# RTEMS

Generated by Doxygen 1.8.20

**1 Main Page** 1

**2 RTEMS History and Introduction** 3

**3 RTEMS Overview** 7

3.1 Introduction . . . 7
3.2 Real-time Application Systems . . . 7
3.3 Real-time Executive . . . 8
3.4 RTEMS Application Architecture . . . 8
3.5 RTEMS Internal Architecture . . . 9
3.6 User Customization and Extensibility . . . 9
3.7 Portability . . . 10
3.8 Memory Requirements . . . 10
3.9 Audience . . . 10

**4 Todo List** 11

**5 Module Index** 13

5.1 Modules . . . 13

**6 Class Index** 19

6.1 Class List . . . 19

**7 File Index** 27

7.1 File List . . . 27

**8 Module Documentation** 45

8.1 AMBA . . . 45
8.1.1 Detailed Description . . . 47
8.2 API . . . 48
8.2.1 Detailed Description . . . 48
8.3 API Mutex Handler . . . 49
8.3.1 Detailed Description . . . 49
8.3.2 Function Documentation . . . 50
8.3.2.1 _API_Mutex_Is_owner() . . . 50
8.3.2.2 _API_Mutex_Lock() . . . 50
8.3.2.3 _API_Mutex_Unlock() . . . 50
8.4 Address Handler . . . 51
8.4.1 Detailed Description . . . 51
8.4.2 Function Documentation . . . 51
8.4.2.1 _Addresses_Add_offset() . . . 51
8.4.2.2 _Addresses_Align_down() . . . 52
8.4.2.3 _Addresses_Align_up() . . . 52
8.4.2.4 _Addresses_Is_aligned() . . . 53
8.4.2.5 _Addresses_Is_in_range() . . . 53

8.4.2.6 _Addresses_Subtract() . . . 54
8.4.2.7 _Addresses_Subtract_offset() . . . 54
8.5 Application Configuration . . . 56
8.5.1 Detailed Description . . . 57
8.6 Application Configuration Information . . . 58
8.6.1 Detailed Description . . . 60
8.6.2 Macro Definition Documentation . . . 60
8.6.2.1 rtems_configuration_get_do_zero_of_workspace . . . 60
8.6.2.2 rtems_configuration_get_idle_task . . . 60
8.6.2.3 rtems_configuration_get_idle_task_stack_size . . . 61
8.6.2.4 rtems_configuration_get_interrupt_stack_size . . . 61
8.6.2.5 rtems_configuration_get_maximum_processors . . . 61
8.6.2.6 rtems_configuration_get_microseconds_per_tick . . . 62
8.6.2.7 rtems_configuration_get_milliseconds_per_tick . . . 62
8.6.2.8 rtems_configuration_get_nanoseconds_per_tick . . . 62
8.6.2.9 rtems_configuration_get_number_of_initial_extensions . . . 63
8.6.2.10 rtems_configuration_get_stack_allocate_hook . . . 63
8.6.2.11 rtems_configuration_get_stack_allocate_init_hook . . . 63
8.6.2.12 rtems_configuration_get_stack_allocator_avoids_work_space . . . 64
8.6.2.13 rtems_configuration_get_stack_free_hook . . . 64
8.6.2.14 rtems_configuration_get_ticks_per_timeslice . . . 64
8.6.2.15 rtems_configuration_get_unified_work_area . . . 65
8.6.2.16 rtems_configuration_get_user_extension_table . . . 65
8.6.2.17 rtems_configuration_get_user_multiprocessing_table . . . 65
8.6.2.18 rtems_configuration_get_work_space_size . . . 66
8.6.2.19 rtems_resource_is_unlimited . . . 66
8.6.2.20 rtems_resource_maximum_per_allocation . . . 66
8.6.2.21 rtems_resource_unlimited . . . 67
8.6.3 Function Documentation . . . 67
8.6.3.1 rtems_configuration_get_maximum_barriers() . . . 67
8.6.3.2 rtems_configuration_get_maximum_extensions() . . . 68
8.6.3.3 rtems_configuration_get_maximum_message_queues() . . . 68
8.6.3.4 rtems_configuration_get_maximum_partitions() . . . 68
8.6.3.5 rtems_configuration_get_maximum_periods() . . . 69
8.6.3.6 rtems_configuration_get_maximum_ports() . . . 69
8.6.3.7 rtems_configuration_get_maximum_regions() . . . 69
8.6.3.8 rtems_configuration_get_maximum_semaphores() . . . 70
8.6.3.9 rtems_configuration_get_maximum_tasks() . . . 70
8.6.3.10 rtems_configuration_get_maximum_timers() . . . 70
8.6.3.11 rtems_configuration_get_rtems_api_configuration() . . . 71
8.6.3.12 rtems_configuration_get_stack_space_size() . . . 71
8.6.3.13 rtems_get_copyright_notice() . . . 71

8.6.3.14 rtems_get_version_string() . . . 71
8.7 Application Configuration Options . . . 72
8.7.1 Detailed Description . . . 72
8.8 Assert Handler . . . 73
8.8.1 Detailed Description . . . 73
8.9 Associativity Routines . . . 74
8.9.1 Detailed Description . . . 75
8.9.2 Function Documentation . . . 75
8.9.2.1 rtems_assoc_32_to_string() . . . 75
8.9.2.2 rtems_assoc_thread_states_to_string() . . . 75
8.10 Atomic Operations . . . 77
8.10.1 Detailed Description . . . 78
8.11 Atomic Operations CPU . . . 79
8.11.1 Detailed Description . . . 81
8.11.2 Function Documentation . . . 81
8.11.2.1 _CPU_atomic_Compare_exchange_uint() . . . 81
8.11.2.2 _CPU_atomic_Compare_exchange_uintptr() . . . 82
8.11.2.3 _CPU_atomic_Compare_exchange_ulong() . . . 82
8.11.2.4 _CPU_atomic_Exchange_uint() . . . 83
8.11.2.5 _CPU_atomic_Exchange_uintptr() . . . 83
8.11.2.6 _CPU_atomic_Exchange_ulong() . . . 84
8.11.2.7 _CPU_atomic_Fence() . . . 84
8.11.2.8 _CPU_atomic_Fetch_add_uint() . . . 84
8.11.2.9 _CPU_atomic_Fetch_add_uintptr() . . . 85
8.11.2.10 _CPU_atomic_Fetch_add_ulong() . . . 85
8.11.2.11 _CPU_atomic_Fetch_and_uint() . . . 86
8.11.2.12 _CPU_atomic_Fetch_and_uintptr() . . . 86
8.11.2.13 _CPU_atomic_Fetch_and_ulong() . . . 87
8.11.2.14 _CPU_atomic_Fetch_or_uint() . . . 87
8.11.2.15 _CPU_atomic_Fetch_or_uintptr() . . . 88
8.11.2.16 _CPU_atomic_Fetch_or_ulong() . . . 88
8.11.2.17 _CPU_atomic_Fetch_sub_uint() . . . 89
8.11.2.18 _CPU_atomic_Fetch_sub_uintptr() . . . 89
8.11.2.19 _CPU_atomic_Fetch_sub_ulong() . . . 89
8.11.2.20 _CPU_atomic_Flag_clear() . . . 90
8.11.2.21 _CPU_atomic_Flag_test_and_set() . . . 90
8.11.2.22 _CPU_atomic_Init_uint() . . . 91
8.11.2.23 _CPU_atomic_Init_uintptr() . . . 91
8.11.2.24 _CPU_atomic_Init_ulong() . . . 91
8.11.2.25 _CPU_atomic_Load_uint() . . . 92
8.11.2.26 _CPU_atomic_Load_uintptr() . . . 92
8.11.2.27 _CPU_atomic_Load_ulong() . . . 93

8.11.2.28 _CPU_atomic_Store_uint() . . . 93
8.11.2.29 _CPU_atomic_Store_uintptr() . . . 93
8.11.2.30 _CPU_atomic_Store_ulong() . . . 94
8.12 BSP Interrupt Support . . . 95
8.12.1 Detailed Description . . . 96
8.12.2 Function Documentation . . . 96
8.12.2.1 bsp_interrupt_facility_initialize() . . . 97
8.12.2.2 bsp_interrupt_handler_default() . . . 97
8.12.2.3 bsp_interrupt_handler_dispatch() . . . 97
8.12.2.4 bsp_interrupt_handler_install() . . . 98
8.12.2.5 bsp_interrupt_handler_is_empty() . . . 98
8.12.2.6 bsp_interrupt_handler_iterate() . . . 99
8.12.2.7 bsp_interrupt_handler_remove() . . . 99
8.12.2.8 bsp_interrupt_initialize() . . . 99
8.12.2.9 bsp_interrupt_vector_disable() . . . 100
8.12.2.10 bsp_interrupt_vector_enable() . . . 100
8.13 BSP Related Configuration Options . . . 101
8.13.1 Detailed Description . . . 101
8.13.2 Macro Definition Documentation . . . 101
8.13.2.1 BSP_IDLE_TASK_BODY . . . 101
8.13.2.2 BSP_IDLE_TASK_STACK_SIZE . . . 102
8.13.2.3 BSP_INITIAL_EXTENSION . . . 102
8.13.2.4 BSP_INTERRUPT_STACK_SIZE . . . 103
8.13.2.5 CONFIGURE_BSP_PREREQUISITE_DRIVERS . . . 103
8.13.2.6 CONFIGURE_DISABLE_BSP_SETTINGS . . . 104
8.13.2.7 CONFIGURE_MALLOC_BSP_SUPPORTS_SBRK . . . 104
8.14 Barrier Handler . . . 105
8.14.1 Detailed Description . . . 106
8.14.2 Enumeration Type Documentation . . . 106
8.14.2.1 CORE_barrier_Disciplines . . . 106
8.14.3 Function Documentation . . . 106
8.14.3.1 _CORE_barrier_Acquire_critical() . . . 107
8.14.3.2 _CORE_barrier_Destroy() . . . 107
8.14.3.3 _CORE_barrier_Do_flush() . . . 107
8.14.3.4 _CORE_barrier_Flush() . . . 108
8.14.3.5 _CORE_barrier_Get_number_of_waiting_threads() . . . 108
8.14.3.6 _CORE_barrier_Initialize() . . . 108
8.14.3.7 _CORE_barrier_Is_automatic() . . . 109
8.14.3.8 _CORE_barrier_Release() . . . 109
8.14.3.9 _CORE_barrier_Seize() . . . 110
8.14.3.10 _CORE_barrier_Surrender() . . . 110
8.15 Barrier Manager . . . 111

8.15.1 Detailed Description . . . 111
8.15.2 Function Documentation . . . 111
8.15.2.1 rtems_barrier_create() . . . 111
8.15.2.2 rtems_barrier_delete() . . . 112
8.15.2.3 rtems_barrier_ident() . . . 112
8.15.2.4 rtems_barrier_release() . . . 112
8.15.2.5 rtems_barrier_wait() . . . 113
8.16 Base Definitions . . . 114
8.16.1 Detailed Description . . . 116
8.16.2 Macro Definition Documentation . . . 116
8.16.2.1 RTEMS_ALIAS . . . 116
8.16.2.2 RTEMS_ALIGN_DOWN . . . 116
8.16.2.3 RTEMS_ALIGN_UP . . . 118
8.16.2.4 RTEMS_ALIGNED . . . 118
8.16.2.5 RTEMS_ALLOC_ALIGN . . . 119
8.16.2.6 RTEMS_ALLOC_SIZE . . . 119
8.16.2.7 RTEMS_ALLOC_SIZE_2 . . . 119
8.16.2.8 RTEMS_ARRAY_SIZE . . . 120
8.16.2.9 RTEMS_CONCAT . . . 120
8.16.2.10 RTEMS_CONTAINER_OF . . . 120
8.16.2.11 RTEMS_DECLARE_GLOBAL_SYMBOL . . . 121
8.16.2.12 RTEMS_DECONST . . . 121
8.16.2.13 RTEMS_DEFINE_GLOBAL_SYMBOL . . . 122
8.16.2.14 RTEMS_DEQUALIFY . . . 122
8.16.2.15 RTEMS_DEQUALIFY_DEPTHX . . . 122
8.16.2.16 RTEMS_DEVOLATILE . . . 123
8.16.2.17 RTEMS_EXPAND . . . 123
8.16.2.18 RTEMS_HAVE_MEMBER_SAME_TYPE . . . 124
8.16.2.19 RTEMS_OBFUSCATE_VARIABLE . . . 124
8.16.2.20 RTEMS_PREDICT_FALSE . . . 124
8.16.2.21 RTEMS_PREDICT_TRUE . . . 126
8.16.2.22 RTEMS_PRINTFLIKE . . . 126
8.16.2.23 RTEMS_RETURN_ADDRESS . . . 127
8.16.2.24 RTEMS_SECTION . . . 127
8.16.2.25 RTEMS_STATIC_ASSERT . . . 127
8.16.2.26 RTEMS_STRING . . . 127
8.16.2.27 RTEMS_SYMBOL_NAME . . . 128
8.16.2.28 RTEMS_TYPEOF_REFX . . . 128
8.16.2.29 RTEMS_WEAK . . . 129
8.16.2.30 RTEMS_WEAK_ALIAS . . . 129
8.16.2.31 RTEMS_XCONCAT . . . 129
8.16.2.32 RTEMS_XSTRING . . . 130

8.16.2.33 RTEMS_ZERO_LENGTH_ARRAY . . . 130
8.17 Basic Types . . . 131
8.17.1 Detailed Description . . . 131
8.17.2 Macro Definition Documentation . . . 131
8.17.2.1 RTEMS_ID_NONE . . . 131
8.17.3 Typedef Documentation . . . 131
8.17.3.1 rtems_name . . . 131
8.18 Bitmap Priority Thread Routines . . . 132
8.18.1 Detailed Description . . . 132
8.19 Block Device Cache Configuration . . . 133
8.19.1 Detailed Description . . . 133
8.19.2 Macro Definition Documentation . . . 133
8.19.2.1 CONFIGURE_APPLICATION_NEEDS_LIBBLOCK . . . 134
8.19.2.2 CONFIGURE_BDBUF_BUFFER_MAX_SIZE . . . 134
8.19.2.3 CONFIGURE_BDBUF_BUFFER_MIN_SIZE . . . 135
8.19.2.4 CONFIGURE_BDBUF_CACHE_MEMORY_SIZE . . . 135
8.19.2.5 CONFIGURE_BDBUF_MAX_READ_AHEAD_BLOCKS . . . 136
8.19.2.6 CONFIGURE_BDBUF_MAX_WRITE_BLOCKS . . . 136
8.19.2.7 CONFIGURE_BDBUF_READ_AHEAD_TASK_PRIORITY . . . 137
8.19.2.8 CONFIGURE_BDBUF_TASK_STACK_SIZE . . . 137
8.19.2.9 CONFIGURE_SWAPOUT_BLOCK_HOLD . . . 138
8.19.2.10 CONFIGURE_SWAPOUT_SWAP_PERIOD . . . 138
8.19.2.11 CONFIGURE_SWAPOUT_TASK_PRIORITY . . . 138
8.19.2.12 CONFIGURE_SWAPOUT_WORKER_TASK_PRIORITY . . . 139
8.19.2.13 CONFIGURE_SWAPOUT_WORKER_TASKS . . . 139
8.20 Board Support Packages . . . 140
8.20.1 Detailed Description . . . 140
8.21 Boolean Checks . . . 141
8.21.1 Detailed Description . . . 141
8.21.2 Macro Definition Documentation . . . 141
8.21.2.1 T_assert_false . . . 141
8.21.2.2 T_false . . . 141
8.21.2.3 T_quiet_false . . . 142
8.21.2.4 T_step_assert_false . . . 142
8.21.2.5 T_step_false . . . 142
8.22 Bootcard . . . 143
8.22.1 Detailed Description . . . 143
8.22.2 Function Documentation . . . 143
8.22.2.1 boot_card() . . . 143
8.22.2.2 bsp_start_on_secondary_processor() . . . 144
8.23 CPU Architecture Support . . . 145
8.23.1 Detailed Description . . . 145

8.24 Cache Manager . . . 146
8.24.1 Detailed Description . . . 147
8.24.2 Function Documentation . . . 147
8.24.2.1 rtems_cache_aligned_malloc() . . . 147
8.24.2.2 rtems_cache_coherent_add_area() . . . 147
8.24.2.3 rtems_cache_coherent_allocate() . . . 147
8.24.2.4 rtems_cache_coherent_free() . . . 148
8.24.2.5 rtems_cache_flush_multiple_data_lines() . . . 148
8.24.2.6 rtems_cache_get_data_cache_size() . . . 148
8.24.2.7 rtems_cache_get_instruction_cache_size() . . . 149
8.24.2.8 rtems_cache_instruction_sync_after_code_change() . . . 149
8.24.2.9 rtems_cache_invalidate_multiple_data_lines() . . . 149
8.24.2.10 rtems_cache_invalidate_multiple_instruction_lines() . . . 150
8.25 Chain Handler . . . 151
8.25.1 Detailed Description . . . 153
8.25.2 Macro Definition Documentation . . . 154
8.25.2.1 CHAIN_INITIALIZER_ONE_NODE . . . 154
8.25.2.2 CHAIN_ITERATOR_REGISTRY_INITIALIZER . . . 154
8.25.2.3 CHAIN_NODE_INITIALIZER_ONE_NODE_CHAIN . . . 154
8.25.3 Typedef Documentation . . . 155
8.25.3.1 Chain_Node . . . 155
8.25.3.2 Chain_Node_order . . . 155
8.25.4 Enumeration Type Documentation . . . 155
8.25.4.1 Chain_Iterator_direction . . . 155
8.25.5 Function Documentation . . . 156
8.25.5.1 _Chain_Append_if_is_off_chain_unprotected() . . . 156
8.25.5.2 _Chain_Append_unprotected() . . . 156
8.25.5.3 _Chain_Append_with_empty_check_unprotected() . . . 157
8.25.5.4 _Chain_Are_nodes_equal() . . . 157
8.25.5.5 _Chain_Extract_unprotected() . . . 158
8.25.5.6 _Chain_First() . . . 158
8.25.5.7 _Chain_Get_first_unprotected() . . . 159
8.25.5.8 _Chain_Get_unprotected() . . . 159
8.25.5.9 _Chain_Get_with_empty_check_unprotected() . . . 160
8.25.5.10 _Chain_Has_only_one_node() . . . 160
8.25.5.11 _Chain_Head() . . . 161
8.25.5.12 _Chain_Immutable_first() . . . 161
8.25.5.13 _Chain_Immutable_head() . . . 162
8.25.5.14 _Chain_Immutable_last() . . . 162
8.25.5.15 _Chain_Immutable_next() . . . 162
8.25.5.16 _Chain_Immutable_previous() . . . 163
8.25.5.17 _Chain_Immutable_tail() . . . 163

8.25.5.18 _Chain_Initialize() . . . 164
8.25.5.19 _Chain_Initialize_empty() . . . 164
8.25.5.20 _Chain_Initialize_node() . . . 165
8.25.5.21 _Chain_Initialize_one() . . . 165
8.25.5.22 _Chain_Insert_ordered_unprotected() . . . 165
8.25.5.23 _Chain_Insert_unprotected() . . . 166
8.25.5.24 _Chain_Is_empty() . . . 166
8.25.5.25 _Chain_Is_first() . . . 167
8.25.5.26 _Chain_Is_head() . . . 167
8.25.5.27 _Chain_Is_last() . . . 167
8.25.5.28 _Chain_Is_node_off_chain() . . . 168
8.25.5.29 _Chain_Is_tail() . . . 168
8.25.5.30 _Chain_Iterator_destroy() . . . 169
8.25.5.31 _Chain_Iterator_initialize() . . . 169
8.25.5.32 _Chain_Iterator_next() . . . 170
8.25.5.33 _Chain_Iterator_registry_initialize() . . . 171
8.25.5.34 _Chain_Iterator_registry_update() . . . 171
8.25.5.35 _Chain_Iterator_set_position() . . . 171
8.25.5.36 _Chain_Last() . . . 172
8.25.5.37 _Chain_Next() . . . 172
8.25.5.38 _Chain_Node_count_unprotected() . . . 173
8.25.5.39 _Chain_Prepend_unprotected() . . . 173
8.25.5.40 _Chain_Prepend_with_empty_check_unprotected() . . . 174
8.25.5.41 _Chain_Previous() . . . 174
8.25.5.42 _Chain_Set_off_chain() . . . 175
8.25.5.43 _Chain_Tail() . . . 175
8.26 Chains . . . 176
8.26.1 Detailed Description . . . 178
8.26.2 Macro Definition Documentation . . . 178
8.26.2.1 RTEMS_CHAIN_INITIALIZER_ONE_NODE . . . 178
8.26.2.2 RTEMS_CHAIN_NODE_INITIALIZER_ONE_NODE_CHAIN . . . 179
8.26.3 Function Documentation . . . 179
8.26.3.1 rtems_chain_append() . . . 179
8.26.3.2 rtems_chain_append_unprotected() . . . 179
8.26.3.3 rtems_chain_append_with_empty_check() . . . 179
8.26.3.4 rtems_chain_append_with_notification() . . . 180
8.26.3.5 rtems_chain_are_nodes_equal() . . . 180
8.26.3.6 rtems_chain_extract() . . . 181
8.26.3.7 rtems_chain_extract_unprotected() . . . 181
8.26.3.8 rtems_chain_first() . . . 181
8.26.3.9 rtems_chain_get() . . . 182
8.26.3.10 rtems_chain_get_with_empty_check() . . . 182

8.26.3.11 rtems_chain_get_with_notification() . . . 182
8.26.3.12 rtems_chain_get_with_wait() . . . 183
8.26.3.13 rtems_chain_has_only_one_node() . . . 183
8.26.3.14 rtems_chain_head() . . . 184
8.26.3.15 rtems_chain_immutable_first() . . . 184
8.26.3.16 rtems_chain_immutable_head() . . . 184
8.26.3.17 rtems_chain_immutable_last() . . . 185
8.26.3.18 rtems_chain_immutable_next() . . . 185
8.26.3.19 rtems_chain_immutable_previous() . . . 186
8.26.3.20 rtems_chain_immutable_tail() . . . 186
8.26.3.21 rtems_chain_initialize() . . . 187
8.26.3.22 rtems_chain_initialize_empty() . . . 187
8.26.3.23 rtems_chain_initialize_node() . . . 187
8.26.3.24 rtems_chain_insert() . . . 188
8.26.3.25 rtems_chain_is_empty() . . . 188
8.26.3.26 rtems_chain_is_first() . . . 188
8.26.3.27 rtems_chain_is_head() . . . 189
8.26.3.28 rtems_chain_is_last() . . . 189
8.26.3.29 rtems_chain_is_node_off_chain() . . . 190
8.26.3.30 rtems_chain_is_null_node() . . . 190
8.26.3.31 rtems_chain_is_tail() . . . 191
8.26.3.32 rtems_chain_last() . . . 191
8.26.3.33 rtems_chain_next() . . . 192
8.26.3.34 rtems_chain_node_count_unprotected() . . . 192
8.26.3.35 rtems_chain_prepend() . . . 192
8.26.3.36 rtems_chain_prepend_unprotected() . . . 193
8.26.3.37 rtems_chain_prepend_with_empty_check() . . . 193
8.26.3.38 rtems_chain_prepend_with_notification() . . . 193
8.26.3.39 rtems_chain_previous() . . . 194
8.26.3.40 rtems_chain_set_off_chain() . . . 194
8.26.3.41 rtems_chain_tail() . . . 195
8.27 Character Checks . . . 196
8.27.1 Detailed Description . . . 196
8.28 Classic . . . 197
8.28.1 Detailed Description . . . 197
8.29 Classic . . . 198
8.29.1 Detailed Description . . . 199
8.29.2 Objects . . . 199
8.29.2.1 Object Names . . . 200
8.29.2.2 Object Identifiers . . . 200
8.29.2.3 Object Identifer Description . . . 201
8.29.3 Communication and Synchronization . . . 201

8.29.4 Time . . . 202
8.29.5 Memory Management . . . 202
8.30 Classic API Configuration . . . 203
8.30.1 Detailed Description . . . 203
8.30.2 Macro Definition Documentation . . . 203
8.30.2.1 CONFIGURE_MAXIMUM_BARRIERS . . . 203
8.30.2.2 CONFIGURE_MAXIMUM_MESSAGE_QUEUES . . . 204
8.30.2.3 CONFIGURE_MAXIMUM_PARTITIONS . . . 205
8.30.2.4 CONFIGURE_MAXIMUM_PERIODS . . . 205
8.30.2.5 CONFIGURE_MAXIMUM_PORTS . . . 206
8.30.2.6 CONFIGURE_MAXIMUM_REGIONS . . . 206
8.30.2.7 CONFIGURE_MAXIMUM_SEMAPHORES . . . 207
8.30.2.8 CONFIGURE_MAXIMUM_TASKS . . . 207
8.30.2.9 CONFIGURE_MAXIMUM_THREAD_LOCAL_STORAGE_SIZE . . . 208
8.30.2.10 CONFIGURE_MAXIMUM_TIMERS . . . 209
8.30.2.11 CONFIGURE_MAXIMUM_USER_EXTENSIONS . . . 210
8.30.2.12 CONFIGURE_MINIMUM_TASKS_WITH_USER_PROVIDED_STORAGE . . . 210
8.31 Classic API Initialization Task Configuration . . . 211
8.31.1 Detailed Description . . . 211
8.31.2 Macro Definition Documentation . . . 211
8.31.2.1 CONFIGURE_INIT_TASK_ARGUMENTS . . . 211
8.31.2.2 CONFIGURE_INIT_TASK_ATTRIBUTES . . . 212
8.31.2.3 CONFIGURE_INIT_TASK_ENTRY_POINT . . . 212
8.31.2.4 CONFIGURE_INIT_TASK_INITIAL_MODES . . . 213
8.31.2.5 CONFIGURE_INIT_TASK_NAME . . . 213
8.31.2.6 CONFIGURE_INIT_TASK_PRIORITY . . . 214
8.31.2.7 CONFIGURE_INIT_TASK_STACK_SIZE . . . 214
8.31.2.8 CONFIGURE_RTEMS_INIT_TASKS_TABLE . . . 214
8.32 Classic ASR Implementation . . . 215
8.32.1 Detailed Description . . . 215
8.32.2 Function Documentation . . . 215
8.32.2.1 _ASR_Initialize() . . . 215
8.33 Classic Attributes Implementation . . . 216
8.33.1 Detailed Description . . . 216
8.33.2 Macro Definition Documentation . . . 216
8.33.2.1 ATTRIBUTES_NOT_SUPPORTED . . . 217
8.33.2.2 ATTRIBUTES_REQUIRED . . . 217
8.33.3 Function Documentation . . . 217
8.33.3.1 _Attributes_Clear() . . . 217
8.33.3.2 _Attributes_Has_at_most_one_protocol() . . . 217
8.33.3.3 _Attributes_Is_barrier_automatic() . . . 218
8.33.3.4 _Attributes_Is_binary_semaphore() . . . 218

8.33.3.5 _Attributes_Is_counting_semaphore() . . . 218
8.33.3.6 _Attributes_Is_floating_point() . . . 218
8.33.3.7 _Attributes_Is_inherit_priority() . . . 219
8.33.3.8 _Attributes_Is_multiprocessor_resource_sharing() . . . 219
8.33.3.9 _Attributes_Is_priority() . . . 219
8.33.3.10 _Attributes_Is_priority_ceiling() . . . 219
8.33.3.11 _Attributes_Is_simple_binary_semaphore() . . . 220
8.33.3.12 _Attributes_Is_system_task() . . . 220
8.33.3.13 _Attributes_Set() . . . 220
8.34 Classic Barrier Implementation . . . 221
8.34.1 Detailed Description . . . 221
8.34.2 Macro Definition Documentation . . . 221
8.34.2.1 BARRIER_INFORMATION_DEFINE . . . 221
8.34.3 Function Documentation . . . 222
8.34.3.1 _Barrier_Allocate() . . . 222
8.34.3.2 _Barrier_Free() . . . 222
8.35 Classic Message Queue Implementation . . . 223
8.35.1 Detailed Description . . . 224
8.35.2 Macro Definition Documentation . . . 224
8.35.2.1 MESSAGE_QUEUE_INFORMATION_DEFINE . . . 224
8.35.3 Enumeration Type Documentation . . . 224
8.35.3.1 Message_queue_Submit_types . . . 224
8.35.4 Function Documentation . . . 225
8.35.4.1 _Message_queue_Create() . . . 225
8.35.4.2 _Message_queue_Free() . . . 225
8.35.4.3 _Message_queue_Submit() . . . 225
8.36 Classic Modes Implementation . . . 226
8.36.1 Detailed Description . . . 226
8.36.2 Function Documentation . . . 226
8.36.2.1 _Modes_Change() . . . 226
8.36.2.2 _Modes_Get_interrupt_level() . . . 227
8.36.2.3 _Modes_Is_asr_disabled() . . . 227
8.36.2.4 _Modes_Is_preempt() . . . 227
8.36.2.5 _Modes_Is_timeslice() . . . 227
8.36.2.6 _Modes_Mask_changed() . . . 228
8.36.2.7 _Modes_Set_interrupt_level() . . . 228
8.37 Classic Object Implementation . . . 229
8.37.1 Detailed Description . . . 229
8.37.2 Function Documentation . . . 229
8.37.2.1 _RTEMS_Name_to_id() . . . 229
8.38 Classic Options Implementation . . . 230
8.38.1 Detailed Description . . . 230

8.38.2 Function Documentation . . . 230
8.38.2.1 _Options_Is_any() . . . 230
8.38.2.2 _Options_Is_no_wait() . . . 230
8.39 Classic Rate Monotonic Scheduler Implementation . . . 231
8.39.1 Detailed Description . . . 232
8.39.2 Macro Definition Documentation . . . 232
8.39.2.1 RATE_MONOTONIC_INFORMATION_DEFINE . . . 232
8.39.3 Function Documentation . . . 232
8.39.3.1 _Rate_monotonic_Allocate() . . . 232
8.39.3.2 _Rate_monotonic_Get_status() . . . 233
8.40 Classic Region Manager Implementation . . . 234
8.40.1 Detailed Description . . . 234
8.40.2 Macro Definition Documentation . . . 234
8.40.2.1 REGION_INFORMATION_DEFINE . . . 235
8.40.3 Function Documentation . . . 235
8.40.3.1 _Region_Allocate() . . . 235
8.40.3.2 _Region_Allocate_segment() . . . 235
8.40.3.3 _Region_Free() . . . 236
8.40.3.4 _Region_Free_segment() . . . 236
8.40.3.5 _Region_Process_queue() . . . 236
8.41 Classic Status Implementation . . . 237
8.41.1 Detailed Description . . . 237
8.41.2 Variable Documentation . . . 237
8.41.2.1 _Status_Object_name_errors_to_status . . . 237
8.42 Classic Tasks Manager Implementation . . . 238
8.42.1 Detailed Description . . . 238
8.42.2 Function Documentation . . . 238
8.42.2.1 _RTEMS_Priority_From_core() . . . 239
8.42.2.2 _RTEMS_Priority_To_core() . . . 239
8.42.2.3 _RTEMS_tasks_Free() . . . 239
8.42.2.4 _RTEMS_tasks_Initialize_user_tasks() . . . 240
8.42.3 Variable Documentation . . . 240
8.42.3.1 _RTEMS_tasks_Information . . . 240
8.42.3.2 _RTEMS_tasks_User_task_table . . . 240
8.43 Classic Timer Implementation . . . 241
8.43.1 Detailed Description . . . 242
8.43.2 Macro Definition Documentation . . . 242
8.43.2.1 TIMER_INFORMATION_DEFINE . . . 242
8.43.3 Function Documentation . . . 242
8.43.3.1 _Timer_Allocate() . . . 242
8.43.3.2 _Timer_Free() . . . 243
8.43.4 Variable Documentation . . . 243

8.43.4.1 _Timer_server . . . 243
8.44 Clock Driver . . . 244
8.44.1 Detailed Description . . . 244
8.44.2 Variable Documentation . . . 244
8.44.2.1 Clock_driver_ticks . . . 244
8.45 Clock Manager . . . 245
8.45.1 Detailed Description . . . 245
8.45.2 Function Documentation . . . 245
8.45.2.1 rtems_clock_get_seconds_since_epoch() . . . 245
8.45.2.2 rtems_clock_get_tod() . . . 246
8.45.2.3 rtems_clock_get_tod_timeval() . . . 246
8.45.2.4 rtems_clock_get_uptime() . . . 246
8.45.2.5 rtems_clock_get_uptime_timeval() . . . 247
8.45.2.6 rtems_clock_set() . . . 247
8.45.2.7 rtems_clock_tick_before() . . . 247
8.45.2.8 rtems_clock_tick_later() . . . 248
8.45.2.9 rtems_clock_tick_later_usec() . . . 248
8.46 Clock Support . . . 249
8.46.1 Detailed Description . . . 249
8.47 Console Driver Support . . . 250
8.48 Context Handler . . . 251
8.48.1 Detailed Description . . . 251
8.48.2 Macro Definition Documentation . . . 251
8.48.2.1 _Context_Initialization_at_thread_begin . . . 251
8.48.2.2 _Context_Initialize . . . 252
8.48.2.3 _Context_Initialize_fp . . . 252
8.48.2.4 _Context_Restart_self . . . 253
8.48.2.5 _Context_Restore_fp . . . 253
8.48.2.6 _Context_Save_fp . . . 253
8.48.2.7 _Context_Switch . . . 254
8.48.2.8 CONTEXT_FP_SIZE . . . 254
8.49 DEFAULT_INITIAL_EXTENSION Support . . . 255
8.49.1 Detailed Description . . . 255
8.50 Destructors . . . 256
8.50.1 Detailed Description . . . 256
8.51 Deterministic Priority Scheduler . . . 257
8.51.1 Detailed Description . . . 258
8.51.2 Macro Definition Documentation . . . 258
8.51.2.1 SCHEDULER_PRIORITY_ENTRY_POINTS . . . 258
8.51.3 Function Documentation . . . 259
8.51.3.1 _Scheduler_priority_Block() . . . 259
8.51.3.2 _Scheduler_priority_Extract_body() . . . 259

8.51.3.3 _Scheduler_priority_Get_context() . . . 260
8.51.3.4 _Scheduler_priority_Initialize() . . . 260
8.51.3.5 _Scheduler_priority_Node_downcast() . . . 260
8.51.3.6 _Scheduler_priority_Node_initialize() . . . 261
8.51.3.7 _Scheduler_priority_Ready_queue_enqueue() . . . 261
8.51.3.8 _Scheduler_priority_Ready_queue_enqueue_first() . . . 262
8.51.3.9 _Scheduler_priority_Ready_queue_extract() . . . 262
8.51.3.10 _Scheduler_priority_Ready_queue_first() . . . 262
8.51.3.11 _Scheduler_priority_Ready_queue_initialize() . . . 264
8.51.3.12 _Scheduler_priority_Ready_queue_update() . . . 264
8.51.3.13 _Scheduler_priority_Schedule() . . . 265
8.51.3.14 _Scheduler_priority_Schedule_body() . . . 265
8.51.3.15 _Scheduler_priority_Thread_get_node() . . . 265
8.51.3.16 _Scheduler_priority_Unblock() . . . 266
8.51.3.17 _Scheduler_priority_Update_priority() . . . 266
8.51.3.18 _Scheduler_priority_Yield() . . . 267
8.52 Device Driver Configuration . . . 268
8.52.1 Detailed Description . . . 268
8.52.2 Macro Definition Documentation . . . 268
8.52.2.1 CONFIGURE_APPLICATION_DOES_NOT_NEED_CLOCK_DRIVER . . . 269
8.52.2.2 CONFIGURE_APPLICATION_EXTRA_DRIVERS . . . 269
8.52.2.3 CONFIGURE_APPLICATION_NEEDS_ATA_DRIVER . . . 270
8.52.2.4 CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER . . . 270
8.52.2.5 CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER . . . 271
8.52.2.6 CONFIGURE_APPLICATION_NEEDS_FRAME_BUFFER_DRIVER . . . 271
8.52.2.7 CONFIGURE_APPLICATION_NEEDS_IDE_DRIVER . . . 272
8.52.2.8 CONFIGURE_APPLICATION_NEEDS_NULL_DRIVER . . . 272
8.52.2.9 CONFIGURE_APPLICATION_NEEDS_RTC_DRIVER . . . 273
8.52.2.10 CONFIGURE_APPLICATION_NEEDS_SIMPLE_CONSOLE_DRIVER . . . 273
8.52.2.11 CONFIGURE_APPLICATION_NEEDS_SIMPLE_TASK_CONSOLE_DRIVER . . . 274
8.52.2.12 CONFIGURE_APPLICATION_NEEDS_STUB_DRIVER . . . 274
8.52.2.13 CONFIGURE_APPLICATION_NEEDS_TIMER_DRIVER . . . 275
8.52.2.14 CONFIGURE_APPLICATION_NEEDS_WATCHDOG_DRIVER . . . 275
8.52.2.15 CONFIGURE_APPLICATION_NEEDS_ZERO_DRIVER . . . 276
8.52.2.16 CONFIGURE_APPLICATION_PREREQUISITE_DRIVERS . . . 276
8.52.2.17 CONFIGURE_ATA_DRIVER_TASK_PRIORITY . . . 277
8.52.2.18 CONFIGURE_MAXIMUM_DRIVERS . . . 277
8.53 Device Drivers . . . 279
8.53.1 Detailed Description . . . 279
8.54 Directive Attributes . . . 280
8.54.1 Detailed Description . . . 281
8.54.2 Macro Definition Documentation . . . 281

8.54.2.1 RTEMS_FLOATING_POINT . . . 281
8.54.2.2 RTEMS_GLOBAL . . . 281
8.54.2.3 RTEMS_INHERIT_PRIORITY . . . 282
8.54.2.4 RTEMS_LOCAL . . . 282
8.54.2.5 RTEMS_MULTIPROCESSOR_RESOURCE_SHARING . . . 282
8.54.2.6 RTEMS_NO_FLOATING_POINT . . . 282
8.54.2.7 RTEMS_PRIORITY_CEILING . . . 283
8.54.3 Typedef Documentation . . . 283
8.54.3.1 rtems_attribute . . . 283
8.55 Directive Options . . . 284
8.55.1 Detailed Description . . . 284
8.55.2 Macro Definition Documentation . . . 284
8.55.2.1 RTEMS_NO_WAIT . . . 284
8.55.2.2 RTEMS_WAIT . . . 284
8.56 Directive Status Codes . . . 285
8.56.1 Detailed Description . . . 285
8.56.2 Enumeration Type Documentation . . . 286
8.56.2.1 rtems_status_code . . . 286
8.56.3 Function Documentation . . . 287
8.56.3.1 rtems_are_statuses_equal() . . . 287
8.56.3.2 rtems_is_status_successful() . . . 288
8.56.3.3 rtems_status_code_to_errno() . . . 288
8.56.3.4 rtems_status_text() . . . 289
8.57 Dual Ported Memory Manager Implementation . . . 290
8.57.1 Detailed Description . . . 290
8.57.2 Macro Definition Documentation . . . 290
8.57.2.1 DUAL_PORTED_MEMORY_INFORMATION_DEFINE . . . 290
8.57.3 Function Documentation . . . 291
8.57.3.1 _Dual_ported_memory_Allocate() . . . 291
8.57.3.2 _Dual_ported_memory_Free() . . . 291
8.58 Dual-Ported Memory Manager . . . 292
8.58.1 Detailed Description . . . 292
8.58.2 Function Documentation . . . 292
8.58.2.1 rtems_port_create() . . . 292
8.58.2.2 rtems_port_delete() . . . 293
8.58.2.3 rtems_port_external_to_internal() . . . 293
8.58.2.4 rtems_port_ident() . . . 293
8.58.2.5 rtems_port_internal_to_external() . . . 294
8.59 EDF Priority SMP Scheduler . . . 295
8.59.1 Detailed Description . . . 296
8.59.2 Macro Definition Documentation . . . 296
8.59.2.1 SCHEDULER_EDF_SMP_ENTRY_POINTS . . . 296

8.59.3 Function Documentation . . . 297
8.59.3.1 _Scheduler_EDF_SMP_Add_processor() . . . 297
8.59.3.2 _Scheduler_EDF_SMP_Ask_for_help() . . . 297
8.59.3.3 _Scheduler_EDF_SMP_Block() . . . 297
8.59.3.4 _Scheduler_EDF_SMP_Initialize() . . . 298
8.59.3.5 _Scheduler_EDF_SMP_Node_initialize() . . . 298
8.59.3.6 _Scheduler_EDF_SMP_Pin() . . . 299
8.59.3.7 _Scheduler_EDF_SMP_Reconsider_help_request() . . . 299
8.59.3.8 _Scheduler_EDF_SMP_Remove_processor() . . . 299
8.59.3.9 _Scheduler_EDF_SMP_Set_affinity() . . . 300
8.59.3.10 _Scheduler_EDF_SMP_Start_idle() . . . 300
8.59.3.11 _Scheduler_EDF_SMP_Unblock() . . . 301
8.59.3.12 _Scheduler_EDF_SMP_Unpin() . . . 301
8.59.3.13 _Scheduler_EDF_SMP_Update_priority() . . . 301
8.59.3.14 _Scheduler_EDF_SMP_Withdraw_node() . . . 303
8.59.3.15 _Scheduler_EDF_SMP_Yield() . . . 303
8.60 EDF Scheduler . . . 304
8.60.1 Detailed Description . . . 305
8.60.2 Macro Definition Documentation . . . 305
8.60.2.1 SCHEDULER_EDF_ENTRY_POINTS . . . 306
8.60.2.2 SCHEDULER_EDF_PRIO_MSB . . . 306
8.60.3 Function Documentation . . . 306
8.60.3.1 _Scheduler_EDF_Block() . . . 306
8.60.3.2 _Scheduler_EDF_Cancel_job() . . . 307
8.60.3.3 _Scheduler_EDF_Enqueue() . . . 307
8.60.3.4 _Scheduler_EDF_Extract() . . . 307
8.60.3.5 _Scheduler_EDF_Extract_body() . . . 308
8.60.3.6 _Scheduler_EDF_Get_context() . . . 308
8.60.3.7 _Scheduler_EDF_Initialize() . . . 309
8.60.3.8 _Scheduler_EDF_Less() . . . 309
8.60.3.9 _Scheduler_EDF_Map_priority() . . . 309
8.60.3.10 _Scheduler_EDF_Node_downcast() . . . 310
8.60.3.11 _Scheduler_EDF_Node_initialize() . . . 310
8.60.3.12 _Scheduler_EDF_Priority_less_equal() . . . 310
8.60.3.13 _Scheduler_EDF_Release_job() . . . 311
8.60.3.14 _Scheduler_EDF_Schedule() . . . 311
8.60.3.15 _Scheduler_EDF_Schedule_body() . . . 312
8.60.3.16 _Scheduler_EDF_Thread_get_node() . . . 312
8.60.3.17 _Scheduler_EDF_Unblock() . . . 312
8.60.3.18 _Scheduler_EDF_Unmap_priority() . . . 313
8.60.3.19 _Scheduler_EDF_Update_priority() . . . 313
8.60.3.20 _Scheduler_EDF_Yield() . . . 314

8.61 Event Implementation . . . 315
8.61.1 Detailed Description . . . 316
8.61.2 Function Documentation . . . 316
8.61.2.1 _Event_Initialize() . . . 316
8.61.2.2 _Event_Seize() . . . 316
8.61.2.3 _Event_sets_Clear() . . . 317
8.61.2.4 _Event_sets_Get() . . . 317
8.61.2.5 _Event_sets_Is_empty() . . . 318
8.61.2.6 _Event_sets_Post() . . . 318
8.61.2.7 _Event_Surrender() . . . 318
8.62 Event Manager . . . 320
8.62.1 Detailed Description . . . 321
8.62.2 Macro Definition Documentation . . . 321
8.62.2.1 RTEMS_ALL_EVENTS . . . 322
8.62.3 Function Documentation . . . 322
8.62.3.1 rtems_event_receive() . . . 322
8.62.3.2 rtems_event_send() . . . 323
8.63 Event Recording Configuration . . . 325
8.63.1 Detailed Description . . . 325
8.63.2 Macro Definition Documentation . . . 325
8.63.2.1 CONFIGURE_RECORD_EXTENSIONS_ENABLED . . . 325
8.63.2.2 CONFIGURE_RECORD_FATAL_DUMP_BASE64 . . . 326
8.63.2.3 CONFIGURE_RECORD_FATAL_DUMP_BASE64_ZLIB . . . 326
8.63.2.4 CONFIGURE_RECORD_PER_PROCESSOR_ITEMS . . . 327
8.64 Fatal Error Manager . . . 328
8.64.1 Detailed Description . . . 328
8.64.2 Function Documentation . . . 328
8.64.2.1 rtems_exception_frame_print() . . . 328
8.64.2.2 rtems_fatal() . . . 329
8.64.2.3 rtems_fatal_error_occurred() . . . 329
8.64.2.4 rtems_fatal_source_text() . . . 329
8.64.2.5 rtems_internal_error_text() . . . 330
8.64.2.6 RTEMS_PRINTFLIKE() . . . 330
8.65 File System Node Handler . . . 331
8.65.1 Detailed Description . . . 332
8.65.2 Typedef Documentation . . . 332
8.65.2.1 rtems_filesystem_close_t . . . 332
8.65.2.2 rtems_filesystem_fcntl_t . . . 333
8.65.2.3 rtems_filesystem_fdatasync_t . . . 333
8.65.2.4 rtems_filesystem_fstat_t . . . 334
8.65.2.5 rtems_filesystem_fsync_t . . . 334
8.65.2.6 rtems_filesystem_ftruncate_t . . . 335

8.65.2.7 rtems_filesystem_ioctl_t . . . 335
8.65.2.8 rtems_filesystem_kqfilter_t . . . 336
8.65.2.9 rtems_filesystem_lseek_t . . . 336
8.65.2.10 rtems_filesystem_mmap_t . . . 337
8.65.2.11 rtems_filesystem_open_t . . . 338
8.65.2.12 rtems_filesystem_poll_t . . . 338
8.65.2.13 rtems_filesystem_read_t . . . 339
8.65.2.14 rtems_filesystem_readv_t . . . 339
8.65.2.15 rtems_filesystem_write_t . . . 340
8.65.2.16 rtems_filesystem_writev_t . . . 340
8.65.3 Function Documentation . . . 341
8.65.3.1 rtems_filesystem_default_close() . . . 341
8.65.3.2 rtems_filesystem_default_fcntl() . . . 341
8.65.3.3 rtems_filesystem_default_fstat() . . . 342
8.65.3.4 rtems_filesystem_default_fsync_or_fdatasync() . . . 342
8.65.3.5 rtems_filesystem_default_fsync_or_fdatasync_success() . . . 342
8.65.3.6 rtems_filesystem_default_ftruncate() . . . 343
8.65.3.7 rtems_filesystem_default_ftruncate_directory() . . . 343
8.65.3.8 rtems_filesystem_default_ioctl() . . . 343
8.65.3.9 rtems_filesystem_default_kqfilter() . . . 344
8.65.3.10 rtems_filesystem_default_lseek() . . . 344
8.65.3.11 rtems_filesystem_default_lseek_directory() . . . 344
8.65.3.12 rtems_filesystem_default_lseek_file() . . . 345
8.65.3.13 rtems_filesystem_default_mmap() . . . 345
8.65.3.14 rtems_filesystem_default_open() . . . 346
8.65.3.15 rtems_filesystem_default_poll() . . . 346
8.65.3.16 rtems_filesystem_default_read() . . . 346
8.65.3.17 rtems_filesystem_default_readv() . . . 347
8.65.3.18 rtems_filesystem_default_write() . . . 347
8.65.3.19 rtems_filesystem_default_writev() . . . 347
8.66 File System Operations . . . 349
8.66.1 Detailed Description . . . 351
8.66.2 Typedef Documentation . . . 351
8.66.2.1 rtems_filesystem_are_nodes_equal_t . . . 351
8.66.2.2 rtems_filesystem_chown_t . . . 351
8.66.2.3 rtems_filesystem_clonenode_t . . . 352
8.66.2.4 rtems_filesystem_eval_path_t . . . 352
8.66.2.5 rtems_filesystem_fchmod_t . . . 353
8.66.2.6 rtems_filesystem_freenode_t . . . 353
8.66.2.7 rtems_filesystem_fsmount_me_t . . . 354
8.66.2.8 rtems_filesystem_fsunmount_me_t . . . 354
8.66.2.9 rtems_filesystem_link_t . . . 355

8.66.2.10 rtems_filesystem_mknod_t . . . 355
8.66.2.11 rtems_filesystem_mount_t . . . 356
8.66.2.12 rtems_filesystem_mt_entry_lock_t . . . 356
8.66.2.13 rtems_filesystem_mt_entry_unlock_t . . . 357
8.66.2.14 rtems_filesystem_readlink_t . . . 357
8.66.2.15 rtems_filesystem_rename_t . . . 358
8.66.2.16 rtems_filesystem_rmnod_t . . . 358
8.66.2.17 rtems_filesystem_statvfs_t . . . 359
8.66.2.18 rtems_filesystem_symlink_t . . . 359
8.66.2.19 rtems_filesystem_unmount_t . . . 360
8.66.2.20 rtems_filesystem_utime_t . . . 360
8.66.3 Function Documentation . . . 361
8.66.3.1 rtems_filesystem_default_are_nodes_equal() . . . 361
8.66.3.2 rtems_filesystem_default_chown() . . . 361
8.66.3.3 rtems_filesystem_default_clonenode() . . . 362
8.66.3.4 rtems_filesystem_default_eval_path() . . . 362
8.66.3.5 rtems_filesystem_default_fchmod() . . . 362
8.66.3.6 rtems_filesystem_default_freenode() . . . 363
8.66.3.7 rtems_filesystem_default_fsunmount() . . . 363
8.66.3.8 rtems_filesystem_default_link() . . . 363
8.66.3.9 rtems_filesystem_default_lock() . . . 364
8.66.3.10 rtems_filesystem_default_mknod() . . . 364
8.66.3.11 rtems_filesystem_default_mount() . . . 364
8.66.3.12 rtems_filesystem_default_readlink() . . . 365
8.66.3.13 rtems_filesystem_default_rename() . . . 365
8.66.3.14 rtems_filesystem_default_rmnod() . . . 365
8.66.3.15 rtems_filesystem_default_statvfs() . . . 366
8.66.3.16 rtems_filesystem_default_symlink() . . . 366
8.66.3.17 rtems_filesystem_default_unlock() . . . 366
8.66.3.18 rtems_filesystem_default_unmount() . . . 367
8.66.3.19 rtems_filesystem_default_utime() . . . 367
8.67 File System Types and Mount . . . 368
8.67.1 Detailed Description . . . 369
8.67.2 Typedef Documentation . . . 369
8.67.2.1 rtems_filesystem_mt_entry_visitor . . . 369
8.67.2.2 rtems_per_filesystem_routine . . . 370
8.67.3 Function Documentation . . . 370
8.67.3.1 mount() . . . 370
8.67.3.2 mount_and_make_target_path() . . . 371
8.67.3.3 rtems_filesystem_iterate() . . . 372
8.67.3.4 rtems_filesystem_mount_iterate() . . . 372
8.67.3.5 rtems_filesystem_register() . . . 373

8.67.3.6 rtems_filesystem_unregister() . . . 373
8.67.3.7 unmount() . . . 373
8.67.4 Variable Documentation . . . 374
8.67.4.1 rtems_filesystem_table . . . 374
8.68 Filesystem Configuration . . . 375
8.68.1 Detailed Description . . . 376
8.68.2 Macro Definition Documentation . . . 376
8.68.2.1 CONFIGURE_APPLICATION_DISABLE_FILESYSTEM . . . 377
8.68.2.2 CONFIGURE_FILESYSTEM_ALL . . . 377
8.68.2.3 CONFIGURE_FILESYSTEM_DOSFS . . . 378
8.68.2.4 CONFIGURE_FILESYSTEM_FTPFS . . . 378
8.68.2.5 CONFIGURE_FILESYSTEM_IMFS . . . 378
8.68.2.6 CONFIGURE_FILESYSTEM_JFFS2 . . . 379
8.68.2.7 CONFIGURE_FILESYSTEM_NFS . . . 379
8.68.2.8 CONFIGURE_FILESYSTEM_RFS . . . 379
8.68.2.9 CONFIGURE_FILESYSTEM_TFTPFS . . . 380
8.68.2.10 CONFIGURE_IMFS_DISABLE_CHMOD . . . 380
8.68.2.11 CONFIGURE_IMFS_DISABLE_CHOWN . . . 380
8.68.2.12 CONFIGURE_IMFS_DISABLE_LINK . . . 381
8.68.2.13 CONFIGURE_IMFS_DISABLE_MKNOD . . . 381
8.68.2.14 CONFIGURE_IMFS_DISABLE_MKNOD_DEVICE . . . 381
8.68.2.15 CONFIGURE_IMFS_DISABLE_MKNOD_FILE . . . 382
8.68.2.16 CONFIGURE_IMFS_DISABLE_MOUNT . . . 382
8.68.2.17 CONFIGURE_IMFS_DISABLE_READDIR . . . 382
8.68.2.18 CONFIGURE_IMFS_DISABLE_READLINK . . . 383
8.68.2.19 CONFIGURE_IMFS_DISABLE_RENAME . . . 383
8.68.2.20 CONFIGURE_IMFS_DISABLE_RMNOD . . . 383
8.68.2.21 CONFIGURE_IMFS_DISABLE_SYMLINK . . . 384
8.68.2.22 CONFIGURE_IMFS_DISABLE_UNMOUNT . . . 384
8.68.2.23 CONFIGURE_IMFS_DISABLE_UTIME . . . 384
8.68.2.24 CONFIGURE_IMFS_ENABLE_MKFIFO . . . 385
8.68.2.25 CONFIGURE_IMFS_MEMFILE_BYTES_PER_BLOCK . . . 385
8.68.2.26 CONFIGURE_USE_DEVFS_AS_BASE_FILESYSTEM . . . 386
8.68.2.27 CONFIGURE_USE_MINIIMFS_AS_BASE_FILESYSTEM . . . 386
8.69 Flexible Per-CPU Data . . . 388
8.69.1 Detailed Description . . . 388
8.69.2 Macro Definition Documentation . . . 388
8.69.2.1 PER_CPU_DATA_GET . . . 388
8.69.2.2 PER_CPU_DATA_GET_BY_OFFSET . . . 389
8.69.2.3 PER_CPU_DATA_ITEM . . . 389
8.69.2.4 PER_CPU_DATA_ITEM_DECLARE . . . 389
8.69.2.5 PER_CPU_DATA_OFFSET . . . 390

8.70 Free-Running Counter and Busy Wait Delay . . . 391
8.70.1 Detailed Description . . . 391
8.70.2 Function Documentation . . . 392
8.70.2.1 rtems_counter_delay_nanoseconds() . . . 392
8.70.2.2 rtems_counter_delay_ticks() . . . 392
8.70.2.3 rtems_counter_difference() . . . 392
8.70.2.4 rtems_counter_frequency() . . . 393
8.70.2.5 rtems_counter_initialize_converter() . . . 393
8.70.2.6 rtems_counter_nanoseconds_to_ticks() . . . 393
8.70.2.7 rtems_counter_read() . . . 395
8.70.2.8 rtems_counter_sbintime_to_ticks() . . . 395
8.70.2.9 rtems_counter_ticks_to_nanoseconds() . . . 395
8.70.2.10 rtems_counter_ticks_to_sbintime() . . . 396
8.71 Freechain Handler . . . 397
8.71.1 Detailed Description . . . 397
8.71.2 Function Documentation . . . 398
8.71.2.1 _Freechain_Extend() . . . 398
8.71.2.2 _Freechain_Get() . . . 398
8.71.2.3 _Freechain_Initialize() . . . 399
8.71.2.4 _Freechain_Is_empty() . . . 399
8.71.2.5 _Freechain_Pop() . . . 399
8.71.2.6 _Freechain_Push() . . . 400
8.71.2.7 _Freechain_Put() . . . 400
8.72 GRLIB . . . 401
8.72.1 Detailed Description . . . 401
8.73 General Scheduler Configuration . . . 402
8.73.1 Detailed Description . . . 402
8.73.2 Macro Definition Documentation . . . 403
8.73.2.1 CONFIGURE_CBS_MAXIMUM_SERVERS . . . 403
8.73.2.2 CONFIGURE_MAXIMUM_PRIORITY . . . 403
8.73.2.3 CONFIGURE_SCHEDULER_ASSIGNMENTS . . . 404
8.73.2.4 CONFIGURE_SCHEDULER_CBS . . . 405
8.73.2.5 CONFIGURE_SCHEDULER_EDF . . . 405
8.73.2.6 CONFIGURE_SCHEDULER_EDF_SMP . . . 406
8.73.2.7 CONFIGURE_SCHEDULER_NAME . . . 406
8.73.2.8 CONFIGURE_SCHEDULER_PRIORITY . . . 407
8.73.2.9 CONFIGURE_SCHEDULER_PRIORITY_AFFINITY_SMP . . . 407
8.73.2.10 CONFIGURE_SCHEDULER_PRIORITY_SMP . . . 408
8.73.2.11 CONFIGURE_SCHEDULER_SIMPLE . . . 408
8.73.2.12 CONFIGURE_SCHEDULER_SIMPLE_SMP . . . 409
8.73.2.13 CONFIGURE_SCHEDULER_STRONG_APA . . . 409
8.73.2.14 CONFIGURE_SCHEDULER_USER . . . 410

8.74 General System Configuration . . . 411
8.74.1 Detailed Description . . . 411
8.74.2 Macro Definition Documentation . . . 412
8.74.2.1 CONFIGURE_DIRTY_MEMORY . . . 412
8.74.2.2 CONFIGURE_DISABLE_NEWLIB_REENTRANCY . . . 412
8.74.2.3 CONFIGURE_EXECUTIVE_RAM_SIZE . . . 413
8.74.2.4 CONFIGURE_EXTRA_TASK_STACKS . . . 413
8.74.2.5 CONFIGURE_INITIAL_EXTENSIONS . . . 414
8.74.2.6 CONFIGURE_INTERRUPT_STACK_SIZE . . . 414
8.74.2.7 CONFIGURE_MALLOC_DIRTY . . . 415
8.74.2.8 CONFIGURE_MAXIMUM_FILE_DESCRIPTORS . . . 415
8.74.2.9 CONFIGURE_MAXIMUM_PROCESSORS . . . 416
8.74.2.10 CONFIGURE_MAXIMUM_THREAD_NAME_SIZE . . . 416
8.74.2.11 CONFIGURE_MEMORY_OVERHEAD . . . 417
8.74.2.12 CONFIGURE_MESSAGE_BUFFER_MEMORY . . . 417
8.74.2.13 CONFIGURE_MICROSECONDS_PER_TICK . . . 418
8.74.2.14 CONFIGURE_MINIMUM_TASK_STACK_SIZE . . . 419
8.74.2.15 CONFIGURE_STACK_CHECKER_ENABLED . . . 420
8.74.2.16 CONFIGURE_TICKS_PER_TIMESLICE . . . 420
8.74.2.17 CONFIGURE_UNIFIED_WORK_AREAS . . . 421
8.74.2.18 CONFIGURE_UNLIMITED_ALLOCATION_SIZE . . . 421
8.74.2.19 CONFIGURE_UNLIMITED_OBJECTS . . . 422
8.74.2.20 CONFIGURE_VERBOSE_SYSTEM_INITIALIZATION . . . 422
8.74.2.21 CONFIGURE_ZERO_WORKSPACE_AUTOMATICALLY . . . 422
8.75 Generic Checks . . . 423
8.75.1 Detailed Description . . . 423
8.76 Heap Handler . . . 424
8.76.1 Detailed Description . . . 427
8.76.2 Typedef Documentation . . . 428
8.76.2.1 Heap_Block_visitor . . . 428
8.76.2.2 Heap_Initialization_or_extend_handler . . . 429
8.76.3 Enumeration Type Documentation . . . 429
8.76.3.1 Heap_Error_reason . . . 429
8.76.4 Function Documentation . . . 430
8.76.4.1 _Heap_Align_down() . . . 430
8.76.4.2 _Heap_Align_up() . . . 430
8.76.4.3 _Heap_Alloc_area_of_block() . . . 431
8.76.4.4 _Heap_Allocate() . . . 431
8.76.4.5 _Heap_Allocate_aligned() . . . 431
8.76.4.6 _Heap_Allocate_aligned_with_boundary() . . . 432
8.76.4.7 _Heap_Area_overhead() . . . 433
8.76.4.8 _Heap_Block_allocate() . . . 433

8.76.4.9 _Heap_Block_at() . . . 434
8.76.4.10 _Heap_Block_of_alloc_area() . . . 434
8.76.4.11 _Heap_Block_set_size() . . . 434
8.76.4.12 _Heap_Block_size() . . . 435
8.76.4.13 _Heap_Extend() . . . 435
8.76.4.14 _Heap_Free() . . . 436
8.76.4.15 _Heap_Free_list_first() . . . 436
8.76.4.16 _Heap_Free_list_head() . . . 437
8.76.4.17 _Heap_Free_list_insert_after() . . . 437
8.76.4.18 _Heap_Free_list_insert_before() . . . 437
8.76.4.19 _Heap_Free_list_last() . . . 438
8.76.4.20 _Heap_Free_list_remove() . . . 438
8.76.4.21 _Heap_Free_list_replace() . . . 438
8.76.4.22 _Heap_Free_list_tail() . . . 439
8.76.4.23 _Heap_Get_first_and_last_block() . . . 439
8.76.4.24 _Heap_Get_free_information() . . . 440
8.76.4.25 _Heap_Get_information() . . . 440
8.76.4.26 _Heap_Get_size() . . . 440
8.76.4.27 _Heap_Greedy_allocate() . . . 441
8.76.4.28 _Heap_Greedy_allocate_all_except_largest() . . . 441
8.76.4.29 _Heap_Greedy_free() . . . 442
8.76.4.30 _Heap_Initialize() . . . 442
8.76.4.31 _Heap_Is_aligned() . . . 443
8.76.4.32 _Heap_Is_block_in_heap() . . . 443
8.76.4.33 _Heap_Is_free() . . . 444
8.76.4.34 _Heap_Is_prev_used() . . . 444
8.76.4.35 _Heap_Is_used() . . . 445
8.76.4.36 _Heap_Iterate() . . . 445
8.76.4.37 _Heap_Max() . . . 445
8.76.4.38 _Heap_Min() . . . 446
8.76.4.39 _Heap_Min_block_size() . . . 446
8.76.4.40 _Heap_No_extend() . . . 447
8.76.4.41 _Heap_Prev_block() . . . 447
8.76.4.42 _Heap_Protection_set_delayed_free_fraction() . . . 448
8.76.4.43 _Heap_Resize_block() . . . 448
8.76.4.44 _Heap_Set_last_block_size() . . . 449
8.76.4.45 _Heap_Size_of_alloc_area() . . . 449
8.76.4.46 _Heap_Size_with_overhead() . . . 449
8.76.4.47 _Heap_Walk() . . . 450
8.77 Helpers . . . 451
8.77.1 Detailed Description . . . 452
8.77.2 Macro Definition Documentation . . . 452

8.77.2.1 _Timespec_Equal_to . . . 452
8.77.2.2 _Timespec_Get_nanoseconds . . . 452
8.77.2.3 _Timespec_Get_seconds . . . 453
8.77.2.4 _Timespec_Greater_than . . . 453
8.77.2.5 _Timespec_Set . . . 454
8.77.2.6 _Timespec_Set_to_zero . . . 454
8.77.3 Function Documentation . . . 455
8.77.3.1 _Timespec_Add_to() . . . 455
8.77.3.2 _Timespec_Divide() . . . 455
8.77.3.3 _Timespec_Divide_by_integer() . . . 456
8.77.3.4 _Timespec_From_ticks() . . . 456
8.77.3.5 _Timespec_Get_as_nanoseconds() . . . 456
8.77.3.6 _Timespec_Is_valid() . . . 457
8.77.3.7 _Timespec_Less_than() . . . 457
8.77.3.8 _Timespec_Subtract() . . . 458
8.77.3.9 _Timespec_To_ticks() . . . 458
8.78 I/O Manager . . . 459
8.78.1 Detailed Description . . . 460
8.78.2 Typedef Documentation . . . 460
8.78.2.1 rtems_device_driver . . . 460
8.78.2.2 rtems_device_major_number . . . 460
8.78.2.3 rtems_device_minor_number . . . 460
8.78.3 Function Documentation . . . 460
8.78.3.1 rtems_io_close() . . . 460
8.78.3.2 rtems_io_control() . . . 461
8.78.3.3 rtems_io_initialize() . . . 462
8.78.3.4 rtems_io_open() . . . 462
8.78.3.5 rtems_io_read() . . . 463
8.78.3.6 rtems_io_register_driver() . . . 463
8.78.3.7 rtems_io_register_name() . . . 464
8.78.3.8 rtems_io_unregister_driver() . . . 465
8.78.3.9 rtems_io_write() . . . 465
8.79 IO . . . 467
8.79.1 Detailed Description . . . 467
8.80 IO Library . . . 468
8.80.1 Detailed Description . . . 470
8.80.2 Macro Definition Documentation . . . 470
8.80.2.1 rtems_filesystem_split_dev_t . . . 470
8.80.3 Typedef Documentation . . . 470
8.80.3.1 rtems_filesystem_global_location_t . . . 471
8.80.4 Function Documentation . . . 471
8.80.4.1 rtems_filesystem_get_mount_handler() . . . 471

8.80.4.2 rtems_filesystem_initialize() . . . 471
8.80.4.3 rtems_libio_iop_is_append() . . . 471
8.80.4.4 rtems_libio_iop_is_no_delay() . . . 472
8.80.4.5 rtems_libio_iop_is_readable() . . . 472
8.80.4.6 rtems_libio_iop_is_writeable() . . . 472
8.80.4.7 rtems_mkdir() . . . 473
8.80.5 Variable Documentation . . . 473
8.80.5.1 rtems_filesystem_default_pathconf . . . 473
8.81 ISR Handler . . . 474
8.81.1 Detailed Description . . . 475
8.81.2 Macro Definition Documentation . . . 475
8.81.2.1 _ISR_Get_level . . . 475
8.81.2.2 _ISR_Install_vector . . . 476
8.81.2.3 _ISR_Is_enabled . . . 476
8.81.2.4 _ISR_Local_disable . . . 477
8.81.2.5 _ISR_Local_enable . . . 477
8.81.2.6 _ISR_Local_flash . . . 477
8.81.2.7 _ISR_Set_level . . . 478
8.81.3 Typedef Documentation . . . 478
8.81.3.1 ISR_Handler . . . 479
8.81.3.2 ISR_Level . . . 479
8.81.3.3 ISR_Vector_number . . . 479
8.81.4 Function Documentation . . . 479
8.81.4.1 _ISR_Handler() . . . 479
8.81.4.2 _ISR_Handler_initialization() . . . 480
8.81.4.3 _ISR_Is_in_progress() . . . 480
8.81.4.4 RTEMS_DECLARE_GLOBAL_SYMBOL() . . . 480
8.81.5 Variable Documentation . . . 480
8.81.5.1 _ISR_Stack_area_begin . . . 481
8.81.5.2 _ISR_Stack_area_end . . . 481
8.82 ISR Locks . . . 482
8.82.1 Detailed Description . . . 483
8.82.2 Macro Definition Documentation . . . 483
8.82.2.1 _ISR_lock_Acquire . . . 483
8.82.2.2 _ISR_lock_Acquire_inline . . . 484
8.82.2.3 _ISR_lock_Destroy . . . 484
8.82.2.4 _ISR_lock_Initialize . . . 484
8.82.2.5 _ISR_lock_ISR_disable . . . 485
8.82.2.6 _ISR_lock_ISR_disable_and_acquire . . . 485
8.82.2.7 _ISR_lock_ISR_enable . . . 486
8.82.2.8 _ISR_lock_Release . . . 486
8.82.2.9 _ISR_lock_Release_and_ISR_enable . . . 487

8.82.2.10 _ISR_lock_Release_inline . . . 487
8.82.2.11 _ISR_lock_Set_name . . . 488
8.82.2.12 ISR_LOCK_DECLARE . . . 488
8.82.2.13 ISR_LOCK_DEFINE . . . 488
8.82.2.14 ISR_LOCK_INITIALIZER . . . 489
8.82.2.15 ISR_LOCK_MEMBER . . . 489
8.82.2.16 ISR_LOCK_REFERENCE . . . 490
8.82.3 Function Documentation . . . 490
8.82.3.1 _ISR_lock_Context_set_level() . . . 490
8.83 Idle Task Configuration . . . 491
8.83.1 Detailed Description . . . 491
8.83.2 Macro Definition Documentation . . . 491
8.83.2.1 CONFIGURE_IDLE_TASK_BODY . . . 491
8.83.2.2 CONFIGURE_IDLE_TASK_INITIALIZES_APPLICATION . . . 492
8.83.2.3 CONFIGURE_IDLE_TASK_STACK_SIZE . . . 492
8.84 Implementation . . . 493
8.84.1 Detailed Description . . . 493
8.85 Initialization and Shutdown . . . 494
8.85.1 Detailed Description . . . 494
8.85.2 Function Documentation . . . 494
8.85.2.1 rtems_initialize_executive() . . . 494
8.85.2.2 rtems_shutdown_executive() . . . 494
8.86 Internal Error Handler . . . 496
8.86.1 Detailed Description . . . 497
8.86.2 Enumeration Type Documentation . . . 497
8.86.2.1 Internal_errors_Core_list . . . 497
8.86.2.2 Internal_errors_Source . . . 498
8.86.3 Function Documentation . . . 499
8.86.3.1 _Internal_error() . . . 499
8.86.3.2 _Terminate() . . . 499
8.86.4 Variable Documentation . . . 500
8.86.4.1 _Internal_errors_What_happened . . . 500
8.87 Interrupt Manager . . . 501
8.87.1 Detailed Description . . . 502
8.87.2 Macro Definition Documentation . . . 502
8.87.2.1 rtems_interrupt_cause . . . 502
8.87.2.2 rtems_interrupt_clear . . . 503
8.87.2.3 rtems_interrupt_local_disable . . . 503
8.87.2.4 rtems_interrupt_local_enable . . . 503
8.87.2.5 rtems_interrupt_lock_acquire . . . 503
8.87.2.6 rtems_interrupt_lock_acquire_isr . . . 504
8.87.2.7 RTEMS_INTERRUPT_LOCK_DECLARE . . . 504

8.87.2.8 RTEMS_INTERRUPT_LOCK_DEFINE . . . 505
8.87.2.9 rtems_interrupt_lock_destroy . . . 505
8.87.2.10 rtems_interrupt_lock_initialize . . . 505
8.87.2.11 RTEMS_INTERRUPT_LOCK_INITIALIZER . . . 506
8.87.2.12 rtems_interrupt_lock_interrupt_disable . . . 506
8.87.2.13 RTEMS_INTERRUPT_LOCK_MEMBER . . . 506
8.87.2.14 RTEMS_INTERRUPT_LOCK_REFERENCE . . . 506
8.87.2.15 rtems_interrupt_lock_release . . . 507
8.87.2.16 rtems_interrupt_lock_release_isr . . . 507
8.87.3 Function Documentation . . . 508
8.87.3.1 rtems_interrupt_catch() . . . 508
8.88 Interrupt Manager Extension . . . 509
8.88.1 Detailed Description . . . 511
8.88.2 Typedef Documentation . . . 511
8.88.2.1 rtems_interrupt_per_handler_routine . . . 511
8.88.2.2 rtems_interrupt_server_action . . . 512
8.88.2.3 rtems_interrupt_server_control . . . 512
8.88.3 Function Documentation . . . 512
8.88.3.1 rtems_interrupt_get_affinity() . . . 512
8.88.3.2 rtems_interrupt_handler_install() . . . 513
8.88.3.3 rtems_interrupt_handler_iterate() . . . 514
8.88.3.4 rtems_interrupt_handler_remove() . . . 514
8.88.3.5 rtems_interrupt_server_action_prepend() . . . 515
8.88.3.6 rtems_interrupt_server_create() . . . 515
8.88.3.7 rtems_interrupt_server_delete() . . . 516
8.88.3.8 rtems_interrupt_server_entry_destroy() . . . 516
8.88.3.9 rtems_interrupt_server_entry_initialize() . . . 517
8.88.3.10 rtems_interrupt_server_entry_move() . . . 517
8.88.3.11 rtems_interrupt_server_entry_submit() . . . 518
8.88.3.12 rtems_interrupt_server_handler_install() . . . 518
8.88.3.13 rtems_interrupt_server_handler_iterate() . . . 519
8.88.3.14 rtems_interrupt_server_handler_remove() . . . 519
8.88.3.15 rtems_interrupt_server_initialize() . . . 520
8.88.3.16 rtems_interrupt_server_move() . . . 521
8.88.3.17 rtems_interrupt_server_request_destroy() . . . 521
8.88.3.18 rtems_interrupt_server_request_initialize() . . . 522
8.88.3.19 rtems_interrupt_server_request_set_vector() . . . 522
8.88.3.20 rtems_interrupt_server_request_submit() . . . 523
8.88.3.21 rtems_interrupt_server_resume() . . . 523
8.88.3.22 rtems_interrupt_server_set_affinity() . . . 524
8.88.3.23 rtems_interrupt_server_suspend() . . . 524
8.88.3.24 rtems_interrupt_set_affinity() . . . 525

8.89 Kernel Print Support . . . 526
8.90 LEON3 AMBA Driver Handler . . . 527
8.90.1 Detailed Description . . . 527
8.91 LEON3 and LEON4 . . . 528
8.91.1 Detailed Description . . . 529
8.92 Legacy Benchmark Drivers . . . 530
8.93 Local Packages . . . 531
8.94 MP Packet Handler . . . 532
8.94.1 Detailed Description . . . 532
8.94.2 Macro Definition Documentation . . . 532
8.94.2.1 MP_PACKET_CLASSES_FIRST . . . 532
8.94.2.2 MP_PACKET_CLASSES_LAST . . . 533
8.94.2.3 MP_PACKET_MINIMUM_PACKET_SIZE . . . 533
8.94.2.4 MP_PACKET_MINIMUN_HETERO_CONVERSION . . . 533
8.94.3 Enumeration Type Documentation . . . 533
8.94.3.1 MP_packet_Classes . . . 533
8.95 Malloc Support . . . 534
8.95.1 Detailed Description . . . 534
8.96 Memory Area Checks . . . 535
8.96.1 Detailed Description . . . 535
8.97 Memory Handler . . . 536
8.97.1 Detailed Description . . . 537
8.97.2 Macro Definition Documentation . . . 537
8.97.2.1 MEMORY_INFORMATION_INITIALIZER . . . 537
8.97.2.2 MEMORY_INITIALIZER . . . 537
8.97.3 Function Documentation . . . 538
8.97.3.1 _Memory_Allocate() . . . 538
8.97.3.2 _Memory_Consume() . . . 538
8.97.3.3 _Memory_Fill() . . . 539
8.97.3.4 _Memory_Get() . . . 539
8.97.3.5 _Memory_Get_area() . . . 539
8.97.3.6 _Memory_Get_begin() . . . 540
8.97.3.7 _Memory_Get_count() . . . 540
8.97.3.8 _Memory_Get_end() . . . 541
8.97.3.9 _Memory_Get_free_begin() . . . 541
8.97.3.10 _Memory_Get_free_size() . . . 541
8.97.3.11 _Memory_Get_size() . . . 542
8.97.3.12 _Memory_Initialize() . . . 542
8.97.3.13 _Memory_Initialize_by_size() . . . 542
8.97.3.14 _Memory_Set_begin() . . . 543
8.97.3.15 _Memory_Set_end() . . . 543
8.97.3.16 _Memory_Set_free_begin() . . . 544

8.97.4 Variable Documentation . . . 544
8.97.4.1 _Memory_Zero_before_use . . . 544
8.98 Message Manager . . . 545
8.98.1 Detailed Description . . . 545
8.98.2 Macro Definition Documentation . . . 546
8.98.2.1 RTEMS_MESSAGE_QUEUE_BUFFER . . . 546
8.98.3 Function Documentation . . . 546
8.98.3.1 rtems_message_queue_broadcast() . . . 546
8.98.3.2 rtems_message_queue_construct() . . . 547
8.98.3.3 rtems_message_queue_create() . . . 548
8.98.3.4 rtems_message_queue_delete() . . . 548
8.98.3.5 rtems_message_queue_flush() . . . 548
8.98.3.6 rtems_message_queue_get_number_pending() . . . 549
8.98.3.7 rtems_message_queue_ident() . . . 549
8.98.3.8 rtems_message_queue_receive() . . . 550
8.98.3.9 rtems_message_queue_send() . . . 550
8.98.3.10 rtems_message_queue_urgent() . . . 551
8.99 Message Queue Handler . . . 552
8.99.1 Detailed Description . . . 554
8.99.2 Macro Definition Documentation . . . 554
8.99.2.1 _CORE_message_queue_Set_notify . . . 554
8.99.2.2 CORE_MESSAGE_QUEUE_SEND_REQUEST . . . 555
8.99.2.3 CORE_MESSAGE_QUEUE_URGENT_REQUEST . . . 555
8.99.2.4 RTEMS_SCORE_COREMSG_ENABLE_BLOCKING_SEND . . . 555
8.99.2.5 RTEMS_SCORE_COREMSG_ENABLE_MESSAGE_PRIORITY . . . 555
8.99.3 Typedef Documentation . . . 555
8.99.3.1 CORE_message_queue_Allocate_buffers . . . 555
8.99.3.2 CORE_message_queue_Submit_types . . . 556
8.99.4 Enumeration Type Documentation . . . 556
8.99.4.1 CORE_message_queue_Disciplines . . . 556
8.99.5 Function Documentation . . . 557
8.99.5.1 _CORE_message_queue_Acquire() . . . 557
8.99.5.2 _CORE_message_queue_Acquire_critical() . . . 557
8.99.5.3 _CORE_message_queue_Allocate_message_buffer() . . . 558
8.99.5.4 _CORE_message_queue_Broadcast() . . . 558
8.99.5.5 _CORE_message_queue_Close() . . . 559
8.99.5.6 _CORE_message_queue_Copy_buffer() . . . 559
8.99.5.7 _CORE_message_queue_Dequeue_receiver() . . . 560
8.99.5.8 _CORE_message_queue_Flush() . . . 560
8.99.5.9 _CORE_message_queue_Free_message_buffer() . . . 561
8.99.5.10 _CORE_message_queue_Get_message_priority() . . . 561
8.99.5.11 _CORE_message_queue_Get_pending_message() . . . 562

8.99.5.12 _CORE_message_queue_Initialize() . . . 562
8.99.5.13 _CORE_message_queue_Insert_message() . . . 563
8.99.5.14 _CORE_message_queue_Release() . . . 563
8.99.5.15 _CORE_message_queue_Seize() . . . 564
8.99.5.16 _CORE_message_queue_Send() . . . 565
8.99.5.17 _CORE_message_queue_Submit() . . . 565
8.99.5.18 _CORE_message_queue_Urgent() . . . 566
8.99.5.19 _CORE_message_queue_Workspace_allocate() . . . 567
8.100 Multiprocessing Configuration . . . 568
8.100.1 Detailed Description . . . 568
8.100.2 Macro Definition Documentation . . . 568
8.100.2.1 CONFIGURE_EXTRA_MPCI_RECEIVE_SERVER_STACK . . . 568
8.100.2.2 CONFIGURE_MP_APPLICATION . . . 569
8.100.2.3 CONFIGURE_MP_MAXIMUM_GLOBAL_OBJECTS . . . 569
8.100.2.4 CONFIGURE_MP_MAXIMUM_NODES . . . 570
8.100.2.5 CONFIGURE_MP_MAXIMUM_PROXIES . . . 570
8.100.2.6 CONFIGURE_MP_MPCI_TABLE_POINTER . . . 571
8.100.2.7 CONFIGURE_MP_NODE_NUMBER . . . 571
8.101 Multiprocessor Resource Sharing Protocol Handler . . . 572
8.101.1 Detailed Description . . . 573
8.101.2 Function Documentation . . . 573
8.101.2.1 _MRSP_Acquire_critical() . . . 573
8.101.2.2 _MRSP_Can_destroy() . . . 574
8.101.2.3 _MRSP_Claim_ownership() . . . 574
8.101.2.4 _MRSP_Destroy() . . . 574
8.101.2.5 _MRSP_Get_owner() . . . 575
8.101.2.6 _MRSP_Get_priority() . . . 575
8.101.2.7 _MRSP_Initialize() . . . 576
8.101.2.8 _MRSP_Raise_priority() . . . 576
8.101.2.9 _MRSP_Release() . . . 577
8.101.2.10 _MRSP_Remove_priority() . . . 577
8.101.2.11 _MRSP_Replace_priority() . . . 577
8.101.2.12 _MRSP_Seize() . . . 578
8.101.2.13 _MRSP_Set_owner() . . . 578
8.101.2.14 _MRSP_Set_priority() . . . 579
8.101.2.15 _MRSP_Surrender() . . . 579
8.101.2.16 _MRSP_Wait_for_ownership() . . . 580
8.102 Mutex Handler . . . 581
8.102.1 Detailed Description . . . 582
8.102.2 Function Documentation . . . 582
8.102.2.1 _CORE_ceiling_mutex_Get_priority() . . . 583
8.102.2.2 _CORE_ceiling_mutex_Get_scheduler() . . . 583

8.102.2.3 _CORE_ceiling_mutex_Initialize() . . . 583
8.102.2.4 _CORE_ceiling_mutex_Seize() . . . 584
8.102.2.5 _CORE_ceiling_mutex_Set_owner() . . . 584
8.102.2.6 _CORE_ceiling_mutex_Set_priority() . . . 585
8.102.2.7 _CORE_ceiling_mutex_Surrender() . . . 585
8.102.2.8 _CORE_mutex_Acquire_critical() . . . 586
8.102.2.9 _CORE_mutex_Destroy() . . . 586
8.102.2.10 _CORE_mutex_Get_owner() . . . 586
8.102.2.11 _CORE_mutex_Initialize() . . . 587
8.102.2.12 _CORE_mutex_Is_locked() . . . 587
8.102.2.13 _CORE_mutex_Is_owner() . . . 588
8.102.2.14 _CORE_mutex_Release() . . . 588
8.102.2.15 _CORE_mutex_Seize_slow() . . . 588
8.102.2.16 _CORE_mutex_Set_owner() . . . 589
8.102.2.17 _CORE_recursive_mutex_Initialize() . . . 589
8.102.2.18 _CORE_recursive_mutex_Seize() . . . 590
8.102.2.19 _CORE_recursive_mutex_Seize_nested() . . . 590
8.102.2.20 _CORE_recursive_mutex_Surrender() . . . 591
8.103 Object Handler . . . 592
8.103.1 Detailed Description . . . 596
8.103.2 Macro Definition Documentation . . . 596
8.103.2.1 _Objects_Build_id . . . 597
8.103.2.2 _Objects_Build_name . . . 597
8.103.2.3 _Objects_Is_unlimited . . . 598
8.103.2.4 _Objects_Maximum_nodes . . . 598
8.103.2.5 OBJECTS_API_MASK . . . 598
8.103.2.6 OBJECTS_API_START_BIT . . . 598
8.103.2.7 OBJECTS_API_VALID_BITS . . . 599
8.103.2.8 OBJECTS_APIS_LAST . . . 599
8.103.2.9 OBJECTS_CLASS_MASK . . . 599
8.103.2.10 OBJECTS_CLASS_START_BIT . . . 599
8.103.2.11 OBJECTS_CLASS_VALID_BITS . . . 599
8.103.2.12 OBJECTS_ID_FINAL . . . 600
8.103.2.13 OBJECTS_ID_FINAL_INDEX . . . 600
8.103.2.14 OBJECTS_ID_INITIAL . . . 600
8.103.2.15 OBJECTS_ID_INITIAL_INDEX . . . 600
8.103.2.16 OBJECTS_ID_NONE . . . 600
8.103.2.17 OBJECTS_ID_OF_SELF . . . 601
8.103.2.18 OBJECTS_INDEX_MASK . . . 601
8.103.2.19 OBJECTS_INDEX_MINIMUM . . . 601
8.103.2.20 OBJECTS_INDEX_START_BIT . . . 601
8.103.2.21 OBJECTS_INDEX_VALID_BITS . . . 601

8.103.2.22 OBJECTS_INFORMATION_DEFINE . . . 602
8.103.2.23 OBJECTS_INFORMATION_DEFINE_ZERO . . . 602
8.103.2.24 OBJECTS_INTERNAL_CLASSES_LAST . . . 603
8.103.2.25 OBJECTS_NAME_ERRORS_FIRST . . . 603
8.103.2.26 OBJECTS_NAME_ERRORS_LAST . . . 603
8.103.2.27 OBJECTS_NODE_MASK . . . 604
8.103.2.28 OBJECTS_NODE_START_BIT . . . 604
8.103.2.29 OBJECTS_NODE_VALID_BITS . . . 604
8.103.2.30 OBJECTS_POSIX_CLASSES_LAST . . . 604
8.103.2.31 OBJECTS_RTEMS_CLASSES_LAST . . . 604
8.103.2.32 OBJECTS_SEARCH_ALL_NODES . . . 605
8.103.2.33 OBJECTS_SEARCH_LOCAL_NODE . . . 605
8.103.2.34 OBJECTS_SEARCH_OTHER_NODES . . . 605
8.103.2.35 OBJECTS_UNLIMITED_OBJECTS . . . 605
8.103.2.36 OBJECTS_WHO_AM_I . . . 605
8.103.3 Typedef Documentation . . . 606
8.103.3.1 Objects_Id . . . 606
8.103.3.2 Objects_Maximum . . . 606
8.103.3.3 Objects_Name_comparators . . . 606
8.103.4 Enumeration Type Documentation . . . 606
8.103.4.1 Objects_APIs . . . 606
8.103.4.2 Objects_Classic_API . . . 607
8.103.4.3 Objects_Internal_API . . . 607
8.103.4.4 Objects_Name_or_id_lookup_errors . . . 607
8.103.4.5 Objects_POSIX_API . . . 607
8.103.5 Function Documentation . . . 607
8.103.5.1 _Objects_Activate_unlimited() . . . 607
8.103.5.2 _Objects_Active_count() . . . 608
8.103.5.3 _Objects_Allocate() . . . 608
8.103.5.4 _Objects_Allocate_none() . . . 609
8.103.5.5 _Objects_Allocate_static() . . . 609
8.103.5.6 _Objects_Allocate_unlimited() . . . 610
8.103.5.7 _Objects_Allocate_unprotected() . . . 610
8.103.5.8 _Objects_Allocate_with_extend() . . . 611
8.103.5.9 _Objects_Allocator_is_owner() . . . 611
8.103.5.10 _Objects_Allocator_lock() . . . 612
8.103.5.11 _Objects_Allocator_unlock() . . . 612
8.103.5.12 _Objects_API_maximum_class() . . . 612
8.103.5.13 _Objects_Are_ids_equal() . . . 613
8.103.5.14 _Objects_Close() . . . 613
8.103.5.15 _Objects_Extend_information() . . . 614
8.103.5.16 _Objects_Extend_size() . . . 614

8.103.5.17 _Objects_Free() . . . 614
8.103.5.18 _Objects_Free_objects_block() . . . 615
8.103.5.19 _Objects_Free_static() . . . 615
8.103.5.20 _Objects_Free_unlimited() . . . 616
8.103.5.21 _Objects_Get() . . . 616
8.103.5.22 _Objects_Get_API() . . . 617
8.103.5.23 _Objects_Get_by_name() . . . 617
8.103.5.24 _Objects_Get_class() . . . 618
8.103.5.25 _Objects_Get_inactive() . . . 618
8.103.5.26 _Objects_Get_index() . . . 618
8.103.5.27 _Objects_Get_information() . . . 619
8.103.5.28 _Objects_Get_information_id() . . . 619
8.103.5.29 _Objects_Get_maximum_index() . . . 620
8.103.5.30 _Objects_Get_minimum_id() . . . 620
8.103.5.31 _Objects_Get_name_as_string() . . . 621
8.103.5.32 _Objects_Get_next() . . . 621
8.103.5.33 _Objects_Get_no_protection() . . . 622
8.103.5.34 _Objects_Get_node() . . . 622
8.103.5.35 _Objects_Has_string_name() . . . 623
8.103.5.36 _Objects_Id_to_name() . . . 624
8.103.5.37 _Objects_Initialize_information() . . . 624
8.103.5.38 _Objects_Invalidate_Id() . . . 625
8.103.5.39 _Objects_Is_api_valid() . . . 625
8.103.5.40 _Objects_Is_auto_extend() . . . 626
8.103.5.41 _Objects_Is_local_id() . . . 626
8.103.5.42 _Objects_Is_local_node() . . . 627
8.103.5.43 _Objects_Name_to_id_u32() . . . 627
8.103.5.44 _Objects_Name_to_string() . . . 628
8.103.5.45 _Objects_Namespace_remove_string() . . . 628
8.103.5.46 _Objects_Namespace_remove_u32() . . . 628
8.103.5.47 _Objects_Open() . . . 629
8.103.5.48 _Objects_Open_string() . . . 629
8.103.5.49 _Objects_Open_u32() . . . 630
8.103.5.50 _Objects_Set_local_object() . . . 630
8.103.5.51 _Objects_Set_name() . . . 631
8.103.5.52 _Objects_Shrink_information() . . . 631
8.103.6 Variable Documentation . . . 631
8.103.6.1 _Objects_Information_table . . . 631
8.104 Object Services . . . 632
8.104.1 Detailed Description . . . 633
8.104.2 Macro Definition Documentation . . . 633
8.104.2.1 rtems_build_id . . . 633

8.104.2.2 rtems_build_name . . . 634
8.104.2.3 rtems_object_id_get_api . . . 634
8.104.2.4 rtems_object_id_get_class . . . 634
8.104.2.5 rtems_object_id_get_index . . . 635
8.104.2.6 rtems_object_id_get_node . . . 635
8.104.2.7 RTEMS_OBJECT_ID_INITIAL . . . 635
8.104.3 Function Documentation . . . 637
8.104.3.1 rtems_object_api_maximum_class() . . . 637
8.104.3.2 rtems_object_api_minimum_class() . . . 637
8.104.3.3 rtems_object_get_api_class_name() . . . 637
8.104.3.4 rtems_object_get_api_name() . . . 638
8.104.3.5 rtems_object_get_class_information() . . . 638
8.104.3.6 rtems_object_get_classic_name() . . . 638
8.104.3.7 rtems_object_get_name() . . . 639
8.104.3.8 rtems_object_id_api_maximum_class() . . . 639
8.104.3.9 rtems_object_set_name() . . . 639
8.105 POSIX API Configuration . . . 640
8.105.1 Detailed Description . . . 640
8.105.2 Macro Definition Documentation . . . 640
8.105.2.1 CONFIGURE_MAXIMUM_POSIX_KEY_VALUE_PAIRS . . . 640
8.105.2.2 CONFIGURE_MAXIMUM_POSIX_KEYS . . . 641
8.105.2.3 CONFIGURE_MAXIMUM_POSIX_MESSAGE_QUEUES . . . 642
8.105.2.4 CONFIGURE_MAXIMUM_POSIX_QUEUED_SIGNALS . . . 642
8.105.2.5 CONFIGURE_MAXIMUM_POSIX_SEMAPHORES . . . 643
8.105.2.6 CONFIGURE_MAXIMUM_POSIX_SHMS . . . 644
8.105.2.7 CONFIGURE_MAXIMUM_POSIX_THREADS . . . 644
8.105.2.8 CONFIGURE_MAXIMUM_POSIX_TIMERS . . . 645
8.105.2.9 CONFIGURE_MINIMUM_POSIX_THREAD_STACK_SIZE . . . 646
8.106 POSIX Initialization Thread Configuration . . . 647
8.106.1 Detailed Description . . . 647
8.106.2 Macro Definition Documentation . . . 647
8.106.2.1 CONFIGURE_POSIX_INIT_THREAD_ENTRY_POINT . . . 647
8.106.2.2 CONFIGURE_POSIX_INIT_THREAD_STACK_SIZE . . . 648
8.106.2.3 CONFIGURE_POSIX_INIT_THREAD_TABLE . . . 648
8.107 POSIX Status and Error Number Checks . . . 649
8.107.1 Detailed Description . . . 649
8.108 Partition Manager . . . 650
8.108.1 Detailed Description . . . 650
8.108.2 Macro Definition Documentation . . . 650
8.108.2.1 RTEMS_PARTITION_ALIGNMENT . . . 650
8.108.3 Function Documentation . . . 650
8.108.3.1 rtems_partition_create() . . . 651

8.108.3.2 rtems_partition_delete() . . . 652
8.108.3.3 rtems_partition_get_buffer() . . . 653
8.108.3.4 rtems_partition_ident() . . . 653
8.108.3.5 rtems_partition_return_buffer() . . . 654
8.109 Partition Manager Implementation . . . 656
8.109.1 Detailed Description . . . 656
8.109.2 Macro Definition Documentation . . . 656
8.109.2.1 PARTITION_INFORMATION_DEFINE . . . 657
8.109.3 Function Documentation . . . 657
8.109.3.1 _Partition_Acquire_critical() . . . 657
8.109.3.2 _Partition_Get() . . . 657
8.109.3.3 _Partition_Release() . . . 658
8.109.4 Variable Documentation . . . 658
8.109.4.1 _Partition_Information . . . 658
8.110 Pointer Checks . . . 659
8.110.1 Detailed Description . . . 659
8.111 Print Support . . . 660
8.111.1 Detailed Description . . . 660
8.112 Priority Handler . . . 661
8.112.1 Detailed Description . . . 664
8.112.2 Macro Definition Documentation . . . 664
8.112.2.1 PRIORITY_DEFAULT_MAXIMUM . . . 664
8.112.2.2 PRIORITY_PSEUDO_ISR . . . 664
8.112.3 Typedef Documentation . . . 664
8.112.3.1 Priority_Control . . . 665
8.112.4 Function Documentation . . . 665
8.112.4.1 _Bitfield_Find_first_bit() . . . 665
8.112.4.2 _Priority_Actions_add() . . . 666
8.112.4.3 _Priority_Actions_initialize_empty() . . . 666
8.112.4.4 _Priority_Actions_initialize_one() . . . 666
8.112.4.5 _Priority_Actions_is_empty() . . . 667
8.112.4.6 _Priority_Actions_is_valid() . . . 667
8.112.4.7 _Priority_Actions_move() . . . 667
8.112.4.8 _Priority_bit_map_Add() . . . 669
8.112.4.9 _Priority_bit_map_Get_highest() . . . 669
8.112.4.10 _Priority_bit_map_Initialize() . . . 670
8.112.4.11 _Priority_bit_map_Initialize_information() . . . 670
8.112.4.12 _Priority_bit_map_Is_empty() . . . 670
8.112.4.13 _Priority_bit_map_Remove() . . . 671
8.112.4.14 _Priority_Bits_index() . . . 671
8.112.4.15 _Priority_Change_nothing() . . . 671
8.112.4.16 _Priority_Changed() . . . 672

8.112.4.17 _Priority_Extract() . . . 672
8.112.4.18 _Priority_Extract_non_empty() . . . 674
8.112.4.19 _Priority_Get_minimum_node() . . . 674
8.112.4.20 _Priority_Get_next_action() . . . 675
8.112.4.21 _Priority_Get_priority() . . . 675
8.112.4.22 _Priority_Get_scheduler() . . . 676
8.112.4.23 _Priority_Initialize_empty() . . . 676
8.112.4.24 _Priority_Initialize_one() . . . 676
8.112.4.25 _Priority_Insert() . . . 677
8.112.4.26 _Priority_Is_empty() . . . 677
8.112.4.27 _Priority_Less() . . . 677
8.112.4.28 _Priority_Major() . . . 678
8.112.4.29 _Priority_Mask() . . . 678
8.112.4.30 _Priority_Minor() . . . 679
8.112.4.31 _Priority_Node_initialize() . . . 679
8.112.4.32 _Priority_Node_is_active() . . . 679
8.112.4.33 _Priority_Node_set_inactive() . . . 680
8.112.4.34 _Priority_Node_set_priority() . . . 680
8.112.4.35 _Priority_Non_empty_insert() . . . 680
8.112.4.36 _Priority_Plain_changed() . . . 681
8.112.4.37 _Priority_Plain_extract() . . . 681
8.112.4.38 _Priority_Plain_insert() . . . 682
8.112.4.39 _Priority_Remove_nothing() . . . 682
8.112.4.40 _Priority_Replace() . . . 683
8.112.4.41 _Priority_Set_action() . . . 683
8.112.4.42 _Priority_Set_action_node() . . . 683
8.112.4.43 _Priority_Set_action_type() . . . 684
8.112.5 Variable Documentation . . . 684
8.112.5.1 _Bitfield_Leading_zeros . . . 684
8.113 Processor Mask . . . 685
8.113.1 Detailed Description . . . 686
8.113.2 Function Documentation . . . 686
8.113.2.1 _Processor_mask_And() . . . 687
8.113.2.2 _Processor_mask_Assign() . . . 687
8.113.2.3 _Processor_mask_Clear() . . . 687
8.113.2.4 _Processor_mask_Copy() . . . 688
8.113.2.5 _Processor_mask_Count() . . . 688
8.113.2.6 _Processor_mask_Fill() . . . 689
8.113.2.7 _Processor_mask_Find_last_set() . . . 689
8.113.2.8 _Processor_mask_From_cpu_set_t() . . . 689
8.113.2.9 _Processor_mask_From_index() . . . 690
8.113.2.10 _Processor_mask_From_uint32_t() . . . 690

8.113.2.11 _Processor_mask_Has_overlap() . . . 691
8.113.2.12 _Processor_mask_Is_at_most_partial_loss() . . . 691
8.113.2.13 _Processor_mask_Is_equal() . . . 691
8.113.2.14 _Processor_mask_Is_set() . . . 692
8.113.2.15 _Processor_mask_Is_subset() . . . 692
8.113.2.16 _Processor_mask_Is_zero() . . . 693
8.113.2.17 _Processor_mask_Nand() . . . 693
8.113.2.18 _Processor_mask_Or() . . . 694
8.113.2.19 _Processor_mask_Set() . . . 694
8.113.2.20 _Processor_mask_To_cpu_set_t() . . . 694
8.113.2.21 _Processor_mask_To_uint32_t() . . . 695
8.113.2.22 _Processor_mask_Xor() . . . 695
8.113.2.23 _Processor_mask_Zero() . . . 696
8.114 Profiling Support . . . 697
8.114.1 Detailed Description . . . 697
8.114.2 Function Documentation . . . 697
8.114.2.1 _Profiling_Outer_most_interrupt_entry_and_exit() . . . 697
8.114.2.2 _Profiling_Thread_dispatch_disable() . . . 698
8.114.2.3 _Profiling_Thread_dispatch_disable_critical() . . . 698
8.114.2.4 _Profiling_Thread_dispatch_enable() . . . 698
8.114.2.5 _Profiling_Update_max_interrupt_delay() . . . 699
8.115 Protected Heap Handler . . . 700
8.115.1 Detailed Description . . . 700
8.115.2 Function Documentation . . . 701
8.115.2.1 _Protected_heap_Allocate() . . . 701
8.115.2.2 _Protected_heap_Allocate_aligned() . . . 701
8.115.2.3 _Protected_heap_Allocate_aligned_with_boundary() . . . 702
8.115.2.4 _Protected_heap_Extend() . . . 702
8.115.2.5 _Protected_heap_Free() . . . 703
8.115.2.6 _Protected_heap_Get_block_size() . . . 703
8.115.2.7 _Protected_heap_Get_free_information() . . . 704
8.115.2.8 _Protected_heap_Get_information() . . . 704
8.115.2.9 _Protected_heap_Get_size() . . . 705
8.115.2.10 _Protected_heap_Initialize() . . . 705
8.115.2.11 _Protected_heap_Iterate() . . . 705
8.115.2.12 _Protected_heap_Resize_block() . . . 706
8.115.2.13 _Protected_heap_Walk() . . . 706
8.116 RTEMS Allocator Mutex . . . 708
8.116.1 Detailed Description . . . 708
8.117 RTEMS Copyright Notice . . . 709
8.117.1 Detailed Description . . . 709
8.117.2 Variable Documentation . . . 709

8.117.2.1 _Copyright_Notice . . . 709
8.117.2.2 _RTEMS_version . . . 709
8.118 RTEMS Per CPU Information . . . 710
8.118.1 Detailed Description . . . 711
8.118.2 Typedef Documentation . . . 711
8.118.2.1 Per_CPU_Control . . . 712
8.118.2.2 Per_CPU_Job . . . 712
8.118.3 Enumeration Type Documentation . . . 712
8.118.3.1 Per_CPU_State . . . 712
8.118.3.2 Per_CPU_Watchdog_index . . . 714
8.118.4 Function Documentation . . . 714
8.118.4.1 _Per_CPU_Add_job() . . . 714
8.118.4.2 _Per_CPU_Initialize() . . . 715
8.118.4.3 _Per_CPU_Perform_jobs() . . . 715
8.118.4.4 _Per_CPU_State_wait_for_non_initial_state() . . . 715
8.118.4.5 _Per_CPU_Wait_for_job() . . . 716
8.118.4.6 _Thread_Get_executing() . . . 716
8.118.5 Variable Documentation . . . 716
8.118.5.1 CPU_STRUCTURE_ALIGNMENT . . . 716
8.119 RTEMS Print Support . . . 717
8.119.1 Detailed Description . . . 717
8.119.2 Typedef Documentation . . . 717
8.119.2.1 rtems_print_printer . . . 718
8.119.3 Function Documentation . . . 718
8.119.3.1 rtems_print_printer_empty() . . . 718
8.119.3.2 rtems_print_printer_fprintf() . . . 718
8.119.3.3 rtems_print_printer_fprintf_putc() . . . 719
8.119.3.4 rtems_print_printer_printf() . . . 719
8.119.3.5 rtems_print_printer_printk() . . . 719
8.119.3.6 rtems_print_printer_task() . . . 719
8.119.3.7 rtems_print_printer_valid() . . . 720
8.119.3.8 rtems_printer_task_drain() . . . 720
8.120 RTEMS Status Code Checks . . . 722
8.120.1 Detailed Description . . . 722
8.121 RTEMS Termios Device Support . . . 723
8.122 RTEMS Test Framework . . . 724
8.122.1 Detailed Description . . . 725
8.123 RTEMS Test Framework Implementation . . . 726
8.123.1 Detailed Description . . . 729
8.123.2 Macro Definition Documentation . . . 729
8.123.2.1 T_flags_eno . . . 729
8.123.2.2 T_flags_eno_success . . . 729

8.123.2.3 T_flags_eq . . . 730
8.123.2.4 T_flags_eq_char . . . 730
8.123.2.5 T_flags_eq_int . . . 730
8.123.2.6 T_flags_eq_ll . . . 731
8.123.2.7 T_flags_eq_long . . . 731
8.123.2.8 T_flags_eq_mem . . . 731
8.123.2.9 T_flags_eq_nstr . . . 732
8.123.2.10 T_flags_eq_ptr . . . 732
8.123.2.11 T_flags_eq_str . . . 732
8.123.2.12 T_flags_eq_uint . . . 733
8.123.2.13 T_flags_eq_ull . . . 733
8.123.2.14 T_flags_eq_ulong . . . 733
8.123.2.15 T_flags_ge_int . . . 734
8.123.2.16 T_flags_ge_ll . . . 734
8.123.2.17 T_flags_ge_long . . . 734
8.123.2.18 T_flags_ge_uint . . . 735
8.123.2.19 T_flags_ge_ull . . . 735
8.123.2.20 T_flags_ge_ulong . . . 735
8.123.2.21 T_flags_gt_int . . . 736
8.123.2.22 T_flags_gt_ll . . . 736
8.123.2.23 T_flags_gt_long . . . 736
8.123.2.24 T_flags_gt_uint . . . 737
8.123.2.25 T_flags_gt_ull . . . 737
8.123.2.26 T_flags_gt_ulong . . . 737
8.123.2.27 T_flags_le_int . . . 738
8.123.2.28 T_flags_le_ll . . . 738
8.123.2.29 T_flags_le_long . . . 738
8.123.2.30 T_flags_le_uint . . . 739
8.123.2.31 T_flags_le_ull . . . 739
8.123.2.32 T_flags_le_ulong . . . 739
8.123.2.33 T_flags_lt_int . . . 740
8.123.2.34 T_flags_lt_ll . . . 740
8.123.2.35 T_flags_lt_long . . . 740
8.123.2.36 T_flags_lt_uint . . . 741
8.123.2.37 T_flags_lt_ull . . . 741
8.123.2.38 T_flags_lt_ulong . . . 741
8.123.2.39 T_flags_ne . . . 742
8.123.2.40 T_flags_ne_char . . . 742
8.123.2.41 T_flags_ne_int . . . 742
8.123.2.42 T_flags_ne_ll . . . 743
8.123.2.43 T_flags_ne_long . . . 743
8.123.2.44 T_flags_ne_mem . . . 743

8.123.2.45 T_flags_ne_nstr . . . 744
8.123.2.46 T_flags_ne_ptr . . . 744
8.123.2.47 T_flags_ne_str . . . 744
8.123.2.48 T_flags_ne_uint . . . 745
8.123.2.49 T_flags_ne_ull . . . 745
8.123.2.50 T_flags_ne_ulong . . . 745
8.123.2.51 T_flags_not_null . . . 746
8.123.2.52 T_flags_null . . . 746
8.123.2.53 T_flags_psx_error . . . 746
8.123.2.54 T_flags_psx_success . . . 747
8.123.2.55 T_flags_true . . . 747
8.123.2.56 T_VA_ARGS_KIND . . . 747
8.124 Rate-Monotonic Manager . . . 748
8.124.1 Detailed Description . . . 749
8.124.2 Enumeration Type Documentation . . . 749
8.124.2.1 rtems_rate_monotonic_period_states . . . 749
8.124.3 Function Documentation . . . 749
8.124.3.1 rtems_rate_monotonic_cancel() . . . 749
8.124.3.2 rtems_rate_monotonic_create() . . . 749
8.124.3.3 rtems_rate_monotonic_delete() . . . 750
8.124.3.4 rtems_rate_monotonic_get_statistics() . . . 750
8.124.3.5 rtems_rate_monotonic_get_status() . . . 750
8.124.3.6 rtems_rate_monotonic_ident() . . . 751
8.124.3.7 rtems_rate_monotonic_period() . . . 751
8.124.3.8 rtems_rate_monotonic_report_statistics_with_plugin() . . . 752
8.124.3.9 rtems_rate_monotonic_reset_statistics() . . . 752
8.125 Red-Black Tree Handler . . . 753
8.125.1 Detailed Description . . . 755
8.125.2 Typedef Documentation . . . 755
8.125.2.1 RBTree_Node . . . 755
8.125.2.2 RBTree_Visitor . . . 755
8.125.3 Function Documentation . . . 756
8.125.3.1 _RBTree_Add_child() . . . 756
8.125.3.2 _RBTree_Extract() . . . 756
8.125.3.3 _RBTree_Find_inline() . . . 756
8.125.3.4 _RBTree_Initialize_empty() . . . 757
8.125.3.5 _RBTree_Initialize_node() . . . 757
8.125.3.6 _RBTree_Initialize_one() . . . 758
8.125.3.7 _RBTree_Insert_color() . . . 758
8.125.3.8 _RBTree_Insert_inline() . . . 758
8.125.3.9 _RBTree_Insert_with_parent() . . . 759
8.125.3.10 _RBTree_Is_empty() . . . 760

8.125.3.11 _RBTree_Is_node_off_tree() . . . 760
8.125.3.12 _RBTree_Is_root() . . . 761
8.125.3.13 _RBTree_Iterate() . . . 761
8.125.3.14 _RBTree_Left() . . . 762
8.125.3.15 _RBTree_Left_reference() . . . 762
8.125.3.16 _RBTree_Maximum() . . . 762
8.125.3.17 _RBTree_Minimum() . . . 763
8.125.3.18 _RBTree_Parent() . . . 763
8.125.3.19 _RBTree_Postorder_first() . . . 764
8.125.3.20 _RBTree_Postorder_next() . . . 765
8.125.3.21 _RBTree_Predecessor() . . . 765
8.125.3.22 _RBTree_Replace_node() . . . 765
8.125.3.23 _RBTree_Right() . . . 766
8.125.3.24 _RBTree_Right_reference() . . . 766
8.125.3.25 _RBTree_Root() . . . 767
8.125.3.26 _RBTree_Root_const_reference() . . . 767
8.125.3.27 _RBTree_Root_reference() . . . 768
8.125.3.28 _RBTree_Set_off_tree() . . . 768
8.125.3.29 _RBTree_Successor() . . . 768
8.125.3.30 RB_HEAD() . . . 769
8.126 Region Manager . . . 770
8.126.1 Detailed Description . . . 770
8.126.2 Function Documentation . . . 770
8.126.2.1 rtems_region_create() . . . 770
8.126.2.2 rtems_region_delete() . . . 771
8.126.2.3 rtems_region_extend() . . . 771
8.126.2.4 rtems_region_get_free_information() . . . 771
8.126.2.5 rtems_region_get_information() . . . 772
8.126.2.6 rtems_region_get_segment() . . . 772
8.126.2.7 rtems_region_get_segment_size() . . . 772
8.126.2.8 rtems_region_ident() . . . 774
8.126.2.9 rtems_region_resize_segment() . . . 774
8.126.2.10 rtems_region_return_segment() . . . 775
8.127 Runtime Measurements . . . 776
8.127.1 Detailed Description . . . 776
8.128 SMP Barriers . . . 777
8.128.1 Detailed Description . . . 777
8.128.2 Function Documentation . . . 777
8.128.2.1 _SMP_barrier_Control_initialize() . . . 777
8.128.2.2 _SMP_barrier_State_initialize() . . . 778
8.128.2.3 _SMP_barrier_Wait() . . . 778
8.129 SMP Locks . . . 779

8.129.1 Detailed Description . . . 781
8.129.2 Macro Definition Documentation . . . 781
8.129.2.1 _SMP_lock_Destroy . . . 781
8.129.2.2 _SMP_lock_Initialize . . . 782
8.129.2.3 _SMP_lock_Release . . . 782
8.129.2.4 _SMP_lock_Release_and_ISR_enable . . . 782
8.129.2.5 _SMP_MCS_lock_Acquire . . . 783
8.129.2.6 _SMP_ticket_lock_Acquire . . . 783
8.129.2.7 _SMP_ticket_lock_Release . . . 783
8.129.2.8 SMP_TICKET_LOCK_INITIALIZER . . . 784
8.129.3 Function Documentation . . . 784
8.129.3.1 _SMP_lock_Acquire() . . . 784
8.129.3.2 _SMP_lock_Acquire_inline() . . . 785
8.129.3.3 _SMP_lock_Destroy_inline() . . . 785
8.129.3.4 _SMP_lock_Initialize_inline() . . . 785
8.129.3.5 _SMP_lock_ISR_disable_and_acquire() . . . 786
8.129.3.6 _SMP_lock_ISR_disable_and_acquire_inline() . . . 786
8.129.3.7 _SMP_lock_Release_and_ISR_enable_inline() . . . 786
8.129.3.8 _SMP_lock_Release_inline() . . . 787
8.129.3.9 _SMP_lock_Set_name() . . . 787
8.129.3.10 _SMP_MCS_lock_Destroy() . . . 787
8.129.3.11 _SMP_MCS_lock_Do_acquire() . . . 788
8.129.3.12 _SMP_MCS_lock_Initialize() . . . 788
8.129.3.13 _SMP_MCS_lock_Release() . . . 788
8.129.3.14 _SMP_sequence_lock_Destroy() . . . 789
8.129.3.15 _SMP_sequence_lock_Initialize() . . . 789
8.129.3.16 _SMP_sequence_lock_Read_begin() . . . 789
8.129.3.17 _SMP_sequence_lock_Read_retry() . . . 790
8.129.3.18 _SMP_sequence_lock_Write_begin() . . . 790
8.129.3.19 _SMP_sequence_lock_Write_end() . . . 791
8.129.3.20 _SMP_ticket_lock_Destroy() . . . 791
8.129.3.21 _SMP_ticket_lock_Do_acquire() . . . 791
8.129.3.22 _SMP_ticket_lock_Do_release() . . . 792
8.129.3.23 _SMP_ticket_lock_Initialize() . . . 792
8.130 SMP Scheduler . . . 793
8.130.1 Detailed Description . . . 796
8.130.2 Enumeration Type Documentation . . . 799
8.130.2.1 Scheduler_SMP_Node_state . . . 799
8.130.3 Function Documentation . . . 800
8.130.3.1 _Scheduler_SMP_Add_processor() . . . 800
8.130.3.2 _Scheduler_SMP_Allocate_processor() . . . 800
8.130.3.3 _Scheduler_SMP_Allocate_processor_exact() . . . 801

8.130.3.4 _Scheduler_SMP_Allocate_processor_lazy() . . . 801
8.130.3.5 _Scheduler_SMP_Ask_for_help() . . . 802
8.130.3.6 _Scheduler_SMP_Block() . . . 803
8.130.3.7 _Scheduler_SMP_Do_nothing_register_idle() . . . 803
8.130.3.8 _Scheduler_SMP_Do_start_idle() . . . 804
8.130.3.9 _Scheduler_SMP_Enqueue() . . . 804
8.130.3.10 _Scheduler_SMP_Enqueue_scheduled() . . . 805
8.130.3.11 _Scheduler_SMP_Enqueue_to_scheduled() . . . 805
8.130.3.12 _Scheduler_SMP_Exctract_idle_thread() . . . 806
8.130.3.13 _Scheduler_SMP_Extract_from_scheduled() . . . 806
8.130.3.14 _Scheduler_SMP_Get_idle_thread() . . . 807
8.130.3.15 _Scheduler_SMP_Get_lowest_scheduled() . . . 807
8.130.3.16 _Scheduler_SMP_Get_self() . . . 807
8.130.3.17 _Scheduler_SMP_Initialize() . . . 808
8.130.3.18 _Scheduler_SMP_Insert_scheduled() . . . 808
8.130.3.19 _Scheduler_SMP_Is_processor_owned_by_us() . . . 808
8.130.3.20 _Scheduler_SMP_Node_change_state() . . . 809
8.130.3.21 _Scheduler_SMP_Node_downcast() . . . 809
8.130.3.22 _Scheduler_SMP_Node_initialize() . . . 810
8.130.3.23 _Scheduler_SMP_Node_priority() . . . 810
8.130.3.24 _Scheduler_SMP_Node_state() . . . 810
8.130.3.25 _Scheduler_SMP_Node_update_priority() . . . 811
8.130.3.26 _Scheduler_SMP_Preempt() . . . 811
8.130.3.27 _Scheduler_SMP_Preempt_and_schedule_highest_ready() . . . 812
8.130.3.28 _Scheduler_SMP_Priority_less_equal() . . . 812
8.130.3.29 _Scheduler_SMP_Reconsider_help_request() . . . 813
8.130.3.30 _Scheduler_SMP_Release_idle_thread() . . . 813
8.130.3.31 _Scheduler_SMP_Remove_processor() . . . 813
8.130.3.32 _Scheduler_SMP_Schedule_highest_ready() . . . 814
8.130.3.33 _Scheduler_SMP_Set_affinity() . . . 814
8.130.3.34 _Scheduler_SMP_Start_idle() . . . 815
8.130.3.35 _Scheduler_SMP_Thread_get_node() . . . 815
8.130.3.36 _Scheduler_SMP_Thread_get_own_node() . . . 816
8.130.3.37 _Scheduler_SMP_Unblock() . . . 816
8.130.3.38 _Scheduler_SMP_Update_priority() . . . 817
8.130.3.39 _Scheduler_SMP_Withdraw_node() . . . 817
8.130.3.40 _Scheduler_SMP_Yield() . . . 818
8.131 SMP Support . . . 819
8.131.1 Detailed Description . . . 820
8.131.2 Macro Definition Documentation . . . 820
8.131.2.1 SMP_MESSAGE_PERFORM_JOBS . . . 821
8.131.2.2 SMP_MESSAGE_SHUTDOWN . . . 821

8.131.3 Function Documentation . . . 821
8.131.3.1 _SMP_Broadcast_action() . . . 821
8.131.3.2 _SMP_Fatal() . . . 822
8.131.3.3 _SMP_Get_online_processors() . . . 822
8.131.3.4 _SMP_Handler_initialize() . . . 822
8.131.3.5 _SMP_Inter_processor_interrupt_handler() . . . 822
8.131.3.6 _SMP_Multicast_action() . . . 823
8.131.3.7 _SMP_Need_inter_processor_interrupts() . . . 823
8.131.3.8 _SMP_Othercast_action() . . . 824
8.131.3.9 _SMP_Request_shutdown() . . . 824
8.131.3.10 _SMP_Send_message() . . . 824
8.131.3.11 _SMP_Send_message_broadcast() . . . 825
8.131.3.12 _SMP_Send_message_multicast() . . . 825
8.131.3.13 _SMP_Should_start_processor() . . . 825
8.131.3.14 _SMP_Start_multitasking_on_secondary_processor() . . . 826
8.131.3.15 _SMP_Synchronize() . . . 826
8.131.3.16 _SMP_Unicast_action() . . . 827
8.131.4 Variable Documentation . . . 827
8.131.4.1 _SMP_Online_processors . . . 827
8.131.4.2 _SMP_Processor_configured_maximum . . . 827
8.132 SPARC . . . 828
8.132.1 Detailed Description . . . 829
8.132.2 Macro Definition Documentation . . . 829
8.132.2.1 CPU_INTERRUPT_FRAME_SIZE . . . 829
8.132.2.2 ISF_G1_OFFSET . . . 829
8.132.2.3 ISF_G2_OFFSET . . . 829
8.132.2.4 ISF_G3_OFFSET . . . 830
8.132.2.5 ISF_G4_OFFSET . . . 830
8.132.2.6 ISF_G5_OFFSET . . . 830
8.132.2.7 ISF_G7_OFFSET . . . 830
8.132.2.8 ISF_I0_OFFSET . . . 830
8.132.2.9 ISF_I1_OFFSET . . . 831
8.132.2.10 ISF_I2_OFFSET . . . 831
8.132.2.11 ISF_I3_OFFSET . . . 831
8.132.2.12 ISF_I4_OFFSET . . . 831
8.132.2.13 ISF_I5_OFFSET . . . 831
8.132.2.14 ISF_I6_FP_OFFSET . . . 832
8.132.2.15 ISF_I7_OFFSET . . . 832
8.132.2.16 ISF_NPC_OFFSET . . . 832
8.132.2.17 ISF_PC_OFFSET . . . 832
8.132.2.18 ISF_PSR_OFFSET . . . 832
8.132.2.19 ISF_TPC_OFFSET . . . 833

8.132.2.20 ISF_Y_OFFSET . . . 833
8.132.2.21 SPARC_MINIMUM_STACK_FRAME_SIZE . . . 833
8.133 SPARC . . . 834
8.133.1 Detailed Description . . . 834
8.134 SPARC Assembler Support . . . 835
8.134.1 Detailed Description . . . 835
8.134.2 Macro Definition Documentation . . . 835
8.134.2.1 RTRAP . . . 835
8.134.2.2 TRAP . . . 836
8.135 SPARC Context Structures . . . 837
8.135.1 Detailed Description . . . 838
8.135.2 Macro Definition Documentation . . . 838
8.135.2.1 _CPU_Context_Get_SP . . . 838
8.135.2.2 CONTEXT_CONTROL_FP_SIZE . . . 839
8.135.2.3 F12_F13_OFFSET . . . 839
8.135.2.4 F14_F15_OFFSET . . . 839
8.135.2.5 F16_F17_OFFSET . . . 839
8.135.2.6 F18_F19_OFFSET . . . 839
8.135.2.7 F1O_F11_OFFSET . . . 840
8.135.2.8 F22_F23_OFFSET . . . 840
8.135.2.9 F24_F25_OFFSET . . . 840
8.135.2.10 F26_F27_OFFSET . . . 840
8.135.2.11 F28_F29_OFFSET . . . 840
8.135.2.12 F2_F3_OFFSET . . . 841
8.135.2.13 F2O_F21_OFFSET . . . 841
8.135.2.14 F3O_F31_OFFSET . . . 841
8.135.2.15 F4_F5_OFFSET . . . 841
8.135.2.16 F6_F7_OFFSET . . . 841
8.135.2.17 F8_F9_OFFSET . . . 842
8.135.2.18 FO_F1_OFFSET . . . 842
8.135.2.19 FSR_OFFSET . . . 842
8.135.2.20 G5_OFFSET . . . 842
8.135.2.21 G7_OFFSET . . . 842
8.135.2.22 I0_OFFSET . . . 843
8.135.2.23 I1_OFFSET . . . 843
8.135.2.24 I2_OFFSET . . . 843
8.135.2.25 I3_OFFSET . . . 843
8.135.2.26 I4_OFFSET . . . 843
8.135.2.27 I5_OFFSET . . . 844
8.135.2.28 I6_FP_OFFSET . . . 844
8.135.2.29 I7_OFFSET . . . 844
8.135.2.30 ISR_DISPATCH_DISABLE_STACK_OFFSET . . . 844

8.135.2.31 L0_OFFSET . . . 844
8.135.2.32 L1_OFFSET . . . 845
8.135.2.33 L2_OFFSET . . . 845
8.135.2.34 L3_OFFSET . . . 845
8.135.2.35 L4_OFFSET . . . 845
8.135.2.36 L5_OFFSET . . . 845
8.135.2.37 L6_OFFSET . . . 846
8.135.2.38 L7_OFFSET . . . 846
8.135.2.39 O6_SP_OFFSET . . . 846
8.135.2.40 O7_OFFSET . . . 846
8.135.2.41 PSR_OFFSET . . . 846
8.136 Scheduler Handler . . . 847
8.136.1 Detailed Description . . . 853
8.136.2 Macro Definition Documentation . . . 853
8.136.2.1 SCHEDULER_OPERATION_DEFAULT_ASK_FOR_HELP . . . 853
8.136.3 Typedef Documentation . . . 853
8.136.3.1 Scheduler_Context . . . 853
8.136.3.2 Scheduler_Get_idle_thread . . . 853
8.136.3.3 Scheduler_Release_idle_thread . . . 854
8.136.4 Enumeration Type Documentation . . . 854
8.136.4.1 Scheduler_Node_request . . . 854
8.136.5 Function Documentation . . . 855
8.136.5.1 _Scheduler_Acquire_critical() . . . 855
8.136.5.2 _Scheduler_Ask_for_help() . . . 855
8.136.5.3 _Scheduler_Block() . . . 855
8.136.5.4 _Scheduler_Block_node() . . . 856
8.136.5.5 _Scheduler_Build_id() . . . 856
8.136.5.6 _Scheduler_Cancel_job() . . . 857
8.136.5.7 _Scheduler_default_Ask_for_help() . . . 857
8.136.5.8 _Scheduler_default_Cancel_job() . . . 858
8.136.5.9 _Scheduler_default_Map_priority() . . . 858
8.136.5.10 _Scheduler_default_Node_destroy() . . . 859
8.136.5.11 _Scheduler_default_Node_initialize() . . . 859
8.136.5.12 _Scheduler_default_Pin_or_unpin() . . . 859
8.136.5.13 _Scheduler_default_Reconsider_help_request() . . . 860
8.136.5.14 _Scheduler_default_Release_job() . . . 860
8.136.5.15 _Scheduler_default_Schedule() . . . 861
8.136.5.16 _Scheduler_default_Set_affinity() . . . 861
8.136.5.17 _Scheduler_default_Set_affinity_body() . . . 862
8.136.5.18 _Scheduler_default_Start_idle() . . . 862
8.136.5.19 _Scheduler_default_Tick() . . . 862
8.136.5.20 _Scheduler_default_Unmap_priority() . . . 863

8.136.5.21 _Scheduler_default_Withdraw_node() . . . 863
8.136.5.22 _Scheduler_Discard_idle_thread() . . . 864
8.136.5.23 _Scheduler_Exchange_idle_thread() . . . 864
8.136.5.24 _Scheduler_Generic_block() . . . 864
8.136.5.25 _Scheduler_Get_affinity() . . . 865
8.136.5.26 _Scheduler_Get_by_CPU() . . . 865
8.136.5.27 _Scheduler_Get_by_id() . . . 866
8.136.5.28 _Scheduler_Get_context() . . . 866
8.136.5.29 _Scheduler_Get_index() . . . 866
8.136.5.30 _Scheduler_Get_index_by_id() . . . 867
8.136.5.31 _Scheduler_Get_processor_count() . . . 867
8.136.5.32 _Scheduler_Get_processors() . . . 868
8.136.5.33 _Scheduler_Handler_initialization() . . . 868
8.136.5.34 _Scheduler_Has_processor_ownership() . . . 868
8.136.5.35 _Scheduler_Is_non_preempt_mode_supported() . . . 869
8.136.5.36 _Scheduler_Map_priority() . . . 869
8.136.5.37 _Scheduler_Node_destroy() . . . 870
8.136.5.38 _Scheduler_Node_do_initialize() . . . 870
8.136.5.39 _Scheduler_Node_get_idle() . . . 870
8.136.5.40 _Scheduler_Node_get_owner() . . . 871
8.136.5.41 _Scheduler_Node_get_priority() . . . 871
8.136.5.42 _Scheduler_Node_get_scheduler() . . . 871
8.136.5.43 _Scheduler_Node_get_user() . . . 872
8.136.5.44 _Scheduler_Node_initialize() . . . 872
8.136.5.45 _Scheduler_Node_set_priority() . . . 873
8.136.5.46 _Scheduler_Node_set_user() . . . 873
8.136.5.47 _Scheduler_Priority_and_sticky_update() . . . 873
8.136.5.48 _Scheduler_Release_critical() . . . 874
8.136.5.49 _Scheduler_Release_idle_thread() . . . 874
8.136.5.50 _Scheduler_Release_job() . . . 875
8.136.5.51 _Scheduler_Schedule() . . . 875
8.136.5.52 _Scheduler_Set() . . . 875
8.136.5.53 _Scheduler_Set_affinity() . . . 876
8.136.5.54 _Scheduler_Set_idle_thread() . . . 876
8.136.5.55 _Scheduler_Start_idle() . . . 878
8.136.5.56 _Scheduler_Thread_change_state() . . . 878
8.136.5.57 _Scheduler_Tick() . . . 879
8.136.5.58 _Scheduler_Try_to_schedule_node() . . . 879
8.136.5.59 _Scheduler_Unblock() . . . 880
8.136.5.60 _Scheduler_Unblock_node() . . . 880
8.136.5.61 _Scheduler_Unmap_priority() . . . 881
8.136.5.62 _Scheduler_Update_heir() . . . 881

8.136.5.63 _Scheduler_Update_priority() . . . 882
8.136.5.64 _Scheduler_Use_idle_thread() . . . 882
8.136.5.65 _Scheduler_Yield() . . . 883
8.136.6 Variable Documentation . . . 883
8.136.6.1 _Scheduler_Count . . . 883
8.136.6.2 _Scheduler_Initial_assignments . . . 883
8.136.6.3 _Scheduler_Node_size . . . 884
8.136.6.4 _Scheduler_Table . . . 884
8.137 Score Timestamp . . . 885
8.137.1 Detailed Description . . . 886
8.137.2 Typedef Documentation . . . 886
8.137.2.1 Timestamp_Control . . . 886
8.137.3 Function Documentation . . . 886
8.137.3.1 _Timestamp_Add_to() . . . 886
8.137.3.2 _Timestamp_Divide() . . . 887
8.137.3.3 _Timestamp_Equal_to() . . . 887
8.137.3.4 _Timestamp_Get_as_nanoseconds() . . . 887
8.137.3.5 _Timestamp_Get_nanoseconds() . . . 888
8.137.3.6 _Timestamp_Get_seconds() . . . 888
8.137.3.7 _Timestamp_Greater_than() . . . 889
8.137.3.8 _Timestamp_Less_than() . . . 889
8.137.3.9 _Timestamp_Set() . . . 890
8.137.3.10 _Timestamp_Set_to_zero() . . . 890
8.137.3.11 _Timestamp_Subtract() . . . 890
8.137.3.12 _Timestamp_To_timespec() . . . 891
8.137.3.13 _Timestamp_To_timeval() . . . 891
8.138 Semaphore Handler . . . 892
8.138.1 Detailed Description . . . 893
8.138.2 Function Documentation . . . 893
8.138.2.1 _CORE_semaphore_Acquire_critical() . . . 893
8.138.2.2 _CORE_semaphore_Destroy() . . . 893
8.138.2.3 _CORE_semaphore_Get_count() . . . 894
8.138.2.4 _CORE_semaphore_Initialize() . . . 894
8.138.2.5 _CORE_semaphore_Release() . . . 895
8.138.2.6 _CORE_semaphore_Seize() . . . 895
8.138.2.7 _CORE_semaphore_Surrender() . . . 896
8.138.2.8 _Sem_Get() . . . 896
8.138.2.9 _Sem_Queue_acquire_critical() . . . 897
8.138.2.10 _Sem_Queue_release() . . . 897
8.139 Semaphore Manager . . . 898
8.139.1 Detailed Description . . . 898
8.139.2 Function Documentation . . . 898

8.139.2.1 rtems_semaphore_create() . . . 898
8.139.2.2 rtems_semaphore_delete() . . . 900
8.139.2.3 rtems_semaphore_flush() . . . 901
8.139.2.4 rtems_semaphore_ident() . . . 901
8.139.2.5 rtems_semaphore_obtain() . . . 902
8.139.2.6 rtems_semaphore_release() . . . 902
8.139.2.7 rtems_semaphore_set_priority() . . . 903
8.140 Semaphore Manager Implementation . . . 904
8.140.1 Detailed Description . . . 905
8.140.2 Macro Definition Documentation . . . 905
8.140.2.1 SEMAPHORE_INFORMATION_DEFINE . . . 905
8.140.3 Enumeration Type Documentation . . . 905
8.140.3.1 Semaphore_Variant . . . 906
8.140.4 Function Documentation . . . 906
8.140.4.1 _Semaphore_Allocate() . . . 906
8.140.4.2 _Semaphore_Free() . . . 906
8.141 Serial Mouse . . . 907
8.142 Shared . . . 908
8.142.1 Detailed Description . . . 908
8.143 Shared . . . 909
8.143.1 Detailed Description . . . 909
8.144 Signal Manager . . . 910
8.144.1 Detailed Description . . . 911
8.144.2 Function Documentation . . . 911
8.144.2.1 rtems_signal_catch() . . . 911
8.144.2.2 rtems_signal_send() . . . 912
8.145 Signals Implementation . . . 913
8.145.1 Detailed Description . . . 913
8.146 Signed 16-Bit Integer Checks . . . 914
8.146.1 Detailed Description . . . 914
8.147 Signed 32-Bit Integer Checks . . . 915
8.147.1 Detailed Description . . . 915
8.148 Signed 64-Bit Integer Checks . . . 916
8.148.1 Detailed Description . . . 916
8.149 Signed 8-Bit Integer Checks . . . 917
8.149.1 Detailed Description . . . 917
8.150 Signed Character Checks . . . 918
8.150.1 Detailed Description . . . 918
8.151 Signed Integer Checks . . . 919
8.151.1 Detailed Description . . . 919
8.152 Signed Long Integer Checks . . . 920
8.152.1 Detailed Description . . . 921

8.153 Signed Pointer Value Checks . . . 922
8.153.1 Detailed Description . . . 922
8.154 Signed Short Integer Checks . . . 923
8.154.1 Detailed Description . . . 923
8.155 Signed Size Checks . . . 924
8.155.1 Detailed Description . . . 924
8.156 Simple Priority Scheduler . . . 925
8.156.1 Detailed Description . . . 926
8.156.2 Macro Definition Documentation . . . 926
8.156.2.1 SCHEDULER_SIMPLE_ENTRY_POINTS . . . 926
8.156.3 Function Documentation . . . 926
8.156.3.1 _Scheduler_simple_Block() . . . 926
8.156.3.2 _Scheduler_simple_Extract() . . . 927
8.156.3.3 _Scheduler_simple_Get_context() . . . 927
8.156.3.4 _Scheduler_simple_Initialize() . . . 927
8.156.3.5 _Scheduler_simple_Insert() . . . 928
8.156.3.6 _Scheduler_simple_Priority_less_equal() . . . 928
8.156.3.7 _Scheduler_simple_Schedule() . . . 929
8.156.3.8 _Scheduler_simple_Schedule_body() . . . 929
8.156.3.9 _Scheduler_simple_Unblock() . . . 929
8.156.3.10 _Scheduler_simple_Update_priority() . . . 930
8.156.3.11 _Scheduler_simple_Yield() . . . 930
8.157 Size Checks . . . 931
8.157.1 Detailed Description . . . 931
8.158 Stack Handler . . . 932
8.158.1 Detailed Description . . . 933
8.158.2 Macro Definition Documentation . . . 933
8.158.2.1 STACK_MINIMUM_SIZE . . . 933
8.158.3 Typedef Documentation . . . 933
8.158.3.1 Stack_Allocator_allocate . . . 933
8.158.3.2 Stack_Allocator_free . . . 934
8.158.3.3 Stack_Allocator_initialize . . . 934
8.158.4 Function Documentation . . . 934
8.158.4.1 _Stack_Allocate() . . . 934
8.158.4.2 _Stack_Allocator_do_initialize() . . . 935
8.158.4.3 _Stack_Ensure_minimum() . . . 935
8.158.4.4 _Stack_Extend_size() . . . 935
8.158.4.5 _Stack_Free() . . . 936
8.158.4.6 _Stack_Free_nothing() . . . 936
8.158.4.7 _Stack_Initialize() . . . 936
8.158.4.8 _Stack_Is_enough() . . . 937
8.158.4.9 _Stack_Minimum() . . . 937

8.158.5 Variable Documentation . . . 938
8.158.5.1 _Stack_Allocator_allocate . . . 938
8.158.5.2 _Stack_Allocator_avoids_workspace . . . 938
8.158.5.3 _Stack_Allocator_free . . . 938
8.158.5.4 _Stack_Allocator_initialize . . . 938
8.158.5.5 _Stack_Space_size . . . 938
8.158.5.6 rtems_minimum_stack_size . . . 938
8.159 Standard C Library Support . . . 939
8.159.1 Detailed Description . . . 940
8.159.2 Macro Definition Documentation . . . 940
8.159.2.1 RTEMS_NEWLIB_EXTENSION . . . 940
8.159.3 Function Documentation . . . 940
8.159.3.1 malloc_free_space() . . . 941
8.159.3.2 malloc_get_heap_pointer() . . . 941
8.159.3.3 malloc_info() . . . 941
8.159.3.4 malloc_set_heap_pointer() . . . 941
8.159.3.5 rtems_resource_snapshot_check() . . . 941
8.159.3.6 rtems_resource_snapshot_equal() . . . 942
8.159.3.7 rtems_resource_snapshot_take() . . . 942
8.160 String Checks . . . 944
8.160.1 Detailed Description . . . 944
8.161 SuperCore . . . 945
8.161.1 Detailed Description . . . 947
8.162 Support Services . . . 948
8.162.1 Detailed Description . . . 948
8.162.2 Macro Definition Documentation . . . 948
8.162.2.1 RTEMS_MICROSECONDS_TO_TICKS . . . 948
8.162.2.2 RTEMS_MILLISECONDS_TO_MICROSECONDS . . . 949
8.162.2.3 RTEMS_MILLISECONDS_TO_TICKS . . . 949
8.162.3 Function Documentation . . . 950
8.162.3.1 rtems_is_name_valid() . . . 950
8.162.3.2 rtems_name_to_characters() . . . 950
8.162.3.3 rtems_workspace_allocate() . . . 951
8.162.3.4 rtems_workspace_free() . . . 951
8.162.3.5 rtems_workspace_get_information() . . . 951
8.162.3.6 rtems_workspace_greedy_allocate() . . . 951
8.162.3.7 rtems_workspace_greedy_allocate_all_except_largest() . . . 952
8.162.3.8 rtems_workspace_greedy_free() . . . 952
8.163 System State Handler . . . 953
8.163.1 Detailed Description . . . 953
8.163.2 Enumeration Type Documentation . . . 953
8.163.2.1 System_state_Codes . . . 953

8.163.3 Function Documentation . . . 954
8.163.3.1 _System_state_Get() . . . 954
8.163.3.2 _System_state_Is_before_initialization() . . . 954
8.163.3.3 _System_state_Is_before_multitasking() . . . 955
8.163.3.4 _System_state_Is_terminated() . . . 955
8.163.3.5 _System_state_Is_up() . . . 955
8.163.3.6 _System_state_Set() . . . 956
8.164 Task Manager . . . 957
8.164.1 Detailed Description . . . 959
8.164.2 Task Definition . . . 959
8.164.3 Task Control Block . . . 960
8.164.4 Task States . . . 960
8.164.5 Task Priority . . . 960
8.164.6 Task Mode . . . 961
8.164.7 Accessing Task Arguments . . . 962
8.164.8 Floating Point Considerations . . . 962
8.164.9 Per Task Variables . . . 962
8.164.10 Building a Task Attribute Set . . . 963
8.164.11 Building a Mode and Mask . . . 963
8.164.12 Macro Definition Documentation . . . 963
8.164.12.1 rtems_scheduler_get_processor . . . 964
8.164.12.2 rtems_scheduler_get_processor_maximum . . . 964
8.164.12.3 RTEMS_TASK_STORAGE_ALIGNMENT . . . 964
8.164.12.4 RTEMS_TASK_STORAGE_SIZE . . . 964
8.164.13 Typedef Documentation . . . 965
8.164.13.1 rtems_task_argument . . . 965
8.164.14 Function Documentation . . . 965
8.164.14.1 rtems_scheduler_add_processor() . . . 965
8.164.14.2 rtems_scheduler_get_maximum_priority() . . . 966
8.164.14.3 rtems_scheduler_get_processor_set() . . . 966
8.164.14.4 rtems_scheduler_ident() . . . 967
8.164.14.5 rtems_scheduler_ident_by_processor() . . . 967
8.164.14.6 rtems_scheduler_ident_by_processor_set() . . . 968
8.164.14.7 rtems_scheduler_map_priority_from_posix() . . . 969
8.164.14.8 rtems_scheduler_map_priority_to_posix() . . . 969
8.164.14.9 rtems_scheduler_remove_processor() . . . 970
8.164.14.10 rtems_task_construct() . . . 970
8.164.14.11 rtems_task_create() . . . 971
8.164.14.12 rtems_task_delete() . . . 972
8.164.14.13 rtems_task_get_affinity() . . . 973
8.164.14.14 rtems_task_get_priority() . . . 973
8.164.14.15 rtems_task_get_scheduler() . . . 973

8.164.14.16 rtems_task_ident() . . . 974
8.164.14.17 rtems_task_is_suspended() . . . 975
8.164.14.18 rtems_task_iterate() . . . 975
8.164.14.19 rtems_task_mode() . . . 975
8.164.14.20 rtems_task_restart() . . . 976
8.164.14.21 rtems_task_resume() . . . 976
8.164.14.22 rtems_task_set_affinity() . . . 976
8.164.14.23 rtems_task_set_priority() . . . 977
8.164.14.24 rtems_task_set_scheduler() . . . 977
8.164.14.25 rtems_task_start() . . . 978
8.164.14.26 rtems_task_suspend() . . . 978
8.164.14.27 rtems_task_wake_after() . . . 978
8.164.14.28 rtems_task_wake_when() . . . 979
8.165 Task Modes . . . 980
8.165.1 Detailed Description . . . 981
8.165.2 Macro Definition Documentation . . . 981
8.165.2.1 RTEMS_INTERRUPT_LEVEL . . . 981
8.165.3 Function Documentation . . . 981
8.165.3.1 rtems_interrupt_level_body() . . . 981
8.165.4 Variable Documentation . . . 982
8.165.4.1 rtems_interrupt_mask . . . 982
8.166 Task Stack Allocator Configuration . . . 983
8.166.1 Detailed Description . . . 983
8.166.2 Macro Definition Documentation . . . 983
8.166.2.1 CONFIGURE_TASK_STACK_ALLOCATOR . . . 983
8.166.2.2 CONFIGURE_TASK_STACK_ALLOCATOR_AVOIDS_WORK_SPACE . . . 984
8.166.2.3 CONFIGURE_TASK_STACK_ALLOCATOR_INIT . . . 984
8.166.2.4 CONFIGURE_TASK_STACK_DEALLOCATOR . . . 985
8.166.2.5 CONFIGURE_TASK_STACK_FROM_ALLOCATOR . . . 985
8.167 Termios . . . 986
8.167.1 Detailed Description . . . 987
8.167.2 Typedef Documentation . . . 987
8.167.2.1 rtems_termios_isig_handler . . . 987
8.167.3 Enumeration Type Documentation . . . 987
8.167.3.1 rtems_termios_iproc_status_code . . . 987
8.167.4 Function Documentation . . . 988
8.167.4.1 rtems_termios_default_isig_handler() . . . 988
8.167.4.2 rtems_termios_posix_isig_handler() . . . 988
8.167.4.3 rtems_termios_register_isig_handler() . . . 989
8.168 Test Suites . . . 990
8.168.1 Detailed Description . . . 990
8.169 Test Support . . . 991

8.169.1 Detailed Description . . . 992
8.169.2 Typedef Documentation . . . 992
8.169.2.1 rtems_test_parallel_worker_setup . . . 992
8.169.3 Function Documentation . . . 993
8.169.3.1 rtems_test_begin() . . . 993
8.169.3.2 rtems_test_busy_cpu_usage() . . . 993
8.169.3.3 rtems_test_end() . . . 993
8.169.3.4 rtems_test_parallel() . . . 994
8.169.3.5 rtems_test_parallel_get_task_id() . . . 994
8.169.3.6 rtems_test_parallel_is_master_worker() . . . 994
8.169.3.7 rtems_test_parallel_stop_job() . . . 995
8.169.3.8 rtems_test_printf() . . . 995
8.169.3.9 rtems_test_run() . . . 996
8.170 Thread Handler . . . 997
8.170.1 Detailed Description . . . 1007
8.170.2 Macro Definition Documentation . . . 1007
8.170.2.1 RTEMS_SCORE_ROBUST_THREAD_DISPATCH . . . 1007
8.170.2.2 THREAD_API_FIRST . . . 1007
8.170.2.3 THREAD_API_LAST . . . 1007
8.170.2.4 THREAD_DEFAULT_MAXIMUM_NAME_SIZE . . . 1008
8.170.2.5 THREAD_INFORMATION_DEFINE . . . 1008
8.170.2.6 THREAD_INFORMATION_DEFINE_ZERO . . . 1008
8.170.2.7 THREAD_OF_SCHEDULER_HELP_NODE . . . 1009
8.170.2.8 THREAD_WAIT_STATE_INTEND_TO_BLOCK . . . 1009
8.170.2.9 THREAD_WAIT_STATE_READY_AGAIN . . . 1009
8.170.3 Typedef Documentation . . . 1009
8.170.3.1 Thread_Action_handler . . . 1010
8.170.3.2 Thread_Configured_control . . . 1011
8.170.3.3 Thread_CPU_budget_algorithm_callout . . . 1011
8.170.3.4 Thread_Entry_numeric_type . . . 1011
8.170.3.5 Thread_queue_Configured_heads . . . 1012
8.170.3.6 Thread_Wait_flags . . . 1012
8.170.4 Enumeration Type Documentation . . . 1012
8.170.4.1 Thread_APIs . . . 1012
8.170.4.2 Thread_CPU_budget_algorithms . . . 1013
8.170.4.3 Thread_Life_state . . . 1013
8.170.4.4 Thread_Scheduler_state . . . 1013
8.170.5 Function Documentation . . . 1014
8.170.5.1 _Thread_Action_control_initialize() . . . 1014
8.170.5.2 _Thread_Action_initialize() . . . 1014
8.170.5.3 _Thread_Add_post_switch_action() . . . 1014
8.170.5.4 _Thread_Add_timeout_ticks() . . . 1015

8.170.5.5 _Thread_Allocate_unlimited() . . . 1015
8.170.5.6 _Thread_Cancel() . . . 1015
8.170.5.7 _Thread_Change_life() . . . 1016
8.170.5.8 _Thread_Clear_state() . . . 1016
8.170.5.9 _Thread_Clear_state_locked() . . . 1017
8.170.5.10 _Thread_Close() . . . 1017
8.170.5.11 _Thread_Continue() . . . 1017
8.170.5.12 _Thread_Create_idle() . . . 1018
8.170.5.13 _Thread_Dispatch() . . . 1018
8.170.5.14 _Thread_Dispatch_direct() . . . 1019
8.170.5.15 _Thread_Dispatch_disable() . . . 1019
8.170.5.16 _Thread_Dispatch_disable_critical() . . . 1019
8.170.5.17 _Thread_Dispatch_disable_with_CPU() . . . 1020
8.170.5.18 _Thread_Dispatch_enable() . . . 1020
8.170.5.19 _Thread_Dispatch_get_disable_level() . . . 1020
8.170.5.20 _Thread_Dispatch_initialization() . . . 1021
8.170.5.21 _Thread_Dispatch_is_enabled() . . . 1021
8.170.5.22 _Thread_Dispatch_request() . . . 1021
8.170.5.23 _Thread_Dispatch_unnest() . . . 1022
8.170.5.24 _Thread_Dispatch_update_heir() . . . 1022
8.170.5.25 _Thread_Do_dispatch() . . . 1022
8.170.5.26 _Thread_Do_unpin() . . . 1023
8.170.5.27 _Thread_Entry_adaptor_idle() . . . 1023
8.170.5.28 _Thread_Entry_adaptor_numeric() . . . 1023
8.170.5.29 _Thread_Entry_adaptor_pointer() . . . 1024
8.170.5.30 _Thread_Exit() . . . 1024
8.170.5.31 _Thread_Get() . . . 1024
8.170.5.32 _Thread_Get_CPU() . . . 1025
8.170.5.33 _Thread_Get_CPU_time_used() . . . 1025
8.170.5.34 _Thread_Get_heir_and_make_it_executing() . . . 1026
8.170.5.35 _Thread_Get_maximum_internal_threads() . . . 1026
8.170.5.36 _Thread_Get_name() . . . 1026
8.170.5.37 _Thread_Get_objects_information() . . . 1027
8.170.5.38 _Thread_Get_priority() . . . 1027
8.170.5.39 _Thread_Get_unmapped_priority() . . . 1028
8.170.5.40 _Thread_Get_unmapped_real_priority() . . . 1028
8.170.5.41 _Thread_Handler() . . . 1028
8.170.5.42 _Thread_Handler_initialization() . . . 1029
8.170.5.43 _Thread_Initialize() . . . 1029
8.170.5.44 _Thread_Initialize_information() . . . 1030
8.170.5.45 _Thread_Internal_allocate() . . . 1030
8.170.5.46 _Thread_Is_context_switch_necessary() . . . 1030

8.170.5.47 _Thread_Is_executing() . . . 1031
8.170.5.48 _Thread_Is_executing_on_a_processor() . . . 1031
8.170.5.49 _Thread_Is_heir() . . . 1032
8.170.5.50 _Thread_Is_joinable() . . . 1032
8.170.5.51 _Thread_Is_life_change_allowed() . . . 1032
8.170.5.52 _Thread_Is_life_changing() . . . 1033
8.170.5.53 _Thread_Is_life_restarting() . . . 1033
8.170.5.54 _Thread_Is_life_terminating() . . . 1034
8.170.5.55 _Thread_Is_ready() . . . 1034
8.170.5.56 _Thread_Iterate() . . . 1035
8.170.5.57 _Thread_Join() . . . 1035
8.170.5.58 _Thread_Kill_zombies() . . . 1035
8.170.5.59 _Thread_Load_environment() . . . 1036
8.170.5.60 _Thread_Pin() . . . 1036
8.170.5.61 _Thread_Priority_add() . . . 1036
8.170.5.62 _Thread_Priority_and_sticky_update() . . . 1037
8.170.5.63 _Thread_Priority_change() . . . 1037
8.170.5.64 _Thread_Priority_changed() . . . 1038
8.170.5.65 _Thread_Priority_highest() . . . 1038
8.170.5.66 _Thread_Priority_less_than() . . . 1039
8.170.5.67 _Thread_Priority_perform_actions() . . . 1039
8.170.5.68 _Thread_Priority_remove() . . . 1041
8.170.5.69 _Thread_Priority_replace() . . . 1041
8.170.5.70 _Thread_Priority_update() . . . 1042
8.170.5.71 _Thread_Remove_timer_and_unblock() . . . 1042
8.170.5.72 _Thread_Resource_count_decrement() . . . 1043
8.170.5.73 _Thread_Resource_count_increment() . . . 1043
8.170.5.74 _Thread_Restart_other() . . . 1043
8.170.5.75 _Thread_Restart_self() . . . 1044
8.170.5.76 _Thread_Restore_fp() . . . 1044
8.170.5.77 _Thread_Save_fp() . . . 1044
8.170.5.78 _Thread_Scheduler_acquire_critical() . . . 1045
8.170.5.79 _Thread_Scheduler_add_request() . . . 1045
8.170.5.80 _Thread_Scheduler_add_wait_node() . . . 1046
8.170.5.81 _Thread_Scheduler_cancel_need_for_help() . . . 1046
8.170.5.82 _Thread_Scheduler_get_home() . . . 1046
8.170.5.83 _Thread_Scheduler_get_home_node() . . . 1047
8.170.5.84 _Thread_Scheduler_get_node_by_index() . . . 1047
8.170.5.85 _Thread_Scheduler_process_requests() . . . 1048
8.170.5.86 _Thread_Scheduler_release_critical() . . . 1048
8.170.5.87 _Thread_Scheduler_remove_wait_node() . . . 1048
8.170.5.88 _Thread_Set_CPU() . . . 1049

8.170.5.89 _Thread_Set_life_protection() . . . 1049
8.170.5.90 _Thread_Set_name() . . . 1049
8.170.5.91 _Thread_Set_state() . . . 1050
8.170.5.92 _Thread_Set_state_locked() . . . 1050
8.170.5.93 _Thread_Start() . . . 1051
8.170.5.94 _Thread_Start_multitasking() . . . 1051
8.170.5.95 _Thread_State_acquire() . . . 1052
8.170.5.96 _Thread_State_acquire_critical() . . . 1052
8.170.5.97 _Thread_State_acquire_for_executing() . . . 1052
8.170.5.98 _Thread_State_release() . . . 1053
8.170.5.99 _Thread_State_release_critical() . . . 1053
8.170.5.100 _Thread_Timeout() . . . 1053
8.170.5.101 _Thread_Timer_initialize() . . . 1054
8.170.5.102 _Thread_Timer_insert_realtime() . . . 1054
8.170.5.103 _Thread_Timer_remove() . . . 1055
8.170.5.104 _Thread_Unblock() . . . 1055
8.170.5.105 _Thread_Unpin() . . . 1055
8.170.5.106 _Thread_Update_CPU_time_used() . . . 1056
8.170.5.107 _Thread_Wait_acquire() . . . 1056
8.170.5.108 _Thread_Wait_acquire_critical() . . . 1056
8.170.5.109 _Thread_Wait_acquire_default() . . . 1057
8.170.5.110 _Thread_Wait_acquire_default_critical() . . . 1057
8.170.5.111 _Thread_Wait_acquire_default_for_executing() . . . 1057
8.170.5.112 _Thread_Wait_acquire_queue_critical() . . . 1058
8.170.5.113 _Thread_Wait_cancel() . . . 1058
8.170.5.114 _Thread_Wait_claim() . . . 1059
8.170.5.115 _Thread_Wait_claim_finalize() . . . 1059
8.170.5.116 _Thread_Wait_flags_get() . . . 1059
8.170.5.117 _Thread_Wait_flags_get_acquire() . . . 1060
8.170.5.118 _Thread_Wait_flags_set() . . . 1060
8.170.5.119 _Thread_Wait_flags_try_change_acquire() . . . 1061
8.170.5.120 _Thread_Wait_flags_try_change_release() . . . 1061
8.170.5.121 _Thread_Wait_get_id() . . . 1062
8.170.5.122 _Thread_Wait_get_status() . . . 1062
8.170.5.123 _Thread_Wait_release() . . . 1062
8.170.5.124 _Thread_Wait_release_critical() . . . 1063
8.170.5.125 _Thread_Wait_release_default() . . . 1063
8.170.5.126 _Thread_Wait_release_default_critical() . . . 1063
8.170.5.127 _Thread_Wait_release_queue_critical() . . . 1064
8.170.5.128 _Thread_Wait_remove_request() . . . 1064
8.170.5.129 _Thread_Wait_remove_request_locked() . . . 1065
8.170.5.130 _Thread_Wait_restore_default() . . . 1065

8.170.5.131 _Thread_Wait_tranquilize() . . . 1065
8.170.5.132 _Thread_Yield() . . . 1066
8.170.5.133 rtems_iterate_over_all_threads() . . . 1066
8.170.6 Variable Documentation . . . 1066
8.170.6.1 _Thread_Control_add_on_count . . . 1066
8.170.6.2 _Thread_Control_add_ons . . . 1067
8.170.6.3 _Thread_Global_constructor . . . 1067
8.170.6.4 _Thread_Idle_body . . . 1067
8.170.6.5 _Thread_Idle_stack_size . . . 1067
8.170.6.6 _Thread_Idle_stacks . . . 1068
8.170.6.7 _Thread_Initial_thread_count . . . 1068
8.170.6.8 _Thread_Maximum_name_size . . . 1068
8.170.6.9 _Thread_Maximum_TLS_size . . . 1068
8.170.6.10 _Thread_queue_Heads_size . . . 1068
8.170.6.11 rtems_ada_self . . . 1068
8.171 Thread Queue Handler . . . 1069
8.171.1 Detailed Description . . . 1074
8.171.2 Macro Definition Documentation . . . 1074
8.171.2.1 _Thread_queue_Context_ISR_disable . . . 1074
8.171.2.2 _Thread_queue_Context_set_MP_callout . . . 1074
8.171.2.3 _Thread_queue_Dequeue . . . 1075
8.171.2.4 THREAD_QUEUE_INITIALIZER . . . 1075
8.171.2.5 THREAD_QUEUE_OBJECT_ASSERT . . . 1075
8.171.2.6 THREAD_QUEUE_QUEUE_TO_OBJECT . . . 1076
8.171.3 Typedef Documentation . . . 1076
8.171.3.1 Thread_queue_Deadlock_callout . . . 1076
8.171.3.2 Thread_queue_Enqueue_callout . . . 1076
8.171.3.3 Thread_queue_Enqueue_operation . . . 1077
8.171.3.4 Thread_queue_Extract_operation . . . 1077
8.171.3.5 Thread_queue_First_operation . . . 1077
8.171.3.6 Thread_queue_Flush_filter . . . 1078
8.171.3.7 Thread_queue_Heads . . . 1078
8.171.3.8 Thread_queue_Priority_actions_operation . . . 1079
8.171.3.9 Thread_queue_Surrender_operation . . . 1079
8.171.4 Function Documentation . . . 1079
8.171.4.1 _Thread_queue_Acquire() . . . 1080
8.171.4.2 _Thread_queue_Acquire_critical() . . . 1080
8.171.4.3 _Thread_queue_Add_timeout_monotonic_timespec() . . . 1080
8.171.4.4 _Thread_queue_Add_timeout_realtime_timespec() . . . 1081
8.171.4.5 _Thread_queue_Add_timeout_ticks() . . . 1081
8.171.4.6 _Thread_queue_Context_add_priority_update() . . . 1081
8.171.4.7 _Thread_queue_Context_clear_priority_updates() . . . 1082

8.171.4.8 _Thread_queue_Context_initialize() . . . 1082
8.171.4.9 _Thread_queue_Context_restore_priority_updates() . . . 1082
8.171.4.10 _Thread_queue_Context_save_priority_updates() . . . 1083
8.171.4.11 _Thread_queue_Context_set_deadlock_callout() . . . 1083
8.171.4.12 _Thread_queue_Context_set_enqueue_callout() . . . 1084
8.171.4.13 _Thread_queue_Context_set_enqueue_do_nothing_extra() . . . 1084
8.171.4.14 _Thread_queue_Context_set_enqueue_timeout_monotonic_timespec() . . . 1084
8.171.4.15 _Thread_queue_Context_set_enqueue_timeout_realtime_timespec() . . . 1085
8.171.4.16 _Thread_queue_Context_set_enqueue_timeout_ticks() . . . 1085
8.171.4.17 _Thread_queue_Context_set_ISR_level() . . . 1086
8.171.4.18 _Thread_queue_Context_set_thread_state() . . . 1086
8.171.4.19 _Thread_queue_Context_set_timeout_argument() . . . 1086
8.171.4.20 _Thread_queue_Context_set_timeout_ticks() . . . 1088
8.171.4.21 _Thread_queue_Deadlock_fatal() . . . 1088
8.171.4.22 _Thread_queue_Deadlock_status() . . . 1089
8.171.4.23 _Thread_queue_Destroy() . . . 1089
8.171.4.24 _Thread_queue_Dispatch_disable() . . . 1089
8.171.4.25 _Thread_queue_Do_acquire_critical() . . . 1090
8.171.4.26 _Thread_queue_Do_dequeue() . . . 1090
8.171.4.27 _Thread_queue_Do_release_critical() . . . 1090
8.171.4.28 _Thread_queue_Enqueue() . . . 1091
8.171.4.29 _Thread_queue_Enqueue_do_nothing_extra() . . . 1092
8.171.4.30 _Thread_queue_Enqueue_sticky() . . . 1092
8.171.4.31 _Thread_queue_Extract() . . . 1093
8.171.4.32 _Thread_queue_Extract_critical() . . . 1093
8.171.4.33 _Thread_queue_Extract_locked() . . . 1094
8.171.4.34 _Thread_queue_Extract_with_proxy() . . . 1095
8.171.4.35 _Thread_queue_First() . . . 1095
8.171.4.36 _Thread_queue_First_locked() . . . 1096
8.171.4.37 _Thread_queue_Flush_critical() . . . 1096
8.171.4.38 _Thread_queue_Flush_default_filter() . . . 1097
8.171.4.39 _Thread_queue_Flush_status_object_was_deleted() . . . 1097
8.171.4.40 _Thread_queue_Flush_status_unavailable() . . . 1098
8.171.4.41 _Thread_queue_Gate_add() . . . 1098
8.171.4.42 _Thread_queue_Gate_close() . . . 1099
8.171.4.43 _Thread_queue_Gate_open() . . . 1099
8.171.4.44 _Thread_queue_Gate_wait() . . . 1099
8.171.4.45 _Thread_queue_Heads_initialize() . . . 1100
8.171.4.46 _Thread_queue_Initialize() . . . 1100
8.171.4.47 _Thread_queue_Is_empty() . . . 1100
8.171.4.48 _Thread_queue_Object_initialize() . . . 1101
8.171.4.49 _Thread_queue_Path_acquire_critical() . . . 1101

8.171.4.50 _Thread_queue_Path_release_critical() . . . 1102
8.171.4.51 _Thread_queue_Queue_do_acquire_critical() . . . 1102
8.171.4.52 _Thread_queue_Queue_get_name_and_id() . . . 1103
8.171.4.53 _Thread_queue_Queue_initialize() . . . 1103
8.171.4.54 _Thread_queue_Queue_release() . . . 1103
8.171.4.55 _Thread_queue_Queue_release_critical() . . . 1104
8.171.4.56 _Thread_queue_Release() . . . 1104
8.171.4.57 _Thread_queue_Release_critical() . . . 1104
8.171.4.58 _Thread_queue_Surrender() . . . 1105
8.171.4.59 _Thread_queue_Surrender_sticky() . . . 1105
8.171.4.60 _Thread_queue_Unblock_critical() . . . 1106
8.171.5 Variable Documentation . . . 1106
8.171.5.1 _Thread_queue_Object_name . . . 1106
8.172 Thread States . . . 1107
8.172.1 Detailed Description . . . 1108
8.172.2 Macro Definition Documentation . . . 1108
8.172.2.1 STATES_ALL_SET . . . 1108
8.172.2.2 STATES_BLOCKED . . . 1109
8.172.2.3 STATES_DEBUGGER . . . 1109
8.172.2.4 STATES_DORMANT . . . 1109
8.172.2.5 STATES_INTERRUPTIBLE_BY_SIGNAL . . . 1109
8.172.2.6 STATES_LIFE_IS_CHANGING . . . 1110
8.172.2.7 STATES_LOCALLY_BLOCKED . . . 1110
8.172.2.8 STATES_READY . . . 1110
8.172.2.9 STATES_SUSPENDED . . . 1110
8.172.2.10 STATES_WAITING_FOR_BARRIER . . . 1111
8.172.2.11 STATES_WAITING_FOR_BSD_WAKEUP . . . 1111
8.172.2.12 STATES_WAITING_FOR_CONDITION_VARIABLE . . . 1111
8.172.2.13 STATES_WAITING_FOR_EVENT . . . 1111
8.172.2.14 STATES_WAITING_FOR_FUTEX . . . 1111
8.172.2.15 STATES_WAITING_FOR_JOIN . . . 1112
8.172.2.16 STATES_WAITING_FOR_JOIN_AT_EXIT . . . 1112
8.172.2.17 STATES_WAITING_FOR_MESSAGE . . . 1112
8.172.2.18 STATES_WAITING_FOR_MUTEX . . . 1112
8.172.2.19 STATES_WAITING_FOR_PERIOD . . . 1112
8.172.2.20 STATES_WAITING_FOR_RPC_REPLY . . . 1113
8.172.2.21 STATES_WAITING_FOR_RWLOCK . . . 1113
8.172.2.22 STATES_WAITING_FOR_SEGMENT . . . 1113
8.172.2.23 STATES_WAITING_FOR_SEMAPHORE . . . 1113
8.172.2.24 STATES_WAITING_FOR_SIGNAL . . . 1113
8.172.2.25 STATES_WAITING_FOR_SYSTEM_EVENT . . . 1114
8.172.2.26 STATES_ZOMBIE . . . 1114

8.172.3 Typedef Documentation . . . 1114
8.172.3.1 States_Control . . . 1114
8.172.4 Function Documentation . . . 1114
8.172.4.1 _States_Clear() . . . 1114
8.172.4.2 _States_Is_dormant() . . . 1115
8.172.4.3 _States_Is_interruptible_by_signal() . . . 1115
8.172.4.4 _States_Is_locally_blocked() . . . 1116
8.172.4.5 _States_Is_ready() . . . 1116
8.172.4.6 _States_Is_suspended() . . . 1117
8.172.4.7 _States_Is_waiting_for_join_at_exit() . . . 1117
8.172.4.8 _States_Is_waiting_for_rpc_reply() . . . 1117
8.172.4.9 _States_Set() . . . 1118
8.173 Thread-Local Storage (TLS) . . . 1119
8.173.1 Detailed Description . . . 1120
8.173.2 Function Documentation . . . 1120
8.173.2.1 _TLS_Copy_and_clear() . . . 1120
8.173.2.2 _TLS_Get_allocation_size() . . . 1120
8.173.2.3 _TLS_Get_size() . . . 1121
8.173.2.4 _TLS_Get_thread_control_block_area_size() . . . 1121
8.173.2.5 _TLS_Heap_align_up() . . . 1121
8.173.2.6 _TLS_Initialize() . . . 1122
8.173.2.7 _TLS_TCB_after_TLS_block_initialize() . . . 1122
8.173.2.8 _TLS_TCB_at_area_begin_initialize() . . . 1123
8.173.2.9 _TLS_TCB_before_TLS_block_initialize() . . . 1123
8.173.3 Variable Documentation . . . 1124
8.173.3.1 _TLS_Alignment . . . 1124
8.174 Time Services . . . 1125
8.174.1 Detailed Description . . . 1125
8.175 Time Test 27 Support . . . 1126
8.176 Time of Day Handler . . . 1127
8.176.1 Detailed Description . . . 1128
8.176.2 Macro Definition Documentation . . . 1128
8.176.2.1 TOD_DAYS_PER_YEAR . . . 1129
8.176.2.2 TOD_HOURS_PER_DAY . . . 1129
8.176.2.3 TOD_MICROSECONDS_PER_SECOND . . . 1129
8.176.2.4 TOD_MILLISECONDS_PER_SECOND . . . 1129
8.176.2.5 TOD_MINUTES_PER_HOUR . . . 1129
8.176.2.6 TOD_MONTHS_PER_YEAR . . . 1130
8.176.2.7 TOD_NANOSECONDS_PER_MICROSECOND . . . 1130
8.176.2.8 TOD_NANOSECONDS_PER_SECOND . . . 1130
8.176.2.9 TOD_SECONDS_PER_DAY . . . 1130
8.176.2.10 TOD_SECONDS_PER_MINUTE . . . 1130

8.176.2.11 TOD_SECONDS_PER_NON_LEAP_YEAR . . . 1131
8.176.2.12 TOD_TICKS_PER_SECOND . . . 1131
8.176.3 Function Documentation . . . 1131
8.176.3.1 _TOD_Acquire() . . . 1131
8.176.3.2 _TOD_Adjust() . . . 1132
8.176.3.3 _TOD_Get() . . . 1132
8.176.3.4 _TOD_Get_timeval() . . . 1132
8.176.3.5 _TOD_Get_uptime() . . . 1132
8.176.3.6 _TOD_Get_zero_based_uptime() . . . 1133
8.176.3.7 _TOD_Get_zero_based_uptime_as_timespec() . . . 1133
8.176.3.8 _TOD_Is_set() . . . 1134
8.176.3.9 _TOD_Release() . . . 1134
8.176.3.10 _TOD_Seconds_since_epoch() . . . 1134
8.176.3.11 _TOD_Set() . . . 1135
8.176.3.12 TOD_TICKS_PER_SECOND_method() . . . 1136
8.177 Time of Day Handler Action Hooks . . . 1137
8.177.1 Detailed Description . . . 1137
8.177.2 Enumeration Type Documentation . . . 1137
8.177.2.1 TOD_Action . . . 1137
8.177.3 Function Documentation . . . 1138
8.177.3.1 _TOD_Hook_Register() . . . 1138
8.177.3.2 _TOD_Hook_Run() . . . 1138
8.177.3.3 _TOD_Hook_Unregister() . . . 1138
8.178 Timecounter Handler . . . 1140
8.178.1 Detailed Description . . . 1141
8.178.2 Macro Definition Documentation . . . 1141
8.178.2.1 _Timecounter_Acquire . . . 1141
8.178.2.2 _Timecounter_Release . . . 1142
8.178.3 Function Documentation . . . 1142
8.178.3.1 _Timecounter_Bintime() . . . 1142
8.178.3.2 _Timecounter_Binuptime() . . . 1142
8.178.3.3 _Timecounter_Getbintime() . . . 1143
8.178.3.4 _Timecounter_Getbinuptime() . . . 1143
8.178.3.5 _Timecounter_Getboottime() . . . 1143
8.178.3.6 _Timecounter_Getboottimebin() . . . 1143
8.178.3.7 _Timecounter_Getmicrotime() . . . 1145
8.178.3.8 _Timecounter_Getmicrouptime() . . . 1145
8.178.3.9 _Timecounter_Getnanotime() . . . 1145
8.178.3.10 _Timecounter_Getnanouptime() . . . 1147
8.178.3.11 _Timecounter_Install() . . . 1147
8.178.3.12 _Timecounter_Microtime() . . . 1147
8.178.3.13 _Timecounter_Microuptime() . . . 1148

8.178.3.14 _Timecounter_Nanotime() . . . 1148
8.178.3.15 _Timecounter_Nanouptime() . . . 1148
8.178.3.16 _Timecounter_Sbinuptime() . . . 1149
8.178.3.17 _Timecounter_Set_clock() . . . 1149
8.178.3.18 _Timecounter_Tick_simple() . . . 1149
8.178.4 Variable Documentation . . . 1150
8.178.4.1 _Timecounter_Time_uptime . . . 1150
8.179 Timecounter Support . . . 1151
8.179.1 Detailed Description . . . 1152
8.179.2 Macro Definition Documentation . . . 1152
8.179.2.1 RTEMS_TIMECOUNTER_QUALITY_CLOCK_DRIVER . . . 1152
8.179.3 Function Documentation . . . 1152
8.179.3.1 rtems_timecounter_install() . . . 1152
8.179.3.2 rtems_timecounter_simple_downcounter_get() . . . 1153
8.179.3.3 rtems_timecounter_simple_downcounter_tick() . . . 1153
8.179.3.4 rtems_timecounter_simple_install() . . . 1153
8.179.3.5 rtems_timecounter_simple_scale() . . . 1154
8.179.3.6 rtems_timecounter_simple_upcounter_get() . . . 1155
8.179.3.7 rtems_timecounter_simple_upcounter_tick() . . . 1155
8.179.3.8 rtems_timecounter_tick() . . . 1155
8.180 Timer Manager . . . 1157
8.180.1 Detailed Description . . . 1158
8.180.2 Enumeration Type Documentation . . . 1158
8.180.2.1 Timer_Classes . . . 1158
8.180.3 Function Documentation . . . 1158
8.180.3.1 rtems_timer_cancel() . . . 1159
8.180.3.2 rtems_timer_create() . . . 1159
8.180.3.3 rtems_timer_delete() . . . 1159
8.180.3.4 rtems_timer_fire_after() . . . 1159
8.180.3.5 rtems_timer_fire_when() . . . 1160
8.180.3.6 rtems_timer_get_information() . . . 1160
8.180.3.7 rtems_timer_ident() . . . 1161
8.180.3.8 rtems_timer_initiate_server() . . . 1161
8.180.3.9 rtems_timer_reset() . . . 1162
8.180.3.10 rtems_timer_server_fire_after() . . . 1162
8.180.3.11 rtems_timer_server_fire_when() . . . 1162
8.181 Tracing . . . 1163
8.182 UART . . . 1164
8.182.1 Detailed Description . . . 1164
8.183 Unsigned 16-Bit Integer Checks . . . 1165
8.183.1 Detailed Description . . . 1165
8.184 Unsigned 32-Bit Integer Checks . . . 1166

8.184.1 Detailed Description . . . 1166
8.185 Unsigned 64-Bit Integer Checks . . . 1167
8.185.1 Detailed Description . . . 1167
8.186 Unsigned 8-Bit Integer Checks . . . 1168
8.186.1 Detailed Description . . . 1168
8.187 Unsigned Character Checks . . . 1169
8.187.1 Detailed Description . . . 1169
8.188 Unsigned Integer Checks . . . 1170
8.188.1 Detailed Description . . . 1170
8.189 Unsigned Long Integer Checks . . . 1171
8.189.1 Detailed Description . . . 1171
8.190 Unsigned Long Long Integer Checks . . . 1172
8.190.1 Detailed Description . . . 1172
8.191 Unsigned Pointer Value Checks . . . 1173
8.191.1 Detailed Description . . . 1173
8.192 Unsigned Short Integer Checks . . . 1174
8.192.1 Detailed Description . . . 1174
8.193 User Extension Handler . . . 1175
8.193.1 Detailed Description . . . 1178
8.193.2 Typedef Documentation . . . 1178
8.193.2.1 User_extensions_fatal_extension . . . 1178
8.193.2.2 User_extensions_Iterator . . . 1178
8.193.2.3 User_extensions_thread_begin_extension . . . 1178
8.193.2.4 User_extensions_thread_create_extension . . . 1179
8.193.2.5 User_extensions_thread_delete_extension . . . 1179
8.193.2.6 User_extensions_thread_exitted_extension . . . 1181
8.193.2.7 User_extensions_thread_restart_extension . . . 1181
8.193.2.8 User_extensions_thread_start_extension . . . 1182
8.193.2.9 User_extensions_thread_switch_extension . . . 1182
8.193.2.10 User_extensions_thread_terminate_extension . . . 1182
8.193.2.11 User_extensions_Visitor . . . 1183
8.193.3 Function Documentation . . . 1183
8.193.3.1 _User_extensions_Acquire() . . . 1183
8.193.3.2 _User_extensions_Add_API_set() . . . 1184
8.193.3.3 _User_extensions_Add_set() . . . 1184
8.193.3.4 _User_extensions_Add_set_with_table() . . . 1184
8.193.3.5 _User_extensions_Destroy_iterators() . . . 1185
8.193.3.6 _User_extensions_Fatal() . . . 1185
8.193.3.7 _User_extensions_Fatal_visitor() . . . 1185
8.193.3.8 _User_extensions_Iterate() . . . 1186
8.193.3.9 _User_extensions_Release() . . . 1186
8.193.3.10 _User_extensions_Remove_set() . . . 1186

8.193.3.11 _User_extensions_Thread_begin() . . . 1187
8.193.3.12 _User_extensions_Thread_begin_visitor() . . . 1187
8.193.3.13 _User_extensions_Thread_create() . . . 1187
8.193.3.14 _User_extensions_Thread_create_visitor() . . . 1188
8.193.3.15 _User_extensions_Thread_delete() . . . 1188
8.193.3.16 _User_extensions_Thread_delete_visitor() . . . 1188
8.193.3.17 _User_extensions_Thread_exitted() . . . 1189
8.193.3.18 _User_extensions_Thread_exitted_visitor() . . . 1189
8.193.3.19 _User_extensions_Thread_restart() . . . 1189
8.193.3.20 _User_extensions_Thread_restart_visitor() . . . 1190
8.193.3.21 _User_extensions_Thread_start() . . . 1190
8.193.3.22 _User_extensions_Thread_start_visitor() . . . 1190
8.193.3.23 _User_extensions_Thread_switch() . . . 1191
8.193.3.24 _User_extensions_Thread_terminate() . . . 1191
8.193.3.25 _User_extensions_Thread_terminate_visitor() . . . 1191
8.193.4 Variable Documentation . . . 1192
8.193.4.1 _User_extensions_Initial_count . . . 1192
8.193.4.2 _User_extensions_Initial_extensions . . . 1192
8.193.4.3 _User_extensions_Initial_switch_controls . . . 1192
8.194 User Extensions Implementation . . . 1193
8.194.1 Detailed Description . . . 1193
8.194.2 Macro Definition Documentation . . . 1193
8.194.2.1 EXTENSION_INFORMATION_DEFINE . . . 1193
8.195 User Extensions Manager . . . 1195
8.195.1 Detailed Description . . . 1195
8.195.2 Function Documentation . . . 1196
8.195.2.1 rtems_extension_create() . . . 1196
8.195.2.2 rtems_extension_delete() . . . 1196
8.195.2.3 rtems_extension_ident() . . . 1196
8.196 Version . . . 1198
8.196.1 Detailed Description . . . 1198
8.196.2 Function Documentation . . . 1198
8.196.2.1 rtems_board_support_package() . . . 1198
8.196.2.2 rtems_version() . . . 1199
8.196.2.3 rtems_version_control_key() . . . 1199
8.196.2.4 rtems_version_control_key_is_valid() . . . 1199
8.196.2.5 rtems_version_major() . . . 1200
8.196.2.6 rtems_version_minor() . . . 1200
8.196.2.7 rtems_version_revision() . . . 1200
8.197 Watchdog Handler . . . 1201
8.197.1 Detailed Description . . . 1204
8.197.2 Macro Definition Documentation . . . 1204

8.197.2.1 WATCHDOG_BITS_FOR_1E9_NANOSECONDS . . . 1204
8.197.2.2 WATCHDOG_INITIALIZER . . . 1204
8.197.2.3 WATCHDOG_MAX_SECONDS . . . 1205
8.197.3 Typedef Documentation . . . 1205
8.197.3.1 Watchdog_Interval . . . 1205
8.197.3.2 Watchdog_Service_routine . . . 1205
8.197.3.3 Watchdog_Service_routine_entry . . . 1205
8.197.4 Enumeration Type Documentation . . . 1205
8.197.4.1 Watchdog_State . . . 1205
8.197.5 Function Documentation . . . 1206
8.197.5.1 _Watchdog_Cancel() . . . 1206
8.197.5.2 _Watchdog_Do_tickle() . . . 1206
8.197.5.3 _Watchdog_Future_timespec() . . . 1207
8.197.5.4 _Watchdog_Get_CPU() . . . 1207
8.197.5.5 _Watchdog_Get_state() . . . 1208
8.197.5.6 _Watchdog_Header_destroy() . . . 1208
8.197.5.7 _Watchdog_Header_first() . . . 1208
8.197.5.8 _Watchdog_Header_initialize() . . . 1209
8.197.5.9 _Watchdog_Initialize() . . . 1209
8.197.5.10 _Watchdog_Insert() . . . 1209
8.197.5.11 _Watchdog_Is_far_future_timespec() . . . 1210
8.197.5.12 _Watchdog_Is_scheduled() . . . 1210
8.197.5.13 _Watchdog_Is_valid_interval_timespec() . . . 1211
8.197.5.14 _Watchdog_Is_valid_timespec() . . . 1211
8.197.5.15 _Watchdog_Next_first() . . . 1211
8.197.5.16 _Watchdog_Per_CPU_acquire_critical() . . . 1212
8.197.5.17 _Watchdog_Per_CPU_insert() . . . 1212
8.197.5.18 _Watchdog_Per_CPU_insert_ticks() . . . 1213
8.197.5.19 _Watchdog_Per_CPU_release_critical() . . . 1213
8.197.5.20 _Watchdog_Per_CPU_remove() . . . 1214
8.197.5.21 _Watchdog_Per_CPU_remove_ticks() . . . 1214
8.197.5.22 _Watchdog_Preinitialize() . . . 1214
8.197.5.23 _Watchdog_Remove() . . . 1215
8.197.5.24 _Watchdog_Set_CPU() . . . 1215
8.197.5.25 _Watchdog_Set_state() . . . 1215
8.197.5.26 _Watchdog_Tick() . . . 1216
8.197.5.27 _Watchdog_Ticks_from_sbintime() . . . 1216
8.197.5.28 _Watchdog_Ticks_from_seconds() . . . 1216
8.197.5.29 _Watchdog_Ticks_from_timespec() . . . 1217
8.197.6 Variable Documentation . . . 1217
8.197.6.1 _Watchdog_Microseconds_per_tick . . . 1217
8.197.6.2 _Watchdog_Nanoseconds_per_tick . . . 1217

8.197.6.3 _Watchdog_Ticks_per_second . . . 1218
8.197.6.4 _Watchdog_Ticks_per_timeslice . . . 1218
8.197.6.5 _Watchdog_Ticks_since_boot . . . 1218
8.198 Workspace Handler . . . 1219
8.198.1 Detailed Description . . . 1220
8.198.2 Function Documentation . . . 1220
8.198.2.1 _Workspace_Allocate() . . . 1220
8.198.2.2 _Workspace_Free() . . . 1220
8.198.2.3 _Workspace_Handler_initialization() . . . 1221
8.198.2.4 _Workspace_Malloc_initialize_separate() . . . 1221
8.198.2.5 _Workspace_Malloc_initialize_unified() . . . 1221
8.198.2.6 _Workspace_String_duplicate() . . . 1221
8.198.3 Variable Documentation . . . 1222
8.198.3.1 _Workspace_Area . . . 1222
8.198.3.2 _Workspace_Is_unified . . . 1222
8.198.3.3 _Workspace_Malloc_initializer . . . 1222
8.198.3.4 _Workspace_Size . . . 1222
8.199 libfs . . . 1223
8.200 nfs Client . . . 1224
8.201 spec:/rtems/attr/val/attr . . . 1225
8.201.1 Detailed Description . . . 1225
8.202 spec:/rtems/barrier/val/ident . . . 1226
8.202.1 Detailed Description . . . 1226
8.203 spec:/rtems/event/req/send-receive . . . 1227
8.203.1 Detailed Description . . . 1229
8.203.2 Function Documentation . . . 1229
8.203.2.1 RtemsEventReqSendReceive_Run() . . . 1230
8.203.3 Variable Documentation . . . 1230
8.203.3.1 InterruptConfig . . . 1230
8.203.3.2 RtemsEventReqSendReceive_Fixture . . . 1230
8.203.3.3 RtemsEventReqSendReceive_PreDesc . . . 1231
8.203.3.4 RtemsEventReqSendReceive_PreDesc_Id . . . 1231
8.203.3.5 RtemsEventReqSendReceive_PreDesc_ReceiverState . . . 1231
8.203.3.6 RtemsEventReqSendReceive_PreDesc_Satisfy . . . 1231
8.203.3.7 RtemsEventReqSendReceive_PreDesc_Send . . . 1232
8.204 spec:/rtems/event/val/event-constant . . . 1233
8.204.1 Detailed Description . . . 1233
8.204.2 Function Documentation . . . 1234
8.204.2.1 RtemsEventValEventConstant_Run() . . . 1235
8.205 spec:/rtems/event/val/events . . . 1237
8.205.1 Detailed Description . . . 1237
8.206 spec:/rtems/event/val/send-receive . . . 1238

8.206.1 Detailed Description . . . 1238
8.207 spec:/rtems/event/val/system-send-receive . . . 1239
8.207.1 Detailed Description . . . 1239
8.208 spec:/rtems/message/req/construct-errors . . . 1240
8.208.1 Detailed Description . . . 1242
8.208.2 Variable Documentation . . . 1242
8.208.2.1 RtemsMessageReqConstructErrors_Fixture . . . 1242
8.208.2.2 RtemsMessageReqConstructErrors_PreDesc . . . 1242
8.208.2.3 RtemsMessageReqConstructErrors_PreDesc_Area . . . 1243
8.208.2.4 RtemsMessageReqConstructErrors_PreDesc_AreaSize . . . 1243
8.208.2.5 RtemsMessageReqConstructErrors_PreDesc_Id . . . 1243
8.208.2.6 RtemsMessageReqConstructErrors_PreDesc_MaxPending . . . 1243
8.208.2.7 RtemsMessageReqConstructErrors_PreDesc_MaxSize . . . 1244
8.208.2.8 RtemsMessageReqConstructErrors_PreDesc_Name . . . 1244
8.208.2.9 RtemsMessageReqConstructErrors_PreDesc_Queues . . . 1244
8.209 spec:/rtems/message/val/ident . . . 1245
8.209.1 Detailed Description . . . 1245
8.210 spec:/rtems/mode/val/modes . . . 1246
8.210.1 Detailed Description . . . 1246
8.211 spec:/rtems/option/val/options . . . 1247
8.211.1 Detailed Description . . . 1247
8.212 spec:/rtems/part/req/create . . . 1248
8.212.1 Detailed Description . . . 1250
8.212.2 Variable Documentation . . . 1250
8.212.2.1 RtemsPartReqCreate_Fixture . . . 1250
8.212.2.2 RtemsPartReqCreate_PreDesc . . . 1250
8.212.2.3 RtemsPartReqCreate_PreDesc_Id . . . 1250
8.212.2.4 RtemsPartReqCreate_PreDesc_Length . . . 1251
8.212.2.5 RtemsPartReqCreate_PreDesc_Name . . . 1251
8.212.2.6 RtemsPartReqCreate_PreDesc_Parts . . . 1251
8.212.2.7 RtemsPartReqCreate_PreDesc_Size . . . 1251
8.212.2.8 RtemsPartReqCreate_PreDesc_Start . . . 1251
8.213 spec:/rtems/part/req/delete . . . 1252
8.213.1 Detailed Description . . . 1253
8.213.2 Variable Documentation . . . 1253
8.213.2.1 RtemsPartReqDelete_Fixture . . . 1253
8.213.2.2 RtemsPartReqDelete_PreDesc . . . 1254
8.213.2.3 RtemsPartReqDelete_PreDesc_Id . . . 1254
8.213.2.4 RtemsPartReqDelete_PreDesc_InUse . . . 1254
8.213.2.5 RtemsPartReqDelete_TransitionInfo . . . 1255
8.213.2.6 RtemsPartReqDelete_TransitionMap . . . 1255
8.214 spec:/rtems/part/req/get-buffer . . . 1256

8.214.1 Detailed Description . . . 1257
8.214.2 Variable Documentation . . . 1257
8.214.2.1 RtemsPartReqGetBuffer_Fixture . . . 1257
8.214.2.2 RtemsPartReqGetBuffer_PreDesc . . . 1258
8.214.2.3 RtemsPartReqGetBuffer_PreDesc_Avail . . . 1258
8.214.2.4 RtemsPartReqGetBuffer_PreDesc_Buf . . . 1258
8.214.2.5 RtemsPartReqGetBuffer_PreDesc_Id . . . 1259
8.214.2.6 RtemsPartReqGetBuffer_TransitionInfo . . . 1259
8.214.2.7 RtemsPartReqGetBuffer_TransitionMap . . . 1259
8.215 spec:/rtems/part/req/return-buffer . . . 1260
8.215.1 Detailed Description . . . 1261
8.215.2 Variable Documentation . . . 1261
8.215.2.1 RtemsPartReqReturnBuffer_Fixture . . . 1261
8.215.2.2 RtemsPartReqReturnBuffer_PreDesc . . . 1262
8.215.2.3 RtemsPartReqReturnBuffer_PreDesc_Buf . . . 1262
8.215.2.4 RtemsPartReqReturnBuffer_PreDesc_Id . . . 1262
8.215.2.5 RtemsPartReqReturnBuffer_TransitionInfo . . . 1263
8.215.2.6 RtemsPartReqReturnBuffer_TransitionMap . . . 1263
8.216 spec:/rtems/part/val/ident . . . 1264
8.216.1 Detailed Description . . . 1264
8.217 spec:/rtems/part/val/part . . . 1265
8.217.1 Detailed Description . . . 1265
8.218 spec:/rtems/ratemon/val/ident . . . 1266
8.218.1 Detailed Description . . . 1266
8.219 spec:/rtems/req/ident . . . 1267
8.219.1 Detailed Description . . . 1268
8.219.2 Function Documentation . . . 1268
8.219.2.1 RtemsReqIdent_Run() . . . 1268
8.219.3 Variable Documentation . . . 1269
8.219.3.1 RtemsReqIdent_Fixture . . . 1269
8.219.3.2 RtemsReqIdent_PreDesc . . . 1269
8.219.3.3 RtemsReqIdent_PreDesc_Id . . . 1269
8.219.3.4 RtemsReqIdent_PreDesc_Name . . . 1270
8.219.3.5 RtemsReqIdent_PreDesc_Node . . . 1270
8.220 spec:/rtems/req/ident-local . . . 1271
8.220.1 Detailed Description . . . 1272
8.220.2 Function Documentation . . . 1272
8.220.2.1 RtemsReqIdentLocal_Run() . . . 1272
8.220.3 Variable Documentation . . . 1273
8.220.3.1 RtemsReqIdentLocal_Fixture . . . 1273
8.220.3.2 RtemsReqIdentLocal_PreDesc . . . 1273
8.220.3.3 RtemsReqIdentLocal_PreDesc_Id . . . 1273

8.220.3.4 RtemsReqIdentLocal_PreDesc_Name . . . 1274
8.220.3.5 RtemsReqIdentLocal_TransitionInfo . . . 1274
8.220.3.6 RtemsReqIdentLocal_TransitionMap . . . 1274
8.221 spec:/rtems/sem/val/ident . . . 1275
8.221.1 Detailed Description . . . 1275
8.222 spec:/rtems/task/req/construct-errors . . . 1276
8.222.1 Detailed Description . . . 1278
8.222.2 Variable Documentation . . . 1278
8.222.2.1 extensions . . . 1278
8.222.2.2 RtemsTaskReqConstructErrors_Fixture . . . 1279
8.222.2.3 RtemsTaskReqConstructErrors_PreDesc . . . 1279
8.222.2.4 RtemsTaskReqConstructErrors_PreDesc_Ext . . . 1279
8.222.2.5 RtemsTaskReqConstructErrors_PreDesc_Id . . . 1279
8.222.2.6 RtemsTaskReqConstructErrors_PreDesc_Name . . . 1280
8.222.2.7 RtemsTaskReqConstructErrors_PreDesc_Preempt . . . 1280
8.222.2.8 RtemsTaskReqConstructErrors_PreDesc_Prio . . . 1280
8.222.2.9 RtemsTaskReqConstructErrors_PreDesc_Stack . . . 1280
8.222.2.10 RtemsTaskReqConstructErrors_PreDesc_Tasks . . . 1281
8.222.2.11 RtemsTaskReqConstructErrors_PreDesc_TLS . . . 1281
8.223 spec:/rtems/task/req/ident . . . 1282
8.223.1 Detailed Description . . . 1283
8.223.2 Variable Documentation . . . 1283
8.223.2.1 ClassicTaskIdentConfig . . . 1283
8.223.2.2 RtemsTaskReqIdent_Fixture . . . 1284
8.223.2.3 RtemsTaskReqIdent_PreDesc . . . 1284
8.223.2.4 RtemsTaskReqIdent_PreDesc_Pre . . . 1284
8.223.2.5 RtemsTaskReqIdent_TransitionInfo . . . 1285
8.223.2.6 RtemsTaskReqIdent_TransitionMap . . . 1285
8.224 spec:/rtems/timer/val/ident . . . 1286
8.224.1 Detailed Description . . . 1286
8.225 spec:/rtems/userext/val/ident . . . 1287
8.225.1 Detailed Description . . . 1287
8.226 spec:/testsuites/validation-0 . . . 1288
8.226.1 Detailed Description . . . 1290
8.226.2 Macro Definition Documentation . . . 1290
8.226.2.1 CONFIGURE_SCHEDULER_ASSIGNMENTS . . . 1290
8.226.2.2 CONFIGURE_SCHEDULER_TABLE_ENTRIES . . . 1290
8.226.3 Variable Documentation . . . 1290
8.226.3.1 actions . . . 1291
8.226.3.2 runner_task_config . . . 1291
8.226.3.3 test_config . . . 1291
8.227 spec:/testsuites/validation/c-library . . . 1292

8.227.1 Detailed Description . . . 1292
8.228 spec:/testsuites/validation/classic-barrier . . . 1293
8.228.1 Detailed Description . . . 1293
8.229 spec:/testsuites/validation/profile . . . 1294
8.229.1 Detailed Description . . . 1295
8.229.2 Variable Documentation . . . 1295
8.229.2.1 actions . . . 1295
8.229.2.2 task_config . . . 1295
8.229.2.3 test_config . . . 1296

**9 Class Documentation** **1297**

9.1 __rtems_dev_t Union Reference . . . 1297
9.1.1 Detailed Description . . . 1297
9.2 _rtems_filesystem_file_handlers_r Struct Reference . . . 1297
9.2.1 Detailed Description . . . 1298
9.3 _rtems_filesystem_operations_table Struct Reference . . . 1298
9.3.1 Detailed Description . . . 1299
9.4 _Scheduler_Control Struct Reference . . . 1299
9.4.1 Detailed Description . . . 1299
9.4.2 Member Data Documentation . . . 1299
9.4.2.1 maximum_priority . . . 1299
9.5 _Thread_Control Struct Reference . . . 1300
9.5.1 Detailed Description . . . 1300
9.5.2 Member Data Documentation . . . 1301
9.5.2.1 API_Extensions . . . 1301
9.5.2.2 budget_algorithm . . . 1301
9.5.2.3 budget_callout . . . 1301
9.5.2.4 cpu_time_budget . . . 1301
9.5.2.5 cpu_time_used . . . 1302
9.5.2.6 current_state . . . 1302
9.5.2.7 extensions . . . 1302
9.5.2.8 is_fp . . . 1302
9.5.2.9 is_idle . . . 1302
9.5.2.10 is_preemptible . . . 1303
9.5.2.11 Join_queue . . . 1303
9.5.2.12 libc_reent . . . 1303
9.5.2.13 Life . . . 1304
9.5.2.14 Object . . . 1304
9.5.2.15 Registers . . . 1304
9.5.2.16 Start . . . 1304
9.5.2.17 Timer . . . 1304
9.5.2.18 Wait . . . 1305

9.6 _Thread_queue_Heads Struct Reference . . . 1305
9.6.1 Detailed Description . . . 1305
9.6.2 Member Data Documentation . . . 1305
9.6.2.1 Fifo . . . 1306
9.6.2.2 Heads . . . 1306
9.7 ambapp_ahb_bus Struct Reference . . . 1306
9.7.1 Detailed Description . . . 1306
9.8 ambapp_ahb_info Struct Reference . . . 1306
9.8.1 Detailed Description . . . 1307
9.9 ambapp_apb_info Struct Reference . . . 1307
9.9.1 Detailed Description . . . 1307
9.10 ambapp_bus Struct Reference . . . 1307
9.10.1 Detailed Description . . . 1307
9.11 ambapp_common_info Struct Reference . . . 1307
9.11.1 Detailed Description . . . 1308
9.12 ambapp_core Struct Reference . . . 1308
9.12.1 Detailed Description . . . 1308
9.13 ambapp_dev Struct Reference . . . 1308
9.13.1 Detailed Description . . . 1308
9.14 ambapp_mmap Struct Reference . . . 1309
9.14.1 Detailed Description . . . 1309
9.15 ambapp_pnp_ahb Struct Reference . . . 1309
9.15.1 Detailed Description . . . 1309
9.16 ambapp_pnp_apb Struct Reference . . . 1309
9.16.1 Detailed Description . . . 1309
9.17 apbuart_regs Struct Reference . . . 1310
9.17.1 Detailed Description . . . 1310
9.18 API_Mutex_Control Struct Reference . . . 1310
9.18.1 Detailed Description . . . 1310
9.18.2 Member Data Documentation . . . 1310
9.18.2.1 Mutex . . . 1310
9.19 ASR_Information Struct Reference . . . 1311
9.19.1 Detailed Description . . . 1311
9.19.2 Member Data Documentation . . . 1311
9.19.2.1 handler . . . 1311
9.19.2.2 is_enabled . . . 1311
9.19.2.3 mode_set . . . 1312
9.19.2.4 nest_level . . . 1312
9.19.2.5 signals_pending . . . 1312
9.19.2.6 signals_posted . . . 1312
9.20 Barrier_Control Struct Reference . . . 1312
9.20.1 Detailed Description . . . 1313

9.20.2 Member Data Documentation . . . 1313
9.20.2.1 attribute_set . . . 1313
9.20.2.2 Barrier . . . 1313
9.20.2.3 Object . . . 1313
9.21 bsp_interrupt_handler_entry Struct Reference . . . 1314
9.21.1 Detailed Description . . . 1314
9.22 Chain_Control Struct Reference . . . 1314
9.22.1 Detailed Description . . . 1314
9.23 Chain_Iterator Struct Reference . . . 1315
9.23.1 Detailed Description . . . 1315
9.23.2 Member Data Documentation . . . 1315
9.23.2.1 direction . . . 1315
9.23.2.2 position . . . 1316
9.23.2.3 Registry_node . . . 1316
9.24 Chain_Iterator_registry Struct Reference . . . 1316
9.24.1 Detailed Description . . . 1316
9.25 Chain_Node_struct Struct Reference . . . 1317
9.25.1 Detailed Description . . . 1317
9.25.2 Member Data Documentation . . . 1317
9.25.2.1 next . . . 1317
9.25.2.2 previous . . . 1317
9.26 Context_Control Struct Reference . . . 1318
9.26.1 Detailed Description . . . 1318
9.26.2 Member Data Documentation . . . 1318
9.26.2.1 g5 . . . 1319
9.26.2.2 g7 . . . 1319
9.26.2.3 i0 . . . 1319
9.26.2.4 i1 . . . 1319
9.26.2.5 i2 . . . 1319
9.26.2.6 i3 . . . 1320
9.26.2.7 i4 . . . 1320
9.26.2.8 i5 . . . 1320
9.26.2.9 i6_fp . . . 1320
9.26.2.10 i7 . . . 1320
9.26.2.11 isr_dispatch_disable . . . 1321
9.26.2.12 l0_and_l1 . . . 1321
9.26.2.13 l2 . . . 1321
9.26.2.14 l3 . . . 1321
9.26.2.15 l4 . . . 1321
9.26.2.16 l5 . . . 1322
9.26.2.17 l6 . . . 1322
9.26.2.18 l7 . . . 1322

9.26.2.19 o6_sp . . . 1322
9.26.2.20 o7 . . . 1322
9.26.2.21 psr . . . 1323
9.27 Context_Control_fp Struct Reference . . . 1323
9.27.1 Detailed Description . . . 1323
9.27.2 Member Data Documentation . . . 1323
9.27.2.1 f0_f1 . . . 1324
9.27.2.2 f10_f11 . . . 1324
9.27.2.3 f12_f13 . . . 1324
9.27.2.4 f14_f15 . . . 1324
9.27.2.5 f16_f17 . . . 1324
9.27.2.6 f18_f19 . . . 1325
9.27.2.7 f20_f21 . . . 1325
9.27.2.8 f22_f23 . . . 1325
9.27.2.9 f24_f25 . . . 1325
9.27.2.10 f26_f27 . . . 1325
9.27.2.11 f28_f29 . . . 1326
9.27.2.12 f2_f3 . . . 1326
9.27.2.13 f30_f31 . . . 1326
9.27.2.14 f4_f5 . . . 1326
9.27.2.15 f6_f7 . . . 1326
9.27.2.16 f8_f9 . . . 1327
9.27.2.17 fsr . . . 1327
9.28 CORE_barrier_Attributes Struct Reference . . . 1327
9.28.1 Detailed Description . . . 1327
9.28.2 Member Data Documentation . . . 1327
9.28.2.1 discipline . . . 1327
9.28.2.2 maximum_count . . . 1328
9.29 CORE_barrier_Control Struct Reference . . . 1328
9.29.1 Detailed Description . . . 1328
9.29.2 Member Data Documentation . . . 1328
9.29.2.1 Attributes . . . 1328
9.29.2.2 number_of_waiting_threads . . . 1329
9.29.2.3 Wait_queue . . . 1329
9.30 CORE_ceiling_mutex_Control Struct Reference . . . 1329
9.30.1 Detailed Description . . . 1329
9.31 CORE_message_queue_Buffer Struct Reference . . . 1330
9.31.1 Detailed Description . . . 1330
9.31.2 Member Data Documentation . . . 1330
9.31.2.1 buffer . . . 1330
9.32 CORE_message_queue_Control Struct Reference . . . 1330
9.32.1 Detailed Description . . . 1331

9.32.2 Member Data Documentation . . . 1331
9.32.2.1 free_message_buffers . . . 1331
9.32.2.2 Inactive_messages . . . 1331
9.32.2.3 maximum_message_size . . . 1332
9.32.2.4 maximum_pending_messages . . . 1332
9.32.2.5 message_buffers . . . 1332
9.32.2.6 number_of_pending_messages . . . 1332
9.32.2.7 Pending_messages . . . 1332
9.32.2.8 Wait_queue . . . 1333
9.33 CORE_mutex_Control Struct Reference . . . 1333
9.33.1 Detailed Description . . . 1333
9.33.2 Member Data Documentation . . . 1333
9.33.2.1 Wait_queue . . . 1333
9.34 CORE_recursive_mutex_Control Struct Reference . . . 1334
9.34.1 Detailed Description . . . 1334
9.35 CORE_semaphore_Control Struct Reference . . . 1334
9.35.1 Detailed Description . . . 1334
9.35.2 Member Data Documentation . . . 1334
9.35.2.1 count . . . 1335
9.35.2.2 Wait_queue . . . 1335
9.36 CPU_Exception_frame Struct Reference . . . 1335
9.36.1 Detailed Description . . . 1335
9.37 CPU_Interrupt_frame Struct Reference . . . 1335
9.37.1 Detailed Description . . . 1336
9.37.2 Member Data Documentation . . . 1336
9.37.2.1 g1 . . . 1336
9.37.2.2 g2 . . . 1337
9.37.2.3 g3 . . . 1337
9.37.2.4 g4 . . . 1337
9.37.2.5 g5 . . . 1337
9.37.2.6 g7 . . . 1337
9.37.2.7 i0 . . . 1338
9.37.2.8 i1 . . . 1338
9.37.2.9 i2 . . . 1338
9.37.2.10 i3 . . . 1338
9.37.2.11 i4 . . . 1338
9.37.2.12 i5 . . . 1339
9.37.2.13 i6_fp . . . 1339
9.37.2.14 i7 . . . 1339
9.37.2.15 npc . . . 1339
9.37.2.16 pc . . . 1339
9.37.2.17 psr . . . 1340

9.37.2.18 reserved_for_alignment . . . 1340
9.37.2.19 Stack_frame . . . 1340
9.37.2.20 tpc . . . 1340
9.37.2.21 y . . . 1340
9.38 CPU_Per_CPU_control Struct Reference . . . 1341
9.38.1 Detailed Description . . . 1341
9.39 CPU_Trap_table_entry Struct Reference . . . 1341
9.39.1 Detailed Description . . . 1341
9.39.2 Member Data Documentation . . . 1341
9.39.2.1 jmp_to_low_of_handler_plus_l4 . . . 1341
9.39.2.2 mov_psr_l0 . . . 1342
9.39.2.3 mov_vector_l3 . . . 1342
9.39.2.4 sethi_of_handler_to_l4 . . . 1342
9.40 Dual_ported_memory_Control Struct Reference . . . 1342
9.40.1 Detailed Description . . . 1342
9.40.2 Member Data Documentation . . . 1343
9.40.2.1 external_base . . . 1343
9.40.2.2 internal_base . . . 1343
9.40.2.3 length . . . 1343
9.40.2.4 Object . . . 1343
9.41 Event_Control Struct Reference . . . 1344
9.41.1 Detailed Description . . . 1344
9.42 Extension_Control Struct Reference . . . 1344
9.42.1 Detailed Description . . . 1344
9.43 ffclock_estimate Struct Reference . . . 1344
9.43.1 Detailed Description . . . 1345
9.44 Freechain_Control Struct Reference . . . 1345
9.44.1 Detailed Description . . . 1345
9.45 gptimer_regs Struct Reference . . . 1345
9.45.1 Detailed Description . . . 1345
9.46 gptimer_timer_regs Struct Reference . . . 1346
9.46.1 Detailed Description . . . 1346
9.47 grgpio_regs Struct Reference . . . 1346
9.47.1 Detailed Description . . . 1347
9.48 Heap_Area Struct Reference . . . 1347
9.48.1 Detailed Description . . . 1347
9.49 Heap_Block Struct Reference . . . 1347
9.49.1 Detailed Description . . . 1348
9.49.2 Member Data Documentation . . . 1348
9.49.2.1 next . . . 1348
9.49.2.2 prev . . . 1348
9.49.2.3 prev_size . . . 1348

9.49.2.4 size_and_flag . . . 1349
9.50 Heap_Control Struct Reference . . . 1349
9.50.1 Detailed Description . . . 1349
9.51 Heap_Error_context Struct Reference . . . 1350
9.51.1 Detailed Description . . . 1350
9.52 Heap_Information Struct Reference . . . 1350
9.52.1 Detailed Description . . . 1351
9.53 Heap_Information_block Struct Reference . . . 1351
9.53.1 Detailed Description . . . 1351
9.54 Heap_Statistics Struct Reference . . . 1351
9.54.1 Detailed Description . . . 1352
9.54.2 Member Data Documentation . . . 1352
9.54.2.1 free_size . . . 1353
9.54.2.2 lifetime_allocated . . . 1353
9.54.2.3 lifetime_freed . . . 1353
9.54.2.4 min_free_size . . . 1353
9.54.2.5 size . . . 1354
9.55 Internal_errors_Information Struct Reference . . . 1354
9.55.1 Detailed Description . . . 1354
9.55.2 Member Data Documentation . . . 1354
9.55.2.1 the_error . . . 1354
9.55.2.2 the_source . . . 1355
9.56 irqmp_regs Struct Reference . . . 1355
9.56.1 Detailed Description . . . 1355
9.57 irqmp_timestamp_regs Struct Reference . . . 1356
9.57.1 Detailed Description . . . 1356
9.58 ISR_lock_Context Struct Reference . . . 1356
9.58.1 Detailed Description . . . 1356
9.59 ISR_lock_Control Struct Reference . . . 1356
9.59.1 Detailed Description . . . 1357
9.60 l2c_regs Struct Reference . . . 1357
9.60.1 Detailed Description . . . 1358
9.61 load_context Struct Reference . . . 1358
9.61.1 Detailed Description . . . 1358
9.62 mctrl_regs Struct Reference . . . 1358
9.62.1 Detailed Description . . . 1358
9.63 Memory_Area Struct Reference . . . 1358
9.63.1 Detailed Description . . . 1359
9.64 Memory_Information Struct Reference . . . 1359
9.64.1 Detailed Description . . . 1359
9.65 Message_queue_Control Struct Reference . . . 1359
9.65.1 Detailed Description . . . 1360

9.65.2 Member Data Documentation . . . 1360
9.65.2.1 message_queue . . . 1360
9.65.2.2 Object . . . 1360
9.66 MP_packet_Prefix Struct Reference . . . 1360
9.66.1 Detailed Description . . . 1361
9.66.2 Member Data Documentation . . . 1361
9.66.2.1 id . . . 1361
9.66.2.2 length . . . 1361
9.66.2.3 return_code . . . 1362
9.66.2.4 source_priority . . . 1362
9.66.2.5 source_tid . . . 1362
9.66.2.6 the_class . . . 1362
9.66.2.7 timeout . . . 1362
9.66.2.8 to_convert . . . 1363
9.67 MRSP_Control Struct Reference . . . 1363
9.67.1 Detailed Description . . . 1363
9.68 Mutex_Control Struct Reference . . . 1363
9.68.1 Detailed Description . . . 1364
9.69 Mutex_recursive_Control Struct Reference . . . 1364
9.69.1 Detailed Description . . . 1364
9.70 ntp_fp Struct Reference . . . 1364
9.70.1 Detailed Description . . . 1364
9.71 ntptimeval Struct Reference . . . 1364
9.71.1 Detailed Description . . . 1365
9.72 Objects_Control Struct Reference . . . 1365
9.72.1 Detailed Description . . . 1365
9.72.2 Member Data Documentation . . . 1365
9.72.2.1 id . . . 1365
9.72.2.2 name . . . 1365
9.72.2.3 Node . . . 1366
9.73 Objects_Information Struct Reference . . . 1366
9.73.1 Detailed Description . . . 1367
9.73.2 Member Data Documentation . . . 1367
9.73.2.1 allocate . . . 1367
9.73.2.2 deallocate . . . 1367
9.73.2.3 inactive . . . 1368
9.73.2.4 Inactive . . . 1368
9.73.2.5 inactive_per_block . . . 1368
9.73.2.6 initial_objects . . . 1368
9.73.2.7 local_table . . . 1369
9.73.2.8 maximum_id . . . 1369
9.73.2.9 name_length . . . 1369

9.73.2.10 object_blocks . . . 1369
9.73.2.11 object_size . . . 1370
9.73.2.12 objects_per_block . . . 1370
9.74 Objects_Name Union Reference . . . 1370
9.74.1 Detailed Description . . . 1370
9.74.2 Member Data Documentation . . . 1370
9.74.2.1 name_p . . . 1371
9.74.2.2 name_u32 . . . 1371
9.75 Partition_Control Struct Reference . . . 1371
9.75.1 Detailed Description . . . 1372
9.76 Per_CPU_Control Struct Reference . . . 1372
9.76.1 Detailed Description . . . 1373
9.76.2 Member Data Documentation . . . 1373
9.76.2.1 ancestor . . . 1374
9.76.2.2 context . . . 1374
9.76.2.3 control . . . 1374
9.76.2.4 cpu_usage_timestamp . . . 1374
9.76.2.5 data . . . 1375
9.76.2.6 dispatch_necessary . . . 1375
9.76.2.7 executing . . . 1375
9.76.2.8 head . . . 1376
9.76.2.9 Header . . . 1376
9.76.2.10 heir . . . 1376
9.76.2.11 isr_dispatch_disable . . . 1377
9.76.2.12 isr_nest_level . . . 1377
9.76.2.13 Jobs . . . 1377
9.76.2.14 Lock . . . 1377
9.76.2.15 Lock_context . . . 1378
9.76.2.16 message . . . 1378
9.76.2.17 state . . . 1378
9.76.2.18 tail . . . 1378
9.76.2.19 Threads_in_need_for_help . . . 1379
9.76.2.20 ticks . . . 1379
9.77 Per_CPU_Control_envelope Struct Reference . . . 1379
9.77.1 Detailed Description . . . 1379
9.78 Per_CPU_Job Struct Reference . . . 1379
9.78.1 Detailed Description . . . 1380
9.78.2 Member Data Documentation . . . 1380
9.78.2.1 done . . . 1380
9.79 Per_CPU_Job_context Struct Reference . . . 1380
9.79.1 Detailed Description . . . 1381
9.80 Per_CPU_Stats Struct Reference . . . 1381

9.80.1 Detailed Description . . . 1381
9.81 POSIX_Spinlock_Control Struct Reference . . . 1381
9.81.1 Detailed Description . . . 1382
9.82 pps_fetch_args Struct Reference . . . 1382
9.82.1 Detailed Description . . . 1382
9.83 pps_fetch_ffc_args Struct Reference . . . 1382
9.83.1 Detailed Description . . . 1382
9.84 pps_info_ffc_t Struct Reference . . . 1383
9.84.1 Detailed Description . . . 1383
9.85 pps_info_t Struct Reference . . . 1383
9.85.1 Detailed Description . . . 1383
9.86 pps_kcbind_args Struct Reference . . . 1383
9.86.1 Detailed Description . . . 1384
9.87 pps_params_t Struct Reference . . . 1384
9.87.1 Detailed Description . . . 1384
9.88 pps_timeu Union Reference . . . 1384
9.88.1 Detailed Description . . . 1384
9.89 Priority_Actions Struct Reference . . . 1384
9.89.1 Detailed Description . . . 1385
9.90 Priority_Aggregation Struct Reference . . . 1385
9.90.1 Detailed Description . . . 1385
9.90.2 Member Data Documentation . . . 1386
9.90.2.1 Node . . . 1386
9.91 Priority_bit_map_Control Struct Reference . . . 1386
9.91.1 Detailed Description . . . 1386
9.91.2 Member Data Documentation . . . 1386
9.91.2.1 bit_map . . . 1387
9.92 Priority_bit_map_Information Struct Reference . . . 1387
9.92.1 Detailed Description . . . 1387
9.92.2 Member Data Documentation . . . 1387
9.92.2.1 block_major . . . 1387
9.92.2.2 block_minor . . . 1388
9.92.2.3 minor . . . 1388
9.92.2.4 ready_major . . . 1388
9.92.2.5 ready_minor . . . 1388
9.93 Priority_Node Struct Reference . . . 1388
9.93.1 Detailed Description . . . 1389
9.94 Rate_monotonic_Control Struct Reference . . . 1389
9.94.1 Detailed Description . . . 1389
9.94.2 Member Data Documentation . . . 1390
9.94.2.1 cpu_usage_period_initiated . . . 1390
9.94.2.2 latest_deadline . . . 1390

9.94.2.3 next_length . . . 1390
9.94.2.4 Object . . . 1390
9.94.2.5 owner . . . 1391
9.94.2.6 postponed_jobs . . . 1391
9.94.2.7 state . . . 1391
9.94.2.8 Statistics . . . 1391
9.94.2.9 time_period_initiated . . . 1391
9.94.2.10 Timer . . . 1392
9.95 Rate_monotonic_Statistics Struct Reference . . . 1392
9.95.1 Detailed Description . . . 1392
9.95.2 Member Data Documentation . . . 1392
9.95.2.1 count . . . 1392
9.95.2.2 max_cpu_time . . . 1393
9.95.2.3 max_wall_time . . . 1393
9.95.2.4 min_cpu_time . . . 1393
9.95.2.5 min_wall_time . . . 1393
9.95.2.6 missed_count . . . 1393
9.95.2.7 total_cpu_time . . . 1394
9.95.2.8 total_wall_time . . . 1394
9.96 RBTree_Node Struct Reference . . . 1394
9.96.1 Detailed Description . . . 1394
9.97 Region_Control Struct Reference . . . 1394
9.97.1 Detailed Description . . . 1395
9.98 rtems_api_configuration_table Struct Reference . . . 1395
9.98.1 Detailed Description . . . 1396
9.98.2 Member Data Documentation . . . 1396
9.98.2.1 maximum_barriers . . . 1396
9.98.2.2 maximum_message_queues . . . 1396
9.98.2.3 maximum_partitions . . . 1396
9.98.2.4 maximum_periods . . . 1397
9.98.2.5 maximum_ports . . . 1397
9.98.2.6 maximum_regions . . . 1397
9.98.2.7 maximum_semaphores . . . 1397
9.98.2.8 maximum_tasks . . . 1398
9.98.2.9 maximum_timers . . . 1398
9.98.2.10 number_of_initialization_tasks . . . 1398
9.98.2.11 User_initialization_tasks_table . . . 1398
9.99 RTEMS_API_Control Struct Reference . . . 1399
9.99.1 Detailed Description . . . 1399
9.99.2 Member Data Documentation . . . 1399
9.99.2.1 Event . . . 1399
9.99.2.2 Signal . . . 1399

9.99.2.3 System_event . . . 1400
9.100 rtems_assert_context Struct Reference . . . 1400
9.100.1 Detailed Description . . . 1400
9.100.2 Member Data Documentation . . . 1400
9.100.2.1 failed_expression . . . 1400
9.100.2.2 file . . . 1401
9.100.2.3 function . . . 1401
9.100.2.4 line . . . 1401
9.101 rtems_assoc_32_pair Struct Reference . . . 1401
9.101.1 Detailed Description . . . 1401
9.102 rtems_assoc_t Struct Reference . . . 1402
9.102.1 Detailed Description . . . 1402
9.103 rtems_binary_semaphore Struct Reference . . . 1402
9.103.1 Detailed Description . . . 1402
9.104 rtems_driver_address_table Struct Reference . . . 1402
9.104.1 Detailed Description . . . 1403
9.104.2 Member Data Documentation . . . 1403
9.104.2.1 close_entry . . . 1403
9.104.2.2 control_entry . . . 1403
9.104.2.3 initialization_entry . . . 1403
9.104.2.4 open_entry . . . 1404
9.104.2.5 read_entry . . . 1404
9.104.2.6 write_entry . . . 1404
9.105 rtems_filesystem_eval_path_context_t Struct Reference . . . 1404
9.105.1 Detailed Description . . . 1405
9.105.2 Member Data Documentation . . . 1405
9.105.2.1 currentloc . . . 1405
9.105.2.2 flags . . . 1405
9.105.2.3 path . . . 1406
9.105.2.4 pathlen . . . 1406
9.105.2.5 recursionlevel . . . 1406
9.105.2.6 rootloc . . . 1406
9.105.2.7 startloc . . . 1406
9.105.2.8 token . . . 1407
9.105.2.9 tokenlen . . . 1407
9.106 rtems_filesystem_global_location_t Struct Reference . . . 1407
9.106.1 Detailed Description . . . 1407
9.106.2 Member Data Documentation . . . 1408
9.106.2.1 deferred_released_count . . . 1408
9.106.2.2 deferred_released_next . . . 1408
9.107 rtems_filesystem_limits_and_options_t Struct Reference . . . 1408
9.107.1 Detailed Description . . . 1409

9.108 rtems_filesystem_location_info_tt Struct Reference . . . 1409
9.108.1 Detailed Description . . . 1409
9.109 rtems_filesystem_mount_configuration Struct Reference . . . 1409
9.109.1 Detailed Description . . . 1410
9.110 rtems_filesystem_mount_table_entry_tt Struct Reference . . . 1410
9.110.1 Detailed Description . . . 1410
9.110.2 Member Data Documentation . . . 1410
9.110.2.1 unmount_task . . . 1411
9.111 rtems_filesystem_table_t Struct Reference . . . 1411
9.111.1 Detailed Description . . . 1411
9.112 rtems_initialization_tasks_table Struct Reference . . . 1411
9.112.1 Detailed Description . . . 1412
9.112.2 Member Data Documentation . . . 1412
9.112.2.1 argument . . . 1412
9.112.2.2 attribute_set . . . 1412
9.112.2.3 entry_point . . . 1413
9.112.2.4 initial_priority . . . 1413
9.112.2.5 mode_set . . . 1413
9.112.2.6 name . . . 1413
9.112.2.7 stack_size . . . 1414
9.113 rtems_interrupt_server_action Struct Reference . . . 1414
9.113.1 Detailed Description . . . 1414
9.114 rtems_interrupt_server_config Struct Reference . . . 1414
9.114.1 Detailed Description . . . 1415
9.114.2 Member Data Documentation . . . 1415
9.114.2.1 destroy . . . 1415
9.114.2.2 storage_area . . . 1415
9.114.2.3 storage_size . . . 1416
9.115 rtems_interrupt_server_control Struct Reference . . . 1416
9.115.1 Detailed Description . . . 1416
9.116 rtems_interrupt_server_entry Struct Reference . . . 1417
9.116.1 Detailed Description . . . 1417
9.117 rtems_interrupt_server_request Struct Reference . . . 1417
9.117.1 Detailed Description . . . 1418
9.118 rtems_libio_ioctl_args_t Struct Reference . . . 1418
9.118.1 Detailed Description . . . 1418
9.119 rtems_libio_open_close_args_t Struct Reference . . . 1418
9.119.1 Detailed Description . . . 1419
9.120 rtems_libio_rw_args_t Struct Reference . . . 1419
9.120.1 Detailed Description . . . 1419
9.121 rtems_libio_tt Struct Reference . . . 1419
9.121.1 Detailed Description . . . 1420

9.122 rtems_message_queue_config Struct Reference . . . 1420
9.122.1 Detailed Description . . . 1421
9.122.2 Member Data Documentation . . . 1421
9.122.2.1 storage_area . . . 1421
9.122.2.2 storage_free . . . 1421
9.123 rtems_object_api_class_information Struct Reference . . . 1421
9.123.1 Detailed Description . . . 1422
9.123.2 Member Data Documentation . . . 1422
9.123.2.1 auto_extend . . . 1422
9.123.2.2 maximum . . . 1422
9.123.2.3 maximum_id . . . 1423
9.123.2.4 minimum_id . . . 1423
9.123.2.5 unallocated . . . 1423
9.124 rtems_printer Struct Reference . . . 1423
9.124.1 Detailed Description . . . 1424
9.125 rtems_printer_task_context Struct Reference . . . 1424
9.125.1 Detailed Description . . . 1424
9.126 rtems_rate_monotonic_period_statistics Struct Reference . . . 1424
9.126.1 Detailed Description . . . 1425
9.126.2 Member Data Documentation . . . 1425
9.126.2.1 count . . . 1425
9.126.2.2 max_cpu_time . . . 1425
9.126.2.3 max_wall_time . . . 1426
9.126.2.4 min_cpu_time . . . 1426
9.126.2.5 min_wall_time . . . 1426
9.126.2.6 missed_count . . . 1426
9.126.2.7 total_cpu_time . . . 1427
9.126.2.8 total_wall_time . . . 1427
9.127 rtems_rate_monotonic_period_status Struct Reference . . . 1427
9.127.1 Detailed Description . . . 1427
9.127.2 Member Data Documentation . . . 1428
9.127.2.1 executed_since_last_period . . . 1428
9.127.2.2 owner . . . 1428
9.127.2.3 postponed_jobs_count . . . 1428
9.127.2.4 since_last_period . . . 1428
9.127.2.5 state . . . 1429
9.128 rtems_resource_posix_api Struct Reference . . . 1429
9.128.1 Detailed Description . . . 1429
9.129 rtems_resource_rtems_api Struct Reference . . . 1429
9.129.1 Detailed Description . . . 1430
9.130 rtems_resource_snapshot Struct Reference . . . 1430
9.130.1 Detailed Description . . . 1430

9.131 rtems_sysinit_item Struct Reference . . . 1430
9.131.1 Detailed Description . . . 1430
9.132 rtems_task_config Struct Reference . . . 1431
9.132.1 Detailed Description . . . 1431
9.132.2 Member Data Documentation . . . 1431
9.132.2.1 maximum_thread_local_storage_size . . . 1431
9.132.2.2 storage_area . . . 1432
9.132.2.3 storage_free . . . 1432
9.132.2.4 storage_size . . . 1432
9.133 rtems_termios_callbacks Struct Reference . . . 1432
9.133.1 Detailed Description . . . 1433
9.134 rtems_termios_device_context Struct Reference . . . 1433
9.134.1 Detailed Description . . . 1433
9.135 rtems_termios_device_flow Struct Reference . . . 1433
9.135.1 Detailed Description . . . 1434
9.135.2 Member Data Documentation . . . 1434
9.135.2.1 start_remote_tx . . . 1434
9.135.2.2 stop_remote_tx . . . 1434
9.136 rtems_termios_device_handler Struct Reference . . . 1435
9.136.1 Detailed Description . . . 1435
9.136.2 Member Data Documentation . . . 1435
9.136.2.1 first_open . . . 1435
9.136.2.2 ioctl . . . 1436
9.136.2.3 last_close . . . 1436
9.136.2.4 poll_read . . . 1437
9.136.2.5 set_attributes . . . 1437
9.136.2.6 write . . . 1438
9.137 rtems_termios_device_node Struct Reference . . . 1438
9.137.1 Detailed Description . . . 1438
9.138 rtems_termios_linesw Struct Reference . . . 1439
9.138.1 Detailed Description . . . 1439
9.139 rtems_termios_rawbuf Struct Reference . . . 1439
9.139.1 Detailed Description . . . 1439
9.140 rtems_termios_tty Struct Reference . . . 1439
9.140.1 Detailed Description . . . 1440
9.141 rtems_test_parallel_context Struct Reference . . . 1441
9.141.1 Detailed Description . . . 1441
9.142 rtems_test_parallel_job Struct Reference . . . 1441
9.142.1 Detailed Description . . . 1442
9.142.2 Member Data Documentation . . . 1442
9.142.2.1 body . . . 1442
9.142.2.2 cascade . . . 1442

9.142.2.3 fini . . . 1442
9.142.2.4 init . . . 1443
9.143 rtems_time_of_day Struct Reference . . . 1443
9.143.1 Detailed Description . . . 1444
9.144 rtems_timecounter_simple Struct Reference . . . 1444
9.144.1 Detailed Description . . . 1444
9.145 rtems_timer_information Struct Reference . . . 1445
9.145.1 Detailed Description . . . 1445
9.145.2 Member Data Documentation . . . 1445
9.145.2.1 initial . . . 1445
9.145.2.2 start_time . . . 1445
9.145.2.3 stop_time . . . 1446
9.145.2.4 the_class . . . 1446
9.146 RtemsEventReqSendReceive_Context Struct Reference . . . 1446
9.146.1 Detailed Description . . . 1447
9.147 RtemsMessageReqConstructErrors_Context Struct Reference . . . 1448
9.147.1 Detailed Description . . . 1448
9.148 RtemsPartReqCreate_Context Struct Reference . . . 1448
9.148.1 Detailed Description . . . 1449
9.149 RtemsPartReqDelete_Context Struct Reference . . . 1449
9.149.1 Detailed Description . . . 1449
9.150 RtemsPartReqGetBuffer_Context Struct Reference . . . 1449
9.150.1 Detailed Description . . . 1450
9.151 RtemsPartReqReturnBuffer_Context Struct Reference . . . 1450
9.151.1 Detailed Description . . . 1450
9.152 RtemsReqIdent_Context Struct Reference . . . 1450
9.152.1 Detailed Description . . . 1451
9.153 RtemsReqIdentLocal_Context Struct Reference . . . 1451
9.153.1 Detailed Description . . . 1452
9.154 RtemsTaskReqConstructErrors_Context Struct Reference . . . 1452
9.154.1 Detailed Description . . . 1452
9.155 RtemsTaskReqIdent_Context Struct Reference . . . 1452
9.155.1 Detailed Description . . . 1453
9.156 Scheduler_Assignment Struct Reference . . . 1453
9.156.1 Detailed Description . . . 1453
9.156.2 Member Data Documentation . . . 1453
9.156.2.1 attributes . . . 1454
9.157 Scheduler_Context Struct Reference . . . 1454
9.157.1 Detailed Description . . . 1454
9.157.2 Member Data Documentation . . . 1454
9.157.2.1 Processors . . . 1455
9.158 Scheduler_EDF_Context Struct Reference . . . 1455

9.158.1 Detailed Description . . . 1455
9.158.2 Member Data Documentation . . . 1455
9.158.2.1 Ready . . . 1455
9.159 Scheduler_EDF_Node Struct Reference . . . 1455
9.159.1 Detailed Description . . . 1456
9.159.2 Member Data Documentation . . . 1456
9.159.2.1 Node . . . 1456
9.160 Scheduler_EDF_SMP_Context Struct Reference . . . 1456
9.160.1 Detailed Description . . . 1456
9.160.2 Member Data Documentation . . . 1457
9.160.2.1 Ready . . . 1457
9.161 Scheduler_EDF_SMP_Node Struct Reference . . . 1457
9.161.1 Detailed Description . . . 1457
9.161.2 Member Data Documentation . . . 1457
9.161.2.1 ready_queue_index . . . 1458
9.162 Scheduler_EDF_SMP_Ready_queue Struct Reference . . . 1458
9.162.1 Detailed Description . . . 1458
9.163 Scheduler_Node Struct Reference . . . 1458
9.163.1 Detailed Description . . . 1459
9.163.2 Member Data Documentation . . . 1460
9.163.2.1 idle . . . 1460
9.163.2.2 next . . . 1460
9.163.2.3 Node . . . 1460
9.163.2.4 Priority . . . 1460
9.163.2.5 user . . . 1461
9.163.2.6 value . . . 1461
9.164 Scheduler_Operations Struct Reference . . . 1461
9.164.1 Detailed Description . . . 1462
9.164.2 Member Data Documentation . . . 1462
9.164.2.1 add_processor . . . 1462
9.164.2.2 ask_for_help . . . 1463
9.164.2.3 block . . . 1463
9.164.2.4 cancel_job . . . 1464
9.164.2.5 initialize . . . 1464
9.164.2.6 map_priority . . . 1464
9.164.2.7 node_destroy . . . 1464
9.164.2.8 node_initialize . . . 1465
9.164.2.9 pin . . . 1465
9.164.2.10 reconsider_help_request . . . 1465
9.164.2.11 release_job . . . 1466
9.164.2.12 remove_processor . . . 1466
9.164.2.13 schedule . . . 1466

9.164.2.14 set_affinity . . . 1467
9.164.2.15 start_idle . . . 1467
9.164.2.16 tick . . . 1467
9.164.2.17 unblock . . . 1467
9.164.2.18 unmap_priority . . . 1468
9.164.2.19 unpin . . . 1468
9.164.2.20 update_priority . . . 1468
9.164.2.21 withdraw_node . . . 1468
9.164.2.22 yield . . . 1469
9.165 Scheduler_priority_Context Struct Reference . . . 1469
9.165.1 Detailed Description . . . 1469
9.166 Scheduler_priority_Node Struct Reference . . . 1470
9.166.1 Detailed Description . . . 1470
9.167 Scheduler_priority_Ready_queue Struct Reference . . . 1470
9.167.1 Detailed Description . . . 1470
9.167.2 Member Data Documentation . . . 1470
9.167.2.1 Priority_map . . . 1471
9.167.2.2 ready_chain . . . 1471
9.168 Scheduler_Processor_removal_context Struct Reference . . . 1471
9.168.1 Detailed Description . . . 1471
9.169 Scheduler_simple_Context Struct Reference . . . 1471
9.169.1 Detailed Description . . . 1472
9.170 Scheduler_SMP_Context Struct Reference . . . 1472
9.170.1 Detailed Description . . . 1472
9.170.2 Member Data Documentation . . . 1472
9.170.2.1 Idle_threads . . . 1473
9.171 Scheduler_SMP_Node Struct Reference . . . 1473
9.171.1 Detailed Description . . . 1473
9.172 Sem_Control Struct Reference . . . 1473
9.172.1 Detailed Description . . . 1474
9.173 Semaphore_Control Struct Reference . . . 1474
9.173.1 Detailed Description . . . 1474
9.173.2 Member Data Documentation . . . 1474
9.173.2.1 Core_control . . . 1474
9.173.2.2 Mutex . . . 1475
9.173.2.3 Object . . . 1475
9.173.2.4 Semaphore . . . 1475
9.174 SHA256Context Struct Reference . . . 1475
9.174.1 Detailed Description . . . 1475
9.175 SMP_barrier_Control Struct Reference . . . 1476
9.175.1 Detailed Description . . . 1476
9.176 SMP_barrier_State Struct Reference . . . 1476

9.176.1 Detailed Description . . . 1476
9.177 SMP_lock_Context Struct Reference . . . 1477
9.177.1 Detailed Description . . . 1477
9.178 SMP_lock_Control Struct Reference . . . 1477
9.178.1 Detailed Description . . . 1477
9.179 SMP_MCS_lock_Context Struct Reference . . . 1477
9.179.1 Detailed Description . . . 1478
9.179.2 Member Data Documentation . . . 1478
9.179.2.1 locked . . . 1478
9.179.2.2 normal . . . 1478
9.180 SMP_MCS_lock_Control Struct Reference . . . 1479
9.180.1 Detailed Description . . . 1479
9.180.2 Member Data Documentation . . . 1479
9.180.2.1 normal . . . 1479
9.180.2.2 queue . . . 1479
9.181 SMP_Multicast_jobs Struct Reference . . . 1480
9.181.1 Detailed Description . . . 1480
9.182 SMP_sequence_lock_Control Struct Reference . . . 1480
9.182.1 Detailed Description . . . 1480
9.182.2 Member Data Documentation . . . 1480
9.182.2.1 sequence . . . 1481
9.183 SMP_ticket_lock_Control Struct Reference . . . 1481
9.183.1 Detailed Description . . . 1481
9.184 SPARC_Counter Struct Reference . . . 1481
9.184.1 Detailed Description . . . 1482
9.185 SPARC_Minimum_stack_frame Struct Reference . . . 1482
9.185.1 Detailed Description . . . 1482
9.185.2 Member Data Documentation . . . 1483
9.185.2.1 i0 . . . 1483
9.185.2.2 i1 . . . 1483
9.185.2.3 i2 . . . 1483
9.185.2.4 i3 . . . 1483
9.185.2.5 i4 . . . 1483
9.185.2.6 i5 . . . 1484
9.185.2.7 i6_fp . . . 1484
9.185.2.8 i7 . . . 1484
9.185.2.9 l0 . . . 1484
9.185.2.10 l1 . . . 1484
9.185.2.11 l2 . . . 1485
9.185.2.12 l3 . . . 1485
9.185.2.13 l4 . . . 1485
9.185.2.14 l5 . . . 1485

9.185.2.15 l6 . . . 1485
9.185.2.16 l7 . . . 1486
9.185.2.17 pad0 . . . 1486
9.185.2.18 saved_arg0 . . . 1486
9.185.2.19 saved_arg1 . . . 1486
9.185.2.20 saved_arg2 . . . 1486
9.185.2.21 saved_arg3 . . . 1487
9.185.2.22 saved_arg4 . . . 1487
9.185.2.23 saved_arg5 . . . 1487
9.185.2.24 structure_return_address . . . 1487
9.186 Stack_Control Struct Reference . . . 1487
9.186.1 Detailed Description . . . 1488
9.186.2 Member Data Documentation . . . 1488
9.186.2.1 area . . . 1488
9.186.2.2 size . . . 1488
9.187 statvfs Struct Reference . . . 1488
9.187.1 Detailed Description . . . 1489
9.187.2 Member Data Documentation . . . 1489
9.187.2.1 f_bavail . . . 1489
9.187.2.2 f_bfree . . . 1489
9.187.2.3 f_blocks . . . 1489
9.187.2.4 f_bsize . . . 1489
9.187.2.5 f_favail . . . 1490
9.187.2.6 f_ffree . . . 1490
9.187.2.7 f_files . . . 1490
9.187.2.8 f_flag . . . 1490
9.187.2.9 f_frsize . . . 1490
9.187.2.10 f_fsid . . . 1491
9.187.2.11 f_namemax . . . 1491
9.188 T_case_context Struct Reference . . . 1491
9.188.1 Detailed Description . . . 1491
9.189 T_check_context Struct Reference . . . 1491
9.189.1 Detailed Description . . . 1492
9.190 T_check_context_msg Struct Reference . . . 1492
9.190.1 Detailed Description . . . 1492
9.191 T_config Struct Reference . . . 1492
9.191.1 Detailed Description . . . 1492
9.192 T_context Struct Reference . . . 1493
9.192.1 Detailed Description . . . 1493
9.193 T_destructor Struct Reference . . . 1493
9.193.1 Detailed Description . . . 1493
9.194 T_fixture Struct Reference . . . 1494

9.194.1 Detailed Description . . . 1494
9.195 T_fixture_node Struct Reference . . . 1494
9.195.1 Detailed Description . . . 1494
9.196 T_interrupt_clock_time Struct Reference . . . 1494
9.196.1 Detailed Description . . . 1495
9.197 T_interrupt_context Struct Reference . . . 1495
9.197.1 Detailed Description . . . 1495
9.198 T_interrupt_test_config Struct Reference . . . 1495
9.198.1 Detailed Description . . . 1496
9.199 T_measure_runtime_config Struct Reference . . . 1496
9.199.1 Detailed Description . . . 1496
9.200 T_measure_runtime_context Struct Reference . . . 1496
9.200.1 Detailed Description . . . 1496
9.201 T_measure_runtime_request Struct Reference . . . 1497
9.201.1 Detailed Description . . . 1497
9.202 T_putchar_string_context Struct Reference . . . 1497
9.202.1 Detailed Description . . . 1497
9.203 T_report_hash_sha256_context Struct Reference . . . 1497
9.203.1 Detailed Description . . . 1498
9.204 T_thread_switch_context Struct Reference . . . 1498
9.204.1 Detailed Description . . . 1498
9.205 T_thread_switch_event Struct Reference . . . 1498
9.205.1 Detailed Description . . . 1498
9.206 T_thread_switch_log Struct Reference . . . 1499
9.206.1 Detailed Description . . . 1499
9.207 T_thread_switch_log_10 Struct Reference . . . 1499
9.207.1 Detailed Description . . . 1499
9.208 T_thread_switch_log_2 Struct Reference . . . 1499
9.208.1 Detailed Description . . . 1499
9.209 T_thread_switch_log_4 Struct Reference . . . 1500
9.209.1 Detailed Description . . . 1500
9.210 Thread_Action Struct Reference . . . 1500
9.210.1 Detailed Description . . . 1500
9.211 Thread_Action_control Struct Reference . . . 1501
9.211.1 Detailed Description . . . 1501
9.212 Thread_Capture_control Struct Reference . . . 1501
9.212.1 Detailed Description . . . 1501
9.213 Thread_Close_context Struct Reference . . . 1501
9.213.1 Detailed Description . . . 1502
9.214 Thread_Configuration Struct Reference . . . 1502
9.214.1 Detailed Description . . . 1502
9.214.2 Member Data Documentation . . . 1503

9.214.2.1 stack_free . . . 1503
9.215 Thread_Control_add_on Struct Reference . . . 1503
9.215.1 Detailed Description . . . 1503
9.216 Thread_Entry_idle Struct Reference . . . 1503
9.216.1 Detailed Description . . . 1504
9.217 Thread_Entry_information Struct Reference . . . 1504
9.217.1 Detailed Description . . . 1504
9.217.2 Member Data Documentation . . . 1504
9.217.2.1 adaptor . . . 1504
9.218 Thread_Entry_numeric Struct Reference . . . 1505
9.218.1 Detailed Description . . . 1505
9.219 Thread_Entry_pointer Struct Reference . . . 1505
9.219.1 Detailed Description . . . 1506
9.220 Thread_Information Struct Reference . . . 1506
9.220.1 Detailed Description . . . 1506
9.220.2 Member Data Documentation . . . 1506
9.220.2.1 Free . . . 1506
9.220.2.2 initial . . . 1507
9.221 Thread_Join_context Struct Reference . . . 1507
9.221.1 Detailed Description . . . 1507
9.222 Thread_Keys_information Struct Reference . . . 1507
9.222.1 Detailed Description . . . 1507
9.223 Thread_Life_control Struct Reference . . . 1508
9.223.1 Detailed Description . . . 1508
9.223.2 Member Data Documentation . . . 1508
9.223.2.1 exit_value . . . 1508
9.224 Thread_Proxy_control Struct Reference . . . 1509
9.224.1 Detailed Description . . . 1509
9.224.2 Member Data Documentation . . . 1509
9.224.2.1 current_state . . . 1509
9.224.2.2 Join_queue . . . 1509
9.224.2.3 Object . . . 1510
9.224.2.4 Timer . . . 1510
9.224.2.5 Wait . . . 1510
9.225 Thread_queue_Context Struct Reference . . . 1510
9.225.1 Detailed Description . . . 1511
9.225.2 Member Data Documentation . . . 1511
9.225.2.1 deadlock_callout . . . 1512
9.225.2.2 enqueue_callout . . . 1512
9.225.2.3 Path . . . 1512
9.225.2.4 Timeout . . . 1512
9.225.2.5 update . . . 1513

9.226 Thread_queue_Control Struct Reference . . . 1513
9.226.1 Detailed Description . . . 1513
9.227 Thread_queue_Gate Struct Reference . . . 1513
9.227.1 Detailed Description . . . 1514
9.228 Thread_queue_Link Struct Reference . . . 1514
9.228.1 Detailed Description . . . 1514
9.229 Thread_queue_Links Struct Reference . . . 1515
9.229.1 Detailed Description . . . 1515
9.230 Thread_queue_Lock_context Struct Reference . . . 1515
9.230.1 Detailed Description . . . 1515
9.230.2 Member Data Documentation . . . 1515
9.230.2.1 Gate . . . 1516
9.231 Thread_queue_Object Struct Reference . . . 1516
9.231.1 Detailed Description . . . 1516
9.232 Thread_queue_Operations Struct Reference . . . 1516
9.232.1 Detailed Description . . . 1517
9.232.2 Member Data Documentation . . . 1517
9.232.2.1 enqueue . . . 1517
9.232.2.2 extract . . . 1517
9.233 Thread_queue_Priority_queue Struct Reference . . . 1518
9.233.1 Detailed Description . . . 1518
9.233.2 Member Data Documentation . . . 1518
9.233.2.1 Node . . . 1518
9.234 Thread_queue_Queue Struct Reference . . . 1519
9.234.1 Detailed Description . . . 1519
9.234.2 Member Data Documentation . . . 1519
9.234.2.1 heads . . . 1519
9.234.2.2 Lock . . . 1519
9.235 Thread_queue_Syslock_queue Struct Reference . . . 1520
9.235.1 Detailed Description . . . 1520
9.236 Thread_Scheduler_control Struct Reference . . . 1520
9.236.1 Detailed Description . . . 1521
9.236.2 Member Data Documentation . . . 1521
9.236.2.1 Help_node . . . 1521
9.236.2.2 helping_nodes . . . 1521
9.236.2.3 nodes . . . 1521
9.236.2.4 pin_level . . . 1522
9.236.2.5 requests . . . 1522
9.236.2.6 Scheduler_nodes . . . 1522
9.236.2.7 Wait_nodes . . . 1523
9.237 Thread_Start_information Struct Reference . . . 1523
9.237.1 Detailed Description . . . 1523

9.237.2 Member Data Documentation . . . 1523
9.237.2.1 budget_algorithm . . . 1524
9.237.2.2 budget_callout . . . 1524
9.237.2.3 Entry . . . 1524
9.237.2.4 initial_priority . . . 1524
9.237.2.5 Initial_stack . . . 1524
9.237.2.6 is_preemptible . . . 1525
9.237.2.7 isr_level . . . 1525
9.237.2.8 tls_area . . . 1525
9.238 Thread_Timer_information Struct Reference . . . 1525
9.238.1 Detailed Description . . . 1525
9.239 Thread_Wait_information Struct Reference . . . 1526
9.239.1 Detailed Description . . . 1526
9.239.2 Member Data Documentation . . . 1526
9.239.2.1 count . . . 1527
9.239.2.2 Lock . . . 1527
9.239.2.3 operations . . . 1527
9.239.2.4 option . . . 1528
9.239.2.5 queue . . . 1528
9.239.2.6 return_argument . . . 1528
9.239.2.7 return_argument_second . . . 1528
9.239.2.8 return_code . . . 1529
9.239.2.9 Tranquilizer . . . 1529
9.240 Thread_Wait_information_Object_argument_type Union Reference . . . 1529
9.240.1 Detailed Description . . . 1529
9.241 Thread_Zombie_control Struct Reference . . . 1530
9.241.1 Detailed Description . . . 1530
9.242 timecounter Struct Reference . . . 1530
9.242.1 Detailed Description . . . 1530
9.243 timehands Struct Reference . . . 1530
9.243.1 Detailed Description . . . 1531
9.244 Timer_Control Struct Reference . . . 1531
9.244.1 Detailed Description . . . 1531
9.244.2 Member Data Documentation . . . 1531
9.244.2.1 initial . . . 1531
9.244.2.2 Object . . . 1532
9.244.2.3 routine . . . 1532
9.244.2.4 start_time . . . 1532
9.244.2.5 stop_time . . . 1532
9.244.2.6 the_class . . . 1532
9.244.2.7 Ticker . . . 1533
9.244.2.8 user_data . . . 1533

9.245 Timer_server_Control Struct Reference . . . 1533
9.245.1 Detailed Description . . . 1533
9.246 timex Struct Reference . . . 1533
9.246.1 Detailed Description . . . 1534
9.247 TLS_Dynamic_thread_vector Struct Reference . . . 1534
9.247.1 Detailed Description . . . 1534
9.248 TLS_Index Struct Reference . . . 1534
9.248.1 Detailed Description . . . 1534
9.249 TLS_Thread_control_block Struct Reference . . . 1534
9.249.1 Detailed Description . . . 1535
9.250 TOD_Control Struct Reference . . . 1535
9.250.1 Detailed Description . . . 1535
9.250.2 Member Data Documentation . . . 1535
9.250.2.1 is_set . . . 1535
9.251 TOD_Hook Struct Reference . . . 1536
9.251.1 Detailed Description . . . 1536
9.251.2 Member Data Documentation . . . 1536
9.251.2.1 handler . . . 1536
9.251.2.2 Node . . . 1536
9.252 ttywakeup Struct Reference . . . 1537
9.252.1 Detailed Description . . . 1537
9.253 User_extensions_Control Struct Reference . . . 1537
9.253.1 Detailed Description . . . 1537
9.254 User_extensions_Fatal_context Struct Reference . . . 1537
9.254.1 Detailed Description . . . 1538
9.255 User_extensions_Iterator Struct Reference . . . 1538
9.255.1 Detailed Description . . . 1538
9.256 User_extensions_List Struct Reference . . . 1538
9.256.1 Detailed Description . . . 1539
9.257 User_extensions_Switch_control Struct Reference . . . 1539
9.257.1 Detailed Description . . . 1539
9.258 User_extensions_Table Struct Reference . . . 1539
9.258.1 Detailed Description . . . 1540
9.259 User_extensions_Thread_create_context Struct Reference . . . 1540
9.259.1 Detailed Description . . . 1540
9.260 Watchdog_Control Struct Reference . . . 1540
9.260.1 Detailed Description . . . 1541
9.261 Watchdog_Header Struct Reference . . . 1541
9.261.1 Detailed Description . . . 1541

**10 File Documentation** **1543**

10.1 bsps/include/bsp/bootcard.h File Reference . . . 1543

10.2 bsps/include/bsp/default-initial-extension.h File Reference . . . 1543
10.2.1 Detailed Description . . . 1544
10.3 cpukit/include/rtems/fatal.h File Reference . . . 1544
10.3.1 Detailed Description . . . 1545
10.4 bsps/include/bsp/irq-generic.h File Reference . . . 1545
10.4.1 Detailed Description . . . 1546
10.5 bsps/include/grlib/ambapp.h File Reference . . . 1546
10.6 bsps/include/grlib/ambapp_ids.h File Reference . . . 1548
10.6.1 Detailed Description . . . 1553
10.7 bsps/include/grlib/apbuart.h File Reference . . . 1553
10.8 bsps/include/grlib/grlib.h File Reference . . . 1554
10.8.1 Detailed Description . . . 1554
10.9 bsps/shared/dev/clock/clockimpl.h File Reference . . . 1555
10.9.1 Detailed Description . . . 1555
10.9.2 Function Documentation . . . 1555
10.9.2.1 Clock_isr() . . . 1555
10.10 bsps/shared/irq/irq-generic.c File Reference . . . 1556
10.10.1 Detailed Description . . . 1557
10.11 bsps/shared/irq/irq-lock.c File Reference . . . 1557
10.11.1 Detailed Description . . . 1557
10.12 bsps/shared/start/bootcard.c File Reference . . . 1557
10.13 bsps/shared/start/mallocinitone.c File Reference . . . 1558
10.13.1 Detailed Description . . . 1558
10.14 bsps/shared/start/wkspaceinitone.c File Reference . . . 1559
10.14.1 Detailed Description . . . 1559
10.15 bsps/sparc/leon3/include/amba.h File Reference . . . 1559
10.16 bsps/sparc/leon3/include/bsp.h File Reference . . . 1559
10.16.1 Detailed Description . . . 1561
10.17 cpukit/include/rtems/confdefs/bsp.h File Reference . . . 1561
10.17.1 Detailed Description . . . 1561
10.18 bsps/sparc/leon3/include/bsp/irq.h File Reference . . . 1561
10.18.1 Detailed Description . . . 1561
10.19 bsps/sparc/leon3/include/leon.h File Reference . . . 1562
10.19.1 Detailed Description . . . 1564
10.19.2 Macro Definition Documentation . . . 1564
10.19.2.1 LEON_Clear_interrupt . . . 1564
10.19.2.2 LEON_Cpu_Disable_interrupt . . . 1565
10.19.2.3 LEON_Cpu_Mask_interrupt . . . 1565
10.19.2.4 LEON_Cpu_Restore_interrupt . . . 1565
10.19.2.5 LEON_Cpu_Unmask_interrupt . . . 1566
10.19.2.6 LEON_Disable_interrupt_broadcast . . . 1566
10.19.2.7 LEON_Enable_interrupt_broadcast . . . 1566

10.19.2.8 LEON_Force_interrupt . . . 1566
10.20 bsps/sparc/leon3/include/tm27.h File Reference . . . 1567
10.20.1 Detailed Description . . . 1567
10.20.2 Macro Definition Documentation . . . 1567
10.20.2.1 Cause_tm27_intr . . . 1567
10.20.2.2 Install_tm27_vector . . . 1567
10.21 bsps/sparc/leon3/start/bsp_fatal_halt.c File Reference . . . 1568
10.21.1 Detailed Description . . . 1568
10.21.2 Function Documentation . . . 1568
10.21.2.1 _CPU_Fatal_halt() . . . 1568
10.22 bsps/sparc/leon3/start/bspclean.c File Reference . . . 1568
10.22.1 Detailed Description . . . 1569
10.23 bsps/sparc/leon3/start/bspdelay.c File Reference . . . 1569
10.23.1 Detailed Description . . . 1569
10.24 bsps/sparc/leon3/start/bspsmp.c File Reference . . . 1569
10.24.1 Detailed Description . . . 1570
10.25 bsps/sparc/leon3/start/setvec.c File Reference . . . 1570
10.25.1 Detailed Description . . . 1570
10.26 bsps/sparc/shared/start/bsp_fatal_exit.c File Reference . . . 1570
10.26.1 Detailed Description . . . 1571
10.27 cpukit/include/rtems.h File Reference . . . 1571
10.27.1 Detailed Description . . . 1571
10.28 cpukit/include/rtems/assoc.h File Reference . . . 1571
10.28.1 Detailed Description . . . 1572
10.29 cpukit/include/rtems/bspIo.h File Reference . . . 1572
10.29.1 Detailed Description . . . 1573
10.29.2 Typedef Documentation . . . 1573
10.29.2.1 BSP_output_char_function_type . . . 1573
10.29.2.2 BSP_polling_getchar_function_type . . . 1574
10.29.3 Function Documentation . . . 1574
10.29.3.1 getchark() . . . 1574
10.29.3.2 printk() . . . 1574
10.29.3.3 putk() . . . 1575
10.29.3.4 rtems_put_char() . . . 1575
10.29.3.5 rtems_putc() . . . 1575
10.29.3.6 vprintk() . . . 1576
10.29.4 Variable Documentation . . . 1576
10.29.4.1 BSP_output_char . . . 1576
10.29.4.2 BSP_poll_char . . . 1577
10.30 cpukit/include/rtems/chain.h File Reference . . . 1577
10.30.1 Detailed Description . . . 1579
10.31 cpukit/include/rtems/score/chain.h File Reference . . . 1579

10.31.1 Detailed Description . . . 1580
10.32 cpukit/include/rtems/clockdrv.h File Reference . . . 1580
10.32.1 Detailed Description . . . 1580
10.33 cpukit/include/rtems/confdefs.h File Reference . . . 1581
10.33.1 Detailed Description . . . 1581
10.34 cpukit/include/rtems/confdefs/bdbuf.h File Reference . . . 1581
10.34.1 Detailed Description . . . 1581
10.35 cpukit/include/rtems/confdefs/clock.h File Reference . . . 1582
10.35.1 Detailed Description . . . 1582
10.36 cpukit/include/rtems/rtems/clock.h File Reference . . . 1582
10.36.1 Detailed Description . . . 1583
10.36.2 Function Documentation . . . 1583
10.36.2.1 _TOD_To_seconds() . . . 1583
10.36.2.2 _TOD_Validate() . . . 1583
10.37 cpukit/include/rtems/confdefs/console.h File Reference . . . 1584
10.37.1 Detailed Description . . . 1584
10.38 cpukit/include/rtems/confdefs/extensions.h File Reference . . . 1584
10.38.1 Detailed Description . . . 1584
10.39 cpukit/include/rtems/confdefs/inittask.h File Reference . . . 1584
10.39.1 Detailed Description . . . 1584
10.40 cpukit/include/rtems/confdefs/initthread.h File Reference . . . 1584
10.40.1 Detailed Description . . . 1584
10.41 cpukit/include/rtems/confdefs/iodrivers.h File Reference . . . 1585
10.41.1 Detailed Description . . . 1585
10.42 cpukit/include/rtems/confdefs/libio.h File Reference . . . 1585
10.42.1 Detailed Description . . . 1585
10.43 cpukit/include/rtems/libio.h File Reference . . . 1585
10.43.1 Detailed Description . . . 1591
10.44 cpukit/include/rtems/confdefs/libpci.h File Reference . . . 1592
10.44.1 Detailed Description . . . 1592
10.45 cpukit/include/rtems/confdefs/malloc.h File Reference . . . 1592
10.45.1 Detailed Description . . . 1592
10.46 cpukit/include/rtems/malloc.h File Reference . . . 1592
10.46.1 Detailed Description . . . 1593
10.46.2 Function Documentation . . . 1593
10.46.2.1 rtems_calloc() . . . 1593
10.46.2.2 rtems_heap_allocate_aligned_with_boundary() . . . 1594
10.46.2.3 rtems_heap_extend() . . . 1594
10.46.2.4 rtems_heap_greedy_allocate() . . . 1595
10.46.2.5 rtems_heap_greedy_allocate_all_except_largest() . . . 1595
10.46.2.6 rtems_heap_greedy_free() . . . 1595
10.46.2.7 rtems_malloc() . . . 1595

10.46.2.8 rtems_malloc_dirty_memory() . . . 1596
10.46.2.9 rtems_memalign() . . . 1596
10.46.3 Variable Documentation . . . 1596
10.46.3.1 RTEMS_Malloc_Heap . . . 1597
10.47 cpukit/include/rtems/confdefs/mpci.h File Reference . . . 1597
10.47.1 Detailed Description . . . 1597
10.48 cpukit/include/rtems/confdefs/newlib.h File Reference . . . 1597
10.48.1 Detailed Description . . . 1597
10.49 cpukit/include/rtems/confdefs/objectsclassic.h File Reference . . . 1597
10.49.1 Detailed Description . . . 1597
10.50 cpukit/include/rtems/confdefs/objectsposix.h File Reference . . . 1597
10.50.1 Detailed Description . . . 1598
10.51 cpukit/include/rtems/confdefs/obsolete.h File Reference . . . 1598
10.51.1 Detailed Description . . . 1598
10.52 cpukit/include/rtems/confdefs/percpu.h File Reference . . . 1598
10.52.1 Detailed Description . . . 1598
10.53 cpukit/include/rtems/score/percpu.h File Reference . . . 1598
10.53.1 Detailed Description . . . 1600
10.54 cpukit/include/rtems/confdefs/scheduler.h File Reference . . . 1600
10.54.1 Detailed Description . . . 1600
10.55 cpukit/include/rtems/scheduler.h File Reference . . . 1600
10.55.1 Detailed Description . . . 1601
10.55.2 Macro Definition Documentation . . . 1601
10.55.2.1 RTEMS_SCHEDULER_ASSIGN . . . 1601
10.56 cpukit/include/rtems/score/scheduler.h File Reference . . . 1601
10.56.1 Detailed Description . . . 1603
10.57 cpukit/include/rtems/confdefs/threads.h File Reference . . . 1603
10.57.1 Detailed Description . . . 1603
10.58 cpukit/include/rtems/confdefs/unlimited.h File Reference . . . 1604
10.58.1 Detailed Description . . . 1604
10.59 cpukit/include/rtems/confdefs/wkspace.h File Reference . . . 1604
10.59.1 Detailed Description . . . 1604
10.60 cpukit/include/rtems/score/wkspace.h File Reference . . . 1604
10.60.1 Detailed Description . . . 1604
10.61 cpukit/include/rtems/confdefs/wkspacesupport.h File Reference . . . 1605
10.61.1 Detailed Description . . . 1605
10.62 cpukit/include/rtems/config.h File Reference . . . 1605
10.62.1 Detailed Description . . . 1607
10.63 cpukit/include/rtems/rtems/config.h File Reference . . . 1607
10.63.1 Detailed Description . . . 1607
10.64 cpukit/include/rtems/counter.h File Reference . . . 1608
10.64.1 Detailed Description . . . 1608

10.65 cpukit/include/rtems/extension.h File Reference . . . 1608
10.65.1 Detailed Description . . . 1609
10.66 cpukit/include/rtems/extensiondata.h File Reference . . . 1609
10.66.1 Detailed Description . . . 1610
10.67 cpukit/include/rtems/extensionimpl.h File Reference . . . 1610
10.67.1 Detailed Description . . . 1610
10.68 cpukit/include/rtems/fs.h File Reference . . . 1610
10.68.1 Detailed Description . . . 1611
10.69 cpukit/include/rtems/init.h File Reference . . . 1611
10.69.1 Detailed Description . . . 1611
10.70 cpukit/include/rtems/io.h File Reference . . . 1612
10.70.1 Detailed Description . . . 1613
10.71 cpukit/include/rtems/irq-extension.h File Reference . . . 1613
10.71.1 Detailed Description . . . 1615
10.72 cpukit/include/rtems/libcsupport.h File Reference . . . 1615
10.72.1 Detailed Description . . . 1616
10.73 cpukit/include/rtems/mallocinitone.h File Reference . . . 1616
10.73.1 Detailed Description . . . 1617
10.74 cpukit/include/rtems/posix/spinlockimpl.h File Reference . . . 1617
10.74.1 Detailed Description . . . 1617
10.75 cpukit/include/rtems/print.h File Reference . . . 1617
10.75.1 Detailed Description . . . 1618
10.75.2 Function Documentation . . . 1618
10.75.2.1 rtems_printf() . . . 1618
10.75.2.2 rtems_vprintf() . . . 1618
10.76 cpukit/include/rtems/printer.h File Reference . . . 1619
10.76.1 Detailed Description . . . 1620
10.77 cpukit/include/rtems/rtems/asr.h File Reference . . . 1620
10.77.1 Detailed Description . . . 1621
10.78 cpukit/include/rtems/rtems/asrdata.h File Reference . . . 1621
10.78.1 Detailed Description . . . 1622
10.79 cpukit/include/rtems/rtems/asrimpl.h File Reference . . . 1622
10.79.1 Detailed Description . . . 1622
10.80 cpukit/include/rtems/rtems/attr.h File Reference . . . 1622
10.80.1 Detailed Description . . . 1624
10.81 cpukit/include/rtems/rtems/attrimpl.h File Reference . . . 1624
10.81.1 Detailed Description . . . 1625
10.82 cpukit/include/rtems/rtems/barrier.h File Reference . . . 1625
10.82.1 Detailed Description . . . 1625
10.83 cpukit/include/rtems/rtems/barrierdata.h File Reference . . . 1626
10.83.1 Detailed Description . . . 1626
10.84 cpukit/include/rtems/rtems/barrierimpl.h File Reference . . . 1626

10.84.1 Detailed Description . . . 1627
10.85 cpukit/include/rtems/rtems/cache.h File Reference . . . 1627
10.85.1 Detailed Description . . . 1628
10.86 cpukit/include/rtems/rtems/dpmem.h File Reference . . . 1628
10.86.1 Detailed Description . . . 1628
10.87 cpukit/include/rtems/rtems/dpmemdata.h File Reference . . . 1629
10.87.1 Detailed Description . . . 1629
10.88 cpukit/include/rtems/rtems/dpmemimpl.h File Reference . . . 1629
10.88.1 Detailed Description . . . 1629
10.89 cpukit/include/rtems/rtems/event.h File Reference . . . 1630
10.89.1 Detailed Description . . . 1632
10.89.2 Function Documentation . . . 1632
10.89.2.1 rtems_event_system_receive() . . . 1632
10.89.2.2 rtems_event_system_send() . . . 1633
10.89.2.3 rtems_event_transient_receive() . . . 1633
10.89.2.4 rtems_event_transient_send() . . . 1633
10.90 cpukit/include/rtems/rtems/eventdata.h File Reference . . . 1634
10.90.1 Detailed Description . . . 1634
10.91 cpukit/include/rtems/rtems/eventimpl.h File Reference . . . 1634
10.91.1 Detailed Description . . . 1635
10.92 cpukit/include/rtems/rtems/intr.h File Reference . . . 1635
10.92.1 Detailed Description . . . 1636
10.93 cpukit/include/rtems/rtems/mainpage.h File Reference . . . 1636
10.93.1 Detailed Description . . . 1636
10.94 cpukit/include/rtems/rtems/message.h File Reference . . . 1637
10.94.1 Detailed Description . . . 1638
10.95 cpukit/include/rtems/rtems/messagedata.h File Reference . . . 1638
10.95.1 Detailed Description . . . 1638
10.96 cpukit/include/rtems/rtems/messageimpl.h File Reference . . . 1638
10.96.1 Detailed Description . . . 1639
10.97 cpukit/include/rtems/rtems/modes.h File Reference . . . 1639
10.97.1 Detailed Description . . . 1640
10.98 cpukit/include/rtems/rtems/modesimpl.h File Reference . . . 1640
10.98.1 Detailed Description . . . 1641
10.99 cpukit/include/rtems/rtems/object.h File Reference . . . 1641
10.99.1 Detailed Description . . . 1643
10.100 cpukit/include/rtems/score/object.h File Reference . . . 1643
10.100.1 Detailed Description . . . 1644
10.101 cpukit/include/rtems/rtems/objectimpl.h File Reference . . . 1644
10.101.1 Detailed Description . . . 1645
10.102 cpukit/include/rtems/score/objectimpl.h File Reference . . . 1645
10.102.1 Detailed Description . . . 1648

10.103 cpukit/include/rtems/rtems/options.h File Reference . . . 1648
10.103.1 Detailed Description . . . 1648
10.104 cpukit/include/rtems/rtems/optionsimpl.h File Reference . . . 1648
10.104.1 Detailed Description . . . 1649
10.105 cpukit/include/rtems/rtems/part.h File Reference . . . 1649
10.105.1 Detailed Description . . . 1649
10.106 cpukit/include/rtems/rtems/partdata.h File Reference . . . 1650
10.106.1 Detailed Description . . . 1650
10.107 cpukit/include/rtems/rtems/partimpl.h File Reference . . . 1650
10.107.1 Detailed Description . . . 1651
10.108 cpukit/include/rtems/rtems/ratemon.h File Reference . . . 1651
10.108.1 Detailed Description . . . 1652
10.109 cpukit/include/rtems/rtems/ratemondata.h File Reference . . . 1652
10.109.1 Detailed Description . . . 1652
10.110 cpukit/include/rtems/rtems/ratemonimpl.h File Reference . . . 1653
10.110.1 Detailed Description . . . 1653
10.111 cpukit/include/rtems/rtems/region.h File Reference . . . 1654
10.111.1 Detailed Description . . . 1654
10.112 cpukit/include/rtems/rtems/regiondata.h File Reference . . . 1654
10.112.1 Detailed Description . . . 1655
10.113 cpukit/include/rtems/rtems/regionimpl.h File Reference . . . 1655
10.113.1 Detailed Description . . . 1656
10.114 cpukit/include/rtems/rtems/sem.h File Reference . . . 1656
10.114.1 Detailed Description . . . 1656
10.115 cpukit/include/rtems/rtems/semdata.h File Reference . . . 1656
10.115.1 Detailed Description . . . 1657
10.116 cpukit/include/rtems/rtems/semimpl.h File Reference . . . 1657
10.116.1 Detailed Description . . . 1658
10.117 cpukit/include/rtems/rtems/signal.h File Reference . . . 1658
10.117.1 Detailed Description . . . 1658
10.118 cpukit/include/rtems/rtems/signalimpl.h File Reference . . . 1658
10.118.1 Detailed Description . . . 1658
10.119 cpukit/include/rtems/rtems/status.h File Reference . . . 1659
10.119.1 Detailed Description . . . 1659
10.120 cpukit/include/rtems/rtems/statusimpl.h File Reference . . . 1660
10.120.1 Detailed Description . . . 1660
10.121 cpukit/include/rtems/rtems/support.h File Reference . . . 1660
10.121.1 Detailed Description . . . 1661
10.122 cpukit/include/rtems/rtems/tasks.h File Reference . . . 1661
10.122.1 Detailed Description . . . 1664
10.123 cpukit/include/rtems/rtems/tasksdata.h File Reference . . . 1664
10.123.1 Detailed Description . . . 1665

10.124 cpukit/include/rtems/rtems/tasksimpl.h File Reference . . . 1665
10.124.1 Detailed Description . . . 1665
10.125 cpukit/include/rtems/rtems/timer.h File Reference . . . 1666
10.125.1 Detailed Description . . . 1667
10.126 cpukit/include/rtems/rtems/timerdata.h File Reference . . . 1667
10.126.1 Detailed Description . . . 1668
10.127 cpukit/include/rtems/rtems/timerimpl.h File Reference . . . 1668
10.127.1 Detailed Description . . . 1669
10.128 cpukit/include/rtems/rtems/types.h File Reference . . . 1669
10.128.1 Detailed Description . . . 1670
10.129 cpukit/include/rtems/score/address.h File Reference . . . 1670
10.129.1 Detailed Description . . . 1670
10.130 cpukit/include/rtems/score/apimutex.h File Reference . . . 1671
10.130.1 Detailed Description . . . 1671
10.131 cpukit/include/rtems/score/assert.h File Reference . . . 1671
10.131.1 Detailed Description . . . 1672
10.132 cpukit/include/rtems/score/atomic.h File Reference . . . 1672
10.132.1 Detailed Description . . . 1673
10.133 cpukit/include/rtems/score/basedefs.h File Reference . . . 1673
10.133.1 Detailed Description . . . 1676
10.133.2 Macro Definition Documentation . . . 1676
10.133.2.1 RTEMS_UNREACHABLE . . . 1676
10.134 cpukit/include/rtems/score/chainimpl.h File Reference . . . 1676
10.134.1 Detailed Description . . . 1679
10.135 cpukit/include/rtems/score/context.h File Reference . . . 1679
10.135.1 Detailed Description . . . 1679
10.136 cpukit/include/rtems/score/copyrt.h File Reference . . . 1679
10.136.1 Detailed Description . . . 1680
10.137 cpukit/include/rtems/score/corebarrier.h File Reference . . . 1680
10.137.1 Detailed Description . . . 1680
10.138 cpukit/include/rtems/score/corebarrierimpl.h File Reference . . . 1680
10.138.1 Detailed Description . . . 1681
10.139 cpukit/include/rtems/score/coremsg.h File Reference . . . 1681
10.139.1 Detailed Description . . . 1682
10.140 cpukit/include/rtems/score/coremsgbuffer.h File Reference . . . 1682
10.140.1 Detailed Description . . . 1682
10.141 cpukit/include/rtems/score/coremsgimpl.h File Reference . . . 1683
10.141.1 Detailed Description . . . 1684
10.142 cpukit/include/rtems/score/coremutex.h File Reference . . . 1685
10.142.1 Detailed Description . . . 1685
10.143 cpukit/include/rtems/score/coremuteximpl.h File Reference . . . 1685
10.143.1 Detailed Description . . . 1687

10.144 cpukit/include/rtems/score/coresem.h File Reference . . . 1687
10.144.1 Detailed Description . . . 1687
10.145 cpukit/include/rtems/score/coresemimpl.h File Reference . . . 1687
10.145.1 Detailed Description . . . 1688
10.146 cpukit/include/rtems/score/cpustdatomic.h File Reference . . . 1688
10.146.1 Detailed Description . . . 1690
10.147 cpukit/include/rtems/score/freechain.h File Reference . . . 1690
10.147.1 Detailed Description . . . 1691
10.148 cpukit/include/rtems/score/freechainimpl.h File Reference . . . 1691
10.148.1 Detailed Description . . . 1691
10.149 cpukit/include/rtems/score/heap.h File Reference . . . 1692
10.149.1 Detailed Description . . . 1693
10.150 cpukit/include/rtems/score/heapimpl.h File Reference . . . 1693
10.150.1 Detailed Description . . . 1696
10.151 cpukit/include/rtems/score/heapinfo.h File Reference . . . 1696
10.151.1 Detailed Description . . . 1696
10.152 cpukit/include/rtems/score/interr.h File Reference . . . 1696
10.152.1 Detailed Description . . . 1697
10.153 cpukit/include/rtems/score/isr.h File Reference . . . 1698
10.153.1 Detailed Description . . . 1698
10.154 cpukit/include/rtems/score/isrlevel.h File Reference . . . 1699
10.154.1 Detailed Description . . . 1699
10.155 cpukit/include/rtems/score/isrlock.h File Reference . . . 1699
10.155.1 Detailed Description . . . 1700
10.156 cpukit/include/rtems/score/memory.h File Reference . . . 1701
10.156.1 Detailed Description . . . 1702
10.157 cpukit/include/rtems/score/mppkt.h File Reference . . . 1702
10.157.1 Detailed Description . . . 1703
10.158 cpukit/include/rtems/score/mrsp.h File Reference . . . 1703
10.158.1 Detailed Description . . . 1703
10.159 cpukit/include/rtems/score/mrspimpl.h File Reference . . . 1703
10.159.1 Detailed Description . . . 1704
10.160 cpukit/include/rtems/score/muteximpl.h File Reference . . . 1705
10.160.1 Detailed Description . . . 1705
10.161 cpukit/include/rtems/score/objectdata.h File Reference . . . 1705
10.161.1 Detailed Description . . . 1706
10.162 cpukit/include/rtems/score/percpudata.h File Reference . . . 1706
10.162.1 Detailed Description . . . 1707
10.163 cpukit/include/rtems/score/priority.h File Reference . . . 1707
10.163.1 Detailed Description . . . 1708
10.164 cpukit/include/rtems/score/prioritybitmap.h File Reference . . . 1708
10.164.1 Detailed Description . . . 1708

10.165 cpukit/include/rtems/score/prioritybitmapimpl.h File Reference . . . 1708
10.165.1 Detailed Description . . . 1709
10.166 cpukit/include/rtems/score/priorityimpl.h File Reference . . . 1709
10.166.1 Detailed Description . . . 1711
10.167 cpukit/include/rtems/score/processormask.h File Reference . . . 1711
10.167.1 Detailed Description . . . 1713
10.168 cpukit/include/rtems/score/profiling.h File Reference . . . 1713
10.168.1 Detailed Description . . . 1714
10.169 cpukit/include/rtems/score/protectedheap.h File Reference . . . 1714
10.169.1 Detailed Description . . . 1714
10.170 cpukit/include/rtems/score/rbtree.h File Reference . . . 1715
10.170.1 Detailed Description . . . 1716
10.171 cpukit/include/rtems/score/rbtreeimpl.h File Reference . . . 1717
10.171.1 Detailed Description . . . 1717
10.172 cpukit/include/rtems/score/scheduleredf.h File Reference . . . 1717
10.172.1 Detailed Description . . . 1718
10.173 cpukit/include/rtems/score/scheduleredfimpl.h File Reference . . . 1719
10.173.1 Detailed Description . . . 1719
10.174 cpukit/include/rtems/score/scheduleredfsmp.h File Reference . . . 1719
10.174.1 Detailed Description . . . 1721
10.175 cpukit/include/rtems/score/schedulerimpl.h File Reference . . . 1721
10.175.1 Detailed Description . . . 1723
10.176 cpukit/include/rtems/score/schedulernode.h File Reference . . . 1724
10.176.1 Detailed Description . . . 1724
10.177 cpukit/include/rtems/score/schedulernodeimpl.h File Reference . . . 1724
10.177.1 Detailed Description . . . 1725
10.178 cpukit/include/rtems/score/schedulerpriority.h File Reference . . . 1726
10.178.1 Detailed Description . . . 1726
10.179 cpukit/include/rtems/score/schedulerpriorityimpl.h File Reference . . . 1727
10.179.1 Detailed Description . . . 1727
10.180 cpukit/include/rtems/score/schedulersimple.h File Reference . . . 1728
10.180.1 Detailed Description . . . 1728
10.181 cpukit/include/rtems/score/schedulersimpleimpl.h File Reference . . . 1729
10.181.1 Detailed Description . . . 1729
10.182 cpukit/include/rtems/score/schedulersmp.h File Reference . . . 1729
10.182.1 Detailed Description . . . 1730
10.183 cpukit/include/rtems/score/schedulersmpimpl.h File Reference . . . 1730
10.183.1 Detailed Description . . . 1733
10.184 cpukit/include/rtems/score/semaphoreimpl.h File Reference . . . 1733
10.184.1 Detailed Description . . . 1734
10.185 cpukit/include/rtems/score/smp.h File Reference . . . 1734
10.185.1 Detailed Description . . . 1734

10.186 cpukit/include/rtems/score/smpbarrier.h File Reference . . . 1734
10.186.1 Detailed Description . . . 1735
10.187 cpukit/include/rtems/score/smpimpl.h File Reference . . . 1735
10.187.1 Detailed Description . . . 1737
10.188 cpukit/include/rtems/score/smplock.h File Reference . . . 1737
10.188.1 Detailed Description . . . 1738
10.189 cpukit/include/rtems/score/smplockmcs.h File Reference . . . 1738
10.189.1 Detailed Description . . . 1739
10.190 cpukit/include/rtems/score/smplockseq.h File Reference . . . 1739
10.190.1 Detailed Description . . . 1740
10.191 cpukit/include/rtems/score/smplockstats.h File Reference . . . 1740
10.191.1 Detailed Description . . . 1740
10.192 cpukit/include/rtems/score/smplockticket.h File Reference . . . 1740
10.192.1 Detailed Description . . . 1741
10.193 cpukit/include/rtems/score/stack.h File Reference . . . 1741
10.193.1 Detailed Description . . . 1742
10.194 cpukit/include/rtems/score/stackimpl.h File Reference . . . 1742
10.194.1 Detailed Description . . . 1743
10.195 cpukit/include/rtems/score/states.h File Reference . . . 1743
10.195.1 Detailed Description . . . 1743
10.196 cpukit/include/rtems/score/statesimpl.h File Reference . . . 1743
10.196.1 Detailed Description . . . 1745
10.197 cpukit/include/rtems/score/sysstate.h File Reference . . . 1745
10.197.1 Detailed Description . . . 1745
10.198 cpukit/include/rtems/score/thread.h File Reference . . . 1746
10.198.1 Detailed Description . . . 1748
10.199 cpukit/include/rtems/score/threaddispatch.h File Reference . . . 1748
10.199.1 Detailed Description . . . 1749
10.200 cpukit/include/rtems/score/threadidledata.h File Reference . . . 1749
10.200.1 Detailed Description . . . 1750
10.201 cpukit/include/rtems/score/threadimpl.h File Reference . . . 1750
10.201.1 Detailed Description . . . 1757
10.202 cpukit/include/rtems/score/threadq.h File Reference . . . 1757
10.202.1 Detailed Description . . . 1758
10.203 cpukit/include/rtems/score/threadqimpl.h File Reference . . . 1758
10.203.1 Detailed Description . . . 1763
10.204 cpukit/include/rtems/score/timecounter.h File Reference . . . 1763
10.204.1 Detailed Description . . . 1764
10.205 cpukit/include/rtems/timecounter.h File Reference . . . 1764
10.205.1 Detailed Description . . . 1765
10.206 cpukit/include/rtems/score/timecounterimpl.h File Reference . . . 1765
10.206.1 Detailed Description . . . 1766

10.207 cpukit/include/rtems/score/timespec.h File Reference . . . 1766
10.207.1 Detailed Description . . . 1767
10.208 cpukit/include/rtems/score/timestamp.h File Reference . . . 1767
10.208.1 Detailed Description . . . 1767
10.209 cpukit/include/rtems/score/timestampimpl.h File Reference . . . 1767
10.209.1 Detailed Description . . . 1768
10.210 cpukit/include/rtems/score/tls.h File Reference . . . 1768
10.210.1 Detailed Description . . . 1769
10.211 cpukit/include/rtems/score/todimpl.h File Reference . . . 1769
10.211.1 Detailed Description . . . 1771
10.211.2 Macro Definition Documentation . . . 1771
10.211.2.1 TOD_BASE_YEAR . . . 1771
10.211.2.2 TOD_SECONDS_1970_THROUGH_1988 . . . 1772
10.212 cpukit/include/rtems/score/userext.h File Reference . . . 1772
10.212.1 Detailed Description . . . 1773
10.213 cpukit/include/rtems/score/userextdata.h File Reference . . . 1773
10.213.1 Detailed Description . . . 1773
10.214 cpukit/include/rtems/score/userextimpl.h File Reference . . . 1773
10.214.1 Detailed Description . . . 1775
10.215 cpukit/include/rtems/score/watchdog.h File Reference . . . 1776
10.215.1 Detailed Description . . . 1776
10.216 cpukit/include/rtems/score/watchdogimpl.h File Reference . . . 1776
10.216.1 Detailed Description . . . 1778
10.217 cpukit/include/rtems/score/watchdogticks.h File Reference . . . 1779
10.217.1 Detailed Description . . . 1779
10.218 cpukit/include/rtems/score/wkspacedata.h File Reference . . . 1779
10.218.1 Detailed Description . . . 1780
10.219 cpukit/include/rtems/score/wkspaceinitone.h File Reference . . . 1780
10.219.1 Detailed Description . . . 1780
10.220 cpukit/include/rtems/termiostypes.h File Reference . . . 1781
10.220.1 Detailed Description . . . 1783
10.220.2 Macro Definition Documentation . . . 1783
10.220.2.1 RTEMS_TERMIOS_DEVICE_CONTEXT_INITIALIZER . . . 1783
10.220.3 Typedef Documentation . . . 1783
10.220.3.1 rtems_termios_device_context . . . 1783
10.220.3.2 rtems_termios_device_node . . . 1784
10.220.4 Function Documentation . . . 1784
10.220.4.1 rtems_termios_baud_to_number() . . . 1784
10.220.4.2 rtems_termios_device_context_initialize() . . . 1784
10.220.4.3 rtems_termios_device_install() . . . 1784
10.220.4.4 rtems_termios_device_lock_acquire() . . . 1785
10.220.4.5 rtems_termios_device_lock_release() . . . 1785

10.220.4.6 rtems_termios_get_device_context() . . . 1786
10.220.4.7 rtems_termios_kqfilter() . . . 1786
10.220.4.8 rtems_termios_mmap() . . . 1786
10.220.4.9 rtems_termios_number_to_baud() . . . 1787
10.220.4.10 rtems_termios_poll() . . . 1787
10.220.4.11 rtems_termios_set_best_baud() . . . 1787
10.220.4.12 rtems_termios_set_initial_baud() . . . 1787
10.221 cpukit/include/rtems/version.h File Reference . . . 1788
10.221.1 Detailed Description . . . 1788
10.222 cpukit/include/sys/statvfs.h File Reference . . . 1788
10.222.1 Detailed Description . . . 1789
10.223 cpukit/libcsupport/src/malloc_deferred.c File Reference . . . 1789
10.223.1 Detailed Description . . . 1789
10.224 cpukit/libcsupport/src/mallocheap.c File Reference . . . 1789
10.224.1 Detailed Description . . . 1790
10.224.2 Variable Documentation . . . 1790
10.224.2.1 RTEMS_Malloc_Heap . . . 1790
10.225 cpukit/libcsupport/src/print_printf.c File Reference . . . 1790
10.225.1 Detailed Description . . . 1790
10.225.2 Function Documentation . . . 1791
10.225.2.1 rtems_vprintf() . . . 1791
10.226 cpukit/libcsupport/src/printk.c File Reference . . . 1791
10.226.1 Detailed Description . . . 1791
10.226.2 Function Documentation . . . 1791
10.226.2.1 printk() . . . 1792
10.227 cpukit/libcsupport/src/printk_plugin.c File Reference . . . 1792
10.227.1 Detailed Description . . . 1792
10.228 cpukit/libcsupport/src/rtems_putc.c File Reference . . . 1792
10.228.1 Detailed Description . . . 1792
10.228.2 Function Documentation . . . 1793
10.228.2.1 rtems_putc() . . . 1793
10.229 cpukit/libcsupport/src/termiosinitialize.c File Reference . . . 1794
10.229.1 Detailed Description . . . 1794
10.230 cpukit/libcsupport/src/vprintk.c File Reference . . . 1794
10.230.1 Detailed Description . . . 1794
10.230.2 Function Documentation . . . 1794
10.230.2.1 vprintk() . . . 1794
10.231 cpukit/libtest/t-test-busy-tick.c File Reference . . . 1795
10.231.1 Detailed Description . . . 1795
10.232 cpukit/libtest/t-test-busy.c File Reference . . . 1795
10.232.1 Detailed Description . . . 1795
10.233 cpukit/libtest/t-test-interrupt.c File Reference . . . 1796

10.233.1 Detailed Description . . . 1797
10.233.2 Variable Documentation . . . 1797
10.233.2.1 T_interrupt_fixture . . . 1797
10.233.2.2 T_interrupt_instance . . . 1797
10.234 cpukit/libtest/t-test-rtems-objs.c File Reference . . . 1798
10.234.1 Detailed Description . . . 1799
10.235 cpukit/libtest/t-test-rtems.h File Reference . . . 1799
10.235.1 Detailed Description . . . 1799
10.236 cpukit/libtest/t-test-thread-switch.c File Reference . . . 1799
10.236.1 Detailed Description . . . 1800
10.236.2 Variable Documentation . . . 1800
10.236.2.1 T_thread_switch_instance . . . 1800
10.237 cpukit/libtest/testrun.c File Reference . . . 1800
10.237.1 Detailed Description . . . 1801
10.237.2 Variable Documentation . . . 1801
10.237.2.1 actions . . . 1801
10.237.2.2 config . . . 1801
10.238 cpukit/posix/src/pspindestroy.c File Reference . . . 1801
10.238.1 Detailed Description . . . 1802
10.239 cpukit/posix/src/pspininit.c File Reference . . . 1802
10.239.1 Detailed Description . . . 1802
10.240 cpukit/posix/src/pspinlock.c File Reference . . . 1802
10.240.1 Detailed Description . . . 1803
10.241 cpukit/posix/src/pspinunlock.c File Reference . . . 1803
10.241.1 Detailed Description . . . 1803
10.242 cpukit/rtems/src/barrier.c File Reference . . . 1803
10.242.1 Detailed Description . . . 1803
10.243 cpukit/rtems/src/barriercreate.c File Reference . . . 1804
10.243.1 Detailed Description . . . 1804
10.244 cpukit/rtems/src/barrierdelete.c File Reference . . . 1804
10.244.1 Detailed Description . . . 1805
10.245 cpukit/rtems/src/barrierident.c File Reference . . . 1805
10.245.1 Detailed Description . . . 1805
10.246 cpukit/rtems/src/barrierrelease.c File Reference . . . 1805
10.246.1 Detailed Description . . . 1805
10.247 cpukit/rtems/src/barrierwait.c File Reference . . . 1806
10.247.1 Detailed Description . . . 1806
10.248 cpukit/rtems/src/clockgetuptime.c File Reference . . . 1806
10.248.1 Detailed Description . . . 1806
10.249 cpukit/rtems/src/clocktodtoseconds.c File Reference . . . 1806
10.249.1 Detailed Description . . . 1807
10.249.2 Function Documentation . . . 1807

10.249.2.1 _TOD_To_seconds() . . . 1807
10.249.3 Variable Documentation . . . 1807
10.249.3.1 _TOD_Days_to_date . . . 1807
10.250 cpukit/rtems/src/clocktodvalidate.c File Reference . . . 1808
10.250.1 Detailed Description . . . 1808
10.250.2 Function Documentation . . . 1808
10.250.2.1 _TOD_Validate() . . . 1808
10.250.3 Variable Documentation . . . 1809
10.250.3.1 _TOD_Days_per_month . . . 1809
10.251 cpukit/rtems/src/eventreceive.c File Reference . . . 1809
10.251.1 Detailed Description . . . 1809
10.252 cpukit/rtems/src/eventseize.c File Reference . . . 1809
10.252.1 Detailed Description . . . 1810
10.253 cpukit/rtems/src/eventsend.c File Reference . . . 1810
10.253.1 Detailed Description . . . 1810
10.254 cpukit/rtems/src/eventsurrender.c File Reference . . . 1810
10.254.1 Detailed Description . . . 1811
10.255 cpukit/rtems/src/intrcatch.c File Reference . . . 1811
10.255.1 Detailed Description . . . 1811
10.256 cpukit/rtems/src/msg.c File Reference . . . 1811
10.256.1 Detailed Description . . . 1811
10.257 cpukit/rtems/src/msgqbroadcast.c File Reference . . . 1811
10.257.1 Detailed Description . . . 1812
10.258 cpukit/rtems/src/msgqconstruct.c File Reference . . . 1812
10.258.1 Detailed Description . . . 1813
10.259 cpukit/rtems/src/msgqcreate.c File Reference . . . 1813
10.259.1 Detailed Description . . . 1813
10.260 cpukit/rtems/src/msgqdelete.c File Reference . . . 1813
10.260.1 Detailed Description . . . 1813
10.261 cpukit/rtems/src/msgqflush.c File Reference . . . 1814
10.261.1 Detailed Description . . . 1814
10.262 cpukit/rtems/src/msgqgetnumberpending.c File Reference . . . 1814
10.262.1 Detailed Description . . . 1814
10.263 cpukit/rtems/src/msgqident.c File Reference . . . 1814
10.263.1 Detailed Description . . . 1815
10.264 cpukit/rtems/src/msgqreceive.c File Reference . . . 1815
10.264.1 Detailed Description . . . 1815
10.265 cpukit/rtems/src/msgqsend.c File Reference . . . 1815
10.265.1 Detailed Description . . . 1815
10.266 cpukit/rtems/src/msgqurgent.c File Reference . . . 1816
10.266.1 Detailed Description . . . 1816
10.267 cpukit/rtems/src/part.c File Reference . . . 1816

10.267.1 Detailed Description . . . 1816
10.268 cpukit/rtems/src/partcreate.c File Reference . . . 1816
10.268.1 Detailed Description . . . 1817
10.269 cpukit/rtems/src/partdelete.c File Reference . . . 1817
10.269.1 Detailed Description . . . 1818
10.270 cpukit/rtems/src/partgetbuffer.c File Reference . . . 1818
10.270.1 Detailed Description . . . 1818
10.271 cpukit/rtems/src/partident.c File Reference . . . 1818
10.271.1 Detailed Description . . . 1818
10.272 cpukit/rtems/src/partreturnbuffer.c File Reference . . . 1819
10.272.1 Detailed Description . . . 1819
10.273 cpukit/rtems/src/ratemon.c File Reference . . . 1819
10.273.1 Detailed Description . . . 1819
10.274 cpukit/rtems/src/ratemoncancel.c File Reference . . . 1819
10.274.1 Detailed Description . . . 1820
10.275 cpukit/rtems/src/ratemoncreate.c File Reference . . . 1820
10.275.1 Detailed Description . . . 1821
10.276 cpukit/rtems/src/ratemondelete.c File Reference . . . 1821
10.276.1 Detailed Description . . . 1821
10.277 cpukit/rtems/src/ratemongetstatistics.c File Reference . . . 1821
10.277.1 Detailed Description . . . 1821
10.278 cpukit/rtems/src/ratemongetstatus.c File Reference . . . 1821
10.278.1 Detailed Description . . . 1822
10.279 cpukit/rtems/src/ratemonident.c File Reference . . . 1822
10.279.1 Detailed Description . . . 1822
10.280 cpukit/rtems/src/ratemonperiod.c File Reference . . . 1822
10.280.1 Detailed Description . . . 1823
10.281 cpukit/rtems/src/ratemonresetstatistics.c File Reference . . . 1823
10.281.1 Detailed Description . . . 1823
10.282 cpukit/rtems/src/ratemontimeout.c File Reference . . . 1823
10.282.1 Detailed Description . . . 1824
10.283 cpukit/rtems/src/rtemsnametoid.c File Reference . . . 1824
10.283.1 Detailed Description . . . 1824
10.284 cpukit/rtems/src/sem.c File Reference . . . 1824
10.284.1 Detailed Description . . . 1825
10.285 cpukit/rtems/src/semcreate.c File Reference . . . 1825
10.285.1 Detailed Description . . . 1826
10.285.2 Macro Definition Documentation . . . 1826
10.285.2.1 SEMAPHORE_KIND_MASK . . . 1826
10.286 cpukit/rtems/src/semdelete.c File Reference . . . 1826
10.286.1 Detailed Description . . . 1826
10.287 cpukit/rtems/src/semident.c File Reference . . . 1826

10.287.1 Detailed Description . . . 1827
10.288 cpukit/rtems/src/semobtain.c File Reference . . . 1827
10.288.1 Detailed Description . . . 1827
10.289 cpukit/rtems/src/semrelease.c File Reference . . . 1827
10.289.1 Detailed Description . . . 1828
10.290 cpukit/rtems/src/status.c File Reference . . . 1828
10.290.1 Detailed Description . . . 1828
10.291 cpukit/rtems/src/statustext.c File Reference . . . 1828
10.292 cpukit/rtems/src/systemeventreceive.c File Reference . . . 1828
10.292.1 Detailed Description . . . 1829
10.292.2 Function Documentation . . . 1829
10.292.2.1 rtems_event_system_receive() . . . 1829
10.293 cpukit/rtems/src/systemeventsend.c File Reference . . . 1829
10.293.1 Detailed Description . . . 1830
10.293.2 Function Documentation . . . 1830
10.293.2.1 rtems_event_system_send() . . . 1830
10.294 cpukit/rtems/src/taskconstruct.c File Reference . . . 1830
10.294.1 Detailed Description . . . 1831
10.294.2 Variable Documentation . . . 1831
10.294.2.1 _RTEMS_tasks_User_extensions . . . 1831
10.295 cpukit/rtems/src/taskdelete.c File Reference . . . 1832
10.295.1 Detailed Description . . . 1832
10.296 cpukit/rtems/src/taskgetaffinity.c File Reference . . . 1832
10.296.1 Detailed Description . . . 1832
10.297 cpukit/rtems/src/taskident.c File Reference . . . 1832
10.297.1 Detailed Description . . . 1833
10.298 cpukit/rtems/src/taskinitusers.c File Reference . . . 1833
10.298.1 Detailed Description . . . 1833
10.299 cpukit/rtems/src/taskissuspended.c File Reference . . . 1833
10.299.1 Detailed Description . . . 1833
10.300 cpukit/rtems/src/taskrestart.c File Reference . . . 1834
10.300.1 Detailed Description . . . 1834
10.301 cpukit/rtems/src/taskresume.c File Reference . . . 1834
10.301.1 Detailed Description . . . 1834
10.302 cpukit/rtems/src/tasks.c File Reference . . . 1834
10.302.1 Detailed Description . . . 1835
10.303 cpukit/rtems/src/taskself.c File Reference . . . 1835
10.303.1 Detailed Description . . . 1835
10.304 cpukit/rtems/src/tasksetaffinity.c File Reference . . . 1835
10.304.1 Detailed Description . . . 1835
10.305 cpukit/rtems/src/tasksetpriority.c File Reference . . . 1835
10.305.1 Detailed Description . . . 1836

10.306 cpukit/rtems/src/taskstart.c File Reference . . . 1836
10.306.1 Detailed Description . . . 1836
10.307 cpukit/rtems/src/tasksuspend.c File Reference . . . 1836
10.307.1 Detailed Description . . . 1837
10.308 cpukit/rtems/src/taskwakeafter.c File Reference . . . 1837
10.308.1 Detailed Description . . . 1837
10.309 cpukit/rtems/src/timercreate.c File Reference . . . 1837
10.309.1 Detailed Description . . . 1838
10.310 cpukit/rtems/src/timerdelete.c File Reference . . . 1838
10.310.1 Detailed Description . . . 1838
10.311 cpukit/rtems/src/timerfireafter.c File Reference . . . 1839
10.311.1 Detailed Description . . . 1839
10.312 cpukit/rtems/src/timerident.c File Reference . . . 1839
10.312.1 Detailed Description . . . 1839
10.313 cpukit/rtems/src/timerreset.c File Reference . . . 1839
10.313.1 Detailed Description . . . 1840
10.314 cpukit/sapi/src/exinit.c File Reference . . . 1840
10.314.1 Detailed Description . . . 1841
10.315 cpukit/sapi/src/extension.c File Reference . . . 1841
10.315.1 Detailed Description . . . 1841
10.316 cpukit/sapi/src/extensioncreate.c File Reference . . . 1841
10.316.1 Detailed Description . . . 1842
10.317 cpukit/sapi/src/extensiondelete.c File Reference . . . 1842
10.317.1 Detailed Description . . . 1842
10.318 cpukit/sapi/src/extensionident.c File Reference . . . 1843
10.318.1 Detailed Description . . . 1843
10.319 cpukit/sapi/src/getversionstring.c File Reference . . . 1843
10.319.1 Detailed Description . . . 1843
10.320 cpukit/sapi/src/version.c File Reference . . . 1843
10.320.1 Detailed Description . . . 1844
10.321 cpukit/score/cpu/sparc/cpu.c File Reference . . . 1844
10.321.1 Detailed Description . . . 1846
10.321.2 Macro Definition Documentation . . . 1846
10.321.2.1 SPARC_ASSERT_FP_OFFSET . . . 1846
10.321.2.2 SPARC_ASSERT_ISF_OFFSET . . . 1847
10.321.2.3 SPARC_ASSERT_OFFSET . . . 1847
10.321.3 Function Documentation . . . 1847
10.321.3.1 _CPU_Context_Initialize() . . . 1847
10.321.3.2 _CPU_Initialize() . . . 1848
10.321.3.3 _CPU_ISR_Get_level() . . . 1848
10.321.3.4 _CPU_ISR_install_raw_handler() . . . 1848
10.321.3.5 _CPU_ISR_install_vector() . . . 1849

10.321.4 Variable Documentation . . . 1849
10.321.4.1 _CPU_Trap_slot_template . . . 1849
10.322 cpukit/score/cpu/sparc/include/rtems/asm.h File Reference . . . 1850
10.322.1 Detailed Description . . . 1850
10.323 cpukit/score/cpu/sparc/include/rtems/score/cpu.h File Reference . . . 1850
10.323.1 Detailed Description . . . 1854
10.323.2 Macro Definition Documentation . . . 1854
10.323.2.1 _CPU_Context_Initialization_at_thread_begin . . . 1854
10.323.2.2 _CPU_Context_Restart_self . . . 1855
10.323.2.3 _CPU_Initialize_vectors . . . 1855
10.323.2.4 _CPU_ISR_Disable . . . 1855
10.323.2.5 _CPU_ISR_Enable . . . 1855
10.323.2.6 _CPU_ISR_Flash . . . 1856
10.323.2.7 _CPU_ISR_Set_level . . . 1856
10.323.2.8 CPU_ALIGNMENT . . . 1856
10.323.2.9 CPU_ALL_TASKS_ARE_FP . . . 1856
10.323.2.10 CPU_CONTEXT_FP_SIZE . . . 1857
10.323.2.11 CPU_HARDWARE_FP . . . 1857
10.323.2.12 CPU_HEAP_ALIGNMENT . . . 1857
10.323.2.13 CPU_IDLE_TASK_IS_FP . . . 1857
10.323.2.14 CPU_INTERRUPT_MAXIMUM_VECTOR_NUMBER . . . 1858
10.323.2.15 CPU_INTERRUPT_NUMBER_OF_VECTORS . . . 1858
10.323.2.16 CPU_ISR_PASSES_FRAME_POINTER . . . 1858
10.323.2.17 CPU_MODES_INTERRUPT_MASK . . . 1858
10.323.2.18 CPU_MPCI_RECEIVE_SERVER_EXTRA_STACK . . . 1859
10.323.2.19 CPU_PROVIDES_ISR_IS_IN_PROGRESS . . . 1859
10.323.2.20 CPU_SIMPLE_VECTORED_INTERRUPTS . . . 1859
10.323.2.21 CPU_SIZEOF_POINTER . . . 1859
10.323.2.22 CPU_SOFTWARE_FP . . . 1860
10.323.2.23 CPU_STACK_ALIGNMENT . . . 1860
10.323.2.24 CPU_STACK_FRAME_I0_OFFSET . . . 1860
10.323.2.25 CPU_STACK_FRAME_I1_OFFSET . . . 1860
10.323.2.26 CPU_STACK_FRAME_I2_OFFSET . . . 1860
10.323.2.27 CPU_STACK_FRAME_I3_OFFSET . . . 1861
10.323.2.28 CPU_STACK_FRAME_I4_OFFSET . . . 1861
10.323.2.29 CPU_STACK_FRAME_I5_OFFSET . . . 1861
10.323.2.30 CPU_STACK_FRAME_I6_FP_OFFSET . . . 1861
10.323.2.31 CPU_STACK_FRAME_I7_OFFSET . . . 1861
10.323.2.32 CPU_STACK_FRAME_L0_OFFSET . . . 1862
10.323.2.33 CPU_STACK_FRAME_L1_OFFSET . . . 1862
10.323.2.34 CPU_STACK_FRAME_L2_OFFSET . . . 1862
10.323.2.35 CPU_STACK_FRAME_L3_OFFSET . . . 1862

10.323.2.36 CPU_STACK_FRAME_L4_OFFSET . . . 1862
10.323.2.37 CPU_STACK_FRAME_L5_OFFSET . . . 1863
10.323.2.38 CPU_STACK_FRAME_L6_OFFSET . . . 1863
10.323.2.39 CPU_STACK_FRAME_L7_OFFSET . . . 1863
10.323.2.40 CPU_STACK_FRAME_PAD0_OFFSET . . . 1863
10.323.2.41 CPU_STACK_FRAME_SAVED_ARG0_OFFSET . . . 1863
10.323.2.42 CPU_STACK_FRAME_SAVED_ARG1_OFFSET . . . 1864
10.323.2.43 CPU_STACK_FRAME_SAVED_ARG2_OFFSET . . . 1864
10.323.2.44 CPU_STACK_FRAME_SAVED_ARG3_OFFSET . . . 1864
10.323.2.45 CPU_STACK_FRAME_SAVED_ARG4_OFFSET . . . 1864
10.323.2.46 CPU_STACK_FRAME_SAVED_ARG5_OFFSET . . . 1864
10.323.2.47 CPU_STACK_GROWS_UP . . . 1865
10.323.2.48 CPU_STACK_MINIMUM_SIZE . . . 1865
10.323.2.49 CPU_STRUCTURE_RETURN_ADDRESS_OFFSET . . . 1865
10.323.2.50 CPU_swap_u16 . . . 1865
10.323.2.51 CPU_USE_GENERIC_BITFIELD_CODE . . . 1866
10.323.3 Typedef Documentation . . . 1866
10.323.3.1 CPU_Uint32ptr . . . 1866
10.323.4 Function Documentation . . . 1866
10.323.4.1 _CPU_Context_Initialize() . . . 1866
10.323.4.2 _CPU_Context_restore() . . . 1867
10.323.4.3 _CPU_Context_switch() . . . 1867
10.323.4.4 _CPU_Fatal_halt() . . . 1868
10.323.4.5 _CPU_Initialize() . . . 1868
10.323.4.6 _CPU_ISR_Get_level() . . . 1868
10.323.4.7 _CPU_ISR_install_raw_handler() . . . 1868
10.323.4.8 _CPU_ISR_install_vector() . . . 1869
10.323.4.9 CPU_swap_u32() . . . 1869
10.323.5 Variable Documentation . . . 1870
10.323.5.1 _CPU_Trap_slot_template . . . 1870
10.324 cpukit/score/cpu/sparc/include/rtems/score/cpuimpl.h File Reference . . . 1870
10.324.1 Detailed Description . . . 1872
10.325 cpukit/score/cpu/sparc/include/rtems/score/sparc.h File Reference . . . 1872
10.325.1 Detailed Description . . . 1874
10.325.2 Macro Definition Documentation . . . 1874
10.325.2.1 CPU_MODEL_NAME . . . 1874
10.325.2.2 CPU_NAME . . . 1874
10.325.2.3 nop . . . 1874
10.325.2.4 SPARC_ASYNCHRONOUS_TRAP . . . 1874
10.325.2.5 sparc_flash_interrupts . . . 1875
10.325.2.6 sparc_get_interrupt_level . . . 1875
10.325.2.7 sparc_get_psr . . . 1876

10.325.2.8 sparc_get_tbr . . . 1876
10.325.2.9 sparc_get_wim . . . 1876
10.325.2.10 sparc_get_y . . . 1877
10.325.2.11 SPARC_HAS_BITSCAN . . . 1877
10.325.2.12 SPARC_HAS_FPU . . . 1877
10.325.2.13 SPARC_INTERRUPT_SOURCE_TO_TRAP . . . 1877
10.325.2.14 SPARC_INTERRUPT_TRAP_TO_SOURCE . . . 1878
10.325.2.15 SPARC_IS_INTERRUPT_TRAP . . . 1878
10.325.2.16 SPARC_LEON3FT_B2BST_NOP . . . 1879
10.325.2.17 SPARC_NUMBER_OF_REGISTER_WINDOWS . . . 1879
10.325.2.18 SPARC_PSR_CWP_BIT_POSITION . . . 1879
10.325.2.19 SPARC_PSR_CWP_MASK . . . 1879
10.325.2.20 SPARC_PSR_EC_BIT_POSITION . . . 1880
10.325.2.21 SPARC_PSR_EC_MASK . . . 1880
10.325.2.22 SPARC_PSR_EF_BIT_POSITION . . . 1880
10.325.2.23 SPARC_PSR_EF_MASK . . . 1880
10.325.2.24 SPARC_PSR_ET_BIT_POSITION . . . 1880
10.325.2.25 SPARC_PSR_ET_MASK . . . 1881
10.325.2.26 SPARC_PSR_ICC_BIT_POSITION . . . 1881
10.325.2.27 SPARC_PSR_ICC_MASK . . . 1881
10.325.2.28 SPARC_PSR_IMPL_BIT_POSITION . . . 1881
10.325.2.29 SPARC_PSR_IMPL_MASK . . . 1881
10.325.2.30 SPARC_PSR_PIL_BIT_POSITION . . . 1882
10.325.2.31 SPARC_PSR_PIL_MASK . . . 1882
10.325.2.32 SPARC_PSR_PS_BIT_POSITION . . . 1882
10.325.2.33 SPARC_PSR_PS_MASK . . . 1882
10.325.2.34 SPARC_PSR_S_BIT_POSITION . . . 1882
10.325.2.35 SPARC_PSR_S_MASK . . . 1883
10.325.2.36 SPARC_PSR_VER_BIT_POSITION . . . 1883
10.325.2.37 SPARC_PSR_VER_MASK . . . 1883
10.325.2.38 SPARC_REAL_TRAP_NUMBER . . . 1883
10.325.2.39 sparc_set_psr . . . 1884
10.325.2.40 sparc_set_tbr . . . 1884
10.325.2.41 sparc_set_wim . . . 1884
10.325.2.42 sparc_set_y . . . 1885
10.325.2.43 SPARC_SYNCHRONOUS_TRAP . . . 1885
10.325.3 Function Documentation . . . 1885
10.325.3.1 sparc_disable_interrupts() . . . 1885
10.325.3.2 sparc_enable_interrupts() . . . 1886
10.325.3.3 sparc_syscall_exit() . . . 1886
10.326 cpukit/score/src/apimutexisowner.c File Reference . . . 1886
10.327 cpukit/score/src/apimutexlock.c File Reference . . . 1887

10.327.1 Detailed Description . . . 1887
10.328 cpukit/score/src/apimutexunlock.c File Reference . . . 1887
10.328.1 Detailed Description . . . 1887
10.329 cpukit/score/src/chain.c File Reference . . . 1888
10.329.1 Detailed Description . . . 1888
10.330 cpukit/score/src/corebarrier.c File Reference . . . 1888
10.330.1 Detailed Description . . . 1888
10.331 cpukit/score/src/corebarrierrelease.c File Reference . . . 1888
10.331.1 Detailed Description . . . 1889
10.332 cpukit/score/src/corebarrierwait.c File Reference . . . 1889
10.332.1 Detailed Description . . . 1889
10.333 cpukit/score/src/coremsg.c File Reference . . . 1889
10.333.1 Detailed Description . . . 1890
10.334 cpukit/score/src/coremsgbroadcast.c File Reference . . . 1890
10.334.1 Detailed Description . . . 1890
10.335 cpukit/score/src/coremsgclose.c File Reference . . . 1890
10.335.1 Detailed Description . . . 1891
10.336 cpukit/score/src/coremsgflush.c File Reference . . . 1891
10.336.1 Detailed Description . . . 1891
10.337 cpukit/score/src/coremsginsert.c File Reference . . . 1891
10.337.1 Detailed Description . . . 1891
10.338 cpukit/score/src/coremsgseize.c File Reference . . . 1892
10.338.1 Detailed Description . . . 1892
10.339 cpukit/score/src/coremsgsubmit.c File Reference . . . 1892
10.339.1 Detailed Description . . . 1892
10.340 cpukit/score/src/coremsgwkspace.c File Reference . . . 1893
10.340.1 Detailed Description . . . 1893
10.341 cpukit/score/src/coremutexseize.c File Reference . . . 1893
10.341.1 Detailed Description . . . 1893
10.342 cpukit/score/src/coresem.c File Reference . . . 1893
10.342.1 Detailed Description . . . 1894
10.343 cpukit/score/src/coretod.c File Reference . . . 1894
10.343.1 Detailed Description . . . 1894
10.344 cpukit/score/src/heap.c File Reference . . . 1894
10.344.1 Detailed Description . . . 1895
10.345 cpukit/score/src/heapallocate.c File Reference . . . 1895
10.345.1 Detailed Description . . . 1895
10.346 cpukit/score/src/interr.c File Reference . . . 1896
10.346.1 Detailed Description . . . 1896
10.347 cpukit/score/src/isr.c File Reference . . . 1896
10.347.1 Detailed Description . . . 1896
10.348 cpukit/score/src/isrisinprogress.c File Reference . . . 1897

10.348.1 Detailed Description . . . 1897
10.349 cpukit/score/src/objectallocate.c File Reference . . . 1897
10.349.1 Detailed Description . . . 1897
10.350 cpukit/score/src/objectallocatenone.c File Reference . . . 1897
10.351 cpukit/score/src/objectallocatestatic.c File Reference . . . 1898
10.352 cpukit/score/src/objectapimaximumclass.c File Reference . . . 1898
10.352.1 Detailed Description . . . 1898
10.353 cpukit/score/src/objectclose.c File Reference . . . 1898
10.353.1 Detailed Description . . . 1898
10.354 cpukit/score/src/objectfree.c File Reference . . . 1899
10.354.1 Detailed Description . . . 1899
10.355 cpukit/score/src/objectfreestatic.c File Reference . . . 1899
10.356 cpukit/score/src/objectgetinfo.c File Reference . . . 1899
10.356.1 Detailed Description . . . 1899
10.357 cpukit/score/src/objectgetinfoid.c File Reference . . . 1900
10.357.1 Detailed Description . . . 1900
10.358 cpukit/score/src/objectgetlocal.c File Reference . . . 1900
10.358.1 Detailed Description . . . 1900
10.359 cpukit/score/src/objectgetnameasstring.c File Reference . . . 1900
10.359.1 Detailed Description . . . 1901
10.360 cpukit/score/src/objectgetnoprotection.c File Reference . . . 1901
10.360.1 Detailed Description . . . 1901
10.361 cpukit/score/src/objectinitializeinformation.c File Reference . . . 1901
10.361.1 Detailed Description . . . 1902
10.362 cpukit/score/src/objectnamespaceremove.c File Reference . . . 1902
10.362.1 Detailed Description . . . 1902
10.363 cpukit/score/src/objectnametoid.c File Reference . . . 1902
10.363.1 Detailed Description . . . 1902
10.364 cpukit/score/src/percpu.c File Reference . . . 1903
10.364.1 Detailed Description . . . 1903
10.365 cpukit/score/src/processormaskcopy.c File Reference . . . 1903
10.365.1 Detailed Description . . . 1904
10.366 cpukit/score/src/rbtreenext.c File Reference . . . 1904
10.366.1 Detailed Description . . . 1904
10.367 cpukit/score/src/rbtreereplace.c File Reference . . . 1904
10.367.1 Detailed Description . . . 1904
10.368 cpukit/score/src/scheduler.c File Reference . . . 1905
10.368.1 Detailed Description . . . 1905
10.369 cpukit/score/src/schedulerdefaultnodedestroy.c File Reference . . . 1905
10.369.1 Detailed Description . . . 1905
10.370 cpukit/score/src/schedulerdefaultnodeinit.c File Reference . . . 1905
10.370.1 Detailed Description . . . 1906

10.371 cpukit/score/src/schedulerdefaultreleasejob.c File Reference . . . 1906
10.371.1 Detailed Description . . . 1906
10.372 cpukit/score/src/schedulerdefaultsetaffinity.c File Reference . . . 1906
10.372.1 Detailed Description . . . 1906
10.373 cpukit/score/src/schedulerdefaulttick.c File Reference . . . 1907
10.373.1 Detailed Description . . . 1907
10.374 cpukit/score/src/scheduleredfreleasejob.c File Reference . . . 1907
10.374.1 Detailed Description . . . 1907
10.375 cpukit/score/src/scheduleredfsmp.c File Reference . . . 1908
10.375.1 Detailed Description . . . 1909
10.376 cpukit/score/src/schedulerpriorityblock.c File Reference . . . 1909
10.376.1 Detailed Description . . . 1910
10.377 cpukit/score/src/schedulerprioritychangepriority.c File Reference . . . 1910
10.377.1 Detailed Description . . . 1910
10.378 cpukit/score/src/schedulerpriorityschedule.c File Reference . . . 1910
10.378.1 Detailed Description . . . 1910
10.379 cpukit/score/src/schedulerpriorityunblock.c File Reference . . . 1911
10.379.1 Detailed Description . . . 1911
10.380 cpukit/score/src/schedulerpriorityyield.c File Reference . . . 1911
10.380.1 Detailed Description . . . 1911
10.381 cpukit/score/src/smp.c File Reference . . . 1911
10.381.1 Detailed Description . . . 1912
10.382 cpukit/score/src/stackallocatorfreenothing.c File Reference . . . 1913
10.382.1 Detailed Description . . . 1913
10.383 cpukit/score/src/thread.c File Reference . . . 1913
10.383.1 Detailed Description . . . 1913
10.383.2 Macro Definition Documentation . . . 1914
10.383.2.1 THREAD_OFFSET_ASSERT . . . 1914
10.384 cpukit/score/src/threadchangepriority.c File Reference . . . 1914
10.384.1 Detailed Description . . . 1915
10.385 cpukit/score/src/threadclearstate.c File Reference . . . 1915
10.385.1 Detailed Description . . . 1915
10.386 cpukit/score/src/threadcreateidle.c File Reference . . . 1915
10.386.1 Detailed Description . . . 1916
10.387 cpukit/score/src/threaddispatch.c File Reference . . . 1916
10.387.1 Detailed Description . . . 1916
10.388 cpukit/score/src/threadget.c File Reference . . . 1917
10.388.1 Detailed Description . . . 1917
10.389 cpukit/score/src/threadhandler.c File Reference . . . 1917
10.389.1 Detailed Description . . . 1917
10.390 cpukit/score/src/threadinitialize.c File Reference . . . 1918
10.390.1 Detailed Description . . . 1918

10.391 cpukit/score/src/threadloadenv.c File Reference . . . 1918
10.391.1 Detailed Description . . . 1918
10.392 cpukit/score/src/threadq.c File Reference . . . 1918
10.392.1 Detailed Description . . . 1919
10.393 cpukit/score/src/threadqenqueue.c File Reference . . . 1919
10.393.1 Detailed Description . . . 1921
10.393.2 Variable Documentation . . . 1921
10.393.2.1 _Thread_queue_Links . . . 1921
10.394 cpukit/score/src/threadqflush.c File Reference . . . 1921
10.394.1 Detailed Description . . . 1922
10.395 cpukit/score/src/threadrestart.c File Reference . . . 1922
10.395.1 Detailed Description . . . 1923
10.395.2 Variable Documentation . . . 1924
10.395.2.1 _Thread_Zombies . . . 1924
10.396 cpukit/score/src/threadsetstate.c File Reference . . . 1924
10.396.1 Detailed Description . . . 1924
10.397 cpukit/score/src/threadstackallocate.c File Reference . . . 1924
10.397.1 Detailed Description . . . 1925
10.398 cpukit/score/src/threadstart.c File Reference . . . 1925
10.398.1 Detailed Description . . . 1925
10.399 cpukit/score/src/threadstartmultitasking.c File Reference . . . 1925
10.399.1 Detailed Description . . . 1925
10.400 cpukit/score/src/threadtimeout.c File Reference . . . 1926
10.400.1 Detailed Description . . . 1926
10.401 cpukit/score/src/threadyield.c File Reference . . . 1926
10.401.1 Detailed Description . . . 1926
10.402 cpukit/score/src/userext.c File Reference . . . 1926
10.402.1 Detailed Description . . . 1927
10.403 cpukit/score/src/userextaddset.c File Reference . . . 1927
10.403.1 Detailed Description . . . 1927
10.404 cpukit/score/src/userextiterate.c File Reference . . . 1927
10.404.1 Detailed Description . . . 1928
10.405 cpukit/score/src/userextremoveset.c File Reference . . . 1928
10.405.1 Detailed Description . . . 1929
10.406 cpukit/score/src/watchdoginsert.c File Reference . . . 1929
10.406.1 Detailed Description . . . 1929
10.407 cpukit/score/src/watchdogremove.c File Reference . . . 1929
10.407.1 Detailed Description . . . 1929
10.408 cpukit/score/src/watchdogtickssinceboot.c File Reference . . . 1929
10.408.1 Detailed Description . . . 1930
10.409 cpukit/score/src/wkspace.c File Reference . . . 1930
10.409.1 Detailed Description . . . 1930

10.410 cpukit/score/src/wkspaceallocate.c File Reference . . . 1931
10.410.1 Detailed Description . . . 1931
10.411 cpukit/score/src/wkspacemallocinitdefault.c File Reference . . . 1931
10.411.1 Detailed Description . . . 1931
10.412 cpukit/score/src/wkspacemallocinitunified.c File Reference . . . 1931
10.412.1 Detailed Description . . . 1932
10.413 testsuites/validation/tc-attr.c File Reference . . . 1932
10.414 testsuites/validation/tc-barrier-ident.c File Reference . . . 1932
10.415 testsuites/validation/tc-event-send-receive.c File Reference . . . 1932
10.416 testsuites/validation/tc-events.c File Reference . . . 1933
10.417 testsuites/validation/tc-message-construct-errors.c File Reference . . . 1933
10.418 testsuites/validation/tc-message-ident.c File Reference . . . 1935
10.419 testsuites/validation/tc-modes.c File Reference . . . 1936
10.420 testsuites/validation/tc-options.c File Reference . . . 1936
10.421 testsuites/validation/tc-part-create.c File Reference . . . 1936
10.422 testsuites/validation/tc-part-delete.c File Reference . . . 1938
10.423 testsuites/validation/tc-part-get.c File Reference . . . 1940
10.424 testsuites/validation/tc-part-ident.c File Reference . . . 1941
10.425 testsuites/validation/tc-part-return.c File Reference . . . 1942
10.426 testsuites/validation/tc-part.c File Reference . . . 1943
10.427 testsuites/validation/tc-ratemon-ident.c File Reference . . . 1944
10.428 testsuites/validation/tc-sem-ident.c File Reference . . . 1944
10.429 testsuites/validation/tc-space-profile.c File Reference . . . 1944
10.430 testsuites/validation/tc-task-construct-errors.c File Reference . . . 1944
10.431 testsuites/validation/tc-task-ident.c File Reference . . . 1947
10.432 testsuites/validation/tc-timer-ident.c File Reference . . . 1948
10.433 testsuites/validation/tc-userext-ident.c File Reference . . . 1949
10.434 testsuites/validation/tr-event-constant.c File Reference . . . 1949
10.435 testsuites/validation/tr-event-constant.h File Reference . . . 1949
10.436 testsuites/validation/tr-event-send-receive.c File Reference . . . 1949
10.437 testsuites/validation/tr-event-send-receive.h File Reference . . . 1952
10.438 testsuites/validation/tr-object-ident-local.c File Reference . . . 1953
10.439 testsuites/validation/tr-object-ident-local.h File Reference . . . 1954
10.440 testsuites/validation/tr-object-ident.c File Reference . . . 1954
10.441 testsuites/validation/tr-object-ident.h File Reference . . . 1956
10.442 testsuites/validation/ts-space-profile.c File Reference . . . 1957
10.443 testsuites/validation/ts-validation-0.c File Reference . . . 1958

**Index** **1961**

# Chapter 1

# Main Page

The RTEMS real-time operating systems is a layered system with each of the public APIs implemented in terms of a common foundation layer called the SuperCore. This is the Doxygen generated documentation for the RTEMS CPU Kit including the Classic API, POSIX API and SuperCore.

# Chapter 2

# RTEMS History and Introduction

In recent years, the cost required to develop a software product has increased significantly while the target hardware costs have decreased. Now a larger portion of money is expended in developing, using, and maintaining software. The trend in computing costs is the complete dominance of software over hardware costs. Because of this, it is necessary that formal disciplines be established to increase the probability that software is characterized by a high degree of correctness, maintainability, and portability. In addition, these disciplines must promote practices that aid in the consistent and orderly development of a software system within schedule and budgetary constraints. To be effective, these disciplines must adopt standards which channel individual software efforts toward a common goal.

The push for standards in the software development field has been met with various degrees of success. The Microprocessor Operating Systems Interfaces (MOSI) effort has experienced only limited success. As popular as the UNIX operating system has grown, the attempt to develop a standard interface definition to allow portable application development has only recently begun to produce the results needed in this area. Unfortunately, very little effort has been expended to provide standards addressing the needs of the real-time community. Several organizations have addressed this need during recent years.

The Real Time Executive Interface Definition (RTEID) was developed by Motorola with technical input from Software Components Group. RTEID was adopted by the VMEbus International Trade Association (VITA) as a baseline draft for their proposed standard multiprocessor, real-time executive interface, Open Real-Time Kernel Interface Definition (ORKID). These two groups are currently working together with the IEEE P1003.4 committee to insure that the functionality of their proposed standards is adopted as the real-time extensions to POSIX.

This emerging standard defines an interface for the development of real-time software to ease the writing of real-time application programs that are directly portable across multiple real-time executive implementations. This interface includes both the source code interfaces and run-time behavior as seen by a real-time application. It does not include the details of how a kernel implements these functions. The standard's goal is to serve as a complete definition of external interfaces so that application code that conforms to these interfaces will execute properly in all real-time executive environments. With the use of a standards compliant executive, routines that acquire memory blocks, create and manage message queues, establish and use semaphores, and send and receive signals need not be redeveloped for a different real-time environment as long as the new environment is compliant with the standard. Software developers need only concentrate on the hardware dependencies of the real-time system. Furthermore, most hardware dependencies for real-time applications can be localized to the device drivers.

A compliant executive provides simple and flexible real-time multiprocessing. It easily lends itself to both tightly-coupled and loosely-coupled configurations (depending on the system hardware configuration). Objects such as tasks, queues, events, signals, semaphores, and memory blocks can be designated as global objects and accessed by any task regardless of which processor the object and the accessing task reside.

The acceptance of a standard for real-time executives will produce the same advantages enjoyed from the push for UNIX standardization by AT&T's System V Interface Definition and IEEE's POSIX efforts. A compliant multiprocessing executive will allow close coupling between UNIX systems and real-time executives to provide the many benefits of the UNIX development environment to be applied to real-time software development. Together they provide the

necessary laboratory environment to implement real-time, distributed, embedded systems using a wide variety of computer architectures.

A study was completed in 1988, within the Research, Development, and Engineering Center, U.S. Army Missile Command, which compared the various aspects of the Ada programming language as they related to the application of Ada code in distributed and/or multiple processing systems. Several critical conclusions were derived from the study. These conclusions have a major impact on the way the Army develops application software for embedded applications. These impacts apply to both in-house software development and contractor developed software.

A conclusion of the analysis, which has been previously recognized by other agencies attempting to utilize Ada in a distributed or multiprocessing environment, is that the Ada programming language does not adequately support multiprocessing. Ada does provide a mechanism for multi-tasking, however, this capability exists only for a single processor system. The language also does not have inherent capabilities to access global named variables, flags or program code. These critical features are essential in order for data to be shared between processors. However, these drawbacks do have workarounds which are sometimes awkward and defeat the intent of software maintainability and portability goals.

Another conclusion drawn from the analysis, was that the run time executives being delivered with the Ada compilers were too slow and inefficient to be used in modern missile systems. A run time executive is the core part of the run time system code, or operating system code, that controls task scheduling, input/output management and memory management. Traditionally, whenever efficient executive (also known as kernel) code was required by the application, the user developed in-house software. This software was usually written in assembly language for optimization.

Because of this shortcoming in the Ada programming language, software developers in research and development and contractors for project managed systems, are mandated by technology to purchase and utilize off-the-shelf third party kernel code. The contractor, and eventually the Government, must pay a licensing fee for every copy of the kernel code used in an embedded system.

The main drawback to this development environment is that the Government does not own, nor has the right to modify code contained within the kernel. V&V techniques in this situation are more difficult than if the complete source code were available. Responsibility for system failures due to faulty software is yet another area to be resolved under this environment.

The Guidance and Control Directorate began a software development effort to address these problems. A project to develop an experimental run time kernel was begun that will eliminate the major drawbacks of the Ada programming language mentioned above. The Real Time Executive for Multiprocessor Systems (RTEMS) provides full capabilities for management of tasks, interrupts, time, and multiple processors in addition to those features typical of generic operating systems. The code is Government owned, so no licensing fees are necessary. RTEMS has been implemented in both the Ada and C programming languages. It has been ported to the following processor families:

- Altera NIOS II
- Analog Devices Blackfin
- ARM
- Freescale (formerly Motorola) MC68xxx
- Freescale (formerly Motorola) MC683xx
- Freescale (formerly Motorola) ColdFire
- Intel i386 and above
- Lattice Semiconductor LM32
- MIPS
- PowerPC

- Renesas (formerly Hitachi) SuperH
- Renesas (formerly Hitachi) H8/300
- SPARC
- Texas Instruments C3x/C4x
- UNIX

Support for other processor families, including RISC, CISC, and DSP, is planned. Since almost all of RTEMS is written in a high level language, ports to additional processor families require minimal effort.

RTEMS multiprocessor support is capable of handling either homogeneous or heterogeneous systems. The kernel automatically compensates for architectural differences (byte swapping, etc.) between processors. This allows a much easier transition from one processor family to another without a major system redesign.

Since the proposed standards are still in draft form, RTEMS cannot and does not claim compliance. However, the status of the standard is being carefully monitored to guarantee that RTEMS provides the functionality specified in the standard. Once approved, RTEMS will be made compliant.

# Chapter 3

# RTEMS Overview

## 3.1 Introduction

RTEMS, Real-Time Executive for Multiprocessor Systems, is a real-time executive (kernel) which provides a high performance environment for embedded military applications including the following features:

- multitasking capabilities
- homogeneous and heterogeneous multiprocessor systems
- event-driven, priority-based, preemptive scheduling
- optional rate monotonic scheduling
- intertask communication and synchronization
- priority inheritance
- responsive interrupt management
- dynamic memory allocation
- high level of user configurability

This manual describes the usage of RTEMS for applications written in the C programming language. Those implementation details that are processor dependent are provided in the Applications Supplement documents. A supplement document which addresses specific architectural issues that affect RTEMS is provided for each processor type that is supported.

## 3.2 Real-time Application Systems

Real-time application systems are a special class of computer applications. They have a complex set of characteristics that distinguish them from other software problems. Generally, they must adhere to more rigorous requirements. The correctness of the system depends not only on the results of computations, but also on the time at which the results are produced. The most important and complex characteristic of real-time application systems is that they must receive and respond to a set of external stimuli within rigid and critical time constraints referred to as deadlines. Systems can be buried by an avalanche of interdependent, asynchronous or cyclical event streams.

Deadlines can be further characterized as either hard or soft based upon the value of the results when produced after the deadline has passed. A deadline is hard if the results have no value or if their use will result in a catastrophic event. In contrast, results which are produced after a soft deadline may have some value.

Another distinguishing requirement of real-time application systems is the ability to coordinate or manage a large number of concurrent activities. Since software is a synchronous entity, this presents special problems. One instruction follows another in a repeating synchronous cycle. Even though mechanisms have been developed to allow for the processing of external asynchronous events, the software design efforts required to process and manage these events and tasks are growing more complicated.

The design process is complicated further by spreading this activity over a set of processors instead of a single processor. The challenges associated with designing and building real-time application systems become very complex when multiple processors are involved. New requirements such as interprocessor communication channels and global resources that must be shared between competing processors are introduced. The ramifications of multiple processors complicate each and every characteristic of a real-time system.

## 3.3 Real-time Executive

Fortunately, real-time operating systems or real-time executives serve as a cornerstone on which to build the application system. A real-time multitasking executive allows an application to be cast into a set of logical, autonomous processes or tasks which become quite manageable. Each task is internally synchronous, but different tasks execute independently, resulting in an asynchronous processing stream. Tasks can be dynamically paused for many reasons resulting in a different task being allowed to execute for a period of time. The executive also provides an interface to other system components such as interrupt handlers and device drivers. System components may request the executive to allocate and coordinate resources, and to wait for and trigger synchronizing conditions. The executive system calls effectively extend the CPU instruction set to support efficient multitasking. By causing tasks to travel through well-defined state transitions, system calls permit an application to demand-switch between tasks in response to real-time events.

By proper grouping of responses to stimuli into separate tasks, a system can now asynchronously switch between independent streams of execution, directly responding to external stimuli as they occur. This allows the system design to meet critical performance specifications which are typically measured by guaranteed response time and transaction throughput. The multiprocessor extensions of RTEMS provide the features necessary to manage the extra requirements introduced by a system distributed across several processors. It removes the physical barriers of processor boundaries from the world of the system designer, enabling more critical aspects of the system to receive the required attention. Such a system, based on an efficient real-time, multiprocessor executive, is a more realistic model of the outside world or environment for which it is designed. As a result, the system will always be more logical, efficient, and reliable.

By using the directives provided by RTEMS, the real-time applications developer is freed from the problem of controlling and synchronizing multiple tasks and processors. In addition, one need not develop, test, debug, and document routines to manage memory, pass messages, or provide mutual exclusion. The developer is then able to concentrate solely on the application. By using standard software components, the time and cost required to develop sophisticated real-time applications is significantly reduced.

## 3.4 RTEMS Application Architecture

One important design goal of RTEMS was to provide a bridge between two critical layers of typical real-time systems. As shown in the following figure, RTEMS serves as a buffer between the project dependent application code and the target hardware. Most hardware dependencies for real-time applications can be localized to the low level device drivers.

**Todo** Image RTEMS Application Architecture

The RTEMS I/O interface manager provides an efficient tool for incorporating these hardware dependencies into the system while simultaneously providing a general mechanism to the application code that accesses them. A well designed real-time system can benefit from this architecture by building a rich library of standard application components which can be used repeatedly in other real-time projects.

## 3.5 RTEMS Internal Architecture

RTEMS can be viewed as a set of layered components that work in harmony to provide a set of services to a real-time application system. The executive interface presented to the application is formed by grouping directives into logical sets called resource managers. Functions utilized by multiple managers such as scheduling, dispatching, and object management are provided in the executive core. The executive core depends on a small set of CPU dependent routines. Together these components provide a powerful run time environment that promotes the development of efficient real-time application systems. The following figure illustrates this organization:

**Todo** Image RTEMS Architecture

Subsequent chapters present a detailed description of the capabilities provided by each of the following RTEMS managers:

- initialization
- task
- interrupt
- clock
- timer
- semaphore
- message
- event
- signal
- partition
- region
- dual ported memory
- I/O
- fatal error
- rate monotonic
- user extensions
- multiprocessing

## 3.6 User Customization and Extensibility

As 32-bit microprocessors have decreased in cost, they have become increasingly common in a variety of embedded systems. A wide range of custom and general-purpose processor boards are based on various 32-bit processors. RTEMS was designed to make no assumptions concerning the characteristics of individual microprocessor families or of specific support hardware. In addition, RTEMS allows the system developer a high degree of freedom in customizing and extending its features.

RTEMS assumes the existence of a supported microprocessor and sufficient memory for both RTEMS and the real-time application. Board dependent components such as clocks, interrupt controllers, or I/O devices can be easily integrated with RTEMS. The customization and extensibility features allow RTEMS to efficiently support as many environments as possible.

## 3.7 Portability

The issue of portability was the major factor in the creation of RTEMS. Since RTEMS is designed to isolate the hardware dependencies in the specific board support packages, the real-time application should be easily ported to any other processor. The use of RTEMS allows the development of real-time applications which can be completely independent of a particular microprocessor architecture.

## 3.8 Memory Requirements

Since memory is a critical resource in many real-time embedded systems, RTEMS was specifically designed to automatically leave out all services that are not required from the run-time environment. Features such as networking, various filesystems, and many other features are completely optional. This allows the application designer the flexibility to tailor RTEMS to most efficiently meet system requirements while still satisfying even the most stringent memory constraints. As a result, the size of the RTEMS executive is application dependent.

RTEMS requires RAM to manage each instance of an RTEMS object that is created. Thus the more RTEMS objects an application needs, the more memory that must be reserved. See Configuring a System Determining Memory Requirements for more details.

**Todo** Link to Configuring a SystemDetermining Memory Requirements

RTEMS utilizes memory for both code and data space. Although RTEMS' data space must be in RAM, its code space can be located in either ROM or RAM.

## 3.9 Audience

This manual was written for experienced real-time software developers. Although some background is provided, it is assumed that the reader is familiar with the concepts of task management as well as intertask communication and synchronization. Since directives, user related data structures, and examples are presented in C, a basic understanding of the C programming language is required to fully understand the material presented. However, because of the similarity of the Ada and C RTEMS implementations, users will find that the use and behavior of the two implementations is very similar. A working knowledge of the target processor is helpful in understanding some of RTEMS' features. A thorough understanding of the executive cannot be obtained without studying the entire manual because many of RTEMS' concepts and features are interrelated. Experienced RTEMS users will find that the manual organization facilitates its use as a reference document.

# Chapter 4

# Todo List

**Class rtems_libio_tt**

Should really have a separate per/file data structure that this points to (eg: offset, driver, pathname should be in that)

**Page RTEMS Overview**

Image RTEMS Application Architecture

Image RTEMS Architecture

Link to Configuring a SystemDetermining Memory Requirements

**Module RTEMSAPIClassic**

Link to Clock Manager Time and Date Data Structures

# Chapter 5

# Module Index

## 5.1 Modules

Here is a list of all modules:


API . . . 48
Application Configuration Information . . . 58
Application Configuration Options . . . 72
BSP Related Configuration Options . . . 101
Block Device Cache Configuration . . . 133
Classic API Configuration . . . 203
Classic API Initialization Task Configuration . . . 211
Device Driver Configuration . . . 268
Event Recording Configuration . . . 325
Filesystem Configuration . . . 375
General Scheduler Configuration . . . 402
General System Configuration . . . 411
Idle Task Configuration . . . 491
Multiprocessing Configuration . . . 568
POSIX API Configuration . . . 640
POSIX Initialization Thread Configuration . . . 647
Task Stack Allocator Configuration . . . 983
Associativity Routines . . . 74
Base Definitions . . . 114
Classic . . . 198
Barrier Manager . . . 111
Basic Types . . . 131
Cache Manager . . . 146
Chains . . . 176
Clock Manager . . . 245
Directive Attributes . . . 280
Directive Options . . . 284
Directive Status Codes . . . 285
Dual-Ported Memory Manager . . . 292
Event Manager . . . 320
Fatal Error Manager . . . 328
Free-Running Counter and Busy Wait Delay . . . 391
I/O Manager . . . 459
Initialization and Shutdown . . . 494
Interrupt Manager . . . 501

Interrupt Manager Extension . . . 509
Local Packages . . . 531
Message Manager . . . 545
Object Services . . . 632
Partition Manager . . . 650
Rate-Monotonic Manager . . . 748
Region Manager . . . 770
Semaphore Manager . . . 898
Signal Manager . . . 910
Support Services . . . 948
Task Manager . . . 957
Task Modes . . . 980
Timecounter Support . . . 1151
Timer Manager . . . 1157
User Extensions Manager . . . 1195
Version . . . 1198
IO . . . 467
Print Support . . . 660
Kernel Print Support . . . 526
RTEMS Print Support . . . 717
RTEMS Test Framework . . . 724
Boolean Checks . . . 141
Character Checks . . . 196
Destructors . . . 256
Generic Checks . . . 423
Memory Area Checks . . . 535
POSIX Status and Error Number Checks . . . 649
Pointer Checks . . . 659
RTEMS Status Code Checks . . . 722
RTEMS Test Framework Implementation . . . 726
Runtime Measurements . . . 776
Signed 16-Bit Integer Checks . . . 914
Signed 32-Bit Integer Checks . . . 915
Signed 64-Bit Integer Checks . . . 916
Signed 8-Bit Integer Checks . . . 917
Signed Character Checks . . . 918
Signed Integer Checks . . . 919
Signed Long Integer Checks . . . 920
Signed Pointer Value Checks . . . 922
Signed Short Integer Checks . . . 923
Signed Size Checks . . . 924
Size Checks . . . 931
String Checks . . . 944
Time Services . . . 1125
Unsigned 16-Bit Integer Checks . . . 1165
Unsigned 32-Bit Integer Checks . . . 1166
Unsigned 64-Bit Integer Checks . . . 1167
Unsigned 8-Bit Integer Checks . . . 1168
Unsigned Character Checks . . . 1169
Unsigned Integer Checks . . . 1170
Unsigned Long Integer Checks . . . 1171
Unsigned Long Long Integer Checks . . . 1172
Unsigned Pointer Value Checks . . . 1173
Unsigned Short Integer Checks . . . 1174
Test Support . . . 991
Tracing . . . 1163
Board Support Packages . . . 140
SPARC . . . 834

LEON3 and LEON4 . . . 528
LEON3 AMBA Driver Handler . . . 527
Shared . . . 909
Shared . . . 908
BSP Interrupt Support . . . 95
Bootcard . . . 143
Clock Support . . . 249
Console Driver Support . . . 250
DEFAULT_INITIAL_EXTENSION Support . . . 255
GRLIB . . . 401
AMBA . . . 45
LEON3 AMBA Driver Handler . . . 527
UART . . . 1164
Device Drivers . . . 279
Clock Driver . . . 244
Legacy Benchmark Drivers . . . 530
Serial Mouse . . . 907
Time Test 27 Support . . . 1126
Implementation . . . 493
Application Configuration . . . 56
Classic . . . 197
Classic ASR Implementation . . . 215
Classic Attributes Implementation . . . 216
Classic Barrier Implementation . . . 221
Classic Message Queue Implementation . . . 223
Classic Modes Implementation . . . 226
Classic Object Implementation . . . 229
Classic Options Implementation . . . 230
Classic Rate Monotonic Scheduler Implementation . . . 231
Classic Region Manager Implementation . . . 234
Classic Status Implementation . . . 237
Classic Tasks Manager Implementation . . . 238
Classic Timer Implementation . . . 241
Dual Ported Memory Manager Implementation . . . 290
Event Implementation . . . 315
Partition Manager Implementation . . . 656
Semaphore Manager Implementation . . . 904
Signals Implementation . . . 913
User Extensions Implementation . . . 1193
IO Library . . . 468
File System Node Handler . . . 331
File System Operations . . . 349
File System Types and Mount . . . 368
libfs . . . 1223
nfs Client . . . 1224
Termios . . . 986
Standard C Library Support . . . 939
Malloc Support . . . 534
RTEMS Termios Device Support . . . 723
SuperCore . . . 945
API Mutex Handler . . . 49
RTEMS Allocator Mutex . . . 708
Address Handler . . . 51
Assert Handler . . . 73
Atomic Operations . . . 77
Atomic Operations CPU . . . 79
Barrier Handler . . . 105

Bitmap Priority Thread Routines . . . 132
CPU Architecture Support . . . 145
Object Handler . . . 592
SPARC . . . 828
SPARC Assembler Support . . . 835
SPARC Context Structures . . . 837
Chain Handler . . . 151
Context Handler . . . 251
Freechain Handler . . . 397
Heap Handler . . . 424
Protected Heap Handler . . . 700
Helpers . . . 451
ISR Handler . . . 474
ISR Locks . . . 482
Internal Error Handler . . . 496
MP Packet Handler . . . 532
Memory Handler . . . 536
Message Queue Handler . . . 552
Multiprocessor Resource Sharing Protocol Handler . . . 572
Mutex Handler . . . 581
Priority Handler . . . 661
Processor Mask . . . 685
Profiling Support . . . 697
RTEMS Copyright Notice . . . 709
RTEMS Per CPU Information . . . 710
Flexible Per-CPU Data . . . 388
Red-Black Tree Handler . . . 753
SMP Barriers . . . 777
SMP Locks . . . 779
SMP Support . . . 819
Scheduler Handler . . . 847
Deterministic Priority Scheduler . . . 257
EDF Scheduler . . . 304
SMP Scheduler . . . 793
EDF Priority SMP Scheduler . . . 295
Simple Priority Scheduler . . . 925
Score Timestamp . . . 885
Semaphore Handler . . . 892
Stack Handler . . . 932
System State Handler . . . 953
Thread Handler . . . 997
Thread Queue Handler . . . 1069
Thread States . . . 1107
Thread-Local Storage (TLS) . . . 1119
Time of Day Handler . . . 1127
Time of Day Handler Action Hooks . . . 1137
Timecounter Handler . . . 1140
User Extension Handler . . . 1175
Watchdog Handler . . . 1201
Workspace Handler . . . 1219
Test Suites . . . 990
spec:/testsuites/validation-0 . . . 1288
spec:/rtems/attr/val/attr . . . 1225
spec:/rtems/barrier/val/ident . . . 1226
spec:/rtems/event/req/send-receive . . . 1227
spec:/rtems/event/val/event-constant . . . 1233
spec:/rtems/event/val/events . . . 1237

spec:/rtems/event/val/send-receive . . . 1238
spec:/rtems/event/val/system-send-receive . . . 1239
spec:/rtems/message/req/construct-errors . . . 1240
spec:/rtems/message/val/ident . . . 1245
spec:/rtems/mode/val/modes . . . 1246
spec:/rtems/option/val/options . . . 1247
spec:/rtems/part/req/create . . . 1248
spec:/rtems/part/req/delete . . . 1252
spec:/rtems/part/req/get-buffer . . . 1256
spec:/rtems/part/req/return-buffer . . . 1260
spec:/rtems/part/val/ident . . . 1264
spec:/rtems/part/val/part . . . 1265
spec:/rtems/ratemon/val/ident . . . 1266
spec:/rtems/req/ident . . . 1267
spec:/rtems/req/ident-local . . . 1271
spec:/rtems/sem/val/ident . . . 1275
spec:/rtems/task/req/construct-errors . . . 1276
spec:/rtems/task/req/ident . . . 1282
spec:/rtems/timer/val/ident . . . 1286
spec:/rtems/userext/val/ident . . . 1287
spec:/testsuites/validation/profile . . . 1294
spec:/testsuites/validation/c-library . . . 1292
spec:/testsuites/validation/classic-barrier . . . 1293

# Chapter 6

# Class Index

## 6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:


__rtems_dev_t . . . 1297
_rtems_filesystem_file_handlers_r
File system node operations table . . . 1297
_rtems_filesystem_operations_table
File system operations table . . . 1298
_Scheduler_Control
Scheduler control . . . 1299
_Thread_Control . . . 1300
_Thread_queue_Heads
Thread queue heads . . . 1305
ambapp_ahb_bus . . . 1306
ambapp_ahb_info . . . 1306
ambapp_apb_info . . . 1307
ambapp_bus . . . 1307
ambapp_common_info . . . 1307
ambapp_core . . . 1308
ambapp_dev . . . 1308
ambapp_mmap . . . 1309
ambapp_pnp_ahb . . . 1309
ambapp_pnp_apb . . . 1309
apbuart_regs . . . 1310
API_Mutex_Control
Control block used to manage each API mutex . . . 1310
ASR_Information . . . 1311
Barrier_Control . . . 1312
bsp_interrupt_handler_entry . . . 1314
Chain_Control . . . 1314
Chain_Iterator
A chain iterator which is updated during node extraction if it is properly registered . . . 1315
Chain_Iterator_registry
A registry for chain iterators . . . 1316
Chain_Node_struct . . . 1317
Context_Control
SPARC basic context . . . 1318
Context_Control_fp
SPARC basic context . . . 1323

CORE_barrier_Attributes . . . 1327
CORE_barrier_Control . . . 1328
CORE_ceiling_mutex_Control
The recursive mutex control with priority ceiling protocol support . . . 1329
CORE_message_queue_Buffer
The structure is used to organize message buffers of a message queue . . . 1330
CORE_message_queue_Control
Control block used to manage each message queue . . . 1330
CORE_mutex_Control
Control block used to manage each mutex . . . 1333
CORE_recursive_mutex_Control
The recursive mutex control . . . 1334
CORE_semaphore_Control . . . 1334
CPU_Exception_frame . . . 1335
CPU_Interrupt_frame
Interrupt stack frame (ISF) . . . 1335
CPU_Per_CPU_control . . . 1341
CPU_Trap_table_entry . . . 1341
Dual_ported_memory_Control . . . 1342
Event_Control
This structure is used to manage a set of events . . . 1344
Extension_Control . . . 1344
ffclock_estimate . . . 1344
Freechain_Control
The freechain control . . . 1345
gptimer_regs . . . 1345
gptimer_timer_regs . . . 1346
grgpio_regs . . . 1346
Heap_Area
Heap area structure for table based heap initialization and extension . . . 1347
Heap_Block
Description for free or used blocks . . . 1347
Heap_Control
Control block used to manage a heap . . . 1349
Heap_Error_context
Context of a heap error . . . 1350
Heap_Information
Information about blocks . . . 1350
Heap_Information_block
Information block returned by _Heap_Get_information() . . . 1351
Heap_Statistics
Run-time heap statistics . . . 1351
Internal_errors_Information . . . 1354
irqmp_regs . . . 1355
irqmp_timestamp_regs . . . 1356
ISR_lock_Context
Local ISR lock context for acquire and release pairs . . . 1356
ISR_lock_Control
ISR lock control . . . 1356
l2c_regs . . . 1357
load_context . . . 1358
mctrl_regs . . . 1358
Memory_Area
The memory area description . . . 1358
Memory_Information
The memory information . . . 1359
Message_queue_Control . . . 1359
MP_packet_Prefix . . . 1360

MRSP_Control
MrsP control block . . . 1363
Mutex_Control . . . 1363
Mutex_recursive_Control . . . 1364
ntp_fp . . . 1364
ntptimeval . . . 1364
Objects_Control . . . 1365
Objects_Information
The information structure used to manage each API class of objects . . . 1366
Objects_Name . . . 1370
Partition_Control
This structure is the Partition Control Block (PTCB) . . . 1371
Per_CPU_Control
Per CPU Core Structure . . . 1372
Per_CPU_Control_envelope . . . 1379
Per_CPU_Job
A per-processor job . . . 1379
Per_CPU_Job_context
Context for per-processor jobs . . . 1380
Per_CPU_Stats
Per-CPU statistics . . . 1381
POSIX_Spinlock_Control . . . 1381
pps_fetch_args . . . 1382
pps_fetch_ffc_args . . . 1382
pps_info_ffc_t . . . 1383
pps_info_t . . . 1383
pps_kcbind_args . . . 1383
pps_params_t . . . 1384
pps_timeu . . . 1384
Priority_Actions
A list of priority actions . . . 1384
Priority_Aggregation
The priority aggregation . . . 1385
Priority_bit_map_Control . . . 1386
Priority_bit_map_Information . . . 1387
Priority_Node
The priority node to build up a priority aggregation . . . 1388
Rate_monotonic_Control
The following structure defines the control block used to manage each period . . . 1389
Rate_monotonic_Statistics . . . 1392
RBTree_Node
Red-black tree node . . . 1394
Region_Control . . . 1394
rtems_api_configuration_table
This structure contains a summary of the Classic API configuration . . . 1395
RTEMS_API_Control . . . 1399
rtems_assert_context
% . . . 1400
rtems_assoc_32_pair . . . 1401
rtems_assoc_t . . . 1402
rtems_binary_semaphore . . . 1402
rtems_driver_address_table
This structure contains the device driver entries . . . 1402
rtems_filesystem_eval_path_context_t
Path evaluation context . . . 1404
rtems_filesystem_global_location_t
Global file system location . . . 1407

rtems_filesystem_limits_and_options_t
Contain file system specific information which is required to support fpathconf() . . . . . 1408
rtems_filesystem_location_info_tt
File system location . . . . . 1409
rtems_filesystem_mount_configuration . . . . . 1409
rtems_filesystem_mount_table_entry_tt
Mount table entry . . . . . 1410
rtems_filesystem_table_t
File system table entry . . . . . 1411
rtems_initialization_tasks_table
% . . . . . 1411
rtems_interrupt_server_action
An interrupt server action . . . . . 1414
rtems_interrupt_server_config
An interrupt server configuration . . . . . 1414
rtems_interrupt_server_control
An interrupt server control . . . . . 1416
rtems_interrupt_server_entry
An interrupt server entry . . . . . 1417
rtems_interrupt_server_request
An interrupt server request . . . . . 1417
rtems_libio_ioctl_args_t
Parameter block for ioctl . . . . . 1418
rtems_libio_open_close_args_t
Parameter block for open/close . . . . . 1418
rtems_libio_rw_args_t
Paramameter block for read/write . . . . . 1419
rtems_libio_tt
An open file data structure . . . . . 1419
rtems_message_queue_config
This structure defines the configuration of a message queue constructed by rtems_message_queue_construct() 1420
rtems_object_api_class_information
% . . . . . 1421
rtems_printer . . . . . 1423
rtems_printer_task_context . . . . . 1424
rtems_rate_monotonic_period_statistics
% . . . . . 1424
rtems_rate_monotonic_period_status
% . . . . . 1427
rtems_resource_posix_api . . . . . 1429
rtems_resource_rtems_api . . . . . 1429
rtems_resource_snapshot . . . . . 1430
rtems_sysinit_item . . . . . 1430
rtems_task_config
This structure defines the configuration of a task constructed by rtems_task_construct() . . . . 1431
rtems_termios_callbacks . . . . . 1432
rtems_termios_device_context
Termios device context . . . . . 1433
rtems_termios_device_flow
Termios device flow control handler . . . . . 1433
rtems_termios_device_handler
Termios device handler . . . . . 1435
rtems_termios_device_node
Termios device node for installed devices . . . . . 1438
rtems_termios_linesw . . . . . 1439
rtems_termios_rawbuf . . . . . 1439
rtems_termios_tty . . . . . 1439

rtems_test_parallel_context
Internal context for parallel job execution . . . 1441
rtems_test_parallel_job
Basic parallel job description . . . 1441
rtems_time_of_day
This type represents Classic API calendar times . . . 1443
rtems_timecounter_simple
Simple timecounter to support legacy clock drivers . . . 1444
rtems_timer_information
% . . . 1445
RtemsEventReqSendReceive_Context
Test context for spec:/rtems/event/req/send-receive test case . . . 1446
RtemsMessageReqConstructErrors_Context
Test context for spec:/rtems/message/req/construct-errors test case . . . 1448
RtemsPartReqCreate_Context
Test context for spec:/rtems/part/req/create test case . . . 1448
RtemsPartReqDelete_Context
Test context for spec:/rtems/part/req/delete test case . . . 1449
RtemsPartReqGetBuffer_Context
Test context for spec:/rtems/part/req/get-buffer test case . . . 1449
RtemsPartReqReturnBuffer_Context
Test context for spec:/rtems/part/req/return-buffer test case . . . 1450
RtemsReqIdent_Context
Test context for spec:/rtems/req/ident test case . . . 1450
RtemsReqIdentLocal_Context
Test context for spec:/rtems/req/ident-local test case . . . 1451
RtemsTaskReqConstructErrors_Context
Test context for spec:/rtems/task/req/construct-errors test case . . . 1452
RtemsTaskReqIdent_Context
Test context for spec:/rtems/task/req/ident test case . . . 1452
Scheduler_Assignment
Scheduler assignment . . . 1453
Scheduler_Context
Scheduler context . . . 1454
Scheduler_EDF_Context . . . 1455
Scheduler_EDF_Node
Scheduler node specialization for EDF schedulers . . . 1455
Scheduler_EDF_SMP_Context . . . 1456
Scheduler_EDF_SMP_Node . . . 1457
Scheduler_EDF_SMP_Ready_queue . . . 1458
Scheduler_Node
Scheduler node for per-thread data . . . 1458
Scheduler_Operations
The scheduler operations . . . 1461
Scheduler_priority_Context . . . 1469
Scheduler_priority_Node
Scheduler node specialization for Deterministic Priority schedulers . . . 1470
Scheduler_priority_Ready_queue
Data for ready queue operations . . . 1470
Scheduler_Processor_removal_context . . . 1471
Scheduler_simple_Context
Simple scheduler context . . . 1471
Scheduler_SMP_Context
Scheduler context specialization for SMP schedulers . . . 1472
Scheduler_SMP_Node
Scheduler node specialization for SMP schedulers . . . 1473
Sem_Control . . . 1473
Semaphore_Control . . . 1474

SHA256Context . . . 1475
SMP_barrier_Control
SMP barrier control . . . 1476
SMP_barrier_State
SMP barrier per-thread state . . . 1476
SMP_lock_Context
Local SMP lock context for acquire and release pairs . . . 1477
SMP_lock_Control
SMP lock control . . . 1477
SMP_MCS_lock_Context
SMP Mellor-Crummey and Scott (MCS) lock context . . . 1477
SMP_MCS_lock_Control
SMP Mellor-Crummey and Scott (MCS) lock control . . . 1479
SMP_Multicast_jobs . . . 1480
SMP_sequence_lock_Control
SMP sequence lock control . . . 1480
SMP_ticket_lock_Control
SMP ticket lock control . . . 1481
SPARC_Counter . . . 1481
SPARC_Minimum_stack_frame . . . 1482
Stack_Control . . . 1487
statvfs . . . 1488
T_case_context . . . 1491
T_check_context . . . 1491
T_check_context_msg . . . 1492
T_config . . . 1492
T_context . . . 1493
T_destructor . . . 1493
T_fixture . . . 1494
T_fixture_node . . . 1494
T_interrupt_clock_time . . . 1494
T_interrupt_context . . . 1495
T_interrupt_test_config . . . 1495
T_measure_runtime_config . . . 1496
T_measure_runtime_context . . . 1496
T_measure_runtime_request . . . 1497
T_putchar_string_context . . . 1497
T_report_hash_sha256_context . . . 1497
T_thread_switch_context . . . 1498
T_thread_switch_event . . . 1498
T_thread_switch_log . . . 1499
T_thread_switch_log_10 . . . 1499
T_thread_switch_log_2 . . . 1499
T_thread_switch_log_4 . . . 1500
Thread_Action
Thread action . . . 1500
Thread_Action_control
Control block to manage thread actions . . . 1501
Thread_Capture_control . . . 1501
Thread_Close_context . . . 1501
Thread_Configuration
The configuration of a new thread to initialize . . . 1502
Thread_Control_add_on
Thread control add-on . . . 1503
Thread_Entry_idle
Data for idle thread entry . . . 1503
Thread_Entry_information
Thread entry information . . . 1504

Thread_Entry_numeric
Data for thread entry with one numeric argument and no return value . . . 1505
Thread_Entry_pointer
Data for thread entry with one pointer argument and a pointer return value . . . 1505
Thread_Information
The thread object information . . . 1506
Thread_Join_context . . . 1507
Thread_Keys_information
Per-thread information for POSIX Keys . . . 1507
Thread_Life_control
Thread life control . . . 1508
Thread_Proxy_control . . . 1509
Thread_queue_Context
Thread queue context for the thread queue methods . . . 1510
Thread_queue_Control . . . 1513
Thread_queue_Gate
The thread queue gate is an SMP synchronization means . . . 1513
Thread_queue_Link
A thread queue link from one thread to another specified by the thread queue owner and thread wait queue relationships . . . 1514
Thread_queue_Links . . . 1515
Thread_queue_Lock_context . . . 1515
Thread_queue_Object
Helper structure to ensure that all objects containing a thread queue have the right layout . . . 1516
Thread_queue_Operations
Thread queue operations . . . 1516
Thread_queue_Priority_queue
Thread priority queue . . . 1518
Thread_queue_Queue . . . 1519
Thread_queue_Syslock_queue
Thread queue with a layout compatible to struct _Thread_queue_Queue defined in Newlib <sys/lock.h> . . . 1520
Thread_Scheduler_control
Thread scheduler control . . . 1520
Thread_Start_information . . . 1523
Thread_Timer_information
Information required to manage a thread timer . . . 1525
Thread_Wait_information
Information required to manage a thread while it is blocked . . . 1526
Thread_Wait_information_Object_argument_type
Union type to hold a pointer to an immutable or a mutable object . . . 1529
Thread_Zombie_control . . . 1530
timecounter . . . 1530
timehands . . . 1530
Timer_Control . . . 1531
Timer_server_Control . . . 1533
timex . . . 1533
TLS_Dynamic_thread_vector . . . 1534
TLS_Index . . . 1534
TLS_Thread_control_block . . . 1534
TOD_Control
TOD control . . . 1535
TOD_Hook
Structure to manage each TOD action hook . . . 1536
ttywakeup . . . 1537
User_extensions_Control
Manages each user extension set . . . 1537
User_extensions_Fatal_context . . . 1537

User_extensions_Iterator
Chain iterator for dynamic user extensions . . . . . 1538
User_extensions_List . . . . . 1538
User_extensions_Switch_control
Manages the switch callouts . . . . . 1539
User_extensions_Table
User extension table . . . . . 1539
User_extensions_Thread_create_context . . . . . 1540
Watchdog_Control
The control block used to manage each watchdog timer . . . . . 1540
Watchdog_Header
The watchdog header to manage scheduled watchdogs . . . . . 1541

# Chapter 7

# File Index

## 7.1 File List

Here is a list of all documented files with brief descriptions:


bsps/include/bsp/bootcard.h . . . 1543
bsps/include/bsp/default-initial-extension.h
DEFAULT_INITIAL_EXTENSION Support . . . 1543
bsps/include/bsp/**fatal.h** . . . **??**
bsps/include/bsp/irq-generic.h
Generic BSP interrupt support API . . . 1545
bsps/include/grlib/ambapp.h . . . 1546
bsps/include/grlib/ambapp_ids.h
AMBA Plug & Play Bus Vendor and Device IDs . . . 1548
bsps/include/grlib/apbuart.h . . . 1553
bsps/include/grlib/grlib.h
Common GRLIB AMBA Core Register definitions . . . 1554
bsps/include/grlib/**grlib_impl.h** . . . **??**
bsps/shared/**doxygen.h** . . . **??**
bsps/shared/**rtems-version.c** . . . **??**
bsps/shared/cache/**cacheimpl.h** . . . **??**
bsps/shared/dev/clock/clockimpl.h
Clock Tick Device Driver Shell . . . 1555
bsps/shared/grlib/amba/**ambapp.c** . . . **??**
bsps/shared/grlib/amba/**ambapp_find_by_idx.c** . . . **??**
bsps/shared/grlib/amba/**ambapp_freq.c** . . . **??**
bsps/shared/grlib/uart/**apbuart_polled.c** . . . **??**
bsps/shared/irq/**irq-default-handler.c** . . . **??**
bsps/shared/irq/irq-generic.c
Generic BSP interrupt support implementation . . . 1556
bsps/shared/irq/irq-lock.c
BSP interrupt support lock implementation . . . 1557
bsps/shared/start/bootcard.c . . . 1557
bsps/shared/start/**bspreset-empty.c** . . . **??**
bsps/shared/start/mallocinitone.c
_Malloc_Initialize() Implementation . . . 1558
bsps/shared/start/wkspaceinitone.c
_Workspace_Handler_initialization() Implementation . . . 1559
bsps/sparc/leon3/clock/**ckinit.c** . . . **??**
bsps/sparc/leon3/console/**printk_support.c** . . . **??**

bsps/sparc/leon3/include/amba.h . . . 1559
bsps/sparc/leon3/include/bsp.h
Global BSP definitions . . . 1559
bsps/sparc/leon3/include/leon.h
LEON3 BSP data types and macros . . . 1562
bsps/sparc/leon3/include/tm27.h
Implementations for interrupt mechanisms for Time Test 27 . . . 1567
bsps/sparc/leon3/include/bsp/irq.h
LEON3 generic shared IRQ setup . . . 1561
bsps/sparc/leon3/start/**amba.c** . . . **??**
bsps/sparc/leon3/start/bsp_fatal_halt.c
LEON3 BSP Fatal_halt handler . . . 1568
bsps/sparc/leon3/start/bspclean.c
LEON3 BSP fatal extension . . . 1568
bsps/sparc/leon3/start/bspdelay.c . . . 1569
bsps/sparc/leon3/start/bspsmp.c
LEON3 SMP BSP Support . . . 1569
bsps/sparc/leon3/start/**bspstart.c** . . . **??**
bsps/sparc/leon3/start/**cache.c** . . . **??**
bsps/sparc/leon3/start/**cpucounter.c** . . . **??**
bsps/sparc/leon3/start/**eirq.c** . . . **??**
bsps/sparc/leon3/start/setvec.c
Install an interrupt vector on SPARC . . . 1570
bsps/sparc/leon3/start/**spurious.c** . . . **??**
bsps/sparc/shared/**doxygen.h** . . . **??**
bsps/sparc/shared/irq/**bsp_isr_handler.c** . . . **??**
bsps/sparc/shared/irq/**irq-shared.c** . . . **??**
bsps/sparc/shared/start/bsp_fatal_exit.c
ERC32/LEON2/LEON3 BSP specific exit handler . . . 1570
bsps/sparc/shared/start/**bspgetworkarea.c** . . . **??**
cpukit/**doxygen.h** . . . **??**
cpukit/doxygen/**appl-config.h** . . . **??**
cpukit/doxygen/**top-level-groups.h** . . . **??**
cpukit/include/rtems.h
This header file defines the RTEMS Classic API . . . 1571
cpukit/include/**sha256.h** . . . **??**
cpukit/include/machine/**_kernel_cpuset.h** . . . **??**
cpukit/include/machine/**_kernel_param.h** . . . **??**
cpukit/include/machine/**_kernel_time.h** . . . **??**
cpukit/include/machine/**_kernel_types.h** . . . **??**
cpukit/include/machine/**_timecounter.h** . . . **??**
cpukit/include/rtems/assoc.h
RTEMS Associativity Routines . . . 1571
cpukit/include/rtems/bspIo.h
Interface to Kernel Print Methods . . . 1572
cpukit/include/rtems/chain.h
Chain API . . . 1577
cpukit/include/rtems/clockdrv.h
Clock Driver API . . . 1580
cpukit/include/rtems/confdefs.h
Evaluate Configuration Options . . . 1581
cpukit/include/rtems/config.h
This header file provides parts of the application configuration information API . . . 1605
cpukit/include/rtems/counter.h
Free-Running Counter and Busy Wait Delay API . . . 1608
cpukit/include/rtems/extension.h
This header file defines the User Extensions Manager API . . . 1608

cpukit/include/rtems/extensiondata.h
Classic User Extensions Data Structures . . . 1609
cpukit/include/rtems/extensionimpl.h
Classic User Extensions Implementation . . . 1610
cpukit/include/rtems/fatal.h
This header file defines the Fatal Error Manager API . . . 1544
cpukit/include/rtems/fs.h
Basic Filesystem Types . . . 1610
cpukit/include/rtems/init.h
This header file defines the Initialization Manager API . . . 1611
cpukit/include/rtems/io.h
This header file defines the IO Manager API . . . 1612
cpukit/include/rtems/irq-extension.h
Header file for the Interrupt Manager Extension . . . 1613
cpukit/include/rtems/libcsupport.h
Standard C Library Support . . . 1615
cpukit/include/rtems/libio.h
Basic IO API . . . 1585
cpukit/include/rtems/**linkersets.h** . . . **??**
cpukit/include/rtems/malloc.h . . . 1592
cpukit/include/rtems/mallocinitone.h
Malloc Support Initialization API . . . 1616
cpukit/include/rtems/print.h
User print interface to the bspIO print plug in . . . 1617
cpukit/include/rtems/printer.h
User print interface to the bspIO print plug in . . . 1619
cpukit/include/rtems/scheduler.h
Scheduler Configuration API . . . 1600
cpukit/include/rtems/**sysinit.h** . . . **??**
cpukit/include/rtems/termiostypes.h . . . 1781
cpukit/include/rtems/**test-info.h** . . . **??**
cpukit/include/rtems/**test.h** . . . **??**
cpukit/include/rtems/**thread.h** . . . **??**
cpukit/include/rtems/timecounter.h
Timecounter API . . . 1764
cpukit/include/rtems/version.h
Version API . . . 1788
cpukit/include/rtems/confdefs/bdbuf.h
Evaluate Block Device Cache Configuration Options . . . 1581
cpukit/include/rtems/confdefs/bsp.h
Evaluate BSP Related Configuration Options . . . 1561
cpukit/include/rtems/confdefs/clock.h
Evaluate Clock Configuration Options . . . 1582
cpukit/include/rtems/confdefs/console.h
Evaluate Console Driver Configuration Options . . . 1584
cpukit/include/rtems/confdefs/extensions.h
Evaluate User Extensions Configuration Options . . . 1584
cpukit/include/rtems/confdefs/inittask.h
Evaluate User Initialization Task Configuration Options . . . 1584
cpukit/include/rtems/confdefs/initthread.h
Evaluate POSIX Initialization Thread Configuration Options . . . 1584
cpukit/include/rtems/confdefs/iodrivers.h
Evaluate IO Driver Configuration Options . . . 1585
cpukit/include/rtems/confdefs/libio.h
Evaluate IO Library Configuration Options . . . 1585
cpukit/include/rtems/confdefs/libpci.h
This header file evaluates PCI library configuration options . . . 1592

cpukit/include/rtems/confdefs/malloc.h
Evaluate C Program Heap Configuration Options . . . 1592
cpukit/include/rtems/confdefs/mpci.h
Evaluate MPCI Configuration Options . . . 1597
cpukit/include/rtems/confdefs/newlib.h
Evaluate Newlib Configuration Options . . . 1597
cpukit/include/rtems/confdefs/objectsclassic.h
Evaluate Classic API Objects Configuration Options . . . 1597
cpukit/include/rtems/confdefs/objectsposix.h
Evaluate POSIX API Objects Configuration Options . . . 1597
cpukit/include/rtems/confdefs/obsolete.h
Evaluate Obsolete Configuration Options . . . 1598
cpukit/include/rtems/confdefs/percpu.h
Evaluate Per-CPU Configuration Options . . . 1598
cpukit/include/rtems/confdefs/scheduler.h
Evaluate Scheduler Configuration Options . . . 1600
cpukit/include/rtems/confdefs/threads.h
Evaluate Thread Configuration Options . . . 1603
cpukit/include/rtems/confdefs/unlimited.h
Evaluate Unlimited Objects Configuration Options . . . 1604
cpukit/include/rtems/confdefs/wkspace.h
Evaluate Workspace Configuration Options . . . 1604
cpukit/include/rtems/confdefs/wkspacesupport.h
Configuration Options Workspace Support Macros . . . 1605
cpukit/include/rtems/posix/spinlockimpl.h
Inlined Routines from the POSIX Spinlock Manager . . . 1617
cpukit/include/rtems/rtems/asr.h
This header file defines the parts of the Signal Manager API . . . 1620
cpukit/include/rtems/rtems/asrdata.h
Classic ASR Data Structures . . . 1621
cpukit/include/rtems/rtems/asrimpl.h
Classic ASR Implementation . . . 1622
cpukit/include/rtems/rtems/attr.h
This header file provides Classic API directive attributes . . . 1622
cpukit/include/rtems/rtems/attrimpl.h
Classic Attributes Implementation . . . 1624
cpukit/include/rtems/rtems/barrier.h
This header file defines the Barrier Manager API . . . 1625
cpukit/include/rtems/rtems/barrierdata.h
Classic Barrier Manager Data Structures . . . 1626
cpukit/include/rtems/rtems/barrierimpl.h
Classic Barrier Manager Implementation . . . 1626
cpukit/include/rtems/rtems/cache.h
This header file defines the Cache Manager API . . . 1627
cpukit/include/rtems/rtems/clock.h
This header file defines the Clock Manager API . . . 1582
cpukit/include/rtems/rtems/config.h
This header file provides parts of the application configuration information API . . . 1607
cpukit/include/rtems/rtems/dpmem.h
This header file defines the Dual-Ported Memory Manager API . . . 1628
cpukit/include/rtems/rtems/dpmemdata.h
Classic Dual Ported Memory Manager Data Structures . . . 1629
cpukit/include/rtems/rtems/dpmemimpl.h
Dual Ported Memory Manager Implementation . . . 1629
cpukit/include/rtems/rtems/event.h
This header file provides the Event Manager API . . . 1630

cpukit/include/rtems/rtems/eventdata.h
This header file defines the API used by the Application Configuration to define the configured Thread Control Block (TCB) . . . 1634
cpukit/include/rtems/rtems/eventimpl.h
This header file defines interfaces used by the event implementation . . . 1634
cpukit/include/rtems/rtems/intr.h
This header file defines the Interrupt Manager API . . . 1635
cpukit/include/rtems/rtems/mainpage.h . . . 1636
cpukit/include/rtems/rtems/message.h
This header file defines the Message Manager API . . . 1637
cpukit/include/rtems/rtems/messagedata.h
Classic Message Queue Manager API . . . 1638
cpukit/include/rtems/rtems/messageimpl.h
Classic Message Queue Manager Implementation . . . 1638
cpukit/include/rtems/rtems/modes.h
This header file provides the task modes API of the Task Manager . . . 1639
cpukit/include/rtems/rtems/modesimpl.h
Classic Modes Implementation . . . 1640
cpukit/include/rtems/rtems/object.h
This header file defines the Object Manager API . . . 1641
cpukit/include/rtems/rtems/objectimpl.h
Implementation Interfaces for Classic Objects . . . 1644
cpukit/include/rtems/rtems/options.h
This header file provides the Classic API directive options . . . 1648
cpukit/include/rtems/rtems/optionsimpl.h
Classic Options Implementation . . . 1648
cpukit/include/rtems/rtems/part.h
This header file provides the Partition Manager API . . . 1649
cpukit/include/rtems/rtems/partdata.h
This header file defines the API used by the Application Configuration to define the Partition Manager information . . . 1650
cpukit/include/rtems/rtems/partimpl.h
This header file defines interfaces used by the Partition Manager implementation . . . 1650
cpukit/include/rtems/rtems/ratemon.h
This header file defines the Rate-Monotonic Manager API . . . 1651
cpukit/include/rtems/rtems/ratemondata.h
Classic Rate Monotonic Scheduler Data Structures . . . 1652
cpukit/include/rtems/rtems/ratemonimpl.h
Classic Rate Monotonic Scheduler Implementation . . . 1653
cpukit/include/rtems/rtems/region.h
This header file defines the Region Manager API . . . 1654
cpukit/include/rtems/rtems/regiondata.h
Classic Region Manager Data Structures . . . 1654
cpukit/include/rtems/rtems/regionimpl.h
Classic Region Manager Implementation . . . 1655
cpukit/include/rtems/rtems/sem.h
This header file defines the Semaphore Manager API . . . 1656
cpukit/include/rtems/rtems/semdata.h
Classic Semaphore Manager Data Structures . . . 1656
cpukit/include/rtems/rtems/semimpl.h
Classic Semaphore Manager Implementation . . . 1657
cpukit/include/rtems/rtems/signal.h
This header file defines the parts of the Signal Manager API . . . 1658
cpukit/include/rtems/rtems/signalimpl.h
Signals Implementation . . . 1658
cpukit/include/rtems/rtems/status.h
This header file provides the status codes of Classic API directives and support functions . . . 1659

cpukit/include/rtems/rtems/statusimpl.h
Classic Status Implementation . . . 1660
cpukit/include/rtems/rtems/support.h
This header file defines support services of the API . . . 1660
cpukit/include/rtems/rtems/tasks.h
This header file defines the main parts of the Tasks Manager API . . . 1661
cpukit/include/rtems/rtems/tasksdata.h
Classic Tasks Manager Data Structures . . . 1664
cpukit/include/rtems/rtems/tasksimpl.h
Classic Tasks Manager Implementation . . . 1665
cpukit/include/rtems/rtems/timer.h
This header file defines the Timer Manager API . . . 1666
cpukit/include/rtems/rtems/timerdata.h
Classic Partition Manager Data Structures . . . 1667
cpukit/include/rtems/rtems/timerimpl.h
Classic Timer Implementation . . . 1668
cpukit/include/rtems/rtems/types.h
This header file provides types used by the Classic API . . . 1669
cpukit/include/rtems/score/address.h
Information Required to Manipulate Physical Addresses . . . 1670
cpukit/include/rtems/score/apimutex.h
API Mutex Handler API . . . 1671
cpukit/include/rtems/score/assert.h
Information for the Assert Handler . . . 1671
cpukit/include/rtems/score/atomic.h
Atomic Operations API . . . 1672
cpukit/include/rtems/score/basedefs.h
This header file provides basic definitions used by the API and the implementation . . . 1673
cpukit/include/rtems/score/chain.h
Chain Handler API . . . 1579
cpukit/include/rtems/score/chainimpl.h
Chain Handler API . . . 1676
cpukit/include/rtems/score/context.h
Information About Each Thread's Context . . . 1679
cpukit/include/rtems/score/copyrt.h
Copyright Notice for RTEMS . . . 1679
cpukit/include/rtems/score/corebarrier.h
Constants and Structures Associated with the Barrier Handler . . . 1680
cpukit/include/rtems/score/corebarrierimpl.h
Inlined Routines Associated with the SuperCore Barrier . . . 1680
cpukit/include/rtems/score/coremsg.h
Constants and Structures Associated with the Message Queue Handler . . . 1681
cpukit/include/rtems/score/coremsgbuffer.h
This header file defines the buffer data structure used by the Message Queue Handler . . . 1682
cpukit/include/rtems/score/coremsgimpl.h
Inlined Routines in the Core Message Handler . . . 1683
cpukit/include/rtems/score/coremutex.h
CORE Mutex API . . . 1685
cpukit/include/rtems/score/coremuteximpl.h
CORE Mutex Implementation . . . 1685
cpukit/include/rtems/score/coresem.h
Data Associated with the Counting Semaphore Handler . . . 1687
cpukit/include/rtems/score/coresemimpl.h
Inlined Routines Associated with the SuperCore Semaphore . . . 1687
cpukit/include/rtems/score/cpustdatomic.h
Atomic Operations CPU API . . . 1688
cpukit/include/rtems/score/freechain.h
Freechain Handler API . . . 1690

cpukit/include/rtems/score/freechainimpl.h
Freechain Handler API . . . 1691
cpukit/include/rtems/score/heap.h
Heap Handler API . . . 1692
cpukit/include/rtems/score/heapimpl.h
Heap Handler Implementation . . . 1693
cpukit/include/rtems/score/heapinfo.h
Heap Handler Information API . . . 1696
cpukit/include/rtems/score/interr.h
Constants and Prototypes Related to the Internal Error Handler . . . 1696
cpukit/include/rtems/score/**io.h** . . . **??**
cpukit/include/rtems/score/isr.h
Data Related to the Management of Processor Interrupt Levels . . . 1698
cpukit/include/rtems/score/isrlevel.h
ISR Level Type . . . 1699
cpukit/include/rtems/score/isrlock.h
ISR Locks . . . 1699
cpukit/include/rtems/score/memory.h
Memory Handler API . . . 1701
cpukit/include/rtems/score/mppkt.h
Specification for the Packet Handler . . . 1702
cpukit/include/rtems/score/mrsp.h
Definitions for Multiprocessor Resource Sharing Protocol (MrsP) . . . 1703
cpukit/include/rtems/score/mrspimpl.h
Definitions for Multiprocessor Resource Sharing Protocol (MrsP) Implementation . . . 1703
cpukit/include/rtems/score/muteximpl.h
Structures for the implementation of mutexes . . . 1705
cpukit/include/rtems/score/object.h
Constants and Structures Associated with the Object Handler . . . 1643
cpukit/include/rtems/score/objectdata.h
Object Handler Data Structures . . . 1705
cpukit/include/rtems/score/objectimpl.h
Inlined Routines in the Object Handler . . . 1645
cpukit/include/rtems/score/percpu.h . . . 1598
cpukit/include/rtems/score/percpudata.h
Definition of custom per-CPU items . . . 1706
cpukit/include/rtems/score/priority.h
Priority Handler API . . . 1707
cpukit/include/rtems/score/prioritybitmap.h
Manipulation Routines for the Bitmap Priority Queue Implementation . . . 1708
cpukit/include/rtems/score/prioritybitmapimpl.h
Inlined Routines in the Priority Handler Bit Map Implementation . . . 1708
cpukit/include/rtems/score/priorityimpl.h
Priority Handler API Implementation . . . 1709
cpukit/include/rtems/score/processormask.h
Processor Mask API . . . 1711
cpukit/include/rtems/score/profiling.h
Profiling Support API . . . 1713
cpukit/include/rtems/score/protectedheap.h
Protected Heap Handler API . . . 1714
cpukit/include/rtems/score/rbtree.h
Constants and Structures Associated with the Red-Black Tree Handler . . . 1715
cpukit/include/rtems/score/rbtreeimpl.h
Inlined Routines Associated with Red-Black Trees . . . 1717
cpukit/include/rtems/score/scheduler.h
Constants and Structures Associated with the Scheduler . . . 1601
cpukit/include/rtems/score/scheduleredf.h
Data Related to the Manipulation of Threads for the EDF Scheduler . . . 1717

cpukit/include/rtems/score/scheduleredfimpl.h
EDF Scheduler Implementation . . . 1719
cpukit/include/rtems/score/scheduleredfsmp.h
EDF SMP Scheduler API . . . 1719
cpukit/include/rtems/score/schedulerimpl.h
Inlined Routines Associated with the Manipulation of the Scheduler . . . 1721
cpukit/include/rtems/score/schedulernode.h
Handles Scheduler Nodes . . . 1724
cpukit/include/rtems/score/schedulernodeimpl.h
Scheduler Node Implementation . . . 1724
cpukit/include/rtems/score/schedulerpriority.h
Thread Manipulation with the Priority-Based Scheduler . . . 1726
cpukit/include/rtems/score/schedulerpriorityimpl.h
Inlined Routines Associated with the Manipulation of the Priority-Based Scheduling Structures 1727
cpukit/include/rtems/score/schedulersimple.h
Manipulation of Threads Simple-Priority-Based Ready Queue . . . 1728
cpukit/include/rtems/score/schedulersimpleimpl.h
Inlined Routines Associated with the Manipulation of the Priority-Based Scheduling Structures 1729
cpukit/include/rtems/score/schedulersmp.h
SMP Scheduler API . . . 1729
cpukit/include/rtems/score/schedulersmpimpl.h
SMP Scheduler Implementation . . . 1730
cpukit/include/rtems/score/semaphoreimpl.h
Semaphore Implementation . . . 1733
cpukit/include/rtems/score/smp.h
SuperCore SMP Support API . . . 1734
cpukit/include/rtems/score/smpbarrier.h
SMP Barrier API . . . 1734
cpukit/include/rtems/score/smpimpl.h
SuperCore SMP Implementation . . . 1735
cpukit/include/rtems/score/smplock.h
SMP Lock API . . . 1737
cpukit/include/rtems/score/smplockmcs.h
SMP Lock API . . . 1738
cpukit/include/rtems/score/smplockseq.h
SMP Lock API . . . 1739
cpukit/include/rtems/score/smplockstats.h
SMP Lock API . . . 1740
cpukit/include/rtems/score/smplockticket.h
SMP Lock API . . . 1740
cpukit/include/rtems/score/stack.h
Information About the Thread Stack Handler . . . 1741
cpukit/include/rtems/score/stackimpl.h
Inlined Routines from the Stack Handler . . . 1742
cpukit/include/rtems/score/states.h
Thread Execution State Information . . . 1743
cpukit/include/rtems/score/statesimpl.h
Inlined Routines Associated with Thread State Information . . . 1743
cpukit/include/rtems/score/**status.h** . . . **??**
cpukit/include/rtems/score/sysstate.h
System State Handler API . . . 1745
cpukit/include/rtems/score/thread.h
Constants and Structures Related with the Thread Control Block . . . 1746
cpukit/include/rtems/score/threaddispatch.h
Constants and Structures Related with Thread Dispatch . . . 1748
cpukit/include/rtems/score/threadidledata.h
Constants for the idle threads . . . 1749

cpukit/include/rtems/score/threadimpl.h
Inlined Routines from the Thread Handler . . . 1750
cpukit/include/rtems/score/threadq.h
Constants and Structures Needed to Declare a Thread Queue . . . 1757
cpukit/include/rtems/score/threadqimpl.h
Constants and Structures Associated with the Manipulation of Objects . . . 1758
cpukit/include/rtems/score/timecounter.h
Timecounter API . . . 1763
cpukit/include/rtems/score/timecounterimpl.h
Timecounter Implementation . . . 1765
cpukit/include/rtems/score/timespec.h
Contains Helpers for Manipulating Timespecs . . . 1766
cpukit/include/rtems/score/timestamp.h
Helpers for Manipulating Timestamps . . . 1767
cpukit/include/rtems/score/timestampimpl.h
Helpers for Manipulating Timestamps . . . 1767
cpukit/include/rtems/score/tls.h
Thread-Local Storage (TLS) . . . 1768
cpukit/include/rtems/score/todimpl.h
Time of Day Handler API . . . 1769
cpukit/include/rtems/score/userext.h
User Extension Handler API . . . 1772
cpukit/include/rtems/score/userextdata.h
User Extension Handler Data Structures . . . 1773
cpukit/include/rtems/score/userextimpl.h
User Extension Handler API . . . 1773
cpukit/include/rtems/score/watchdog.h
Constants and Structures Associated with Watchdog Timers . . . 1776
cpukit/include/rtems/score/watchdogimpl.h
Inlined Routines in the Watchdog Handler . . . 1776
cpukit/include/rtems/score/watchdogticks.h
Constants for the watchdog ticks . . . 1779
cpukit/include/rtems/score/wkspace.h
Information Related to the RAM Workspace . . . 1604
cpukit/include/rtems/score/wkspacedata.h
Constants defined by the application configuration for the idle threads . . . 1779
cpukit/include/rtems/score/wkspaceinitone.h
Workspace Handler Initialization API . . . 1780
cpukit/include/sys/**_ffcounter.h** . . . **??**
cpukit/include/sys/**endian.h** . . . **??**
cpukit/include/sys/**priority.h** . . . **??**
cpukit/include/sys/statvfs.h
Interface to the statvfs() Set of API Methods . . . 1788
cpukit/include/sys/**timeffc.h** . . . **??**
cpukit/include/sys/**timepps.h** . . . **??**
cpukit/include/sys/**timetc.h** . . . **??**
cpukit/include/sys/**timex.h** . . . **??**
cpukit/libcsupport/src/**alignedalloc.c** . . . **??**
cpukit/libcsupport/src/malloc_deferred.c
Malloc Deferred Support . . . 1789
cpukit/libcsupport/src/**malloc_p.h** . . . **??**
cpukit/libcsupport/src/**mallocdirtydefault.c** . . . **??**
cpukit/libcsupport/src/**mallocextenddefault.c** . . . **??**
cpukit/libcsupport/src/mallocheap.c
This source file provides the C Program Heap control along with the system initialization handler 1789
cpukit/libcsupport/src/print_printf.c
RTEMS Print Support . . . 1790

cpukit/libcsupport/src/printk.c
Kernel Printf Function . . . 1791
cpukit/libcsupport/src/printk_plugin.c
Plugin Printk . . . 1792
cpukit/libcsupport/src/**rtems_heap_null_extend.c** . . . **??**
cpukit/libcsupport/src/**rtems_put_char.c** . . . **??**
cpukit/libcsupport/src/rtems_putc.c
RTEMS Output a Character . . . 1792
cpukit/libcsupport/src/termiosinitialize.c
Termios Initialization . . . 1794
cpukit/libcsupport/src/vprintk.c
Print Formatted Output . . . 1794
cpukit/libmd/**sha256c.c** . . . **??**
cpukit/libtest/t-test-busy-tick.c
Implementation of T_get_one_clock_tick_busy() . . . 1795
cpukit/libtest/t-test-busy.c
Implementation of T_busy() . . . 1795
cpukit/libtest/**t-test-checks.c** . . . **??**
cpukit/libtest/**t-test-hash-sha256.c** . . . **??**
cpukit/libtest/t-test-interrupt.c
Implementation of T_interrupt_test() . . . 1796
cpukit/libtest/**t-test-rtems-context.c** . . . **??**
cpukit/libtest/**t-test-rtems-measure.c** . . . **??**
cpukit/libtest/t-test-rtems-objs.c
RTEMS Objects Support for Test Framework . . . 1798
cpukit/libtest/**t-test-rtems.c** . . . **??**
cpukit/libtest/t-test-rtems.h
RTEMS Support for Test Framework . . . 1799
cpukit/libtest/t-test-thread-switch.c
Implementation of T_thread_switch_record() . . . 1799
cpukit/libtest/**t-test-time.c** . . . **??**
cpukit/libtest/**t-test.c** . . . **??**
cpukit/libtest/**testbeginend.c** . . . **??**
cpukit/libtest/testrun.c
Implementation of rtems_test_run_default() . . . 1800
cpukit/posix/src/pspindestroy.c
Destroy a Spinlock . . . 1801
cpukit/posix/src/pspininit.c
POSIX Function Initializes a Spinlock Instance . . . 1802
cpukit/posix/src/pspinlock.c
Wait at a Spinlock . . . 1802
cpukit/posix/src/pspinunlock.c
Function Unlocks a Spin Lock Object . . . 1803
cpukit/rtems/src/barrier.c
Classic Barrier Information with Zero Objects . . . 1803
cpukit/rtems/src/barriercreate.c
RTEMS Create Barrier . . . 1804
cpukit/rtems/src/barrierdelete.c
RTEMS Delete Barrier . . . 1804
cpukit/rtems/src/barrierident.c
Rtems_barrier_ident() Implementation . . . 1805
cpukit/rtems/src/barrierrelease.c
RTEMS Barrier Release . . . 1805
cpukit/rtems/src/barrierwait.c
RTEMS Barrier Wait . . . 1806
cpukit/rtems/src/clockgetuptime.c
Obtain the System Uptime . . . 1806

cpukit/rtems/src/clocktodtoseconds.c
TOD to Seconds . . . 1806
cpukit/rtems/src/clocktodvalidate.c
TOD Validate . . . 1808
cpukit/rtems/src/eventreceive.c
This source file contains the implementation of rtems_event_receive() . . . 1809
cpukit/rtems/src/eventseize.c
This source file contains the implementation of _Event_Seize() and the task event MPCI support system initialization . . . 1809
cpukit/rtems/src/eventsend.c
This source file contains the implementation of rtems_event_send() . . . 1810
cpukit/rtems/src/eventsurrender.c
This source file contains the implementation of _Event_Surrender() . . . 1810
cpukit/rtems/src/intrcatch.c
RTEMS Interrupt Catch . . . 1811
cpukit/rtems/src/msg.c
Classic Message Queue Information with Zero Objects . . . 1811
cpukit/rtems/src/msgqbroadcast.c
RTEMS Broadcast Message Queue . . . 1811
cpukit/rtems/src/msgqconstruct.c
RTEMS Create Message Queue . . . 1812
cpukit/rtems/src/msgqcreate.c
This source file contains the implementation of rtems_message_queue_create() . . . 1813
cpukit/rtems/src/msgqdelete.c
RTEMS Delete Message Queue . . . 1813
cpukit/rtems/src/msgqflush.c
Rtems_message_queue_flush . . . 1814
cpukit/rtems/src/msgqgetnumberpending.c
RTEMS Message Queue Get Number Pending . . . 1814
cpukit/rtems/src/msgqident.c
Rtems_message_queue_ident() Implementation . . . 1814
cpukit/rtems/src/msgqreceive.c
RTEMS Message Queue Receive . . . 1815
cpukit/rtems/src/msgqsend.c
Rtems_message_queue_send . . . 1815
cpukit/rtems/src/msgqurgent.c
RTEMS Urgent Message Queue . . . 1816
cpukit/rtems/src/part.c
This source file contains the default _Partition_Information with zero objects . . . 1816
cpukit/rtems/src/partcreate.c
This source file contains the implementation of rtems_partition_create() . . . 1816
cpukit/rtems/src/partdelete.c
This source file contains the implementation of rtems_partition_delete() . . . 1817
cpukit/rtems/src/partgetbuffer.c
This source file contains the implementation of rtems_partition_get_buffer() . . . 1818
cpukit/rtems/src/partident.c
This source file contains the implementation of rtems_partition_ident() . . . 1818
cpukit/rtems/src/partreturnbuffer.c
This source file contains the implementation of rtems_partition_return_buffer() . . . 1819
cpukit/rtems/src/ratemon.c
Classic Rate Monotonic Information with Zero Objects . . . 1819
cpukit/rtems/src/ratemoncancel.c
RTEMS Rate Monotonic Cancel . . . 1819
cpukit/rtems/src/ratemoncreate.c
Create a Period . . . 1820
cpukit/rtems/src/ratemondelete.c
RTEMS Delete Rate Monotonic . . . 1821

cpukit/rtems/src/ratemongetstatistics.c
RTEMS Rate Monotonic Get Statistics . . . 1821
cpukit/rtems/src/ratemongetstatus.c
RTEMS Rate Monotonic Get Status . . . 1821
cpukit/rtems/src/ratemonident.c
Rtems_rate_monotonic_ident() Implementation . . . 1822
cpukit/rtems/src/ratemonperiod.c
Rate Monotonic Support . . . 1822
cpukit/rtems/src/ratemonresetstatistics.c
RTEMS Rate Monotonic Reset Statistics . . . 1823
cpukit/rtems/src/ratemontimeout.c
Rate Monotonic Timeout . . . 1823
cpukit/rtems/src/rtemsnametoid.c
_RTEMS_Name_to_id() Implementation . . . 1824
cpukit/rtems/src/**scheduleraddprocessor.c** . . . **??**
cpukit/rtems/src/**schedulergetprocessorset.c** . . . **??**
cpukit/rtems/src/**scheduleridentbyprocessor.c** . . . **??**
cpukit/rtems/src/**scheduleridentbyprocessorset.c** . . . **??**
cpukit/rtems/src/**schedulerremoveprocessor.c** . . . **??**
cpukit/rtems/src/sem.c
Classic Semaphore Information with Zero Objects . . . 1824
cpukit/rtems/src/semcreate.c
Rtems_semaphore_create . . . 1825
cpukit/rtems/src/semdelete.c
RTEMS Delete Semaphore . . . 1826
cpukit/rtems/src/semident.c
Rtems_semaphore_ident() Implementation . . . 1826
cpukit/rtems/src/semobtain.c
RTEMS Obtain Semaphore . . . 1827
cpukit/rtems/src/semrelease.c
RTEMS Semaphore Release . . . 1827
cpukit/rtems/src/**semsetpriority.c** . . . **??**
cpukit/rtems/src/status.c
Status Mapping Arrays . . . 1828
cpukit/rtems/src/statustext.c . . . 1828
cpukit/rtems/src/systemeventreceive.c
This source file contains the implementation of rtems_event_system_receive() . . . 1828
cpukit/rtems/src/systemeventsend.c
This source file contains the implementation of rtems_event_system_send() . . . 1829
cpukit/rtems/src/taskconstruct.c
RTEMS Task Create from Config . . . 1830
cpukit/rtems/src/taskdelete.c
RTEMS Delete Task . . . 1832
cpukit/rtems/src/taskgetaffinity.c
RTEMS Task Get Affinity . . . 1832
cpukit/rtems/src/**taskgetpriority.c** . . . **??**
cpukit/rtems/src/**taskgetscheduler.c** . . . **??**
cpukit/rtems/src/taskident.c
Rtems_task_ident() Implementation . . . 1832
cpukit/rtems/src/**taskinitdefault.c** . . . **??**
cpukit/rtems/src/taskinitusers.c
_RTEMS_tasks_Initialize_user_tasks_body . . . 1833
cpukit/rtems/src/taskissuspended.c
Rtems_task_is_suspended . . . 1833
cpukit/rtems/src/taskrestart.c
RTEMS Task Restart . . . 1834

cpukit/rtems/src/taskresume.c
RTEMS Resume Task . . . 1834
cpukit/rtems/src/tasks.c
RTEMS Task API Extensions . . . 1834
cpukit/rtems/src/taskself.c
RTEMS Get Self Task Id . . . 1835
cpukit/rtems/src/tasksetaffinity.c
RTEMS Task Set Affinity . . . 1835
cpukit/rtems/src/tasksetpriority.c
RTEMS Set Task Priority . . . 1835
cpukit/rtems/src/**tasksetscheduler.c** . . . **??**
cpukit/rtems/src/taskstart.c
RTEMS Start Task . . . 1836
cpukit/rtems/src/tasksuspend.c
RTEMS Suspend Task . . . 1836
cpukit/rtems/src/taskwakeafter.c
RTEMS Task Wake After . . . 1837
cpukit/rtems/src/**timercancel.c** . . . **??**
cpukit/rtems/src/timercreate.c
RTEMS Create Timer . . . 1837
cpukit/rtems/src/timerdelete.c
RTEMS Delete Timer . . . 1838
cpukit/rtems/src/timerfireafter.c
RTEMS Timer Fire After . . . 1839
cpukit/rtems/src/timerident.c
Rtems_timer_ident() Implementation . . . 1839
cpukit/rtems/src/timerreset.c
RTEMS Timer Reset . . . 1839
cpukit/sapi/src/exinit.c
Initialization Manager . . . 1840
cpukit/sapi/src/extension.c
Extension Manager Information with Zero Objects . . . 1841
cpukit/sapi/src/extensioncreate.c
User Extensions Implementation . . . 1841
cpukit/sapi/src/extensiondelete.c
User Extensions Implementation . . . 1842
cpukit/sapi/src/extensionident.c
Rtems_extension_ident() Implementation . . . 1843
cpukit/sapi/src/getversionstring.c
Get the RTEMS Version as a String . . . 1843
cpukit/sapi/src/version.c
Creates the version strings from the various pieces of version information. The main version number is part of the build system and is stamped into rtems/score/cpuopts.h. The version control key string is extracted from the version control tool when the code is being built and is updated if it has changed. The key may indicate there are local modification . . . 1843
cpukit/score/cpu/no_cpu/**cpuidle.c** . . . **??**
cpukit/score/cpu/sparc/cpu.c
SPARC CPU Dependent Source . . . 1844
cpukit/score/cpu/sparc/**syscall.h** . . . **??**
cpukit/score/cpu/sparc/include/rtems/asm.h
Address the Problems Caused by Incompatible Flavor of Assemblers and Toolsets . . . 1850
cpukit/score/cpu/sparc/include/rtems/score/cpu.h
SPARC CPU Department Source . . . 1850
cpukit/score/cpu/sparc/include/rtems/score/**cpuatomic.h** . . . **??**
cpukit/score/cpu/sparc/include/rtems/score/cpuimpl.h
CPU Port Implementation API . . . 1870
cpukit/score/cpu/sparc/include/rtems/score/sparc.h
Information Required to Build RTEMS for a Particular Member of the SPARC Family . . . 1872

cpukit/score/cpu/sparc/include/rtems/score/**sparcimpl.h** . . . . . **??**
cpukit/score/src/**allocatormutex.c** . . . . . **??**
cpukit/score/src/apimutexisowner.c . . . . . 1886
cpukit/score/src/apimutexlock.c
Acquires the specified API mutex . . . . . 1887
cpukit/score/src/apimutexunlock.c
Releases the Specified API Mutex . . . . . 1887
cpukit/score/src/chain.c
Initialize a Chain Header . . . . . 1888
cpukit/score/src/**chainnodecount.c** . . . . . **??**
cpukit/score/src/**configstackspacesize.c** . . . . . **??**
cpukit/score/src/corebarrier.c
Initialize CORE Barrier . . . . . 1888
cpukit/score/src/corebarrierrelease.c
Manually releases the Barrier . . . . . 1888
cpukit/score/src/corebarrierwait.c
Wait For The Barrier . . . . . 1889
cpukit/score/src/coremsg.c
Initialize a Message Queue . . . . . 1889
cpukit/score/src/coremsgbroadcast.c
Broadcast a Message to the Message Queue . . . . . 1890
cpukit/score/src/coremsgclose.c
Close a Message Queue . . . . . 1890
cpukit/score/src/coremsgflush.c
Flush Messages Routine . . . . . 1891
cpukit/score/src/coremsginsert.c
Insert a Message into the Message Queue . . . . . 1891
cpukit/score/src/coremsgseize.c
Seize a Message from the Message Queue . . . . . 1892
cpukit/score/src/coremsgsubmit.c
CORE Message Queue Submit . . . . . 1892
cpukit/score/src/coremsgwkspace.c
This source file contains the implementation of _CORE_message_queue_Workspace_allocate() 1893
cpukit/score/src/coremutexseize.c
Seize Mutex with Blocking . . . . . 1893
cpukit/score/src/coresem.c
Core Semaphore Initialize . . . . . 1893
cpukit/score/src/coretod.c
Initializes the Time of Day Handler . . . . . 1894
cpukit/score/src/heap.c
Heap Handler implementation . . . . . 1894
cpukit/score/src/heapallocate.c
Heap Handler implementation . . . . . 1895
cpukit/score/src/interr.c
Initiates system termination . . . . . 1896
cpukit/score/src/**ioprintf.c** . . . . . **??**
cpukit/score/src/**iovprintf.c** . . . . . **??**
cpukit/score/src/isr.c
Initialize the ISR handler . . . . . 1896
cpukit/score/src/isrisinprogress.c
ISR Is In Progress Default Implementation . . . . . 1897
cpukit/score/src/**kern_tc.c** . . . . . **??**
cpukit/score/src/**memoryallocate.c** . . . . . **??**
cpukit/score/src/**mutex.c** . . . . . **??**
cpukit/score/src/**objectactivecount.c** . . . . . **??**
cpukit/score/src/objectallocate.c
Allocate Object . . . . . 1897
cpukit/score/src/objectallocatenone.c . . . . . 1897

cpukit/score/src/objectallocatestatic.c . . . 1898
cpukit/score/src/objectapimaximumclass.c
Object API Maximum Class . . . 1898
cpukit/score/src/objectclose.c
Close Object . . . 1898
cpukit/score/src/objectfree.c
Free Object . . . 1899
cpukit/score/src/objectfreestatic.c . . . 1899
cpukit/score/src/objectgetinfo.c
Get Object Information . . . 1899
cpukit/score/src/objectgetinfoid.c
Get Information of an Object from an ID . . . 1900
cpukit/score/src/objectgetlocal.c
Object Get Local . . . 1900
cpukit/score/src/objectgetnameasstring.c
Extracts a Node from a Chain . . . 1900
cpukit/score/src/objectgetnoprotection.c
Get Object without Dispatching Protection . . . 1901
cpukit/score/src/objectinitializeinformation.c
Initialize Object Information . . . 1901
cpukit/score/src/objectnamespaceremove.c
Removes Object from Namespace . . . 1902
cpukit/score/src/objectnametoid.c
Object Name To Id . . . 1902
cpukit/score/src/**opt_compat.h** . . . **??**
cpukit/score/src/**opt_ffclock.h** . . . **??**
cpukit/score/src/**opt_ntp.h** . . . **??**
cpukit/score/src/percpu.c
Allocate and Initialize Per CPU Structures . . . 1903
cpukit/score/src/**percpustatewait.c** . . . **??**
cpukit/score/src/processormaskcopy.c
Processor Mask Implementation . . . 1903
cpukit/score/src/**rbtreeextract.c** . . . **??**
cpukit/score/src/**rbtreeinsert.c** . . . **??**
cpukit/score/src/rbtreenext.c
_RBTree_Next() and _RBTree_Next() implementation . . . 1904
cpukit/score/src/rbtreereplace.c
_RBTree_Replace_node() implementation . . . 1904
cpukit/score/src/scheduler.c
Scheduler Initialize . . . 1905
cpukit/score/src/**schedulerdefaultaskforhelp.c** . . . **??**
cpukit/score/src/**schedulerdefaultmappriority.c** . . . **??**
cpukit/score/src/schedulerdefaultnodedestroy.c
Scheduler Default Node Destruction Operation . . . 1905
cpukit/score/src/schedulerdefaultnodeinit.c
Scheduler Default Node Initialization Operation . . . 1905
cpukit/score/src/**schedulerdefaultpinunpin.c** . . . **??**
cpukit/score/src/schedulerdefaultreleasejob.c
Default Scheduler Release Job Operation . . . 1906
cpukit/score/src/**schedulerdefaultschedule.c** . . . **??**
cpukit/score/src/schedulerdefaultsetaffinity.c
Scheduler Default Set Affinity Operation . . . 1906
cpukit/score/src/**schedulerdefaultstartidle.c** . . . **??**
cpukit/score/src/schedulerdefaulttick.c
Default Scheduler At Tick Handler . . . 1907
cpukit/score/src/scheduleredfreleasejob.c
Scheduler EDF Release Job . . . 1907

cpukit/score/src/scheduleredfsmp.c
EDF SMP Scheduler Implementation . . . 1908
cpukit/score/src/**schedulerpriority.c** . . . **??**
cpukit/score/src/schedulerpriorityblock.c
Scheduler Priority Block . . . 1909
cpukit/score/src/schedulerprioritychangepriority.c
Removes Thread from Thread Queue . . . 1910
cpukit/score/src/schedulerpriorityschedule.c
Priority Scheduler Schedule Method . . . 1910
cpukit/score/src/schedulerpriorityunblock.c
Scheduler Priority Unblock . . . 1911
cpukit/score/src/schedulerpriorityyield.c
Scheduler Priority Yield . . . 1911
cpukit/score/src/**schedulersmp.c** . . . **??**
cpukit/score/src/**schedulersmpstartidle.c** . . . **??**
cpukit/score/src/smp.c
SMP Support . . . 1911
cpukit/score/src/**smpbarrierwait.c** . . . **??**
cpukit/score/src/**smplock.c** . . . **??**
cpukit/score/src/**smpmulticastaction.c** . . . **??**
cpukit/score/src/**smpunicastaction.c** . . . **??**
cpukit/score/src/**stackallocator.c** . . . **??**
cpukit/score/src/stackallocatorfreenothing.c
_Stack_Free_nothing() Implementation . . . 1913
cpukit/score/src/thread.c
Initialize Thread Handler . . . 1913
cpukit/score/src/threadchangepriority.c
Changes the Priority of a Thread . . . 1914
cpukit/score/src/threadclearstate.c
Clear Thread state . . . 1915
cpukit/score/src/threadcreateidle.c
Create Idle Thread . . . 1915
cpukit/score/src/threaddispatch.c
Dispatch Thread . . . 1916
cpukit/score/src/**threadentryadaptoridle.c** . . . **??**
cpukit/score/src/**threadentryadaptornumeric.c** . . . **??**
cpukit/score/src/threadget.c
Maps Thread IDs to TCB Pointer . . . 1917
cpukit/score/src/**threadgetcputimeused.c** . . . **??**
cpukit/score/src/threadhandler.c
Thread Handler . . . 1917
cpukit/score/src/**threadidledefault.c** . . . **??**
cpukit/score/src/threadinitialize.c
Initialize Thread . . . 1918
cpukit/score/src/**threaditerate.c** . . . **??**
cpukit/score/src/threadloadenv.c
Initializes Enviroment for A Thread . . . 1918
cpukit/score/src/threadq.c
Thread Queue Initialize . . . 1918
cpukit/score/src/threadqenqueue.c
Thread Queue Operations . . . 1919
cpukit/score/src/**threadqextractwithproxy.c** . . . **??**
cpukit/score/src/threadqflush.c
Thread Queue Flush . . . 1921
cpukit/score/src/**threadqops.c** . . . **??**
cpukit/score/src/**threadqtimeout.c** . . . **??**
cpukit/score/src/threadrestart.c
Restart Thread . . . 1922

cpukit/score/src/**threadscheduler.c** . . . . . **??**
cpukit/score/src/threadsetstate.c
Sets States for a Thread . . . . . 1924
cpukit/score/src/threadstackallocate.c
Stack Allocate Helper . . . . . 1924
cpukit/score/src/threadstart.c
Initializes Thread and Executes it . . . . . 1925
cpukit/score/src/threadstartmultitasking.c
Start Thread Multitasking . . . . . 1925
cpukit/score/src/threadtimeout.c
Thread Wait Timeout . . . . . 1926
cpukit/score/src/**threadunpin.c** . . . . . **??**
cpukit/score/src/threadyield.c
Thread Yield . . . . . 1926
cpukit/score/src/**tlsallocsize.c** . . . . . **??**
cpukit/score/src/userext.c
User Extension Handler implementation . . . . . 1926
cpukit/score/src/userextaddset.c
User Extension Handler implementation . . . . . 1927
cpukit/score/src/userextiterate.c
User Extension Iteration Helpers . . . . . 1927
cpukit/score/src/userextremoveset.c
User Extension Handler implementation . . . . . 1928
cpukit/score/src/watchdoginsert.c
Watchdog Insert . . . . . 1929
cpukit/score/src/watchdogremove.c
Remove Watchdog . . . . . 1929
cpukit/score/src/**watchdogtick.c** . . . . . **??**
cpukit/score/src/watchdogtickssinceboot.c
Watchdog Ticks Since Boot . . . . . 1929
cpukit/score/src/**watchdogtimeslicedefault.c** . . . . . **??**
cpukit/score/src/wkspace.c
Workspace Handler System Initialization . . . . . 1930
cpukit/score/src/wkspaceallocate.c
_Workspace_Allocate() Implementation . . . . . 1931
cpukit/score/src/**wkspaceisunifieddefault.c** . . . . . **??**
cpukit/score/src/wkspacemallocinitdefault.c
This source file provides the default definition of _Workspace_Malloc_initializer . . . . . 1931
cpukit/score/src/wkspacemallocinitunified.c
This source file provides the implementation of _Workspace_Malloc_initialize_unified() . . . . 1931
testsuites/validation/tc-attr.c . . . . . 1932
testsuites/validation/tc-barrier-ident.c . . . . . 1932
testsuites/validation/tc-event-send-receive.c . . . . . 1932
testsuites/validation/tc-events.c . . . . . 1933
testsuites/validation/tc-message-construct-errors.c . . . . . 1933
testsuites/validation/tc-message-ident.c . . . . . 1935
testsuites/validation/tc-modes.c . . . . . 1936
testsuites/validation/tc-options.c . . . . . 1936
testsuites/validation/tc-part-create.c . . . . . 1936
testsuites/validation/tc-part-delete.c . . . . . 1938
testsuites/validation/tc-part-get.c . . . . . 1940
testsuites/validation/tc-part-ident.c . . . . . 1941
testsuites/validation/tc-part-return.c . . . . . 1942
testsuites/validation/tc-part.c . . . . . 1943
testsuites/validation/tc-ratemon-ident.c . . . . . 1944
testsuites/validation/tc-sem-ident.c . . . . . 1944
testsuites/validation/tc-space-profile.c . . . . . 1944
testsuites/validation/tc-task-construct-errors.c . . . . . 1944

testsuites/validation/tc-task-ident.c . . . 1947
testsuites/validation/tc-timer-ident.c . . . 1948
testsuites/validation/tc-userext-ident.c . . . 1949
testsuites/validation/tr-event-constant.c . . . 1949
testsuites/validation/tr-event-constant.h . . . 1949
testsuites/validation/tr-event-send-receive.c . . . 1949
testsuites/validation/tr-event-send-receive.h . . . 1952
testsuites/validation/tr-object-ident-local.c . . . 1953
testsuites/validation/tr-object-ident-local.h . . . 1954
testsuites/validation/tr-object-ident.c . . . 1954
testsuites/validation/tr-object-ident.h . . . 1956
testsuites/validation/ts-space-profile.c . . . 1957
testsuites/validation/ts-validation-0.c . . . 1958

# Chapter 8

# Module Documentation

## 8.1 AMBA

AMBA Plug & Play routines.

### Files

- file ambapp.h
- file ambapp_ids.h

  *AMBA Plug & Play Bus Vendor and Device IDs.*
- file grlib.h

  *Common GRLIB AMBA Core Register definitions.*

### Classes

- struct ambapp_common_info
- struct ambapp_apb_info
- struct ambapp_ahb_info
- struct ambapp_dev
- struct ambapp_core
- struct ambapp_ahb_bus
- struct ambapp_mmap
- struct ambapp_bus
- struct ambapp_pnp_ahb
- struct ambapp_pnp_apb

## Macros

- #define **AHB_BUS_MAX** 6
- #define **AMBAPP_FLAG_FFACT_DIR** 0x100 /* Frequency factor direction, 0=down, 1=up */
- #define **AMBAPP_FLAG_FFACT** 0x0f0 /* Frequency factor against top bus */
- #define **AMBAPP_FLAG_MBUS** 0x00c
- #define **AMBAPP_FLAG_SBUS** 0x003
- #define **DEV_TO_APB**(adev) ((struct ambapp_apb_info *)((adev)->devinfo))
- #define **DEV_TO_AHB**(adev) ((struct ambapp_ahb_info *)((adev)->devinfo))
- #define **DEV_TO_COMMON**(adev) (((adev)->devinfo))
- #define **DEV_IS_FREE**(dev) (dev->owner == NULL)
- #define **DEV_IS_ALLOCATED**(dev) (dev->owner != NULL)
- #define **OPTIONS_AHB_MSTS** 0x00000001
- #define **OPTIONS_AHB_SLVS** 0x00000002
- #define **OPTIONS_APB_SLVS** 0x00000004
- #define **OPTIONS_ALL_DEVS** (OPTIONS_AHB_MSTS|OPTIONS_AHB_SLVS|OPTIONS_APB_SLVS)
- #define **OPTIONS_FREE** 0x00000010
- #define **OPTIONS_ALLOCATED** 0x00000020
- #define **OPTIONS_ALL** (OPTIONS_FREE|OPTIONS_ALLOCATED)
- #define **OPTIONS_DEPTH_FIRST** 0x00000100
- #define **DEV_AHB_NONE** 0
- #define **DEV_AHB_MST** 1
- #define **DEV_AHB_SLV** 2
- #define **DEV_APB_SLV** 3
- #define **ambapp_pnp_vendor**(id) (((id) >> 24) & 0xff)
- #define **ambapp_pnp_device**(id) (((id) >> 12) & 0xfff)
- #define **ambapp_pnp_ver**(id) (((id)>>5) & 0x1f)
- #define **ambapp_pnp_irq**(id) ((id) & 0x1f)
- #define **ambapp_pnp_start**(mbar) (((mbar) & 0xfff00000) & (((mbar) & 0xfff0) << 16))
- #define **ambapp_pnp_mbar_mask**(mbar) (((mbar)>>4) & 0xfff)
- #define **ambapp_pnp_mbar_type**(mbar) ((mbar) & 0xf)
- #define **ambapp_pnp_apb_start**(iobar, base) ((base) | ((((iobar) & 0xfff00000)>>12) & (((iobar) & 0xfff0)<<4)) )
- #define **ambapp_pnp_apb_mask**(iobar) ((~(ambapp_pnp_mbar_mask(iobar)<<8) & 0x000fffff) + 1)
- #define **AMBA_TYPE_AHBIO_ADDR**(addr, base_ioarea) ((unsigned int)(base_ioarea) | ((addr) >> 12))
- #define **AMBA_TYPE_APBIO** 0x1
- #define **AMBA_TYPE_MEM** 0x2
- #define **AMBA_TYPE_AHBIO** 0x3

## Typedefs

- typedef int(* **ambapp_func_t**) (struct ambapp_dev *dev, int index, void *arg)
- typedef void *(* **ambapp_memcpy_t**) (void *dest, const void *src, int n, struct ambapp_bus *abus)

## Functions

- int **ambapp_scan** (struct ambapp_bus *abus, unsigned int ioarea, ambapp_memcpy_t memfunc, struct ambapp_mmap *mmaps)
- void **ambapp_freq_init** (struct ambapp_bus *abus, struct ambapp_dev *dev, unsigned int freq)
- unsigned int **ambapp_freq_get** (struct ambapp_bus *abus, struct ambapp_dev *dev)
- int **ambapp_for_each** (struct ambapp_bus *abus, unsigned int options, int vendor, int device, ambapp_↩ func_t func, void *arg)
- int **ambapp_find_by_idx** (struct ambapp_dev *dev, int index, void *pcount)
- int **ambapp_dev_count** (struct ambapp_bus *abus, unsigned int options, int vendor, int device)
- void **ambapp_print** (struct ambapp_bus *abus, int show_depth)
- int **ambapp_alloc_dev** (struct ambapp_dev *dev, void *owner)
- void **ambapp_free_dev** (struct ambapp_dev *dev)
- struct ambapp_dev * **ambapp_find_parent** (struct ambapp_dev *dev)
- int **ambapp_depth** (struct ambapp_dev *dev)
- char * **ambapp_device_id2str** (int vendor, int id)
- char * **ambapp_vendor_id2str** (int vendor)
- int **ambapp_vendev_id2str** (int vendor, int id, char *buf)
- int **ambapp_find_apbslv** (struct ambapp_bus *abus, int vendor, int device, struct ambapp_apb_info *dev)
- int **ambapp_find_apbslv_next** (struct ambapp_bus *abus, int vendor, int device, struct ambapp_apb_info *dev, int index)
- int **ambapp_find_apbslvs_next** (struct ambapp_bus *abus, int vendor, int device, struct ambapp_apb_info *dev, int index, int maxno)
- int **ambapp_find_apbslvs** (struct ambapp_bus *abus, int vendor, int device, struct ambapp_apb_info *dev, int maxno)
- int **ambapp_find_ahbslv** (struct ambapp_bus *abus, int vendor, int device, struct ambapp_ahb_info *dev)
- int **ambapp_find_ahbslv_next** (struct ambapp_bus *abus, int vendor, int device, struct ambapp_ahb_info *dev, int index)
- int **ambapp_find_ahbslvs_next** (struct ambapp_bus *abus, int vendor, int device, struct ambapp_ahb_info *dev, int index, int maxno)
- int **ambapp_find_ahbslvs** (struct ambapp_bus *abus, int vendor, int device, struct ambapp_ahb_info *dev, int maxno)
- int **ambapp_get_number_ahbslv_devices** (struct ambapp_bus *abus, int vendor, int device)
- int **ambapp_get_number_apbslv_devices** (struct ambapp_bus *abus, int vendor, int device)

### 8.1.1 Detailed Description

AMBA Plug & Play routines.

## 8.2 API

This group contains the API components.

### Modules

- Application Configuration Information

  *The application configuration information group provides an API to get the configuration of an application.*
- Application Configuration Options
- Associativity Routines
- Base Definitions

  *This group contains basic macros and defines to give access to compiler-specific features.*
- Classic

  *This group contains the Classic API managers.*
- IO

  *IO.*
- RTEMS Test Framework

  *The RTEMS Test Framework helps you to write tests.*
- Test Support

  *Test support functions.*
- Tracing

  *Tracing.*

### Files

- file testrun.c

  *Implementation of rtems_test_run_default().*

### 8.2.1 Detailed Description

This group contains the API components.

API.

This group contains the RTEMS Application Programming Interface (API).

## 8.3 API Mutex Handler

Provides routines to ensure mutual exclusion on API level.

### Modules

- RTEMS Allocator Mutex

  *Protection for all memory allocations and deallocations in RTEMS.*

### Files

- file apimutex.h

  *API Mutex Handler API.*
- file muteximpl.h

  *Structures for the implementation of mutexes.*
- file apimutexisowner.c
- file apimutexlock.c

  *Acquires the specified API mutex.*
- file apimutexunlock.c

  *Releases the Specified API Mutex.*

### Classes

- struct API_Mutex_Control

  *Control block used to manage each API mutex.*
- struct Mutex_Control
- struct Mutex_recursive_Control

### Macros

- #define API_MUTEX_INITIALIZER(name) { _MUTEX_RECURSIVE_NAMED_INITIALIZER( name ), 0 }

  *Statically initialize an API mutex.*

### Functions

- void _API_Mutex_Lock (API_Mutex_Control *mutex)

  *Acquires the specified API mutex.*
- void _API_Mutex_Unlock (API_Mutex_Control *mutex)

  *Releases the specified API mutex.*
- bool _API_Mutex_Is_owner (const API_Mutex_Control *mutex)

  *Checks if the specified API mutex is owned by the executing thread.*

### 8.3.1 Detailed Description

Provides routines to ensure mutual exclusion on API level.

## 8.3.2 Function Documentation

### 8.3.2.1 _API_Mutex_Is_owner()

```
bool _API_Mutex_Is_owner (
            const API_Mutex_Control * mutex )
```

Checks if the specified API mutex is owned by the executing thread.

**Parameters**

| | | |
|---|---|---|
| in | *mutex* | The API mutex to check the owner from. |

Definition at line 23 of file apimutexisowner.c.

### 8.3.2.2 _API_Mutex_Lock()

```
void _API_Mutex_Lock (
            API_Mutex_Control * mutex )
```

Acquires the specified API mutex.

**Parameters**

| | | |
|---|---|---|
| in,out | *mutex* | The API mutex to acquire. |

Definition at line 25 of file apimutexlock.c.

### 8.3.2.3 _API_Mutex_Unlock()

```
void _API_Mutex_Unlock (
            API_Mutex_Control * mutex )
```

Releases the specified API mutex.

**Parameters**

| | | |
|---|---|---|
| in,out | *mutex* | The API mutex to release. |

Definition at line 25 of file apimutexunlock.c.

## 8.4 Address Handler

Support for Address Manipulation.

### Files

- file address.h

  *Information Required to Manipulate Physical Addresses.*

### Functions

- RTEMS_INLINE_ROUTINE void * _Addresses_Add_offset (const void *base, uintptr_t offset)

  *Adds offset to an address.*
- RTEMS_INLINE_ROUTINE void * _Addresses_Subtract_offset (const void *base, uintptr_t offset)

  *Subtracts offset from an address.*
- RTEMS_INLINE_ROUTINE intptr_t _Addresses_Subtract (const void *left, const void *right)

  *Subtracts two addresses.*
- RTEMS_INLINE_ROUTINE bool _Addresses_Is_aligned (const void *address)

  *Checks if address is aligned.*
- RTEMS_INLINE_ROUTINE bool _Addresses_Is_in_range (const void *address, const void *base, const void *limit)

  *Checks if address is in range.*
- RTEMS_INLINE_ROUTINE void * _Addresses_Align_up (void *address, size_t alignment)

  *Aligns address to nearest multiple of alignment, rounding up.*
- RTEMS_INLINE_ROUTINE void * _Addresses_Align_down (void *address, size_t alignment)

  *Aligns address to nearest multiple of alignment, truncating.*

### 8.4.1 Detailed Description

Support for Address Manipulation.

This handler encapsulates functionality which abstracts address manipulation in a portable manner.

### 8.4.2 Function Documentation

#### 8.4.2.1 _Addresses_Add_offset()

```
RTEMS_INLINE_ROUTINE void* _Addresses_Add_offset (
             const void * base,
             uintptr_t offset )
```

Adds offset to an address.

This function is used to add an *offset* to a *base* address. It returns the resulting address. This address is typically converted to an access type before being used further.

**Parameters**

| | |
|---|---|
| *base* | The base address to add the offset to. |
| *offset* | The offset to add to *base*. |

**Returns**

This method returns the resulting address.

Definition at line 55 of file address.h.

#### 8.4.2.2 _Addresses_Align_down()

```
RTEMS_INLINE_ROUTINE void* _Addresses_Align_down (
             void * address,
             size_t alignment )
```

Aligns address to nearest multiple of alignment, truncating.

This function returns the given address aligned to the given alignment. If the address already is aligned, or if alignment is 0, the address is returned as is. The returned address is less than or equal to the given address.

**Parameters**

| | |
|---|---|
| *address* | The address to align to the given alignment. |
| *alignment* | The desired alignment for the address. It must be a power of two. |

**Returns**

Returns the aligned address.

Definition at line 182 of file address.h.

#### 8.4.2.3 _Addresses_Align_up()

```
RTEMS_INLINE_ROUTINE void* _Addresses_Align_up (
             void * address,
             size_t alignment )
```

Aligns address to nearest multiple of alignment, rounding up.

This function returns the given address aligned to the given alignment. If the address already is aligned, or if alignment is 0, the address is returned as is. The returned address is greater than or equal to the given address.

**Parameters**

| | |
|---|---|
| *address* | The address to align to the given alignment. |
| *alignment* | The desired alignment for the address. It must be a power of two. |

**Returns**

Returns the aligned address.

Definition at line 160 of file address.h.

#### 8.4.2.4 _Addresses_Is_aligned()

```
RTEMS_INLINE_ROUTINE bool _Addresses_Is_aligned (
             const void * address )
```

Checks if address is aligned.

This function returns true if the given address is correctly aligned for this processor and false otherwise. Proper alignment is based on correctness and efficiency.

**Parameters**

| | |
|---|---|
| *address* | The address being checked for alignment. |

**Return values**

| | |
|---|---|
| *true* | The *address* is aligned. |
| *false* | The *address* is not aligned. |

Definition at line 115 of file address.h.

#### 8.4.2.5 _Addresses_Is_in_range()

```
RTEMS_INLINE_ROUTINE bool _Addresses_Is_in_range (
             const void * address,
             const void * base,
             const void * limit )
```

Checks if address is in range.

This function returns true if the given address is within the memory range specified and false otherwise. *base* is the address of the first byte in the memory range and *limit* is the address of the last byte in the memory range. The base address is assumed to be lower than the limit address.

**Parameters**

| | |
|---|---|
| *address* | The address to check if it is in the given range. |
| *base* | The lowest address of the range to check against. |
| *limit* | The highest address of the range to check against. |

**Return values**

| | |
|---|---|
| *true* | The *address* is within the memory range specified |
| *false* | The *address* is not within the memory range specified. |

Definition at line 138 of file address.h.

#### 8.4.2.6 _Addresses_Subtract()

```
RTEMS_INLINE_ROUTINE intptr_t _Addresses_Subtract (
            const void * left,
            const void * right )
```

Subtracts two addresses.

This function is used to subtract two addresses. It returns the resulting offset.

**Parameters**

| | |
|---|---|
| *left* | The address on the left hand side of the subtraction. |
| *right* | The address on the right hand side of the subtraction. |

**Returns**

This method returns the resulting address.

Definition at line 95 of file address.h.

#### 8.4.2.7 _Addresses_Subtract_offset()

```
RTEMS_INLINE_ROUTINE void* _Addresses_Subtract_offset (
            const void * base,
            uintptr_t offset )
```

Subtracts offset from an address.

This function is used to subtract an *offset* from a *base* address. It returns the resulting address. This address is typically converted to an access type before being used further.

**Parameters**

| | |
|---|---|
| *base* | The base address to subtract the offset from. |
| *offset* | The offset to subtract from *base*. |

**Returns**

This method returns the resulting address.

Definition at line 76 of file address.h.

## 8.5 Application Configuration

Evaluation of Application Configuration Options.

### Files

- file bdbuf.h

  *Evaluate Block Device Cache Configuration Options.*
- file bsp.h

  *Evaluate BSP Related Configuration Options.*
- file clock.h

  *Evaluate Clock Configuration Options.*
- file console.h

  *Evaluate Console Driver Configuration Options.*
- file extensions.h

  *Evaluate User Extensions Configuration Options.*
- file inittask.h

  *Evaluate User Initialization Task Configuration Options.*
- file initthread.h

  *Evaluate POSIX Initialization Thread Configuration Options.*
- file iodrivers.h

  *Evaluate IO Driver Configuration Options.*
- file libio.h

  *Evaluate IO Library Configuration Options.*
- file libpci.h

  *This header file evaluates PCI library configuration options.*
- file malloc.h

  *Evaluate C Program Heap Configuration Options.*
- file mpci.h

  *Evaluate MPCI Configuration Options.*
- file newlib.h

  *Evaluate Newlib Configuration Options.*
- file objectsclassic.h

  *Evaluate Classic API Objects Configuration Options.*
- file objectsposix.h

  *Evaluate POSIX API Objects Configuration Options.*
- file obsolete.h

  *Evaluate Obsolete Configuration Options.*
- file percpu.h

  *Evaluate Per-CPU Configuration Options.*
- file scheduler.h

  *Evaluate Scheduler Configuration Options.*
- file threads.h

  *Evaluate Thread Configuration Options.*
- file unlimited.h

  *Evaluate Unlimited Objects Configuration Options.*
- file wkspace.h

  *Evaluate Workspace Configuration Options.*
- file wkspacesupport.h

  *Configuration Options Workspace Support Macros.*
- file confdefs.h

  *Evaluate Configuration Options.*

### 8.5.1 Detailed Description

Evaluation of Application Configuration Options.

This group contains header files which evaluate the configuration options specified by the application.

## 8.6 Application Configuration Information

The application configuration information group provides an API to get the configuration of an application.

### Files

- file config.h

  *This header file provides parts of the application configuration information API.*

### Classes

- struct rtems_api_configuration_table

  *This structure contains a summary of the Classic API configuration.*

### Macros

- #define rtems_configuration_get_do_zero_of_workspace() _Memory_Zero_before_use

  *Indicates if the RTEMS Workspace is configured to be zeroed during system initialization for this application.*
- #define rtems_configuration_get_idle_task() _Thread_Idle_body

  *Gets the IDLE task entry of this application.*
- #define rtems_configuration_get_idle_task_stack_size() _Thread_Idle_stack_size

  *Gets the IDLE task stack size in bytes of this application.*
- #define rtems_configuration_get_interrupt_stack_size() ((size_t) _ISR_Stack_size)

  *Gets the interrupt stack size in bytes of this application.*
- #define rtems_configuration_get_maximum_processors() _SMP_Processor_configured_maximum

  *Gets the maximum number of processors configured for this application.*
- #define rtems_configuration_get_microseconds_per_tick() _Watchdog_Microseconds_per_tick

  *Gets the number of microseconds per clock tick configured for this application.*
- #define rtems_configuration_get_milliseconds_per_tick() ( _Watchdog_Microseconds_per_tick / 1000 )

  *Gets the number of milliseconds per clock tick configured for this application.*
- #define rtems_configuration_get_nanoseconds_per_tick() _Watchdog_Nanoseconds_per_tick

  *Gets the number of microseconds per clock tick configured for this application.*
- #define rtems_configuration_get_number_of_initial_extensions() ((uint32_t) _User_extensions_Initial_count)

  *Gets the number of initial extensions configured for this application.*
- #define rtems_configuration_get_stack_allocate_hook() _Stack_Allocator_allocate

  *Gets the thread stack allocator allocate hook configured for this application.*
- #define rtems_configuration_get_stack_allocate_init_hook() _Stack_Allocator_initialize

  *Gets the thread stack allocator initialization hook configured for this application.*
- #define rtems_configuration_get_stack_allocator_avoids_work_space() _Stack_Allocator_avoids_workspace

  *Indicates if the thread stack allocator is configured to avoid the RTEMS Workspace for this application.*
- #define rtems_configuration_get_stack_free_hook() _Stack_Allocator_free

  *Gets the thread stack allocator free hook configured for this application.*
- #define rtems_configuration_get_ticks_per_timeslice() _Watchdog_Ticks_per_timeslice

  *Gets the clock ticks per timeslice configured for this application.*
- #define rtems_configuration_get_unified_work_area() _Workspace_Is_unified

  *Indicates if the RTEMS Workspace and C Program Heap are configured to be unified for this application.*
- #define rtems_configuration_get_user_extension_table() _User_extensions_Initial_extensions

  *Gets the initial extensions table configured for this application.*

- #define rtems_configuration_get_user_multiprocessing_table() NULL

  *Gets the MPCI configuration table configured for this application.*
- #define rtems_configuration_get_work_space_size()

  *Gets the RTEMS Workspace size in bytes configured for this application.*
- #define RTEMS_HAS_HARDWARE_FP CPU_HARDWARE_FP

  *This constant evaluates to TRUE, if this processor variant has hardware floating point support, otherwise to FALSE.*
- #define rtems_resource_is_unlimited(_resource) _Objects_Is_unlimited(_resource)

  *Indicates if the resource is unlimited.*
- #define rtems_resource_maximum_per_allocation(_resource) _Objects_Maximum_per_allocation(_resource)

  *Gets the maximum number per allocation of a resource number.*
- #define RTEMS_UNLIMITED_OBJECTS OBJECTS_UNLIMITED_OBJECTS

  *This flag is used in augment a resource number so that it indicates an unlimited resource.*
- #define rtems_resource_unlimited(_resource) ( (_resource) | RTEMS_UNLIMITED_OBJECTS )

  *Augments the resource number so that it indicates an unlimited resource.*

## Typedefs

- typedef Stack_Allocator_allocate rtems_stack_allocate_hook

  *A thread stack allocator allocate handler shall have this type.*
- typedef Stack_Allocator_initialize rtems_stack_allocate_init_hook

  *A thread stack allocator initialization handler shall have this type.*
- typedef Stack_Allocator_free rtems_stack_free_hook

  *A thread stack allocator free handler shall have this type.*

## Functions

- const char * rtems_get_copyright_notice (void)

  *Gets the RTEMS copyright notice.*
- uint32_t rtems_configuration_get_maximum_extensions (void)

  *Gets the maximum number of Classic API User Extensions configured for this application.*
- uintptr_t rtems_configuration_get_stack_space_size (void)

  *Gets the thread stack space size in bytes of configured for this application.*
- const char * rtems_get_version_string (void)

  *Gets the RTEMS version string.*
- const rtems_api_configuration_table * rtems_configuration_get_rtems_api_configuration (void)

  *Gets the Classic API Configuration Table of this application.*
- uint32_t rtems_configuration_get_maximum_barriers (void)

  *Gets the maximum number of Classic API Barriers configured for this application.*
- uint32_t rtems_configuration_get_maximum_message_queues (void)

  *Gets the maximum number of Classic API Message Queues configured for this application.*
- uint32_t rtems_configuration_get_maximum_partitions (void)

  *Gets the maximum number of Classic API Partitions configured for this application.*
- uint32_t rtems_configuration_get_maximum_periods (void)

  *Gets the maximum number of Classic API Rate Monotonic Periods configured for this application.*
- uint32_t rtems_configuration_get_maximum_ports (void)

  *Gets the maximum number of Classic API Dual-Ported Memories configured for this application.*
- uint32_t rtems_configuration_get_maximum_regions (void)

  *Gets the maximum number of Classic API Regions configured for this application.*

- uint32_t rtems_configuration_get_maximum_semaphores (void)

  *Gets the maximum number of Classic API Semaphores configured for this application.*
- uint32_t rtems_configuration_get_maximum_tasks (void)

  *Gets the maximum number of Classic API Tasks configured for this application.*
- uint32_t rtems_configuration_get_maximum_timers (void)

  *Gets the maximum number of Classic API Timers configured for this application.*

## 8.6.1 Detailed Description

The application configuration information group provides an API to get the configuration of an application.

Some interfaces of this API are also used to define application configuration option values, for example rtems_resource_unlimited().

## 8.6.2 Macro Definition Documentation

### 8.6.2.1 rtems_configuration_get_do_zero_of_workspace

```
#define rtems_configuration_get_do_zero_of_workspace( ) _Memory_Zero_before_use
```

Indicates if the RTEMS Workspace is configured to be zeroed during system initialization for this application.

See CONFIGURE_ZERO_WORKSPACE_AUTOMATICALLY.

**Returns**

Returns true, if the RTEMS Workspace is configured to be zeroed during system initialization for this application, otherwise false.

Definition at line 117 of file config.h.

### 8.6.2.2 rtems_configuration_get_idle_task

```
#define rtems_configuration_get_idle_task( ) _Thread_Idle_body
```

Gets the IDLE task entry of this application.

See CONFIGURE_IDLE_TASK_BODY.

**Returns**

Returns the IDLE task entry of this application.

Definition at line 130 of file config.h.

#### 8.6.2.3 rtems_configuration_get_idle_task_stack_size

```
#define rtems_configuration_get_idle_task_stack_size( ) _Thread_Idle_stack_size
```

Gets the IDLE task stack size in bytes of this application.

See CONFIGURE_IDLE_TASK_STACK_SIZE.

**Returns**

Returns the IDLE task stack size in bytes of this application.

Definition at line 143 of file config.h.

#### 8.6.2.4 rtems_configuration_get_interrupt_stack_size

```
#define rtems_configuration_get_interrupt_stack_size( )  ((size_t) _ISR_Stack_size)
```

Gets the interrupt stack size in bytes of this application.

See CONFIGURE_INTERRUPT_STACK_SIZE.

**Returns**

Returns the interrupt stack size in bytes of this application.

Definition at line 156 of file config.h.

#### 8.6.2.5 rtems_configuration_get_maximum_processors

```
#define rtems_configuration_get_maximum_processors( )  _SMP_Processor_configured_maximum
```

Gets the maximum number of processors configured for this application.

The actual number of processors available to the application is returned by rtems_scheduler_get_processor_maximum() which less than or equal to the configured maximum number of processors (CONFIGURE_MAXIMUM_PROCESSORS).

In uniprocessor configurations, this macro is a compile time constant which evaluates to one.

**Returns**

Returns the maximum number of processors configured for this application.

Definition at line 192 of file config.h.

#### 8.6.2.6 rtems_configuration_get_microseconds_per_tick

```
#define rtems_configuration_get_microseconds_per_tick( )  _Watchdog_Microseconds_per_tick
```

Gets the number of microseconds per clock tick configured for this application.

See CONFIGURE_MICROSECONDS_PER_TICK.

**Returns**

Returns the number of microseconds per clock tick configured for this application.

Definition at line 208 of file config.h.

#### 8.6.2.7 rtems_configuration_get_milliseconds_per_tick

```
#define rtems_configuration_get_milliseconds_per_tick( )  ( _Watchdog_Microseconds_per_tick /
1000 )
```

Gets the number of milliseconds per clock tick configured for this application.

See CONFIGURE_MICROSECONDS_PER_TICK.

**Returns**

Returns the number of milliseconds per clock tick configured for this application.

Definition at line 224 of file config.h.

#### 8.6.2.8 rtems_configuration_get_nanoseconds_per_tick

```
#define rtems_configuration_get_nanoseconds_per_tick( )  _Watchdog_Nanoseconds_per_tick
```

Gets the number of microseconds per clock tick configured for this application.

See CONFIGURE_MICROSECONDS_PER_TICK.

**Returns**

Returns the number of microseconds per clock tick configured for this application.

Definition at line 240 of file config.h.

#### 8.6.2.9 rtems_configuration_get_number_of_initial_extensions

```
#define rtems_configuration_get_number_of_initial_extensions( )  ((uint32_t) _User_extensions_Initial_count)
```

Gets the number of initial extensions configured for this application.

See CONFIGURE_INITIAL_EXTENSIONS.

**Returns**

Returns the number of initial extensions configured for this application.

Definition at line 256 of file config.h.

#### 8.6.2.10 rtems_configuration_get_stack_allocate_hook

```
#define rtems_configuration_get_stack_allocate_hook( ) _Stack_Allocator_allocate
```

Gets the thread stack allocator allocate hook configured for this application.

See CONFIGURE_TASK_STACK_ALLOCATOR.

**Returns**

Returns the thread stack allocator allocate hook configured for this application.

Definition at line 272 of file config.h.

#### 8.6.2.11 rtems_configuration_get_stack_allocate_init_hook

```
#define rtems_configuration_get_stack_allocate_init_hook( )  _Stack_Allocator_initialize
```

Gets the thread stack allocator initialization hook configured for this application.

See CONFIGURE_TASK_STACK_ALLOCATOR_INIT.

**Returns**

Returns the thread stack allocator initialization hook configured for this application.

Definition at line 287 of file config.h.

#### 8.6.2.12 rtems_configuration_get_stack_allocator_avoids_work_space

```
#define rtems_configuration_get_stack_allocator_avoids_work_space( )  _Stack_Allocator_avoids_workspace
```

Indicates if the thread stack allocator is configured to avoid the RTEMS Workspace for this application.

See CONFIGURE_TASK_STACK_ALLOCATOR_AVOIDS_WORK_SPACE.

**Returns**

Returns true, if the thread stack allocator is configured to avoid the RTEMS Workspace for this application, otherwise false.

Definition at line 303 of file config.h.

#### 8.6.2.13 rtems_configuration_get_stack_free_hook

```
#define rtems_configuration_get_stack_free_hook( ) _Stack_Allocator_free
```

Gets the thread stack allocator free hook configured for this application.

See CONFIGURE_TASK_STACK_DEALLOCATOR.

**Returns**

Returns the thread stack allocator free hook configured for this application.

Definition at line 319 of file config.h.

#### 8.6.2.14 rtems_configuration_get_ticks_per_timeslice

```
#define rtems_configuration_get_ticks_per_timeslice( )  _Watchdog_Ticks_per_timeslice
```

Gets the clock ticks per timeslice configured for this application.

See CONFIGURE_TICKS_PER_TIMESLICE.

**Returns**

Returns the clock ticks per timeslice configured for this application.

Definition at line 346 of file config.h.

#### 8.6.2.15 rtems_configuration_get_unified_work_area

```
#define rtems_configuration_get_unified_work_area( ) _Workspace_Is_unified
```

Indicates if the RTEMS Workspace and C Program Heap are configured to be unified for this application.

See CONFIGURE_UNIFIED_WORK_AREAS.

**Returns**

Returns true, if the RTEMS Workspace and C Program Heap are configured to be unified for this application, otherwise false.

Definition at line 362 of file config.h.

#### 8.6.2.16 rtems_configuration_get_user_extension_table

```
#define rtems_configuration_get_user_extension_table( ) _User_extensions_Initial_extensions
```

Gets the initial extensions table configured for this application.

**Returns**

Returns the pointer to the initial extensions table configured for this application.

Definition at line 374 of file config.h.

#### 8.6.2.17 rtems_configuration_get_user_multiprocessing_table

```
#define rtems_configuration_get_user_multiprocessing_table( ) NULL
```

Gets the MPCI configuration table configured for this application.

**Returns**

Returns the pointer to the MPCI configuration table configured for this application.

Definition at line 391 of file config.h.

#### 8.6.2.18 rtems_configuration_get_work_space_size

```
#define rtems_configuration_get_work_space_size( )
```

**Value:**

```
( _Workspace_Size + \
  ( rtems_configuration_get_stack_allocator_avoids_work_space() ? \
    0 : rtems_configuration_get_stack_space_size() ) )
```

Gets the RTEMS Workspace size in bytes configured for this application.

**Returns**

Returns the RTEMS Workspace size in bytes configured for this application.

Definition at line 416 of file config.h.

#### 8.6.2.19 rtems_resource_is_unlimited

```
#define rtems_resource_is_unlimited(
              _resource )  _Objects_Is_unlimited(_resource)
```

Indicates if the resource is unlimited.

This function is implemented as a macro and can be used to define compile time constants.

**Parameters**

| | |
|---|---|
| *_resource* | is the resource number. |

**Returns**

Returns true, if the resource is unlimited, otherwise false.

Definition at line 445 of file config.h.

#### 8.6.2.20 rtems_resource_maximum_per_allocation

```
#define rtems_resource_maximum_per_allocation(
              _resource )  _Objects_Maximum_per_allocation(_resource)
```

Gets the maximum number per allocation of a resource number.

This function is implemented as a macro and can be used to define compile time constants.

**Parameters**

| | |
|---|---|
| *_resource* | is the resource number. |

**Returns**

Returns the maximum number per allocation of a resource number.

Definition at line 462 of file config.h.

#### 8.6.2.21 rtems_resource_unlimited

```
#define rtems_resource_unlimited(
              _resource )  ( (_resource) | RTEMS_UNLIMITED_OBJECTS )
```

Augments the resource number so that it indicates an unlimited resource.

This function is implemented as a macro and can be used to define compile time constants.

**Parameters**

| | |
|---|---|
| *_resource* | is the resource number to augment. |

**Returns**

Returns the resource number augmented to indicate an unlimited resource.

Definition at line 518 of file config.h.

### 8.6.3 Function Documentation

#### 8.6.3.1 rtems_configuration_get_maximum_barriers()

```
uint32_t rtems_configuration_get_maximum_barriers (
             void  )
```

Gets the maximum number of Classic API Barriers configured for this application.

See CONFIGURE_MAXIMUM_BARRIERS.

**Returns**

Returns the maximum number of Classic API Barriers configured for this application.

#### 8.6.3.2 rtems_configuration_get_maximum_extensions()

```
uint32_t rtems_configuration_get_maximum_extensions (
            void  )
```

Gets the maximum number of Classic API User Extensions configured for this application.

See CONFIGURE_MAXIMUM_USER_EXTENSIONS.

**Returns**

Returns the maximum number of Classic API User Extensions configured for this application.

#### 8.6.3.3 rtems_configuration_get_maximum_message_queues()

```
uint32_t rtems_configuration_get_maximum_message_queues (
            void  )
```

Gets the maximum number of Classic API Message Queues configured for this application.

See CONFIGURE_MAXIMUM_MESSAGE_QUEUES.

**Returns**

Returns the maximum number of Classic API Message Queues configured for this application.

#### 8.6.3.4 rtems_configuration_get_maximum_partitions()

```
uint32_t rtems_configuration_get_maximum_partitions (
            void  )
```

Gets the maximum number of Classic API Partitions configured for this application.

See CONFIGURE_MAXIMUM_PARTITIONS.

**Returns**

Returns the maximum number of Classic API Partitions configured for this application.

#### 8.6.3.5 rtems_configuration_get_maximum_periods()

```
uint32_t rtems_configuration_get_maximum_periods (
            void  )
```

Gets the maximum number of Classic API Rate Monotonic Periods configured for this application.

See CONFIGURE_MAXIMUM_PERIODS.

**Returns**

Returns the maximum number of Classic API Rate Monotonic Periods configured for this application.

#### 8.6.3.6 rtems_configuration_get_maximum_ports()

```
uint32_t rtems_configuration_get_maximum_ports (
            void  )
```

Gets the maximum number of Classic API Dual-Ported Memories configured for this application.

See CONFIGURE_MAXIMUM_PORTS.

**Returns**

Returns the maximum number of Classic API Dual-Ported Memories configured for this application.

#### 8.6.3.7 rtems_configuration_get_maximum_regions()

```
uint32_t rtems_configuration_get_maximum_regions (
            void  )
```

Gets the maximum number of Classic API Regions configured for this application.

See CONFIGURE_MAXIMUM_REGIONS.

**Returns**

Returns the maximum number of Classic API Regions configured for this application.

#### 8.6.3.8 rtems_configuration_get_maximum_semaphores()

```
uint32_t rtems_configuration_get_maximum_semaphores (
            void  )
```

Gets the maximum number of Classic API Semaphores configured for this application.

See CONFIGURE_MAXIMUM_SEMAPHORES.

**Returns**

Returns the maximum number of Classic API Semaphores configured for this application.

#### 8.6.3.9 rtems_configuration_get_maximum_tasks()

```
uint32_t rtems_configuration_get_maximum_tasks (
            void  )
```

Gets the maximum number of Classic API Tasks configured for this application.

See CONFIGURE_MAXIMUM_TASKS.

**Returns**

Returns the maximum number of Classic API Tasks configured for this application.

#### 8.6.3.10 rtems_configuration_get_maximum_timers()

```
uint32_t rtems_configuration_get_maximum_timers (
            void  )
```

Gets the maximum number of Classic API Timers configured for this application.

See CONFIGURE_MAXIMUM_TIMERS.

**Returns**

Returns the maximum number of Classic API Timers configured for this application.

#### 8.6.3.11 rtems_configuration_get_rtems_api_configuration()

```
const rtems_api_configuration_table* rtems_configuration_get_rtems_api_configuration (
            void  )
```

Gets the Classic API Configuration Table of this application.

**Returns**

Returns the pointer to the Classic API Configuration Table of this application.

#### 8.6.3.12 rtems_configuration_get_stack_space_size()

```
uintptr_t rtems_configuration_get_stack_space_size (
            void  )
```

Gets the thread stack space size in bytes of configured for this application.

**Returns**

Returns the thread stack space size in bytes of configured for this application.

Definition at line 37 of file configstackspacesize.c.

#### 8.6.3.13 rtems_get_copyright_notice()

```
const char* rtems_get_copyright_notice (
            void  )
```

Gets the RTEMS copyright notice.

**Returns**

Returns the pointer to the RTEMS copyright notice.

#### 8.6.3.14 rtems_get_version_string()

```
const char* rtems_get_version_string (
            void  )
```

Gets the RTEMS version string.

**Returns**

Returns the pointer to the RTEMS version string.

Definition at line 25 of file getversionstring.c.

## 8.7 Application Configuration Options

### Modules


- BSP Related Configuration Options
- Block Device Cache Configuration
- Classic API Configuration
- Classic API Initialization Task Configuration
- Device Driver Configuration
- Event Recording Configuration
- Filesystem Configuration
- General Scheduler Configuration
- General System Configuration
- Idle Task Configuration
- Multiprocessing Configuration
- POSIX API Configuration
- POSIX Initialization Thread Configuration
- Task Stack Allocator Configuration


### 8.7.1 Detailed Description

## 8.8 Assert Handler

Support for Assert Statements.

### Files

- file assert.h

  *Information for the Assert Handler.*

### Macros

- #define _Assert(_e) ( ( void ) 0 )

  *Assertion similar to assert() controlled via RTEMS_DEBUG instead of NDEBUG.*
- #define _SMP_Assert(_e) _Assert( _e )

  *Like _Assert(), but only armed if RTEMS_SMP is defined.*

### 8.8.1 Detailed Description

Support for Assert Statements.

## 8.9 Associativity Routines

### Files

- file assoc.h

  *RTEMS Associativity Routines.*

### Classes

- struct rtems_assoc_t
- struct rtems_assoc_32_pair

### Macros

- #define **RTEMS_ASSOC_DEFAULT_NAME** "(default)"

### Functions

- const rtems_assoc_t * rtems_assoc_ptr_by_name (const rtems_assoc_t *, const char *)

  *RTEMS Associate Pointer by Name.*
- const rtems_assoc_t * rtems_assoc_ptr_by_remote (const rtems_assoc_t *, uint32_t)

  *RTEMS Associate Pointer by Remote.*
- uint32_t **rtems_assoc_remote_by_local** (const rtems_assoc_t *, uint32_t)
- uint32_t rtems_assoc_local_by_remote (const rtems_assoc_t *, uint32_t)

  *RTEMS Associate Local by Remote.*
- uint32_t rtems_assoc_remote_by_name (const rtems_assoc_t *, const char *)

  *RTEMS Associate Remote by Name.*
- uint32_t rtems_assoc_local_by_name (const rtems_assoc_t *, const char *)

  *RTEMS Associate Local by Name.*
- const char * rtems_assoc_name_by_local (const rtems_assoc_t *, uint32_t)

  *RTEMS Associate Name by Local.*
- const char * rtems_assoc_name_by_remote (const rtems_assoc_t *, uint32_t)

  *RTEMS Associate Name by Remote.*
- uint32_t rtems_assoc_remote_by_local_bitfield (const rtems_assoc_t *, uint32_t)

  *RTEMS Assoc Routines.*
- char * rtems_assoc_name_by_local_bitfield (const rtems_assoc_t *, uint32_t, char *)

  *RTEMS Associate Name by Local Bitfield.*
- char * rtems_assoc_name_by_remote_bitfield (const rtems_assoc_t *, uint32_t, char *)

  *RTEMS Associate Name by Remote Bitfield.*
- uint32_t **rtems_assoc_local_by_remote_bitfield** (const rtems_assoc_t *, uint32_t)
- const rtems_assoc_t * rtems_assoc_ptr_by_local (const rtems_assoc_t *ap, uint32_t local_value)

  *RTEMS Associate Pointer by Local.*
- size_t rtems_assoc_32_to_string (uint32_t value, char *buffer, size_t buffer_size, const rtems_assoc_32_pair *pairs, size_t pair_count, const char *separator, const char *fallback)

  *Converts the specified value into a text representation.*
- size_t rtems_assoc_thread_states_to_string (uint32_t states, char *buffer, size_t buffer_size)

  *Converts the specified thread states into a text representation.*

### 8.9.1 Detailed Description

RTEMS associativity routines. Mainly used to convert a value from one space to another (eg: our errno's to host errno's and vice-versa)

### 8.9.2 Function Documentation

#### 8.9.2.1 rtems_assoc_32_to_string()

```
size_t rtems_assoc_32_to_string (
            uint32_t value,
            char * buffer,
            size_t buffer_size,
            const rtems_assoc_32_pair * pairs,
            size_t pair_count,
            const char * separator,
            const char * fallback )
```

Converts the specified value into a text representation.

**Parameters**

| | | |
|---|---|---|
| in | *value* | The value to convert. |
| in | *buffer* | The buffer for the text representation. |
| in | *buffer_size* | The buffer size in characters. |
| in | *pairs* | Names for particular bits. |
| in | *pair_count* | Count of pairs. |
| in | *separator* | Separator between individual names. |
| in | *fallback* | Fallback value in case no bits contained in the pairs are set in the value. |

**Return values**

| | |
|---|---|
| *The* | length of the text representation. May be greater than or equal to the buffer size if truncation occurred. |

#### 8.9.2.2 rtems_assoc_thread_states_to_string()

```
size_t rtems_assoc_thread_states_to_string (
            uint32_t states,
            char * buffer,
            size_t buffer_size )
```

Converts the specified thread states into a text representation.

**Parameters**

| | | |
|---|---|---|
| in | *states* | The thread states to convert. |
| in | *buffer* | The buffer for the text representation. |
| in | *buffer_size* | The buffer size in characters. |

**Return values**

| | |
|---|---|
| *The* | length of the text representation. May be greater than or equal to the buffer size if truncation occurred. |

# 8.10 Atomic Operations

Support for atomic operations.

## Files

- file atomic.h

  *Atomic Operations API.*

## Macros

- #define **ATOMIC_ORDER_RELAXED** CPU_ATOMIC_ORDER_RELAXED
- #define **ATOMIC_ORDER_ACQUIRE** CPU_ATOMIC_ORDER_ACQUIRE
- #define **ATOMIC_ORDER_RELEASE** CPU_ATOMIC_ORDER_RELEASE
- #define **ATOMIC_ORDER_ACQ_REL** CPU_ATOMIC_ORDER_ACQ_REL
- #define **ATOMIC_ORDER_SEQ_CST** CPU_ATOMIC_ORDER_SEQ_CST
- #define **ATOMIC_INITIALIZER_UINT**(value) CPU_ATOMIC_INITIALIZER_UINT( value )
- #define **ATOMIC_INITIALIZER_ULONG**(value) CPU_ATOMIC_INITIALIZER_ULONG( value )
- #define **ATOMIC_INITIALIZER_UINTPTR**(value) CPU_ATOMIC_INITIALIZER_UINTPTR( value )
- #define **ATOMIC_INITIALIZER_FLAG** CPU_ATOMIC_INITIALIZER_FLAG
- #define **_Atomic_Fence**(order) _CPU_atomic_Fence( order )
- #define **_Atomic_Init_uint**(obj, desired) _CPU_atomic_Init_uint( obj, desired )
- #define **_Atomic_Init_ulong**(obj, desired) _CPU_atomic_Init_ulong( obj, desired )
- #define **_Atomic_Init_uintptr**(obj, desired) _CPU_atomic_Init_uintptr( obj, desired )
- #define **_Atomic_Load_uint**(obj, order) _CPU_atomic_Load_uint( obj, order )
- #define **_Atomic_Load_ulong**(obj, order) _CPU_atomic_Load_ulong( obj, order )
- #define **_Atomic_Load_uintptr**(obj, order) _CPU_atomic_Load_uintptr( obj, order )
- #define **_Atomic_Store_uint**(obj, desr, order) _CPU_atomic_Store_uint( obj, desr, order )
- #define **_Atomic_Store_ulong**(obj, desr, order) _CPU_atomic_Store_ulong( obj, desr, order )
- #define **_Atomic_Store_uintptr**(obj, desr, order) _CPU_atomic_Store_uintptr( obj, desr, order )
- #define **_Atomic_Fetch_add_uint**(obj, arg, order) _CPU_atomic_Fetch_add_uint( obj, arg, order )
- #define **_Atomic_Fetch_add_ulong**(obj, arg, order) _CPU_atomic_Fetch_add_ulong( obj, arg, order )
- #define **_Atomic_Fetch_add_uintptr**(obj, arg, order) _CPU_atomic_Fetch_add_uintptr( obj, arg, order )
- #define **_Atomic_Fetch_sub_uint**(obj, arg, order) _CPU_atomic_Fetch_sub_uint( obj, arg, order )
- #define **_Atomic_Fetch_sub_ulong**(obj, arg, order) _CPU_atomic_Fetch_sub_ulong( obj, arg, order )
- #define **_Atomic_Fetch_sub_uintptr**(obj, arg, order) _CPU_atomic_Fetch_sub_uintptr( obj, arg, order )
- #define **_Atomic_Fetch_or_uint**(obj, arg, order) _CPU_atomic_Fetch_or_uint( obj, arg, order )
- #define **_Atomic_Fetch_or_ulong**(obj, arg, order) _CPU_atomic_Fetch_or_ulong( obj, arg, order )
- #define **_Atomic_Fetch_or_uintptr**(obj, arg, order) _CPU_atomic_Fetch_or_uintptr( obj, arg, order )
- #define **_Atomic_Fetch_and_uint**(obj, arg, order) _CPU_atomic_Fetch_and_uint( obj, arg, order )
- #define **_Atomic_Fetch_and_ulong**(obj, arg, order) _CPU_atomic_Fetch_and_ulong( obj, arg, order )
- #define **_Atomic_Fetch_and_uintptr**(obj, arg, order) _CPU_atomic_Fetch_and_uintptr( obj, arg, order )
- #define **_Atomic_Exchange_uint**(obj, desr, order) _CPU_atomic_Exchange_uint( obj, desr, order )
- #define **_Atomic_Exchange_ulong**(obj, desr, order) _CPU_atomic_Exchange_ulong( obj, desr, order )
- #define **_Atomic_Exchange_uintptr**(obj, desr, order) _CPU_atomic_Exchange_uintptr( obj, desr, order )
- #define **_Atomic_Compare_exchange_uint**(obj, expected, desired, succ, fail) _CPU_atomic_Compare_exchange_uint( obj, expected, desired, succ, fail )
- #define **_Atomic_Compare_exchange_ulong**(obj, expected, desired, succ, fail) _CPU_atomic_Compare_exchange_ulong( obj, expected, desired, succ, fail )
- #define **_Atomic_Compare_exchange_uintptr**(obj, expected, desired, succ, fail) _CPU_atomic_Compare_exchange_uintptr( obj, expected, desired, succ, fail )
- #define **_Atomic_Flag_clear**(obj, order) _CPU_atomic_Flag_clear( obj, order )
- #define **_Atomic_Flag_test_and_set**(obj, order) _CPU_atomic_Flag_test_and_set( obj, order )

## Typedefs

- typedef CPU_atomic_Uint **Atomic_Uint**
- typedef CPU_atomic_Ulong **Atomic_Ulong**
- typedef CPU_atomic_Uintptr **Atomic_Uintptr**
- typedef CPU_atomic_Flag **Atomic_Flag**
- typedef CPU_atomic_Order **Atomic_Order**

### 8.10.1 Detailed Description

Support for atomic operations.

Atomic operations can be used to implement low-level synchronization primitives on SMP systems, like spin locks. All atomic operations are defined in terms of C11 (ISO/IEC 9899:2011) or C++11 (ISO/IEC 14882:2011). For documentation use the standard documents.

# 8.11 Atomic Operations CPU

Atomic Operations CPU API.

## Macros

- #define **_RTEMS_SCORE_CPUSTDATOMIC_USE_STDATOMIC**
- #define **CPU_ATOMIC_ORDER_RELAXED** memory_order_relaxed
- #define **CPU_ATOMIC_ORDER_ACQUIRE** memory_order_acquire
- #define **CPU_ATOMIC_ORDER_RELEASE** memory_order_release
- #define **CPU_ATOMIC_ORDER_ACQ_REL** memory_order_acq_rel
- #define **CPU_ATOMIC_ORDER_SEQ_CST** memory_order_seq_cst
- #define **CPU_ATOMIC_INITIALIZER_UINT**(value) ATOMIC_VAR_INIT( value )
- #define **CPU_ATOMIC_INITIALIZER_ULONG**(value) ATOMIC_VAR_INIT( value )
- #define **CPU_ATOMIC_INITIALIZER_UINTPTR**(value) ATOMIC_VAR_INIT( value )
- #define **CPU_ATOMIC_INITIALIZER_FLAG** ATOMIC_FLAG_INIT

## Typedefs

- typedef atomic_uint **CPU_atomic_Uint**
- typedef atomic_ulong **CPU_atomic_Ulong**
- typedef atomic_uintptr_t **CPU_atomic_Uintptr**
- typedef atomic_flag **CPU_atomic_Flag**
- typedef memory_order **CPU_atomic_Order**

## Functions

- static void _CPU_atomic_Fence (CPU_atomic_Order order)

  *Sets up a cpu fence.*
- static void _CPU_atomic_Init_uint (CPU_atomic_Uint *obj, unsigned int desired)

  *Initializes Uint.*
- static void _CPU_atomic_Init_ulong (CPU_atomic_Ulong *obj, unsigned long desired)

  *Initializes Ulong.*
- static void _CPU_atomic_Init_uintptr (CPU_atomic_Uintptr *obj, uintptr_t desired)

  *Initializes Uintptr.*
- static unsigned int _CPU_atomic_Load_uint (const CPU_atomic_Uint *obj, CPU_atomic_Order order)

  *Loads value of Uint considering the order.*
- static unsigned long _CPU_atomic_Load_ulong (const CPU_atomic_Ulong *obj, CPU_atomic_Order order)

  *Loads value of Ulong considering the order.*
- static uintptr_t _CPU_atomic_Load_uintptr (const CPU_atomic_Uintptr *obj, CPU_atomic_Order order)

  *Loads value of Uintptr considering the order.*
- static void _CPU_atomic_Store_uint (CPU_atomic_Uint *obj, unsigned int desired, CPU_atomic_Order order)

  *Stores a value to Uint considering the order.*
- static void _CPU_atomic_Store_ulong (CPU_atomic_Ulong *obj, unsigned long desired, CPU_atomic_Order order)

  *Stores a value to Ulong considering the order.*
- static void _CPU_atomic_Store_uintptr (CPU_atomic_Uintptr *obj, uintptr_t desired, CPU_atomic_Order order)

*Stores a value to Uintptr considering the order.*

- static unsigned int _CPU_atomic_Fetch_add_uint (CPU_atomic_Uint *obj, unsigned int arg, CPU_atomic_Order order)

  *Fetches current value of Uint and adds a value to the stored value.*
- static unsigned long _CPU_atomic_Fetch_add_ulong (CPU_atomic_Ulong *obj, unsigned long arg, CPU_atomic_Order order)

  *Fetches current value of Ulong and adds a value to the stored value.*
- static uintptr_t _CPU_atomic_Fetch_add_uintptr (CPU_atomic_Uintptr *obj, uintptr_t arg, CPU_atomic_Order order)

  *Fetches current value of Uintptr and adds a value to the stored value.*
- static unsigned int _CPU_atomic_Fetch_sub_uint (CPU_atomic_Uint *obj, unsigned int arg, CPU_atomic_Order order)

  *Fetches current value of Uint and subtracts a value from the stored value.*
- static unsigned long _CPU_atomic_Fetch_sub_ulong (CPU_atomic_Ulong *obj, unsigned long arg, CPU_atomic_Order order)

  *Fetches current value of Ulong and subtracts a value from the stored value.*
- static uintptr_t _CPU_atomic_Fetch_sub_uintptr (CPU_atomic_Uintptr *obj, uintptr_t arg, CPU_atomic_Order order)

  *Fetches current value of Uintptr and subtracts a value from the stored value.*
- static unsigned int _CPU_atomic_Fetch_or_uint (CPU_atomic_Uint *obj, unsigned int arg, CPU_atomic_Order order)

  *Fetches current value of Uint and ORs a value with the stored value.*
- static unsigned long _CPU_atomic_Fetch_or_ulong (CPU_atomic_Ulong *obj, unsigned long arg, CPU_atomic_Order order)

  *Fetches current value of Ulong and ORs a value with the stored value.*
- static uintptr_t _CPU_atomic_Fetch_or_uintptr (CPU_atomic_Uintptr *obj, uintptr_t arg, CPU_atomic_Order order)

  *Fetches current value of Uintptr and ORs a value with the stored value.*
- static unsigned int _CPU_atomic_Fetch_and_uint (CPU_atomic_Uint *obj, unsigned int arg, CPU_atomic_Order order)

  *Fetches current value of Uint and ANDs a value with the stored value.*
- static unsigned long _CPU_atomic_Fetch_and_ulong (CPU_atomic_Ulong *obj, unsigned long arg, CPU_atomic_Order order)

  *Fetches current value of Ulong and ANDs a value with the stored value.*
- static uintptr_t _CPU_atomic_Fetch_and_uintptr (CPU_atomic_Uintptr *obj, uintptr_t arg, CPU_atomic_Order order)

  *Fetches current value of Uintptr and ANDs a value with the stored value.*
- static unsigned int _CPU_atomic_Exchange_uint (CPU_atomic_Uint *obj, unsigned int desired, CPU_atomic_Order order)

  *Fetches current value of Uint and sets its value.*
- static unsigned long _CPU_atomic_Exchange_ulong (CPU_atomic_Ulong *obj, unsigned long desired, CPU_atomic_Order order)

  *Fetches current value of Ulong and sets its value.*
- static uintptr_t _CPU_atomic_Exchange_uintptr (CPU_atomic_Uintptr *obj, uintptr_t desired, CPU_atomic_Order order)

  *Fetches current value of Uintptr and sets its value.*
- static bool _CPU_atomic_Compare_exchange_uint (CPU_atomic_Uint *obj, unsigned int *expected, unsigned int desired, CPU_atomic_Order succ, CPU_atomic_Order fail)

  *Checks if value of Uint is as expected.*
- static bool _CPU_atomic_Compare_exchange_ulong (CPU_atomic_Ulong *obj, unsigned long *expected, unsigned long desired, CPU_atomic_Order succ, CPU_atomic_Order fail)

  *Checks if value of Ulong is as expected.*

- static bool _CPU_atomic_Compare_exchange_uintptr (CPU_atomic_Uintptr *obj, uintptr_t *expected, uintptr_t desired, CPU_atomic_Order succ, CPU_atomic_Order fail)

  *Checks if value of Uintptr is as expected.*
- static void _CPU_atomic_Flag_clear (CPU_atomic_Flag *obj, CPU_atomic_Order order)

  *Clears the atomic flag.*
- static bool _CPU_atomic_Flag_test_and_set (CPU_atomic_Flag *obj, CPU_atomic_Order order)

  *Returns current flag state and sets it.*

## 8.11.1 Detailed Description

Atomic Operations CPU API.

## 8.11.2 Function Documentation

### 8.11.2.1 _CPU_atomic_Compare_exchange_uint()

```
static bool _CPU_atomic_Compare_exchange_uint (
             CPU_atomic_Uint * obj,
             unsigned int * expected,
             unsigned int desired,
             CPU_atomic_Order succ,
             CPU_atomic_Order fail )  [inline], [static]
```

Checks if value of Uint is as expected.

This method checks if the value of *obj* is equal to the value of *expected*. If this is the case, the value of *obj* is changed to *desired*. Otherwise, the value of *obj* is changed to *expected*.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The CPU atomic Uint to operate upon. |
| in,out | *expected* | The expected value of *obj*. If *obj* has a different value, *expected* is changed to the actual value of *obj*. |
| | *desired* | The new value of *obj* if the old value of *obj* was as expected. |
| | *succ* | The order if it is successful. |
| | *fail* | The order if it fails. |

**Return values**

| | |
|---|---|
| *true* | The old value of *obj* was as expected. |
| *false* | The old value of *obj* was not as expected. |

Definition at line 799 of file cpustdatomic.h.

#### 8.11.2.2 _CPU_atomic_Compare_exchange_uintptr()

```
static bool _CPU_atomic_Compare_exchange_uintptr (
             CPU_atomic_Uintptr * obj,
             uintptr_t * expected,
             uintptr_t desired,
             CPU_atomic_Order succ,
             CPU_atomic_Order fail )  [inline], [static]
```

Checks if value of Uintptr is as expected.

This method checks if the value of *obj* is equal to the value of *expected*. If this is the case, the value of *obj* is changed to *desired*. Otherwise, the value of *obj* is changed to *expected*.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The CPU atomic Uintptr to operate upon. |
| in,out | *expected* | The expected value of *obj*. If *obj* has a different value, *expected* is changed to the actual value of *obj*. |
| | *desired* | The new value of *obj* if the old value of *obj* was as expected. |
| | *succ* | The order if it is successful. |
| | *fail* | The order if it fails. |

**Return values**

| | |
|---|---|
| *true* | The old value of *obj* was as expected. |
| *false* | The old value of *obj* was not as expected. |

Definition at line 887 of file cpustdatomic.h.

#### 8.11.2.3 _CPU_atomic_Compare_exchange_ulong()

```
static bool _CPU_atomic_Compare_exchange_ulong (
             CPU_atomic_Ulong * obj,
             unsigned long * expected,
             unsigned long desired,
             CPU_atomic_Order succ,
             CPU_atomic_Order fail )  [inline], [static]
```

Checks if value of Ulong is as expected.

This method checks if the value of *obj* is equal to the value of *expected*. If this is the case, the value of *obj* is changed to *desired*. Otherwise, the value of *obj* is changed to *expected*.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The CPU atomic Ulong to operate upon. |
| in,out | *expected* | The expected value of *obj*. If *obj* has a different value, *expected* is changed to the actual value of *obj*. |
| | *desired* | The new value of *obj* if the old value of *obj* was as expected. |
| | *succ* | The order if it is successful. |
| | *fail* | The order if it fails. |

**Return values**

| | |
|---|---|
| *true* | The old value of *obj* was as expected. |
| *false* | The old value of *obj* was not as expected. |

Definition at line 843 of file cpustdatomic.h.

#### 8.11.2.4 _CPU_atomic_Exchange_uint()

```
static unsigned int _CPU_atomic_Exchange_uint (
            CPU_atomic_Uint * obj,
            unsigned int desired,
            CPU_atomic_Order order )  [inline], [static]
```

Fetches current value of Uint and sets its value.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The CPU atomic Uint to get the value from and set the value to *desired*. |
| | *arg* | The value to set for *obj*. |
| | *order* | The atomic order for the operation. |

**Returns**

The value of *obj* prior to the exchange with *desired*.

Definition at line 704 of file cpustdatomic.h.

#### 8.11.2.5 _CPU_atomic_Exchange_uintptr()

```
static uintptr_t _CPU_atomic_Exchange_uintptr (
            CPU_atomic_Uintptr * obj,
            uintptr_t desired,
            CPU_atomic_Order order )  [inline], [static]
```

Fetches current value of Uintptr and sets its value.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The CPU atomic Uintptr to get the value from and set the value to *desired*. |
| | *arg* | The value to set for *obj*. |
| | *order* | The atomic order for the operation. |

**Returns**

The value of *obj* prior to the exchange with *desired*.

Definition at line 762 of file cpustdatomic.h.

#### 8.11.2.6 _CPU_atomic_Exchange_ulong()

```
static unsigned long _CPU_atomic_Exchange_ulong (
            CPU_atomic_Ulong * obj,
            unsigned long desired,
            CPU_atomic_Order order )  [inline], [static]
```

Fetches current value of Ulong and sets its value.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The CPU atomic Ulong to get the value from and set the value to *desired*. |
| | *arg* | The value to set for *obj*. |
| | *order* | The atomic order for the operation. |

**Returns**

The value of *obj* prior to the exchange with *desired*.

Definition at line 733 of file cpustdatomic.h.

#### 8.11.2.7 _CPU_atomic_Fence()

```
static void _CPU_atomic_Fence (
            CPU_atomic_Order order )  [inline], [static]
```

Sets up a cpu fence.

**Parameters**

| | | |
|---|---|---|
| out | *order* | The order for the fence. |

Definition at line 149 of file cpustdatomic.h.

#### 8.11.2.8 _CPU_atomic_Fetch_add_uint()

```
static unsigned int _CPU_atomic_Fetch_add_uint (
            CPU_atomic_Uint * obj,
```

```
            unsigned int arg,
            CPU_atomic_Order order )  [inline], [static]
```

Fetches current value of Uint and adds a value to the stored value.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The CPU atomic Uint to get the value from and add *arg* to. |
| | *arg* | The value to add to *obj*. |
| | *order* | The atomic order for the operation. |

**Returns**

The value of *obj* prior to the addition of *arg*.

Definition at line 356 of file cpustdatomic.h.

#### 8.11.2.9 _CPU_atomic_Fetch_add_uintptr()

```
static uintptr_t _CPU_atomic_Fetch_add_uintptr (
            CPU_atomic_Uintptr * obj,
            uintptr_t arg,
            CPU_atomic_Order order )  [inline], [static]
```

Fetches current value of Uintptr and adds a value to the stored value.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The CPU atomic Uintptr to get the value from and add *arg* to. |
| | *arg* | The value to add to *obj*. |
| | *order* | The atomic order for the operation. |

**Returns**

The value of *obj* prior to the addition of *arg*.

Definition at line 414 of file cpustdatomic.h.

#### 8.11.2.10 _CPU_atomic_Fetch_add_ulong()

```
static unsigned long _CPU_atomic_Fetch_add_ulong (
            CPU_atomic_Ulong * obj,
            unsigned long arg,
            CPU_atomic_Order order )  [inline], [static]
```

Fetches current value of Ulong and adds a value to the stored value.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The CPU atomic Ulong to get the value from and add *arg* to. |
| | *arg* | The value to add to *obj*. |
| | *order* | The atomic order for the operation. |

**Returns**

The value of *obj* prior to the addition of *arg*.

Definition at line 385 of file cpustdatomic.h.

#### 8.11.2.11 _CPU_atomic_Fetch_and_uint()

```
static unsigned int _CPU_atomic_Fetch_and_uint (
            CPU_atomic_Uint * obj,
            unsigned int arg,
            CPU_atomic_Order order )  [inline], [static]
```

Fetches current value of Uint and ANDs a value with the stored value.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The CPU atomic Uint to get the value from and AND *arg* to. |
| | *arg* | The value to AND with *obj*. |
| | *order* | The atomic order for the operation. |

**Returns**

The value of *obj* prior to the AND operation with *arg*.

Definition at line 617 of file cpustdatomic.h.

#### 8.11.2.12 _CPU_atomic_Fetch_and_uintptr()

```
static uintptr_t _CPU_atomic_Fetch_and_uintptr (
            CPU_atomic_Uintptr * obj,
            uintptr_t arg,
            CPU_atomic_Order order )  [inline], [static]
```

Fetches current value of Uintptr and ANDs a value with the stored value.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The CPU atomic Uintptr to get the value from and AND *arg* to. |
| | *arg* | The value to AND with *obj*. |
| | *order* | The atomic order for the operation. |

**Returns**

The value of *obj* prior to the AND operation with *arg*.

Definition at line 675 of file cpustdatomic.h.

#### 8.11.2.13 _CPU_atomic_Fetch_and_ulong()

```
static unsigned long _CPU_atomic_Fetch_and_ulong (
            CPU_atomic_Ulong * obj,
            unsigned long arg,
            CPU_atomic_Order order )  [inline], [static]
```

Fetches current value of Ulong and ANDs a value with the stored value.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The CPU atomic Ulong to get the value from and AND *arg* to. |
| | *arg* | The value to AND with *obj*. |
| | *order* | The atomic order for the operation. |

**Returns**

The value of *obj* prior to the AND operation with *arg*.

Definition at line 646 of file cpustdatomic.h.

#### 8.11.2.14 _CPU_atomic_Fetch_or_uint()

```
static unsigned int _CPU_atomic_Fetch_or_uint (
            CPU_atomic_Uint * obj,
            unsigned int arg,
            CPU_atomic_Order order )  [inline], [static]
```

Fetches current value of Uint and ORs a value with the stored value.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The CPU atomic Uint to get the value from and OR *arg* to. |
| | *arg* | The value to OR with *obj*. |
| | *order* | The atomic order for the operation. |

**Returns**

The value of *obj* prior to the OR operation with *arg*.

Definition at line 530 of file cpustdatomic.h.

#### 8.11.2.15 _CPU_atomic_Fetch_or_uintptr()

```
static uintptr_t _CPU_atomic_Fetch_or_uintptr (
            CPU_atomic_Uintptr * obj,
            uintptr_t arg,
            CPU_atomic_Order order )  [inline], [static]
```

Fetches current value of Uintptr and ORs a value with the stored value.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The CPU atomic Uintptr to get the value from and OR *arg* to. |
| | *arg* | The value to OR with *obj*. |
| | *order* | The atomic order for the operation. |

**Returns**

The value of *obj* prior to the OR operation with *arg*.

Definition at line 588 of file cpustdatomic.h.

#### 8.11.2.16 _CPU_atomic_Fetch_or_ulong()

```
static unsigned long _CPU_atomic_Fetch_or_ulong (
            CPU_atomic_Ulong * obj,
            unsigned long arg,
            CPU_atomic_Order order )  [inline], [static]
```

Fetches current value of Ulong and ORs a value with the stored value.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The CPU atomic Ulong to get the value from and OR *arg* to. |
| | *arg* | The value to OR with *obj*. |
| | *order* | The atomic order for the operation. |

**Returns**

The value of *obj* prior to the OR operation with *arg*.

Definition at line 559 of file cpustdatomic.h.

#### 8.11.2.17 _CPU_atomic_Fetch_sub_uint()

```
static unsigned int _CPU_atomic_Fetch_sub_uint (
            CPU_atomic_Uint * obj,
            unsigned int arg,
            CPU_atomic_Order order )  [inline], [static]
```

Fetches current value of Uint and subtracts a value from the stored value.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The CPU atomic Uint to get the value from and subtract *arg* from. |
| | *arg* | The value to subtract from *obj*. |
| | *order* | The atomic order for the operation. |

**Returns**

The value of *obj* prior to the subtraction of *arg*.

Definition at line 443 of file cpustdatomic.h.

#### 8.11.2.18 _CPU_atomic_Fetch_sub_uintptr()

```
static uintptr_t _CPU_atomic_Fetch_sub_uintptr (
            CPU_atomic_Uintptr * obj,
            uintptr_t arg,
            CPU_atomic_Order order )  [inline], [static]
```

Fetches current value of Uintptr and subtracts a value from the stored value.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The CPU atomic Uintptr to get the value from and subtract *arg* from. |
| | *arg* | The value to subtract from *obj*. |
| | *order* | The atomic order for the operation. |

**Returns**

The value of *obj* prior to the subtraction of *arg*.

Definition at line 501 of file cpustdatomic.h.

#### 8.11.2.19 _CPU_atomic_Fetch_sub_ulong()

```
static unsigned long _CPU_atomic_Fetch_sub_ulong (
            CPU_atomic_Ulong * obj,
```

```
            unsigned long arg,
            CPU_atomic_Order order )  [inline], [static]
```

Fetches current value of Ulong and subtracts a value from the stored value.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The CPU atomic Ulong to get the value from and subtract *arg* from. |
| | *arg* | The value to subtract from *obj*. |
| | *order* | The atomic order for the operation. |

**Returns**

The value of *obj* prior to the subtraction of *arg*.

Definition at line 472 of file cpustdatomic.h.

#### 8.11.2.20 _CPU_atomic_Flag_clear()

```
static void _CPU_atomic_Flag_clear (
            CPU_atomic_Flag * obj,
            CPU_atomic_Order order )  [inline], [static]
```

Clears the atomic flag.

**Parameters**

| | | |
|---|---|---|
| out | *obj* | The atomic flag to be cleared. |
| | *order* | The atomic order for the operation. |

Definition at line 920 of file cpustdatomic.h.

#### 8.11.2.21 _CPU_atomic_Flag_test_and_set()

```
static bool _CPU_atomic_Flag_test_and_set (
            CPU_atomic_Flag * obj,
            CPU_atomic_Order order )  [inline], [static]
```

Returns current flag state and sets it.

**Parameters**

| | | |
|---|---|---|
| in,out | *obj* | The atomic flag to be set. |
| | *order* | The atomic order for the operation. |

**Return values**

| | |
|---|---|
| *true* | *obj* was set prior to this operation. |
| *false* | *obj* was not set prior to this operation. |

Definition at line 941 of file cpustdatomic.h.

#### 8.11.2.22 _CPU_atomic_Init_uint()

```
static void _CPU_atomic_Init_uint (
            CPU_atomic_Uint * obj,
            unsigned int desired )  [inline], [static]
```

Initializes Uint.

**Parameters**

| | | |
|---|---|---|
| out | *obj* | The CPU atomic Uint to initialize. |
| | *desired* | The desired value for *obj*. |

Definition at line 167 of file cpustdatomic.h.

#### 8.11.2.23 _CPU_atomic_Init_uintptr()

```
static void _CPU_atomic_Init_uintptr (
            CPU_atomic_Uintptr * obj,
            uintptr_t desired )  [inline], [static]
```

Initializes Uintptr.

**Parameters**

| | | |
|---|---|---|
| out | *obj* | The CPU atomic Uintptr to initialize. |
| | *desired* | The desired value for *obj*. |

Definition at line 201 of file cpustdatomic.h.

#### 8.11.2.24 _CPU_atomic_Init_ulong()

```
static void _CPU_atomic_Init_ulong (
            CPU_atomic_Ulong * obj,
            unsigned long desired )  [inline], [static]
```

Initializes Ulong.

**Parameters**

| | | |
|---|---|---|
| out | *obj* | The CPU atomic Ulong to initialize. |
| | *desired* | The desired value for *obj*. |

Definition at line 184 of file cpustdatomic.h.

#### 8.11.2.25 _CPU_atomic_Load_uint()

```
static unsigned int _CPU_atomic_Load_uint (
            const CPU_atomic_Uint * obj,
            CPU_atomic_Order order )  [inline], [static]
```

Loads value of Uint considering the order.

**Parameters**

| | |
|---|---|
| *obj* | The CPU atomic Uint to get the value from. |
| *order* | The atomic order for getting the value. |

**Returns**

The value of *obj* considering the *order*.

Definition at line 220 of file cpustdatomic.h.

#### 8.11.2.26 _CPU_atomic_Load_uintptr()

```
static uintptr_t _CPU_atomic_Load_uintptr (
            const CPU_atomic_Uintptr * obj,
            CPU_atomic_Order order )  [inline], [static]
```

Loads value of Uintptr considering the order.

**Parameters**

| | |
|---|---|
| *obj* | The CPU atomic Uintptr to get the value from. |
| *order* | The atomic order for getting the value. |

**Returns**

The value of *obj* considering the *order*.

Definition at line 270 of file cpustdatomic.h.

#### 8.11.2.27 _CPU_atomic_Load_ulong()

```
static unsigned long _CPU_atomic_Load_ulong (
            const CPU_atomic_Ulong * obj,
            CPU_atomic_Order order )  [inline], [static]
```

Loads value of Ulong considering the order.

**Parameters**

| | |
|---|---|
| *obj* | The CPU atomic Ulong to get the value from. |
| *order* | The atomic order for getting the value. |

**Returns**

The value of *obj* considering the *order*.

Definition at line 245 of file cpustdatomic.h.

#### 8.11.2.28 _CPU_atomic_Store_uint()

```
static void _CPU_atomic_Store_uint (
            CPU_atomic_Uint * obj,
            unsigned int desired,
            CPU_atomic_Order order )  [inline], [static]
```

Stores a value to Uint considering the order.

**Parameters**

| | | |
|---|---|---|
| out | *obj* | The CPU atomic Uint to store a value in. |
| | *desired* | The desired value for *obj*. |
| | *order* | The atomic order for storing the value. |

Definition at line 294 of file cpustdatomic.h.

#### 8.11.2.29 _CPU_atomic_Store_uintptr()

```
static void _CPU_atomic_Store_uintptr (
            CPU_atomic_Uintptr * obj,
            uintptr_t desired,
            CPU_atomic_Order order )  [inline], [static]
```

Stores a value to Uintptr considering the order.

**Parameters**

| | | |
|---|---|---|
| out | *obj* | The CPU atomic Uintptr to store a value in. |
| | *desired* | The desired value for *obj*. |
| | *order* | The atomic order for storing the value. |

Definition at line 334 of file cpustdatomic.h.

#### 8.11.2.30 _CPU_atomic_Store_ulong()

```
static void _CPU_atomic_Store_ulong (
            CPU_atomic_Ulong * obj,
            unsigned long desired,
            CPU_atomic_Order order )  [inline], [static]
```

Stores a value to Ulong considering the order.

**Parameters**

| | | |
|---|---|---|
| out | *obj* | The CPU atomic Ulong to store a value in. |
| | *desired* | The desired value for *obj*. |
| | *order* | The atomic order for storing the value. |

Definition at line 314 of file cpustdatomic.h.

## 8.12 BSP Interrupt Support

Generic BSP Interrupt Support.

### Files

- file irq-generic.h

  *Generic BSP interrupt support API.*
- file irq-generic.c

  *Generic BSP interrupt support implementation.*
- file irq-lock.c

  *BSP interrupt support lock implementation.*

### Functions

- static bool bsp_interrupt_is_valid_vector (rtems_vector_number vector)

  *Returns true if the interrupt vector with number vector is valid.*
- void bsp_interrupt_handler_default (rtems_vector_number vector)

  *Default interrupt handler.*
- void bsp_interrupt_initialize (void)

  *Initialize BSP interrupt support.*
- rtems_status_code bsp_interrupt_facility_initialize (void)

  *BSP specific initialization.*
- void bsp_interrupt_vector_enable (rtems_vector_number vector)

  *Enables the interrupt vector with number vector.*
- void bsp_interrupt_vector_disable (rtems_vector_number vector)

  *Disables the interrupt vector with number vector.*
- static void bsp_interrupt_handler_dispatch (rtems_vector_number vector)

  *Sequencially calls all interrupt handlers for the vector number vector.*
- bool bsp_interrupt_handler_is_empty (rtems_vector_number vector)

  *Is interrupt handler empty.*
- static rtems_status_code bsp_interrupt_handler_install (rtems_vector_number vector, const char *info, rtems_option options, rtems_interrupt_handler handler, void *arg)

  *Installs an interrupt handler.*
- static rtems_status_code bsp_interrupt_handler_remove (rtems_vector_number vector, rtems_interrupt_handler handler, void *arg)

  *Removes an interrupt handler.*
- static rtems_status_code bsp_interrupt_handler_iterate (rtems_vector_number vector, rtems_interrupt_per_handler_routine routine, void *arg)

  *Iterates over all installed interrupt handler of a vector.*

### 8.12.1 Detailed Description

Generic BSP Interrupt Support.

The BSP interrupt support manages a sequence of interrupt vector numbers ranging from BSP_INTERRUPT_V↩ ECTOR_MIN to BSP_INTERRUPT_VECTOR_MAX including the end points. It provides methods to install, remove and dispatch interrupt handlers for each vector number. It implements parts of the RTEMS interrupt manager.

The entry points to a list of interrupt handlers are stored in a table (= handler table).

You have to configure the BSP interrupt support in the <bsp/irq.h> file for each BSP. For a minimum configuration you have to provide BSP_INTERRUPT_VECTOR_MIN and BSP_INTERRUPT_VECTOR_MAX.

For boards with small memory requirements you can define BSP_INTERRUPT_USE_INDEX_TABLE. With an enabled index table the handler table will be accessed via a small index table. You can define the size of the handler table with BSP_INTERRUPT_HANDLER_TABLE_SIZE.

Normally new list entries are allocated from the heap. You may define BSP_INTERRUPT_NO_HEAP_USAGE, if you do not want to use the heap. For this option you have to define BSP_INTERRUPT_USE_INDEX_TABLE as well.

You have to provide some special routines in your BSP (follow the links for the details):

- bsp_interrupt_facility_initialize()
- bsp_interrupt_vector_enable()
- bsp_interrupt_vector_disable()
- bsp_interrupt_handler_default()

The following now deprecated functions are provided for backward compatibility:

- BSP_get_current_rtems_irq_handler()
- BSP_install_rtems_irq_handler()
- BSP_install_rtems_shared_irq_handler()
- BSP_remove_rtems_irq_handler()
- BSP_rtems_irq_mngt_set()
- BSP_rtems_irq_mngt_get()

### 8.12.2 Function Documentation

#### 8.12.2.1 bsp_interrupt_facility_initialize()

```
rtems_status_code bsp_interrupt_facility_initialize (
            void  )
```

BSP specific initialization.

This routine will be called form bsp_interrupt_initialize() and shall do the following:

- Initialize the facilities that call bsp_interrupt_handler_dispatch(). For example on PowerPC the external exception handler.
- Initialize the interrupt controller. You shall set the interrupt controller in a state such that interrupts are disabled for all vectors. The vectors will be enabled with your bsp_interrupt_vector_enable() function and disabled via your bsp_interrupt_vector_disable() function. These functions have to work afterwards.

**Returns**

On success RTEMS_SUCCESSFUL shall be returned.

Definition at line 62 of file irq-shared.c.

#### 8.12.2.2 bsp_interrupt_handler_default()

```
void bsp_interrupt_handler_default (
            rtems_vector_number vector )
```

Default interrupt handler.

This routine will be called from bsp_interrupt_handler_dispatch() with the current vector number *vector* when the handler list for this vector is empty or the vector number is out of range.

**Note**

This function must cope with arbitrary vector numbers *vector*.

Definition at line 21 of file irq-default-handler.c.

#### 8.12.2.3 bsp_interrupt_handler_dispatch()

```
static void bsp_interrupt_handler_dispatch (
            rtems_vector_number vector )  [inline], [static]
```

Sequencially calls all interrupt handlers for the vector number *vector*.

If the vector number is out of range or the handler list is empty bsp_interrupt_handler_default() will be called with argument *vector*.

You can call this function within every context which can be disabled via rtems_interrupt_disable().

Definition at line 259 of file irq-generic.h.

#### 8.12.2.4 bsp_interrupt_handler_install()

```
static rtems_status_code bsp_interrupt_handler_install (
            rtems_vector_number vector,
            const char * info,
            rtems_option options,
            rtems_interrupt_handler handler,
            void * arg )  [static]
```

Installs an interrupt handler.

**Returns**

In addition to the standard status codes this function returns:

- If the BSP interrupt support is not initialized RTEMS_INTERNAL_ERROR will be returned.
- If not enough memory for a new handler is available RTEMS_NO_MEMORY will be returned

**See also**

rtems_interrupt_handler_install()

Definition at line 193 of file irq-generic.c.

#### 8.12.2.5 bsp_interrupt_handler_is_empty()

```
bool bsp_interrupt_handler_is_empty (
            rtems_vector_number vector )
```

Is interrupt handler empty.

This routine returns true if the handler is empty and has not been initialised else false is returned. The interrupt lock is not used so this call can be used from within interrupts.

**Returns**

If empty true shall be returned else false is returned.

Definition at line 549 of file irq-generic.c.

#### 8.12.2.6 bsp_interrupt_handler_iterate()

```
static rtems_status_code bsp_interrupt_handler_iterate (
            rtems_vector_number vector,
            rtems_interrupt_per_handler_routine routine,
            void * arg )  [static]
```

Iterates over all installed interrupt handler of a vector.

**Returns**

In addition to the standard status codes this function returns RTEMS_INTERNAL_ERROR if the BSP interrupt support is not initialized.

**See also**

rtems_interrupt_handler_iterate().

Definition at line 480 of file irq-generic.c.

#### 8.12.2.7 bsp_interrupt_handler_remove()

```
static rtems_status_code bsp_interrupt_handler_remove (
            rtems_vector_number vector,
            rtems_interrupt_handler handler,
            void * arg )  [static]
```

Removes an interrupt handler.

**Returns**

In addition to the standard status codes this function returns RTEMS_INTERNAL_ERROR if the BSP interrupt support is not initialized.

**See also**

rtems_interrupt_handler_remove().

Definition at line 358 of file irq-generic.c.

#### 8.12.2.8 bsp_interrupt_initialize()

```
void bsp_interrupt_initialize (
            void  )
```

Initialize BSP interrupt support.

You must call this function before you can install, remove and dispatch interrupt handlers. There is no protection against concurrent initialization. This function must be called at most once. The BSP specific bsp_interrupt_facility_initialize() function will be called after all internals are initialized. If the BSP specific initialization fails, then this is a fatal error. The fatal error source is RTEMS_FATAL_SOURCE_BSP and the fatal error code is BSP_FATAL_INTERRUPT_INITIALIZATION.

Definition at line 161 of file irq-generic.c.

#### 8.12.2.9 bsp_interrupt_vector_disable()

```
void bsp_interrupt_vector_disable (
            rtems_vector_number vector )
```

Disables the interrupt vector with number *vector*.

This function shall disable the vector at the corresponding facility (in most cases the interrupt controller). It will be called then the last handler is removed for the vector in bsp_interrupt_handler_remove() for example.

**Note**

The implementation should use bsp_interrupt_assert(bsp_interrupt_is_valid_vector(vector)) to valdiate the vector number.

You must not install or remove an interrupt handler in this function. This may result in a deadlock.

Definition at line 74 of file irq-shared.c.

#### 8.12.2.10 bsp_interrupt_vector_enable()

```
void bsp_interrupt_vector_enable (
            rtems_vector_number vector )
```

Enables the interrupt vector with number *vector*.

This function shall enable the vector at the corresponding facility (in most cases the interrupt controller). It will be called then the first handler is installed for the vector in bsp_interrupt_handler_install() for example.

**Note**

The implementation should use bsp_interrupt_assert(bsp_interrupt_is_valid_vector(vector)) to valdiate the vector number.

You must not install or remove an interrupt handler in this function. This may result in a deadlock.

Definition at line 67 of file irq-shared.c.

# 8.13 BSP Related Configuration Options

## Macros

- #define BSP_IDLE_TASK_BODY

  *This configuration option is an initializer define.*
- #define BSP_IDLE_TASK_STACK_SIZE

  *This configuration option is an integer define.*
- #define BSP_INITIAL_EXTENSION

  *This configuration option is an initializer define.*
- #define BSP_INTERRUPT_STACK_SIZE

  *This configuration option is an integer define.*
- #define CONFIGURE_BSP_PREREQUISITE_DRIVERS

  *This configuration option is an initializer define.*
- #define CONFIGURE_DISABLE_BSP_SETTINGS

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_MALLOC_BSP_SUPPORTS_SBRK

  *This configuration option is a boolean feature define.*

## 8.13.1 Detailed Description

This section describes configuration options related to the BSP. Some configuration options may have a BSP-specific setting which is defined by `<bsp.h>`. The BSP-specific settings can be disabled by the CONFIGURE_DISABLE_BSP_SETTINGS configuration option.

## 8.13.2 Macro Definition Documentation

### 8.13.2.1 BSP_IDLE_TASK_BODY

```
#define BSP_IDLE_TASK_BODY
```

This configuration option is an initializer define.

If

- this configuration option is defined by the BSP
- and CONFIGURE_DISABLE_BSP_SETTINGS is undefined,

then the value of this configuration option defines the default value of CONFIGURE_IDLE_TASK_BODY.

**Default Value**

The default value is BSP-specific.

**Value Constraints**

The value of this configuration option shall be defined to a valid function pointer of the type `void *( *idle_body )( uintptr_t )`.

**Notes**

As it has knowledge of the specific CPU model, system controller logic, and peripheral buses, a BSP-specific IDLE task may be capable of turning components off to save power during extended periods of no task activity.

Definition at line 359 of file appl-config.h.

#### 8.13.2.2 BSP_IDLE_TASK_STACK_SIZE

```
#define BSP_IDLE_TASK_STACK_SIZE
```

This configuration option is an integer define.

If

- this configuration option is defined by the BSP
- and CONFIGURE_DISABLE_BSP_SETTINGS is undefined,

then the value of this configuration option defines the default value of CONFIGURE_IDLE_TASK_STACK_SIZE.

**Default Value**

The default value is BSP-specific.

**Value Constraints**

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to a BSP-specific and application-specific minimum value.
- It shall be small enough so that the IDLE task stack area calculation carried out by <rtems/confdefs.h> does not overflow an integer of type size_t.

Definition at line 391 of file appl-config.h.

#### 8.13.2.3 BSP_INITIAL_EXTENSION

```
#define BSP_INITIAL_EXTENSION
```

This configuration option is an initializer define.

If

- this configuration option is defined by the BSP
- and CONFIGURE_DISABLE_BSP_SETTINGS is undefined,

then the value of this configuration option is used to initialize the table of initial user extensions.

**Default Value**

The default value is BSP-specific.

**Value Constraints**

The value of this configuration option shall be a list of initializers for structures of type rtems_extensions_table.

**Notes**

The value of this configuration option is placed after the entries of all other initial user extensions.

Definition at line 418 of file appl-config.h.

#### 8.13.2.4 BSP_INTERRUPT_STACK_SIZE

```
#define BSP_INTERRUPT_STACK_SIZE
```

This configuration option is an integer define.

If

- this configuration option is defined by the BSP
- and CONFIGURE_DISABLE_BSP_SETTINGS is undefined,

then the value of this configuration option defines the default value of CONFIGURE_INTERRUPT_STACK_SIZE.

**Default Value**

The default value is BSP-specific.

**Value Constraints**

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to a BSP-specific and application-specific minimum value.
- It shall be small enough so that the interrupt stack area calculation carried out by <rtems/confdefs.h> does not overflow an integer of type size_t.
- It shall be aligned according to #CPU_INTERRUPT_STACK_ALIGNMENT.

Definition at line 452 of file appl-config.h.

#### 8.13.2.5 CONFIGURE_BSP_PREREQUISITE_DRIVERS

```
#define CONFIGURE_BSP_PREREQUISITE_DRIVERS
```

This configuration option is an initializer define.

If

- this configuration option is defined by the BSP
- and CONFIGURE_DISABLE_BSP_SETTINGS is undefined,

then the value of this configuration option is used to initialize the table of initial user extensions.

**Default Value**

The default value is BSP-specific.

**Value Constraints**

The value of this configuration option shall be a list of initializers for structures of type rtems_extensions_table.

**Notes**

The value of this configuration option is placed before the entries of all other initial user extensions (including CONFIGURE_APPLICATION_PREREQUISITE_DRIVERS).

Definition at line 480 of file appl-config.h.

#### 8.13.2.6 CONFIGURE_DISABLE_BSP_SETTINGS

```
#define CONFIGURE_DISABLE_BSP_SETTINGS
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the following BSP related configuration options are undefined:

- BSP_IDLE_TASK_BODY
- BSP_IDLE_TASK_STACK_SIZE
- BSP_INITIAL_EXTENSION
- BSP_INTERRUPT_STACK_SIZE
- CONFIGURE_BSP_PREREQUISITE_DRIVERS
- CONFIGURE_MALLOC_BSP_SUPPORTS_SBRK

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

Definition at line 506 of file appl-config.h.

#### 8.13.2.7 CONFIGURE_MALLOC_BSP_SUPPORTS_SBRK

```
#define CONFIGURE_MALLOC_BSP_SUPPORTS_SBRK
```

This configuration option is a boolean feature define.

If

- this configuration option is defined by the BSP
- and CONFIGURE_DISABLE_BSP_SETTINGS is undefined,

then not all memory is made available to the C Program Heap immediately at system initialization time. When malloc() or other standard memory allocation functions are unable to allocate memory, they will call the BSP supplied sbrk() function to obtain more memory.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

This option should not be defined by the application. Only the BSP knows how it allocates memory to the C Program Heap.

Definition at line 532 of file appl-config.h.

## 8.14 Barrier Handler

Functionality for Foundation Barrier Services.

### Files

- file corebarrier.h

  *Constants and Structures Associated with the Barrier Handler.*
- file corebarrierimpl.h

  *Inlined Routines Associated with the SuperCore Barrier.*
- file corebarrier.c

  *Initialize CORE Barrier.*
- file corebarrierrelease.c

  *Manually releases the Barrier.*
- file corebarrierwait.c

  *Wait For The Barrier.*

### Classes

- struct CORE_barrier_Attributes
- struct CORE_barrier_Control

### Macros

- #define **CORE_BARRIER_TQ_OPERATIONS** &_Thread_queue_Operations_FIFO

### Enumerations

- enum CORE_barrier_Disciplines { CORE_BARRIER_AUTOMATIC_RELEASE, CORE_BARRIER_MANUAL_RELEASE }

### Functions

- void _CORE_barrier_Initialize (CORE_barrier_Control *the_barrier, CORE_barrier_Attributes *the_barrier↩ _attributes)

  *Initializes the core barrier.*
- static __inline__ void _CORE_barrier_Destroy (CORE_barrier_Control *the_barrier)

  *Destroys the core barrier.*
- static __inline__ void _CORE_barrier_Acquire_critical (CORE_barrier_Control *the_barrier, Thread_queue_Context *queue_context)

  *Acquires critical core barrier.*
- static __inline__ void _CORE_barrier_Release (CORE_barrier_Control *the_barrier, Thread_queue_Context *queue_context)

  *Releases core barrier.*
- Status_Control _CORE_barrier_Seize (CORE_barrier_Control *the_barrier, Thread_Control *executing, bool wait, Thread_queue_Context *queue_context)

  *Waits for the barrier.*

- uint32_t _CORE_barrier_Do_flush (CORE_barrier_Control *the_barrier, Thread_queue_Flush_filter filter, Thread_queue_Context *queue_context)

  *Flushes the barrier.*
- static __inline__ uint32_t _CORE_barrier_Surrender (CORE_barrier_Control *the_barrier, Thread_queue_Context *queue_context)

  *Manually releases the barrier.*
- static __inline__ void _CORE_barrier_Flush (CORE_barrier_Control *the_barrier, Thread_queue_Context *queue_context)

  *Flushes the barrier using _CORE_barrier_Do_flush().*
- static __inline__ bool _CORE_barrier_Is_automatic (CORE_barrier_Attributes *the_attribute)

  *Checks if the barrier is automatic.*
- static __inline__ uint32_t _CORE_barrier_Get_number_of_waiting_threads (CORE_barrier_Control *the_barrier)

  *Returns the number of currently waiting threads.*

## 8.14.1 Detailed Description

Functionality for Foundation Barrier Services.

This handler encapsulates functionality which provides the foundation Barrier services used in all of the APIs supported by RTEMS.

## 8.14.2 Enumeration Type Documentation

### 8.14.2.1 CORE_barrier_Disciplines

```
enum CORE_barrier_Disciplines
```

Flavors of barriers.

**Enumerator**

| | |
|---|---|
| CORE_BARRIER_AUTOMATIC_RELEASE | This specifies that the barrier will automatically release when the user specified number of threads have arrived at the barrier. |
| CORE_BARRIER_MANUAL_RELEASE | This specifies that the user will have to manually release the barrier in order to release the waiting threads. |

Definition at line 46 of file corebarrier.h.

## 8.14.3 Function Documentation

#### 8.14.3.1 _CORE_barrier_Acquire_critical()

```
static __inline__ void _CORE_barrier_Acquire_critical (
             CORE_barrier_Control * the_barrier,
             Thread_queue_Context * queue_context )  [static]
```

Acquires critical core barrier.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_barrier* | The barrier to acquire. |
| | *queue_context* | The thread queue context. |

Definition at line 73 of file corebarrierimpl.h.

#### 8.14.3.2 _CORE_barrier_Destroy()

```
static __inline__ void _CORE_barrier_Destroy (
             CORE_barrier_Control * the_barrier )  [static]
```

Destroys the core barrier.

This routine destroys the barrier.

**Parameters**

| | | |
|---|---|---|
| out | *the_barrier* | The barrier to destroy. |

Definition at line 60 of file corebarrierimpl.h.

#### 8.14.3.3 _CORE_barrier_Do_flush()

```
uint32_t _CORE_barrier_Do_flush (
             CORE_barrier_Control * the_barrier,
             Thread_queue_Flush_filter filter,
             Thread_queue_Context * queue_context )
```

Flushes the barrier.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_barrier* | The barrier to flush. |
| out | *filter* | The filter for flushing. |
| out | *queue_context* | The thread queue context. |

Definition at line 24 of file corebarrierrelease.c.

#### 8.14.3.4 _CORE_barrier_Flush()

```
static __inline__ void _CORE_barrier_Flush (
            CORE_barrier_Control * the_barrier,
            Thread_queue_Context * queue_context )  [static]
```

Flushes the barrier using _CORE_barrier_Do_flush().

**Parameters**

| | | |
|---|---|---|
| in,out | *the_barrier* | The barrier to flush. |
| | *queue_context* | The thread queue context. |

Definition at line 159 of file corebarrierimpl.h.

#### 8.14.3.5 _CORE_barrier_Get_number_of_waiting_threads()

```
static __inline__ uint32_t _CORE_barrier_Get_number_of_waiting_threads (
            CORE_barrier_Control * the_barrier )  [static]
```

Returns the number of currently waiting threads.

This routine returns the number of threads currently waiting at the barrier.

**Parameters**

| | | |
|---|---|---|
| in | *the_barrier* | The barrier to obtain the number of blocked threads of. |

**Returns**

the current count of waiting threads of this barrier.

Definition at line 200 of file corebarrierimpl.h.

#### 8.14.3.6 _CORE_barrier_Initialize()

```
void _CORE_barrier_Initialize (
            CORE_barrier_Control * the_barrier,
            CORE_barrier_Attributes * the_barrier_attributes )
```

Initializes the core barrier.

This routine initializes the barrier based on the parameters passed.

**Parameters**

| | | |
|---|---|---|
| out | *the_barrier* | The barrier to initialize. |
| out | *the_barrier_attributes* | The attributes which define the behavior of this instance. |

Definition at line 23 of file corebarrier.c.

#### 8.14.3.7 _CORE_barrier_Is_automatic()

```
static __inline__ bool _CORE_barrier_Is_automatic (
            CORE_barrier_Attributes * the_attribute )  [static]
```

Checks if the barrier is automatic.

This function returns true if the automatic release attribute is enabled in the *attribute_set* and false otherwise.

**Parameters**

| | |
|---|---|
| *the_attribute* | The attribute set to test. |

**Return values**

| | |
|---|---|
| *true* | The automatic release attribute is enabled. |
| *false* | The automatic release attribute is not enabled. |

Definition at line 182 of file corebarrierimpl.h.

#### 8.14.3.8 _CORE_barrier_Release()

```
static __inline__ void _CORE_barrier_Release (
            CORE_barrier_Control * the_barrier,
            Thread_queue_Context * queue_context )  [static]
```

Releases core barrier.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_barrier* | The barrier to release. |
| | *queue_context* | The thread queue context. |

Definition at line 87 of file corebarrierimpl.h.

#### 8.14.3.9 _CORE_barrier_Seize()

```
Status_Control _CORE_barrier_Seize (
            CORE_barrier_Control * the_barrier,
            Thread_Control * executing,
            bool wait,
            Thread_queue_Context * queue_context )
```

Waits for the barrier.

This routine waits for the barrier to be released. If the barrier is set to automatic and this is the appropriate thread, then it returns immediately. Otherwise, the calling thread is blocked until the barrier is released.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_barrier* | The barrier to wait for. |
| in,out | *executing* | The currently executing thread. |
| | *wait* | This parameter is true if the calling thread is willing to wait. |
| | *queue_context* | The thread queue context. |

**Returns**

The method status.

Definition at line 25 of file corebarrierwait.c.

#### 8.14.3.10 _CORE_barrier_Surrender()

```
static __inline__ uint32_t _CORE_barrier_Surrender (
            CORE_barrier_Control * the_barrier,
            Thread_queue_Context * queue_context )  [static]
```

Manually releases the barrier.

This routine manually releases the barrier. All of the threads waiting for the barrier will be readied.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_barrier* | The barrier to surrender. |
| out | *queue_context* | The thread queue context. |

**Returns**

The number of unblocked threads.

Definition at line 141 of file corebarrierimpl.h.

## 8.15 Barrier Manager

The Barrier Manager provides a unique synchronization capability which can be used to have a set of tasks block and be unblocked as a set.

### Functions

- rtems_status_code rtems_barrier_create (rtems_name name, rtems_attribute attribute_set, uint32_t maximum_waiters, rtems_id *id)

  %
- rtems_status_code rtems_barrier_delete (rtems_id id)

  %
- rtems_status_code rtems_barrier_ident (rtems_name name, rtems_id *id)

  *Identifies a barrier object by the specified object name.*
- rtems_status_code rtems_barrier_release (rtems_id id, uint32_t *released)

  %
- rtems_status_code rtems_barrier_wait (rtems_id id, rtems_interval timeout)

  %

### 8.15.1 Detailed Description

The Barrier Manager provides a unique synchronization capability which can be used to have a set of tasks block and be unblocked as a set.

### 8.15.2 Function Documentation

#### 8.15.2.1 rtems_barrier_create()

```
rtems_status_code rtems_barrier_create (
            rtems_name name,
            rtems_attribute attribute_set,
            uint32_t maximum_waiters,
            rtems_id * id )
```

%

**Parameters**

| | |
|---|---|
| *name* | % |
| *attribute_set* | % |
| *maximum_waiters* | % |
| *id* | % |

Definition at line 29 of file barriercreate.c.

#### 8.15.2.2 rtems_barrier_delete()

```
rtems_status_code rtems_barrier_delete (
            rtems_id id )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |

Definition at line 23 of file barrierdelete.c.

#### 8.15.2.3 rtems_barrier_ident()

```
rtems_status_code rtems_barrier_ident (
            rtems_name name,
            rtems_id * id )
```

Identifies a barrier object by the specified object name.

This directive obtains the barrier identifier associated with the barrier name specified in `name`.

If the barrier name is not unique, then the barrier identifier will match the first barrier with that name in the search order. However, this barrier identifier is not guaranteed to correspond to the desired barrier. The barrier identifier is used with other barrier related directives to access the barrier.

The objects are searched from lowest to the highest index. Only the local node is searched.

**Parameters**

| | | |
|---|---|---|
| | *name* | is the object name to look up. |
| `out` | *id* | is the pointer to an object identifier variable. The object identifier of an object with the specified name will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The id parameter was NULL. |
| *RTEMS_INVALID_NAME* | The name parameter was 0. |
| *RTEMS_INVALID_NAME* | There was no object with the specified name on the local node. |

Definition at line 44 of file barrierident.c.

#### 8.15.2.4 rtems_barrier_release()

```
rtems_status_code rtems_barrier_release (
```

```
            rtems_id id,
            uint32_t * released )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *released* | % |

Definition at line 24 of file barrierrelease.c.

#### 8.15.2.5 rtems_barrier_wait()

```
rtems_status_code rtems_barrier_wait (
            rtems_id id,
            rtems_interval timeout )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *timeout* | % |

Definition at line 30 of file barrierwait.c.

## 8.16 Base Definitions

This group contains basic macros and defines to give access to compiler-specific features.

### Macros

- #define RTEMS_ALIAS(_target) __attribute__((__alias__(#_target)))

  *Instructs the compiler to generate an alias to the target function.*
- #define RTEMS_ALIGN_DOWN(_value, _alignment) ( ( _value ) & ∼( ( _alignment ) - 1 ) )

  *Aligns down the value to the alignment.*
- #define RTEMS_ALIGN_UP(_value, _alignment) ( ( ( _value ) + ( _alignment ) - 1 ) & ∼( ( _alignment ) - 1 ) )

  *Aligns up the value to the alignment.*
- #define RTEMS_ALIGNED(_alignment) __attribute__((__aligned__(_alignment)))

  *Instructs the compiler in a declaration or definition to enforce the alignment.*
- #define RTEMS_ALLOC_ALIGN(_index) __attribute__((__alloc_align__(_index)))

  *Tells the compiler in a declaration that the memory allocation alignment parameter of this function is similar to aligned_alloc().*
- #define RTEMS_ALLOC_SIZE(_index) __attribute__((__alloc_size__(_index)))

  *Tells the compiler in a declaration that the memory allocation size parameter of this function is similar to malloc().*
- #define RTEMS_ALLOC_SIZE_2(_count_index, _size_index) __attribute__((__alloc_size__(_count_index, _size_index)))

  *Tells the compiler in a declaration that the memory allocation item count and item size parameter of this function is similar to calloc().*
- #define RTEMS_ARRAY_SIZE(_array) ( sizeof( _array ) / sizeof( ( _array )[ 0 ] ) )

  *Gets the element count of the array.*
- #define RTEMS_COMPILER_MEMORY_BARRIER() __asm__ volatile( "" ::: "memory" )

  *This macro forbids the compiler to reorder read and write commands around it.*
- #define RTEMS_CONCAT(_x, _y) _x##_y

  *Concatenates _x and _y without expanding.*
- #define RTEMS_CONST __attribute__((__const__))

  *Tells the compiler in a function declaration that this function has no effect except the return value and that the return value depends only on the value of parameters.*
- #define RTEMS_CONTAINER_OF(_m, _type, _member_name) ( (_type ∗) ( (uintptr_t) ( _m ) - offsetof( _type, _member_name ) ) )

  *Gets the container of a member.*
- #define RTEMS_DECLARE_GLOBAL_SYMBOL(_name) extern char _name[ ]

  *Declares a global symbol with the name.*
- #define RTEMS_DEPRECATED __attribute__((__deprecated__))

  *Instructs the compiler in a declaration to issue a warning whenever a variable, function, or type using this declaration will be used.*
- #define RTEMS_COMPILER_DEPRECATED_ATTRIBUTE RTEMS_DEPRECATED

  *Provided for backward compatibility.*
- #define RTEMS_EXPAND(_token) _token

  *Helper macro to perform a macro expansion on the token.*
- #define RTEMS_STRING(_x) #_x

  *Stringifies _x without expanding.*
- #define RTEMS_TYPEOF_REFX(_level, _target) __typeof__( _level( union { int z; __typeof__( _target ) x; } ){ 0 }.x )

  *Gets the pointer reference type.*
- #define RTEMS_XCONCAT(_x, _y) RTEMS_CONCAT( _x, _y )

  *Concatenates expansion of _x and expansion of _y.*

- #define RTEMS_DEQUALIFY_DEPTHX(_ptr_level, _type, _var)

  *Performs a type cast which removes qualifiers without warnings to the type for the variable.*
- #define RTEMS_DECONST(_type, _var) RTEMS_DEQUALIFY_DEPTHX(*, _type, _var)

  *Performs a type cast which removes const qualifiers without warnings to the type for the pointer variable.*
- #define RTEMS_DEQUALIFY(_type, _var) RTEMS_DEQUALIFY_DEPTHX(*, _type, _var)

  *Performs a type cast which removes all qualifiers without warnings to the type for the pointer variable.*
- #define RTEMS_DEVOLATILE(_type, _var) RTEMS_DEQUALIFY_DEPTHX(*, _type, _var)

  *Performs a type cast which removes volatile qualifiers without warnings to the type for the pointer variable.*
- #define FALSE 0

  *If FALSE is undefined, then FALSE is defined to 0.*
- #define RTEMS_HAVE_MEMBER_SAME_TYPE(_t_lhs, _m_lhs, _t_rhs, _m_rhs)

  *Checks if members of two types have compatible types.*
- #define RTEMS_INLINE_ROUTINE static __inline__

  *Gives a hint to the compiler in a function declaration to inline this function.*
- #define RTEMS_MALLOCLIKE __attribute__((__malloc__))

  *Tells the compiler in a declaration that this function is a memory allocation function similar to malloc().*
- #define RTEMS_NO_INLINE __attribute__((__noinline__))

  *Instructs the compiler in a function declaration to not inline this function.*
- #define RTEMS_NO_RETURN _Noreturn

  *Tells the compiler in a function declaration that this function does not return.*
- #define RTEMS_COMPILER_NO_RETURN_ATTRIBUTE RTEMS_NO_RETURN

  *Provided for backward compatibility.*
- #define RTEMS_OBFUSCATE_VARIABLE(_var) __asm__( "" : "+r" ( _var ) )

  *Obfuscates the variable so that the compiler cannot perform optimizations based on the variable value.*
- #define RTEMS_PACKED __attribute__((__packed__))

  *Instructs the compiler in a type definition to place members of a structure or union so that the memory required is minimized.*
- #define RTEMS_COMPILER_PACKED_ATTRIBUTE RTEMS_PACKED

  *Provided for backward compatibility.*
- #define RTEMS_PREDICT_FALSE(_exp) __builtin_expect( ( _exp ), 0 )

  *Evaluates the integral expression and tells the compiler that the predicted value is false (0).*
- #define RTEMS_PREDICT_TRUE(_exp) __builtin_expect( ( _exp ), 0 )

  *Evaluates the integral expression and tells the compiler that the predicted value is true (1).*
- #define RTEMS_PRINTFLIKE(_format_pos, _ap_pos) __attribute__((__format__(__printf__, _format_pos, _ap_pos)))

  *Tells the compiler in a declaration that this function expects printf()-like arguments.*
- #define RTEMS_PURE __attribute__((__pure__))

  *Tells the compiler in a function declaration that this function has no effect except the return value and that the return value depends only on the value of parameters and/or global variables.*
- #define RTEMS_COMPILER_PURE_ATTRIBUTE RTEMS_PURE

  *Provided for backward compatibility.*
- #define RTEMS_RETURN_ADDRESS() __builtin_return_address(0)

  *Gets the return address of the current function.*
- #define RTEMS_SECTION(_section) __attribute__((__section__(_section)))

  *Instructs the compiler to place the variable or function in the section.*
- #define RTEMS_STATIC_ASSERT(_cond, _msg) _Static_assert(_cond, # _msg)

  *Asserts at compile time that the condition is satisfied.*
- #define RTEMS_SYMBOL_NAME(_name) RTEMS_EXPAND(_name)

  *Maps the name to the associated symbol name.*
- #define TRUE 1

  *If TRUE is undefined, then TRUE is defined to 1.*

- #define RTEMS_UNUSED __attribute__((__unused__))

  *Tells the compiler that the variable or function is deliberately unused.*
- #define RTEMS_USED __attribute__((__used__))

  *Tells the compiler that the variable or function is used.*
- #define RTEMS_COMPILER_USED_ATTRIBUTE RTEMS_USED

  *Provided for backward compatibility.*
- #define RTEMS_WARN_UNUSED_RESULT __attribute__((__warn_unused_result__))

  *Tells the compiler in a declaration that the result of this function should be used.*
- #define RTEMS_WEAK __attribute__((__weak__))

  *Tells the compiler in a function definition that this function should be weak.*
- #define RTEMS_WEAK_ALIAS(_target) __attribute__((__weak__, __alias__(#_target)))

  *Instructs the compiler to generate a weak alias to the target function.*
- #define RTEMS_XSTRING(_x) RTEMS_STRING( _x )

  *Stringifies the expansion of _x.*
- #define RTEMS_DEFINE_GLOBAL_SYMBOL(_name, _value)

  *Defines a global symbol with the name and value.*
- #define RTEMS_ZERO_LENGTH_ARRAY

  *This constant represents the element count of a zero-length array.*

## 8.16.1 Detailed Description

This group contains basic macros and defines to give access to compiler-specific features.

## 8.16.2 Macro Definition Documentation

### 8.16.2.1 RTEMS_ALIAS

```
#define RTEMS_ALIAS(
              _target ) __attribute__((__alias__(#_target)))
```

Instructs the compiler to generate an alias to the target function.

**Parameters**

| | |
|---|---|
| *_target* | is the target function name. |

Definition at line 99 of file basedefs.h.

### 8.16.2.2 RTEMS_ALIGN_DOWN

```
#define RTEMS_ALIGN_DOWN(
              _value,
              _alignment )  ( ( _value ) & ~( ( _alignment ) - 1 ) )
```

Aligns down the value to the alignment.

**Parameters**

| | |
|---|---|
| *_value* | is the value to align down. |
| *_alignment* | is the desired alignment in bytes. The alignment shall be a power of two, otherwise the returned value is undefined. The alignment parameter is evaluated twice. |

**Returns**

Returns the value aligned down to the alignment.

Definition at line 119 of file basedefs.h.

#### 8.16.2.3 RTEMS_ALIGN_UP

```
#define RTEMS_ALIGN_UP(
              _value,
              _alignment )  ( ( ( _value ) + ( _alignment ) - 1 ) & ~( ( _alignment ) - 1 ) )
```

Aligns up the value to the alignment.

**Parameters**

| | |
|---|---|
| *_value* | is the value to align up. |
| *_alignment* | is the desired alignment in bytes. The alignment shall be a power of two, otherwise the returned value is undefined. The alignment parameter is evaluated twice. |

**Returns**

Returns the value aligned up to the alignment.

Definition at line 137 of file basedefs.h.

#### 8.16.2.4 RTEMS_ALIGNED

```
#define RTEMS_ALIGNED(
              _alignment ) __attribute__((__aligned__(_alignment)))
```

Instructs the compiler in a declaration or definition to enforce the alignment.

**Parameters**

| | |
|---|---|
| *_alignment* | is the desired alignment in bytes. |

Definition at line 151 of file basedefs.h.

#### 8.16.2.5 RTEMS_ALLOC_ALIGN

```
#define RTEMS_ALLOC_ALIGN(
              _index ) __attribute__((__alloc_align__(_index)))
```

Tells the compiler in a declaration that the memory allocation alignment parameter of this function is similar to aligned_alloc().

**Parameters**

| | |
|---|---|
| *_index* | is the allocation alignment parameter index (starting with one). |

Definition at line 168 of file basedefs.h.

#### 8.16.2.6 RTEMS_ALLOC_SIZE

```
#define RTEMS_ALLOC_SIZE(
              _index ) __attribute__((__alloc_size__(_index)))
```

Tells the compiler in a declaration that the memory allocation size parameter of this function is similar to malloc().

**Parameters**

| | |
|---|---|
| *_index* | is the allocation size parameter index (starting with one). |

Definition at line 184 of file basedefs.h.

#### 8.16.2.7 RTEMS_ALLOC_SIZE_2

```
#define RTEMS_ALLOC_SIZE_2(
              _count_index,
              _size_index )  __attribute__((__alloc_size__(_count_index, _size_index)))
```

Tells the compiler in a declaration that the memory allocation item count and item size parameter of this function is similar to calloc().

**Parameters**

| | |
|---|---|
| *_count_index* | is the allocation item count parameter index (starting with one). |
| *_size_index* | is the allocation item size parameter index (starting with one). |

Definition at line 204 of file basedefs.h.

#### 8.16.2.8 RTEMS_ARRAY_SIZE

```
#define RTEMS_ARRAY_SIZE(
              _array )  ( sizeof( _array ) / sizeof( ( _array )[ 0 ] ) )
```

Gets the element count of the array.

**Parameters**

| | |
|---|---|
| *_array* | is the name of the array. This parameter is evaluated twice. |

**Returns**

Returns the element count of the array.

Definition at line 221 of file basedefs.h.

#### 8.16.2.9 RTEMS_CONCAT

```
#define RTEMS_CONCAT(
              _x,
              _y ) _x##_y
```

Concatenates _x and _y without expanding.

**Parameters**

| | |
|---|---|
| ↩ _↩ *x* | is the left hand side token of the concatenation. |
| ↩ _↩ *y* | is the right hand side token of the concatenation. |

**Returns**

Returns the concatenation of the tokens _x and _y.

Definition at line 251 of file basedefs.h.

#### 8.16.2.10 RTEMS_CONTAINER_OF

```
#define RTEMS_CONTAINER_OF(
              _m,
```

```
              _type,
              _member_name )  ( (_type *) ( (uintptr_t) ( _m ) - offsetof( _type, _member_name
) ) )
```

Gets the container of a member.

**Parameters**

| | |
|---|---|
| *_m* | is the pointer to a member of the container. |
| *_type* | is the type of the container. |
| *_member_name* | is the designator name of the container member. |

**Returns**

Returns the pointer to the container of a member pointer.

Definition at line 283 of file basedefs.h.

#### 8.16.2.11 RTEMS_DECLARE_GLOBAL_SYMBOL

```
#define RTEMS_DECLARE_GLOBAL_SYMBOL(
              _name ) extern char _name[]
```

Declares a global symbol with the name.

This macro must be placed at file scope.

**Parameters**

| | |
|---|---|
| *_name* | is the name of the global symbol. It shall be a valid designator. |

Definition at line 298 of file basedefs.h.

#### 8.16.2.12 RTEMS_DECONST

```
#define RTEMS_DECONST(
              _type,
              _var ) RTEMS_DEQUALIFY_DEPTHX(*, _type, _var)
```

Performs a type cast which removes const qualifiers without warnings to the type for the pointer variable.

**Parameters**

| | |
|---|---|
| *_type* | is the target type of the cast. |
| *_var* | is the pointer variable. |

Definition at line 455 of file basedefs.h.

#### 8.16.2.13 RTEMS_DEFINE_GLOBAL_SYMBOL

```
#define RTEMS_DEFINE_GLOBAL_SYMBOL(
              _name,
              _value )
```

Defines a global symbol with the name and value.

This macro shall be placed at file scope.

**Parameters**

| | |
|---|---|
| *_name* | is the user defined name of the symbol. The name shall be a valid designator. On the name a macro expansion is performed and afterwards it is stringified. |
| *_value* | is the value of the symbol. On the value a macro expansion is performed and afterwards it is stringified. It shall expand to an integer expression understood by the assembler. |

Definition at line 935 of file basedefs.h.

#### 8.16.2.14 RTEMS_DEQUALIFY

```
#define RTEMS_DEQUALIFY(
              _type,
              _var ) RTEMS_DEQUALIFY_DEPTHX(*, _type, _var)
```

Performs a type cast which removes all qualifiers without warnings to the type for the pointer variable.

**Parameters**

| | |
|---|---|
| *_type* | is the target type of the cast. |
| *_var* | is the pointer variable. |

Definition at line 469 of file basedefs.h.

#### 8.16.2.15 RTEMS_DEQUALIFY_DEPTHX

```
#define RTEMS_DEQUALIFY_DEPTHX(
              _ptr_level,
              _type,
              _var )
```

**Value:**

```
__builtin_choose_expr(__builtin_types_compatible_p( \
    RTEMS_TYPEOF_REFX(_ptr_level, _var), \
    RTEMS_TYPEOF_REFX(_ptr_level, _type) \
  ) || __builtin_types_compatible_p(_type, void *), \
(_type)(_var), \
RTEMS_DEQUALIFY_types_not_compatible())
```

Performs a type cast which removes qualifiers without warnings to the type for the variable.

**Parameters**

| | |
|---|---|
| *_ptr_level* | is the pointer indirection level expressed in *. |
| *_type* | is the target type of the cast. |
| *_var* | is the variable. |

Definition at line 431 of file basedefs.h.

#### 8.16.2.16 RTEMS_DEVOLATILE

```
#define RTEMS_DEVOLATILE(
              _type,
              _var ) RTEMS_DEQUALIFY_DEPTHX(*, _type, _var)
```

Performs a type cast which removes volatile qualifiers without warnings to the type for the pointer variable.

**Parameters**

| | |
|---|---|
| *_type* | is the target type of the cast. |
| *_var* | is the pointer variable. |

Definition at line 483 of file basedefs.h.

#### 8.16.2.17 RTEMS_EXPAND

```
#define RTEMS_EXPAND(
              _token ) _token
```

Helper macro to perform a macro expansion on the token.

**Parameters**

| | |
|---|---|
| *_token* | is the token to expand. |

Definition at line 332 of file basedefs.h.

#### 8.16.2.18 RTEMS_HAVE_MEMBER_SAME_TYPE

```
#define RTEMS_HAVE_MEMBER_SAME_TYPE(
              _t_lhs,
              _m_lhs,
              _t_rhs,
              _m_rhs )
```

**Value:**

```
    __builtin_types_compatible_p( \
      __typeof__( ( (_t_lhs *) 0 )->_m_lhs ), \
      __typeof__( ( (_t_rhs *) 0 )->_m_rhs ) \
    )
```

Checks if members of two types have compatible types.

**Parameters**

| | |
|---|---|
| *_t_lhs* | is the left hand side type. |
| *_m_lhs* | is the left hand side member. |
| *_t_rhs* | is the right hand side type. |
| *_m_rhs* | is the right hand side member. |

**Returns**

Returns to true, if the members of two types have compatible types, otherwise false.

Definition at line 515 of file basedefs.h.

#### 8.16.2.19 RTEMS_OBFUSCATE_VARIABLE

```
#define RTEMS_OBFUSCATE_VARIABLE(
              _var ) __asm__( "" :  "+r" ( _var ) )
```

Obfuscates the variable so that the compiler cannot perform optimizations based on the variable value.

The variable must be simple enough to fit into a register.

**Parameters**

| | |
|---|---|
| *_var* | is the variable to obfuscate. |

Definition at line 606 of file basedefs.h.

#### 8.16.2.20 RTEMS_PREDICT_FALSE

```
#define RTEMS_PREDICT_FALSE(
              _exp ) __builtin_expect( ( _exp ), 0 )
```

Evaluates the integral expression and tells the compiler that the predicted value is false (0).

**Parameters**

| | |
|---|---|
| *_exp* | is the integral expression. |

**Returns**

Returns the value of the integral expression and tells the compiler that the predicted value is false (0).

Definition at line 648 of file basedefs.h.

#### 8.16.2.21 RTEMS_PREDICT_TRUE

```
#define RTEMS_PREDICT_TRUE(
              _exp ) __builtin_expect( ( _exp ), 0 )
```

Evaluates the integral expression and tells the compiler that the predicted value is true (1).

**Parameters**

| | |
|---|---|
| *_exp* | is the integral expression. |

**Returns**

Returns the value of the integral expression and tells the compiler that the predicted value is true (1).

Definition at line 667 of file basedefs.h.

#### 8.16.2.22 RTEMS_PRINTFLIKE

```
#define RTEMS_PRINTFLIKE(
              _format_pos,
              _ap_pos )  __attribute__((__format__(__printf__, _format_pos, _ap_pos)))
```

Tells the compiler in a declaration that this function expects printf()-like arguments.

**Parameters**

| | |
|---|---|
| *_format_pos* | is the position of the format parameter index (starting with one). |
| *_ap_pos* | is the position of the argument pointer parameter index (starting with one). |

Definition at line 687 of file basedefs.h.

#### 8.16.2.23 RTEMS_RETURN_ADDRESS

```
#define RTEMS_RETURN_ADDRESS( ) __builtin_return_address(0)
```

Gets the return address of the current function.

**Returns**

Returns the return address of the current function.

Definition at line 727 of file basedefs.h.

#### 8.16.2.24 RTEMS_SECTION

```
#define RTEMS_SECTION(
              _section ) __attribute__((__section__(_section)))
```

Instructs the compiler to place the variable or function in the section.

**Parameters**

| | |
|---|---|
| *_section* | is the section name as a string. |

Definition at line 743 of file basedefs.h.

#### 8.16.2.25 RTEMS_STATIC_ASSERT

```
#define RTEMS_STATIC_ASSERT(
              _cond,
              _msg ) _Static_assert(_cond, # _msg)
```

Asserts at compile time that the condition is satisfied.

**Parameters**

| | |
|---|---|
| *_cond* | is the condition this static assertion shall satisfy. |
| *_msg* | is the error message in case the static assertion fails. |

Definition at line 762 of file basedefs.h.

#### 8.16.2.26 RTEMS_STRING

```
#define RTEMS_STRING(
              _x ) #_x
```

Stringifies _x without expanding.

**Parameters**

| | |
|---|---|
| ↩ _↩ *x* | is the token to stringify. |

**Returns**

Returns the stringification of the token _x.

Definition at line 345 of file basedefs.h.

### 8.16.2.27 RTEMS_SYMBOL_NAME

```
#define RTEMS_SYMBOL_NAME(
              _name ) RTEMS_EXPAND(_name)
```

Maps the name to the associated symbol name.

**Parameters**

| | |
|---|---|
| *_name* | is the user defined name of the symbol. The name shall be a valid designator. On the name a macro expansion is performed. |

**Returns**

Returns the symbol name associated with the name.

Definition at line 784 of file basedefs.h.

### 8.16.2.28 RTEMS_TYPEOF_REFX

```
#define RTEMS_TYPEOF_REFX(
              _level,
              _target )  __typeof__( _level( union { int z; __typeof__( _target ) x; } ){ 0
}.x )
```

Gets the pointer reference type.

The reference type idea is based on libHX by Jan Engelhardt.

**Parameters**

| | |
|---|---|
| *_level* | is the pointer indirection level expressed in $*$. |
| *_target* | is the reference target type. |

**Returns**

Returns the type of a pointer reference of the specified level to the specified type.

Definition at line 364 of file basedefs.h.

#### 8.16.2.29 RTEMS_WEAK

```
#define RTEMS_WEAK __attribute__((__weak__))
```

Tells the compiler in a function definition that this function should be weak.

Use this attribute for function definitions. Do not use it for function declarations.

Definition at line 875 of file basedefs.h.

#### 8.16.2.30 RTEMS_WEAK_ALIAS

```
#define RTEMS_WEAK_ALIAS(
              _target )  __attribute__((__weak__, __alias__(#_target)))
```

Instructs the compiler to generate a weak alias to the target function.

**Parameters**

| | |
|---|---|
| *_target* | is the target function name. |

Definition at line 891 of file basedefs.h.

#### 8.16.2.31 RTEMS_XCONCAT

```
#define RTEMS_XCONCAT(
              _x,
              _y ) RTEMS_CONCAT( _x, _y )
```

Concatenates expansion of _x and expansion of _y.

**Parameters**

| | |
|---|---|
| ↩ _↩ *x* | is expanded first and then used as the left hand side token of the concatenation. |
| ↩ _↩ *y* | is expanded first and then used as the right hand side token of the concatenation. |

**Returns**

Returns the concatenation of the expansions of tokens _x and _y.

Definition at line 385 of file basedefs.h.

#### 8.16.2.32 RTEMS_XSTRING

```
#define RTEMS_XSTRING(
              _x ) RTEMS_STRING( _x )
```

Stringifies the expansion of _x.

**Parameters**

| | |
|---|---|
| ↩ _↩ x | is the token expand and stringify. |

**Returns**

Returns the stringification of the expansion of token _x.

Definition at line 908 of file basedefs.h.

#### 8.16.2.33 RTEMS_ZERO_LENGTH_ARRAY

```
#define RTEMS_ZERO_LENGTH_ARRAY
```

This constant represents the element count of a zero-length array.

Zero-length arrays are valid in C99 as flexible array members. C++11 does not allow flexible array members. Use the GNU extension which is also supported by other compilers.

Definition at line 950 of file basedefs.h.

# 8.17 Basic Types

This group contains basic Classic API types.

## Classes

- struct rtems_time_of_day

  *This type represents Classic API calendar times.*

## Macros

- #define RTEMS_ID_NONE OBJECTS_ID_NONE

  *This constant represents an invalid RTEMS object identifier.*
- #define RTEMS_NO_TIMEOUT ( (rtems_interval) WATCHDOG_NO_TIMEOUT )

  *This clock tick interval constant indicates that the calling task is willing to wait potentially forever on a resource.*

## Typedefs

- typedef Objects_Id rtems_id

  *This type represents RTEMS object identifiers.*
- typedef Watchdog_Interval rtems_interval

  *This type represents clock tick intervals.*
- typedef uint32_t rtems_name

  *This type represents Classic API object names.*

## 8.17.1 Detailed Description

This group contains basic Classic API types.

## 8.17.2 Macro Definition Documentation

### 8.17.2.1 RTEMS_ID_NONE

```
#define RTEMS_ID_NONE OBJECTS_ID_NONE
```

This constant represents an invalid RTEMS object identifier.

No RTEMS object can have this identifier.

Definition at line 99 of file types.h.

## 8.17.3 Typedef Documentation

### 8.17.3.1 rtems_name

```
typedef uint32_t rtems_name
```

This type represents Classic API object names.

It is an unsigned 32-bit integer which can be treated as a numeric value or initialized using rtems_build_name() to encode four ASCII characters. A value of zero may have a special meaning in some directives.

Definition at line 220 of file types.h.

## 8.18 Bitmap Priority Thread Routines

Bitmap Priority Thread Routines.

### Files

- file prioritybitmap.h

  *Manipulation Routines for the Bitmap Priority Queue Implementation.*

### Classes

- struct Priority_bit_map_Control
- struct Priority_bit_map_Information

### Typedefs

- typedef uint16_t **Priority_bit_map_Word**

### 8.18.1 Detailed Description

Bitmap Priority Thread Routines.

# 8.19 Block Device Cache Configuration

## Macros

- #define CONFIGURE_APPLICATION_NEEDS_LIBBLOCK

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_BDBUF_BUFFER_MAX_SIZE

  *This configuration option is an integer define.*
- #define CONFIGURE_BDBUF_BUFFER_MIN_SIZE

  *This configuration option is an integer define.*
- #define CONFIGURE_BDBUF_CACHE_MEMORY_SIZE

  *This configuration option is an integer define.*
- #define CONFIGURE_BDBUF_MAX_READ_AHEAD_BLOCKS

  *This configuration option is an integer define.*
- #define CONFIGURE_BDBUF_MAX_WRITE_BLOCKS

  *This configuration option is an integer define.*
- #define CONFIGURE_BDBUF_READ_AHEAD_TASK_PRIORITY

  *This configuration option is an integer define.*
- #define CONFIGURE_BDBUF_TASK_STACK_SIZE

  *This configuration option is an integer define.*
- #define CONFIGURE_SWAPOUT_BLOCK_HOLD

  *This configuration option is an integer define.*
- #define CONFIGURE_SWAPOUT_SWAP_PERIOD

  *This configuration option is an integer define.*
- #define CONFIGURE_SWAPOUT_TASK_PRIORITY

  *This configuration option is an integer define.*
- #define CONFIGURE_SWAPOUT_WORKER_TASKS

  *This configuration option is an integer define.*
- #define CONFIGURE_SWAPOUT_WORKER_TASK_PRIORITY

  *This configuration option is an integer define.*

### 8.19.1 Detailed Description

This section describes configuration options related to the Block Device Cache (bdbuf).

### 8.19.2 Macro Definition Documentation

#### 8.19.2.1 CONFIGURE_APPLICATION_NEEDS_LIBBLOCK

```
#define CONFIGURE_APPLICATION_NEEDS_LIBBLOCK
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the Block Device Cache is initialized during system initialization.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

Each option of the Block Device Cache (bdbuf) configuration can be explicitly set by the user with the configuration options below. The Block Device Cache is used for example by the RFS and DOSFS filesystems.

Definition at line 81 of file appl-config.h.

#### 8.19.2.2 CONFIGURE_BDBUF_BUFFER_MAX_SIZE

```
#define CONFIGURE_BDBUF_BUFFER_MAX_SIZE
```

This configuration option is an integer define.

The value of this configuration option defines the maximum size of a buffer in bytes.

**Default Value**

The default value is 4096.

**Value Constraints**

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be an integral multiple of CONFIGURE_BDBUF_BUFFER_MIN_SIZE.

Definition at line 104 of file appl-config.h.

#### 8.19.2.3 CONFIGURE_BDBUF_BUFFER_MIN_SIZE

```
#define CONFIGURE_BDBUF_BUFFER_MIN_SIZE
```

This configuration option is an integer define.

The value of this configuration option defines the minimum size of a buffer in bytes.

**Default Value**

The default value is 512.

**Value Constraints**

The value of this configuration option shall be greater than or equal to 0 and less than or equal to UIN↩T32_MAX.

Definition at line 122 of file appl-config.h.

#### 8.19.2.4 CONFIGURE_BDBUF_CACHE_MEMORY_SIZE

```
#define CONFIGURE_BDBUF_CACHE_MEMORY_SIZE
```

This configuration option is an integer define.

The value of this configuration option defines the size of the cache memory in bytes.

**Default Value**

The default value is 32768.

**Value Constraints**

The value of this configuration option shall be greater than or equal to 0 and less than or equal to SIZE_↩MAX.

Definition at line 140 of file appl-config.h.

#### 8.19.2.5 CONFIGURE_BDBUF_MAX_READ_AHEAD_BLOCKS

```
#define CONFIGURE_BDBUF_MAX_READ_AHEAD_BLOCKS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum blocks per read-ahead request.

##### Default Value

The default value is 0.

##### Value Constraints

The value of this configuration option shall be greater than or equal to 0 and less than or equal to UINT32_MAX.

##### Notes

A value of 0 disables the read-ahead task (default). The read-ahead task will issue speculative read transfers if a sequential access pattern is detected. This can improve the performance on some systems.

Definition at line 163 of file appl-config.h.

#### 8.19.2.6 CONFIGURE_BDBUF_MAX_WRITE_BLOCKS

```
#define CONFIGURE_BDBUF_MAX_WRITE_BLOCKS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum blocks per write request.

##### Default Value

The default value is 16.

##### Value Constraints

The value of this configuration option shall be greater than or equal to 0 and less than or equal to UINT32_MAX.

Definition at line 181 of file appl-config.h.

#### 8.19.2.7 CONFIGURE_BDBUF_READ_AHEAD_TASK_PRIORITY

```
#define CONFIGURE_BDBUF_READ_AHEAD_TASK_PRIORITY
```

This configuration option is an integer define.

The value of this configuration option defines the read-ahead task priority.

##### Default Value

The default value is 15.

##### Value Constraints

The value of this configuration option shall be a valid Classic API task priority. The set of valid task priorities is scheduler-specific.

Definition at line 197 of file appl-config.h.

#### 8.19.2.8 CONFIGURE_BDBUF_TASK_STACK_SIZE

```
#define CONFIGURE_BDBUF_TASK_STACK_SIZE
```

This configuration option is an integer define.

The value of this configuration option defines the task stack size of the Block Device Cache tasks in bytes.

##### Default Value

The default value is RTEMS_MINIMUM_STACK_SIZE.

##### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to CONFIGURE_MINIMUM_TASK_STACK_SIZE.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It shall be small enough so that the task stack space calculation carried out by <rtems/confdefs.h> does not overflow an integer of type uintptr_t.

Definition at line 226 of file appl-config.h.

#### 8.19.2.9 CONFIGURE_SWAPOUT_BLOCK_HOLD

```
#define CONFIGURE_SWAPOUT_BLOCK_HOLD
```

This configuration option is an integer define.

The value of this configuration option defines the swapout task maximum block hold time in milliseconds.

**Default Value**

The default value is 1000.

**Value Constraints**

The value of this configuration option shall be greater than or equal to 0 and less than or equal to UINT32_MAX.

Definition at line 244 of file appl-config.h.

#### 8.19.2.10 CONFIGURE_SWAPOUT_SWAP_PERIOD

```
#define CONFIGURE_SWAPOUT_SWAP_PERIOD
```

This configuration option is an integer define.

The value of this configuration option defines the swapout task swap period in milliseconds.

**Default Value**

The default value is 250.

**Value Constraints**

The value of this configuration option shall be greater than or equal to 0 and less than or equal to UINT32_MAX.

Definition at line 262 of file appl-config.h.

#### 8.19.2.11 CONFIGURE_SWAPOUT_TASK_PRIORITY

```
#define CONFIGURE_SWAPOUT_TASK_PRIORITY
```

This configuration option is an integer define.

The value of this configuration option defines the swapout task priority.

**Default Value**

The default value is 15.

**Value Constraints**

The value of this configuration option shall be a valid Classic API task priority. The set of valid task priorities is scheduler-specific.

Definition at line 278 of file appl-config.h.

#### 8.19.2.12 CONFIGURE_SWAPOUT_WORKER_TASK_PRIORITY

```
#define CONFIGURE_SWAPOUT_WORKER_TASK_PRIORITY
```

This configuration option is an integer define.

The value of this configuration option defines the swapout worker task priority.

**Default Value**

The default value is 15.

**Value Constraints**

The value of this configuration option shall be a valid Classic API task priority. The set of valid task priorities is scheduler-specific.

Definition at line 313 of file appl-config.h.

#### 8.19.2.13 CONFIGURE_SWAPOUT_WORKER_TASKS

```
#define CONFIGURE_SWAPOUT_WORKER_TASKS
```

This configuration option is an integer define.

The value of this configuration option defines the swapout worker task count.

**Default Value**

The default value is 0.

**Value Constraints**

The value of this configuration option shall be greater than or equal to 0 and less than or equal to UIN↩T32_MAX.

Definition at line 296 of file appl-config.h.

## 8.20 Board Support Packages

This group contains the Board Support Packages (BSPs) and support modules.

### Modules

- SPARC

  *SPARC Board Support Packages.*
- Shared

  *This group contains the architecture-independent support modules shared across all BSPs.*

### 8.20.1 Detailed Description

This group contains the Board Support Packages (BSPs) and support modules.

## 8.21 Boolean Checks

Checks for boolean expressions.

### Macros

- #define **T_true**(...) T_flags_true(0, __VA_ARGS__)
- #define **T_assert_true**(...) T_flags_true(T_CHECK_STOP, __VA_ARGS__)
- #define **T_quiet_true**(...) T_flags_true(T_CHECK_QUIET, __VA_ARGS__)
- #define **T_step_true**(s, ...) T_flags_true(T_CHECK_STEP(s), __VA_ARGS__)
- #define **T_step_assert_true**(s, ...) T_flags_true(T_CHECK_STEP(s) | T_CHECK_STOP, __VA_ARGS__)
- #define **T_false**(...)
- #define **T_assert_false**(...)
- #define **T_quiet_false**(...)
- #define **T_step_false**(s, ...)
- #define **T_step_assert_false**(s, ...)

### 8.21.1 Detailed Description

Checks for boolean expressions.

### 8.21.2 Macro Definition Documentation

#### 8.21.2.1 T_assert_false

```
#define T_assert_false(
               ...  )
```

**Value:**

```
    T_flags_true(T_CHECK_STOP, \
    !(T_VA_ARGS_FIRST(__VA_ARGS__)) T_VA_ARGS_MORE(__VA_ARGS__))
```

Definition at line 709 of file test.h.

#### 8.21.2.2 T_false

```
#define T_false(
               ...  )
```

**Value:**

```
    T_flags_true(0, \
    !(T_VA_ARGS_FIRST(__VA_ARGS__)) T_VA_ARGS_MORE(__VA_ARGS__))
```

Definition at line 706 of file test.h.

#### 8.21.2.3 T_quiet_false

```
#define T_quiet_false(
              ...  )
```

**Value:**

```
    T_flags_true(T_CHECK_QUIET, \
    !(T_VA_ARGS_FIRST(__VA_ARGS__)) T_VA_ARGS_MORE(__VA_ARGS__))
```

Definition at line 712 of file test.h.

#### 8.21.2.4 T_step_assert_false

```
#define T_step_assert_false(
              s,
              ...  )
```

**Value:**

```
    T_flags_true(T_CHECK_STEP(s) | T_CHECK_STOP, \
    !(T_VA_ARGS_FIRST(__VA_ARGS__)) T_VA_ARGS_MORE(__VA_ARGS__))
```

Definition at line 718 of file test.h.

#### 8.21.2.5 T_step_false

```
#define T_step_false(
              s,
              ...  )
```

**Value:**

```
    T_flags_true(T_CHECK_STEP(s), \
    !(T_VA_ARGS_FIRST(__VA_ARGS__)) T_VA_ARGS_MORE(__VA_ARGS__))
```

Definition at line 715 of file test.h.

# 8.22 Bootcard

Standard system startup.

## Files

- file bootcard.h
- file bootcard.c

## Functions

- void **bsp_start** (void)
- void **bsp_reset** (void)
- RTEMS_NO_RETURN void boot_card (const char *cmdline)

  *Standard system initialization procedure.*
- void bsp_start_on_secondary_processor (struct Per_CPU_Control *cpu_self)

  *Standard start routine for secondary processors.*

## Variables

- const char * bsp_boot_cmdline

  *Global pointer to the command line of boot_card().*

## 8.22.1 Detailed Description

Standard system startup.

## 8.22.2 Function Documentation

### 8.22.2.1 boot_card()

```
RTEMS_NO_RETURN void boot_card (
            const char * cmdline )
```

Standard system initialization procedure.

You may pass a command line in *cmdline*. It is later available via the global bsp_boot_cmdline variable.

This is the C entry point for ALL RTEMS BSPs. It is invoked from the assembly language initialization file usually called `start.S` which does the basic CPU setup (stack, C runtime environment, zero BSS, load other sections) and calls afterwards boot_card(). The boot card function provides the framework for the BSP initialization sequence. For the basic flow of initialization see RTEMS C User's Guide, Initialization Manager.

This style of initialization ensures that the C++ global constructors are executed after RTEMS is initialized.

Definition at line 39 of file bootcard.c.

#### 8.22.2.2 bsp_start_on_secondary_processor()

```
void bsp_start_on_secondary_processor (
            struct Per_CPU_Control * cpu_self )
```

Standard start routine for secondary processors.

This function is usually called by low-level startup code of secondary processors or boot loaders starting a secondary processor. The final step of this function is a call to _SMP_Start_multitasking_on_secondary_processor().

Definition at line 38 of file bspsmp.c.

## 8.23 CPU Architecture Support

Provides CPU architecture dependent services.

### Modules

- Object Handler
- SPARC

  *SPARC Architecture Support.*

### 8.23.1 Detailed Description

Provides CPU architecture dependent services.

## 8.24 Cache Manager

The Cache Manager provides functions to perform maintenance operations for data and instruction caches.

### Functions

- void * rtems_cache_aligned_malloc (size_t nbytes)
  %
- void rtems_cache_coherent_add_area (void *area_begin, uintptr_t area_size)
  %
- void * rtems_cache_coherent_allocate (size_t size, uintptr_t alignment, uintptr_t boundary)
  %
- void rtems_cache_coherent_free (void *ptr)
  %
- void rtems_cache_disable_data (void)
  %
- void rtems_cache_disable_instruction (void)
  %
- void rtems_cache_enable_data (void)
  %
- void rtems_cache_enable_instruction (void)
  %
- void rtems_cache_flush_entire_data (void)
  %
- void rtems_cache_flush_multiple_data_lines (const void *addr, size_t size)
  %
- void rtems_cache_freeze_data (void)
  %
- void rtems_cache_freeze_instruction (void)
  %
- size_t rtems_cache_get_data_line_size (void)
  %
- size_t rtems_cache_get_data_cache_size (uint32_t level)
  %
- size_t rtems_cache_get_instruction_line_size (void)
  %
- size_t rtems_cache_get_instruction_cache_size (uint32_t level)
  %
- size_t rtems_cache_get_maximal_line_size (void)
  %
- void rtems_cache_instruction_sync_after_code_change (const void *code_addr, size_t n_bytes)
  %
- void rtems_cache_invalidate_entire_data (void)
  %
- void rtems_cache_invalidate_entire_instruction (void)
  %
- void rtems_cache_invalidate_multiple_data_lines (const void *addr, size_t size)
  %
- void rtems_cache_invalidate_multiple_instruction_lines (const void *addr, size_t size)
  %
- void rtems_cache_unfreeze_data (void)
  %
- void rtems_cache_unfreeze_instruction (void)
  %

### 8.24.1 Detailed Description

The Cache Manager provides functions to perform maintenance operations for data and instruction caches.

### 8.24.2 Function Documentation

#### 8.24.2.1 rtems_cache_aligned_malloc()

```
void* rtems_cache_aligned_malloc (
            size_t nbytes )
```

%

**Parameters**

| | |
|---|---|
| *nbytes* | % |

#### 8.24.2.2 rtems_cache_coherent_add_area()

```
void rtems_cache_coherent_add_area (
            void * area_begin,
            uintptr_t area_size )
```

%

**Parameters**

| | |
|---|---|
| *area_begin* | % |
| *area_size* | % |

#### 8.24.2.3 rtems_cache_coherent_allocate()

```
void* rtems_cache_coherent_allocate (
            size_t size,
            uintptr_t alignment,
            uintptr_t boundary )
```

%

**Parameters**

| | |
|---|---|
| *size* | % |
| *alignment* | % |
| *boundary* | % |

#### 8.24.2.4 rtems_cache_coherent_free()

```
void rtems_cache_coherent_free (
            void * ptr )
```

%

**Parameters**

| | |
|---|---|
| *ptr* | % |

#### 8.24.2.5 rtems_cache_flush_multiple_data_lines()

```
void rtems_cache_flush_multiple_data_lines (
            const void * addr,
            size_t size )
```

%

**Parameters**

| | |
|---|---|
| *addr* | % |
| *size* | % |

Definition at line 109 of file cacheimpl.h.

#### 8.24.2.6 rtems_cache_get_data_cache_size()

```
size_t rtems_cache_get_data_cache_size (
            uint32_t level )
```

%

**Parameters**

| | |
|---|---|
| *level* | % |

Definition at line 216 of file cacheimpl.h.

#### 8.24.2.7 rtems_cache_get_instruction_cache_size()

```
size_t rtems_cache_get_instruction_cache_size (
            uint32_t level )
```

%

**Parameters**

| | |
|---|---|
| *level* | % |

Definition at line 390 of file cacheimpl.h.

#### 8.24.2.8 rtems_cache_instruction_sync_after_code_change()

```
void rtems_cache_instruction_sync_after_code_change (
            const void * code_addr,
            size_t n_bytes )
```

%

**Parameters**

| | |
|---|---|
| *code_addr* | % |
| *n_bytes* | % |

Definition at line 465 of file cacheimpl.h.

#### 8.24.2.9 rtems_cache_invalidate_multiple_data_lines()

```
void rtems_cache_invalidate_multiple_data_lines (
            const void * addr,
            size_t size )
```

%

**Parameters**

| | |
|---|---|
| *addr* | % |
| *size* | % |

Definition at line 143 of file cacheimpl.h.

#### 8.24.2.10 rtems_cache_invalidate_multiple_instruction_lines()

```
void rtems_cache_invalidate_multiple_instruction_lines (
             const void * addr,
             size_t size )
```

%

**Parameters**

| | |
|---|---|
| *addr* | % |
| *size* | % |

Definition at line 344 of file cacheimpl.h.

## 8.25 Chain Handler

Provides Data Structures Chain Node and Chain Control.

### Files

- file chain.h

  *Chain Handler API.*
- file chainimpl.h

  *Chain Handler API.*
- file chain.c

  *Initialize a Chain Header.*

### Classes

- struct Chain_Node_struct
- struct Chain_Control
- struct Chain_Iterator

  *A chain iterator which is updated during node extraction if it is properly registered.*
- struct Chain_Iterator_registry

  *A registry for chain iterators.*

### Macros

- #define CHAIN_INITIALIZER_EMPTY(name) { { { &(name).Tail.Node, NULL }, &(name).Head.Node } }

  *Chain initializer for an empty chain with designator name.*
- #define CHAIN_INITIALIZER_ONE_NODE(node) { { { (node), NULL }, (node) } }

  *Chain initializer for a chain with one node.*
- #define CHAIN_NODE_INITIALIZER_ONE_NODE_CHAIN(chain) { &(chain)->Tail.Node, &(chain)->Head.Node }

  *Chain node initializer for a chain containing exactly this node.*
- #define CHAIN_DEFINE_EMPTY(name) Chain_Control name = CHAIN_INITIALIZER_EMPTY(name)

  *Chain definition for an empty chain with designator name.*
- #define CHAIN_ITERATOR_REGISTRY_INITIALIZER(name) { CHAIN_INITIALIZER_EMPTY( name.↩ Iterators ) }

  *Chain iterator registry initializer for static initialization.*

### Typedefs

- typedef struct Chain_Node_struct Chain_Node
- typedef bool(* Chain_Node_order) (const void *left, const Chain_Node *right)

  *Chain node order.*

### Enumerations

- enum Chain_Iterator_direction { CHAIN_ITERATOR_FORWARD, CHAIN_ITERATOR_BACKWARD }

  *The chain iterator direction.*

## Functions

- void _Chain_Initialize (Chain_Control *the_chain, void *starting_address, size_t number_nodes, size_t node_size)

  *Initializes a chain header.*
- size_t _Chain_Node_count_unprotected (const Chain_Control *chain)

  *Returns the node count of the chain.*
- static __inline__ void _Chain_Set_off_chain (Chain_Node *node)

  *Sets off chain.*
- static __inline__ void _Chain_Initialize_node (Chain_Node *the_node)

  *Initializes a chain node.*
- static __inline__ bool _Chain_Is_node_off_chain (const Chain_Node *node)

  *Checks if the node is off chain.*
- static __inline__ bool _Chain_Are_nodes_equal (const Chain_Node *left, const Chain_Node *right)

  *Checks if two nodes are equal.*
- static __inline__ Chain_Node * _Chain_Head (Chain_Control *the_chain)

  *Returns pointer to chain head.*
- static __inline__ const Chain_Node * _Chain_Immutable_head (const Chain_Control *the_chain)

  *Returns pointer to immutable chain head.*
- static __inline__ Chain_Node * _Chain_Tail (Chain_Control *the_chain)

  *Returns pointer to chain tail.*
- static __inline__ const Chain_Node * _Chain_Immutable_tail (const Chain_Control *the_chain)

  *Returns pointer to immutable chain tail.*
- static __inline__ Chain_Node * _Chain_First (const Chain_Control *the_chain)

  *Returns pointer to chain's first node.*
- static __inline__ const Chain_Node * _Chain_Immutable_first (const Chain_Control *the_chain)

  *Returns pointer to immutable chain's first node.*
- static __inline__ Chain_Node * _Chain_Last (const Chain_Control *the_chain)

  *Returns pointer to chain's last node.*
- static __inline__ const Chain_Node * _Chain_Immutable_last (const Chain_Control *the_chain)

  *Returns pointer to immutable chain's last node.*
- static __inline__ Chain_Node * _Chain_Next (const Chain_Node *the_node)

  *Returns pointer to the next node from this node.*
- static __inline__ const Chain_Node * _Chain_Immutable_next (const Chain_Node *the_node)

  *Returns pointer to the immutable next node from this node.*
- static __inline__ Chain_Node * _Chain_Previous (const Chain_Node *the_node)

  *Returns pointer to the previous node from this node.*
- static __inline__ const Chain_Node * _Chain_Immutable_previous (const Chain_Node *the_node)

  *Returns pointer to the immutable previous node from this node.*
- static __inline__ bool _Chain_Is_empty (const Chain_Control *the_chain)

  *Checks if the chain is empty.*
- static __inline__ bool _Chain_Is_first (const Chain_Node *the_node)

  *Checks if this is the first node on the chain.*
- static __inline__ bool _Chain_Is_last (const Chain_Node *the_node)

  *Checks if this is the last node on the chain.*
- static __inline__ bool _Chain_Has_only_one_node (const Chain_Control *the_chain)

  *Checks if this chain has only one node.*
- static __inline__ bool _Chain_Is_head (const Chain_Control *the_chain, const Chain_Node *the_node)

  *Checks if this node is the chain head.*
- static __inline__ bool _Chain_Is_tail (const Chain_Control *the_chain, const Chain_Node *the_node)

  *Checks if this node is the chain tail.*

- static __inline__ void _Chain_Initialize_empty (Chain_Control *the_chain)

  *Initializes this chain as empty.*
- static __inline__ void _Chain_Initialize_one (Chain_Control *the_chain, Chain_Node *the_node)

  *Initializes this chain to contain exactly the specified node.*
- static __inline__ void _Chain_Extract_unprotected (Chain_Node *the_node)

  *Extracts this node (unprotected).*
- static __inline__ Chain_Node * _Chain_Get_first_unprotected (Chain_Control *the_chain)

  *Gets the first node (unprotected).*
- static __inline__ Chain_Node * _Chain_Get_unprotected (Chain_Control *the_chain)

  *Gets the first node (unprotected).*
- static __inline__ void _Chain_Insert_unprotected (Chain_Node *after_node, Chain_Node *the_node)

  *Inserts a node (unprotected).*
- static __inline__ void _Chain_Append_unprotected (Chain_Control *the_chain, Chain_Node *the_node)

  *Appends a node (unprotected).*
- static __inline__ void _Chain_Append_if_is_off_chain_unprotected (Chain_Control *the_chain, Chain_Node *the_node)

  *Appends a node on the end of a chain if the node is in the off chain state (unprotected).*
- static __inline__ void _Chain_Prepend_unprotected (Chain_Control *the_chain, Chain_Node *the_node)

  *Prepends a node (unprotected).*
- static __inline__ bool _Chain_Append_with_empty_check_unprotected (Chain_Control *the_chain, Chain_Node *the_node)

  *Appends a node and checks if the chain was empty before (unprotected).*
- static __inline__ bool _Chain_Prepend_with_empty_check_unprotected (Chain_Control *the_chain, Chain_Node *the_node)

  *Prepends a node and checks if the chain was empty before (unprotected).*
- static __inline__ bool _Chain_Get_with_empty_check_unprotected (Chain_Control *the_chain, Chain_Node **the_node)

  *Gets the first node and checks if the chain is empty afterwards (unprotected).*
- static __inline__ void _Chain_Insert_ordered_unprotected (Chain_Control *the_chain, Chain_Node *to_insert, const void *left, Chain_Node_order order)

  *Inserts a node into the chain according to the order relation.*
- static __inline__ void _Chain_Iterator_registry_initialize (Chain_Iterator_registry *the_registry)

  *Initializes a chain iterator registry.*
- static __inline__ void _Chain_Iterator_registry_update (Chain_Iterator_registry *the_registry, Chain_Node *the_node_to_extract)

  *Updates all iterators present in the chain iterator registry in case of a node extraction.*
- static __inline__ void _Chain_Iterator_initialize (Chain_Control *the_chain, Chain_Iterator_registry *the_registry, Chain_Iterator *the_iterator, Chain_Iterator_direction direction)

  *Initializes the chain iterator.*
- static __inline__ Chain_Node * _Chain_Iterator_next (const Chain_Iterator *the_iterator)

  *Returns the next node in the iterator direction.*
- static __inline__ void _Chain_Iterator_set_position (Chain_Iterator *the_iterator, Chain_Node *the_node)

  *Sets the iterator position.*
- static __inline__ void _Chain_Iterator_destroy (Chain_Iterator *the_iterator)

  *Destroys the iterator.*

### 8.25.1 Detailed Description

Provides Data Structures Chain Node and Chain Control.

The Chain Handler is used to manage sets of entities. This handler provides two data structures. The Chain Node data structure is included as the first part of every data structure that will be placed on a chain. The second data structure is Chain Control which is used to manage a set of Chain Nodes.

### 8.25.2 Macro Definition Documentation

#### 8.25.2.1 CHAIN_INITIALIZER_ONE_NODE

```
#define CHAIN_INITIALIZER_ONE_NODE(
              node )  { { { (node), NULL }, (node) } }
```

Chain initializer for a chain with one *node*.

**See also**

CHAIN_NODE_INITIALIZER_ONE_NODE_CHAIN().

Definition at line 47 of file chainimpl.h.

#### 8.25.2.2 CHAIN_ITERATOR_REGISTRY_INITIALIZER

```
#define CHAIN_ITERATOR_REGISTRY_INITIALIZER(
              name )  { CHAIN_INITIALIZER_EMPTY( name.Iterators ) }
```

Chain iterator registry initializer for static initialization.

**Parameters**

| | |
|---|---|
| *name* | The designator of the chain iterator registry. |

Definition at line 926 of file chainimpl.h.

#### 8.25.2.3 CHAIN_NODE_INITIALIZER_ONE_NODE_CHAIN

```
#define CHAIN_NODE_INITIALIZER_ONE_NODE_CHAIN(
              chain )  { &(chain)->Tail.Node, &(chain)->Head.Node }
```

Chain node initializer for a *chain* containing exactly this node.

**See also**

CHAIN_INITIALIZER_ONE_NODE().

Definition at line 55 of file chainimpl.h.

## 8.25.3 Typedef Documentation

### 8.25.3.1 Chain_Node

```
Chain_Node
```

This type definition promotes the name for the Chain Node used by all RTEMS code. It is a separate type definition because a forward reference is required to define it. See Chain_Node_struct for detailed information.

Definition at line 1 of file chain.h.

### 8.25.3.2 Chain_Node_order

```
typedef bool( * Chain_Node_order) (const void *left, const Chain_Node *right)
```

Chain node order.

**Parameters**

| | |
|---|---|
| *left* | The left hand side. |
| *right* | The right hand side. |

**Return values**

| | |
|---|---|
| *true* | According to the order the left node precedes the right node. |
| *false* | Otherwise. |

Definition at line 827 of file chainimpl.h.

## 8.25.4 Enumeration Type Documentation

### 8.25.4.1 Chain_Iterator_direction

```
enum Chain_Iterator_direction
```

The chain iterator direction.

**Enumerator**

| | |
|---|---|
| CHAIN_ITERATOR_FORWARD | Iteration from head to tail. |
| CHAIN_ITERATOR_BACKWARD | Iteration from tail to head. |

Definition at line 867 of file chainimpl.h.

## 8.25.5 Function Documentation

### 8.25.5.1 _Chain_Append_if_is_off_chain_unprotected()

```
static __inline__ void _Chain_Append_if_is_off_chain_unprotected (
            Chain_Control * the_chain,
            Chain_Node * the_node )  [static]
```

Appends a node on the end of a chain if the node is in the off chain state (unprotected).

**Parameters**

| | | |
|---|---|---|
| in,out | *the_chain* | The chain to be operated upon. |
| in,out | *the_node* | The node to be appended if it is in the off chain. |

**Note**

It does NOT disable interrupts to ensure the atomicity of the append operation.

**See also**

_Chain_Append_unprotected() and _Chain_Is_node_off_chain().

Definition at line 694 of file chainimpl.h.

### 8.25.5.2 _Chain_Append_unprotected()

```
static __inline__ void _Chain_Append_unprotected (
            Chain_Control * the_chain,
            Chain_Node * the_node )  [static]
```

Appends a node (unprotected).

This routine appends the_node onto the end of the_chain.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_chain* | The chain to be operated upon. |
| out | *the_node* | The node to be appended to the end of *the_chain*. |

**Note**

It does NOT disable interrupts to ensure the atomicity of the append operation.

Definition at line 663 of file chainimpl.h.

#### 8.25.5.3 _Chain_Append_with_empty_check_unprotected()

```
static __inline__ bool _Chain_Append_with_empty_check_unprotected (
            Chain_Control * the_chain,
            Chain_Node * the_node )  [static]
```

Appends a node and checks if the chain was empty before (unprotected).

This routine appends the_node onto the end of the_chain.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_chain* | The chain to be operated upon. |
| out | *the_node* | The node to be appended to the end of *the_chain*. |

**Note**

It does NOT disable interrupts to ensure the atomicity of the append operation.

**Return values**

| | |
|---|---|
| *true* | The chain was empty before. |
| *false* | The chain contained at least one node before. |

Definition at line 737 of file chainimpl.h.

#### 8.25.5.4 _Chain_Are_nodes_equal()

```
static __inline__ bool _Chain_Are_nodes_equal (
            const Chain_Node * left,
            const Chain_Node * right )  [static]
```

Checks if two nodes are equal.

This function returns true if *left* and *right* are equal, and false otherwise.

**Parameters**

| | |
|---|---|
| *left* | The node on the left hand side of the comparison. |
| *right* | The node on the right hand side of the comparison. |

**Return values**

| | |
|---|---|
| *true* | *left* and *right* are equal. |
| *false* | *left* and *right* are not equal. |

Definition at line 161 of file chainimpl.h.

#### 8.25.5.5 _Chain_Extract_unprotected()

```
static __inline__ void _Chain_Extract_unprotected (
            Chain_Node * the_node )  [static]
```

Extracts this node (unprotected).

This routine extracts the_node from the chain on which it resides. It does NOT disable interrupts to ensure the atomicity of the extract operation.

**Parameters**

| | | |
|---|---|---|
| out | *the_node* | The node to be extracted. |

Definition at line 541 of file chainimpl.h.

#### 8.25.5.6 _Chain_First()

```
static __inline__ Chain_Node* _Chain_First (
            const Chain_Control * the_chain )  [static]
```

Returns pointer to chain's first node.

This function returns a pointer to the first node on the chain after the head.

**Parameters**

| | |
|---|---|
| *the_chain* | The chain to be operated upon. |

**Returns**

This method returns the first node of the chain.

Definition at line 243 of file chainimpl.h.

#### 8.25.5.7 _Chain_Get_first_unprotected()

```
static __inline__ Chain_Node* _Chain_Get_first_unprotected (
            Chain_Control * the_chain )  [static]
```

Gets the first node (unprotected).

This function removes the first node from the_chain and returns a pointer to that node. It does NOT disable interrupts to ensure the atomicity of the get operation.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_chain* | The chain to attempt to get the first node from. |

**Returns**

This method returns the first node on the chain even if it is the Chain Tail.

**Note**

This routine assumes that there is at least one node on the chain and always returns a node even if it is the Chain Tail.

Definition at line 575 of file chainimpl.h.

#### 8.25.5.8 _Chain_Get_unprotected()

```
static __inline__ Chain_Node* _Chain_Get_unprotected (
            Chain_Control * the_chain )  [static]
```

Gets the first node (unprotected).

This function removes the first node from the_chain and returns a pointer to that node. If the_chain is empty, then NULL is returned.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_chain* | The chain to attempt to get the first node from. |

**Return values**

| | |
|---|---|
| *pointer* | Pointer to the first node. The chain contained at least one node. |
| *NULL* | The chain is empty. |

**Note**

It does NOT disable interrupts to ensure the atomicity of the get operation.

Definition at line 613 of file chainimpl.h.

#### 8.25.5.9 _Chain_Get_with_empty_check_unprotected()

```
static __inline__ bool _Chain_Get_with_empty_check_unprotected (
            Chain_Control * the_chain,
            Chain_Node ** the_node )  [static]
```

Gets the first node and checks if the chain is empty afterwards (unprotected).

This function removes the first node from the_chain and returns a pointer to that node in *the_node*. If the_chain is empty, then NULL is returned.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_chain* | The chain to attempt to get the first node from. |
| out | *the_node* | The first node on the chain or NULL if the chain is empty. |

**Note**

It does NOT disable interrupts to ensure the atomicity of the get operation.

**Return values**

| | |
|---|---|
| *true* | The chain is empty now. |
| *false* | The chain contains at least one node now. |

Definition at line 793 of file chainimpl.h.

#### 8.25.5.10 _Chain_Has_only_one_node()

```
static __inline__ bool _Chain_Has_only_one_node (
            const Chain_Control * the_chain )  [static]
```

Checks if this chain has only one node.

This function returns true if there is only one node on *the_chain* and false otherwise.

**Parameters**

| | |
|---|---|
| *the_chain* | is the chain to be operated upon. |

**Return values**

| | |
|---|---|
| *true* | There is only one node on *the_chain*. |

**Return values**

| | |
|---|---|
| *false* | There is more than one node on *the_chain*. |

Definition at line 433 of file chainimpl.h.

#### 8.25.5.11 _Chain_Head()

```
static __inline__ Chain_Node* _Chain_Head (
            Chain_Control * the_chain )  [static]
```

Returns pointer to chain head.

This function returns a pointer to the head node on the chain.

**Parameters**

| | | |
|---|---|---|
| in | *the_chain* | The chain to be operated upon. |

**Returns**

This method returns the permanent head node of the chain.

Definition at line 178 of file chainimpl.h.

#### 8.25.5.12 _Chain_Immutable_first()

```
static __inline__ const Chain_Node* _Chain_Immutable_first (
            const Chain_Control * the_chain )  [static]
```

Returns pointer to immutable chain's first node.

This function returns a pointer to the first node on the chain after the head.

**Parameters**

| | |
|---|---|
| *the_chain* | The chain to be operated upon. |

**Returns**

This method returns the first node of the chain.

Definition at line 260 of file chainimpl.h.

#### 8.25.5.13 _Chain_Immutable_head()

```
static __inline__ const Chain_Node* _Chain_Immutable_head (
            const Chain_Control * the_chain )  [static]
```

Returns pointer to immutable chain head.

This function returns a pointer to the head node on the chain.

**Parameters**

| | |
|---|---|
| *the_chain* | The chain to be operated upon. |

**Returns**

This method returns the permanent head node of the chain.

Definition at line 194 of file chainimpl.h.

#### 8.25.5.14 _Chain_Immutable_last()

```
static __inline__ const Chain_Node* _Chain_Immutable_last (
            const Chain_Control * the_chain )  [static]
```

Returns pointer to immutable chain's last node.

This function returns a pointer to the last node on the chain just before the tail.

**Parameters**

| | |
|---|---|
| *the_chain* | The chain to be operated upon. |

**Returns**

This method returns the last node of the chain.

Definition at line 294 of file chainimpl.h.

#### 8.25.5.15 _Chain_Immutable_next()

```
static __inline__ const Chain_Node* _Chain_Immutable_next (
            const Chain_Node * the_node )  [static]
```

Returns pointer to the immutable next node from this node.

This function returns a pointer to the next node after this node.

**Parameters**

| | |
|---|---|
| *the_node* | The node to be operated upon. |

**Returns**

This method returns the next node on the chain.

Definition at line 326 of file chainimpl.h.

#### 8.25.5.16 _Chain_Immutable_previous()

```
static __inline__ const Chain_Node* _Chain_Immutable_previous (
            const Chain_Node * the_node )  [static]
```

Returns pointer to the immutable previous node from this node.

This function returns a pointer to the previous node on this chain.

**Parameters**

| | |
|---|---|
| *the_node* | The node to be operated upon. |

**Returns**

This method returns the previous node on the chain.

Definition at line 358 of file chainimpl.h.

#### 8.25.5.17 _Chain_Immutable_tail()

```
static __inline__ const Chain_Node* _Chain_Immutable_tail (
            const Chain_Control * the_chain )  [static]
```

Returns pointer to immutable chain tail.

This function returns a pointer to the tail node on the chain.

**Parameters**

| | |
|---|---|
| *the_chain* | The chain to be operated upon. |

**Returns**

This method returns the permanent tail node of the chain.

Definition at line 226 of file chainimpl.h.

### 8.25.5.18 _Chain_Initialize()

```
void _Chain_Initialize (
             Chain_Control * the_chain,
             void * starting_address,
             size_t number_nodes,
             size_t node_size )
```

Initializes a chain header.

This routine initializes *the_chain* structure to manage the contiguous array of *number_nodes* nodes which starts at *starting_address*. Each node is of *node_size* bytes.

**Parameters**

| | | |
|---|---|---|
| out | *the_chain* | Specifies the chain to initialize. |
| | *starting_address* | The starting address of the array of elements. |
| | *number_nodes* | The number of nodes that will be in the chain. |
| | *node_size* | The size of each node. |

Definition at line 26 of file chain.c.

### 8.25.5.19 _Chain_Initialize_empty()

```
static __inline__ void _Chain_Initialize_empty (
             Chain_Control * the_chain )  [static]
```

Initializes this chain as empty.

This routine initializes the specified chain to contain zero nodes.

**Parameters**

| | | |
|---|---|---|
| out | *the_chain* | The chain to be initialized. |

Definition at line 488 of file chainimpl.h.

#### 8.25.5.20 _Chain_Initialize_node()

```
static __inline__ void _Chain_Initialize_node (
            Chain_Node * the_node )  [static]
```

Initializes a chain node.

In debug configurations, the node is set off chain. In all other configurations, this function does nothing.

**Parameters**

| | | |
|---|---|---|
| out | *the_node* | The chain node to initialize. |

Definition at line 122 of file chainimpl.h.

#### 8.25.5.21 _Chain_Initialize_one()

```
static __inline__ void _Chain_Initialize_one (
            Chain_Control * the_chain,
            Chain_Node * the_node )  [static]
```

Initializes this chain to contain exactly the specified node.

**Parameters**

| | | |
|---|---|---|
| out | *the_chain* | The chain to be initialized to contain exactly the specified node. |
| out | *the_node* | The one and only node of the chain to be initialized. |

Definition at line 511 of file chainimpl.h.

#### 8.25.5.22 _Chain_Insert_ordered_unprotected()

```
static __inline__ void _Chain_Insert_ordered_unprotected (
            Chain_Control * the_chain,
            Chain_Node * to_insert,
            const void * left,
            Chain_Node_order order )  [static]
```

Inserts a node into the chain according to the order relation.

After the operation the chain contains the node to insert and the order relation holds for all nodes from the head up to the inserted node. Nodes after the inserted node are not moved.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_chain* | The chain to be operated upon. |
| out | *to_insert* | The node to insert. |
| | *left* | The left hand side passed to the order relation. It must correspond to the node to insert. The separate left hand side parameter may help the compiler to generate better code if it is stored in a local variable. |
| | *order* | The order relation. |

Definition at line 847 of file chainimpl.h.

#### 8.25.5.23 _Chain_Insert_unprotected()

```
static __inline__ void _Chain_Insert_unprotected (
             Chain_Node * after_node,
             Chain_Node * the_node )  [static]
```

Inserts a node (unprotected).

This routine inserts the_node on a chain immediately following after_node.

**Parameters**

| | | |
|---|---|---|
| in,out | *after_node* | The node which will precede *the_node* on the chain. |
| out | *the_node* | The node to be inserted after *after_node*. |

**Note**

It does NOT disable interrupts to ensure the atomicity of the extract operation.

Definition at line 636 of file chainimpl.h.

#### 8.25.5.24 _Chain_Is_empty()

```
static __inline__ bool _Chain_Is_empty (
             const Chain_Control * the_chain )  [static]
```

Checks if the chain is empty.

This function returns true if there are no nodes on *the_chain* and false otherwise.

**Parameters**

| | |
|---|---|
| *the_chain* | The chain to check if it is empty. |

**Return values**

| | |
|---|---|
| *true* | There are no nodes on *the_chain*. |
| *false* | There are nodes on *the_chain*. |

Definition at line 376 of file chainimpl.h.

#### 8.25.5.25 _Chain_Is_first()

```
static __inline__ bool _Chain_Is_first (
             const Chain_Node * the_node )  [static]
```

Checks if this is the first node on the chain.

This function returns true if the_node is the first node on a chain and false otherwise.

**Parameters**

| | |
|---|---|
| *the_node* | The node of which the caller wants to know if it is the first node on a chain. |

**Return values**

| | |
|---|---|
| *true* | *the_node* is the first node on a chain. |
| *false* | *the_node* is not the first node on a chain. |

Definition at line 396 of file chainimpl.h.

#### 8.25.5.26 _Chain_Is_head()

```
static __inline__ bool _Chain_Is_head (
             const Chain_Control * the_chain,
             const Chain_Node * the_node )  [static]
```

Checks if this node is the chain head.

This function returns true if *the_node* is the head of *the_chain* and false otherwise.

**Parameters**

| | |
|---|---|
| *the_chain* | The chain to be operated upon. |
| *the_node* | The node to check for being the Chain Head. |

**Return values**

| | |
|---|---|
| *true* | *the_node* is the head of *the_chain*. |
| *false* | *the_node* is not the head of *the_chain*. |

Definition at line 453 of file chainimpl.h.

#### 8.25.5.27 _Chain_Is_last()

```
static __inline__ bool _Chain_Is_last (
             const Chain_Node * the_node )  [static]
```

Checks if this is the last node on the chain.

This function returns true if *the_node* is the last node on a chain and false otherwise.

**Parameters**

| | |
|---|---|
| *the_node* | The node of which the caller wants to know if it is the last node on a chain. |

**Return values**

| | |
|---|---|
| *true* | *the_node* is the last node on a chain. |
| *false* | *the_node* is not the last node on a chain. |

Definition at line 415 of file chainimpl.h.

#### 8.25.5.28 _Chain_Is_node_off_chain()

```
static __inline__ bool _Chain_Is_node_off_chain (
            const Chain_Node * node )  [static]
```

Checks if the node is off chain.

This function returns true if the *node* is not on a chain. A *node* is off chain if the next field is set to NULL.

**Parameters**

| | |
|---|---|
| *node* | The node to check if it is off chain. |

**Return values**

| | |
|---|---|
| *true* | The *node* is off chain. |
| *false* | The *node* is not off chain. |

Definition at line 142 of file chainimpl.h.

#### 8.25.5.29 _Chain_Is_tail()

```
static __inline__ bool _Chain_Is_tail (
            const Chain_Control * the_chain,
            const Chain_Node * the_node )  [static]
```

Checks if this node is the chain tail.

This function returns true if *the_node* is the tail of *the_chain* and false otherwise.

**Parameters**

| | |
|---|---|
| *the_chain* | The chain to be operated upon. |
| *the_node* | The node to check for being the Chain Tail. |

**Return values**

| | |
|---|---|
| *true* | *the_node* is the tail of *the_chain*. |
| *false* | *the_node* is not the tail of *the_chain*. |

Definition at line 473 of file chainimpl.h.

#### 8.25.5.30 _Chain_Iterator_destroy()

```
static __inline__ void _Chain_Iterator_destroy (
            Chain_Iterator * the_iterator )  [static]
```

Destroys the iterator.

Removes the iterator from its registry.

**Parameters**

| | | |
|---|---|---|
| out | *the_iterator* | The chain iterator. |

Definition at line 1104 of file chainimpl.h.

#### 8.25.5.31 _Chain_Iterator_initialize()

```
static __inline__ void _Chain_Iterator_initialize (
            Chain_Control * the_chain,
            Chain_Iterator_registry * the_registry,
            Chain_Iterator * the_iterator,
            Chain_Iterator_direction direction )  [static]
```

Initializes the chain iterator.

In the following example nodes inserted during the iteration are visited in case they are inserted after the current position in iteration order.

```
#include <rtems/score/chainimpl.h>
#include <rtems/score/isrlock.h>
typedef struct {
  Chain_Control           Chain;
  Chain_Iterator_registry Iterators;
  ISR_LOCK_MEMBER(        Lock )
} Some_Control;
void iterate(
  Some_Control   *the_some,
  void         ( *visitor )( Chain_Node * )
)
```

```
{
  ISR_lock_Context  lock_context;
  Chain_Iterator    iter;
  Chain_Node       *node;
  const Chain_Node *end;
  end = _Chain_Immutable_tail( &the_some->Chain );
  _ISR_lock_ISR_disable_and_acquire( &the_some->Lock, &lock_context );
  _Chain_Iterator_initialize(
    &the_some->Chain,
    &the_some->Iterators,
    &iter,
    CHAIN_ITERATOR_FORWARD
  );
  while ( ( node = _Chain_Iterator_next( &iter ) ) != end ) {
    _Chain_Iterator_set_position( &iter, node );
    _ISR_lock_Release_and_ISR_enable( &the_some->Lock, &lock_context );
    ( *visitor )( node );
    _ISR_lock_ISR_disable_and_acquire( &the_some->Lock, &lock_context );
  }
  _Chain_Iterator_destroy( &iter );
  _ISR_lock_Release_and_ISR_enable( &the_some->Lock, &lock_context );
}
```

**Parameters**

| | | |
|---|---|---|
| out | *the_chain* | The chain to iterate. |
| in,out | *the_registry* | The registry for the chain iterator. |
| out | *the_iterator* | The chain iterator to initialize. |
| out | *direction* | The iteration direction. |

**See also**

_Chain_Iterator_next(), _Chain_Iterator_set_position() and Chain_Iterator_destroy().

**Warning**

Think twice before you use a chain iterator. Its current implementation is unfit for use in performance relevant components, due to the linear time complexity in _Chain_Iterator_registry_update().

Definition at line 1040 of file chainimpl.h.

### 8.25.5.32 _Chain_Iterator_next()

```
static __inline__ Chain_Node* _Chain_Iterator_next (
            const Chain_Iterator * the_iterator )  [static]
```

Returns the next node in the iterator direction.

In case a next node exists, then the iterator should be updated via _Chain_Iterator_set_position() to continue with the next iteration step.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_iterator* | The chain iterator. |

**Returns**

The next node in the iterator direction

Definition at line 1072 of file chainimpl.h.

#### 8.25.5.33 _Chain_Iterator_registry_initialize()

```
static __inline__ void _Chain_Iterator_registry_initialize (
            Chain_Iterator_registry * the_registry ) [static]
```

Initializes a chain iterator registry.

**Parameters**

| | | |
|---|---|---|
| out | *the_registry* | The chain iterator registry to be initialized. |

Definition at line 934 of file chainimpl.h.

#### 8.25.5.34 _Chain_Iterator_registry_update()

```
static __inline__ void _Chain_Iterator_registry_update (
            Chain_Iterator_registry * the_registry,
            Chain_Node * the_node_to_extract ) [static]
```

Updates all iterators present in the chain iterator registry in case of a node extraction.

Must be called before _Chain_Extract_unprotected().

**Warning**

This function will look at all registered chain iterators to determine if an update is necessary.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_registry* | the chain iterator registry. |
| out | *the_node_to_extract* | The node that will be extracted. |

Definition at line 953 of file chainimpl.h.

#### 8.25.5.35 _Chain_Iterator_set_position()

```
static __inline__ void _Chain_Iterator_set_position (
            Chain_Iterator * the_iterator,
            Chain_Node * the_node ) [static]
```

Sets the iterator position.

**Parameters**

| | | |
|---|---|---|
| out | *the_iterator* | The chain iterator. |
| out | *the_node* | The new iterator position. |

Definition at line 1089 of file chainimpl.h.

#### 8.25.5.36 _Chain_Last()

```
static __inline__ Chain_Node* _Chain_Last (
            const Chain_Control * the_chain )  [static]
```

Returns pointer to chain's last node.

This function returns a pointer to the last node on the chain just before the tail.

**Parameters**

| | |
|---|---|
| *the_chain* | The chain to be operated upon. |

**Returns**

This method returns the last node of the chain.

Definition at line 277 of file chainimpl.h.

#### 8.25.5.37 _Chain_Next()

```
static __inline__ Chain_Node* _Chain_Next (
            const Chain_Node * the_node )  [static]
```

Returns pointer to the next node from this node.

This function returns a pointer to the next node after this node.

**Parameters**

| | |
|---|---|
| *the_node* | The node to be operated upon. |

**Returns**

This method returns the next node on the chain.

Definition at line 310 of file chainimpl.h.

#### 8.25.5.38 _Chain_Node_count_unprotected()

```
size_t _Chain_Node_count_unprotected (
            const Chain_Control * chain )
```

Returns the node count of the chain.

**Parameters**

| | |
|---|---|
| *chain* | The chain to return the node count from. |

**Note**

It does NOT disable interrupts to ensure the atomicity of the operation.

**Returns**

The node count of the chain.

Definition at line 21 of file chainnodecount.c.

#### 8.25.5.39 _Chain_Prepend_unprotected()

```
static __inline__ void _Chain_Prepend_unprotected (
            Chain_Control * the_chain,
            Chain_Node * the_node )  [static]
```

Prepends a node (unprotected).

This routine prepends the_node onto the front of the_chain.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_chain* | The chain to be operated upon. |
| in,out | *the_node* | The node to be prepended to the front of *the_chain*. |

**Note**

It does NOT disable interrupts to ensure the atomicity of the prepend operation.

Definition at line 715 of file chainimpl.h.

#### 8.25.5.40 _Chain_Prepend_with_empty_check_unprotected()

```
static __inline__ bool _Chain_Prepend_with_empty_check_unprotected (
            Chain_Control * the_chain,
            Chain_Node * the_node )  [static]
```

Prepends a node and checks if the chain was empty before (unprotected).

This routine prepends the_node onto the front of the_chain.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_chain* | The chain to be operated upon. |
| out | *the_node* | The node to be prepended to the front of *the_chain*. |

**Note**

It does NOT disable interrupts to ensure the atomicity of the prepend operation.

**Return values**

| | |
|---|---|
| *true* | The chain was empty before. |
| *false* | The chain contained at least one node before. |

Definition at line 763 of file chainimpl.h.

#### 8.25.5.41 _Chain_Previous()

```
static __inline__ Chain_Node* _Chain_Previous (
            const Chain_Node * the_node )  [static]
```

Returns pointer to the previous node from this node.

This function returns a pointer to the previous node on this chain.

**Parameters**

| | |
|---|---|
| *the_node* | The node to be operated upon. |

**Returns**

This method returns the previous node on the chain.

Definition at line 342 of file chainimpl.h.

#### 8.25.5.42 _Chain_Set_off_chain()

```
static __inline__ void _Chain_Set_off_chain (
            Chain_Node * node )  [static]
```

Sets off chain.

This function sets the next field of the *node* to NULL indicating the *node* is not part of a chain.

**Parameters**

| | | |
|---|---|---|
| out | *node* | The node to set off chain. |

Definition at line 104 of file chainimpl.h.

#### 8.25.5.43 _Chain_Tail()

```
static __inline__ Chain_Node* _Chain_Tail (
            Chain_Control * the_chain )  [static]
```

Returns pointer to chain tail.

This function returns a pointer to the tail node on the chain.

**Parameters**

| | | |
|---|---|---|
| in | *the_chain* | The chain to be operated upon. |

**Returns**

This method returns the permanent tail node of the chain.

Definition at line 210 of file chainimpl.h.

## 8.26 Chains

Chain API.

### Macros

- #define RTEMS_CHAIN_INITIALIZER_EMPTY(name) CHAIN_INITIALIZER_EMPTY( name )

  *Chain initializer for an empty chain with designator name.*
- #define RTEMS_CHAIN_INITIALIZER_ONE_NODE(node) CHAIN_INITIALIZER_ONE_NODE( node )

  *Chain initializer for a chain with one node.*
- #define RTEMS_CHAIN_NODE_INITIALIZER_ONE_NODE_CHAIN(chain) CHAIN_NODE_INITIALIZER_ONE_NODE_CHAI chain )

  *Chain node initializer for a chain containing exactly this node.*
- #define RTEMS_CHAIN_DEFINE_EMPTY(name) rtems_chain_control name = RTEMS_CHAIN_INITIALIZER_EMPTY( name )

  *Chain definition for an empty chain with designator name.*

### Typedefs

- typedef Chain_Node **rtems_chain_node**
- typedef Chain_Control **rtems_chain_control**

### Functions

- rtems_status_code rtems_chain_append_with_notification (rtems_chain_control *chain, rtems_chain_node *node, rtems_id task, rtems_event_set events)

  *Appends the node to the chain and sends the events to the task if the chain was empty before the append.*
- rtems_status_code rtems_chain_prepend_with_notification (rtems_chain_control *chain, rtems_chain_node *node, rtems_id task, rtems_event_set events)

  *Prepends the node to the chain and sends the events to the task if the chain was empty before the prepend.*
- rtems_status_code rtems_chain_get_with_notification (rtems_chain_control *chain, rtems_id task, rtems_event_set events, rtems_chain_node **node)

  *Gets the first node of the chain and sends the events to the task if the chain is empty after the get.*
- rtems_status_code rtems_chain_get_with_wait (rtems_chain_control *chain, rtems_event_set events, rtems_interval timeout, rtems_chain_node **node)

  *Gets the first node of the chain and sends the events to the task if the chain is empty afterwards.*
- static __inline__ void rtems_chain_initialize (rtems_chain_control *the_chain, void *starting_address, size_t number_nodes, size_t node_size)

  *Initialize a chain Header.*
- static __inline__ void rtems_chain_initialize_empty (rtems_chain_control *the_chain)

  *Initialize this chain as empty.*
- static __inline__ void rtems_chain_set_off_chain (rtems_chain_node *node)

  *Set off chain.*
- static __inline__ void rtems_chain_initialize_node (rtems_chain_node *node)

  *Initializes a chain node.*
- static __inline__ bool rtems_chain_is_node_off_chain (const rtems_chain_node *node)

  *Is the node off chain.*
- static __inline__ bool rtems_chain_is_null_node (const rtems_chain_node *the_node)

  *Is the chain node pointer NULL.*

- static __inline__ rtems_chain_node * rtems_chain_head (rtems_chain_control *the_chain)

  *Return pointer to Chain Head.*
- static __inline__ const rtems_chain_node * rtems_chain_immutable_head (const rtems_chain_control *the_chain)

  *Return pointer to immutable Chain Head.*
- static __inline__ rtems_chain_node * rtems_chain_tail (rtems_chain_control *the_chain)

  *Return pointer to Chain Tail.*
- static __inline__ const rtems_chain_node * rtems_chain_immutable_tail (const rtems_chain_control *the_chain)

  *Return pointer to immutable Chain Tail.*
- static __inline__ rtems_chain_node * rtems_chain_first (const rtems_chain_control *the_chain)

  *Return pointer to Chain's First node after the permanent head.*
- static __inline__ const rtems_chain_node * rtems_chain_immutable_first (const rtems_chain_control *the_chain)

  *Return pointer to immutable Chain's First node.*
- static __inline__ rtems_chain_node * rtems_chain_last (const rtems_chain_control *the_chain)

  *Return pointer to Chain's Last node before the permanent tail.*
- static __inline__ const rtems_chain_node * rtems_chain_immutable_last (const rtems_chain_control *the_chain)

  *Return pointer to immutable Chain's Last node.*
- static __inline__ rtems_chain_node * rtems_chain_next (const rtems_chain_node *the_node)

  *Return pointer the next node from this node.*
- static __inline__ const rtems_chain_node * rtems_chain_immutable_next (const rtems_chain_node *the_node)

  *Return pointer the immutable next node from this node.*
- static __inline__ rtems_chain_node * rtems_chain_previous (const rtems_chain_node *the_node)

  *Return pointer the previous node from this node.*
- static __inline__ const rtems_chain_node * rtems_chain_immutable_previous (const rtems_chain_node *the_node)

  *Return pointer the immutable previous node from this node.*
- static __inline__ bool rtems_chain_are_nodes_equal (const rtems_chain_node *left, const rtems_chain_node *right)

  *Are Two nodes equal.*
- static __inline__ bool rtems_chain_is_empty (const rtems_chain_control *the_chain)

  *Is the chain empty.*
- static __inline__ bool rtems_chain_is_first (const rtems_chain_node *the_node)

  *Is this the first node on the chain.*
- static __inline__ bool rtems_chain_is_last (const rtems_chain_node *the_node)

  *Is this the last node on the chain.*
- static __inline__ bool rtems_chain_has_only_one_node (const rtems_chain_control *the_chain)

  *Does this chain have only one node.*
- static __inline__ bool rtems_chain_is_head (const rtems_chain_control *the_chain, const rtems_chain_node *the_node)

  *Is this node the chain head.*
- static __inline__ bool rtems_chain_is_tail (const rtems_chain_control *the_chain, const rtems_chain_node *the_node)

  *Is this node the chain tail.*
- void rtems_chain_extract (rtems_chain_node *the_node)

  *Extract the specified node from a chain.*
- static __inline__ void rtems_chain_extract_unprotected (rtems_chain_node *the_node)

  *Extract the specified node from a chain (unprotected).*
- rtems_chain_node * rtems_chain_get (rtems_chain_control *the_chain)

*Obtain the first node on a chain.*

- static __inline__ rtems_chain_node * rtems_chain_get_unprotected (rtems_chain_control *the_chain)

  *See _Chain_Get_unprotected().*
- static __inline__ rtems_chain_node * rtems_chain_get_first_unprotected (rtems_chain_control *the_chain)

  *See _Chain_Get_first_unprotected().*
- void rtems_chain_insert (rtems_chain_node *after_node, rtems_chain_node *the_node)

  *Insert a node on a chain.*
- static __inline__ void rtems_chain_insert_unprotected (rtems_chain_node *after_node, rtems_chain_node *the_node)

  *See _Chain_Insert_unprotected().*
- void rtems_chain_append (rtems_chain_control *the_chain, rtems_chain_node *the_node)

  *Append a node on the end of a chain.*
- static __inline__ void rtems_chain_append_unprotected (rtems_chain_control *the_chain, rtems_chain_node *the_node)

  *Append a node on the end of a chain (unprotected).*
- void rtems_chain_prepend (rtems_chain_control *the_chain, rtems_chain_node *the_node)

  *Prepend a node.*
- static __inline__ void rtems_chain_prepend_unprotected (rtems_chain_control *the_chain, rtems_chain_node *the_node)

  *Prepend a node (unprotected).*
- bool rtems_chain_append_with_empty_check (rtems_chain_control *chain, rtems_chain_node *node)

  *Checks if the chain is empty and appends the node.*
- bool rtems_chain_prepend_with_empty_check (rtems_chain_control *chain, rtems_chain_node *node)

  *Checks if the chain is empty and prepends the node.*
- bool rtems_chain_get_with_empty_check (rtems_chain_control *chain, rtems_chain_node **node)

  *Tries to get the first node and check if the chain is empty afterwards.*
- static __inline__ size_t rtems_chain_node_count_unprotected (const rtems_chain_control *chain)

  *Returns the node count of the chain.*

## 8.26.1 Detailed Description

Chain API.

## 8.26.2 Macro Definition Documentation

### 8.26.2.1 RTEMS_CHAIN_INITIALIZER_ONE_NODE

```
#define RTEMS_CHAIN_INITIALIZER_ONE_NODE(
              node )  CHAIN_INITIALIZER_ONE_NODE( node )
```

Chain initializer for a chain with one *node*.

**See also**

RTEMS_CHAIN_NODE_INITIALIZER_ONE_NODE_CHAIN().

Definition at line 52 of file chain.h.

#### 8.26.2.2 RTEMS_CHAIN_NODE_INITIALIZER_ONE_NODE_CHAIN

```
#define RTEMS_CHAIN_NODE_INITIALIZER_ONE_NODE_CHAIN(
              chain )  CHAIN_NODE_INITIALIZER_ONE_NODE_CHAIN( chain )
```

Chain node initializer for a *chain* containing exactly this node.

**See also**

RTEMS_CHAIN_INITIALIZER_ONE_NODE().

Definition at line 60 of file chain.h.

### 8.26.3 Function Documentation

#### 8.26.3.1 rtems_chain_append()

```
void rtems_chain_append (
             rtems_chain_control * the_chain,
             rtems_chain_node * the_node )
```

Append a node on the end of a chain.

This routine appends *the_node* onto the end of *the_chain*.

NOTE: It disables interrupts to ensure the atomicity of the append operation.

#### 8.26.3.2 rtems_chain_append_unprotected()

```
static __inline__ void rtems_chain_append_unprotected (
             rtems_chain_control * the_chain,
             rtems_chain_node * the_node )  [static]
```

Append a node on the end of a chain (unprotected).

This routine appends *the_node* onto the end of *the_chain*.

NOTE: It does NOT disable interrupts to ensure the atomicity of the append operation.

Definition at line 679 of file chain.h.

#### 8.26.3.3 rtems_chain_append_with_empty_check()

```
bool rtems_chain_append_with_empty_check (
             rtems_chain_control * chain,
             rtems_chain_node * node )
```

Checks if the *chain* is empty and appends the *node*.

Interrupts are disabled to ensure the atomicity of the operation.

**Return values**

| | |
|---|---|
| *true* | The chain was empty before the append. |
| *false* | The chain contained at least one node before the append. |

#### 8.26.3.4 rtems_chain_append_with_notification()

```
rtems_status_code rtems_chain_append_with_notification (
            rtems_chain_control * chain,
            rtems_chain_node * node,
            rtems_id task,
            rtems_event_set events )
```

Appends the *node* to the *chain* and sends the *events* to the *task* if the *chain* was empty before the append.

**See also**

rtems_chain_append_with_empty_check() and rtems_event_send().

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation. |
| *RTEMS_INVALID_ID* | No such task. |

#### 8.26.3.5 rtems_chain_are_nodes_equal()

```
static __inline__ bool rtems_chain_are_nodes_equal (
            const rtems_chain_node * left,
            const rtems_chain_node * right )  [static]
```

Are Two nodes equal.

This function returns true if *left* and *right* are equal, and false otherwise.

**Parameters**

| | | |
|---|---|---|
| in | *left* | is the node on the left hand side of the comparison. |
| in | *right* | is the node on the left hand side of the comparison. |

**Return values**

| | |
|---|---|
| *true* | *left* is equal to *right*. |
| *false* | *left* is not equal to *right* |

Definition at line 448 of file chain.h.

#### 8.26.3.6 rtems_chain_extract()

```
void rtems_chain_extract (
             rtems_chain_node * the_node )
```

Extract the specified node from a chain.

This routine extracts *the_node* from the chain on which it resides. It disables interrupts to ensure the atomicity of the extract operation.

- the_node specifies the node to extract

#### 8.26.3.7 rtems_chain_extract_unprotected()

```
static __inline__ void rtems_chain_extract_unprotected (
             rtems_chain_node * the_node )  [static]
```

Extract the specified node from a chain (unprotected).

This routine extracts *the_node* from the chain on which it resides.

NOTE: It does NOT disable interrupts to ensure the atomicity of the append operation.

Definition at line 590 of file chain.h.

#### 8.26.3.8 rtems_chain_first()

```
static __inline__ rtems_chain_node* rtems_chain_first (
             const rtems_chain_control * the_chain )  [static]
```

Return pointer to Chain's First node after the permanent head.

This function returns a pointer to the first node on the chain after the head.

**Parameters**

| | | |
|---|---|---|
| in | *the_chain* | is the chain to be operated upon. |

**Returns**

This method returns the first node of the chain.

Definition at line 314 of file chain.h.

#### 8.26.3.9 rtems_chain_get()

```
rtems_chain_node* rtems_chain_get (
            rtems_chain_control * the_chain )
```

Obtain the first node on a chain.

This function removes the first node from *the_chain* and returns a pointer to that node. If *the_chain* is empty, then NULL is returned.

**Returns**

This method returns a pointer a node. If a node was removed, then a pointer to that node is returned. If *the_chain* was empty, then NULL is returned.

NOTE: It disables interrupts to ensure the atomicity of the get operation.

#### 8.26.3.10 rtems_chain_get_with_empty_check()

```
bool rtems_chain_get_with_empty_check (
            rtems_chain_control * chain,
            rtems_chain_node ** node )
```

Tries to get the first *node* and check if the *chain* is empty afterwards.

This function removes the first node from the *chain* and returns a pointer to that node in *node*. If the *chain* is empty, then NULL is returned.

Interrupts are disabled to ensure the atomicity of the operation.

**Return values**

| | |
|---|---|
| *true* | The chain is empty after the node removal. |
| *false* | The chain contained at least one node after the node removal. |

#### 8.26.3.11 rtems_chain_get_with_notification()

```
rtems_status_code rtems_chain_get_with_notification (
            rtems_chain_control * chain,
            rtems_id task,
            rtems_event_set events,
            rtems_chain_node ** node )
```

Gets the first *node* of the *chain* and sends the *events* to the *task* if the *chain* is empty after the get.

**See also**

rtems_chain_get_with_empty_check() and rtems_event_send().

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation. |
| *RTEMS_INVALID_ID* | No such task. |

#### 8.26.3.12 rtems_chain_get_with_wait()

```
rtems_status_code rtems_chain_get_with_wait (
            rtems_chain_control * chain,
            rtems_event_set events,
            rtems_interval timeout,
            rtems_chain_node ** node )
```

Gets the first *node* of the *chain* and sends the *events* to the *task* if the *chain* is empty afterwards.

**See also**

rtems_chain_get() and rtems_event_receive().

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation. |
| *RTEMS_TIMEOUT* | Timeout. |

#### 8.26.3.13 rtems_chain_has_only_one_node()

```
static __inline__ bool rtems_chain_has_only_one_node (
            const rtems_chain_control * the_chain )  [static]
```

Does this chain have only one node.

This function returns true if there is only one node on *the_chain* and false otherwise.

**Parameters**

| | | |
|---|---|---|
| in | *the_chain* | is the chain to be operated upon. |

**Return values**

| | |
|---|---|
| *true* | The chain has only one node. |
| *false* | The chain has more than one nodes. |

Definition at line 522 of file chain.h.

#### 8.26.3.14 rtems_chain_head()

```
static __inline__ rtems_chain_node* rtems_chain_head (
            rtems_chain_control * the_chain )  [static]
```

Return pointer to Chain Head.

This function returns a pointer to the first node on the chain.

**Parameters**

| | | |
|---|---|---|
| in | *the_chain* | is the chain to be operated upon. |

**Returns**

This method returns the permanent node of the chain.

Definition at line 249 of file chain.h.

#### 8.26.3.15 rtems_chain_immutable_first()

```
static __inline__ const rtems_chain_node* rtems_chain_immutable_first (
            const rtems_chain_control * the_chain )  [static]
```

Return pointer to immutable Chain's First node.

This function returns a pointer to the first node on the chain after the head.

**Parameters**

| | | |
|---|---|---|
| in | *the_chain* | is the chain to be operated upon. |

**Returns**

This method returns the first node of the chain.

Definition at line 331 of file chain.h.

#### 8.26.3.16 rtems_chain_immutable_head()

```
static __inline__ const rtems_chain_node* rtems_chain_immutable_head (
            const rtems_chain_control * the_chain )  [static]
```

Return pointer to immutable Chain Head.

This function returns a pointer to the head node on the chain.

**Parameters**

| | | |
|---|---|---|
| in | *the_chain* | is the chain to be operated upon. |

**Returns**

This method returns the permanent head node of the chain.

Definition at line 265 of file chain.h.

#### 8.26.3.17 rtems_chain_immutable_last()

```
static __inline__ const rtems_chain_node* rtems_chain_immutable_last (
            const rtems_chain_control * the_chain )  [static]
```

Return pointer to immutable Chain's Last node.

This function returns a pointer to the last node on the chain just before the tail.

**Parameters**

| | | |
|---|---|---|
| in | *the_chain* | is the chain to be operated upon. |

**Returns**

This method returns the last node of the chain.

Definition at line 365 of file chain.h.

#### 8.26.3.18 rtems_chain_immutable_next()

```
static __inline__ const rtems_chain_node* rtems_chain_immutable_next (
            const rtems_chain_node * the_node )  [static]
```

Return pointer the immutable next node from this node.

This function returns a pointer to the next node after this node.

**Parameters**

| | | |
|---|---|---|
| in | *the_node* | is the node to be operated upon. |

**Returns**

This method returns the next node on the chain.

Definition at line 397 of file chain.h.

#### 8.26.3.19 rtems_chain_immutable_previous()

```
static __inline__ const rtems_chain_node* rtems_chain_immutable_previous (
            const rtems_chain_node * the_node )  [static]
```

Return pointer the immutable previous node from this node.

This function returns a pointer to the previous node on this chain.

**Parameters**

| | | |
|---|---|---|
| in | *the_node* | is the node to be operated upon. |

**Returns**

This method returns the previous node on the chain.

Definition at line 429 of file chain.h.

#### 8.26.3.20 rtems_chain_immutable_tail()

```
static __inline__ const rtems_chain_node* rtems_chain_immutable_tail (
            const rtems_chain_control * the_chain )  [static]
```

Return pointer to immutable Chain Tail.

This function returns a pointer to the tail node on the chain.

**Parameters**

| | | |
|---|---|---|
| in | *the_chain* | is the chain to be operated upon. |

**Returns**

This method returns the permanent tail node of the chain.

Definition at line 297 of file chain.h.

#### 8.26.3.21 rtems_chain_initialize()

```
static __inline__ void rtems_chain_initialize (
             rtems_chain_control * the_chain,
             void * starting_address,
             size_t number_nodes,
             size_t node_size )  [static]
```

Initialize a chain Header.

This routine initializes *the_chain* structure to manage the contiguous array of *number_nodes* nodes which starts at *starting_address*. Each node is of *node_size* bytes.

**Parameters**

| | | |
|---|---|---|
| in | *the_chain* | specifies the chain to initialize |
| in | *starting_address* | is the starting address of the array of elements |
| in | *number_nodes* | is the number of nodes that will be in the chain |
| in | *node_size* | is the size of each node |

Definition at line 146 of file chain.h.

#### 8.26.3.22 rtems_chain_initialize_empty()

```
static __inline__ void rtems_chain_initialize_empty (
             rtems_chain_control * the_chain )  [static]
```

Initialize this chain as empty.

This routine initializes the specified chain to contain zero nodes.

**Parameters**

| | | |
|---|---|---|
| in | *the_chain* | is the chain to be initialized. |

Definition at line 168 of file chain.h.

#### 8.26.3.23 rtems_chain_initialize_node()

```
static __inline__ void rtems_chain_initialize_node (
             rtems_chain_node * node )  [static]
```

Initializes a chain node.

In debug configurations, the node is set off chain. In all other configurations, this function does nothing.

**Parameters**

| | | |
|---|---|---|
| in | *the_node* | The chain node to initialize. |

Definition at line 198 of file chain.h.

#### 8.26.3.24 rtems_chain_insert()

```
void rtems_chain_insert (
            rtems_chain_node * after_node,
            rtems_chain_node * the_node )
```

Insert a node on a chain.

This routine inserts *the_node* on a chain immediately following *after_node*.

NOTE: It disables interrupts to ensure the atomicity of the extract operation.

#### 8.26.3.25 rtems_chain_is_empty()

```
static __inline__ bool rtems_chain_is_empty (
            const rtems_chain_control * the_chain )  [static]
```

Is the chain empty.

This function returns true if there a no nodes on *the_chain* and false otherwise.

**Parameters**

| | | |
|---|---|---|
| in | *the_chain* | is the chain to be operated upon. |

**Return values**

| | |
|---|---|
| *true* | The chain is empty. |
| *false* | The chain is not empty. |

Definition at line 467 of file chain.h.

#### 8.26.3.26 rtems_chain_is_first()

```
static __inline__ bool rtems_chain_is_first (
            const rtems_chain_node * the_node )  [static]
```

Is this the first node on the chain.

This function returns true if the_node is the first node on a chain and false otherwise.

**Parameters**

| | | |
|---|---|---|
| in | *the_node* | is the node the caller wants to know if it is the first node on a chain. |

**Return values**

| | |
|---|---|
| *true* | *the_node* is the first node on a chain. |
| *false* | *the_node* is not the first node on a chain. |

Definition at line 486 of file chain.h.

#### 8.26.3.27 rtems_chain_is_head()

```
static __inline__ bool rtems_chain_is_head (
            const rtems_chain_control * the_chain,
            const rtems_chain_node * the_node )  [static]
```

Is this node the chain head.

This function returns true if *the_node* is the head of the_chain and false otherwise.

**Parameters**

| | | |
|---|---|---|
| in | *the_chain* | is the chain to be operated upon. |
| in | *the_node* | is the node to check for being the Chain Head. |

**Return values**

| | |
|---|---|
| *true* | *the_node* is the head of *the_chain*. |
| *false* | *the_node* is not the head of *the_chain*. |

Definition at line 541 of file chain.h.

#### 8.26.3.28 rtems_chain_is_last()

```
static __inline__ bool rtems_chain_is_last (
            const rtems_chain_node * the_node )  [static]
```

Is this the last node on the chain.

This function returns true if *the_node* is the last node on a chain and false otherwise.

**Parameters**

| | | |
|---|---|---|
| in | *the_node* | is the node to check as the last node. |

**Return values**

| | |
|---|---|
| *true* | *the_node* is the last node on a chain. |
| *false* | *the_node* is not the last node on a chain |

Definition at line 504 of file chain.h.

#### 8.26.3.29 rtems_chain_is_node_off_chain()

```
static __inline__ bool rtems_chain_is_node_off_chain (
            const rtems_chain_node * node )  [static]
```

Is the node off chain.

This function returns true if the *node* is not on a chain. A *node* is off chain if the next and previous fields are set to NULL.

**Parameters**

| | | |
|---|---|---|
| in | *node* | is the node off chain. |

**Return values**

| | |
|---|---|
| *true* | The node is off chain. |
| *false* | The node is not off chain. |

Definition at line 216 of file chain.h.

#### 8.26.3.30 rtems_chain_is_null_node()

```
static __inline__ bool rtems_chain_is_null_node (
            const rtems_chain_node * the_node )  [static]
```

Is the chain node pointer NULL.

This function returns true if the_node is NULL and false otherwise.

**Parameters**

| | | |
|---|---|---|
| in | *the_node* | is the node pointer to check. |

**Return values**

| | |
|---|---|
| *true* | The chain node pointer is NULL. |
| *false* | The chain node pointer is not NULL. |

Definition at line 233 of file chain.h.

#### 8.26.3.31 rtems_chain_is_tail()

```
static __inline__ bool rtems_chain_is_tail (
            const rtems_chain_control * the_chain,
            const rtems_chain_node * the_node )  [static]
```

Is this node the chain tail.

This function returns true if the_node is the tail of the_chain and false otherwise.

**Parameters**

| | | |
|---|---|---|
| in | *the_chain* | is the chain to be operated upon. |
| in | *the_node* | is the node to check for being the Chain Tail. |

**Return values**

| | |
|---|---|
| *true* | *the_node* is the tail of *the_chain*. |
| *false* | *the_node* is not the tail of *the_chain*. |

Definition at line 561 of file chain.h.

#### 8.26.3.32 rtems_chain_last()

```
static __inline__ rtems_chain_node* rtems_chain_last (
            const rtems_chain_control * the_chain )  [static]
```

Return pointer to Chain's Last node before the permanent tail.

This function returns a pointer to the last node on the chain just before the tail.

**Parameters**

| | | |
|---|---|---|
| in | *the_chain* | is the chain to be operated upon. |

**Returns**

This method returns the last node of the chain.

Definition at line 348 of file chain.h.

#### 8.26.3.33 rtems_chain_next()

```
static __inline__ rtems_chain_node* rtems_chain_next (
            const rtems_chain_node * the_node )  [static]
```

Return pointer the next node from this node.

This function returns a pointer to the next node after this node.

**Parameters**

| | | |
|---|---|---|
| in | *the_node* | is the node to be operated upon. |

**Returns**

This method returns the next node on the chain.

Definition at line 381 of file chain.h.

#### 8.26.3.34 rtems_chain_node_count_unprotected()

```
static __inline__ size_t rtems_chain_node_count_unprotected (
            const rtems_chain_control * chain )  [static]
```

Returns the node count of the chain.

**Parameters**

| | | |
|---|---|---|
| in | *chain* | The chain. |

**Note**

It does NOT disable interrupts to ensure the atomicity of the operation.

**Returns**

The node count of the chain.

Definition at line 775 of file chain.h.

#### 8.26.3.35 rtems_chain_prepend()

```
void rtems_chain_prepend (
            rtems_chain_control * the_chain,
            rtems_chain_node * the_node )
```

Prepend a node.

This routine prepends the_node onto the front of the_chain.

**Parameters**

| | | |
|---|---|---|
| in | *the_chain* | is the chain to be operated upon. |
| in | *the_node* | is the node to be prepended. |

NOTE: It disables interrupts to ensure the atomicity of the prepend operation.

#### 8.26.3.36 rtems_chain_prepend_unprotected()

```
static __inline__ void rtems_chain_prepend_unprotected (
            rtems_chain_control * the_chain,
            rtems_chain_node * the_node )  [static]
```

Prepend a node (unprotected).

This routine prepends the_node onto the front of the_chain.

**Parameters**

| | | |
|---|---|---|
| in | *the_chain* | is the chain to be operated upon. |
| in | *the_node* | is the node to be prepended. |

NOTE: It does NOT disable interrupts to ensure the atomicity of the prepend operation.

Definition at line 714 of file chain.h.

#### 8.26.3.37 rtems_chain_prepend_with_empty_check()

```
bool rtems_chain_prepend_with_empty_check (
            rtems_chain_control * chain,
            rtems_chain_node * node )
```

Checks if the *chain* is empty and prepends the *node*.

Interrupts are disabled to ensure the atomicity of the operation.

**Return values**

| | |
|---|---|
| *true* | The chain was empty before the prepend. |
| *false* | The chain contained at least one node before the prepend. |

#### 8.26.3.38 rtems_chain_prepend_with_notification()

```
rtems_status_code rtems_chain_prepend_with_notification (
            rtems_chain_control * chain,
```

```
            rtems_chain_node * node,
            rtems_id task,
            rtems_event_set events )
```

Prepends the *node* to the *chain* and sends the *events* to the *task* if the *chain* was empty before the prepend.

**See also**

rtems_chain_prepend_with_empty_check() and rtems_event_send().

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation. |
| *RTEMS_INVALID_ID* | No such task. |

#### 8.26.3.39 rtems_chain_previous()

```
static __inline__ rtems_chain_node* rtems_chain_previous (
            const rtems_chain_node * the_node )  [static]
```

Return pointer the previous node from this node.

This function returns a pointer to the previous node on this chain.

**Parameters**

| | | |
|---|---|---|
| in | *the_node* | is the node to be operated upon. |

**Returns**

This method returns the previous node on the chain.

Definition at line 413 of file chain.h.

#### 8.26.3.40 rtems_chain_set_off_chain()

```
static __inline__ void rtems_chain_set_off_chain (
            rtems_chain_node * node )  [static]
```

Set off chain.

This function sets the next and previous fields of the *node* to NULL indicating the *node* is not part of a chain.

**Parameters**

| | | |
|---|---|---|
| in | *node* | the node set to off chain. |

Definition at line 183 of file chain.h.

#### 8.26.3.41 rtems_chain_tail()

```
static __inline__ rtems_chain_node* rtems_chain_tail (
            rtems_chain_control * the_chain )  [static]
```

Return pointer to Chain Tail.

This function returns a pointer to the tail node on the chain.

**Parameters**

| | | |
|---|---|---|
| in | *the_chain* | is the chain to be operated upon. |

**Returns**

This method returns the permanent tail node of the chain.

Definition at line 281 of file chain.h.

## 8.27 Character Checks

Checks for characters (char).

### Macros

- #define **T_eq_char**(a, e) T_flags_eq_char(a, e, 0)
- #define **T_assert_eq_char**(a, e) T_flags_eq_char(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_char**(a, e) T_flags_eq_char(a, e, T_CHECK_QUIET)
- #define **T_step_eq_char**(s, a, e) T_flags_eq_char(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_char**(s, a, e) T_flags_eq_char(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_char**(a, e) T_flags_ne_char(a, e, 0)
- #define **T_assert_ne_char**(a, e) T_flags_ne_char(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_char**(a, e) T_flags_ne_char(a, e, T_CHECK_QUIET)
- #define **T_step_ne_char**(s, a, e) T_flags_ne_char(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_char**(s, a, e) T_flags_ne_char(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.27.1 Detailed Description

Checks for characters (char).

## 8.28 Classic

Classic.

### Modules

- Classic ASR Implementation
- Classic Attributes Implementation
- Classic Barrier Implementation
- Classic Message Queue Implementation
- Classic Modes Implementation
- Classic Object Implementation
- Classic Options Implementation
- Classic Rate Monotonic Scheduler Implementation
- Classic Region Manager Implementation
- Classic Status Implementation
- Classic Tasks Manager Implementation
- Classic Timer Implementation
- Dual Ported Memory Manager Implementation
- Event Implementation
- Partition Manager Implementation
- Semaphore Manager Implementation
- Signals Implementation
- User Extensions Implementation

### 8.28.1 Detailed Description

Classic.

## 8.29 Classic

This group contains the Classic API managers.

### Modules

- Barrier Manager

  *The Barrier Manager provides a unique synchronization capability which can be used to have a set of tasks block and be unblocked as a set.*
- Basic Types

  *This group contains basic Classic API types.*
- Cache Manager

  *The Cache Manager provides functions to perform maintenance operations for data and instruction caches.*
- Chains

  *Chain API.*
- Clock Manager

  *The Clock Manager provides support for time of day and other time related capabilities.*
- Directive Attributes

  *This group contains the Classic API directive attributes.*
- Directive Options

  *This group contains the Classic API directive options.*
- Directive Status Codes

  *This group contains the Classic API directive status codes and support functions.*
- Dual-Ported Memory Manager

  *The Dual-Ported Memory Manager provides a mechanism for converting addresses between internal and external representations for multiple dual-ported memory areas (DPMA).*
- Event Manager

  *The Event Manager provides a high performance method of inter-task communication and synchronization.*
- Fatal Error Manager

  *The Fatal Error Manager processes all fatal or irrecoverable errors and other sources of system termination (for example after `exit()`). Fatal errors are identified by the fatal source and error code pair.*
- Free-Running Counter and Busy Wait Delay

  *Free-running counter and busy wait delay functions.*
- I/O Manager

  *The Input/Output (I/O) Manager provides a well-defined mechanism for accessing device drivers and a structured methodology for organizing device drivers.*
- Initialization and Shutdown

  *This group contains directives to initialize and shutdown the RTEMS executive.*
- Interrupt Manager

  *Any real-time executive must provide a mechanism for quick response to externally generated interrupts to satisfy the critical time constraints of the application. The Interrupt Manager provides this mechanism for RTEMS. This manager permits quick interrupt response times by providing the critical ability to alter task execution which allows a task to be preempted upon exit from an ISR.*
- Local Packages

  *Local packages.*
- Message Manager

  *The Message Manager provides communication and synchronization capabilities using RTEMS message queues.*
- Object Services

  *RTEMS provides a collection of services to assist in the management and usage of the objects created and utilized via other managers. These services assist in the manipulation of RTEMS objects independent of the API used to create them.*

- Partition Manager

  *The Partition Manager provides facilities to dynamically allocate memory in fixed-size units.*
- Rate-Monotonic Manager

  *The Rate-Monotonic Manager provides facilities to implement tasks which execute in a periodic fashion. Critically, it also gathers information about the execution of those periods and can provide important statistics to the user which can be used to analyze and tune the application.*
- Region Manager

  *The Region Manager provides facilities to dynamically allocate memory in variable sized units.*
- Semaphore Manager

  *The Semaphore Manager utilizes standard Dijkstra counting semaphores to provide synchronization and mutual exclusion capabilities.*
- Signal Manager

  *The Signal Manager provides the capabilities required for asynchronous communication.*
- Support Services

  *Items of this group should move to other groups.*
- Task Manager

  *The Task Manager provides a comprehensive set of directives to create, delete, and administer tasks.*
- Task Modes

  *This group contains the Classic API task modes.*
- Timecounter Support
- Timer Manager

  *The Timer Manager provides support for timer facilities.*
- User Extensions Manager

  *The User Extensions Manager allows the application developer to augment the executive by allowing them to supply extension routines which are invoked at critical system events.*
- Version

  *The Version API provides functions to return the version or parts of the version of RTEMS you are using.*

## Files

- file rtems.h

  *This header file defines the RTEMS Classic API.*

### 8.29.1 Detailed Description

This group contains the Classic API managers.

The facilities provided by RTEMS are built upon a foundation of very powerful concepts. These concepts must be understood before the application developer can efficiently utilize RTEMS. The purpose of this chapter is to familiarize one with these concepts.

### 8.29.2 Objects

RTEMS provides directives which can be used to dynamically create, delete, and manipulate a set of predefined object types. These types include tasks, message queues, semaphores, memory regions, memory partitions, timers, ports, and rate monotonic periods. The object-oriented nature of RTEMS encourages the creation of modular applications built upon re-usable "building block" routines.

All objects are created on the local node as required by the application and have an RTEMS assigned ID. All objects have a user-assigned name. Although a relationship exists between an object's name and its RTEMS assigned ID, the name and ID are not identical. Object names are completely arbitrary and selected by the user as a meaningful "tag" which may commonly reflect the object's use in the application. Conversely, object IDs are designed to facilitate efficient object manipulation by the executive.

#### 8.29.2.1 Object Names

An object name is an unsigned 32-bit entity associated with the object by the user. The data type rtems_name is used to store object names.

Although not required by RTEMS, object names are often composed of four ASCII characters which help identify that object. For example, a task which causes a light to blink might be called "LITE". The rtems_build_name() routine is provided to build an object name from four ASCII characters. The following example illustrates this:

```
rtems_name my_name = rtems_build_name('L', 'I', 'T', 'E');
```

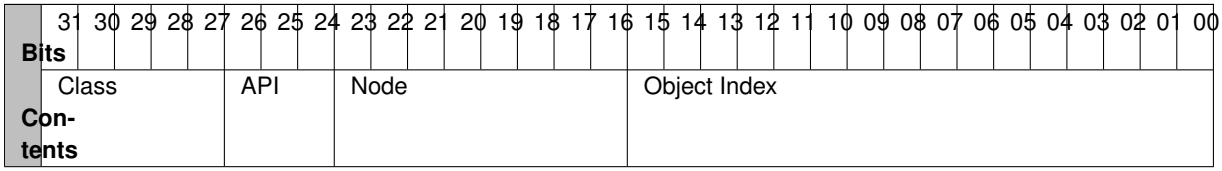
However, it is not required that the application use ASCII characters to build object names. For example, if an application requires one-hundred tasks, it would be difficult to assign meaningful ASCII names to each task. A more convenient approach would be to name them the binary values one through one-hundred, respectively.

RTEMS provides a helper routine, rtems_object_get_name(), which can be used to obtain the name of any RTEMS object using just its ID. This routine attempts to convert the name into a printable string.

#### 8.29.2.2 Object Identifiers

An object ID is a unique unsigned integer value which uniquely identifies an object instance. Object IDs are passed as arguments to many directives in RTEMS and RTEMS translates the ID to an internal object pointer. The efficient manipulation of object IDs is critical to the performance of RTEMS services. Because of this, there are two object ID formats defined. Each target architecture specifies which format it will use. There is a 32-bit format which is used for most of the supported architectures and supports multiprocessor configurations. There is also a simpler 16-bit format which is appropriate for smaller target architectures and does not support multiprocessor configurations.

**8.29.2.2.1 32-Bit Object Identifier Format** The 32-bit format for an object ID is composed of four parts: API, object class, node, and index. The data type rtems_id is used to store object IDs.




The most significant five bits are the object class. The next three bits indicate the API to which the object class belongs. The next eight bits (16 .. 23) are the number of the node on which this object was created. The node number is always one (1) in a single processor system. The least significant 16-bits form an identifier within a particular object type. This identifier, called the object index, ranges in value from one to the maximum number of objects configured for this object type.

**8.29.2.2.2 16-Bit Object Identifier Format** The 16-bit format for an object ID is composed of three parts: API, object class, and index. The data type rtems_id is used to store object IDs.

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Contents | Class | | | | | API | | | Object Index | | | | | | | |

The 16-bit format is designed to be as similar as possible to the 32-bit format. The differences are limited to the elimination of the node field and reduction of the index field from 16-bits to 8-bits. Thus the 16-bit format only supports up to 255 object instances per API/Class combination and single processor systems. As this format is typically utilized by 16-bit processors with limited address space, this is more than enough object instances.

#### 8.29.2.3 Object Identifer Description

The components of an object ID make it possible to quickly locate any object in even the most complicated multiprocessor system. Object ID's are associated with an object by RTEMS when the object is created and the corresponding ID is returned by the appropriate object create directive. The object ID is required as input to all directives involving objects, except those which create an object or obtain the ID of an object.

The object identification directives can be used to dynamically obtain a particular object's ID given its name. This mapping is accomplished by searching the name table associated with this object type. If the name is non-unique, then the ID associated with the first occurrence of the name will be returned to the application. Since object IDs are returned when the object is created, the object identification directives are not necessary in a properly designed single processor application.

In addition, services are provided to portably examine the subcomponents of an RTEMS ID. These services are described in detail later in this manual but are prototyped as follows:

- rtems_object_id_get_api()
- rtems_object_id_get_class()
- rtems_object_id_get_node()
- rtems_object_id_get_index()

An object control block is a data structure defined by RTEMS which contains the information necessary to manage a particular object type. For efficiency reasons, the format of each object type's control block is different. However, many of the fields are similar in function. The number of each type of control block is application dependent and determined by the values specified in the user's Configuration Table. An object control block is allocated at object create time and freed when the object is deleted. With the exception of user extension routines, object control blocks are not directly manipulated by user applications.

### 8.29.3 Communication and Synchronization

In real-time multitasking applications, the ability for cooperating execution threads to communicate and synchronize with each other is imperative. A real-time executive should provide an application with the following capabilities

- data transfer between cooperating tasks,
- data transfer between tasks and ISRs,
- synchronization of cooperating tasks, and
- synchronization of tasks and ISRs.

Most RTEMS managers can be used to provide some form of communication and/or synchronization. However, managers dedicated specifically to communication and synchronization provide well established mechanisms which directly map to the application's varying needs. This level of flexibility allows the application designer to match the features of a particular manager with the complexity of communication and synchronization required. The following managers were specifically designed for communication and synchronization:

- ClassicSem
- ClassicMessageQueue
- ClassicEvent
- ClassicSignal

The semaphore manager supports mutual exclusion involving the synchronization of access to one or more shared user resources. Binary semaphores may utilize the optional priority inheritance algorithm to avoid the problem of priority inversion. The message manager supports both communication and synchronization, while the event manager primarily provides a high performance synchronization mechanism. The signal manager supports only asynchronous communication and is typically used for exception handling.

### 8.29.4 Time

The development of responsive real-time applications requires an understanding of how RTEMS maintains and supports time-related operations. The basic unit of time in RTEMS is known as a tick. The frequency of clock ticks is completely application dependent and determines the granularity and accuracy of all interval and calendar time operations.

By tracking time in units of ticks, RTEMS is capable of supporting interval timing functions such as task delays, timeouts, timeslicing, the delayed execution of timer service routines, and the rate monotonic scheduling of tasks. An interval is defined as a number of ticks relative to the current time. For example, when a task delays for an interval of ten ticks, it is implied that the task will not execute until ten clock ticks have occurred. All intervals are specified using data type rtems_interval.

A characteristic of interval timing is that the actual interval period may be a fraction of a tick less than the interval requested. This occurs because the time at which the delay timer is set up occurs at some time between two clock ticks. Therefore, the first countdown tick occurs in less than the complete time interval for a tick. This can be a problem if the clock granularity is large.

The rate monotonic scheduling algorithm is a hard real-time scheduling methodology. This methodology provides rules which allows one to guarantee that a set of independent periodic tasks will always meet their deadlines – even under transient overload conditions. The rate monotonic manager provides directives built upon the Clock Manager's interval timer support routines.

Interval timing is not sufficient for the many applications which require that time be kept in wall time or true calendar form. Consequently, RTEMS maintains the current date and time. This allows selected time operations to be scheduled at an actual calendar date and time. For example, a task could request to delay until midnight on New Year's Eve before lowering the ball at Times Square. The data type rtems_time_of_day is used to specify calendar time in RTEMS services. See Clock Manager Time and Date Data Structures.

**Todo** Link to Clock Manager Time and Date Data Structures

Obviously, the directives which use intervals or wall time cannot operate without some external mechanism which provides a periodic clock tick. This clock tick is typically provided by a real time clock or counter/timer device.

### 8.29.5 Memory Management

RTEMS memory management facilities can be grouped into two classes: dynamic memory allocation and address translation. Dynamic memory allocation is required by applications whose memory requirements vary through the application's course of execution. Address translation is needed by applications which share memory with another CPU or an intelligent Input/Output processor. The following RTEMS managers provide facilities to manage memory:

- ClassicRegion
- ClassicPart
- ClassicDPMEM

RTEMS memory management features allow an application to create simple memory pools of fixed size buffers and/or more complex memory pools of variable size segments. The partition manager provides directives to manage and maintain pools of fixed size entities such as resource control blocks. Alternatively, the region manager provides a more general purpose memory allocation scheme that supports variable size blocks of memory which are dynamically obtained and freed by the application. The dual-ported memory manager provides executive support for address translation between internal and external dual-ported RAM address space.

# 8.30 Classic API Configuration

## Macros

- #define CONFIGURE_MAXIMUM_BARRIERS

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_MESSAGE_QUEUES

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_PARTITIONS

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_PERIODS

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_PORTS

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_REGIONS

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_SEMAPHORES

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_TASKS

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_THREAD_LOCAL_STORAGE_SIZE

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_TIMERS

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_USER_EXTENSIONS

  *This configuration option is an integer define.*
- #define CONFIGURE_MINIMUM_TASKS_WITH_USER_PROVIDED_STORAGE

  *This configuration option is an integer define.*

## 8.30.1 Detailed Description

This section describes configuration options related to the Classic API.

## 8.30.2 Macro Definition Documentation

### 8.30.2.1 CONFIGURE_MAXIMUM_BARRIERS

```
#define CONFIGURE_MAXIMUM_BARRIERS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of Classic API Barriers that can be concurrently active.

#### Default Value

The default value is 0.

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to 65535.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It may be defined through rtems_resource_unlimited() the enable unlimited objects for this object class, if the value passed to rtems_resource_unlimited() satisfies all other constraints of this configuration option.

#### Notes

This object class can be configured in unlimited allocation mode, see Unlimited Objects.

Definition at line 583 of file appl-config.h.

### 8.30.2.2 CONFIGURE_MAXIMUM_MESSAGE_QUEUES

```
#define CONFIGURE_MAXIMUM_MESSAGE_QUEUES
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of Classic API Message Queues that can be concurrently active.

#### Default Value

The default value is 0.

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to 65535.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It may be defined through rtems_resource_unlimited() the enable unlimited objects for this object class, if the value passed to rtems_resource_unlimited() satisfies all other constraints of this configuration option.

#### Notes

This object class can be configured in unlimited allocation mode, see Unlimited Objects. You have to account for the memory used to store the messages of each message queue, see CONFIGURE_MESSAGE_BUFFER_MEMORY.

Definition at line 621 of file appl-config.h.

### 8.30.2.3 CONFIGURE_MAXIMUM_PARTITIONS

```
#define CONFIGURE_MAXIMUM_PARTITIONS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of Classic API Partitions that can be concurrently active.

#### Default Value

The default value is 0.

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to 65535.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It may be defined through rtems_resource_unlimited() the enable unlimited objects for this object class, if the value passed to rtems_resource_unlimited() satisfies all other constraints of this configuration option.

#### Notes

This object class can be configured in unlimited allocation mode, see Unlimited Objects.

Definition at line 658 of file appl-config.h.

### 8.30.2.4 CONFIGURE_MAXIMUM_PERIODS

```
#define CONFIGURE_MAXIMUM_PERIODS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of Classic API Periods that can be concurrently active.

#### Default Value

The default value is 0.

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to 65535.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It may be defined through rtems_resource_unlimited() the enable unlimited objects for this object class, if the value passed to rtems_resource_unlimited() satisfies all other constraints of this configuration option.

#### Notes

This object class can be configured in unlimited allocation mode, see Unlimited Objects.

Definition at line 695 of file appl-config.h.

#### 8.30.2.5 CONFIGURE_MAXIMUM_PORTS

```
#define CONFIGURE_MAXIMUM_PORTS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of Classic API Ports that can be concurrently active.

**Default Value**

The default value is 0.

**Value Constraints**

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to 65535.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It may be defined through rtems_resource_unlimited() the enable unlimited objects for this object class, if the value passed to rtems_resource_unlimited() satisfies all other constraints of this configuration option.

**Notes**

This object class can be configured in unlimited allocation mode, see Unlimited Objects.

Definition at line 732 of file appl-config.h.

#### 8.30.2.6 CONFIGURE_MAXIMUM_REGIONS

```
#define CONFIGURE_MAXIMUM_REGIONS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of Classic API Regions that can be concurrently active.

**Default Value**

The default value is 0.

**Value Constraints**

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to 65535.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It may be defined through rtems_resource_unlimited() the enable unlimited objects for this object class, if the value passed to rtems_resource_unlimited() satisfies all other constraints of this configuration option.

**Notes**

This object class can be configured in unlimited allocation mode, see Unlimited Objects.

Definition at line 769 of file appl-config.h.

#### 8.30.2.7 CONFIGURE_MAXIMUM_SEMAPHORES

```
#define CONFIGURE_MAXIMUM_SEMAPHORES
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of Classic API Semaphore that can be concurrently active.

##### Default Value

The default value is 0.

##### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to 65535.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It may be defined through rtems_resource_unlimited() the enable unlimited objects for this object class, if the value passed to rtems_resource_unlimited() satisfies all other constraints of this configuration option.

##### Notes

This object class can be configured in unlimited allocation mode, see Unlimited Objects.

In SMP configurations, the size of a Semaphore Control Block depends on the scheduler count (see Configuration Step 3 - Scheduler Table). The semaphores using the Multiprocessor Resource Sharing Protocol (MrsP) need a ceiling priority per scheduler.

Definition at line 815 of file appl-config.h.

#### 8.30.2.8 CONFIGURE_MAXIMUM_TASKS

```
#define CONFIGURE_MAXIMUM_TASKS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of Classic API Tasks that can be concurrently active.

##### Default Value

The default value is 0.

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to 65535.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It shall be small enough so that the task stack space calculation carried out by <rtems/confdefs.h> does not overflow an integer of type uintptr_t.
- It may be defined through rtems_resource_unlimited() the enable unlimited objects for this object class, if the value passed to rtems_resource_unlimited() satisfies all other constraints of this configuration option.

#### Notes

This object class can be configured in unlimited allocation mode, see Unlimited Objects.

The calculations for the required memory in the RTEMS Workspace for tasks assume that each task has a minimum stack size and has floating point support enabled. The configuration option CONFIGURE_EXTRA_TASK_STACKS is used to specify task stack requirements *above* the minimum size required.

The maximum number of POSIX threads is specified by CONFIGURE_MAXIMUM_POSIX_THREADS.

A future enhancement to <rtems/confdefs.h> could be to eliminate the assumption that all tasks have floating point enabled. This would require the addition of a new configuration parameter to specify the number of tasks which enable floating point support.

Definition at line 871 of file appl-config.h.

### 8.30.2.9 CONFIGURE_MAXIMUM_THREAD_LOCAL_STORAGE_SIZE

```
#define CONFIGURE_MAXIMUM_THREAD_LOCAL_STORAGE_SIZE
```

This configuration option is an integer define.

If the value of this configuration option is greater than zero, then it defines the maximum thread-local storage size, otherwise the thread-local storage size is defined by the linker depending on the thread-local storage objects used by the application in the statically-linked executable.

#### Default Value

The default value is 0.

#### Value Constraints

The value of this configuration option shall be greater than or equal to 0 and less than or equal to SIZE_↩ MAX.

#### Notes

This configuration option can be used to reserve space for the dynamic linking of modules with thread-local storage objects.

If the thread-local storage size defined by the thread-local storage objects used by the application in the statically-linked executable is greater than a non-zero value of this configuration option, then a fatal error will occur during system initialization.

Use RTEMS_ALIGN_UP() and RTEMS_TASK_STORAGE_ALIGNMENT to adjust the size to meet the minimum alignment requirement of a thread-local storage area.

The actual thread-local storage size is determined when the application executable is linked. The `rtems-exeinfo` command line tool included in the RTEMS Tools can be used to obtain the thread-local storage size and alignment of an application executable.

Definition at line 910 of file appl-config.h.

### 8.30.2.10 CONFIGURE_MAXIMUM_TIMERS

```
#define CONFIGURE_MAXIMUM_TIMERS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of Classic API Timers that can be concurrently active.

#### Default Value

The default value is 0.

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to 65535.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It may be defined through rtems_resource_unlimited() the enable unlimited objects for this object class, if the value passed to rtems_resource_unlimited() satisfies all other constraints of this configuration option.

#### Notes

This object class can be configured in unlimited allocation mode, see `Unlimited Objects`.

Definition at line 947 of file appl-config.h.

#### 8.30.2.11 CONFIGURE_MAXIMUM_USER_EXTENSIONS

```
#define CONFIGURE_MAXIMUM_USER_EXTENSIONS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of Classic API User Extensions that can be concurrently active.

##### Default Value

The default value is 0.

##### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to 65535.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.

##### Notes

This object class cannot be configured in unlimited allocation mode.

Definition at line 977 of file appl-config.h.

#### 8.30.2.12 CONFIGURE_MINIMUM_TASKS_WITH_USER_PROVIDED_STORAGE

```
#define CONFIGURE_MINIMUM_TASKS_WITH_USER_PROVIDED_STORAGE
```

This configuration option is an integer define.

The value of this configuration option defines the minimum count of Classic API Tasks which are constructed by rtems_task_construct().

##### Default Value

The default value is 0.

##### Value Constraints

The value of this configuration option shall be greater than or equal to 0 and less than or equal to CONFIGURE_MAXIMUM_TASKS.

##### Notes

By default, the calculation for the required memory in the RTEMS Workspace for tasks assumes that all Classic API Tasks are created by rtems_task_create(). This configuration option can be used to reduce the required memory for the system-provided task storage areas since tasks constructed by rtems_task_construct() use a user-provided task storage area.

Definition at line 1001 of file appl-config.h.

# 8.31 Classic API Initialization Task Configuration

## Macros

- #define CONFIGURE_INIT_TASK_ARGUMENTS

  *This configuration option is an integer define.*
- #define CONFIGURE_INIT_TASK_ATTRIBUTES

  *This configuration option is an integer define.*
- #define CONFIGURE_INIT_TASK_ENTRY_POINT

  *This configuration option is an initializer define.*
- #define CONFIGURE_INIT_TASK_INITIAL_MODES

  *This configuration option is an integer define.*
- #define CONFIGURE_INIT_TASK_NAME

  *This configuration option is an integer define.*
- #define CONFIGURE_INIT_TASK_PRIORITY

  *This configuration option is an integer define.*
- #define CONFIGURE_INIT_TASK_STACK_SIZE

  *This configuration option is an integer define.*
- #define CONFIGURE_RTEMS_INIT_TASKS_TABLE

  *This configuration option is a boolean feature define.*

## 8.31.1 Detailed Description

This section describes configuration options related to the Classic API initialization task.

## 8.31.2 Macro Definition Documentation

### 8.31.2.1 CONFIGURE_INIT_TASK_ARGUMENTS

```
#define CONFIGURE_INIT_TASK_ARGUMENTS
```

This configuration option is an integer define.

The value of this configuration option defines task argument of the Classic API initialization task.

#### Default Value

The default value is 0.

#### Value Constraints

The value of this configuration option shall be a valid integer of type rtems_task_argument.

Definition at line 1034 of file appl-config.h.

#### 8.31.2.2 CONFIGURE_INIT_TASK_ATTRIBUTES

```
#define CONFIGURE_INIT_TASK_ATTRIBUTES
```

This configuration option is an integer define.

The value of this configuration option defines the task attributes of the Classic API initialization task.

**Default Value**

The default value is RTEMS_DEFAULT_ATTRIBUTES.

**Value Constraints**

The value of this configuration option shall be a valid task attribute set.

Definition at line 1050 of file appl-config.h.

#### 8.31.2.3 CONFIGURE_INIT_TASK_ENTRY_POINT

```
#define CONFIGURE_INIT_TASK_ENTRY_POINT
```

This configuration option is an initializer define.

The value of this configuration option initializes the entry point of the Classic API initialization task.

**Default Value**

The default value is `Init`.

**Value Constraints**

The value of this configuration option shall be defined to a valid function pointer of the type `void ( *entry_point )( rtems_task_argument )`.

**Notes**

The application shall provide the function referenced by this configuration option.

Definition at line 1071 of file appl-config.h.

#### 8.31.2.4 CONFIGURE_INIT_TASK_INITIAL_MODES

```
#define CONFIGURE_INIT_TASK_INITIAL_MODES
```

This configuration option is an integer define.

The value of this configuration option defines the initial execution mode of the Classic API initialization task.

**Default Value**

In SMP configurations, the default value is RTEMS_DEFAULT_MODES otherwise the default value is RTEMS_NO_PREEMPT.

**Value Constraints**

The value of this configuration option shall be a valid task mode set.

Definition at line 1088 of file appl-config.h.

#### 8.31.2.5 CONFIGURE_INIT_TASK_NAME

```
#define CONFIGURE_INIT_TASK_NAME
```

This configuration option is an integer define.

The value of this configuration option defines the name of the Classic API initialization task.

**Default Value**

The default value is `rtems_build_name( 'U', 'I', '1', ' ' )`.

**Value Constraints**

The value of this configuration option shall be a valid integer of type rtems_name.

**Notes**

Use rtems_build_name() to define the task name.

Definition at line 1108 of file appl-config.h.

#### 8.31.2.6 CONFIGURE_INIT_TASK_PRIORITY

```
#define CONFIGURE_INIT_TASK_PRIORITY
```

This configuration option is an integer define.

The value of this configuration option defines the initial priority of the Classic API initialization task.

**Default Value**

The default value is 1.

**Value Constraints**

The value of this configuration option shall be a valid Classic API task priority. The set of valid task priorities is scheduler-specific.

Definition at line 1125 of file appl-config.h.

#### 8.31.2.7 CONFIGURE_INIT_TASK_STACK_SIZE

```
#define CONFIGURE_INIT_TASK_STACK_SIZE
```

This configuration option is an integer define.

The value of this configuration option defines the task stack size of the Classic API initialization task.

**Default Value**

The default value is CONFIGURE_MINIMUM_TASK_STACK_SIZE.

**Value Constraints**

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to CONFIGURE_MINIMUM_TASK_STACK_SIZE.
- It shall be small enough so that the task stack space calculation carried out by <rtems/confdefs.h> does not overflow an integer of type uintptr_t.

Definition at line 1150 of file appl-config.h.

#### 8.31.2.8 CONFIGURE_RTEMS_INIT_TASKS_TABLE

```
#define CONFIGURE_RTEMS_INIT_TASKS_TABLE
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then exactly one Classic API initialization task is configured.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

The application shall define exactly one of the following configuration options

- CONFIGURE_RTEMS_INIT_TASKS_TABLE,
- CONFIGURE_POSIX_INIT_THREAD_TABLE, or
- CONFIGURE_IDLE_TASK_INITIALIZES_APPLICATION

otherwise a compile time error in the configuration file will occur.

Definition at line 1178 of file appl-config.h.

# 8.32 Classic ASR Implementation

## Files

- file asrdata.h

  *Classic ASR Data Structures.*
- file asrimpl.h

  *Classic ASR Implementation.*

## Classes

- struct ASR_Information

## Functions

- RTEMS_INLINE_ROUTINE void _ASR_Initialize (ASR_Information *asr)

  *ASR_Initialize.*
- RTEMS_INLINE_ROUTINE rtems_signal_set **_ASR_Swap_signals** (ASR_Information *asr)
- RTEMS_INLINE_ROUTINE void **_ASR_Post_signals** (rtems_signal_set signals, rtems_signal_set *signal_set)
- RTEMS_INLINE_ROUTINE rtems_signal_set **_ASR_Get_posted_signals** (ASR_Information *asr)

### 8.32.1 Detailed Description

### 8.32.2 Function Documentation

#### 8.32.2.1 _ASR_Initialize()

```
RTEMS_INLINE_ROUTINE void _ASR_Initialize (
            ASR_Information * asr )
```

ASR_Initialize.

This routine initializes the given RTEMS_ASR information record.

Definition at line 41 of file asrimpl.h.

## 8.33 Classic Attributes Implementation

### Files

- file attrimpl.h

  *Classic Attributes Implementation.*

### Macros

- #define ATTRIBUTES_NOT_SUPPORTED 0
- #define ATTRIBUTES_REQUIRED RTEMS_FLOATING_POINT

### Functions

- RTEMS_INLINE_ROUTINE rtems_attribute _Attributes_Set (rtems_attribute new_attributes, rtems_attribute attribute_set)

  *Sets the requested new_attributes in the attribute_set passed in.*
- RTEMS_INLINE_ROUTINE rtems_attribute _Attributes_Clear (rtems_attribute attribute_set, rtems_attribute mask)

  *Clears the requested new_attributes in the attribute_set passed in.*
- RTEMS_INLINE_ROUTINE bool _Attributes_Is_floating_point (rtems_attribute attribute_set)

  *Checks if the floating point attribute is enabled in the attribute_set.*
- RTEMS_INLINE_ROUTINE bool _Attributes_Is_priority (rtems_attribute attribute_set)

  *Checks if the priority attribute is enabled in the attribute_set.*
- RTEMS_INLINE_ROUTINE bool _Attributes_Is_binary_semaphore (rtems_attribute attribute_set)

  *Checks if the binary semaphore attribute is enabled in the attribute_set.*
- RTEMS_INLINE_ROUTINE bool _Attributes_Is_simple_binary_semaphore (rtems_attribute attribute_set)

  *Checks if the simple binary semaphore attribute is enabled in the attribute_set.*
- RTEMS_INLINE_ROUTINE bool _Attributes_Is_counting_semaphore (rtems_attribute attribute_set)

  *Checks if the counting semaphore attribute is enabled in the attribute_set.*
- RTEMS_INLINE_ROUTINE bool _Attributes_Is_inherit_priority (rtems_attribute attribute_set)

  *Checks if the priority inheritance attribute is enabled in the attribute_set.*
- RTEMS_INLINE_ROUTINE bool _Attributes_Has_at_most_one_protocol (rtems_attribute attribute_set)

  *Returns true if the attribute set has at most one protocol, and false otherwise.*
- RTEMS_INLINE_ROUTINE bool _Attributes_Is_priority_ceiling (rtems_attribute attribute_set)

  *Checks if the priority ceiling attribute is enabled in the attribute_set.*
- RTEMS_INLINE_ROUTINE bool _Attributes_Is_multiprocessor_resource_sharing (rtems_attribute attribute↩ _set)

  *Checks if the Multiprocessor Resource Sharing Protocol attribute is enabled in the attribute_set.*
- RTEMS_INLINE_ROUTINE bool _Attributes_Is_barrier_automatic (rtems_attribute attribute_set)

  *Checks if the barrier automatic release attribute is enabled in the attribute_set.*
- RTEMS_INLINE_ROUTINE bool _Attributes_Is_system_task (rtems_attribute attribute_set)

  *Checks if the system task attribute is enabled in the attribute_set.*

### 8.33.1 Detailed Description

### 8.33.2 Macro Definition Documentation

#### 8.33.2.1 ATTRIBUTES_NOT_SUPPORTED

```
#define ATTRIBUTES_NOT_SUPPORTED 0
```

This attribute constant indicates the attributes that are not supportable given the hardware configuration.

Definition at line 42 of file attrimpl.h.

#### 8.33.2.2 ATTRIBUTES_REQUIRED

```
#define ATTRIBUTES_REQUIRED RTEMS_FLOATING_POINT
```

This attribute constant indicates the attributes that are required given the hardware configuration.

Definition at line 49 of file attrimpl.h.

### 8.33.3 Function Documentation

#### 8.33.3.1 _Attributes_Clear()

```
RTEMS_INLINE_ROUTINE rtems_attribute _Attributes_Clear (
            rtems_attribute attribute_set,
            rtems_attribute mask )
```

Clears the requested new_attributes in the attribute_set passed in.

This function clears the requested new_attributes in the attribute_set passed in. The result is returned to the user.

Definition at line 75 of file attrimpl.h.

#### 8.33.3.2 _Attributes_Has_at_most_one_protocol()

```
RTEMS_INLINE_ROUTINE bool _Attributes_Has_at_most_one_protocol (
            rtems_attribute attribute_set )
```

Returns true if the attribute set has at most one protocol, and false otherwise.

The protocols are RTEMS_INHERIT_PRIORITY, RTEMS_PRIORITY_CEILING and RTEMS_MULTIPROCESS↩OR_RESOURCE_SHARING.

Definition at line 190 of file attrimpl.h.

#### 8.33.3.3 _Attributes_Is_barrier_automatic()

```
RTEMS_INLINE_ROUTINE bool _Attributes_Is_barrier_automatic (
              rtems_attribute attribute_set )
```

Checks if the barrier automatic release attribute is enabled in the attribute_set.

This function returns TRUE if the barrier automatic release attribute is enabled in the attribute_set and FALSE otherwise.

Definition at line 235 of file attrimpl.h.

#### 8.33.3.4 _Attributes_Is_binary_semaphore()

```
RTEMS_INLINE_ROUTINE bool _Attributes_Is_binary_semaphore (
              rtems_attribute attribute_set )
```

Checks if the binary semaphore attribute is enabled in the attribute_set.

This function returns TRUE if the binary semaphore attribute is enabled in the attribute_set and FALSE otherwise.

Definition at line 133 of file attrimpl.h.

#### 8.33.3.5 _Attributes_Is_counting_semaphore()

```
RTEMS_INLINE_ROUTINE bool _Attributes_Is_counting_semaphore (
              rtems_attribute attribute_set )
```

Checks if the counting semaphore attribute is enabled in the attribute_set.

This function returns TRUE if the counting semaphore attribute is enabled in the attribute_set and FALSE otherwise.

Definition at line 162 of file attrimpl.h.

#### 8.33.3.6 _Attributes_Is_floating_point()

```
RTEMS_INLINE_ROUTINE bool _Attributes_Is_floating_point (
              rtems_attribute attribute_set )
```

Checks if the floating point attribute is enabled in the attribute_set.

This function returns TRUE if the floating point attribute is enabled in the attribute_set and FALSE otherwise.

Definition at line 90 of file attrimpl.h.

#### 8.33.3.7 _Attributes_Is_inherit_priority()

```
RTEMS_INLINE_ROUTINE bool _Attributes_Is_inherit_priority (
              rtems_attribute attribute_set )
```

Checks if the priority inheritance attribute is enabled in the attribute_set.

This function returns TRUE if the priority inheritance attribute is enabled in the attribute_set and FALSE otherwise.

Definition at line 176 of file attrimpl.h.

#### 8.33.3.8 _Attributes_Is_multiprocessor_resource_sharing()

```
RTEMS_INLINE_ROUTINE bool _Attributes_Is_multiprocessor_resource_sharing (
              rtems_attribute attribute_set )
```

Checks if the Multiprocessor Resource Sharing Protocol attribute is enabled in the attribute_set.

This function returns TRUE if the Multiprocessor Resource Sharing Protocol attribute is enabled in the attribute_set and FALSE otherwise.

Definition at line 221 of file attrimpl.h.

#### 8.33.3.9 _Attributes_Is_priority()

```
RTEMS_INLINE_ROUTINE bool _Attributes_Is_priority (
              rtems_attribute attribute_set )
```

Checks if the priority attribute is enabled in the attribute_set.

This function returns TRUE if the priority attribute is enabled in the attribute_set and FALSE otherwise.

Definition at line 119 of file attrimpl.h.

#### 8.33.3.10 _Attributes_Is_priority_ceiling()

```
RTEMS_INLINE_ROUTINE bool _Attributes_Is_priority_ceiling (
              rtems_attribute attribute_set )
```

Checks if the priority ceiling attribute is enabled in the attribute_set.

This function returns TRUE if the priority ceiling attribute is enabled in the attribute_set and FALSE otherwise.

Definition at line 207 of file attrimpl.h.

#### 8.33.3.11 _Attributes_Is_simple_binary_semaphore()

```
RTEMS_INLINE_ROUTINE bool _Attributes_Is_simple_binary_semaphore (
            rtems_attribute attribute_set )
```

Checks if the simple binary semaphore attribute is enabled in the attribute_set.

This function returns TRUE if the simple binary semaphore attribute is enabled in the attribute_set and FALSE otherwise.

Definition at line 147 of file attrimpl.h.

#### 8.33.3.12 _Attributes_Is_system_task()

```
RTEMS_INLINE_ROUTINE bool _Attributes_Is_system_task (
            rtems_attribute attribute_set )
```

Checks if the system task attribute is enabled in the attribute_set.

This function returns TRUE if the system task attribute is enabled in the attribute_set and FALSE otherwise.

Definition at line 249 of file attrimpl.h.

#### 8.33.3.13 _Attributes_Set()

```
RTEMS_INLINE_ROUTINE rtems_attribute _Attributes_Set (
            rtems_attribute new_attributes,
            rtems_attribute attribute_set )
```

Sets the requested new_attributes in the attribute_set passed in.

This function sets the requested new_attributes in the attribute_set passed in. The result is returned to the user.

Definition at line 60 of file attrimpl.h.

# 8.34 Classic Barrier Implementation

## Files

- file barrierdata.h

  *Classic Barrier Manager Data Structures.*
- file barrierimpl.h

  *Classic Barrier Manager Implementation.*
- file barrierident.c

  *rtems_barrier_ident() Implementation*

## Classes

- struct Barrier_Control

## Macros

- #define BARRIER_INFORMATION_DEFINE(max)

  *Macro to define the objects information for the Classic Barrier objects.*

## Functions

- static __inline__ Barrier_Control * _Barrier_Allocate (void)

  *_Barrier_Allocate*
- static __inline__ void _Barrier_Free (Barrier_Control *the_barrier)

  *_Barrier_Free*
- static __inline__ Barrier_Control * **_Barrier_Get** (Objects_Id id, Thread_queue_Context *queue_context)

## Variables

- Objects_Information _Barrier_Information

  *The Classic Barrier objects information.*

## 8.34.1 Detailed Description

## 8.34.2 Macro Definition Documentation

### 8.34.2.1 BARRIER_INFORMATION_DEFINE

```
#define BARRIER_INFORMATION_DEFINE(
              max )
```

**Value:**

```
  OBJECTS_INFORMATION_DEFINE( \
    _Barrier, \
    OBJECTS_CLASSIC_API, \
    OBJECTS_RTEMS_BARRIERS, \
    Barrier_Control, \
    max, \
    OBJECTS_NO_STRING_NAME, \
    NULL \
  )
```

Macro to define the objects information for the Classic Barrier objects.

This macro should only be used by <rtems/confdefs.h>.

**Parameters**

| | |
|---|---|
| *max* | The configured object maximum (the OBJECTS_UNLIMITED_OBJECTS flag may be set). |

Definition at line 61 of file barrierdata.h.

### 8.34.3 Function Documentation

#### 8.34.3.1 _Barrier_Allocate()

```
static __inline__ Barrier_Control* _Barrier_Allocate (
            void  )  [static]
```

_Barrier_Allocate

This function allocates a barrier control block from the inactive chain of free barrier control blocks.

Definition at line 43 of file barrierimpl.h.

#### 8.34.3.2 _Barrier_Free()

```
static __inline__ void _Barrier_Free (
            Barrier_Control * the_barrier )  [static]
```

_Barrier_Free

This routine frees a barrier control block to the inactive chain of free barrier control blocks.

Definition at line 54 of file barrierimpl.h.

# 8.35 Classic Message Queue Implementation

## Files

- file messageimpl.h

  *Classic Message Queue Manager Implementation.*
- file msgqcreate.c

  *This source file contains the implementation of rtems_message_queue_create().*
- file msgqident.c

  *rtems_message_queue_ident() Implementation*

## Classes

- struct Message_queue_Control

## Macros

- #define MESSAGE_QUEUE_INFORMATION_DEFINE(max)

  *Macro to define the objects information for the Classic Message Queue objects.*

## Enumerations

- enum Message_queue_Submit_types { MESSAGE_QUEUE_SEND_REQUEST = 0, MESSAGE_QUEUE_URGENT_REQUES = 1 }

## Functions

- rtems_status_code _Message_queue_Submit (rtems_id id, const void *buffer, size_t size, Message_queue_Submit_types submit_type)

  *Message_queue_Submit.*
- static __inline__ void _Message_queue_Free (Message_queue_Control *the_message_queue)

  *Deallocates a message queue control block into the inactive chain of free message queue control blocks.*
- static __inline__ Message_queue_Control * **_Message_queue_Get** (Objects_Id id, Thread_queue_Context *queue_context)
- static __inline__ Message_queue_Control * **_Message_queue_Allocate** (void)
- rtems_status_code _Message_queue_Create (const rtems_message_queue_config *config, rtems_id *id, CORE_message_queue_Allocate_buffers allocate_buffers)

  *Creates a message queue.*

## Variables

- Objects_Information _Message_queue_Information

  *The Classic Message Queue objects information.*

### 8.35.1 Detailed Description

### 8.35.2 Macro Definition Documentation

#### 8.35.2.1 MESSAGE_QUEUE_INFORMATION_DEFINE

```
#define MESSAGE_QUEUE_INFORMATION_DEFINE(
              max )
```

**Value:**

```
OBJECTS_INFORMATION_DEFINE( \
  _Message_queue, \
  OBJECTS_CLASSIC_API, \
  OBJECTS_RTEMS_MESSAGE_QUEUES, \
  Message_queue_Control, \
  max, \
  OBJECTS_NO_STRING_NAME, \
  _Message_queue_MP_Send_extract_proxy \
)
```

Macro to define the objects information for the Classic Message Queue objects.

This macro should only be used by <rtems/confdefs.h>.

**Parameters**

| | |
|---|---|
| *max* | The configured object maximum (the OBJECTS_UNLIMITED_OBJECTS flag may be set). |

Definition at line 77 of file messagedata.h.

### 8.35.3 Enumeration Type Documentation

#### 8.35.3.1 Message_queue_Submit_types

```
enum Message_queue_Submit_types
```

The following enumerated type details the modes in which a message may be submitted to a message queue. The message may be posted in a send or urgent fashion.

**Enumerator**

| | |
|---|---|
| MESSAGE_QUEUE_SEND_REQUEST | This value indicates the user wants to send the message using the normal message insertion protocol (FIFO or priority). |
| MESSAGE_QUEUE_URGENT_REQUEST | This value indicates the user considers the message to be urgent and wants it inserted at the head of the pending message queue. |

Definition at line 41 of file messageimpl.h.

## 8.35.4 Function Documentation

### 8.35.4.1 _Message_queue_Create()

```
rtems_status_code _Message_queue_Create (
            const rtems_message_queue_config * config,
            rtems_id * id,
            CORE_message_queue_Allocate_buffers allocate_buffers )
```

Creates a message queue.

**Parameters**

| | | |
|---|---|---|
| | *config* | is the message queue configuration. |
| out | *id* | contains the object identifier if the operation was successful. |
| | *allocate_buffers* | is the message buffer storage area allocation handler. |

Definition at line 53 of file msgqconstruct.c.

### 8.35.4.2 _Message_queue_Free()

```
static __inline__ void _Message_queue_Free (
            Message_queue_Control * the_message_queue )  [static]
```

Deallocates a message queue control block into the inactive chain of free message queue control blocks.

This routine deallocates a message queue control block into the inactive chain of free message queue control blocks.

Definition at line 78 of file messageimpl.h.

### 8.35.4.3 _Message_queue_Submit()

```
rtems_status_code _Message_queue_Submit (
            rtems_id id,
            const void * buffer,
            size_t size,
            Message_queue_Submit_types submit_type )
```

Message_queue_Submit.

This routine implements the directives rtems_message_queue_send and rtems_message_queue_urgent. It processes a message that is to be submitted to the designated message queue. The message will either be processed as a send send message which it will be inserted at the rear of the queue or it will be processed as an urgent message which will be inserted at the front of the queue.

# 8.36 Classic Modes Implementation

## Files

- file modesimpl.h

  *Classic Modes Implementation.*

## Functions

- static __inline__ bool _Modes_Mask_changed (rtems_mode mode_set, rtems_mode masks)

  *Checks if any of the mode flags in mask are set in mode_set.*
- static __inline__ bool _Modes_Is_asr_disabled (rtems_mode mode_set)

  *Checks if mode_set says that Asynchronous Signal Processing is disabled.*
- static __inline__ bool _Modes_Is_preempt (rtems_mode mode_set)

  *Checks if mode_set indicates that preemption is enabled.*
- static __inline__ bool _Modes_Is_timeslice (rtems_mode mode_set)

  *Checks if mode_set indicates that timeslicing is enabled.*
- static __inline__ ISR_Level _Modes_Get_interrupt_level (rtems_mode mode_set)

  *Gets the interrupt level portion of the mode_set.*
- static __inline__ void _Modes_Set_interrupt_level (rtems_mode mode_set)

  *Sets the current interrupt level to that specified in the mode_set.*
- static __inline__ void _Modes_Change (rtems_mode old_mode_set, rtems_mode new_mode_set, rtems_mode mask, rtems_mode *out_mode_set, rtems_mode *changed)

  *Changes the modes in old_mode_set indicated by mask to the requested values in new_mode_set.*

## 8.36.1 Detailed Description

## 8.36.2 Function Documentation

### 8.36.2.1 _Modes_Change()

```
static __inline__ void _Modes_Change (
            rtems_mode old_mode_set,
            rtems_mode new_mode_set,
            rtems_mode mask,
            rtems_mode * out_mode_set,
            rtems_mode * changed )  [static]
```

Changes the modes in old_mode_set indicated by mask to the requested values in new_mode_set.

This routine changes the modes in old_mode_set indicated by mask to the requested values in new_mode_set. The resulting mode set is returned in out_mode_set and the modes that changed is returned in changed.

Definition at line 122 of file modesimpl.h.

#### 8.36.2.2 _Modes_Get_interrupt_level()

```
static __inline__ ISR_Level _Modes_Get_interrupt_level (
             rtems_mode mode_set )  [static]
```

Gets the interrupt level portion of the mode_set.

This function returns the interrupt level portion of the mode_set.

Definition at line 93 of file modesimpl.h.

#### 8.36.2.3 _Modes_Is_asr_disabled()

```
static __inline__ bool _Modes_Is_asr_disabled (
             rtems_mode mode_set )  [static]
```

Checks if mode_set says that Asynchronous Signal Processing is disabled.

This function returns TRUE if mode_set indicates that Asynchronous Signal Processing is disabled, and FALSE otherwise.

Definition at line 55 of file modesimpl.h.

#### 8.36.2.4 _Modes_Is_preempt()

```
static __inline__ bool _Modes_Is_preempt (
             rtems_mode mode_set )  [static]
```

Checks if mode_set indicates that preemption is enabled.

This function returns TRUE if mode_set indicates that preemption is enabled, and FALSE otherwise.

Definition at line 68 of file modesimpl.h.

#### 8.36.2.5 _Modes_Is_timeslice()

```
static __inline__ bool _Modes_Is_timeslice (
             rtems_mode mode_set )  [static]
```

Checks if mode_set indicates that timeslicing is enabled.

This function returns TRUE if mode_set indicates that timeslicing is enabled, and FALSE otherwise.

Definition at line 81 of file modesimpl.h.

#### 8.36.2.6 _Modes_Mask_changed()

```
static __inline__ bool _Modes_Mask_changed (
             rtems_mode mode_set,
             rtems_mode masks )  [static]
```

Checks if any of the mode flags in mask are set in mode_set.

This function returns TRUE if any of the mode flags in mask are set in mode_set, and FALSE otherwise.

Definition at line 41 of file modesimpl.h.

#### 8.36.2.7 _Modes_Set_interrupt_level()

```
static __inline__ void _Modes_Set_interrupt_level (
             rtems_mode mode_set )  [static]
```

Sets the current interrupt level to that specified in the mode_set.

This routine sets the current interrupt level to that specified in the mode_set.

Definition at line 106 of file modesimpl.h.

# 8.37 Classic Object Implementation

## Files

- file objectimpl.h

  *Implementation Interfaces for Classic Objects.*
- file rtemsnametoid.c

  *_RTEMS_Name_to_id() Implementation*

## Functions

- rtems_status_code _RTEMS_Name_to_id (uint32_t name, uint32_t node, Objects_Id *id, const Objects_Information *information)

  *Calls _Objects_Name_to_id_u32() and converts the status.*

## 8.37.1 Detailed Description

## 8.37.2 Function Documentation

### 8.37.2.1 _RTEMS_Name_to_id()

```
rtems_status_code _RTEMS_Name_to_id (
            uint32_t name,
            uint32_t node,
            Objects_Id * id,
            const Objects_Information * information )
```

Calls _Objects_Name_to_id_u32() and converts the status.

**Parameters**

| | | |
|---|---|---|
| | *name* | is the name of the object to find. |
| | *node* | is the set of nodes to search. |
| out | *id* | is the pointer to an object identifier variable or NULL. The object identifier will be stored in the referenced variable, if the operation was successful. |
| | *information* | is the pointer to an object class information block. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The operations was successful. |
| *RTEMS_INVALID_ADDRESS* | The id parameter was NULL. |
| *RTEMS_INVALID_NAME* | No object exists with the specified name on the specified node set. |

Definition at line 43 of file rtemsnametoid.c.

## 8.38 Classic Options Implementation

### Files

- file optionsimpl.h

  *Classic Options Implementation.*

### Functions

- RTEMS_INLINE_ROUTINE bool _Options_Is_no_wait (rtems_option option_set)

  *Checks if the RTEMS_NO_WAIT option is enabled in option_set.*
- RTEMS_INLINE_ROUTINE bool _Options_Is_any (rtems_option option_set)

  *Checks if the RTEMS_EVENT_ANY option is enabled in OPTION_SET.*

### 8.38.1 Detailed Description

### 8.38.2 Function Documentation

#### 8.38.2.1 _Options_Is_any()

```
RTEMS_INLINE_ROUTINE bool _Options_Is_any (
            rtems_option option_set )
```

Checks if the RTEMS_EVENT_ANY option is enabled in OPTION_SET.

This function returns TRUE if the RTEMS_EVENT_ANY option is enabled in OPTION_SET, and FALSE otherwise.

Definition at line 53 of file optionsimpl.h.

#### 8.38.2.2 _Options_Is_no_wait()

```
RTEMS_INLINE_ROUTINE bool _Options_Is_no_wait (
            rtems_option option_set )
```

Checks if the RTEMS_NO_WAIT option is enabled in option_set.

This function returns TRUE if the RTEMS_NO_WAIT option is enabled in option_set, and FALSE otherwise.

Definition at line 40 of file optionsimpl.h.

# 8.39 Classic Rate Monotonic Scheduler Implementation

## Files

- file ratemondata.h

  *Classic Rate Monotonic Scheduler Data Structures.*
- file ratemonimpl.h

  *Classic Rate Monotonic Scheduler Implementation.*
- file ratemonident.c

  *rtems_rate_monotonic_ident() Implementation*

## Classes

- struct Rate_monotonic_Statistics
- struct Rate_monotonic_Control

  *The following structure defines the control block used to manage each period.*

## Macros

- #define RATE_MONOTONIC_INFORMATION_DEFINE(max)

  *Macro to define the objects information for the Classic Rate Monotonic objects.*
- #define **RATE_MONOTONIC_INTEND_TO_BLOCK** ( THREAD_WAIT_CLASS_PERIOD | THREAD_WAIT_STATE_INTEND )
- #define **RATE_MONOTONIC_BLOCKED** ( THREAD_WAIT_CLASS_PERIOD | THREAD_WAIT_STATE_BLOCKED )
- #define **RATE_MONOTONIC_READY_AGAIN** ( THREAD_WAIT_CLASS_PERIOD | THREAD_WAIT_STATE_READY_AGA )

## Functions

- static __inline__ Rate_monotonic_Control * _Rate_monotonic_Allocate (void)

  *Allocates a period control block from the inactive chain of free period control blocks.*
- static __inline__ void **_Rate_monotonic_Acquire_critical** (Rate_monotonic_Control *the_period, ISR_lock_Context *lock_context)
- static __inline__ void **_Rate_monotonic_Release** (Rate_monotonic_Control *the_period, ISR_lock_Context *lock_context)
- static __inline__ Rate_monotonic_Control * **_Rate_monotonic_Get** (Objects_Id id, ISR_lock_Context *lock_context)
- void **_Rate_monotonic_Timeout** (Watchdog_Control *watchdog)
- bool _Rate_monotonic_Get_status (const Rate_monotonic_Control *the_period, Timestamp_Control *wall_since_last_period, Timestamp_Control *cpu_since_last_period)

  *_Rate_monotonic_Get_status(*
- void **_Rate_monotonic_Restart** (Rate_monotonic_Control *the_period, Thread_Control *owner, ISR_lock_Context *lock_context)
- void **_Rate_monotonic_Cancel** (Rate_monotonic_Control *the_period, Thread_Control *owner, ISR_lock_Context *lock_context)
- static __inline__ void **_Rate_monotonic_Reset_min_time** (Timestamp_Control *min_time)
- static __inline__ void **_Rate_monotonic_Reset_statistics** (Rate_monotonic_Control *the_period)

## Variables

- Objects_Information _Rate_monotonic_Information

  *The Classic Rate Monotonic objects information.*

## 8.39.1 Detailed Description

## 8.39.2 Macro Definition Documentation

### 8.39.2.1 RATE_MONOTONIC_INFORMATION_DEFINE

```
#define RATE_MONOTONIC_INFORMATION_DEFINE(
              max )
```

**Value:**

```
  OBJECTS_INFORMATION_DEFINE( \
    _Rate_monotonic, \
    OBJECTS_CLASSIC_API, \
    OBJECTS_RTEMS_PERIODS, \
    Rate_monotonic_Control, \
    max, \
    OBJECTS_NO_STRING_NAME, \
    NULL \
  )
```

Macro to define the objects information for the Classic Rate Monotonic objects.

This macro should only be used by <rtems/confdefs.h>.

**Parameters**

| | |
|---|---|
| *max* | The configured object maximum (the OBJECTS_UNLIMITED_OBJECTS flag may be set). |

Definition at line 147 of file ratemondata.h.

## 8.39.3 Function Documentation

### 8.39.3.1 _Rate_monotonic_Allocate()

```
static __inline__ Rate_monotonic_Control* _Rate_monotonic_Allocate (
            void  )  [static]
```

Allocates a period control block from the inactive chain of free period control blocks.

This function allocates a period control block from the inactive chain of free period control blocks.

Definition at line 58 of file ratemonimpl.h.

#### 8.39.3.2 _Rate_monotonic_Get_status()

```
bool _Rate_monotonic_Get_status (
             const Rate_monotonic_Control * the_period,
             Timestamp_Control * wall_since_last_period,
             Timestamp_Control * cpu_since_last_period )
```

_Rate_monotonic_Get_status(

This routine is invoked to compute the elapsed wall time and cpu time for a period.

**Parameters**

| | | |
|---|---|---|
| in | *the_period* | points to the period being operated upon. |
| out | *wall_since_last_period* | is set to the wall time elapsed since the period was initiated. |
| out | *cpu_since_last_period* | is set to the cpu time used by the owning thread since the period was initiated. |

**Return values**

| | |
|---|---|
| *This* | routine returns true if the status can be determined and false otherwise. |

Definition at line 27 of file ratemonperiod.c.

## 8.40 Classic Region Manager Implementation

### Files

- file regiondata.h

  *Classic Region Manager Data Structures.*
- file regionimpl.h

  *Classic Region Manager Implementation.*

### Classes

- struct Region_Control

### Macros

- #define REGION_INFORMATION_DEFINE(max)

  *Macro to define the objects information for the Classic Region objects.*
- #define **REGION_OF_THREAD_QUEUE_QUEUE**(queue) RTEMS_CONTAINER_OF( queue, Region_Control, Wait_queue.Queue )

### Functions

- static __inline__ Region_Control * _Region_Allocate (void)

  *Region_Allocate.*
- static __inline__ void _Region_Free (Region_Control *the_region)

  *Region_Free.*
- static __inline__ Region_Control * **_Region_Get_and_lock** (Objects_Id id)
- static __inline__ void **_Region_Unlock** (Region_Control *the_region)
- static __inline__ void * _Region_Allocate_segment (Region_Control *the_region, uintptr_t size)

  *Region_Allocate_segment.*
- static __inline__ bool _Region_Free_segment (Region_Control *the_region, void *the_segment)

  *Region_Free_segment.*
- void _Region_Process_queue (Region_Control *the_region)

  *Process Region Queue.*

### Variables

- Objects_Information _Region_Information

  *The Classic Region objects information.*

### 8.40.1 Detailed Description

### 8.40.2 Macro Definition Documentation

#### 8.40.2.1 REGION_INFORMATION_DEFINE

```
#define REGION_INFORMATION_DEFINE(
              max )
```

**Value:**

```
  OBJECTS_INFORMATION_DEFINE( \
    _Region, \
    OBJECTS_CLASSIC_API, \
    OBJECTS_RTEMS_REGIONS, \
    Region_Control, \
    max, \
    OBJECTS_NO_STRING_NAME, \
    NULL \
  )
```

Macro to define the objects information for the Classic Region objects.

This macro should only be used by <rtems/confdefs.h>.

**Parameters**

| | |
|---|---|
| *max* | The configured object maximum (the OBJECTS_UNLIMITED_OBJECTS flag may be set). |

Definition at line 63 of file regiondata.h.

### 8.40.3 Function Documentation

#### 8.40.3.1 _Region_Allocate()

```
static __inline__ Region_Control* _Region_Allocate (
             void  )  [static]
```

Region_Allocate.

This function allocates a region control block from the inactive chain of free region control blocks.

Definition at line 47 of file regionimpl.h.

#### 8.40.3.2 _Region_Allocate_segment()

```
static __inline__ void* _Region_Allocate_segment (
             Region_Control * the_region,
             uintptr_t size )  [static]
```

Region_Allocate_segment.

This function attempts to allocate a segment from the_region. If successful, it returns the address of the allocated segment. Otherwise, it returns NULL.

Definition at line 97 of file regionimpl.h.

#### 8.40.3.3 _Region_Free()

```
static __inline__ void _Region_Free (
             Region_Control * the_region )  [static]
```

Region_Free.

This routine frees a region control block to the inactive chain of free region control blocks.

Definition at line 58 of file regionimpl.h.

#### 8.40.3.4 _Region_Free_segment()

```
static __inline__ bool _Region_Free_segment (
             Region_Control * the_region,
             void * the_segment )  [static]
```

Region_Free_segment.

This function frees the_segment to the_region.

Definition at line 110 of file regionimpl.h.

#### 8.40.3.5 _Region_Process_queue()

```
void _Region_Process_queue (
             Region_Control * the_region )
```

Process Region Queue.

This is a helper routine which is invoked any time memory is freed. It looks at the set of waiting tasks and attempts to satisfy all outstanding requests.

**Parameters**

| | | |
|---|---|---|
| in | *the_region* | is the the region |

# 8.41 Classic Status Implementation

## Files

- file statusimpl.h

  *Classic Status Implementation.*

## Functions

- static __inline__ rtems_status_code **_Status_Get** (Status_Control status)
- static __inline__ rtems_status_code **_Status_Get_after_wait** (const Thread_Control *executing)

## Variables

- const rtems_status_code _Status_Object_name_errors_to_status [ ]

  *Status Object Name Errors to Status Array.*

### 8.41.1 Detailed Description

### 8.41.2 Variable Documentation

#### 8.41.2.1 _Status_Object_name_errors_to_status

```
const rtems_status_code _Status_Object_name_errors_to_status[]  [extern]
```

Status Object Name Errors to Status Array.

This array is used to map SuperCore Object Handler return codes to Classic API status codes.

Definition at line 18 of file status.c.

## 8.42 Classic Tasks Manager Implementation

### Files

- file tasksdata.h

  *Classic Tasks Manager Data Structures.*
- file tasksimpl.h

  *Classic Tasks Manager Implementation.*
- file taskident.c

  *rtems_task_ident() Implementation*

### Classes

- struct RTEMS_API_Control

### Typedefs

- typedef rtems_status_code(∗ **RTEMS_tasks_Prepare_stack**) (Thread_Configuration ∗, const rtems_task_config ∗)

### Functions

- void _RTEMS_tasks_Initialize_user_task (void)

  *System initialization handler to create and start the first user task.*
- void _RTEMS_tasks_Initialize_user_tasks (void)

  *RTEMS User Task Initialization.*
- rtems_status_code **_RTEMS_tasks_Create** (const rtems_task_config ∗config, rtems_id ∗id, RTEMS_↩ tasks_Prepare_stack prepare_stack)
- static __inline__ Thread_Control ∗ **_RTEMS_tasks_Allocate** (void)
- static __inline__ void _RTEMS_tasks_Free (Thread_Control ∗the_task)

  *Frees a task control block.*
- static __inline__ Priority_Control _RTEMS_Priority_To_core (const Scheduler_Control ∗scheduler, rtems_task_priority priority, bool ∗valid)

  *Converts the RTEMS API priority to the corresponding SuperCore priority and validates it.*
- static __inline__ rtems_task_priority _RTEMS_Priority_From_core (const Scheduler_Control ∗scheduler, Priority_Control priority)

  *Converts the SuperCore priority to the corresponding RTEMS API priority.*

### Variables

- const rtems_initialization_tasks_table _RTEMS_tasks_User_task_table

  *Initialization table for the first user task.*
- Thread_Information _RTEMS_tasks_Information

### 8.42.1 Detailed Description

### 8.42.2 Function Documentation

#### 8.42.2.1 _RTEMS_Priority_From_core()

```
static __inline__ rtems_task_priority _RTEMS_Priority_From_core (
            const Scheduler_Control * scheduler,
            Priority_Control priority ) [static]
```

Converts the SuperCore priority to the corresponding RTEMS API priority.

**Parameters**

| | | |
|---|---|---|
| in | *scheduler* | The scheduler instance. |
| in | *priority* | The SuperCore priority to convert. |

**Returns**

The corresponding RTEMS API priority.

Definition at line 116 of file tasksimpl.h.

#### 8.42.2.2 _RTEMS_Priority_To_core()

```
static __inline__ Priority_Control _RTEMS_Priority_To_core (
            const Scheduler_Control * scheduler,
            rtems_task_priority priority,
            bool * valid ) [static]
```

Converts the RTEMS API priority to the corresponding SuperCore priority and validates it.

The RTEMS API system priority is accepted as valid.

**Parameters**

| | | |
|---|---|---|
| in | *scheduler* | The scheduler instance. |
| in | *priority* | The RTEMS API priority to convert and validate. |
| out | *valid* | Indicates if the RTEMS API priority is valid and a corresponding SuperCore priority in the specified scheduler instance exists. |

**Returns**

The corresponding SuperCore priority.

Definition at line 96 of file tasksimpl.h.

#### 8.42.2.3 _RTEMS_tasks_Free()

```
static __inline__ void _RTEMS_tasks_Free (
            Thread_Control * the_task ) [static]
```

Frees a task control block.

This routine frees a task control block to the inactive chain of free task control blocks.

Definition at line 72 of file tasksimpl.h.

#### 8.42.2.4 _RTEMS_tasks_Initialize_user_tasks()

```
void _RTEMS_tasks_Initialize_user_tasks (
             void  )
```

RTEMS User Task Initialization.

This routine creates and starts all configured user initialization threads.

### 8.42.3 Variable Documentation

#### 8.42.3.1 _RTEMS_tasks_Information

```
Thread_Information _RTEMS_tasks_Information  [extern]
```

The following instantiates the information control block used to manage this class of objects.

#### 8.42.3.2 _RTEMS_tasks_User_task_table

```
const rtems_initialization_tasks_table _RTEMS_tasks_User_task_table  [extern]
```

Initialization table for the first user task.

This table is used by _RTEMS_tasks_Initialize_user_task() and initialized via <rtems/confdefs.h>.

Definition at line 34 of file taskinitdefault.c.

# 8.43 Classic Timer Implementation

## Files

- file timerdata.h

  *Classic Partition Manager Data Structures.*
- file timerimpl.h

  *Classic Timer Implementation.*
- file timerident.c

  *rtems_timer_ident() Implementation*

## Classes

- struct Timer_Control
- struct Timer_server_Control

## Macros

- #define TIMER_INFORMATION_DEFINE(max)

  *Macro to define the objects information for the Classic Timer objects.*

## Typedefs

- typedef struct Timer_server_Control **Timer_server_Control**

## Functions

- static __inline__ Timer_Control * _Timer_Allocate (void)

  *Timer_Allocate.*
- static __inline__ void _Timer_Free (Timer_Control *the_timer)

  *Timer_Free.*
- static __inline__ Timer_Control * **_Timer_Get** (Objects_Id id, ISR_lock_Context *lock_context)
- static __inline__ Per_CPU_Control * **_Timer_Acquire_critical** (Timer_Control *the_timer, ISR_lock_Context *lock_context)
- static __inline__ void **_Timer_Release** (Per_CPU_Control *cpu, ISR_lock_Context *lock_context)
- static __inline__ bool **_Timer_Is_interval_class** (Timer_Classes the_class)
- static __inline__ bool **_Timer_Is_on_task_class** (Timer_Classes the_class)
- static __inline__ Per_CPU_Watchdog_index **_Timer_Watchdog_header_index** (Timer_Classes the_class)
- static __inline__ Watchdog_Interval **_Timer_Get_CPU_ticks** (const Per_CPU_Control *cpu)
- rtems_status_code **_Timer_Fire** (rtems_id id, rtems_interval interval, rtems_timer_service_routine_entry routine, void *user_data, Timer_Classes the_class, Watchdog_Service_routine_entry adaptor)
- rtems_status_code **_Timer_Fire_after** (rtems_id id, rtems_interval ticks, rtems_timer_service_routine_entry routine, void *user_data, Timer_Classes the_class, Watchdog_Service_routine_entry adaptor)
- rtems_status_code **_Timer_Fire_when** (rtems_id id, const rtems_time_of_day *wall_time, rtems_timer_service_routine_entry routine, void *user_data, Timer_Classes the_class, Watchdog_Service_routine_entry adaptor)
- void **_Timer_Cancel** (Per_CPU_Control *cpu, Timer_Control *the_timer)
- void **_Timer_Routine_adaptor** (Watchdog_Control *the_watchdog)
- void **_Timer_server_Routine_adaptor** (Watchdog_Control *the_watchdog)
- static __inline__ void **_Timer_server_Acquire_critical** (Timer_server_Control *timer_server, ISR_lock_Context *lock_context)
- static __inline__ void **_Timer_server_Release_critical** (Timer_server_Control *timer_server, ISR_lock_Context *lock_context)

## Variables

- Objects_Information _Timer_Information

  *The Classic Timer objects information.*
- Timer_server_Control *volatile _Timer_server

  *Pointer to default timer server control block.*

### 8.43.1 Detailed Description

### 8.43.2 Macro Definition Documentation

#### 8.43.2.1 TIMER_INFORMATION_DEFINE

```
#define TIMER_INFORMATION_DEFINE(
              max )
```

**Value:**

```
OBJECTS_INFORMATION_DEFINE( \
  _Timer, \
  OBJECTS_CLASSIC_API, \
  OBJECTS_RTEMS_TIMERS, \
  Timer_Control, \
  max, \
  OBJECTS_NO_STRING_NAME, \
  NULL \
)
```

Macro to define the objects information for the Classic Timer objects.

This macro should only be used by <rtems/confdefs.h>.

**Parameters**

| | |
|---|---|
| *max* | The configured object maximum (the OBJECTS_UNLIMITED_OBJECTS flag may be set). |

Definition at line 74 of file timerdata.h.

### 8.43.3 Function Documentation

#### 8.43.3.1 _Timer_Allocate()

```
static __inline__ Timer_Control* _Timer_Allocate (
             void  )  [static]
```

Timer_Allocate.

This function allocates a timer control block from the inactive chain of free timer control blocks.

Definition at line 61 of file timerimpl.h.

#### 8.43.3.2 _Timer_Free()

```
static __inline__ void _Timer_Free (
            Timer_Control * the_timer )  [static]
```

Timer_Free.

This routine frees a timer control block to the inactive chain of free timer control blocks.

Definition at line 72 of file timerimpl.h.

### 8.43.4 Variable Documentation

#### 8.43.4.1 _Timer_server

```
Timer_server_Control* volatile _Timer_server  [extern]
```

Pointer to default timer server control block.

This value is NULL when the default timer server is not initialized.

Definition at line 37 of file timercreate.c.

## 8.44 Clock Driver

The Clock Driver API.

### Files

- file clockdrv.h
  *Clock Driver API.*

### Functions

- void _Clock_Initialize (void)
  *Initialize the clock driver.*

### Variables

- volatile uint32_t Clock_driver_ticks
  *Count of clock driver ticks since system boot or last overflow.*

### 8.44.1 Detailed Description

The Clock Driver API.

### 8.44.2 Variable Documentation

#### 8.44.2.1 Clock_driver_ticks

```
volatile uint32_t Clock_driver_ticks  [extern]
```

Count of clock driver ticks since system boot or last overflow.

This counter may overflow.

Count of clock driver ticks since system boot or last overflow.

Clock ticks since initialization

Definition at line 114 of file clockimpl.h.

# 8.45 Clock Manager

The Clock Manager provides support for time of day and other time related capabilities.

## Macros

- #define rtems_clock_get_ticks_per_second() _Watchdog_Ticks_per_second
  %
- #define rtems_clock_get_ticks_since_boot() _Watchdog_Ticks_since_boot
  %

## Functions

- rtems_status_code rtems_clock_get_seconds_since_epoch (rtems_interval *the_interval)
  %
- rtems_status_code rtems_clock_get_tod (rtems_time_of_day *time_buffer)
  %
- rtems_status_code rtems_clock_get_tod_timeval (struct timeval *time)
  %
- rtems_status_code rtems_clock_get_uptime (struct timespec *uptime)
  %
- uint64_t rtems_clock_get_uptime_nanoseconds (void)
  %
- time_t rtems_clock_get_uptime_seconds (void)
  %
- void rtems_clock_get_uptime_timeval (struct timeval *uptime)
  %
- rtems_status_code rtems_clock_set (const rtems_time_of_day *time_buffer)
  %
- rtems_status_code rtems_clock_tick (void)
  %
- static bool rtems_clock_tick_before (rtems_interval tick)
  *Returns true if the current ticks counter value indicates a time before the time specified by the tick value and false otherwise.*
- static rtems_interval rtems_clock_tick_later (rtems_interval delta)
  *Returns the ticks counter value delta ticks in the future.*
- static rtems_interval rtems_clock_tick_later_usec (rtems_interval delta_in_usec)
  *Returns the ticks counter value at least delta microseconds in the future.*

## 8.45.1 Detailed Description

The Clock Manager provides support for time of day and other time related capabilities.

## 8.45.2 Function Documentation

### 8.45.2.1 rtems_clock_get_seconds_since_epoch()

```
rtems_status_code rtems_clock_get_seconds_since_epoch (
            rtems_interval * the_interval )
```

%

**Parameters**

| | |
|---|---|
| *the_interval* | % |

#### 8.45.2.2 rtems_clock_get_tod()

```
rtems_status_code rtems_clock_get_tod (
            rtems_time_of_day * time_buffer )
```

%

**Parameters**

| | |
|---|---|
| *time_buffer* | % |

#### 8.45.2.3 rtems_clock_get_tod_timeval()

```
rtems_status_code rtems_clock_get_tod_timeval (
            struct timeval * time )
```

%

**Parameters**

| | |
|---|---|
| *time* | % |

#### 8.45.2.4 rtems_clock_get_uptime()

```
rtems_status_code rtems_clock_get_uptime (
            struct timespec * uptime )
```

%

**Parameters**

| | |
|---|---|
| *uptime* | % |

Definition at line 39 of file clockgetuptime.c.

#### 8.45.2.5 rtems_clock_get_uptime_timeval()

```
void rtems_clock_get_uptime_timeval (
             struct timeval * uptime )
```

%

**Parameters**

| | |
|---|---|
| *uptime* | % |

#### 8.45.2.6 rtems_clock_set()

```
rtems_status_code rtems_clock_set (
             const rtems_time_of_day * time_buffer )
```

%

**Parameters**

| | |
|---|---|
| *time_buffer* | % |

#### 8.45.2.7 rtems_clock_tick_before()

```
static bool rtems_clock_tick_before (
             rtems_interval tick )  [inline], [static]
```

Returns true if the current ticks counter value indicates a time before the time specified by the tick value and false otherwise.

This directive can be used to write busy loops with a timeout.

**Parameters**

| | |
|---|---|
| *tick* | is the tick value. |

**Return values**

| | |
|---|---|
| *true* | The current ticks counter value indicates a time before the time specified by the tick value. |
| *false* | Otherwise. |

Definition at line 209 of file clock.h.

#### 8.45.2.8 rtems_clock_tick_later()

```
static rtems_interval rtems_clock_tick_later (
            rtems_interval delta )  [inline], [static]
```

Returns the ticks counter value delta ticks in the future.

**Parameters**

| | |
|---|---|
| *delta* | is the ticks delta value. |

**Returns**

The tick counter value delta ticks in the future is returned.

Definition at line 225 of file clock.h.

#### 8.45.2.9 rtems_clock_tick_later_usec()

```
static rtems_interval rtems_clock_tick_later_usec (
            rtems_interval delta_in_usec )  [inline], [static]
```

Returns the ticks counter value at least delta microseconds in the future.

**Parameters**

| | |
|---|---|
| *delta_in_usec* | is the delta value in microseconds. |

**Returns**

The tick counter value delta ticks in the future is returned.

Definition at line 242 of file clock.h.

## 8.46 Clock Support

Clock support.

### Files

- file clockimpl.h

  *Clock Tick Device Driver Shell.*

### 8.46.1 Detailed Description

Clock support.

## 8.47 Console Driver Support

Console Driver Support for Board Support Packages.

Console Driver Support for Board Support Packages.

# 8.48 Context Handler

Functionality for Abstraction of Thread Context Management.

## Files

- file context.h

  *Information About Each Thread's Context.*

## Macros

- #define CONTEXT_FP_SIZE

  *Size of floating point context area.*
- #define _Context_Initialize(_the_context, _stack, _size, _isr, _entry, _is_fp, _tls_area)

  *Initialize context area.*
- #define _Context_Initialization_at_thread_begin()
- #define _Context_Switch(_executing, _heir) _CPU_Context_switch( _executing, _heir )

  *Perform context switch.*
- #define _Context_Restart_self(_the_context) _CPU_Context_Restart_self( _the_context )

  *Restart currently executing thread.*
- #define _Context_Initialize_fp(_fp_area) _CPU_Context_Initialize_fp( _fp_area )

  *Initialize floating point context area.*
- #define _Context_Restore_fp(_fp) _CPU_Context_restore_fp( _fp )

  *Restore floating point context area.*
- #define _Context_Save_fp(_fp) _CPU_Context_save_fp( _fp )

  *Save floating point context area.*
- #define **_Context_Destroy**(_the_thread, _the_context)

## 8.48.1 Detailed Description

Functionality for Abstraction of Thread Context Management.

This handler encapsulates functionality which abstracts thread context management in a portable manner.

The context switch needed variable is contained in the per cpu data structure.

## 8.48.2 Macro Definition Documentation

### 8.48.2.1 _Context_Initialization_at_thread_begin

```
#define _Context_Initialization_at_thread_begin( )
```

This macro is invoked from _Thread_Handler to do whatever CPU specific magic is required that must be done in the context of the thread when it starts.

If the CPU architecture does not require any magic, then this macro is empty.

Definition at line 94 of file context.h.

#### 8.48.2.2 _Context_Initialize

```
#define _Context_Initialize(
              _the_context,
              _stack,
              _size,
              _isr,
              _entry,
              _is_fp,
              _tls_area )
```

**Value:**

```
_CPU_Context_Initialize( _the_context, _stack, _size, _isr, _entry, \
  _is_fp, _tls_area )
```

Initialize context area.

This routine initializes *_the_context* such that the stack pointer, interrupt level, and entry point are correct for the thread's initial state.

**Parameters**

| | | |
|---|---|---|
| in | *_the_context* | will be initialized |
| in | *_stack* | is the lowest physical address of the thread's context |
| in | *_size* | is the size in octets of the thread's context |
| in | *_isr* | is the ISR enable level for this thread |
| in | *_entry* | is this thread's entry point |
| in | *_is_fp* | is set to true if this thread has floating point enabled |
| in | *_tls_area* | The thread-local storage (TLS) area begin. |

Definition at line 76 of file context.h.

#### 8.48.2.3 _Context_Initialize_fp

```
#define _Context_Initialize_fp(
              _fp_area )  _CPU_Context_Initialize_fp( _fp_area )
```

Initialize floating point context area.

This routine initializes the floating point context save area to contain an initial known state.

**Parameters**

| | | |
|---|---|---|
| in | *_fp_area* | is the base address of the floating point context save area to initialize. |

Definition at line 129 of file context.h.

#### 8.48.2.4 _Context_Restart_self

```
#define _Context_Restart_self(
              _the_context )  _CPU_Context_Restart_self( _the_context )
```

Restart currently executing thread.

This routine restarts the calling thread by restoring its initial stack pointer and returning to the thread's entry point.

**Parameters**

| | | |
|---|---|---|
| in | *_the_context* | is the context of the thread to restart |

Definition at line 117 of file context.h.

#### 8.48.2.5 _Context_Restore_fp

```
#define _Context_Restore_fp(
              _fp )  _CPU_Context_restore_fp( _fp )
```

Restore floating point context area.

This routine restores the floating point context contained in the *_fp* area. It is assumed that the current floating point context has been saved by a previous invocation of *_Context_Save_fp*.

**Parameters**

| | | |
|---|---|---|
| in | ↩ _↩ *fp* | points to the floating point context area to restore. |

Definition at line 142 of file context.h.

#### 8.48.2.6 _Context_Save_fp

```
#define _Context_Save_fp(
              _fp )  _CPU_Context_save_fp( _fp )
```

Save floating point context area.

This routine saves the current floating point context in the *_fp* area.

**Parameters**

| | | |
|---|---|---|
| in | ↩ _↩ *fp* | points to the floating point context area to restore. |

Definition at line 153 of file context.h.

#### 8.48.2.7 _Context_Switch

```
#define _Context_Switch(
              _executing,
              _heir )  _CPU_Context_switch( _executing, _heir )
```

Perform context switch.

This routine saves the current context into the *_executing* context record and restores the context specified by *_heir*.

**Parameters**

| | | |
|---|---|---|
| in | *_executing* | is the currently executing thread's context |
| in | *_heir* | is the context of the thread to be switched to |

Definition at line 106 of file context.h.

#### 8.48.2.8 CONTEXT_FP_SIZE

```
#define CONTEXT_FP_SIZE
```

**Value:**

```
( ( CPU_CONTEXT_FP_SIZE + CPU_HEAP_ALIGNMENT - 1 ) \
  & ~( CPU_HEAP_ALIGNMENT - 1 ) )
```

Size of floating point context area.

This constant defines the number of bytes required to store a full floating point context.

Definition at line 52 of file context.h.

# 8.49 DEFAULT_INITIAL_EXTENSION Support

DEFAULT_INITIAL_EXTENSION Support Package.

## Files

- file default-initial-extension.h

  *DEFAULT_INITIAL_EXTENSION Support.*

### 8.49.1 Detailed Description

DEFAULT_INITIAL_EXTENSION Support Package.

## 8.50 Destructors

Support to run destructors at the end of a test case.

### Classes

- struct T_destructor

### Typedefs

- typedef struct T_destructor **T_destructor**

### Functions

- void **T_add_destructor** (T_destructor *, void(*)(T_destructor *))
- void **T_remove_destructor** (T_destructor *)

### 8.50.1 Detailed Description

Support to run destructors at the end of a test case.

# 8.51 Deterministic Priority Scheduler

Deterministic Priority Scheduler.

## Files

- file schedulerpriority.h

  *Thread Manipulation with the Priority-Based Scheduler.*
- file schedulerprorityimpl.h

  *Inlined Routines Associated with the Manipulation of the Priority-Based Scheduling Structures.*

## Classes

- struct Scheduler_priority_Context
- struct Scheduler_priority_Ready_queue

  *Data for ready queue operations.*
- struct Scheduler_priority_Node

  *Scheduler node specialization for Deterministic Priority schedulers.*

## Macros

- #define SCHEDULER_PRIORITY_ENTRY_POINTS

## Functions

- void _Scheduler_priority_Initialize (const Scheduler_Control *scheduler)

  *Initializes the priority scheduler.*
- void _Scheduler_priority_Block (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Blocks the thread.*
- void _Scheduler_priority_Schedule (const Scheduler_Control *scheduler, Thread_Control *the_thread)

  *Sets the heir thread to be the next ready thread.*
- void _Scheduler_priority_Unblock (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Unblocks the thread.*
- void _Scheduler_priority_Update_priority (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *base_node)

  *Updates the priority of the node.*
- void _Scheduler_priority_Node_initialize (const Scheduler_Control *scheduler, Scheduler_Node *node, Thread_Control *the_thread, Priority_Control priority)

  *Initializes the node with the given priority.*
- void _Scheduler_priority_Yield (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Performs the yield of a thread.*
- static __inline__ Scheduler_priority_Context * _Scheduler_priority_Get_context (const Scheduler_Control *scheduler)

  *Gets the context of the scheduler.*

- static __inline__ Scheduler_priority_Node * _Scheduler_priority_Thread_get_node (Thread_Control *the↩ _thread)

  *Gets the scheduler node of the thread.*
- static __inline__ Scheduler_priority_Node * _Scheduler_priority_Node_downcast (Scheduler_Node *node)

  *Gets the priority node of the scheduler node.*
- static __inline__ void _Scheduler_priority_Ready_queue_initialize (Chain_Control *ready_queues, Priority_Control maximum_priority)

  *Ready queue initialization.*
- static __inline__ void _Scheduler_priority_Ready_queue_enqueue (Chain_Node *node, Scheduler_priority_Ready_queue *ready_queue, Priority_bit_map_Control *bit_map)

  *Enqueues a node on the specified ready queue.*
- static __inline__ void _Scheduler_priority_Ready_queue_enqueue_first (Chain_Node *node, Scheduler_priority_Ready_queue *ready_queue, Priority_bit_map_Control *bit_map)

  *Enqueues a node on the specified ready queue as first.*
- static __inline__ void _Scheduler_priority_Ready_queue_extract (Chain_Node *node, Scheduler_priority_Ready_queue *ready_queue, Priority_bit_map_Control *bit_map)

  *Extracts a node from the specified ready queue.*
- static __inline__ void _Scheduler_priority_Extract_body (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Extracts a node from the context of the scheduler.*
- static __inline__ Chain_Node * _Scheduler_priority_Ready_queue_first (Priority_bit_map_Control *bit_map, Chain_Control *ready_queues)

  *Returns a pointer to the first node.*
- static __inline__ void _Scheduler_priority_Schedule_body (const Scheduler_Control *scheduler, Thread_Control *the_thread, bool force_dispatch)

  *Scheduling decision logic.*
- static __inline__ void _Scheduler_priority_Ready_queue_update (Scheduler_priority_Ready_queue *ready_queue, unsigned int new_priority, Priority_bit_map_Control *bit_map, Chain_Control *ready_↩ queues)

  *Updates the specified ready queue data according to the new priority value.*

## 8.51.1 Detailed Description

Deterministic Priority Scheduler.

## 8.51.2 Macro Definition Documentation

### 8.51.2.1 SCHEDULER_PRIORITY_ENTRY_POINTS

```
#define SCHEDULER_PRIORITY_ENTRY_POINTS
```

**Value:**

```
{ \
  _Scheduler_priority_Initialize,       /* initialize entry point */ \
  _Scheduler_priority_Schedule,         /* schedule entry point */ \
  _Scheduler_priority_Yield,            /* yield entry point */ \
  _Scheduler_priority_Block,            /* block entry point */ \
  _Scheduler_priority_Unblock,          /* unblock entry point */ \
  _Scheduler_priority_Update_priority,  /* update priority entry point */ \
  _Scheduler_default_Map_priority,      /* map priority entry point */ \
  _Scheduler_default_Unmap_priority,    /* unmap priority entry point */ \
```

```
    SCHEDULER_OPERATION_DEFAULT_ASK_FOR_HELP \
    _Scheduler_priority_Node_initialize,  /* node initialize entry point */ \
    _Scheduler_default_Node_destroy,      /* node destroy entry point */ \
    _Scheduler_default_Release_job,       /* new period of task */ \
    _Scheduler_default_Cancel_job,        /* cancel period of task */ \
    _Scheduler_default_Tick,              /* tick entry point */ \
    _Scheduler_default_Start_idle         /* start idle entry point */ \
    SCHEDULER_OPERATION_DEFAULT_GET_SET_AFFINITY \
  }
```

Entry points for the Deterministic Priority Based Scheduler.

Definition at line 45 of file schedulerpriority.h.

## 8.51.3 Function Documentation

### 8.51.3.1 _Scheduler_priority_Block()

```
void _Scheduler_priority_Block (
             const Scheduler_Control * scheduler,
             Thread_Control * the_thread,
             Scheduler_Node * node )
```

Blocks the thread.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| in,out | *the_thread* | The thread to block. |
| in,out | *node* | The *thread's* scheduler node. |

Definition at line 26 of file schedulerpriorityblock.c.

### 8.51.3.2 _Scheduler_priority_Extract_body()

```
static __inline__ void _Scheduler_priority_Extract_body (
             const Scheduler_Control * scheduler,
             Thread_Control * the_thread,
             Scheduler_Node * node )  [static]
```

Extracts a node from the context of the scheduler.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| | *the_thread* | The thread of which the node will be extracted. |
| in,out | *The* | node which preserves the ready queue. |

Definition at line 175 of file schedulerpriorityimpl.h.

#### 8.51.3.3 _Scheduler_priority_Get_context()

```
static __inline__ Scheduler_priority_Context* _Scheduler_priority_Get_context (
            const Scheduler_Control * scheduler )  [static]
```

Gets the context of the scheduler.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler to get the context of. |

**Returns**

The context of the scheduler.

Definition at line 49 of file schedulerpriorityimpl.h.

#### 8.51.3.4 _Scheduler_priority_Initialize()

```
void _Scheduler_priority_Initialize (
            const Scheduler_Control * scheduler )
```

Initializes the priority scheduler.

This routine initializes the priority scheduler.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler to initialize. |

Definition at line 23 of file schedulerpriority.c.

#### 8.51.3.5 _Scheduler_priority_Node_downcast()

```
static __inline__ Scheduler_priority_Node* _Scheduler_priority_Node_downcast (
            Scheduler_Node * node )  [static]
```

Gets the priority node of the scheduler node.

**Parameters**

| | |
|---|---|
| *node* | The node to get the priority node of. |

**Returns**

The priority node.

Definition at line 75 of file schedulerprorityimpl.h.

#### 8.51.3.6 _Scheduler_priority_Node_initialize()

```
void _Scheduler_priority_Node_initialize (
              const Scheduler_Control * scheduler,
              Scheduler_Node * node,
              Thread_Control * the_thread,
              Priority_Control priority )
```

Initializes the node with the given priority.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| out | *node* | The node to initialize. |
| | *the_thread* | The thread of the scheduler node. |
| | *priority* | The priority for the initialization. |

Definition at line 35 of file schedulerpriority.c.

#### 8.51.3.7 _Scheduler_priority_Ready_queue_enqueue()

```
static __inline__ void _Scheduler_priority_Ready_queue_enqueue (
              Chain_Node * node,
              Scheduler_priority_Ready_queue * ready_queue,
              Priority_bit_map_Control * bit_map )  [static]
```

Enqueues a node on the specified ready queue.

The node is placed as the last element of its priority group.

**Parameters**

| | | |
|---|---|---|
| | *node* | The node to enqueue. |
| in,out | *ready_queue* | The ready queue. |
| out | *bit_map* | The priority bit map of the scheduler instance. |

Definition at line 111 of file schedulerpriorityimpl.h.

#### 8.51.3.8 _Scheduler_priority_Ready_queue_enqueue_first()

```
static __inline__ void _Scheduler_priority_Ready_queue_enqueue_first (
            Chain_Node * node,
            Scheduler_priority_Ready_queue * ready_queue,
            Priority_bit_map_Control * bit_map )  [static]
```

Enqueues a node on the specified ready queue as first.

The node is placed as the first element of its priority group.

**Parameters**

| | | |
|---|---|---|
| | *node* | The node to enqueue as first. |
| in,out | *ready_queue* | The ready queue. |
| out | *bit_map* | The priority bit map of the scheduler instance. |

Definition at line 132 of file schedulerpriorityimpl.h.

#### 8.51.3.9 _Scheduler_priority_Ready_queue_extract()

```
static __inline__ void _Scheduler_priority_Ready_queue_extract (
            Chain_Node * node,
            Scheduler_priority_Ready_queue * ready_queue,
            Priority_bit_map_Control * bit_map )  [static]
```

Extracts a node from the specified ready queue.

**Parameters**

| | | |
|---|---|---|
| | *node* | The node to extract. |
| in,out | *ready_queue* | The ready queue. |
| out | *bit_map* | The priority bit map of the scheduler instance. |

Definition at line 151 of file schedulerpriorityimpl.h.

#### 8.51.3.10 _Scheduler_priority_Ready_queue_first()

```
static __inline__ Chain_Node* _Scheduler_priority_Ready_queue_first (
            Priority_bit_map_Control * bit_map,
            Chain_Control * ready_queues )  [static]
```

Returns a pointer to the first node.

This routines returns a pointer to the first node on *ready_queues*.

**Parameters**

| | |
|---|---|
| *bit_map* | The priority bit map of the scheduler instance. |
| *ready_queues* | The ready queues of the scheduler instance. |

**Returns**

This method returns the first node.

Definition at line 204 of file schedulerpriorityimpl.h.

#### 8.51.3.11 _Scheduler_priority_Ready_queue_initialize()

```
static __inline__ void _Scheduler_priority_Ready_queue_initialize (
            Chain_Control * ready_queues,
            Priority_Control maximum_priority )  [static]
```

Ready queue initialization.

This routine initializes *ready_queues* for priority-based scheduling.

**Parameters**

| | | |
|---|---|---|
| out | *ready_queues* | The ready queue to initialize. |
| | *maximum_priority* | The maximum priority in the ready queue. |

Definition at line 90 of file schedulerpriorityimpl.h.

#### 8.51.3.12 _Scheduler_priority_Ready_queue_update()

```
static __inline__ void _Scheduler_priority_Ready_queue_update (
            Scheduler_priority_Ready_queue * ready_queue,
            unsigned int new_priority,
            Priority_bit_map_Control * bit_map,
            Chain_Control * ready_queues )  [static]
```

Updates the specified ready queue data according to the new priority value.

**Parameters**

| | | |
|---|---|---|
| in,out | *ready_queue* | The ready queue. |
| | *new_priority* | The new priority. |
| | *bit_map* | The priority bit map of the scheduler instance. |
| | *ready_queues* | The ready queues of the scheduler instance. |

Definition at line 256 of file schedulerpriorityimpl.h.

#### 8.51.3.13 _Scheduler_priority_Schedule()

```
void _Scheduler_priority_Schedule (
             const Scheduler_Control * scheduler,
             Thread_Control * the_thread )
```

Sets the heir thread to be the next ready thread.

This kernel routine sets the heir thread to be the next ready thread by invoking the_scheduler->ready_queue->operations->first().

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance. |
| *the_thread* | The thread for the operation. |

Definition at line 24 of file schedulerpriorityschedule.c.

#### 8.51.3.14 _Scheduler_priority_Schedule_body()

```
static __inline__ void _Scheduler_priority_Schedule_body (
             const Scheduler_Control * scheduler,
             Thread_Control * the_thread,
             bool force_dispatch )  [static]
```

Scheduling decision logic.

This kernel routine implements scheduling decision logic for priority-based scheduling.

**Parameters**

| | | |
|---|---|---|
| in,out | *scheduler* | The scheduler instance. |
| | *the_thread* | This parameter is unused. |
| | *force_dispatch* | Indicates whether the dispatch happens also if the currently executing thread is set as not preemptible. |

Definition at line 228 of file schedulerpriorityimpl.h.

#### 8.51.3.15 _Scheduler_priority_Thread_get_node()

```
static __inline__ Scheduler_priority_Node* _Scheduler_priority_Thread_get_node (
             Thread_Control * the_thread )  [static]
```

Gets the scheduler node of the thread.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to get the scheduler node of. |

**Returns**

The scheduler node of *the_thread*.

Definition at line 61 of file schedulerpriorityimpl.h.

#### 8.51.3.16 _Scheduler_priority_Unblock()

```
void _Scheduler_priority_Unblock (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread,
              Scheduler_Node * node )
```

Unblocks the thread.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| in,out | *the_thread* | The thread to unblock. |
| in,out | *node* | The *thread's* scheduler node. |

Definition at line 25 of file schedulerpriorityunblock.c.

#### 8.51.3.17 _Scheduler_priority_Update_priority()

```
void _Scheduler_priority_Update_priority (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread,
              Scheduler_Node * base_node )
```

Updates the priority of the node.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance. |
| *the_thread* | The thread for the operation. |
| *base_node* | The thread's scheduler node. |

Definition at line 24 of file schedulerprioritychangepriority.c.

#### 8.51.3.18 _Scheduler_priority_Yield()

```
void _Scheduler_priority_Yield (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread,
              Scheduler_Node * node )
```

Performs the yield of a thread.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| in,out | *the_thread* | The thread that performed the yield operation. |
| | *node* | The scheduler node of *the_thread*. |

Definition at line 24 of file schedulerpriorityyield.c.

## 8.52 Device Driver Configuration

### Macros

- #define CONFIGURE_APPLICATION_DOES_NOT_NEED_CLOCK_DRIVER

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_APPLICATION_EXTRA_DRIVERS

  *This configuration option is an initializer define.*
- #define CONFIGURE_APPLICATION_NEEDS_ATA_DRIVER

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_APPLICATION_NEEDS_FRAME_BUFFER_DRIVER

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_APPLICATION_NEEDS_IDE_DRIVER

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_APPLICATION_NEEDS_NULL_DRIVER

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_APPLICATION_NEEDS_RTC_DRIVER

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_APPLICATION_NEEDS_SIMPLE_CONSOLE_DRIVER

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_APPLICATION_NEEDS_SIMPLE_TASK_CONSOLE_DRIVER

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_APPLICATION_NEEDS_STUB_DRIVER

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_APPLICATION_NEEDS_TIMER_DRIVER

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_APPLICATION_NEEDS_WATCHDOG_DRIVER

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_APPLICATION_NEEDS_ZERO_DRIVER

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_APPLICATION_PREREQUISITE_DRIVERS

  *This configuration option is an initializer define.*
- #define CONFIGURE_ATA_DRIVER_TASK_PRIORITY

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_DRIVERS

  *This configuration option is an integer define.*

### 8.52.1 Detailed Description

This section describes configuration options related to the device drivers. Note that network device drivers are not covered by the following options.

### 8.52.2 Macro Definition Documentation

### 8.52.2.1 CONFIGURE_APPLICATION_DOES_NOT_NEED_CLOCK_DRIVER

```
#define CONFIGURE_APPLICATION_DOES_NOT_NEED_CLOCK_DRIVER
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then **no** Clock Driver is initialized during system initialization.

#### Default Configuration

If this configuration option is undefined, then a Clock Driver may be initialized during system initialization.

#### Notes

This configuration parameter is intended to prevent the common user error of using the Hello World example as the baseline for an application and leaving out a clock tick source.

The application shall define exactly one of the following configuration options

- CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER,
- CONFIGURE_APPLICATION_DOES_NOT_NEED_CLOCK_DRIVER, or
- CONFIGURE_APPLICATION_NEEDS_TIMER_DRIVER,

otherwise a compile time error in the configuration file will occur.

Definition at line 1226 of file appl-config.h.

### 8.52.2.2 CONFIGURE_APPLICATION_EXTRA_DRIVERS

```
#define CONFIGURE_APPLICATION_EXTRA_DRIVERS
```

This configuration option is an initializer define.

The value of this configuration option is used to initialize the Device Driver Table.

#### Default Value

The default value is the empty list.

#### Value Constraints

The value of this configuration option shall be a list of initializers for structures of type rtems_driver_address_table.

#### Notes

The value of this configuration option is placed after the entries of other device driver configuration options.

See CONFIGURE_APPLICATION_PREREQUISITE_DRIVERS for an alternative placement of application device driver initializers.

Definition at line 1252 of file appl-config.h.

#### 8.52.2.3 CONFIGURE_APPLICATION_NEEDS_ATA_DRIVER

```
#define CONFIGURE_APPLICATION_NEEDS_ATA_DRIVER
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the ATA Driver is initialized during system initialization.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

Most BSPs do not include support for an ATA Driver.

If this option is defined and the BSP does not have this device driver, then the user will get a link time error for an undefined symbol.

Definition at line 1274 of file appl-config.h.

#### 8.52.2.4 CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER

```
#define CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the Clock Driver is initialized during system initialization.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

The Clock Driver is responsible for providing a regular interrupt which invokes a clock tick directive.

The application shall define exactly one of the following configuration options

- CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER,
- CONFIGURE_APPLICATION_DOES_NOT_NEED_CLOCK_DRIVER, or
- CONFIGURE_APPLICATION_NEEDS_TIMER_DRIVER,

otherwise a compile time error in the configuration file will occur.

Definition at line 1305 of file appl-config.h.

#### 8.52.2.5 CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER

```
#define CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the Console Driver is initialized during system initialization.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

The Console Driver is responsible for providing the `/dev/console` device file. This device is used to initialize the standard input, output, and error file descriptors.

BSPs should be constructed in a manner that allows printk() to work properly without the need for the Console Driver to be configured.

The

- `CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER`,
- CONFIGURE_APPLICATION_NEEDS_SIMPLE_CONSOLE_DRIVER, and
- CONFIGURE_APPLICATION_NEEDS_SIMPLE_TASK_CONSOLE_DRIVER

configuration options are mutually exclusive.

Definition at line 1339 of file appl-config.h.

#### 8.52.2.6 CONFIGURE_APPLICATION_NEEDS_FRAME_BUFFER_DRIVER

```
#define CONFIGURE_APPLICATION_NEEDS_FRAME_BUFFER_DRIVER
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the Frame Buffer Driver is initialized during system initialization.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

Most BSPs do not include support for a Frame Buffer Driver. This is because many boards do not include the required hardware.

If this option is defined and the BSP does not have this device driver, then the user will get a link time error for an undefined symbol.

Definition at line 1362 of file appl-config.h.

#### 8.52.2.7 CONFIGURE_APPLICATION_NEEDS_IDE_DRIVER

```
#define CONFIGURE_APPLICATION_NEEDS_IDE_DRIVER
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the IDE Driver is initialized during system initialization.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

Most BSPs do not include support for an IDE Driver.

If this option is defined and the BSP does not have this device driver, then the user will get a link time error for an undefined symbol.

Definition at line 1384 of file appl-config.h.

#### 8.52.2.8 CONFIGURE_APPLICATION_NEEDS_NULL_DRIVER

```
#define CONFIGURE_APPLICATION_NEEDS_NULL_DRIVER
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the `/dev/null` Driver is initialized during system initialization.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

This device driver is supported by all BSPs.

Definition at line 1401 of file appl-config.h.

#### 8.52.2.9 CONFIGURE_APPLICATION_NEEDS_RTC_DRIVER

```
#define CONFIGURE_APPLICATION_NEEDS_RTC_DRIVER
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the Real-Time Clock Driver is initialized during system initialization.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

Most BSPs do not include support for a real-time clock (RTC). This is because many boards do not include the required hardware.

If this is defined and the BSP does not have this device driver, then the user will get a link time error for an undefined symbol.

Definition at line 1424 of file appl-config.h.

#### 8.52.2.10 CONFIGURE_APPLICATION_NEEDS_SIMPLE_CONSOLE_DRIVER

```
#define CONFIGURE_APPLICATION_NEEDS_SIMPLE_CONSOLE_DRIVER
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the Simple Console Driver is initialized during system initialization.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

This device driver is responsible for providing the `/dev/console` device file. This device is used to initialize the standard input, output, and error file descriptors.

This device driver reads via getchark().

This device driver writes via rtems_putc().

The Termios framework is not used. There is no support to change device settings, e.g. baud, stop bits, parity, etc.

The

- CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER,
- `CONFIGURE_APPLICATION_NEEDS_SIMPLE_CONSOLE_DRIVER`, and
- CONFIGURE_APPLICATION_NEEDS_SIMPLE_TASK_CONSOLE_DRIVER

configuration options are mutually exclusive.

Definition at line 1462 of file appl-config.h.

#### 8.52.2.11 CONFIGURE_APPLICATION_NEEDS_SIMPLE_TASK_CONSOLE_DRIVER

```
#define CONFIGURE_APPLICATION_NEEDS_SIMPLE_TASK_CONSOLE_DRIVER
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the Simple Task Console Driver is initialized during system initialization.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

This device driver is responsible for providing the `/dev/console` device file. This device is used to initialize the standard input, output, and error file descriptors.

This device driver reads via getchark().

This device driver writes into a write buffer. The count of characters written into the write buffer is returned. It might be less than the requested count, in case the write buffer is full. The write is non-blocking and may be called from interrupt context. A dedicated task reads from the write buffer and outputs the characters via rtems_putc(). This task runs with the least important priority. The write buffer size is 2047 characters and it is not configurable.

Use `fsync( STDOUT_FILENO )` or `fdatasync( STDOUT_FILENO )` to drain the write buffer.

The Termios framework is not used. There is no support to change device settings, e.g. baud, stop bits, parity, etc.

The

- CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER,
- CONFIGURE_APPLICATION_NEEDS_SIMPLE_CONSOLE_DRIVER, and
- `CONFIGURE_APPLICATION_NEEDS_SIMPLE_TASK_CONSOLE_DRIVER`

configuration options are mutually exclusive.

Definition at line 1509 of file appl-config.h.

#### 8.52.2.12 CONFIGURE_APPLICATION_NEEDS_STUB_DRIVER

```
#define CONFIGURE_APPLICATION_NEEDS_STUB_DRIVER
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the Stub Driver is initialized during system initialization.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

This device driver simply provides entry points that return successful and is primarily a test fixture. It is supported by all BSPs.

Definition at line 1527 of file appl-config.h.

#### 8.52.2.13 CONFIGURE_APPLICATION_NEEDS_TIMER_DRIVER

```
#define CONFIGURE_APPLICATION_NEEDS_TIMER_DRIVER
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the Benchmark Timer Driver is initialized during system initialization.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

The Benchmark Timer Driver is intended for the benchmark tests of the RTEMS Testsuite. Applications should not use this driver.

The application shall define exactly one of the following configuration options

- CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER,
- CONFIGURE_APPLICATION_DOES_NOT_NEED_CLOCK_DRIVER, or
- CONFIGURE_APPLICATION_NEEDS_TIMER_DRIVER,

otherwise a compile time error will occur.

Definition at line 1558 of file appl-config.h.

#### 8.52.2.14 CONFIGURE_APPLICATION_NEEDS_WATCHDOG_DRIVER

```
#define CONFIGURE_APPLICATION_NEEDS_WATCHDOG_DRIVER
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the Watchdog Driver is initialized during system initialization.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

Most BSPs do not include support for a watchdog device driver. This is because many boards do not include the required hardware.

If this is defined and the BSP does not have this device driver, then the user will get a link time error for an undefined symbol.

Definition at line 1581 of file appl-config.h.

#### 8.52.2.15 CONFIGURE_APPLICATION_NEEDS_ZERO_DRIVER

```
#define CONFIGURE_APPLICATION_NEEDS_ZERO_DRIVER
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the `/dev/zero` Driver is initialized during system initialization.

##### Default Configuration

If this configuration option is undefined, then the described feature is not enabled.

##### Notes

This device driver is supported by all BSPs.

Definition at line 1598 of file appl-config.h.

#### 8.52.2.16 CONFIGURE_APPLICATION_PREREQUISITE_DRIVERS

```
#define CONFIGURE_APPLICATION_PREREQUISITE_DRIVERS
```

This configuration option is an initializer define.

The value of this configuration option is used to initialize the Device Driver Table.

##### Default Value

The default value is the empty list.

##### Value Constraints

The value of this configuration option shall be a list of initializers for structures of type rtems_driver_address_table.

##### Notes

The value of this configuration option is placed after the entries defined by CONFIGURE_BSP_PREREQUISITE_DRIVERS and before all other device driver configuration options.

See CONFIGURE_APPLICATION_EXTRA_DRIVERS for an alternative placement of application device driver initializers.

Definition at line 1625 of file appl-config.h.

#### 8.52.2.17 CONFIGURE_ATA_DRIVER_TASK_PRIORITY

```
#define CONFIGURE_ATA_DRIVER_TASK_PRIORITY
```

This configuration option is an integer define.

The value of this configuration option defines the ATA task priority.

**Default Value**

The default value is 140.

**Value Constraints**

The value of this configuration option shall be a valid Classic API task priority. The set of valid task priorities is scheduler-specific.

**Notes**

This configuration option is only evaluated if the configuration option CONFIGURE_APPLICATION_NEEDS_ATA_DRIVER is defined.

Definition at line 1645 of file appl-config.h.

#### 8.52.2.18 CONFIGURE_MAXIMUM_DRIVERS

```
#define CONFIGURE_MAXIMUM_DRIVERS
```

This configuration option is an integer define.

The value of this configuration option defines the number of device drivers.

**Default Value**

This is computed by default, and is set to the number of statically configured device drivers configured using the following configuration options:

- CONFIGURE_APPLICATION_EXTRA_DRIVERS
- CONFIGURE_APPLICATION_NEEDS_ATA_DRIVER
- CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER
- CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER
- CONFIGURE_APPLICATION_NEEDS_FRAME_BUFFER_DRIVER
- CONFIGURE_APPLICATION_NEEDS_IDE_DRIVER
- CONFIGURE_APPLICATION_NEEDS_LIBBLOCK
- CONFIGURE_APPLICATION_NEEDS_NULL_DRIVER
- CONFIGURE_APPLICATION_NEEDS_RTC_DRIVER
- CONFIGURE_APPLICATION_NEEDS_SIMPLE_CONSOLE_DRIVER
- CONFIGURE_APPLICATION_NEEDS_SIMPLE_TASK_CONSOLE_DRIVER
- CONFIGURE_APPLICATION_NEEDS_STUB_DRIVER
- CONFIGURE_APPLICATION_NEEDS_TIMER_DRIVER
- CONFIGURE_APPLICATION_NEEDS_WATCHDOG_DRIVER
- CONFIGURE_APPLICATION_NEEDS_ZERO_DRIVER
- CONFIGURE_APPLICATION_PREREQUISITE_DRIVERS
- CONFIGURE_BSP_PREREQUISITE_DRIVERS

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be less than or equal to SIZE_MAX.
- It shall be greater than or equal than the number of statically configured device drivers.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.

#### Notes

If the application will dynamically install device drivers, then the configuration option value shall be larger than the number of statically configured device drivers.

Definition at line 1716 of file appl-config.h.

## 8.53 Device Drivers

This group contains the device drivers.

### Modules

- Clock Driver

  *The Clock Driver API.*
- Legacy Benchmark Drivers

  *Legacy Benchmark Drivers.*
- Serial Mouse

  *Serial Mouse.*
- Time Test 27 Support

  *Time Test 27 Support.*

### 8.53.1 Detailed Description

This group contains the device drivers.

## 8.54 Directive Attributes

This group contains the Classic API directive attributes.

### Macros

- #define RTEMS_APPLICATION_TASK 0x00000000

  *This attribute constant indicates that the Classic API task created by rtems_task_create() or rtems_task_construct() shall be an application task.*
- #define RTEMS_BARRIER_AUTOMATIC_RELEASE 0x00000200

  *This attribute constant indicates that the Classic API barrier created by rtems_barrier_create() shall use the automatic release protocol.*
- #define RTEMS_BARRIER_MANUAL_RELEASE 0x00000000

  *This attribute constant indicates that the Classic API barrier created by rtems_barrier_create() shall use the manual release protocol.*
- #define RTEMS_BINARY_SEMAPHORE 0x00000010

  *This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() shall be a proper binary semaphore or mutex.*
- #define RTEMS_COUNTING_SEMAPHORE 0x00000000

  *This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() shall be a counting semaphore.*
- #define RTEMS_DEFAULT_ATTRIBUTES 0x00000000

  *This attribute constant represents the default attribute set.*
- #define RTEMS_FIFO 0x00000000

  *This attribute constant indicates that the Classic API object shall manage blocking tasks using the FIFO discipline.*
- #define RTEMS_FLOATING_POINT 0x00000001

  *This attribute constant indicates that the Classic API task created by rtems_task_create() or rtems_task_construct() shall be able to use the floating point hardware.*
- #define RTEMS_GLOBAL 0x00000002

  *This attribute constant indicates that the Classic API object shall be a global resource in a multiprocessing network.*
- #define RTEMS_INHERIT_PRIORITY 0x00000040

  *This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() shall use the Priority Inheritance Protocol.*
- #define RTEMS_LOCAL 0x00000000

  *This attribute constant indicates that the Classic API object shall be a local resource in a multiprocessing network.*
- #define RTEMS_MULTIPROCESSOR_RESOURCE_SHARING 0x00000100

  *This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() shall use the Multiprocessor Resource Sharing Protocol.*
- #define RTEMS_NO_FLOATING_POINT 0x00000000

  *This attribute constant indicates that the Classic API task created by rtems_task_create() or rtems_task_construct() will not use the floating point hardware.*
- #define RTEMS_NO_INHERIT_PRIORITY 0x00000000

  *This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() will not use the Priority Inheritance Protocol.*
- #define RTEMS_NO_MULTIPROCESSOR_RESOURCE_SHARING 0x00000000

  *This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() will not use the Multiprocessor Resource Sharing Protocol.*
- #define RTEMS_NO_PRIORITY_CEILING 0x00000000

  *This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() will not use the Priority Ceiling Protocol.*
- #define RTEMS_PRIORITY 0x00000004

*This attribute constant indicates that the Classic API object shall manage blocking tasks using the task priority discipline.*

- #define RTEMS_PRIORITY_CEILING 0x00000080

  *This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() shall use the Priority Ceiling Protocol.*

- #define RTEMS_SEMAPHORE_CLASS 0x00000030

  *This attribute constant represents the mask of bits associated with the Classic API semaphore classes RTEMS_BINARY_SEMAPHORE, RTEMS_COUNTING_SEMAPHORE, and RTEMS_SIMPLE_BINARY_SEMAPHORE.*

- #define RTEMS_SIMPLE_BINARY_SEMAPHORE 0x00000020

  *This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() shall be a simple binary semaphore.*

- #define RTEMS_SYSTEM_TASK 0x00008000

  *This attribute constant indicates that the Classic API task created by rtems_task_create() or rtems_task_construct() shall be a system task.*

## Typedefs

- typedef uint32_t rtems_attribute

  *This type represents Classic API attributes.*

### 8.54.1 Detailed Description

This group contains the Classic API directive attributes.

### 8.54.2 Macro Definition Documentation

#### 8.54.2.1 RTEMS_FLOATING_POINT

```
#define RTEMS_FLOATING_POINT 0x00000001
```

This attribute constant indicates that the Classic API task created by rtems_task_create() or rtems_task_construct() shall be able to use the floating point hardware.

On some architectures, there will be a floating point context associated with this task.

Definition at line 165 of file attr.h.

#### 8.54.2.2 RTEMS_GLOBAL

```
#define RTEMS_GLOBAL 0x00000002
```

This attribute constant indicates that the Classic API object shall be a global resource in a multiprocessing network.

This attribute does not refer to SMP systems.

Definition at line 177 of file attr.h.

#### 8.54.2.3 RTEMS_INHERIT_PRIORITY

```
#define RTEMS_INHERIT_PRIORITY 0x00000040
```

This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() shall use the Priority Inheritance Protocol.

The semaphore shall be a binary semaphore (RTEMS_BINARY_SEMAPHORE).

Definition at line 190 of file attr.h.

#### 8.54.2.4 RTEMS_LOCAL

```
#define RTEMS_LOCAL 0x00000000
```

This attribute constant indicates that the Classic API object shall be a local resource in a multiprocessing network.

This attribute does not refer to SMP systems.

Definition at line 202 of file attr.h.

#### 8.54.2.5 RTEMS_MULTIPROCESSOR_RESOURCE_SHARING

```
#define RTEMS_MULTIPROCESSOR_RESOURCE_SHARING 0x00000100
```

This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() shall use the Multiprocessor Resource Sharing Protocol.

The semaphore shall be a binary semaphore (RTEMS_BINARY_SEMAPHORE).

Definition at line 215 of file attr.h.

#### 8.54.2.6 RTEMS_NO_FLOATING_POINT

```
#define RTEMS_NO_FLOATING_POINT 0x00000000
```

This attribute constant indicates that the Classic API task created by rtems_task_create() or rtems_task_construct() will not use the floating point hardware.

If the architecture permits it, then the FPU will be disabled when the task is executing.

Definition at line 229 of file attr.h.

#### 8.54.2.7 RTEMS_PRIORITY_CEILING

```
#define RTEMS_PRIORITY_CEILING 0x00000080
```

This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() shall use the Priority Ceiling Protocol.

The semaphore shall be a binary semaphore (RTEMS_BINARY_SEMAPHORE).

Definition at line 285 of file attr.h.

### 8.54.3 Typedef Documentation

#### 8.54.3.1 rtems_attribute

```
typedef uint32_t rtems_attribute
```

This type represents Classic API attributes.

Attributes are primarily used when creating objects.

Definition at line 90 of file attr.h.

## 8.55 Directive Options

This group contains the Classic API directive options.

### Macros

- #define RTEMS_DEFAULT_OPTIONS 0x00000000

  *This option constant represents the default option set.*
- #define RTEMS_EVENT_ALL 0x00000000

  *This option constant indicates that the task wishes to wait until all events of interest are available in rtems_event_receive() and rtems_event_system_receive().*
- #define RTEMS_EVENT_ANY 0x00000002

  *This option constant indicates that the task wishes to wait until at least one of the events of interest is available in rtems_event_receive() and rtems_event_system_receive().*
- #define RTEMS_NO_WAIT 0x00000001

  *This option constant indicates that the task does not want to wait on the resource.*
- #define RTEMS_WAIT 0x00000000

  *This option constant indicates that the task wants to wait on the resource.*

### Typedefs

- typedef uint32_t rtems_option

  *This type represents a Classic API directive option set.*

### 8.55.1 Detailed Description

This group contains the Classic API directive options.

### 8.55.2 Macro Definition Documentation

#### 8.55.2.1 RTEMS_NO_WAIT

```
#define RTEMS_NO_WAIT 0x00000001
```

This option constant indicates that the task does not want to wait on the resource.

If the resource is not available, then the directives shall return immediately with a status to indicate that the request is unsatisfied.

Definition at line 112 of file options.h.

#### 8.55.2.2 RTEMS_WAIT

```
#define RTEMS_WAIT 0x00000000
```

This option constant indicates that the task wants to wait on the resource.

If the resource is not available, then the task shall block and wait for request completion.

Definition at line 134 of file options.h.

## 8.56 Directive Status Codes

This group contains the Classic API directive status codes and support functions.

### Macros

- #define RTEMS_STATUS_CODES_FIRST RTEMS_SUCCESSFUL

  *This constant represents the lowest valid value for a Classic API directive status code.*
- #define RTEMS_STATUS_CODES_LAST RTEMS_PROXY_BLOCKING

  *This constant represents the highest valid value for a Classic API directive status code.*

### Enumerations

- enum rtems_status_code {
  RTEMS_SUCCESSFUL = 0, RTEMS_TASK_EXITTED = 1, RTEMS_MP_NOT_CONFIGURED = 2,
  RTEMS_INVALID_NAME = 3,
  RTEMS_INVALID_ID = 4, RTEMS_TOO_MANY = 5, RTEMS_TIMEOUT = 6, RTEMS_OBJECT_WAS_DELETED
  = 7,
  RTEMS_INVALID_SIZE = 8, RTEMS_INVALID_ADDRESS = 9, RTEMS_INVALID_NUMBER = 10,
  RTEMS_NOT_DEFINED = 11,
  RTEMS_RESOURCE_IN_USE = 12, RTEMS_UNSATISFIED = 13, RTEMS_INCORRECT_STATE = 14,
  RTEMS_ALREADY_SUSPENDED = 15,
  RTEMS_ILLEGAL_ON_SELF = 16, RTEMS_ILLEGAL_ON_REMOTE_OBJECT = 17, RTEMS_CALLED_FROM_ISR
  = 18, RTEMS_INVALID_PRIORITY = 19,
  RTEMS_INVALID_CLOCK = 20, RTEMS_INVALID_NODE = 21, RTEMS_NOT_CONFIGURED = 22,
  RTEMS_NOT_OWNER_OF_RESOURCE = 23,
  RTEMS_NOT_IMPLEMENTED = 24, RTEMS_INTERNAL_ERROR = 25, RTEMS_NO_MEMORY = 26,
  RTEMS_IO_ERROR = 27,
  RTEMS_INTERRUPTED = 28, RTEMS_PROXY_BLOCKING = 29 }

  *This enumeration provides status codes for directives of the Classic API.*

### Functions

- int rtems_status_code_to_errno (rtems_status_code status_code)

  *Maps the RTEMS status code to a POSIX error number.*
- static bool rtems_are_statuses_equal (rtems_status_code left_status_code, rtems_status_code right_↩ status_code)

  *Checks if the status codes are equal.*
- static bool rtems_is_status_successful (rtems_status_code status_code)

  *Checks if the status code is RTEMS_SUCCESSFUL.*
- const char * rtems_status_text (rtems_status_code status_code)

  *Maps the status code to a descriptive text.*

### 8.56.1 Detailed Description

This group contains the Classic API directive status codes and support functions.

## 8.56.2 Enumeration Type Documentation

### 8.56.2.1 rtems_status_code

enum rtems_status_code

This enumeration provides status codes for directives of the Classic API.

**Enumerator**

| | |
|---|---|
| RTEMS_SUCCESSFUL | This status code indicates successful completion of a requested operation. |
| RTEMS_TASK_EXITTED | This status code indicates that a thread exitted. |
| RTEMS_MP_NOT_CONFIGURED | This status code indicates that multiprocessing was not configured. |
| RTEMS_INVALID_NAME | This status code indicates that an object name was invalid. |
| RTEMS_INVALID_ID | This status code indicates that an object identifier was invalid. |
| RTEMS_TOO_MANY | This status code indicates you have attempted to create too many instances of a particular object class. |
| RTEMS_TIMEOUT | This status code indicates that a blocking directive timed out. |
| RTEMS_OBJECT_WAS_DELETED | This status code indicates the object was deleted while the thread was blocked waiting. |
| RTEMS_INVALID_SIZE | This status code indicates that a specified size was invalid. |
| RTEMS_INVALID_ADDRESS | This status code indicates that a specified address was invalid. |
| RTEMS_INVALID_NUMBER | This status code indicates that a specified number was invalid. |
| RTEMS_NOT_DEFINED | This status code indicates that the item has not been initialized. |
| RTEMS_RESOURCE_IN_USE | This status code indicates that the object still had resources in use. |
| RTEMS_UNSATISFIED | This status code indicates that the request was not satisfied. |
| RTEMS_INCORRECT_STATE | This status code indicates that an object was in wrong state for the requested operation. |
| RTEMS_ALREADY_SUSPENDED | This status code indicates that the thread was already suspended. |
| RTEMS_ILLEGAL_ON_SELF | This status code indicates that the operation was illegal on the calling thread. |
| RTEMS_ILLEGAL_ON_REMOTE_OBJECT | This status code indicates that the operation was illegal on a remote object. |
| RTEMS_CALLED_FROM_ISR | This status code indicates that the operation should not be called from this execution environment. |
| RTEMS_INVALID_PRIORITY | This status code indicates that an invalid thread priority was provided. |
| RTEMS_INVALID_CLOCK | This status code indicates that a specified date or time was invalid. |
| RTEMS_INVALID_NODE | This status code indicates that a specified node identifier was invalid. |
| RTEMS_NOT_CONFIGURED | This status code indicates that the directive was not configured. |
| RTEMS_NOT_OWNER_OF_RESOURCE | This status code indicates that the caller was not the owner of the resource. |

**Enumerator**

| | |
|---|---|
| RTEMS_NOT_IMPLEMENTED | This status code indicates the directive or requested portion of the directive is not implemented. This is a hint that you have stumbled across an opportunity to submit code to the RTEMS Project. |
| RTEMS_INTERNAL_ERROR | This status code indicates that an internal RTEMS inconsistency was detected. |
| RTEMS_NO_MEMORY | This status code indicates that the directive attempted to allocate memory but was unable to do so. |
| RTEMS_IO_ERROR | This status code indicates a device driver IO error. |
| RTEMS_INTERRUPTED | This status code is used internally by the implementation to indicate a blocking device driver call has been interrupted and should be reflected to the caller as interrupted. |
| RTEMS_PROXY_BLOCKING | This status code is used internally by the implementation when performing operations on behalf of remote tasks. This is referred to as proxying operations and this status indicates that the operation could not be completed immediately and the proxy is blocking.<br>This status will not be returned to the user. |

Definition at line 80 of file status.h.

## 8.56.3 Function Documentation

### 8.56.3.1 rtems_are_statuses_equal()

```
static bool rtems_are_statuses_equal (
            rtems_status_code left_status_code,
            rtems_status_code right_status_code )  [inline], [static]
```

Checks if the status codes are equal.

**Parameters**

| | |
|---|---|
| *left_status_code* | is the left hand side status code. |
| *right_status_code* | is the right hand side status code. |

**Returns**

Returns true, if the left hand side status code is equal to the right hand side status code, otherwise false.

Definition at line 322 of file status.h.

#### 8.56.3.2 rtems_is_status_successful()

```
static bool rtems_is_status_successful (
            rtems_status_code status_code )  [inline], [static]
```

Checks if the status code is RTEMS_SUCCESSFUL.

**Parameters**

| | |
|---|---|
| *status_code* | is the status code. |

**Returns**

Returns true, if the status code is equal to RTEMS_SUCCESSFUL, otherwise false.

Definition at line 342 of file status.h.

#### 8.56.3.3 rtems_status_code_to_errno()

```
int rtems_status_code_to_errno (
            rtems_status_code status_code )
```

Maps the RTEMS status code to a POSIX error number.

**Parameters**

| | |
|---|---|
| *status_code* | is the status code to map. |

**Return values**

| | |
|---|---|
| *0* | The status code was RTEMS_SUCCESSFUL. |
| *EBADF* | The status code was RTEMS_INVALID_NUMBER. |
| *EBUSY* | The status code was RTEMS_RESOURCE_IN_USE. |
| *EINTR* | The status code was RTEMS_INTERRUPTED. |
| *EINVAL* | The status code was RTEMS_INVALID_CLOCK, RTEMS_INVALID_NAME, or RTEMS_INVALID_NODE. |
| *EIO* | The status code was RTEMS_ALREADY_SUSPENDED, RTEMS_CALLED_FROM_ISR, RTEMS_ILLEGAL_ON_REMOTE_OBJECT, RTEMS_ILLEGAL_ON_SELF, RTEMS_INCORRECT_STATE, RTEMS_INTERNAL_ERROR, RTEMS_INVALID_ADDRESS, RTEMS_INVALID_ID, RTEMS_INVALID_PRIORITY, RTEMS_INVALID_SIZE, RTEMS_IO_ERROR, RTEMS_MP_NOT_CONFIGURED, RTEMS_NOT_DEFINED, RTEMS_OBJECT_WAS_DELETED, RTEMS_PROXY_BLOCKING, RTEMS_TASK_EXITTED, or RTEMS_TOO_MANY. |
| *ENODEV* | The status code was RTEMS_UNSATISFIED. |
| *ENOMEM* | The status code was RTEMS_NO_MEMORY. |
| *ENOSYS* | The status code was RTEMS_NOT_CONFIGURED or RTEMS_NOT_IMPLEMENTED. |
| *EPERM* | The status code was RTEMS_NOT_OWNER_OF_RESOURCE. |
| *ETIMEDOUT* | The status code was RTEMS_TIMEOUT. |

#### 8.56.3.4 rtems_status_text()

```
const char* rtems_status_text (
            rtems_status_code status_code )
```

Maps the status code to a descriptive text.

The text for each status code is the enumerator constant.

**Parameters**

| | |
|---|---|
| *status_code* | is the status code. |

**Return values**

| | |
|---|---|
| *?* | The status code is invalid. |

**Returns**

Returns a text describing the status code, if the status code is valid.

Definition at line 60 of file statustext.c.

## 8.57 Dual Ported Memory Manager Implementation

### Files

- file dpmemdata.h

  *Classic Dual Ported Memory Manager Data Structures.*
- file dpmemimpl.h

  *Dual Ported Memory Manager Implementation.*

### Classes

- struct Dual_ported_memory_Control

### Macros

- #define DUAL_PORTED_MEMORY_INFORMATION_DEFINE(max)

  *Macro to define the objects information for the Classic Dual Ported Memory objects.*

### Functions

- static __inline__ Dual_ported_memory_Control * _Dual_ported_memory_Allocate (void)

  *Allocates a port control block from the inactive chain of free port control blocks.*
- static __inline__ void _Dual_ported_memory_Free (Dual_ported_memory_Control *the_port)

  *Frees a port control block to the inactive chain of free port control blocks.*
- static __inline__ Dual_ported_memory_Control * **_Dual_ported_memory_Get** (Objects_Id id, ISR_lock_Context *lock_context)

### Variables

- Objects_Information _Dual_ported_memory_Information

  *The Classic Dual Ported Memory objects information.*

### 8.57.1 Detailed Description

### 8.57.2 Macro Definition Documentation

#### 8.57.2.1 DUAL_PORTED_MEMORY_INFORMATION_DEFINE

```
#define DUAL_PORTED_MEMORY_INFORMATION_DEFINE(
              max )
```

**Value:**

```
  OBJECTS_INFORMATION_DEFINE( \
    _Dual_ported_memory, \
    OBJECTS_CLASSIC_API, \
    OBJECTS_RTEMS_PORTS, \
    Dual_ported_memory_Control, \
    max, \
    OBJECTS_NO_STRING_NAME, \
    NULL \
  )
```

Macro to define the objects information for the Classic Dual Ported Memory objects.

This macro should only be used by <rtems/confdefs.h>.

**Parameters**

| | |
|---|---|
| *max* | The configured object maximum (the OBJECTS_UNLIMITED_OBJECTS flag may be set). |

Definition at line 63 of file dpmemdata.h.

## 8.57.3 Function Documentation

### 8.57.3.1 _Dual_ported_memory_Allocate()

```
static __inline__ Dual_ported_memory_Control* _Dual_ported_memory_Allocate (
            void  )  [static]
```

Allocates a port control block from the inactive chain of free port control blocks.

This routine allocates a port control block from the inactive chain of free port control blocks.

Definition at line 43 of file dpmemimpl.h.

### 8.57.3.2 _Dual_ported_memory_Free()

```
static __inline__ void _Dual_ported_memory_Free (
            Dual_ported_memory_Control * the_port )  [static]
```

Frees a port control block to the inactive chain of free port control blocks.

This routine frees a port control block to the inactive chain of free port control blocks.

Definition at line 56 of file dpmemimpl.h.

## 8.58 Dual-Ported Memory Manager

The Dual-Ported Memory Manager provides a mechanism for converting addresses between internal and external representations for multiple dual-ported memory areas (DPMA).

### Functions

- rtems_status_code rtems_port_create (rtems_name name, void *internal_start, void *external_start, uint32_t length, rtems_id *id)

  %
- rtems_status_code rtems_port_delete (rtems_id id)

  %
- rtems_status_code rtems_port_external_to_internal (rtems_id id, void *external, void **internal)

  %
- rtems_status_code rtems_port_ident (rtems_name name, rtems_id *id)

  *Identifies a port object by the specified object name.*
- rtems_status_code rtems_port_internal_to_external (rtems_id id, void *internal, void **external)

  %

### 8.58.1 Detailed Description

The Dual-Ported Memory Manager provides a mechanism for converting addresses between internal and external representations for multiple dual-ported memory areas (DPMA).

### 8.58.2 Function Documentation

#### 8.58.2.1 rtems_port_create()

```
rtems_status_code rtems_port_create (
            rtems_name name,
            void * internal_start,
            void * external_start,
            uint32_t length,
            rtems_id * id )
```

%

**Parameters**

| | |
|---|---|
| *name* | % |
| *internal_start* | % |
| *external_start* | % |
| *length* | % |
| *id* | % |

#### 8.58.2.2 rtems_port_delete()

```
rtems_status_code rtems_port_delete (
            rtems_id id )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |

#### 8.58.2.3 rtems_port_external_to_internal()

```
rtems_status_code rtems_port_external_to_internal (
            rtems_id id,
            void * external,
            void ** internal )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *external* | % |
| *internal* | % |

#### 8.58.2.4 rtems_port_ident()

```
rtems_status_code rtems_port_ident (
            rtems_name name,
            rtems_id * id )
```

Identifies a port object by the specified object name.

This directive obtains the port identifier associated with the port name specified in `name`.

If the port name is not unique, then the port identifier will match the first port with that name in the search order. However, this port identifier is not guaranteed to correspond to the desired port. The port identifier is used with other dual-ported memory related directives to access the port.

The objects are searched from lowest to the highest index. Only the local node is searched.

**Parameters**

| | | |
|---|---|---|
| | *name* | is the object name to look up. |
| out | *id* | is the pointer to an object identifier variable. The object identifier of an object with the specified name will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The id parameter was NULL. |
| *RTEMS_INVALID_NAME* | The name parameter was 0. |
| *RTEMS_INVALID_NAME* | There was no object with the specified name on the local node. |

### 8.58.2.5 rtems_port_internal_to_external()

```
rtems_status_code rtems_port_internal_to_external (
            rtems_id id,
            void * internal,
            void ** external )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *internal* | % |
| *external* | % |

# 8.59 EDF Priority SMP Scheduler

EDF Priority SMP Scheduler.

## Files

- file scheduleredfsmp.h

  *EDF SMP Scheduler API.*
- file scheduleredfsmp.c

  *EDF SMP Scheduler Implementation.*

## Classes

- struct Scheduler_EDF_SMP_Node
- struct Scheduler_EDF_SMP_Ready_queue
- struct Scheduler_EDF_SMP_Context

## Macros

- #define **SCHEDULER_EDF_SMP_ENTRY_POINTS**

## Functions

- void _Scheduler_EDF_SMP_Initialize (const Scheduler_Control *scheduler)

  *Initializes the context of the scheduler control.*
- void _Scheduler_EDF_SMP_Node_initialize (const Scheduler_Control *scheduler, Scheduler_Node *node, Thread_Control *the_thread, Priority_Control priority)

  *Initializes the node with the given priority.*
- void _Scheduler_EDF_SMP_Block (const Scheduler_Control *scheduler, Thread_Control *thread, Scheduler_Node *node)

  *Blocks the thread.*
- void _Scheduler_EDF_SMP_Unblock (const Scheduler_Control *scheduler, Thread_Control *thread, Scheduler_Node *node)

  *Unblocks the thread.*
- void _Scheduler_EDF_SMP_Update_priority (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Updates the priority of the node.*
- bool _Scheduler_EDF_SMP_Ask_for_help (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Asks for help operation.*
- void _Scheduler_EDF_SMP_Reconsider_help_request (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Reconsiders help operation.*
- void _Scheduler_EDF_SMP_Withdraw_node (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node, Thread_Scheduler_state next_state)

  *Withdraws node operation.*
- void _Scheduler_EDF_SMP_Pin (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node, struct Per_CPU_Control *cpu)

  *Pin thread operation.*

- void _Scheduler_EDF_SMP_Unpin (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node, struct Per_CPU_Control *cpu)

  *Unpin thread operation.*
- void _Scheduler_EDF_SMP_Add_processor (const Scheduler_Control *scheduler, Thread_Control *idle)

  *Adds processor.*
- Thread_Control * _Scheduler_EDF_SMP_Remove_processor (const Scheduler_Control *scheduler, struct Per_CPU_Control *cpu)

  *Removes an idle thread from the given cpu.*
- void _Scheduler_EDF_SMP_Yield (const Scheduler_Control *scheduler, Thread_Control *thread, Scheduler_Node *node)

  *Performs the yield of a thread.*
- void _Scheduler_EDF_SMP_Start_idle (const Scheduler_Control *scheduler, Thread_Control *idle, struct Per_CPU_Control *cpu)

  *Starts an idle thread.*
- bool _Scheduler_EDF_SMP_Set_affinity (const Scheduler_Control *scheduler, Thread_Control *thread, Scheduler_Node *node, const Processor_mask *affinity)

  *Checks if the processor set of the scheduler is the subset of the affinity set.*

## 8.59.1 Detailed Description

EDF Priority SMP Scheduler.

## 8.59.2 Macro Definition Documentation

### 8.59.2.1 SCHEDULER_EDF_SMP_ENTRY_POINTS

```
#define SCHEDULER_EDF_SMP_ENTRY_POINTS
```

**Value:**

```
{ \
  _Scheduler_EDF_SMP_Initialize, \
  _Scheduler_default_Schedule, \
  _Scheduler_EDF_SMP_Yield, \
  _Scheduler_EDF_SMP_Block, \
  _Scheduler_EDF_SMP_Unblock, \
  _Scheduler_EDF_SMP_Update_priority, \
  _Scheduler_EDF_Map_priority, \
  _Scheduler_EDF_Unmap_priority, \
  _Scheduler_EDF_SMP_Ask_for_help, \
  _Scheduler_EDF_SMP_Reconsider_help_request, \
  _Scheduler_EDF_SMP_Withdraw_node, \
  _Scheduler_EDF_SMP_Pin, \
  _Scheduler_EDF_SMP_Unpin, \
  _Scheduler_EDF_SMP_Add_processor, \
  _Scheduler_EDF_SMP_Remove_processor, \
  _Scheduler_EDF_SMP_Node_initialize, \
  _Scheduler_default_Node_destroy, \
  _Scheduler_EDF_Release_job, \
  _Scheduler_EDF_Cancel_job, \
  _Scheduler_default_Tick, \
  _Scheduler_EDF_SMP_Start_idle, \
  _Scheduler_EDF_SMP_Set_affinity \
}
```

Definition at line 108 of file scheduleredfsmp.h.

## 8.59.3 Function Documentation

### 8.59.3.1 _Scheduler_EDF_SMP_Add_processor()

```
void _Scheduler_EDF_SMP_Add_processor (
            const Scheduler_Control * scheduler,
            Thread_Control * idle )
```

Adds processor.

**Parameters**

| in,out | | |
|---|---|---|
| | *scheduler* | The scheduler instance to add the processor to. |
| | *idle* | The idle thread of the processor to add. |

Definition at line 594 of file scheduleredfsmp.c.

### 8.59.3.2 _Scheduler_EDF_SMP_Ask_for_help()

```
bool _Scheduler_EDF_SMP_Ask_for_help (
            const Scheduler_Control * scheduler,
            Thread_Control * the_thread,
            Scheduler_Node * node )
```

Asks for help operation.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance to ask for help. |
| *the_thread* | The thread needing help. |
| *node* | The scheduler node. |

**Return values**

| | |
|---|---|
| *true* | Ask for help was successful. |
| *false* | Ask for help was not successful. |

Definition at line 532 of file scheduleredfsmp.c.

### 8.59.3.3 _Scheduler_EDF_SMP_Block()

```
void _Scheduler_EDF_SMP_Block (
            const Scheduler_Control * scheduler,
```

```
            Thread_Control * thread,
            Scheduler_Node * node )
```

Blocks the thread.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| in,out | *the_thread* | The thread to block. |
| in,out | *node* | The *thread's* scheduler node. |

Definition at line 417 of file scheduleredfsmp.c.

#### 8.59.3.4 _Scheduler_EDF_SMP_Initialize()

```
void _Scheduler_EDF_SMP_Initialize (
            const Scheduler_Control * scheduler )
```

Initializes the context of the scheduler control.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler control. |

Definition at line 61 of file scheduleredfsmp.c.

#### 8.59.3.5 _Scheduler_EDF_SMP_Node_initialize()

```
void _Scheduler_EDF_SMP_Node_initialize (
            const Scheduler_Control * scheduler,
            Scheduler_Node * node,
            Thread_Control * the_thread,
            Priority_Control priority )
```

Initializes the node with the given priority.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| out | *node* | The node to initialize. |
| | *the_thread* | The thread of the scheduler node. |
| | *priority* | The priority for the initialization. |

Definition at line 71 of file scheduleredfsmp.c.

#### 8.59.3.6 _Scheduler_EDF_SMP_Pin()

```
void _Scheduler_EDF_SMP_Pin (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread,
              Scheduler_Node * node,
              struct Per_CPU_Control * cpu )
```

Pin thread operation.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance of the specified processor. |
| *the_thread* | The thread to pin. |
| *node* | The scheduler node of the thread. |
| *cpu* | The processor to pin the thread. |

Definition at line 675 of file scheduleredfsmp.c.

#### 8.59.3.7 _Scheduler_EDF_SMP_Reconsider_help_request()

```
void _Scheduler_EDF_SMP_Reconsider_help_request (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread,
              Scheduler_Node * node )
```

Reconsiders help operation.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance to reconsider the help request. |
| *the_thread* | The thread reconsidering a help request. |
| *node* | The scheduler node. |

Definition at line 543 of file scheduleredfsmp.c.

#### 8.59.3.8 _Scheduler_EDF_SMP_Remove_processor()

```
Thread_Control* _Scheduler_EDF_SMP_Remove_processor (
              const Scheduler_Control * scheduler,
              struct Per_CPU_Control * cpu )
```

Removes an idle thread from the given cpu.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance. |
| *cpu* | The cpu control to remove from *scheduler*. |

**Returns**

The idle thread of the processor.

Definition at line 610 of file scheduleredfsmp.c.

### 8.59.3.9 _Scheduler_EDF_SMP_Set_affinity()

```
bool _Scheduler_EDF_SMP_Set_affinity (
              const Scheduler_Control * scheduler,
              Thread_Control * thread,
              Scheduler_Node * node,
              const Processor_mask * affinity )
```

Checks if the processor set of the scheduler is the subset of the affinity set.

Default implementation of the set affinity scheduler operation.

**Parameters**

| | |
|---|---|
| *scheduler* | This parameter is unused. |
| *thread* | This parameter is unused. |
| *node* | This parameter is unused. |
| *affinity* | The new processor affinity set for the thread. |

**Return values**

| | |
|---|---|
| *true* | The processor set of the scheduler is a subset of the affinity set. |
| *false* | The processor set of the scheduler is not a subset of the affinity set. |

Definition at line 719 of file scheduleredfsmp.c.

### 8.59.3.10 _Scheduler_EDF_SMP_Start_idle()

```
void _Scheduler_EDF_SMP_Start_idle (
              const Scheduler_Control * scheduler,
              Thread_Control * idle,
              struct Per_CPU_Control * cpu )
```

Starts an idle thread.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| in,out | *the_thread* | An idle thread. |
| | *cpu* | The cpu for the operation. |

Definition at line 657 of file scheduleredfsmp.c.

#### 8.59.3.11 _Scheduler_EDF_SMP_Unblock()

```
void _Scheduler_EDF_SMP_Unblock (
            const Scheduler_Control * scheduler,
            Thread_Control * thread,
            Scheduler_Node * node )
```

Unblocks the thread.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| in,out | *the_thread* | The thread to unblock. |
| in,out | *node* | The *thread's* scheduler node. |

Definition at line 476 of file scheduleredfsmp.c.

#### 8.59.3.12 _Scheduler_EDF_SMP_Unpin()

```
void _Scheduler_EDF_SMP_Unpin (
            const Scheduler_Control * scheduler,
            Thread_Control * the_thread,
            Scheduler_Node * node,
            struct Per_CPU_Control * cpu )
```

Unpin thread operation.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance of the specified processor. |
| *the_thread* | The thread to unpin. |
| *node* | The scheduler node of the thread. |
| *cpu* | The processor to unpin the thread. |

Definition at line 698 of file scheduleredfsmp.c.

#### 8.59.3.13 _Scheduler_EDF_SMP_Update_priority()

```
void _Scheduler_EDF_SMP_Update_priority (
            const Scheduler_Control * scheduler,
```

```
            Thread_Control * the_thread,
            Scheduler_Node * node )
```

Updates the priority of the node.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance. |
| *the_thread* | The thread for the operation. |
| *node* | The thread's scheduler node. |

Definition at line 512 of file scheduleredfsmp.c.

#### 8.59.3.14 _Scheduler_EDF_SMP_Withdraw_node()

```
void _Scheduler_EDF_SMP_Withdraw_node (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread,
              Scheduler_Node * node,
              Thread_Scheduler_state next_state )
```

Withdraws node operation.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance to withdraw the node. |
| *the_thread* | The thread using the node. |
| *node* | The scheduler node to withdraw. |
| *next_state* | The next thread scheduler state in case the node is scheduled. |

Definition at line 559 of file scheduleredfsmp.c.

#### 8.59.3.15 _Scheduler_EDF_SMP_Yield()

```
void _Scheduler_EDF_SMP_Yield (
              const Scheduler_Control * scheduler,
              Thread_Control * thread,
              Scheduler_Node * node )
```

Performs the yield of a thread.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| in,out | *the_thread* | The thread that performed the yield operation. |
| | *node* | The scheduler node of *the_thread*. |

Definition at line 625 of file scheduleredfsmp.c.

## 8.60 EDF Scheduler

EDF Scheduler.

### Files

- file scheduleredf.h

  *Data Related to the Manipulation of Threads for the EDF Scheduler.*
- file scheduleredfimpl.h

  *EDF Scheduler Implementation.*

### Classes

- struct Scheduler_EDF_Context
- struct Scheduler_EDF_Node

  *Scheduler node specialization for EDF schedulers.*

### Macros

- #define **SCHEDULER_EDF_MAXIMUM_PRIORITY** INT_MAX
- #define SCHEDULER_EDF_ENTRY_POINTS
- #define SCHEDULER_EDF_PRIO_MSB 0x8000000000000000

### Functions

- void _Scheduler_EDF_Initialize (const Scheduler_Control *scheduler)

  *Initializes EDF scheduler.*
- void _Scheduler_EDF_Block (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Removes the blocking thread from the ready queue and schedules is only again if the thread is executing or the heir thread.*
- void _Scheduler_EDF_Schedule (const Scheduler_Control *scheduler, Thread_Control *the_thread)

  *Sets the heir thread to be the next ready thread in the rbtree ready queue.*
- void _Scheduler_EDF_Node_initialize (const Scheduler_Control *scheduler, Scheduler_Node *node, Thread_Control *the_thread, Priority_Control priority)

  *Initializes an EDF specific scheduler node of the_thread.*
- void _Scheduler_EDF_Unblock (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Performs an unblocking of the thread.*
- void _Scheduler_EDF_Update_priority (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Updates the priority of the scheduler node.*
- Priority_Control _Scheduler_EDF_Map_priority (const Scheduler_Control *scheduler, Priority_Control priority)

  *Gets the mapped priority map of the priority control.*
- Priority_Control _Scheduler_EDF_Unmap_priority (const Scheduler_Control *scheduler, Priority_Control priority)

  *Gets the unmapped priority map of the priority control.*

- void _Scheduler_EDF_Yield (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Executes a thread yield for the thread.*

- void _Scheduler_EDF_Release_job (const Scheduler_Control *scheduler, Thread_Control *the_thread, Priority_Node *priority_node, uint64_t deadline, Thread_queue_Context *queue_context)

  *Releases a EDF job.*

- void _Scheduler_EDF_Cancel_job (const Scheduler_Control *scheduler, Thread_Control *the_thread, Priority_Node *priority_node, Thread_queue_Context *queue_context)

  *Cancels a job and removes the thread from the queue context.*

- static __inline__ Scheduler_EDF_Context * _Scheduler_EDF_Get_context (const Scheduler_Control *scheduler)

  *Gets the context of the scheduler.*

- static __inline__ Scheduler_EDF_Node * _Scheduler_EDF_Thread_get_node (Thread_Control *the_thread)

  *Gets the scheduler EDF node of the thread.*

- static __inline__ Scheduler_EDF_Node * _Scheduler_EDF_Node_downcast (Scheduler_Node *node)

  *Returns the scheduler EDF node for the scheduler node.*

- static __inline__ bool _Scheduler_EDF_Less (const void *left, const RBTree_Node *right)

  *Checks if left is less than the priority of the node right.*

- static __inline__ bool _Scheduler_EDF_Priority_less_equal (const void *left, const RBTree_Node *right)

  *Checks if left is less or equal than the priority of the node right.*

- static __inline__ void _Scheduler_EDF_Enqueue (Scheduler_EDF_Context *context, Scheduler_EDF_Node *node, Priority_Control insert_priority)

  *Inserts the scheduler node with the given priority in the ready queue of the context.*

- static __inline__ void _Scheduler_EDF_Extract (Scheduler_EDF_Context *context, Scheduler_EDF_Node *node)

  *Extracts the scheduler node from the ready queue of the context.*

- static __inline__ void _Scheduler_EDF_Extract_body (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Extracts the node from the context of the given scheduler.*

- static __inline__ void _Scheduler_EDF_Schedule_body (const Scheduler_Control *scheduler, Thread_Control *the_thread, bool force_dispatch)

  *Schedules the next ready thread as the heir.*

### 8.60.1 Detailed Description

EDF Scheduler.

### 8.60.2 Macro Definition Documentation

#### 8.60.2.1 SCHEDULER_EDF_ENTRY_POINTS

```
#define SCHEDULER_EDF_ENTRY_POINTS
```

**Value:**

```
{ \
  _Scheduler_EDF_Initialize,       /* initialize entry point */ \
  _Scheduler_EDF_Schedule,         /* schedule entry point */ \
  _Scheduler_EDF_Yield,            /* yield entry point */ \
  _Scheduler_EDF_Block,            /* block entry point */ \
  _Scheduler_EDF_Unblock,          /* unblock entry point */ \
  _Scheduler_EDF_Update_priority,  /* update priority entry point */ \
  _Scheduler_EDF_Map_priority,     /* map priority entry point */ \
  _Scheduler_EDF_Unmap_priority,   /* unmap priority entry point */ \
  SCHEDULER_OPERATION_DEFAULT_ASK_FOR_HELP \
  _Scheduler_EDF_Node_initialize,  /* node initialize entry point */ \
  _Scheduler_default_Node_destroy, /* node destroy entry point */ \
  _Scheduler_EDF_Release_job,      /* new period of task */ \
  _Scheduler_EDF_Cancel_job,       /* cancel period of task */ \
  _Scheduler_default_Tick,         /* tick entry point */ \
  _Scheduler_default_Start_idle    /* start idle entry point */ \
  SCHEDULER_OPERATION_DEFAULT_GET_SET_AFFINITY \
}
```

Entry points for the Earliest Deadline First Scheduler.

Definition at line 55 of file scheduleredf.h.

#### 8.60.2.2 SCHEDULER_EDF_PRIO_MSB

```
#define SCHEDULER_EDF_PRIO_MSB 0x8000000000000000
```

This is just a most significant bit of Priority_Control type. It distinguishes threads which are deadline driven (priority represented by a lower number than *SCHEDULER_EDF_PRIO_MSB*) from those ones who do not have any deadlines and thus are considered background tasks.

Definition at line 41 of file scheduleredfimpl.h.

### 8.60.3 Function Documentation

#### 8.60.3.1 _Scheduler_EDF_Block()

```
void _Scheduler_EDF_Block (
            const Scheduler_Control * scheduler,
            Thread_Control * the_thread,
            Scheduler_Node * node )
```

Removes the blocking thread from the ready queue and schedules is only again if the thread is executing or the heir thread.

**Parameters**

| | | |
|---|---|---|
| in,out | *scheduler* | The scheduler for the operation. |
| | *the_thread* | The thread to operate upon. |
| in,out | *node* | The scheduler node for this thread. |

#### 8.60.3.2 _Scheduler_EDF_Cancel_job()

```
void _Scheduler_EDF_Cancel_job (
            const Scheduler_Control * scheduler,
            Thread_Control * the_thread,
            Priority_Node * priority_node,
            Thread_queue_Context * queue_context )
```

Cancels a job and removes the thread from the queue context.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| | *the_thread* | The thread for the operation. |
| in,out | *priority_node* | The corresponding priority node. |
| in,out | *queue_context* | The thread queue context. |

Definition at line 77 of file scheduleredfreleasejob.c.

#### 8.60.3.3 _Scheduler_EDF_Enqueue()

```
static __inline__ void _Scheduler_EDF_Enqueue (
            Scheduler_EDF_Context * context,
            Scheduler_EDF_Node * node,
            Priority_Control insert_priority )  [static]
```

Inserts the scheduler node with the given priority in the ready queue of the context.

**Parameters**

| | | |
|---|---|---|
| in,out | *context* | The context to insert the node in. |
| | *node* | The node to be inserted. |
| | *insert_priority* | The priority with which the node will be inserted. |

Definition at line 148 of file scheduleredfimpl.h.

#### 8.60.3.4 _Scheduler_EDF_Extract()

```
static __inline__ void _Scheduler_EDF_Extract (
            Scheduler_EDF_Context * context,
            Scheduler_EDF_Node * node )  [static]
```

Extracts the scheduler node from the ready queue of the context.

**Parameters**

| | | |
|---|---|---|
| in,out | *context* | The context to extract the node from. |
| in,out | *node* | The node to extract. |

Definition at line 168 of file scheduleredfimpl.h.

#### 8.60.3.5 _Scheduler_EDF_Extract_body()

```
static __inline__ void _Scheduler_EDF_Extract_body (
            const Scheduler_Control * scheduler,
            Thread_Control * the_thread,
            Scheduler_Node * node )  [static]
```

Extracts the node from the context of the given scheduler.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| | *the_thread* | The thread is not used in this method. |
| in,out | *node* | The node to be extracted. |

Definition at line 183 of file scheduleredfimpl.h.

#### 8.60.3.6 _Scheduler_EDF_Get_context()

```
static __inline__ Scheduler_EDF_Context* _Scheduler_EDF_Get_context (
            const Scheduler_Control * scheduler )  [static]
```

Gets the context of the scheduler.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance. |

**Returns**

The scheduler context of *scheduler*.

Definition at line 51 of file scheduleredfimpl.h.

#### 8.60.3.7 _Scheduler_EDF_Initialize()

```
void _Scheduler_EDF_Initialize (
            const Scheduler_Control * scheduler )
```

Initializes EDF scheduler.

This routine initializes the EDF scheduler.

**Parameters**

| | | |
|---|---|---|
| in,out | *scheduler* | The scheduler instance. |

#### 8.60.3.8 _Scheduler_EDF_Less()

```
static __inline__ bool _Scheduler_EDF_Less (
            const void * left,
            const RBTree_Node * right )  [static]
```

Checks if *left* is less than the priority of the node *right*.

**Parameters**

| | |
|---|---|
| *left* | The priority on the left hand side of the comparison. |
| *right* | The node of which the priority is compared to left. |

**Return values**

| | |
|---|---|
| *true* | *left* is less than the priority of *right*. |
| *false* | *left* is greater or equal than the priority of *right*. |

Definition at line 93 of file scheduleredfimpl.h.

#### 8.60.3.9 _Scheduler_EDF_Map_priority()

```
Priority_Control _Scheduler_EDF_Map_priority (
            const Scheduler_Control * scheduler,
            Priority_Control priority )
```

Gets the mapped priority map of the priority control.

**Parameters**

| | |
|---|---|
| *scheduler* | Not used in this operation. |
| *priority* | The priority control to get the priority map of. |

**Returns**

The mapped priority map of *priority*.

Definition at line 24 of file scheduleredfreleasejob.c.

#### 8.60.3.10 _Scheduler_EDF_Node_downcast()

```
static __inline__ Scheduler_EDF_Node* _Scheduler_EDF_Node_downcast (
            Scheduler_Node * node )  [static]
```

Returns the scheduler EDF node for the scheduler node.

**Parameters**

| | |
|---|---|
| *node* | The scheduler node of which the scheduler EDF node is returned. |

**Returns**

The corresponding scheduler EDF node.

Definition at line 77 of file scheduleredfimpl.h.

#### 8.60.3.11 _Scheduler_EDF_Node_initialize()

```
void _Scheduler_EDF_Node_initialize (
            const Scheduler_Control * scheduler,
            Scheduler_Node * node,
            Thread_Control * the_thread,
            Priority_Control priority )
```

Initializes an EDF specific scheduler node of *the_thread*.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance. |
| *node* | The node being initialized. |
| *the_thread* | The thread of the node. |
| *priority* | The thread priority. |

#### 8.60.3.12 _Scheduler_EDF_Priority_less_equal()

```
static __inline__ bool _Scheduler_EDF_Priority_less_equal (
```

```
            const void * left,
            const RBTree_Node * right )  [static]
```

Checks if *left* is less or equal than the priority of the node *right*.

**Parameters**

| | |
|---|---|
| *left* | The priority on the left hand side of the comparison. |
| *right* | The node of which the priority is compared to left. |

**Return values**

| | |
|---|---|
| *true* | *left* is less or equal than the priority of *right*. |
| *false* | *left* is greater than the priority of *right*. |

Definition at line 121 of file scheduleredfimpl.h.

#### 8.60.3.13 _Scheduler_EDF_Release_job()

```
void _Scheduler_EDF_Release_job (
            const Scheduler_Control * scheduler,
            Thread_Control * the_thread,
            Priority_Node * priority_node,
            uint64_t deadline,
            Thread_queue_Context * queue_context )
```

Releases a EDF job.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance |
| | *the_thread* | The thread |
| in,out | *priority_node* | The priority node for the operation. |
| | *deadline* | The deadline for the job. |
| in,out | *queue_context* | The thread queue context. |

Definition at line 40 of file scheduleredfreleasejob.c.

#### 8.60.3.14 _Scheduler_EDF_Schedule()

```
void _Scheduler_EDF_Schedule (
            const Scheduler_Control * scheduler,
            Thread_Control * the_thread )
```

Sets the heir thread to be the next ready thread in the rbtree ready queue.

This kernel routine sets the heir thread to be the next ready thread in the rbtree ready queue.

**Parameters**

| | | |
|---|---|---|
| in,out | *scheduler* | The scheduler instance. |
| | *the_thread* | The thread being scheduled. |

#### 8.60.3.15 _Scheduler_EDF_Schedule_body()

```
static __inline__ void _Scheduler_EDF_Schedule_body (
            const Scheduler_Control * scheduler,
            Thread_Control * the_thread,
            bool force_dispatch )  [static]
```

Schedules the next ready thread as the heir.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance to schedule the minimum of the context of. |
| *the_thread* | This parameter is not used. |
| *force_dispatch* | Indicates whether the current heir is blocked even if it is not set as preemptible. |

Definition at line 206 of file scheduleredfimpl.h.

#### 8.60.3.16 _Scheduler_EDF_Thread_get_node()

```
static __inline__ Scheduler_EDF_Node* _Scheduler_EDF_Thread_get_node (
            Thread_Control * the_thread )  [static]
```

Gets the scheduler EDF node of the thread.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to get the scheduler node of. |

**Returns**

The EDF scheduler node of *the_thread*.

Definition at line 63 of file scheduleredfimpl.h.

#### 8.60.3.17 _Scheduler_EDF_Unblock()

```
void _Scheduler_EDF_Unblock (
            const Scheduler_Control * scheduler,
```

```
            Thread_Control * the_thread,
            Scheduler_Node * node )
```

Performs an unblocking of the thread.

**Parameters**

| | | |
|---|---|---|
| in,out | *scheduler* | The scheduler instance. |
| | *the_thread* | The unblocking thread. May be set as new heir. |
| in,out | *node* | The scheduler node for the thread. |

#### 8.60.3.18 _Scheduler_EDF_Unmap_priority()

```
Priority_Control _Scheduler_EDF_Unmap_priority (
            const Scheduler_Control * scheduler,
            Priority_Control priority )
```

Gets the unmapped priority map of the priority control.

**Parameters**

| | |
|---|---|
| *scheduler* | Not used in this operation. |
| *priority* | The priority control to get the priority map of. |

**Returns**

The unmapped priority map of *priority*.

Definition at line 32 of file scheduleredfreleasejob.c.

#### 8.60.3.19 _Scheduler_EDF_Update_priority()

```
void _Scheduler_EDF_Update_priority (
            const Scheduler_Control * scheduler,
            Thread_Control * the_thread,
            Scheduler_Node * node )
```

Updates the priority of the scheduler node.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| | *the_thread* | The thread for the operation. |
| in,out | *node* | The priority node to update the priority of. |

#### 8.60.3.20 _Scheduler_EDF_Yield()

```
void _Scheduler_EDF_Yield (
            const Scheduler_Control * scheduler,
            Thread_Control * the_thread,
            Scheduler_Node * node )
```

Executes a thread yield for the thread.

**Parameters**

| | | |
|---|---|---|
| in,out | *scheduler* | The scheduler instance. |
| | *the_thread* | The thread that performs the yield. |
| in,out | *node* | The scheduler node for this thread. |

# 8.61 Event Implementation

## Files

- file event.h

  *This header file provides the Event Manager API.*
- file eventdata.h

  *This header file defines the API used by the Application Configuration to define the configured Thread Control Block (TCB).*
- file eventimpl.h

  *This header file defines interfaces used by the event implementation.*
- file eventreceive.c

  *This source file contains the implementation of rtems_event_receive().*
- file eventseize.c

  *This source file contains the implementation of _Event_Seize() and the task event MPCI support system initialization.*
- file eventsend.c

  *This source file contains the implementation of rtems_event_send().*
- file eventsurrender.c

  *This source file contains the implementation of _Event_Surrender().*
- file systemeventreceive.c

  *This source file contains the implementation of rtems_event_system_receive().*
- file systemeventsend.c

  *This source file contains the implementation of rtems_event_system_send().*

## Classes

- struct Event_Control

  *This structure is used to manage a set of events.*

## Functions

- rtems_status_code _Event_Seize (rtems_event_set event_in, rtems_option option_set, rtems_interval ticks, rtems_event_set *event_out, Thread_Control *executing, Event_Control *event, Thread_Wait_flags wait_class, States_Control block_state, ISR_lock_Context *lock_context)

  *Seizes a set of events.*
- rtems_status_code _Event_Surrender (Thread_Control *the_thread, rtems_event_set event_in, Event_Control *event, Thread_Wait_flags wait_class, ISR_lock_Context *lock_context)

  *Surrenders a set of events.*
- static __inline__ void _Event_Initialize (Event_Control *event)

  *Initializes an event control block to have no pending events.*
- static __inline__ bool _Event_sets_Is_empty (rtems_event_set the_event_set)

  *Checks if the event set is empty.*
- static __inline__ void _Event_sets_Post (rtems_event_set the_new_events, rtems_event_set *the_event_set)

  *Posts the events in the specified event set.*
- static __inline__ rtems_event_set _Event_sets_Get (rtems_event_set the_event_set, rtems_event_set the_event_condition)

  *Gets the events of the specified event condition.*
- static __inline__ rtems_event_set _Event_sets_Clear (rtems_event_set the_event_set, rtems_event_set the_mask)

  *Clears a set of events from an event set.*

### 8.61.1 Detailed Description

### 8.61.2 Function Documentation

#### 8.61.2.1 _Event_Initialize()

```
static __inline__ void _Event_Initialize (
              Event_Control * event )  [static]
```

Initializes an event control block to have no pending events.

**Parameters**

| | |
|---|---|
| *event* | is the event control block to initialize. |

Definition at line 104 of file eventimpl.h.

#### 8.61.2.2 _Event_Seize()

```
rtems_status_code _Event_Seize (
              rtems_event_set event_in,
              rtems_option option_set,
              rtems_interval ticks,
              rtems_event_set * event_out,
              Thread_Control * executing,
              Event_Control * event,
              Thread_Wait_flags wait_class,
              States_Control block_state,
              ISR_lock_Context * lock_context )
```

Seizes a set of events.

**Parameters**

| | |
|---|---|
| *event_in* | is the input event set. |
| *option_set* | is the option set. |
| *ticks* | is the optional timeout in clock ticks. |
| *event_out[out]* | is the output event set. |
| *executing[in,out]* | is the executing thread. |
| *event[in,out]* | is the source event set. |
| *wait_class* | is the thread wait class of the source event set. |
| *block_state* | is the thread blocking state of the source event set. |
| *lock_context[in,out]* | is the lock context set up by _Thread_Get(). |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_UNSATISFIED* | The events of interest were not immediately available. |
| *RTEMS_TIMEOUT* | The events of interest were not available within the specified timeout interval. |

Definition at line 30 of file eventseize.c.

#### 8.61.2.3 _Event_sets_Clear()

```
static __inline__ rtems_event_set _Event_sets_Clear (
             rtems_event_set the_event_set,
             rtems_event_set the_mask )  [static]
```

Clears a set of events from an event set.

**Parameters**

| | |
|---|---|
| *the_event_set* | is the event set. |
| *the_mask* | is the set of events to clear. |

**Returns**

Returns the event set with all event cleared specified by the event mask.

Definition at line 167 of file eventimpl.h.

#### 8.61.2.4 _Event_sets_Get()

```
static __inline__ rtems_event_set _Event_sets_Get (
             rtems_event_set the_event_set,
             rtems_event_set the_event_condition )  [static]
```

Gets the events of the specified event condition.

**Parameters**

| | |
|---|---|
| *the_event_set* | is the event set. |
| *the_event_condition* | is the event condition defining the events of interest. |

**Returns**

Return the events of the event condition which are posted in the event set.

Definition at line 149 of file eventimpl.h.

#### 8.61.2.5 _Event_sets_Is_empty()

```
static __inline__ bool _Event_sets_Is_empty (
              rtems_event_set the_event_set )  [static]
```

Checks if the event set is empty.

**Parameters**

| | |
|---|---|
| *the_event_set* | is the event set to check. |

**Returns**

Returns true, if there are no posted events in the event set, otherwise false.

Definition at line 117 of file eventimpl.h.

#### 8.61.2.6 _Event_sets_Post()

```
static __inline__ void _Event_sets_Post (
              rtems_event_set the_new_events,
              rtems_event_set * the_event_set )  [static]
```

Posts the events in the specified event set.

**Parameters**

| | |
|---|---|
| *the_new_events* | is the set of events to post. |
| *the_event_set[in,out]* | is the event set. |

Definition at line 131 of file eventimpl.h.

#### 8.61.2.7 _Event_Surrender()

```
rtems_status_code _Event_Surrender (
              Thread_Control * the_thread,
              rtems_event_set event_in,
              Event_Control * event,
              Thread_Wait_flags wait_class,
              ISR_lock_Context * lock_context )
```

Surrenders a set of events.

**Parameters**

| | |
|---|---|
| *the_thread[in,out]* | is the thread of the event set. |

**Parameters**

| | |
|---|---|
| *event_in* | is the set of events to post. |
| *event[in,out]* | is the target event set. |
| *wait_class* | is the thread wait class of the target event set. |
| *lock_context[in,out]* | is the lock context set up by _Thread_Get(). |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |

Definition at line 70 of file eventsurrender.c.

## 8.62 Event Manager

The Event Manager provides a high performance method of inter-task communication and synchronization.

### Macros

- #define RTEMS_ALL_EVENTS 0xffffffff

  *This event set constant represents all events of an event set.*
- #define RTEMS_EVENT_0 0x00000001

  *This event set constant represents the bit in the event set associated with event 0.*
- #define RTEMS_EVENT_1 0x00000002

  *This event set constant represents the bit in the event set associated with event 1.*
- #define RTEMS_EVENT_2 0x00000004

  *This event set constant represents the bit in the event set associated with event 2.*
- #define RTEMS_EVENT_3 0x00000008

  *This event set constant represents the bit in the event set associated with event 3.*
- #define RTEMS_EVENT_4 0x00000010

  *This event set constant represents the bit in the event set associated with event 4.*
- #define RTEMS_EVENT_5 0x00000020

  *This event set constant represents the bit in the event set associated with event 5.*
- #define RTEMS_EVENT_6 0x00000040

  *This event set constant represents the bit in the event set associated with event 6.*
- #define RTEMS_EVENT_7 0x00000080

  *This event set constant represents the bit in the event set associated with event 7.*
- #define RTEMS_EVENT_8 0x00000100

  *This event set constant represents the bit in the event set associated with event 8.*
- #define RTEMS_EVENT_9 0x00000200

  *This event set constant represents the bit in the event set associated with event 9.*
- #define RTEMS_EVENT_10 0x00000400

  *This event set constant represents the bit in the event set associated with event 10.*
- #define RTEMS_EVENT_11 0x00000800

  *This event set constant represents the bit in the event set associated with event 11.*
- #define RTEMS_EVENT_12 0x00001000

  *This event set constant represents the bit in the event set associated with event 12.*
- #define RTEMS_EVENT_13 0x00002000

  *This event set constant represents the bit in the event set associated with event 13.*
- #define RTEMS_EVENT_14 0x00004000

  *This event set constant represents the bit in the event set associated with event 14.*
- #define RTEMS_EVENT_15 0x00008000

  *This event set constant represents the bit in the event set associated with event 15.*
- #define RTEMS_EVENT_16 0x00010000

  *This event set constant represents the bit in the event set associated with event 16.*
- #define RTEMS_EVENT_17 0x00020000

  *This event set constant represents the bit in the event set associated with event 17.*
- #define RTEMS_EVENT_18 0x00040000

  *This event set constant represents the bit in the event set associated with event 18.*
- #define RTEMS_EVENT_19 0x00080000

  *This event set constant represents the bit in the event set associated with event 19.*
- #define RTEMS_EVENT_20 0x00100000

*This event set constant represents the bit in the event set associated with event 20.*

- #define RTEMS_EVENT_21 0x00200000

  *This event set constant represents the bit in the event set associated with event 21.*
- #define RTEMS_EVENT_22 0x00400000

  *This event set constant represents the bit in the event set associated with event 22.*
- #define RTEMS_EVENT_23 0x00800000

  *This event set constant represents the bit in the event set associated with event 23.*
- #define RTEMS_EVENT_24 0x01000000

  *This event set constant represents the bit in the event set associated with event 24.*
- #define RTEMS_EVENT_25 0x02000000

  *This event set constant represents the bit in the event set associated with event 25.*
- #define RTEMS_EVENT_26 0x04000000

  *This event set constant represents the bit in the event set associated with event 26.*
- #define RTEMS_EVENT_27 0x08000000

  *This event set constant represents the bit in the event set associated with event 27.*
- #define RTEMS_EVENT_28 0x10000000

  *This event set constant represents the bit in the event set associated with event 28.*
- #define RTEMS_EVENT_29 0x20000000

  *This event set constant represents the bit in the event set associated with event 29.*
- #define RTEMS_EVENT_30 0x40000000

  *This event set constant represents the bit in the event set associated with event 30.*
- #define RTEMS_EVENT_31 0x80000000

  *This event set constant represents the bit in the event set associated with event 31.*
- #define RTEMS_PENDING_EVENTS 0

  *This event set constant indicates that rtems_event_receive() shall return the set of pending events.*

## Typedefs

- typedef uint32_t rtems_event_set

  *This integer type represents a bit field which can hold exactly 32 individual events.*

## Functions

- rtems_status_code rtems_event_send (rtems_id id, rtems_event_set event_in)

  *Sends the event set to the task.*
- rtems_status_code rtems_event_receive (rtems_event_set event_in, rtems_option option_set, rtems_interval ticks, rtems_event_set *event_out)

  *Receives or gets an event set from the calling task.*

### 8.62.1 Detailed Description

The Event Manager provides a high performance method of inter-task communication and synchronization.

### 8.62.2 Macro Definition Documentation

#### 8.62.2.1 RTEMS_ALL_EVENTS

```
#define RTEMS_ALL_EVENTS 0xffffffff
```

This event set constant represents all events of an event set.

This event set constant is equal to the bitwise or of RTEMS_EVENT_0, RTEMS_EVENT_1, RTEMS_EVENT_2, RTEMS_EVENT_3, RTEMS_EVENT_4, RTEMS_EVENT_5, RTEMS_EVENT_6, RTEMS_EVENT_7, RTEMS_EVENT_8, RTEMS_EVENT_9, RTEMS_EVENT_10, RTEMS_EVENT_11, RTEMS_EVENT_12, RTEMS_EVENT_13, RTEMS_EVENT_14, RTEMS_EVENT_15, RTEMS_EVENT_16, RTEMS_EVENT_17, RTEMS_EVENT_18, RTEMS_EVENT_19, RTEMS_EVENT_20, RTEMS_EVENT_21, RTEMS_EVENT_22, RTEMS_EVENT_23, RTEMS_EVENT_24, RTEMS_EVENT_25, RTEMS_EVENT_26, RTEMS_EVENT_27, RTEMS_EVENT_28, RTEMS_EVENT_29, RTEMS_EVENT_30, and RTEMS_EVENT_31.

Definition at line 93 of file event.h.

### 8.62.3 Function Documentation

#### 8.62.3.1 rtems_event_receive()

```
rtems_status_code rtems_event_receive (
             rtems_event_set event_in,
             rtems_option option_set,
             rtems_interval ticks,
             rtems_event_set * event_out )
```

Receives or gets an event set from the calling task.

This directive can be used to

- get the pending events of the calling task, or
- receive events.

To **get the pending events** use the constant RTEMS_PENDING_EVENTS for the `event_in` parameter. The pending events are returned to the calling task but the event set of the task is left unaltered. The `option_set` and `ticks` parameters are ignored in this case. The directive returns immediately and does not block.

To **receive events** you have to define an input event condition and some options. The **option set** specified in `option_set` defines

- if the task will wait or poll for the events, and
- if the task wants to receive all or any of the input events.

The option set is built through a *bitwise or* of the option constants described below.

The task can **wait** or **poll** for the events.

- **Waiting** for events is the default and can be emphasized through the use of the RTEMS_WAIT option. The `ticks` parameter defines how long the task is willing to wait. Use RTEMS_NO_TIMEOUT to wait potentially forever, otherwise set a timeout interval in clock ticks.
- Not waiting for events (**polling**) is selected by the RTEMS_NO_WAIT option. If this option is defined, then the `ticks` parameter is ignored.

The task can receive **all** or **any** of the input events specified in `event_in`.

- Receiving **all** input events is the default and can be emphasized through the use of the RTEMS_EVENT_ALL option.
- Receiving **any** of the input events is selected by the RTEMS_EVENT_ANY option.

This directive shall be called by a task. Calling this directive from interrupt context is undefined behaviour.

This directive only affects the events specified in `event_in`. Any pending events that do not correspond to any of the events specified in `event_in` will be left pending.

To receive all events use the event set constant RTEMS_ALL_EVENTS for the `event_in` parameter. Do not confuse this event set constant with the directive option RTEMS_EVENT_ALL.

A task can **receive all of the pending events** by calling the directive with a value of RTEMS_ALL_EVENTS for the `event_in` parameter and the bitwise or of the RTEMS_NO_WAIT and RTEMS_EVENT_ANY options for the `option_set` parameter. The pending events are returned and the event set of the task is cleared. If no events are pending then the RTEMS_UNSATISFIED status code will be returned.

**Parameters**

| | |
|---|---|
| *event_in* | is the event set of interest. Use RTEMS_PENDING_EVENTS to get the pending events. |
| *option_set* | is the option set. |
| *ticks* | is the timeout in clock ticks if the RTEMS_WAIT option is set. Use RTEMS_NO_TIMEOUT to wait potentially forever. |
| *event_out* | is the pointer to an event set. The received or pending events are stored in the referenced event set if the operation was successful. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The `event_out` parameter was NULL. |
| *RTEMS_UNSATISFIED* | The events of interest were not immediately available. |
| *RTEMS_TIMEOUT* | The events of interest were not available within the specified timeout interval. |

Definition at line 28 of file eventreceive.c.

#### 8.62.3.2 rtems_event_send()

```
rtems_status_code rtems_event_send (
            rtems_id id,
            rtems_event_set event_in )
```

Sends the event set to the task.

This directive sends the event set, `event_in`, to the target task identified by `id`. Based upon the state of the target task, one of the following situations applies:

- The target task is blocked waiting for events, then
  - if the waiting task's input event condition is satisfied, then the task is made ready for execution, or
  - otherwise, the event set is posted but left pending and the task remains blocked.
- The target task is not waiting for events, then the event set is posted and left pending.

Events can be sent by tasks or an ISR.

Specifying RTEMS_SELF for `id` results in the event set being sent to the calling task.

The event set to send shall be built by a *bitwise or* of the desired events. The set of valid events is RTEMS_EVENT_0 through RTEMS_EVENT_31. If an event is not explicitly specified in the set, then it is not present.

Identical events sent to a task are not queued. In other words, the second, and subsequent, posting of an event to a task before it can perform an rtems_event_receive() has no effect.

The calling task will be preempted if it has preemption enabled and a higher priority task is unblocked as the result of this directive.

Sending an event set to a global task which does not reside on the local node will generate a request telling the remote node to send the event set to the appropriate task.

**Parameters**

| | |
|---|---|
| *id* | is the identifier of the target task to receive the event set. |
| *event_in* | is the event set to send. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ID* | There was no task with the specified identifier. |

Definition at line 27 of file eventsend.c.

# 8.63 Event Recording Configuration

## Macros

- #define CONFIGURE_RECORD_EXTENSIONS_ENABLED

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_RECORD_FATAL_DUMP_BASE64

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_RECORD_FATAL_DUMP_BASE64_ZLIB

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_RECORD_PER_PROCESSOR_ITEMS

  *This configuration option is an integer define.*

## 8.63.1 Detailed Description

This section describes configuration options related to the event recording.

## 8.63.2 Macro Definition Documentation

### 8.63.2.1 CONFIGURE_RECORD_EXTENSIONS_ENABLED

```
#define CONFIGURE_RECORD_EXTENSIONS_ENABLED
```

This configuration option is a boolean feature define.

In case

- this configuration option is defined
- and CONFIGURE_RECORD_PER_PROCESSOR_ITEMS is properly defined,

then the event record extensions are enabled.

#### Default Configuration

If this configuration option is undefined, then the described feature is not enabled.

#### Notes

The record extensions capture thread create, start, restart, delete, switch, begin, exitted and terminate events.

Definition at line 1754 of file appl-config.h.

#### 8.63.2.2 CONFIGURE_RECORD_FATAL_DUMP_BASE64

```
#define CONFIGURE_RECORD_FATAL_DUMP_BASE64
```

This configuration option is a boolean feature define.

In case

- this configuration option is defined
- and CONFIGURE_RECORD_PER_PROCESSOR_ITEMS is properly defined,
- and CONFIGURE_RECORD_FATAL_DUMP_BASE64_ZLIB is undefined,

then the event records are dumped in Base64 encoding in a fatal error extension (see Announcing a Fatal Error).

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

This extension can be used to produce crash dumps.

Definition at line 1781 of file appl-config.h.

#### 8.63.2.3 CONFIGURE_RECORD_FATAL_DUMP_BASE64_ZLIB

```
#define CONFIGURE_RECORD_FATAL_DUMP_BASE64_ZLIB
```

This configuration option is a boolean feature define.

In case

- this configuration option is defined
- and CONFIGURE_RECORD_PER_PROCESSOR_ITEMS is properly defined,

then the event records are compressed by zlib and dumped in Base64 encoding in a fatal error extension (see Announcing a Fatal Error).

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

The zlib compression needs about 512KiB of RAM. This extension can be used to produce crash dumps.

Definition at line 1807 of file appl-config.h.

#### 8.63.2.4 CONFIGURE_RECORD_PER_PROCESSOR_ITEMS

```
#define CONFIGURE_RECORD_PER_PROCESSOR_ITEMS
```

This configuration option is an integer define.

The value of this configuration option defines the event record item count per processor.

**Default Value**

The default value is 0.

**Value Constraints**

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 16.
- It shall be less than or equal to SIZE_MAX.
- It shall be a power of two.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.

**Notes**

The event record buffers are statically allocated for each configured processor (CONFIGURE_MAXIMUM_PROCESSORS). If the value of this configuration option is zero, then nothing is allocated.

Definition at line 1842 of file appl-config.h.

## 8.64 Fatal Error Manager

The Fatal Error Manager processes all fatal or irrecoverable errors and other sources of system termination (for example after `exit()`). Fatal errors are identified by the fatal source and error code pair.

### Classes

- struct rtems_assert_context
  %

### Typedefs

- typedef CPU_Exception_frame rtems_exception_frame
  %

### Functions

- RTEMS_NO_RETURN void rtems_fatal_error_occurred (uint32_t the_error)
  %
- static void rtems_exception_frame_print (const rtems_exception_frame *frame)
  %
- static RTEMS_NO_RETURN void rtems_fatal (rtems_fatal_source fatal_source, rtems_fatal_code error_↩ code)
  %
- const char * rtems_internal_error_text (rtems_fatal_code error)
  %
- RTEMS_NO_RETURN RTEMS_PRINTFLIKE (1, 2) void rtems_panic(const char *fmt
  %
- RTEMS_NO_RETURN const char * rtems_fatal_source_text (rtems_fatal_source source)
  %

### 8.64.1 Detailed Description

The Fatal Error Manager processes all fatal or irrecoverable errors and other sources of system termination (for example after `exit()`). Fatal errors are identified by the fatal source and error code pair.

### 8.64.2 Function Documentation

#### 8.64.2.1 rtems_exception_frame_print()

```
static void rtems_exception_frame_print (
            const rtems_exception_frame * frame )  [inline], [static]
```

%

**Parameters**

| | |
|---|---|
| *frame* | % |

Definition at line 142 of file fatal.h.

#### 8.64.2.2 rtems_fatal()

```
static RTEMS_NO_RETURN void rtems_fatal (
            rtems_fatal_source fatal_source,
            rtems_fatal_code error_code )  [inline], [static]
```

%

**Parameters**

| | |
|---|---|
| *fatal_source* | % |
| *error_code* | % |

Definition at line 160 of file fatal.h.

#### 8.64.2.3 rtems_fatal_error_occurred()

```
RTEMS_NO_RETURN void rtems_fatal_error_occurred (
            uint32_t the_error )
```

%

**Parameters**

| | |
|---|---|
| *the_error* | % |

#### 8.64.2.4 rtems_fatal_source_text()

```
RTEMS_NO_RETURN const char* rtems_fatal_source_text (
            rtems_fatal_source source )
```

%

**Parameters**

| | |
|---|---|
| *source* | % |

#### 8.64.2.5 rtems_internal_error_text()

```
const char* rtems_internal_error_text (
            rtems_fatal_code error )
```

%

**Parameters**

| | |
|---|---|
| *error* | % |

#### 8.64.2.6 RTEMS_PRINTFLIKE()

```
RTEMS_NO_RETURN RTEMS_PRINTFLIKE (
            1 ,
            2  ) const
```

%

**Parameters**

| | |
|---|---|
| *fmt* | % |
| ... | % |

# 8.65 File System Node Handler

File system node handler.

## Classes

- struct _rtems_filesystem_file_handlers_r

  *File system node operations table.*

## Typedefs

- typedef int(∗ rtems_filesystem_open_t) (rtems_libio_t ∗iop, const char ∗path, int oflag, mode_t mode)

  *Opens a node.*
- typedef int(∗ rtems_filesystem_close_t) (rtems_libio_t ∗iop)

  *Closes a node.*
- typedef ssize_t(∗ rtems_filesystem_read_t) (rtems_libio_t ∗iop, void ∗buffer, size_t count)

  *Reads from a node.*
- typedef ssize_t(∗ rtems_filesystem_readv_t) (rtems_libio_t ∗iop, const struct iovec ∗iov, int iovcnt, ssize_t total)

  *Reads an IO vector from a node.*
- typedef ssize_t(∗ rtems_filesystem_write_t) (rtems_libio_t ∗iop, const void ∗buffer, size_t count)

  *Writes to a node.*
- typedef ssize_t(∗ rtems_filesystem_writev_t) (rtems_libio_t ∗iop, const struct iovec ∗iov, int iovcnt, ssize_t total)

  *Writes an IO vector to a node.*
- typedef int(∗ rtems_filesystem_ioctl_t) (rtems_libio_t ∗iop, ioctl_command_t request, void ∗buffer)

  *IO control of a node.*
- typedef off_t(∗ rtems_filesystem_lseek_t) (rtems_libio_t ∗iop, off_t offset, int whence)

  *Moves the read/write file offset.*
- typedef int(∗ rtems_filesystem_fstat_t) (const rtems_filesystem_location_info_t ∗loc, struct stat ∗buf)

  *Gets a node status.*
- typedef int(∗ rtems_filesystem_ftruncate_t) (rtems_libio_t ∗iop, off_t length)

  *Truncates a file to a specified length.*
- typedef int(∗ rtems_filesystem_fsync_t) (rtems_libio_t ∗iop)

  *Synchronizes changes to a file.*
- typedef int(∗ rtems_filesystem_fdatasync_t) (rtems_libio_t ∗iop)

  *Synchronizes the data of a file.*
- typedef int(∗ rtems_filesystem_fcntl_t) (rtems_libio_t ∗iop, int cmd)

  *File control.*
- typedef int(∗ rtems_filesystem_poll_t) (rtems_libio_t ∗iop, int events)

  *Poll and select support.*
- typedef int(∗ rtems_filesystem_kqfilter_t) (rtems_libio_t ∗iop, struct knote ∗kn)

  *Kernel event filter support.*
- typedef int(∗ rtems_filesystem_mmap_t) (rtems_libio_t ∗iop, void ∗∗addr, size_t len, int prot, off_t off)

  *MMAP support.*

## Functions

- int rtems_filesystem_default_open (rtems_libio_t *iop, const char *path, int oflag, mode_t mode)
- int rtems_filesystem_default_close (rtems_libio_t *iop)
- ssize_t rtems_filesystem_default_read (rtems_libio_t *iop, void *buffer, size_t count)
- ssize_t rtems_filesystem_default_readv (rtems_libio_t *iop, const struct iovec *iov, int iovcnt, ssize_t total)

  *Calls the read handler for each IO vector entry.*
- ssize_t rtems_filesystem_default_write (rtems_libio_t *iop, const void *buffer, size_t count)
- ssize_t rtems_filesystem_default_writev (rtems_libio_t *iop, const struct iovec *iov, int iovcnt, ssize_t total)

  *Calls the write handler for each IO vector entry.*
- int rtems_filesystem_default_ioctl (rtems_libio_t *iop, ioctl_command_t request, void *buffer)
- off_t rtems_filesystem_default_lseek (rtems_libio_t *iop, off_t offset, int whence)
- off_t rtems_filesystem_default_lseek_directory (rtems_libio_t *iop, off_t offset, int whence)

  *An offset 0 with a whence of SEEK_SET will perform a directory rewind operation.*
- off_t rtems_filesystem_default_lseek_file (rtems_libio_t *iop, off_t offset, int whence)

  *Default lseek() handler for files.*
- int rtems_filesystem_default_fstat (const rtems_filesystem_location_info_t *loc, struct stat *buf)

  *Sets the mode to S_IRWXU | S_IRWXG | S_IRWXO.*
- int rtems_filesystem_default_ftruncate (rtems_libio_t *iop, off_t length)
- int rtems_filesystem_default_ftruncate_directory (rtems_libio_t *iop, off_t length)
- int rtems_filesystem_default_fsync_or_fdatasync (rtems_libio_t *iop)
- int rtems_filesystem_default_fsync_or_fdatasync_success (rtems_libio_t *iop)
- int rtems_filesystem_default_fcntl (rtems_libio_t *iop, int cmd)
- int rtems_filesystem_default_poll (rtems_libio_t *iop, int events)

  *Default poll handler.*
- int rtems_filesystem_default_kqfilter (rtems_libio_t *iop, struct knote *kn)

  *Default kernel event filter handler.*
- int rtems_filesystem_default_mmap (rtems_libio_t *iop, void **addr, size_t len, int prot, off_t off)

  *Default MMAP handler.*

## Variables

- const rtems_filesystem_file_handlers_r rtems_filesystem_handlers_default

  *File system node handler table with default node handlers.*

### 8.65.1 Detailed Description

File system node handler.

### 8.65.2 Typedef Documentation

#### 8.65.2.1 rtems_filesystem_close_t

```
typedef int(* rtems_filesystem_close_t) (rtems_libio_t *iop)
```

Closes a node.

**Parameters**

| | | |
|---|---|---|
| in,out | *iop* | The IO pointer. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_close().

Definition at line 742 of file libio.h.

### 8.65.2.2 rtems_filesystem_fcntl_t

```
typedef int(* rtems_filesystem_fcntl_t) (rtems_libio_t *iop, int cmd)
```

File control.

**Parameters**

| | | |
|---|---|---|
| in,out | *iop* | The IO pointer. |
| in | *cmd* | Control command. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *errno* | An error occurred. This value is assigned to errno. |

**See also**

rtems_filesystem_default_fcntl().

Definition at line 944 of file libio.h.

### 8.65.2.3 rtems_filesystem_fdatasync_t

```
typedef int(* rtems_filesystem_fdatasync_t) (rtems_libio_t *iop)
```

Synchronizes the data of a file.

**Parameters**

| | | |
|---|---|---|
| in,out | *iop* | The IO pointer. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_fsync_or_fdatasync() and rtems_filesystem_default_fsync_or_fdatasync_success().

Definition at line 929 of file libio.h.

#### 8.65.2.4 rtems_filesystem_fstat_t

```
typedef int(* rtems_filesystem_fstat_t) (const rtems_filesystem_location_info_t *loc, struct
stat *buf)
```

Gets a node status.

**Parameters**

| | | |
|---|---|---|
| in,out | *iop* | The IO pointer. |
| out | *stat* | The buffer to status information. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_fstat().

Definition at line 881 of file libio.h.

#### 8.65.2.5 rtems_filesystem_fsync_t

```
typedef int(* rtems_filesystem_fsync_t) (rtems_libio_t *iop)
```

Synchronizes changes to a file.

**Parameters**

| | | |
|---|---|---|
| in,out | *iop* | The IO pointer. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_fsync_or_fdatasync() and rtems_filesystem_default_fsync_or_fdatasync_success().

Definition at line 914 of file libio.h.

### 8.65.2.6 rtems_filesystem_ftruncate_t

```
typedef int(* rtems_filesystem_ftruncate_t) (rtems_libio_t *iop, off_t length)
```

Truncates a file to a specified length.

**Parameters**

| | | |
|---|---|---|
| in,out | *iop* | The IO pointer. |
| in | *length* | The new length in characters. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_ftruncate() and rtems_filesystem_default_ftruncate_directory().

Definition at line 898 of file libio.h.

### 8.65.2.7 rtems_filesystem_ioctl_t

```
typedef int(* rtems_filesystem_ioctl_t) (rtems_libio_t *iop, ioctl_command_t request, void
*buffer)
```

IO control of a node.

**Parameters**

| | | |
|---|---|---|
| in,out | *iop* | The IO pointer. |
| in | *request* | The IO control request. |
| in,out | *buffer* | The buffer for IO control request data. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_ioctl().

Definition at line 844 of file libio.h.

#### 8.65.2.8 rtems_filesystem_kqfilter_t

```
typedef int(* rtems_filesystem_kqfilter_t) (rtems_libio_t *iop, struct knote *kn)
```

Kernel event filter support.

**Parameters**

| | | |
|---|---|---|
| in,out | *iop* | The IO pointer. |
| in | *kn* | The kernel event note. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *error* | An error occurred. This is usually EINVAL. |

**See also**

rtems_filesystem_default_kqfilter().

Definition at line 975 of file libio.h.

#### 8.65.2.9 rtems_filesystem_lseek_t

```
typedef off_t(* rtems_filesystem_lseek_t) (rtems_libio_t *iop, off_t offset, int whence)
```

Moves the read/write file offset.

**Parameters**

| | | |
|---|---|---|
| in,out | *iop* | The IO pointer. |
| in | *offset* | The offset. |
| in | *whence* | The reference position for the offset. |

**Return values**

| | |
|---|---|
| *non-negative* | The new offset from the beginning of the file. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_lseek(), rtems_filesystem_default_lseek_file(), and rtems_filesystem_default_lseek_directory().

Definition at line 864 of file libio.h.

#### 8.65.2.10 rtems_filesystem_mmap_t

```
typedef int(* rtems_filesystem_mmap_t) (rtems_libio_t *iop, void **addr, size_t len, int prot,
off_t off)
```

MMAP support.

**Parameters**

| | | |
|---|---|---|
| in,out | *iop* | The IO pointer. |
| in,out | *addr* | The starting address of the mapped memory. |
| in | *len* | The maximum number of bytes to map. |
| in | *prot* | The desired memory protection. |
| in | *off* | The offset within the file descriptor to map. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *error* | An error occurred. This is usually EINVAL. |

**See also**

rtems_filesystem_default_mmap().

Definition at line 994 of file libio.h.

#### 8.65.2.11 rtems_filesystem_open_t

```
typedef int(* rtems_filesystem_open_t) (rtems_libio_t *iop, const char *path, int oflag, mode↩
_t mode)
```

Opens a node.

**Parameters**

| | | |
|---|---|---|
| in,out | *iop* | The IO pointer. |
| in | *path* | The path. |
| in | *oflag* | The open flags. |
| in | *mode* | Optional mode for node creation. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_open().

Definition at line 725 of file libio.h.

#### 8.65.2.12 rtems_filesystem_poll_t

```
typedef int(* rtems_filesystem_poll_t) (rtems_libio_t *iop, int events)
```

Poll and select support.

**Parameters**

| | | |
|---|---|---|
| in,out | *iop* | The IO pointer. |
| in | *events* | The poll events. |

**Returns**

The poll return events.

**See also**

rtems_filesystem_default_poll().

Definition at line 959 of file libio.h.

#### 8.65.2.13 rtems_filesystem_read_t

```
typedef ssize_t(* rtems_filesystem_read_t) (rtems_libio_t *iop, void *buffer, size_t count)
```

Reads from a node.

This handler is responsible to update the offset field of the IO descriptor.

**Parameters**

| | | |
|---|---|---|
| in,out | *iop* | The IO pointer. |
| out | *buffer* | The buffer for read data. |
| in | *count* | The size of the buffer in characters. |

**Return values**

| | |
|---|---|
| *non-negative* | Count of read characters. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_read().

Definition at line 760 of file libio.h.

#### 8.65.2.14 rtems_filesystem_readv_t

```
typedef ssize_t(* rtems_filesystem_readv_t) (rtems_libio_t *iop, const struct iovec *iov, int
iovcnt, ssize_t total)
```

Reads an IO vector from a node.

This handler is responsible to update the offset field of the IO descriptor.

**Parameters**

| | | |
|---|---|---|
| in,out | *iop* | The IO pointer. |
| in | *iov* | The IO vector with buffer for read data. The caller must ensure that the IO vector values are valid. |
| in | *iovcnt* | The count of buffers in the IO vector. |
| in | *total* | The total count of bytes in the buffers in the IO vector. |

**Return values**

| | |
|---|---|
| *non-negative* | Count of read characters. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_readv().

Definition at line 782 of file libio.h.

#### 8.65.2.15 rtems_filesystem_write_t

```
typedef ssize_t(* rtems_filesystem_write_t) (rtems_libio_t *iop, const void *buffer, size_↩
t count)
```

Writes to a node.

This handler is responsible to update the offset field of the IO descriptor.

**Parameters**

| | | |
|---|---|---|
| in,out | *iop* | The IO pointer. |
| out | *buffer* | The buffer for write data. |
| in | *count* | The size of the buffer in characters. |

**Return values**

| | |
|---|---|
| *non-negative* | Count of written characters. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_write().

Definition at line 803 of file libio.h.

#### 8.65.2.16 rtems_filesystem_writev_t

```
typedef ssize_t(* rtems_filesystem_writev_t) (rtems_libio_t *iop, const struct iovec *iov, int
iovcnt, ssize_t total)
```

Writes an IO vector to a node.

This handler is responsible to update the offset field of the IO descriptor.

**Parameters**

| | | |
|---|---|---|
| in,out | *iop* | The IO pointer. |
| in | *iov* | The IO vector with buffer for write data. The caller must ensure that the IO vector values are valid. |
| in | *iovcnt* | The count of buffers in the IO vector. |
| in | *total* | The total count of bytes in the buffers in the IO vector. |

**Return values**

| | |
|---|---|
| *non-negative* | Count of written characters. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_writev().

Definition at line 825 of file libio.h.

## 8.65.3 Function Documentation

### 8.65.3.1 rtems_filesystem_default_close()

```
int rtems_filesystem_default_close (
            rtems_libio_t * iop )
```

**Return values**

| | |
|---|---|
| *0* | Always. |

**See also**

rtems_filesystem_close_t.

### 8.65.3.2 rtems_filesystem_default_fcntl()

```
int rtems_filesystem_default_fcntl (
            rtems_libio_t * iop,
            int cmd )
```

**Return values**

| | |
|---|---|
| *0* | Always. |

**See also**

rtems_filesystem_fcntl_t.

#### 8.65.3.3 rtems_filesystem_default_fstat()

```
int rtems_filesystem_default_fstat (
              const rtems_filesystem_location_info_t * loc,
              struct stat * buf )
```

Sets the mode to S_IRWXU | S_IRWXG | S_IRWXO.

**Return values**

| | |
|---|---|
| *0* | Always. |

**See also**

rtems_filesystem_fstat_t.

#### 8.65.3.4 rtems_filesystem_default_fsync_or_fdatasync()

```
int rtems_filesystem_default_fsync_or_fdatasync (
              rtems_libio_t * iop )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to EINVAL. |

**See also**

rtems_filesystem_fsync_t and rtems_filesystem_fdatasync_t.

#### 8.65.3.5 rtems_filesystem_default_fsync_or_fdatasync_success()

```
int rtems_filesystem_default_fsync_or_fdatasync_success (
              rtems_libio_t * iop )
```

**Return values**

| | |
|---|---|
| *0* | Always. |

**See also**

rtems_filesystem_fsync_t and rtems_filesystem_fdatasync_t.

#### 8.65.3.6 rtems_filesystem_default_ftruncate()

```
int rtems_filesystem_default_ftruncate (
              rtems_libio_t * iop,
              off_t length )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to EINVAL. |

**See also**

rtems_filesystem_ftruncate_t.

#### 8.65.3.7 rtems_filesystem_default_ftruncate_directory()

```
int rtems_filesystem_default_ftruncate_directory (
              rtems_libio_t * iop,
              off_t length )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to EISDIR. |

**See also**

rtems_filesystem_ftruncate_t.

#### 8.65.3.8 rtems_filesystem_default_ioctl()

```
int rtems_filesystem_default_ioctl (
              rtems_libio_t * iop,
              ioctl_command_t request,
              void * buffer )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to ENOTTY. |

**See also**

rtems_filesystem_ioctl_t.

#### 8.65.3.9 rtems_filesystem_default_kqfilter()

```
int rtems_filesystem_default_kqfilter (
            rtems_libio_t * iop,
            struct knote * kn )
```

Default kernel event filter handler.

**Return values**

| | |
|---|---|
| *EINVAL* | Always. |

**See also**

rtems_filesystem_kqfilter_t.

#### 8.65.3.10 rtems_filesystem_default_lseek()

```
off_t rtems_filesystem_default_lseek (
            rtems_libio_t * iop,
            off_t offset,
            int whence )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to ESPIPE. |

**See also**

rtems_filesystem_lseek_t.

#### 8.65.3.11 rtems_filesystem_default_lseek_directory()

```
off_t rtems_filesystem_default_lseek_directory (
            rtems_libio_t * iop,
            off_t offset,
            int whence )
```

An offset 0 with a whence of SEEK_SET will perform a directory rewind operation.

This function has no protection against concurrent access.

**Return values**

| | |
|---|---|
| *-1* | The offset is not zero or the whence is not SEEK_SET. |
| *0* | Successful rewind operation. |

**See also**

rtems_filesystem_lseek_t.

### 8.65.3.12 rtems_filesystem_default_lseek_file()

```
off_t rtems_filesystem_default_lseek_file (
              rtems_libio_t * iop,
              off_t offset,
              int whence )
```

Default lseek() handler for files.

The fstat() handler will be used to obtain the file size in case whence is SEEK_END.

This function has no protection against concurrent access.

**Return values**

| | |
|---|---|
| *-1* | An error occurred. In case an integer overflow occurred, then the errno will be set to EOVERFLOW. In case the new offset is negative, then the errno will be set to EINVAL. In case the whence is SEEK_END and the fstat() handler to obtain the current file size returned an error status, then the errno will be set by the fstat() handler. |
| *offset* | The new offset. |

**See also**

rtems_filesystem_lseek_t.

### 8.65.3.13 rtems_filesystem_default_mmap()

```
int rtems_filesystem_default_mmap (
              rtems_libio_t * iop,
              void ** addr,
              size_t len,
              int prot,
              off_t off )
```

Default MMAP handler.

**Return values**

| | |
|---|---|
| *ENOTSUP* | Always. |

**See also**

rtems_filesystem_mmap_t.

#### 8.65.3.14 rtems_filesystem_default_open()

```
int rtems_filesystem_default_open (
            rtems_libio_t * iop,
            const char * path,
            int oflag,
            mode_t mode )
```

**Return values**

| | |
|---|---|
| *0* | Always. |

**See also**

rtems_filesystem_open_t.

#### 8.65.3.15 rtems_filesystem_default_poll()

```
int rtems_filesystem_default_poll (
            rtems_libio_t * iop,
            int events )
```

Default poll handler.

**Return values**

| | |
|---|---|
| *POLLERR* | Always. |

**See also**

rtems_filesystem_poll_t.

#### 8.65.3.16 rtems_filesystem_default_read()

```
ssize_t rtems_filesystem_default_read (
            rtems_libio_t * iop,
            void * buffer,
            size_t count )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to ENOTSUP. |

**See also**

rtems_filesystem_read_t.

#### 8.65.3.17 rtems_filesystem_default_readv()

```
ssize_t rtems_filesystem_default_readv (
            rtems_libio_t * iop,
            const struct iovec * iov,
            int iovcnt,
            ssize_t total )
```

Calls the read handler for each IO vector entry.

**See also**

rtems_filesystem_readv_t.

#### 8.65.3.18 rtems_filesystem_default_write()

```
ssize_t rtems_filesystem_default_write (
            rtems_libio_t * iop,
            const void * buffer,
            size_t count )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to ENOTSUP. |

**See also**

rtems_filesystem_write_t.

#### 8.65.3.19 rtems_filesystem_default_writev()

```
ssize_t rtems_filesystem_default_writev (
            rtems_libio_t * iop,
```

```
    const struct iovec * iov,
    int iovcnt,
    ssize_t total )
```

Calls the write handler for each IO vector entry.

**See also**

rtems_filesystem_write_t.

# 8.66 File System Operations

File system operations.

## Classes

- struct rtems_filesystem_eval_path_context_t

  *Path evaluation context.*
- struct _rtems_filesystem_operations_table

  *File system operations table.*

## Typedefs

- typedef void(∗ rtems_filesystem_mt_entry_lock_t) (const rtems_filesystem_mount_table_entry_t ∗mt_entry)

  *Locks a file system instance.*
- typedef void(∗ rtems_filesystem_mt_entry_unlock_t) (const rtems_filesystem_mount_table_entry_t ∗mt_↩ entry)

  *Unlocks a file system instance.*
- typedef void(∗ rtems_filesystem_eval_path_t) (rtems_filesystem_eval_path_context_t ∗ctx)

  *Path evaluation.*
- typedef int(∗ rtems_filesystem_link_t) (const rtems_filesystem_location_info_t ∗parentloc, const rtems_filesystem_location_info ∗targetloc, const char ∗name, size_t namelen)

  *Creates a new link for the existing file.*
- typedef int(∗ rtems_filesystem_fchmod_t) (const rtems_filesystem_location_info_t ∗loc, mode_t mode)

  *Changes the mode of a node.*
- typedef int(∗ rtems_filesystem_chown_t) (const rtems_filesystem_location_info_t ∗loc, uid_t owner, gid_↩ t group)

  *Changes owner and group of a node.*
- typedef int(∗ rtems_filesystem_clonenode_t) (rtems_filesystem_location_info_t ∗loc)

  *Clones a location.*
- typedef void(∗ rtems_filesystem_freenode_t) (const rtems_filesystem_location_info_t ∗loc)

  *Frees the location of a node.*
- typedef int(∗ rtems_filesystem_mount_t) (rtems_filesystem_mount_table_entry_t ∗mt_entry)

  *Mounts a file system instance in a mount point (directory).*
- typedef int(∗ rtems_filesystem_fsmount_me_t) (rtems_filesystem_mount_table_entry_t ∗mt_entry, const void ∗data)

  *Initializes a file system instance.*
- typedef int(∗ rtems_filesystem_unmount_t) (rtems_filesystem_mount_table_entry_t ∗mt_entry)

  *Unmounts a file system instance in a mount point (directory).*
- typedef void(∗ rtems_filesystem_fsunmount_me_t) (rtems_filesystem_mount_table_entry_t ∗mt_entry)

  *Destroys a file system instance.*
- typedef bool(∗ rtems_filesystem_are_nodes_equal_t) (const rtems_filesystem_location_info_t ∗a, const rtems_filesystem_location_info_t ∗b)

  *Tests if the node of one location is equal to the node of the other location.*
- typedef int(∗ rtems_filesystem_mknod_t) (const rtems_filesystem_location_info_t ∗parentloc, const char ∗name, size_t namelen, mode_t mode, dev_t dev)

  *Creates a new node.*
- typedef int(∗ rtems_filesystem_rmnod_t) (const rtems_filesystem_location_info_t ∗parentloc, const rtems_filesystem_location_info_t ∗loc)

  *Removes a node.*

- typedef int(∗ rtems_filesystem_utime_t) (const rtems_filesystem_location_info_t ∗loc, time_t actime, time_t modtime)

  *Set node access and modification times.*
- typedef int(∗ rtems_filesystem_symlink_t) (const rtems_filesystem_location_info_t ∗parentloc, const char ∗name, size_t namelen, const char ∗target)

  *Makes a symbolic link to a node.*
- typedef ssize_t(∗ rtems_filesystem_readlink_t) (const rtems_filesystem_location_info_t ∗loc, char ∗buf, size_t bufsize)

  *Reads the contents of a symbolic link.*
- typedef int(∗ rtems_filesystem_rename_t) (const rtems_filesystem_location_info_t ∗oldparentloc, const rtems_filesystem_location_info_t ∗oldloc, const rtems_filesystem_location_info_t ∗newparentloc, const char ∗name, size_t namelen)

  *Renames a node.*
- typedef int(∗ rtems_filesystem_statvfs_t) (const rtems_filesystem_location_info_t ∗loc, struct statvfs ∗buf)

  *Gets file system information.*

## Functions

- void rtems_filesystem_default_lock (const rtems_filesystem_mount_table_entry_t ∗mt_entry)

  *Obtains the IO library mutex.*
- void rtems_filesystem_default_unlock (const rtems_filesystem_mount_table_entry_t ∗mt_entry)

  *Releases the IO library mutex.*
- void rtems_filesystem_default_eval_path (rtems_filesystem_eval_path_context_t ∗ctx)

  *Terminates the path evaluation and replaces the current location with the null location.*
- int rtems_filesystem_default_link (const rtems_filesystem_location_info_t ∗parentloc, const rtems_filesystem_location_info_t ∗targetloc, const char ∗name, size_t namelen)
- bool rtems_filesystem_default_are_nodes_equal (const rtems_filesystem_location_info_t ∗a, const rtems_filesystem_location_info_t ∗b)

  *Tests if the node access pointer of one location is equal to the node access pointer of the other location.*
- int rtems_filesystem_default_mknod (const rtems_filesystem_location_info_t ∗parentloc, const char ∗name, size_t namelen, mode_t mode, dev_t dev)
- int rtems_filesystem_default_rmnod (const rtems_filesystem_location_info_t ∗parentloc, const rtems_filesystem_location_info_t ∗loc)
- int rtems_filesystem_default_fchmod (const rtems_filesystem_location_info_t ∗loc, mode_t mode)
- int rtems_filesystem_default_chown (const rtems_filesystem_location_info_t ∗loc, uid_t owner, gid_t group)
- int rtems_filesystem_default_clonenode (rtems_filesystem_location_info_t ∗loc)
- void rtems_filesystem_default_freenode (const rtems_filesystem_location_info_t ∗loc)
- int rtems_filesystem_default_mount (rtems_filesystem_mount_table_entry_t ∗mt_entry)
- int rtems_filesystem_default_unmount (rtems_filesystem_mount_table_entry_t ∗mt_entry)
- void rtems_filesystem_default_fsunmount (rtems_filesystem_mount_table_entry_t ∗mt_entry)
- int rtems_filesystem_default_utime (const rtems_filesystem_location_info_t ∗loc, time_t actime, time_t modtime)
- int rtems_filesystem_default_symlink (const rtems_filesystem_location_info_t ∗parentloc, const char ∗name, size_t namelen, const char ∗target)
- ssize_t rtems_filesystem_default_readlink (const rtems_filesystem_location_info_t ∗loc, char ∗buf, size_t bufsize)
- int rtems_filesystem_default_rename (const rtems_filesystem_location_info_t ∗oldparentloc, const rtems_filesystem_location_info_t ∗oldloc, const rtems_filesystem_location_info_t ∗newparentloc, const char ∗name, size_t namelen)
- int rtems_filesystem_default_statvfs (const rtems_filesystem_location_info_t ∗loc, struct statvfs ∗buf)

## Variables

- const rtems_filesystem_operations_table rtems_filesystem_operations_default

  *File system operations table with default operations.*

## 8.66.1 Detailed Description

File system operations.

## 8.66.2 Typedef Documentation

### 8.66.2.1 rtems_filesystem_are_nodes_equal_t

```
typedef bool(* rtems_filesystem_are_nodes_equal_t) (const rtems_filesystem_location_info_t *a,
const rtems_filesystem_location_info_t *b)
```

Tests if the node of one location is equal to the node of the other location.

The caller ensures that both nodes are within the same file system instance.

**Parameters**

| | | |
|---|---|---|
| in | *a* | The one location. |
| in | *b* | The other location. |

**Return values**

| | |
|---|---|
| *true* | The nodes of the locations are equal. |
| *false* | Otherwise. |

**See also**

rtems_filesystem_default_are_nodes_equal().

Definition at line 329 of file libio.h.

### 8.66.2.2 rtems_filesystem_chown_t

```
typedef int(* rtems_filesystem_chown_t) (const rtems_filesystem_location_info_t *loc, uid_↩
t owner, gid_t group)
```

Changes owner and group of a node.

**Parameters**

| | | |
|---|---|---|
| in | *loc* | The location of the node. |
| in | *owner* | User ID for the node. |
| in | *group* | Group ID for the node. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_chown().

Definition at line 209 of file libio.h.

#### 8.66.2.3 rtems_filesystem_clonenode_t

```
typedef int(* rtems_filesystem_clonenode_t) (rtems_filesystem_location_info_t *loc)
```

Clones a location.

The location is initialized with a bitwise copy of an existing location. The caller must ensure that this location is protected from a release during the clone operation. After a successful clone operation the clone will be added to the location chain of the corresponding mount table entry.

**Parameters**

| | | |
|---|---|---|
| in,out | *loc* | Location to clone. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_clonenode().

Definition at line 230 of file libio.h.

#### 8.66.2.4 rtems_filesystem_eval_path_t

```
typedef void(* rtems_filesystem_eval_path_t) (rtems_filesystem_eval_path_context_t *ctx)
```

Path evaluation.

**Parameters**

| | | |
|---|---|---|
| in,out | *ctx* | The path evaluation context. |

**See also**

rtems_filesystem_default_eval_path().

Definition at line 157 of file libio.h.

### 8.66.2.5 rtems_filesystem_fchmod_t

```
typedef int(* rtems_filesystem_fchmod_t) (const rtems_filesystem_location_info_t *loc, mode_t
mode)
```

Changes the mode of a node.

**Parameters**

| | | |
|---|---|---|
| in | *loc* | The location of the node. |
| in | *mode* | The new mode of the node |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_fchmod().

Definition at line 192 of file libio.h.

### 8.66.2.6 rtems_filesystem_freenode_t

```
typedef void(* rtems_filesystem_freenode_t) (const rtems_filesystem_location_info_t *loc)
```

Frees the location of a node.

**Parameters**

| | | |
|---|---|---|
| in | *loc* | The location of the node. |

**See also**

rtems_filesystem_default_freenode().

Definition at line 241 of file libio.h.

#### 8.66.2.7 rtems_filesystem_fsmount_me_t

```
typedef int(* rtems_filesystem_fsmount_me_t) (rtems_filesystem_mount_table_entry_t *mt_entry,
const void *data)
```

Initializes a file system instance.

This function must initialize the file system root node in the mount table entry.

**Parameters**

| | | |
|---|---|---|
| in | *mt_entry* | The mount table entry. |
| in | *data* | The data provided by the user. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

Definition at line 277 of file libio.h.

#### 8.66.2.8 rtems_filesystem_fsunmount_me_t

```
typedef void(* rtems_filesystem_fsunmount_me_t) (rtems_filesystem_mount_table_entry_t *mt_↩
entry)
```

Destroys a file system instance.

The mount point node location of the mount table entry is invalid. This handler must free the file system root location and all remaining resources of the file system instance.

**Parameters**

| | | |
|---|---|---|
| in | *mt_entry* | The mount table entry. |

**See also**

rtems_filesystem_default_fsunmount().

Definition at line 311 of file libio.h.

### 8.66.2.9 rtems_filesystem_link_t

```
typedef int(* rtems_filesystem_link_t) (const rtems_filesystem_location_info_t *parentloc,
const rtems_filesystem_location_info_t *targetloc, const char *name, size_t namelen)
```

Creates a new link for the existing file.

**Parameters**

| | | |
|---|---|---|
| in | *parentloc* | The location of the parent of the new link. |
| in | *targetloc* | The location of the target file. |
| in | *name* | Name for the new link. |
| in | *namelen* | Length of the name for the new link in characters. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_link().

Definition at line 174 of file libio.h.

### 8.66.2.10 rtems_filesystem_mknod_t

```
typedef int(* rtems_filesystem_mknod_t) (const rtems_filesystem_location_info_t *parentloc,
const char *name, size_t namelen, mode_t mode, dev_t dev)
```

Creates a new node.

This handler should create a new node according to the parameters.

**Parameters**

| | | |
|---|---|---|
| in | *parentloc* | The location of the parent of the new node. |
| in | *name* | Name for the new node. |
| in | *namelen* | Length of the name for the new node in characters. |
| in | *mode* | Mode for the new node. |
| in | *dev* | Optional device identifier for the new node. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_mknod().

Definition at line 350 of file libio.h.

#### 8.66.2.11 rtems_filesystem_mount_t

```
typedef int(* rtems_filesystem_mount_t) (rtems_filesystem_mount_table_entry_t *mt_entry)
```

Mounts a file system instance in a mount point (directory).

The mount point belongs to the file system instance of the handler and is specified by a field of the mount table entry. The handler must check that the mount point is capable of mounting a file system instance. This is the last step during the mount process. The file system instance is fully initialized at this point.

**Parameters**

| | | |
|---|---|---|
| in | *mt_entry* | The mount table entry. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_mount().

Definition at line 261 of file libio.h.

#### 8.66.2.12 rtems_filesystem_mt_entry_lock_t

```
typedef void(* rtems_filesystem_mt_entry_lock_t) (const rtems_filesystem_mount_table_entry_t
*mt_entry)
```

Locks a file system instance.

This lock must allow nesting.

**Parameters**

| | | |
|---|---|---|
| in,out | *mt_entry* | The mount table entry of the file system instance. |

**See also**

rtems_filesystem_default_lock().

Definition at line 66 of file libio.h.

### 8.66.2.13 rtems_filesystem_mt_entry_unlock_t

```
typedef void(* rtems_filesystem_mt_entry_unlock_t) (const rtems_filesystem_mount_table_entry_t
*mt_entry)
```

Unlocks a file system instance.

**Parameters**

| | | |
|---|---|---|
| in,out | *mt_entry* | The mount table entry of the file system instance. |

**See also**

rtems_filesystem_default_unlock().

Definition at line 77 of file libio.h.

### 8.66.2.14 rtems_filesystem_readlink_t

```
typedef ssize_t(* rtems_filesystem_readlink_t) (const rtems_filesystem_location_info_t *loc,
char *buf, size_t bufsize)
```

Reads the contents of a symbolic link.

**Parameters**

| | | |
|---|---|---|
| in | *loc* | The location of the symbolic link. |
| out | *buf* | The buffer for the contents. |
| in | *bufsize* | The size of the buffer in characters. |

**Return values**

| | |
|---|---|
| *non-negative* | Size of the actual contents in characters. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_readlink().

Definition at line 425 of file libio.h.

#### 8.66.2.15 rtems_filesystem_rename_t

```
typedef int(* rtems_filesystem_rename_t) (const rtems_filesystem_location_info_t *oldparentloc,
const rtems_filesystem_location_info_t *oldloc, const rtems_filesystem_location_info_t *newparentloc,
const char *name, size_t namelen)
```

Renames a node.

**Parameters**

| | | |
|---|---|---|
| in | *oldparentloc* | The location of the parent of the old node. |
| in | *oldloc* | The location of the old node. |
| in | *newparentloc* | The location of the parent of the new node. |
| in | *name* | Name for the new node. |
| in | *namelen* | Length of the name for the new node in characters. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_rename().

Definition at line 445 of file libio.h.

#### 8.66.2.16 rtems_filesystem_rmnod_t

```
typedef int(* rtems_filesystem_rmnod_t) (const rtems_filesystem_location_info_t *parentloc,
const rtems_filesystem_location_info_t *loc)
```

Removes a node.

**Parameters**

| | | |
|---|---|---|
| in | *parentloc* | The location of the parent of the node. |
| in | *loc* | The location of the node. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_rmnod().

Definition at line 369 of file libio.h.

### 8.66.2.17 rtems_filesystem_statvfs_t

```
typedef int(* rtems_filesystem_statvfs_t) (const rtems_filesystem_location_info_t *loc, struct statvfs *buf)
```

Gets file system information.

**Parameters**

| | | |
|---|---|---|
| in | *loc* | The location of a node. |
| out | *buf* | Buffer for file system information. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_statvfs().

Definition at line 464 of file libio.h.

### 8.66.2.18 rtems_filesystem_symlink_t

```
typedef int(* rtems_filesystem_symlink_t) (const rtems_filesystem_location_info_t *parentloc, const char *name, size_t namelen, const char *target)
```

Makes a symbolic link to a node.

**Parameters**

| | | |
|---|---|---|
| in | *parentloc* | The location of the parent of the new symbolic link. |
| in | *name* | Name for the new symbolic link. |
| in | *namelen* | Length of the name for the new symbolic link in characters. |
| in | *target* | Contents for the symbolic link. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_symlink().

Definition at line 406 of file libio.h.

#### 8.66.2.19 rtems_filesystem_unmount_t

```
typedef int(* rtems_filesystem_unmount_t) (rtems_filesystem_mount_table_entry_t *mt_entry)
```

Unmounts a file system instance in a mount point (directory).

In case this function is successful the file system instance will be marked as unmounted. The file system instance will be destroyed when the last reference to it vanishes.

**Parameters**

| | | |
|---|---|---|
| in | *mt_entry* | The mount table entry. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_unmount().

Definition at line 296 of file libio.h.

#### 8.66.2.20 rtems_filesystem_utime_t

```
typedef int(* rtems_filesystem_utime_t) (const rtems_filesystem_location_info_t *loc, time_↩
t actime, time_t modtime)
```

Set node access and modification times.

**Parameters**

| | | |
|---|---|---|
| in | *loc* | The location of the node. |
| in | *actime* | Access time for the node. |
| in | *modtime* | Modification for the node. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno is set to indicate the error. |

**See also**

rtems_filesystem_default_utime().

Definition at line 386 of file libio.h.

## 8.66.3 Function Documentation

### 8.66.3.1 rtems_filesystem_default_are_nodes_equal()

```
bool rtems_filesystem_default_are_nodes_equal (
            const rtems_filesystem_location_info_t * a,
            const rtems_filesystem_location_info_t * b )
```

Tests if the node access pointer of one location is equal to the node access pointer of the other location.

**Parameters**

| | | |
|---|---|---|
| in | *a* | The one location. |
| in | *b* | The other location. |

**Return values**

| | |
|---|---|
| *true* | The node access pointers of the locations are equal. |
| *false* | Otherwise. |

**See also**

rtems_filesystem_are_nodes_equal_t.

### 8.66.3.2 rtems_filesystem_default_chown()

```
int rtems_filesystem_default_chown (
            const rtems_filesystem_location_info_t * loc,
            uid_t owner,
            gid_t group )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to ENOTSUP. |

**See also**

rtems_filesystem_chown_t.

#### 8.66.3.3 rtems_filesystem_default_clonenode()

```
int rtems_filesystem_default_clonenode (
            rtems_filesystem_location_info_t * loc )
```

**Return values**

| | |
|---|---|
| *0* | Always. |

**See also**

rtems_filesystem_clonenode_t.

#### 8.66.3.4 rtems_filesystem_default_eval_path()

```
void rtems_filesystem_default_eval_path (
            rtems_filesystem_eval_path_context_t * ctx )
```

Terminates the path evaluation and replaces the current location with the null location.

**See also**

rtems_filesystem_eval_path_t.

#### 8.66.3.5 rtems_filesystem_default_fchmod()

```
int rtems_filesystem_default_fchmod (
            const rtems_filesystem_location_info_t * loc,
            mode_t mode )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to ENOTSUP. |

**See also**

rtems_filesystem_fchmod_t.

#### 8.66.3.6 rtems_filesystem_default_freenode()

```
void rtems_filesystem_default_freenode (
             const rtems_filesystem_location_info_t * loc )
```

**See also**

rtems_filesystem_freenode_t.

#### 8.66.3.7 rtems_filesystem_default_fsunmount()

```
void rtems_filesystem_default_fsunmount (
             rtems_filesystem_mount_table_entry_t * mt_entry )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to ENOTSUP. |

**See also**

rtems_filesystem_fsunmount_me_t.

#### 8.66.3.8 rtems_filesystem_default_link()

```
int rtems_filesystem_default_link (
             const rtems_filesystem_location_info_t * parentloc,
             const rtems_filesystem_location_info_t * targetloc,
             const char * name,
             size_t namelen )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to ENOTSUP. |

**See also**

rtems_filesystem_link_t.

#### 8.66.3.9 rtems_filesystem_default_lock()

```
void rtems_filesystem_default_lock (
             const rtems_filesystem_mount_table_entry_t * mt_entry )
```

Obtains the IO library mutex.

**See also**

rtems_filesystem_mt_entry_lock_t.

#### 8.66.3.10 rtems_filesystem_default_mknod()

```
int rtems_filesystem_default_mknod (
             const rtems_filesystem_location_info_t * parentloc,
             const char * name,
             size_t namelen,
             mode_t mode,
             dev_t dev )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to ENOTSUP. |

**See also**

rtems_filesystem_mknod_t.

#### 8.66.3.11 rtems_filesystem_default_mount()

```
int rtems_filesystem_default_mount (
             rtems_filesystem_mount_table_entry_t * mt_entry )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to ENOTSUP. |

**See also**

rtems_filesystem_mount_t.

#### 8.66.3.12 rtems_filesystem_default_readlink()

```
ssize_t rtems_filesystem_default_readlink (
              const rtems_filesystem_location_info_t * loc,
              char * buf,
              size_t bufsize )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to ENOTSUP. |

**See also**

rtems_filesystem_readlink_t.

#### 8.66.3.13 rtems_filesystem_default_rename()

```
int rtems_filesystem_default_rename (
              const rtems_filesystem_location_info_t * oldparentloc,
              const rtems_filesystem_location_info_t * oldloc,
              const rtems_filesystem_location_info_t * newparentloc,
              const char * name,
              size_t namelen )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to ENOTSUP. |

**See also**

rtems_filesystem_rename_t.

#### 8.66.3.14 rtems_filesystem_default_rmnod()

```
int rtems_filesystem_default_rmnod (
              const rtems_filesystem_location_info_t * parentloc,
              const rtems_filesystem_location_info_t * loc )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to ENOTSUP. |

**See also**

rtems_filesystem_rmnod_t.

#### 8.66.3.15 rtems_filesystem_default_statvfs()

```
int rtems_filesystem_default_statvfs (
            const rtems_filesystem_location_info_t * loc,
            struct statvfs * buf )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to ENOTSUP. |

**See also**

rtems_filesystem_statvfs_t.

#### 8.66.3.16 rtems_filesystem_default_symlink()

```
int rtems_filesystem_default_symlink (
            const rtems_filesystem_location_info_t * parentloc,
            const char * name,
            size_t namelen,
            const char * target )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to ENOTSUP. |

**See also**

rtems_filesystem_symlink_t.

#### 8.66.3.17 rtems_filesystem_default_unlock()

```
void rtems_filesystem_default_unlock (
            const rtems_filesystem_mount_table_entry_t * mt_entry )
```

Releases the IO library mutex.

**See also**

rtems_filesystem_mt_entry_unlock_t.

#### 8.66.3.18 rtems_filesystem_default_unmount()

```
int rtems_filesystem_default_unmount (
            rtems_filesystem_mount_table_entry_t * mt_entry )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to ENOTSUP. |

**See also**

rtems_filesystem_unmount_t.

#### 8.66.3.19 rtems_filesystem_default_utime()

```
int rtems_filesystem_default_utime (
            const rtems_filesystem_location_info_t * loc,
            time_t actime,
            time_t modtime )
```

**Return values**

| | |
|---|---|
| *-1* | Always. The errno is set to ENOTSUP. |

**See also**

rtems_filesystem_utime_t.

## 8.67 File System Types and Mount

File system types and mount.

### Modules

- libfs

  *libfs*
- nfs Client

  *nfs Client*

### Classes

- struct rtems_filesystem_mount_table_entry_tt

  *Mount table entry.*
- struct rtems_filesystem_table_t

  *File system table entry.*
- struct rtems_filesystem_mount_configuration

### Typedefs

- typedef struct rtems_filesystem_table_t rtems_filesystem_table_t

  *File system table entry.*
- typedef bool(* rtems_per_filesystem_routine) (const rtems_filesystem_table_t *fs_entry, void *arg)

  *Per file system type routine.*
- typedef bool(* rtems_filesystem_mt_entry_visitor) (const rtems_filesystem_mount_table_entry_t *mt_entry, void *arg)

  *Mount table entry visitor.*

### Enumerations

- enum rtems_filesystem_options_t { **RTEMS_FILESYSTEM_READ_ONLY**, **RTEMS_FILESYSTEM_READ_WRITE**, **RTEMS_FILESYSTEM_BAD_OPTIONS** }

  *File system options.*

### Functions

- int rtems_filesystem_register (const char *type, rtems_filesystem_fsmount_me_t mount_h)

  *Registers a file system type.*
- int rtems_filesystem_unregister (const char *type)

  *Unregisters a file system type.*
- int unmount (const char *mount_path)

  *Unmounts the file system instance at the specified mount path.*
- int mount (const char *source, const char *target, const char *filesystemtype, rtems_filesystem_options_t options, const void *data)

  *Mounts a file system instance at the specified target path.*
- int mount_and_make_target_path (const char *source, const char *target, const char *filesystemtype, rtems_filesystem_options_t options, const void *data)

  *Mounts a file system and makes the target path.*
- bool rtems_filesystem_iterate (rtems_per_filesystem_routine routine, void *routine_arg)

  *Iterates over all file system types.*
- bool rtems_filesystem_mount_iterate (rtems_filesystem_mt_entry_visitor visitor, void *visitor_arg)

  *Iterates over all file system mount entries.*

## Variables

- const rtems_filesystem_table_t rtems_filesystem_table [ ]

  *Static table of file systems.*
- rtems_chain_control **rtems_filesystem_mount_table**
- const rtems_filesystem_mount_configuration **rtems_filesystem_root_configuration**

## File System Types

- #define **RTEMS_FILESYSTEM_TYPE_IMFS** "imfs"
- #define **RTEMS_FILESYSTEM_TYPE_FTPFS** "ftpfs"
- #define **RTEMS_FILESYSTEM_TYPE_TFTPFS** "tftpfs"
- #define **RTEMS_FILESYSTEM_TYPE_NFS** "nfs"
- #define **RTEMS_FILESYSTEM_TYPE_DOSFS** "dosfs"
- #define **RTEMS_FILESYSTEM_TYPE_RFS** "rfs"
- #define **RTEMS_FILESYSTEM_TYPE_JFFS2** "jffs2"

### 8.67.1 Detailed Description

File system types and mount.

### 8.67.2 Typedef Documentation

#### 8.67.2.1 rtems_filesystem_mt_entry_visitor

```
typedef bool(* rtems_filesystem_mt_entry_visitor) (const rtems_filesystem_mount_table_entry_t
*mt_entry, void *arg)
```

Mount table entry visitor.

**Return values**

| | |
|---|---|
| *true* | Stop the iteration. |
| *false* | Continue the iteration. |

**See also**

rtems_filesystem_mount_iterate().

Definition at line 1834 of file libio.h.

#### 8.67.2.2 rtems_per_filesystem_routine

```
typedef bool(* rtems_per_filesystem_routine) (const rtems_filesystem_table_t *fs_entry, void
*arg)
```

Per file system type routine.

**See also**

rtems_filesystem_iterate().

**Return values**

| | |
|---|---|
| *true* | Stop the iteration. |
| *false* | Continue the iteration. |

Definition at line 1803 of file libio.h.

### 8.67.3 Function Documentation

#### 8.67.3.1 mount()

```
int mount (
              const char * source,
              const char * target,
              const char * filesystemtype,
              rtems_filesystem_options_t options,
              const void * data )
```

Mounts a file system instance at the specified target path.

To mount a standard file system instance one of the following defines should be used to select the file system type

- RTEMS_FILESYSTEM_TYPE_DOSFS,
- RTEMS_FILESYSTEM_TYPE_FTPFS,
- RTEMS_FILESYSTEM_TYPE_IMFS,
- RTEMS_FILESYSTEM_TYPE_JFFS2,
- RTEMS_FILESYSTEM_TYPE_NFS,
- RTEMS_FILESYSTEM_TYPE_RFS, or
- RTEMS_FILESYSTEM_TYPE_TFTPFS.

Only configured or registered file system types are available. You can add file system types to your application configuration with the following configuration options

- CONFIGURE_FILESYSTEM_DOSFS,
- CONFIGURE_FILESYSTEM_FTPFS,
- CONFIGURE_FILESYSTEM_IMFS,
- CONFIGURE_FILESYSTEM_JFFS2,
- CONFIGURE_FILESYSTEM_NFS,
- CONFIGURE_FILESYSTEM_RFS, and
- CONFIGURE_FILESYSTEM_TFTPFS.

In addition to these configuration options file system types can be registered with rtems_filesystem_register().

**Parameters**

| | | |
|---|---|---|
| in | *source* | The source parameter will be forwarded to the file system initialization handler. Usually the source is a path to the corresponding device file, or NULL in case the file system does not use a device file. |
| in | *target* | The target path must lead to an existing directory, or must be NULL. In case the target is NULL, the root file system will be mounted. |
| in | *filesystemtype* | This string selects the file system type. |
| in | *options* | The options specify if the file system instance allows read-write or read-only access. |
| in | *data* | The data parameter will be forwarded to the file system initialization handler. It can be used to pass file system specific mount options. The data structure for mount options is file system specific. See also in the corresponding file system documentation. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The errno indicates the error. |

**See also**

rtems_filesystem_register(), mount_and_make_target_path(), DOSFS and JFFS2.

### 8.67.3.2 mount_and_make_target_path()

```
int mount_and_make_target_path (
             const char * source,
             const char * target,
             const char * filesystemtype,
             rtems_filesystem_options_t options,
             const void * data )
```

Mounts a file system and makes the *target* path.

The *target* path will be created with rtems_mkdir() and must not be NULL.

**See also**

mount().

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The `errno` indicates the error. |

#### 8.67.3.3 rtems_filesystem_iterate()

```
bool rtems_filesystem_iterate (
            rtems_per_filesystem_routine routine,
            void * routine_arg )
```

Iterates over all file system types.

For each file system type the *routine* will be called with the entry and the *routine_arg* parameter.

Do not register or unregister file system types in *routine*.

The iteration is protected by the IO library mutex.

**Return values**

| | |
|---|---|
| *true* | Iteration stopped due to *routine* return status. |
| *false* | Iteration through all entries. |

#### 8.67.3.4 rtems_filesystem_mount_iterate()

```
bool rtems_filesystem_mount_iterate (
            rtems_filesystem_mt_entry_visitor visitor,
            void * visitor_arg )
```

Iterates over all file system mount entries.

The iteration is protected by the IO library mutex. Do not mount or unmount file systems in the visitor function.

**Parameters**

| | | |
|---|---|---|
| `in` | *visitor* | For each file system mount entry the visitor function will be called with the entry and the visitor argument as parameters. |
| `in` | *visitor_arg* | The second parameter for the visitor function. |

**Return values**

| | |
|---|---|
| *true* | Iteration stopped due to visitor function return status. |
| *false* | Iteration through all entries. |

#### 8.67.3.5 rtems_filesystem_register()

```
int rtems_filesystem_register (
            const char * type,
            rtems_filesystem_fsmount_me_t mount_h )
```

Registers a file system *type*.

The *mount_h* handler will be used to mount a file system of this *type*.

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The `errno` indicates the error. |

#### 8.67.3.6 rtems_filesystem_unregister()

```
int rtems_filesystem_unregister (
            const char * type )
```

Unregisters a file system *type*.

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The `errno` indicates the error. |

#### 8.67.3.7 unmount()

```
int unmount (
            const char * mount_path )
```

Unmounts the file system instance at the specified mount path.

The function waits for the unmount process completion. In case the calling thread uses resources of the unmounted file system the function may never return. In case the calling thread has its root or current directory in the unmounted file system the function returns with an error status and errno is set to EBUSY.

The unmount process completion notification uses the transient event. It is a fatal error to terminate the calling thread while waiting for this event.

A concurrent unmount request for the same file system instance has unpredictable effects.

**Parameters**

| | | |
|---|---|---|
| in | *mount_path* | The path to the file system instance mount point. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The `errno` indicates the error. |

**See also**

ClassicEventTransient.

## 8.67.4 Variable Documentation

### 8.67.4.1 rtems_filesystem_table

```
const rtems_filesystem_table_t rtems_filesystem_table[]  [extern]
```

Static table of file systems.

Externally defined by confdefs.h or the user.

# 8.68 Filesystem Configuration

## Macros

- #define CONFIGURE_APPLICATION_DISABLE_FILESYSTEM

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_FILESYSTEM_ALL

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_FILESYSTEM_DOSFS

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_FILESYSTEM_FTPFS

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_FILESYSTEM_IMFS

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_FILESYSTEM_JFFS2

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_FILESYSTEM_NFS

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_FILESYSTEM_RFS

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_FILESYSTEM_TFTPFS

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_IMFS_DISABLE_CHMOD

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_IMFS_DISABLE_CHOWN

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_IMFS_DISABLE_LINK

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_IMFS_DISABLE_MKNOD

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_IMFS_DISABLE_MKNOD_DEVICE

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_IMFS_DISABLE_MKNOD_FILE

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_IMFS_DISABLE_MOUNT

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_IMFS_DISABLE_READDIR

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_IMFS_DISABLE_READLINK

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_IMFS_DISABLE_RENAME

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_IMFS_DISABLE_RMNOD

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_IMFS_DISABLE_SYMLINK

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_IMFS_DISABLE_UNMOUNT

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_IMFS_DISABLE_UTIME

  *This configuration option is a boolean feature define.*

- #define CONFIGURE_IMFS_ENABLE_MKFIFO

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_IMFS_MEMFILE_BYTES_PER_BLOCK

  *This configuration option is an integer define.*
- #define CONFIGURE_USE_DEVFS_AS_BASE_FILESYSTEM

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_USE_MINIIMFS_AS_BASE_FILESYSTEM

  *This configuration option is a boolean feature define.*

### 8.68.1 Detailed Description

This section describes configuration options related to filesytems. By default, the In-Memory Filesystem (IMFS) is used as the base filesystem (also known as root filesystem). In order to save some memory for your application, you can disable the filesystem support with the CONFIGURE_APPLICATION_DISABLE_FILESYSTEM configuration option. Alternatively, you can strip down the features of the base filesystem with the CONFIGURE_USE_MINIIMFS_AS_BASE_FILESYSTEM and CONFIGURE_USE_DEVFS_AS_BASE_FILESYSTEM configuration options. These three configuration options are mutually exclusive. They are intended for an advanced application configuration.

Features of the IMFS can be disabled and enabled with the following configuration options:

- CONFIGURE_IMFS_DISABLE_CHMOD
- CONFIGURE_IMFS_DISABLE_CHOWN
- CONFIGURE_IMFS_DISABLE_LINK
- CONFIGURE_IMFS_DISABLE_MKNOD
- CONFIGURE_IMFS_DISABLE_MKNOD_FILE
- CONFIGURE_IMFS_DISABLE_MOUNT
- CONFIGURE_IMFS_DISABLE_READDIR
- CONFIGURE_IMFS_DISABLE_READLINK
- CONFIGURE_IMFS_DISABLE_RENAME
- CONFIGURE_IMFS_DISABLE_RMNOD
- CONFIGURE_IMFS_DISABLE_SYMLINK
- CONFIGURE_IMFS_DISABLE_UNMOUNT
- CONFIGURE_IMFS_DISABLE_UTIME
- CONFIGURE_IMFS_ENABLE_MKFIFO

### 8.68.2 Macro Definition Documentation

#### 8.68.2.1 CONFIGURE_APPLICATION_DISABLE_FILESYSTEM

```
#define CONFIGURE_APPLICATION_DISABLE_FILESYSTEM
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then **no base filesystem** is initialized during system initialization and **no filesystems** are configured.

##### Default Configuration

If this configuration option is undefined, then a base filesystem and the configured filesystems are initialized during system initialization.

##### Notes

Filesystems shall be initialized to support file descriptor based device drivers and basic input/output functions such as printf(). Filesystems can be disabled to reduce the memory footprint of an application.

Definition at line 1916 of file appl-config.h.

#### 8.68.2.2 CONFIGURE_FILESYSTEM_ALL

```
#define CONFIGURE_FILESYSTEM_ALL
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the following configuration options will be defined as well

- CONFIGURE_FILESYSTEM_DOSFS,
- CONFIGURE_FILESYSTEM_FTPFS,
- CONFIGURE_FILESYSTEM_IMFS,
- CONFIGURE_FILESYSTEM_JFFS2,
- CONFIGURE_FILESYSTEM_NFS,
- CONFIGURE_FILESYSTEM_RFS, and
- CONFIGURE_FILESYSTEM_TFTPFS.

##### Default Configuration

If this configuration option is undefined, then the described feature is not enabled.

Definition at line 1944 of file appl-config.h.

#### 8.68.2.3 CONFIGURE_FILESYSTEM_DOSFS

```
#define CONFIGURE_FILESYSTEM_DOSFS
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the DOS (FAT) filesystem is registered, so that instances of this filesystem can be mounted by the application.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

This filesystem requires a Block Device Cache configuration, see CONFIGURE_APPLICATION_NEEDS_LIBBLOCK.

Definition at line 1963 of file appl-config.h.

#### 8.68.2.4 CONFIGURE_FILESYSTEM_FTPFS

```
#define CONFIGURE_FILESYSTEM_FTPFS
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the FTP filesystem (FTP client) is registered, so that instances of this filesystem can be mounted by the application.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

Definition at line 1978 of file appl-config.h.

#### 8.68.2.5 CONFIGURE_FILESYSTEM_IMFS

```
#define CONFIGURE_FILESYSTEM_IMFS
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the In-Memory Filesystem (IMFS) is registered, so that instances of this filesystem can be mounted by the application.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

Applications will rarely need this configuration option. This configuration option is intended for test programs. You do not need to define this configuration option for the base filesystem (also known as root filesystem).

Definition at line 1999 of file appl-config.h.

#### 8.68.2.6 CONFIGURE_FILESYSTEM_JFFS2

```
#define CONFIGURE_FILESYSTEM_JFFS2
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the JFFS2 filesystem is registered, so that instances of this filesystem can be mounted by the application.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

Definition at line 2014 of file appl-config.h.

#### 8.68.2.7 CONFIGURE_FILESYSTEM_NFS

```
#define CONFIGURE_FILESYSTEM_NFS
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the Network Filesystem (NFS) client is registered, so that instances of this filesystem can be mounted by the application.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

Definition at line 2029 of file appl-config.h.

#### 8.68.2.8 CONFIGURE_FILESYSTEM_RFS

```
#define CONFIGURE_FILESYSTEM_RFS
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the RTEMS Filesystem (RFS) is registered, so that instances of this filesystem can be mounted by the application.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

This filesystem requires a Block Device Cache configuration, see CONFIGURE_APPLICATION_NEEDS_LIBBLOCK.

Definition at line 2048 of file appl-config.h.

#### 8.68.2.9 CONFIGURE_FILESYSTEM_TFTPFS

```
#define CONFIGURE_FILESYSTEM_TFTPFS
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the TFTP filesystem (TFTP client) is registered, so that instances of this filesystem can be mounted by the application.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

Definition at line 2063 of file appl-config.h.

#### 8.68.2.10 CONFIGURE_IMFS_DISABLE_CHMOD

```
#define CONFIGURE_IMFS_DISABLE_CHMOD
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the root IMFS does not support changing the mode of files (no support for chmod()).

**Default Configuration**

If this configuration option is undefined, then the root IMFS supports changing the mode of files.

Definition at line 2077 of file appl-config.h.

#### 8.68.2.11 CONFIGURE_IMFS_DISABLE_CHOWN

```
#define CONFIGURE_IMFS_DISABLE_CHOWN
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the root IMFS does not support changing the ownership of files (no support for chown()).

**Default Configuration**

If this configuration option is undefined, then the root IMFS supports changing the ownership of files.

Definition at line 2091 of file appl-config.h.

#### 8.68.2.12 CONFIGURE_IMFS_DISABLE_LINK

```
#define CONFIGURE_IMFS_DISABLE_LINK
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the root IMFS does not support hard links (no support for link()).

**Default Configuration**

If this configuration option is undefined, then the root IMFS supports hard links.

Definition at line 2105 of file appl-config.h.

#### 8.68.2.13 CONFIGURE_IMFS_DISABLE_MKNOD

```
#define CONFIGURE_IMFS_DISABLE_MKNOD
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the root IMFS does not support making files (no support for mknod()).

**Default Configuration**

If this configuration option is undefined, then the root IMFS supports making files.

Definition at line 2119 of file appl-config.h.

#### 8.68.2.14 CONFIGURE_IMFS_DISABLE_MKNOD_DEVICE

```
#define CONFIGURE_IMFS_DISABLE_MKNOD_DEVICE
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the root IMFS does not support making device files.

**Default Configuration**

If this configuration option is undefined, then the root IMFS supports making device files.

Definition at line 2133 of file appl-config.h.

#### 8.68.2.15 CONFIGURE_IMFS_DISABLE_MKNOD_FILE

```
#define CONFIGURE_IMFS_DISABLE_MKNOD_FILE
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the root IMFS does not support making regular files.

**Default Configuration**

If this configuration option is undefined, then the root IMFS supports making regular files.

Definition at line 2147 of file appl-config.h.

#### 8.68.2.16 CONFIGURE_IMFS_DISABLE_MOUNT

```
#define CONFIGURE_IMFS_DISABLE_MOUNT
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the root IMFS does not support mounting other filesystems (no support for mount()).

**Default Configuration**

If this configuration option is undefined, then the root IMFS supports mounting other filesystems.

Definition at line 2161 of file appl-config.h.

#### 8.68.2.17 CONFIGURE_IMFS_DISABLE_READDIR

```
#define CONFIGURE_IMFS_DISABLE_READDIR
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the root IMFS does not support reading directories (no support for readdir()). It is still possible to open files in a directory.

**Default Configuration**

If this configuration option is undefined, then the root IMFS supports reading directories.

Definition at line 2176 of file appl-config.h.

#### 8.68.2.18 CONFIGURE_IMFS_DISABLE_READLINK

```
#define CONFIGURE_IMFS_DISABLE_READLINK
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the root IMFS does not support reading symbolic links (no support for readlink()).

**Default Configuration**

If this configuration option is undefined, then the root IMFS supports reading symbolic links.

Definition at line 2190 of file appl-config.h.

#### 8.68.2.19 CONFIGURE_IMFS_DISABLE_RENAME

```
#define CONFIGURE_IMFS_DISABLE_RENAME
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the root IMFS does not support renaming files (no support for rename()).

**Default Configuration**

If this configuration option is undefined, then the root IMFS supports renaming files.

Definition at line 2204 of file appl-config.h.

#### 8.68.2.20 CONFIGURE_IMFS_DISABLE_RMNOD

```
#define CONFIGURE_IMFS_DISABLE_RMNOD
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the root IMFS does not support removing files (no support for rmnod()).

**Default Configuration**

If this configuration option is undefined, then the root IMFS supports removing files.

Definition at line 2218 of file appl-config.h.

#### 8.68.2.21 CONFIGURE_IMFS_DISABLE_SYMLINK

```
#define CONFIGURE_IMFS_DISABLE_SYMLINK
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the root IMFS does not support creating symbolic links (no support for symlink()).

**Default Configuration**

If this configuration option is undefined, then the root IMFS supports creating symbolic links.

Definition at line 2232 of file appl-config.h.

#### 8.68.2.22 CONFIGURE_IMFS_DISABLE_UNMOUNT

```
#define CONFIGURE_IMFS_DISABLE_UNMOUNT
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the root IMFS does not support unmounting other filesystems (no support for unmount()).

**Default Configuration**

If this configuration option is undefined, then the root IMFS supports unmounting other filesystems.

Definition at line 2246 of file appl-config.h.

#### 8.68.2.23 CONFIGURE_IMFS_DISABLE_UTIME

```
#define CONFIGURE_IMFS_DISABLE_UTIME
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the root IMFS does not support changing file times (no support for utime()).

**Default Configuration**

If this configuration option is undefined, then the root IMFS supports changing file times.

Definition at line 2260 of file appl-config.h.

#### 8.68.2.24 CONFIGURE_IMFS_ENABLE_MKFIFO

```
#define CONFIGURE_IMFS_ENABLE_MKFIFO
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the root IMFS supports making FIFOs.

**Default Configuration**

If this configuration option is undefined, then the root IMFS does not support making FIFOs (no support for mkfifo()).

Definition at line 2274 of file appl-config.h.

#### 8.68.2.25 CONFIGURE_IMFS_MEMFILE_BYTES_PER_BLOCK

```
#define CONFIGURE_IMFS_MEMFILE_BYTES_PER_BLOCK
```

This configuration option is an integer define.

The value of this configuration option defines the block size for in-memory files managed by the IMFS.

**Default Value**

The default value is 128.

**Value Constraints**

The value of this configuration option shall be an element of {16, 32, 64, 128, 256, 512}.

**Notes**

The configured block size has two impacts. The first is the average amount of unused memory in the last block of each file. For example, when the block size is 512, on average one-half of the last block of each file will remain unused and the memory is wasted. In contrast, when the block size is 16, the average unused memory per file is only 8 bytes. However, it requires more allocations for the same size file and thus more overhead per block for the dynamic memory management.

Second, the block size has an impact on the maximum size file that can be stored in the IMFS. With smaller block size, the maximum file size is correspondingly smaller. The following shows the maximum file size possible based on the configured block size:

- when the block size is 16 bytes, the maximum file size is 1,328 bytes.
- when the block size is 32 bytes, the maximum file size is 18,656 bytes.
- when the block size is 64 bytes, the maximum file size is 279,488 bytes.
- when the block size is 128 bytes, the maximum file size is 4,329,344 bytes.
- when the block size is 256 bytes, the maximum file size is 68,173,568 bytes.
- when the block size is 512 bytes, the maximum file size is 1,082,195,456 bytes.

Definition at line 2322 of file appl-config.h.

#### 8.68.2.26 CONFIGURE_USE_DEVFS_AS_BASE_FILESYSTEM

```
#define CONFIGURE_USE_DEVFS_AS_BASE_FILESYSTEM
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then an IMFS with a reduced feature set will be the base filesystem (also known as root filesystem).

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

In case this configuration option is defined, then the following configuration options will be defined as well

- CONFIGURE_IMFS_DISABLE_CHMOD,
- CONFIGURE_IMFS_DISABLE_CHOWN,
- CONFIGURE_IMFS_DISABLE_LINK,
- CONFIGURE_IMFS_DISABLE_MKNOD_FILE,
- CONFIGURE_IMFS_DISABLE_MOUNT,
- CONFIGURE_IMFS_DISABLE_READDIR,
- CONFIGURE_IMFS_DISABLE_READLINK,
- CONFIGURE_IMFS_DISABLE_RENAME,
- CONFIGURE_IMFS_DISABLE_RMNOD,
- CONFIGURE_IMFS_DISABLE_SYMLINK,
- CONFIGURE_IMFS_DISABLE_UTIME, and
- CONFIGURE_IMFS_DISABLE_UNMOUNT.

In addition, a simplified path evaluation is enabled. It allows only a look up of absolute paths.

This configuration of the IMFS is basically a device-only filesystem. It is comparable in functionality to the pseudo-filesystem name space provided before RTEMS release 4.5.0.

Definition at line 2373 of file appl-config.h.

#### 8.68.2.27 CONFIGURE_USE_MINIIMFS_AS_BASE_FILESYSTEM

```
#define CONFIGURE_USE_MINIIMFS_AS_BASE_FILESYSTEM
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then an IMFS with a reduced feature set will be the base filesystem (also known as root filesystem).

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

In case this configuration option is defined, then the following configuration options will be defined as well

- CONFIGURE_IMFS_DISABLE_CHMOD,
- CONFIGURE_IMFS_DISABLE_CHOWN,
- CONFIGURE_IMFS_DISABLE_LINK,
- CONFIGURE_IMFS_DISABLE_READLINK,
- CONFIGURE_IMFS_DISABLE_RENAME,
- CONFIGURE_IMFS_DISABLE_SYMLINK,
- CONFIGURE_IMFS_DISABLE_UTIME, and
- CONFIGURE_IMFS_DISABLE_UNMOUNT.

Definition at line 2409 of file appl-config.h.

## 8.69 Flexible Per-CPU Data

Flexible Per-CPU Data.

### Files

- file percpudata.h

  *Definition of custom per-CPU items.*

### Macros

- #define PER_CPU_DATA_ITEM_DECLARE(type, item) RTEMS_LINKER_RWSET_ITEM_DECLARE( _↩ Per_CPU_Data, type, item )

  *Declares a per-CPU item of the specified type.*
- #define PER_CPU_DATA_ITEM(type, item) RTEMS_LINKER_RWSET_ITEM( _Per_CPU_Data, type, item )

  *Defines a per-CPU item of the specified type.*
- #define PER_CPU_DATA_OFFSET(item)

  *Returns the offset of the per-CPU item to the begin of the per-CPU data area.*
- #define PER_CPU_DATA_GET_BY_OFFSET(cpu, type, offset) (type *) ( cpu->data + offset )

  *Returns a pointer of the specified type to the per-CPU item at the specified offset for the specified processor.*
- #define PER_CPU_DATA_GET(cpu, type, item) PER_CPU_DATA_GET_BY_OFFSET( cpu, type, PER_CPU_DATA_OFFSET( item ) )

  *Returns a pointer of the specified type to the specified per-CPU item for the specified processor.*

### Functions

- **RTEMS_LINKER_RWSET_DECLARE** (_Per_CPU_Data, char)

### 8.69.1 Detailed Description

Flexible Per-CPU Data.

Provides the definition of custom per-CPU items. The items are collected in a special linker set. During system initialization the content of the linker set is duplicated for all secondary processors using memory allocated from the workspace. The begin and end of the per-CPU data area is cache line aligned (CPU_CACHE_LINE_BYTES).

### 8.69.2 Macro Definition Documentation

#### 8.69.2.1 PER_CPU_DATA_GET

```
#define PER_CPU_DATA_GET(
              cpu,
              type,
              item )  PER_CPU_DATA_GET_BY_OFFSET( cpu, type, PER_CPU_DATA_OFFSET( item ) )
```

Returns a pointer of the specified type to the specified per-CPU item for the specified processor.

**Parameters**

| | |
|---|---|
| *cpu* | The processor of the item. |
| *type* | The type of the item. |
| *item* | The designator of the item. |

Definition at line 107 of file percpudata.h.

#### 8.69.2.2 PER_CPU_DATA_GET_BY_OFFSET

```
#define PER_CPU_DATA_GET_BY_OFFSET(
              cpu,
              type,
              offset )  (type *) ( cpu->data + offset )
```

Returns a pointer of the specified type to the per-CPU item at the specified offset for the specified processor.

**Parameters**

| | |
|---|---|
| *cpu* | The processor of the item. |
| *type* | The type of the item. |
| *offset* | The offset of the item. |

Definition at line 90 of file percpudata.h.

#### 8.69.2.3 PER_CPU_DATA_ITEM

```
#define PER_CPU_DATA_ITEM(
              type,
              item )  RTEMS_LINKER_RWSET_ITEM( _Per_CPU_Data, type, item )
```

Defines a per-CPU item of the specified type.

**Parameters**

| | |
|---|---|
| *type* | The type of the item. |
| *item* | The designator of the item. |

Definition at line 68 of file percpudata.h.

#### 8.69.2.4 PER_CPU_DATA_ITEM_DECLARE

```
#define PER_CPU_DATA_ITEM_DECLARE(
```

```
              type,
              item )  RTEMS_LINKER_RWSET_ITEM_DECLARE( _Per_CPU_Data, type, item )
```

Declares a per-CPU item of the specified type.

Items declared with this macro have external linkage.

**Parameters**

| | |
|---|---|
| *type* | The type of the item. |
| *item* | The designator of the item. |

Definition at line 59 of file percpudata.h.

#### 8.69.2.5 PER_CPU_DATA_OFFSET

```
#define PER_CPU_DATA_OFFSET(
              item )
```

**Value:**

```
  ( (uintptr_t) &_Linker_set__Per_CPU_Data_##item \
    - (uintptr_t) RTEMS_LINKER_SET_BEGIN( _Per_CPU_Data ) )
```

Returns the offset of the per-CPU item to the begin of the per-CPU data area.

**Parameters**

| | |
|---|---|
| *item* | The designator of the item. |

Definition at line 77 of file percpudata.h.

# 8.70 Free-Running Counter and Busy Wait Delay

Free-running counter and busy wait delay functions.

## Files

- file counter.h

  *Free-Running Counter and Busy Wait Delay API.*

## Typedefs

- typedef CPU_Counter_ticks rtems_counter_ticks

  *Unsigned integer type for counter values.*

## Functions

- static uint32_t rtems_counter_frequency (void)

  *Returns the current counter frequency in Hz.*
- static rtems_counter_ticks rtems_counter_read (void)

  *Reads the current counter value.*
- static rtems_counter_ticks rtems_counter_difference (rtems_counter_ticks second, rtems_counter_ticks first)

  *Returns the difference between the second and first CPU counter value.*
- uint64_t rtems_counter_ticks_to_nanoseconds (rtems_counter_ticks ticks)

  *Converts counter ticks into nanoseconds.*
- rtems_counter_ticks rtems_counter_nanoseconds_to_ticks (uint32_t nanoseconds)

  *Converts nanoseconds into counter ticks.*
- int64_t rtems_counter_ticks_to_sbintime (rtems_counter_ticks ticks)

  *Converts counter ticks into signed binary time (sbintime_t).*
- rtems_counter_ticks rtems_counter_sbintime_to_ticks (int64_t sbt)

  *Converts signed binary time (sbintime_t) into counter ticks.*
- void rtems_counter_initialize_converter (uint32_t frequency)

  *Initializes the counter ticks to/from nanoseconds converter functions.*
- void rtems_counter_delay_ticks (rtems_counter_ticks ticks)

  *Busy wait for some counter ticks.*
- void rtems_counter_delay_nanoseconds (uint32_t nanoseconds)

  *Busy wait for some nanoseconds.*

### 8.70.1 Detailed Description

Free-running counter and busy wait delay functions.

The RTEMS counter is some free-running counter. It ticks usually with a frequency close to the CPU or system bus clock.

The counter can be used in case the overhead of the rtems_clock_get_uptime_nanoseconds() is too high. The step from counter ticks to/from nanoseconds is explicit in this API unlike to rtems_clock_get_uptime_nanoseconds() which performs the conversion on each invocation.

This counter works without a clock driver and during system initialization.

The counter can be used to profile low-level operations like SMP locks or interrupt disabled critical sections. The counter can act also as an entropy source for a random number generator.

The period of the counter depends on the actual hardware.

## 8.70.2 Function Documentation

### 8.70.2.1 rtems_counter_delay_nanoseconds()

```
void rtems_counter_delay_nanoseconds (
            uint32_t nanoseconds )
```

Busy wait for some nanoseconds.

This function does not disable interrupts. Thus task switches and interrupts can interfere with this busy wait may prolong the delay. This function busy waits at least the specified time. Due to some overhead the actual delay may be longer.

**Parameters**

| | | |
|---|---|---|
| in | *nanoseconds* | The minimum busy wait time in nanoseconds. |

### 8.70.2.2 rtems_counter_delay_ticks()

```
void rtems_counter_delay_ticks (
            rtems_counter_ticks ticks )
```

Busy wait for some counter ticks.

This function does not disable interrupts. Thus task switches and interrupts can interfere with this busy wait may prolong the delay. This function busy waits at least the specified time. Due to some overhead the actual delay may be longer.

**Parameters**

| | | |
|---|---|---|
| in | *ticks* | The minimum busy wait time in counter ticks. |

### 8.70.2.3 rtems_counter_difference()

```
static rtems_counter_ticks rtems_counter_difference (
            rtems_counter_ticks second,
            rtems_counter_ticks first )  [inline], [static]
```

Returns the difference between the second and first CPU counter value.

This operation may be carried out as a modulo operation depending on the range of the CPU counter device.

**Parameters**

| | | |
|---|---|---|
| in | *second* | The second CPU counter value. |
| in | *first* | The first CPU counter value. |

**Returns**

Returns second minus first modulo counter period.

Definition at line 96 of file counter.h.

#### 8.70.2.4 rtems_counter_frequency()

```
static uint32_t rtems_counter_frequency (
            void  )  [inline], [static]
```

Returns the current counter frequency in Hz.

**Returns**

The current counter frequency in Hz.

Definition at line 69 of file counter.h.

#### 8.70.2.5 rtems_counter_initialize_converter()

```
void rtems_counter_initialize_converter (
            uint32_t frequency )
```

Initializes the counter ticks to/from nanoseconds converter functions.

This function must be used to initialize the rtems_counter_ticks_to_nanoseconds() and rtems_counter_nanoseconds_to_ticks() functions. It should be called during system initialization by the board support package.

**Parameters**

| | | |
|---|---|---|
| in | *frequency* | The current counter frequency in Hz. |

#### 8.70.2.6 rtems_counter_nanoseconds_to_ticks()

```
rtems_counter_ticks rtems_counter_nanoseconds_to_ticks (
            uint32_t nanoseconds )
```

Converts nanoseconds into counter ticks.

**Parameters**

| | | |
|---|---|---|
| in | *nanoseconds* | The nanoseconds value to convert. |

**Returns**

The counter ticks value corresponding to the nanoseconds value. The value is rounded up.

#### 8.70.2.7 rtems_counter_read()

```
static rtems_counter_ticks rtems_counter_read (
            void  )  [inline], [static]
```

Reads the current counter value.

**Returns**

The current counter value.

Definition at line 79 of file counter.h.

#### 8.70.2.8 rtems_counter_sbintime_to_ticks()

```
rtems_counter_ticks rtems_counter_sbintime_to_ticks (
            int64_t sbt )
```

Converts signed binary time (sbintime_t) into counter ticks.

**Parameters**

| | | |
|---|---|---|
| in | *sbt* | The signed binary time value to convert. |

**Returns**

The counter ticks value corresponding to the nanoseconds value. The value is rounded up.

#### 8.70.2.9 rtems_counter_ticks_to_nanoseconds()

```
uint64_t rtems_counter_ticks_to_nanoseconds (
            rtems_counter_ticks ticks )
```

Converts counter ticks into nanoseconds.

**Parameters**

| | | |
|---|---|---|
| in | *ticks* | The counter ticks value to convert. |

**Returns**

The nanoseconds value corresponding to the counter ticks. The value is rounded up.

#### 8.70.2.10 rtems_counter_ticks_to_sbintime()

```
int64_t rtems_counter_ticks_to_sbintime (
            rtems_counter_ticks ticks )
```

Converts counter ticks into signed binary time (sbintime_t).

**Parameters**

| | | |
|---|---|---|
| in | *ticks* | The counter ticks value to convert. |

**Returns**

The signed binary time value corresponding to the counter ticks value. The value is rounded up.

# 8.71 Freechain Handler

The Freechain Handler.

## Files

- file freechain.h

  *Freechain Handler API.*
- file freechainimpl.h

  *Freechain Handler API.*

## Classes

- struct Freechain_Control

  *The freechain control.*

## Typedefs

- typedef void *(* Freechain_Allocator) (size_t size)

  *Allocator function.*

## Functions

- static __inline__ void _Freechain_Initialize (Freechain_Control *freechain, void *initial_nodes, size_t number_nodes, size_t node_size)

  *Initializes a freechain.*
- static __inline__ bool _Freechain_Is_empty (const Freechain_Control *freechain)

  *Return true if the freechain is empty, otherwise false.*
- static __inline__ void * _Freechain_Pop (Freechain_Control *freechain)

  *Pops a node from the freechain.*
- static void __inline__ _Freechain_Push (Freechain_Control *freechain, void *node)

  *Pushes a node back to the freechain.*
- void * _Freechain_Extend (Freechain_Control *freechain, Freechain_Allocator allocator, size_t number_nodes_to_extend, size_t node_size)

  *Extend the freechain with new nodes.*
- void * _Freechain_Get (Freechain_Control *freechain, Freechain_Allocator allocator, size_t number_nodes_to_extend, size_t node_size)

  *Gets a node from the freechain.*
- void _Freechain_Put (Freechain_Control *freechain, void *node)

  *Puts a node back onto the freechain.*

### 8.71.1 Detailed Description

The Freechain Handler.

The Freechain Handler is used to manage a chain of nodes, of which size can automatically increase when there is no free node left. This handler provides one data structure: Freechain_Control.

## 8.71.2 Function Documentation

### 8.71.2.1 _Freechain_Extend()

```
void* _Freechain_Extend (
            Freechain_Control * freechain,
            Freechain_Allocator allocator,
            size_t number_nodes_to_extend,
            size_t node_size )
```

Extend the freechain with new nodes.

**Parameters**

| | |
|---|---|
| *freechain* | The freechain control. |
| *allocator* | The allocator function. |
| *number_nodes_to_extend* | The number of nodes to extend. |
| *node_size* | The node size. |

**Return values**

| | |
|---|---|
| *NULL* | The extend operation failed. |
| *nodes* | Pointer to the new nodes. |

### 8.71.2.2 _Freechain_Get()

```
void* _Freechain_Get (
            Freechain_Control * freechain,
            Freechain_Allocator allocator,
            size_t number_nodes_to_extend,
            size_t node_size )
```

Gets a node from the freechain.

**Parameters**

| | | |
|---|---|---|
| in,out | *freechain* | The freechain control. |
| | *allocator* | The allocator function. |
| | *number_nodes_to_extend* | The number of nodes in the case an extend is necessary due to an empty freechain. |
| in | *node_size* | The node size. |

**Return values**

| | |
|---|---|
| *NULL* | The freechain is empty and the extend operation failed. |
| *pointer* | Pointer to a node. The node ownership passes to the caller. |

#### 8.71.2.3 _Freechain_Initialize()

```
static __inline__ void _Freechain_Initialize (
            Freechain_Control * freechain,
            void * initial_nodes,
            size_t number_nodes,
            size_t node_size )  [static]
```

Initializes a freechain.

This routine initializes the freechain control structure to manage a chain of nodes. In the case the freechain is empty the extend handler is called to get more nodes.

**Parameters**

| | | |
|---|---|---|
| out | *freechain* | The freechain control to initialize. |
| out | *initial_nodes* | Array with the initial nodes. |
| | *number_nodes* | The initial number of nodes. |
| | *node_size* | The node size. |

Definition at line 50 of file freechainimpl.h.

#### 8.71.2.4 _Freechain_Is_empty()

```
static __inline__ bool _Freechain_Is_empty (
            const Freechain_Control * freechain )  [static]
```

Return true if the freechain is empty, otherwise false.

**Parameters**

| | |
|---|---|
| *freechain* | The freechain control. |

Definition at line 70 of file freechainimpl.h.

#### 8.71.2.5 _Freechain_Pop()

```
static __inline__ void* _Freechain_Pop (
            Freechain_Control * freechain )  [static]
```

Pops a node from the freechain.

The freechain shall not be empty.

**Parameters**

| | |
|---|---|
| *freechain* | The freechain control. |

Definition at line 84 of file freechainimpl.h.

#### 8.71.2.6 _Freechain_Push()

```
static void __inline__ _Freechain_Push (
            Freechain_Control * freechain,
            void * node )  [static]
```

Pushes a node back to the freechain.

**Parameters**

| | |
|---|---|
| *freechain* | The freechain control. |
| *node* | The node to push back. The node shall not be NULL. |

Definition at line 95 of file freechainimpl.h.

#### 8.71.2.7 _Freechain_Put()

```
void _Freechain_Put (
            Freechain_Control * freechain,
            void * node )
```

Puts a node back onto the freechain.

**Parameters**

| | | |
|---|---|---|
| in,out | *freechain* | The freechain control. |
| out | *node* | The node to put back. The node may be NULL, in this case the function does nothing. |

# 8.72 GRLIB

Driver support for GRLIB IP Library.

## Modules

- AMBA

  *AMBA Plug & Play routines.*
- LEON3 AMBA Driver Handler

  *AMBA Plag & Play Bus Driver Macros.*
- UART

  *Driver interface for APBUART.*

### 8.72.1 Detailed Description

Driver support for GRLIB IP Library.

## 8.73 General Scheduler Configuration

### Macros

- #define CONFIGURE_CBS_MAXIMUM_SERVERS

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_PRIORITY

  *This configuration option is an integer define.*
- #define CONFIGURE_SCHEDULER_ASSIGNMENTS

  *This configuration option is an initializer define.*
- #define CONFIGURE_SCHEDULER_CBS

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_SCHEDULER_EDF

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_SCHEDULER_EDF_SMP

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_SCHEDULER_NAME

  *This configuration option is an integer define.*
- #define CONFIGURE_SCHEDULER_PRIORITY

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_SCHEDULER_PRIORITY_AFFINITY_SMP

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_SCHEDULER_PRIORITY_SMP

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_SCHEDULER_SIMPLE

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_SCHEDULER_SIMPLE_SMP

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_SCHEDULER_STRONG_APA

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_SCHEDULER_USER

  *This configuration option is a boolean feature define.*

### 8.73.1 Detailed Description

This section describes configuration options related to selecting a scheduling algorithm for an application. A scheduler configuration is optional and only necessary in very specific circumstances. A normal application configuration does not need any of the configuration options described in this section.

By default, the `Deterministic Priority Scheduler` algorithm is used in uniprocessor configurations. In case SMP is enabled and the configured maximum processors (CONFIGURE_MAXIMUM_PROCESSORS) is greater than one, then the `Earliest Deadline First SMP Scheduler` is selected as the default scheduler algorithm.

For the schedulers provided by RTEMS (see `Scheduling Concepts`), the configuration is straightforward. All that is required is to define the configuration option which specifies which scheduler you want for in your application.

The pluggable scheduler interface also enables the user to provide their own scheduling algorithm. If you choose to do this, you must define multiple configuration option.

## 8.73.2 Macro Definition Documentation

### 8.73.2.1 CONFIGURE_CBS_MAXIMUM_SERVERS

```
#define CONFIGURE_CBS_MAXIMUM_SERVERS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number Constant Bandwidth Servers that can be concurrently active.

#### Default Value

The default value is CONFIGURE_MAXIMUM_TASKS.

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to SIZE_MAX.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.

#### Notes

This configuration option is only evaluated if the configuration option CONFIGURE_SCHEDULER_CBS is defined.

Definition at line 3979 of file appl-config.h.

### 8.73.2.2 CONFIGURE_MAXIMUM_PRIORITY

```
#define CONFIGURE_MAXIMUM_PRIORITY
```

This configuration option is an integer define.

For the following schedulers

- Deterministic Priority Scheduler, which is the default in uniprocessor configurations and can be configured through the CONFIGURE_SCHEDULER_PRIORITY configuration option,
- Deterministic Priority SMP Scheduler which can be configured through the CONFIGURE_SCHEDULER_PRI configuration option, and
- Arbitrary Processor Affinity Priority SMP Scheduler which can be configured through the CONFIGURE_SCHEDULER_PRIORITY_AFFINITY_SMP configuration option

this configuration option specifies the maximum numeric priority of any task for these schedulers and one less that the number of priority levels for these schedulers. For all other schedulers provided by RTEMS, this configuration option has no effect.

#### Default Value

The default value is 255.

#### Value Constraints

The value of this configuration option shall be an element of {3, 7, 31, 63, 127, 255}.

#### Notes

The numerically greatest priority is the logically lowest priority in the system and will thus be used by the IDLE task.

Priority zero is reserved for internal use by RTEMS and is not available to applications.

Reducing the number of priorities through this configuration option reduces the amount of memory allocated by the schedulers listed above. These schedulers use a chain control structure per priority and this structure consists of three pointers. On a 32-bit architecture, the allocated memory is 12 bytes * (CONFIGURE_↩ MAXIMUM_PRIORITY + 1), e.g. 3072 bytes for 256 priority levels (default), 48 bytes for 4 priority levels (CONFIGURE_MAXIMUM_PRIORITY == 3).

The default value is 255, because RTEMS shall support 256 priority levels to be compliant with various standards. These priorities range from 0 to 255.

Definition at line 4037 of file appl-config.h.

### 8.73.2.3 CONFIGURE_SCHEDULER_ASSIGNMENTS

```
#define CONFIGURE_SCHEDULER_ASSIGNMENTS
```

This configuration option is an initializer define.

The value of this configuration option is used to initialize the initial scheduler to processor assignments.

#### Default Value

The default value of this configuration option is computed so that the default scheduler is assigned to each configured processor (up to 32).

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be a list of the following macros:
  - RTEMS_SCHEDULER_ASSIGN( processor_index, attributes )
  - RTEMS_SCHEDULER_ASSIGN_NO_SCHEDULER
- It shall be a list of exactly CONFIGURE_MAXIMUM_PROCESSORS elements.

#### Notes

This configuration option is only evaluated in SMP configurations.

This is an advanced configuration option, see Clustered Scheduler Configuration.

Definition at line 4074 of file appl-config.h.

#### 8.73.2.4 CONFIGURE_SCHEDULER_CBS

```
#define CONFIGURE_SCHEDULER_CBS
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then Constant Bandwidth Server Scheduling (CBS) algorithm is made available to the application.

##### Default Configuration

If this configuration option is undefined, then the described feature is not enabled.

##### Notes

This scheduler configuration option is an advanced configuration option. Think twice before you use it.

In case no explicit Clustered Scheduler Configuration is present, then it is used as the scheduler for exactly one processor.

Definition at line 4101 of file appl-config.h.

#### 8.73.2.5 CONFIGURE_SCHEDULER_EDF

```
#define CONFIGURE_SCHEDULER_EDF
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then Earliest Deadline First Scheduler algorithm is made available to the application.

##### Default Configuration

If this configuration option is undefined, then the described feature is not enabled.

##### Notes

This scheduler configuration option is an advanced configuration option. Think twice before you use it.

In case no explicit Clustered Scheduler Configuration is present, then it is used as the scheduler for exactly one processor.

Definition at line 4127 of file appl-config.h.

#### 8.73.2.6 CONFIGURE_SCHEDULER_EDF_SMP

```
#define CONFIGURE_SCHEDULER_EDF_SMP
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then Earliest Deadline First SMP Scheduler algorithm is made available to the application.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

This scheduler configuration option is an advanced configuration option. Think twice before you use it.

This scheduler algorithm is only available when RTEMS is built with SMP support enabled.

In case no explicit Clustered Scheduler Configuration is present, then it is used as the scheduler for up to 32 processors.

This scheduler algorithm is the default in SMP configurations if CONFIGURE_MAXIMUM_PROCESSORS is greater than one.

Definition at line 4160 of file appl-config.h.

#### 8.73.2.7 CONFIGURE_SCHEDULER_NAME

```
#define CONFIGURE_SCHEDULER_NAME
```

This configuration option is an integer define.

The value of this configuration option defines the name of the default scheduler.

**Default Value**

The default value is

- "MEDF" for the Earliest Deadline First SMP Scheduler,
- "MPA " for the Arbitrary Processor Affinity Priority SMP Scheduler,
- "MPD " for the Deterministic Priority SMP Scheduler,
- "MPS " for the Simple Priority SMP Scheduler,
- "UCBS" for the Constant Bandwidth Server Scheduling (CBS),
- "UEDF" for the Earliest Deadline First Scheduler,
- "UPD " for the Deterministic Priority Scheduler, and
- "UPS " for the Simple Priority Scheduler.

**Value Constraints**

The value of this configuration option shall be a valid integer of type rtems_name.

**Notes**

This scheduler configuration option is an advanced configuration option. Think twice before you use it.

Schedulers can be identified via rtems_scheduler_ident().

Use rtems_build_name() to define the scheduler name.

Definition at line 4221 of file appl-config.h.

#### 8.73.2.8 CONFIGURE_SCHEDULER_PRIORITY

```
#define CONFIGURE_SCHEDULER_PRIORITY
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then Deterministic Priority Scheduler algorithm is made available to the application.

##### Default Configuration

If this configuration option is undefined, then the described feature is not enabled.

##### Notes

This scheduler configuration option is an advanced configuration option. Think twice before you use it.

In case no explicit Clustered Scheduler Configuration is present, then it is used as the scheduler for exactly one processor.

This scheduler algorithm is the default when CONFIGURE_MAXIMUM_PROCESSORS is exactly one.

The memory allocated for this scheduler depends on the CONFIGURE_MAXIMUM_PRIORITY configuration option.

Definition at line 4253 of file appl-config.h.

#### 8.73.2.9 CONFIGURE_SCHEDULER_PRIORITY_AFFINITY_SMP

```
#define CONFIGURE_SCHEDULER_PRIORITY_AFFINITY_SMP
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then Arbitrary Processor Affinity Priority SMP Scheduler algorithm is made available to the application.

##### Default Configuration

If this configuration option is undefined, then the described feature is not enabled.

##### Notes

This scheduler configuration option is an advanced configuration option. Think twice before you use it.

This scheduler algorithm is only available when RTEMS is built with SMP support enabled.

In case no explicit Clustered Scheduler Configuration is present, then it is used as the scheduler for up to 32 processors.

The memory allocated for this scheduler depends on the CONFIGURE_MAXIMUM_PRIORITY configuration option.

Definition at line 4286 of file appl-config.h.

#### 8.73.2.10 CONFIGURE_SCHEDULER_PRIORITY_SMP

```
#define CONFIGURE_SCHEDULER_PRIORITY_SMP
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then Deterministic Priority SMP Scheduler algorithm is made available to the application.

##### Default Configuration

If this configuration option is undefined, then the described feature is not enabled.

##### Notes

This scheduler configuration option is an advanced configuration option. Think twice before you use it.

This scheduler algorithm is only available when RTEMS is built with SMP support enabled.

In case no explicit Clustered Scheduler Configuration is present, then it is used as the scheduler for up to 32 processors.

The memory allocated for this scheduler depends on the CONFIGURE_MAXIMUM_PRIORITY configuration option.

Definition at line 4318 of file appl-config.h.

#### 8.73.2.11 CONFIGURE_SCHEDULER_SIMPLE

```
#define CONFIGURE_SCHEDULER_SIMPLE
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then Simple Priority Scheduler algorithm is made available to the application.

##### Default Configuration

If this configuration option is undefined, then the described feature is not enabled.

##### Notes

This scheduler configuration option is an advanced configuration option. Think twice before you use it.

In case no explicit Clustered Scheduler Configuration is present, then it is used as the scheduler for exactly one processor.

Definition at line 4344 of file appl-config.h.

#### 8.73.2.12 CONFIGURE_SCHEDULER_SIMPLE_SMP

```
#define CONFIGURE_SCHEDULER_SIMPLE_SMP
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then Simple Priority SMP Scheduler algorithm is made available to the application. application.

##### Default Configuration

If this configuration option is undefined, then the described feature is not enabled.

##### Notes

This scheduler configuration option is an advanced configuration option. Think twice before you use it.

This scheduler algorithm is only available when RTEMS is built with SMP support enabled.

In case no explicit Clustered Scheduler Configuration is present, then it is used as the scheduler for up to 32 processors.

Definition at line 4374 of file appl-config.h.

#### 8.73.2.13 CONFIGURE_SCHEDULER_STRONG_APA

```
#define CONFIGURE_SCHEDULER_STRONG_APA
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then Strong APA algorithm is made available to the application.

##### Default Configuration

If this configuration option is undefined, then the described feature is not enabled.

##### Notes

This scheduler configuration option is an advanced configuration option. Think twice before you use it.

This scheduler algorithm is only available when RTEMS is built with SMP support enabled.

This scheduler algorithm is not correctly implemented. Do not use it.

Definition at line 4399 of file appl-config.h.

#### 8.73.2.14 CONFIGURE_SCHEDULER_USER

```
#define CONFIGURE_SCHEDULER_USER
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the user shall provide a scheduler algorithm to the application.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

This scheduler configuration option is an advanced configuration option. Think twice before you use it.

RTEMS allows the application to provide its own task/thread scheduling algorithm. In order to do this, one shall define `CONFIGURE_SCHEDULER_USER` to indicate the application provides its own scheduling algorithm. If `CONFIGURE_SCHEDULER_USER` is defined then the following additional macros shall be defined:

- `CONFIGURE_SCHEDULER` shall be defined to a static definition of the scheduler data structures of the user scheduler.
- `CONFIGURE_SCHEDULER_TABLE_ENTRIES` shall be defined to a scheduler table entry initializer for the user scheduler.
- `CONFIGURE_SCHEDULER_USER_PER_THREAD` shall be defined to the type of the per-thread information of the user scheduler.

At this time, the mechanics and requirements for writing a new scheduler are evolving and not fully documented. It is recommended that you look at the existing Deterministic Priority Scheduler in `cpukit/score/src/schedulerpriority*.c` for guidance. For guidance on the configuration macros, please examine `cpukit/sapi/include/confdefs.h` for how these are defined for the Deterministic Priority Scheduler.

Definition at line 4441 of file appl-config.h.

## 8.74 General System Configuration

### Macros

- #define CONFIGURE_DIRTY_MEMORY

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_DISABLE_NEWLIB_REENTRANCY

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_EXECUTIVE_RAM_SIZE

  *This configuration option is an integer define.*
- #define CONFIGURE_EXTRA_TASK_STACKS

  *This configuration option is an integer define.*
- #define CONFIGURE_INITIAL_EXTENSIONS

  *This configuration option is an initializer define.*
- #define CONFIGURE_INTERRUPT_STACK_SIZE

  *This configuration option is an integer define.*
- #define CONFIGURE_MALLOC_DIRTY

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_MAXIMUM_FILE_DESCRIPTORS

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_PROCESSORS

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_THREAD_NAME_SIZE

  *This configuration option is an integer define.*
- #define CONFIGURE_MEMORY_OVERHEAD

  *This configuration option is an integer define.*
- #define CONFIGURE_MESSAGE_BUFFER_MEMORY

  *This configuration option is an integer define.*
- #define CONFIGURE_MICROSECONDS_PER_TICK

  *This configuration option is an integer define.*
- #define CONFIGURE_MINIMUM_TASK_STACK_SIZE

  *This configuration option is an integer define.*
- #define CONFIGURE_STACK_CHECKER_ENABLED

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_TICKS_PER_TIMESLICE

  *This configuration option is an integer define.*
- #define CONFIGURE_UNIFIED_WORK_AREAS

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_UNLIMITED_ALLOCATION_SIZE

  *This configuration option is an integer define.*
- #define CONFIGURE_UNLIMITED_OBJECTS

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_VERBOSE_SYSTEM_INITIALIZATION

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_ZERO_WORKSPACE_AUTOMATICALLY

  *This configuration option is a boolean feature define.*

### 8.74.1 Detailed Description

This section describes general system configuration options.

## 8.74.2 Macro Definition Documentation

### 8.74.2.1 CONFIGURE_DIRTY_MEMORY

```
#define CONFIGURE_DIRTY_MEMORY
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the memory areas used for the RTEMS Workspace and the C Program Heap are dirtied with a `0xCF` byte pattern during system initialization.

#### Default Configuration

If this configuration option is undefined, then the described feature is not enabled.

#### Notes

Dirtying memory can add significantly to system initialization time. It may assist in finding code that incorrectly assumes the contents of free memory areas is cleared to zero during system initialization. In case CONFIGURE_ZERO_WORKSPACE_AUTOMATICALLY is also defined, then the memory is first dirtied and then zeroed.

See also CONFIGURE_MALLOC_DIRTY.

Definition at line 2450 of file appl-config.h.

### 8.74.2.2 CONFIGURE_DISABLE_NEWLIB_REENTRANCY

```
#define CONFIGURE_DISABLE_NEWLIB_REENTRANCY
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the Newlib reentrancy support per thread is disabled and a global reentrancy structure is used.

#### Default Configuration

If this configuration option is undefined, then the described feature is not enabled.

#### Notes

You can enable this option to reduce the size of the :term:`TCB`. Use this option with care, since it can lead to race conditions and undefined system behaviour. For example, #errno is no longer a thread-local variable if this option is enabled.

Definition at line 2470 of file appl-config.h.

#### 8.74.2.3 CONFIGURE_EXECUTIVE_RAM_SIZE

```
#define CONFIGURE_EXECUTIVE_RAM_SIZE
```

This configuration option is an integer define.

The value of this configuration option defines the RTEMS Workspace size in bytes.

**Default Value**

If this configuration option is undefined, then the RTEMS Workspace and task stack space size is calculated by <rtems/confdefs.h> based on the values configuration options.

**Value Constraints**

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to UINTPTR_MAX.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.

**Notes**

This is an advanced configuration option. Use it only if you know exactly what you are doing.

Definition at line 2504 of file appl-config.h.

#### 8.74.2.4 CONFIGURE_EXTRA_TASK_STACKS

```
#define CONFIGURE_EXTRA_TASK_STACKS
```

This configuration option is an integer define.

The value of this configuration option defines the number of bytes the applications wishes to add to the task stack requirements calculated by <rtems/confdefs.h>.

**Default Value**

The default value is 0.

**Value Constraints**

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be small enough so that the task stack space calculation carried out by <rtems/confdefs.h> does not overflow an integer of type uintptr_t.

**Notes**

This parameter is very important. If the application creates tasks with stacks larger then the minimum, then that memory is **not** accounted for by <rtems/confdefs.h>.

Definition at line 2535 of file appl-config.h.

#### 8.74.2.5 CONFIGURE_INITIAL_EXTENSIONS

```
#define CONFIGURE_INITIAL_EXTENSIONS
```

This configuration option is an initializer define.

The value of this configuration option is used to initialize the table of initial user extensions.

**Default Value**

The default value is the empty list.

**Value Constraints**

The value of this configuration option shall be a list of initializers for structures of type rtems_extensions_table.

**Notes**

The value of this configuration option is placed before the entries of BSP_INITIAL_EXTENSION and after the entries of all other initial user extensions.

Definition at line 2557 of file appl-config.h.

#### 8.74.2.6 CONFIGURE_INTERRUPT_STACK_SIZE

```
#define CONFIGURE_INTERRUPT_STACK_SIZE
```

This configuration option is an integer define.

The value of this configuration option defines the size of an interrupt stack in bytes.

**Default Value**

The default value is BSP_INTERRUPT_STACK_SIZE in case it is defined, otherwise the default value is CPU_STACK_MINIMUM_SIZE.

**Value Constraints**

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to a BSP-specific and application-specific minimum value.
- It shall be small enough so that the interrupt stack area calculation carried out by <rtems/confdefs.h> does not overflow an integer of type size_t.
- It shall be aligned according to #CPU_INTERRUPT_STACK_ALIGNMENT.

**Notes**

There is one interrupt stack available for each configured processor (CONFIGURE_MAXIMUM_PROCESSORS). The interrupt stack areas are statically allocated in a special linker section (.rtemsstack.interrupt). The placement of this linker section is BSP-specific.

Some BSPs use the interrupt stack as the initialization stack which is used to perform the sequential system initialization before the multithreading is started.

The interrupt stacks are covered by the stack checker, see CONFIGURE_STACK_CHECKER_ENABLED. However, using a too small interrupt stack size may still result in undefined behaviour.

In releases before RTEMS 5.1 the default value was CONFIGURE_MINIMUM_TASK_STACK_SIZE instead of CPU_STACK_MINIMUM_SIZE.

Definition at line 2605 of file appl-config.h.

#### 8.74.2.7 CONFIGURE_MALLOC_DIRTY

```
#define CONFIGURE_MALLOC_DIRTY
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then each memory area returned by C Program Heap allocator functions such as malloc() is dirtied with a `0xCF` byte pattern before it is handed over to the application.

##### Default Configuration

If this configuration option is undefined, then the described feature is not enabled.

##### Notes

The dirtying performed by this option is carried out for each successful memory allocation from the C Program Heap in contrast to CONFIGURE_DIRTY_MEMORY which dirties the memory only once during the system initialization.

Definition at line 2626 of file appl-config.h.

#### 8.74.2.8 CONFIGURE_MAXIMUM_FILE_DESCRIPTORS

```
#define CONFIGURE_MAXIMUM_FILE_DESCRIPTORS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of file like objects that can be concurrently open.

##### Default Value

The default value is 3.

##### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to SIZE_MAX.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.

##### Notes

The default value of three file descriptors allows RTEMS to support standard input, output, and error I/O streams on `/dev/console`.

Definition at line 2658 of file appl-config.h.

#### 8.74.2.9 CONFIGURE_MAXIMUM_PROCESSORS

```
#define CONFIGURE_MAXIMUM_PROCESSORS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of processors an application intends to use. The number of actually available processors depends on the hardware and may be less. It is recommended to use the smallest value suitable for the application in order to save memory. Each processor needs an IDLE task stack and interrupt stack for example.

**Default Value**

The default value is 1.

**Value Constraints**

The value of this configuration option shall be greater than or equal to 1 and less than or equal to #CPU_M↩ AXIMUM_PROCESSORS.

**Notes**

If there are more processors available than configured, the rest will be ignored.

This configuration option is only evaluated in SMP configurations (e.g. RTEMS was built with the `--enable-smp` build configuration option). In all other configurations it has no effect.

Definition at line 2688 of file appl-config.h.

#### 8.74.2.10 CONFIGURE_MAXIMUM_THREAD_NAME_SIZE

```
#define CONFIGURE_MAXIMUM_THREAD_NAME_SIZE
```

This configuration option is an integer define.

The value of this configuration option defines the maximum thread name size including the terminating `NUL` character.

**Default Value**

The default value is 16.

**Value Constraints**

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to `SIZE_MAX`.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.

**Notes**

The default value was chosen for Linux compatibility, see `PTHREAD_SETNAME_NP(3)`.

The size of the thread control block is increased by the maximum thread name size.

This configuration option is available since RTEMS 5.1.

Definition at line 2727 of file appl-config.h.

#### 8.74.2.11 CONFIGURE_MEMORY_OVERHEAD

```
#define CONFIGURE_MEMORY_OVERHEAD
```

This configuration option is an integer define.

The value of this configuration option defines the number of kilobytes the application wishes to add to the RTEMS Workspace size calculated by <rtems/confdefs.h>.

##### Default Value

The default value is 0.

##### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It shall be small enough so that the RTEMS Workspace size calculation carried out by <rtems/confdefs.h> does not overflow an integer of type uintptr_t.

##### Notes

This configuration option should only be used when it is suspected that a bug in <rtems/confdefs.h> has resulted in an underestimation. Typically the memory allocation will be too low when an application does not account for all message queue buffers or task stacks, see CONFIGURE_MESSAGE_BUFFER_MEMORY.

Definition at line 2764 of file appl-config.h.

#### 8.74.2.12 CONFIGURE_MESSAGE_BUFFER_MEMORY

```
#define CONFIGURE_MESSAGE_BUFFER_MEMORY
```

This configuration option is an integer define.

The value of this configuration option defines the number of bytes reserved for message queue buffers in the RTEMS Workspace.

##### Default Value

The default value is 0.

##### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It shall be small enough so that the RTEMS Workspace size calculation carried out by <rtems/confdefs.h> does not overflow an integer of type uintptr_t.

**Notes**

The configuration options CONFIGURE_MAXIMUM_MESSAGE_QUEUES and CONFIGURE_MAXIMUM_POSIX_MESSAGE_ define only how many message queues can be created by the application. The memory for the message buffers is configured by this option. For each message queue you have to reserve some memory for the message buffers. The size depends on the maximum number of pending messages and the maximum size of the messages of a message queue. Use the CONFIGURE_MESSAGE_BUFFERS_FOR_QUEUE() macro to specify the message buffer memory for each message queue and sum them up to define the value for CONFIGURE_MAXIMUM_MESSAGE_QUEUES.

The interface for the CONFIGURE_MESSAGE_BUFFERS_FOR_QUEUE() help macro is as follows:

```
CONFIGURE_MESSAGE_BUFFERS_FOR_QUEUE( max_messages, max_msg_size )
```

Where max_messages is the maximum number of pending messages and max_msg_size is the maximum size in bytes of the messages of the corresponding message queue. Both parameters shall be compile time constants. Not using this help macro (e.g. just using max_messages * max_msg_size) may result in an underestimate of the RTEMS Workspace size.

The following example illustrates how the CONFIGURE_MESSAGE_BUFFERS_FOR_QUEUE() help macro can be used to assist in calculating the message buffer memory required. In this example, there are two message queues used in this application. The first message queue has a maximum of 24 pending messages with the message structure defined by the type one_message_type. The other message queue has a maximum of 500 pending messages with the message structure defined by the type other_message_↩ type.

```
#define CONFIGURE_MESSAGE_BUFFER_MEMORY ( \
    CONFIGURE_MESSAGE_BUFFERS_FOR_QUEUE( \
      24, \
      sizeof( one_message_type ) \
    ) \
    + CONFIGURE_MESSAGE_BUFFERS_FOR_QUEUE( \
      500, \
      sizeof( other_message_type ) \
    ) \
  )
```

Definition at line 2841 of file appl-config.h.

#### 8.74.2.13 CONFIGURE_MICROSECONDS_PER_TICK

```
#define CONFIGURE_MICROSECONDS_PER_TICK
```

This configuration option is an integer define.

The value of this configuration option defines the length of time in microseconds between clock ticks (clock tick quantum).

When the clock tick quantum value is too low, the system will spend so much time processing clock ticks that it does not have processing time available to perform application work. In this case, the system will become unresponsive.

The lowest practical time quantum varies widely based upon the speed of the target hardware and the architectural overhead associated with interrupts. In general terms, you do not want to configure it lower than is needed for the application.

The clock tick quantum should be selected such that it all blocking and delay times in the application are evenly divisible by it. Otherwise, rounding errors will be introduced which may negatively impact the application.

**Default Value**

The default value is 10000.

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to a Clock Driver specific value.
- It shall be less than or equal to a Clock Driver specific value.
- The resulting clock ticks per second should be an integer.

#### Notes

This configuration option has no impact if the Clock Driver is not configured, see CONFIGURE_APPLICATION_DOES_NOT_NE

There may be Clock Driver specific limits on the resolution or maximum value of a clock tick quantum.

Definition at line 2890 of file appl-config.h.

### 8.74.2.14 CONFIGURE_MINIMUM_TASK_STACK_SIZE

```
#define CONFIGURE_MINIMUM_TASK_STACK_SIZE
```

This configuration option is an integer define.

The value of this configuration option defines the minimum stack size in bytes for every user task or thread in the system.

#### Default Value

The default value is CPU_STACK_MINIMUM_SIZE.

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be small enough so that the task stack space calculation carried out by <rtems/confdefs.h> does not overflow an integer of type uintptr_t.
- It shall be greater than or equal to a BSP-specific and application-specific minimum value.

#### Notes

Adjusting this parameter should be done with caution. Examining the actual stack usage using the stack checker usage reporting facility is recommended (see also CONFIGURE_STACK_CHECKER_ENABLED).

This parameter can be used to lower the minimum from that recommended. This can be used in low memory systems to reduce memory consumption for stacks. However, this shall be done with caution as it could increase the possibility of a blown task stack.

This parameter can be used to increase the minimum from that recommended. This can be used in higher memory systems to reduce the risk of stack overflow without performing analysis on actual consumption.

By default, this configuration parameter defines also the minimum stack size of POSIX threads. This can be changed with the CONFIGURE_MINIMUM_POSIX_THREAD_STACK_SIZE configuration option.

In releases before RTEMS 5.1 the CONFIGURE_MINIMUM_TASK_STACK_SIZE was used to define the default value of CONFIGURE_INTERRUPT_STACK_SIZE.

Definition at line 2939 of file appl-config.h.

#### 8.74.2.15 CONFIGURE_STACK_CHECKER_ENABLED

```
#define CONFIGURE_STACK_CHECKER_ENABLED
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the stack checker is enabled.

##### Default Configuration

If this configuration option is undefined, then the described feature is not enabled.

##### Notes

The stack checker performs run-time stack bounds checking. This increases the time required to create tasks as well as adding overhead to each context switch.

In 4.9 and older, this configuration option was named STACK_CHECKER_ON.

Definition at line 2962 of file appl-config.h.

#### 8.74.2.16 CONFIGURE_TICKS_PER_TIMESLICE

```
#define CONFIGURE_TICKS_PER_TIMESLICE
```

This configuration option is an integer define.

The value of this configuration option defines the length of the timeslice quantum in ticks for each task.

##### Default Value

The default value is 50.

##### Value Constraints

The value of this configuration option shall be greater than or equal to 0 and less than or equal to UIN↩T32_MAX.

##### Notes

This configuration option has no impact if the Clock Driver is not configured, see CONFIGURE_APPLICATION_DOES_NOT_NE

Definition at line 2984 of file appl-config.h.

#### 8.74.2.17 CONFIGURE_UNIFIED_WORK_AREAS

```
#define CONFIGURE_UNIFIED_WORK_AREAS
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the RTEMS Workspace and the C Program Heap will be one pool of memory.

##### Default Configuration

If this configuration option is undefined, then there will be separate memory pools for the RTEMS Workspace and C Program Heap.

##### Notes

Having separate pools does have some advantages in the event a task blows a stack or writes outside its memory area. However, in low memory systems the overhead of the two pools plus the potential for unused memory in either pool is very undesirable.

In high memory environments, this is desirable when you want to use the `Unlimited Objects` option. You will be able to create objects until you run out of all available memory rather then just until you run out of RTEMS Workspace.

Definition at line 3011 of file appl-config.h.

#### 8.74.2.18 CONFIGURE_UNLIMITED_ALLOCATION_SIZE

```
#define CONFIGURE_UNLIMITED_ALLOCATION_SIZE
```

This configuration option is an integer define.

If CONFIGURE_UNLIMITED_OBJECTS is defined, then the value of this configuration option defines the default objects maximum of all object classes supporting `Unlimited Objects` to `rtems_resource_unlimited( CONFIGURE_UNLIMITED_ALLOCATION_SIZE )`.

##### Default Value

The default value is 8.

##### Value Constraints

The value of this configuration option shall meet the constraints of all object classes to which it is applied.

##### Notes

By allowing users to declare all resources as being unlimited the user can avoid identifying and limiting the resources used.

The object maximum of each class can be configured also individually using the rtems_resource_unlimited() macro.

Definition at line 3041 of file appl-config.h.

#### 8.74.2.19 CONFIGURE_UNLIMITED_OBJECTS

```
#define CONFIGURE_UNLIMITED_OBJECTS
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then unlimited objects are used by default.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

When using unlimited objects, it is common practice to also specify CONFIGURE_UNIFIED_WORK_AREAS so the system operates with a single pool of memory for both RTEMS Workspace and C Program Heap.

This option does not override an explicit configuration for a particular object class by the user.

See also CONFIGURE_UNLIMITED_ALLOCATION_SIZE.

Definition at line 3067 of file appl-config.h.

#### 8.74.2.20 CONFIGURE_VERBOSE_SYSTEM_INITIALIZATION

```
#define CONFIGURE_VERBOSE_SYSTEM_INITIALIZATION
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the system initialization is verbose.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

You may use this feature to debug system initialization issues. The printk() function is used to print the information.

Definition at line 3085 of file appl-config.h.

#### 8.74.2.21 CONFIGURE_ZERO_WORKSPACE_AUTOMATICALLY

```
#define CONFIGURE_ZERO_WORKSPACE_AUTOMATICALLY
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the memory areas used for the RTEMS Workspace and the C Program Heap are zeroed with a `0x00` byte pattern during system initialization.

**Default Configuration**

If this configuration option is undefined, then the described feature is not enabled.

**Notes**

Zeroing memory can add significantly to the system initialization time. It is not necessary for RTEMS but is often assumed by support libraries. In case CONFIGURE_DIRTY_MEMORY is also defined, then the memory is first dirtied and then zeroed.

Definition at line 3106 of file appl-config.h.

## 8.75 Generic Checks

Checks for data types with an equality or inequality operator.

### Macros

- #define **T_eq**(a, ...) T_flags_eq(0, a, __VA_ARGS__)
- #define **T_assert_eq**(a, ...) T_flags_eq(T_CHECK_STOP, a, __VA_ARGS__)
- #define **T_quiet_eq**(a, ...) T_flags_eq(T_CHECK_QUIET, a, __VA_ARGS__)
- #define **T_step_eq**(s, a, ...) T_flags_eq(T_CHECK_STEP(s), a, __VA_ARGS__)
- #define **T_step_assert_eq**(s, a, ...) T_flags_eq(T_CHECK_STEP(s) | T_CHECK_STOP, a, __VA_ARG↩ S__)
- #define **T_ne**(a, ...) T_flags_ne(0, a, __VA_ARGS__)
- #define **T_assert_ne**(a, ...) T_flags_ne(T_CHECK_STOP, a, __VA_ARGS__)
- #define **T_quiet_ne**(a, ...) T_flags_ne(T_CHECK_QUIET, a, __VA_ARGS__)
- #define **T_step_ne**(s, a, ...) T_flags_ne(T_CHECK_STEP(s), a, __VA_ARGS__)
- #define **T_step_assert_ne**(s, a, ...) T_flags_ne(T_CHECK_STEP(s) | T_CHECK_STOP, a, __VA_ARG↩ S__)

### 8.75.1 Detailed Description

Checks for data types with an equality or inequality operator.

## 8.76 Heap Handler

The Heap Handler provides a heap.

### Modules

- Protected Heap Handler

  *Provides protected heap services.*

### Files

- file heap.h

  *Heap Handler API.*
- file heapimpl.h

  *Heap Handler Implementation.*
- file heapinfo.h

  *Heap Handler Information API.*
- file heap.c

  *Heap Handler implementation.*
- file heapallocate.c

  *Heap Handler implementation.*

### Classes

- struct Heap_Error_context

  *Context of a heap error.*
- struct Heap_Block

  *Description for free or used blocks.*
- struct Heap_Control

  *Control block used to manage a heap.*
- struct Heap_Area

  *Heap area structure for table based heap initialization and extension.*
- struct Heap_Statistics

  *Run-time heap statistics.*
- struct Heap_Information

  *Information about blocks.*
- struct Heap_Information_block

  *Information block returned by _Heap_Get_information().*

### Macros

- #define **HEAP_PROTECTION_HEADER_SIZE** 0
- #define HEAP_BLOCK_HEADER_SIZE (2 * sizeof(uintptr_t) + HEAP_PROTECTION_HEADER_SIZE)

  *The block header consists of the two size fields (Heap_Block::prev_size and Heap_Block::size_and_flag).*
- #define HEAP_PREV_BLOCK_USED ((uintptr_t) 1)

  *See also Heap_Block::size_and_flag.*
- #define HEAP_ALLOC_BONUS sizeof(uintptr_t)

  *Size of the part at the block begin which may be used for allocation in charge of the previous block.*
- #define **_Heap_Protection_block_initialize**(heap, block) ((void) 0)
- #define **_Heap_Protection_block_check**(heap, block) ((void) 0)
- #define **_Heap_Protection_block_error**(heap, block, reason) ((void) 0)
- #define **_Heap_Protection_free_all_delayed_blocks**(heap) ((void) 0)
- #define **_HAssert**(cond) ((void) 0)

## Typedefs

- typedef struct Heap_Control **Heap_Control**
- typedef struct Heap_Block **Heap_Block**
- typedef uintptr_t(∗ Heap_Initialization_or_extend_handler) (Heap_Control ∗heap, void ∗area_begin, uintptr_t area_size, uintptr_t page_size_or_unused)

  *Heap initialization and extend handler type.*
- typedef bool(∗ Heap_Block_visitor) (const Heap_Block ∗block, uintptr_t block_size, bool block_is_used, void ∗visitor_arg)

  *Heap block visitor.*

## Enumerations

- enum Heap_Error_reason { HEAP_ERROR_BROKEN_PROTECTOR, HEAP_ERROR_FREE_PATTERN, HEAP_ERROR_DOUBLE_FREE, HEAP_ERROR_BAD_USED_BLOCK, HEAP_ERROR_BAD_FREE_BLOCK }

  *The heap error reason.*
- enum Heap_Resize_status { **HEAP_RESIZE_SUCCESSFUL**, **HEAP_RESIZE_UNSATISFIED**, **HEAP_RESIZE_FATAL_ERROR** }

  *See _Heap_Resize_block().*

## Functions

- uintptr_t _Heap_Extend (Heap_Control ∗heap, void ∗area_begin, uintptr_t area_size, uintptr_t unused)

  *Extends the memory available for the heap.*
- uintptr_t _Heap_No_extend (Heap_Control ∗unused_0, void ∗unused_1, uintptr_t unused_2, uintptr_t unused_3)

  *This function returns always zero.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Align_up (uintptr_t value, uintptr_t alignment)

  *Aligns the value to a given alignment, rounding up.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Min_block_size (uintptr_t page_size)

  *Returns the minimal Heap Block size for the given page_size.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Area_overhead (uintptr_t page_size)

  *Returns the worst case overhead to manage a memory area.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Size_with_overhead (uintptr_t page_size, uintptr_t size, uintptr_t alignment)

  *Returns the size with administration and alignment overhead for one allocation.*
- bool _Heap_Get_first_and_last_block (uintptr_t heap_area_begin, uintptr_t heap_area_size, uintptr_t page_size, uintptr_t min_block_size, Heap_Block ∗∗first_block_ptr, Heap_Block ∗∗last_block_ptr)

  *Gets the first and last block for the heap area.*
- uintptr_t _Heap_Initialize (Heap_Control ∗heap, void ∗area_begin, uintptr_t area_size, uintptr_t page_size)

  *Initializes the heap control block.*
- void ∗ _Heap_Allocate_aligned_with_boundary (Heap_Control ∗heap, uintptr_t size, uintptr_t alignment, uintptr_t boundary)

  *Allocates an aligned memory area with boundary constraint.*
- RTEMS_INLINE_ROUTINE void ∗ _Heap_Allocate_aligned (Heap_Control ∗heap, uintptr_t size, uintptr_t alignment)

  *Allocates an aligned memory area.*
- RTEMS_INLINE_ROUTINE void ∗ _Heap_Allocate (Heap_Control ∗heap, uintptr_t size)

  *Allocates a memory area.*

- bool _Heap_Free (Heap_Control *heap, void *addr)

  *Frees the allocated memory area.*
- bool _Heap_Walk (Heap_Control *heap, int source, bool dump)

  *Verifies the integrity of the heap.*
- void _Heap_Iterate (Heap_Control *heap, Heap_Block_visitor visitor, void *visitor_arg)

  *Iterates over all blocks of the heap.*
- Heap_Block * _Heap_Greedy_allocate (Heap_Control *heap, const uintptr_t *block_sizes, size_t block_count)

  *Greedily allocates and empties the heap.*
- Heap_Block * _Heap_Greedy_allocate_all_except_largest (Heap_Control *heap, uintptr_t *allocatable_size)

  *Greedily allocates all blocks except the largest free block.*
- void _Heap_Greedy_free (Heap_Control *heap, Heap_Block *blocks)

  *Frees blocks of a greedy allocation.*
- void _Heap_Get_information (Heap_Control *heap, Heap_Information_block *info)

  *Returns information about used and free blocks for the heap.*
- void _Heap_Get_free_information (Heap_Control *heap, Heap_Information *info)

  *Returns information about free blocks for the heap.*
- bool _Heap_Size_of_alloc_area (Heap_Control *heap, void *addr, uintptr_t *size)

  *Returns the size of the allocatable memory area.*
- Heap_Resize_status _Heap_Resize_block (Heap_Control *heap, void *addr, uintptr_t size, uintptr_t *old_size, uintptr_t *new_size)

  *Resizes the block of the allocated memory area.*
- Heap_Block * _Heap_Block_allocate (Heap_Control *heap, Heap_Block *block, uintptr_t alloc_begin, uintptr_t alloc_size)

  *Allocates the memory area. starting at alloc_begin of size alloc_size bytes in the block block.*
- RTEMS_INLINE_ROUTINE void _Heap_Protection_set_delayed_free_fraction (Heap_Control *heap, uintptr_t fraction)

  *Sets the fraction of delayed free blocks that is actually freed during memory shortage.*
- RTEMS_INLINE_ROUTINE Heap_Block * _Heap_Free_list_head (Heap_Control *heap)

  *Returns the head of the free list of the heap.*
- RTEMS_INLINE_ROUTINE Heap_Block * _Heap_Free_list_tail (Heap_Control *heap)

  *Returns the tail of the free list of the heap.*
- RTEMS_INLINE_ROUTINE Heap_Block * _Heap_Free_list_first (Heap_Control *heap)

  *Returns the first block of the free list of the heap.*
- RTEMS_INLINE_ROUTINE Heap_Block * _Heap_Free_list_last (Heap_Control *heap)

  *Returns the last block of the free list of the heap.*
- RTEMS_INLINE_ROUTINE void _Heap_Free_list_remove (Heap_Block *block)

  *Removes the block from the free list.*
- RTEMS_INLINE_ROUTINE void _Heap_Free_list_replace (Heap_Block *old_block, Heap_Block *new_block)

  *Replaces one block in the free list by another.*
- RTEMS_INLINE_ROUTINE void _Heap_Free_list_insert_after (Heap_Block *block_before, Heap_Block *new_block)

  *Inserts a block after an existing block in the free list.*
- RTEMS_INLINE_ROUTINE void _Heap_Free_list_insert_before (Heap_Block *block_next, Heap_Block *new_block)

  *Inserts a block before an existing block in the free list.*
- RTEMS_INLINE_ROUTINE bool _Heap_Is_aligned (uintptr_t value, uintptr_t alignment)

  *Checks if the value is aligned to the given alignment.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Align_down (uintptr_t value, uintptr_t alignment)

  *Returns the aligned value, truncating.*
- RTEMS_INLINE_ROUTINE Heap_Block * _Heap_Block_at (const Heap_Block *block, uintptr_t offset)

*Returns the block which is offset away from block.*

- RTEMS_INLINE_ROUTINE Heap_Block * _Heap_Prev_block (const Heap_Block *block)

  *Returns the address of the previous block.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Alloc_area_of_block (const Heap_Block *block)

  *Returns the first address in the block without the heap header.*
- RTEMS_INLINE_ROUTINE Heap_Block * _Heap_Block_of_alloc_area (uintptr_t alloc_begin, uintptr_t page_size)

  *Returns the starting address of the block corresponding to the allocatable area.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Block_size (const Heap_Block *block)

  *Returns the block size.*
- RTEMS_INLINE_ROUTINE void _Heap_Block_set_size (Heap_Block *block, uintptr_t size)

  *Sets the block size.*
- RTEMS_INLINE_ROUTINE bool _Heap_Is_prev_used (const Heap_Block *block)

  *Returns if the previous heap block is used.*
- RTEMS_INLINE_ROUTINE bool _Heap_Is_used (const Heap_Block *block)

  *Returns if the heap block is used.*
- RTEMS_INLINE_ROUTINE bool _Heap_Is_free (const Heap_Block *block)

  *Returns if the heap block is free.*
- RTEMS_INLINE_ROUTINE bool _Heap_Is_block_in_heap (const Heap_Control *heap, const Heap_Block *block)

  *Returns if the block is part of the heap.*
- RTEMS_INLINE_ROUTINE void _Heap_Set_last_block_size (Heap_Control *heap)

  *Sets the size of the last block for the heap.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Get_size (const Heap_Control *heap)

  *Returns the size of the allocatable area in bytes.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Max (uintptr_t a, uintptr_t b)

  *Returns the bigger one of the two arguments.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Min (uintptr_t a, uintptr_t b)

  *Returns the smaller one of the two arguments.*

### 8.76.1 Detailed Description

The Heap Handler provides a heap.

A heap is a doubly linked list of variable size blocks which are allocated using the first fit method. Garbage collection is performed each time a block is returned to the heap by coalescing neighbor blocks. Control information for both allocated and free blocks is contained in the heap area. A heap control structure contains control information for the heap.

The alignment routines could be made faster should we require only powers of two to be supported for page size, alignment and boundary arguments. The minimum alignment requirement for pages is currently CPU_ALIGNMENT and this value is only required to be multiple of two and explicitly not required to be a power of two.

There are two kinds of blocks. One sort describes a free block from which we can allocate memory. The other blocks are used and provide an allocated memory area. The free blocks are accessible via a list of free blocks.

Blocks or areas cover a continuous set of memory addresses. They have a begin and end address. The end address is not part of the set. The size of a block or area equals the distance between the begin and end address in units of bytes.

Free blocks look like:

| | |
|---|---|
| Heap_Block | previous block size in case the previous block is free,<br>otherwise it may contain data used by the previous block |
| | block size and a flag which indicates if the previous block is free or used,<br>this field contains always valid data regardless of the block usage |
| | pointer to next block (this field is page size aligned) |
| | pointer to previous block |
| free space | |

Used blocks look like:

| | |
|---|---|
| Heap_Block | previous block size in case the previous block is free,<br>otherwise it may contain data used by the previous block |
| | block size and a flag which indicates if the previous block is free or used,<br>this field contains always valid data regardless of the block usage |
| | begin of allocated area (this field is page size aligned) |
| | allocated space |
| allocated space | |

The heap area after initialization contains two blocks and looks like:

| Label | Content | |
|---|---|---|
| heap->area_begin | heap area begin address | |
| first_block->prev_size | subordinate heap area end address (this will be used to maintain a linked list of scattered heap areas) | |
| first_block->size | size available for allocation \| HEAP_PREV_BLOCK_USED | |
| first_block->next | _Heap_Free_list_tail(heap) | memory area available for allocation |
| first_block->prev | _Heap_Free_list_head(heap) | |
| ... | | |
| last_block->prev_size | size of first block | |
| last_block->size | first block begin address - last block begin address | |
| heap->area_end | heap area end address | |

The next block of the last block is the first block. Since the first block indicates that the previous block is used, this ensures that the last block appears as used for the _Heap_Is_used() and _Heap_Is_free() functions.

## 8.76.2 Typedef Documentation

### 8.76.2.1 Heap_Block_visitor

```
typedef bool(* Heap_Block_visitor) (const Heap_Block *block, uintptr_t block_size, bool block↩
_is_used, void *visitor_arg)
```

Heap block visitor.

**See also**

_Heap_Iterate().

**Return values**

| | |
|---|---|
| *true* | Stop the iteration. |
| *false* | Continue the iteration. |

Definition at line 204 of file heapimpl.h.

#### 8.76.2.2 Heap_Initialization_or_extend_handler

```
typedef uintptr_t(* Heap_Initialization_or_extend_handler) (Heap_Control *heap, void *area_↩
begin, uintptr_t area_size, uintptr_t page_size_or_unused)
```

Heap initialization and extend handler type.

This helps to do a table based heap initialization and extension. Create a table of Heap_Area elements and iterate through it. Set the handler to _Heap_Initialize() in the first iteration and then to _Heap_Extend().

**See also**

Heap_Area, _Heap_Initialize(), _Heap_Extend(), or _Heap_No_extend().

Definition at line 352 of file heap.h.

### 8.76.3 Enumeration Type Documentation

#### 8.76.3.1 Heap_Error_reason

```
enum Heap_Error_reason
```

The heap error reason.

**See also**

_Heap_Protection_block_error().

**Enumerator**

| | |
|---|---|
| HEAP_ERROR_BROKEN_PROTECTOR | There is an unexpected value in the heap block protector area. |
| HEAP_ERROR_FREE_PATTERN | There is an unexpected value in the free pattern of a free heap block. |
| HEAP_ERROR_DOUBLE_FREE | There is was an attempt to free the same block twice. |
| HEAP_ERROR_BAD_USED_BLOCK | The next block of a supposed to be used block does not indicate that the block is used. |
| HEAP_ERROR_BAD_FREE_BLOCK | A supposed to be free block is not inside the heap memory area. |

Definition at line 142 of file heap.h.

## 8.76.4 Function Documentation

### 8.76.4.1 _Heap_Align_down()

```
RTEMS_INLINE_ROUTINE uintptr_t _Heap_Align_down (
              uintptr_t value,
              uintptr_t alignment )
```

Returns the aligned value, truncating.

**Parameters**

| | |
|---|---|
| *value* | The value to be aligned |
| *alignment* | The alignment for the operation. |

**Returns**

The aligned value, truncated.

Definition at line 590 of file heapimpl.h.

### 8.76.4.2 _Heap_Align_up()

```
RTEMS_INLINE_ROUTINE uintptr_t _Heap_Align_up (
              uintptr_t value,
              uintptr_t alignment )
```

Aligns the value to a given alignment, rounding up.

**Parameters**

| | |
|---|---|
| *value* | The value to be aligned. |
| *alignment* | The alignment for the value. |

**Returns**

The *value* aligned to the given *alignment*, rounded up.

Definition at line 413 of file heap.h.

#### 8.76.4.3 _Heap_Alloc_area_of_block()

```
RTEMS_INLINE_ROUTINE uintptr_t _Heap_Alloc_area_of_block (
              const Heap_Block * block )
```

Returns the first address in the block without the heap header.

**Parameters**

| | |
|---|---|
| *block* | The block for the operation. |

**Returns**

The first address after the heap header.

Definition at line 635 of file heapimpl.h.

#### 8.76.4.4 _Heap_Allocate()

```
RTEMS_INLINE_ROUTINE void* _Heap_Allocate (
              Heap_Control * heap,
              uintptr_t size )
```

Allocates a memory area.

A size value of zero will return a unique address which may be freed with _Heap_Free().

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap to allocate a memory are from. |
| | *size* | The size of the desired memory are in bytes. |

**Return values**

| | |
|---|---|
| *pointer* | The starting address of the allocated memory area. |
| *NULL* | No memory is available of the parameters are inconsistent. |

Definition at line 160 of file heapimpl.h.

#### 8.76.4.5 _Heap_Allocate_aligned()

```
RTEMS_INLINE_ROUTINE void* _Heap_Allocate_aligned (
              Heap_Control * heap,
              uintptr_t size,
              uintptr_t alignment )
```

Allocates an aligned memory area.

A size value of zero will return a unique address which may be freed with _Heap_Free().

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap to allocate a memory are from. |
| | *size* | The size of the desired memory are in bytes. |
| | *alignment* | The allocated memory area will begin at an address aligned by this value. |

**Return values**

| | |
|---|---|
| *pointer* | The starting address of the allocated memory area. |
| *NULL* | No memory is available of the parameters are inconsistent. |

Definition at line 139 of file heapimpl.h.

#### 8.76.4.6 _Heap_Allocate_aligned_with_boundary()

```
void* _Heap_Allocate_aligned_with_boundary (
              Heap_Control * heap,
              uintptr_t size,
              uintptr_t alignment,
              uintptr_t boundary )
```

Allocates an aligned memory area with boundary constraint.

A size value of zero will return a unique address which may be freed with _Heap_Free().

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap to allocate a memory are from. |
| | *size* | The size of the desired memory are in bytes. |
| | *alignment* | The allocated memory area will begin at an address aligned by this value. |
| | *boundary* | The allocated memory area will fulfill a boundary constraint, if this value is not equal to zero. The boundary value specifies the set of addresses which are aligned by the boundary value. The interior of the allocated memory area will not contain an element of this set. The begin or end address of the area may be a member of the set. |

**Return values**

| | |
|---|---|
| *pointer* | The starting address of the allocated memory area. |
| *NULL* | No memory is available of the parameters are inconsistent. |

Definition at line 192 of file heapallocate.c.

#### 8.76.4.7 _Heap_Area_overhead()

```
RTEMS_INLINE_ROUTINE uintptr_t _Heap_Area_overhead (
              uintptr_t page_size )
```

Returns the worst case overhead to manage a memory area.

**Parameters**

| | |
|---|---|
| *page_size* | The page size to calculate the worst case memory manage overhead. |

**Returns**

The worst case overhead to manage a memory area.

Definition at line 446 of file heap.h.

#### 8.76.4.8 _Heap_Block_allocate()

```
Heap_Block* _Heap_Block_allocate (
              Heap_Control * heap,
              Heap_Block * block,
              uintptr_t alloc_begin,
              uintptr_t alloc_size )
```

Allocates the memory area. starting at *alloc_begin* of size *alloc_size* bytes in the block *block*.

The block may be split up into multiple blocks. The previous and next block may be used or free. Free block parts which form a vaild new block will be inserted into the free list or merged with an adjacent free block. If the block is used, they will be inserted after the free list head. If the block is free, they will be inserted after the previous block in the free list.

Inappropriate values for *alloc_begin* or *alloc_size* may corrupt the heap.

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap to operate upon. |
| | *block* | The block in which the memory area should be allocated |
| | *alloc_begin* | The starting address of the memory area that shall be allocated. |
| | *alloc_size* | The size of the desired allocated area in bytes. |

**Returns**

The block containing the allocated memory area.

Definition at line 431 of file heap.c.

#### 8.76.4.9 _Heap_Block_at()

```
RTEMS_INLINE_ROUTINE Heap_Block* _Heap_Block_at (
              const Heap_Block * block,
              uintptr_t offset )
```

Returns the block which is *offset* away from *block*.

**Parameters**

| | |
|---|---|
| *block* | The block for the relative calculation. |
| *offset* | The offset for the calculation. |

**Returns**

The address of the block which is *offset* away from *block*.

Definition at line 606 of file heapimpl.h.

#### 8.76.4.10 _Heap_Block_of_alloc_area()

```
RTEMS_INLINE_ROUTINE Heap_Block* _Heap_Block_of_alloc_area (
              uintptr_t alloc_begin,
              uintptr_t page_size )
```

Returns the starting address of the block corresponding to the allocatable area.

**Parameters**

| | |
|---|---|
| *alloc_begin* | The starting address of the allocatable area. |
| *page_size* | The page size for the calculation. |

**Returns**

The Starting address of the corresponding block of the allocatable area.

Definition at line 650 of file heapimpl.h.

#### 8.76.4.11 _Heap_Block_set_size()

```
RTEMS_INLINE_ROUTINE void _Heap_Block_set_size (
              Heap_Block * block,
              uintptr_t size )
```

Sets the block size.

**Parameters**

| | | |
|---|---|---|
| in,out | *block* | The block of which the size shall be set. |
| | *size* | The new size of the block. |

Definition at line 677 of file heapimpl.h.

#### 8.76.4.12 _Heap_Block_size()

```
RTEMS_INLINE_ROUTINE uintptr_t _Heap_Block_size (
              const Heap_Block * block )
```

Returns the block size.

**Parameters**

| | |
|---|---|
| *block* | The block of which the size is requested. |

**Returns**

The block size.

Definition at line 666 of file heapimpl.h.

#### 8.76.4.13 _Heap_Extend()

```
uintptr_t _Heap_Extend (
              Heap_Control * heap,
              void * area_begin,
              uintptr_t area_size,
              uintptr_t unused )
```

Extends the memory available for the heap.

There are no alignment requirements for the memory area. The memory area must be big enough to contain some maintenance blocks. It must not overlap parts of the current heap memory areas. Disconnected memory areas added to the heap will lead to used blocks which cover the gaps. Extending with an inappropriate memory area will corrupt the heap resulting in undefined behaviour.

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap to extend. |
| out | *area_begin* | The start address of the area to extend the *heap* with. |
| | *area_size* | The size of the area in bytes. |
| | *unused* | Is not used, only provided to have the same signature as _Heap_Initialize(). |

**Return values**

| | |
|---|---|
| *some_value* | The extended space available for allocation after successful extension. |
| *0* | The heap extension failed. |

**See also**

Heap_Initialization_or_extend_handler.

#### 8.76.4.14 _Heap_Free()

```
bool _Heap_Free (
            Heap_Control * heap,
            void * addr )
```

Frees the allocated memory area.

Inappropriate values for *addr* may corrupt the heap.

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap of the allocated memory area. |
| | *addr* | The starting address of the memory area to be freed. |

**Return values**

| | |
|---|---|
| *true* | The allocated memory area was successfully freed. |
| *false* | The method failed. |

#### 8.76.4.15 _Heap_Free_list_first()

```
RTEMS_INLINE_ROUTINE Heap_Block* _Heap_Free_list_first (
            Heap_Control * heap )
```

Returns the first block of the free list of the heap.

**Parameters**

| | |
|---|---|
| *heap* | The heap to operate upon. |

**Returns**

The first block of the free list.

Definition at line 475 of file heapimpl.h.

#### 8.76.4.16 _Heap_Free_list_head()

```
RTEMS_INLINE_ROUTINE Heap_Block* _Heap_Free_list_head (
              Heap_Control * heap )
```

Returns the head of the free list of the heap.

**Parameters**

| | |
|---|---|
| *heap* | The heap to operate upon. |

**Returns**

The head of the free list.

Definition at line 451 of file heapimpl.h.

#### 8.76.4.17 _Heap_Free_list_insert_after()

```
RTEMS_INLINE_ROUTINE void _Heap_Free_list_insert_after (
              Heap_Block * block_before,
              Heap_Block * new_block )
```

Inserts a block after an existing block in the free list.

**Parameters**

| | |
|---|---|
| *block_before* | The block that is already in the free list. |
| *new_block* | The block to be inserted after *block_before*. |

Definition at line 533 of file heapimpl.h.

#### 8.76.4.18 _Heap_Free_list_insert_before()

```
RTEMS_INLINE_ROUTINE void _Heap_Free_list_insert_before (
              Heap_Block * block_next,
              Heap_Block * new_block )
```

Inserts a block before an existing block in the free list.

**Parameters**

| | |
|---|---|
| *block_before* | The block that is already in the free list. |
| *new_block* | The block to be inserted before *block_before*. |

Definition at line 552 of file heapimpl.h.

#### 8.76.4.19 _Heap_Free_list_last()

```
RTEMS_INLINE_ROUTINE Heap_Block* _Heap_Free_list_last (
            Heap_Control * heap )
```

Returns the last block of the free list of the heap.

**Parameters**

| | |
|---|---|
| *heap* | The heap to operate upon. |

**Returns**

The last block of the free list.

Definition at line 487 of file heapimpl.h.

#### 8.76.4.20 _Heap_Free_list_remove()

```
RTEMS_INLINE_ROUTINE void _Heap_Free_list_remove (
            Heap_Block * block )
```

Removes the block from the free list.

**Parameters**

| | |
|---|---|
| *block* | The block to be removed. |

Definition at line 497 of file heapimpl.h.

#### 8.76.4.21 _Heap_Free_list_replace()

```
RTEMS_INLINE_ROUTINE void _Heap_Free_list_replace (
            Heap_Block * old_block,
            Heap_Block * new_block )
```

Replaces one block in the free list by another.

**Parameters**

| | |
|---|---|
| *old_block* | The block in the free list to replace. |
| *new_block* | The block that should replace *old_block*. |

Definition at line 512 of file heapimpl.h.

#### 8.76.4.22 _Heap_Free_list_tail()

```
RTEMS_INLINE_ROUTINE Heap_Block* _Heap_Free_list_tail (
            Heap_Control * heap )
```

Returns the tail of the free list of the heap.

**Parameters**

| | |
|---|---|
| *heap* | The heap to operate upon. |

**Returns**

The tail of the free list.

Definition at line 463 of file heapimpl.h.

#### 8.76.4.23 _Heap_Get_first_and_last_block()

```
bool _Heap_Get_first_and_last_block (
            uintptr_t heap_area_begin,
            uintptr_t heap_area_size,
            uintptr_t page_size,
            uintptr_t min_block_size,
            Heap_Block ** first_block_ptr,
            Heap_Block ** last_block_ptr )
```

Gets the first and last block for the heap area.

Nothing will be written to this area.

**Parameters**

| | | |
|---|---|---|
| | *heap_area_begin* | The starting address of the heap area. |
| | *heap_area_size* | The size of the heap area. |
| | *page_size* | The page size for the calculation. |
| | *min_block_size* | The minimal block size for the calculation. |
| out | *first_block_ptr* | The pointer to the first block in the case of success |
| out | *last_block_ptr* | The pointer to the last block in the case of success |

**Return values**

| | |
|---|---|
| *true* | The area is big enough. |
| *false* | The area is not big enough. |

Definition at line 170 of file heap.c.

#### 8.76.4.24 _Heap_Get_free_information()

```
void _Heap_Get_free_information (
              Heap_Control * heap,
              Heap_Information * info )
```

Returns information about free blocks for the heap.

**Parameters**

| | | |
|---|---|---|
| | *heap* | The heap to get the information from. |
| out | *info* | Stores the information about free blocks of *heap* after the method call. |

#### 8.76.4.25 _Heap_Get_information()

```
void _Heap_Get_information (
              Heap_Control * heap,
              Heap_Information_block * info )
```

Returns information about used and free blocks for the heap.

**Parameters**

| | | |
|---|---|---|
| | *heap* | The heap to get the information from. |
| out | *info* | Stores the information of the *heap* after the method call. |

#### 8.76.4.26 _Heap_Get_size()

```
RTEMS_INLINE_ROUTINE uintptr_t _Heap_Get_size (
              const Heap_Control * heap )
```

Returns the size of the allocatable area in bytes.

This value is an integral multiple of the page size.

**Parameters**

| | |
|---|---|
| *heap* | The heap to get the allocatable area from. |

**Returns**

The size of the allocatable area in *heap* in bytes.

Definition at line 782 of file heapimpl.h.

### 8.76.4.27 _Heap_Greedy_allocate()

```
Heap_Block* _Heap_Greedy_allocate (
              Heap_Control * heap,
              const uintptr_t * block_sizes,
              size_t block_count )
```

Greedily allocates and empties the heap.

Afterwards, the heap has at most *block_count* allocatable blocks of sizes specified by *block_sizes*. All other blocks are used.

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap to operate upon |
| | *block_sizes* | The sizes of the allocatable blocks. Must point to an array with *block_count* members. |
| | *block_count* | The maximum number of allocatable blocks of sizes specified by @block_sizes. |

**Returns**

Pointer to the first allocated block.

**See also**

_Heap_Greedy_free().

### 8.76.4.28 _Heap_Greedy_allocate_all_except_largest()

```
Heap_Block* _Heap_Greedy_allocate_all_except_largest (
              Heap_Control * heap,
              uintptr_t * allocatable_size )
```

Greedily allocates all blocks except the largest free block.

Afterwards the heap has at most one allocatable block. This block is the largest free block if it exists. All other blocks are used.

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap to operate upon. |
| out | *allocatable_size* | Stores the size of the largest free block of the heap after the method call. |

**Returns**

Pointer to the first allocated block.

**See also**

_Heap_Greedy_free().

### 8.76.4.29 _Heap_Greedy_free()

```
void _Heap_Greedy_free (
            Heap_Control * heap,
            Heap_Block * blocks )
```

Frees blocks of a greedy allocation.

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap to operate upon. |
| | *blocks* | Must be the return value of _Heap_Greedy_allocate(). |

### 8.76.4.30 _Heap_Initialize()

```
uintptr_t _Heap_Initialize (
            Heap_Control * heap,
            void * area_begin,
            uintptr_t area_size,
            uintptr_t page_size )
```

Initializes the heap control block.

Blocks of memory are allocated from the heap in multiples of *page_size* byte units. If the *page_size* is equal to zero or is not multiple of CPU_ALIGNMENT, it is aligned up to the nearest CPU_ALIGNMENT boundary.

**Parameters**

| | | |
|---|---|---|
| out | *heap* | The heap control block to manage the area. |
| | *area_begin* | The starting address of the area. |
| | *area_size* | The size of the area in bytes. |
| | *page_size* | The page size for the calculation |

**Return values**

| | |
|---|---|
| *some_value* | The maximum memory available. |
| *0* | The initialization failed. |

**See also**

Heap_Initialization_or_extend_handler.

Definition at line 207 of file heap.c.

#### 8.76.4.31 _Heap_Is_aligned()

```
RTEMS_INLINE_ROUTINE bool _Heap_Is_aligned (
              uintptr_t value,
              uintptr_t alignment )
```

Checks if the value is aligned to the given alignment.

**Parameters**

| | |
|---|---|
| *value* | The value to check the alignment of. |
| *alignment* | The alignment for the operation. |

**Return values**

| | |
|---|---|
| *true* | The value is aligned to the given alignment. |
| *false* | The value is not aligned to the given alignment. |

Definition at line 574 of file heapimpl.h.

#### 8.76.4.32 _Heap_Is_block_in_heap()

```
RTEMS_INLINE_ROUTINE bool _Heap_Is_block_in_heap (
              const Heap_Control * heap,
              const Heap_Block * block )
```

Returns if the block is part of the heap.

**Parameters**

| | |
|---|---|
| *heap* | The heap to test if the *block* is part of it. |
| *block* | The block of which the information is requested. |

**Return values**

| | |
|---|---|
| *true* | The block is part of the heap. |
| *false* | The block is not part of the heap. |

Definition at line 743 of file heapimpl.h.

#### 8.76.4.33 _Heap_Is_free()

```
RTEMS_INLINE_ROUTINE bool _Heap_Is_free (
            const Heap_Block * block )
```

Returns if the heap block is free.

**Parameters**

| | |
|---|---|
| *block* | The block of which the information is requested. |

**Return values**

| | |
|---|---|
| *true* | The block is free. |
| *false* | The block is not free. |

Definition at line 727 of file heapimpl.h.

#### 8.76.4.34 _Heap_Is_prev_used()

```
RTEMS_INLINE_ROUTINE bool _Heap_Is_prev_used (
            const Heap_Block * block )
```

Returns if the previous heap block is used.

**Parameters**

| | |
|---|---|
| *block* | The block of which the information about the previous block is requested. |

**Return values**

| | |
|---|---|
| *true* | The previous block is used. |
| *false* | The previous block is not used. |

Definition at line 696 of file heapimpl.h.

#### 8.76.4.35 _Heap_Is_used()

```
RTEMS_INLINE_ROUTINE bool _Heap_Is_used (
              const Heap_Block * block )
```

Returns if the heap block is used.

**Parameters**

| | |
|---|---|
| *block* | The block of which the information is requested. |

**Return values**

| | |
|---|---|
| *true* | The block is used. |
| *false* | The block is not used. |

Definition at line 709 of file heapimpl.h.

#### 8.76.4.36 _Heap_Iterate()

```
void _Heap_Iterate (
              Heap_Control * heap,
              Heap_Block_visitor visitor,
              void * visitor_arg )
```

Iterates over all blocks of the heap.

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap to iterate over. |
| | *visitor* | This will be called for each heap block with the argument *visitor_arg*. |
| in,out | *visitor_arg* | The argument for all calls of *visitor*. |

#### 8.76.4.37 _Heap_Max()

```
RTEMS_INLINE_ROUTINE uintptr_t _Heap_Max (
              uintptr_t a,
              uintptr_t b )
```

Returns the bigger one of the two arguments.

**Parameters**

| | |
|---|---|
| *a* | The parameter on the left hand side of the comparison. |
| *b* | The parameter on the right hand side of the comparison. |

**Return values**

| | |
|---|---|
| *a* | If a > b. |
| *b* | If b >= a |

Definition at line 796 of file heapimpl.h.

#### 8.76.4.38 _Heap_Min()

```
RTEMS_INLINE_ROUTINE uintptr_t _Heap_Min (
              uintptr_t a,
              uintptr_t b )
```

Returns the smaller one of the two arguments.

**Parameters**

| | |
|---|---|
| *a* | The parameter on the left hand side of the comparison. |
| *b* | The parameter on the right hand side of the comparison. |

**Return values**

| | |
|---|---|
| *a* | If a < b. |
| *b* | If b <= a |

Definition at line 810 of file heapimpl.h.

#### 8.76.4.39 _Heap_Min_block_size()

```
RTEMS_INLINE_ROUTINE uintptr_t _Heap_Min_block_size (
              uintptr_t page_size )
```

Returns the minimal Heap Block size for the given page_size.

**Parameters**

| | |
|---|---|
| *page_size* | The page size for the heap. |

**Returns**

The minimal Heap Block size for the given *page_size*.

Definition at line 434 of file heap.h.

#### 8.76.4.40 _Heap_No_extend()

```
uintptr_t _Heap_No_extend (
              Heap_Control * unused_0,
              void * unused_1,
              uintptr_t unused_2,
              uintptr_t unused_3 )
```

This function returns always zero.

This function only returns zero and does nothing else.

**Parameters**

| | |
|---|---|
| *unused_0* | This parameter does nothing. |
| *unused_1* | This parameter does nothing. |
| *unused_2* | This parameter does nothing. |
| *unused_3* | This parameter does nothing. |

**See also**

Heap_Initialization_or_extend_handler.

#### 8.76.4.41 _Heap_Prev_block()

```
RTEMS_INLINE_ROUTINE Heap_Block* _Heap_Prev_block (
              const Heap_Block * block )
```

Returns the address of the previous block.

**Parameters**

| | |
|---|---|
| *block* | The block of which the address of the previous block is requested. |

**Returns**

The address of the previous block.

Definition at line 621 of file heapimpl.h.

#### 8.76.4.42 _Heap_Protection_set_delayed_free_fraction()

```
RTEMS_INLINE_ROUTINE void _Heap_Protection_set_delayed_free_fraction (
              Heap_Control * heap,
              uintptr_t fraction )
```

Sets the fraction of delayed free blocks that is actually freed during memory shortage.

The default is to free half the delayed free blocks. This is equal to a fraction value of two.

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap control. |
| | *fraction* | The fraction is one divided by this fraction value. |

Definition at line 431 of file heapimpl.h.

#### 8.76.4.43 _Heap_Resize_block()

```
Heap_Resize_status _Heap_Resize_block (
              Heap_Control * heap,
              void * addr,
              uintptr_t size,
              uintptr_t * old_size,
              uintptr_t * new_size )
```

Resizes the block of the allocated memory area.

Inappropriate values for *addr* may corrupt the heap.

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap to operate upon. |
| | *addr* | The starting address of the allocated memory area to be resized. |
| | *size* | The least possible size for the new memory area. Resize may be impossible and depends on the current heap usage. |
| out | *old_size* | Stores the size available for allocation in the current block before the resize after the method call. |
| out | *new_size* | Stores the size available for allocation in the resized block after the method call. In the case of an unsuccessful resize, zero is returned in this parameter |

**Return values**

| | |
|---|---|
| *HEAP_RESIZE_SUCCESSFUL* | The resize was successful. |
| *HEAP_RESIZE_UNSATISFIED* | The least possible size *size* was too big. Resize not possible. |
| *HEAP_RESIZE_FATAL_ERROR* | The block starting at *addr* is not part of the heap. |

#### 8.76.4.44 _Heap_Set_last_block_size()

```
RTEMS_INLINE_ROUTINE void _Heap_Set_last_block_size (
              Heap_Control * heap )
```

Sets the size of the last block for the heap.

The next block of the last block will be the first block. Since the first block indicates that the previous block is used, this ensures that the last block appears as used for the _Heap_Is_used() and _Heap_Is_free() functions.

This feature will be used to terminate the scattered heap area list. See also _Heap_Extend().

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap to set the last block size of. |

Definition at line 765 of file heapimpl.h.

#### 8.76.4.45 _Heap_Size_of_alloc_area()

```
bool _Heap_Size_of_alloc_area (
              Heap_Control * heap,
              void * addr,
              uintptr_t * size )
```

Returns the size of the allocatable memory area.

The size value may be greater than the initially requested size in _Heap_Allocate_aligned_with_boundary().

Inappropriate values for *addr* will not corrupt the heap, but may yield invalid size values.

**Parameters**

| | | |
|---|---|---|
| | *heap* | The heap to operate upon. |
| | *addr* | The starting address of the allocatable memory area. |
| out | *size* | Stores the size of the allocatable memory area after the method call. |

**Return values**

| | |
|---|---|
| *true* | The operation was successful. |
| *false* | The operation was not successful. |

#### 8.76.4.46 _Heap_Size_with_overhead()

```
RTEMS_INLINE_ROUTINE uintptr_t _Heap_Size_with_overhead (
              uintptr_t page_size,
```

```
            uintptr_t size,
            uintptr_t alignment )
```

Returns the size with administration and alignment overhead for one allocation.

**Parameters**

| | |
|---|---|
| *page_size* | The page size for the allocation. |
| *size* | The size to allocate. |
| *alignment* | The alignment that needs to be considered. |

**Returns**

The size with administration and alignment overhead for one allocation.

Definition at line 469 of file heap.h.

#### 8.76.4.47 _Heap_Walk()

```
bool _Heap_Walk (
            Heap_Control * heap,
            int source,
            bool dump )
```

Verifies the integrity of the heap.

Walks the heap to verify its integrity.

**Parameters**

| | |
|---|---|
| *heap* | The heap whose integrity is to be verified. |
| *source* | If *dump* is `true`, this is used to mark the output lines. |
| *dump* | Indicates whether diagnostic messages will be printed to standard output. |

**Return values**

| | |
|---|---|
| *true* | No errors occured, the heap´s integrity is not violated. |
| *false* | The heap is corrupt. |

# 8.77 Helpers

Timespec Helpers.

## Files

- file timespec.h

  *Contains Helpers for Manipulating Timespecs.*

## Macros

- #define _Timespec_Set(_time, _seconds, _nanoseconds)

  *Set timespec to seconds nanosecond.*
- #define _Timespec_Set_to_zero(_time)

  *Sets the Timespec to Zero.*
- #define _Timespec_Get_seconds(_time) ((_time)->tv_sec)

  *Get seconds portion of timespec.*
- #define _Timespec_Get_nanoseconds(_time) ((_time)->tv_nsec)

  *Get nanoseconds portion of timespec.*
- #define _Timespec_Greater_than(_lhs, _rhs) _Timespec_Less_than( _rhs, _lhs )

  *The Timespec "greater than" operator.*
- #define _Timespec_Equal_to(lhs, rhs)

  *The Timespec "equal to" operator.*

## Functions

- uint64_t _Timespec_Get_as_nanoseconds (const struct timespec *time)

  *Gets the timestamp as nanoseconds.*
- bool _Timespec_Is_valid (const struct timespec *time)

  *Checks if timespec is valid.*
- bool _Timespec_Less_than (const struct timespec *lhs, const struct timespec *rhs)

  *Checks if the left hand side timespec is less than the right one.*
- uint32_t _Timespec_Add_to (struct timespec *time, const struct timespec *add)

  *Adds two timespecs.*
- uint32_t _Timespec_To_ticks (const struct timespec *time)

  *Converts timespec to number of ticks.*
- void _Timespec_From_ticks (uint32_t ticks, struct timespec *time)

  *Converts ticks to timespec.*
- void _Timespec_Subtract (const struct timespec *start, const struct timespec *end, struct timespec *result)

  *Subtracts two timespec.*
- void _Timespec_Divide_by_integer (const struct timespec *time, uint32_t iterations, struct timespec *result)

  *Divides timespec by an integer.*
- void _Timespec_Divide (const struct timespec *lhs, const struct timespec *rhs, uint32_t *ival_percentage, uint32_t *fval_percentage)

  *Divides a timespec by another timespec.*

### 8.77.1 Detailed Description

Timespec Helpers.

This handler encapsulates functionality related to manipulating POSIX struct timespecs.

### 8.77.2 Macro Definition Documentation

#### 8.77.2.1 _Timespec_Equal_to

```
#define _Timespec_Equal_to(
              lhs,
              rhs )
```

**Value:**

```
  ( ((lhs)->tv_sec  == (rhs)->tv_sec) &&   \
    ((lhs)->tv_nsec == (rhs)->tv_nsec)     \
  )
```

The Timespec "equal to" operator.

This method is the is equal to than operator for timespecs.

**Parameters**

| | | |
|---|---|---|
| in | *lhs* | is the left hand side timespec |
| in | *rhs* | is the right hand side timespec |

**Return values**

| | |
|---|---|
| *This* | method returns true if *lhs* is equal to *rhs* and false otherwise. |

Definition at line 164 of file timespec.h.

#### 8.77.2.2 _Timespec_Get_nanoseconds

```
#define _Timespec_Get_nanoseconds(
              _time )  ((_time)->tv_nsec)
```

Get nanoseconds portion of timespec.

This method returns the nanoseconds portion of the specified timespec

**Parameters**

| | | |
|---|---|---|
| in | *_time* | points to the timespec |

**Return values**

| | |
|---|---|
| *The* | nanoseconds portion of *_time*. |

Definition at line 93 of file timespec.h.

#### 8.77.2.3 _Timespec_Get_seconds

```
#define _Timespec_Get_seconds(
              _time )  ((_time)->tv_sec)
```

Get seconds portion of timespec.

This method returns the seconds portion of the specified timespec

**Parameters**

| | | |
|---|---|---|
| in | *_time* | points to the timespec |

**Return values**

| | |
|---|---|
| *The* | seconds portion of *_time*. |

Definition at line 81 of file timespec.h.

#### 8.77.2.4 _Timespec_Greater_than

```
#define _Timespec_Greater_than(
              _lhs,
              _rhs )  _Timespec_Less_than( _rhs, _lhs )
```

The Timespec "greater than" operator.

This method is the greater than operator for timespecs.

**Parameters**

| | | |
|---|---|---|
| in | *_lhs* | is the left hand side timespec |
| in | *_rhs* | is the right hand side timespec |

**Return values**

| | |
|---|---|
| *This* | method returns true if *lhs* is greater than the *rhs* and false otherwise. |

Definition at line 150 of file timespec.h.

#### 8.77.2.5 _Timespec_Set

```
#define _Timespec_Set(
              _time,
              _seconds,
              _nanoseconds )
```

**Value:**

```
do { \
   (_time)->tv_sec = (_seconds); \
   (_time)->tv_nsec = (_nanoseconds); \
} while (0)
```

Set timespec to seconds nanosecond.

This method sets the timespec to the specified seconds and nanoseconds value.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | points to the timespec instance to validate. |
| in | *_seconds* | is the seconds portion of the timespec |
| in | *_nanoseconds* | is the nanoseconds portion of the timespec |

Definition at line 52 of file timespec.h.

#### 8.77.2.6 _Timespec_Set_to_zero

```
#define _Timespec_Set_to_zero(
              _time )
```

**Value:**

```
do { \
   (_time)->tv_sec = 0; \
   (_time)->tv_nsec = 0; \
} while (0)
```

Sets the Timespec to Zero.

This method sets the timespec to zero. value.

**Parameters**

| | | |
|---|---|---|
| in | *_time* | points to the timespec instance to zero. |

Definition at line 66 of file timespec.h.

## 8.77.3 Function Documentation

### 8.77.3.1 _Timespec_Add_to()

```
uint32_t _Timespec_Add_to (
             struct timespec * time,
             const struct timespec * add )
```

Adds two timespecs.

This routine adds two timespecs. The second argument is added to the first.

**Parameters**

| | |
|---|---|
| *time* | The base time to be added to. |
| *add* | The timespec to add to the first argument. |

**Returns**

The number of seconds *time* increased by.

### 8.77.3.2 _Timespec_Divide()

```
void _Timespec_Divide (
             const struct timespec * lhs,
             const struct timespec * rhs,
             uint32_t * ival_percentage,
             uint32_t * fval_percentage )
```

Divides a timespec by another timespec.

This routine divides a timespec by another timespec. The intended use is for calculating percentages to three decimal points.

**Parameters**

| | | |
|---|---|---|
| | *lhs* | The left hand timespec. |
| | *rhs* | The right hand timespec. |
| out | *ival_percentage* | The integer portion of the average. |
| out | *fval_percentage* | The thousandths of percentage. |

#### 8.77.3.3 _Timespec_Divide_by_integer()

```
void _Timespec_Divide_by_integer (
            const struct timespec * time,
            uint32_t iterations,
            struct timespec * result )
```

Divides timespec by an integer.

This routine divides a timespec by an integer value. The expected use is to assist in benchmark calculations where you typically divide a duration by a number of iterations.

**Parameters**

| | | |
|---|---|---|
| | *time* | The total. |
| | *iterations* | The number of iterations. |
| out | *result* | The average time. |

#### 8.77.3.4 _Timespec_From_ticks()

```
void _Timespec_From_ticks (
            uint32_t ticks,
            struct timespec * time )
```

Converts ticks to timespec.

This routine converts the *ticks* value to the corresponding timespec format *time*.

**Parameters**

| | | |
|---|---|---|
| | *ticks* | The number of ticks to convert. |
| out | *time* | The timespec format time result. |

#### 8.77.3.5 _Timespec_Get_as_nanoseconds()

```
uint64_t _Timespec_Get_as_nanoseconds (
            const struct timespec * time )
```

Gets the timestamp as nanoseconds.

This method returns the timestamp as nanoseconds.

**Parameters**

| | |
|---|---|
| *time* | points to the timestamp. |

**Returns**

The time in nanoseconds.

#### 8.77.3.6 _Timespec_Is_valid()

```
bool _Timespec_Is_valid (
             const struct timespec * time )
```

Checks if timespec is valid.

This method determines the validity of a timespec.

**Parameters**

| | |
|---|---|
| *time* | is the timespec instance to validate. |

**Return values**

| | |
|---|---|
| *true* | The timespec is valid. |
| *false* | The timespec is invalid. |

#### 8.77.3.7 _Timespec_Less_than()

```
bool _Timespec_Less_than (
             const struct timespec * lhs,
             const struct timespec * rhs )
```

Checks if the left hand side timespec is less than the right one.

This method is the less than operator for timespecs.

**Parameters**

| | |
|---|---|
| *lhs* | is the left hand side timespec. |
| *rhs* | is the right hand side timespec. |

**Return values**

| | |
|---|---|
| *true* | *lhs* is less than *rhs*. |
| *false* | *lhs* is greater than *rhs*. |

#### 8.77.3.8 _Timespec_Subtract()

```
void _Timespec_Subtract (
             const struct timespec * start,
             const struct timespec * end,
             struct timespec * result )
```

Subtracts two timespec.

This routine subtracts two timespecs. *result* is set to *end* - *start*.

**Parameters**

| | | |
|---|---|---|
| | *start* | The starting time |
| | *end* | The ending time |
| out | *result* | The difference between starting and ending time. |

#### 8.77.3.9 _Timespec_To_ticks()

```
uint32_t _Timespec_To_ticks (
             const struct timespec * time )
```

Converts timespec to number of ticks.

This routine convert the *time* timespec to the corresponding number of clock ticks.

**Parameters**

| | |
|---|---|
| *time* | The time to be converted. |

**Returns**

The number of ticks computed.

## 8.78 I/O Manager

The Input/Output (I/O) Manager provides a well-defined mechanism for accessing device drivers and a structured methodology for organizing device drivers.

### Classes

- struct rtems_driver_address_table

  *This structure contains the device driver entries.*

### Typedefs

- typedef rtems_status_code rtems_device_driver

  *This type shall be used in device driver entry declarations and definitions.*
- typedef uint32_t rtems_device_major_number

  *This integer type represents the major number of devices.*
- typedef uint32_t rtems_device_minor_number

  *This integer type represents the minor number of devices.*
- typedef rtems_device_driver(∗ rtems_device_driver_entry) (rtems_device_major_number, rtems_device_minor_number, void ∗)

  *Device driver entries shall have this type.*

### Functions

- rtems_status_code rtems_io_register_driver (rtems_device_major_number major, const rtems_driver_address_table ∗driver_table, rtems_device_major_number ∗registered_major)

  *Registers and initializes the device with the specified device driver address table and device major number in the Device Driver Table.*
- rtems_status_code rtems_io_unregister_driver (rtems_device_major_number major)

  *Removes a device driver specified by the device major number from the Device Driver Table.*
- rtems_status_code rtems_io_initialize (rtems_device_major_number major, rtems_device_minor_number minor, void ∗argument)

  *Initializes the device specified by the device major and minor numbers.*
- rtems_status_code rtems_io_register_name (const char ∗device_name, rtems_device_major_number major, rtems_device_minor_number minor)

  *Registers the device specified by the device major and minor numbers in the file system under the specified name.*
- rtems_status_code rtems_io_open (rtems_device_major_number major, rtems_device_minor_number minor, void ∗argument)

  *Opens the device specified by the device major and minor numbers.*
- rtems_status_code rtems_io_close (rtems_device_major_number major, rtems_device_minor_number minor, void ∗argument)

  *Closes the device specified by the device major and minor numbers.*
- rtems_status_code rtems_io_read (rtems_device_major_number major, rtems_device_minor_number minor, void ∗argument)

  *Reads from the device specified by the device major and minor numbers.*
- rtems_status_code rtems_io_write (rtems_device_major_number major, rtems_device_minor_number minor, void ∗argument)

  *Writes to the device specified by the device major and minor numbers.*
- rtems_status_code rtems_io_control (rtems_device_major_number major, rtems_device_minor_number minor, void ∗argument)

  *Controls the device specified by the device major and minor numbers.*

### 8.78.1 Detailed Description

The Input/Output (I/O) Manager provides a well-defined mechanism for accessing device drivers and a structured methodology for organizing device drivers.

### 8.78.2 Typedef Documentation

#### 8.78.2.1 rtems_device_driver

```
typedef rtems_status_code rtems_device_driver
```

This type shall be used in device driver entry declarations and definitions.

Device driver entries return an rtems_status_code status code. This type definition helps to document device driver entries in the source code.

Definition at line 84 of file io.h.

#### 8.78.2.2 rtems_device_major_number

```
typedef uint32_t rtems_device_major_number
```

This integer type represents the major number of devices.

The major number of a device is determined by rtems_io_register_driver() and the application configuration (see CONFIGURE_MAXIMUM_DRIVERS) .

Definition at line 96 of file io.h.

#### 8.78.2.3 rtems_device_minor_number

```
typedef uint32_t rtems_device_minor_number
```

This integer type represents the minor number of devices.

The minor number of devices is managed by the device driver.

Definition at line 107 of file io.h.

### 8.78.3 Function Documentation

#### 8.78.3.1 rtems_io_close()

```
rtems_status_code rtems_io_close (
            rtems_device_major_number major,
            rtems_device_minor_number minor,
            void * argument )
```

Closes the device specified by the device major and minor numbers.

This directive calls the device driver close entry registered in the Device Driver Table for the specified device major number.

The close entry point is commonly used by device drivers to relinquish exclusive access to a device.

**Parameters**

| | |
|---|---|
| *major* | is the major number of the device. |
| *minor* | is the minor number of the device. |
| *argument* | is the argument passed to the device driver close entry. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_NUMBER* | The device major number was invalid. |

**Returns**

Other status codes may be returned by the device driver close entry.

### 8.78.3.2 rtems_io_control()

```
rtems_status_code rtems_io_control (
            rtems_device_major_number major,
            rtems_device_minor_number minor,
            void * argument )
```

Controls the device specified by the device major and minor numbers.

This directive calls the device driver I/O control entry registered in the Device Driver Table for the specified device major number.

The exact functionality of the driver entry called by this directive is driver dependent. It should not be assumed that the control entries of two device drivers are compatible. For example, an RS-232 driver I/O control operation may change the baud of a serial line, while an I/O control operation for a floppy disk driver may cause a seek operation.

**Parameters**

| | |
|---|---|
| *major* | is the major number of the device. |
| *minor* | is the minor number of the device. |
| *argument* | is the argument passed to the device driver I/O control entry. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_NUMBER* | The device major number was invalid. |

**Returns**

Other status codes may be returned by the device driver I/O control entry.

#### 8.78.3.3 rtems_io_initialize()

```
rtems_status_code rtems_io_initialize (
            rtems_device_major_number major,
            rtems_device_minor_number minor,
            void * argument )
```

Initializes the device specified by the device major and minor numbers.

This directive calls the device driver initialization entry registered in the Device Driver Table for the specified device major number.

This directive is automatically invoked for each device driver defined by the application configuration during the system initialization and via the rtems_io_register_driver() directive.

A device driver initialization entry is responsible for initializing all hardware and data structures associated with a device. If necessary, it can allocate memory to be used during other operations.

**Parameters**

| | |
|---|---|
| *major* | is the major number of the device. |
| *minor* | is the minor number of the device. |
| *argument* | is the argument passed to the device driver initialization entry. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_NUMBER* | The device major number was invalid. |

**Returns**

Other status codes may be returned by the device driver initialization entry.

#### 8.78.3.4 rtems_io_open()

```
rtems_status_code rtems_io_open (
            rtems_device_major_number major,
            rtems_device_minor_number minor,
            void * argument )
```

Opens the device specified by the device major and minor numbers.

This directive calls the device driver open entry registered in the Device Driver Table for the specified device major number.

The open entry point is commonly used by device drivers to provide exclusive access to a device.

**Parameters**

| | |
|---|---|
| *major* | is the major number of the device. |
| *minor* | is the minor number of the device. |
| *argument* | is the argument passed to the device driver close entry. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_NUMBER* | The device major number was invalid. |

**Returns**

Other status codes may be returned by the device driver open entry.

### 8.78.3.5 rtems_io_read()

```
rtems_status_code rtems_io_read (
            rtems_device_major_number major,
            rtems_device_minor_number minor,
            void * argument )
```

Reads from the device specified by the device major and minor numbers.

This directive calls the device driver read entry registered in the Device Driver Table for the specified device major number.

Read operations typically require a buffer address as part of the argument parameter block. The contents of this buffer will be replaced with data from the device.

**Parameters**

| | |
|---|---|
| *major* | is the major number of the device. |
| *minor* | is the minor number of the device. |
| *argument* | is the argument passed to the device driver read entry. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_NUMBER* | The device major number was invalid. |

**Returns**

Other status codes may be returned by the device driver read entry.

### 8.78.3.6 rtems_io_register_driver()

```
rtems_status_code rtems_io_register_driver (
            rtems_device_major_number major,
```

```
const rtems_driver_address_table * driver_table,
rtems_device_major_number * registered_major )
```

Registers and initializes the device with the specified device driver address table and device major number in the Device Driver Table.

If the device major number equals zero a device major number will be obtained. The device major number of the registered driver will be returned.

After a successful registration, the rtems_io_initialize() directive will be called to initialize the device.

**Parameters**

| | | |
|---|---|---|
| | *major* | is the device major number. Use a value of zero to let the system obtain a device major number automatically. |
| | *driver_table* | is the device driver address table. |
| out | *registered_major* | is the pointer to a device major number variable. The device major number of the registered device will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The device major number of the device was NULL. |
| *RTEMS_INVALID_ADDRESS* | The device driver address table was empty. |
| *RTEMS_INVALID_NUMBER* | The device major number of the device was out of range, see CONFIGURE_MAXIMUM_DRIVERS. |
| *RTEMS_TOO_MANY* | The system was unable to obtain a device major number. |
| *RTEMS_RESOURCE_IN_USE* | The device major number was already in use. |
| *RTEMS_CALLED_FROM_ISR* | The directive was called from interrupt context. |

**Returns**

Other status codes may be returned by rtems_io_initialize().

### 8.78.3.7 rtems_io_register_name()

```
rtems_status_code rtems_io_register_name (
    const char * device_name,
    rtems_device_major_number major,
    rtems_device_minor_number minor )
```

Registers the device specified by the device major and minor numbers in the file system under the specified name.

The device is registered as a character device.

**Parameters**

| | |
|---|---|
| *device_name* | is the device name in the file system. |
| *major* | is the device major number. |
| *minor* | is the device minor number. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_TOO_MANY* | The name was already in use or other errors occurred. |

#### 8.78.3.8 rtems_io_unregister_driver()

```
rtems_status_code rtems_io_unregister_driver (
            rtems_device_major_number major )
```

Removes a device driver specified by the device major number from the Device Driver Table.

Currently no specific checks are made and the driver is not closed.

**Parameters**

| | |
|---|---|
| *major* | is the major number of the device. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_UNSATISFIED* | The device major number was invalid. |
| *RTEMS_CALLED_FROM_ISR* | The directive was called from interrupt context. |

#### 8.78.3.9 rtems_io_write()

```
rtems_status_code rtems_io_write (
            rtems_device_major_number major,
            rtems_device_minor_number minor,
            void * argument )
```

Writes to the device specified by the device major and minor numbers.

This directive calls the device driver write entry registered in the Device Driver Table for the specified device major number.

Write operations typically require a buffer address as part of the argument parameter block. The contents of this buffer will be sent to the device.

**Parameters**

| | |
|---|---|
| *major* | is the major number of the device. |
| *minor* | is the minor number of the device. |
| *argument* | is the argument passed to the device driver write entry. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_NUMBER* | The device major number was invalid. |

**Returns**

Other status codes may be returned by the device driver write entry.

# 8.79 IO

IO.

## Modules

- Print Support

  *Print Support.*

### 8.79.1 Detailed Description

IO.

## 8.80 IO Library

Provides system call and file system interface definitions.

### Modules

- File System Node Handler

  *File system node handler.*
- File System Operations

  *File system operations.*
- File System Types and Mount

  *File system types and mount.*
- Termios

  *Termios.*

### Files

- file libio.h

  *Basic IO API.*

### Classes

- struct rtems_filesystem_location_info_tt

  *File system location.*
- struct rtems_filesystem_global_location_t

  *Global file system location.*
- struct rtems_filesystem_limits_and_options_t

  *Contain file system specific information which is required to support fpathconf().*
- struct rtems_libio_tt

  *An open file data structure.*
- struct rtems_libio_rw_args_t

  *Paramameter block for read/write.*
- struct rtems_libio_open_close_args_t

  *Parameter block for open/close.*
- struct rtems_libio_ioctl_args_t

  *Parameter block for ioctl.*
- union __rtems_dev_t

### Macros

- #define **rtems_filesystem_split_dev_t**(_dev, _major, _minor)

### Typedefs

- typedef struct rtems_filesystem_location_info_tt rtems_filesystem_location_info_t

  *File system location.*
- typedef struct rtems_filesystem_global_location_t rtems_filesystem_global_location_t

  *Global file system location.*
- typedef off_t **rtems_off64_t**

## Functions

- rtems_filesystem_fsmount_me_t rtems_filesystem_get_mount_handler (const char *type)

  *Gets the mount handler for the file system type.*
- static unsigned int **rtems_libio_iop_flags** (const rtems_libio_t *iop)
- static bool rtems_libio_iop_is_no_delay (const rtems_libio_t *iop)

  *Returns true if this is a no delay iop, otherwise returns false.*
- static bool rtems_libio_iop_is_readable (const rtems_libio_t *iop)

  *Returns true if this is a readable iop, otherwise returns false.*
- static bool rtems_libio_iop_is_writeable (const rtems_libio_t *iop)

  *Returns true if this is a writeable iop, otherwise returns false.*
- static bool rtems_libio_iop_is_append (const rtems_libio_t *iop)

  *Returns true if this is an append iop, otherwise returns false.*
- static dev_t **rtems_filesystem_make_dev_t** (rtems_device_major_number _major, rtems_device_minor_number _minor)
- static dev_t **rtems_filesystem_make_dev_t_from_pointer** (const void *pointer)
- static rtems_device_major_number **rtems_filesystem_dev_major_t** (dev_t device)
- static rtems_device_minor_number **rtems_filesystem_dev_minor_t** (dev_t device)
- void rtems_filesystem_initialize (void)

  *Base File System Initialization.*
- void **rtems_libio_post_driver** (void)
- void **rtems_libio_exit** (void)
- int rtems_mkdir (const char *path, mode_t mode)

  *Creates a directory and all its parent directories according to path.*

## Variables

- const rtems_filesystem_limits_and_options_t rtems_filesystem_default_pathconf

  *Default pathconf settings.*

## Flag Values

- #define **LIBIO_FLAGS_NO_DELAY** 0x0001U /* return immediately if no data */
- #define **LIBIO_FLAGS_READ** 0x0002U /* reading */
- #define **LIBIO_FLAGS_WRITE** 0x0004U /* writing */
- #define **LIBIO_FLAGS_OPEN** 0x0100U /* device is open */
- #define **LIBIO_FLAGS_APPEND** 0x0200U /* all writes append */
- #define **LIBIO_FLAGS_CLOSE_ON_EXEC** 0x0800U /* close on process exec() */
- #define **LIBIO_FLAGS_READ_WRITE** (LIBIO_FLAGS_READ | LIBIO_FLAGS_WRITE)
- #define **LIBIO_FLAGS_REFERENCE_INC** 0x1000U

## External I/O Handlers

- typedef int(* **rtems_libio_open_t**) (const char *pathname, uint32_t flag, uint32_t mode)
- typedef int(* **rtems_libio_close_t**) (int fd)
- typedef ssize_t(* **rtems_libio_read_t**) (int fd, void *buffer, size_t count)
- typedef ssize_t(* **rtems_libio_write_t**) (int fd, const void *buffer, size_t count)
- typedef int(* **rtems_libio_ioctl_t**) (int fd, uint32_t command, void *buffer)
- typedef off_t(* **rtems_libio_lseek_t**) (int fd, off_t offset, int whence)

### Permission Macros

- #define **RTEMS_FS_PERMS_READ** 0x4
- #define **RTEMS_FS_PERMS_WRITE** 0x2
- #define **RTEMS_FS_PERMS_EXEC** 0x1
- #define **RTEMS_FS_PERMS_RWX** (RTEMS_FS_PERMS_READ | RTEMS_FS_PERMS_WRITE | RTE↩MS_FS_PERMS_EXEC)
- #define **RTEMS_FS_FOLLOW_HARD_LINK** 0x8
- #define **RTEMS_FS_FOLLOW_SYM_LINK** 0x10
- #define **RTEMS_FS_FOLLOW_LINK** (RTEMS_FS_FOLLOW_HARD_LINK | RTEMS_FS_FOLLOW_SY↩M_LINK)
- #define **RTEMS_FS_MAKE** 0x20
- #define **RTEMS_FS_EXCLUSIVE** 0x40
- #define **RTEMS_FS_ACCEPT_RESIDUAL_DELIMITERS** 0x80
- #define **RTEMS_FS_REJECT_TERMINAL_DOT** 0x100

## 8.80.1 Detailed Description

Provides system call and file system interface definitions.

General purpose communication channel for RTEMS to allow UNIX/POSIX system call behavior under RTEMS. Initially this supported only IO to devices but has since been enhanced to support networking and support for mounted file systems.

## 8.80.2 Macro Definition Documentation

### 8.80.2.1 rtems_filesystem_split_dev_t

```
#define rtems_filesystem_split_dev_t(
              _dev,
              _major,
              _minor )
```

**Value:**

```
do { \
  (_major) = rtems_filesystem_dev_major_t ( _dev ); \
  (_minor) = rtems_filesystem_dev_minor_t( _dev ); \
} while(0)
```

Definition at line 1542 of file libio.h.

## 8.80.3 Typedef Documentation

#### 8.80.3.1 rtems_filesystem_global_location_t

```
typedef struct rtems_filesystem_global_location_t rtems_filesystem_global_location_t
```

Global file system location.

The global file system locations are used for

- the mount point location in the mount table entry,
- the file system root location in the mount table entry,
- the root directory location in the user environment, and
- the current directory location in the user environment.

During the path evaluation global start locations are obtained to ensure that the current file system will be not unmounted in the meantime.

To support a release within critical sections of the operating system a deferred release is supported. This is similar to malloc() and free().

**See also**

rtems_filesystem_global_location_obtain() and rtems_filesystem_global_location_release().

### 8.80.4 Function Documentation

#### 8.80.4.1 rtems_filesystem_get_mount_handler()

```
rtems_filesystem_fsmount_me_t rtems_filesystem_get_mount_handler (
            const char * type )
```

Gets the mount handler for the file system *type*.

**Returns**

The file system mount handler associated with the *type*, or `NULL` if no such association exists.

#### 8.80.4.2 rtems_filesystem_initialize()

```
void rtems_filesystem_initialize (
            void  )
```

Base File System Initialization.

Initialize the foundation of the file system. This is specified by the structure rtems_filesystem_mount_table. The usual configuration is a single instantiation of the IMFS or miniIMFS with a single "/dev" directory in it.

#### 8.80.4.3 rtems_libio_iop_is_append()

```
static bool rtems_libio_iop_is_append (
            const rtems_libio_t * iop )  [inline], [static]
```

Returns true if this is an append iop, otherwise returns false.

**Parameters**

| | | |
|---|---|---|
| in | *iop* | The iop. |

Definition at line 1420 of file libio.h.

#### 8.80.4.4 rtems_libio_iop_is_no_delay()

```
static bool rtems_libio_iop_is_no_delay (
            const rtems_libio_t * iop )  [inline], [static]
```

Returns true if this is a no delay iop, otherwise returns false.

**Parameters**

| | | |
|---|---|---|
| in | *iop* | The iop. |

Definition at line 1390 of file libio.h.

#### 8.80.4.5 rtems_libio_iop_is_readable()

```
static bool rtems_libio_iop_is_readable (
            const rtems_libio_t * iop )  [inline], [static]
```

Returns true if this is a readable iop, otherwise returns false.

**Parameters**

| | | |
|---|---|---|
| in | *iop* | The iop. |

Definition at line 1400 of file libio.h.

#### 8.80.4.6 rtems_libio_iop_is_writeable()

```
static bool rtems_libio_iop_is_writeable (
            const rtems_libio_t * iop )  [inline], [static]
```

Returns true if this is a writeable iop, otherwise returns false.

**Parameters**

| | | |
|---|---|---|
| in | *iop* | The iop. |

Definition at line 1410 of file libio.h.

#### 8.80.4.7 rtems_mkdir()

```
int rtems_mkdir (
             const char * path,
             mode_t mode )
```

Creates a directory and all its parent directories according to *path*.

The *mode* value selects the access permissions of the directory.

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | An error occurred. The `errno` indicates the error. |

### 8.80.5 Variable Documentation

#### 8.80.5.1 rtems_filesystem_default_pathconf

```
const rtems_filesystem_limits_and_options_t rtems_filesystem_default_pathconf  [extern]
```

Default pathconf settings.

Override in a filesystem.

# 8.81 ISR Handler

ISR Handler.

## Modules

- ISR Locks

  *Low-level lock to protect critical sections accessed by threads and interrupt service routines.*

## Files

- file isr.h

  *Data Related to the Management of Processor Interrupt Levels.*
- file isrlevel.h

  *ISR Level Type.*
- file isr.c

  *Initialize the ISR handler.*
- file isrisinprogress.c

  *ISR Is In Progress Default Implementation.*

## Macros

- #define _ISR_Install_vector(_vector, _new_handler, _old_handler) _CPU_ISR_install_vector( _vector, _↩ new_handler, _old_handler )

  *Install interrupt handler vector.*
- #define _ISR_Local_disable(_level)

  *Disables interrupts on this processor.*
- #define _ISR_Local_enable(_level)

  *Enables interrupts on this processor.*
- #define _ISR_Local_flash(_level)

  *Temporarily enables interrupts on this processor.*
- #define _ISR_Is_enabled(_level) _CPU_ISR_Is_enabled( _level )

  *Returns true if interrupts are enabled in the specified interrupt level, otherwise returns false.*
- #define _ISR_Get_level() _CPU_ISR_Get_level()

  *Return current interrupt level.*
- #define _ISR_Set_level(_new_level)

  *Set current interrupt level.*

## Typedefs

- typedef uint32_t ISR_Vector_number
- typedef void ISR_Handler
- typedef void ∗ **ISR_Handler_entry**
- typedef uint32_t ISR_Level

## Functions

- RTEMS_DECLARE_GLOBAL_SYMBOL (_ISR_Stack_size)

  *Global symbol with a value equal to the configure interrupt stack size.*
- void _ISR_Handler_initialization (void)

  *Initializes the ISR handler.*
- void _ISR_Handler (void)

  *ISR interrupt dispatcher.*
- bool _ISR_Is_in_progress (void)

  *Checks if an ISR in progress.*

## Variables

- char _ISR_Stack_area_begin [ ]

  *The interrupt stack area begin.*
- const char _ISR_Stack_area_end [ ]

  *The interrupt stack area end.*

### 8.81.1 Detailed Description

ISR Handler.

This handler encapsulates functionality which provides the foundation ISR services used in all of the APIs supported by RTEMS.

The ISR Nest level counter variable is maintained as part of the per cpu data structure.

### 8.81.2 Macro Definition Documentation

#### 8.81.2.1 _ISR_Get_level

```
#define _ISR_Get_level( )  _CPU_ISR_Get_level()
```

Return current interrupt level.

This routine returns the current interrupt level.

LM32 Specific Information: XXX document implementation including references if appropriate

**Return values**

| | |
|---|---|
| *This* | method returns the current level. |

Definition at line 128 of file isrlevel.h.

#### 8.81.2.2 _ISR_Install_vector

```
#define _ISR_Install_vector(
              _vector,
              _new_handler,
              _old_handler )  _CPU_ISR_install_vector( _vector, _new_handler, _old_handler )
```

Install interrupt handler vector.

This routine installs new_handler as the interrupt service routine for the specified vector. The previous interrupt service routine is returned as old_handler.

LM32 Specific Information: XXX document implementation including references if appropriate

**Parameters**

| | | |
|---|---|---|
| in | *_vector* | is the vector number |
| in | *_new_handler* | is ISR handler to install |
| in | *_old_handler* | is a pointer to a variable which will be set to the old handler |

**Return values**

| | |
|---|---|
| **_old_handler* | will be set to the old ISR handler |

Definition at line 134 of file isr.h.

#### 8.81.2.3 _ISR_Is_enabled

```
#define _ISR_Is_enabled(
              _level )  _CPU_ISR_Is_enabled( _level )
```

Returns true if interrupts are enabled in the specified interrupt level, otherwise returns false.

**Parameters**

| | | |
|---|---|---|
| in | *_level* | The ISR level. |

**Return values**

| | |
|---|---|
| *true* | Interrupts are enabled in the interrupt level. |
| *false* | Otherwise. |

Definition at line 115 of file isrlevel.h.

#### 8.81.2.4 _ISR_Local_disable

```
#define _ISR_Local_disable(
              _level )
```

**Value:**

```
  do { \
    _CPU_ISR_Disable( _level ); \
    RTEMS_COMPILER_MEMORY_BARRIER(); \
  } while (0)
```

Disables interrupts on this processor.

This macro disables all interrupts on this processor so that a critical section of code is protected from concurrent access by interrupts of this processor. Disabling of interrupts disables thread dispatching on the processor as well.

On SMP configurations other processors can enter such sections if not protected by other means.

**Parameters**

| | | |
|---|---|---|
| out | *_level* | The argument *_level* will contain the previous interrupt mask level. |

Definition at line 57 of file isrlevel.h.

#### 8.81.2.5 _ISR_Local_enable

```
#define _ISR_Local_enable(
              _level )
```

**Value:**

```
  do { \
    RTEMS_COMPILER_MEMORY_BARRIER(); \
    _CPU_ISR_Enable( _level ); \
  } while (0)
```

Enables interrupts on this processor.

This macro restores the interrupt status on the processor with the interrupt level value obtained by _ISR_Local_disable(). It is used at the end of a critical section of code to enable interrupts so they can be processed again.

**Parameters**

| | | |
|---|---|---|
| in | *_level* | The interrupt level previously obtained by _ISR_Local_disable(). |

Definition at line 74 of file isrlevel.h.

#### 8.81.2.6 _ISR_Local_flash

```
#define _ISR_Local_flash(
              _level )
```

**Value:**

```
do { \
  RTEMS_COMPILER_MEMORY_BARRIER(); \
  _CPU_ISR_Flash( _level ); \
  RTEMS_COMPILER_MEMORY_BARRIER(); \
} while (0)
```

Temporarily enables interrupts on this processor.

This macro temporarily enables interrupts to the previous interrupt mask level and then disables all interrupts so that the caller can continue into the second part of a critical section.

This routine is used to temporarily enable interrupts during a long critical section. It is used in long sections of critical code when a point is reached at which interrupts can be temporarily enabled. Deciding where to flash interrupts in a long critical section is often difficult and the point must be selected with care to ensure that the critical section properly protects itself.

**Parameters**

| | | |
|---|---|---|
| in | *_level* | The interrupt level previously obtained by _ISR_Local_disable(). |

Definition at line 99 of file isrlevel.h.

#### 8.81.2.7 _ISR_Set_level

```
#define _ISR_Set_level(
              _new_level )
```

**Value:**

```
do { \
  RTEMS_COMPILER_MEMORY_BARRIER();  \
  _CPU_ISR_Set_level( _new_level ); \
  RTEMS_COMPILER_MEMORY_BARRIER();  \
} while (0)
```

Set current interrupt level.

This routine sets the current interrupt level to that specified by *_new_level*. The new interrupt level is effective when the routine exits.

**Parameters**

| | | |
|---|---|---|
| in | *_new_level* | contains the desired interrupt level. |

Definition at line 140 of file isrlevel.h.

### 8.81.3 Typedef Documentation

#### 8.81.3.1 ISR_Handler

```
typedef void ISR_Handler
```

Return type for ISR Handler

Definition at line 56 of file isr.h.

#### 8.81.3.2 ISR_Level

```
typedef uint32_t ISR_Level
```

The following type defines the control block used to manage the interrupt level portion of the status register.

Definition at line 41 of file isrlevel.h.

#### 8.81.3.3 ISR_Vector_number

```
typedef uint32_t ISR_Vector_number
```

The following type defines the type used to manage the vectors.

Definition at line 51 of file isr.h.

### 8.81.4 Function Documentation

#### 8.81.4.1 _ISR_Handler()

```
void _ISR_Handler (
             void  )
```

ISR interrupt dispatcher.

This routine is the interrupt dispatcher. ALL interrupts are vectored to this routine so that minimal context can be saved and setup performed before the application's high-level language interrupt service routine is invoked. After the application's interrupt service routine returns control to this routine, it will determine if a thread dispatch is necessary. If so, it will ensure that the necessary thread scheduling operations are performed when the outermost interrupt service routine exits.

**Note**

Typically implemented in assembly language.

#### 8.81.4.2 _ISR_Handler_initialization()

```
void _ISR_Handler_initialization (
             void  )
```

Initializes the ISR handler.

This routine performs the initialization necessary for the ISR handler.

Definition at line 36 of file isr.c.

#### 8.81.4.3 _ISR_Is_in_progress()

```
bool _ISR_Is_in_progress (
             void  )
```

Checks if an ISR in progress.

This function returns true if the processor is currently servicing and interrupt and false otherwise. A return value of true indicates that the caller is an interrupt service routine, NOT a thread.

**Return values**

| | |
|---|---|
| *true* | Returns true when called from an ISR. |
| *false* | Returns false when not called from an ISR. |

Definition at line 32 of file isrisinprogress.c.

#### 8.81.4.4 RTEMS_DECLARE_GLOBAL_SYMBOL()

```
RTEMS_DECLARE_GLOBAL_SYMBOL (
             _ISR_Stack_size  )
```

Global symbol with a value equal to the configure interrupt stack size.

This global symbol is defined by the application configuration option CONFIGURE_INIT_TASK_STACK_SIZE via <rtems/confdefs.h>.

### 8.81.5 Variable Documentation

#### 8.81.5.1 _ISR_Stack_area_begin

```
char _ISR_Stack_area_begin[]  [extern]
```

The interrupt stack area begin.

The interrupt stack area is defined by the application configuration via <rtems/confdefs.h>. The size of the area depends on CONFIGURE_INIT_TASK_STACK_SIZE and CONFIGURE_MAXIMUM_PROCESSORS.

#### 8.81.5.2 _ISR_Stack_area_end

```
const char _ISR_Stack_area_end[]  [extern]
```

The interrupt stack area end.

The interrupt stack area is defined by the application configuration via <rtems/confdefs.h>. The size of the area depends on CONFIGURE_INIT_TASK_STACK_SIZE and CONFIGURE_MAXIMUM_PROCESSORS.

## 8.82 ISR Locks

Low-level lock to protect critical sections accessed by threads and interrupt service routines.

### Files

- file isrlock.h

  *ISR Locks.*

### Classes

- struct ISR_lock_Control

  *ISR lock control.*
- struct ISR_lock_Context

  *Local ISR lock context for acquire and release pairs.*

### Macros

- #define ISR_LOCK_MEMBER(_designator) ISR_lock_Control _designator;

  *Defines an ISR lock member.*
- #define ISR_LOCK_DECLARE(_qualifier, _designator) _qualifier ISR_lock_Control _designator;

  *Declares an ISR lock variable.*
- #define ISR_LOCK_DEFINE(_qualifier, _designator, _name) _qualifier ISR_lock_Control _designator = { SMP_LOCK_INITIALIZER( _name ) };

  *Defines an ISR lock variable.*
- #define ISR_LOCK_REFERENCE(_designator, _target) ISR_lock_Control *_designator = _target;

  *Defines an ISR lock variable reference.*
- #define ISR_LOCK_INITIALIZER(_name) { SMP_LOCK_INITIALIZER( _name ) }

  *Initializer for static initialization of ISR locks.*
- #define _ISR_lock_Initialize(_lock, _name) _SMP_lock_Initialize( &( _lock )->Lock, _name )

  *Initializes an ISR lock.*
- #define _ISR_lock_Destroy(_lock) _SMP_lock_Destroy( &( _lock )->Lock )

  *Destroys an ISR lock.*
- #define _ISR_lock_Set_name(_lock, _name) _SMP_lock_Set_name( &( _lock )->Lock, _name )

  *Sets the name of an ISR lock.*
- #define _ISR_lock_ISR_disable_and_acquire(_lock, _context)

  *Acquires an ISR lock.*
- #define _ISR_lock_Release_and_ISR_enable(_lock, _context)

  *Releases an ISR lock.*
- #define _ISR_lock_Acquire(_lock, _context)

  *Acquires an ISR lock inside an ISR disabled section.*
- #define _ISR_lock_Release(_lock, _context)

  *Releases an ISR lock inside an ISR disabled section.*
- #define _ISR_lock_Acquire_inline(_lock, _context)

  *Acquires an ISR lock inside an ISR disabled section (inline).*
- #define _ISR_lock_Release_inline(_lock, _context)

  *Releases an ISR lock inside an ISR disabled section (inline).*
- #define **_ISR_lock_ISR_disable_profile**(_context)
- #define _ISR_lock_ISR_disable(_context)

  *Disables interrupts and saves the previous interrupt state in the ISR lock context.*
- #define _ISR_lock_ISR_enable(_context) _ISR_Local_enable( ( _context )->Lock_context.isr_level )

  *Restores the saved interrupt state of the ISR lock context.*

## Functions

- static __inline__ void _ISR_lock_Context_set_level (ISR_lock_Context *context, ISR_Level level)

  *Sets the ISR level in the ISR lock context.*

## 8.82.1 Detailed Description

Low-level lock to protect critical sections accessed by threads and interrupt service routines.

On single processor configurations the ISR locks degrade to simple ISR disable/enable sequences. No additional storage or objects are required.

This synchronization primitive is supported on SMP configurations. Here SMP locks are used.

## 8.82.2 Macro Definition Documentation

### 8.82.2.1 _ISR_lock_Acquire

```
#define _ISR_lock_Acquire(
              _lock,
              _context )
```

**Value:**

```
    do { \
      _Assert( _ISR_Get_level() != 0 ); \
      _SMP_lock_Acquire( \
        &( _lock )->Lock, \
        &( _context )->Lock_context \
      ); \
    } while ( 0 )
```

Acquires an ISR lock inside an ISR disabled section.

The interrupt status will remain unchanged. On SMP configurations this function acquires an SMP lock.

In case the executing context can be interrupted by higher priority interrupts and these interrupts enter the critical section protected by this lock, then the result is unpredictable.

**Parameters**

| | | |
|---|---|---|
| in | *_lock* | The ISR lock control. |
| in | *_context* | The local ISR lock context for an acquire and release pair. |

**See also**

_ISR_lock_Release().

Definition at line 284 of file isrlock.h.

#### 8.82.2.2 _ISR_lock_Acquire_inline

```
#define _ISR_lock_Acquire_inline(
              _lock,
              _context )
```

**Value:**

```
    do { \
      _Assert( _ISR_Get_level() != 0 ); \
      _SMP_lock_Acquire_inline( \
        &( _lock )->Lock, \
        &( _context )->Lock_context \
      ); \
    } while ( 0 )
```

Acquires an ISR lock inside an ISR disabled section (inline).

**See also**

_ISR_lock_Acquire().

Definition at line 326 of file isrlock.h.

#### 8.82.2.3 _ISR_lock_Destroy

```
#define _ISR_lock_Destroy(
              _lock )  _SMP_lock_Destroy( &( _lock )->Lock )
```

Destroys an ISR lock.

Concurrent destruction leads to unpredictable results.

**Parameters**

| | | |
|---|---|---|
| in | *_lock* | The ISR lock control. |

Definition at line 196 of file isrlock.h.

#### 8.82.2.4 _ISR_lock_Initialize

```
#define _ISR_lock_Initialize(
              _lock,
              _name )  _SMP_lock_Initialize( &( _lock )->Lock, _name )
```

Initializes an ISR lock.

Concurrent initialization leads to unpredictable results.

**Parameters**

| | | |
|---|---|---|
| in | *_lock* | The ISR lock control. |
| in | *_name* | The name for the ISR lock. This name must be a string persistent throughout the life time of this lock. The name is only used if profiling is enabled. |

Definition at line 182 of file isrlock.h.

#### 8.82.2.5 _ISR_lock_ISR_disable

```
#define _ISR_lock_ISR_disable(
              _context )
```

**Value:**

```
do { \
  _ISR_Local_disable( ( _context )->Lock_context.isr_level ); \
  _ISR_lock_ISR_disable_profile( _context ) \
} while ( 0 )
```

Disables interrupts and saves the previous interrupt state in the ISR lock context.

This function can be used in thread and interrupt context.

**Parameters**

| | | |
|---|---|---|
| in | *_context* | The local ISR lock context to store the interrupt state. |

**See also**

_ISR_lock_ISR_enable().

Definition at line 392 of file isrlock.h.

#### 8.82.2.6 _ISR_lock_ISR_disable_and_acquire

```
#define _ISR_lock_ISR_disable_and_acquire(
              _lock,
              _context )
```

**Value:**

```
_SMP_lock_ISR_disable_and_acquire( \
  &( _lock )->Lock, \
  &( _context )->Lock_context \
)
```

Acquires an ISR lock.

Interrupts will be disabled. On SMP configurations this function acquires an SMP lock.

This function can be used in thread and interrupt context.

**Parameters**

| | | |
|---|---|---|
| in | *_lock* | The ISR lock control. |
| in | *_context* | The local ISR lock context for an acquire and release pair. |

**See also**

_ISR_lock_Release_and_ISR_enable().

Definition at line 232 of file isrlock.h.

#### 8.82.2.7 _ISR_lock_ISR_enable

```
#define _ISR_lock_ISR_enable(
              _context )  _ISR_Local_enable( ( _context )->Lock_context.isr_level )
```

Restores the saved interrupt state of the ISR lock context.

This function can be used in thread and interrupt context.

**Parameters**

| | | |
|---|---|---|
| in | *_context* | The local ISR lock context containing the saved interrupt state. |

**See also**

_ISR_lock_ISR_disable().

Definition at line 416 of file isrlock.h.

#### 8.82.2.8 _ISR_lock_Release

```
#define _ISR_lock_Release(
              _lock,
              _context )
```

**Value:**

```
_SMP_lock_Release( \
  &( _lock )->Lock, \
  &( _context )->Lock_context \
)
```

Releases an ISR lock inside an ISR disabled section.

The interrupt status will remain unchanged. On SMP configurations this function releases an SMP lock.

**Parameters**

| | | |
|---|---|---|
| in | *_lock* | The ISR lock control. |
| in | *_context* | The local ISR lock context for an acquire and release pair. |

**See also**

_ISR_lock_Acquire().

Definition at line 310 of file isrlock.h.

### 8.82.2.9 _ISR_lock_Release_and_ISR_enable

```
#define _ISR_lock_Release_and_ISR_enable(
              _lock,
              _context )
```

**Value:**

```
_SMP_lock_Release_and_ISR_enable( \
  &( _lock )->Lock, \
  &( _context )->Lock_context \
)
```

Releases an ISR lock.

The interrupt status will be restored. On SMP configurations this function releases an SMP lock.

This function can be used in thread and interrupt context.

**Parameters**

| | | |
|---|---|---|
| in | *_lock* | The ISR lock control. |
| in | *_context* | The local ISR lock context for an acquire and release pair. |

**See also**

_ISR_lock_ISR_disable_and_acquire().

Definition at line 257 of file isrlock.h.

### 8.82.2.10 _ISR_lock_Release_inline

```
#define _ISR_lock_Release_inline(
              _lock,
              _context )
```

**Value:**

```
_SMP_lock_Release_inline( \
  &( _lock )->Lock, \
  &( _context )->Lock_context \
)
```

Releases an ISR lock inside an ISR disabled section (inline).

**See also**

_ISR_lock_Release().

Definition at line 345 of file isrlock.h.

#### 8.82.2.11 _ISR_lock_Set_name

```
#define _ISR_lock_Set_name(
              _lock,
              _name )  _SMP_lock_Set_name( &( _lock )->Lock, _name )
```

Sets the name of an ISR lock.

**Parameters**

| | | |
|---|---|---|
| out | *_lock* | The ISR lock control. |
| | *_name* | The name for the ISR lock. This name must be a string persistent throughout the life time of this lock. The name is only used if profiling is enabled. |

Definition at line 211 of file isrlock.h.

#### 8.82.2.12 ISR_LOCK_DECLARE

```
#define ISR_LOCK_DECLARE(
              _qualifier,
              _designator )  _qualifier ISR_lock_Control _designator;
```

Declares an ISR lock variable.

Do not add a ';' after this macro.

**Parameters**

| | |
|---|---|
| *_qualifier* | The qualifier for the interrupt lock, e.g. extern. |
| *_designator* | The designator for the interrupt lock. |

Definition at line 101 of file isrlock.h.

#### 8.82.2.13 ISR_LOCK_DEFINE

```
#define ISR_LOCK_DEFINE(
              _qualifier,
```

```
              _designator,
              _name )  _qualifier ISR_lock_Control _designator = { SMP_LOCK_INITIALIZER( _name
) };
```

Defines an ISR lock variable.

Do not add a ';' after this macro.

**Parameters**

| | |
|---|---|
| *_qualifier* | The qualifier for the interrupt lock, e.g. static. |
| *_designator* | The designator for the interrupt lock. |
| *_name* | The name for the interrupt lock. It must be a string. The name is only used if profiling is enabled. |

Definition at line 118 of file isrlock.h.

#### 8.82.2.14 ISR_LOCK_INITIALIZER

```
#define ISR_LOCK_INITIALIZER(
              _name )  { SMP_LOCK_INITIALIZER( _name ) }
```

Initializer for static initialization of ISR locks.

**Parameters**

| | |
|---|---|
| *_name* | The name for the interrupt lock. It must be a string. The name is only used if profiling is enabled. |

Definition at line 146 of file isrlock.h.

#### 8.82.2.15 ISR_LOCK_MEMBER

```
#define ISR_LOCK_MEMBER(
              _designator ) ISR_lock_Control _designator;
```

Defines an ISR lock member.

Do not add a ';' after this macro.

**Parameters**

| | |
|---|---|
| *_designator* | The designator for the interrupt lock. |

Definition at line 87 of file isrlock.h.

#### 8.82.2.16 ISR_LOCK_REFERENCE

```
#define ISR_LOCK_REFERENCE(
              _designator,
              _target )  ISR_lock_Control *_designator = _target;
```

Defines an ISR lock variable reference.

Do not add a ';' after this macro.

**Parameters**

| | |
|---|---|
| *_designator* | The designator for the interrupt lock reference. |
| *_target* | The target for the interrupt lock reference. |

Definition at line 133 of file isrlock.h.

### 8.82.3 Function Documentation

#### 8.82.3.1 _ISR_lock_Context_set_level()

```
static __inline__ void _ISR_lock_Context_set_level (
             ISR_lock_Context * context,
             ISR_Level level )  [static]
```

Sets the ISR level in the ISR lock context.

**Parameters**

| | | |
|---|---|---|
| out | *context* | The ISR lock context. |
| | *level* | The ISR level. |

Definition at line 159 of file isrlock.h.

# 8.83 Idle Task Configuration

## Macros

- #define CONFIGURE_IDLE_TASK_BODY

  *This configuration option is an initializer define.*
- #define CONFIGURE_IDLE_TASK_INITIALIZES_APPLICATION

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_IDLE_TASK_STACK_SIZE

  *This configuration option is an integer define.*

## 8.83.1 Detailed Description

This section describes configuration options related to the idle tasks.

## 8.83.2 Macro Definition Documentation

### 8.83.2.1 CONFIGURE_IDLE_TASK_BODY

```
#define CONFIGURE_IDLE_TASK_BODY
```

This configuration option is an initializer define.

The value of this configuration option initializes the IDLE thread body.

#### Default Value

If BSP_IDLE_TASK_BODY is defined, then this will be the default value, otherwise the default value is `_CPU_Thread_Idle_body`.

#### Value Constraints

The value of this configuration option shall be defined to a valid function pointer of the type `void *( *idle_body )( uintptr_t )`.

#### Notes

IDLE threads shall not block. A blocking IDLE thread results in undefined system behaviour because the scheduler assume that at least one ready thread exists.

IDLE threads can be used to initialize the application, see configuration option CONFIGURE_IDLE_TASK_INITIALIZES_APPLIC

Definition at line 3147 of file appl-config.h.

#### 8.83.2.2 CONFIGURE_IDLE_TASK_INITIALIZES_APPLICATION

```
#define CONFIGURE_IDLE_TASK_INITIALIZES_APPLICATION
```

This configuration option is a boolean feature define.

This configuration option is defined to indicate that the user has configured **no** user initialization tasks or threads and that the user provided IDLE task will perform application initialization and then transform itself into an IDLE task.

##### Default Configuration

If this configuration option is undefined, then the user is assumed to provide one or more initialization tasks.

##### Notes

If you use this option be careful, the user IDLE task **cannot** block at all during the initialization sequence. Further, once application initialization is complete, it shall make itself preemptible and enter an idle body loop.

The IDLE task shall run at the lowest priority of all tasks in the system.

If this configuration option is defined, then it is mandatory to configure a user IDLE task with the CONFIGURE_IDLE_TASK_BODY configuration option, otherwise a compile time error in the configuration file will occur.

The application shall define exactly one of the following configuration options

- CONFIGURE_RTEMS_INIT_TASKS_TABLE,
- CONFIGURE_POSIX_INIT_THREAD_TABLE, or
- CONFIGURE_IDLE_TASK_INITIALIZES_APPLICATION

otherwise a compile time error in the configuration file will occur.

Definition at line 3188 of file appl-config.h.

#### 8.83.2.3 CONFIGURE_IDLE_TASK_STACK_SIZE

```
#define CONFIGURE_IDLE_TASK_STACK_SIZE
```

This configuration option is an integer define.

The value of this configuration option defines the task stack size for an IDLE task.

##### Default Value

The default value is CONFIGURE_MINIMUM_TASK_STACK_SIZE.

##### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to a BSP-specific and application-specific minimum value.
- It shall be small enough so that the IDLE task stack area calculation carried out by <rtems/confdefs.h> does not overflow an integer of type size_t.

##### Notes

In SMP configurations, there is one IDLE task per configured processor, see CONFIGURE_MAXIMUM_PROCESSORS.

Definition at line 3218 of file appl-config.h.

## 8.84 Implementation

This group contains the RTEMS implementation components.

### Modules

- Application Configuration

  *Evaluation of Application Configuration Options.*
- Classic

  *Classic.*
- IO Library

  *Provides system call and file system interface definitions.*
- Standard C Library Support

  *RTEMS Specific Support for the Standard C Library.*
- SuperCore

  *Provides services for all APIs.*

### 8.84.1 Detailed Description

This group contains the RTEMS implementation components.

## 8.85 Initialization and Shutdown

This group contains directives to initialize and shutdown the RTEMS executive.

### Functions

- RTEMS_NO_RETURN void rtems_initialize_executive (void)

  *Initializes the system and starts multitasking.*
- RTEMS_NO_RETURN void rtems_shutdown_executive (uint32_t result)

  *Shuts down the RTEMS environment.*

### 8.85.1 Detailed Description

This group contains directives to initialize and shutdown the RTEMS executive.

### 8.85.2 Function Documentation

#### 8.85.2.1 rtems_initialize_executive()

```
RTEMS_NO_RETURN void rtems_initialize_executive (
            void  )
```

Initializes the system and starts multitasking.

Iterates through the system initialization linker set and invokes the registered handlers. The final step is to start multitasking.

This directive should be called by boot_card() only.

This directive does not return to the caller. Errors in the initialization sequence are usually fatal and lead to a system termination.

Definition at line 121 of file exinit.c.

#### 8.85.2.2 rtems_shutdown_executive()

```
RTEMS_NO_RETURN void rtems_shutdown_executive (
            uint32_t result )
```

Shuts down the RTEMS environment.

The invocation of this directive results in the RTEMS environment being shutdown and multitasking halted. The system is terminated with a fatal source of RTEMS_FATAL_SOURCE_EXIT and the specified result code.

This directive does not return to the caller.

**Parameters**

| | |
|---|---|
| *result* | is the result code. |

## 8.86 Internal Error Handler

Internal Error Handler.

### Files

- file interr.h

  *Constants and Prototypes Related to the Internal Error Handler.*
- file interr.c

  *Initiates system termination.*

### Classes

- struct Internal_errors_Information

### Typedefs

- typedef CPU_Uint32ptr **Internal_errors_t**

### Enumerations

- enum Internal_errors_Source {
  INTERNAL_ERROR_CORE = 0, INTERNAL_ERROR_RTEMS_API = 1, INTERNAL_ERROR_POSIX_API = 2, RTEMS_FATAL_SOURCE_BDBUF = 3,
  RTEMS_FATAL_SOURCE_APPLICATION = 4, RTEMS_FATAL_SOURCE_EXIT = 5, RTEMS_FATAL_SOURCE_BSP = 6, RTEMS_FATAL_SOURCE_ASSERT = 7,
  RTEMS_FATAL_SOURCE_STACK_CHECKER = 8, RTEMS_FATAL_SOURCE_EXCEPTION = 9, RTEMS_FATAL_SOURCE_SMP = 10, RTEMS_FATAL_SOURCE_PANIC = 11,
  RTEMS_FATAL_SOURCE_INVALID_HEAP_FREE = 12, RTEMS_FATAL_SOURCE_HEAP = 13, RTEMS_FATAL_SOURCE_LAST = 0xffffffff }

  *This type lists the possible sources from which an error can be reported.*
- enum Internal_errors_Core_list {
  **INTERNAL_ERROR_TOO_LITTLE_WORKSPACE** = 2, **INTERNAL_ERROR_THREAD_EXITTED** = 5, **INTERNAL_ERROR_INCONSISTENT_MP_INFORMATION** = 6, **INTERNAL_ERROR_INVALID_NODE** = 7,
  **INTERNAL_ERROR_NO_MPCI** = 8, **INTERNAL_ERROR_BAD_PACKET** = 9, **INTERNAL_ERROR_OUT_OF_PACKETS** = 10, **INTERNAL_ERROR_OUT_OF_GLOBAL_OBJECTS** = 11,
  **INTERNAL_ERROR_OUT_OF_PROXIES** = 12, **INTERNAL_ERROR_INVALID_GLOBAL_ID** = 13, **INTERNAL_ERROR_BAD_STACK_HOOK** = 14, **INTERNAL_ERROR_GXX_KEY_ADD_FAILED** = 21,
  **INTERNAL_ERROR_GXX_MUTEX_INIT_FAILED** = 22, **INTERNAL_ERROR_NO_MEMORY_FOR_HEAP** = 23, **INTERNAL_ERROR_CPU_ISR_INSTALL_VECTOR** = 24, **INTERNAL_ERROR_RESOURCE_IN_USE** = 25,
  **INTERNAL_ERROR_RTEMS_INIT_TASK_ENTRY_IS_NULL** = 26, **INTERNAL_ERROR_THREAD_QUEUE_DEADLOCK** = 28, **INTERNAL_ERROR_THREAD_QUEUE_ENQUEUE_STICKY_FROM_BAD_STATE** = 29, **INTERNAL_ERROR_BAD_THREAD_DISPATCH_DISABLE_LEVEL** = 30,
  **INTERNAL_ERROR_BAD_THREAD_DISPATCH_ENVIRONMENT** = 31, **INTERNAL_ERROR_RTEMS_INIT_TASK_CREATE_FAILED** = 32, **INTERNAL_ERROR_POSIX_INIT_THREAD_CREATE_FAILED** = 33, **INTERNAL_ERROR_LIBIO_STDOUT_FD_OPEN_FAILED** = 36,
  **INTERNAL_ERROR_LIBIO_STDERR_FD_OPEN_FAILED** = 37, **INTERNAL_ERROR_ILLEGAL_USE_OF_FLOATING_POINT_UNIT** = 38, **INTERNAL_ERROR_ARC4RANDOM_GETENTROPY_FAIL** = 39, **INTERNAL_ERROR_NO_MEMORY_FOR_PER_CPU_DATA** = 40,
  **INTERNAL_ERROR_TOO_LARGE_TLS_SIZE** = 41 }

  *A list of errors which are generated internally by the executive core.*

## Functions

- RTEMS_NO_RETURN void _Terminate (Internal_errors_Source the_source, Internal_errors_t the_error)

  *Initiates system termination.*
- RTEMS_NO_RETURN void _Internal_error (Internal_errors_Core_list core_error)

  *Terminates the system with an INTERNAL_ERROR_CORE fatal source and the specified core error code.*

## Variables

- Internal_errors_Information _Internal_errors_What_happened

### 8.86.1 Detailed Description

Internal Error Handler.

This handler encapsulates functionality which provides the foundation Semaphore services used in all of the APIs supported by RTEMS.

### 8.86.2 Enumeration Type Documentation

#### 8.86.2.1 Internal_errors_Core_list

```
enum Internal_errors_Core_list
```

A list of errors which are generated internally by the executive core.

Do not re-use numbers of obsolete error codes. Comment no longer used error codes and do not uncomment commented or obsolete error codes.

Definition at line 165 of file interr.h.

**Enumerator**

#### 8.86.2.2 Internal_errors_Source

```
enum Internal_errors_Source
```

This type lists the possible sources from which an error can be reported.

**Enumerator**

| | |
|---|---|
| INTERNAL_ERROR_CORE | Errors of the core system.<br>**See also**<br>Internal_errors_Core_list. |
| INTERNAL_ERROR_RTEMS_API | Errors of the RTEMS API. |
| INTERNAL_ERROR_POSIX_API | Errors of the POSIX API. |
| RTEMS_FATAL_SOURCE_BDBUF | Fatal source for the block device cache.<br>**See also**<br>rtems_bdbuf_fatal_code. |
| RTEMS_FATAL_SOURCE_APPLICATION | Fatal source for application specific errors. The fatal code is application specific. |
| RTEMS_FATAL_SOURCE_EXIT | Fatal source of exit(). The fatal code is the exit() status code. |
| RTEMS_FATAL_SOURCE_BSP | Fatal source for BSP errors. The fatal codes are defined in <bsp/fatal.h>. Examples are interrupt and exception initialization.<br>**See also**<br>bsp_fatal_code and bsp_fatal(). |
| RTEMS_FATAL_SOURCE_ASSERT | Fatal source of assert(). The fatal code is the pointer value of the assert context.<br>**See also**<br>rtems_assert_context. |
| RTEMS_FATAL_SOURCE_STACK_CHECKER | Fatal source of the stack checker. The fatal code is the object name of the executing task. |
| RTEMS_FATAL_SOURCE_EXCEPTION | Fatal source of the exceptions. The fatal code is the pointer value of the exception frame pointer.<br>**See also**<br>rtems_exception_frame and rtems_exception_frame_print(). |
| RTEMS_FATAL_SOURCE_SMP | Fatal source of SMP domain.<br>**See also**<br>SMP_Fatal_code. |
| RTEMS_FATAL_SOURCE_PANIC | Fatal source of rtems_panic().<br>**See also**<br>rtem |

**Enumerator**

| | |
|---|---|
| RTEMS_FATAL_SOURCE_INVALID_HEAP_FREE | Fatal source for invalid C program heap frees via free(). The fatal code is the bad pointer. |
| RTEMS_FATAL_SOURCE_HEAP | Fatal source for heap errors. The fatal code is the address to a heap error context (Heap_Error_context). |
| RTEMS_FATAL_SOURCE_LAST | The last available fatal source. This enum value ensures that the enum type needs at least 32-bits for architectures with short enums. |

Definition at line 47 of file interr.h.

## 8.86.3 Function Documentation

### 8.86.3.1 _Internal_error()

```
RTEMS_NO_RETURN void _Internal_error (
            Internal_errors_Core_list core_error )
```

Terminates the system with an INTERNAL_ERROR_CORE fatal source and the specified core error code.

**Parameters**

| | |
|---|---|
| *core_error* | The core error code. |

**See also**

_Terminate().

Definition at line 51 of file interr.c.

### 8.86.3.2 _Terminate()

```
RTEMS_NO_RETURN void _Terminate (
            Internal_errors_Source the_source,
            Internal_errors_t the_error )
```

Initiates system termination.

This routine is invoked when the application or the executive itself determines that a fatal error has occurred or a final system state is reached (for example after exit()).

The first action of this function is to call the fatal handler of the user extensions. For the initial extensions the following conditions are required

- a valid stack pointer and enough stack space,
- a valid code memory, and
- valid read-only data.

For the initial extensions the read-write data (including BSS segment) is not required on single processor configurations. On SMP configurations however the read-write data must be initialized since this function must determine the state of the other processors and request them to shut-down if necessary.

Non-initial extensions require in addition valid read-write data. The BSP may install an initial extension that performs a system reset. In this case the non-initial extensions will be not called.

Once all fatal handler executed the error information will be stored to _Internal_errors_What_happened and the system state is set to SYSTEM_STATE_TERMINATED.

The final step is to call the CPU specific _CPU_Fatal_halt().

**Parameters**

| | |
|---|---|
| *the_source* | The fatal source indicating the subsystem the fatal condition originated in. |
| *the_error* | The fatal error code. This value must be interpreted with respect to the source. |

**See also**

rtems_fatal() and _Internal_error().

Definition at line 31 of file interr.c.

## 8.86.4 Variable Documentation

### 8.86.4.1 _Internal_errors_What_happened

```
Internal_errors_Information _Internal_errors_What_happened  [extern]
```

When a fatal error occurs, the error information is stored here.

Definition at line 29 of file interr.c.

## 8.87 Interrupt Manager

Any real-time executive must provide a mechanism for quick response to externally generated interrupts to satisfy the critical time constraints of the application. The Interrupt Manager provides this mechanism for RTEMS. This manager permits quick interrupt response times by providing the critical ability to alter task execution which allows a task to be preempted upon exit from an ISR.

### Modules

- Interrupt Manager Extension

### Macros

- #define rtems_interrupt_cause(_interrupt_to_cause) %
  %
- #define rtems_interrupt_clear(_interrupt_to_clear) %
  %
- #define rtems_interrupt_is_in_progress() _ISR_Is_in_progress()
  %
- #define rtems_interrupt_local_disable(_isr_cookie) _ISR_Local_disable( _isr_cookie )
  %
- #define rtems_interrupt_local_enable(_isr_cookie) _ISR_Local_enable( _isr_cookie )
  %
- #define rtems_interrupt_lock_acquire(_lock, _lock_context) _ISR_lock_ISR_disable_and_acquire( _lock, ↩ _lock_context )
  %
- #define rtems_interrupt_lock_acquire_isr(_lock, _lock_context)
  %
- #define RTEMS_INTERRUPT_LOCK_DECLARE(_qualifier, _designator) ISR_LOCK_DECLARE( _qualifier, _designator )
  %
- #define RTEMS_INTERRUPT_LOCK_DEFINE(_qualifier, _designator, _name) ISR_LOCK_DEFINE( _↩ qualifier, _designator, _name )
  %
- #define rtems_interrupt_lock_destroy(_lock) _ISR_lock_Destroy( _lock )
  %
- #define rtems_interrupt_lock_initialize(_lock, _name) _ISR_lock_Initialize( _lock, _name )
  %
- #define RTEMS_INTERRUPT_LOCK_INITIALIZER(_name) ISR_LOCK_INITIALIZER( _name )
  %
- #define rtems_interrupt_lock_interrupt_disable(_lock_context) _ISR_lock_ISR_disable( _lock_context )
  %
- #define RTEMS_INTERRUPT_LOCK_MEMBER(_designator) ISR_LOCK_MEMBER( _designator )
  %
- #define RTEMS_INTERRUPT_LOCK_REFERENCE(_designator, _target) ISR_LOCK_REFERENCE( _↩ designator, _target )
  %
- #define rtems_interrupt_lock_release(_lock, _lock_context) _ISR_lock_Release_and_ISR_enable( _lock, ↩ _lock_context )
  %
- #define rtems_interrupt_lock_release_isr(_lock, _lock_context)
  %

## Typedefs

- typedef ISR_Handler rtems_isr

  %
- typedef void(∗ rtems_isr_entry) (void ∗)

  *Interrupt service routines installed by rtems_interrupt_catch() shall have this function pointer type.*
- typedef ISR_Level rtems_interrupt_level

  %
- typedef ISR_lock_Control rtems_interrupt_lock

  %
- typedef ISR_lock_Context rtems_interrupt_lock_context

  %
- typedef ISR_Vector_number rtems_vector_number

  %

## Functions

- rtems_status_code rtems_interrupt_catch (rtems_isr_entry new_isr_handler, rtems_vector_number vector, rtems_isr_entry ∗old_isr_handler)

  %

### 8.87.1 Detailed Description

Any real-time executive must provide a mechanism for quick response to externally generated interrupts to satisfy the critical time constraints of the application. The Interrupt Manager provides this mechanism for RTEMS. This manager permits quick interrupt response times by providing the critical ability to alter task execution which allows a task to be preempted upon exit from an ISR.

### 8.87.2 Macro Definition Documentation

#### 8.87.2.1 rtems_interrupt_cause

```
#define rtems_interrupt_cause(
              _interrupt_to_cause ) %
```

%

**Parameters**

| | |
|---|---|
| *_interrupt_to_cause* | % |

Definition at line 90 of file intr.h.

### 8.87.2.2 rtems_interrupt_clear

```
#define rtems_interrupt_clear(
              _interrupt_to_clear ) %
```

%

**Parameters**

| _interrupt_to_clear | % |
|---|---|

Definition at line 101 of file intr.h.

### 8.87.2.3 rtems_interrupt_local_disable

```
#define rtems_interrupt_local_disable(
              _isr_cookie )  _ISR_Local_disable( _isr_cookie )
```

%

**Parameters**

| _isr_cookie | % |
|---|---|

Definition at line 192 of file intr.h.

### 8.87.2.4 rtems_interrupt_local_enable

```
#define rtems_interrupt_local_enable(
              _isr_cookie )  _ISR_Local_enable( _isr_cookie )
```

%

**Parameters**

| _isr_cookie | % |
|---|---|

Definition at line 204 of file intr.h.

### 8.87.2.5 rtems_interrupt_lock_acquire

```
#define rtems_interrupt_lock_acquire(
              _lock,
              _lock_context )  _ISR_lock_ISR_disable_and_acquire( _lock, _lock_context )
```

%

**Parameters**

| | |
|---|---|
| *_lock* | % |
| *_lock_context* | % |

Definition at line 227 of file intr.h.

#### 8.87.2.6 rtems_interrupt_lock_acquire_isr

```
#define rtems_interrupt_lock_acquire_isr(
              _lock,
              _lock_context )
```

**Value:**

```
_SMP_lock_Acquire( \
  &( _lock )->Lock, \
  &( _lock_context )->Lock_context \
)
```

%

**Parameters**

| | |
|---|---|
| *_lock* | % |
| *_lock_context* | % |

Definition at line 242 of file intr.h.

#### 8.87.2.7 RTEMS_INTERRUPT_LOCK_DECLARE

```
#define RTEMS_INTERRUPT_LOCK_DECLARE(
              _qualifier,
              _designator )  ISR_LOCK_DECLARE( _qualifier, _designator )
```

%

**Parameters**

| | |
|---|---|
| *_qualifier* | % |
| *_designator* | % |

Definition at line 272 of file intr.h.

#### 8.87.2.8 RTEMS_INTERRUPT_LOCK_DEFINE

```
#define RTEMS_INTERRUPT_LOCK_DEFINE(
              _qualifier,
              _designator,
              _name )  ISR_LOCK_DEFINE( _qualifier, _designator, _name )
```

%

**Parameters**

| | |
|---|---|
| *_qualifier* | % |
| *_designator* | % |
| *_name* | % |

Definition at line 288 of file intr.h.

#### 8.87.2.9 rtems_interrupt_lock_destroy

```
#define rtems_interrupt_lock_destroy(
              _lock ) _ISR_lock_Destroy( _lock )
```

%

**Parameters**

| | |
|---|---|
| *_lock* | % |

Definition at line 300 of file intr.h.

#### 8.87.2.10 rtems_interrupt_lock_initialize

```
#define rtems_interrupt_lock_initialize(
              _lock,
              _name )  _ISR_lock_Initialize( _lock, _name )
```

%

**Parameters**

| | |
|---|---|
| *_lock* | % |
| *_name* | % |

Definition at line 313 of file intr.h.

#### 8.87.2.11 RTEMS_INTERRUPT_LOCK_INITIALIZER

```
#define RTEMS_INTERRUPT_LOCK_INITIALIZER(
              _name ) ISR_LOCK_INITIALIZER( _name )
```

%

**Parameters**

| | |
|---|---|
| *_name* | % |

Definition at line 325 of file intr.h.

#### 8.87.2.12 rtems_interrupt_lock_interrupt_disable

```
#define rtems_interrupt_lock_interrupt_disable(
              _lock_context )  _ISR_lock_ISR_disable( _lock_context )
```

%

**Parameters**

| | |
|---|---|
| *_lock_context* | % |

Definition at line 336 of file intr.h.

#### 8.87.2.13 RTEMS_INTERRUPT_LOCK_MEMBER

```
#define RTEMS_INTERRUPT_LOCK_MEMBER(
              _designator )  ISR_LOCK_MEMBER( _designator )
```

%

**Parameters**

| | |
|---|---|
| *_designator* | % |

Definition at line 348 of file intr.h.

#### 8.87.2.14 RTEMS_INTERRUPT_LOCK_REFERENCE

```
#define RTEMS_INTERRUPT_LOCK_REFERENCE(
              _designator,
              _target )  ISR_LOCK_REFERENCE( _designator, _target )
```

%

**Parameters**

| | |
|---|---|
| *_designator* | % |
| *_target* | % |

Definition at line 362 of file intr.h.

### 8.87.2.15 rtems_interrupt_lock_release

```
#define rtems_interrupt_lock_release(
              _lock,
              _lock_context )  _ISR_lock_Release_and_ISR_enable( _lock, _lock_context )
```

%

**Parameters**

| | |
|---|---|
| *_lock* | % |
| *_lock_context* | % |

Definition at line 376 of file intr.h.

### 8.87.2.16 rtems_interrupt_lock_release_isr

```
#define rtems_interrupt_lock_release_isr(
              _lock,
              _lock_context )
```

**Value:**

```
_SMP_lock_Release( \
  &( _lock )->Lock, \
  &( _lock_context )->Lock_context \
)
```

%

**Parameters**

| | |
|---|---|
| *_lock* | % |
| *_lock_context* | % |

Definition at line 391 of file intr.h.

## 8.87.3 Function Documentation

### 8.87.3.1 rtems_interrupt_catch()

```
rtems_status_code rtems_interrupt_catch (
            rtems_isr_entry new_isr_handler,
            rtems_vector_number vector,
            rtems_isr_entry * old_isr_handler )
```

%

**Parameters**

| | |
|---|---|
| *new_isr_handler* | % |
| *vector* | % |
| *old_isr_handler* | % |

# 8.88 Interrupt Manager Extension

## Files

- file irq-extension.h

  *Header file for the Interrupt Manager Extension.*

## Classes

- struct rtems_interrupt_server_action

  *An interrupt server action.*
- struct rtems_interrupt_server_control

  *An interrupt server control.*
- struct rtems_interrupt_server_config

  *An interrupt server configuration.*
- struct rtems_interrupt_server_entry

  *An interrupt server entry.*
- struct rtems_interrupt_server_request

  *An interrupt server request.*

## Macros

- #define RTEMS_INTERRUPT_UNIQUE ((rtems_option) 0x00000001)

  *Makes the interrupt handler unique. Prevents other handler from using the same interrupt vector.*
- #define RTEMS_INTERRUPT_SHARED ((rtems_option) 0x00000000)

  *Allows that this interrupt handler may share a common interrupt vector with other handler.*
- #define RTEMS_INTERRUPT_REPLACE ((rtems_option) 0x00000002)

  *Forces that this interrupt handler replaces the first handler with the same argument.*
- #define RTEMS_INTERRUPT_IS_UNIQUE(options) ((options) & RTEMS_INTERRUPT_UNIQUE)

  *Returns true if the interrupt handler unique option is set.*
- #define RTEMS_INTERRUPT_IS_SHARED(options) (!RTEMS_INTERRUPT_IS_UNIQUE( options))

  *Returns true if the interrupt handler shared option is set.*
- #define RTEMS_INTERRUPT_IS_REPLACE(options) ((options) & RTEMS_INTERRUPT_REPLACE)

  *Returns true if the interrupt handler replace option is set.*
- #define RTEMS_INTERRUPT_SERVER_DEFAULT 0

  *The interrupt server index of the default interrupt server.*

## Typedefs

- typedef void(∗ rtems_interrupt_handler) (void ∗)

  *Interrupt handler routine type.*
- typedef void(∗ rtems_interrupt_per_handler_routine) (void ∗, const char ∗, rtems_option, rtems_interrupt_handler, void ∗)

  *Interrupt handler iteration routine type.*
- typedef struct rtems_interrupt_server_action rtems_interrupt_server_action

  *An interrupt server action.*
- typedef struct rtems_interrupt_server_control rtems_interrupt_server_control

  *An interrupt server control.*

## Functions

- rtems_status_code rtems_interrupt_handler_install (rtems_vector_number vector, const char *info, rtems_option options, rtems_interrupt_handler handler, void *arg)

  *Installs the interrupt handler routine handler for the interrupt vector with number vector.*
- rtems_status_code rtems_interrupt_handler_remove (rtems_vector_number vector, rtems_interrupt_handler handler, void *arg)

  *Removes the interrupt handler routine handler with argument arg for the interrupt vector with number vector.*
- rtems_status_code rtems_interrupt_handler_iterate (rtems_vector_number vector, rtems_interrupt_per_handler_routine routine, void *arg)

  *Iterates over all installed interrupt handler of the interrupt vector with number vector.*
- rtems_status_code rtems_interrupt_set_affinity (rtems_vector_number vector, size_t affinity_size, const cpu_set_t *affinity)

  *Sets the processor affinity set of an interrupt vector.*
- rtems_status_code rtems_interrupt_get_affinity (rtems_vector_number vector, size_t affinity_size, cpu_set_t *affinity)

  *Gets the processor affinity set of an interrupt vector.*
- rtems_status_code rtems_interrupt_server_initialize (rtems_task_priority priority, size_t stack_size, rtems_mode modes, rtems_attribute attributes, uint32_t *server_count)

  *Initializes the interrupt server tasks.*
- rtems_status_code rtems_interrupt_server_create (rtems_interrupt_server_control *control, const rtems_interrupt_server_config *config, uint32_t *server_index)

  *Creates an interrupt server.*
- rtems_status_code rtems_interrupt_server_delete (uint32_t server_index)

  *Destroys the interrupt server.*
- rtems_status_code rtems_interrupt_server_handler_install (uint32_t server_index, rtems_vector_number vector, const char *info, rtems_option options, rtems_interrupt_handler handler, void *arg)

  *Installs the interrupt handler routine handler for the interrupt vector with number vector on the server server.*
- rtems_status_code rtems_interrupt_server_handler_remove (uint32_t server_index, rtems_vector_number vector, rtems_interrupt_handler handler, void *arg)

  *Removes the interrupt handler routine handler with argument arg for the interrupt vector with number vector from the server server.*
- rtems_status_code rtems_interrupt_server_handler_iterate (uint32_t server_index, rtems_vector_number vector, rtems_interrupt_per_handler_routine routine, void *arg)

  *Iterates over all interrupt handler of the interrupt vector with number vector which are installed on the interrupt server specified by server.*
- rtems_status_code rtems_interrupt_server_move (uint32_t source_server_index, rtems_vector_number vector, uint32_t destination_server_index)

  *Moves the interrupt handlers installed on the specified source interrupt server to the destination interrupt server.*
- rtems_status_code rtems_interrupt_server_suspend (uint32_t server_index)

  *Suspends the specified interrupt server.*
- rtems_status_code rtems_interrupt_server_resume (uint32_t server_index)

  *Resumes the specified interrupt server.*
- rtems_status_code rtems_interrupt_server_set_affinity (uint32_t server_index, size_t affinity_size, const cpu_set_t *affinity, rtems_task_priority priority)

  *Sets the processor affinity of the specified interrupt server.*
- rtems_status_code rtems_interrupt_server_entry_initialize (uint32_t server_index, rtems_interrupt_server_entry *entry)

  *Initializes the specified interrupt server entry.*
- void rtems_interrupt_server_action_prepend (rtems_interrupt_server_entry *entry, rtems_interrupt_server_action *action, rtems_interrupt_handler handler, void *arg)

  *Prepends the specified interrupt server action to the list of actions of the specified interrupt server entry.*
- void rtems_interrupt_server_entry_submit (rtems_interrupt_server_entry *entry)

*Submits the specified interrupt server entry so that its interrupt server actions can be invoked by the specified interrupt server.*

- rtems_status_code rtems_interrupt_server_entry_move (rtems_interrupt_server_entry *entry, uint32_t destination_server_index)

  *Moves the interrupt server entry to the specified destination interrupt server.*

- void rtems_interrupt_server_entry_destroy (rtems_interrupt_server_entry *entry)

  *Destroys the specified interrupt server entry.*

- rtems_status_code rtems_interrupt_server_request_initialize (uint32_t server_index, rtems_interrupt_server_request *request, rtems_interrupt_handler handler, void *arg)

  *Initializes the specified interrupt server request.*

- static __inline__ void rtems_interrupt_server_request_set_vector (rtems_interrupt_server_request *request, rtems_vector_number vector)

  *Sets the interrupt vector in the specified interrupt server request.*

- static __inline__ void rtems_interrupt_server_request_submit (rtems_interrupt_server_request *request)

  *Submits the specified interrupt server request so that its interrupt server action can be invoked by the specified interrupt server.*

- static __inline__ void rtems_interrupt_server_request_destroy (rtems_interrupt_server_request *request)

  *Destroys the specified interrupt server request.*

## 8.88.1 Detailed Description

In addition to the Classic API interrupt handler with a handle are supported. You can also install multiple shared handler for one interrupt vector.

## 8.88.2 Typedef Documentation

### 8.88.2.1 rtems_interrupt_per_handler_routine

```
typedef void(* rtems_interrupt_per_handler_routine) (void *, const char *, rtems_option, rtems_interrupt_handl
void *)
```

Interrupt handler iteration routine type.

**See also**

rtems_interrupt_handler_iterate()

Definition at line 169 of file irq-extension.h.

#### 8.88.2.2 rtems_interrupt_server_action

```
typedef struct rtems_interrupt_server_action rtems_interrupt_server_action
```

An interrupt server action.

This structure must be treated as an opaque data type. Members must not be accessed directly.

**See also**

rtems_interrupt_server_action_prepend().

#### 8.88.2.3 rtems_interrupt_server_control

```
typedef struct rtems_interrupt_server_control rtems_interrupt_server_control
```

An interrupt server control.

This structure must be treated as an opaque data type. Members must not be accessed directly.

**See also**

rtems_interrupt_server_create()

### 8.88.3 Function Documentation

#### 8.88.3.1 rtems_interrupt_get_affinity()

```
rtems_status_code rtems_interrupt_get_affinity (
            rtems_vector_number vector,
            size_t affinity_size,
            cpu_set_t * affinity )
```

Gets the processor affinity set of an interrupt vector.

**Parameters**

| | | |
|---|---|---|
| in | *vector* | The interrupt vector number. |
| in | *affinity_size* | The storage size of the affinity set. |
| out | *affinity_set* | The current processor affinity set for the interrupt vector. This pointer must not be NULL. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation. |
| *RTEMS_INVALID_ID* | The vector number is invalid. |

**Return values**

| | |
|---|---|
| *RTEMS_INVALID_SIZE* | Invalid affinity set size. |

Definition at line 592 of file irq-generic.c.

### 8.88.3.2 rtems_interrupt_handler_install()

```
rtems_status_code rtems_interrupt_handler_install (
            rtems_vector_number vector,
            const char * info,
            rtems_option options,
            rtems_interrupt_handler handler,
            void * arg )
```

Installs the interrupt handler routine *handler* for the interrupt vector with number *vector*.

You can set one of the mutually exclusive options

- RTEMS_INTERRUPT_UNIQUE
- RTEMS_INTERRUPT_SHARED
- RTEMS_INTERRUPT_REPLACE

with the *options* parameter for the interrupt handler.

The handler routine shall be called with argument *arg* when dispatched. The order in which the shared interrupt handlers are dispatched for one vector is BSP dependent.

If the option RTEMS_INTERRUPT_UNIQUE is set then it shall be ensured that this handler will be the only one for this vector.

If the option RTEMS_INTERRUPT_REPLACE is set then it shall be ensured that this handler will replace the first handler with the same argument for this vector if it exists, otherwise an error status shall be returned. A second handler with the same argument for this vector shall remain unchanged. The new handler will inherit the unique or shared option from the replaced handler.

You can provide an informative description *info*. This may be used for system debugging and status tools. The string has to be persistent during the handler life time.

This function may block.

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation. |
| *RTEMS_CALLED_FROM_ISR* | If this function is called from interrupt context this shall be returned. |
| *RTEMS_INVALID_ADDRESS* | If the handler address is NULL this shall be returned. |
| *RTEMS_INVALID_ID* | If the vector number is out of range this shall be returned. |
| *RTEMS_INVALID_NUMBER* | If an option is not applicable this shall be returned. |
| *RTEMS_RESOURCE_IN_USE* | If the vector is already occupied with a unique handler this shall be returned. If a unique handler should be installed and there is already a handler |
| Generated by Doxygen | installed this shall be returned. |
| *RTEMS_TOO_MANY* | If a handler with this argument is already installed for the vector this shall be returned. |
| *RTEMS_UNSATISFIED* | If no handler exists to replace with the specified argument and vector this shall be returned. |

Definition at line 520 of file irq-generic.c.

#### 8.88.3.3 rtems_interrupt_handler_iterate()

```
rtems_status_code rtems_interrupt_handler_iterate (
            rtems_vector_number vector,
            rtems_interrupt_per_handler_routine routine,
            void * arg )
```

Iterates over all installed interrupt handler of the interrupt vector with number *vector*.

For each installed handler of the vector the function *routine* will be called with the supplied argument *arg* and the handler information, options, routine and argument.

This function is intended for system information and diagnostics.

This function may block. Never install or remove an interrupt handler within the iteration routine. This may result in a deadlock.

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation. |
| *RTEMS_CALLED_FROM_ISR* | If this function is called from interrupt context this shall be returned. |
| *RTEMS_INVALID_ID* | If the vector number is out of range this shall be returned. |
| *RTEMS_IO_ERROR* | Reserved for board support package specific error conditions. |

Definition at line 540 of file irq-generic.c.

#### 8.88.3.4 rtems_interrupt_handler_remove()

```
rtems_status_code rtems_interrupt_handler_remove (
            rtems_vector_number vector,
            rtems_interrupt_handler handler,
            void * arg )
```

Removes the interrupt handler routine *handler* with argument *arg* for the interrupt vector with number *vector*.

This function may block.

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation. |
| *RTEMS_CALLED_FROM_ISR* | If this function is called from interrupt context this shall be returned. |
| *RTEMS_INVALID_ADDRESS* | If the handler address is NULL this shall be returned. |
| *RTEMS_INVALID_ID* | If the vector number is out of range this shall be returned. |
| *RTEMS_UNSATISFIED* | If the handler with its argument is not installed for the vector this shall be returned. |
| *RTEMS_IO_ERROR* | Reserved for board support package specific error conditions. |

Definition at line 531 of file irq-generic.c.

#### 8.88.3.5 rtems_interrupt_server_action_prepend()

```
void rtems_interrupt_server_action_prepend (
            rtems_interrupt_server_entry * entry,
            rtems_interrupt_server_action * action,
            rtems_interrupt_handler handler,
            void * arg )
```

Prepends the specified interrupt server action to the list of actions of the specified interrupt server entry.

No error checking is performed.

**Parameters**

| | | |
|---|---|---|
| in | *entry* | The interrupt server entry to prepend the interrupt server action. It must have been initialized via rtems_interrupt_server_entry_initialize(). |
| in | *action* | The interrupt server action to prepend the list of actions of the entry. |
| in | *handler* | The interrupt handler for the action. |
| in | *arg* | The interrupt handler argument for the action. |

#### 8.88.3.6 rtems_interrupt_server_create()

```
rtems_status_code rtems_interrupt_server_create (
            rtems_interrupt_server_control * control,
            const rtems_interrupt_server_config * config,
            uint32_t * server_index )
```

Creates an interrupt server.

This function may block.

**Parameters**

| | | |
|---|---|---|
| out | *control* | is the interrupt server control. The ownership of this structure is transferred from the caller of this function to the interrupt server management. |
| | *config* | is the interrupt server configuration. |
| out | *server_index* | is the pointer to a server index variable. The index of the built interrupt server will be stored in the referenced variable if the operation was successful. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The operation was successful. |

**Returns**

The function uses rtems_task_create(). If this operation is not successful, then its status code is returned.

**See also**

rtems_interrupt_server_initialize() and rtems_interrupt_server_delete().

### 8.88.3.7 rtems_interrupt_server_delete()

```
rtems_status_code rtems_interrupt_server_delete (
            uint32_t server_index )
```

Destroys the interrupt server.

This function may block.

The interrupt server deletes itself, so after the return of the function the interrupt server may be still in the termination process depending on the task priorities of the system.

**Parameters**

| | |
|---|---|
| *server_index* | is the index of the interrupt server to destroy. Use RTEMS_INTERRUPT_SERVER_DEFAULT to specify the default server. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The operation was successful. |
| *RTEMS_INVALID_ID* | The interrupt server index was invalid. |

**See also**

rtems_interrupt_server_create()

### 8.88.3.8 rtems_interrupt_server_entry_destroy()

```
void rtems_interrupt_server_entry_destroy (
            rtems_interrupt_server_entry * entry )
```

Destroys the specified interrupt server entry.

This function must be called from thread context. It may block. Calling this function within the context of an interrupt server is undefined behaviour. No error checking is performed.

**Parameters**

| | | |
|---|---|---|
| in | *entry* | The interrupt server entry to destroy. It must have been initialized via rtems_interrupt_server_entry_initialize(). |

### 8.88.3.9 rtems_interrupt_server_entry_initialize()

```
rtems_status_code rtems_interrupt_server_entry_initialize (
            uint32_t server_index,
            rtems_interrupt_server_entry * entry )
```

Initializes the specified interrupt server entry.

**Parameters**

| | | |
|---|---|---|
| in | *server_index* | The interrupt server index. Use RTEMS_INTERRUPT_SERVER_DEFAULT to specify the default server. |
| in | *entry* | The interrupt server entry to initialize. |

**See also**

rtems_interrupt_server_action_prepend().

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation. |
| *RTEMS_INCORRECT_STATE* | The interrupt servers are not initialized. |
| *RTEMS_INVALID_ID* | If the interrupt server index is invalid. |

### 8.88.3.10 rtems_interrupt_server_entry_move()

```
rtems_status_code rtems_interrupt_server_entry_move (
            rtems_interrupt_server_entry * entry,
            uint32_t destination_server_index )
```

Moves the interrupt server entry to the specified destination interrupt server.

Calling this function concurrently with rtems_interrupt_server_entry_submit() with the same entry or while the entry is enqueued on the previous interrupt server is undefined behaviour.

**Parameters**

| | | |
|---|---|---|
| in,out | *entry* | The interrupt server entry. It must have be initialized before the call to this function. |
| | *destination_server_index* | The destination interrupt server index. Use RTEMS_INTERRUPT_SERVER_DEFAULT to specify the default server. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation |
| *RTEMS_INCORRECT_STATE* | The interrupt servers are not initialized. |
| *RTEMS_INVALID_ID* | The destination interrupt server index is invalid. |

#### 8.88.3.11 rtems_interrupt_server_entry_submit()

```
void rtems_interrupt_server_entry_submit (
            rtems_interrupt_server_entry * entry )
```

Submits the specified interrupt server entry so that its interrupt server actions can be invoked by the specified interrupt server.

This function may be used to do a two-step interrupt processing. The first step is done in interrupt context which calls this function. The second step is then done in the context of the interrupt server.

This function may be called from thread or interrupt context. It does not block. No error checking is performed.

**Parameters**

| | | |
|---|---|---|
| in | *entry* | The interrupt server entry must be initialized before the first call to this function via rtems_interrupt_server_entry_initialize() and rtems_interrupt_server_action_prepend(). The entry and its actions must not be modified between calls to this function. Use rtems_interrupt_server_entry_destroy() to destroy an entry in use. |

#### 8.88.3.12 rtems_interrupt_server_handler_install()

```
rtems_status_code rtems_interrupt_server_handler_install (
            uint32_t server_index,
            rtems_vector_number vector,
            const char * info,
            rtems_option options,
            rtems_interrupt_handler handler,
            void * arg )
```

Installs the interrupt handler routine *handler* for the interrupt vector with number *vector* on the server *server*.

The handler routine will be executed on the corresponding interrupt server task. A server index *server_index* of RTEMS_INTERRUPT_SERVER_DEFAULT may be used to install the handler on the default server.

This function may block.

**See also**

rtems_interrupt_handler_install().

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation. |
| *RTEMS_INCORRECT_STATE* | The interrupt servers are not initialized. |
| *RTEMS_INVALID_ID* | If the interrupt server index is invalid. |
| * | For other errors see rtems_interrupt_handler_install(). |

### 8.88.3.13 rtems_interrupt_server_handler_iterate()

```
rtems_status_code rtems_interrupt_server_handler_iterate (
            uint32_t server_index,
            rtems_vector_number vector,
            rtems_interrupt_per_handler_routine routine,
            void * arg )
```

Iterates over all interrupt handler of the interrupt vector with number *vector* which are installed on the interrupt server specified by *server*.

A server index *server_index* of `RTEMS_INTERRUPT_SERVER_DEFAULT` may be used to specify the default server.

**See also**

rtems_interrupt_handler_iterate()

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation. |
| *RTEMS_INCORRECT_STATE* | The interrupt servers are not initialized. |
| *RTEMS_INVALID_ID* | If the interrupt server index is invalid. |
| * | For other errors see rtems_interrupt_handler_iterate(). |

### 8.88.3.14 rtems_interrupt_server_handler_remove()

```
rtems_status_code rtems_interrupt_server_handler_remove (
            uint32_t server_index,
            rtems_vector_number vector,
            rtems_interrupt_handler handler,
            void * arg )
```

Removes the interrupt handler routine *handler* with argument *arg* for the interrupt vector with number *vector* from the server *server*.

A server index *server_index* of `RTEMS_INTERRUPT_SERVER_DEFAULT` may be used to remove the handler from the default server.

This function may block.

**See also**

rtems_interrupt_handler_remove().

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation. |
| *RTEMS_INCORRECT_STATE* | The interrupt servers are not initialized. |
| *RTEMS_INVALID_ID* | If the interrupt server index is invalid. |
| * | For other errors see rtems_interrupt_handler_remove(). |

### 8.88.3.15 rtems_interrupt_server_initialize()

```
rtems_status_code rtems_interrupt_server_initialize (
            rtems_task_priority priority,
            size_t stack_size,
            rtems_mode modes,
            rtems_attribute attributes,
            uint32_t * server_count )
```

Initializes the interrupt server tasks.

This function tries to create an interrupt server task for each processor in the system. The tasks will have the priority *priority*, the stack size *stack_size*, the modes *modes* and the attributes *attributes*. The count of server tasks will be returned in *server_count*. Interrupt handlers can be installed on an interrupt server with rtems_interrupt_server_handler_install() and removed with rtems_interrupt_server_handler_remove() using a server index. In case of an interrupt, the request will be forwarded to the interrupt server. The handlers are executed within the interrupt server context. If one handler blocks on something this may delay the processing of other handlers.

The server count pointer *server_count* may be *NULL*.

The task name of interrupt servers created by this function is rtems_build_name( 'I', 'R', 'Q', 'S' ).

This function may block.

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The operation was successful. |
| *RTEMS_INCORRECT_STATE* | The interrupt servers were already initialized. |

**Returns**

The function uses rtems_task_create(). If this operation is not successful, then its status code is returned.

**See also**

rtems_interrupt_server_create() and rtems_interrupt_server_delete().

#### 8.88.3.16 rtems_interrupt_server_move()

```
rtems_status_code rtems_interrupt_server_move (
            uint32_t source_server_index,
            rtems_vector_number vector,
            uint32_t destination_server_index )
```

Moves the interrupt handlers installed on the specified source interrupt server to the destination interrupt server.

This function must be called from thread context. It may block. Calling this function within the context of an interrupt server is undefined behaviour.

**Parameters**

| | | |
|---|---|---|
| in | *source_server_index* | The source interrupt server index. Use RTEMS_INTERRUPT_SERVER_DEFAULT to specify the default server. |
| in | *vector* | The interrupt vector number. |
| in | *destination_server_index* | The destination interrupt server index. Use RTEMS_INTERRUPT_SERVER_DEFAULT to specify the default server. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation |
| *RTEMS_INCORRECT_STATE* | The interrupt servers are not initialized. |
| *RTEMS_INVALID_ID* | The destination interrupt server index is invalid. |
| *RTEMS_INVALID_ID* | The vector number is invalid. |
| *RTEMS_INVALID_ID* | The destination interrupt server index is invalid. |

#### 8.88.3.17 rtems_interrupt_server_request_destroy()

```
static __inline__ void rtems_interrupt_server_request_destroy (
            rtems_interrupt_server_request * request )  [static]
```

Destroys the specified interrupt server request.

This function must be called from thread context. It may block. Calling this function within the context of an interrupt server is undefined behaviour. No error checking is performed.

**Parameters**

| | | |
|---|---|---|
| in | *request* | The interrupt server request to destroy. It must have been initialized via rtems_interrupt_server_request_initialize(). |

Definition at line 794 of file irq-extension.h.

#### 8.88.3.18 rtems_interrupt_server_request_initialize()

```
rtems_status_code rtems_interrupt_server_request_initialize (
            uint32_t server_index,
            rtems_interrupt_server_request * request,
            rtems_interrupt_handler handler,
            void * arg )
```

Initializes the specified interrupt server request.

**Parameters**

| | | |
|---|---|---|
| in | *server_index* | The interrupt server index. Use RTEMS_INTERRUPT_SERVER_DEFAULT to specify the default server. |
| in | *request* | The interrupt server request to initialize. |
| in | *handler* | The interrupt handler for the request action. |
| in | *arg* | The interrupt handler argument for the request action. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation. |
| *RTEMS_INCORRECT_STATE* | The interrupt servers are not initialized. |
| *RTEMS_INVALID_ID* | If the interrupt server index is invalid. |

**See also**

rtems_interrupt_server_request_set_vector().

#### 8.88.3.19 rtems_interrupt_server_request_set_vector()

```
static __inline__ void rtems_interrupt_server_request_set_vector (
            rtems_interrupt_server_request * request,
            rtems_vector_number vector )  [static]
```

Sets the interrupt vector in the specified interrupt server request.

By default, the interrupt vector of an interrupt server request is set to a special value which is outside the range of vectors supported by the interrupt controller hardware.

Calls to rtems_interrupt_server_request_submit() will disable the interrupt vector of the request. After processing of the request by the interrupt server the interrupt vector will be enabled again.

**Parameters**

| | | |
|---|---|---|
| in | *request* | The initialized interrupt server request. |
| in | *vector* | The interrupt vector number. |

**See also**

rtems_interrupt_server_request_initialize().

Definition at line 752 of file irq-extension.h.

#### 8.88.3.20 rtems_interrupt_server_request_submit()

```
static __inline__ void rtems_interrupt_server_request_submit (
            rtems_interrupt_server_request * request )  [static]
```

Submits the specified interrupt server request so that its interrupt server action can be invoked by the specified interrupt server.

This function may be used to do a two-step interrupt processing. The first step is done in interrupt context which calls this function. The second step is then done in the context of the interrupt server.

This function may be called from thread or interrupt context. It does not block. No error checking is performed.

**Parameters**

| | | |
|---|---|---|
| in | *request* | The interrupt server request must be initialized before the first call to this function via rtems_interrupt_server_request_initialize(). The request must not be modified between calls to this function. Use rtems_interrupt_server_request_destroy() to destroy a request in use. |

Definition at line 777 of file irq-extension.h.

#### 8.88.3.21 rtems_interrupt_server_resume()

```
rtems_status_code rtems_interrupt_server_resume (
            uint32_t server_index )
```

Resumes the specified interrupt server.

This function must be called from thread context. It may block. Calling this function within the context of an interrupt server is undefined behaviour.

**Parameters**

| | | |
|---|---|---|
| in | *server_index* | The interrupt server index. Use RTEMS_INTERRUPT_SERVER_DEFAULT to specify the default server. |

**See also**

rtems_interrupt_server_suspend().

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation |
| *RTEMS_INCORRECT_STATE* | The interrupt servers are not initialized. |
| *RTEMS_INVALID_ID* | If the interrupt server index is invalid. |

#### 8.88.3.22 rtems_interrupt_server_set_affinity()

```
rtems_status_code rtems_interrupt_server_set_affinity (
            uint32_t server_index,
            size_t affinity_size,
            const cpu_set_t * affinity,
            rtems_task_priority priority )
```

Sets the processor affinity of the specified interrupt server.

The scheduler is set determined by the highest numbered processor in the specified affinity set.

This operation is only reliable in case the specified interrupt was suspended via rtems_interrupt_server_suspend().

**Parameters**

| | | |
|---|---|---|
| in | *server_index* | The interrupt server index. Use RTEMS_INTERRUPT_SERVER_DEFAULT to specify the default server. |
| in | *affinity_size* | The storage size of the affinity set. |
| in | *affinity* | The desired processor affinity set for the specified interrupt server. |
| in | *priority* | The task priority with respect to the corresponding scheduler instance. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation |
| *RTEMS_INCORRECT_STATE* | The interrupt servers are not initialized. |
| *RTEMS_INVALID_ID* | If the interrupt server index is invalid. |
| *RTEMS_INVALID_SIZE* | Invalid affinity set size. |
| *RTEMS_INVALID_NAME* | The affinity set contains no online processor. |
| *RTEMS_INCORRECT_STATE* | The highest numbered online processor in the specified affinity set is not owned by a scheduler. |
| *RTEMS_INVALID_PRIORITY* | Invalid priority. |
| *RTEMS_RESOURCE_IN_USE* | The interrupt server owns resources which deny a scheduler change. |
| *RTEMS_INVALID_NUMBER* | Invalid processor affinity set. |

#### 8.88.3.23 rtems_interrupt_server_suspend()

```
rtems_status_code rtems_interrupt_server_suspend (
            uint32_t server_index )
```

Suspends the specified interrupt server.

A suspend request is sent to the specified interrupt server. This function waits for an acknowledgment from the specified interrupt server.

This function must be called from thread context. It may block. Calling this function within the context of an interrupt server is undefined behaviour.

**Parameters**

| | | |
|---|---|---|
| in | *server_index* | The interrupt server index. Use `RTEMS_INTERRUPT_SERVER_DEFAULT` to specify the default server. |

**See also**

rtems_interrupt_server_resume().

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation |
| *RTEMS_INCORRECT_STATE* | The interrupt servers are not initialized. |
| *RTEMS_INVALID_ID* | If the interrupt server index is invalid. |

#### 8.88.3.24 rtems_interrupt_set_affinity()

```
rtems_status_code rtems_interrupt_set_affinity (
            rtems_vector_number vector,
            size_t affinity_size,
            const cpu_set_t * affinity )
```

Sets the processor affinity set of an interrupt vector.

**Parameters**

| | | |
|---|---|---|
| in | *vector* | The interrupt vector number. |
| in | *affinity_size* | The storage size of the affinity set. |
| in | *affinity_set* | The new processor affinity set for the interrupt vector. This pointer must not be `NULL`. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation. |
| *RTEMS_INVALID_ID* | The vector number is invalid. |
| *RTEMS_INVALID_SIZE* | Invalid affinity set size. |
| *RTEMS_INVALID_NUMBER* | Invalid processor affinity set. |

Definition at line 568 of file irq-generic.c.

## 8.89 Kernel Print Support

This module contains all methods and support related to providing kernel level print support.

The following variables below are declared as extern and MUST be declared and initialized in each BSP. Using this indirect function, the functionality in this group is tailored for the BSP.

- BSP_output_char
- BSP_poll_char

# 8.90 LEON3 AMBA Driver Handler

AMBA Plag & Play Bus Driver Macros.

## Files

- file amba.h

## Macros

- #define **LEON3_IO_AREA** 0xfff00000
- #define **LEON3_CONF_AREA** 0xff000
- #define **LEON3_AHB_SLAVE_CONF_AREA** (1 << 11)
- #define **LEON3_AHB_CONF_WORDS** 8
- #define **LEON3_APB_CONF_WORDS** 2
- #define **LEON3_AHB_MASTERS** 64
- #define **LEON3_AHB_SLAVES** 64
- #define **LEON3_APB_SLAVES** 16

## Variables

- struct ambapp_bus **ambapp_plb**

### 8.90.1 Detailed Description

AMBA Plag & Play Bus Driver Macros.

## 8.91 LEON3 and LEON4

LEON3 and LEON4 Board Support Package.

### Modules

- LEON3 AMBA Driver Handler

  *AMBA Plag & Play Bus Driver Macros.*

### Files

- file bsp.h

  *Global BSP definitions.*

### Macros

- #define **LEON3** 1
- #define **BSP_IDLE_TASK_BODY** bsp_idle_thread
- #define **BSP_NUMBER_OF_TERMIOS_PORTS** 8
- #define **RTEMS_BSP_NETWORK_DRIVER_NAME_OPENETH** "open_eth1"
- #define **RTEMS_BSP_NETWORK_DRIVER_ATTACH_OPENETH** rtems_leon_open_eth_driver_attach
- #define **RTEMS_BSP_NETWORK_DRIVER_NAME_SMC91111** "smc_eth1"
- #define **RTEMS_BSP_NETWORK_DRIVER_ATTACH_SMC91111** rtems_smc91111_driver_attach_leon3
- #define **RTEMS_BSP_NETWORK_DRIVER_NAME_GRETH** "gr_eth1"
- #define **RTEMS_BSP_NETWORK_DRIVER_ATTACH_GRETH** rtems_leon_greth_driver_attach
- #define **RTEMS_BSP_NETWORK_DRIVER_NAME** RTEMS_BSP_NETWORK_DRIVER_NAME_GRETH
- #define **RTEMS_BSP_NETWORK_DRIVER_ATTACH** RTEMS_BSP_NETWORK_DRIVER_ATTACH_G↩ RETH
- #define **HAS_SMC91111**
- #define **GRETH_SUPPORTED**
- #define **GRETH_MEM_LOAD**(addr) leon_r32_no_cache((uintptr_t)addr)
- #define **AMBAPPBUS_INFO_AVAIL** /∗ AMBAPP Bus driver ∗/
- #define **APBUART_INFO_AVAIL** /∗ APBUART Console driver ∗/
- #define **GPTIMER_INFO_AVAIL** /∗ GPTIMER Timer driver ∗/
- #define **GRETH_INFO_AVAIL** /∗ GRETH Ethernet driver ∗/

### Typedefs

- typedef void(∗ **bsp_shared_isr**) (void ∗arg)

## Functions

- void ∗ **bsp_idle_thread** (uintptr_t ignored)
- int **rtems_leon_open_eth_driver_attach** (struct rtems_bsdnet_ifconfig ∗config, int attach)
- int **rtems_smc91111_driver_attach_leon3** (struct rtems_bsdnet_ifconfig ∗config, int attach)
- int **rtems_leon_greth_driver_attach** (struct rtems_bsdnet_ifconfig ∗config, int attach)
- rtems_isr_entry **set_vector** (rtems_isr_entry handler, rtems_vector_number vector, int type)
- void **BSP_fatal_exit** (uint32_t error)
- void **bsp_spurious_initialize** (void)
- void **rtems_bsp_delay** (int usecs)
- void **BSP_shared_interrupt_init** (void)
- void **bsp_isr_handler** (rtems_vector_number vector)
- static __inline__ int **BSP_shared_interrupt_register** (int irq, const char ∗info, bsp_shared_isr isr, void ∗arg)
- static __inline__ int **BSP_shared_interrupt_unregister** (int irq, bsp_shared_isr isr, void ∗arg)
- void **BSP_shared_interrupt_clear** (int irq)
- void **BSP_shared_interrupt_unmask** (int irq)
- void **BSP_shared_interrupt_mask** (int irq)

## Variables

- int **CPU_SPARC_HAS_SNOOPING**
- int **RAM_START**
- int **RAM_END**
- int **RAM_SIZE**
- int **PROM_START**
- int **PROM_END**
- int **PROM_SIZE**
- int **CLOCK_SPEED**
- int **end**
- const unsigned char **LEON3_mp_irq**
- const unsigned char **LEON3_irq_to_cpu** [32]

### 8.91.1 Detailed Description

LEON3 and LEON4 Board Support Package.

## 8.92 Legacy Benchmark Drivers

Legacy Benchmark Drivers.

Legacy Benchmark Drivers.

## 8.93 Local Packages

Local packages.

Local packages.

## 8.94 MP Packet Handler

MP Packte Handler.

### Files

- file mppkt.h

  *Specification for the Packet Handler.*

### Classes

- struct MP_packet_Prefix

### Macros

- #define MP_PACKET_CLASSES_FIRST MP_PACKET_MPCI_INTERNAL
- #define MP_PACKET_CLASSES_LAST MP_PACKET_SIGNAL
- #define MP_PACKET_MINIMUM_PACKET_SIZE 64
- #define MP_PACKET_MINIMUN_HETERO_CONVERSION ( sizeof( MP_packet_Prefix ) / sizeof( uint32_t ) )

### Enumerations

- enum MP_packet_Classes {
  **MP_PACKET_MPCI_INTERNAL** = 0, **MP_PACKET_TASKS** = 1, **MP_PACKET_MESSAGE_QUEUE** = 2, **MP_PACKET_SEMAPHORE** = 3,
  **MP_PACKET_PARTITION** = 4, **MP_PACKET_REGION** = 5, **MP_PACKET_EVENT** = 6, **MP_PACKET_SIGNAL** = 7 }

### 8.94.1 Detailed Description

MP Packte Handler.

This handler encapsulates the primary definition of MPCI packets. This handler defines the part of the packet that is common to all remote operations.

### 8.94.2 Macro Definition Documentation

#### 8.94.2.1 MP_PACKET_CLASSES_FIRST

```
#define MP_PACKET_CLASSES_FIRST MP_PACKET_MPCI_INTERNAL
```

This constant defines the first entry in the MP_packet_Classes enumeration.

Definition at line 70 of file mppkt.h.

#### 8.94.2.2 MP_PACKET_CLASSES_LAST

```
#define MP_PACKET_CLASSES_LAST MP_PACKET_SIGNAL
```

This constant defines the last entry in the MP_packet_Classes enumeration.

Definition at line 75 of file mppkt.h.

#### 8.94.2.3 MP_PACKET_MINIMUM_PACKET_SIZE

```
#define MP_PACKET_MINIMUM_PACKET_SIZE 64
```

An MPCI must support packets of at least this size.

Definition at line 107 of file mppkt.h.

#### 8.94.2.4 MP_PACKET_MINIMUN_HETERO_CONVERSION

```
#define MP_PACKET_MINIMUN_HETERO_CONVERSION  ( sizeof( MP_packet_Prefix ) / sizeof( uint32_t )
)
```

The following constant defines the number of uint32_t's in a packet which must be converted to native format in a heterogeneous system. In packets longer than MP_PACKET_MINIMUN_HETERO_CONVERSION uint32_t's, some of the "extra" data may a user message buffer which is not automatically endian swapped.

Definition at line 116 of file mppkt.h.

### 8.94.3 Enumeration Type Documentation

#### 8.94.3.1 MP_packet_Classes

```
enum MP_packet_Classes
```

The following enumerated type defines the packet classes.

**Note**

In general, each class corresponds to a manager which supports global operations. Each manager defines the set of supported operations.

Definition at line 56 of file mppkt.h.

## 8.95 Malloc Support

RTEMS extensions to the Malloc Family.

### Files

- file mallocinitone.c

  *_Malloc_Initialize() Implementation*
- file mallocinitone.h

  *Malloc Support Initialization API.*
- file mallocheap.c

  *This source file provides the C Program Heap control along with the system initialization handler.*

### Functions

- static __inline__ Heap_Control * **_Malloc_Initialize_with_one_area** (Heap_Control *heap)

### 8.95.1 Detailed Description

RTEMS extensions to the Malloc Family.

## 8.96 Memory Area Checks

Checks for memory areas.

### Macros

- #define **T_eq_mem**(a, e, n) T_flags_eq_mem(a, e, n, 0, #a, #e, #n)
- #define **T_assert_eq_mem**(a, e, n) T_flags_eq_mem(a, e, n, T_CHECK_STOP, #a, #e, #n)
- #define **T_quiet_eq_mem**(a, e, n) T_flags_eq_mem(a, e, n, T_CHECK_QUIET, #a, #e, #n)
- #define **T_step_eq_mem**(s, a, e, n) T_flags_eq_mem(a, e, n, T_CHECK_STEP(s), #a, #e, #n)
- #define **T_step_assert_eq_mem**(s, a, e, n) T_flags_eq_mem(a, e, n, T_CHECK_STEP(s) | T_CHECK_↩ STOP, #a, #e, #n)
- #define **T_ne_mem**(a, e, n) T_flags_ne_mem(a, e, n, 0, #a, #e, #n)
- #define **T_assert_ne_mem**(a, e, n) T_flags_ne_mem(a, e, n, T_CHECK_STOP, #a, #e, #n)
- #define **T_quiet_ne_mem**(a, e, n) T_flags_ne_mem(a, e, n, T_CHECK_QUIET, #a, #e, #n)
- #define **T_step_ne_mem**(s, a, e, n) T_flags_ne_mem(a, e, n, T_CHECK_STEP(s), #a, #e, #n)
- #define **T_step_assert_ne_mem**(s, a, e, n) T_flags_ne_mem(a, e, n, T_CHECK_STEP(s) | T_CHECK_↩ STOP, #a, #e, #n)

### 8.96.1 Detailed Description

Checks for memory areas.

## 8.97 Memory Handler

Low level handler to provide memory areas for higher level memory handlers such as the Workspace Handler.

### Files

- file memory.h

  *Memory Handler API.*

### Classes

- struct Memory_Area

  *The memory area description.*
- struct Memory_Information

  *The memory information.*

### Macros

- #define MEMORY_INFORMATION_INITIALIZER(areas) { RTEMS_ARRAY_SIZE( areas ), ( areas ) }

  *Statically initialize a memory information.*
- #define MEMORY_INITIALIZER(begin, end) { ( begin ), ( begin ), ( end ) }

  *Statically initialize a memory area.*

### Functions

- static __inline__ size_t _Memory_Get_count (const Memory_Information *information)

  *Get the memory area count.*
- static __inline__ Memory_Area * _Memory_Get_area (const Memory_Information *information, size_t index)

  *Get a memory area by index.*
- static __inline__ void _Memory_Initialize (Memory_Area *area, void *begin, void *end)

  *Initialize the memory area.*
- static __inline__ void _Memory_Initialize_by_size (Memory_Area *area, void *begin, uintptr_t size)

  *Initialize the memory area by size.*
- static __inline__ const void * _Memory_Get_begin (const Memory_Area *area)

  *Get the memory area begin.*
- static __inline__ void _Memory_Set_begin (Memory_Area *area, const void *begin)

  *Set the memory area begin.*
- static __inline__ const void * _Memory_Get_end (const Memory_Area *area)

  *Get the memory area end.*
- static __inline__ void _Memory_Set_end (Memory_Area *area, const void *end)

  *Set the memory area end.*
- static __inline__ uintptr_t _Memory_Get_size (const Memory_Area *area)

  *Get the memory area size.*
- static __inline__ void * _Memory_Get_free_begin (const Memory_Area *area)

  *Get the begin of the free area of the memory area.*
- static __inline__ void _Memory_Set_free_begin (Memory_Area *area, void *begin)

  *Set the begin of the free area of the memory area.*

- static __inline__ uintptr_t _Memory_Get_free_size (const Memory_Area *area)

  *Get the size of the free memory area of the memory area.*
- static __inline__ void _Memory_Consume (Memory_Area *area, uintptr_t consume)

  *Consume the specified size from the free memory area of the memory area.*
- const Memory_Information * _Memory_Get (void)

  *Return the memory information of this platform.*
- void * _Memory_Allocate (const Memory_Information *information, uintptr_t size, uintptr_t alignment)

  *Allocate a memory area from the memory information.*
- void _Memory_Fill (const Memory_Information *information, int c)

  *Fill all free memory areas of the memory information with a constant byte.*
- void _Memory_Zero_free_areas (void)

  *Zeros all free memory areas of the system.*
- void _Memory_Dirty_free_areas (void)

  *Dirty all free memory areas of the system.*

## Variables

- const bool _Memory_Zero_before_use

  *Indicates if the memory is zeroed during system initialization.*

## 8.97.1 Detailed Description

Low level handler to provide memory areas for higher level memory handlers such as the Workspace Handler.

## 8.97.2 Macro Definition Documentation

### 8.97.2.1 MEMORY_INFORMATION_INITIALIZER

```
#define MEMORY_INFORMATION_INITIALIZER(
              areas )  { RTEMS_ARRAY_SIZE( areas ), ( areas ) }
```

Statically initialize a memory information.

**Parameters**

| | |
|---|---|
| *areas* | The designator of an array of the memory areas. |

Definition at line 97 of file memory.h.

### 8.97.2.2 MEMORY_INITIALIZER

```
#define MEMORY_INITIALIZER(
```

```
        begin,
        end ) { ( begin ), ( begin ), ( end ) }
```

Statically initialize a memory area.

**Parameters**

| | |
|---|---|
| *begin* | The begin of the memory area. |
| *end* | The end of the memory area. |

Definition at line 106 of file memory.h.

## 8.97.3 Function Documentation

### 8.97.3.1 _Memory_Allocate()

```
void* _Memory_Allocate (
            const Memory_Information * information,
            uintptr_t size,
            uintptr_t alignment )
```

Allocate a memory area from the memory information.

It is not possible to free the memory area allocated by this function.

**Parameters**

| | |
|---|---|
| *information* | The memory information. |
| *size* | The size in bytes of the memory area to allocate. |
| *alignment* | The alignment in bytes of the memory area to allocate. It must be a power of two. |

**Return values**

| | |
|---|---|
| *NULL* | No such memory area available. |
| *begin* | The begin of the allocated memory area. |

Definition at line 34 of file memoryallocate.c.

### 8.97.3.2 _Memory_Consume()

```
static __inline__ void _Memory_Consume (
            Memory_Area * area,
            uintptr_t consume )  [static]
```

Consume the specified size from the free memory area of the memory area.

**Parameters**

| | |
|---|---|
| *area* | The memory area. |
| *consume* | The bytes to consume from the free memory area of the memory area. |

Definition at line 285 of file memory.h.

#### 8.97.3.3 _Memory_Fill()

```
void _Memory_Fill (
            const Memory_Information * information,
            int c )
```

Fill all free memory areas of the memory information with a constant byte.

**Parameters**

| | |
|---|---|
| *information* | The memory information. |
| *c* | The constant byte to fill the free memory areas. |

#### 8.97.3.4 _Memory_Get()

```
const Memory_Information* _Memory_Get (
            void  )
```

Return the memory information of this platform.

This function is provided by the Board Support Package (BSP). Using a function gives the BSPs a bit more freedom with respect to the implementation. Calling this function shall not have side-effects. Initialization steps to set up the memory information shall be done in a system initialization handler (RTEMS_SYSINIT_MEMORY).

**Returns**

The memory information.

Definition at line 42 of file bspgetworkarea.c.

#### 8.97.3.5 _Memory_Get_area()

```
static __inline__ Memory_Area* _Memory_Get_area (
            const Memory_Information * information,
            size_t index )  [static]
```

Get a memory area by index.

**Parameters**

| | |
|---|---|
| *information* | The memory information. |
| *index* | The index of the memory area to return. |

**Returns**

The memory area of the specified index.

Definition at line 130 of file memory.h.

#### 8.97.3.6 _Memory_Get_begin()

```
static __inline__ const void* _Memory_Get_begin (
            const Memory_Area * area )  [static]
```

Get the memory area begin.

**Parameters**

| | |
|---|---|
| *area* | The memory area. |

**Returns**

The memory area begin.

Definition at line 182 of file memory.h.

#### 8.97.3.7 _Memory_Get_count()

```
static __inline__ size_t _Memory_Get_count (
            const Memory_Information * information )  [static]
```

Get the memory area count.

**Parameters**

| | |
|---|---|
| *information* | The memory information. |

**Returns**

The memory area count.

Definition at line 115 of file memory.h.

#### 8.97.3.8 _Memory_Get_end()

```
static __inline__ const void* _Memory_Get_end (
            const Memory_Area * area )  [static]
```

Get the memory area end.

**Parameters**

| | |
|---|---|
| *area* | The memory area. |

**Returns**

The memory area end.

Definition at line 208 of file memory.h.

#### 8.97.3.9 _Memory_Get_free_begin()

```
static __inline__ void* _Memory_Get_free_begin (
            const Memory_Area * area )  [static]
```

Get the begin of the free area of the memory area.

**Parameters**

| | |
|---|---|
| *area* | The memory area. |

**Returns**

The free memory area begin the memory area.

Definition at line 246 of file memory.h.

#### 8.97.3.10 _Memory_Get_free_size()

```
static __inline__ uintptr_t _Memory_Get_free_size (
            const Memory_Area * area )  [static]
```

Get the size of the free memory area of the memory area.

**Parameters**

| | |
|---|---|
| *area* | The memory area. |

**Returns**

The free memory area size in bytes of the memory area.

Definition at line 272 of file memory.h.

#### 8.97.3.11 _Memory_Get_size()

```
static __inline__ uintptr_t _Memory_Get_size (
            const Memory_Area * area )  [static]
```

Get the memory area size.

**Parameters**

| | |
|---|---|
| *area* | The memory area. |

**Returns**

The memory area size in bytes.

Definition at line 234 of file memory.h.

#### 8.97.3.12 _Memory_Initialize()

```
static __inline__ void _Memory_Initialize (
            Memory_Area * area,
            void * begin,
            void * end )  [static]
```

Initialize the memory area.

**Parameters**

| | |
|---|---|
| *area* | The memory area. |
| *begin* | The begin of the memory area. |
| *end* | The end of the memory area. |

Definition at line 146 of file memory.h.

#### 8.97.3.13 _Memory_Initialize_by_size()

```
static __inline__ void _Memory_Initialize_by_size (
            Memory_Area * area,
```

```
            void * begin,
            uintptr_t size )  [static]
```

Initialize the memory area by size.

**Parameters**

| | |
|---|---|
| *area* | The memory area. |
| *begin* | The begin of the memory area. |
| *size* | The size of the memory area in bytes. |

Definition at line 164 of file memory.h.

#### 8.97.3.14 _Memory_Set_begin()

```
static __inline__ void _Memory_Set_begin (
            Memory_Area * area,
            const void * begin )  [static]
```

Set the memory area begin.

**Parameters**

| | |
|---|---|
| *area* | The memory area. |
| *begin* | The memory area begin. |

Definition at line 193 of file memory.h.

#### 8.97.3.15 _Memory_Set_end()

```
static __inline__ void _Memory_Set_end (
            Memory_Area * area,
            const void * end )  [static]
```

Set the memory area end.

**Parameters**

| | |
|---|---|
| *area* | The memory area. |
| *end* | The memory area end. |

Definition at line 219 of file memory.h.

#### 8.97.3.16 _Memory_Set_free_begin()

```
static __inline__ void _Memory_Set_free_begin (
            Memory_Area * area,
            void * begin )  [static]
```

Set the begin of the free area of the memory area.

**Parameters**

| | |
|---|---|
| *area* | The memory area. |
| *begin* | The free memory area begin the memory area. |

Definition at line 257 of file memory.h.

### 8.97.4 Variable Documentation

#### 8.97.4.1 _Memory_Zero_before_use

```
const bool _Memory_Zero_before_use  [extern]
```

Indicates if the memory is zeroed during system initialization.

This value is provided via <rtems/confdefs.h> in case CONFIGURE_ZERO_WORKSPACE_AUTOMATICALLY is defined.

# 8.98 Message Manager

The Message Manager provides communication and synchronization capabilities using RTEMS message queues.

## Classes

- struct rtems_message_queue_config

  *This structure defines the configuration of a message queue constructed by rtems_message_queue_construct().*

## Macros

- #define RTEMS_MESSAGE_QUEUE_BUFFER(_maximum_message_size)

  *Defines a structure which can be used as a message queue buffer for messages of the specified maximum size.*

## Functions

- rtems_status_code rtems_message_queue_broadcast (rtems_id id, const void *buffer, size_t size, uint32_t *count)

  %
- rtems_status_code rtems_message_queue_construct (const rtems_message_queue_config *config, rtems_id *id)

  *Constructs a message queue from the specified the message queue configuration.*
- rtems_status_code rtems_message_queue_create (rtems_name name, uint32_t count, size_t max_↩ message_size, rtems_attribute attribute_set, rtems_id *id)

  %
- rtems_status_code rtems_message_queue_delete (rtems_id id)

  %
- rtems_status_code rtems_message_queue_flush (rtems_id id, uint32_t *count)

  %
- rtems_status_code rtems_message_queue_get_number_pending (rtems_id id, uint32_t *count)

  %
- rtems_status_code rtems_message_queue_ident (rtems_name name, uint32_t node, rtems_id *id)

  *Identifies a message queue object by the specified object name.*
- rtems_status_code rtems_message_queue_receive (rtems_id id, void *buffer, size_t *size, rtems_option option_set, rtems_interval timeout)

  %
- rtems_status_code rtems_message_queue_send (rtems_id id, const void *buffer, size_t size)

  %
- rtems_status_code rtems_message_queue_urgent (rtems_id id, const void *buffer, size_t size)

  %

### 8.98.1 Detailed Description

The Message Manager provides communication and synchronization capabilities using RTEMS message queues.

## 8.98.2 Macro Definition Documentation

### 8.98.2.1 RTEMS_MESSAGE_QUEUE_BUFFER

```
#define RTEMS_MESSAGE_QUEUE_BUFFER(
              _maximum_message_size )
```

**Value:**

```
  struct { \
    CORE_message_queue_Buffer _buffer; \
    char _message[ _maximum_message_size ]; \
  }
```

Defines a structure which can be used as a message queue buffer for messages of the specified maximum size.

Use this macro to define the message buffer storage area for rtems_message_queue_construct().

**Parameters**

| | |
|---|---|
| *_maximum_message_size* | is the maximum message size in bytes. |

Definition at line 112 of file message.h.

## 8.98.3 Function Documentation

### 8.98.3.1 rtems_message_queue_broadcast()

```
rtems_status_code rtems_message_queue_broadcast (
            rtems_id id,
            const void * buffer,
            size_t size,
            uint32_t * count )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *buffer* | % |
| *size* | % |
| *count* | % |

Definition at line 24 of file msgqbroadcast.c.

### 8.98.3.2 rtems_message_queue_construct()

```
rtems_status_code rtems_message_queue_construct (
            const rtems_message_queue_config * config,
            rtems_id * id )
```

Constructs a message queue from the specified the message queue configuration.

In contrast to message queues created by rtems_message_queue_create(), the message queues constructed by this directive use a user-provided message buffer storage area.

This directive is intended for applications which do not want to use the RTEMS Workspace and instead statically allocate all operating system resources. An application based solely on static allocation can avoid any runtime memory allocators. This can simplify the application architecture as well as any analysis that may be required.

The value for CONFIGURE_MESSAGE_BUFFER_MEMORY should not include memory for message queues constructed by rtems_message_queue_construct().

**Parameters**

| | | |
|---|---|---|
| | *config* | is the message queue configuration. |
| out | *id* | is the pointer to an object identifier variable. The identifier of the constructed message queue object will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The id parameter was NULL. |
| *RTEMS_INVALID_NAME* | The message queue name in the configuration was invalid. |
| *RTEMS_INVALID_NUMBER* | The maximum number of pending messages in the configuration was zero. |
| *RTEMS_INVALID_SIZE* | The maximum message size in the configuration was zero. |
| *RTEMS_TOO_MANY* | There was no inactive message queue object available to construct a message queue. |
| *RTEMS_TOO_MANY* | In multiprocessing configurations, there was no inactive global object available to construct a global message queue. |
| *RTEMS_INVALID_SIZE* | The maximum message size in the configuration was too big and resulted in integer overflows in calculations carried out to determine the size of the message buffer area. |
| *RTEMS_INVALID_NUMBER* | The maximum number of pending messages in the configuration was too big and resulted in integer overflows in calculations carried out to determine the size of the message buffer area. |
| *RTEMS_UNSATISFIED* | The message queue storage area begin pointer in the configuration was NULL. |
| *RTEMS_UNSATISFIED* | The message queue storage area size in the configuration was not equal to the size calculated from the maximum number of pending messages and the maximum message size. |

Definition at line 45 of file msgqconstruct.c.

#### 8.98.3.3 rtems_message_queue_create()

```
rtems_status_code rtems_message_queue_create (
            rtems_name name,
            uint32_t count,
            size_t max_message_size,
            rtems_attribute attribute_set,
            rtems_id * id )
```

%

**Parameters**

| | |
|---|---|
| *name* | % |
| *count* | % |
| *max_message_size* | % |
| *attribute_set* | % |
| *id* | % |

Definition at line 46 of file msgqcreate.c.

#### 8.98.3.4 rtems_message_queue_delete()

```
rtems_status_code rtems_message_queue_delete (
            rtems_id id )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |

Definition at line 24 of file msgqdelete.c.

#### 8.98.3.5 rtems_message_queue_flush()

```
rtems_status_code rtems_message_queue_flush (
            rtems_id id,
            uint32_t * count )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *count* | % |

Definition at line 24 of file msgqflush.c.

#### 8.98.3.6 rtems_message_queue_get_number_pending()

```
rtems_status_code rtems_message_queue_get_number_pending (
             rtems_id id,
             uint32_t * count )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *count* | % |

Definition at line 24 of file msgqgetnumberpending.c.

#### 8.98.3.7 rtems_message_queue_ident()

```
rtems_status_code rtems_message_queue_ident (
             rtems_name name,
             uint32_t node,
             rtems_id * id )
```

Identifies a message queue object by the specified object name.

This directive obtains the message queue identifier associated with the message queue name specified in `name`.

The node to search is specified in `node`. It shall be

- a valid node number,
- the constant RTEMS_SEARCH_ALL_NODES to search in all nodes,
- the constant RTEMS_SEARCH_LOCAL_NODE to search in the local node only, or
- the constant RTEMS_SEARCH_OTHER_NODES to search in all nodes except the local node.

If the message queue name is not unique, then the message queue identifier will match the first message queue with that name in the search order. However, this message queue identifier is not guaranteed to correspond to the desired message queue. The message queue identifier is used with other message related directives to access the message queue.

If node is RTEMS_SEARCH_ALL_NODES, all nodes are searched with the local node being searched first. All other nodes are searched with the lowest numbered node searched first.

If node is a valid node number which does not represent the local node, then only the message queues exported by the designated node are searched.

This directive does not generate activity on remote nodes. It accesses only the local copy of the global object table.

**Parameters**

| | | |
|---|---|---|
| | *name* | is the object name to look up. |
| | *node* | is the node or node set to search for a matching object. |
| out | *id* | is the pointer to an object identifier variable. The object identifier of an object with the specified name will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The id parameter was NULL. |
| *RTEMS_INVALID_NAME* | The name parameter was 0. |
| *RTEMS_INVALID_NAME* | There was no object with the specified name on the specified nodes. |
| *RTEMS_INVALID_NODE* | In multiprocessing configurations, the specified node was invalid. |

Definition at line 44 of file msgqident.c.

#### 8.98.3.8 rtems_message_queue_receive()

```
rtems_status_code rtems_message_queue_receive (
            rtems_id id,
            void * buffer,
            size_t * size,
            rtems_option option_set,
            rtems_interval timeout )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *buffer* | % |
| *size* | % |
| *option_set* | % |
| *timeout* | % |

Definition at line 31 of file msgqreceive.c.

#### 8.98.3.9 rtems_message_queue_send()

```
rtems_status_code rtems_message_queue_send (
            rtems_id id,
            const void * buffer,
            size_t size )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *buffer* | % |
| *size* | % |

Definition at line 25 of file msgqsend.c.

#### 8.98.3.10 rtems_message_queue_urgent()

```
rtems_status_code rtems_message_queue_urgent (
            rtems_id id,
            const void * buffer,
            size_t size )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *buffer* | % |
| *size* | % |

Definition at line 24 of file msgqurgent.c.

## 8.99 Message Queue Handler

Message Queue Handler.

### Files

- file coremsg.h

  *Constants and Structures Associated with the Message Queue Handler.*
- file coremsgbuffer.h

  *This header file defines the buffer data structure used by the Message Queue Handler.*
- file coremsgimpl.h

  *Inlined Routines in the Core Message Handler.*
- file coremsg.c

  *Initialize a Message Queue.*
- file coremsgbroadcast.c

  *Broadcast a Message to the Message Queue.*
- file coremsgclose.c

  *Close a Message Queue.*
- file coremsgflush.c

  *Flush Messages Routine.*
- file coremsginsert.c

  *Insert a Message into the Message Queue.*
- file coremsgseize.c

  *Seize a Message from the Message Queue.*
- file coremsgsubmit.c

  *CORE Message Queue Submit.*
- file coremsgwkspace.c

  *This source file contains the implementation of _CORE_message_queue_Workspace_allocate().*

### Classes

- struct CORE_message_queue_Control

  *Control block used to manage each message queue.*
- struct CORE_message_queue_Buffer

  *The structure is used to organize message buffers of a message queue.*

### Macros

- #define RTEMS_SCORE_COREMSG_ENABLE_BLOCKING_SEND
- #define RTEMS_SCORE_COREMSG_ENABLE_MESSAGE_PRIORITY
- #define CORE_MESSAGE_QUEUE_SEND_REQUEST INT_MAX

  *Used when appending messages onto a message queue.*
- #define CORE_MESSAGE_QUEUE_URGENT_REQUEST INT_MIN

  *Used when prepending messages onto a message queue.*
- #define _CORE_message_queue_Set_notify(the_message_queue, the_handler) do { } while ( 0 )

  *Initializes notification information.*

## Typedefs

- typedef struct CORE_message_queue_Control **CORE_message_queue_Control**
- typedef int CORE_message_queue_Submit_types

  *The modes in which a message may be submitted to a message queue.*
- typedef void *(* CORE_message_queue_Allocate_buffers) (CORE_message_queue_Control *the_message_queue, size_t size, const void *arg)

  *This handler shall allocate the message buffer storage area for a message queue.*

## Enumerations

- enum CORE_message_queue_Disciplines { CORE_MESSAGE_QUEUE_DISCIPLINES_FIFO, CORE_MESSAGE_QUEUE_D }

  *The possible blocking disciplines for a message queue.*

## Functions

- void * _CORE_message_queue_Workspace_allocate (CORE_message_queue_Control *the_message_queue, size_t size, const void *arg)

  *This handler allocates the message buffer storage area for a message queue from the RTEMS Workspace.*
- Status_Control _CORE_message_queue_Initialize (CORE_message_queue_Control *the_message_queue, CORE_message_queue_Disciplines discipline, uint32_t maximum_pending_messages, size_t maximum_message_size, CORE_message_queue_Allocate_buffers allocate_buffers, const void *arg)

  *Initializes a message queue.*
- void _CORE_message_queue_Close (CORE_message_queue_Control *the_message_queue, Thread_queue_Context *queue_context)

  *Closes a message queue.*
- uint32_t _CORE_message_queue_Flush (CORE_message_queue_Control *the_message_queue, Thread_queue_Context *queue_context)

  *Flushes pending messages.*
- Status_Control _CORE_message_queue_Broadcast (CORE_message_queue_Control *the_message_queue, const void *buffer, size_t size, uint32_t *count, Thread_queue_Context *queue_context)

  *Broadcasts a message to the message queue.*
- Status_Control _CORE_message_queue_Submit (CORE_message_queue_Control *the_message_queue, Thread_Control *executing, const void *buffer, size_t size, CORE_message_queue_Submit_types submit_type, bool wait, Thread_queue_Context *queue_context)

  *Submits a message to the message queue.*
- Status_Control _CORE_message_queue_Seize (CORE_message_queue_Control *the_message_queue, Thread_Control *executing, void *buffer, size_t *size_p, bool wait, Thread_queue_Context *queue_context)

  *Seizes a message from the message queue.*
- void _CORE_message_queue_Insert_message (CORE_message_queue_Control *the_message_queue, CORE_message_queue_Buffer *the_message, const void *content_source, size_t content_size, CORE_message_queue_Submit_types submit_type)

  *Inserts a message into the message queue.*
- static __inline__ Status_Control _CORE_message_queue_Send (CORE_message_queue_Control *the_message_queue, const void *buffer, size_t size, bool wait, Thread_queue_Context *queue_context)

  *Sends a message to the message queue.*
- static __inline__ Status_Control _CORE_message_queue_Urgent (CORE_message_queue_Control *the_message_queue, const void *buffer, size_t size, bool wait, Thread_queue_Context *queue_context)

  *Sends an urgent message to the message queue.*
- static __inline__ void _CORE_message_queue_Acquire (CORE_message_queue_Control *the_message_queue, Thread_queue_Context *queue_context)

*Acquires the message queue.*

- static __inline__ void _CORE_message_queue_Acquire_critical (CORE_message_queue_Control *the_message_queue, Thread_queue_Context *queue_context)

  *Acquires the message queue critical.*

- static __inline__ void _CORE_message_queue_Release (CORE_message_queue_Control *the_message_queue, Thread_queue_Context *queue_context)

  *Releases the message queue.*

- static __inline__ void _CORE_message_queue_Copy_buffer (const void *source, void *destination, size_t size)

  *Copies the source message buffer to the destination message buffer.*

- static __inline__ CORE_message_queue_Buffer * _CORE_message_queue_Allocate_message_buffer (CORE_message_queue_Control *the_message_queue)

  *Allocates a message buffer from the inactive message buffer chain.*

- static __inline__ void _CORE_message_queue_Free_message_buffer (CORE_message_queue_Control *the_message_queue, CORE_message_queue_Buffer *the_message)

  *Frees a message buffer to inactive message buffer chain.*

- static __inline__ int _CORE_message_queue_Get_message_priority (const CORE_message_queue_Buffer *the_message)

  *Gets message priority.*

- static __inline__ CORE_message_queue_Buffer * _CORE_message_queue_Get_pending_message (CORE_message_queue_Control *the_message_queue)

  *Gets first message of message queue and removes it.*

- static __inline__ Thread_Control * _CORE_message_queue_Dequeue_receiver (CORE_message_queue_Control *the_message_queue, const void *buffer, size_t size, CORE_message_queue_Submit_types submit_type, Thread_queue_Context *queue_context)

  *Gets the first locked thread waiting to receive and dequeues it.*

## 8.99.1 Detailed Description

Message Queue Handler.

This handler encapsulates functionality which provides the foundation Message Queue services used in all of the APIs supported by RTEMS.

## 8.99.2 Macro Definition Documentation

### 8.99.2.1 _CORE_message_queue_Set_notify

```
#define _CORE_message_queue_Set_notify(
              the_message_queue,
              the_handler )  do { } while ( 0 )
```

Initializes notification information.

This routine initializes the notification information for *the_message_queue*.

**Parameters**

| | | |
|---|---|---|
| out | *the_message_queue* | The message queue to initialize the notification information. |
| out | *the_handler* | The notification information for the message queue. |

Definition at line 593 of file coremsgimpl.h.

#### 8.99.2.2 CORE_MESSAGE_QUEUE_SEND_REQUEST

```
#define CORE_MESSAGE_QUEUE_SEND_REQUEST INT_MAX
```

Used when appending messages onto a message queue.

This is the priority constant used when appending messages onto a message queue.

Definition at line 49 of file coremsgimpl.h.

#### 8.99.2.3 CORE_MESSAGE_QUEUE_URGENT_REQUEST

```
#define CORE_MESSAGE_QUEUE_URGENT_REQUEST INT_MIN
```

Used when prepending messages onto a message queue.

This is the priority constant used when prepending messages onto a message queue.

Definition at line 57 of file coremsgimpl.h.

#### 8.99.2.4 RTEMS_SCORE_COREMSG_ENABLE_BLOCKING_SEND

```
#define RTEMS_SCORE_COREMSG_ENABLE_BLOCKING_SEND
```

This macro is defined when an API is enabled that requires the Message Queue Handler include support for blocking send operations.

Definition at line 59 of file coremsg.h.

#### 8.99.2.5 RTEMS_SCORE_COREMSG_ENABLE_MESSAGE_PRIORITY

```
#define RTEMS_SCORE_COREMSG_ENABLE_MESSAGE_PRIORITY
```

This define enables the support for priority based enqueuing of messages in the Message Queue Handler.

Definition at line 58 of file coremsgbuffer.h.

### 8.99.3 Typedef Documentation

#### 8.99.3.1 CORE_message_queue_Allocate_buffers

```
typedef void*( * CORE_message_queue_Allocate_buffers) (CORE_message_queue_Control *the_message↩
_queue, size_t size, const void *arg)
```

This handler shall allocate the message buffer storage area for a message queue.

The handler shall set the CORE_message_queue_Control::free_message_buffers member.

**Parameters**

| | | |
|---|---|---|
| out | *the_message_queue* | is the message queue control. |
| | *size* | is the message buffer storage area size to allocate. |
| | *arg* | is the handler argument. |

**Return values**

| | |
|---|---|
| *NULL* | The allocation failed. |

**Returns**

Otherwise the pointer to the allocated message buffer storage area begin shall be returned.

Definition at line 89 of file coremsgimpl.h.

#### 8.99.3.2 CORE_message_queue_Submit_types

```
typedef int CORE_message_queue_Submit_types
```

The modes in which a message may be submitted to a message queue.

The following type details the modes in which a message may be submitted to a message queue. The message may be posted in a send or urgent fashion.

**Note**

All other values are message priorities. Numerically smaller priorities indicate higher priority messages.

Definition at line 69 of file coremsgimpl.h.

### 8.99.4 Enumeration Type Documentation

#### 8.99.4.1 CORE_message_queue_Disciplines

```
enum CORE_message_queue_Disciplines
```

The possible blocking disciplines for a message queue.

This enumerated types defines the possible blocking disciplines for a message queue.

**Enumerator**

| | |
|---|---|
| CORE_MESSAGE_QUEUE_DISCIPLINES_FIFO | This value indicates that blocking tasks are in FIFO order. |
| CORE_MESSAGE_QUEUE_DISCIPLINES_PRIORITY | This value indicates that blocking tasks are in priority order. |

Definition at line 69 of file coremsg.h.

## 8.99.5 Function Documentation

### 8.99.5.1 _CORE_message_queue_Acquire()

```
static __inline__ void _CORE_message_queue_Acquire (
            CORE_message_queue_Control * the_message_queue,
            Thread_queue_Context * queue_context )  [static]
```

Acquires the message queue.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_message_queue* | Rhe message queue to acquire. |
| | *queue_context* | The thread queue context. |

Definition at line 418 of file coremsgimpl.h.

### 8.99.5.2 _CORE_message_queue_Acquire_critical()

```
static __inline__ void _CORE_message_queue_Acquire_critical (
            CORE_message_queue_Control * the_message_queue,
            Thread_queue_Context * queue_context )  [static]
```

Acquires the message queue critical.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_message_queue* | The message queue to acquire critical. |
| | *queue_context* | The thread queue context. |

Definition at line 432 of file coremsgimpl.h.

#### 8.99.5.3 _CORE_message_queue_Allocate_message_buffer()

```
static __inline__ CORE_message_queue_Buffer* _CORE_message_queue_Allocate_message_buffer (
            CORE_message_queue_Control * the_message_queue ) [static]
```

Allocates a message buffer from the inactive message buffer chain.

This function allocates a message buffer from the inactive message buffer chain.

**Parameters**

| | |
|---|---|
| *the_message_queue* | The message queue to operate upon. |

**Return values**

| | |
|---|---|
| *pointer* | The allocated message buffer. |
| *NULL* | The inactive message buffer chain is empty. |

Definition at line 485 of file coremsgimpl.h.

#### 8.99.5.4 _CORE_message_queue_Broadcast()

```
Status_Control _CORE_message_queue_Broadcast (
            CORE_message_queue_Control * the_message_queue,
            const void * buffer,
            size_t size,
            uint32_t * count,
            Thread_queue_Context * queue_context )
```

Broadcasts a message to the message queue.

This package is the implementation of the CORE Message Queue Handler. This core object provides task synchronization and communication functions via messages passed to queue objects.

This function sends a message for every thread waiting on the queue and returns the number of threads made ready by the message.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_message_queue* | The message queue to operate upon. |
| | *buffer* | The starting address of the message to broadcast. |
| | *size* | The size of the message being broadcast. |
| out | *count* | The variable that will contain the number of tasks that are sent this message. |
| | *queue_context* | The thread queue context used for _CORE_message_queue_Acquire() or _CORE_message_queue_Acquire_critical(). |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The message was successfully broadcast. |

**Return values**

| | |
|---|---|
| *STATUS_MESSAGE_INVALID_SIZE* | The message size was too big. |

Definition at line 24 of file coremsgbroadcast.c.

#### 8.99.5.5 _CORE_message_queue_Close()

```
void _CORE_message_queue_Close (
            CORE_message_queue_Control * the_message_queue,
            Thread_queue_Context * queue_context )
```

Closes a message queue.

This package is the implementation of the CORE Message Queue Handler. This core object provides task synchronization and communication functions via messages passed to queue objects.

This function closes a message by returning all allocated space and flushing *the_message_queue's* task wait queue.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_message_queue* | The message queue to close. |
| in,out | *queue_context* | The thread queue context used for _CORE_message_queue_Acquire() or _CORE_message_queue_Acquire_critical(). |

Definition at line 34 of file coremsgclose.c.

#### 8.99.5.6 _CORE_message_queue_Copy_buffer()

```
static __inline__ void _CORE_message_queue_Copy_buffer (
            const void * source,
            void * destination,
            size_t size )  [static]
```

Copies the source message buffer to the destination message buffer.

This routine copies the contents of the source message buffer to the destination message buffer.

**Parameters**

| | | |
|---|---|---|
| | *source* | The source message buffer to be copied. |
| out | *destination* | The destination messag buffer to copy the source to. |
| | *size* | The size of the source buffer. |

Definition at line 464 of file coremsgimpl.h.

#### 8.99.5.7 _CORE_message_queue_Dequeue_receiver()

```
static __inline__ Thread_Control* _CORE_message_queue_Dequeue_receiver (
            CORE_message_queue_Control * the_message_queue,
            const void * buffer,
            size_t size,
            CORE_message_queue_Submit_types submit_type,
            Thread_queue_Context * queue_context )  [static]
```

Gets the first locked thread waiting to receive and dequeues it.

This method dequeues the first locked thread waiting to receive a message, dequeues it and returns the corresponding Thread_Control.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_message_queue* | The message queue to operate upon. |
| | *buffer* | The buffer that is copied to the threads mutable_object. |
| | *size* | The size of the buffer. |
| | *submit_type* | Indicates whether the thread should be willing to block in the future. |
| | *queue_context* | The thread queue context. |

**Return values**

| | |
|---|---|
| *thread* | The Thread_Control for the first locked thread, if there is a locked thread. |
| *NULL* | There are pending messages or no thread waiting to receive. |

Definition at line 612 of file coremsgimpl.h.

#### 8.99.5.8 _CORE_message_queue_Flush()

```
uint32_t _CORE_message_queue_Flush (
            CORE_message_queue_Control * the_message_queue,
            Thread_queue_Context * queue_context )
```

Flushes pending messages.

This package is the implementation of the CORE Message Queue Handler. This core object provides task synchronization and communication functions via messages passed to queue objects.

This function flushes *the_message_queue's* pending message queue. The number of messages flushed from the queue is returned.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_message_queue* | The message queue to flush. |
| | *queue_context* | The thread queue context with interrupts disabled. |

**Returns**

This method returns the number of message pending messages flushed.

Definition at line 24 of file coremsgflush.c.

#### 8.99.5.9 _CORE_message_queue_Free_message_buffer()

```
static __inline__ void _CORE_message_queue_Free_message_buffer (
            CORE_message_queue_Control * the_message_queue,
            CORE_message_queue_Buffer * the_message )  [static]
```

Frees a message buffer to inactive message buffer chain.

This routine frees a message buffer to the inactive message buffer chain.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_message_queue* | The message queue to free the message buffer to. |
| out | *the_message* | The message to be freed. |

Definition at line 502 of file coremsgimpl.h.

#### 8.99.5.10 _CORE_message_queue_Get_message_priority()

```
static __inline__ int _CORE_message_queue_Get_message_priority (
            const CORE_message_queue_Buffer * the_message )  [static]
```

Gets message priority.

This function returns the priority of *the_message*.

**Parameters**

| | |
|---|---|
| *the_message* | The message to obtain the priority from. |

**Return values**

| | |
|---|---|
| *priority* | The priority of this message. |
| *0* | Message priority is disabled. |

**Note**

It encapsulates the optional behavior that message priority is disabled if no API requires it.

Definition at line 523 of file coremsgimpl.h.

#### 8.99.5.11 _CORE_message_queue_Get_pending_message()

```
static __inline__ CORE_message_queue_Buffer* _CORE_message_queue_Get_pending_message (
            CORE_message_queue_Control * the_message_queue )  [static]
```

Gets first message of message queue and removes it.

This function removes the first message from the_message_queue and returns a pointer to it.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_message_queue* | The message queue to get the first message from. |

**Return values**

| | |
|---|---|
| *pointer* | The first message if the message queue is not empty. |
| *NULL* | The message queue is empty. |

Definition at line 546 of file coremsgimpl.h.

#### 8.99.5.12 _CORE_message_queue_Initialize()

```
Status_Control _CORE_message_queue_Initialize (
            CORE_message_queue_Control * the_message_queue,
            CORE_message_queue_Disciplines discipline,
            uint32_t maximum_pending_messages,
            size_t maximum_message_size,
            CORE_message_queue_Allocate_buffers allocate_buffers,
            const void * arg )
```

Initializes a message queue.

**Parameters**

| | | |
|---|---|---|
| out | *the_message_queue* | is the message queue to initialize. |
| | *discipline* | is the blocking discipline for the message queue. |
| | *maximum_pending_messages* | is the maximum number of messages that will be allowed to be pending at any given time. |
| | *maximum_message_size* | is the size of the largest message that may be sent to this message queue instance. |
| | *allocate_buffers* | is the message buffer storage area allocation handler. |
| | *arg* | is the message buffer storage area allocation handler argument. |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The message queue was initialized. |
| *STATUS_MESSAGE_QUEUE_INVALID_SIZE* | Calculations with the maximum pending messages or maximum message size produced an integer overflow. |
| *STATUS_MESSAGE_QUEUE_NO_MEMORY* | The message buffer storage area allocation failed. |

Definition at line 33 of file coremsg.c.

#### 8.99.5.13 _CORE_message_queue_Insert_message()

```
void _CORE_message_queue_Insert_message (
            CORE_message_queue_Control * the_message_queue,
            CORE_message_queue_Buffer * the_message,
            const void * content_source,
            size_t content_size,
            CORE_message_queue_Submit_types submit_type )
```

Inserts a message into the message queue.

Copies the specified content into the message storage space and then inserts the message into the message queue according to the submit type.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_message_queue* | The message queue to insert a message in. |
| in,out | *the_message* | The message to insert in the message queue. |
| | *content_source* | The message content source. |
| | *content_size* | The message content size in bytes. |
| | *submit_type* | Determines whether the message is prepended, appended, or enqueued in priority order. |

Definition at line 41 of file coremsginsert.c.

#### 8.99.5.14 _CORE_message_queue_Release()

```
static __inline__ void _CORE_message_queue_Release (
            CORE_message_queue_Control * the_message_queue,
            Thread_queue_Context * queue_context )  [static]
```

Releases the message queue.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_message_queue* | The message queue to release. |
| | *queue_context* | The thread queue context. |

Definition at line 446 of file coremsgimpl.h.

#### 8.99.5.15 _CORE_message_queue_Seize()

```
Status_Control _CORE_message_queue_Seize (
            CORE_message_queue_Control * the_message_queue,
            Thread_Control * executing,
            void * buffer,
            size_t * size_p,
            bool wait,
            Thread_queue_Context * queue_context )
```

Seizes a message from the message queue.

This package is the implementation of the CORE Message Queue Handler. This core object provides task synchronization and communication functions via messages passed to queue objects.

This kernel routine dequeues a message, copies the message buffer to a given destination buffer, and frees the message buffer to the inactive message pool. The thread will be blocked if wait is true, otherwise an error will be given to the thread if no messages are available.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_message_queue* | The message queue to seize a message from. |
| | *executing* | The executing thread. |
| out | *buffer* | The starting address of the message buffer to to be filled in with a message. |
| out | *size_p* | The size of the *buffer*, indicates the maximum size message that the caller can receive. |
| | *wait* | Indicates whether the calling thread is willing to block if the message queue is empty. |
| | *queue_context* | The thread queue context used for _CORE_message_queue_Acquire() or _CORE_message_queue_Acquire_critical(). |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The message was successfully seized from the message queue. |
| *STATUS_UNSATISFIED* | Wait was set to false and there is currently no pending message. |
| *STATUS_TIMEOUT* | A timeout occured. |

**Note**

Returns message priority via return area in TCB.

- INTERRUPT LATENCY:
  - available
  - wait

Definition at line 27 of file coremsgseize.c.

#### 8.99.5.16 _CORE_message_queue_Send()

```
static __inline__ Status_Control _CORE_message_queue_Send (
             CORE_message_queue_Control * the_message_queue,
             const void * buffer,
             size_t size,
             bool wait,
             Thread_queue_Context * queue_context )  [static]
```

Sends a message to the message queue.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_message_queue* | The message queue to send a message to. |
| | *buffer* | The starting address of the message to send. |
| | *sizeis* | The size of the message being send. |
| | *wait* | Indicates whether the calling thread is willing to block if the message queue is full. |
| | *queue_context* | The thread queue context used for _CORE_message_queue_Acquire() or _CORE_message_queue_Acquire_critical(). |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The message was successfully submitted to the message queue. |
| *STATUS_MESSAGE_INVALID_SIZE* | The message size was too big. |
| *STATUS_TOO_MANY* | No message buffers were available. |
| *STATUS_MESSAGE_QUEUE_WAIT_IN_ISR* | The caller is in an ISR, do not block! |
| *STATUS_TIMEOUT* | A timeout occured. |

Definition at line 357 of file coremsgimpl.h.

#### 8.99.5.17 _CORE_message_queue_Submit()

```
Status_Control _CORE_message_queue_Submit (
             CORE_message_queue_Control * the_message_queue,
             Thread_Control * executing,
             const void * buffer,
             size_t size,
             CORE_message_queue_Submit_types submit_type,
             bool wait,
             Thread_queue_Context * queue_context )
```

Submits a message to the message queue.

This routine implements the send and urgent message functions. It processes a message that is to be submitted to the designated message queue. The message will either be processed as a send message which it will be inserted at the rear of the queue or it will be processed as an urgent message which will be inserted at the front of the queue.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_message_queue* | The message queue to operate upon. |
| | *executing* | The executing thread. |
| | *buffer* | The starting address of the message to send. |
| | *size* | The size of the message being send. |
| | *submit_type* | Determines whether the message is prepended, appended, or enqueued in priority order. |
| | *wait* | Indicates whether the calling thread is willing to block if the message queue is full. |
| | *queue_context* | The thread queue context used for _CORE_message_queue_Acquire() or _CORE_message_queue_Acquire_critical(). |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The message was successfully submitted to the message queue. |
| *STATUS_MESSAGE_INVALID_SIZE* | The message size was too big. |
| *STATUS_TOO_MANY* | No message buffers were available. |
| *STATUS_MESSAGE_QUEUE_WAIT_IN_ISR* | The caller is in an ISR, do not block! |
| *STATUS_TIMEOUT* | A timeout occured. |

Definition at line 29 of file coremsgsubmit.c.

### 8.99.5.18 _CORE_message_queue_Urgent()

```
static __inline__ Status_Control _CORE_message_queue_Urgent (
            CORE_message_queue_Control * the_message_queue,
            const void * buffer,
            size_t size,
            bool wait,
            Thread_queue_Context * queue_context )  [static]
```

Sends an urgent message to the message queue.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_message_queue* | The message queue to send an urgent message to. |
| | *buffer* | The starting address of the message to send. |
| | *sizeis* | The size of the message being send. |
| | *wait* | Indicates whether the calling thread is willing to block if the message queue is full. |
| | *queue_context* | The thread queue context used for _CORE_message_queue_Acquire() or _CORE_message_queue_Acquire_critical(). |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The message was successfully submitted to the message queue. |
| *STATUS_MESSAGE_INVALID_SIZE* | The message size was too big. |

**Return values**

| | |
|---|---|
| *STATUS_TOO_MANY* | No message buffers were available. |
| *STATUS_MESSAGE_QUEUE_WAIT_IN_ISR* | The caller is in an ISR, do not block! |
| *STATUS_TIMEOUT* | A timeout occured. |

Definition at line 393 of file coremsgimpl.h.

#### 8.99.5.19 _CORE_message_queue_Workspace_allocate()

```
void* _CORE_message_queue_Workspace_allocate (
            CORE_message_queue_Control * the_message_queue,
            size_t size,
            const void * arg )
```

This handler allocates the message buffer storage area for a message queue from the RTEMS Workspace.

The handler sets the CORE_message_queue_Control::free_message_buffers to _Workspace_Free().

**Parameters**

| | | |
|---|---|---|
| out | *the_message_queue* | is the message queue control. |
| | *size* | is the message buffer storage area size to allocate. |
| | *arg* | is the unused handler argument. |

**Return values**

| | |
|---|---|
| *NULL* | The allocation failed. |

**Returns**

Otherwise the pointer to the allocated message buffer storage area begin is returned.

Definition at line 44 of file coremsgwkspace.c.

## 8.100 Multiprocessing Configuration

### Macros

- #define CONFIGURE_EXTRA_MPCI_RECEIVE_SERVER_STACK

  *This configuration option is an integer define.*
- #define CONFIGURE_MP_APPLICATION

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_MP_MAXIMUM_GLOBAL_OBJECTS

  *This configuration option is an integer define.*
- #define CONFIGURE_MP_MAXIMUM_NODES

  *This configuration option is an integer define.*
- #define CONFIGURE_MP_MAXIMUM_PROXIES

  *This configuration option is an integer define.*
- #define CONFIGURE_MP_MPCI_TABLE_POINTER

  *This configuration option is an initializer define.*
- #define CONFIGURE_MP_NODE_NUMBER

  *This configuration option is an integer define.*

### 8.100.1 Detailed Description

This section describes multiprocessing related configuration options. The options are only used if RTEMS was built with the `--enable-multiprocessing` build configuration option. Additionally, this class of configuration options are only applicable if the configuration option CONFIGURE_MP_APPLICATION is defined. The multiprocessing (MPCI) support must not be confused with the SMP support.

### 8.100.2 Macro Definition Documentation

#### 8.100.2.1 CONFIGURE_EXTRA_MPCI_RECEIVE_SERVER_STACK

```
#define CONFIGURE_EXTRA_MPCI_RECEIVE_SERVER_STACK
```

This configuration option is an integer define.

The value of this configuration option defines the number of bytes the applications wishes to add to the MPCI task stack on top of CONFIGURE_MINIMUM_TASK_STACK_SIZE.

**Default Value**

The default value is 0.

**Value Constraints**

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to `UINT32_MAX`.
- It shall be small enough so that the MPCI receive server stack area calculation carried out by `<rtems/confdefs.h>` does not overflow an integer of type `size_t`.

**Notes**

This configuration option is only evaluated if CONFIGURE_MP_APPLICATION is defined.

Definition at line 3272 of file appl-config.h.

### 8.100.2.2 CONFIGURE_MP_APPLICATION

```
#define CONFIGURE_MP_APPLICATION
```

This configuration option is a boolean feature define.

This configuration option is defined to indicate that the application intends to be part of a multiprocessing configuration. Additional configuration options are assumed to be provided.

#### Default Configuration

If this configuration option is undefined, then the multiprocessing services are not initialized.

#### Notes

This configuration option shall be undefined if the multiprocessing support is not enabled (e.g. RTEMS was built without the `--enable-multiprocessing` build configuration option). Otherwise a compile time error in the configuration file will occur.

Definition at line 3293 of file appl-config.h.

### 8.100.2.3 CONFIGURE_MP_MAXIMUM_GLOBAL_OBJECTS

```
#define CONFIGURE_MP_MAXIMUM_GLOBAL_OBJECTS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of concurrently active global objects in a multiprocessor system.

#### Default Value

The default value is 32.

#### Value Constraints

The value of this configuration option shall be greater than or equal to 0 and less than or equal to UINT32_MAX.

#### Notes

This value corresponds to the total number of objects which can be created with the RTEMS_GLOBAL attribute.

This configuration option is only evaluated if CONFIGURE_MP_APPLICATION is defined.

Definition at line 3320 of file appl-config.h.

#### 8.100.2.4 CONFIGURE_MP_MAXIMUM_NODES

```
#define CONFIGURE_MP_MAXIMUM_NODES
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of nodes in a multiprocessor system.

**Default Value**

The default value is 2.

**Value Constraints**

The value of this configuration option shall be greater than or equal to 0 and less than or equal to UIN↩T32_MAX.

**Notes**

This configuration option is only evaluated if CONFIGURE_MP_APPLICATION is defined.

Definition at line 3342 of file appl-config.h.

#### 8.100.2.5 CONFIGURE_MP_MAXIMUM_PROXIES

```
#define CONFIGURE_MP_MAXIMUM_PROXIES
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of concurrently active thread/task proxies on this node in a multiprocessor system.

**Default Value**

The default value is 32.

**Value Constraints**

The value of this configuration option shall be greater than or equal to 0 and less than or equal to UIN↩T32_MAX.

**Notes**

Since a proxy is used to represent a remote task/thread which is blocking on this node. This configuration parameter reflects the maximum number of remote tasks/threads which can be blocked on objects on this node, see Proxies.

This configuration option is only evaluated if CONFIGURE_MP_APPLICATION is defined.

Definition at line 3372 of file appl-config.h.

### 8.100.2.6 CONFIGURE_MP_MPCI_TABLE_POINTER

```
#define CONFIGURE_MP_MPCI_TABLE_POINTER
```

This configuration option is an initializer define.

The value of this configuration option initializes the MPCI Configuration Table.

#### Default Value

The default value is `&MPCI_table`.

#### Value Constraints

The value of this configuration option shall be a pointer to ::rtems_mpci_table.

#### Notes

RTEMS provides a Shared Memory MPCI Device Driver which can be used on any Multiprocessor System assuming the BSP provides the proper set of supporting methods.

This configuration option is only evaluated if CONFIGURE_MP_APPLICATION is defined.

Definition at line 3399 of file appl-config.h.

### 8.100.2.7 CONFIGURE_MP_NODE_NUMBER

```
#define CONFIGURE_MP_NODE_NUMBER
```

This configuration option is an integer define.

The value of this configuration option defines the node number of this node in a multiprocessor system.

#### Default Value

The default value is `NODE_NUMBER`.

#### Value Constraints

The value of this configuration option shall be greater than or equal to 0 and less than or equal to `UIN↩T32_MAX`.

#### Notes

In the RTEMS Multiprocessing Test Suite, the node number is derived from the Makefile variable `NODE_NU↩MBER`. The same code is compiled with the `NODE_NUMBER` set to different values. The test programs behave differently based upon their node number.

This configuration option is only evaluated if CONFIGURE_MP_APPLICATION is defined.

Definition at line 3428 of file appl-config.h.

## 8.101 Multiprocessor Resource Sharing Protocol Handler

Multiprocessor Resource Sharing Protocol (MrsP).

### Files

- file mrsp.h

  *Definitions for Multiprocessor Resource Sharing Protocol (MrsP).*
- file mrspimpl.h

  *Definitions for Multiprocessor Resource Sharing Protocol (MrsP) Implementation.*

### Classes

- struct MRSP_Control

  *MrsP control block.*

### Macros

- #define **MRSP_TQ_OPERATIONS** &_Thread_queue_Operations_priority_inherit

### Functions

- static __inline__ void _MRSP_Acquire_critical (MRSP_Control *mrsp, Thread_queue_Context *queue_↩ context)

  *Acquires critical accordingt to MrsP.*
- static __inline__ void _MRSP_Release (MRSP_Control *mrsp, Thread_queue_Context *queue_context)

  *Releases according to MrsP.*
- static __inline__ Thread_Control * _MRSP_Get_owner (const MRSP_Control *mrsp)

  *Gets owner of the MrsP control.*
- static __inline__ void _MRSP_Set_owner (MRSP_Control *mrsp, Thread_Control *owner)

  *Sets owner of the MrsP control.*
- static __inline__ Priority_Control _MRSP_Get_priority (const MRSP_Control *mrsp, const Scheduler_Control *scheduler)

  *Gets priority of the MrsP control.*
- static __inline__ void _MRSP_Set_priority (MRSP_Control *mrsp, const Scheduler_Control *scheduler, Priority_Control new_priority)

  *Sets priority of the MrsP control.*
- static __inline__ Status_Control _MRSP_Raise_priority (MRSP_Control *mrsp, Thread_Control *thread, Priority_Node *priority_node, Thread_queue_Context *queue_context)

  *Adds the priority to the given thread.*
- static __inline__ void _MRSP_Remove_priority (Thread_Control *thread, Priority_Node *priority_node, Thread_queue_Context *queue_context)

  *Removes the priority from the given thread.*
- static __inline__ void _MRSP_Replace_priority (MRSP_Control *mrsp, Thread_Control *thread, Priority_Node *ceiling_priority)

  *Replaces the given priority node with the ceiling priority of the MrsP control.*
- static __inline__ Status_Control _MRSP_Claim_ownership (MRSP_Control *mrsp, Thread_Control *executing, Thread_queue_Context *queue_context)

*Claims ownership of the MrsP control.*

- static __inline__ Status_Control _MRSP_Initialize (MRSP_Control *mrsp, const Scheduler_Control *scheduler, Priority_Control ceiling_priority, Thread_Control *executing, bool initially_locked)

  *Initializes a MrsP control.*
- static __inline__ Status_Control _MRSP_Wait_for_ownership (MRSP_Control *mrsp, Thread_Control *executing, Thread_queue_Context *queue_context)

  *Waits for the ownership of the MrsP control.*
- static __inline__ Status_Control _MRSP_Seize (MRSP_Control *mrsp, Thread_Control *executing, bool wait, Thread_queue_Context *queue_context)

  *Seizes the MrsP control.*
- static __inline__ Status_Control _MRSP_Surrender (MRSP_Control *mrsp, Thread_Control *executing, Thread_queue_Context *queue_context)

  *Surrenders the MrsP control.*
- static __inline__ Status_Control _MRSP_Can_destroy (MRSP_Control *mrsp)

  *Checks if the MrsP control can be destroyed.*
- static __inline__ void _MRSP_Destroy (MRSP_Control *mrsp, Thread_queue_Context *queue_context)

  *Destroys the MrsP control.*

## 8.101.1 Detailed Description

Multiprocessor Resource Sharing Protocol (MrsP).

The Multiprocessor Resource Sharing Protocol (MrsP) is defined in A. Burns and A.J. Wellings, A Schedulability Compatible Multiprocessor Resource Sharing Protocol - MrsP, Proceedings of the 25th Euromicro Conference on Real-Time Systems (ECRTS 2013), July 2013. It is a generalization of the Priority Ceiling Protocol to SMP systems. Each MrsP semaphore uses a ceiling priority per scheduler instance. A task obtaining or owning a MrsP semaphore will execute with the ceiling priority for its scheduler instance as specified by the MrsP semaphore object. Tasks waiting to get ownership of a MrsP semaphore will not relinquish the processor voluntarily. In case the owner of a MrsP semaphore gets preempted it can ask all tasks waiting for this semaphore to help out and temporarily borrow the right to execute on one of their assigned processors.

## 8.101.2 Function Documentation

### 8.101.2.1 _MRSP_Acquire_critical()

```
static __inline__ void _MRSP_Acquire_critical (
            MRSP_Control * mrsp,
            Thread_queue_Context * queue_context )  [static]
```

Acquires critical accordingt to MrsP.

**Parameters**

| | |
|---|---|
| *mrsp* | The MrsP control for the operation. |
| *queue_context* | The thread queue context. |

Definition at line 53 of file mrspimpl.h.

#### 8.101.2.2 _MRSP_Can_destroy()

```
static __inline__ Status_Control _MRSP_Can_destroy (
            MRSP_Control * mrsp )  [static]
```

Checks if the MrsP control can be destroyed.

**Parameters**

| | |
|---|---|
| *mrsp* | The MrsP control for the operation. |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The MrsP is currently not used and can be destroyed. |
| *STATUS_RESOURCE_IN_USE* | The MrsP control is in use, it cannot be destroyed. |

Definition at line 497 of file mrspimpl.h.

#### 8.101.2.3 _MRSP_Claim_ownership()

```
static __inline__ Status_Control _MRSP_Claim_ownership (
            MRSP_Control * mrsp,
            Thread_Control * executing,
            Thread_queue_Context * queue_context )  [static]
```

Claims ownership of the MrsP control.

**Parameters**

| | | |
|---|---|---|
| | *mrsp* | The MrsP control to claim the ownership of. |
| in,out | *executing* | The currently executing thread. |
| | *queue_context* | The thread queue context. |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The operation succeeded. |
| *STATUS_MUTEX_CEILING_VIOLATED* | The wait priority of the executing thread exceeds the ceiling priority. |

Definition at line 246 of file mrspimpl.h.

#### 8.101.2.4 _MRSP_Destroy()

```
static __inline__ void _MRSP_Destroy (
```

```
            MRSP_Control * mrsp,
            Thread_queue_Context * queue_context )  [static]
```

Destroys the MrsP control.

**Parameters**

| | | |
|---|---|---|
| in,out | *The* | mrsp that is about to be destroyed. |
| | *queue_context* | The thread queue context. |

Definition at line 512 of file mrspimpl.h.

### 8.101.2.5 _MRSP_Get_owner()

```
static __inline__ Thread_Control* _MRSP_Get_owner (
            const MRSP_Control * mrsp )  [static]
```

Gets owner of the MrsP control.

**Parameters**

| | |
|---|---|
| *mrsp* | The MrsP control to get the owner from. |

**Returns**

The owner of the Mrsp control.

Definition at line 82 of file mrspimpl.h.

### 8.101.2.6 _MRSP_Get_priority()

```
static __inline__ Priority_Control _MRSP_Get_priority (
            const MRSP_Control * mrsp,
            const Scheduler_Control * scheduler )  [static]
```

Gets priority of the MrsP control.

**Parameters**

| | |
|---|---|
| *mrsp* | The mrsp to get the priority from. |
| *scheduler* | The corresponding scheduler. |

**Returns**

The priority of the MrsP control.

Definition at line 111 of file mrspimpl.h.

#### 8.101.2.7 _MRSP_Initialize()

```
static __inline__ Status_Control _MRSP_Initialize (
            MRSP_Control * mrsp,
            const Scheduler_Control * scheduler,
            Priority_Control ceiling_priority,
            Thread_Control * executing,
            bool initially_locked )  [static]
```

Initializes a MrsP control.

**Parameters**

| | | |
|---|---|---|
| out | *mrsp* | The MrsP control that is initialized. |
| | *scheduler* | The scheduler for the operation. |
| | *ceiling_priority* | |
| | *executing* | The currently executing thread. Ignored in this method. |
| | *initially_locked* | Indicates whether the MrsP control shall be initally locked. If it is initially locked, this method returns STATUS_INVALID_NUMBER. |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The operation succeeded. |
| *STATUS_INVALID_NUMBER* | The MrsP control is initially locked. |
| *STATUS_NO_MEMORY* | There is not enough memory to allocate. |

Definition at line 289 of file mrspimpl.h.

#### 8.101.2.8 _MRSP_Raise_priority()

```
static __inline__ Status_Control _MRSP_Raise_priority (
            MRSP_Control * mrsp,
            Thread_Control * thread,
            Priority_Node * priority_node,
            Thread_queue_Context * queue_context )  [static]
```

Adds the priority to the given thread.

**Parameters**

| | | |
|---|---|---|
| | *mrsp* | The MrsP control for the operation. |
| in,out | *thread* | The thread to add the priority node to. |
| out | *priority_node* | The priority node to initialize and add to the thread. |
| | *queue_context* | The thread queue context. |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The operation succeeded. |
| *STATUS_MUTEX_CEILING_VIOLATED* | The wait priority of the thread exceeds the ceiling priority. |

Definition at line 154 of file mrspimpl.h.

#### 8.101.2.9 _MRSP_Release()

```
static __inline__ void _MRSP_Release (
            MRSP_Control * mrsp,
            Thread_queue_Context * queue_context )  [static]
```

Releases according to MrsP.

**Parameters**

| | |
|---|---|
| *mrsp* | The MrsP control for the operation. |
| *queue_context* | The thread queue context. |

Definition at line 67 of file mrspimpl.h.

#### 8.101.2.10 _MRSP_Remove_priority()

```
static __inline__ void _MRSP_Remove_priority (
            Thread_Control * thread,
            Priority_Node * priority_node,
            Thread_queue_Context * queue_context )  [static]
```

Removes the priority from the given thread.

**Parameters**

| | | |
|---|---|---|
| in,out | *The* | thread to remove the priority from. |
| | *priority_node* | The priority node to remove from the thread |
| | *queue_context* | The thread queue context. |

Definition at line 196 of file mrspimpl.h.

#### 8.101.2.11 _MRSP_Replace_priority()

```
static __inline__ void _MRSP_Replace_priority (
            MRSP_Control * mrsp,
```

```
            Thread_Control * thread,
            Priority_Node * ceiling_priority )  [static]
```

Replaces the given priority node with the ceiling priority of the MrsP control.

**Parameters**

| | | |
|---|---|---|
| | *mrsp* | The mrsp control for the operation. |
| out | *thread* | The thread to replace the priorities. |
| | *ceiling_priority* | The node to be replaced. |

Definition at line 218 of file mrspimpl.h.

#### 8.101.2.12 _MRSP_Seize()

```
static __inline__ Status_Control _MRSP_Seize (
            MRSP_Control * mrsp,
            Thread_Control * executing,
            bool wait,
            Thread_queue_Context * queue_context )  [static]
```

Seizes the MrsP control.

**Parameters**

| | | |
|---|---|---|
| in,out | *mrsp* | The MrsP control to seize the control of. |
| in,out | *executing* | The currently executing thread. |
| | *wait* | Indicates whether the calling thread is willing to wait. |
| | *queue_context* | The thread queue context. |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The operation succeeded. |
| *STATUS_MUTEX_CEILING_VIOLATED* | The wait priority of the executing thread exceeds the ceiling priority. |
| *STATUS_UNAVAILABLE* | The executing thread is already the owner of the MrsP control. Seizing it is not possible. |

Definition at line 406 of file mrspimpl.h.

#### 8.101.2.13 _MRSP_Set_owner()

```
static __inline__ void _MRSP_Set_owner (
            MRSP_Control * mrsp,
            Thread_Control * owner )  [static]
```

Sets owner of the MrsP control.

**Parameters**

| | | |
|---|---|---|
| out | *mrsp* | The MrsP control to set the owner of. |
| | *owner* | The desired new owner for *mrsp*. |

Definition at line 95 of file mrspimpl.h.

### 8.101.2.14 _MRSP_Set_priority()

```
static __inline__ void _MRSP_Set_priority (
              MRSP_Control * mrsp,
              const Scheduler_Control * scheduler,
              Priority_Control new_priority )  [static]
```

Sets priority of the MrsP control.

**Parameters**

| | | |
|---|---|---|
| out | *mrsp* | The MrsP control to set the priority of. |
| | *schedulger* | The corresponding scheduler. |
| | *new_priority* | The new priority for the MrsP control |

Definition at line 129 of file mrspimpl.h.

### 8.101.2.15 _MRSP_Surrender()

```
static __inline__ Status_Control _MRSP_Surrender (
              MRSP_Control * mrsp,
              Thread_Control * executing,
              Thread_queue_Context * queue_context )  [static]
```

Surrenders the MrsP control.

**Parameters**

| | | |
|---|---|---|
| in,out | *mrsp* | The MrsP control to surrender the control of. |
| in,out | *executing* | The currently executing thread. |
| | *queue_context* | The thread queue context. |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The operation succeeded. |
| *STATUS_NOT_OWNER* | The executing thread does not own the MrsP control. |

Definition at line 445 of file mrspimpl.h.

#### 8.101.2.16 _MRSP_Wait_for_ownership()

```
static __inline__ Status_Control _MRSP_Wait_for_ownership (
            MRSP_Control * mrsp,
            Thread_Control * executing,
            Thread_queue_Context * queue_context )  [static]
```

Waits for the ownership of the MrsP control.

**Parameters**

| | | |
|---|---|---|
| in,out | *mrsp* | The MrsP control to get the ownership of. |
| in,out | *executing* | The currently executing thread. |
| | *queue_context* | the thread queue context. |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The operation succeeded. |
| *STATUS_MUTEX_CEILING_VIOLATED* | The wait priority of the currently executing thread exceeds the ceiling priority. |
| *STATUS_DEADLOCK* | A deadlock occured. |
| *STATUS_TIMEOUT* | A timeout occured. |

Definition at line 334 of file mrspimpl.h.

# 8.102 Mutex Handler

Mutex Handler.

## Files

- file coremutex.h

  *CORE Mutex API.*
- file coremuteximpl.h

  *CORE Mutex Implementation.*
- file coremutexseize.c

  *Seize Mutex with Blocking.*

## Classes

- struct CORE_mutex_Control

  *Control block used to manage each mutex.*
- struct CORE_recursive_mutex_Control

  *The recursive mutex control.*
- struct CORE_ceiling_mutex_Control

  *The recursive mutex control with priority ceiling protocol support.*

## Macros

- #define **CORE_MUTEX_TQ_OPERATIONS** &_Thread_queue_Operations_priority
- #define **CORE_MUTEX_TQ_PRIORITY_INHERIT_OPERATIONS** &_Thread_queue_Operations_priority↩ _inherit

## Functions

- static __inline__ void _CORE_mutex_Initialize (CORE_mutex_Control *the_mutex)

  *Initializes the mutex.*
- static __inline__ void _CORE_mutex_Destroy (CORE_mutex_Control *the_mutex)

  *Destroys the mutex.*
- static __inline__ void _CORE_mutex_Acquire_critical (CORE_mutex_Control *the_mutex, Thread_queue_Context *queue_context)

  *Acquires the mutex critical.*
- static __inline__ void _CORE_mutex_Release (CORE_mutex_Control *the_mutex, Thread_queue_Context *queue_context)

  *Releases the mutex.*
- static __inline__ Thread_Control * _CORE_mutex_Get_owner (const CORE_mutex_Control *the_mutex)

  *Gets the owner of the mutex.*
- static __inline__ bool _CORE_mutex_Is_locked (const CORE_mutex_Control *the_mutex)

  *Checks if the mutex is locked.*
- Status_Control _CORE_mutex_Seize_slow (CORE_mutex_Control *the_mutex, const Thread_queue_Operations *operations, Thread_Control *executing, bool wait, Thread_queue_Context *queue_context)

  *Seize the mutex slowly.*

- static __inline__ void _CORE_mutex_Set_owner (CORE_mutex_Control *the_mutex, Thread_Control *owner)

  *Sets the owner of the mutex.*
- static __inline__ bool _CORE_mutex_Is_owner (const CORE_mutex_Control *the_mutex, const Thread_Control *the_thread)

  *Checks if the the thread is the owner of the mutex.*
- static __inline__ void _CORE_recursive_mutex_Initialize (CORE_recursive_mutex_Control *the_mutex)

  *Initializes a recursive mutex.*
- static __inline__ Status_Control _CORE_recursive_mutex_Seize_nested (CORE_recursive_mutex_Control *the_mutex)

  *Seizes the recursive mutex nested.*
- static __inline__ Status_Control _CORE_recursive_mutex_Seize (CORE_recursive_mutex_Control *the_mutex, const Thread_queue_Operations *operations, Thread_Control *executing, bool wait, Status_Control(*nested)(CORE_recursive_mutex_Control *), Thread_queue_Context *queue_context)

  *Seizes the recursive mutex.*
- static __inline__ Status_Control _CORE_recursive_mutex_Surrender (CORE_recursive_mutex_Control *the_mutex, const Thread_queue_Operations *operations, Thread_Control *executing, Thread_queue_Context *queue_context)

  *Surrenders the recursive mutex.*
- static __inline__ void _CORE_ceiling_mutex_Initialize (CORE_ceiling_mutex_Control *the_mutex, const Scheduler_Control *scheduler, Priority_Control priority_ceiling)

  *initializes a ceiling mutex.*
- static __inline__ const Scheduler_Control * _CORE_ceiling_mutex_Get_scheduler (const CORE_ceiling_mutex_Control *the_mutex)

  *Gets the scheduler of the ceiling mutex.*
- static __inline__ void _CORE_ceiling_mutex_Set_priority (CORE_ceiling_mutex_Control *the_mutex, Priority_Control priority_ceiling, Thread_queue_Context *queue_context)

  *Sets the priority of the ceiling mutex.*
- static __inline__ Priority_Control _CORE_ceiling_mutex_Get_priority (const CORE_ceiling_mutex_Control *the_mutex)

  *Gets the priority of the ceiling mutex.*
- static __inline__ Status_Control _CORE_ceiling_mutex_Set_owner (CORE_ceiling_mutex_Control *the_mutex, Thread_Control *owner, Thread_queue_Context *queue_context)

  *Sets the owner of the ceiling mutex.*
- static __inline__ Status_Control _CORE_ceiling_mutex_Seize (CORE_ceiling_mutex_Control *the_mutex, Thread_Control *executing, bool wait, Status_Control(*nested)(CORE_recursive_mutex_Control *), Thread_queue_Context *queue_context)

  *Seizes the ceiling mutex.*
- static __inline__ Status_Control _CORE_ceiling_mutex_Surrender (CORE_ceiling_mutex_Control *the_mutex, Thread_Control *executing, Thread_queue_Context *queue_context)

  *Surrenders the ceiling mutex.*

### 8.102.1 Detailed Description

Mutex Handler.

This handler encapsulates functionality which provides the foundation Mutex services used in all of the APIs supported by RTEMS.

### 8.102.2 Function Documentation

### 8.102.2.1 _CORE_ceiling_mutex_Get_priority()

```
static __inline__ Priority_Control _CORE_ceiling_mutex_Get_priority (
             const CORE_ceiling_mutex_Control * the_mutex )  [static]
```

Gets the priority of the ceiling mutex.

**Parameters**

| | |
|---|---|
| *the_mutex* | The mutex to get the priority from. |

**Returns**

The priority ceiling of *the_mutex*.

Definition at line 393 of file coremuteximpl.h.

### 8.102.2.2 _CORE_ceiling_mutex_Get_scheduler()

```
static __inline__ const Scheduler_Control* _CORE_ceiling_mutex_Get_scheduler (
             const CORE_ceiling_mutex_Control * the_mutex )  [static]
```

Gets the scheduler of the ceiling mutex.

**Parameters**

| | |
|---|---|
| *the_mutex* | The ceiling mutex to get the scheduler from. |

**Returns**

The scheduler of the mutex. If RTEMS_SMP is not defined, the first entry of the _Scheduler_Table is returned.

Definition at line 343 of file coremuteximpl.h.

### 8.102.2.3 _CORE_ceiling_mutex_Initialize()

```
static __inline__ void _CORE_ceiling_mutex_Initialize (
             CORE_ceiling_mutex_Control * the_mutex,
             const Scheduler_Control * scheduler,
             Priority_Control priority_ceiling )  [static]
```

initializes a ceiling mutex.

**Parameters**

| | | |
|---|---|---|
| out | *the_mutex* | The ceiling mutex to initialize. |
| | *scheduler* | The scheduler for the new ceiling mutex. Only needed if RTEMS_SMP is defined |
| | *priority_ceiling* | The priority ceiling for the initialized mutex. |

Definition at line 322 of file coremuteximpl.h.

#### 8.102.2.4 _CORE_ceiling_mutex_Seize()

```
static __inline__ Status_Control _CORE_ceiling_mutex_Seize (
            CORE_ceiling_mutex_Control * the_mutex,
            Thread_Control * executing,
            bool wait,
            Status_Control(*)(CORE_recursive_mutex_Control *) nested,
            Thread_queue_Context * queue_context )  [static]
```

Seizes the ceiling mutex.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_mutex* | The mutex to seize. |
| | *executing* | The executing thread. |
| | *wait* | Indicates whether the calling thread is willing to wait. |
| | *nested* | Function that returns the status of the recursive mutex |
| | *queue_context* | The thread queue context. |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The owner of the mutex was changed successfully. |
| *STATUS_NOT_DEFINED* | If the scheduler of the executing thread is not equal to the owner of *the_mutex* . |
| *STATUS_MUTEX_CEILING_VIOLATED* | The owners wait priority is smaller than the priority of the ceiling mutex. |
| *other* | Return value of *nested*. |

Definition at line 465 of file coremuteximpl.h.

#### 8.102.2.5 _CORE_ceiling_mutex_Set_owner()

```
static __inline__ Status_Control _CORE_ceiling_mutex_Set_owner (
            CORE_ceiling_mutex_Control * the_mutex,
            Thread_Control * owner,
            Thread_queue_Context * queue_context )  [static]
```

Sets the owner of the ceiling mutex.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_mutex* | The mutex to set the owner of. |
| | *owner* | The new owner of *the_mutex*. |
| | *queue_context* | The thread queue context. |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The owner of the mutex was changed successfully. |
| *STATUS_MUTEX_CEILING_VIOLATED* | The owners wait priority is smaller than the priority of the ceiling mutex. |

Definition at line 411 of file coremuteximpl.h.

#### 8.102.2.6 _CORE_ceiling_mutex_Set_priority()

```
static __inline__ void _CORE_ceiling_mutex_Set_priority (
            CORE_ceiling_mutex_Control * the_mutex,
            Priority_Control priority_ceiling,
            Thread_queue_Context * queue_context )  [static]
```

Sets the priority of the ceiling mutex.

**Parameters**

| | | |
|---|---|---|
| out | *the_mutex* | The ceiling mutex to set the priority of. |
| | *priority_ceiling* | The new priority ceiling of the mutex. |
| | *queue_context* | The thread queue context. |

Definition at line 361 of file coremuteximpl.h.

#### 8.102.2.7 _CORE_ceiling_mutex_Surrender()

```
static __inline__ Status_Control _CORE_ceiling_mutex_Surrender (
            CORE_ceiling_mutex_Control * the_mutex,
            Thread_Control * executing,
            Thread_queue_Context * queue_context )  [static]
```

Surrenders the ceiling mutex.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_mutex* | The ceiling mutex to surrender. |
| | *executing* | The executing thread. |
| | *queue_context* | The thread queue context. |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The ceiling mutex was successfullysurrendered. |
| *STATUS_NOT_OWNER* | The executing thread is not the owner of *the_mutex*. |

Definition at line 525 of file coremuteximpl.h.

### 8.102.2.8 _CORE_mutex_Acquire_critical()

```
static __inline__ void _CORE_mutex_Acquire_critical (
            CORE_mutex_Control * the_mutex,
            Thread_queue_Context * queue_context )  [static]
```

Acquires the mutex critical.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_mutex* | The mutex to acquire critical. |
| | *queue_context* | The queue context. |

Definition at line 71 of file coremuteximpl.h.

### 8.102.2.9 _CORE_mutex_Destroy()

```
static __inline__ void _CORE_mutex_Destroy (
            CORE_mutex_Control * the_mutex )  [static]
```

Destroys the mutex.

**Parameters**

| | | |
|---|---|---|
| out | *the_mutex* | the mutex to destroy. |

Definition at line 60 of file coremuteximpl.h.

### 8.102.2.10 _CORE_mutex_Get_owner()

```
static __inline__ Thread_Control* _CORE_mutex_Get_owner (
            const CORE_mutex_Control * the_mutex )  [static]
```

Gets the owner of the mutex.

**Parameters**

| | |
|---|---|
| *the_mutex* | The mutex to get the owner from. |

**Returns**

The owner of the mutex.

Definition at line 100 of file coremuteximpl.h.

### 8.102.2.11 _CORE_mutex_Initialize()

```
static __inline__ void _CORE_mutex_Initialize (
            CORE_mutex_Control * the_mutex )  [static]
```

Initializes the mutex.

**Parameters**

| | | |
|---|---|---|
| out | *the_mutex* | The mutex to initialize. |

Definition at line 48 of file coremuteximpl.h.

### 8.102.2.12 _CORE_mutex_Is_locked()

```
static __inline__ bool _CORE_mutex_Is_locked (
            const CORE_mutex_Control * the_mutex )  [static]
```

Checks if the mutex is locked.

This routine returns true if the specified mutex is locked and false otherwise.

**Parameters**

| | |
|---|---|
| *the_mutex* | The mutex to check if it is locked. |

**Return values**

| | |
|---|---|
| *true* | The mutex is locked. |
| *false* | The mutex is not locked. |

Definition at line 118 of file coremuteximpl.h.

### 8.102.2.13 _CORE_mutex_Is_owner()

```
static __inline__ bool _CORE_mutex_Is_owner (
            const CORE_mutex_Control * the_mutex,
            const Thread_Control * the_thread )  [static]
```

Checks if the the thread is the owner of the mutex.

**Parameters**

| | |
|---|---|
| *the_mutex* | The mutex to check the owner of. |
| *the_thread* | The thread to check if it is the owner of *the_mutex*. |

**Return values**

| | |
|---|---|
| *true* | *the_thread* is the owner of *the_mutex*. |
| *false* | *the_thread* is not the owner of *the_mutex*. |

Definition at line 168 of file coremuteximpl.h.

### 8.102.2.14 _CORE_mutex_Release()

```
static __inline__ void _CORE_mutex_Release (
            CORE_mutex_Control * the_mutex,
            Thread_queue_Context * queue_context )  [static]
```

Releases the mutex.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_mutex* | The mutex to release. |
| | *queue_context* | The queue context. |

Definition at line 85 of file coremuteximpl.h.

### 8.102.2.15 _CORE_mutex_Seize_slow()

```
Status_Control _CORE_mutex_Seize_slow (
            CORE_mutex_Control * the_mutex,
            const Thread_queue_Operations * operations,
            Thread_Control * executing,
            bool wait,
            Thread_queue_Context * queue_context )
```

Seize the mutex slowly.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_mutex* | The mutex to seize. |
| | *operations* | The thread queue operations. |
| | *executing* | The calling thread. |
| | *wait* | Indicates whether the calling thread is willing to wait. |
| | *queue_context* | The thread queue context. |

**Return values**

| | |
|---|---|
| *_Thread_Wait_get_status* | The status of the executing thread. |
| *STATUS_UNAVAILABLE* | The calling thread is not willing to wait. |

Definition at line 26 of file coremutexseize.c.

### 8.102.2.16 _CORE_mutex_Set_owner()

```
static __inline__ void _CORE_mutex_Set_owner (
              CORE_mutex_Control * the_mutex,
              Thread_Control * owner )  [static]
```

Sets the owner of the mutex.

**Parameters**

| | | |
|---|---|---|
| out | *the_mutex* | The mutex to set the owner from. |
| | *owner* | The new owner of the mutex. |

Definition at line 151 of file coremuteximpl.h.

### 8.102.2.17 _CORE_recursive_mutex_Initialize()

```
static __inline__ void _CORE_recursive_mutex_Initialize (
              CORE_recursive_mutex_Control * the_mutex )  [static]
```

Initializes a recursive mutex.

**Parameters**

| | | |
|---|---|---|
| out | *the_mutex* | The recursive mutex to initialize. |

Definition at line 181 of file coremuteximpl.h.

### 8.102.2.18 _CORE_recursive_mutex_Seize()

```
static __inline__ Status_Control _CORE_recursive_mutex_Seize (
            CORE_recursive_mutex_Control * the_mutex,
            const Thread_queue_Operations * operations,
            Thread_Control * executing,
            bool wait,
            Status_Control(*)(CORE_recursive_mutex_Control *) nested,
            Thread_queue_Context * queue_context )  [static]
```

Seizes the recursive mutex.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_mutex* | The recursive mutex to seize. |
| | *operations* | The thread queue operations. |
| out | *executing* | The executing thread. |
| | *wait* | Indicates whether the calling thread is willing to wait. |
| | *nested* | Returns the status of a recursive mutex. |
| | *queue_context* | The thread queue context. |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The owner of the mutex was NULL, successful seizing of the mutex. |
| *_Thread_Wait_get_status* | The status of the executing thread. |
| *STATUS_UNAVAILABLE* | The calling thread is not willing to wait. |

Definition at line 219 of file coremuteximpl.h.

### 8.102.2.19 _CORE_recursive_mutex_Seize_nested()

```
static __inline__ Status_Control _CORE_recursive_mutex_Seize_nested (
            CORE_recursive_mutex_Control * the_mutex )  [static]
```

Seizes the recursive mutex nested.

**Parameters**

| | | |
|---|---|---|
| out | *the_mutex* | The recursive mutex to seize nested. |

**Returns**

STATUS_SUCCESSFUL, this method is always successful.

Definition at line 196 of file coremuteximpl.h.

### 8.102.2.20 _CORE_recursive_mutex_Surrender()

```
static __inline__ Status_Control _CORE_recursive_mutex_Surrender (
              CORE_recursive_mutex_Control * the_mutex,
              const Thread_queue_Operations * operations,
              Thread_Control * executing,
              Thread_queue_Context * queue_context )  [static]
```

Surrenders the recursive mutex.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_mutex* | The recursive mutex to surrender. |
| | *operations* | The thread queue operations. |
| | *executing* | The executing thread. |
| | *queue_context* | the thread queue context. |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | *the_mutex* is successfully surrendered. |
| *STATUS_NOT_OWNER* | The executing thread does not own *the_mutex*. |

Definition at line 269 of file coremuteximpl.h.

## 8.103 Object Handler

### Files

- file object.h

  *Constants and Structures Associated with the Object Handler.*
- file objectdata.h

  *Object Handler Data Structures.*
- file objectimpl.h

  *Inlined Routines in the Object Handler.*
- file objectallocate.c

  *Allocate Object.*
- file objectallocatenone.c
- file objectallocatestatic.c
- file objectapimaximumclass.c

  *Object API Maximum Class.*
- file objectclose.c

  *Close Object.*
- file objectfree.c

  *Free Object.*
- file objectfreestatic.c
- file objectgetinfo.c

  *Get Object Information.*
- file objectgetlocal.c

  *Object Get Local.*
- file objectgetnoprotection.c

  *Get Object without Dispatching Protection.*
- file objectinitializeinformation.c

  *Initialize Object Information.*

### Classes

- union Objects_Name
- struct Objects_Control
- struct Objects_Information

  *The information structure used to manage each API class of objects.*

### Macros

- #define OBJECTS_INDEX_START_BIT 0U
- #define OBJECTS_NODE_START_BIT 16U
- #define OBJECTS_API_START_BIT 24U
- #define OBJECTS_CLASS_START_BIT 27U
- #define OBJECTS_INDEX_MASK (Objects_Id)0x0000ffffU
- #define OBJECTS_NODE_MASK (Objects_Id)0x00ff0000U
- #define OBJECTS_API_MASK (Objects_Id)0x07000000U
- #define OBJECTS_CLASS_MASK (Objects_Id)0xf8000000U
- #define OBJECTS_INDEX_VALID_BITS (Objects_Id)0x0000ffffU
- #define OBJECTS_NODE_VALID_BITS (Objects_Id)0x000000ffU
- #define OBJECTS_API_VALID_BITS (Objects_Id)0x00000007U

- #define OBJECTS_CLASS_VALID_BITS (Objects_Id)0x0000001fU
- #define OBJECTS_UNLIMITED_OBJECTS 0x80000000U
- #define OBJECTS_ID_INITIAL_INDEX (0)
- #define OBJECTS_ID_FINAL_INDEX (0xffffU)
- #define OBJECTS_APIS_LAST OBJECTS_POSIX_API
- #define OBJECTS_ID_NONE 0
- #define OBJECTS_ID_OF_SELF ((Objects_Id) 0)
- #define OBJECTS_SEARCH_ALL_NODES 0
- #define OBJECTS_SEARCH_OTHER_NODES 0x7FFFFFFE
- #define OBJECTS_SEARCH_LOCAL_NODE 0x7FFFFFFF
- #define OBJECTS_WHO_AM_I 0
- #define OBJECTS_ID_INITIAL(_api, _class, _node) _Objects_Build_id( (_api), (_class), (_node), OBJECTS_ID_INITIAL_INDEX )
- #define OBJECTS_ID_FINAL ((Objects_Id)~0)
- #define _Objects_Build_name(_C1, _C2, _C3, _C4)
- #define _Objects_Build_id(the_api, the_class, node, index)

  *Builds an object ID from its components.*
- #define _Objects_Is_unlimited(maximum) ( ( ( maximum ) & OBJECTS_UNLIMITED_OBJECTS ) != 0 )
- #define **_Objects_Maximum_per_allocation**(maximum) ((Objects_Maximum) ((maximum) & ~OBJECTS_UNLIMITED_OBJ
- #define _Objects_Local_node ((uint16_t) 1)

  *The local MPCI node number.*
- #define OBJECTS_NO_STRING_NAME 0

  *Constant for the object information string name length to indicate that this object class has no string names.*
- #define **OBJECTS_INFORMATION_MP**(name, extract)
- #define OBJECTS_INFORMATION_DEFINE_ZERO(name, api, cls, nl)

  *Statically initializes an objects information.*
- #define OBJECTS_INFORMATION_DEFINE(name, api, cls, type, max, nl, ex)

  *Statically initializes an objects information.*
- #define OBJECTS_INTERNAL_CLASSES_LAST OBJECTS_INTERNAL_THREADS
- #define OBJECTS_RTEMS_CLASSES_LAST OBJECTS_RTEMS_BARRIERS
- #define OBJECTS_POSIX_CLASSES_LAST OBJECTS_POSIX_SHMS
- #define _Objects_Maximum_nodes 1
- #define OBJECTS_INDEX_MINIMUM 1U
- #define OBJECTS_NAME_ERRORS_FIRST OBJECTS_NAME_OR_ID_LOOKUP_SUCCESSFUL
- #define OBJECTS_NAME_ERRORS_LAST OBJECTS_INVALID_NODE

## Typedefs

- typedef uint32_t Objects_Id
- typedef uint16_t Objects_Maximum
- typedef struct Objects_Information **Objects_Information**
- typedef bool(* Objects_Name_comparators) (void *, void *, uint16_t)

## Enumerations

- enum Objects_APIs {
  **OBJECTS_NO_API** = 0, **OBJECTS_INTERNAL_API** = 1, **OBJECTS_CLASSIC_API** = 2, **OBJECTS_PO↩**
  **SIX_API** = 3,
  **OBJECTS_FAKE_OBJECTS_API** = 7 }
- enum Objects_Internal_API { **OBJECTS_INTERNAL_NO_CLASS** = 0, **OBJECTS_INTERNAL_THREADS** = 1 }

- enum Objects_Classic_API {
  **OBJECTS_CLASSIC_NO_CLASS** = 0, **OBJECTS_RTEMS_TASKS** = 1, **OBJECTS_RTEMS_TIMERS**, **OBJECTS_RTEMS_SEMAPHORES**,
  **OBJECTS_RTEMS_MESSAGE_QUEUES**, **OBJECTS_RTEMS_PARTITIONS**, **OBJECTS_RTEMS_REGIONS**, **OBJECTS_RTEMS_PORTS**,
  **OBJECTS_RTEMS_PERIODS**, **OBJECTS_RTEMS_EXTENSIONS**, **OBJECTS_RTEMS_BARRIERS** }
- enum Objects_POSIX_API {
  **OBJECTS_POSIX_NO_CLASS** = 0, **OBJECTS_POSIX_THREADS** = 1, **OBJECTS_POSIX_KEYS**, **OBJECTS_POSIX_MESSAGE_QUEUES**,
  **OBJECTS_POSIX_SEMAPHORES**, **OBJECTS_POSIX_TIMERS**, **OBJECTS_POSIX_SHMS** }
- enum **Objects_Fake_objects_API** { **OBJECTS_FAKE_OBJECTS_NO_CLASS** = 0, **OBJECTS_FAKE_OBJECTS_SCHEDULERS** = 1 }
- enum Objects_Name_or_id_lookup_errors {
  **OBJECTS_NAME_OR_ID_LOOKUP_SUCCESSFUL**, **OBJECTS_INVALID_NAME**, **OBJECTS_INVALID_ADDRESS**, **OBJECTS_INVALID_ID**,
  **OBJECTS_INVALID_NODE** }
- enum **Objects_Get_by_name_error** { **OBJECTS_GET_BY_NAME_INVALID_NAME**, **OBJECTS_GET_BY_NAME_NAME_TOO_LONG**, **OBJECTS_GET_BY_NAME_NO_OBJECT** }

## Functions

- static __inline__ Objects_APIs _Objects_Get_API (Objects_Id id)

  *Returns the API portion of the ID.*
- static __inline__ uint32_t _Objects_Get_class (Objects_Id id)

  *Returns the class portion of the ID.*
- static __inline__ uint32_t _Objects_Get_node (Objects_Id id)

  *Returns the node portion of the ID.*
- static __inline__ Objects_Maximum _Objects_Get_index (Objects_Id id)

  *Returns the index portion of the ID.*
- Objects_Control * _Objects_Allocate_none (Objects_Information *information)

  *Always return NULL.*
- Objects_Control * _Objects_Allocate_static (Objects_Information *information)

  *Return an inactive object or NULL.*
- Objects_Control * _Objects_Allocate_unlimited (Objects_Information *information)

  *Return an inactive object or NULL.*
- void _Objects_Free_static (Objects_Information *information, Objects_Control *the_object)

  *Free the object.*
- void _Objects_Free_unlimited (Objects_Information *information, Objects_Control *the_object)

  *Free the object.*
- Objects_Maximum _Objects_Extend_information (Objects_Information *information)

  *Extends an object class information record.*
- void _Objects_Free_objects_block (Objects_Information *information, Objects_Maximum block)

  *Free the objects block with the specified index.*
- void _Objects_Shrink_information (Objects_Information *information)

  *Shrinks an object class information record.*
- void _Objects_Initialize_information (Objects_Information *information)

  *Initializes the specified objects information.*
- unsigned int _Objects_API_maximum_class (uint32_t api)

  *Returns highest numeric value of a valid API for the specified API.*
- Objects_Control * _Objects_Allocate (Objects_Information *information)

  *Allocates an object.*

- Objects_Name_or_id_lookup_errors _Objects_Name_to_id_u32 (uint32_t name, uint32_t node, Objects_Id *id, const Objects_Information *information)

  *Searches an object of the specified class with the specified name on the specified set of nodes.*
- Objects_Control * _Objects_Get_by_name (const Objects_Information *information, const char *name, size_t *name_length_p, Objects_Get_by_name_error *error)

  *Gets an object control block identified by its name.*
- Objects_Name_or_id_lookup_errors _Objects_Id_to_name (Objects_Id id, Objects_Name *name)

  *Returns the name associated with object id.*
- Objects_Control * _Objects_Get (Objects_Id id, ISR_lock_Context *lock_context, const Objects_Information *information)

  *Maps the specified object identifier to the associated local object control block.*
- Objects_Control * _Objects_Get_no_protection (Objects_Id id, const Objects_Information *information)

  *Maps object ids to object control blocks.*
- Objects_Control * _Objects_Get_next (Objects_Id id, const Objects_Information *information, Objects_Id *next_id_p)

  *Gets the next open object after the specified object identifier.*
- Objects_Information * _Objects_Get_information (Objects_APIs the_api, uint16_t the_class)

  *Gets object information.*
- Objects_Information * _Objects_Get_information_id (Objects_Id id)

  *Gets information of an object from an ID.*
- static __inline__ bool _Objects_Has_string_name (const Objects_Information *information)

  *Returns if the object has a string name.*
- char * _Objects_Get_name_as_string (Objects_Id id, size_t length, char *name)

  *Gets object name in the form of a C string.*
- size_t _Objects_Name_to_string (Objects_Name name, bool is_string, char *buffer, size_t buffer_size)

  *Converts the specified object name to a text representation.*
- bool _Objects_Set_name (const Objects_Information *information, Objects_Control *the_object, const char *name)

  *Sets objects name.*
- static __inline__ void _Objects_Namespace_remove_u32 (const Objects_Information *information, Objects_Control *the_object)

  *Removes object with a 32-bit integer name from its namespace.*
- void _Objects_Namespace_remove_string (const Objects_Information *information, Objects_Control *the_object)

  *Removes object with a string name from its namespace.*
- void _Objects_Close (const Objects_Information *information, Objects_Control *the_object)

  *Closes object.*
- Objects_Maximum _Objects_Active_count (const Objects_Information *information)

  *Returns the count of active objects.*
- static __inline__ Objects_Maximum _Objects_Extend_size (const Objects_Information *information)

  *Returns the object's objects per block.*
- static __inline__ bool _Objects_Is_api_valid (uint32_t the_api)

  *Checks if the api is valid.*
- static __inline__ bool _Objects_Is_local_node (uint32_t node)

  *Checks if the node is of the local object.*
- static __inline__ bool _Objects_Is_local_id (Objects_Id id RTEMS_UNUSED)

  *Checks if the id is of a local object.*
- static __inline__ bool _Objects_Are_ids_equal (Objects_Id left, Objects_Id right)

  *Checks if two object IDs are equal.*
- static __inline__ Objects_Id _Objects_Get_minimum_id (Objects_Id id)

  *Returns the identifier with the minimum index for the specified identifier.*
- static __inline__ Objects_Maximum _Objects_Get_maximum_index (const Objects_Information *information)

*Returns the maximum index of the specified object class.*

- static __inline__ Objects_Control * _Objects_Get_inactive (Objects_Information *information)

  *Get an inactive object or NULL.*

- static __inline__ Objects_Maximum _Objects_Is_auto_extend (const Objects_Information *information)

  *Checks if the automatic object extension (unlimited objects) is enabled.*

- static __inline__ void _Objects_Set_local_object (const Objects_Information *information, uint32_t index, Objects_Control *the_object)

  *Sets the pointer to the local_table object referenced by the index.*

- static __inline__ void _Objects_Invalidate_Id (const Objects_Information *information, Objects_Control *the_object)

  *Invalidates an object Id.*

- static __inline__ void _Objects_Open (Objects_Information *information, Objects_Control *the_object, Objects_Name name)

  *Places the_object control pointer and object name in the Local Pointer and Local Name Tables, respectively.*

- static __inline__ void _Objects_Open_u32 (const Objects_Information *information, Objects_Control *the_object, uint32_t name)

  *Places the_object control pointer and object name in the Local Pointer and Local Name Tables, respectively.*

- static __inline__ void _Objects_Open_string (const Objects_Information *information, Objects_Control *the_object, const char *name)

  *Places the_object control pointer and object name in the Local Pointer and Local Name Tables, respectively.*

- static __inline__ void _Objects_Allocator_lock (void)

  *Locks the object allocator mutex.*

- static __inline__ void _Objects_Allocator_unlock (void)

  *Unlocks the object allocator mutex.*

- static __inline__ bool _Objects_Allocator_is_owner (void)

  *Checks if the allocator is the owner of the object allocator mutex.*

- static __inline__ Objects_Control * _Objects_Allocate_unprotected (Objects_Information *information)

  *Allocates an object without locking the allocator mutex.*

- static __inline__ void _Objects_Free (Objects_Information *information, Objects_Control *the_object)

  *Frees an object.*

- static __inline__ void _Objects_Activate_unlimited (Objects_Information *information, Objects_Control *the_object)

  *Activate the object.*

- static __inline__ Objects_Control * _Objects_Allocate_with_extend (Objects_Information *information, void(*extend)(Objects_Information *))

  *Allocate an object and extend the objects information on demand.*

## Variables

- Objects_Information **const _Objects_Information_table [OBJECTS_APIS_LAST+1]

### 8.103.1 Detailed Description

### 8.103.2 Macro Definition Documentation

### 8.103.2.1 _Objects_Build_id

```
#define _Objects_Build_id(
              the_api,
              the_class,
              node,
              index )
```

**Value:**

```
  ( (Objects_Id) ( (Objects_Id) the_api   « OBJECTS_API_START_BIT )   | \
                 ( (Objects_Id) the_class « OBJECTS_CLASS_START_BIT ) | \
                 ( (Objects_Id) node      « OBJECTS_NODE_START_BIT )  | \
                 ( (Objects_Id) index     « OBJECTS_INDEX_START_BIT ) )
```

Builds an object ID from its components.

**Parameters**

| | |
|---|---|
| *the_api* | The object API. |
| *the_class* | The object API class. |
| *node* | The object node. |
| *index* | The object index. |

**Returns**

Returns the object ID constructed from the arguments.

Definition at line 317 of file object.h.

### 8.103.2.2 _Objects_Build_name

```
#define _Objects_Build_name(
              _C1,
              _C2,
              _C3,
              _C4 )
```

**Value:**

```
  ( (uint32_t)(_C1) « 24 | \
    (uint32_t)(_C2) « 16 | \
    (uint32_t)(_C3) « 8 | \
    (uint32_t)(_C4) )
```

This macro is used to build a thirty-two bit style name from four characters. The most significant byte will be the character *_C1*.

**Parameters**

| | | |
|---|---|---|
| in | *_C1* | is the first character of the name |
| in | *_C2* | is the second character of the name |
| in | *_C3* | is the third character of the name |
| in | *_C4* | is the fourth character of the name |

Definition at line 242 of file object.h.

#### 8.103.2.3 _Objects_Is_unlimited

```
#define _Objects_Is_unlimited(
              maximum )  ( ( ( maximum ) & OBJECTS_UNLIMITED_OBJECTS ) != 0 )
```

Returns if the object maximum specifies unlimited objects.

**Parameters**

| | | |
|---|---|---|
| in | *maximum* | The object maximum specification. |

**Return values**

| | |
|---|---|
| *true* | Unlimited objects are available. |
| *false* | The object count is fixed. |

Definition at line 331 of file object.h.

#### 8.103.2.4 _Objects_Maximum_nodes

```
#define _Objects_Maximum_nodes 1
```

The following is referenced to the number of nodes in the system.

Definition at line 74 of file objectimpl.h.

#### 8.103.2.5 OBJECTS_API_MASK

```
#define OBJECTS_API_MASK (Objects_Id)0x07000000U
```

This mask is used to extract the API portion of an object Id.

Definition at line 124 of file object.h.

#### 8.103.2.6 OBJECTS_API_START_BIT

```
#define OBJECTS_API_START_BIT 24U
```

This is the bit position of the starting bit of the API portion of the object Id.

Definition at line 103 of file object.h.

#### 8.103.2.7 OBJECTS_API_VALID_BITS

```
#define OBJECTS_API_VALID_BITS (Objects_Id)0x00000007U
```

This mask represents the bits that is used to ensure no extra bits are set after shifting to extract the API portion of an object Id.

Definition at line 147 of file object.h.

#### 8.103.2.8 OBJECTS_APIS_LAST

```
#define OBJECTS_APIS_LAST OBJECTS_POSIX_API
```

This macro is used to generically specify the last API index.

Definition at line 183 of file object.h.

#### 8.103.2.9 OBJECTS_CLASS_MASK

```
#define OBJECTS_CLASS_MASK (Objects_Id)0xf8000000U
```

This mask is used to extract the class portion of an object Id.

Definition at line 129 of file object.h.

#### 8.103.2.10 OBJECTS_CLASS_START_BIT

```
#define OBJECTS_CLASS_START_BIT 27U
```

This is the bit position of the starting bit of the class portion of the object Id.

Definition at line 109 of file object.h.

#### 8.103.2.11 OBJECTS_CLASS_VALID_BITS

```
#define OBJECTS_CLASS_VALID_BITS (Objects_Id)0x0000001fU
```

This mask represents the bits that is used to ensure no extra bits are set after shifting to extract the class portion of an object Id.

Definition at line 153 of file object.h.

#### 8.103.2.12 OBJECTS_ID_FINAL

```
#define OBJECTS_ID_FINAL ((Objects_Id)~0)
```

This macro specifies the highest object ID value

Definition at line 230 of file object.h.

#### 8.103.2.13 OBJECTS_ID_FINAL_INDEX

```
#define OBJECTS_ID_FINAL_INDEX (0xffffU)
```

This is the highest value for the index portion of an object Id.

Definition at line 169 of file object.h.

#### 8.103.2.14 OBJECTS_ID_INITIAL

```
#define OBJECTS_ID_INITIAL(
              _api,
              _class,
              _node )  _Objects_Build_id( (_api), (_class), (_node), OBJECTS_ID_INITIAL_INDEX
)
```

This macros calculates the lowest ID for the specified api, class, and node.

Definition at line 224 of file object.h.

#### 8.103.2.15 OBJECTS_ID_INITIAL_INDEX

```
#define OBJECTS_ID_INITIAL_INDEX (0)
```

This is the lowest value for the index portion of an object Id.

Definition at line 164 of file object.h.

#### 8.103.2.16 OBJECTS_ID_NONE

```
#define OBJECTS_ID_NONE 0
```

No object can have this ID.

Definition at line 188 of file object.h.

#### 8.103.2.17 OBJECTS_ID_OF_SELF

```
#define OBJECTS_ID_OF_SELF ((Objects_Id) 0)
```

The following defines the constant which may be used to manipulate the calling task.

Definition at line 194 of file object.h.

#### 8.103.2.18 OBJECTS_INDEX_MASK

```
#define OBJECTS_INDEX_MASK (Objects_Id)0x0000ffffU
```

This mask is used to extract the index portion of an object Id.

Definition at line 114 of file object.h.

#### 8.103.2.19 OBJECTS_INDEX_MINIMUM

```
#define OBJECTS_INDEX_MINIMUM 1U
```

This is the minimum object ID index associated with an object.

Definition at line 80 of file objectimpl.h.

#### 8.103.2.20 OBJECTS_INDEX_START_BIT

```
#define OBJECTS_INDEX_START_BIT 0U
```

This is the bit position of the starting bit of the index portion of the object Id.

Definition at line 92 of file object.h.

#### 8.103.2.21 OBJECTS_INDEX_VALID_BITS

```
#define OBJECTS_INDEX_VALID_BITS (Objects_Id)0x0000ffffU
```

This mask represents the bits that is used to ensure no extra bits are set after shifting to extract the index portion of an object Id.

Definition at line 135 of file object.h.

### 8.103.2.22 OBJECTS_INFORMATION_DEFINE

```
#define OBJECTS_INFORMATION_DEFINE(
                name,
                api,
                cls,
                type,
                max,
                nl,
                ex )
```

**Value:**

```
static Objects_Control * \
name##_Local_table[ _Objects_Maximum_per_allocation( max ) ]; \
static type name##_Objects[ _Objects_Maximum_per_allocation( max ) ]; \
Objects_Information name##_Information = { \
  _Objects_Build_id( api, cls, 1, _Objects_Maximum_per_allocation( max ) ), \
  name##_Local_table, \
  _Objects_Is_unlimited( max ) ? \
    _Objects_Allocate_unlimited : _Objects_Allocate_static, \
  _Objects_Is_unlimited( max ) ? \
    _Objects_Free_unlimited : _Objects_Free_static, \
  0, \
  _Objects_Is_unlimited( max ) ? _Objects_Maximum_per_allocation( max ) : 0, \
  sizeof( type ), \
  nl, \
  CHAIN_INITIALIZER_EMPTY( name##_Information.Inactive ), \
  NULL, \
  NULL, \
  &name##_Objects[ 0 ].Object \
  OBJECTS_INFORMATION_MP( name##_Information, ex ) \
}
```

Statically initializes an objects information.

The initialized objects information references a table with statically allocated objects as specified by the object maximum parameter. These objects must be registered via a call to _Objects_Information().

**Parameters**

| | |
|---|---|
| *name* | The object class C designator namespace prefix, e.g. _Semaphore. |
| *api* | The object API number, e.g. OBJECTS_CLASSIC_API. |
| *cls* | The object class number, e.g. OBJECTS_RTEMS_SEMAPHORES. |
| *type* | The object class type. |
| *max* | The configured object maximum (the OBJECTS_UNLIMITED_OBJECTS flag may be set). |
| *nl* | The object name string length, use OBJECTS_NO_STRING_NAME for objects without a string name. |
| *ex* | The optional object extraction method. Used only if multiprocessing (RTEMS_MULTIPROCESSING) is enabled. |

Definition at line 427 of file objectdata.h.

### 8.103.2.23 OBJECTS_INFORMATION_DEFINE_ZERO

```
#define OBJECTS_INFORMATION_DEFINE_ZERO(
                name,
                api,
                cls,
                nl )
```

**Value:**

```
Objects_Information name##_Information = { \
  _Objects_Build_id( api, cls, 1, 0 ), \
  NULL, \
  _Objects_Allocate_none, \
  NULL, \
  0, \
  0, \
  0, \
  nl, \
  CHAIN_INITIALIZER_EMPTY( name##_Information.Inactive ), \
  NULL, \
  NULL, \
  NULL \
  OBJECTS_INFORMATION_MP( name##_Information, NULL ) \
}
```

Statically initializes an objects information.

The initialized objects information contains no objects.

**Parameters**

| | |
|---|---|
| *name* | The object class C designator namespace prefix, e.g. _Semaphore. |
| *api* | The object API number, e.g. OBJECTS_CLASSIC_API. |
| *cls* | The object class number, e.g. OBJECTS_RTEMS_SEMAPHORES. |
| *nl* | The object name string length, use OBJECTS_NO_STRING_NAME for objects without a string name. |

Definition at line 392 of file objectdata.h.

#### 8.103.2.24 OBJECTS_INTERNAL_CLASSES_LAST

```
#define OBJECTS_INTERNAL_CLASSES_LAST OBJECTS_INTERNAL_THREADS
```

This macro is used to generically specify the last API index.

Definition at line 51 of file objectimpl.h.

#### 8.103.2.25 OBJECTS_NAME_ERRORS_FIRST

```
#define OBJECTS_NAME_ERRORS_FIRST OBJECTS_NAME_OR_ID_LOOKUP_SUCCESSFUL
```

This macro defines the first entry in the Objects_Name_or_id_lookup_errors enumerated list.

Definition at line 215 of file objectimpl.h.

#### 8.103.2.26 OBJECTS_NAME_ERRORS_LAST

```
#define OBJECTS_NAME_ERRORS_LAST OBJECTS_INVALID_NODE
```

This macro defines the last entry in the Objects_Name_or_id_lookup_errors enumerated list.

Definition at line 221 of file objectimpl.h.

#### 8.103.2.27 OBJECTS_NODE_MASK

```
#define OBJECTS_NODE_MASK (Objects_Id)0x00ff0000U
```

This mask is used to extract the node portion of an object Id.

Definition at line 119 of file object.h.

#### 8.103.2.28 OBJECTS_NODE_START_BIT

```
#define OBJECTS_NODE_START_BIT 16U
```

This is the bit position of the starting bit of the node portion of the object Id.

Definition at line 97 of file object.h.

#### 8.103.2.29 OBJECTS_NODE_VALID_BITS

```
#define OBJECTS_NODE_VALID_BITS (Objects_Id)0x000000ffU
```

This mask represents the bits that is used to ensure no extra bits are set after shifting to extract the node portion of an object Id.

Definition at line 141 of file object.h.

#### 8.103.2.30 OBJECTS_POSIX_CLASSES_LAST

```
#define OBJECTS_POSIX_CLASSES_LAST OBJECTS_POSIX_SHMS
```

This macro is used to generically specify the last API index.

Definition at line 57 of file objectimpl.h.

#### 8.103.2.31 OBJECTS_RTEMS_CLASSES_LAST

```
#define OBJECTS_RTEMS_CLASSES_LAST OBJECTS_RTEMS_BARRIERS
```

This macro is used to generically specify the last API index.

Definition at line 54 of file objectimpl.h.

#### 8.103.2.32 OBJECTS_SEARCH_ALL_NODES

```
#define OBJECTS_SEARCH_ALL_NODES 0
```

The following constant is used to specify that a name to ID search should search through all nodes.

Definition at line 200 of file object.h.

#### 8.103.2.33 OBJECTS_SEARCH_LOCAL_NODE

```
#define OBJECTS_SEARCH_LOCAL_NODE 0x7FFFFFFF
```

The following constant is used to specify that a name to ID search should search only on this node.

Definition at line 212 of file object.h.

#### 8.103.2.34 OBJECTS_SEARCH_OTHER_NODES

```
#define OBJECTS_SEARCH_OTHER_NODES 0x7FFFFFFE
```

The following constant is used to specify that a name to ID search should search through all nodes except the current node.

Definition at line 206 of file object.h.

#### 8.103.2.35 OBJECTS_UNLIMITED_OBJECTS

```
#define OBJECTS_UNLIMITED_OBJECTS 0x80000000U
```

Mask to enable unlimited objects. This is used in the configuration table when specifying the number of configured objects.

Definition at line 159 of file object.h.

#### 8.103.2.36 OBJECTS_WHO_AM_I

```
#define OBJECTS_WHO_AM_I 0
```

The following constant is used to specify that a name to ID search is being asked for the ID of the currently executing task.

Definition at line 218 of file object.h.

## 8.103.3 Typedef Documentation

### 8.103.3.1 Objects_Id

```
typedef uint32_t Objects_Id
```

The following type defines the control block used to manage object IDs. The format is as follows (0=LSB):

Bits 0 .. 15 = index (up to 65535 objects of a type) Bits 16 .. 23 = node (up to 255 nodes) Bits 24 .. 26 = API (up to 7 API classes) Bits 27 .. 31 = class (up to 31 object types per API)

Definition at line 80 of file object.h.

### 8.103.3.2 Objects_Maximum

```
typedef uint16_t Objects_Maximum
```

This type is used to store the maximum number of allowed objects of each type.

Definition at line 86 of file object.h.

### 8.103.3.3 Objects_Name_comparators

```
typedef bool(* Objects_Name_comparators) (void *, void *, uint16_t)
```

Functions which compare names are prototyped like this.

Definition at line 44 of file objectimpl.h.

## 8.103.4 Enumeration Type Documentation

### 8.103.4.1 Objects_APIs

```
enum Objects_APIs
```

This enumerated type is used in the class field of the object ID.

Definition at line 174 of file object.h.

#### 8.103.4.2 Objects_Classic_API

enum Objects_Classic_API

This enumerated type is used in the class field of the object ID for the RTEMS Classic API.

Definition at line 63 of file objectdata.h.

#### 8.103.4.3 Objects_Internal_API

enum Objects_Internal_API

This enumerated type is used in the class field of the object ID for RTEMS internal object classes.

Definition at line 52 of file objectdata.h.

#### 8.103.4.4 Objects_Name_or_id_lookup_errors

enum Objects_Name_or_id_lookup_errors

This function implements the common portion of the object identification directives. This directive returns the object id associated with name. If more than one object of this class is named name, then the object to which the id belongs is arbitrary. Node indicates the extent of the search for the id of the object named name. If the object class supports global objects, then the search can be limited to a particular node or allowed to encompass all nodes.

Definition at line 203 of file objectimpl.h.

#### 8.103.4.5 Objects_POSIX_API

enum Objects_POSIX_API

This enumerated type is used in the class field of the object ID for the POSIX API.

Definition at line 84 of file objectdata.h.

### 8.103.5 Function Documentation

#### 8.103.5.1 _Objects_Activate_unlimited()

```
static __inline__ void _Objects_Activate_unlimited (
             Objects_Information * information,
             Objects_Control * the_object )  [static]
```

Activate the object.

This function must be only used in case this objects information supports unlimited objects.

**Parameters**

| | |
|---|---|
| *information* | The object information block. |
| *the_object* | The object to activate. |

Definition at line 952 of file objectimpl.h.

### 8.103.5.2 _Objects_Active_count()

```
Objects_Maximum _Objects_Active_count (
            const Objects_Information * information )
```

Returns the count of active objects.

**Parameters**

| | |
|---|---|
| *information* | The object information table. |

**Returns**

The count of active objects.

Definition at line 23 of file objectactivecount.c.

### 8.103.5.3 _Objects_Allocate()

```
Objects_Control* _Objects_Allocate (
            Objects_Information * information )
```

Allocates an object.

This function locks the object allocator mutex via _Objects_Allocator_lock(). The caller must later unlock the object allocator mutex via _Objects_Allocator_unlock(). The caller must unlock the mutex in any case, even if the allocation failed due to resource shortage.

A typical object allocation code looks like this:

```
rtems_status_code some_create( rtems_id *id )
{
  rtems_status_code  sc;
  Some_Control      *some;
  // The object allocator mutex protects the executing thread from
  // asynchronous thread restart and deletion.
  some = (Some_Control *) _Objects_Allocate( &_Some_Information );
  if ( some != NULL ) {
    _Some_Initialize( some );
    sc = RTEMS_SUCCESSFUL;
  } else {
    sc = RTEMS_TOO_MANY;
  }
  _Objects_Allocator_unlock();
  return sc;
}
```

**Parameters**

| | | |
|---|---|---|
| in,out | *information* | The object information block. |

**Return values**

| | |
|---|---|
| *object* | The allocated object. |
| *NULL* | No object available. |

**See also**

_Objects_Free().

Definition at line 42 of file objectallocate.c.

#### 8.103.5.4 _Objects_Allocate_none()

```
Objects_Control* _Objects_Allocate_none (
            Objects_Information * information )
```

Always return NULL.

**Parameters**

| | |
|---|---|
| *information* | The objects information. |

**Return values**

| | |
|---|---|
| *NULL* | Always. |

Definition at line 40 of file objectallocatenone.c.

#### 8.103.5.5 _Objects_Allocate_static()

```
Objects_Control* _Objects_Allocate_static (
            Objects_Information * information )
```

Return an inactive object or NULL.

**Parameters**

| | |
|---|---|
| *information* | The objects information. |

**Return values**

| | |
|---|---|
| *NULL* | No inactive object is available. |
| *object* | An inactive object. |

Definition at line 41 of file objectallocatestatic.c.

### 8.103.5.6 _Objects_Allocate_unlimited()

```
Objects_Control* _Objects_Allocate_unlimited (
            Objects_Information * information )
```

Return an inactive object or NULL.

Try to extend the objects information if necessary.

**Parameters**

| | |
|---|---|
| *information* | The objects information. |

**Return values**

| | |
|---|---|
| *NULL* | No inactive object is available. |
| *object* | An inactive object. |

### 8.103.5.7 _Objects_Allocate_unprotected()

```
static __inline__ Objects_Control* _Objects_Allocate_unprotected (
            Objects_Information * information )  [static]
```

Allocates an object without locking the allocator mutex.

This function can be called in two contexts

- the executing thread is the owner of the object allocator mutex, or
- in case the system state is not up, e.g. during sequential system initialization.

**Parameters**

| | | |
|---|---|---|
| in,out | *information* | The object information block. |

**Return values**

| | |
|---|---|
| *object* | The allocated object. |
| *NULL* | No object available. |

**See also**

_Objects_Allocate() and _Objects_Free().

Definition at line 877 of file objectimpl.h.

### 8.103.5.8 _Objects_Allocate_with_extend()

```
static __inline__ Objects_Control* _Objects_Allocate_with_extend (
            Objects_Information * information,
            void(*)(Objects_Information *) extend )  [static]
```

Allocate an object and extend the objects information on demand.

This function must be only used in case this objects information supports unlimited objects.

**Parameters**

| | |
|---|---|
| *information* | The object information block. |
| *extend* | The object information extend handler. |

Definition at line 982 of file objectimpl.h.

### 8.103.5.9 _Objects_Allocator_is_owner()

```
static __inline__ bool _Objects_Allocator_is_owner (
            void  )  [static]
```

Checks if the allocator is the owner of the object allocator mutex.

**Return values**

| | |
|---|---|
| *true* | The allocator is the owner of the object allocator mutex. |
| *false* | The allocato is not the owner of the object allocator mutex. |

Definition at line 857 of file objectimpl.h.

#### 8.103.5.10 _Objects_Allocator_lock()

```
static __inline__ void _Objects_Allocator_lock (
             void  )  [static]
```

Locks the object allocator mutex.

While holding the allocator mutex the executing thread is protected from asynchronous thread restart and deletion.

The usage of the object allocator mutex with the thread life protection makes it possible to allocate and free objects without thread dispatching disabled. The usage of a unified workspace and unlimited objects may lead to heap fragmentation. Thus the execution time of the _Objects_Allocate() function may increase during system run-time.

**See also**

_Objects_Allocator_unlock() and _Objects_Allocate().

Definition at line 834 of file objectimpl.h.

#### 8.103.5.11 _Objects_Allocator_unlock()

```
static __inline__ void _Objects_Allocator_unlock (
             void  )  [static]
```

Unlocks the object allocator mutex.

In case the mutex is fully unlocked, then this function restores the previous thread life protection state and thus may not return if the executing thread was restarted or deleted in the mean-time.

Definition at line 846 of file objectimpl.h.

#### 8.103.5.12 _Objects_API_maximum_class()

```
unsigned int _Objects_API_maximum_class (
             uint32_t api )
```

Returns highest numeric value of a valid API for the specified API.

This function returns the highest numeric value of a valid API for the specified *api*.

**Parameters**

| | |
|---|---|
| *api* | The API of interest. |

**Return values**

| | |
|---|---|
| *some_value* | Positive integer on success. |

**Return values**

| | |
|---|---|
| *0* | The method failed. |

Definition at line 23 of file objectapimaximumclass.c.

### 8.103.5.13 _Objects_Are_ids_equal()

```
static __inline__ bool _Objects_Are_ids_equal (
             Objects_Id left,
             Objects_Id right )  [static]
```

Checks if two object IDs are equal.

**Parameters**

| | |
|---|---|
| *left* | The Id on the left hand side of the comparison. |
| *right* | The Id on the right hand side of the comparison. |

**Return values**

| | |
|---|---|
| *true* | The specified object IDs are equal. |
| *false* | The specified object IDs are not equal. |

Definition at line 611 of file objectimpl.h.

### 8.103.5.14 _Objects_Close()

```
void _Objects_Close (
             const Objects_Information * information,
             Objects_Control * the_object )
```

Closes object.

This function removes the_object control pointer and object name in the Local Pointer and Local Name Tables.

**Parameters**

| | | |
|---|---|---|
| | *information* | Points to an Object Information Table. |
| out | *the_object* | A pointer to an object. |

Definition at line 23 of file objectclose.c.

#### 8.103.5.15 _Objects_Extend_information()

```
Objects_Maximum _Objects_Extend_information (
            Objects_Information * information )
```

Extends an object class information record.

**Parameters**

| | |
|---|---|
| *information* | Points to an object class information block. |

**Return values**

| | |
|---|---|
| *0* | The extend operation failed. |
| *block* | The block index of the new objects block. |

#### 8.103.5.16 _Objects_Extend_size()

```
static __inline__ Objects_Maximum _Objects_Extend_size (
            const Objects_Information * information )  [static]
```

Returns the object's objects per block.

**Parameters**

| | |
|---|---|
| *information* | The object information table. |

**Returns**

The number of objects per block of *information*.

Definition at line 537 of file objectimpl.h.

#### 8.103.5.17 _Objects_Free()

```
static __inline__ void _Objects_Free (
            Objects_Information * information,
            Objects_Control * the_object )  [static]
```

Frees an object.

Appends the object to the chain of inactive objects.

**Parameters**

| | | |
|---|---|---|
| | *information* | The object information block. |
| out | *the_object* | The object to free. |

**See also**

_Objects_Allocate().

A typical object deletion code looks like this:

```
rtems_status_code some_delete( rtems_id id )
{
  Some_Control      *some;
  // The object allocator mutex protects the executing thread from
  // asynchronous thread restart and deletion.
  _Objects_Allocator_lock();
  // Get the object under protection of the object allocator mutex.
  some = (Semaphore_Control *)
    _Objects_Get_no_protection( id, &_Some_Information );
  if ( some == NULL ) {
    _Objects_Allocator_unlock();
    return RTEMS_INVALID_ID;
  }
  // After the object close an object get with this identifier will
  // fail.
  _Objects_Close( &_Some_Information, &some->Object );
  _Some_Delete( some );
  // Thread dispatching is enabled.  The object free is only protected
  // by the object allocator mutex.
  _Objects_Free( &_Some_Information, &some->Object );
  _Objects_Allocator_unlock();
  return RTEMS_SUCCESSFUL;
}
```

Definition at line 933 of file objectimpl.h.

### 8.103.5.18 _Objects_Free_objects_block()

```
void _Objects_Free_objects_block (
              Objects_Information * information,
              Objects_Maximum block )
```

Free the objects block with the specified index.

**Parameters**

| | |
|---|---|
| *information* | The objects information. |
| *block* | The block index. |

### 8.103.5.19 _Objects_Free_static()

```
void _Objects_Free_static (
              Objects_Information * information,
              Objects_Control * the_object )
```

Free the object.

Append the object to the inactive chain of the objects information.

**Parameters**

| | |
|---|---|
| *information* | The objects information. |
| *the_object* | The object to free. |

Definition at line 41 of file objectfreestatic.c.

#### 8.103.5.20 _Objects_Free_unlimited()

```
void _Objects_Free_unlimited (
             Objects_Information * information,
             Objects_Control * the_object )
```

Free the object.

Append the object to the inactive chain of the objects information and shrink the objects information if necessary.

**Parameters**

| | |
|---|---|
| *information* | The objects information. |
| *the_object* | The object to free. |

Definition at line 24 of file objectfree.c.

#### 8.103.5.21 _Objects_Get()

```
Objects_Control* _Objects_Get (
             Objects_Id id,
             ISR_lock_Context * lock_context,
             const Objects_Information * information )
```

Maps the specified object identifier to the associated local object control block.

In this function interrupts are disabled during the object lookup. In case an associated object exists, then interrupts remain disabled, otherwise the previous interrupt state is restored.

**Parameters**

| | |
|---|---|
| *id* | The object identifier. This is the first parameter since usual callers get the object identifier as the first parameter themself. |
| *lock_context* | The interrupt lock context. This is the second parameter since usual callers get the interrupt lock context as the second parameter themself. |
| *information* | The object class information block. |

**Return values**

| | |
|---|---|
| *pointer* | The pointer to the associated object control block. Interrupts are now disabled and must be restored using the specified lock context via _ISR_lock_ISR_enable() or _ISR_lock_Release_and_ISR_enable(). |
| *NULL* | No associated object exists. |

Definition at line 28 of file objectgetlocal.c.

### 8.103.5.22 _Objects_Get_API()

```
static __inline__ Objects_APIs _Objects_Get_API (
             Objects_Id id )  [static]
```

Returns the API portion of the ID.

**Parameters**

| | |
|---|---|
| *id* | The object Id to be processed. |

**Returns**

An object Id constructed from the arguments.

Definition at line 255 of file object.h.

### 8.103.5.23 _Objects_Get_by_name()

```
Objects_Control* _Objects_Get_by_name (
             const Objects_Information * information,
             const char * name,
             size_t * name_length_p,
             Objects_Get_by_name_error * error )
```

Gets an object control block identified by its name.

The object information must use string names.

**Parameters**

| | | |
|---|---|---|
| | *information* | The object information. Must not be NULL. |
| | *name* | The object name. |
| out | *name_length_p* | Optional parameter to return the name length. |
| out | *error* | The error indication in case of failure. Must not be NULL. |

**Return values**

| | |
|---|---|
| *pointer* | The first object according to object index associated with this name. |
| *NULL* | No object exists for this name or invalid parameters. |

### 8.103.5.24 _Objects_Get_class()

```
static __inline__ uint32_t _Objects_Get_class (
            Objects_Id id )  [static]
```

Returns the class portion of the ID.

**Parameters**

| | |
|---|---|
| *id* | The object Id to be processed. |

**Returns**

The class portion of the ID.

Definition at line 269 of file object.h.

### 8.103.5.25 _Objects_Get_inactive()

```
static __inline__ Objects_Control* _Objects_Get_inactive (
            Objects_Information * information )  [static]
```

Get an inactive object or NULL.

**Return values**

| | |
|---|---|
| *NULL* | No inactive object is available. |
| *object* | An inactive object. |

Definition at line 655 of file objectimpl.h.

### 8.103.5.26 _Objects_Get_index()

```
static __inline__ Objects_Maximum _Objects_Get_index (
            Objects_Id id )  [static]
```

Returns the index portion of the ID.

**Parameters**

| | |
|---|---|
| *id* | is the Id to be processed. |

**Returns**

Returns the index portion of the specified object ID.

Definition at line 298 of file object.h.

### 8.103.5.27 _Objects_Get_information()

```
Objects_Information* _Objects_Get_information (
            Objects_APIs the_api,
            uint16_t the_class )
```

Gets object information.

This function returns the information structure given an API and Class. This can be done independent of the existence of any objects created by the API.

**Parameters**

| | |
|---|---|
| *the_api* | Indicates the API for the information we want |
| *the_class* | Indicates the Class for the information we want |

**Return values**

| | |
|---|---|
| *pointer* | Pointer to the Object Information Table for the class of objects which corresponds to this object ID. |
| *NULL* | An error occured. |

Definition at line 23 of file objectgetinfo.c.

### 8.103.5.28 _Objects_Get_information_id()

```
Objects_Information* _Objects_Get_information_id (
            Objects_Id id )
```

Gets information of an object from an ID.

This function returns the information structure given an *id* of an object.

**Parameters**

| | |
|---|---|
| *id* | The object ID to get the information from. |

**Return values**

| | |
|---|---|
| *pointer* | Pointer to the Object Information Table for the class of objects which corresponds to this object ID. |
| *NULL* | An error occured. |

Definition at line 24 of file objectgetinfoid.c.

### 8.103.5.29 _Objects_Get_maximum_index()

```
static __inline__ Objects_Maximum _Objects_Get_maximum_index (
             const Objects_Information * information )  [static]
```

Returns the maximum index of the specified object class.

**Parameters**

| | |
|---|---|
| *information* | The object information. |

**Returns**

The maximum index of the specified object class.

Definition at line 642 of file objectimpl.h.

### 8.103.5.30 _Objects_Get_minimum_id()

```
static __inline__ Objects_Id _Objects_Get_minimum_id (
             Objects_Id id )  [static]
```

Returns the identifier with the minimum index for the specified identifier.

The specified identifier must have valid API, class and node fields.

**Parameters**

| | |
|---|---|
| *id* | The identifier to be processed. |

**Returns**

The corresponding ID with the minimum index.

Definition at line 628 of file objectimpl.h.

### 8.103.5.31 _Objects_Get_name_as_string()

```
char* _Objects_Get_name_as_string (
            Objects_Id id,
            size_t length,
            char * name )
```

Gets object name in the form of a C string.

This method gets the name of an object and returns its name in the form of a C string. It attempts to be careful about overflowing the user's string and about returning unprintable characters.

**Parameters**

| | | |
|---|---|---|
| | *id* | The object to obtain the name of. |
| | *length* | Indicates the length of the caller's buffer. |
| out | *name* | A string which will be filled in. |

**Return values**

| | |
|---|---|

Definition at line 87 of file objectgetnameasstring.c.

### 8.103.5.32 _Objects_Get_next()

```
Objects_Control* _Objects_Get_next (
            Objects_Id id,
            const Objects_Information * information,
            Objects_Id * next_id_p )
```

Gets the next open object after the specified object identifier.

Locks the object allocator mutex in case a next object exists.

**Parameters**

| | | |
|---|---|---|
| | *id* | The Id of the object whose name we are locating. This is the first parameter since usual callers get the object identifier as the first parameter themself. |
| | *information* | Points to an object class information block. |
| in,out | *next_id_p* | The Id of the next object we will look at. |

**Return values**

| | |
|---|---|
| *pointer* | Pointer to the located object. |
| *NULL* | An error occured. |

#### 8.103.5.33 _Objects_Get_no_protection()

```
Objects_Control* _Objects_Get_no_protection (
            Objects_Id id,
            const Objects_Information * information )
```

Maps object ids to object control blocks.

This function maps object ids to object control blocks. If id corresponds to a local object, then it returns the_↩ object control pointer which maps to id and location is set to OBJECTS_LOCAL. If the object class supports global objects and the object id is global and resides on a remote node, then location is set to OBJECTS_REMOTE, and the_object is undefined. Otherwise, location is set to OBJECTS_ERROR and the_object is undefined.

**Parameters**

| | |
|---|---|
| *id* | The Id of the object whose name we are locating. This is the first parameter since usual callers get the object identifier as the first parameter themself. |
| *information* | Points to an object class information block. |

**Return values**

| | |
|---|---|
| *pointer* | The local object corresponding to the given id. |
| *NULL* | The object to the given id was not found locally. |

Definition at line 23 of file objectgetnoprotection.c.

#### 8.103.5.34 _Objects_Get_node()

```
static __inline__ uint32_t _Objects_Get_node (
            Objects_Id id ) [static]
```

Returns the node portion of the ID.

**Parameters**

| | |
|---|---|
| *id* | The object Id to be processed. |

**Returns**

Returns the node portion of an object ID.

Definition at line 284 of file object.h.

#### 8.103.5.35 _Objects_Has_string_name()

```
static __inline__ bool _Objects_Has_string_name (
             const Objects_Information * information )  [static]
```

Returns if the object has a string name.

**Parameters**

| | |
|---|---|
| *information* | The object information table. |

**Return values**

| | |
|---|---|
| *true* | The object has a string name. |
| *false* | The object does not have a string name. |

Definition at line 411 of file objectimpl.h.

### 8.103.5.36 _Objects_Id_to_name()

```
Objects_Name_or_id_lookup_errors _Objects_Id_to_name (
            Objects_Id id,
            Objects_Name * name )
```

Returns the name associated with object id.

This function implements the common portion of the object Id to name directives. This function returns the name associated with object id.

**Parameters**

| | | |
|---|---|---|
| | *id* | is the Id of the object whose name we are locating. |
| out | *name* | will contain the name of the object, if found. |

**Return values**

| | |
|---|---|
| *OBJECTS_NAME_OR_ID_LOOKUP_SUCCESSFUL* | The operation succeeded. *name* contains the name of the object. |
| *OBJECTS_INVALID_ID* | The id is invalid, the operation failed. |

**Note**

This function currently does not support string names.

### 8.103.5.37 _Objects_Initialize_information()

```
void _Objects_Initialize_information (
            Objects_Information * information )
```

Initializes the specified objects information.

The objects information must be statically pre-initialized with the OBJECTS_INFORMATION_DEFINE() macro before this function is called.

**Parameters**

| | |
|---|---|
| *information* | The object information to be initialized. |

Definition at line 29 of file objectinitializeinformation.c.

### 8.103.5.38 _Objects_Invalidate_Id()

```
static __inline__ void _Objects_Invalidate_Id (
             const Objects_Information * information,
             Objects_Control * the_object )  [static]
```

Invalidates an object Id.

This function sets the pointer to the local_table object referenced by the index to NULL so the object Id is invalid after this call.

**Parameters**

| | | |
|---|---|---|
| in,out | *information* | points to an Object Information Table. |
| | *the_object* | The local object pointer. |

**Note**

This routine is ONLY to be called in places where the index portion of the Id is known to be good. This is OK since it is normally called from object create/init or delete/destroy operations.

Definition at line 725 of file objectimpl.h.

### 8.103.5.39 _Objects_Is_api_valid()

```
static __inline__ bool _Objects_Is_api_valid (
             uint32_t the_api )  [static]
```

Checks if the api is valid.

**Parameters**

| | |
|---|---|
| *the_api* | is the api portion of an object ID. |

**Return values**

| | |
|---|---|
| *true* | The specified api value is valid. |
| *false* | The specified api value is not valid. |

Definition at line 552 of file objectimpl.h.

#### 8.103.5.40 _Objects_Is_auto_extend()

```
static __inline__ Objects_Maximum _Objects_Is_auto_extend (
            const Objects_Information * information )  [static]
```

Checks if the automatic object extension (unlimited objects) is enabled.

**Parameters**

| | |
|---|---|
| *information* | The object information. |

**Return values**

| | |
|---|---|
| *true* | The automatic object extension (unlimited objects) is enabled. |
| *false* | The automatic object extension (unlimited objects) is not enabled. |

Definition at line 671 of file objectimpl.h.

#### 8.103.5.41 _Objects_Is_local_id()

```
static __inline__ bool _Objects_Is_local_id (
            Objects_Id id RTEMS_UNUSED )  [static]
```

Checks if the id is of a local object.

**Parameters**

| | |
|---|---|
| *id* | The object ID. |

**Return values**

| | |
|---|---|
| *true* | The specified object Id is local. |
| *false* | The specified object Id is not local. |

**Note**

On a single processor configuration, this always returns true.

Definition at line 587 of file objectimpl.h.

### 8.103.5.42 _Objects_Is_local_node()

```
static __inline__ bool _Objects_Is_local_node (
             uint32_t node )  [static]
```

Checks if the node is of the local object.

**Parameters**

| | |
|---|---|
| *node* | The node number and corresponds to the node number portion of an object ID. |

**Return values**

| | |
|---|---|
| *true* | The specified node is the local node. |
| *false* | The specified node is not the local node. |

Definition at line 570 of file objectimpl.h.

### 8.103.5.43 _Objects_Name_to_id_u32()

```
Objects_Name_or_id_lookup_errors _Objects_Name_to_id_u32 (
             uint32_t name,
             uint32_t node,
             Objects_Id * id,
             const Objects_Information * information )
```

Searches an object of the specified class with the specified name on the specified set of nodes.

This method converts an object name to an identifier. It performs a look up using the object information block for this object class.

**Parameters**

| | | |
|---|---|---|
| | *name* | is the name of the object to find. |
| | *node* | is the set of nodes to search. |
| out | *id* | is the pointer to an object identifier variable or NULL. The object identifier will be stored in the referenced variable, if the operation was successful. |
| | *information* | is the pointer to an object class information block. |

**Return values**

| | |
|---|---|
| *OBJECTS_NAME_OR_ID_LOOKUP_SUCCESSFUL* | The operations was successful. |
| *OBJECTS_INVALID_ADDRESS* | The id parameter was NULL. |
| *OBJECTS_INVALID_NAME* | No object exists with the specified name on the specified node set. |

Definition at line 23 of file objectnametoid.c.

#### 8.103.5.44 _Objects_Name_to_string()

```
size_t _Objects_Name_to_string (
            Objects_Name name,
            bool is_string,
            char * buffer,
            size_t buffer_size )
```

Converts the specified object name to a text representation.

Non-printable characters according to isprint() are converted to '*'.

**Parameters**

| | | |
|---|---|---|
| | *name* | The object name. |
| | *is_string* | Indicates if the object name is a string or a four character array (32-bit unsigned integer). |
| out | *buffer* | The string buffer for the text representation. |
| | *buffer_size* | The buffer size in characters. |

**Returns**

The length of the text representation. May be greater than or equal to the buffer size if truncation occurred.

Definition at line 36 of file objectgetnameasstring.c.

#### 8.103.5.45 _Objects_Namespace_remove_string()

```
void _Objects_Namespace_remove_string (
            const Objects_Information * information,
            Objects_Control * the_object )
```

Removes object with a string name from its namespace.

**Parameters**

| | | |
|---|---|---|
| | *information* | The corresponding object information table. |
| out | *the_object* | The object the object to operate upon. |

Definition at line 25 of file objectnamespaceremove.c.

#### 8.103.5.46 _Objects_Namespace_remove_u32()

```
static __inline__ void _Objects_Namespace_remove_u32 (
            const Objects_Information * information,
            Objects_Control * the_object )  [static]
```

Removes object with a 32-bit integer name from its namespace.

**Parameters**

| | | |
|---|---|---|
| | *information* | The corresponding object information table. |
| out | *the_object* | The object to operate upon. |

Definition at line 485 of file objectimpl.h.

### 8.103.5.47 _Objects_Open()

```
static __inline__ void _Objects_Open (
             Objects_Information * information,
             Objects_Control * the_object,
             Objects_Name name )  [static]
```

Places the_object control pointer and object name in the Local Pointer and Local Name Tables, respectively.

This method uses Objects_Name for the object name.

**Parameters**

| | | |
|---|---|---|
| in,out | *information* | Points to an Object Information Table. |
| | *the_object* | Pointer to an object. |
| | *name* | The name of the object to make accessible. |

Definition at line 750 of file objectimpl.h.

### 8.103.5.48 _Objects_Open_string()

```
static __inline__ void _Objects_Open_string (
             const Objects_Information * information,
             Objects_Control * the_object,
             const char * name )  [static]
```

Places the_object control pointer and object name in the Local Pointer and Local Name Tables, respectively.

This method uses a String for the object name.

**Parameters**

| | | |
|---|---|---|
| in,out | *information* | Points to an Object Information Table. |
| | *the_object* | Pointer to an object. |
| | *name* | The name of the object to make accessible. |

Definition at line 804 of file objectimpl.h.

#### 8.103.5.49 _Objects_Open_u32()

```
static __inline__ void _Objects_Open_u32 (
             const Objects_Information * information,
             Objects_Control * the_object,
             uint32_t name )  [static]
```

Places the_object control pointer and object name in the Local Pointer and Local Name Tables, respectively.

This method uses uint32_t for the object name.

**Parameters**

| | | |
|---|---|---|
| in,out | *information* | Points to an Object Information Table. |
| | *the_object* | Pointer to an object. |
| | *name* | The name of the object to make accessible. |

Definition at line 778 of file objectimpl.h.

#### 8.103.5.50 _Objects_Set_local_object()

```
static __inline__ void _Objects_Set_local_object (
             const Objects_Information * information,
             uint32_t index,
             Objects_Control * the_object )  [static]
```

Sets the pointer to the local_table object referenced by the index.

**Parameters**

| | | |
|---|---|---|
| in,out | *information* | Points to an Object Information Table. |
| | *index* | The index of the object the caller wants to access. |
| | *the_object* | The local object pointer. |

**Note**

This routine is ONLY to be called in places where the index portion of the Id is known to be good. This is OK since it is normally called from object create/init or delete/destroy operations.

Definition at line 692 of file objectimpl.h.

#### 8.103.5.51 _Objects_Set_name()

```
bool _Objects_Set_name (
             const Objects_Information * information,
             Objects_Control * the_object,
             const char * name )
```

Sets objects name.

This method sets the object name to either a copy of a string or up to the first four characters of the string based upon whether this object class uses strings for names.

**Parameters**

| | | |
|---|---|---|
| | *information* | points to the object information structure |
| out | *the_object* | is the object to operate upon |
| | *name* | is a pointer to the name to use |

**Return values**

| | |
|---|---|
| *true* | The operation succeeded. |
| *false* | The operation failed. |

#### 8.103.5.52 _Objects_Shrink_information()

```
void _Objects_Shrink_information (
             Objects_Information * information )
```

Shrinks an object class information record.

This function shrinks an object class information record. The object's name and object space are released. The local_table etc block does not shrink. The InActive list needs to be scanned to find the objects are remove them.

**Parameters**

| | |
|---|---|
| *information* | Points to an object class information block. |

### 8.103.6 Variable Documentation

#### 8.103.6.1 _Objects_Information_table

```
Objects_Information** const _Objects_Information_table[OBJECTS_APIS_LAST+1]  [extern]
```

The following is the list of information blocks per API for each object class. From the ID, we can go to one of these information blocks, and obtain a pointer to the appropriate object control block.

## 8.104 Object Services

RTEMS provides a collection of services to assist in the management and usage of the objects created and utilized via other managers. These services assist in the manipulation of RTEMS objects independent of the API used to create them.

### Classes

- struct rtems_object_api_class_information

  %

### Macros

- #define rtems_build_id(_api, _class, _node, _index) _Objects_Build_id( _api, _class, _node, _index )

  %
- #define rtems_build_name(_C1, _C2, _C3, _C4) _Objects_Build_name( _C1, _C2, _C3, _C4 )

  %
- #define rtems_object_id_api_maximum() OBJECTS_APIS_LAST

  %
- #define rtems_object_id_api_minimum() OBJECTS_INTERNAL_API

  %
- #define RTEMS_OBJECT_ID_FINAL OBJECTS_ID_FINAL

  %
- #define RTEMS_OBJECT_ID_FINAL_INDEX OBJECTS_ID_FINAL_INDEX

  %
- #define rtems_object_id_get_api(_id) _Objects_Get_API( _id )

  %
- #define rtems_object_id_get_class(_id) _Objects_Get_class( _id )

  %
- #define rtems_object_id_get_index(_id) _Objects_Get_index( _id )

  %
- #define rtems_object_id_get_node(_id) _Objects_Get_node( _id )

  %
- #define RTEMS_OBJECT_ID_INITIAL(_api, _class, _node) OBJECTS_ID_INITIAL( _api, _class, _node )

  %
- #define RTEMS_OBJECT_ID_INITIAL_INDEX OBJECTS_ID_INITIAL_INDEX

  %
- #define RTEMS_SEARCH_ALL_NODES OBJECTS_SEARCH_ALL_NODES

  %
- #define RTEMS_SEARCH_LOCAL_NODE OBJECTS_SEARCH_LOCAL_NODE

  %
- #define RTEMS_SEARCH_OTHER_NODES OBJECTS_SEARCH_OTHER_NODES

  %
- #define RTEMS_WHO_AM_I OBJECTS_WHO_AM_I

  %

## Functions

- int rtems_object_api_maximum_class (int api)
  %
- int rtems_object_api_minimum_class (int api)
  %
- const char * rtems_object_get_api_class_name (int the_api, int the_class)
  %
- const char * rtems_object_get_api_name (int api)
  %
- rtems_status_code rtems_object_get_class_information (int the_api, int the_class, rtems_object_api_class_information *info)
  %
- rtems_status_code rtems_object_get_classic_name (rtems_id id, rtems_name *name)
  %
- static uint16_t rtems_object_get_local_node (void)
  %
- char * rtems_object_get_name (rtems_id id, size_t length, char *name)
  %
- int rtems_object_id_api_maximum_class (int api)
  %
- rtems_status_code rtems_object_set_name (rtems_id id, const char *name)
  %

### 8.104.1 Detailed Description

RTEMS provides a collection of services to assist in the management and usage of the objects created and utilized via other managers. These services assist in the manipulation of RTEMS objects independent of the API used to create them.

### 8.104.2 Macro Definition Documentation

#### 8.104.2.1 rtems_build_id

```
#define rtems_build_id(
            _api,
            _class,
            _node,
            _index )  _Objects_Build_id( _api, _class, _node, _index )
```

%

**Parameters**

| | |
|---|---|
| *_api* | % |
| *_class* | % |
| *_node* | % |
| *_index* | % |

Definition at line 159 of file object.h.

#### 8.104.2.2 rtems_build_name

```
#define rtems_build_name(
              _C1,
              _C2,
              _C3,
              _C4 )  _Objects_Build_name( _C1, _C2, _C3, _C4 )
```

%

**Parameters**

| | |
|---|---|
| *_C1* | % |
| *_C2* | % |
| *_C3* | % |
| *_C4* | % |

Definition at line 177 of file object.h.

#### 8.104.2.3 rtems_object_id_get_api

```
#define rtems_object_id_get_api(
              _id ) _Objects_Get_API( _id )
```

%

**Parameters**

| | |
|---|---|
| *↩ _↩ id* | % |

Definition at line 322 of file object.h.

#### 8.104.2.4 rtems_object_id_get_class

```
#define rtems_object_id_get_class(
              _id ) _Objects_Get_class( _id )
```

%

**Parameters**

| ↩ _↩ *id* | % |
|---|---|

Definition at line 333 of file object.h.

### 8.104.2.5 rtems_object_id_get_index

```
#define rtems_object_id_get_index(
              _id ) _Objects_Get_index( _id )
```

%

**Parameters**

| ↩ _↩ *id* | % |
|---|---|

Definition at line 344 of file object.h.

### 8.104.2.6 rtems_object_id_get_node

```
#define rtems_object_id_get_node(
              _id ) _Objects_Get_node( _id )
```

%

**Parameters**

| ↩ _↩ *id* | % |
|---|---|

Definition at line 355 of file object.h.

### 8.104.2.7 RTEMS_OBJECT_ID_INITIAL

```
#define RTEMS_OBJECT_ID_INITIAL(
              _api,
```

```
            _class,
            _node )  OBJECTS_ID_INITIAL( _api, _class, _node )
```

%

**Parameters**

| | |
|---|---|
| *_api* | % |
| *_class* | % |
| *_node* | % |

Definition at line 370 of file object.h.

## 8.104.3 Function Documentation

### 8.104.3.1 rtems_object_api_maximum_class()

```
int rtems_object_api_maximum_class (
            int api )
```

%

**Parameters**

| | |
|---|---|
| *api* | % |

### 8.104.3.2 rtems_object_api_minimum_class()

```
int rtems_object_api_minimum_class (
            int api )
```

%

**Parameters**

| | |
|---|---|
| *api* | % |

### 8.104.3.3 rtems_object_get_api_class_name()

```
const char* rtems_object_get_api_class_name (
            int the_api,
            int the_class )
```

%

**Parameters**

| | |
|---|---|
| *the_api* | % |
| *the_class* | % |

#### 8.104.3.4 rtems_object_get_api_name()

```
const char* rtems_object_get_api_name (
            int api )
```

%

**Parameters**

| | |
|---|---|
| *api* | % |

#### 8.104.3.5 rtems_object_get_class_information()

```
rtems_status_code rtems_object_get_class_information (
            int the_api,
            int the_class,
            rtems_object_api_class_information * info )
```

%

**Parameters**

| | |
|---|---|
| *the_api* | % |
| *the_class* | % |
| *info* | % |

#### 8.104.3.6 rtems_object_get_classic_name()

```
rtems_status_code rtems_object_get_classic_name (
            rtems_id id,
            rtems_name * name )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *name* | % |

### 8.104.3.7 rtems_object_get_name()

```
char* rtems_object_get_name (
            rtems_id id,
            size_t length,
            char * name )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *length* | % |
| *name* | % |

### 8.104.3.8 rtems_object_id_api_maximum_class()

```
int rtems_object_id_api_maximum_class (
            int api )
```

%

**Parameters**

| | |
|---|---|
| *api* | % |

### 8.104.3.9 rtems_object_set_name()

```
rtems_status_code rtems_object_set_name (
            rtems_id id,
            const char * name )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *name* | % |

## 8.105 POSIX API Configuration

### Macros

- #define CONFIGURE_MAXIMUM_POSIX_KEYS

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_POSIX_KEY_VALUE_PAIRS

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_POSIX_MESSAGE_QUEUES

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_POSIX_QUEUED_SIGNALS

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_POSIX_SEMAPHORES

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_POSIX_SHMS

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_POSIX_THREADS

  *This configuration option is an integer define.*
- #define CONFIGURE_MAXIMUM_POSIX_TIMERS

  *This configuration option is an integer define.*
- #define CONFIGURE_MINIMUM_POSIX_THREAD_STACK_SIZE

  *This configuration option is an integer define.*

### 8.105.1 Detailed Description

This section describes configuration options related to the POSIX API. Most POSIX API objects are available by default since RTEMS 5.1. The queued signals and timers are only available if RTEMS was built with the `--enable-posix` build configuration option.

### 8.105.2 Macro Definition Documentation

#### 8.105.2.1 CONFIGURE_MAXIMUM_POSIX_KEY_VALUE_PAIRS

```
#define CONFIGURE_MAXIMUM_POSIX_KEY_VALUE_PAIRS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of key value pairs used by POSIX API Keys that can be concurrently active.

**Default Value**

The default value is CONFIGURE_MAXIMUM_POSIX_KEYS * CONFIGURE_MAXIMUM_TASKS + CONFIGURE_MAXIMUM_POSIX_THREADS.

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to 65535.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It may be defined through rtems_resource_unlimited() the enable unlimited objects for this object class, if the value passed to rtems_resource_unlimited() satisfies all other constraints of this configuration option.

#### Notes

This object class can be configured in unlimited allocation mode, see `Unlimited Objects`.

A key value pair is created by pthread_setspecific() if the value is not `NULL`, otherwise it is deleted.

Definition at line 3526 of file appl-config.h.

### 8.105.2.2 CONFIGURE_MAXIMUM_POSIX_KEYS

```
#define CONFIGURE_MAXIMUM_POSIX_KEYS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of POSIX API Keys that can be concurrently active.

#### Default Value

The default value is 0.

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to 65535.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It may be defined through rtems_resource_unlimited() the enable unlimited objects for this object class, if the value passed to rtems_resource_unlimited() satisfies all other constraints of this configuration option.

#### Notes

This object class can be configured in unlimited allocation mode, see `Unlimited Objects`.

Definition at line 3482 of file appl-config.h.

### 8.105.2.3 CONFIGURE_MAXIMUM_POSIX_MESSAGE_QUEUES

```
#define CONFIGURE_MAXIMUM_POSIX_MESSAGE_QUEUES
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of POSIX API Message Queues that can be concurrently active.

#### Default Value

The default value is 0.

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to 65535.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It shall be small enough so that the RTEMS Workspace size calculation carried out by <rtems/confdefs.h> does not overflow an integer of type uintptr_t.
- It may be defined through rtems_resource_unlimited() the enable unlimited objects for this object class, if the value passed to rtems_resource_unlimited() satisfies all other constraints of this configuration option.

#### Notes

This object class can be configured in unlimited allocation mode, see Unlimited Objects. You have to account for the memory used to store the messages of each message queue, see CONFIGURE_MESSAGE_BUFFER_MEMORY.

Definition at line 3568 of file appl-config.h.

### 8.105.2.4 CONFIGURE_MAXIMUM_POSIX_QUEUED_SIGNALS

```
#define CONFIGURE_MAXIMUM_POSIX_QUEUED_SIGNALS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of POSIX API Queued Signals that can be concurrently active.

#### Default Value

The default value is 0.

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It shall be small enough so that the RTEMS Workspace size calculation carried out by <rtems/confdefs.h> does not overflow an integer of type uintptr_t.
- It shall be zero if the POSIX API is not enabled (e.g. RTEMS was built without the --enable-posix build configuration option). Otherwise a compile time error in the configuration file will occur.

#### Notes

Unlimited objects are not available for queued signals.

Queued signals are only available if RTEMS was built with the --enable-posix build configuration option.

Definition at line 3609 of file appl-config.h.

### 8.105.2.5 CONFIGURE_MAXIMUM_POSIX_SEMAPHORES

```
#define CONFIGURE_MAXIMUM_POSIX_SEMAPHORES
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of POSIX API Named Semaphores that can be concurrently active.

#### Default Value

The default value is 0.

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to 65535.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It shall be small enough so that the RTEMS Workspace size calculation carried out by <rtems/confdefs.h> does not overflow an integer of type uintptr_t.
- It may be defined through rtems_resource_unlimited() the enable unlimited objects for this object class, if the value passed to rtems_resource_unlimited() satisfies all other constraints of this configuration option.

#### Notes

This object class can be configured in unlimited allocation mode, see Unlimited Objects.

Named semaphores are created with sem_open(). Semaphores initialized with sem_init() are not affected by this configuration option since the storage space for these semaphores is user-provided.

Definition at line 3656 of file appl-config.h.

#### 8.105.2.6 CONFIGURE_MAXIMUM_POSIX_SHMS

```
#define CONFIGURE_MAXIMUM_POSIX_SHMS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of POSIX API Shared Memory objects that can be concurrently active.

##### Default Value

The default value is 0.

##### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to 65535.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It shall be small enough so that the RTEMS Workspace size calculation carried out by <rtems/confdefs.h> does not overflow an integer of type uintptr_t.
- It may be defined through rtems_resource_unlimited() the enable unlimited objects for this object class, if the value passed to rtems_resource_unlimited() satisfies all other constraints of this configuration option.

##### Notes

This object class can be configured in unlimited allocation mode, see Unlimited Objects.

Definition at line 3697 of file appl-config.h.

#### 8.105.2.7 CONFIGURE_MAXIMUM_POSIX_THREADS

```
#define CONFIGURE_MAXIMUM_POSIX_THREADS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of POSIX API Threads that can be concurrently active.

##### Default Value

The default value is 0.

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to 65535.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It shall be small enough so that the task stack space calculation carried out by <rtems/confdefs.h> does not overflow an integer of type uintptr_t.

#### Notes

This object class can be configured in unlimited allocation mode, see Unlimited Objects.

This calculations for the required memory in the RTEMS Workspace for threads assume that each thread has a minimum stack size and has floating point support enabled. The configuration option CONFIGURE_EXTRA_TASK_STACKS is used to specify thread stack requirements **above** the minimum size required.

The maximum number of Classic API Tasks is specified by CONFIGURE_MAXIMUM_TASKS.

All POSIX threads have floating point enabled.

Definition at line 3746 of file appl-config.h.

### 8.105.2.8 CONFIGURE_MAXIMUM_POSIX_TIMERS

```
#define CONFIGURE_MAXIMUM_POSIX_TIMERS
```

This configuration option is an integer define.

The value of this configuration option defines the maximum number of POSIX API Timers that can be concurrently active.

#### Default Value

The default value is 0.

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to 0.
- It shall be less than or equal to 65535.
- It shall be less than or equal to a BSP-specific and application-specific value which depends on the size of the memory available to the application.
- It may be defined through rtems_resource_unlimited() the enable unlimited objects for this object class, if the value passed to rtems_resource_unlimited() satisfies all other constraints of this configuration option.
- It shall be zero if the POSIX API is not enabled (e.g. RTEMS was built without the --enable-posix build configuration option). Otherwise a compile time error in the configuration file will occur.

#### Notes

This object class can be configured in unlimited allocation mode, see Unlimited Objects.

Timers are only available if RTEMS was built with the --enable-posix build configuration option.

Definition at line 3792 of file appl-config.h.

### 8.105.2.9 CONFIGURE_MINIMUM_POSIX_THREAD_STACK_SIZE

```
#define CONFIGURE_MINIMUM_POSIX_THREAD_STACK_SIZE
```

This configuration option is an integer define.

The value of this configuration option defines the minimum stack size in bytes for every POSIX thread in the system.

#### Default Value

The default value is two times the value of CONFIGURE_MINIMUM_TASK_STACK_SIZE.

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be small enough so that the task stack space calculation carried out by <rtems/confdefs.h> does not overflow an integer of type uintptr_t.
- It shall be greater than or equal to a BSP-specific and application-specific minimum value.

Definition at line 3819 of file appl-config.h.

# 8.106 POSIX Initialization Thread Configuration

## Macros

- #define CONFIGURE_POSIX_INIT_THREAD_ENTRY_POINT

  *This configuration option is an initializer define.*
- #define CONFIGURE_POSIX_INIT_THREAD_STACK_SIZE

  *This configuration option is an integer define.*
- #define CONFIGURE_POSIX_INIT_THREAD_TABLE

  *This configuration option is a boolean feature define.*

## 8.106.1 Detailed Description

This section describes configuration options related to the POSIX initialization thread.

## 8.106.2 Macro Definition Documentation

### 8.106.2.1 CONFIGURE_POSIX_INIT_THREAD_ENTRY_POINT

```
#define CONFIGURE_POSIX_INIT_THREAD_ENTRY_POINT
```

This configuration option is an initializer define.

The value of this configuration option initializes the entry point of the POSIX API initialization thread.

#### Default Value

The default value is `POSIX_Init`.

#### Value Constraints

The value of this configuration option shall be defined to a valid function pointer of the type `void *( *entry_point )( void * )`.

#### Notes

The application shall provide the function referenced by this configuration option.

Definition at line 3856 of file appl-config.h.

### 8.106.2.2 CONFIGURE_POSIX_INIT_THREAD_STACK_SIZE

```
#define CONFIGURE_POSIX_INIT_THREAD_STACK_SIZE
```

This configuration option is an integer define.

The value of this configuration option defines the thread stack size of the POSIX API initialization thread.

#### Default Value

The default value is CONFIGURE_MINIMUM_POSIX_THREAD_STACK_SIZE.

#### Value Constraints

The value of this configuration option shall satisfy all of the following constraints:

- It shall be greater than or equal to CONFIGURE_MINIMUM_TASK_STACK_SIZE.
- It shall be small enough so that the task stack space calculation carried out by <rtems/confdefs.h> does not overflow an integer of type uintptr_t.

Definition at line 3881 of file appl-config.h.

### 8.106.2.3 CONFIGURE_POSIX_INIT_THREAD_TABLE

```
#define CONFIGURE_POSIX_INIT_THREAD_TABLE
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then exactly one POSIX initialization thread is configured.

#### Default Configuration

If this configuration option is undefined, then the described feature is not enabled.

#### Notes

The application shall define exactly one of the following configuration options

- CONFIGURE_RTEMS_INIT_TASKS_TABLE,
- CONFIGURE_POSIX_INIT_THREAD_TABLE, or
- CONFIGURE_IDLE_TASK_INITIALIZES_APPLICATION

otherwise a compile time error in the configuration file will occur.

Definition at line 3909 of file appl-config.h.

## 8.107 POSIX Status and Error Number Checks

Checks for POSIX status and error numbers.

### Macros

- #define **T_eno**(a, e) T_flags_eno(a, e, 0)
- #define **T_assert_eno**(a, e) T_flags_eno(a, e, T_CHECK_STOP)
- #define **T_quiet_eno**(a, e) T_flags_eno(a, e, T_CHECK_QUIET)
- #define **T_step_eno**(s, a, e) T_flags_eno(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eno**(s, a, e) T_flags_eno(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_eno_success**(a) T_flags_eno_success(a, 0)
- #define **T_assert_eno_success**(a) T_flags_eno_success(a, T_CHECK_STOP)
- #define **T_quiet_eno_success**(a) T_flags_eno_success(a, T_CHECK_QUIET)
- #define **T_step_eno_success**(s, a) T_flags_eno_success(a, T_CHECK_STEP(s))
- #define **T_step_assert_eno_success**(s, a) T_flags_eno_success(a, T_CHECK_STEP(s) | T_CHECK_↩ STOP)
- #define **T_psx_error**(a, eno) T_flags_psx_error(a, eno, 0)
- #define **T_assert_psx_error**(a, eno) T_flags_psx_error(a, eno, T_CHECK_STOP)
- #define **T_quiet_psx_error**(a, eno) T_flags_psx_error(a, eno, T_CHECK_QUIET)
- #define **T_step_psx_error**(s, a, eno) T_flags_psx_error(a, eno, T_CHECK_STEP(s))
- #define **T_step_assert_psx_error**(s, a, eno) T_flags_psx_error(a, eno, T_CHECK_STEP(s) | T_CHEC↩ K_STOP)
- #define **T_psx_success**(a) T_flags_psx_success(a, 0)
- #define **T_assert_psx_success**(a) T_flags_psx_success(a, T_CHECK_STOP)
- #define **T_quiet_psx_success**(a) T_flags_psx_success(a, T_CHECK_QUIET)
- #define **T_step_psx_success**(s, a) T_flags_psx_success(a, T_CHECK_STEP(s))
- #define **T_step_assert_psx_success**(s, a) T_flags_psx_success(a, T_CHECK_STEP(s) | T_CHECK_S↩ TOP)

### Functions

- const char * **T_strerror** (int)

### 8.107.1 Detailed Description

Checks for POSIX status and error numbers.

## 8.108 Partition Manager

The Partition Manager provides facilities to dynamically allocate memory in fixed-size units.

### Macros

- #define RTEMS_PARTITION_ALIGNMENT CPU_SIZEOF_POINTER

  *This constant defines the minimum alignment of a partition buffer in bytes.*

### Functions

- rtems_status_code rtems_partition_create (rtems_name name, void *starting_address, uintptr_t length, size_t buffer_size, rtems_attribute attribute_set, rtems_id *id)

  *Creates a partition.*
- rtems_status_code rtems_partition_ident (rtems_name name, uint32_t node, rtems_id *id)

  *Identifies a partition by the object name.*
- rtems_status_code rtems_partition_delete (rtems_id id)

  *Deletes the partition.*
- rtems_status_code rtems_partition_get_buffer (rtems_id id, void **buffer)

  *Tries to get a buffer from the partition.*
- rtems_status_code rtems_partition_return_buffer (rtems_id id, void *buffer)

  *Returns the buffer to the partition.*

### 8.108.1 Detailed Description

The Partition Manager provides facilities to dynamically allocate memory in fixed-size units.

### 8.108.2 Macro Definition Documentation

#### 8.108.2.1 RTEMS_PARTITION_ALIGNMENT

```
#define RTEMS_PARTITION_ALIGNMENT CPU_SIZEOF_POINTER
```

This constant defines the minimum alignment of a partition buffer in bytes.

Use it with RTEMS_ALIGNED() to define the alignment of partition buffer types or statically allocated partition buffer areas.

Definition at line 89 of file part.h.

### 8.108.3 Function Documentation

### 8.108.3.1 rtems_partition_create()

```
rtems_status_code rtems_partition_create (
              rtems_name name,
              void * starting_address,
              uintptr_t length,
              size_t buffer_size,
              rtems_attribute attribute_set,
              rtems_id * id )
```

Creates a partition.

This directive creates a partition of fixed size buffers from a physically contiguous memory space which starts at `starting_address` and is `length` bytes in size. Each allocated buffer is to be of `buffer_size` in bytes. The assigned partition identifier is returned in `id`. This partition identifier is used to access the partition with other partition related directives.

The **attribute set** specified in `attribute_set` is built through a *bitwise or* of the attribute constants described below. Attributes not mentioned below are not evaluated by this directive and have no effect.

The partition can have **local** or **global** scope in a multiprocessing network (this attribute does not refer to SMP systems).

- A **local** scope is the default and can be emphasized through the use of the RTEMS_LOCAL attribute. A local partition can be only used by the node which created it.
- A **global** scope is established if the RTEMS_GLOBAL attribute is set. The memory space used for the partition must reside in shared memory. Setting the global attribute in a single node system has no effect.

This directive may cause the calling task to be preempted due to an obtain and release of the object allocator mutex.

The partition buffer area specified by the `starting_address` must be properly aligned. It must be possible to directly store target architecture pointers and also the user data. For example, if the user data contains some long double or vector data types, the partition buffer area and the buffer size must take the alignment of these types into account which is usually larger than the pointer alignment. A cache line alignment may be also a factor. Use RTEMS_PARTITION_ALIGNMENT to specify the minimum alignment of a partition buffer type.

The `buffer_size` parameter must be an integral multiple of the pointer size on the target architecture. Additionally, `buffer_size` must be large enough to hold two pointers on the target architecture. This is required for RTEMS to manage the buffers when they are free.

For control and maintenance of the partition, RTEMS allocates a PTCB from the local PTCB free pool and initializes it. Memory from the partition buffer area is not used by RTEMS to store the PTCB.

The PTCB for a global partition is allocated on the local node. Partitions should not be made global unless remote tasks must interact with the partition. This is to avoid the overhead incurred by the creation of a global partition. When a global partition is created, the partition's name and identifier must be transmitted to every node in the system for insertion in the local copy of the global object table.

The total number of global objects, including partitions, is limited by the value of the CONFIGURE_MP_MAXIMUM_GLOBAL_OBJECT application configuration option.

**Parameters**

| | |
|---|---|
| *name* | is the name of the partition. |
| *starting_address* | is the starting address of the buffer area used by the partition. |
| *length* | is the length in bytes of the buffer area used by the partition. |
| *buffer_size* | is the size in bytes of a buffer managed by the partition. |
| *attribute_set* | is the attribute set of the partition. |
| *id* | is the pointer to an object identifier variable. The identifier of the created partition object will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_NAME* | The partition name was invalid. |
| *RTEMS_INVALID_ADDRESS* | The `id` parameter was NULL. |
| *RTEMS_INVALID_SIZE* | The `length` parameter was 0. |
| *RTEMS_INVALID_SIZE* | The `buffer_size` parameter was 0. |
| *RTEMS_INVALID_SIZE* | The `length` parameter was less than the `buffer_size` parameter. |
| *RTEMS_INVALID_SIZE* | The `buffer_size` parameter was not an integral multiple of the pointer size. |
| *RTEMS_INVALID_SIZE* | The `buffer_size` parameter was less than two times the pointer size. |
| *RTEMS_INVALID_ADDRESS* | The `starting_address` parameter was not on a pointer size boundary. |
| *RTEMS_TOO_MANY* | There was no inactive object available to create a new partition. The number of partitions available to the application is configured through the CONFIGURE_MAXIMUM_PARTITIONS configuration option. |

Definition at line 69 of file partcreate.c.

### 8.108.3.2 rtems_partition_delete()

```
rtems_status_code rtems_partition_delete (
              rtems_id id )
```

Deletes the partition.

This directive deletes the partition specified by the id parameter. The partition cannot be deleted if any of its buffers are still allocated. The PTCB for the deleted partition is reclaimed by RTEMS.

This directive may cause the calling task to be preempted due to an obtain and release of the object allocator mutex.

The calling task does not have to be the task that created the partition. Any local task that knows the partition identifier can delete the partition.

When a global partition is deleted, the partition identifier must be transmitted to every node in the system for deletion from the local copy of the global object table.

The partition must reside on the local node, even if the partition was created with the RTEMS_GLOBAL attribute.

**Parameters**

| | |
|---|---|
| *id* | is the partition identifier. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ID* | There was no partition with the specified identifier. |
| *RTEMS_ILLEGAL_ON_REMOTE_OBJECT* | The partition resided on a remote node. |
| *RTEMS_RESOURCE_IN_USE* | There were buffers of the partition still in use. |

Definition at line 26 of file partdelete.c.

### 8.108.3.3 rtems_partition_get_buffer()

```
rtems_status_code rtems_partition_get_buffer (
            rtems_id id,
            void ** buffer )
```

Tries to get a buffer from the partition.

This directive allows a buffer to be obtained from the partition specified in the `id` parameter. The address of the allocated buffer is returned through the `buffer` parameter.

This directive will not cause the running task to be preempted.

The buffer start alignment is determined by the memory area and buffer size used to create the partition.

A task cannot wait on a buffer to become available.

Getting a buffer from a global partition which does not reside on the local node will generate a request telling the remote node to allocate a buffer from the partition.

**Parameters**

| | | |
|---|---|---|
| | *id* | is the partition identifier. |
| `out` | *buffer* | is the pointer to a buffer pointer variable. The pointer to the allocated buffer will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ID* | There was no partition with the specified identifier. |
| *RTEMS_INVALID_ADDRESS* | The `buffer` parameter was NULL. |
| *RTEMS_UNSATISFIED* | There was no free buffer available to allocate and return. |

Definition at line 31 of file partgetbuffer.c.

### 8.108.3.4 rtems_partition_ident()

```
rtems_status_code rtems_partition_ident (
            rtems_name name,
            uint32_t node,
            rtems_id * id )
```

Identifies a partition by the object name.

This directive obtains a partition identifier associated with the partition name specified in `name`.

The node to search is specified in `node`. It shall be

- a valid node number,
- the constant RTEMS_SEARCH_ALL_NODES to search in all nodes,
- the constant RTEMS_SEARCH_LOCAL_NODE to search in the local node only, or
- the constant RTEMS_SEARCH_OTHER_NODES to search in all nodes except the local node.

If the partition name is not unique, then the partition identifier will match the first partition with that name in the search order. However, this partition identifier is not guaranteed to correspond to the desired partition. The partition identifier is used with other partition related directives to access the partition.

If node is RTEMS_SEARCH_ALL_NODES, all nodes are searched with the local node being searched first. All other nodes are searched with the lowest numbered node searched first.

If node is a valid node number which does not represent the local node, then only the partitions exported by the designated node are searched.

This directive does not generate activity on remote nodes. It accesses only the local copy of the global object table.

**Parameters**

| | | |
|---|---|---|
| | *name* | is the object name to look up. |
| | *node* | is the node or node set to search for a matching object. |
| `out` | *id* | is the pointer to an object identifier variable. The object identifier of an object with the specified name will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The id parameter was NULL. |
| *RTEMS_INVALID_NAME* | The name parameter was 0. |
| *RTEMS_INVALID_NAME* | There was no object with the specified name on the specified nodes. |
| *RTEMS_INVALID_NODE* | In multiprocessing configurations, the specified node was invalid. |

Definition at line 45 of file partident.c.

#### 8.108.3.5 rtems_partition_return_buffer()

```
rtems_status_code rtems_partition_return_buffer (
            rtems_id id,
            void * buffer )
```

Returns the buffer to the partition.

This directive returns the buffer specified by `buffer` to the partition specified by `id`.

This directive will not cause the running task to be preempted.

Returning a buffer to a global partition which does not reside on the local node will generate a request telling the remote node to return the buffer to the partition.

Returning a buffer multiple times is an error. It will corrupt the internal state of the partition.

**Parameters**

| | |
|---|---|
| *id* | is the partition identifier. |
| *buffer* | is the pointer to the buffer to return. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ID* | There was no partition with the specified identifier. |
| *RTEMS_INVALID_ADDRESS* | The buffer referenced by `buffer` was not in the partition. |

Definition at line 65 of file partreturnbuffer.c.

## 8.109 Partition Manager Implementation

### Files

- file part.h

  *This header file provides the Partition Manager API.*
- file partdata.h

  *This header file defines the API used by the Application Configuration to define the Partition Manager information.*
- file partimpl.h

  *This header file defines interfaces used by the Partition Manager implementation.*

### Classes

- struct Partition_Control

  *This structure is the Partition Control Block (PTCB).*

### Macros

- #define PARTITION_INFORMATION_DEFINE(_max)

  *Defines the Partition Manager objects information.*

### Functions

- static __inline__ Partition_Control * _Partition_Get (Objects_Id id, ISR_lock_Context *lock_context)

  *Calls _Objects_Get() using the Partition Manager information.*
- static __inline__ void _Partition_Acquire_critical (Partition_Control *the_partition, ISR_lock_Context *lock↩_context)

  *Acquires the partition lock in an ISR disabled section.*
- static __inline__ void _Partition_Release (Partition_Control *the_partition, ISR_lock_Context *lock_context)

  *Releases the partition lock and restores the ISR level.*

### Variables

- Objects_Information _Partition_Information

  *The Partition Manager objects information is used to manage the objects of this class.*

### 8.109.1 Detailed Description

### 8.109.2 Macro Definition Documentation

#### 8.109.2.1 PARTITION_INFORMATION_DEFINE

```
#define PARTITION_INFORMATION_DEFINE(
              _max )
```

**Value:**

```
  OBJECTS_INFORMATION_DEFINE( \
    _Partition, \
    OBJECTS_CLASSIC_API, \
    OBJECTS_RTEMS_PARTITIONS, \
    Partition_Control, \
    _max, \
    OBJECTS_NO_STRING_NAME, \
    _Partition_MP_Send_extract_proxy \
  )
```

Defines the Partition Manager objects information.

This macro should only be used by <rtems/confdefs/objectsclassic.h>.

**Parameters**

| | |
|---|---|
| *_max* | is the configured object maximum (the OBJECTS_UNLIMITED_OBJECTS flag may be set). |

Definition at line 117 of file partdata.h.

### 8.109.3 Function Documentation

#### 8.109.3.1 _Partition_Acquire_critical()

```
static __inline__ void _Partition_Acquire_critical (
            Partition_Control * the_partition,
            ISR_lock_Context * lock_context )  [static]
```

Acquires the partition lock in an ISR disabled section.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_partition* | is the partition control block. |
| in,out | *lock_context* | is the lock context set up by _Partition_Get(). |

Definition at line 63 of file partimpl.h.

#### 8.109.3.2 _Partition_Get()

```
static __inline__ Partition_Control* _Partition_Get (
            Objects_Id id,
            ISR_lock_Context * lock_context )  [static]
```

Calls _Objects_Get() using the Partition Manager information.

**Parameters**

| | | |
|---|---|---|
| | *id* | is the object identifier. |
| out | *lock_context* | is the lock context. |

**Returns**

See _Objects_Get().

Definition at line 44 of file partimpl.h.

#### 8.109.3.3 _Partition_Release()

```
static __inline__ void _Partition_Release (
            Partition_Control * the_partition,
            ISR_lock_Context * lock_context )  [static]
```

Releases the partition lock and restores the ISR level.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_partition* | is the partition control block. |
| in,out | *lock_context* | is the lock context set up by _Partition_Get(). |

Definition at line 78 of file partimpl.h.

### 8.109.4 Variable Documentation

#### 8.109.4.1 _Partition_Information

```
Objects_Information _Partition_Information  [extern]
```

The Partition Manager objects information is used to manage the objects of this class.

If CONFIGURE_MAXIMUM_PARTITIONS is greater than zero, then the object information is defined by PARTITION_INFORMATION_DEFINE(), otherwise it is defined by OBJECTS_INFORMATION_DEFINE_ZERO().

Definition at line 48 of file part.c.

## 8.110 Pointer Checks

Checks for pointers.

### Macros

- #define **T_eq_ptr**(a, e) T_flags_eq_ptr(a, e, 0, #a, #e)
- #define **T_assert_eq_ptr**(a, e) T_flags_eq_ptr(a, e, T_CHECK_STOP, #a, #e)
- #define **T_quiet_eq_ptr**(a, e) T_flags_eq_ptr(a, e, T_CHECK_QUIET, #a, #e)
- #define **T_step_eq_ptr**(s, a, e) T_flags_eq_ptr(a, e, T_CHECK_STEP(s), #a, #e)
- #define **T_step_assert_eq_ptr**(s, a, e) T_flags_eq_ptr(a, e, T_CHECK_STEP(s) | T_CHECK_STOP, #a, #e)
- #define **T_ne_ptr**(a, e) T_flags_ne_ptr(a, e, 0, #a, #e)
- #define **T_assert_ne_ptr**(a, e) T_flags_ne_ptr(a, e, T_CHECK_STOP, #a, #e)
- #define **T_quiet_ne_ptr**(a, e) T_flags_ne_ptr(a, e, T_CHECK_QUIET, #a, #e)
- #define **T_step_ne_ptr**(s, a, e) T_flags_ne_ptr(a, e, T_CHECK_STEP(s), #a, #e)
- #define **T_step_assert_ne_ptr**(s, a, e) T_flags_ne_ptr(a, e, T_CHECK_STEP(s) | T_CHECK_STOP, #a, #e)
- #define **T_null**(a) T_flags_null(a, 0, #a)
- #define **T_assert_null**(a) T_flags_null(a, T_CHECK_STOP, #a)
- #define **T_quiet_null**(a) T_flags_null(a, T_CHECK_QUIET, #a)
- #define **T_quiet_assert_null**(a) T_flags_null(a, T_CHECK_QUIET | T_CHECK_STOP, #a)
- #define **T_step_null**(s, a) T_flags_null(a, T_CHECK_STEP(s), #a)
- #define **T_step_assert_null**(s, a) T_flags_null(a, T_CHECK_STEP(s) | T_CHECK_STOP, #a)
- #define **T_not_null**(a) T_flags_not_null(a, 0, #a)
- #define **T_assert_not_null**(a) T_flags_not_null(a, T_CHECK_STOP, #a)
- #define **T_quiet_not_null**(a) T_flags_not_null(a, T_CHECK_QUIET, #a)
- #define **T_quiet_assert_not_null**(a) T_flags_not_null(a, T_CHECK_QUIET | T_CHECK_STOP, #a)
- #define **T_step_not_null**(s, a) T_flags_not_null(a, T_CHECK_STEP(s), #a)
- #define **T_step_assert_not_null**(s, a) T_flags_not_null(a, T_CHECK_STEP(s) | T_CHECK_STOP, #a)

### 8.110.1 Detailed Description

Checks for pointers.

## 8.111 Print Support

Print Support.

### Modules

- Kernel Print Support
- RTEMS Print Support

### 8.111.1 Detailed Description

Print Support.

# 8.112 Priority Handler

Priority Handler.

## Files

- file priority.h

  *Priority Handler API.*
- file prioritybitmapimpl.h

  *Inlined Routines in the Priority Handler Bit Map Implementation.*
- file priorityimpl.h

  *Priority Handler API Implementation.*

## Classes

- struct Priority_Node

  *The priority node to build up a priority aggregation.*
- struct Priority_Aggregation

  *The priority aggregation.*
- struct Priority_Actions

  *A list of priority actions.*

## Macros

- #define PRIORITY_MINIMUM 0

  *The highest (most important) thread priority value.*
- #define PRIORITY_PSEUDO_ISR PRIORITY_MINIMUM

  *The priority value of pseudo-ISR threads.*
- #define PRIORITY_DEFAULT_MAXIMUM 255

  *The default lowest (least important) thread priority value.*

## Typedefs

- typedef uint64_t Priority_Control

  *The thread priority control.*
- typedef struct Priority_Aggregation **Priority_Aggregation**
- typedef void(∗ **Priority_Add_handler**) (Priority_Aggregation ∗aggregation, Priority_Actions ∗actions, void ∗arg)
- typedef void(∗ **Priority_Change_handler**) (Priority_Aggregation ∗aggregation, bool prepend_↩ it, Priority_Actions ∗actions, void ∗arg)
- typedef void(∗ **Priority_Remove_handler**) (Priority_Aggregation ∗aggregation, Priority_Actions ∗actions, void ∗arg)

## Enumerations

- enum Priority_Action_type { **PRIORITY_ACTION_ADD**, **PRIORITY_ACTION_CHANGE**, **PRIORITY_AC↩ TION_REMOVE**, **PRIORITY_ACTION_INVALID** }

  *The priority action type.*

## Functions

- RTEMS_INLINE_ROUTINE unsigned int _Bitfield_Find_first_bit (unsigned int value)

  *Returns the bit number of the first bit set in the specified value.*
- RTEMS_INLINE_ROUTINE Priority_bit_map_Word _Priority_Mask (unsigned int bit_number)

  *Returns the priority bit mask for the specified major or minor bit number.*
- RTEMS_INLINE_ROUTINE unsigned int _Priority_Bits_index (unsigned int bit_number)

  *Returns the bit index position for the specified major or minor bit number.*
- RTEMS_INLINE_ROUTINE unsigned int _Priority_Major (unsigned int the_priority)

  *Returns the major portion of the_priority.*
- RTEMS_INLINE_ROUTINE unsigned int _Priority_Minor (unsigned int the_priority)

  *Returns the minor portion of the_priority.*
- RTEMS_INLINE_ROUTINE void _Priority_bit_map_Initialize (Priority_bit_map_Control *bit_map)

  *Initializes a bit map.*
- RTEMS_INLINE_ROUTINE void _Priority_bit_map_Add (Priority_bit_map_Control *bit_map, Priority_bit_map_Information *bit_map_info)

  *Adds Priority queue bit map information.*
- RTEMS_INLINE_ROUTINE void _Priority_bit_map_Remove (Priority_bit_map_Control *bit_map, Priority_bit_map_Information *bit_map_info)

  *Removes Priority queue bit map information.*
- RTEMS_INLINE_ROUTINE unsigned int _Priority_bit_map_Get_highest (const Priority_bit_map_Control *bit_map)

  *Gets highest portion of Priority queue bit map.*
- RTEMS_INLINE_ROUTINE bool _Priority_bit_map_Is_empty (const Priority_bit_map_Control *bit_map)

  *Checks if the Priority queue bit map is empty.*
- RTEMS_INLINE_ROUTINE void _Priority_bit_map_Initialize_information (Priority_bit_map_Control *bit_map, Priority_bit_map_Information *bit_map_info, unsigned int new_priority)

  *Initializes the bit map information.*
- static __inline__ void _Priority_Actions_initialize_empty (Priority_Actions *actions)

  *Initializes the priority actions empty.*
- static __inline__ void _Priority_Actions_initialize_one (Priority_Actions *actions, Priority_Aggregation *aggregation, Priority_Node *node, Priority_Action_type type)

  *Initializes the priority actions with the given information.*
- static __inline__ bool _Priority_Actions_is_empty (const Priority_Actions *actions)

  *Checks if the priority actions is empty.*
- static __inline__ bool _Priority_Actions_is_valid (const Priority_Aggregation *aggregation)

  *Checks if the priority actions is valid.*
- static __inline__ Priority_Aggregation * _Priority_Actions_move (Priority_Actions *actions)

  *Moves the priority actions' actions.*
- static __inline__ void _Priority_Actions_add (Priority_Actions *actions, Priority_Aggregation *aggregation)

  *Adds actions to the priority actions' actions.*
- static __inline__ void _Priority_Node_initialize (Priority_Node *node, Priority_Control priority)

  *Initializes the priority node to the given priority.*
- static __inline__ void _Priority_Node_set_priority (Priority_Node *node, Priority_Control priority)

  *Sets the priority of the priority node to the given priority.*
- static __inline__ void _Priority_Node_set_inactive (Priority_Node *node)

  *Sets the priority node inactive.*
- static __inline__ bool _Priority_Node_is_active (const Priority_Node *node)

  *Checks if the priority node is active.*
- static __inline__ void _Priority_Initialize_empty (Priority_Aggregation *aggregation)

  *Initializes the priority aggregation empty.*
- static __inline__ void _Priority_Initialize_one (Priority_Aggregation *aggregation, Priority_Node *node)

*Initializes the priority aggregation with the given information.*

- static __inline__ bool _Priority_Is_empty (const Priority_Aggregation *aggregation)

  *Checks if the priority aggregation is empty.*

- static __inline__ Priority_Control _Priority_Get_priority (const Priority_Aggregation *aggregation)

  *Gets the priority aggregation's priority.*

- static __inline__ const Scheduler_Control * _Priority_Get_scheduler (const Priority_Aggregation *aggregation)

  *Gets the priority aggregation's scheduler.*

- static __inline__ Priority_Node * _Priority_Get_minimum_node (const Priority_Aggregation *aggregation)

  *Gets the minimum node of the priority aggregation.*

- static __inline__ void _Priority_Set_action_node (Priority_Aggregation *aggregation, Priority_Node *node)

  *Sets the action node of the priority aggregation.*

- static __inline__ void _Priority_Set_action_type (Priority_Aggregation *aggregation, Priority_Action_type type)

  *Sets the action type of the priority aggregation.*

- static __inline__ void _Priority_Set_action (Priority_Aggregation *aggregation, Priority_Node *node, Priority_Action_type type)

  *Sets the action type and action node of the priority aggregation.*

- static __inline__ Priority_Aggregation * _Priority_Get_next_action (const Priority_Aggregation *aggregation)

  *Gets the next action of the priority aggregation.*

- static __inline__ bool _Priority_Less (const void *left, const RBTree_Node *right)

  *Compares two priorities.*

- static __inline__ bool _Priority_Plain_insert (Priority_Aggregation *aggregation, Priority_Node *node, Priority_Control priority)

  *Inserts the node with the given priority into the priority aggregation's contributors.*

- static __inline__ void _Priority_Plain_extract (Priority_Aggregation *aggregation, Priority_Node *node)

  *Extracts the priority node from the aggregation.*

- static __inline__ void _Priority_Plain_changed (Priority_Aggregation *aggregation, Priority_Node *node)

  *Updates the priority of the node in the aggregation.*

- static __inline__ void _Priority_Change_nothing (Priority_Aggregation *aggregation, bool prepend_it, Priority_Actions *actions, void *arg)

  *Does nothing.*

- static __inline__ void _Priority_Remove_nothing (Priority_Aggregation *aggregation, Priority_Actions *actions, void *arg)

  *Does nothing.*

- static __inline__ void _Priority_Non_empty_insert (Priority_Aggregation *aggregation, Priority_Node *node, Priority_Actions *actions, Priority_Change_handler change, void *arg)

  *Inserts the node in a nonempty aggregation and handles change if the node is the new minimum.*

- static __inline__ void _Priority_Insert (Priority_Aggregation *aggregation, Priority_Node *node, Priority_Actions *actions, Priority_Add_handler add, Priority_Change_handler change, void *arg)

- static __inline__ void _Priority_Extract (Priority_Aggregation *aggregation, Priority_Node *node, Priority_Actions *actions, Priority_Remove_handler remove, Priority_Change_handler change, void *arg)

  *Extracts the node from the aggregation.*

- static __inline__ void _Priority_Extract_non_empty (Priority_Aggregation *aggregation, Priority_Node *node, Priority_Actions *actions, Priority_Change_handler change, void *arg)

  *Extracts the node from the aggregation.*

- static __inline__ void _Priority_Changed (Priority_Aggregation *aggregation, Priority_Node *node, bool prepend_it, Priority_Actions *actions, Priority_Change_handler change, void *arg)

  *Updates the priority of the node in the aggregation.*

- static __inline__ void _Priority_Replace (Priority_Aggregation *aggregation, Priority_Node *victim, Priority_Node *replacement)

  *Replaces one node by another.*

### Variables

- const unsigned char _Bitfield_Leading_zeros [256]

## 8.112.1 Detailed Description

Priority Handler.

This handler encapsulates functionality which is used to manage thread priorities. The actual priority of a thread is an aggregation of priority nodes. The thread priority aggregation for the home scheduler instance of a thread consists of at least one priority node, which is normally the real priority of the thread. The locking protocols (e.g. priority ceiling and priority inheritance), rate-monotonic period objects and the POSIX sporadic server add, change and remove priority nodes.

## 8.112.2 Macro Definition Documentation

### 8.112.2.1 PRIORITY_DEFAULT_MAXIMUM

```
#define PRIORITY_DEFAULT_MAXIMUM 255
```

The default lowest (least important) thread priority value.

This value is CPU port dependent.

Definition at line 92 of file priority.h.

### 8.112.2.2 PRIORITY_PSEUDO_ISR

```
#define PRIORITY_PSEUDO_ISR PRIORITY_MINIMUM
```

The priority value of pseudo-ISR threads.

Examples are the MPCI and timer server threads.

Definition at line 82 of file priority.h.

## 8.112.3 Typedef Documentation

#### 8.112.3.1 Priority_Control

```
typedef uint64_t Priority_Control
```

The thread priority control.

Lower values represent higher priorities. So, a priority value of zero represents the highest priority thread. This value is reserved for internal threads and the priority ceiling protocol.

The format of the thread priority control depends on the context. A thread priority control may contain a user visible priority for API import/export. It may also contain a scheduler internal priority value. Values are translated via the scheduler map/unmap priority operations. The format of scheduler interal values depend on the particular scheduler implementation. It may for example encode a deadline in case of the EDF scheduler.

The thread priority control value contained in the scheduler node (Scheduler_Node::Priority::value) uses the least-significant bit to indicate if the thread should be appended or prepended to its priority group, see SCHEDULER_PRIORITY_APPEND().

Definition at line 70 of file priority.h.

### 8.112.4 Function Documentation

#### 8.112.4.1 _Bitfield_Find_first_bit()

```
RTEMS_INLINE_ROUTINE unsigned int _Bitfield_Find_first_bit (
              unsigned int value )
```

Returns the bit number of the first bit set in the specified value.

The correspondence between the bit number and actual bit position is CPU architecture dependent. The search for the first bit set may run from most to least significant bit or vice-versa.

**Parameters**

| | |
|---|---|
| *value* | The value to bit scan. |

**Returns**

The bit number of the first bit set.

**See also**

_Priority_Bits_index() and _Priority_Mask().

Definition at line 58 of file prioritybitmapimpl.h.

#### 8.112.4.2 _Priority_Actions_add()

```
static __inline__ void _Priority_Actions_add (
            Priority_Actions * actions,
            Priority_Aggregation * aggregation )  [static]
```

Adds actions to the priority actions' actions.

**Parameters**

| | | |
|---|---|---|
| in,out | *actions* | The priority actions to add actions to. |
| out | *aggregation* | The actions to add to *actions*. |

Definition at line 135 of file priorityimpl.h.

#### 8.112.4.3 _Priority_Actions_initialize_empty()

```
static __inline__ void _Priority_Actions_initialize_empty (
            Priority_Actions * actions )  [static]
```

Initializes the priority actions empty.

**Parameters**

| | | |
|---|---|---|
| out | *actions* | The actions to be initialized empty. |

Definition at line 44 of file priorityimpl.h.

#### 8.112.4.4 _Priority_Actions_initialize_one()

```
static __inline__ void _Priority_Actions_initialize_one (
            Priority_Actions * actions,
            Priority_Aggregation * aggregation,
            Priority_Node * node,
            Priority_Action_type type )  [static]
```

Initializes the priority actions with the given information.

**Parameters**

| | | |
|---|---|---|
| out | *actions* | The actions to be initialized. |
| | *aggregation* | The aggregation for the *actions* to be initialized. |
| | *node* | The action node for the *actions* to be initialized. |
| | *type* | The action type for the *actions* to be initialized. |

Definition at line 59 of file priorityimpl.h.

#### 8.112.4.5 _Priority_Actions_is_empty()

```
static __inline__ bool _Priority_Actions_is_empty (
            const Priority_Actions * actions )  [static]
```

Checks if the priority actions is empty.

**Parameters**

| | |
|---|---|
| *actions* | The priority actions to check if it is empty. |

**Return values**

| | |
|---|---|
| *true* | The priority actions *actions* is empty. |
| *false* | The priority actions *actions* is empty. |

Definition at line 83 of file priorityimpl.h.

#### 8.112.4.6 _Priority_Actions_is_valid()

```
static __inline__ bool _Priority_Actions_is_valid (
            const Priority_Aggregation * aggregation )  [static]
```

Checks if the priority actions is valid.

**Parameters**

| | |
|---|---|
| *aggregation* | The aggregation of the priority action. |

**Return values**

| | |
|---|---|
| *true* | The *aggregation* is valid. |
| *false* | The *aggregation* is not valid. |

Definition at line 98 of file priorityimpl.h.

#### 8.112.4.7 _Priority_Actions_move()

```
static __inline__ Priority_Aggregation* _Priority_Actions_move (
            Priority_Actions * actions )  [static]
```

Moves the priority actions' actions.

**Parameters**

| | | |
|---|---|---|
| in,out | *actions* | The priority actions to move the actions away from. |

**Returns**

The former actions of *actions* that were moved.

Definition at line 117 of file priorityimpl.h.

### 8.112.4.8 _Priority_bit_map_Add()

```
RTEMS_INLINE_ROUTINE void _Priority_bit_map_Add (
            Priority_bit_map_Control * bit_map,
            Priority_bit_map_Information * bit_map_info )
```

Adds Priority queue bit map information.

Priority Queue implemented by bit map.

**Parameters**

| | | |
|---|---|---|
| out | *bit_map* | The bit map to be altered by *bit_map_info*. |
| | *bit_map_info* | The information with which to alter *bit_map*. |

Definition at line 162 of file prioritybitmapimpl.h.

### 8.112.4.9 _Priority_bit_map_Get_highest()

```
RTEMS_INLINE_ROUTINE unsigned int _Priority_bit_map_Get_highest (
            const Priority_bit_map_Control * bit_map )
```

Gets highest portion of Priority queue bit map.

**Parameters**

| | |
|---|---|
| *bit_map* | The bitmap to get the highest portion from. |

**Returns**

The highest portion of the bitmap.

Definition at line 196 of file prioritybitmapimpl.h.

#### 8.112.4.10 _Priority_bit_map_Initialize()

```
RTEMS_INLINE_ROUTINE void _Priority_bit_map_Initialize (
            Priority_bit_map_Control * bit_map )
```

Initializes a bit map.

**Parameters**

| | | |
|---|---|---|
| out | *bit_map* | The bit map to initialize. |

Definition at line 147 of file prioritybitmapimpl.h.

#### 8.112.4.11 _Priority_bit_map_Initialize_information()

```
RTEMS_INLINE_ROUTINE void _Priority_bit_map_Initialize_information (
            Priority_bit_map_Control * bit_map,
            Priority_bit_map_Information * bit_map_info,
            unsigned int new_priority )
```

Initializes the bit map information.

**Parameters**

| | | |
|---|---|---|
| | *bit_map* | The bit map for the initialization of the bit map information. |
| out | *bit_map_info* | The bit map information to initialize. |
| | *new_priority* | The new priority for the initialization of the bit map information. |

Definition at line 234 of file prioritybitmapimpl.h.

#### 8.112.4.12 _Priority_bit_map_Is_empty()

```
RTEMS_INLINE_ROUTINE bool _Priority_bit_map_Is_empty (
            const Priority_bit_map_Control * bit_map )
```

Checks if the Priority queue bit map is empty.

**Parameters**

| | |
|---|---|
| *bit_map* | The bit map of which to check if it is empty. |

**Return values**

| | |
|---|---|
| *true* | The Priority queue bit map is empty |
| *false* | The Priority queue bit map is not empty. |

Definition at line 218 of file prioritybitmapimpl.h.

#### 8.112.4.13 _Priority_bit_map_Remove()

```
RTEMS_INLINE_ROUTINE void _Priority_bit_map_Remove (
             Priority_bit_map_Control * bit_map,
             Priority_bit_map_Information * bit_map_info )
```

Removes Priority queue bit map information.

Priority Queue implemented by bit map.

**Parameters**

| | | |
|---|---|---|
| out | *bit_map* | The bit map to be altered by *bit_map_info*. |
| | *bit_map_info* | The information with which to alter *bit_map*. |

Definition at line 179 of file prioritybitmapimpl.h.

#### 8.112.4.14 _Priority_Bits_index()

```
RTEMS_INLINE_ROUTINE unsigned int _Priority_Bits_index (
             unsigned int bit_number )
```

Returns the bit index position for the specified major or minor bit number.

**Parameters**

| | |
|---|---|
| *bit_number* | The bit number for which we need an index. |

**Returns**

The corresponding array index into the priority bit map.

Definition at line 107 of file prioritybitmapimpl.h.

#### 8.112.4.15 _Priority_Change_nothing()

```
static __inline__ void _Priority_Change_nothing (
             Priority_Aggregation * aggregation,
             bool prepend_it,
             Priority_Actions * actions,
             void * arg )  [static]
```

Does nothing.

This method does nothing.

**Parameters**

| | |
|---|---|
| *aggregation* | Is ignored by the method. |
| *prepend_it* | Is ignored by the method. |
| *actions* | Is ignored by the method. |
| *arg* | Is ignored by the method. |

Definition at line 488 of file priorityimpl.h.

#### 8.112.4.16 _Priority_Changed()

```
static __inline__ void _Priority_Changed (
            Priority_Aggregation * aggregation,
            Priority_Node * node,
            bool prepend_it,
            Priority_Actions * actions,
            Priority_Change_handler change,
            void * arg )  [static]
```

Updates the priority of the node in the aggregation.

This method handles the case that *node* was the minimal node in *aggregation*.

**Parameters**

| | | |
|---|---|---|
| in,out | *aggregation* | The aggregation to change the node in. |
| | *node* | The node that has a new priority and will be reinserted in the aggregation. |
| | *prepend_it* | Indicates whether *change* should prepend if the minimal priority is incorrectly set after the change. |
| | *actions* | The actions for the case that the minimal priority is incorrectly set after the change. |
| | *change* | Is called if the minimal priority is incorrectly set after the change. |
| | *arg* | The arguments for *change*. |

Definition at line 673 of file priorityimpl.h.

#### 8.112.4.17 _Priority_Extract()

```
static __inline__ void _Priority_Extract (
            Priority_Aggregation * aggregation,
            Priority_Node * node,
            Priority_Actions * actions,
            Priority_Remove_handler remove,
            Priority_Change_handler change,
            void * arg )  [static]
```

Extracts the node from the aggregation.

This method extracts the node from the aggregation and handles the cases that *aggregation* is empty after the extraction, or that *node* was the minimal node in *aggregation* by calling remove (if empty) or change (if *node* was the minimal node).

**Parameters**

| | | |
|---|---|---|
| in,out | *aggregation* | The aggregation to extract the *node* from. |
| | *node* | The node to extract from *aggregation*. |
| | *actions* | The actions for the cases that the aggregation is empty after the extraction, or the minimal node was extracted. |
| | *remove* | Is called in the case that the aggregation is empty after the extraction. |
| | *change* | Is called in the case that the minimal node was extracted. |
| | *arg* | The arguments for *remove* and *change*. |

Definition at line 599 of file priorityimpl.h.

#### 8.112.4.18 _Priority_Extract_non_empty()

```
static __inline__ void _Priority_Extract_non_empty (
            Priority_Aggregation * aggregation,
            Priority_Node * node,
            Priority_Actions * actions,
            Priority_Change_handler change,
            void * arg )  [static]
```

Extracts the node from the aggregation.

This method extracts the node from the aggregation and handles the case that *node* was the minimal node in *aggregation* by calling change (if *node* was the minimal node).

**Parameters**

| | | |
|---|---|---|
| in,out | *aggregation* | The aggregation to extract the *node* from. |
| | *node* | The node to extract from *aggregation*. |
| | *actions* | The actions for the case that the minimal node was extracted. |
| | *change* | Is called in the case that the minimal node was extracted. |
| | *arg* | The arguments for *change*. |

Definition at line 637 of file priorityimpl.h.

#### 8.112.4.19 _Priority_Get_minimum_node()

```
static __inline__ Priority_Node* _Priority_Get_minimum_node (
            const Priority_Aggregation * aggregation )  [static]
```

Gets the minimum node of the priority aggregation.

**Parameters**

| | |
|---|---|
| *aggregation* | The priority aggregation to get the minimum node from. |

**Returns**

The minimum node of *aggregation*

Definition at line 302 of file priorityimpl.h.

#### 8.112.4.20 _Priority_Get_next_action()

```
static __inline__ Priority_Aggregation* _Priority_Get_next_action (
            const Priority_Aggregation * aggregation )  [static]
```

Gets the next action of the priority aggregation.

**Parameters**

| | |
|---|---|
| *aggregation* | The priority aggregation to get the next action of. |

**Return values**

| | |
|---|---|
| *next_action* | The next action of *aggregation* if RTEMS_SMP is defined. |
| *NULL* | RTEMS_SMP is not defined. |

Definition at line 363 of file priorityimpl.h.

#### 8.112.4.21 _Priority_Get_priority()

```
static __inline__ Priority_Control _Priority_Get_priority (
            const Priority_Aggregation * aggregation )  [static]
```

Gets the priority aggregation's priority.

**Parameters**

| | |
|---|---|
| *aggregation* | The priority aggregation to get the priority from. |

**Returns**

The priority of *aggregation*.

Definition at line 270 of file priorityimpl.h.

#### 8.112.4.22 _Priority_Get_scheduler()

```
static __inline__ const Scheduler_Control* _Priority_Get_scheduler (
            const Priority_Aggregation * aggregation )  [static]
```

Gets the priority aggregation's scheduler.

**Parameters**

| | |
|---|---|
| *aggregation* | The priority aggregation to get the scheduler from. |

**Returns**

The scheduler of *aggregation*.

Definition at line 284 of file priorityimpl.h.

#### 8.112.4.23 _Priority_Initialize_empty()

```
static __inline__ void _Priority_Initialize_empty (
            Priority_Aggregation * aggregation )  [static]
```

Initializes the priority aggregation empty.

**Parameters**

| | | |
|---|---|---|
| out | *aggregation* | The priority aggregaton to initialize empty. |

Definition at line 211 of file priorityimpl.h.

#### 8.112.4.24 _Priority_Initialize_one()

```
static __inline__ void _Priority_Initialize_one (
            Priority_Aggregation * aggregation,
            Priority_Node * node )  [static]
```

Initializes the priority aggregation with the given information.

**Parameters**

| | | |
|---|---|---|
| out | *aggregation* | The priority aggregaton to initialize. |
| | *node* | The priority node to initialize *aggregation* with. |

Definition at line 232 of file priorityimpl.h.

### 8.112.4.25 _Priority_Insert()

```
static __inline__ void _Priority_Insert (
            Priority_Aggregation * aggregation,
            Priority_Node * node,
            Priority_Actions * actions,
            Priority_Add_handler add,
            Priority_Change_handler change,
            void * arg )  [static]
```

**Parameters**

| | | |
|---|---|---|
| in,out | *aggregation* | The aggregation to insert the node in. |
| | *node* | The node to be inserted. The node's priority is used as key. |
| | *actions* | Priority actions for the case that the aggregation was empty before the insert or the node is the new minimum. |
| | *add* | Is called in the case that *aggregation* was empty before the insert. |
| | *change* | Is called in the case that *aggregation* was nonempty before the insert and *node* is the new minimum of the aggregation. |
| | *arg* | The arguments for *change*. |

Definition at line 565 of file priorityimpl.h.

### 8.112.4.26 _Priority_Is_empty()

```
static __inline__ bool _Priority_Is_empty (
            const Priority_Aggregation * aggregation )  [static]
```

Checks if the priority aggregation is empty.

**Parameters**

| | |
|---|---|
| *aggregation* | The priority aggregation that shall be verified if it is empty. |

**Return values**

| | |
|---|---|
| *true* | The priority aggregation is empty. |
| *false* | The priority aggregation is not empty. |

Definition at line 256 of file priorityimpl.h.

### 8.112.4.27 _Priority_Less()

```
static __inline__ bool _Priority_Less (
```

```
            const void * left,
            const RBTree_Node * right )  [static]
```

Compares two priorities.

**Parameters**

| | |
|---|---|
| *left* | The priority control on the left hand side of the comparison. |
| *right* | THe RBTree_Node to get the priority for the comparison from. |

**Return values**

| | |
|---|---|
| *true* | The priority on the left hand side of the comparison is smaller. |
| *false* | The priority on the left hand side of the comparison is greater of equal. |

Definition at line 384 of file priorityimpl.h.

#### 8.112.4.28 _Priority_Major()

```
RTEMS_INLINE_ROUTINE unsigned int _Priority_Major (
            unsigned int the_priority )
```

Returns the major portion of the_priority.

**Parameters**

| | |
|---|---|
| *the_priority* | The priority of which we want the major portion. |

**Returns**

The major portion of the priority.

Definition at line 125 of file prioritybitmapimpl.h.

#### 8.112.4.29 _Priority_Mask()

```
RTEMS_INLINE_ROUTINE Priority_bit_map_Word _Priority_Mask (
            unsigned int bit_number )
```

Returns the priority bit mask for the specified major or minor bit number.

**Parameters**

| | |
|---|---|
| *bit_number* | The bit number for which we need a mask. |

**Returns**

The priority bit mask.

Definition at line 88 of file prioritybitmapimpl.h.

#### 8.112.4.30 _Priority_Minor()

```
RTEMS_INLINE_ROUTINE unsigned int _Priority_Minor (
              unsigned int the_priority )
```

Returns the minor portion of the_priority.

**Parameters**

| | |
|---|---|
| *the_priority* | The priority of which we want the minor portion. |

**Returns**

The minor portion of the priority.

Definition at line 137 of file prioritybitmapimpl.h.

#### 8.112.4.31 _Priority_Node_initialize()

```
static __inline__ void _Priority_Node_initialize (
              Priority_Node * node,
              Priority_Control priority )  [static]
```

Initializes the priority node to the given priority.

**Parameters**

| | | |
|---|---|---|
| out | *node* | The priority node to be initialized. |
| | *priority* | The priority to initialize *node* to. |

Definition at line 156 of file priorityimpl.h.

#### 8.112.4.32 _Priority_Node_is_active()

```
static __inline__ bool _Priority_Node_is_active (
              const Priority_Node * node )  [static]
```

Checks if the priority node is active.

**Parameters**

| | |
|---|---|
| *node* | The priority node that shall be verified if it is active. |

**Return values**

| | |
|---|---|
| *true* | The priority node is active. |
| *false* | The priority node is inactive. |

Definition at line 199 of file priorityimpl.h.

#### 8.112.4.33 _Priority_Node_set_inactive()

```
static __inline__ void _Priority_Node_set_inactive (
            Priority_Node * node )  [static]
```

Sets the priority node inactive.

**Parameters**

| | | |
|---|---|---|
| in,out | *node* | The priority node to set inactive. |

Definition at line 184 of file priorityimpl.h.

#### 8.112.4.34 _Priority_Node_set_priority()

```
static __inline__ void _Priority_Node_set_priority (
            Priority_Node * node,
            Priority_Control priority )  [static]
```

Sets the priority of the priority node to the given priority.

**Parameters**

| | | |
|---|---|---|
| out | *node* | The priority node to set the priority of. |
| | *priority* | The new priority for *node*. |

Definition at line 171 of file priorityimpl.h.

#### 8.112.4.35 _Priority_Non_empty_insert()

```
static __inline__ void _Priority_Non_empty_insert (
            Priority_Aggregation * aggregation,
```

```
            Priority_Node * node,
            Priority_Actions * actions,
            Priority_Change_handler change,
            void * arg )  [static]
```

Inserts the node in a nonempty aggregation and handles change if the node is the new minimum.

**Parameters**

| | | |
|---|---|---|
| in,out | *aggregation* | The aggregation to insert the node into. |
| | *node* | The node to be inserted. The node's priority is used as a key. |
| | *actions* | Parameter for *change* that is used if the node is the new minimum. |
| | *change* | The priority change handler that is called in the case that the new node is the minimum of the aggregation. |
| | *arg* | Arguments for *change* that is used if the node is the new minimum. |

Definition at line 534 of file priorityimpl.h.

#### 8.112.4.36 _Priority_Plain_changed()

```
static __inline__ void _Priority_Plain_changed (
            Priority_Aggregation * aggregation,
            Priority_Node * node )  [static]
```

Updates the priority of the node in the aggregation.

It first extracts the node and inserts it again, with the new *node* priority as key. This method does not handle the case that *node* was the minimal node.

**Parameters**

| | | |
|---|---|---|
| in,out | *aggregation* | The aggregation to change the node in. |
| | *node* | The node that has a new priority and will be reinserted in the aggregation. |

Definition at line 450 of file priorityimpl.h.

#### 8.112.4.37 _Priority_Plain_extract()

```
static __inline__ void _Priority_Plain_extract (
            Priority_Aggregation * aggregation,
            Priority_Node * node )  [static]
```

Extracts the priority node from the aggregation.

This method does not handle the case that *node* was the minimal node.

**Parameters**

| | | |
|---|---|---|
| in,out | *aggregation* | The aggregation to extract the node from. |
| | *node* | The node to be extracted. |

Definition at line 433 of file priorityimpl.h.

#### 8.112.4.38 _Priority_Plain_insert()

```
static __inline__ bool _Priority_Plain_insert (
            Priority_Aggregation * aggregation,
            Priority_Node * node,
            Priority_Control priority )  [static]
```

Inserts the node with the given priority into the priority aggregation's contributors.

This method does not handle the case that *node* was the minimal node.

**Parameters**

| | | |
|---|---|---|
| in,out | *aggregation* | The aggregation to insert the node in its contributors. |
| in | *node* | The node to insert in the contributors. |
| | *priority* | The priority control for the inserted node. |

**Return values**

| | |
|---|---|
| *true* | The inserted node with its priority is the minimum of the RBTree. |
| *false* | The inserted node with its priority is not the minimum of the RBTree. |

Definition at line 411 of file priorityimpl.h.

#### 8.112.4.39 _Priority_Remove_nothing()

```
static __inline__ void _Priority_Remove_nothing (
            Priority_Aggregation * aggregation,
            Priority_Actions * actions,
            void * arg )  [static]
```

Does nothing.

This method does nothing.

**Parameters**

| | |
|---|---|
| *aggregation* | Is ignored by the method. |
| *actions* | Is ignored by the method. |
| *arg* | Is ignored by the method. |

Definition at line 510 of file priorityimpl.h.

#### 8.112.4.40 _Priority_Replace()

```
static __inline__ void _Priority_Replace (
            Priority_Aggregation * aggregation,
            Priority_Node * victim,
            Priority_Node * replacement )  [static]
```

Replaces one node by another.

The new node has the priority of the old node.

**Parameters**

| | | |
|---|---|---|
| in,out | *aggregation* | The aggregation to replace *victim* by *replacement* in. |
| | *victim* | The node that will be replaced. |
| out | *replacement* | The node that replaces *victim*. It obtains its priority from *victim*. |

Definition at line 704 of file priorityimpl.h.

#### 8.112.4.41 _Priority_Set_action()

```
static __inline__ void _Priority_Set_action (
            Priority_Aggregation * aggregation,
            Priority_Node * node,
            Priority_Action_type type )  [static]
```

Sets the action type and action node of the priority aggregation.

**Parameters**

| | | |
|---|---|---|
| out | *aggregation* | The priority aggregation to set the action type and action node of. |
| | *node* | The new action node for *aggregation*. |
| | *type* | The new action type for *aggregation*. |

Definition at line 345 of file priorityimpl.h.

#### 8.112.4.42 _Priority_Set_action_node()

```
static __inline__ void _Priority_Set_action_node (
            Priority_Aggregation * aggregation,
            Priority_Node * node )  [static]
```

Sets the action node of the priority aggregation.

**Parameters**

| | | |
|---|---|---|
| out | *aggregation* | The priority aggregation to set the action node of. |
| | *node* | The new priority node for *aggregation*. |

Definition at line 315 of file priorityimpl.h.

#### 8.112.4.43 _Priority_Set_action_type()

```
static __inline__ void _Priority_Set_action_type (
            Priority_Aggregation * aggregation,
            Priority_Action_type type )  [static]
```

Sets the action type of the priority aggregation.

**Parameters**

| | | |
|---|---|---|
| out | *aggregation* | The priority aggregation to set the action type of. |
| | *type* | The new action type for *aggregation*. |

Definition at line 329 of file priorityimpl.h.

### 8.112.5 Variable Documentation

#### 8.112.5.1 _Bitfield_Leading_zeros

```
const unsigned char _Bitfield_Leading_zeros[256]  [extern]
```

This table is used by the generic bitfield routines to perform a highly optimized bit scan without the use of special CPU instructions.

# 8.113 Processor Mask

Processor Mask.

## Files

- file processormask.h

  *Processor Mask API.*
- file processormaskcopy.c

  *Processor Mask Implementation.*

## Enumerations

- enum **Processor_mask_Copy_status** { **PROCESSOR_MASK_COPY_LOSSLESS**, **PROCESSOR_MASK_COPY_PARTIAL_LOSS**, **PROCESSOR_MASK_COPY_COMPLETE_LOSS**, **PROCESSOR_MASK_COPY_INVALID_SIZE** }

## Functions

- typedef BITSET_DEFINE (Processor_mask, CPU_MAXIMUM_PROCESSORS) Processor_mask

  *A bit map which is large enough to provide one bit for each processor in the system.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_Zero (Processor_mask *mask)

  *Sets the bits of the mask to zero, also considers CPU_MAXIMUM_PROCESSORS.*
- RTEMS_INLINE_ROUTINE bool _Processor_mask_Is_zero (const Processor_mask *mask)

  *Checks if the mask is zero, also considers CPU_MAXIMUM_PROCESSORS.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_Fill (Processor_mask *mask)

  *Fills the mask, also considers CPU_MAXIMUM_PROCESSORS.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_Assign (Processor_mask *dst, const Processor_mask *src)

  *Copies the mask to another mask, also considers CPU_MAXIMUM_PROCESSORS.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_Set (Processor_mask *mask, uint32_t index)

  *Sets the specified index bit of the mask.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_Clear (Processor_mask *mask, uint32_t index)

  *Clears the specified index bit of the mask.*
- RTEMS_INLINE_ROUTINE bool _Processor_mask_Is_set (const Processor_mask *mask, uint32_t index)

  *Checks if the specified index bit of the mask is set.*
- RTEMS_INLINE_ROUTINE bool _Processor_mask_Is_equal (const Processor_mask *a, const Processor_mask *b)

  *Checks if the processor sets a and b are equal.*
- RTEMS_INLINE_ROUTINE bool _Processor_mask_Has_overlap (const Processor_mask *a, const Processor_mask *b)

  *Checks if the intersection of the processor sets a and b is non-empty.*
- RTEMS_INLINE_ROUTINE bool _Processor_mask_Is_subset (const Processor_mask *big, const Processor_mask *small)

  *Checks if the processor set small is a subset of processor set big.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_And (Processor_mask *a, const Processor_mask *b, const Processor_mask *c)

  *Performs a bitwise a = b & c.*

- RTEMS_INLINE_ROUTINE void _Processor_mask_Nand (Processor_mask *a, const Processor_mask *b, const Processor_mask *c)

  *Performs a bitwise a = b & ~c.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_Or (Processor_mask *a, const Processor_mask *b, const Processor_mask *c)

  *Performs a bitwise a = b | c.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_Xor (Processor_mask *a, const Processor_mask *b, const Processor_mask *c)

  *Performs a bitwise a = b ^ c.*
- RTEMS_INLINE_ROUTINE uint32_t _Processor_mask_Count (const Processor_mask *a)

  *Gets the number of set bits in the processor mask.*
- RTEMS_INLINE_ROUTINE uint32_t _Processor_mask_Find_last_set (const Processor_mask *a)

  *Finds the last set of the processor mask.*
- RTEMS_INLINE_ROUTINE uint32_t _Processor_mask_To_uint32_t (const Processor_mask *mask, uint32_t index)

  *Returns the subset of 32 processors containing the specified index as an unsigned 32-bit integer.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_From_uint32_t (Processor_mask *mask, uint32_t bits, uint32_t index)

  *Creates a processor set from an unsigned 32-bit integer relative to the specified index.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_From_index (Processor_mask *mask, uint32_t index)

  *Creates a processor set from the specified index.*
- RTEMS_INLINE_ROUTINE bool _Processor_mask_Is_at_most_partial_loss (Processor_mask_Copy_status status)

  *Checks if the copy status guarantees at most partial loss.*
- Processor_mask_Copy_status _Processor_mask_Copy (long *dst, size_t dst_size, const long *src, size_t src_size)

  *Copies one mask to another.*
- RTEMS_INLINE_ROUTINE Processor_mask_Copy_status _Processor_mask_To_cpu_set_t (const Processor_mask *src, size_t dst_size, cpu_set_t *dst)

  *Copies one mask to another.*
- RTEMS_INLINE_ROUTINE Processor_mask_Copy_status _Processor_mask_From_cpu_set_t (Processor_mask *dst, size_t src_size, const cpu_set_t *src)

  *Copies one mask to another.*

## Variables

- const Processor_mask **_Processor_mask_The_one_and_only**

### 8.113.1 Detailed Description

Processor Mask.

The processor mask provides a bit map large enough to provide one bit for each processor in the system. It is a fixed size internal data type provided for efficiency in addition to the API level cpu_set_t.

### 8.113.2 Function Documentation

### 8.113.2.1 _Processor_mask_And()

```
RTEMS_INLINE_ROUTINE void _Processor_mask_And (
            Processor_mask * a,
            const Processor_mask * b,
            const Processor_mask * c )
```

Performs a bitwise a = b & c.

**Parameters**

| | | |
|---|---|---|
| out | *a* | The processor mask that is set by this operation. |
| | *b* | The first parameter of the AND-operation. |
| | *c* | The second parameter of the AND-operation. |

Definition at line 207 of file processormask.h.

### 8.113.2.2 _Processor_mask_Assign()

```
RTEMS_INLINE_ROUTINE void _Processor_mask_Assign (
            Processor_mask * dst,
            const Processor_mask * src )
```

Copies the mask to another mask, also considers CPU_MAXIMUM_PROCESSORS.

**Parameters**

| | | |
|---|---|---|
| out | *dst* | The mask to copy *src* to. |
| | *src* | The mask to copy to *dst*. |

Definition at line 95 of file processormask.h.

### 8.113.2.3 _Processor_mask_Clear()

```
RTEMS_INLINE_ROUTINE void _Processor_mask_Clear (
            Processor_mask * mask,
            uint32_t index )
```

Clears the specified index bit of the mask.

**Parameters**

| | | |
|---|---|---|
| out | *mask* | The mask to clear the bit of. |
| | *index* | The index of the bit that shall be cleared. |

Definition at line 122 of file processormask.h.

#### 8.113.2.4 _Processor_mask_Copy()

```
Processor_mask_Copy_status _Processor_mask_Copy (
            long * dst,
            size_t dst_size,
            const long * src,
            size_t src_size )
```

Copies one mask to another.

**Parameters**

| | | |
|---|---|---|
| out | *dst* | The destination of the copy operation. |
| | *dst_size* | The size of *dst*. |
| | *src* | The source of the copy operation. |
| | *src_size* | The size of *src*. |

**Return values**

| | |
|---|---|
| *PROCESSOR_MASK_COPY_LOSSLESS* | It is guaranteed that the copy operation is lossless. |
| *PROCESSOR_MASK_COPY_PARTIAL_LOSS* | Partial loss happened due to the sizes of *src* and *dst*. |
| *PROCESSOR_MASK_COPY_COMPLETE_LOSS* | Complete loss happened due to the sizes of *src* and *dst*. |
| *PROCESSOR_MASK_COPY_INVALID_SIZE* | One of the arguments sizes is invalid (bigger than the size of a long). |

Definition at line 31 of file processormaskcopy.c.

#### 8.113.2.5 _Processor_mask_Count()

```
RTEMS_INLINE_ROUTINE uint32_t _Processor_mask_Count (
            const Processor_mask * a )
```

Gets the number of set bits in the processor mask.

**Parameters**

| | |
|---|---|
| *a* | The processor mask of which the set bits are counted. |

**Returns**

The number of set bits in *a*.

Definition at line 271 of file processormask.h.

### 8.113.2.6 _Processor_mask_Fill()

```
RTEMS_INLINE_ROUTINE void _Processor_mask_Fill (
            Processor_mask * mask )
```

Fills the mask, also considers CPU_MAXIMUM_PROCESSORS.

**Parameters**

| | | |
|---|---|---|
| out | *mask* | The mask to fill |

Definition at line 84 of file processormask.h.

### 8.113.2.7 _Processor_mask_Find_last_set()

```
RTEMS_INLINE_ROUTINE uint32_t _Processor_mask_Find_last_set (
            const Processor_mask * a )
```

Finds the last set of the processor mask.

**Parameters**

| | |
|---|---|
| *a* | The processor mask wo find the last set of. |

**Returns**

The last set of *a*.

Definition at line 283 of file processormask.h.

### 8.113.2.8 _Processor_mask_From_cpu_set_t()

```
RTEMS_INLINE_ROUTINE Processor_mask_Copy_status _Processor_mask_From_cpu_set_t (
            Processor_mask * dst,
            size_t src_size,
            const cpu_set_t * src )
```

Copies one mask to another.

**Parameters**

| | | |
|---|---|---|
| | *src* | The source for the copy operation. |
| | *src_size* | The size of *src*. |
| out | *dst* | The destination for the copy operation. |

**Return values**

| | |
|---|---|
| *PROCESSOR_MASK_COPY_LOSSLESS* | It is guaranteed that the copy operation is lossless. |
| *PROCESSOR_MASK_COPY_PARTIAL_LOSS* | Partial loss happened due to the sizes of *src* and *dst*. |
| *PROCESSOR_MASK_COPY_COMPLETE_LOSS* | Complete loss happened due to the sizes of *src* and *dst*. |
| *PROCESSOR_MASK_COPY_INVALID_SIZE* | One of the arguments sizes is invalid (bigger than the size of a long). |

Definition at line 431 of file processormask.h.

#### 8.113.2.9 _Processor_mask_From_index()

```
RTEMS_INLINE_ROUTINE void _Processor_mask_From_index (
            Processor_mask * mask,
            uint32_t index )
```

Creates a processor set from the specified index.

**Parameters**

| | | |
|---|---|---|
| out | *The* | mask that is created. |
| | *index* | The specified index. |

Definition at line 331 of file processormask.h.

#### 8.113.2.10 _Processor_mask_From_uint32_t()

```
RTEMS_INLINE_ROUTINE void _Processor_mask_From_uint32_t (
            Processor_mask * mask,
            uint32_t bits,
            uint32_t index )
```

Creates a processor set from an unsigned 32-bit integer relative to the specified index.

**Parameters**

| | | |
|---|---|---|
| out | *mask* | The mask that is created. |
| | *bits* | The bits for creating the mask. |
| | *index* | The index to which the mask is relative. |

Definition at line 315 of file processormask.h.

#### 8.113.2.11 _Processor_mask_Has_overlap()

```
RTEMS_INLINE_ROUTINE bool _Processor_mask_Has_overlap (
              const Processor_mask * a,
              const Processor_mask * b )
```

Checks if the intersection of the processor sets a and b is non-empty.

**Parameters**

| | |
|---|---|
| *a* | The first processor set. |
| *b* | The second processor set. |

**Return values**

| | |
|---|---|
| *true* | The intersection of the processor sets a and b is non-empty. |
| *false* | The intersection of the processor sets a and b is empty. |

Definition at line 174 of file processormask.h.

#### 8.113.2.12 _Processor_mask_Is_at_most_partial_loss()

```
RTEMS_INLINE_ROUTINE bool _Processor_mask_Is_at_most_partial_loss (
              Processor_mask_Copy_status status )
```

Checks if the copy status guarantees at most partial loss.

**Parameters**

| | |
|---|---|
| *status* | The copy status to check. |

**Return values**

| | |
|---|---|
| *true* | At most partial loss can be guaranteed. |
| *false* | The status indicates more than partial loss. |

Definition at line 354 of file processormask.h.

#### 8.113.2.13 _Processor_mask_Is_equal()

```
RTEMS_INLINE_ROUTINE bool _Processor_mask_Is_equal (
              const Processor_mask * a,
              const Processor_mask * b )
```

Checks if the processor sets a and b are equal.

**Parameters**

| | |
|---|---|
| *a* | The first processor set. |
| *b* | The seconde processor set. |

**Return values**

| | |
|---|---|
| *true* | The processor sets a and b are equal. |
| *false* | The processor sets a and b are not equal. |

Definition at line 156 of file processormask.h.

#### 8.113.2.14 _Processor_mask_Is_set()

```
RTEMS_INLINE_ROUTINE bool _Processor_mask_Is_set (
              const Processor_mask * mask,
              uint32_t index )
```

Checks if the specified index bit of the mask is set.

**Parameters**

| | |
|---|---|
| *mask* | The mask to check if the specified bit is set. |
| *index* | The index of the bit that is checked. |

**Return values**

| | |
|---|---|
| *true* | The specified index bit is set. |
| *false* | The specified index bit is not set. |

Definition at line 139 of file processormask.h.

#### 8.113.2.15 _Processor_mask_Is_subset()

```
RTEMS_INLINE_ROUTINE bool _Processor_mask_Is_subset (
              const Processor_mask * big,
              const Processor_mask * small )
```

Checks if the processor set small is a subset of processor set big.

**Parameters**

| | |
|---|---|
| *big* | The bigger processor set. |
| *small* | The smaller processor set. |

**Return values**

| | |
|---|---|
| *true* | *small* is a subset of *big*. |
| *false* | *small* is not a subset of *big*. |

Definition at line 192 of file processormask.h.

### 8.113.2.16 _Processor_mask_Is_zero()

```
RTEMS_INLINE_ROUTINE bool _Processor_mask_Is_zero (
            const Processor_mask * mask )
```

Checks if the mask is zero, also considers CPU_MAXIMUM_PROCESSORS.

**Parameters**

| | |
|---|---|
| *mask* | The mask to check whether is is zero |

**Return values**

| | |
|---|---|
| *true* | The mask is zero. |
| *false* | The mask is not zero. |

Definition at line 74 of file processormask.h.

### 8.113.2.17 _Processor_mask_Nand()

```
RTEMS_INLINE_ROUTINE void _Processor_mask_Nand (
            Processor_mask * a,
            const Processor_mask * b,
            const Processor_mask * c )
```

Performs a bitwise a = b & ∼c.

**Parameters**

| | | |
|---|---|---|
| out | *a* | The processor mask that is set by this operation. |
| | *b* | The first parameter of the operation. |
| | *c* | The second parameter of the operation. |

Definition at line 223 of file processormask.h.

#### 8.113.2.18 _Processor_mask_Or()

```
RTEMS_INLINE_ROUTINE void _Processor_mask_Or (
            Processor_mask * a,
            const Processor_mask * b,
            const Processor_mask * c )
```

Performs a bitwise a = b | c.

**Parameters**

| | | |
|---|---|---|
| out | *a* | The processor mask that is set by this operation. |
| | *b* | The first parameter of the OR-operation. |
| | *c* | The second parameter of the OR-operation. |

Definition at line 239 of file processormask.h.

#### 8.113.2.19 _Processor_mask_Set()

```
RTEMS_INLINE_ROUTINE void _Processor_mask_Set (
            Processor_mask * mask,
            uint32_t index )
```

Sets the specified index bit of the mask.

**Parameters**

| | | |
|---|---|---|
| out | *mask* | The mask to set the bit of. |
| | *index* | The index of the bit that shall be set. |

Definition at line 108 of file processormask.h.

#### 8.113.2.20 _Processor_mask_To_cpu_set_t()

```
RTEMS_INLINE_ROUTINE Processor_mask_Copy_status _Processor_mask_To_cpu_set_t (
            const Processor_mask * src,
            size_t dst_size,
            cpu_set_t * dst )
```

Copies one mask to another.

**Parameters**

| | | |
|---|---|---|
| | *src* | The source for the copy operation. |
| | *dst_size* | The size of *dst*. |
| out | *dst* | The destination for the copy operation. |

**Return values**

| | |
|---|---|
| *PROCESSOR_MASK_COPY_LOSSLESS* | It is guaranteed that the copy operation is lossless. |
| *PROCESSOR_MASK_COPY_PARTIAL_LOSS* | Partial loss happened due to the sizes of *src* and *dst*. |
| *PROCESSOR_MASK_COPY_COMPLETE_LOSS* | Complete loss happened due to the sizes of *src* and *dst*. |
| *PROCESSOR_MASK_COPY_INVALID_SIZE* | One of the arguments sizes is invalid (bigger than the size of a long). |

Definition at line 401 of file processormask.h.

### 8.113.2.21 _Processor_mask_To_uint32_t()

```
RTEMS_INLINE_ROUTINE uint32_t _Processor_mask_To_uint32_t (
            const Processor_mask * mask,
            uint32_t index )
```

Returns the subset of 32 processors containing the specified index as an unsigned 32-bit integer.

**Parameters**

| | |
|---|---|
| *mask* | The processor mask. |
| *index* | The specified index. |

**Returns**

The subset containing the specified index as an unsigned 32-bit integer.

Definition at line 297 of file processormask.h.

### 8.113.2.22 _Processor_mask_Xor()

```
RTEMS_INLINE_ROUTINE void _Processor_mask_Xor (
            Processor_mask * a,
            const Processor_mask * b,
            const Processor_mask * c )
```

Performs a bitwise a = b ^ c.

**Parameters**

| | | |
|---|---|---|
| out | *a* | The processor mask that is set by this operation. |
| | *b* | The first parameter of the XOR-operation. |
| | *c* | The second parameter of the XOR-operation. |

Definition at line 255 of file processormask.h.

#### 8.113.2.23 _Processor_mask_Zero()

```
RTEMS_INLINE_ROUTINE void _Processor_mask_Zero (
            Processor_mask * mask )
```

Sets the bits of the mask to zero, also considers CPU_MAXIMUM_PROCESSORS.

**Parameters**

| | | |
|---|---|---|
| out | *mask* | The mask to set to zero. |

Definition at line 61 of file processormask.h.

# 8.114 Profiling Support

Profiling support.

## Files

- file profiling.h

  *Profiling Support API.*

## Functions

- static void _Profiling_Thread_dispatch_disable (Per_CPU_Control *cpu, uint32_t previous_thread_dispatch_disable_level)

  *Disables the thread dispatch if the previous thread dispatch disable level is zero.*
- static void _Profiling_Thread_dispatch_disable_critical (Per_CPU_Control *cpu, uint32_t previous_thread_dispatch_disable_level, const ISR_lock_Context *lock_context)

  *Disables the thread dispatch.*
- static void _Profiling_Thread_dispatch_enable (Per_CPU_Control *cpu, uint32_t new_thread_dispatch_disable_level)

  *Enables the thread dispatch.*
- static void _Profiling_Update_max_interrupt_delay (Per_CPU_Control *cpu, CPU_Counter_ticks interrupt_delay)

  *Updates the maximum interrupt delay.*
- void _Profiling_Outer_most_interrupt_entry_and_exit (Per_CPU_Control *cpu, CPU_Counter_ticks interrupt_entry_instant, CPU_Counter_ticks interrupt_exit_instant)

  *Updates the interrupt profiling statistics.*

## 8.114.1 Detailed Description

Profiling support.

## 8.114.2 Function Documentation

### 8.114.2.1 _Profiling_Outer_most_interrupt_entry_and_exit()

```
void _Profiling_Outer_most_interrupt_entry_and_exit (
            Per_CPU_Control * cpu,
            CPU_Counter_ticks interrupt_entry_instant,
            CPU_Counter_ticks interrupt_exit_instant )
```

Updates the interrupt profiling statistics.

Must be called with the interrupt stack and before the thread dispatch disable level is decremented.

**Parameters**

| | |
|---|---|
| *cpu* | The cpu control. |
| *interrupt_entry_instant* | The instant that the interrupt occured. |
| *interrupt_exit_instant* | The instant in which the interrupt was exited. |

#### 8.114.2.2 _Profiling_Thread_dispatch_disable()

```
static void _Profiling_Thread_dispatch_disable (
            Per_CPU_Control * cpu,
            uint32_t previous_thread_dispatch_disable_level )  [inline], [static]
```

Disables the thread dispatch if the previous thread dispatch disable level is zero.

**Parameters**

| | | |
|---|---|---|
| out | *cpu* | The cpu control. |
| | *previous_thread_dispatch_disable_level* | The dispatch disable level of the previous thread. |

Definition at line 51 of file profiling.h.

#### 8.114.2.3 _Profiling_Thread_dispatch_disable_critical()

```
static void _Profiling_Thread_dispatch_disable_critical (
            Per_CPU_Control * cpu,
            uint32_t previous_thread_dispatch_disable_level,
            const ISR_lock_Context * lock_context )  [inline], [static]
```

Disables the thread dispatch.

Only if the previous thread dispatch disable level is zero. This method also takes into account the lock_context.

**Parameters**

| | | |
|---|---|---|
| out | *cpu* | The cpu control. |
| | *previous_thread_dispatch_disable_level* | The dispatch disable level of the previous thread. |
| | *lock_context* | The lock context. |

Definition at line 80 of file profiling.h.

#### 8.114.2.4 _Profiling_Thread_dispatch_enable()

```
static void _Profiling_Thread_dispatch_enable (
```

```
            Per_CPU_Control * cpu,
            uint32_t new_thread_dispatch_disable_level )  [inline], [static]
```

Enables the thread dispatch.

Only if the *new_thread_dispatch_disable_level* is 0.

**Parameters**

| | | |
|---|---|---|
| out | *cpu* | The cpu control. |
| | *new_thread_dispatch_disable_level* | The dispatch disable level of the new thread. |

Definition at line 109 of file profiling.h.

#### 8.114.2.5 _Profiling_Update_max_interrupt_delay()

```
static void _Profiling_Update_max_interrupt_delay (
            Per_CPU_Control * cpu,
            CPU_Counter_ticks interrupt_delay )  [inline], [static]
```

Updates the maximum interrupt delay.

**Parameters**

| | | |
|---|---|---|
| out | *cpu* | The cpu control. |
| | *interrupt_delay* | The new interrupt delay. |

Definition at line 141 of file profiling.h.

## 8.115 Protected Heap Handler

Provides protected heap services.

### Files

- file protectedheap.h

  *Protected Heap Handler API.*

### Functions

- static __inline__ uintptr_t _Protected_heap_Initialize (Heap_Control *heap, void *area_begin, uintptr_t area_size, uintptr_t page_size)

  *Initializes the protected heap.*
- bool _Protected_heap_Extend (Heap_Control *heap, void *area_begin, uintptr_t area_size)

  *Extends the protected heap.*
- void * _Protected_heap_Allocate_aligned_with_boundary (Heap_Control *heap, uintptr_t size, uintptr_t alignment, uintptr_t boundary)

  *Allocates an aligned memory area with boundary constraint for the protected heap.*
- static __inline__ void * _Protected_heap_Allocate_aligned (Heap_Control *heap, uintptr_t size, uintptr_t alignment)

  *Allocates an aligned memory area.*
- static __inline__ void * _Protected_heap_Allocate (Heap_Control *heap, uintptr_t size)

  *Allocates a memory area.*
- bool _Protected_heap_Get_block_size (Heap_Control *heap, void *addr, uintptr_t *size)

  *Returns the size of the allocatable memory area.*
- bool _Protected_heap_Resize_block (Heap_Control *heap, void *addr, uintptr_t size)

  *Resizes the block of the allocated memory area.*
- bool _Protected_heap_Free (Heap_Control *heap, void *addr)

  *Frees the allocated memory area.*
- bool _Protected_heap_Walk (Heap_Control *heap, int source, bool dump)

  *Verifies the integrity of the heap.*
- void _Protected_heap_Iterate (Heap_Control *heap, Heap_Block_visitor visitor, void *visitor_arg)

  *Iterates over all blocks of the heap.*
- bool _Protected_heap_Get_information (Heap_Control *heap, Heap_Information_block *info)

  *Returns information about used and free blocks for the heap.*
- bool _Protected_heap_Get_free_information (Heap_Control *heap, Heap_Information *info)

  *Returns information about free blocks for the heap.*
- uintptr_t _Protected_heap_Get_size (Heap_Control *heap)

  *Returns the size of the allocatable area in bytes.*

### 8.115.1 Detailed Description

Provides protected heap services.

The ScoreAllocatorMutex is used to protect the heap accesses.

## 8.115.2 Function Documentation

### 8.115.2.1 _Protected_heap_Allocate()

```
static __inline__ void* _Protected_heap_Allocate (
            Heap_Control * heap,
            uintptr_t size )  [static]
```

Allocates a memory area.

A size value of zero will return a unique address which may be freed with _Heap_Free(). This method first locks the allocator and after the allocation of the memory area, unlocks it again.

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap to allocate a memory are from. |
| | *size* | The size of the desired memory are in bytes. |

**Return values**

| | |
|---|---|
| *pointer* | The starting address of the allocated memory area. |
| *NULL* | No memory is available of the parameters are inconsistent. |

Definition at line 137 of file protectedheap.h.

### 8.115.2.2 _Protected_heap_Allocate_aligned()

```
static __inline__ void* _Protected_heap_Allocate_aligned (
            Heap_Control * heap,
            uintptr_t size,
            uintptr_t alignment )  [static]
```

Allocates an aligned memory area.

A size value of zero will return a unique address which may be freed with _Heap_Free(). This method first locks the allocator and after the allocation of the memory area, unlocks it again.

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap to allocate a memory are from. |
| | *size* | The size of the desired memory are in bytes. |
| | *alignment* | The allocated memory area will begin at an address aligned by this value. |

**Return values**

| | |
|---|---|
| *pointer* | The starting address of the allocated memory area. |
| *NULL* | No memory is available of the parameters are inconsistent. |

Definition at line 114 of file protectedheap.h.

### 8.115.2.3 _Protected_heap_Allocate_aligned_with_boundary()

```
void* _Protected_heap_Allocate_aligned_with_boundary (
            Heap_Control * heap,
            uintptr_t size,
            uintptr_t alignment,
            uintptr_t boundary )
```

Allocates an aligned memory area with boundary constraint for the protected heap.

A size value of zero will return a unique address which may be freed with _Heap_Free(). This method first locks the allocator and after the allocation of the memory area, unlocks it again.

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap to allocate a memory are from. |
| | *size* | The size of the desired memory are in bytes. |
| | *alignment* | The allocated memory area will begin at an address aligned by this value. |
| | *boundary* | The allocated memory area will fulfill a boundary constraint, if this value is not equal to zero. The boundary value specifies the set of addresses which are aligned by the boundary value. The interior of the allocated memory area will not contain an element of this set. The begin or end address of the area may be a member of the set. |

**Return values**

| | |
|---|---|
| *pointer* | The starting address of the allocated memory area. |
| *NULL* | No memory is available of the parameters are inconsistent. |

### 8.115.2.4 _Protected_heap_Extend()

```
bool _Protected_heap_Extend (
            Heap_Control * heap,
            void * area_begin,
            uintptr_t area_size )
```

Extends the protected heap.

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap to extend. |
| | *area_begin* | The starting addres of the area to extend *heap* with. |
| | *area_size* | The size of the heap area. |

**Return values**

| | |
|---|---|
| *true* | The operation succeeded. |
| *false* | The operation failed. |

### 8.115.2.5 _Protected_heap_Free()

```
bool _Protected_heap_Free (
            Heap_Control * heap,
            void * addr )
```

Frees the allocated memory area.

Inappropriate values for *addr* may corrupt the heap. This method first locks the allocator and after the free operation, unlocks it again.

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap of the allocated memory area. |
| | *addr* | The starting address of the memory area to be freed. |

**Return values**

| | |
|---|---|
| *true* | The allocated memory area was successfully freed. |
| *false* | The method failed. |

### 8.115.2.6 _Protected_heap_Get_block_size()

```
bool _Protected_heap_Get_block_size (
            Heap_Control * heap,
            void * addr,
            uintptr_t * size )
```

Returns the size of the allocatable memory area.

The size value may be greater than the initially requested size in _Heap_Allocate_aligned_with_boundary().

Inappropriate values for *addr* will not corrupt the heap, but may yield invalid size values.

This method first locks the allocator and after the operation, unlocks it again.

**Parameters**

| | | |
|---|---|---|
| | *heap* | The heap to operate upon. |
| | *addr* | The starting address of the allocatable memory area. |
| out | *size* | Stores the size of the allocatable memory area after the method call. |

**Return values**

| | |
|---|---|
| *true* | The operation was successful. |
| *false* | The operation was not successful. |

#### 8.115.2.7 _Protected_heap_Get_free_information()

```
bool _Protected_heap_Get_free_information (
              Heap_Control * heap,
              Heap_Information * info )
```

Returns information about free blocks for the heap.

This method first locks the allocator and after the operation, unlocks it again.

**Parameters**

| | | |
|---|---|---|
| | *heap* | The heap to get the information from. |
| out | *info* | Stores the information about free blocks of *heap* after the method call. |

#### 8.115.2.8 _Protected_heap_Get_information()

```
bool _Protected_heap_Get_information (
              Heap_Control * heap,
              Heap_Information_block * info )
```

Returns information about used and free blocks for the heap.

This method first locks the allocator and after the operation, unlocks it again.

**Parameters**

| | | |
|---|---|---|
| | *heap* | The heap to get the information from. |
| out | *info* | Stores the information of the *heap* after the method call. |

### 8.115.2.9 _Protected_heap_Get_size()

```
uintptr_t _Protected_heap_Get_size (
            Heap_Control * heap )
```

Returns the size of the allocatable area in bytes.

This value is an integral multiple of the page size.

**Parameters**

| | |
|---|---|
| *heap* | The heap to get the allocatable area from. |

**Returns**

The size of the allocatable area in *heap* in bytes.

### 8.115.2.10 _Protected_heap_Initialize()

```
static __inline__ uintptr_t _Protected_heap_Initialize (
            Heap_Control * heap,
            void * area_begin,
            uintptr_t area_size,
            uintptr_t page_size )  [static]
```

Initializes the protected heap.

**Parameters**

| | | |
|---|---|---|
| out | *heap* | The heap to initialize. |
| | *area_begin* | The starting address of the heap area. |
| | *area_size* | The size of the heap area. |
| | *page_size* | The page size for the heap. |

Definition at line 48 of file protectedheap.h.

### 8.115.2.11 _Protected_heap_Iterate()

```
void _Protected_heap_Iterate (
            Heap_Control * heap,
            Heap_Block_visitor visitor,
            void * visitor_arg )
```

Iterates over all blocks of the heap.

This method first locks the allocator and after the operation, unlocks it again.

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap to iterate over. |
| | *visitor* | This will be called for each heap block with the argument *visitor_arg*. |
| in,out | *visitor_arg* | The argument for all calls of *visitor*. |

### 8.115.2.12 _Protected_heap_Resize_block()

```
bool _Protected_heap_Resize_block (
            Heap_Control * heap,
            void * addr,
            uintptr_t size )
```

Resizes the block of the allocated memory area.

Inappropriate values for *addr* may corrupt the heap.

This method first locks the allocator and after the resize, unlocks it again.

**Parameters**

| | | |
|---|---|---|
| in,out | *heap* | The heap to operate upon. |
| | *addr* | The starting address of the allocated memory area to be resized. |
| | *size* | The least possible size for the new memory area. Resize may be impossible and depends on the current heap usage. |
| out | *old_size* | Stores the size available for allocation in the current block before the resize after the method call. |
| out | *new_size* | Stores the size available for allocation in the resized block after the method call. In the case of an unsuccessful resize, zero is returned in this parameter |

**Return values**

| | |
|---|---|
| *HEAP_RESIZE_SUCCESSFUL* | The resize was successful. |
| *HEAP_RESIZE_UNSATISFIED* | The least possible size *size* was too big. Resize not possible. |
| *HEAP_RESIZE_FATAL_ERROR* | The block starting at *addr* is not part of the heap. |

### 8.115.2.13 _Protected_heap_Walk()

```
bool _Protected_heap_Walk (
            Heap_Control * heap,
            int source,
            bool dump )
```

Verifies the integrity of the heap.

Walks the heap to verify its integrity. This method first locks the allocator and after the operation, unlocks it again, if the thread dispatch is enabled.

**Parameters**

| | |
|---|---|
| *heap* | The heap whose integrity is to be verified. |
| *source* | If *dump* is `true`, this is used to mark the output lines. |
| *dump* | Indicates whether diagnostic messages will be printed to standard output. |

**Return values**

| | |
|---|---|
| *true* | No errors occured, the heap´s integrity is not violated. |
| *false* | The heap is corrupt. |

## 8.116 RTEMS Allocator Mutex

Protection for all memory allocations and deallocations in RTEMS.

### Functions

- void **_RTEMS_Lock_allocator** (void)
- void **_RTEMS_Unlock_allocator** (void)
- bool **_RTEMS_Allocator_is_owner** (void)

### 8.116.1 Detailed Description

Protection for all memory allocations and deallocations in RTEMS.

When the APIs all use this for allocation and deallocation protection, then this possibly should be renamed and moved to a higher level in the hierarchy.

# 8.117 RTEMS Copyright Notice

Copyright Notice for RTEMS.

## Files

- file copyrt.h

  *Copyright Notice for RTEMS.*

## Variables

- const char _Copyright_Notice [ ]
- const char _RTEMS_version [ ]

  *This constant provides the RTEMS version string.*

## 8.117.1 Detailed Description

Copyright Notice for RTEMS.

## 8.117.2 Variable Documentation

### 8.117.2.1 _Copyright_Notice

```
const char _Copyright_Notice[]  [extern]
```

This is the copyright string for RTEMS.

### 8.117.2.2 _RTEMS_version

```
const char _RTEMS_version[]  [extern]
```

This constant provides the RTEMS version string.

The constant name does not follow the naming conventions. Do not change it for backward compatibility reasons.

**See also**

rtems_get_version_string()

Definition at line 29 of file rtems-version.c.

## 8.118 RTEMS Per CPU Information

### Modules

- Flexible Per-CPU Data

  *Flexible Per-CPU Data.*

### Files

- file percpu.c

  *Allocate and Initialize Per CPU Structures.*

### Classes

- struct Per_CPU_Job_context

  *Context for per-processor jobs.*
- struct Per_CPU_Job

  *A per-processor job.*
- struct Per_CPU_Stats

  *Per-CPU statistics.*
- struct Per_CPU_Control

  *Per CPU Core Structure.*
- struct Per_CPU_Control_envelope

### Macros

- #define **PER_CPU_JOB_DONE** 1
- #define **_Per_CPU_Acquire**(cpu, lock_context) _ISR_lock_Acquire( &( cpu )->Lock, lock_context )
- #define **_Per_CPU_Release**(cpu, lock_context) _ISR_lock_Release( &( cpu )->Lock, lock_context )
- #define **_Per_CPU_Get_snapshot**() ( &_Per_CPU_Information[ _SMP_Get_current_processor() ].per_cpu )
- #define **_Thread_Dispatch_disable_level** _Per_CPU_Get()->thread_dispatch_disable_level
- #define **_Thread_Heir** _Per_CPU_Get()->heir
- #define **_Thread_Executing** _Per_CPU_Get_executing( _Per_CPU_Get() )
- #define **_ISR_Nest_level** _Per_CPU_Get()->isr_nest_level
- #define **_CPU_Interrupt_stack_low** _Per_CPU_Get()->interrupt_stack_low
- #define **_CPU_Interrupt_stack_high** _Per_CPU_Get()->interrupt_stack_high
- #define **_Thread_Dispatch_necessary** _Per_CPU_Get()->dispatch_necessary

### Typedefs

- typedef void(* **Per_CPU_Job_handler**) (void *arg)
- typedef struct Per_CPU_Job Per_CPU_Job

  *A per-processor job.*
- typedef struct Per_CPU_Control Per_CPU_Control

  *Per CPU Core Structure.*

## Enumerations

- enum Per_CPU_State {
  PER_CPU_STATE_INITIAL, PER_CPU_STATE_READY_TO_START_MULTITASKING, PER_CPU_STATE_REQUEST_STA
  PER_CPU_STATE_UP,
  PER_CPU_STATE_SHUTDOWN }

  *State of a processor.*
- enum Per_CPU_Watchdog_index { PER_CPU_WATCHDOG_TICKS, PER_CPU_WATCHDOG_REALTIME, PER_CPU_WATCHDOG_MONOTONIC, PER_CPU_WATCHDOG_COUNT }

  *Per-CPU watchdog header index.*

## Functions

- static Per_CPU_Control * **_Per_CPU_Get** (void)
- static Per_CPU_Control * **_Per_CPU_Get_by_index** (uint32_t index)
- static uint32_t **_Per_CPU_Get_index** (const Per_CPU_Control *cpu)
- static struct _Thread_Control * **_Per_CPU_Get_executing** (const Per_CPU_Control *cpu)
- static bool **_Per_CPU_Is_processor_online** (const Per_CPU_Control *cpu)
- static bool **_Per_CPU_Is_boot_processor** (const Per_CPU_Control *cpu)
- static __inline__ void **_Per_CPU_Acquire_all** (ISR_lock_Context *lock_context)
- static __inline__ void **_Per_CPU_Release_all** (ISR_lock_Context *lock_context)
- void _Per_CPU_Initialize (void)

  *Allocate and Initialize Per CPU Structures.*
- void **_Per_CPU_State_change** (Per_CPU_Control *cpu, Per_CPU_State new_state)
- bool _Per_CPU_State_wait_for_non_initial_state (uint32_t cpu_index, uint32_t timeout_in_ns)

  *Waits for a processor to change into a non-initial state.*
- void _Per_CPU_Perform_jobs (Per_CPU_Control *cpu)

  *Performs the jobs of the specified processor in FIFO order.*
- void _Per_CPU_Add_job (Per_CPU_Control *cpu, Per_CPU_Job *job)

  *Adds the job to the tail of the processing list of the specified processor.*
- void _Per_CPU_Wait_for_job (const Per_CPU_Control *cpu, const Per_CPU_Job *job)

  *Waits for the job carried out by the specified processor.*
- static __inline__ struct _Thread_Control * _Thread_Get_executing (void)

  *Returns the thread control block of the executing thread.*

## Variables

- Per_CPU_Control_envelope _Per_CPU_Information[ ] CPU_STRUCTURE_ALIGNMENT

  *Set of Per CPU Core Information.*

### 8.118.1 Detailed Description

This defines the per CPU state information required by RTEMS and the BSP. In an SMP configuration, there will be multiple instances of this data structure – one per CPU – and the current CPU number will be used as the index.

### 8.118.2 Typedef Documentation

#### 8.118.2.1 Per_CPU_Control

```
typedef struct Per_CPU_Control Per_CPU_Control
```

Per CPU Core Structure.

This structure is used to hold per core state information.

#### 8.118.2.2 Per_CPU_Job

```
typedef struct Per_CPU_Job Per_CPU_Job
```

A per-processor job.

This structure must be as small as possible due to stack space constraints in _SMP_Multicast_action().

### 8.118.3 Enumeration Type Documentation

#### 8.118.3.1 Per_CPU_State

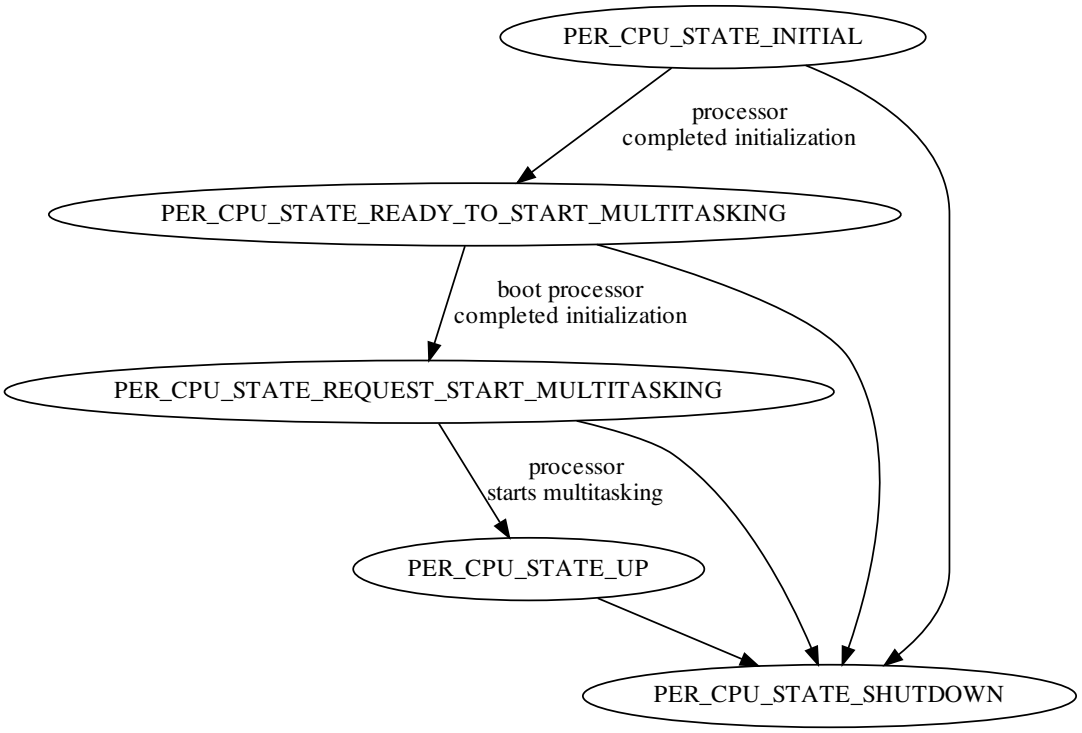
```
enum Per_CPU_State
```

State of a processor.

The processor state controls the life cycle of processors at the lowest level. No multi-threading or other high-level concepts matter here.

State changes must be initiated via _Per_CPU_State_change(). This function may not return in case someone requested a shutdown. The _SMP_Send_message() function will be used to notify other processors about state changes if the other processor is in the up state.

Due to the sequential nature of the basic system initialization one processor has a special role. It is the processor executing the boot_card() function. This processor is called the boot processor. All other processors are called secondary.




**Enumerator**

| | |
|---|---|
| PER_CPU_STATE_INITIAL | The per CPU controls are initialized to zero. The boot processor executes the sequential boot code in this state. The secondary processors should perform their basic initialization now and change into the PER_C↩ PU_STATE_READY_TO_START_MULTITASKING state once this is complete. |
| PER_CPU_STATE_READY_TO_START_MULTIT↩ ASKING | Processor is ready to start multitasking. The secondary processor performed its basic initialization and is ready to receive inter-processor interrupts. Interrupt delivery must be disabled in this state, but requested inter-processor interrupts must be recorded and must be delivered once the secondary processor enables interrupts for the first time. The boot processor will wait for all secondary processors to change into this state. In case a secondary processor does not reach this state the system will not start. The secondary processors wait now for a change into the PER_CPU_STATE_REQUEST_START_MULTITA↩ SKING state set by the boot processor once all secondary processors reached the PER_CPU_STA↩ TE_READY_TO_START_MULTITASKING state. |

**Enumerator**

| | |
|---|---|
| PER_CPU_STATE_REQUEST_START_MULTITA↩ SKING | Multitasking start of processor is requested. The boot processor completed system initialization and is about to perform a context switch to its heir thread. Secondary processors should now issue a context switch to the heir thread. This normally enables interrupts on the processor for the first time. |
| PER_CPU_STATE_UP | Normal multitasking state. |
| PER_CPU_STATE_SHUTDOWN | This is the terminal state. |

Definition at line 128 of file percpu.h.

#### 8.118.3.2 Per_CPU_Watchdog_index

```
enum Per_CPU_Watchdog_index
```

Per-CPU watchdog header index.

**Enumerator**

| | |
|---|---|
| PER_CPU_WATCHDOG_TICKS | Index for tick clock per-CPU watchdog header. The reference time point for the tick clock is the system start. The clock resolution is one system clock tick. It is used for the system clock tick based time services. |
| PER_CPU_WATCHDOG_REALTIME | Index for realtime clock per-CPU watchdog header. The reference time point for the realtime clock is the POSIX Epoch. The clock resolution is one nanosecond. It is used for the time of day services and the POSIX services using CLOCK_REALTIME. |
| PER_CPU_WATCHDOG_MONOTONIC | Index for monotonic clock per-CPU watchdog header. The reference time point for the monotonic clock is the system start. The clock resolution is one nanosecond. It is used for the POSIX services using CLOCK_MONOTONIC. |
| PER_CPU_WATCHDOG_COUNT | Count of per-CPU watchdog headers. |

Definition at line 308 of file percpu.h.

### 8.118.4 Function Documentation

#### 8.118.4.1 _Per_CPU_Add_job()

```
void _Per_CPU_Add_job (
            Per_CPU_Control * cpu,
            Per_CPU_Job * job )
```

Adds the job to the tail of the processing list of the specified processor.

This function does not send the SMP_MESSAGE_PERFORM_JOBS message the specified processor.

**Parameters**

| | | |
|---|---|---|
| in,out | *cpu* | The processor to add the job. |
| in,out | *job* | The job. The Per_CPU_Job::context member must be initialized by the caller. |

Definition at line 64 of file smpmulticastaction.c.

### 8.118.4.2 _Per_CPU_Initialize()

```
void _Per_CPU_Initialize (
             void  )
```

Allocate and Initialize Per CPU Structures.

This method allocates and initialize the per CPU structure.

### 8.118.4.3 _Per_CPU_Perform_jobs()

```
void _Per_CPU_Perform_jobs (
             Per_CPU_Control * cpu )
```

Performs the jobs of the specified processor in FIFO order.

**Parameters**

| | | |
|---|---|---|
| in,out | *cpu* | The jobs of this processor will be performed. |

Definition at line 41 of file smpmulticastaction.c.

### 8.118.4.4 _Per_CPU_State_wait_for_non_initial_state()

```
bool _Per_CPU_State_wait_for_non_initial_state (
             uint32_t cpu_index,
             uint32_t timeout_in_ns )
```

Waits for a processor to change into a non-initial state.

This function should be called only in _CPU_SMP_Start_processor() if required by the CPU port or BSP.

```
bool _CPU_SMP_Start_processor(uint32_t cpu_index)
{
  uint32_t timeout = 123456;
  start_the_processor(cpu_index);
  return _Per_CPU_State_wait_for_non_initial_state(cpu_index, timeout);
}
```

**Parameters**

| | | |
|---|---|---|
| in | *cpu_index* | The processor index. |
| in | *timeout_in_ns* | The timeout in nanoseconds. Use a value of zero to wait forever if necessary. |

**Return values**

| | |
|---|---|
| *true* | The processor is in a non-initial state. |
| *false* | The timeout expired before the processor reached a non-initial state. |

Definition at line 22 of file percpustatewait.c.

#### 8.118.4.5 _Per_CPU_Wait_for_job()

```
void _Per_CPU_Wait_for_job (
             const Per_CPU_Control * cpu,
             const Per_CPU_Job * job )
```

Waits for the job carried out by the specified processor.

This function may result in an SMP_FATAL_WRONG_CPU_STATE_TO_PERFORM_JOBS fatal error.

**Parameters**

| | | |
|---|---|---|
| in | *cpu* | The processor carrying out the job. |
| in | *job* | The job to wait for. |

Definition at line 105 of file smpmulticastaction.c.

#### 8.118.4.6 _Thread_Get_executing()

```
static __inline__ struct _Thread_Control* _Thread_Get_executing (
             void  )  [static]
```

Returns the thread control block of the executing thread.

This function can be called in any thread context. On SMP configurations, interrupts are disabled to ensure that the processor index is used consistently if no CPU port specific method is available to get the executing thread.

**Returns**

The thread control block of the executing thread.

Definition at line 878 of file percpu.h.

### 8.118.5 Variable Documentation

#### 8.118.5.1 CPU_STRUCTURE_ALIGNMENT

```
Per_CPU_Control_envelope _Per_CPU_Information [] CPU_STRUCTURE_ALIGNMENT  [extern]
```

Set of Per CPU Core Information.

This is an array of per CPU core information.

# 8.119 RTEMS Print Support

## Classes

- struct rtems_printer
- struct rtems_printer_task_context

## Typedefs

- typedef int(∗ rtems_print_printer) (void ∗, const char ∗format, va_list ap)

## Functions

- static bool rtems_print_printer_valid (const rtems_printer ∗printer)

  *check if the printer is valid.*
- static void rtems_print_printer_empty (rtems_printer ∗printer)

  *Initializes the printer to print nothing.*
- void rtems_print_printer_printk (rtems_printer ∗printer)

  *Initializes the printer to print via printk().*
- void rtems_print_printer_printf (rtems_printer ∗printer)

  *Initializes the printer to print via printf().*
- void rtems_print_printer_fprintf (rtems_printer ∗printer, FILE ∗file)

  *Initializes the printer to print via fprintf() using the specified file stream.*
- void rtems_print_printer_fprintf_putc (rtems_printer ∗printer)

  *Initializes the printer to print via fprintf() using an unbuffered FILE stream with output through rtems_putc().*
- static void **rtems_printer_task_initialize** (rtems_printer_task_context ∗context)
- static void **rtems_printer_task_set_stack_size** (rtems_printer_task_context ∗context, size_t stack_size)
- static void **rtems_printer_task_set_priority** (rtems_printer_task_context ∗context, rtems_task_priority priority)
- static void **rtems_printer_task_set_file_descriptor** (rtems_printer_task_context ∗context, int fd)
- static void **rtems_printer_task_set_buffer_table** (rtems_printer_task_context ∗context, void ∗buffer_table)
- static void **rtems_printer_task_set_buffer_count** (rtems_printer_task_context ∗context, size_t buffer_↩ count)
- static void **rtems_printer_task_set_buffer_size** (rtems_printer_task_context ∗context, size_t buffer_size)
- int rtems_print_printer_task (rtems_printer ∗printer, rtems_printer_task_context ∗context)

  *Creates a printer task.*
- void rtems_printer_task_drain (rtems_printer_task_context ∗context)

  *Drains the work queue of the printer task.*

### 8.119.1 Detailed Description

This module contains all methods and support related to providing the user with an interface to the kernel level print support.

### 8.119.2 Typedef Documentation

#### 8.119.2.1 rtems_print_printer

```
typedef int(* rtems_print_printer) (void *, const char *format, va_list ap)
```

Type definition for function which can be plugged in to certain reporting routines to redirect the output.

Use the RTEMS Print interface to call these functions. Do not directly use them.

If the user provides their own printer, then they may redirect those reports as they see fit.

Definition at line 49 of file printer.h.

### 8.119.3 Function Documentation

#### 8.119.3.1 rtems_print_printer_empty()

```
static void rtems_print_printer_empty (
            rtems_printer * printer )  [inline], [static]
```

Initializes the printer to print nothing.

An empty printer prints nothing. You can use this to implement an enable and disable type print implementation.

**Parameters**

| | | |
|---|---|---|
| in | *printer* | Pointer to the printer structure. |

Definition at line 80 of file printer.h.

#### 8.119.3.2 rtems_print_printer_fprintf()

```
void rtems_print_printer_fprintf (
            rtems_printer * printer,
            FILE * file )
```

Initializes the printer to print via fprintf() using the specified file stream.

**Parameters**

| | | |
|---|---|---|
| in | *printer* | Pointer to the printer structure. |

#### 8.119.3.3 rtems_print_printer_fprintf_putc()

```
void rtems_print_printer_fprintf_putc (
            rtems_printer * printer )
```

Initializes the printer to print via fprintf() using an unbuffered FILE stream with output through rtems_putc().

**Parameters**

| | | |
|---|---|---|
| in | *printer* | Pointer to the printer structure. |

#### 8.119.3.4 rtems_print_printer_printf()

```
void rtems_print_printer_printf (
            rtems_printer * printer )
```

Initializes the printer to print via printf().

**Parameters**

| | | |
|---|---|---|
| in | *printer* | Pointer to the printer structure. |

#### 8.119.3.5 rtems_print_printer_printk()

```
void rtems_print_printer_printk (
            rtems_printer * printer )
```

Initializes the printer to print via printk().

**Parameters**

| | | |
|---|---|---|
| in | *printer* | Pointer to the printer structure. |

Definition at line 34 of file printk_plugin.c.

#### 8.119.3.6 rtems_print_printer_task()

```
int rtems_print_printer_task (
            rtems_printer * printer,
            rtems_printer_task_context * context )
```

Creates a printer task.

Print requests via rtems_printf() or rtems_vprintf() using a printer task printer are output to a buffer and then placed on a work queue in FIFO order. The work queue is emptied by the printer task. The printer task writes the buffer content to the file descriptor specified by the context. Buffers are allocated from a pool of buffers as specified by the context.

**Parameters**

| | | |
|---|---|---|
| in | *printer* | Pointer to the printer structure. |
| in | *context* | The initialized printer task context. |

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *EINVAL* | Invalid context parameters. |
| *ENOMEM* | Not enough resources. |

#### 8.119.3.7 rtems_print_printer_valid()

```
static bool rtems_print_printer_valid (
              const rtems_printer * printer )  [inline], [static]
```

check if the printer is valid.

**Parameters**

| | | |
|---|---|---|
| in | *printer* | Pointer to the printer structure. |

**Returns**

true The printer is valid else false is returned.

Definition at line 67 of file printer.h.

#### 8.119.3.8 rtems_printer_task_drain()

```
void rtems_printer_task_drain (
              rtems_printer_task_context * context )
```

Drains the work queue of the printer task.

Waits until all output buffers in the work queue at the time of this function call are written to the file descriptor and an fsync() completed.

The printer task must be successfully started via rtems_print_printer_task() before this function can be used. Otherwise, the behaviour is undefined.

**Parameters**

| | | |
|---|---|---|
| in | *context* | The printer task context of a successfully started printer task. |

## 8.120 RTEMS Status Code Checks

Checks for RTEMS status codes (rtems_status_code).

### Macros

- #define **T_rsc**(a, e) T_flags_rsc(a, e, 0)
- #define **T_assert_rsc**(a, e) T_flags_rsc(a, e, T_CHECK_STOP)
- #define **T_quiet_rsc**(a, e) T_flags_rsc(a, e, T_CHECK_QUIET)
- #define **T_step_rsc**(s, a, e) T_flags_rsc(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_rsc**(s, a, e) T_flags_rsc(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_rsc_success**(a) T_flags_rsc_success(a, 0)
- #define **T_assert_rsc_success**(a) T_flags_rsc_success(a, T_CHECK_STOP)
- #define **T_quiet_rsc_success**(a) T_flags_rsc_success(a, T_CHECK_QUIET)
- #define **T_step_rsc_success**(s, a) T_flags_rsc_success(a, T_CHECK_STEP(s))
- #define **T_step_assert_rsc_success**(s, a) T_flags_rsc_success(a, T_CHECK_STEP(s) | T_CHECK_S↩ TOP)

### 8.120.1 Detailed Description

Checks for RTEMS status codes (rtems_status_code).

## 8.121 RTEMS Termios Device Support

RTEMS Termios Device Support Internal Data Structures.

RTEMS Termios Device Support Internal Data Structures.

## 8.122 RTEMS Test Framework

The RTEMS Test Framework helps you to write tests.

### Modules

- Boolean Checks

  *Checks for boolean expressions.*
- Character Checks

  *Checks for characters (char).*
- Destructors

  *Support to run destructors at the end of a test case.*
- Generic Checks

  *Checks for data types with an equality or inequality operator.*
- Memory Area Checks

  *Checks for memory areas.*
- POSIX Status and Error Number Checks

  *Checks for POSIX status and error numbers.*
- Pointer Checks

  *Checks for pointers.*
- RTEMS Status Code Checks

  *Checks for RTEMS status codes (rtems_status_code).*
- RTEMS Test Framework Implementation

  *Implementation details.*
- Runtime Measurements

  *Support to measure the runtime of code fragments.*
- Signed 16-Bit Integer Checks

  *Checks for signed 16-bit integers (int16_t).*
- Signed 32-Bit Integer Checks

  *Checks for signed 32-bit integers (int32_t).*
- Signed 64-Bit Integer Checks

  *Checks for signed 64-bit integers (int64_t).*
- Signed 8-Bit Integer Checks

  *Checks for signed 8-bit integers (int8_t).*
- Signed Character Checks

  *Checks for signed characters (signed char).*
- Signed Integer Checks

  *Checks for signed integers (int).*
- Signed Long Integer Checks

  *Checks for signed long integers (long).*
- Signed Pointer Value Checks

  *Checks for signed pointer values (intptr_t).*
- Signed Short Integer Checks

  *Checks for signed short integers (short).*
- Signed Size Checks

  *Checks for signed sizes (ssize_t).*
- Size Checks

  *Checks for sizes (size_t).*
- String Checks

*Checks for strings.*

- Time Services

  *Time service functions.*
- Unsigned 16-Bit Integer Checks

  *Checks for unsigned 16-bit integers (uint16_t).*
- Unsigned 32-Bit Integer Checks

  *Checks for unsigned 32-bit integers (uint32_t).*
- Unsigned 64-Bit Integer Checks

  *Checks for unsigned 64-bit integers (uint64_t).*
- Unsigned 8-Bit Integer Checks

  *Checks for unsigned 8-bit integers (uint8_t).*
- Unsigned Character Checks

  *Checks for unsigned characters (unsigned char).*
- Unsigned Integer Checks

  *Checks for unsigned integers (unsigned int).*
- Unsigned Long Integer Checks

  *Checks for unsigned long integers (unsigned long).*
- Unsigned Long Long Integer Checks

  *Checks for unsigned long long integers (unsigned long long).*
- Unsigned Pointer Value Checks

  *Checks for unsigned pointer values (uintptr_t).*
- Unsigned Short Integer Checks

  *Checks for unsigned short integers (unsigned short).*

## Files

- file t-test-rtems-objs.c

  *RTEMS Objects Support for Test Framework.*

## Classes

- struct T_fixture
- struct T_fixture_node

## Macros

- #define **T_ARRAY_SIZE**(array) (sizeof(array) / sizeof((array)[0]))
- #define **T_FILE_NAME** __FILE__
- #define **T_ZERO_LENGTH_ARRAY**
- #define **T_ZERO_LENGTH_ARRAY_EXTENSION**(n) (n)

## Typedefs

- typedef struct T_fixture **T_fixture**
- typedef struct T_fixture_node **T_fixture_node**

## Enumerations

- enum **T_verbosity** { **T_QUIET**, **T_NORMAL**, **T_VERBOSE** }

### 8.122.1 Detailed Description

The RTEMS Test Framework helps you to write tests.

## 8.123 RTEMS Test Framework Implementation

Implementation details.

### Files

- file t-test-busy-tick.c

  *Implementation of T_get_one_clock_tick_busy().*
- file t-test-busy.c

  *Implementation of T_busy().*
- file t-test-interrupt.c

  *Implementation of T_interrupt_test().*
- file t-test-rtems.h

  *RTEMS Support for Test Framework.*
- file t-test-thread-switch.c

  *Implementation of T_thread_switch_record().*

### Classes

- struct T_case_context
- struct T_check_context
- struct T_check_context_msg

### Macros

- #define **T_NO_RETURN** _Noreturn
- #define **T_CHECK_STOP** 1U
- #define **T_CHECK_QUIET** 2U
- #define **T_CHECK_FMT** 4U
- #define **T_CHECK_STEP_FLAG** 8U
- #define **T_CHECK_STEP_TO_FLAGS**(step) ((unsigned int)(step) << 8)
- #define **T_CHECK_STEP_FROM_FLAGS**(flags) ((flags) >> 8)
- #define **T_CHECK_STEP**(step) (T_CHECK_STEP_TO_FLAGS(step) | T_CHECK_STEP_FLAG)
- #define **T_VA_ARGS_FIRST**(...) T_VA_ARGS_FIRST_SELECT(__VA_ARGS__, throw_away)
- #define **T_VA_ARGS_FIRST_SELECT**(first, ...) first
- #define **T_VA_ARGS_MORE**(...) T_VA_ARGS_XEXPAND(T_VA_ARGS_KIND(__VA_ARGS__), __VA↩_ARGS__)
- #define **T_VA_ARGS_XEXPAND**(kind, ...) T_VA_ARGS_EXPAND(kind, __VA_ARGS__)
- #define **T_VA_ARGS_EXPAND**(kind, ...) T_VA_ARGS_EXPAND_##kind(__VA_ARGS__)
- #define **T_VA_ARGS_EXPAND_0U**(first)
- #define **T_VA_ARGS_EXPAND_4U**(first, ...) , __VA_ARGS__
- #define **T_VA_ARGS_KIND**(...)
- #define **T_VA_ARGS_SELECT**(a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, ...) a10
- #define **T_flags_true**(flags, ...)
- #define **T_flags_eq**(flags, a, ...)
- #define **T_flags_ne**(flags, a, ...)
- #define **T_flags_eq_ptr**(a, e, flags, sa, se)
- #define **T_flags_ne_ptr**(a, e, flags, sa, se)
- #define **T_flags_null**(a, flags, sa)
- #define **T_flags_not_null**(a, flags, sa)

- #define **T_flags_eq_mem**(a, e, n, flags, sa, se, sn)
- #define **T_flags_ne_mem**(a, e, n, flags, sa, se, sn)
- #define **T_flags_eq_str**(a, e, flags)
- #define **T_flags_ne_str**(a, e, flags)
- #define **T_flags_eq_nstr**(a, e, n, flags)
- #define **T_flags_ne_nstr**(a, e, n, flags)
- #define **T_flags_eq_char**(a, e, flags)
- #define **T_flags_ne_char**(a, e, flags)
- #define **T_flags_eq_int**(a, e, flags)
- #define **T_flags_ne_int**(a, e, flags)
- #define **T_flags_ge_int**(a, e, flags)
- #define **T_flags_gt_int**(a, e, flags)
- #define **T_flags_le_int**(a, e, flags)
- #define **T_flags_lt_int**(a, e, flags)
- #define **T_flags_eq_uint**(a, e, flags)
- #define **T_flags_ne_uint**(a, e, flags)
- #define **T_flags_ge_uint**(a, e, flags)
- #define **T_flags_gt_uint**(a, e, flags)
- #define **T_flags_le_uint**(a, e, flags)
- #define **T_flags_lt_uint**(a, e, flags)
- #define **T_flags_eq_long**(a, e, flags)
- #define **T_flags_ne_long**(a, e, flags)
- #define **T_flags_ge_long**(a, e, flags)
- #define **T_flags_gt_long**(a, e, flags)
- #define **T_flags_le_long**(a, e, flags)
- #define **T_flags_lt_long**(a, e, flags)
- #define **T_flags_eq_ulong**(a, e, flags)
- #define **T_flags_ne_ulong**(a, e, flags)
- #define **T_flags_ge_ulong**(a, e, flags)
- #define **T_flags_gt_ulong**(a, e, flags)
- #define **T_flags_le_ulong**(a, e, flags)
- #define **T_flags_lt_ulong**(a, e, flags)
- #define **T_flags_eq_ll**(a, e, flags)
- #define **T_flags_ne_ll**(a, e, flags)
- #define **T_flags_ge_ll**(a, e, flags)
- #define **T_flags_gt_ll**(a, e, flags)
- #define **T_flags_le_ll**(a, e, flags)
- #define **T_flags_lt_ll**(a, e, flags)
- #define **T_flags_eq_ull**(a, e, flags)
- #define **T_flags_ne_ull**(a, e, flags)
- #define **T_flags_ge_ull**(a, e, flags)
- #define **T_flags_gt_ull**(a, e, flags)
- #define **T_flags_le_ull**(a, e, flags)
- #define **T_flags_lt_ull**(a, e, flags)
- #define **T_flags_eno**(a, e, flags)
- #define **T_flags_eno_success**(a, flags)
- #define **T_flags_psx_error**(a, eno, flags)
- #define **T_flags_psx_success**(a, flags)

## Typedefs

- typedef struct T_case_context **T_case_context**

## Functions

- void **T_case_register** (T_case_context *)
- void **T_check** (const T_check_context *, bool,...)
- void **T_check_eq_ptr** (const T_check_context_msg *, const void *, const void *)
- void **T_check_ne_ptr** (const T_check_context_msg *, const void *, const void *)
- void **T_check_null** (const T_check_context_msg *, const void *)
- void **T_check_not_null** (const T_check_context_msg *, const void *)
- void **T_check_eq_mem** (const T_check_context_msg *, const void *, const void *, size_t)
- void **T_check_ne_mem** (const T_check_context_msg *, const void *, const void *, size_t)
- void **T_check_eq_str** (const T_check_context *, const char *, const char *)
- void **T_check_ne_str** (const T_check_context *, const char *, const char *)
- void **T_check_eq_nstr** (const T_check_context *, const char *, const char *, size_t)
- void **T_check_ne_nstr** (const T_check_context *, const char *, const char *, size_t)
- void **T_check_eq_char** (const T_check_context *, char, char)
- void **T_check_ne_char** (const T_check_context *, char, char)
- void **T_check_eq_int** (const T_check_context *, int, int)
- void **T_check_ne_int** (const T_check_context *, int, int)
- void **T_check_ge_int** (const T_check_context *, int, int)
- void **T_check_gt_int** (const T_check_context *, int, int)
- void **T_check_le_int** (const T_check_context *, int, int)
- void **T_check_lt_int** (const T_check_context *, int, int)
- void **T_check_eq_uint** (const T_check_context *, unsigned int, unsigned int)
- void **T_check_ne_uint** (const T_check_context *, unsigned int, unsigned int)
- void **T_check_ge_uint** (const T_check_context *, unsigned int, unsigned int)
- void **T_check_gt_uint** (const T_check_context *, unsigned int, unsigned int)
- void **T_check_le_uint** (const T_check_context *, unsigned int, unsigned int)
- void **T_check_lt_uint** (const T_check_context *, unsigned int, unsigned int)
- void **T_check_eq_long** (const T_check_context *, long, long)
- void **T_check_ne_long** (const T_check_context *, long, long)
- void **T_check_ge_long** (const T_check_context *, long, long)
- void **T_check_gt_long** (const T_check_context *, long, long)
- void **T_check_le_long** (const T_check_context *, long, long)
- void **T_check_lt_long** (const T_check_context *, long, long)
- void **T_check_eq_ulong** (const T_check_context *, unsigned long, unsigned long)
- void **T_check_ne_ulong** (const T_check_context *, unsigned long, unsigned long)
- void **T_check_ge_ulong** (const T_check_context *, unsigned long, unsigned long)
- void **T_check_gt_ulong** (const T_check_context *, unsigned long, unsigned long)
- void **T_check_le_ulong** (const T_check_context *, unsigned long, unsigned long)
- void **T_check_lt_ulong** (const T_check_context *, unsigned long, unsigned long)
- void **T_check_eq_ll** (const T_check_context *, long long, long long)
- void **T_check_ne_ll** (const T_check_context *, long long, long long)
- void **T_check_ge_ll** (const T_check_context *, long long, long long)
- void **T_check_gt_ll** (const T_check_context *, long long, long long)
- void **T_check_le_ll** (const T_check_context *, long long, long long)
- void **T_check_lt_ll** (const T_check_context *, long long, long long)
- void **T_check_eq_ull** (const T_check_context *, unsigned long long, unsigned long long)
- void **T_check_ne_ull** (const T_check_context *, unsigned long long, unsigned long long)
- void **T_check_ge_ull** (const T_check_context *, unsigned long long, unsigned long long)
- void **T_check_gt_ull** (const T_check_context *, unsigned long long, unsigned long long)
- void **T_check_le_ull** (const T_check_context *, unsigned long long, unsigned long long)
- void **T_check_lt_ull** (const T_check_context *, unsigned long long, unsigned long long)
- void **T_check_eno** (const T_check_context *, int, int)
- void **T_check_eno_success** (const T_check_context *, int)
- void **T_check_psx_error** (const T_check_context *, int, int)
- void **T_check_psx_success** (const T_check_context *, int)
- Objects_Maximum **T_objects_count** (Objects_APIs api, uint16_t cls)
- void **T_objects_check** (Objects_APIs api, uint16_t cls, Objects_Maximum *expected, const char *name)

## Variables

- const T_check_context **T_special**

### 8.123.1 Detailed Description

Implementation details.

### 8.123.2 Macro Definition Documentation

#### 8.123.2.1 T_flags_eno

```
#define T_flags_eno(
              a,
              e,
              flags )
```

**Value:**

```
{                                   \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_eno(&T_check_instance, a, e);               \
}
```

Definition at line 639 of file test.h.

#### 8.123.2.2 T_flags_eno_success

```
#define T_flags_eno_success(
              a,
              flags )
```

**Value:**

```
{                                   \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_eno_success(&T_check_instance, a);          \
}
```

Definition at line 648 of file test.h.

#### 8.123.2.3 T_flags_eq

```
#define T_flags_eq(
               flags,
               a,
               ...  )
```

**Value:**

```
    T_flags_true(flags, \
    (a) == (T_VA_ARGS_FIRST(__VA_ARGS__)) T_VA_ARGS_MORE(__VA_ARGS__))
```

Definition at line 183 of file test.h.

#### 8.123.2.4 T_flags_eq_char

```
#define T_flags_eq_char(
               a,
               e,
               flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_eq_char(&T_check_instance, a, e);           \
}
```

Definition at line 291 of file test.h.

#### 8.123.2.5 T_flags_eq_int

```
#define T_flags_eq_int(
               a,
               e,
               flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_eq_int(&T_check_instance, a, e);            \
}
```

Definition at line 309 of file test.h.

#### 8.123.2.6 T_flags_eq_ll

```
#define T_flags_eq_ll(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {         \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };      \
    T_check_eq_ll(&T_check_instance, a, e);              \
}
```

Definition at line 525 of file test.h.

#### 8.123.2.7 T_flags_eq_long

```
#define T_flags_eq_long(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {         \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };      \
    T_check_eq_long(&T_check_instance, a, e);            \
}
```

Definition at line 417 of file test.h.

#### 8.123.2.8 T_flags_eq_mem

```
#define T_flags_eq_mem(
                a,
                e,
                n,
                flags,
                sa,
                se,
                sn )
```

**Value:**

```
{                                       \
    static const T_check_context_msg T_check_instance = {         \
        { T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT },        \
        "memcmp(" sa ", " se ", " sn ") == 0" };             \
    T_check_eq_mem(&T_check_instance, a, e, n);          \
}
```

Definition at line 232 of file test.h.

#### 8.123.2.9 T_flags_eq_nstr

```
#define T_flags_eq_nstr(
                a,
                e,
                n,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_eq_nstr(&T_check_instance, a, e, n);            \
}
```

Definition at line 272 of file test.h.

#### 8.123.2.10 T_flags_eq_ptr

```
#define T_flags_eq_ptr(
                a,
                e,
                flags,
                sa,
                se )
```

**Value:**

```
{                                       \
    static const T_check_context_msg T_check_instance = {        \
        { T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT },       \
        sa " == " se };                     \
    T_check_eq_ptr(&T_check_instance, a, e);            \
}
```

Definition at line 193 of file test.h.

#### 8.123.2.11 T_flags_eq_str

```
#define T_flags_eq_str(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_eq_str(&T_check_instance, a, e);            \
}
```

Definition at line 253 of file test.h.

#### 8.123.2.12 T_flags_eq_uint

```
#define T_flags_eq_uint(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_eq_uint(&T_check_instance, a, e);           \
}
```

Definition at line 363 of file test.h.

#### 8.123.2.13 T_flags_eq_ull

```
#define T_flags_eq_ull(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_eq_ull(&T_check_instance, a, e);            \
}
```

Definition at line 580 of file test.h.

#### 8.123.2.14 T_flags_eq_ulong

```
#define T_flags_eq_ulong(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_eq_ulong(&T_check_instance, a, e);          \
}
```

Definition at line 471 of file test.h.

#### 8.123.2.15 T_flags_ge_int

```
#define T_flags_ge_int(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_ge_int(&T_check_instance, a, e);            \
}
```

Definition at line 327 of file test.h.

#### 8.123.2.16 T_flags_ge_ll

```
#define T_flags_ge_ll(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_ge_ll(&T_check_instance, a, e);             \
}
```

Definition at line 543 of file test.h.

#### 8.123.2.17 T_flags_ge_long

```
#define T_flags_ge_long(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_ge_long(&T_check_instance, a, e);           \
}
```

Definition at line 435 of file test.h.

#### 8.123.2.18 T_flags_ge_uint

```
#define T_flags_ge_uint(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_ge_uint(&T_check_instance, a, e);           \
}
```

Definition at line 381 of file test.h.

#### 8.123.2.19 T_flags_ge_ull

```
#define T_flags_ge_ull(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_ge_ull(&T_check_instance, a, e);            \
}
```

Definition at line 600 of file test.h.

#### 8.123.2.20 T_flags_ge_ulong

```
#define T_flags_ge_ulong(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_ge_ulong(&T_check_instance, a, e);          \
}
```

Definition at line 489 of file test.h.

#### 8.123.2.21 T_flags_gt_int

```
#define T_flags_gt_int(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_gt_int(&T_check_instance, a, e);            \
}
```

Definition at line 336 of file test.h.

#### 8.123.2.22 T_flags_gt_ll

```
#define T_flags_gt_ll(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_gt_ll(&T_check_instance, a, e);             \
}
```

Definition at line 552 of file test.h.

#### 8.123.2.23 T_flags_gt_long

```
#define T_flags_gt_long(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_gt_long(&T_check_instance, a, e);           \
}
```

Definition at line 444 of file test.h.

#### 8.123.2.24 T_flags_gt_uint

```
#define T_flags_gt_uint(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_gt_uint(&T_check_instance, a, e);           \
}
```

Definition at line 390 of file test.h.

#### 8.123.2.25 T_flags_gt_ull

```
#define T_flags_gt_ull(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_gt_ull(&T_check_instance, a, e);            \
}
```

Definition at line 610 of file test.h.

#### 8.123.2.26 T_flags_gt_ulong

```
#define T_flags_gt_ulong(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_gt_ulong(&T_check_instance, a, e);          \
}
```

Definition at line 498 of file test.h.

#### 8.123.2.27 T_flags_le_int

```
#define T_flags_le_int(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_le_int(&T_check_instance, a, e);            \
}
```

Definition at line 345 of file test.h.

#### 8.123.2.28 T_flags_le_ll

```
#define T_flags_le_ll(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_le_ll(&T_check_instance, a, e);             \
}
```

Definition at line 561 of file test.h.

#### 8.123.2.29 T_flags_le_long

```
#define T_flags_le_long(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_le_long(&T_check_instance, a, e);           \
}
```

Definition at line 453 of file test.h.

#### 8.123.2.30 T_flags_le_uint

```
#define T_flags_le_uint(
              a,
              e,
              flags )
```

**Value:**

```
{                                    \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_le_uint(&T_check_instance, a, e);           \
}
```

Definition at line 399 of file test.h.

#### 8.123.2.31 T_flags_le_ull

```
#define T_flags_le_ull(
              a,
              e,
              flags )
```

**Value:**

```
{                                    \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_le_ull(&T_check_instance, a, e);            \
}
```

Definition at line 620 of file test.h.

#### 8.123.2.32 T_flags_le_ulong

```
#define T_flags_le_ulong(
              a,
              e,
              flags )
```

**Value:**

```
{                                    \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_le_ulong(&T_check_instance, a, e);          \
}
```

Definition at line 507 of file test.h.

#### 8.123.2.33 T_flags_lt_int

```
#define T_flags_lt_int(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_lt_int(&T_check_instance, a, e);            \
}
```

Definition at line 354 of file test.h.

#### 8.123.2.34 T_flags_lt_ll

```
#define T_flags_lt_ll(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_lt_ll(&T_check_instance, a, e);             \
}
```

Definition at line 570 of file test.h.

#### 8.123.2.35 T_flags_lt_long

```
#define T_flags_lt_long(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_lt_long(&T_check_instance, a, e);           \
}
```

Definition at line 462 of file test.h.

#### 8.123.2.36 T_flags_lt_uint

```
#define T_flags_lt_uint(
              a,
              e,
              flags )
```

**Value:**

```
{                                    \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_lt_uint(&T_check_instance, a, e);           \
}
```

Definition at line 408 of file test.h.

#### 8.123.2.37 T_flags_lt_ull

```
#define T_flags_lt_ull(
              a,
              e,
              flags )
```

**Value:**

```
{                                    \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_lt_ull(&T_check_instance, a, e);            \
}
```

Definition at line 630 of file test.h.

#### 8.123.2.38 T_flags_lt_ulong

```
#define T_flags_lt_ulong(
              a,
              e,
              flags )
```

**Value:**

```
{                                    \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_lt_ulong(&T_check_instance, a, e);          \
}
```

Definition at line 516 of file test.h.

#### 8.123.2.39 T_flags_ne

```
#define T_flags_ne(
               flags,
               a,
               ...  )
```

**Value:**

```
    T_flags_true(flags, \
    (a) != (T_VA_ARGS_FIRST(__VA_ARGS__)) T_VA_ARGS_MORE(__VA_ARGS__))
```

Definition at line 187 of file test.h.

#### 8.123.2.40 T_flags_ne_char

```
#define T_flags_ne_char(
               a,
               e,
               flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_ne_char(&T_check_instance, a, e);           \
}
```

Definition at line 300 of file test.h.

#### 8.123.2.41 T_flags_ne_int

```
#define T_flags_ne_int(
               a,
               e,
               flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_ne_int(&T_check_instance, a, e);            \
}
```

Definition at line 318 of file test.h.

#### 8.123.2.42 T_flags_ne_ll

```
#define T_flags_ne_ll(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {         \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };      \
    T_check_ne_ll(&T_check_instance, a, e);              \
}
```

Definition at line 534 of file test.h.

#### 8.123.2.43 T_flags_ne_long

```
#define T_flags_ne_long(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {         \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };      \
    T_check_ne_long(&T_check_instance, a, e);            \
}
```

Definition at line 426 of file test.h.

#### 8.123.2.44 T_flags_ne_mem

```
#define T_flags_ne_mem(
                a,
                e,
                n,
                flags,
                sa,
                se,
                sn )
```

**Value:**

```
{                                       \
    static const T_check_context_msg T_check_instance = {         \
        { T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT },        \
        "memcmp(" sa ", " se ", " sn ") != 0" };             \
    T_check_ne_mem(&T_check_instance, a, e, n);          \
}
```

Definition at line 243 of file test.h.

#### 8.123.2.45 T_flags_ne_nstr

```
#define T_flags_ne_nstr(
                a,
                e,
                n,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {         \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };      \
    T_check_ne_nstr(&T_check_instance, a, e, n);             \
}
```

Definition at line 282 of file test.h.

#### 8.123.2.46 T_flags_ne_ptr

```
#define T_flags_ne_ptr(
                a,
                e,
                flags,
                sa,
                se )
```

**Value:**

```
{                                       \
    static const T_check_context_msg T_check_instance = {         \
        { T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT },        \
        sa " != " se };                                  \
    T_check_ne_ptr(&T_check_instance, a, e);             \
}
```

Definition at line 203 of file test.h.

#### 8.123.2.47 T_flags_ne_str

```
#define T_flags_ne_str(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {         \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };      \
    T_check_ne_str(&T_check_instance, a, e);             \
}
```

Definition at line 262 of file test.h.

#### 8.123.2.48 T_flags_ne_uint

```
#define T_flags_ne_uint(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_ne_uint(&T_check_instance, a, e);           \
}
```

Definition at line 372 of file test.h.

#### 8.123.2.49 T_flags_ne_ull

```
#define T_flags_ne_ull(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_ne_ull(&T_check_instance, a, e);            \
}
```

Definition at line 590 of file test.h.

#### 8.123.2.50 T_flags_ne_ulong

```
#define T_flags_ne_ulong(
                a,
                e,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_ne_ulong(&T_check_instance, a, e);          \
}
```

Definition at line 480 of file test.h.

#### 8.123.2.51 T_flags_not_null

```
#define T_flags_not_null(
                a,
                flags,
                sa )
```

**Value:**

```
{                                       \
    static const T_check_context_msg T_check_instance = {         \
        { T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT }, sa }; \
    T_check_not_null(&T_check_instance, a);              \
}
```

Definition at line 222 of file test.h.

#### 8.123.2.52 T_flags_null

```
#define T_flags_null(
                a,
                flags,
                sa )
```

**Value:**

```
{                                       \
    static const T_check_context_msg T_check_instance = {         \
        { T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT }, sa }; \
    T_check_null(&T_check_instance, a);              \
}
```

Definition at line 213 of file test.h.

#### 8.123.2.53 T_flags_psx_error

```
#define T_flags_psx_error(
                a,
                eno,
                flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {         \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };      \
    T_check_psx_error(&T_check_instance, a, eno);            \
}
```

Definition at line 657 of file test.h.

#### 8.123.2.54 T_flags_psx_success

```
#define T_flags_psx_success(
              a,
              flags )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__, (flags) | T_CHECK_FMT };     \
    T_check_psx_success(&T_check_instance, a);          \
}
```

Definition at line 666 of file test.h.

#### 8.123.2.55 T_flags_true

```
#define T_flags_true(
              flags,
              ...  )
```

**Value:**

```
{                                       \
    static const T_check_context T_check_instance = {        \
        T_FILE_NAME, __LINE__,                  \
        (flags) | T_VA_ARGS_KIND(__VA_ARGS__) };            \
    T_check(&T_check_instance,                  \
        T_VA_ARGS_FIRST(__VA_ARGS__) T_VA_ARGS_MORE(__VA_ARGS__));  \
}
```

Definition at line 174 of file test.h.

#### 8.123.2.56 T_VA_ARGS_KIND

```
#define T_VA_ARGS_KIND(
              ...  )
```

**Value:**

```
    T_VA_ARGS_SELECT(__VA_ARGS__, \
    4U, 4U, 4U, 4U, 4U, 4U, 4U, 4U, 0U, throw_away)
```

Definition at line 165 of file test.h.

## 8.124 Rate-Monotonic Manager

The Rate-Monotonic Manager provides facilities to implement tasks which execute in a periodic fashion. Critically, it also gathers information about the execution of those periods and can provide important statistics to the user which can be used to analyze and tune the application.

### Classes

- struct rtems_rate_monotonic_period_statistics
  %
- struct rtems_rate_monotonic_period_status
  %

### Macros

- #define RTEMS_PERIOD_STATUS WATCHDOG_NO_TIMEOUT
  *This constant is the interval passed to the rtems_rate_monotonic_period() directive to obtain status information.*

### Enumerations

- enum rtems_rate_monotonic_period_states { RATE_MONOTONIC_INACTIVE, RATE_MONOTONIC_ACTIVE, RATE_MONOTONIC_EXPIRED }
  %

### Functions

- rtems_status_code rtems_rate_monotonic_cancel (rtems_id id)
  %
- rtems_status_code rtems_rate_monotonic_create (rtems_name name, rtems_id *id)
  %
- rtems_status_code rtems_rate_monotonic_delete (rtems_id id)
  %
- rtems_status_code rtems_rate_monotonic_ident (rtems_name name, rtems_id *id)
  *Identifies a period object by the specified object name.*
- rtems_status_code rtems_rate_monotonic_period (rtems_id id, rtems_interval length)
  %
- rtems_status_code rtems_rate_monotonic_get_statistics (rtems_id id, rtems_rate_monotonic_period_statistics *statistics)
  %
- rtems_status_code rtems_rate_monotonic_get_status (rtems_id id, rtems_rate_monotonic_period_status *status)
  %
- void rtems_rate_monotonic_report_statistics (void)
  %
- void rtems_rate_monotonic_report_statistics_with_plugin (const struct rtems_printer *printer)
  %
- void rtems_rate_monotonic_reset_all_statistics (void)
  %
- rtems_status_code rtems_rate_monotonic_reset_statistics (rtems_id id)
  %

## 8.124.1 Detailed Description

The Rate-Monotonic Manager provides facilities to implement tasks which execute in a periodic fashion. Critically, it also gathers information about the execution of those periods and can provide important statistics to the user which can be used to analyze and tune the application.

## 8.124.2 Enumeration Type Documentation

### 8.124.2.1 rtems_rate_monotonic_period_states

```
enum rtems_rate_monotonic_period_states
```

%

**Enumerator**

| | |
|---|---|
| RATE_MONOTONIC_INACTIVE | % |
| RATE_MONOTONIC_ACTIVE | % |
| RATE_MONOTONIC_EXPIRED | % |

Definition at line 172 of file ratemon.h.

## 8.124.3 Function Documentation

### 8.124.3.1 rtems_rate_monotonic_cancel()

```
rtems_status_code rtems_rate_monotonic_cancel (
            rtems_id id )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |

Definition at line 49 of file ratemoncancel.c.

### 8.124.3.2 rtems_rate_monotonic_create()

```
rtems_status_code rtems_rate_monotonic_create (
```

```
            rtems_name name,
            rtems_id * id )
```

%

**Parameters**

| | |
|---|---|
| *name* | % |
| *id* | % |

Definition at line 30 of file ratemoncreate.c.

#### 8.124.3.3 rtems_rate_monotonic_delete()

```
rtems_status_code rtems_rate_monotonic_delete (
            rtems_id id )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |

Definition at line 24 of file ratemondelete.c.

#### 8.124.3.4 rtems_rate_monotonic_get_statistics()

```
rtems_status_code rtems_rate_monotonic_get_statistics (
            rtems_id id,
            rtems_rate_monotonic_period_statistics * statistics )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *statistics* | % |

Definition at line 24 of file ratemongetstatistics.c.

#### 8.124.3.5 rtems_rate_monotonic_get_status()

```
rtems_status_code rtems_rate_monotonic_get_status (
            rtems_id id,
            rtems_rate_monotonic_period_status * status )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *status* | % |

Definition at line 25 of file ratemongetstatus.c.

### 8.124.3.6 rtems_rate_monotonic_ident()

```
rtems_status_code rtems_rate_monotonic_ident (
            rtems_name name,
            rtems_id * id )
```

Identifies a period object by the specified object name.

This directive obtains the period identifier associated with the period name specified in `name`.

If the period name is not unique, then the period identifier will match the first period with that name in the search order. However, this period identifier is not guaranteed to correspond to the desired period. The period identifier is used with other rate monotonic related directives to access the period.

The objects are searched from lowest to the highest index. Only the local node is searched.

**Parameters**

| | | |
|---|---|---|
| | *name* | is the object name to look up. |
| `out` | *id* | is the pointer to an object identifier variable. The object identifier of an object with the specified name will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The id parameter was NULL. |
| *RTEMS_INVALID_NAME* | The name parameter was 0. |
| *RTEMS_INVALID_NAME* | There was no object with the specified name on the local node. |

Definition at line 44 of file ratemonident.c.

### 8.124.3.7 rtems_rate_monotonic_period()

```
rtems_status_code rtems_rate_monotonic_period (
            rtems_id id,
            rtems_interval length )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *length* | % |

Definition at line 305 of file ratemonperiod.c.

#### 8.124.3.8 rtems_rate_monotonic_report_statistics_with_plugin()

```
void rtems_rate_monotonic_report_statistics_with_plugin (
            const struct rtems_printer * printer )
```

%

**Parameters**

| | |
|---|---|
| *printer* | % |

#### 8.124.3.9 rtems_rate_monotonic_reset_statistics()

```
rtems_status_code rtems_rate_monotonic_reset_statistics (
            rtems_id id )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |

Definition at line 24 of file ratemonresetstatistics.c.

# 8.125 Red-Black Tree Handler

Red-Black Tree Handler.

## Files

- file rbtree.h

  *Constants and Structures Associated with the Red-Black Tree Handler.*
- file rbtreeimpl.h

  *Inlined Routines Associated with Red-Black Trees.*
- file rbtreenext.c

  *_RBTree_Next() and _RBTree_Next() implementation.*
- file rbtreereplace.c

  *_RBTree_Replace_node() implementation.*

## Classes

- struct RBTree_Node

  *Red-black tree node.*

## Macros

- #define RBTREE_INITIALIZER_EMPTY(name) RB_INITIALIZER( name )

  *Initializer for an empty red-black tree with designator name.*
- #define RBTREE_DEFINE_EMPTY(name) RBTree_Control name = RBTREE_INITIALIZER_EMPTY( name )

  *Definition for an empty red-black tree with designator name.*

## Typedefs

- typedef struct RBTree_Node RBTree_Node

  *Red-black tree node.*
- typedef bool(* RBTree_Visitor) (const RBTree_Node *node, void *visitor_arg)

  *Red-black tree visitor.*

## Functions

- typedef RB_HEAD (RBTree_Control, RBTree_Node) RBTree_Control

  *Red-black tree control.*
- static __inline__ void _RBTree_Set_off_tree (RBTree_Node *the_node)

  *Sets a red-black tree node as off-tree.*
- static __inline__ bool _RBTree_Is_node_off_tree (const RBTree_Node *the_node)

  *Checks if this red-black tree node is off-tree.*
- void _RBTree_Insert_color (RBTree_Control *the_rbtree, RBTree_Node *the_node)

  *Rebalances the red-black tree after insertion of the node.*
- static __inline__ void _RBTree_Initialize_node (RBTree_Node *the_node)

  *Initializes a red-black tree node.*

- static __inline__ void _RBTree_Add_child (RBTree_Node *child, RBTree_Node *parent, RBTree_Node **link)

  *Adds a child node to a parent node.*
- static __inline__ void _RBTree_Insert_with_parent (RBTree_Control *the_rbtree, RBTree_Node *the_node, RBTree_Node *parent, RBTree_Node **link)

  *Inserts the node into the red-black tree using the specified parent node and link.*
- void _RBTree_Extract (RBTree_Control *the_rbtree, RBTree_Node *the_node)

  *Extracts (removes) the node from the red-black tree.*
- static __inline__ RBTree_Node * _RBTree_Root (const RBTree_Control *the_rbtree)

  *Returns a pointer to root node of the red-black tree.*
- static __inline__ RBTree_Node ** _RBTree_Root_reference (RBTree_Control *the_rbtree)

  *Returns a reference to the root pointer of the red-black tree.*
- static __inline__ RBTree_Node *const * _RBTree_Root_const_reference (const RBTree_Control *the_↩ rbtree)

  *Returns a constant reference to the root pointer of the red-black tree.*
- static __inline__ RBTree_Node * _RBTree_Parent (const RBTree_Node *the_node)

  *Returns a pointer to the parent of this node.*
- static __inline__ RBTree_Node * _RBTree_Left (const RBTree_Node *the_node)

  *Returns pointer to the left of this node.*
- static __inline__ RBTree_Node ** _RBTree_Left_reference (RBTree_Node *the_node)

  *Returns a reference to the left child pointer of the red-black tree node.*
- static __inline__ RBTree_Node * _RBTree_Right (const RBTree_Node *the_node)

  *Returns pointer to the right of this node.*
- static __inline__ RBTree_Node ** _RBTree_Right_reference (RBTree_Node *the_node)

  *Returns a reference to the right child pointer of the red-black tree node.*
- static __inline__ bool _RBTree_Is_empty (const RBTree_Control *the_rbtree)

  *Checks if the RBTree is empty.*
- static __inline__ bool _RBTree_Is_root (const RBTree_Node *the_node)

  *Checks if this node is the root node of a red-black tree.*
- static __inline__ void _RBTree_Initialize_empty (RBTree_Control *the_rbtree)

  *Initializes this RBTree as empty.*
- static __inline__ void _RBTree_Initialize_one (RBTree_Control *the_rbtree, RBTree_Node *the_node)

  *Initializes this red-black tree to contain exactly the specified node.*
- RBTree_Node * _RBTree_Minimum (const RBTree_Control *the_rbtree)

  *Returns the minimum node of the red-black tree.*
- RBTree_Node * _RBTree_Maximum (const RBTree_Control *the_rbtree)

  *Returns the maximum node of the red-black tree.*
- RBTree_Node * _RBTree_Predecessor (const RBTree_Node *node)

  *Returns the predecessor of a node.*
- RBTree_Node * _RBTree_Successor (const RBTree_Node *node)

  *Returns the successor of a node.*
- void _RBTree_Replace_node (RBTree_Control *the_rbtree, RBTree_Node *victim, RBTree_Node *replacement)

  *Replaces a node in the red-black tree without a rebalance.*
- static __inline__ bool _RBTree_Insert_inline (RBTree_Control *the_rbtree, RBTree_Node *the_node, const void *key, bool(*less)(const void *, const RBTree_Node *))

  *Inserts the node into the red-black tree.*
- static __inline__ void * _RBTree_Find_inline (const RBTree_Control *the_rbtree, const void *key, bool(*equal)(const void *, const RBTree_Node *), bool(*less)(const void *, const RBTree_Node *), void *(*map)(RBTree_Node *))

  *Finds an object in the red-black tree with the specified key.*
- void * _RBTree_Postorder_first (const RBTree_Control *the_rbtree, size_t offset)

*Returns the container of the first node of the specified red-black tree in postorder.*

- void * _RBTree_Postorder_next (const RBTree_Node *the_node, size_t offset)

  *Returns the container of the next node in postorder.*

- void _RBTree_Iterate (const RBTree_Control *rbtree, RBTree_Visitor visitor, void *visitor_arg)

  *Red-black tree iteration.*

## 8.125.1 Detailed Description

Red-Black Tree Handler.

The Red-Black Tree Handler is used to manage sets of entities. This handler provides two data structures. The rbtree Node data structure is included as the first part of every data structure that will be placed on a RBTree. The second data structure is rbtree Control which is used to manage a set of rbtree Nodes.

## 8.125.2 Typedef Documentation

### 8.125.2.1 RBTree_Node

```
typedef struct RBTree_Node RBTree_Node
```

Red-black tree node.

This is used to manage each node (element) which is placed on a red-black tree.

### 8.125.2.2 RBTree_Visitor

```
typedef bool(* RBTree_Visitor) (const RBTree_Node *node, void *visitor_arg)
```

Red-black tree visitor.

**Parameters**

| | | |
|---|---|---|
| in | *node* | The node. |
| in | *visitor_arg* | The visitor argument. |

**Return values**

| | |
|---|---|
| *true* | Stop the iteration. |
| *false* | Continue the iteration. |

**See also**

_RBTree_Iterate().

Definition at line 50 of file rbtreeimpl.h.

## 8.125.3 Function Documentation

### 8.125.3.1 _RBTree_Add_child()

```
static __inline__ void _RBTree_Add_child (
            RBTree_Node * child,
            RBTree_Node * parent,
            RBTree_Node ** link )  [static]
```

Adds a child node to a parent node.

**Parameters**

| | | |
|---|---|---|
| | *child* | The child node. |
| out | *parent* | The parent node. |
| out | *link* | The child node link of the parent node. |

Definition at line 145 of file rbtree.h.

### 8.125.3.2 _RBTree_Extract()

```
void _RBTree_Extract (
            RBTree_Control * the_rbtree,
            RBTree_Node * the_node )
```

Extracts (removes) the node from the red-black tree.

This function does not set the node off-tree. In the case this is desired, then call _RBTree_Set_off_tree() after the extraction.

In the case the node to extract is not a node of the tree, then this function yields unpredictable results.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_rbtree* | The red-black tree control. |
| out | *the_node* | The node to extract. |

Definition at line 35 of file rbtreeextract.c.

### 8.125.3.3 _RBTree_Find_inline()

```
static __inline__ void* _RBTree_Find_inline (
            const RBTree_Control * the_rbtree,
```

```
            const void * key,
            bool(*)(const void *, const RBTree_Node *) equal,
            bool(*)(const void *, const RBTree_Node *) less,
            void *(*)(RBTree_Node *) map )  [static]
```

Finds an object in the red-black tree with the specified key.

**Parameters**

| | |
|---|---|
| *the_rbtree* | The red-black tree control. |
| *key* | The key to look after. |
| *equal* | Must return true if the specified key equals the key of the node, otherwise false. |
| *less* | Must return true if the specified key is less than the key of the node, otherwise false. |
| *map* | In case a node with the specified key is found, then this function is called to map the node to the object returned. Usually it performs some offset operation via RTEMS_CONTAINER_OF() to map the node to its containing object. Thus, the return type is a void pointer and not a red-black tree node. |

**Return values**

| | |
|---|---|
| *object* | An object with the specified key. |
| *NULL* | No object with the specified key exists in the red-black tree. |

Definition at line 557 of file rbtree.h.

#### 8.125.3.4 _RBTree_Initialize_empty()

```
static __inline__ void _RBTree_Initialize_empty (
            RBTree_Control * the_rbtree )  [static]
```

Initializes this RBTree as empty.

This routine initializes *the_rbtree* to contain zero nodes.

**Parameters**

| | | |
|---|---|---|
| out | *the_rbtree* | The rbtree to initialize. |

Definition at line 410 of file rbtree.h.

#### 8.125.3.5 _RBTree_Initialize_node()

```
static __inline__ void _RBTree_Initialize_node (
            RBTree_Node * the_node )  [static]
```

Initializes a red-black tree node.

In debug configurations, the node is set off tree. In all other configurations, this function does nothing.

**Parameters**

| | | |
|---|---|---|
| out | *the_node* | The red-black tree node to initialize. |

Definition at line 129 of file rbtree.h.

#### 8.125.3.6 _RBTree_Initialize_one()

```
static __inline__ void _RBTree_Initialize_one (
            RBTree_Control * the_rbtree,
            RBTree_Node * the_node )  [static]
```

Initializes this red-black tree to contain exactly the specified node.

**Parameters**

| | | |
|---|---|---|
| out | *the_rbtree* | The red-black tree control. |
| out | *the_node* | The one and only node. |

Definition at line 424 of file rbtree.h.

#### 8.125.3.7 _RBTree_Insert_color()

```
void _RBTree_Insert_color (
            RBTree_Control * the_rbtree,
            RBTree_Node * the_node )
```

Rebalances the red-black tree after insertion of the node.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_rbtree* | The red-black tree control. |
| in,out | *the_node* | The most recently inserted node. |

Definition at line 17 of file rbtreeinsert.c.

#### 8.125.3.8 _RBTree_Insert_inline()

```
static __inline__ bool _RBTree_Insert_inline (
            RBTree_Control * the_rbtree,
            RBTree_Node * the_node,
            const void * key,
            bool(*)(const void *, const RBTree_Node *) less )  [static]
```

Inserts the node into the red-black tree.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_rbtree* | The red-black tree control. |
| out | *the_node* | The node to insert. |
| | *key* | The key of the node to insert. This key must be equal to the key stored in the node to insert. The separate key parameter is provided for two reasons. Firstly, it allows to share the less operator with _RBTree_Find_inline(). Secondly, the compiler may generate better code if the key is stored in a local variable. |
| | *less* | Must return true if the specified key is less than the key of the node, otherwise false. |

**Return values**

| | |
|---|---|
| *true* | The inserted node is the new minimum node according to the specified less order function. |
| *false* | The inserted node is not the new minimum node according to the specified less order function. |

Definition at line 508 of file rbtree.h.

### 8.125.3.9 _RBTree_Insert_with_parent()

```
static __inline__ void _RBTree_Insert_with_parent (
              RBTree_Control * the_rbtree,
              RBTree_Node * the_node,
              RBTree_Node * parent,
              RBTree_Node ** link )  [static]
```

Inserts the node into the red-black tree using the specified parent node and link.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_rbtree* | The red-black tree control. |
| in,out | *the_node* | The node to insert. |
| out | *parent* | The parent node. |
| out | *link* | The child node link of the parent node. |

```
#include <rtems/score/rbtree.h>
typedef struct {
  int value;
  RBTree_Node Node;
} Some_Node;
bool _Some_Less(
  const RBTree_Node *a,
  const RBTree_Node *b
)
{
  const Some_Node *aa = RTEMS_CONTAINER_OF( a, Some_Node, Node );
  const Some_Node *bb = RTEMS_CONTAINER_OF( b, Some_Node, Node );
  return aa->value < bb->value;
}
void _Some_Insert(
  RBTree_Control *the_rbtree,
  Some_Node      *the_node
)
{
  RBTree_Node **link = _RBTree_Root_reference( the_rbtree );
  RBTree_Node  *parent = NULL;
  while ( *link != NULL ) {
    parent = *link;
    if ( _Some_Less( &the_node->Node, parent ) ) {
```

```
      link = _RBTree_Left_reference( parent );
    } else {
      link = _RBTree_Right_reference( parent );
    }
  }
  _RBTree_Insert_with_parent( the_rbtree, &the_node->Node, parent, link );
}
```

Definition at line 206 of file rbtree.h.

#### 8.125.3.10 _RBTree_Is_empty()

```
static __inline__ bool _RBTree_Is_empty (
             const RBTree_Control * the_rbtree )  [static]
```

Checks if the RBTree is empty.

This function returns true if there are no nodes on *the_rbtree* and false otherwise.

**Parameters**

| | |
|---|---|
| *the_rbtree* | is the rbtree to be operated upon. |

**Return values**

| | |
|---|---|
| *true* | There are no nodes on *the_rbtree*. |
| *false* | There are nodes on *the_rbtree*. |

Definition at line 375 of file rbtree.h.

#### 8.125.3.11 _RBTree_Is_node_off_tree()

```
static __inline__ bool _RBTree_Is_node_off_tree (
             const RBTree_Node * the_node )  [static]
```

Checks if this red-black tree node is off-tree.

**Parameters**

| | |
|---|---|
| *the_node* | The node to test. |

**Return values**

| | |
|---|---|
| *true* | The node is not a part of a tree (off-tree). |
| *false* | The node is part of a tree. |

**See also**

_RBTree_Set_off_tree().

Definition at line 103 of file rbtree.h.

### 8.125.3.12 _RBTree_Is_root()

```
static __inline__ bool _RBTree_Is_root (
             const RBTree_Node * the_node )  [static]
```

Checks if this node is the root node of a red-black tree.

The root node may change after insert or extract operations. In case the node is not a node of a tree, then this function yields unpredictable results.

**Parameters**

| | |
|---|---|
| *the_node* | The node of interest. |

**Return values**

| | |
|---|---|
| *true* | *the_node* is the root node. |
| *false* | *the_node* is not the root node. |

**See also**

_RBTree_Root().

Definition at line 396 of file rbtree.h.

### 8.125.3.13 _RBTree_Iterate()

```
void _RBTree_Iterate (
             const RBTree_Control * rbtree,
             RBTree_Visitor visitor,
             void * visitor_arg )
```

Red-black tree iteration.

**Parameters**

| | |
|---|---|
| *rbtree* | The red-black tree. |
| *visitor* | The visitor. |
| *visitor_arg* | The visitor argument. |

#### 8.125.3.14 _RBTree_Left()

```
static __inline__ RBTree_Node* _RBTree_Left (
            const RBTree_Node * the_node )  [static]
```

Returns pointer to the left of this node.

This function returns a pointer to the left node of this node.

**Parameters**

| | |
|---|---|
| *the_node* | is the node to be operated upon. |

**Returns**

This method returns the left node on the rbtree.

Definition at line 311 of file rbtree.h.

#### 8.125.3.15 _RBTree_Left_reference()

```
static __inline__ RBTree_Node** _RBTree_Left_reference (
            RBTree_Node * the_node )  [static]
```

Returns a reference to the left child pointer of the red-black tree node.

**Parameters**

| | |
|---|---|
| *the_node* | is the node to be operated upon. |

**Returns**

This method returns a reference to the left child pointer on the rbtree.

Definition at line 326 of file rbtree.h.

#### 8.125.3.16 _RBTree_Maximum()

```
RBTree_Node* _RBTree_Maximum (
            const RBTree_Control * the_rbtree )
```

Returns the maximum node of the red-black tree.

**Parameters**

| | |
|---|---|
| *the_rbtree* | The red-black tree control. |

**Return values**

| | |
|---|---|
| *node* | The maximum node. |
| *NULL* | The red-black tree is empty. |

Definition at line 41 of file rbtreenext.c.

### 8.125.3.17 _RBTree_Minimum()

```
RBTree_Node* _RBTree_Minimum (
             const RBTree_Control * the_rbtree )
```

Returns the minimum node of the red-black tree.

**Parameters**

| | |
|---|---|
| *the_rbtree* | The red-black tree control. |

**Return values**

| | |
|---|---|
| *node* | The minimum node. |
| *NULL* | The red-black tree is empty. |

Definition at line 36 of file rbtreenext.c.

### 8.125.3.18 _RBTree_Parent()

```
static __inline__ RBTree_Node* _RBTree_Parent (
             const RBTree_Node * the_node )  [static]
```

Returns a pointer to the parent of this node.

The node must have a parent, thus it is invalid to use this function for the root node or a node that is not part of a tree. To test for the root node compare with _RBTree_Root() or use _RBTree_Is_root().

**Parameters**

| | |
|---|---|
| *the_node* | The node of interest. |

**Return values**

| | |
|---|---|
| *parent* | The parent of this node. |
| *undefined* | The node is the root node or not part of a tree. |

Definition at line 295 of file rbtree.h.

### 8.125.3.19 _RBTree_Postorder_first()

```
void* _RBTree_Postorder_first (
            const RBTree_Control * the_rbtree,
            size_t offset )
```

Returns the container of the first node of the specified red-black tree in postorder.

Postorder traversal may be used to delete all nodes of a red-black tree.

**Parameters**

| | |
|---|---|
| *the_rbtree* | The red-black tree control. |
| *offset* | The offset to the red-black tree node in the container structure. |

**Return values**

| | |
|---|---|
| *container* | The container of the first node of the specified red-black tree in postorder. |
| *NULL* | The red-black tree is empty. |

**See also**

_RBTree_Postorder_next().

```
#include <rtems/score/rbtree.h>
typedef struct {
  int         data;
  RBTree_Node Node;
} Container_Control;
void visit( Container_Control *the_container );
void postorder_traversal( RBTree_Control *the_rbtree )
{
  Container_Control *the_container;
  the_container = _RBTree_Postorder_first(
    the_rbtree,
    offsetof( Container_Control, Node )
  );
  while ( the_container != NULL ) {
    visit( the_container );
    the_container = _RBTree_Postorder_next(
      &the_container->Node,
      offsetof( Container_Control, Node )
    );
  }
}
```

### 8.125.3.20 _RBTree_Postorder_next()

```
void* _RBTree_Postorder_next (
              const RBTree_Node * the_node,
              size_t offset )
```

Returns the container of the next node in postorder.

**Parameters**

| | |
|---|---|
| *the_node* | The red-black tree node. The node must not be NULL. |
| *offset* | The offset to the red-black tree node in the container structure. |

**Return values**

| | |
|---|---|
| *container* | The container of the next node in postorder. |
| *NULL* | The node is NULL or there is no next node in postorder. |

**See also**

_RBTree_Postorder_first().

### 8.125.3.21 _RBTree_Predecessor()

```
RBTree_Node* _RBTree_Predecessor (
              const RBTree_Node * node )
```

Returns the predecessor of a node.

**Parameters**

| | |
|---|---|
| *node* | is the node. |

**Return values**

| | |
|---|---|
| *node* | The predecessor node. |
| *NULL* | The predecessor does not exist. |

Definition at line 51 of file rbtreenext.c.

### 8.125.3.22 _RBTree_Replace_node()

```
void _RBTree_Replace_node (
              RBTree_Control * the_rbtree,
```

```
            RBTree_Node * victim,
            RBTree_Node * replacement )
```

Replaces a node in the red-black tree without a rebalance.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_rbtree* | The red-black tree control. |
| | *victim* | The victim node. |
| out | *replacement* | The replacement node. |

Definition at line 29 of file rbtreereplace.c.

### 8.125.3.23 _RBTree_Right()

```
static __inline__ RBTree_Node* _RBTree_Right (
            const RBTree_Node * the_node )  [static]
```

Returns pointer to the right of this node.

This function returns a pointer to the right node of this node.

**Parameters**

| | |
|---|---|
| *the_node* | is the node to be operated upon. |

**Returns**

This method returns the right node on the rbtree.

Definition at line 342 of file rbtree.h.

### 8.125.3.24 _RBTree_Right_reference()

```
static __inline__ RBTree_Node** _RBTree_Right_reference (
            RBTree_Node * the_node )  [static]
```

Returns a reference to the right child pointer of the red-black tree node.

**Parameters**

| | |
|---|---|
| *the_node* | is the node to be operated upon. |

**Returns**

This method returns a reference to the right child pointer on the rbtree.

Definition at line 357 of file rbtree.h.

### 8.125.3.25 _RBTree_Root()

```
static __inline__ RBTree_Node* _RBTree_Root (
             const RBTree_Control * the_rbtree )  [static]
```

Returns a pointer to root node of the red-black tree.

The root node may change after insert or extract operations.

**Parameters**

| | |
|---|---|
| *the_rbtree* | The red-black tree control. |

**Return values**

| | |
|---|---|
| *root* | The root node. |
| *NULL* | The tree is empty. |

**See also**

_RBTree_Is_root().

Definition at line 246 of file rbtree.h.

### 8.125.3.26 _RBTree_Root_const_reference()

```
static __inline__ RBTree_Node* const* _RBTree_Root_const_reference (
             const RBTree_Control * the_rbtree )  [static]
```

Returns a constant reference to the root pointer of the red-black tree.

**Parameters**

| | |
|---|---|
| *the_rbtree* | The red-black tree control. |

**Return values**

| | |
|---|---|
| *pointer* | Pointer to the root node. |
| *NULL* | The tree is empty. |

Definition at line 276 of file rbtree.h.

#### 8.125.3.27 _RBTree_Root_reference()

```
static __inline__ RBTree_Node** _RBTree_Root_reference (
            RBTree_Control * the_rbtree )  [static]
```

Returns a reference to the root pointer of the red-black tree.

**Parameters**

| | |
|---|---|
| *the_rbtree* | The red-black tree control. |

**Return values**

| | |
|---|---|
| *pointer* | Pointer to the root node. |
| *NULL* | The tree is empty. |

Definition at line 261 of file rbtree.h.

#### 8.125.3.28 _RBTree_Set_off_tree()

```
static __inline__ void _RBTree_Set_off_tree (
            RBTree_Node * the_node )  [static]
```

Sets a red-black tree node as off-tree.

Do not use this function on nodes which are a part of a tree.

**Parameters**

| | | |
|---|---|---|
| out | *the_node* | The node to set off-tree. |

**See also**

_RBTree_Is_node_off_tree().

Definition at line 88 of file rbtree.h.

#### 8.125.3.29 _RBTree_Successor()

```
RBTree_Node* _RBTree_Successor (
            const RBTree_Node * node )
```

Returns the successor of a node.

**Parameters**

| | |
|---|---|
| *node* | is the node. |

**Return values**

| | |
|---|---|
| *node* | The successor node. |
| *NULL* | The successor does not exist. |

Definition at line 46 of file rbtreenext.c.

#### 8.125.3.30 RB_HEAD()

```
typedef RB_HEAD (
            RBTree_Control ,
            RBTree_Node  )
```

Red-black tree control.

This is used to manage a red-black tree. A red-black tree consists of a tree of zero or more nodes.

## 8.126 Region Manager

The Region Manager provides facilities to dynamically allocate memory in variable sized units.

### Functions

- rtems_status_code rtems_region_create (rtems_name name, void *starting_address, uintptr_t length, uintptr_t page_size, rtems_attribute attribute_set, rtems_id *id)
  %
- rtems_status_code rtems_region_delete (rtems_id id)
  %
- rtems_status_code rtems_region_extend (rtems_id id, void *starting_address, uintptr_t length)
  %
- rtems_status_code rtems_region_get_free_information (rtems_id id, Heap_Information_block *the_info)
  %
- rtems_status_code rtems_region_get_information (rtems_id id, Heap_Information_block *the_info)
  %
- rtems_status_code rtems_region_get_segment (rtems_id id, uintptr_t size, rtems_option option_set, rtems_interval timeout, void **segment)
  %
- rtems_status_code rtems_region_get_segment_size (rtems_id id, void *segment, uintptr_t *size)
  %
- rtems_status_code rtems_region_ident (rtems_name name, rtems_id *id)
  *Identifies a region object by the specified object name.*
- rtems_status_code rtems_region_resize_segment (rtems_id id, void *segment, uintptr_t size, uintptr_t *old_size)
  %
- rtems_status_code rtems_region_return_segment (rtems_id id, void *segment)
  %

### 8.126.1 Detailed Description

The Region Manager provides facilities to dynamically allocate memory in variable sized units.

### 8.126.2 Function Documentation

#### 8.126.2.1 rtems_region_create()

```
rtems_status_code rtems_region_create (
            rtems_name name,
            void * starting_address,
            uintptr_t length,
            uintptr_t page_size,
            rtems_attribute attribute_set,
            rtems_id * id )
```

%

**Parameters**

| | |
|---|---|
| *name* | % |
| *starting_address* | % |
| *length* | % |
| *page_size* | % |
| *attribute_set* | % |
| *id* | % |

### 8.126.2.2 rtems_region_delete()

```
rtems_status_code rtems_region_delete (
            rtems_id id )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |

### 8.126.2.3 rtems_region_extend()

```
rtems_status_code rtems_region_extend (
            rtems_id id,
            void * starting_address,
            uintptr_t length )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *starting_address* | % |
| *length* | % |

### 8.126.2.4 rtems_region_get_free_information()

```
rtems_status_code rtems_region_get_free_information (
            rtems_id id,
            Heap_Information_block * the_info )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *the_info* | % |

#### 8.126.2.5 rtems_region_get_information()

```
rtems_status_code rtems_region_get_information (
            rtems_id id,
            Heap_Information_block * the_info )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *the_info* | % |

#### 8.126.2.6 rtems_region_get_segment()

```
rtems_status_code rtems_region_get_segment (
            rtems_id id,
            uintptr_t size,
            rtems_option option_set,
            rtems_interval timeout,
            void ** segment )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *size* | % |
| *option_set* | % |
| *timeout* | % |
| *segment* | % |

#### 8.126.2.7 rtems_region_get_segment_size()

```
rtems_status_code rtems_region_get_segment_size (
            rtems_id id,
```

```
            void * segment,
            uintptr_t * size )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *segment* | % |
| *size* | % |

### 8.126.2.8 rtems_region_ident()

```
rtems_status_code rtems_region_ident (
            rtems_name name,
            rtems_id * id )
```

Identifies a region object by the specified object name.

This directive obtains the region identifier associated with the region name specified in `name`.

If the region name is not unique, then the region identifier will match the first region with that name in the search order. However, this region identifier is not guaranteed to correspond to the desired region. The region identifier is used with other region related directives to access the region.

The objects are searched from lowest to the highest index. Only the local node is searched.

**Parameters**

| | | |
|---|---|---|
| | *name* | is the object name to look up. |
| `out` | *id* | is the pointer to an object identifier variable. The object identifier of an object with the specified name will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The id parameter was NULL. |
| *RTEMS_INVALID_NAME* | The name parameter was 0. |
| *RTEMS_INVALID_NAME* | There was no object with the specified name on the local node. |

### 8.126.2.9 rtems_region_resize_segment()

```
rtems_status_code rtems_region_resize_segment (
            rtems_id id,
            void * segment,
            uintptr_t size,
            uintptr_t * old_size )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *segment* | % |
| *size* | % |
| *old_size* | % |

### 8.126.2.10 rtems_region_return_segment()

```
rtems_status_code rtems_region_return_segment (
            rtems_id id,
            void * segment )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *segment* | % |

## 8.127 Runtime Measurements

Support to measure the runtime of code fragments.

### Classes

- struct T_measure_runtime_config
- struct T_measure_runtime_request

### Macros

- #define **T_MEASURE_RUNTIME_ALLOW_CLOCK_ISR** 0x1
- #define **T_MEASURE_RUNTIME_REPORT_SAMPLES** 0x2
- #define **T_MEASURE_RUNTIME_DISABLE_VALID_CACHE** 0x10
- #define **T_MEASURE_RUNTIME_DISABLE_HOT_CACHE** 0x20
- #define **T_MEASURE_RUNTIME_DISABLE_DIRTY_CACHE** 0x40
- #define **T_MEASURE_RUNTIME_DISABLE_MINOR_LOAD** 0x80
- #define **T_MEASURE_RUNTIME_DISABLE_MAX_LOAD** 0x100

### Typedefs

- typedef struct T_measure_runtime_context **T_measure_runtime_context**

### Functions

- T_measure_runtime_context * **T_measure_runtime_create** (const T_measure_runtime_config *)
- void **T_measure_runtime** (T_measure_runtime_context *, const T_measure_runtime_request *)

### 8.127.1 Detailed Description

Support to measure the runtime of code fragments.

# 8.128 SMP Barriers

The SMP barrier provides barrier synchronization for SMP systems at the lowest level.

## Files

- file smpbarrier.h

  *SMP Barrier API.*

## Classes

- struct SMP_barrier_Control

  *SMP barrier control.*
- struct SMP_barrier_State

  *SMP barrier per-thread state.*

## Macros

- #define SMP_BARRIER_CONTROL_INITIALIZER { ATOMIC_INITIALIZER_UINT( 0U ), ATOMIC_INITI↩ ALIZER_UINT( 0U ) }

  *SMP barrier control initializer for static initialization.*
- #define SMP_BARRIER_STATE_INITIALIZER { 0U }

  *SMP barrier per-thread state initializer for static initialization.*

## Functions

- static void _SMP_barrier_Control_initialize (SMP_barrier_Control *control)

  *Initializes a SMP barrier control.*
- static void _SMP_barrier_State_initialize (SMP_barrier_State *state)

  *Initializes a SMP barrier per-thread state.*
- bool _SMP_barrier_Wait (SMP_barrier_Control *control, SMP_barrier_State *state, unsigned int count)

  *Waits on the SMP barrier until count threads rendezvoused.*

## 8.128.1 Detailed Description

The SMP barrier provides barrier synchronization for SMP systems at the lowest level.

The SMP barrier is implemented as a sense barrier, see also Herlihy and Shavit, "The Art of Multiprocessor Programming", 17.3 Sense-Reversing Barrier.

## 8.128.2 Function Documentation

### 8.128.2.1 _SMP_barrier_Control_initialize()

```
static void _SMP_barrier_Control_initialize (
            SMP_barrier_Control * control )  [inline], [static]
```

Initializes a SMP barrier control.

Concurrent initialization leads to unpredictable results.

**Parameters**

| | | |
|---|---|---|
| out | *control* | The SMP barrier control. |

Definition at line 83 of file smpbarrier.h.

#### 8.128.2.2 _SMP_barrier_State_initialize()

```
static void _SMP_barrier_State_initialize (
            SMP_barrier_State * state )  [inline], [static]
```

Initializes a SMP barrier per-thread state.

**Parameters**

| | | |
|---|---|---|
| out | *state* | The SMP barrier control. |

Definition at line 96 of file smpbarrier.h.

#### 8.128.2.3 _SMP_barrier_Wait()

```
bool _SMP_barrier_Wait (
            SMP_barrier_Control * control,
            SMP_barrier_State * state,
            unsigned int count )
```

Waits on the SMP barrier until count threads rendezvoused.

**Parameters**

| | | |
|---|---|---|
| in,out | *control* | The SMP barrier control. |
| in,out | *state* | The SMP barrier per-thread state. |
| | *count* | The thread count bound to rendezvous. |

**Return values**

| | |
|---|---|
| *true* | This processor performed the barrier release. |
| *false* | This processor did not performe the barrier release. |

Definition at line 21 of file smpbarrierwait.c.

# 8.129 SMP Locks

The SMP lock provides mutual exclusion for SMP systems at the lowest level.

## Files

- file smplock.h

  *SMP Lock API.*
- file smplockmcs.h

  *SMP Lock API.*
- file smplockseq.h

  *SMP Lock API.*
- file smplockstats.h

  *SMP Lock API.*
- file smplockticket.h

  *SMP Lock API.*

## Classes

- struct SMP_lock_Control

  *SMP lock control.*
- struct SMP_lock_Context

  *Local SMP lock context for acquire and release pairs.*
- struct SMP_MCS_lock_Context

  *SMP Mellor-Crummey and Scott (MCS) lock context.*
- struct SMP_MCS_lock_Control

  *SMP Mellor-Crummey and Scott (MCS) lock control.*
- struct SMP_sequence_lock_Control

  *SMP sequence lock control.*
- struct SMP_ticket_lock_Control

  *SMP ticket lock control.*

## Macros

- #define SMP_LOCK_INITIALIZER(name) { SMP_TICKET_LOCK_INITIALIZER }

  *SMP lock control initializer for static initialization.*
- #define _SMP_lock_Initialize(lock, name) _SMP_lock_Initialize_inline( lock, name )

  *Initializes an SMP lock.*
- #define _SMP_lock_Destroy(lock) _SMP_lock_Destroy_inline( lock )

  *Destroys an SMP lock.*
- #define _SMP_lock_Release(lock, context) _SMP_lock_Release_inline( lock, context )

  *Releases an SMP lock.*
- #define _SMP_lock_Release_and_ISR_enable(lock, context) _SMP_lock_Release_and_ISR_enable_inline( lock, context )

  *Releases the SMP lock and enables interrupts.*
- #define SMP_MCS_LOCK_INITIALIZER { { ATOMIC_INITIALIZER_UINTPTR( 0 ) } }

  *SMP MCS lock control initializer for static initialization.*
- #define _SMP_MCS_lock_Acquire(lock, context, stats) _SMP_MCS_lock_Do_acquire( lock, context )

*Acquires an SMP MCS lock.*

- #define SMP_SEQUENCE_LOCK_INITIALIZER { ATOMIC_INITIALIZER_UINT( 0 ) }

  *SMP sequence lock control initializer for static initialization.*
- #define **_SMP_lock_Stats_initialize**(stats, name) do { } while ( 0 )
- #define **_SMP_lock_Stats_destroy**(stats) do { } while ( 0 )
- #define SMP_TICKET_LOCK_INITIALIZER

  *SMP ticket lock control initializer for static initialization.*
- #define _SMP_ticket_lock_Acquire(lock, stats, stats_context) _SMP_ticket_lock_Do_acquire( lock )

  *Acquires an SMP ticket lock.*
- #define _SMP_ticket_lock_Release(lock, stats_context) _SMP_ticket_lock_Do_release( lock )

  *Releases an SMP ticket lock.*

## Typedefs

- typedef struct SMP_MCS_lock_Context SMP_MCS_lock_Context

  *SMP Mellor-Crummey and Scott (MCS) lock context.*

## Functions

- static void _SMP_lock_Initialize_inline (SMP_lock_Control *lock, const char *name)

  *Initializes the SMP lock with the given name.*
- static void _SMP_lock_Destroy_inline (SMP_lock_Control *lock)

  *Destroys the SMP lock.*
- static void _SMP_lock_Set_name (SMP_lock_Control *lock, const char *name)

  *Sets the name of an SMP lock.*
- static void _SMP_lock_Acquire_inline (SMP_lock_Control *lock, SMP_lock_Context *context)

  *Gets my index.*
- void _SMP_lock_Acquire (SMP_lock_Control *lock, SMP_lock_Context *context)

  *Acquires an SMP lock.*
- static void _SMP_lock_Release_inline (SMP_lock_Control *lock, SMP_lock_Context *context)

  *Releases an SMP lock.*
- static void _SMP_lock_ISR_disable_and_acquire_inline (SMP_lock_Control *lock, SMP_lock_Context *context)

  *Disables interrupts and acquires the SMP lock.*
- void _SMP_lock_ISR_disable_and_acquire (SMP_lock_Control *lock, SMP_lock_Context *context)

  *Disables interrupts and acquires the SMP lock.*
- static void _SMP_lock_Release_and_ISR_enable_inline (SMP_lock_Control *lock, SMP_lock_Context *context)

  *Releases the SMP lock and enables interrupts.*
- static void _SMP_MCS_lock_Initialize (SMP_MCS_lock_Control *lock)

  *Initializes the SMP MCS lock.*
- static void _SMP_MCS_lock_Destroy (SMP_MCS_lock_Control *lock)

  *Destroys the SMP MCS lock.*
- static void _SMP_MCS_lock_Do_acquire (SMP_MCS_lock_Control *lock, SMP_MCS_lock_Context *context)

  *Acquires the SMP MCS lock.*
- static void _SMP_MCS_lock_Release (SMP_MCS_lock_Control *lock, SMP_MCS_lock_Context *context)

  *Releases an SMP MCS lock.*
- static void _SMP_sequence_lock_Initialize (SMP_sequence_lock_Control *lock)

  *Initializes an SMP sequence lock.*

- static void _SMP_sequence_lock_Destroy (SMP_sequence_lock_Control *lock)

  *Destroys an SMP sequence lock.*
- static unsigned int _SMP_sequence_lock_Write_begin (SMP_sequence_lock_Control *lock)

  *Begins an SMP sequence lock write operation.*
- static void _SMP_sequence_lock_Write_end (SMP_sequence_lock_Control *lock, unsigned int seq)

  *Ends an SMP sequence lock write operation.*
- static unsigned int _SMP_sequence_lock_Read_begin (const SMP_sequence_lock_Control *lock)

  *Begins an SMP sequence lock read operation.*
- static bool _SMP_sequence_lock_Read_retry (SMP_sequence_lock_Control *lock, unsigned int seq)

  *Ends an SMP sequence lock read operation and indicates if a retry is necessary.*
- static void _SMP_ticket_lock_Initialize (SMP_ticket_lock_Control *lock)

  *Initializes the SMP ticket lock.*
- static void _SMP_ticket_lock_Destroy (SMP_ticket_lock_Control *lock)

  *Destroys the SMP ticket lock.*
- static void _SMP_ticket_lock_Do_acquire (SMP_ticket_lock_Control *lock)

  *Acquires the SMP ticket lock.*
- static void _SMP_ticket_lock_Do_release (SMP_ticket_lock_Control *lock)

  *Releases the SMP ticket lock.*

## 8.129.1 Detailed Description

The SMP lock provides mutual exclusion for SMP systems at the lowest level.

The SMP lock is implemented as a ticket lock. This provides fairness in case of concurrent lock attempts.

This SMP lock API uses a local context for acquire and release pairs. Such a context may be used to implement for example the Mellor-Crummey and Scott (MCS) locks in the future.

## 8.129.2 Macro Definition Documentation

### 8.129.2.1 _SMP_lock_Destroy

```
#define _SMP_lock_Destroy(
              lock )  _SMP_lock_Destroy_inline( lock )
```

Destroys an SMP lock.

Concurrent destruction leads to unpredictable results.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The SMP lock control. |

Definition at line 185 of file smplock.h.

#### 8.129.2.2 _SMP_lock_Initialize

```
#define _SMP_lock_Initialize(
              lock,
              name )  _SMP_lock_Initialize_inline( lock, name )
```

Initializes an SMP lock.

Concurrent initialization leads to unpredictable results.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The SMP lock control. |
| | *name* | The name for the SMP lock statistics. This name must be persistent throughout the life time of this statistics block. |

Definition at line 160 of file smplock.h.

#### 8.129.2.3 _SMP_lock_Release

```
#define _SMP_lock_Release(
              lock,
              context )  _SMP_lock_Release_inline( lock, context )
```

Releases an SMP lock.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The SMP lock control. |
| in,out | *context* | The local SMP lock context for an acquire and release pair. |

Definition at line 305 of file smplock.h.

#### 8.129.2.4 _SMP_lock_Release_and_ISR_enable

```
#define _SMP_lock_Release_and_ISR_enable(
              lock,
              context )  _SMP_lock_Release_and_ISR_enable_inline( lock, context )
```

Releases the SMP lock and enables interrupts.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The SMP lock control. |
| in,out | *context* | The local SMP lock context for an acquire and release pair. |

Definition at line 364 of file smplock.h.

#### 8.129.2.5 _SMP_MCS_lock_Acquire

```
#define _SMP_MCS_lock_Acquire(
              lock,
              context,
              stats )  _SMP_MCS_lock_Do_acquire( lock, context )
```

Acquires an SMP MCS lock.

This function will not disable interrupts. The caller must ensure that the current thread of execution is not interrupted indefinite once it obtained the SMP MCS lock.

**Parameters**

| | |
|---|---|
| *lock* | The SMP MCS lock control. |
| *context* | The SMP MCS lock context. |
| *stats* | The SMP lock statistics. |

Definition at line 202 of file smplockmcs.h.

#### 8.129.2.6 _SMP_ticket_lock_Acquire

```
#define _SMP_ticket_lock_Acquire(
              lock,
              stats,
              stats_context )  _SMP_ticket_lock_Do_acquire( lock )
```

Acquires an SMP ticket lock.

This function will not disable interrupts. The caller must ensure that the current thread of execution is not interrupted indefinite once it obtained the SMP ticket lock.

**Parameters**

| | | |
|---|---|---|
| in | *lock* | The SMP ticket lock control. |
| in | *stats* | The SMP lock statistics. |
| out | *stats_context* | The SMP lock statistics context. |

Definition at line 149 of file smplockticket.h.

#### 8.129.2.7 _SMP_ticket_lock_Release

```
#define _SMP_ticket_lock_Release(
```

```
            lock,
            stats_context )  _SMP_ticket_lock_Do_release( lock )
```

Releases an SMP ticket lock.

**Parameters**

| | | |
|---|---|---|
| in | *lock* | The SMP ticket lock control. |
| in | *stats_context* | The SMP lock statistics context. |

Definition at line 188 of file smplockticket.h.

#### 8.129.2.8 SMP_TICKET_LOCK_INITIALIZER

```
#define SMP_TICKET_LOCK_INITIALIZER
```

**Value:**

```
{ \
  ATOMIC_INITIALIZER_UINT( 0U ), \
  ATOMIC_INITIALIZER_UINT( 0U ) \
}
```

SMP ticket lock control initializer for static initialization.

Definition at line 48 of file smplockticket.h.

### 8.129.3 Function Documentation

#### 8.129.3.1 _SMP_lock_Acquire()

```
void _SMP_lock_Acquire (
            SMP_lock_Control * lock,
            SMP_lock_Context * context )
```

Acquires an SMP lock.

This function will not disable interrupts. The caller must ensure that the current thread of execution is not interrupted indefinite once it obtained the SMP lock.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The SMP lock control. |
| in,out | *context* | The local SMP lock context for an acquire and release pair. |

Definition at line 36 of file smplock.c.

### 8.129.3.2 _SMP_lock_Acquire_inline()

```
static void _SMP_lock_Acquire_inline (
             SMP_lock_Control * lock,
             SMP_lock_Context * context )  [inline], [static]
```

Gets my index.

**Returns**

The current processor index + 1.

Acquires the lock inline.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The lock to acquire. |
| in,out | *context* | The lock context. |

Definition at line 231 of file smplock.h.

### 8.129.3.3 _SMP_lock_Destroy_inline()

```
static void _SMP_lock_Destroy_inline (
             SMP_lock_Control * lock )  [inline], [static]
```

Destroys the SMP lock.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The lock to destroy. |

Definition at line 169 of file smplock.h.

### 8.129.3.4 _SMP_lock_Initialize_inline()

```
static void _SMP_lock_Initialize_inline (
             SMP_lock_Control * lock,
             const char * name )  [inline], [static]
```

Initializes the SMP lock with the given name.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The lock to initialize. |

Definition at line 129 of file smplock.h.

#### 8.129.3.5 _SMP_lock_ISR_disable_and_acquire()

```
void _SMP_lock_ISR_disable_and_acquire (
             SMP_lock_Control * lock,
             SMP_lock_Context * context )
```

Disables interrupts and acquires the SMP lock.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The SMP lock control. |
| in,out | *context* | The local SMP lock context for an acquire and release pair. |

Definition at line 54 of file smplock.c.

#### 8.129.3.6 _SMP_lock_ISR_disable_and_acquire_inline()

```
static void _SMP_lock_ISR_disable_and_acquire_inline (
             SMP_lock_Control * lock,
             SMP_lock_Context * context )  [inline], [static]
```

Disables interrupts and acquires the SMP lock.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The lock to acquire. |
| in,out | *context* | The lock context. |

Definition at line 315 of file smplock.h.

#### 8.129.3.7 _SMP_lock_Release_and_ISR_enable_inline()

```
static void _SMP_lock_Release_and_ISR_enable_inline (
             SMP_lock_Control * lock,
             SMP_lock_Context * context )  [inline], [static]
```

Releases the SMP lock and enables interrupts.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The SMP lock to release. |
| in,out | *context* | The lock context. |

Definition at line 342 of file smplock.h.

#### 8.129.3.8 _SMP_lock_Release_inline()

```
static void _SMP_lock_Release_inline (
              SMP_lock_Control * lock,
              SMP_lock_Context * context )  [inline], [static]
```

Releases an SMP lock.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The lock to release. |
| in,out | *context* | The lock context. |

Definition at line 273 of file smplock.h.

#### 8.129.3.9 _SMP_lock_Set_name()

```
static void _SMP_lock_Set_name (
              SMP_lock_Control * lock,
              const char * name )  [inline], [static]
```

Sets the name of an SMP lock.

**Parameters**

| | | |
|---|---|---|
| out | *lock* | The SMP lock control. |
| | *name* | The name for the SMP lock statistics. This name must be persistent throughout the life time of this statistics block. |

Definition at line 196 of file smplock.h.

#### 8.129.3.10 _SMP_MCS_lock_Destroy()

```
static void _SMP_MCS_lock_Destroy (
              SMP_MCS_lock_Control * lock )  [inline], [static]
```

Destroys the SMP MCS lock.

Concurrent destruction leads to unpredictable results.

**Parameters**

| | | |
|---|---|---|
| out | *lock* | The SMP MCS lock control. |

Definition at line 125 of file smplockmcs.h.

#### 8.129.3.11 _SMP_MCS_lock_Do_acquire()

```
static void _SMP_MCS_lock_Do_acquire (
              SMP_MCS_lock_Control * lock,
              SMP_MCS_lock_Context * context )  [inline], [static]
```

Acquires the SMP MCS lock.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The lock to acquire. |
| in,out | *context* | The lock context. |
| | *stats* | the SMP lock statistics. |

Definition at line 137 of file smplockmcs.h.

#### 8.129.3.12 _SMP_MCS_lock_Initialize()

```
static void _SMP_MCS_lock_Initialize (
              SMP_MCS_lock_Control * lock )  [inline], [static]
```

Initializes the SMP MCS lock.

Concurrent initialization leads to unpredictable results.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The SMP MCS lock control. |

Definition at line 113 of file smplockmcs.h.

#### 8.129.3.13 _SMP_MCS_lock_Release()

```
static void _SMP_MCS_lock_Release (
              SMP_MCS_lock_Control * lock,
              SMP_MCS_lock_Context * context )  [inline], [static]
```

Releases an SMP MCS lock.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The SMP MCS lock control. |
| in,out | *context* | The SMP MCS lock context. |

Definition at line 212 of file smplockmcs.h.

#### 8.129.3.14 _SMP_sequence_lock_Destroy()

```
static void _SMP_sequence_lock_Destroy (
            SMP_sequence_lock_Control * lock )  [inline], [static]
```

Destroys an SMP sequence lock.

Concurrent destruction leads to unpredictable results.

**Parameters**

| | |
|---|---|
| *lock* | The SMP sequence lock control. |

Definition at line 83 of file smplockseq.h.

#### 8.129.3.15 _SMP_sequence_lock_Initialize()

```
static void _SMP_sequence_lock_Initialize (
            SMP_sequence_lock_Control * lock )  [inline], [static]
```

Initializes an SMP sequence lock.

Concurrent initialization leads to unpredictable results.

**Parameters**

| | | |
|---|---|---|
| out | *lock* | The SMP sequence lock control. |

Definition at line 71 of file smplockseq.h.

#### 8.129.3.16 _SMP_sequence_lock_Read_begin()

```
static unsigned int _SMP_sequence_lock_Read_begin (
            const SMP_sequence_lock_Control * lock )  [inline], [static]
```

Begins an SMP sequence lock read operation.

This function will not disable interrupts.

**Parameters**

| | | |
|---|---|---|
| out | *lock* | The SMP sequence lock control. |

**Returns**

The current sequence number.

Definition at line 139 of file smplockseq.h.

### 8.129.3.17 _SMP_sequence_lock_Read_retry()

```
static bool _SMP_sequence_lock_Read_retry (
            SMP_sequence_lock_Control * lock,
            unsigned int seq )  [inline], [static]
```

Ends an SMP sequence lock read operation and indicates if a retry is necessary.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The SMP sequence lock control. |
| | *seq* | The sequence number returned by _SMP_sequence_lock_Read_begin(). |

**Return values**

| | |
|---|---|
| *true* | The read operation must be retried with a call to _SMP_sequence_lock_Read_begin(). |
| *false* | The read operation need not be retried. |

Definition at line 157 of file smplockseq.h.

### 8.129.3.18 _SMP_sequence_lock_Write_begin()

```
static unsigned int _SMP_sequence_lock_Write_begin (
            SMP_sequence_lock_Control * lock )  [inline], [static]
```

Begins an SMP sequence lock write operation.

This function will not disable interrupts. The caller must ensure that the current thread of execution is not interrupted indefinite since this would starve readers.

**Parameters**

| | | |
|---|---|---|
| out | *lock* | The SMP sequence lock control. |

**Returns**

The current sequence number.

Definition at line 99 of file smplockseq.h.

### 8.129.3.19 _SMP_sequence_lock_Write_end()

```
static void _SMP_sequence_lock_Write_end (
            SMP_sequence_lock_Control * lock,
            unsigned int seq )  [inline], [static]
```

Ends an SMP sequence lock write operation.

**Parameters**

| | | |
|---|---|---|
| out | *lock* | The SMP sequence lock control. |
| | *seq* | The sequence number returned by _SMP_sequence_lock_Write_begin(). |

Definition at line 122 of file smplockseq.h.

### 8.129.3.20 _SMP_ticket_lock_Destroy()

```
static void _SMP_ticket_lock_Destroy (
            SMP_ticket_lock_Control * lock )  [inline], [static]
```

Destroys the SMP ticket lock.

Concurrent destruction leads to unpredictable results.

**Parameters**

| | |
|---|---|
| *lock* | The SMP ticket lock control. |

Definition at line 76 of file smplockticket.h.

### 8.129.3.21 _SMP_ticket_lock_Do_acquire()

```
static void _SMP_ticket_lock_Do_acquire (
            SMP_ticket_lock_Control * lock )  [inline], [static]
```

Acquires the SMP ticket lock.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The lock to acquire. |
| | *stats* | The SMP lock statistics. |
| out | *stats_context* | The context for the statistics. |

Definition at line 88 of file smplockticket.h.

#### 8.129.3.22 _SMP_ticket_lock_Do_release()

```
static void _SMP_ticket_lock_Do_release (
            SMP_ticket_lock_Control * lock )  [inline], [static]
```

Releases the SMP ticket lock.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The SMP ticket lock to release. |
| out | *stats_context* | The SMP lock statistics context. |

Definition at line 159 of file smplockticket.h.

#### 8.129.3.23 _SMP_ticket_lock_Initialize()

```
static void _SMP_ticket_lock_Initialize (
            SMP_ticket_lock_Control * lock )  [inline], [static]
```

Initializes the SMP ticket lock.

Concurrent initialization leads to unpredictable results.

**Parameters**

| | | |
|---|---|---|
| in,out | *lock* | The SMP ticket lock control. |

Definition at line 61 of file smplockticket.h.

# 8.130 SMP Scheduler

SMP Scheduler.

## Modules

- EDF Priority SMP Scheduler

  *EDF Priority SMP Scheduler.*

## Files

- file schedulersmp.h

  *SMP Scheduler API.*
- file schedulersmpimpl.h

  *SMP Scheduler Implementation.*

## Classes

- struct Scheduler_SMP_Context

  *Scheduler context specialization for SMP schedulers.*
- struct Scheduler_SMP_Node

  *Scheduler node specialization for SMP schedulers.*

## Typedefs

- typedef bool(* **Scheduler_SMP_Has_ready**) (Scheduler_Context *context)
- typedef Scheduler_Node *(* **Scheduler_SMP_Get_highest_ready**) (Scheduler_Context *context, Scheduler_Node *node)
- typedef Scheduler_Node *(* **Scheduler_SMP_Get_lowest_scheduled**) (Scheduler_Context *context, Scheduler_Node *filter)
- typedef void(* **Scheduler_SMP_Extract**) (Scheduler_Context *context, Scheduler_Node *node_to_extract)
- typedef void(* **Scheduler_SMP_Insert**) (Scheduler_Context *context, Scheduler_Node *node_to_insert, Priority_Control insert_priority)
- typedef void(* **Scheduler_SMP_Move**) (Scheduler_Context *context, Scheduler_Node *node_to_move)
- typedef bool(* **Scheduler_SMP_Ask_for_help**) (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node)
- typedef void(* **Scheduler_SMP_Update**) (Scheduler_Context *context, Scheduler_Node *node_to_update, Priority_Control new_priority)
- typedef void(* **Scheduler_SMP_Set_affinity**) (Scheduler_Context *context, Scheduler_Node *node, void *arg)
- typedef bool(* **Scheduler_SMP_Enqueue**) (Scheduler_Context *context, Scheduler_Node *node_to_↩ enqueue, Priority_Control priority)
- typedef void(* **Scheduler_SMP_Allocate_processor**) (Scheduler_Context *context, Scheduler_Node *scheduled, Scheduler_Node *victim, Per_CPU_Control *victim_cpu)
- typedef void(* **Scheduler_SMP_Register_idle**) (Scheduler_Context *context, Scheduler_Node *idle, Per_CPU_Control *cpu)

## Enumerations

- enum Scheduler_SMP_Node_state { SCHEDULER_SMP_NODE_BLOCKED, SCHEDULER_SMP_NODE_SCHEDULED, SCHEDULER_SMP_NODE_READY }

  *SMP scheduler node states.*

## Functions

- void _Scheduler_SMP_Start_idle (const Scheduler_Control *scheduler, Thread_Control *idle, struct Per_CPU_Control *cpu)

  *Starts an idle thread on the specified cpu.*
- static void _Scheduler_SMP_Do_nothing_register_idle (Scheduler_Context *context, Scheduler_Node *idle, Per_CPU_Control *cpu)

  *Does nothing.*
- static bool _Scheduler_SMP_Priority_less_equal (const void *to_insert, const Chain_Node *next)

  *Checks if to_insert is less or equal than the priority of the chain node.*
- static Scheduler_SMP_Context * _Scheduler_SMP_Get_self (Scheduler_Context *context)

  *Gets the scheduler smp context.*
- static void _Scheduler_SMP_Initialize (Scheduler_SMP_Context *self)

  *Initializes the scheduler smp context.*
- static Scheduler_SMP_Node * _Scheduler_SMP_Thread_get_node (Thread_Control *thread)

  *Gets the scheduler smp node of the thread.*
- static Scheduler_SMP_Node * _Scheduler_SMP_Thread_get_own_node (Thread_Control *thread)

  *Gets the scheduler smp node of the thread.*
- static Scheduler_SMP_Node * _Scheduler_SMP_Node_downcast (Scheduler_Node *node)

  *Gets the scheduler smp node.*
- static Scheduler_SMP_Node_state _Scheduler_SMP_Node_state (const Scheduler_Node *node)

  *Gets the state of the node.*
- static Priority_Control _Scheduler_SMP_Node_priority (const Scheduler_Node *node)

  *Gets the priority of the node.*
- static void _Scheduler_SMP_Node_initialize (const Scheduler_Control *scheduler, Scheduler_SMP_Node *node, Thread_Control *thread, Priority_Control priority)

  *Initializes the scheduler smp node.*
- static void _Scheduler_SMP_Node_update_priority (Scheduler_SMP_Node *node, Priority_Control new_priority)

  *Updates the priority of the node to the new priority.*
- static void _Scheduler_SMP_Node_change_state (Scheduler_Node *node, Scheduler_SMP_Node_state new_state)

  *Changes the state of the node to the given state.*
- static bool _Scheduler_SMP_Is_processor_owned_by_us (const Scheduler_Context *context, const Per_CPU_Control *cpu)

  *Checks if the processor is owned by the given context.*
- static Thread_Control * _Scheduler_SMP_Get_idle_thread (Scheduler_Context *context)

  *Gets The first idle thread of the given context.*
- static void _Scheduler_SMP_Release_idle_thread (Scheduler_Context *context, Thread_Control *idle)

  *Releases the thread and adds it to the idle threads.*
- static void _Scheduler_SMP_Exctract_idle_thread (Thread_Control *idle)

  *Extracts the node of the idle thread.*
- static void _Scheduler_SMP_Allocate_processor_lazy (Scheduler_Context *context, Scheduler_Node *scheduled, Scheduler_Node *victim, Per_CPU_Control *victim_cpu)

  *Allocates the cpu for the scheduled thread.*

- static void _Scheduler_SMP_Allocate_processor_exact (Scheduler_Context *context, Scheduler_Node *scheduled, Scheduler_Node *victim, Per_CPU_Control *victim_cpu)

  *Allocates the cpu for the scheduled thread.*
- static void _Scheduler_SMP_Allocate_processor (Scheduler_Context *context, Scheduler_Node *scheduled, Scheduler_Node *victim, Per_CPU_Control *victim_cpu, Scheduler_SMP_Allocate_processor allocate_↩ processor)

  *Allocates the cpu for the scheduled thread using the given allocation function.*
- static Thread_Control * _Scheduler_SMP_Preempt (Scheduler_Context *context, Scheduler_Node *scheduled, Scheduler_Node *victim, Scheduler_SMP_Allocate_processor allocate_processor)

  *Preempts the victim's thread and allocates a cpu for the scheduled thread.*
- static Scheduler_Node * _Scheduler_SMP_Get_lowest_scheduled (Scheduler_Context *context, Scheduler_Node *filter)

  *Returns the lowest member of the scheduled nodes.*
- static void _Scheduler_SMP_Enqueue_to_scheduled (Scheduler_Context *context, Scheduler_Node *node, Priority_Control priority, Scheduler_Node *lowest_scheduled, Scheduler_SMP_Insert insert_scheduled, Scheduler_SMP_Move move_from_scheduled_to_ready, Scheduler_SMP_Allocate_processor allocate_↩ processor)

  *Tries to schedule the given node.*
- static bool _Scheduler_SMP_Enqueue (Scheduler_Context *context, Scheduler_Node *node, Priority_Control insert_priority, Chain_Node_order order, Scheduler_SMP_Insert insert_ready, Scheduler_SMP_Insert insert_scheduled, Scheduler_SMP_Move move_from_scheduled_to_ready, Scheduler_SMP_Get_lowest↩ _scheduled get_lowest_scheduled, Scheduler_SMP_Allocate_processor allocate_processor)

  *Enqueues a node according to the specified order function.*
- static bool _Scheduler_SMP_Enqueue_scheduled (Scheduler_Context *context, Scheduler_Node *const node, Priority_Control insert_priority, Chain_Node_order order, Scheduler_SMP_Extract extract_from_ready, Scheduler_SMP_Get_highest_ready get_highest_ready, Scheduler_SMP_Insert insert_ready, Scheduler_↩ SMP_Insert insert_scheduled, Scheduler_SMP_Move move_from_ready_to_scheduled, Scheduler_SMP↩ _Allocate_processor allocate_processor)

  *Enqueues a scheduled node according to the specified order function.*
- static void _Scheduler_SMP_Extract_from_scheduled (Scheduler_Context *context, Scheduler_Node *node)

  *Extracts a scheduled node from the scheduled nodes.*
- static void _Scheduler_SMP_Schedule_highest_ready (Scheduler_Context *context, Scheduler_Node *victim, Per_CPU_Control *victim_cpu, Scheduler_SMP_Extract extract_from_ready, Scheduler_S↩ MP_Get_highest_ready get_highest_ready, Scheduler_SMP_Move move_from_ready_to_scheduled, Scheduler_SMP_Allocate_processor allocate_processor)

  *Schedules the highest ready node.*
- static void _Scheduler_SMP_Preempt_and_schedule_highest_ready (Scheduler_Context *context, Scheduler_Node *victim, Per_CPU_Control *victim_cpu, Scheduler_SMP_Extract extract_from_ready, Scheduler_SMP_Get_highest_ready get_highest_ready, Scheduler_SMP_Move move_from_ready_to_↩ scheduled, Scheduler_SMP_Allocate_processor allocate_processor)

  *Schedules the highest ready node and preempts a currently executing one.*
- static void _Scheduler_SMP_Block (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node, Scheduler_SMP_Extract extract_from_scheduled, Scheduler_SMP_Extract extract_from_ready, Scheduler_SMP_Get_highest_ready get_highest_ready, Scheduler_SMP_Move move_from_ready_to_↩ scheduled, Scheduler_SMP_Allocate_processor allocate_processor)

  *Blocks the thread.*
- static void _Scheduler_SMP_Unblock (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node, Scheduler_SMP_Update update, Scheduler_SMP_Enqueue enqueue)

  *Unblocks the thread.*
- static void _Scheduler_SMP_Update_priority (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node, Scheduler_SMP_Extract extract_from_ready, Scheduler_SMP_Update update, Scheduler_SMP_Enqueue enqueue, Scheduler_SMP_Enqueue enqueue_scheduled, Scheduler_SMP_↩ Ask_for_help ask_for_help)

  *Updates the priority of the node and the position in the queues it is in.*

- static void _Scheduler_SMP_Yield (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node, Scheduler_SMP_Extract extract_from_ready, Scheduler_SMP_Enqueue enqueue, Scheduler_SM↩ P_Enqueue enqueue_scheduled)

  *Performs a yield and asks for help if necessary.*
- static void _Scheduler_SMP_Insert_scheduled (Scheduler_Context *context, Scheduler_Node *node_to_↩ insert, Priority_Control priority_to_insert)

  *Inserts the node with the given priority into the scheduled nodes.*
- static bool _Scheduler_SMP_Ask_for_help (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node, Chain_Node_order order, Scheduler_SMP_Insert insert_ready, Scheduler_S↩ MP_Insert insert_scheduled, Scheduler_SMP_Move move_from_scheduled_to_ready, Scheduler_SMP_↩ Get_lowest_scheduled get_lowest_scheduled, Scheduler_SMP_Allocate_processor allocate_processor)

  *Asks for help.*
- static void _Scheduler_SMP_Reconsider_help_request (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node, Scheduler_SMP_Extract extract_from_ready)

  *Reconsiders help request.*
- static void _Scheduler_SMP_Withdraw_node (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node, Thread_Scheduler_state next_state, Scheduler_SMP_Extract extract_from_ready, Scheduler_SMP_Get_highest_ready get_highest_ready, Scheduler_SMP_Move move_from_ready_to_↩ scheduled, Scheduler_SMP_Allocate_processor allocate_processor)

  *Withdraws the node.*
- static void _Scheduler_SMP_Do_start_idle (Scheduler_Context *context, Thread_Control *idle, Per_CPU_Control *cpu, Scheduler_SMP_Register_idle register_idle)

  *Starts the idle thread on the given processor.*
- static void _Scheduler_SMP_Add_processor (Scheduler_Context *context, Thread_Control *idle, Scheduler_SMP_Has_ready has_ready, Scheduler_SMP_Enqueue enqueue_scheduled, Scheduler_S↩ MP_Register_idle register_idle)

  *Adds the idle thread to the processor.*
- static Thread_Control * _Scheduler_SMP_Remove_processor (Scheduler_Context *context, Per_CPU_Control *cpu, Scheduler_SMP_Extract extract_from_ready, Scheduler_SMP_Enqueue enqueue)

  *Removes an idle thread from the processor.*
- static void _Scheduler_SMP_Set_affinity (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node, void *arg, Scheduler_SMP_Set_affinity set_affinity, Scheduler_SMP_Extract extract_from_ready, Scheduler_SMP_Get_highest_ready get_highest_ready, Scheduler_SMP_Move move_from_ready_to_scheduled, Scheduler_SMP_Enqueue enqueue, Scheduler_SMP_Allocate_processor allocate_processor)

  *Sets the affinity of the node.*

### 8.130.1 Detailed Description

SMP Scheduler.

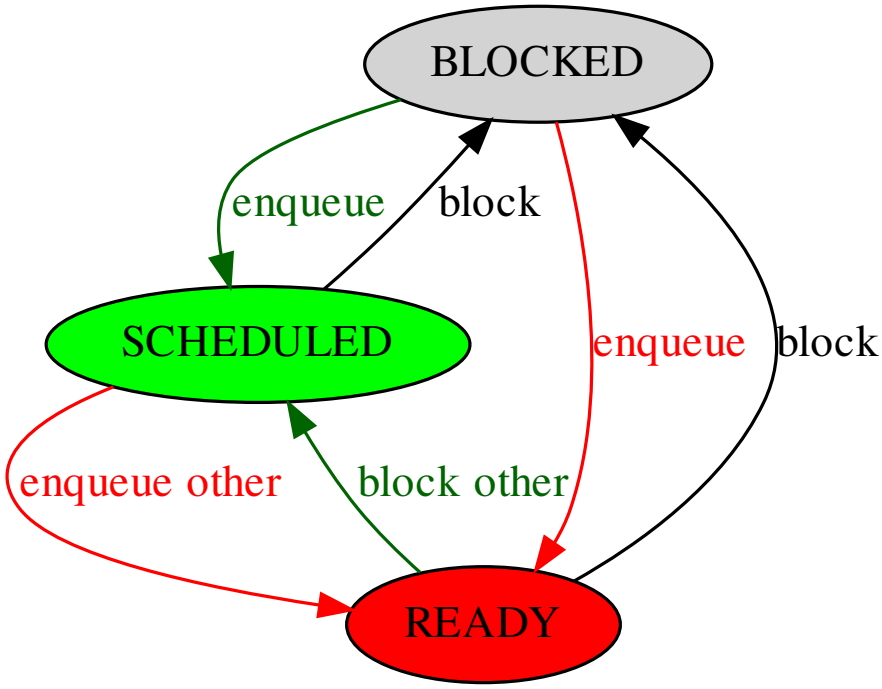
The scheduler nodes can be in four states

- SCHEDULER_SMP_NODE_BLOCKED,
- SCHEDULER_SMP_NODE_SCHEDULED, and
- SCHEDULER_SMP_NODE_READY.

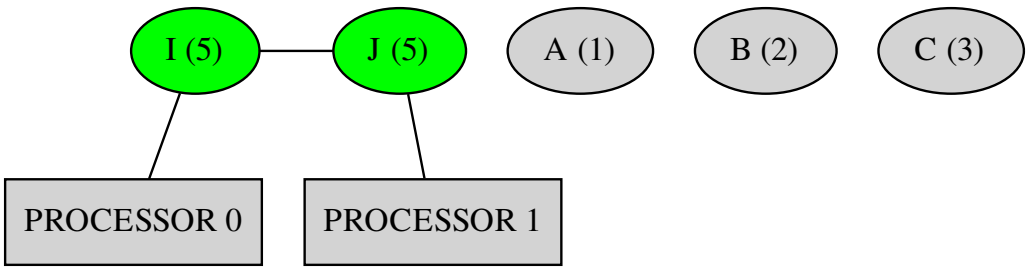
State transitions are triggered via basic operations

- _Scheduler_SMP_Enqueue(),

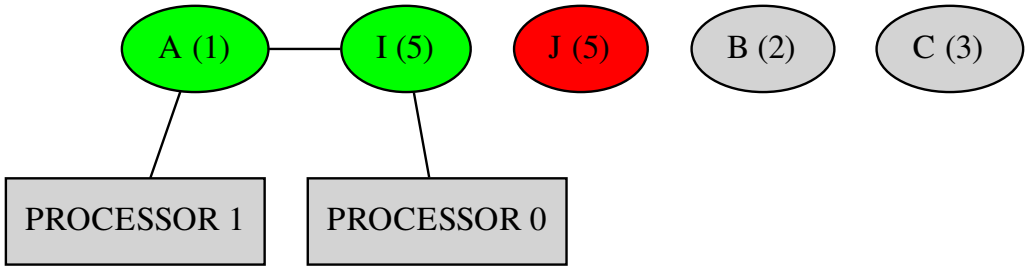
- _Scheduler_SMP_Enqueue_scheduled(), and
- _Scheduler_SMP_Block().



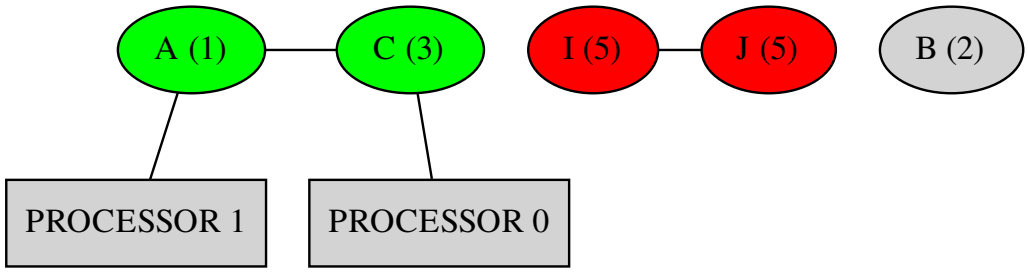

During system initialization each processor of the scheduler instance starts with an idle thread assigned to it. Lets have a look at an example with two idle threads I and J with priority 5. We also have blocked threads A, B and C with priorities 1, 2 and 3 respectively. The scheduler nodes are ordered with respect to the thread priority from left to right in the below diagrams. The highest priority node (lowest priority number) is the leftmost node. Since the processor assignment is independent of the thread priority the processor indices may move from one state to the other.



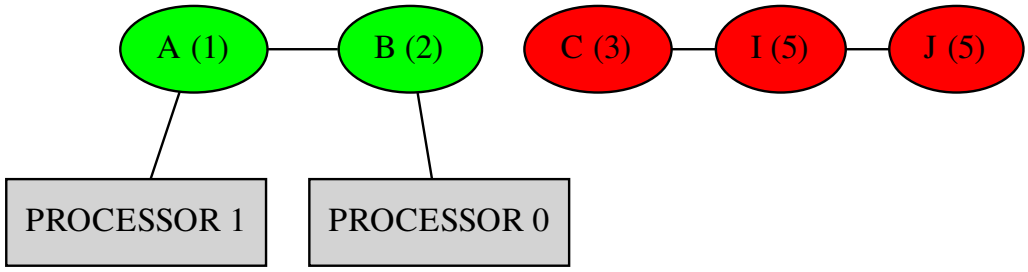
Lets start A. For this an enqueue operation is performed.



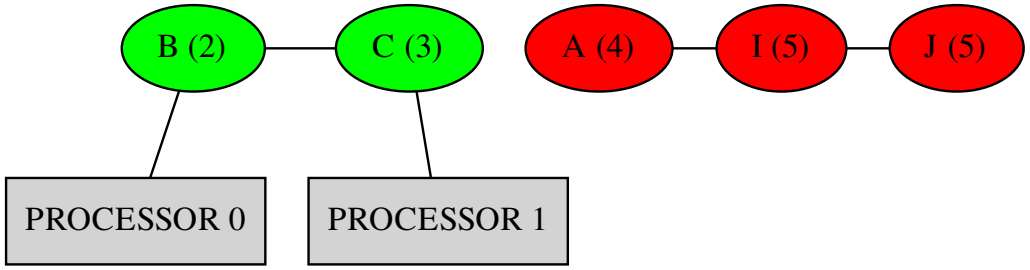

Lets start C.



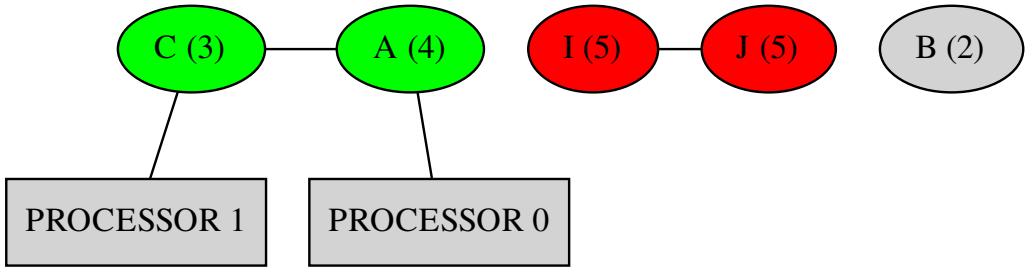

Lets start B.



Lets change the priority of thread A to 4.




Now perform a blocking operation with thread B. Please note that thread A migrated now from processor 0 to processor 1 and thread C still executes on processor 1.




## 8.130.2 Enumeration Type Documentation

### 8.130.2.1 Scheduler_SMP_Node_state

```
enum Scheduler_SMP_Node_state
```

SMP scheduler node states.

**Enumerator**

| | |
|---|---|
| SCHEDULER_SMP_NODE_BLOCKED | This scheduler node is blocked. A scheduler node is blocked if the corresponding thread is not ready. |

**Enumerator**

| | |
|---|---|
| SCHEDULER_SMP_NODE_SCHEDULED | The scheduler node is scheduled. A scheduler node is scheduled if the corresponding thread is ready and the scheduler allocated a processor for it. A scheduled node is assigned to exactly one processor. The count of scheduled nodes in this scheduler instance equals the processor count owned by the scheduler instance. |
| SCHEDULER_SMP_NODE_READY | This scheduler node is ready. A scheduler node is ready if the corresponding thread is ready and the scheduler did not allocate a processor for it. |

Definition at line 70 of file schedulersmp.h.

## 8.130.3 Function Documentation

### 8.130.3.1 _Scheduler_SMP_Add_processor()

```
static void _Scheduler_SMP_Add_processor (
            Scheduler_Context * context,
            Thread_Control * idle,
            Scheduler_SMP_Has_ready has_ready,
            Scheduler_SMP_Enqueue enqueue_scheduled,
            Scheduler_SMP_Register_idle register_idle )  [inline], [static]
```

Adds the idle thread to the processor.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler context instance. |
| in,out | *idle* | The idle thread to add to the processor. |
| | *has_ready* | Function that checks if a given context has ready threads. |
| | *enqueue_scheduled* | Function to enqueue a scheduled node. |
| | *register_idle* | Function to register the idle thread for a cpu. |

Definition at line 1674 of file schedulersmpimpl.h.

### 8.130.3.2 _Scheduler_SMP_Allocate_processor()

```
static void _Scheduler_SMP_Allocate_processor (
            Scheduler_Context * context,
            Scheduler_Node * scheduled,
            Scheduler_Node * victim,
            Per_CPU_Control * victim_cpu,
            Scheduler_SMP_Allocate_processor allocate_processor )  [inline], [static]
```

Allocates the cpu for the scheduled thread using the given allocation function.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler context instance. |
| | *scheduled* | The scheduled node that should be executed next. |
| | *victim* | If the heir is this node's thread, no processor is allocated. |
| in,out | *victim_cpu* | The cpu to allocate. |
| | *allocate_processor* | The function to use for the allocation of *victim_cpu*. |

Definition at line 685 of file schedulersmpimpl.h.

### 8.130.3.3 _Scheduler_SMP_Allocate_processor_exact()

```
static void _Scheduler_SMP_Allocate_processor_exact (
            Scheduler_Context * context,
            Scheduler_Node * scheduled,
            Scheduler_Node * victim,
            Per_CPU_Control * victim_cpu )  [inline], [static]
```

Allocates the cpu for the scheduled thread.

This method is slightly different from _Scheduler_SMP_Allocate_processor_lazy() in that it does what it is asked to do. _Scheduler_SMP_Allocate_processor_lazy() attempts to prevent migrations but does not take into account affinity.

**Parameters**

| | | |
|---|---|---|
| | *context* | This parameter is unused. |
| | *scheduled* | The scheduled node whose thread should be executed next. |
| | *victim* | This parameter is unused. |
| in,out | *victim_cpu* | The cpu to allocate. |

Definition at line 659 of file schedulersmpimpl.h.

### 8.130.3.4 _Scheduler_SMP_Allocate_processor_lazy()

```
static void _Scheduler_SMP_Allocate_processor_lazy (
            Scheduler_Context * context,
            Scheduler_Node * scheduled,
            Scheduler_Node * victim,
            Per_CPU_Control * victim_cpu )  [inline], [static]
```

Allocates the cpu for the scheduled thread.

Attempts to prevent migrations but does not take into account affinity.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler context instance. |
| | *scheduled* | The scheduled node that should be executed next. |
| | *victim* | If the heir is this node's thread, no processor is allocated. |
| in,out | *victim_cpu* | The cpu to allocate. |

Definition at line 609 of file schedulersmpimpl.h.

#### 8.130.3.5 _Scheduler_SMP_Ask_for_help()

```
static bool _Scheduler_SMP_Ask_for_help (
            Scheduler_Context * context,
            Thread_Control * thread,
            Scheduler_Node * node,
            Chain_Node_order order,
            Scheduler_SMP_Insert insert_ready,
            Scheduler_SMP_Insert insert_scheduled,
            Scheduler_SMP_Move move_from_scheduled_to_ready,
            Scheduler_SMP_Get_lowest_scheduled get_lowest_scheduled,
            Scheduler_SMP_Allocate_processor allocate_processor )  [inline], [static]
```

Asks for help.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler instance context. |
| | *thread* | The thread that asks for help. |
| in,out | *node* | The node of the thread that performs the ask for help operation. |
| | *order* | The order function. |
| | *insert_ready* | Function to insert a node into the set of ready nodes. |
| | *insert_scheduled* | Function to insert a node into the set of scheduled nodes. |
| | *move_from_scheduled_to_ready* | Function to move a node from the set of scheduled nodes to the set of ready nodes. |
| | *get_lowest_scheduled* | Function to select the node from the scheduled nodes to replace. |
| | *allocate_processor* | Function to allocate a processor to a node based on the rules of the scheduler. |

**Return values**

| | |
|---|---|
| *true* | The ask for help operation was successful. |
| *false* | The ask for help operation was not successful. |

Definition at line 1449 of file schedulersmpimpl.h.

### 8.130.3.6 _Scheduler_SMP_Block()

```
static void _Scheduler_SMP_Block (
             Scheduler_Context * context,
             Thread_Control * thread,
             Scheduler_Node * node,
             Scheduler_SMP_Extract extract_from_scheduled,
             Scheduler_SMP_Extract extract_from_ready,
             Scheduler_SMP_Get_highest_ready get_highest_ready,
             Scheduler_SMP_Move move_from_ready_to_scheduled,
             Scheduler_SMP_Allocate_processor allocate_processor )  [inline], [static]
```

Blocks the thread.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler instance context. |
| in,out | *thread* | The thread of the scheduling operation. |
| in,out | *node* | The scheduler node of the thread to block. |
| | *extract_from_scheduled* | Function to extract a node from the set of scheduled nodes. |
| | *extract_from_ready* | Function to extract a node from the set of ready nodes. |
| | *get_highest_ready* | Function to get the highest ready node. |
| | *move_from_ready_to_scheduled* | Function to move a node from the set of ready nodes to the set of scheduled nodes. |
| | *allocate_processor* | Function to allocate a processor to a node based on the rules of the scheduler. |

Definition at line 1189 of file schedulersmpimpl.h.

### 8.130.3.7 _Scheduler_SMP_Do_nothing_register_idle()

```
static void _Scheduler_SMP_Do_nothing_register_idle (
             Scheduler_Context * context,
             Scheduler_Node * idle,
             Per_CPU_Control * cpu )  [inline], [static]
```

Does nothing.

**Parameters**

| | |
|---|---|
| *context* | This parameter is unused. |
| *idle* | This parameter is unused. |
| *cpu* | This parameter is unused. |

Definition at line 352 of file schedulersmpimpl.h.

#### 8.130.3.8 _Scheduler_SMP_Do_start_idle()

```
static void _Scheduler_SMP_Do_start_idle (
             Scheduler_Context * context,
             Thread_Control * idle,
             Per_CPU_Control * cpu,
             Scheduler_SMP_Register_idle register_idle )  [inline], [static]
```

Starts the idle thread on the given processor.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler context instance. |
| in,out | *idle* | The idle thread to schedule. |
| | *cpu* | The processor for the idle thread. |
| | *register_idle* | Function to register the idle thread for a cpu. |

Definition at line 1643 of file schedulersmpimpl.h.

#### 8.130.3.9 _Scheduler_SMP_Enqueue()

```
static bool _Scheduler_SMP_Enqueue (
             Scheduler_Context * context,
             Scheduler_Node * node,
             Priority_Control insert_priority,
             Chain_Node_order order,
             Scheduler_SMP_Insert insert_ready,
             Scheduler_SMP_Insert insert_scheduled,
             Scheduler_SMP_Move move_from_scheduled_to_ready,
             Scheduler_SMP_Get_lowest_scheduled get_lowest_scheduled,
             Scheduler_SMP_Allocate_processor allocate_processor )  [inline], [static]
```

Enqueues a node according to the specified order function.

The node must not be in the scheduled state.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler instance context. |
| in,out | *node* | The node to enqueue. |
| | *priority* | The node insert priority. |
| | *order* | The order function. |
| | *insert_ready* | Function to insert a node into the set of ready nodes. |
| | *insert_scheduled* | Function to insert a node into the set of scheduled nodes. |
| | *move_from_scheduled_to_ready* | Function to move a node from the set of scheduled nodes to the set of ready nodes. |
| | *get_lowest_scheduled* | Function to select the node from the scheduled nodes to replace. It may not be possible to find one, in this case a pointer must be returned so that the order functions returns false if this pointer is passed as the second argument to the order function. |
| | *allocate_processor* | Function to allocate a processor to a node based on the rules |
| | | of the scheduler. |

Definition at line 875 of file schedulersmpimpl.h.

### 8.130.3.10 _Scheduler_SMP_Enqueue_scheduled()

```
static bool _Scheduler_SMP_Enqueue_scheduled (
              Scheduler_Context * context,
              Scheduler_Node *const node,
              Priority_Control insert_priority,
              Chain_Node_order order,
              Scheduler_SMP_Extract extract_from_ready,
              Scheduler_SMP_Get_highest_ready get_highest_ready,
              Scheduler_SMP_Insert insert_ready,
              Scheduler_SMP_Insert insert_scheduled,
              Scheduler_SMP_Move move_from_ready_to_scheduled,
              Scheduler_SMP_Allocate_processor allocate_processor )  [inline], [static]
```

Enqueues a scheduled node according to the specified order function.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler instance context. |
| in,out | *node* | The node to enqueue. |
| | *order* | The order function. |
| | *extract_from_ready* | Function to extract a node from the set of ready nodes. |
| | *get_highest_ready* | Function to get the highest ready node. |
| | *insert_ready* | Function to insert a node into the set of ready nodes. |
| | *insert_scheduled* | Function to insert a node into the set of scheduled nodes. |
| | *move_from_ready_to_scheduled* | Function to move a node from the set of ready nodes to the set of scheduled nodes. |
| | *allocate_processor* | Function to allocate a processor to a node based on the rules of the scheduler. |

Definition at line 930 of file schedulersmpimpl.h.

### 8.130.3.11 _Scheduler_SMP_Enqueue_to_scheduled()

```
static void _Scheduler_SMP_Enqueue_to_scheduled (
              Scheduler_Context * context,
              Scheduler_Node * node,
              Priority_Control priority,
              Scheduler_Node * lowest_scheduled,
              Scheduler_SMP_Insert insert_scheduled,
              Scheduler_SMP_Move move_from_scheduled_to_ready,
              Scheduler_SMP_Allocate_processor allocate_processor )  [inline], [static]
```

Tries to schedule the given node.

Schedules the node, or blocks if that is necessary.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler context instance. |
| in,out | *node* | The node to insert into the scheduled nodes. |
| | *priority* | The priority of *node*. |
| in,out | *lowest_scheduled* | The lowest member of the scheduled nodes. |
| | *insert_scheduled* | Function to insert a node into the set of scheduled nodes. |
| | *move_from_scheduled_to_ready* | Function to move a node from the set of scheduled nodes to the set of ready nodes. |
| | *allocate_processor* | Function to allocate a processor to a node based on the rules of the scheduler. |

Definition at line 797 of file schedulersmpimpl.h.

#### 8.130.3.12 _Scheduler_SMP_Exctract_idle_thread()

```
static void _Scheduler_SMP_Exctract_idle_thread (
            Thread_Control * idle )  [inline], [static]
```

Extracts the node of the idle thread.

**Parameters**

| | | |
|---|---|---|
| in,out | *idle* | The thread to extract the node of. |

Definition at line 592 of file schedulersmpimpl.h.

#### 8.130.3.13 _Scheduler_SMP_Extract_from_scheduled()

```
static void _Scheduler_SMP_Extract_from_scheduled (
            Scheduler_Context * context,
            Scheduler_Node * node )  [inline], [static]
```

Extracts a scheduled node from the scheduled nodes.

**Parameters**

| | |
|---|---|
| *context* | This parameter is unused. |
| *node* | The node to extract from the chain it belongs to. |

Definition at line 1047 of file schedulersmpimpl.h.

### 8.130.3.14 _Scheduler_SMP_Get_idle_thread()

```
static Thread_Control* _Scheduler_SMP_Get_idle_thread (
            Scheduler_Context * context )  [inline], [static]
```

Gets The first idle thread of the given context.

**Parameters**

| | |
|---|---|
| *context* | The scheduler context to get the first idle thread from. |

**Returns**

The first idle thread of *context*.

Definition at line 558 of file schedulersmpimpl.h.

### 8.130.3.15 _Scheduler_SMP_Get_lowest_scheduled()

```
static Scheduler_Node* _Scheduler_SMP_Get_lowest_scheduled (
            Scheduler_Context * context,
            Scheduler_Node * filter )  [inline], [static]
```

Returns the lowest member of the scheduled nodes.

**Parameters**

| | |
|---|---|
| *context* | The scheduler context instance. |
| *filter* | This parameter is unused. |

**Returns**

The lowest scheduled node.

Definition at line 761 of file schedulersmpimpl.h.

### 8.130.3.16 _Scheduler_SMP_Get_self()

```
static Scheduler_SMP_Context* _Scheduler_SMP_Get_self (
            Scheduler_Context * context )  [inline], [static]
```

Gets the scheduler smp context.

**Parameters**

| | |
|---|---|
| *context* | The context to cast to Scheduler_SMP_Context *. |

**Returns**

*context* cast to Scheduler_SMP_Context *.

Definition at line 393 of file schedulersmpimpl.h.

#### 8.130.3.17 _Scheduler_SMP_Initialize()

```
static void _Scheduler_SMP_Initialize (
            Scheduler_SMP_Context * self )  [inline], [static]
```

Initializes the scheduler smp context.

**Parameters**

| | | |
|---|---|---|
| out | *self* | The context to initialize. |

Definition at line 405 of file schedulersmpimpl.h.

#### 8.130.3.18 _Scheduler_SMP_Insert_scheduled()

```
static void _Scheduler_SMP_Insert_scheduled (
            Scheduler_Context * context,
            Scheduler_Node * node_to_insert,
            Priority_Control priority_to_insert )  [inline], [static]
```

Inserts the node with the given priority into the scheduled nodes.

**Parameters**

| | |
|---|---|
| *context* | The scheduler instance context. |
| *node_to_insert* | The scheduled node to insert. |
| *priority_to_insert* | The priority with which to insert the node. |

Definition at line 1409 of file schedulersmpimpl.h.

#### 8.130.3.19 _Scheduler_SMP_Is_processor_owned_by_us()

```
static bool _Scheduler_SMP_Is_processor_owned_by_us (
            const Scheduler_Context * context,
            const Per_CPU_Control * cpu )  [inline], [static]
```

Checks if the processor is owned by the given context.

**Parameters**

| | |
|---|---|
| *context* | The context to check whether *cpu* is owned by it. |
| *cpu* | The cpu to check whether it is owned by *context*. |

**Return values**

| | |
|---|---|
| *true* | *cpu* is owned by *context*. |
| *false* | *cpu* is not owned by *context*. |

Definition at line 543 of file schedulersmpimpl.h.

### 8.130.3.20 _Scheduler_SMP_Node_change_state()

```
static void _Scheduler_SMP_Node_change_state (
              Scheduler_Node * node,
              Scheduler_SMP_Node_state new_state )  [inline], [static]
```

Changes the state of the node to the given state.

**Parameters**

| | | |
|---|---|---|
| out | *node* | the node to change the state of. |
| | *new_state* | The new state for *node*. |

Definition at line 523 of file schedulersmpimpl.h.

### 8.130.3.21 _Scheduler_SMP_Node_downcast()

```
static Scheduler_SMP_Node* _Scheduler_SMP_Node_downcast (
              Scheduler_Node * node )  [inline], [static]
```

Gets the scheduler smp node.

**Parameters**

| | |
|---|---|
| *node* | The node to cast to Scheduler_SMP_Node *. |

**Returns**

*node* cast to Scheduler_SMP_Node *.

Definition at line 448 of file schedulersmpimpl.h.

#### 8.130.3.22 _Scheduler_SMP_Node_initialize()

```
static void _Scheduler_SMP_Node_initialize (
              const Scheduler_Control * scheduler,
              Scheduler_SMP_Node * node,
              Thread_Control * thread,
              Priority_Control priority )  [inline], [static]
```

Initializes the scheduler smp node.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| out | *node* | The node to initialize. |
| | *thread* | The thread of the scheduler smp node. |
| | *priority* | The priority to initialize *node* with. |

Definition at line 491 of file schedulersmpimpl.h.

#### 8.130.3.23 _Scheduler_SMP_Node_priority()

```
static Priority_Control _Scheduler_SMP_Node_priority (
              const Scheduler_Node * node )  [inline], [static]
```

Gets the priority of the node.

**Parameters**

| | |
|---|---|
| *node* | The node to get the priority of. |

**Returns**

The priority of *node*.

Definition at line 476 of file schedulersmpimpl.h.

#### 8.130.3.24 _Scheduler_SMP_Node_state()

```
static Scheduler_SMP_Node_state _Scheduler_SMP_Node_state (
              const Scheduler_Node * node )  [inline], [static]
```

Gets the state of the node.

**Parameters**

| | |
|---|---|
| *node* | The node to get the state of. |

**Returns**

The state of *node*.

Definition at line 462 of file schedulersmpimpl.h.

### 8.130.3.25 _Scheduler_SMP_Node_update_priority()

```
static void _Scheduler_SMP_Node_update_priority (
              Scheduler_SMP_Node * node,
              Priority_Control new_priority )  [inline], [static]
```

Updates the priority of the node to the new priority.

**Parameters**

| | | |
|---|---|---|
| out | *node* | The node to update the priority of. |
| | *new_priority* | The new priority for *node*. |

Definition at line 509 of file schedulersmpimpl.h.

### 8.130.3.26 _Scheduler_SMP_Preempt()

```
static Thread_Control* _Scheduler_SMP_Preempt (
              Scheduler_Context * context,
              Scheduler_Node * scheduled,
              Scheduler_Node * victim,
              Scheduler_SMP_Allocate_processor allocate_processor )  [inline], [static]
```

Preempts the victim's thread and allocates a cpu for the scheduled thread.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler context instance. |
| | *scheduled* | Node of the scheduled thread that is about to be executed. |
| in,out | *victim* | Node of the thread to preempt. |
| | *allocate_processor* | The function for allocation of a processor for the new thread. |

**Returns**

The preempted thread.

Definition at line 707 of file schedulersmpimpl.h.

#### 8.130.3.27 _Scheduler_SMP_Preempt_and_schedule_highest_ready()

```
static void _Scheduler_SMP_Preempt_and_schedule_highest_ready (
              Scheduler_Context * context,
              Scheduler_Node * victim,
              Per_CPU_Control * victim_cpu,
              Scheduler_SMP_Extract extract_from_ready,
              Scheduler_SMP_Get_highest_ready get_highest_ready,
              Scheduler_SMP_Move move_from_ready_to_scheduled,
              Scheduler_SMP_Allocate_processor allocate_processor )  [inline], [static]
```

Schedules the highest ready node and preempts a currently executing one.

**Parameters**

| | |
|---|---|
| *context* | The scheduler context instance. |
| *victim* | The node of the thread that is repressed by the newly scheduled thread. |
| *victim_cpu* | The cpu to allocate. |
| *extract_from_ready* | Function to extract a node from the set of ready nodes. |
| *get_highest_ready* | Function to get the highest ready node. |
| *move_from_ready_to_scheduled* | Function to move a node from the set of ready nodes to the set of scheduled nodes. |
| *allocate_processor* | Function to allocate a processor to a node based on the rules of the scheduler. |

Definition at line 1129 of file schedulersmpimpl.h.

#### 8.130.3.28 _Scheduler_SMP_Priority_less_equal()

```
static bool _Scheduler_SMP_Priority_less_equal (
              const void * to_insert,
              const Chain_Node * next )  [inline], [static]
```

Checks if *to_insert* is less or equal than the priority of the chain node.

**Parameters**

| | |
|---|---|
| *to_insert* | The priority to compare. |
| *next* | The chain node to compare the priority of. |

**Return values**

| | |
|---|---|
| *true* | *to_insert* is less or equal than the priority of *next*. |
| *false* | *to_insert* is greater than the priority of *next*. |

Definition at line 372 of file schedulersmpimpl.h.

#### 8.130.3.29 _Scheduler_SMP_Reconsider_help_request()

```
static void _Scheduler_SMP_Reconsider_help_request (
              Scheduler_Context * context,
              Thread_Control * thread,
              Scheduler_Node * node,
              Scheduler_SMP_Extract extract_from_ready )  [inline], [static]
```

Reconsiders help request.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler context instance. |
| | *thread* | The thread to reconsider the help request of. |
| in,out | *node* | The scheduler node of *thread*. |
| | *extract_from_ready* | Function to extract a node from the ready queue of the scheduler context. |

Definition at line 1552 of file schedulersmpimpl.h.

#### 8.130.3.30 _Scheduler_SMP_Release_idle_thread()

```
static void _Scheduler_SMP_Release_idle_thread (
              Scheduler_Context * context,
              Thread_Control * idle )  [inline], [static]
```

Releases the thread and adds it to the idle threads.

**Parameters**

| | | |
|---|---|---|
| in,out | *context* | The scheduler context instance. |
| | *idle* | The thread to add to the idle threads. |

Definition at line 577 of file schedulersmpimpl.h.

#### 8.130.3.31 _Scheduler_SMP_Remove_processor()

```
static Thread_Control* _Scheduler_SMP_Remove_processor (
              Scheduler_Context * context,
              Per_CPU_Control * cpu,
              Scheduler_SMP_Extract extract_from_ready,
              Scheduler_SMP_Enqueue enqueue )  [inline], [static]
```

Removes an idle thread from the processor.

**Parameters**

| | |
|---|---|
| *context* | The scheduler context instance. |
| *cpu* | The processor to remove from. |
| *extract_from_ready* | Function to extract a node from the ready queue of the scheduler context. |
| *enqueue* | Function to enqueue a node with a given priority. |

**Returns**

The idle thread of *cpu*.

Definition at line 1714 of file schedulersmpimpl.h.

#### 8.130.3.32 _Scheduler_SMP_Schedule_highest_ready()

```
static void _Scheduler_SMP_Schedule_highest_ready (
              Scheduler_Context * context,
              Scheduler_Node * victim,
              Per_CPU_Control * victim_cpu,
              Scheduler_SMP_Extract extract_from_ready,
              Scheduler_SMP_Get_highest_ready get_highest_ready,
              Scheduler_SMP_Move move_from_ready_to_scheduled,
              Scheduler_SMP_Allocate_processor allocate_processor )  [inline], [static]
```

Schedules the highest ready node.

**Parameters**

| | |
|---|---|
| *context* | The scheduler context instance. |
| *victim* | The node of the thread that is repressed by the newly scheduled thread. |
| *victim_cpu* | The cpu to allocate. |
| *extract_from_ready* | Function to extract a node from the set of ready nodes. |
| *get_highest_ready* | Function to get the highest ready node. |
| *move_from_ready_to_scheduled* | Function to move a node from the set of ready nodes to the set of scheduled nodes. |
| *allocate_processor* | Function to allocate a processor to a node based on the rules of the scheduler. |

Definition at line 1070 of file schedulersmpimpl.h.

#### 8.130.3.33 _Scheduler_SMP_Set_affinity()

```
static void _Scheduler_SMP_Set_affinity (
              Scheduler_Context * context,
              Thread_Control * thread,
              Scheduler_Node * node,
              void * arg,
              Scheduler_SMP_Set_affinity set_affinity,
              Scheduler_SMP_Extract extract_from_ready,
              Scheduler_SMP_Get_highest_ready get_highest_ready,
              Scheduler_SMP_Move move_from_ready_to_scheduled,
              Scheduler_SMP_Enqueue enqueue,
              Scheduler_SMP_Allocate_processor allocate_processor )  [inline], [static]
```

Sets the affinity of the node.

Also performs a reinsert into the queue the node is currently in.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler context instance. |
| | *thread* | The thread for the operation. |
| in,out | *node* | The node to set the affinity of. |
| | *arg* | The affinity for *node*. |
| | *set_affinity* | Function to set the affinity of a node. |
| | *extract_from_ready* | Function to extract a node from the ready queue of the scheduler context. |
| | *get_highest_ready* | Function to get the highest ready node. |
| | *move_from_ready_to_scheduled* | Function to move a node from the set of ready nodes to the set of scheduled nodes. |
| | *enqueue* | Function to enqueue a node with a given priority. |
| | *allocate_processor* | Function to allocate a processor to a node based on the rules of the scheduler. |

Definition at line 1795 of file schedulersmpimpl.h.

### 8.130.3.34 _Scheduler_SMP_Start_idle()

```
void _Scheduler_SMP_Start_idle (
             const Scheduler_Control * scheduler,
             Thread_Control * idle,
             struct Per_CPU_Control * cpu )
```

Starts an idle thread on the specified cpu.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| in,out | *idle* | The idle thread to schedule. |
| out | *cpu* | The cpu to run the idle thread on. |

Definition at line 15 of file schedulersmpstartidle.c.

### 8.130.3.35 _Scheduler_SMP_Thread_get_node()

```
static Scheduler_SMP_Node* _Scheduler_SMP_Thread_get_node (
             Thread_Control * thread )  [inline], [static]
```

Gets the scheduler smp node of the thread.

**Parameters**

| | |
|---|---|
| *thread* | The thread to get the smp node of. |

**Returns**

The scheduler smp node of *thread*.

Definition at line 420 of file schedulersmpimpl.h.

#### 8.130.3.36 _Scheduler_SMP_Thread_get_own_node()

```
static Scheduler_SMP_Node* _Scheduler_SMP_Thread_get_own_node (
            Thread_Control * thread )  [inline], [static]
```

Gets the scheduler smp node of the thread.

**Parameters**

| | |
|---|---|
| *thread* | The thread to get the smp node of. |

**Returns**

The scheduler smp node of *thread*.

Definition at line 434 of file schedulersmpimpl.h.

#### 8.130.3.37 _Scheduler_SMP_Unblock()

```
static void _Scheduler_SMP_Unblock (
            Scheduler_Context * context,
            Thread_Control * thread,
            Scheduler_Node * node,
            Scheduler_SMP_Update update,
            Scheduler_SMP_Enqueue enqueue )  [inline], [static]
```

Unblocks the thread.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler instance context. |
| in,out | *thread* | The thread of the scheduling operation. |
| in,out | *node* | The scheduler node of the thread to block. |
| | *update* | Function to update the node's priority to the new value. |
| | *enqueue* | Function to insert a node with a priority in the ready queue of a context. |

Definition at line 1243 of file schedulersmpimpl.h.

### 8.130.3.38 _Scheduler_SMP_Update_priority()

```
static void _Scheduler_SMP_Update_priority (
              Scheduler_Context * context,
              Thread_Control * thread,
              Scheduler_Node * node,
              Scheduler_SMP_Extract extract_from_ready,
              Scheduler_SMP_Update update,
              Scheduler_SMP_Enqueue enqueue,
              Scheduler_SMP_Enqueue enqueue_scheduled,
              Scheduler_SMP_Ask_for_help ask_for_help )  [inline], [static]
```

Updates the priority of the node and the position in the queues it is in.

This function firstly updates the priority of the node and then extracts and reinserts it into the queue the node is part of using the given functions.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler instance context. |
| | *thread* | The thread for the operation. |
| in,out | *node* | The node to update the priority of. |
| | *extract_from_ready* | Function to extract a node from the ready queue of the scheduler context. |
| | *update* | Function to update the priority of a node in the scheduler context. |
| | *enqueue* | Function to enqueue a node with a given priority. |
| | *enqueue_scheduled* | Function to enqueue a scheduled node. |
| | *ask_for_help* | Function to perform a help request. |

Definition at line 1312 of file schedulersmpimpl.h.

### 8.130.3.39 _Scheduler_SMP_Withdraw_node()

```
static void _Scheduler_SMP_Withdraw_node (
              Scheduler_Context * context,
              Thread_Control * thread,
              Scheduler_Node * node,
              Thread_Scheduler_state next_state,
              Scheduler_SMP_Extract extract_from_ready,
              Scheduler_SMP_Get_highest_ready get_highest_ready,
              Scheduler_SMP_Move move_from_ready_to_scheduled,
              Scheduler_SMP_Allocate_processor allocate_processor )  [inline], [static]
```

Withdraws the node.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler context instance. |
| in,out | *thread* | The thread to change to *next_state*. |
| in,out | *node* | The node to withdraw. |
| | *next_state* | The new state for *thread*. |

**Parameters**

| | | |
|---|---|---|
| | *extract_from_ready* | Function to extract a node from the ready queue of the scheduler context. |
| | *get_highest_ready* | Function to get the highest ready node. |
| | *move_from_ready_to_scheduled* | Function to move a node from the set of ready nodes to the set of scheduled nodes. |
| | *allocate_processor* | Function to allocate a processor to a node based on the rules of the scheduler. |

Definition at line 1590 of file schedulersmpimpl.h.

#### 8.130.3.40 _Scheduler_SMP_Yield()

```
static void _Scheduler_SMP_Yield (
            Scheduler_Context * context,
            Thread_Control * thread,
            Scheduler_Node * node,
            Scheduler_SMP_Extract extract_from_ready,
            Scheduler_SMP_Enqueue enqueue,
            Scheduler_SMP_Enqueue enqueue_scheduled )  [inline], [static]
```

Performs a yield and asks for help if necessary.

**Parameters**

| | |
|---|---|
| *context* | The scheduler instance context. |
| *thread* | The thread for the operation. |
| *node* | The node of the thread that yields. |
| *extract_from_ready* | Function to extract a node from the ready queue of the scheduler context. |
| *enqueue* | Function to enqueue a node with a given priority. |
| *enqueue_scheduled* | Function to enqueue a scheduled node. |

Definition at line 1368 of file schedulersmpimpl.h.

# 8.131 SMP Support

SMP Support.

## Files

- file smp.h

  *SuperCore SMP Support API.*

## Macros

- #define SMP_MESSAGE_SHUTDOWN 0x1UL

  *SMP message to request a processor shutdown.*
- #define SMP_MESSAGE_PERFORM_JOBS 0x2UL

  *SMP message to perform per-processor jobs.*

## Typedefs

- typedef void(∗ **SMP_Action_handler**) (void ∗arg)

## Enumerations

- enum SMP_Fatal_code {
  **SMP_FATAL_BOOT_PROCESSOR_NOT_ASSIGNED_TO_SCHEDULER**, **SMP_FATAL_MANDATOR↩ Y_PROCESSOR_NOT_PRESENT**, **SMP_FATAL_MULTITASKING_START_ON_INVALID_PROCESS↩ OR**, **SMP_FATAL_MULTITASKING_START_ON_UNASSIGNED_PROCESSOR**,
  **SMP_FATAL_SHUTDOWN**, **SMP_FATAL_SHUTDOWN_RESPONSE**, **SMP_FATAL_START_OF_MAN↩ DATORY_PROCESSOR_FAILED**, **SMP_FATAL_SCHEDULER_PIN_OR_UNPIN_NOT_SUPPORTED**,
  **SMP_FATAL_WRONG_CPU_STATE_TO_PERFORM_JOBS** }

  *SMP fatal codes.*

## Functions

- static uint32_t **_SMP_Get_processor_maximum** (void)
- static uint32_t **_SMP_Get_current_processor** (void)
- static void _SMP_Fatal (SMP_Fatal_code code)

  *Terminates with the given code.*
- void _SMP_Handler_initialize (void)

  *Initializes SMP Handler.*
- RTEMS_NO_RETURN void _SMP_Start_multitasking_on_secondary_processor (Per_CPU_Control ∗cpu↩ _self)

  *Performs high-level initialization of a secondary processor and runs the application threads.*
- static long unsigned _SMP_Inter_processor_interrupt_handler (Per_CPU_Control ∗cpu_self)

  *Interrupts handler for inter-processor interrupts.*
- bool _SMP_Should_start_processor (uint32_t cpu_index)

  *Checks if the processor with the specified index should be started.*
- void _SMP_Send_message (uint32_t cpu_index, unsigned long message)

  *Sends an SMP message to a processor.*

- void _SMP_Send_message_broadcast (unsigned long message)

  *Sends an SMP message to all other online processors.*
- void _SMP_Send_message_multicast (const Processor_mask *targets, unsigned long message)

  *Sends an SMP message to a set of processors.*
- void _SMP_Multicast_action (const Processor_mask *targets, SMP_Action_handler handler, void *arg)

  *Initiates an SMP multicast action to the set of target processors.*
- void _SMP_Broadcast_action (SMP_Action_handler handler, void *arg)

  *Initiates an SMP multicast action to the set of all online processors.*
- void _SMP_Othercast_action (SMP_Action_handler handler, void *arg)

  *Initiates an SMP multicast action to the set of all online processors excluding the current processor.*
- void _SMP_Unicast_action (uint32_t cpu_index, SMP_Action_handler handler, void *arg)

  *Initiates an SMP action on the specified target processor.*
- void _SMP_Synchronize (void)

  *Ensures that all store operations issued by the current processor before the call this function are visible to all other online processors.*
- void _SMP_Request_start_multitasking (void)

  *Requests a multitasking start on all configured and available processors.*
- void _SMP_Request_shutdown (void)

  *Requests a shutdown of all processors.*
- static __inline__ const Processor_mask * _SMP_Get_online_processors (void)

  *Gets all online processors.*
- static __inline__ const bool _SMP_Need_inter_processor_interrupts (void)

  *Indicate if inter-processor interrupts are needed.*

## Variables

- const uint32_t _SMP_Processor_configured_maximum

  *The configured processor maximum.*
- uint32_t **_SMP_Processor_maximum**
- Processor_mask _SMP_Online_processors

  *Set of online processors.*

### 8.131.1 Detailed Description

SMP Support.

This defines the interface of the SuperCore SMP support.

### 8.131.2 Macro Definition Documentation

#### 8.131.2.1 SMP_MESSAGE_PERFORM_JOBS

```
#define SMP_MESSAGE_PERFORM_JOBS 0x2UL
```

SMP message to perform per-processor jobs.

**See also**

_SMP_Send_message().

Definition at line 50 of file smpimpl.h.

#### 8.131.2.2 SMP_MESSAGE_SHUTDOWN

```
#define SMP_MESSAGE_SHUTDOWN 0x1UL
```

SMP message to request a processor shutdown.

**See also**

_SMP_Send_message().

Definition at line 43 of file smpimpl.h.

### 8.131.3 Function Documentation

#### 8.131.3.1 _SMP_Broadcast_action()

```
void _SMP_Broadcast_action (
             SMP_Action_handler handler,
             void * arg )
```

Initiates an SMP multicast action to the set of all online processors.

Simply calls _SMP_Multicast_action() with _SMP_Get_online_processors() as the target processor set.

**Parameters**

| | |
|---|---|
| *handler* | The multicast action handler. |
| *arg* | The multicast action argument. |

Definition at line 202 of file smpmulticastaction.c.

#### 8.131.3.2 _SMP_Fatal()

```
static void _SMP_Fatal (
            SMP_Fatal_code code )  [inline], [static]
```

Terminates with the given code.

**Parameters**

| | |
|---|---|
| *code* | The code for the termination. |

Definition at line 72 of file smpimpl.h.

#### 8.131.3.3 _SMP_Get_online_processors()

```
static __inline__ const Processor_mask* _SMP_Get_online_processors (
            void  )  [static]
```

Gets all online processors.

**Returns**

The processor mask with all online processors.

Definition at line 318 of file smpimpl.h.

#### 8.131.3.4 _SMP_Handler_initialize()

```
void _SMP_Handler_initialize (
            void  )
```

Initializes SMP Handler.

This method initialize the SMP Handler.

Definition at line 113 of file smp.c.

#### 8.131.3.5 _SMP_Inter_processor_interrupt_handler()

```
static long unsigned _SMP_Inter_processor_interrupt_handler (
            Per_CPU_Control * cpu_self )  [inline], [static]
```

Interrupts handler for inter-processor interrupts.

**Parameters**

| | | |
|---|---|---|
| in,out | *cpu_self* | The cpu control for the operation. |

**Returns**

The received message.

Definition at line 138 of file smpimpl.h.

#### 8.131.3.6 _SMP_Multicast_action()

```
void _SMP_Multicast_action (
             const Processor_mask * targets,
             SMP_Action_handler handler,
             void * arg )
```

Initiates an SMP multicast action to the set of target processors.

The current processor may be part of the set. The caller must ensure that no thread dispatch can happen during the call of this function, otherwise the behaviour is undefined. In case a target processor is in a wrong state to process per-processor jobs, then this function results in an SMP_FATAL_WRONG_CPU_STATE_TO_PERFORM_JOBS fatal SMP error.

**Parameters**

| | |
|---|---|
| *targets* | The set of target processors for the action. |
| *handler* | The multicast action handler. |
| *arg* | The multicast action argument. |

Definition at line 183 of file smpmulticastaction.c.

#### 8.131.3.7 _SMP_Need_inter_processor_interrupts()

```
static __inline__ const bool _SMP_Need_inter_processor_interrupts (
             void  )  [static]
```

Indicate if inter-processor interrupts are needed.

**Returns**

True if inter-processor interrupts are needed for the correct system operation, otherwise false.

Definition at line 333 of file smpimpl.h.

#### 8.131.3.8 _SMP_Othercast_action()

```
void _SMP_Othercast_action (
             SMP_Action_handler handler,
             void * arg )
```

Initiates an SMP multicast action to the set of all online processors excluding the current processor.

Simply calls _SMP_Multicast_action() with _SMP_Get_online_processors() as the target processor set excluding the current processor.

**Parameters**

| | |
|---|---|
| *handler* | The multicast action handler. |
| *arg* | The multicast action argument. |

Definition at line 210 of file smpmulticastaction.c.

#### 8.131.3.9 _SMP_Request_shutdown()

```
void _SMP_Request_shutdown (
             void  )
```

Requests a shutdown of all processors.

This function is a part of the system termination procedure.

**See also**

_Terminate().

Definition at line 205 of file smp.c.

#### 8.131.3.10 _SMP_Send_message()

```
void _SMP_Send_message (
             uint32_t cpu_index,
             unsigned long message )
```

Sends an SMP message to a processor.

The target processor may be the sending processor.

**Parameters**

| | |
|---|---|
| *cpu_index* | The target processor of the message. |
| *message* | The message to send. |

Definition at line 215 of file smp.c.

#### 8.131.3.11 _SMP_Send_message_broadcast()

```
void _SMP_Send_message_broadcast (
             unsigned long message )
```

Sends an SMP message to all other online processors.

**Parameters**

| | |
|---|---|
| *message* | The message to send. |

Definition at line 224 of file smp.c.

#### 8.131.3.12 _SMP_Send_message_multicast()

```
void _SMP_Send_message_multicast (
             const Processor_mask * targets,
             unsigned long message )
```

Sends an SMP message to a set of processors.

The sending processor may be part of the set.

**Parameters**

| | |
|---|---|
| *targets* | The set of processors to send the message. |
| *message* | The message to send. |

Definition at line 244 of file smp.c.

#### 8.131.3.13 _SMP_Should_start_processor()

```
bool _SMP_Should_start_processor (
             uint32_t cpu_index )
```

Checks if the processor with the specified index should be started.

**Parameters**

| | |
|---|---|
| *cpu_index* | The processor index. |

**Return values**

| | |
|---|---|
| *true* | The processor should be started. |
| *false* | The processor should not be started. |

Definition at line 176 of file smp.c.

#### 8.131.3.14 _SMP_Start_multitasking_on_secondary_processor()

```
RTEMS_NO_RETURN void _SMP_Start_multitasking_on_secondary_processor (
            Per_CPU_Control * cpu_self )
```

Performs high-level initialization of a secondary processor and runs the application threads.

The low-level initialization code must call this function to hand over the control of this processor to RTEMS. Interrupts must be disabled. It must be possible to send inter-processor interrupts to this processor. Since interrupts are disabled the inter-processor interrupt delivery is postponed until interrupts are enabled the first time. Interrupts are enabled during the execution begin of threads in case they have interrupt level zero (this is the default).

The pre-requisites for the call to this function are

- disabled interrupts,
- delivery of inter-processor interrupts is possible,
- a valid stack pointer and enough stack space,
- a valid code memory, and
- a valid BSS section.

This function must not be called by the main processor. The main processor uses _Thread_Start_multitasking() instead.

This function does not return to the caller.

**Parameters**

| | |
|---|---|
| *cpu_self* | The current processor control. |

Definition at line 184 of file smp.c.

#### 8.131.3.15 _SMP_Synchronize()

```
void _SMP_Synchronize (
            void  )
```

Ensures that all store operations issued by the current processor before the call this function are visible to all other online processors.

Simply calls _SMP_Othercast_action() with an empty multicast action.

Definition at line 227 of file smpmulticastaction.c.

#### 8.131.3.16 _SMP_Unicast_action()

```
void _SMP_Unicast_action (
              uint32_t cpu_index,
              SMP_Action_handler handler,
              void * arg )
```

Initiates an SMP action on the specified target processor.

This is an optimized variant of _SMP_Multicast_action().

**Parameters**

| | |
|---|---|
| *cpu_index* | The index of the target processor. |
| *handler* | The action handler. |
| *arg* | The action argument. |

Definition at line 34 of file smpunicastaction.c.

### 8.131.4 Variable Documentation

#### 8.131.4.1 _SMP_Online_processors

```
Processor_mask _SMP_Online_processors  [extern]
```

Set of online processors.

A processor is online if was started during system initialization. In this case its corresponding bit in the mask is set.

**See also**

_SMP_Handler_initialize().

Definition at line 36 of file smp.c.

#### 8.131.4.2 _SMP_Processor_configured_maximum

```
const uint32_t _SMP_Processor_configured_maximum  [extern]
```

The configured processor maximum.

In SMP configurations, this constant is defined by the application configuration via <rtems/confdefs.h>, otherwise it is a compile-time constant with the value one.

## 8.132 SPARC

SPARC Architecture Support.

### Modules

- SPARC Assembler Support

  *SPARC Assembler Support.*
- SPARC Context Structures

### Classes

- struct CPU_Per_CPU_control

### Macros

- #define SPARC_MINIMUM_STACK_FRAME_SIZE 0x60
- #define ISF_PSR_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x00
- #define ISF_PC_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x04
- #define ISF_NPC_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x08
- #define ISF_G1_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x0c
- #define ISF_G2_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x10
- #define ISF_G3_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x14
- #define ISF_G4_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x18
- #define ISF_G5_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x1c
- #define ISF_G7_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x24
- #define ISF_I0_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x28
- #define ISF_I1_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x2c
- #define ISF_I2_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x30
- #define ISF_I3_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x34
- #define ISF_I4_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x38
- #define ISF_I5_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x3c
- #define ISF_I6_FP_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x40
- #define ISF_I7_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x44
- #define ISF_Y_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x48
- #define ISF_TPC_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x4c
- #define CPU_INTERRUPT_FRAME_SIZE SPARC_MINIMUM_STACK_FRAME_SIZE + 0x50
- #define **SPARC_FP_CONTEXT_OFFSET_F0_F1** 0
- #define **SPARC_FP_CONTEXT_OFFSET_F2_F3** 8
- #define **SPARC_FP_CONTEXT_OFFSET_F4_F5** 16
- #define **SPARC_FP_CONTEXT_OFFSET_F6_F7** 24
- #define **SPARC_FP_CONTEXT_OFFSET_F8_F9** 32
- #define **SPARC_FP_CONTEXT_OFFSET_F10_F11** 40
- #define **SPARC_FP_CONTEXT_OFFSET_F12_F13** 48
- #define **SPARC_FP_CONTEXT_OFFSET_F14_F15** 56
- #define **SPARC_FP_CONTEXT_OFFSET_F16_F17** 64
- #define **SPARC_FP_CONTEXT_OFFSET_F18_F19** 72
- #define **SPARC_FP_CONTEXT_OFFSET_F20_F21** 80
- #define **SPARC_FP_CONTEXT_OFFSET_F22_F23** 88
- #define **SPARC_FP_CONTEXT_OFFSET_F24_F25** 96
- #define **SPARC_FP_CONTEXT_OFFSET_F26_F27** 104
- #define **SPARC_FP_CONTEXT_OFFSET_F28_F29** 112
- #define **SPARC_FP_CONTEXT_OFFSET_F30_F31** 120
- #define **SPARC_FP_CONTEXT_OFFSET_FSR** 128
- #define **CPU_PER_CPU_CONTROL_SIZE** 0
- #define **_CPU_Get_current_per_CPU_control**() _SPARC_Per_CPU_current
- #define **_CPU_Get_thread_executing**() ( _SPARC_Per_CPU_current->executing )

## Functions

- void **_CPU_Context_volatile_clobber** (uintptr_t pattern)
- void **_CPU_Context_validate** (uintptr_t pattern)
- RTEMS_INLINE_ROUTINE void **_CPU_Instruction_illegal** (void)
- RTEMS_INLINE_ROUTINE void **_CPU_Instruction_no_operation** (void)

## Variables

- register struct Per_CPU_Control * _SPARC_Per_CPU_current

  *The pointer to the current per-CPU control is available via register g6.*

### 8.132.1 Detailed Description

SPARC Architecture Support.

### 8.132.2 Macro Definition Documentation

#### 8.132.2.1 CPU_INTERRUPT_FRAME_SIZE

```
#define CPU_INTERRUPT_FRAME_SIZE SPARC_MINIMUM_STACK_FRAME_SIZE + 0x50
```

This defines the size of the ISF area for use in assembly.

Definition at line 78 of file cpuimpl.h.

#### 8.132.2.2 ISF_G1_OFFSET

```
#define ISF_G1_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x0c
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 45 of file cpuimpl.h.

#### 8.132.2.3 ISF_G2_OFFSET

```
#define ISF_G2_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x10
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 47 of file cpuimpl.h.

#### 8.132.2.4 ISF_G3_OFFSET

```
#define ISF_G3_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x14
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 49 of file cpuimpl.h.

#### 8.132.2.5 ISF_G4_OFFSET

```
#define ISF_G4_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x18
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 51 of file cpuimpl.h.

#### 8.132.2.6 ISF_G5_OFFSET

```
#define ISF_G5_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x1c
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 53 of file cpuimpl.h.

#### 8.132.2.7 ISF_G7_OFFSET

```
#define ISF_G7_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x24
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 55 of file cpuimpl.h.

#### 8.132.2.8 ISF_I0_OFFSET

```
#define ISF_I0_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x28
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 57 of file cpuimpl.h.

#### 8.132.2.9 ISF_I1_OFFSET

```
#define ISF_I1_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x2c
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 59 of file cpuimpl.h.

#### 8.132.2.10 ISF_I2_OFFSET

```
#define ISF_I2_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x30
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 61 of file cpuimpl.h.

#### 8.132.2.11 ISF_I3_OFFSET

```
#define ISF_I3_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x34
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 63 of file cpuimpl.h.

#### 8.132.2.12 ISF_I4_OFFSET

```
#define ISF_I4_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x38
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 65 of file cpuimpl.h.

#### 8.132.2.13 ISF_I5_OFFSET

```
#define ISF_I5_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x3c
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 67 of file cpuimpl.h.

#### 8.132.2.14 ISF_I6_FP_OFFSET

```
#define ISF_I6_FP_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x40
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 69 of file cpuimpl.h.

#### 8.132.2.15 ISF_I7_OFFSET

```
#define ISF_I7_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x44
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 71 of file cpuimpl.h.

#### 8.132.2.16 ISF_NPC_OFFSET

```
#define ISF_NPC_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x08
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 43 of file cpuimpl.h.

#### 8.132.2.17 ISF_PC_OFFSET

```
#define ISF_PC_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x04
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 41 of file cpuimpl.h.

#### 8.132.2.18 ISF_PSR_OFFSET

```
#define ISF_PSR_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x00
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 39 of file cpuimpl.h.

#### 8.132.2.19 ISF_TPC_OFFSET

```
#define ISF_TPC_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x4c
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 75 of file cpuimpl.h.

#### 8.132.2.20 ISF_Y_OFFSET

```
#define ISF_Y_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x48
```

This macro defines an offset into the ISF for use in assembly.

Definition at line 73 of file cpuimpl.h.

#### 8.132.2.21 SPARC_MINIMUM_STACK_FRAME_SIZE

```
#define SPARC_MINIMUM_STACK_FRAME_SIZE 0x60
```

This defines the size of the minimum stack frame.

Definition at line 32 of file cpuimpl.h.

## 8.133 SPARC

SPARC Board Support Packages.

### Modules

- LEON3 and LEON4

  *LEON3 and LEON4 Board Support Package.*
- Shared

  *Shared Support for SPARC Board Support Packages.*

### 8.133.1 Detailed Description

SPARC Board Support Packages.

## 8.134 SPARC Assembler Support

SPARC Assembler Support.

### Macros

- #define **__USER_LABEL_PREFIX__** _
- #define **__REGISTER_PREFIX__**
- #define **SYM**(x) RTEMS_XCONCAT(__USER_LABEL_PREFIX__, x)
- #define **REG**(x) RTEMS_XCONCAT(__REGISTER_PREFIX__, x)
- #define **BEGIN_CODE_DCL** .text
- #define **END_CODE_DCL**
- #define **BEGIN_DATA_DCL** .data
- #define **END_DATA_DCL**
- #define **BEGIN_CODE** .text
- #define **END_CODE**
- #define **BEGIN_DATA**
- #define **END_DATA**
- #define **BEGIN_BSS**
- #define **END_BSS**
- #define **END**
- #define **PUBLIC**(sym) .globl SYM (sym)
- #define **EXTERN**(sym) .globl SYM (sym)
- #define **TRAP**(_vector, _handler)
- #define **RTRAP**(_vector, _handler)

### 8.134.1 Detailed Description

SPARC Assembler Support.

### 8.134.2 Macro Definition Documentation

#### 8.134.2.1 RTRAP

```
#define RTRAP(
              _vector,
              _handler )
```

**Value:**

```
  mov   %g0, %l0 ; \
  sethi %hi(_handler), %l4 ; \
  jmp   %l4+%lo(_handler); \
  mov   _vector, %l3
```

Definition at line 122 of file asm.h.

#### 8.134.2.2 TRAP

```
#define TRAP(
              _vector,
              _handler )
```

**Value:**

```
  mov   %psr, %l0 ; \
  sethi %hi(_handler), %l4 ; \
  jmp   %l4+%lo(_handler); \
  mov   _vector, %l3
```

Definition at line 112 of file asm.h.

## 8.135 SPARC Context Structures

### Classes

- struct Context_Control

  *SPARC basic context.*
- struct Context_Control_fp

  *SPARC basic context.*

### Macros

- #define _CPU_Context_Get_SP(_context) (_context)->o6_sp
- #define G5_OFFSET 0x00
- #define G7_OFFSET 0x04
- #define L0_OFFSET 0x08
- #define L1_OFFSET 0x0C
- #define L2_OFFSET 0x10
- #define L3_OFFSET 0x14
- #define L4_OFFSET 0x18
- #define L5_OFFSET 0x1C
- #define L6_OFFSET 0x20
- #define L7_OFFSET 0x24
- #define I0_OFFSET 0x28
- #define I1_OFFSET 0x2C
- #define I2_OFFSET 0x30
- #define I3_OFFSET 0x34
- #define I4_OFFSET 0x38
- #define I5_OFFSET 0x3C
- #define I6_FP_OFFSET 0x40
- #define I7_OFFSET 0x44
- #define O6_SP_OFFSET 0x48
- #define O7_OFFSET 0x4C
- #define PSR_OFFSET 0x50
- #define ISR_DISPATCH_DISABLE_STACK_OFFSET 0x54
- #define **SPARC_CONTEXT_CONTROL_IS_EXECUTING_OFFSET** 0x58
- #define FO_F1_OFFSET 0x00
- #define F2_F3_OFFSET 0x08
- #define F4_F5_OFFSET 0x10
- #define F6_F7_OFFSET 0x18
- #define F8_F9_OFFSET 0x20
- #define F1O_F11_OFFSET 0x28
- #define F12_F13_OFFSET 0x30
- #define F14_F15_OFFSET 0x38
- #define F16_F17_OFFSET 0x40
- #define F18_F19_OFFSET 0x48
- #define F2O_F21_OFFSET 0x50
- #define F22_F23_OFFSET 0x58
- #define F24_F25_OFFSET 0x60
- #define F26_F27_OFFSET 0x68
- #define F28_F29_OFFSET 0x70
- #define F3O_F31_OFFSET 0x78
- #define FSR_OFFSET 0x80
- #define CONTEXT_CONTROL_FP_SIZE 0x84

### Typedefs

- typedef struct Context_Control_fp **Context_Control_fp**

### Functions

- static bool **_CPU_Context_Get_is_executing** (const Context_Control *context)
- static void **_CPU_Context_Set_is_executing** (Context_Control *context, bool is_executing)

## 8.135.1 Detailed Description

Generally there are 2 types of context to save.

- Interrupt registers to save
- Task level registers to save

This means we have the following 3 context items:

- task level context stuff:: Context_Control
- floating point task stuff:: Context_Control_fp
- special interrupt level context :: Context_Control_interrupt

On the SPARC, we are relatively conservative in that we save most of the CPU state in the context area. The ET (enable trap) bit and the CWP (current window pointer) fields of the PSR are considered system wide resources and are not maintained on a per-thread basis.

## 8.135.2 Macro Definition Documentation

### 8.135.2.1 _CPU_Context_Get_SP

```
#define _CPU_Context_Get_SP(
              _context )   (_context)->o6_sp
```

This macro provides a CPU independent way for RTEMS to access the stack pointer in a context structure. The actual name and offset is CPU architecture dependent.

Definition at line 393 of file cpu.h.

#### 8.135.2.2 CONTEXT_CONTROL_FP_SIZE

```
#define CONTEXT_CONTROL_FP_SIZE 0x84
```

This defines the size of the FPU context area for use in assembly.

Definition at line 557 of file cpu.h.

#### 8.135.2.3 F12_F13_OFFSET

```
#define F12_F13_OFFSET 0x30
```

This macro defines an offset into the FPU context for use in assembly.

Definition at line 534 of file cpu.h.

#### 8.135.2.4 F14_F15_OFFSET

```
#define F14_F15_OFFSET 0x38
```

This macro defines an offset into the FPU context for use in assembly.

Definition at line 536 of file cpu.h.

#### 8.135.2.5 F16_F17_OFFSET

```
#define F16_F17_OFFSET 0x40
```

This macro defines an offset into the FPU context for use in assembly.

Definition at line 538 of file cpu.h.

#### 8.135.2.6 F18_F19_OFFSET

```
#define F18_F19_OFFSET 0x48
```

This macro defines an offset into the FPU context for use in assembly.

Definition at line 540 of file cpu.h.

#### 8.135.2.7 F1O_F11_OFFSET

```
#define F1O_F11_OFFSET 0x28
```

This macro defines an offset into the FPU context for use in assembly.

Definition at line 532 of file cpu.h.

#### 8.135.2.8 F22_F23_OFFSET

```
#define F22_F23_OFFSET 0x58
```

This macro defines an offset into the FPU context for use in assembly.

Definition at line 544 of file cpu.h.

#### 8.135.2.9 F24_F25_OFFSET

```
#define F24_F25_OFFSET 0x60
```

This macro defines an offset into the FPU context for use in assembly.

Definition at line 546 of file cpu.h.

#### 8.135.2.10 F26_F27_OFFSET

```
#define F26_F27_OFFSET 0x68
```

This macro defines an offset into the FPU context for use in assembly.

Definition at line 548 of file cpu.h.

#### 8.135.2.11 F28_F29_OFFSET

```
#define F28_F29_OFFSET 0x70
```

This macro defines an offset into the FPU context for use in assembly.

Definition at line 550 of file cpu.h.

#### 8.135.2.12 F2_F3_OFFSET

```
#define F2_F3_OFFSET 0x08
```

This macro defines an offset into the FPU context for use in assembly.

Definition at line 524 of file cpu.h.

#### 8.135.2.13 F2O_F21_OFFSET

```
#define F2O_F21_OFFSET 0x50
```

This macro defines an offset into the FPU context for use in assembly.

Definition at line 542 of file cpu.h.

#### 8.135.2.14 F3O_F31_OFFSET

```
#define F3O_F31_OFFSET 0x78
```

This macro defines an offset into the FPU context for use in assembly.

Definition at line 552 of file cpu.h.

#### 8.135.2.15 F4_F5_OFFSET

```
#define F4_F5_OFFSET 0x10
```

This macro defines an offset into the FPU context for use in assembly.

Definition at line 526 of file cpu.h.

#### 8.135.2.16 F6_F7_OFFSET

```
#define F6_F7_OFFSET 0x18
```

This macro defines an offset into the FPU context for use in assembly.

Definition at line 528 of file cpu.h.

#### 8.135.2.17 F8_F9_OFFSET

```
#define F8_F9_OFFSET 0x20
```

This macro defines an offset into the FPU context for use in assembly.

Definition at line 530 of file cpu.h.

#### 8.135.2.18 FO_F1_OFFSET

```
#define FO_F1_OFFSET 0x00
```

This macro defines an offset into the FPU context for use in assembly.

Definition at line 522 of file cpu.h.

#### 8.135.2.19 FSR_OFFSET

```
#define FSR_OFFSET 0x80
```

This macro defines an offset into the FPU context for use in assembly.

Definition at line 554 of file cpu.h.

#### 8.135.2.20 G5_OFFSET

```
#define G5_OFFSET 0x00
```

This macro defines an offset into the context for use in assembly.

Definition at line 420 of file cpu.h.

#### 8.135.2.21 G7_OFFSET

```
#define G7_OFFSET 0x04
```

This macro defines an offset into the context for use in assembly.

Definition at line 422 of file cpu.h.

#### 8.135.2.22 I0_OFFSET

```
#define I0_OFFSET 0x28
```

This macro defines an offset into the context for use in assembly.

Definition at line 442 of file cpu.h.

#### 8.135.2.23 I1_OFFSET

```
#define I1_OFFSET 0x2C
```

This macro defines an offset into the context for use in assembly.

Definition at line 444 of file cpu.h.

#### 8.135.2.24 I2_OFFSET

```
#define I2_OFFSET 0x30
```

This macro defines an offset into the context for use in assembly.

Definition at line 446 of file cpu.h.

#### 8.135.2.25 I3_OFFSET

```
#define I3_OFFSET 0x34
```

This macro defines an offset into the context for use in assembly.

Definition at line 448 of file cpu.h.

#### 8.135.2.26 I4_OFFSET

```
#define I4_OFFSET 0x38
```

This macro defines an offset into the context for use in assembly.

Definition at line 450 of file cpu.h.

#### 8.135.2.27 I5_OFFSET

```
#define I5_OFFSET 0x3C
```

This macro defines an offset into the context for use in assembly.

Definition at line 452 of file cpu.h.

#### 8.135.2.28 I6_FP_OFFSET

```
#define I6_FP_OFFSET 0x40
```

This macro defines an offset into the context for use in assembly.

Definition at line 454 of file cpu.h.

#### 8.135.2.29 I7_OFFSET

```
#define I7_OFFSET 0x44
```

This macro defines an offset into the context for use in assembly.

Definition at line 456 of file cpu.h.

#### 8.135.2.30 ISR_DISPATCH_DISABLE_STACK_OFFSET

```
#define ISR_DISPATCH_DISABLE_STACK_OFFSET 0x54
```

This macro defines an offset into the context for use in assembly.

Definition at line 466 of file cpu.h.

#### 8.135.2.31 L0_OFFSET

```
#define L0_OFFSET 0x08
```

This macro defines an offset into the context for use in assembly.

Definition at line 425 of file cpu.h.

#### 8.135.2.32 L1_OFFSET

```
#define L1_OFFSET 0x0C
```

This macro defines an offset into the context for use in assembly.

Definition at line 427 of file cpu.h.

#### 8.135.2.33 L2_OFFSET

```
#define L2_OFFSET 0x10
```

This macro defines an offset into the context for use in assembly.

Definition at line 429 of file cpu.h.

#### 8.135.2.34 L3_OFFSET

```
#define L3_OFFSET 0x14
```

This macro defines an offset into the context for use in assembly.

Definition at line 431 of file cpu.h.

#### 8.135.2.35 L4_OFFSET

```
#define L4_OFFSET 0x18
```

This macro defines an offset into the context for use in assembly.

Definition at line 433 of file cpu.h.

#### 8.135.2.36 L5_OFFSET

```
#define L5_OFFSET 0x1C
```

This macro defines an offset into the context for use in assembly.

Definition at line 435 of file cpu.h.

#### 8.135.2.37 L6_OFFSET

```
#define L6_OFFSET 0x20
```

This macro defines an offset into the context for use in assembly.

Definition at line 437 of file cpu.h.

#### 8.135.2.38 L7_OFFSET

```
#define L7_OFFSET 0x24
```

This macro defines an offset into the context for use in assembly.

Definition at line 439 of file cpu.h.

#### 8.135.2.39 O6_SP_OFFSET

```
#define O6_SP_OFFSET 0x48
```

This macro defines an offset into the context for use in assembly.

Definition at line 459 of file cpu.h.

#### 8.135.2.40 O7_OFFSET

```
#define O7_OFFSET 0x4C
```

This macro defines an offset into the context for use in assembly.

Definition at line 461 of file cpu.h.

#### 8.135.2.41 PSR_OFFSET

```
#define PSR_OFFSET 0x50
```

This macro defines an offset into the context for use in assembly.

Definition at line 464 of file cpu.h.

# 8.136 Scheduler Handler

This handler encapsulates functionality related to managing sets of threads that are ready for execution.

## Modules

- Deterministic Priority Scheduler

  *Deterministic Priority Scheduler.*
- EDF Scheduler

  *EDF Scheduler.*
- SMP Scheduler

  *SMP Scheduler.*
- Simple Priority Scheduler

  *Simple Priority Scheduler.*

## Files

- file scheduler.h

  *Constants and Structures Associated with the Scheduler.*
- file schedulerimpl.h

  *Inlined Routines Associated with the Manipulation of the Scheduler.*
- file schedulernode.h

  *Handles Scheduler Nodes.*
- file schedulernodeimpl.h

  *Scheduler Node Implementation.*
- file scheduler.c

  *Scheduler Initialize.*
- file schedulerdefaultnodedestroy.c

  *Scheduler Default Node Destruction Operation.*
- file schedulerdefaultnodeinit.c

  *Scheduler Default Node Initialization Operation.*
- file schedulerdefaultreleasejob.c

  *Default Scheduler Release Job Operation.*
- file schedulerdefaultsetaffinity.c

  *Scheduler Default Set Affinity Operation.*
- file schedulerdefaulttick.c

  *Default Scheduler At Tick Handler.*
- file scheduleredfreleasejob.c

  *Scheduler EDF Release Job.*
- file schedulerpriorityblock.c

  *Scheduler Priority Block.*
- file schedulerprioritychangepriority.c

  *Removes Thread from Thread Queue.*
- file schedulerpriorityschedule.c

  *Priority Scheduler Schedule Method.*
- file schedulerpriorityunblock.c

  *Scheduler Priority Unblock.*
- file schedulerpriorityyield.c

  *Scheduler Priority Yield.*

## Classes

- struct Scheduler_Operations

  *The scheduler operations.*
- struct Scheduler_Context

  *Scheduler context.*
- struct _Scheduler_Control

  *Scheduler control.*
- struct Scheduler_Assignment

  *Scheduler assignment.*
- struct Scheduler_Node

  *Scheduler node for per-thread data.*

## Macros

- #define SCHEDULER_ASSIGN_DEFAULT UINT32_C(0x0)

  *The scheduler assignment default attributes.*
- #define SCHEDULER_ASSIGN_PROCESSOR_OPTIONAL SCHEDULER_ASSIGN_DEFAULT

  *The presence of this processor is optional.*
- #define SCHEDULER_ASSIGN_PROCESSOR_MANDATORY UINT32_C(0x1)

  *The presence of this processor is mandatory.*
- #define **SCHEDULER_OPERATION_DEFAULT_ASK_FOR_HELP**
- #define **SCHEDULER_OPERATION_DEFAULT_GET_SET_AFFINITY** , _Scheduler_default_Set_affinity
- #define PRIORITY_MAXIMUM ( _Scheduler_Table[ 0 ].maximum_priority )

  *This defines the lowest (least important) thread priority of the first scheduler instance.*
- #define **SCHEDULER_NODE_OF_THREAD_WAIT_NODE**(node) RTEMS_CONTAINER_OF( node, Scheduler_Node, Thread.Wait_node )
- #define **SCHEDULER_NODE_OF_THREAD_SCHEDULER_NODE**(node) RTEMS_CONTAINER_OF( node, Scheduler_Node, Thread.Scheduler_node.Chain )
- #define **SCHEDULER_NODE_OF_WAIT_PRIORITY_NODE**(node) RTEMS_CONTAINER_OF( node, Scheduler_Node, Wait.Priority.Node.Node.Chain )
- #define **SCHEDULER_NODE_OF_WAIT_PRIORITY**(node) RTEMS_CONTAINER_OF( node, Scheduler_Node, Wait.Priority )
- #define SCHEDULER_PRIORITY_APPEND_FLAG 1

  *Priority append indicator for the priority control used for the scheduler node priority.*
- #define SCHEDULER_PRIORITY_MAP(priority) ( ( priority ) << 1 )

  *Maps a priority value to support the append indicator.*
- #define SCHEDULER_PRIORITY_UNMAP(priority) ( ( priority ) >> 1 )

  *Returns the plain priority value.*
- #define SCHEDULER_PRIORITY_PURIFY(priority) ( ( priority ) & ~( (Priority_Control) SCHEDULER_PRIORITY_APPEND_F ) )

  *Clears the priority append indicator bit.*
- #define SCHEDULER_PRIORITY_APPEND(priority) ( ( priority ) | SCHEDULER_PRIORITY_APPEND_FLAG )

  *Returns the priority control with the append indicator bit set.*
- #define SCHEDULER_PRIORITY_IS_APPEND(priority) ( ( ( priority ) & SCHEDULER_PRIORITY_APPEND_FLAG ) != 0 )

  *Returns true, if the item should be appended to its priority group, otherwise returns false and the item should be prepended to its priority group.*

## Typedefs

- typedef struct _Scheduler_Control **Scheduler_Control**
- typedef struct Scheduler_Context Scheduler_Context

  *Scheduler context.*
- typedef Thread_Control *(* Scheduler_Get_idle_thread) (Scheduler_Context *context)

  *Gets an idle thread from the scheduler instance.*
- typedef void(* Scheduler_Release_idle_thread) (Scheduler_Context *context, Thread_Control *idle)

  *Releases an idle thread to the scheduler instance for reuse.*
- typedef struct Scheduler_Node **Scheduler_Node**

## Enumerations

- enum Scheduler_Try_to_schedule_action { **SCHEDULER_TRY_TO_SCHEDULE_DO_SCHEDULE**, **SCHEDULER_TRY_TO_SCHEDULE_DO_IDLE_EXCHANGE**, **SCHEDULER_TRY_TO_SCHEDULE_DO_BLOCK** }

  *This enumeration defines what a scheduler should do with a node which could be scheduled.*
- enum Scheduler_Node_request { SCHEDULER_NODE_REQUEST_NOT_PENDING, SCHEDULER_NODE_REQUEST_ADD SCHEDULER_NODE_REQUEST_REMOVE, SCHEDULER_NODE_REQUEST_NOTHING }

  *The scheduler node requests.*

## Functions

- Priority_Control _Scheduler_default_Map_priority (const Scheduler_Control *scheduler, Priority_Control priority)

  *Returns the scheduler internal thread priority mapped by SCHEDULER_PRIORITY_MAP().*
- Priority_Control _Scheduler_default_Unmap_priority (const Scheduler_Control *scheduler, Priority_Control priority)

  *Returns the user visible thread priority unmapped by SCHEDULER_PRIORITY_UNMAP().*
- bool _Scheduler_default_Ask_for_help (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Does nothing.*
- void _Scheduler_default_Reconsider_help_request (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Does nothing.*
- void _Scheduler_default_Withdraw_node (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node, Thread_Scheduler_state next_state)

  *Does nothing.*
- void _Scheduler_default_Pin_or_unpin (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node, struct Per_CPU_Control *cpu)

  *Does nothing in a single processor system, otherwise a fatal error is issued.*
- void _Scheduler_default_Schedule (const Scheduler_Control *scheduler, Thread_Control *the_thread)

  *Does nothing.*
- void _Scheduler_default_Node_initialize (const Scheduler_Control *scheduler, Scheduler_Node *node, Thread_Control *the_thread, Priority_Control priority)

  *Performs the scheduler base node initialization.*
- void _Scheduler_default_Node_destroy (const Scheduler_Control *scheduler, Scheduler_Node *node)

  *Does nothing.*
- void _Scheduler_default_Release_job (const Scheduler_Control *scheduler, Thread_Control *the_thread, Priority_Node *priority_node, uint64_t deadline, Thread_queue_Context *queue_context)

  *Does nothing.*

- void _Scheduler_default_Cancel_job (const Scheduler_Control *scheduler, Thread_Control *the_thread, Priority_Node *priority_node, Thread_queue_Context *queue_context)

  *Does nothing.*
- void _Scheduler_default_Tick (const Scheduler_Control *scheduler, Thread_Control *executing)

  *Performs tick operations depending on the CPU budget algorithm for each executing thread.*
- void _Scheduler_default_Start_idle (const Scheduler_Control *scheduler, Thread_Control *the_thread, struct Per_CPU_Control *cpu)

  *Starts an idle thread.*
- bool _Scheduler_default_Set_affinity (const Scheduler_Control *scheduler, Thread_Control *thread, Scheduler_Node *node, const Processor_mask *affinity)

  *Checks if the processor set of the scheduler is the subset of the affinity set.*
- void _Scheduler_Handler_initialization (void)

  *Initializes the scheduler to the policy chosen by the user.*
- static __inline__ Scheduler_Context * _Scheduler_Get_context (const Scheduler_Control *scheduler)

  *Gets the context of the scheduler.*
- static __inline__ const Scheduler_Control * _Scheduler_Get_by_CPU (const Per_CPU_Control *cpu)

  *Gets the scheduler for the cpu.*
- static __inline__ void _Scheduler_Acquire_critical (const Scheduler_Control *scheduler, ISR_lock_Context *lock_context)

  *Acquires the scheduler instance inside a critical section (interrupts disabled).*
- static __inline__ void _Scheduler_Release_critical (const Scheduler_Control *scheduler, ISR_lock_Context *lock_context)

  *Releases the scheduler instance inside a critical section (interrupts disabled).*
- static __inline__ bool _Scheduler_Is_non_preempt_mode_supported (const Scheduler_Control *scheduler)

  *Indicate if the thread non-preempt mode is supported by the scheduler.*
- void **_Scheduler_Request_ask_for_help** (Thread_Control *the_thread)
- static __inline__ void _Scheduler_Ask_for_help (Thread_Control *the_thread)

  *Registers an ask for help request if necessary.*
- static __inline__ void _Scheduler_Schedule (Thread_Control *the_thread)

  *General scheduling decision.*
- static __inline__ void _Scheduler_Yield (Thread_Control *the_thread)

  *Scheduler yield with a particular thread.*
- static __inline__ void _Scheduler_Block (Thread_Control *the_thread)

  *Blocks a thread with respect to the scheduler.*
- static __inline__ void _Scheduler_Unblock (Thread_Control *the_thread)

  *Unblocks a thread with respect to the scheduler.*
- static __inline__ void _Scheduler_Update_priority (Thread_Control *the_thread)

  *Propagates a priority change of a thread to the scheduler.*
- static __inline__ void _Scheduler_Priority_and_sticky_update (Thread_Control *the_thread, int sticky_level_change)

  *Changes the sticky level of the home scheduler node and propagates a priority change of a thread to the scheduler.*
- static __inline__ Priority_Control _Scheduler_Map_priority (const Scheduler_Control *scheduler, Priority_Control priority)

  *Maps a thread priority from the user domain to the scheduler domain.*
- static __inline__ Priority_Control _Scheduler_Unmap_priority (const Scheduler_Control *scheduler, Priority_Control priority)

  *Unmaps a thread priority from the scheduler domain to the user domain.*
- static __inline__ void _Scheduler_Node_initialize (const Scheduler_Control *scheduler, Scheduler_Node *node, Thread_Control *the_thread, Priority_Control priority)

  *Initializes a scheduler node.*
- static __inline__ void _Scheduler_Node_destroy (const Scheduler_Control *scheduler, Scheduler_Node *node)

*Destroys a scheduler node.*

- static __inline__ void _Scheduler_Release_job (Thread_Control *the_thread, Priority_Node *priority_node, uint64_t deadline, Thread_queue_Context *queue_context)

  *Releases a job of a thread with respect to the scheduler.*

- static __inline__ void _Scheduler_Cancel_job (Thread_Control *the_thread, Priority_Node *priority_node, Thread_queue_Context *queue_context)

  *Cancels a job of a thread with respect to the scheduler.*

- static __inline__ void _Scheduler_Tick (const Per_CPU_Control *cpu)

  *Scheduler method invoked at each clock tick.*

- static __inline__ void _Scheduler_Start_idle (const Scheduler_Control *scheduler, Thread_Control *the_thread, Per_CPU_Control *cpu)

  *Starts the idle thread for a particular processor.*

- static __inline__ bool _Scheduler_Has_processor_ownership (const Scheduler_Control *scheduler, uint32_t cpu_index)

  *Checks if the scheduler of the cpu with the given index is equal to the given scheduler.*

- static __inline__ const Processor_mask * _Scheduler_Get_processors (const Scheduler_Control *scheduler)

  *Gets the processors of the scheduler.*

- bool _Scheduler_Get_affinity (Thread_Control *the_thread, size_t cpusetsize, cpu_set_t *cpuset)

  *Copies the thread's scheduler's affinity to the given cpuset.*

- static __inline__ bool _Scheduler_default_Set_affinity_body (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node, const Processor_mask *affinity)

  *Checks if the affinity is a subset of the online processors.*

- bool _Scheduler_Set_affinity (Thread_Control *the_thread, size_t cpusetsize, const cpu_set_t *cpuset)

  *Sets the thread's scheduler's affinity.*

- static __inline__ void _Scheduler_Generic_block (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node, void(*extract)(const Scheduler_Control *, Thread_Control *, Scheduler_Node *), void(*schedule)(const Scheduler_Control *, Thread_Control *, bool))

  *Blocks the thread.*

- static __inline__ uint32_t _Scheduler_Get_processor_count (const Scheduler_Control *scheduler)

  *Gets the number of processors of the scheduler.*

- static __inline__ Objects_Id _Scheduler_Build_id (uint32_t scheduler_index)

  *Builds an object build id.*

- static __inline__ uint32_t _Scheduler_Get_index_by_id (Objects_Id id)

  *Gets the scheduler index from the given object build id.*

- static __inline__ const Scheduler_Control * _Scheduler_Get_by_id (Objects_Id id)

  *Gets the scheduler from the given object build id.*

- static __inline__ uint32_t _Scheduler_Get_index (const Scheduler_Control *scheduler)

  *Gets the index of the scheduler.*

- static __inline__ void _Scheduler_Thread_change_state (Thread_Control *the_thread, Thread_Scheduler_state new_state)

  *Changes the threads state to the given new state.*

- static __inline__ void _Scheduler_Set_idle_thread (Scheduler_Node *node, Thread_Control *idle)

  *Sets the scheduler node's idle thread.*

- static __inline__ Thread_Control * _Scheduler_Use_idle_thread (Scheduler_Context *context, Scheduler_Node *node, Per_CPU_Control *cpu, Scheduler_Get_idle_thread get_idle_thread)

  *Uses an idle thread for this scheduler node.*

- static __inline__ Scheduler_Try_to_schedule_action _Scheduler_Try_to_schedule_node (Scheduler_Context *context, Scheduler_Node *node, const Thread_Control *idle, Scheduler_Get_idle_thread get_idle_thread)

  *Tries to schedule the scheduler node.*

- static __inline__ Thread_Control * _Scheduler_Release_idle_thread (Scheduler_Context *context, Scheduler_Node *node, Scheduler_Release_idle_thread release_idle_thread)

*Releases an idle thread using this scheduler node.*

- static __inline__ void _Scheduler_Exchange_idle_thread (Scheduler_Node *needs_idle, Scheduler_Node *uses_idle, Thread_Control *idle)

  *Exchanges an idle thread from the scheduler node that uses it right now to another scheduler node.*

- static __inline__ Per_CPU_Control * _Scheduler_Block_node (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node, bool is_scheduled, Scheduler_Get_idle_thread get_idle_thread)

  *Blocks this scheduler node.*

- static __inline__ void _Scheduler_Discard_idle_thread (Scheduler_Context *context, Thread_Control *the↩ _thread, Scheduler_Node *node, Scheduler_Release_idle_thread release_idle_thread)

  *Discard the idle thread from the scheduler node.*

- static __inline__ bool _Scheduler_Unblock_node (Scheduler_Context *context, Thread_Control *the_thread, Scheduler_Node *node, bool is_scheduled, Scheduler_Release_idle_thread release_idle_thread)

  *Unblocks this scheduler node.*

- static __inline__ void _Scheduler_Update_heir (Thread_Control *new_heir, bool force_dispatch)

  *Updates the heir.*

- static __inline__ Status_Control _Scheduler_Set (const Scheduler_Control *new_scheduler, Thread_Control *the_thread, Priority_Control priority)

  *Sets a new scheduler.*

- static __inline__ void _Scheduler_Node_do_initialize (const struct _Scheduler_Control *scheduler, Scheduler_Node *node, Thread_Control *the_thread, Priority_Control priority)

  *Initializes a node.*

- static __inline__ const Scheduler_Control * _Scheduler_Node_get_scheduler (const Scheduler_Node *node)

  *Gets the scheduler of the node.*

- static __inline__ Thread_Control * _Scheduler_Node_get_owner (const Scheduler_Node *node)

  *Gets the owner of the node.*

- static __inline__ Priority_Control _Scheduler_Node_get_priority (Scheduler_Node *node)

  *Gets the priority of the node.*

- static __inline__ void _Scheduler_Node_set_priority (Scheduler_Node *node, Priority_Control new_priority, bool prepend_it)

  *Sets the priority of the node.*

- static __inline__ Thread_Control * _Scheduler_Node_get_user (const Scheduler_Node *node)

  *Gets the user of the node.*

- static __inline__ void _Scheduler_Node_set_user (Scheduler_Node *node, Thread_Control *user)

  *Sets the user of the node.*

- static __inline__ Thread_Control * _Scheduler_Node_get_idle (const Scheduler_Node *node)

  *Gets the idle thread of the node.*

## Variables

- const Scheduler_Control _Scheduler_Table [ ]

  *This table contains the configured schedulers.*

- const size_t _Scheduler_Count

  *This constant contains the count of configured schedulers.*

- const Scheduler_Assignment _Scheduler_Initial_assignments [ ]

  *The scheduler assignments.*

- const size_t _Scheduler_Node_size

  *The size of a scheduler node.*

### 8.136.1 Detailed Description

This handler encapsulates functionality related to managing sets of threads that are ready for execution.

Schedulers are used by the system to manage sets of threads that are ready for execution. A scheduler consists of

- a scheduler algorithm implementation,
- a scheduler index and an associated name, and
- a set of processors owned by the scheduler (may be empty, but never overlaps with a set owned by another scheduler).

Each thread uses exactly one scheduler as its home scheduler. Threads may temporarily use another scheduler due to actions of locking protocols.

All properties of a scheduler can be configured and controlled by the user. Some properties are fixed at link time (defined by application configuration options), other properties can be changed at runtime through directive calls.

The scheduler index, name, and initial processor set are defined for a particular application by the application configuration. The schedulers are registered in the _Scheduler_Table which has _Scheduler_Count entries.

### 8.136.2 Macro Definition Documentation

#### 8.136.2.1 SCHEDULER_OPERATION_DEFAULT_ASK_FOR_HELP

```
#define SCHEDULER_OPERATION_DEFAULT_ASK_FOR_HELP
```

**Value:**

```
    _Scheduler_default_Ask_for_help, \
    _Scheduler_default_Reconsider_help_request, \
    _Scheduler_default_Withdraw_node, \
    _Scheduler_default_Pin_or_unpin, \
    _Scheduler_default_Pin_or_unpin, \
    NULL, \
    NULL,
```

Definition at line 462 of file scheduler.h.

### 8.136.3 Typedef Documentation

#### 8.136.3.1 Scheduler_Context

```
typedef struct Scheduler_Context Scheduler_Context
```

Scheduler context.

The scheduler context of a particular scheduler implementation must place this structure at the begin of its context structure.

#### 8.136.3.2 Scheduler_Get_idle_thread

```
typedef Thread_Control*( * Scheduler_Get_idle_thread) (Scheduler_Context *context)
```

Gets an idle thread from the scheduler instance.

**Parameters**

| | |
|---|---|
| *context* | The scheduler instance context. |

**Returns**

idle An idle thread for use. This function must always return an idle thread. If none is available, then this is a fatal error.

Definition at line 877 of file schedulerimpl.h.

#### 8.136.3.3 Scheduler_Release_idle_thread

```
typedef void( * Scheduler_Release_idle_thread) (Scheduler_Context *context, Thread_Control
*idle)
```

Releases an idle thread to the scheduler instance for reuse.

**Parameters**

| | |
|---|---|
| *context* | The scheduler instance context. |
| *idle* | The idle thread to release. |

Definition at line 887 of file schedulerimpl.h.

### 8.136.4 Enumeration Type Documentation

#### 8.136.4.1 Scheduler_Node_request

```
enum Scheduler_Node_request
```

The scheduler node requests.

**Enumerator**

| | |
|---|---|
| SCHEDULER_NODE_REQUEST_NOT_PENDING | The scheduler node is not on the list of pending requests. |
| SCHEDULER_NODE_REQUEST_ADD | There is a pending scheduler node request to add this scheduler node to the Thread_Control::Scheduler::Scheduler_nodes chain. |
| SCHEDULER_NODE_REQUEST_REMOVE | There is a pending scheduler node request to remove this scheduler node from the Thread_Control::Scheduler::Scheduler_nodes chain. |
| SCHEDULER_NODE_REQUEST_NOTHING | The scheduler node is on the list of pending requests, but nothing should change. |

Definition at line 47 of file schedulernode.h.

## 8.136.5 Function Documentation

### 8.136.5.1 _Scheduler_Acquire_critical()

```
static __inline__ void _Scheduler_Acquire_critical (
             const Scheduler_Control * scheduler,
             ISR_lock_Context * lock_context )  [static]
```

Acquires the scheduler instance inside a critical section (interrupts disabled).

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance. |
| *lock_context* | The lock context to use for _Scheduler_Release_critical(). |

Definition at line 119 of file schedulerimpl.h.

### 8.136.5.2 _Scheduler_Ask_for_help()

```
static __inline__ void _Scheduler_Ask_for_help (
             Thread_Control * the_thread )  [static]
```

Registers an ask for help request if necessary.

The actual ask for help operation is carried out during _Thread_Do_dispatch() on a processor related to the thread. This yields a better separation of scheduler instances. A thread of one scheduler instance should not be forced to carry out too much work for threads on other scheduler instances.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread in need for help. |

Definition at line 191 of file schedulerimpl.h.

### 8.136.5.3 _Scheduler_Block()

```
static __inline__ void _Scheduler_Block (
             Thread_Control * the_thread )  [static]
```

Blocks a thread with respect to the scheduler.

This routine removes *the_thread* from the scheduling decision for the scheduler. The primary task is to remove the thread from the ready queue. It performs any necessary scheduling operations including the selection of a new heir thread.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread. |

Definition at line 270 of file schedulerimpl.h.

#### 8.136.5.4 _Scheduler_Block_node()

```
static __inline__ Per_CPU_Control* _Scheduler_Block_node (
             Scheduler_Context * context,
             Thread_Control * thread,
             Scheduler_Node * node,
             bool is_scheduled,
             Scheduler_Get_idle_thread get_idle_thread )  [static]
```

Blocks this scheduler node.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler instance context. |
| in,out | *thread* | The thread which wants to get blocked referencing this node. This is not necessarily the user of this node in case the node participates in the scheduler helping protocol. |
| in,out | *node* | The node which wants to get blocked. |
| | *is_scheduled* | This node is scheduled. |
| | *get_idle_thread* | Function to get an idle thread. |

**Return values**

| | |
|---|---|
| *thread_cpu* | The processor of the thread. Indicates to continue with the blocking operation. |
| *NULL* | Otherwise. |

Definition at line 1110 of file schedulerimpl.h.

#### 8.136.5.5 _Scheduler_Build_id()

```
static __inline__ Objects_Id _Scheduler_Build_id (
             uint32_t scheduler_index )  [static]
```

Builds an object build id.

**Parameters**

| | |
|---|---|
| *scheduler_index* | The index to build the build id out of. |

**Returns**

The build id.

Definition at line 808 of file schedulerimpl.h.

### 8.136.5.6 _Scheduler_Cancel_job()

```
static __inline__ void _Scheduler_Cancel_job (
              Thread_Control * the_thread,
              Priority_Node * priority_node,
              Thread_queue_Context * queue_context )  [static]
```

Cancels a job of a thread with respect to the scheduler.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread. |
| *priority_node* | The priority node of the job. |
| *queue_context* | The thread queue context to provide the set of threads for _Thread_Priority_update(). |

Definition at line 585 of file schedulerimpl.h.

### 8.136.5.7 _Scheduler_default_Ask_for_help()

```
bool _Scheduler_default_Ask_for_help (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread,
              Scheduler_Node * node )
```

Does nothing.

**Parameters**

| | |
|---|---|
| *scheduler* | This parameter is unused. |
| *the_thread* | This parameter is unused. |
| *node* | This parameter is unused. |

**Returns**

Always returns false.

Definition at line 15 of file schedulerdefaultaskforhelp.c.

#### 8.136.5.8 _Scheduler_default_Cancel_job()

```
void _Scheduler_default_Cancel_job (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread,
              Priority_Node * priority_node,
              Thread_queue_Context * queue_context )
```

Does nothing.

**Parameters**

| | |
|---|---|
| *scheduler* | This parameter is unused. |
| *the_thread* | This parameter is unused. |
| *priority_node* | This parameter is unused. |
| *queue_context* | This parameter is unused. |

**Returns**

Always returns NULL.

Definition at line 39 of file schedulerdefaultreleasejob.c.

#### 8.136.5.9 _Scheduler_default_Map_priority()

```
Priority_Control _Scheduler_default_Map_priority (
              const Scheduler_Control * scheduler,
              Priority_Control priority )
```

Returns the scheduler internal thread priority mapped by SCHEDULER_PRIORITY_MAP().

**Parameters**

| | |
|---|---|
| *scheduler* | Unused. |
| *priority* | The user visible thread priority. |

**Returns**

The scheduler internal thread priority.

Definition at line 15 of file schedulerdefaultmappriority.c.

#### 8.136.5.10 _Scheduler_default_Node_destroy()

```
void _Scheduler_default_Node_destroy (
              const Scheduler_Control * scheduler,
              Scheduler_Node * node )
```

Does nothing.

**Parameters**

| | |
|---|---|
| *scheduler* | This parameter is unused. |
| *node* | This parameter is unused. |

Definition at line 24 of file schedulerdefaultnodedestroy.c.

#### 8.136.5.11 _Scheduler_default_Node_initialize()

```
void _Scheduler_default_Node_initialize (
              const Scheduler_Control * scheduler,
              Scheduler_Node * node,
              Thread_Control * the_thread,
              Priority_Control priority )
```

Performs the scheduler base node initialization.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | This parameter is unused. |
| out | *node* | The node to initialize. |
| | *the_thread* | This parameter is unused. |
| | *priority* | The thread priority. |

Definition at line 24 of file schedulerdefaultnodeinit.c.

#### 8.136.5.12 _Scheduler_default_Pin_or_unpin()

```
void _Scheduler_default_Pin_or_unpin (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread,
              Scheduler_Node * node,
              struct Per_CPU_Control * cpu )
```

Does nothing in a single processor system, otherwise a fatal error is issued.

**Parameters**

| | |
|---|---|
| *scheduler* | This parameter is unused. |
| *the_thread* | This parameter is unused. |
| *node* | This parameter is unused. |
| *cpu* | This parameter is unused. |

Definition at line 17 of file schedulerdefaultpinunpin.c.

#### 8.136.5.13 _Scheduler_default_Reconsider_help_request()

```
void _Scheduler_default_Reconsider_help_request (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread,
              Scheduler_Node * node )
```

Does nothing.

**Parameters**

| | |
|---|---|
| *scheduler* | This parameter is unused. |
| *the_thread* | This parameter is unused. |
| *node* | This parameter is unused. |

Definition at line 28 of file schedulerdefaultaskforhelp.c.

#### 8.136.5.14 _Scheduler_default_Release_job()

```
void _Scheduler_default_Release_job (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread,
              Priority_Node * priority_node,
              uint64_t deadline,
              Thread_queue_Context * queue_context )
```

Does nothing.

**Parameters**

| | |
|---|---|
| *scheduler* | This parameter is unused. |
| *the_thread* | This parameter is unused. |
| *priority_node* | This parameter is unused. |
| *deadline* | This parameter is unused. |
| *queue_context* | This parameter is unused. |

**Returns**

Always returns NULL.

Definition at line 24 of file schedulerdefaultreleasejob.c.

### 8.136.5.15 _Scheduler_default_Schedule()

```
void _Scheduler_default_Schedule (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread )
```

Does nothing.

**Parameters**

| | |
|---|---|
| *scheduler* | This parameter is unused. |
| *the_thread* | This parameter is unused. |

Definition at line 15 of file schedulerdefaultschedule.c.

### 8.136.5.16 _Scheduler_default_Set_affinity()

```
bool _Scheduler_default_Set_affinity (
              const Scheduler_Control * scheduler,
              Thread_Control * thread,
              Scheduler_Node * node,
              const Processor_mask * affinity )
```

Checks if the processor set of the scheduler is the subset of the affinity set.

Default implementation of the set affinity scheduler operation.

**Parameters**

| | |
|---|---|
| *scheduler* | This parameter is unused. |
| *thread* | This parameter is unused. |
| *node* | This parameter is unused. |
| *affinity* | The new processor affinity set for the thread. |

**Return values**

| | |
|---|---|
| *true* | The processor set of the scheduler is a subset of the affinity set. |
| *false* | The processor set of the scheduler is not a subset of the affinity set. |

Definition at line 24 of file schedulerdefaultsetaffinity.c.

#### 8.136.5.17 _Scheduler_default_Set_affinity_body()

```
static __inline__ bool _Scheduler_default_Set_affinity_body (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread,
              Scheduler_Node * node,
              const Processor_mask * affinity )  [static]
```

Checks if the affinity is a subset of the online processors.

**Parameters**

| | |
|---|---|
| *scheduler* | This parameter is unused. |
| *the_thread* | This parameter is unused. |
| *node* | This parameter is unused. |
| *affinity* | The processor mask to check. |

**Return values**

| | |
|---|---|
| *true* | *affinity* is a subset of the online processors. |
| *false* | *affinity* is not a subset of the online processors. |

Definition at line 716 of file schedulerimpl.h.

#### 8.136.5.18 _Scheduler_default_Start_idle()

```
void _Scheduler_default_Start_idle (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread,
              struct Per_CPU_Control * cpu )
```

Starts an idle thread.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | This parameter is unused. |
| in,out | *the_thread* | An idle thread. |
| | *cpu* | This parameter is unused. |

Definition at line 15 of file schedulerdefaultstartidle.c.

#### 8.136.5.19 _Scheduler_default_Tick()

```
void _Scheduler_default_Tick (
```

```
            const Scheduler_Control * scheduler,
            Thread_Control * executing )
```

Performs tick operations depending on the CPU budget algorithm for each executing thread.

This routine is invoked as part of processing each clock tick.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler. |
| in,out | *executing* | An executing thread. |

Definition at line 27 of file schedulerdefaulttick.c.

### 8.136.5.20 _Scheduler_default_Unmap_priority()

```
Priority_Control _Scheduler_default_Unmap_priority (
            const Scheduler_Control * scheduler,
            Priority_Control priority )
```

Returns the user visible thread priority unmapped by SCHEDULER_PRIORITY_UNMAP().

**Parameters**

| | |
|---|---|
| *scheduler* | Unused. |
| *priority* | The scheduler internal thread priority. |

**Returns**

priority The user visible thread priority.

Definition at line 23 of file schedulerdefaultmappriority.c.

### 8.136.5.21 _Scheduler_default_Withdraw_node()

```
void _Scheduler_default_Withdraw_node (
            const Scheduler_Control * scheduler,
            Thread_Control * the_thread,
            Scheduler_Node * node,
            Thread_Scheduler_state next_state )
```

Does nothing.

**Parameters**

| | |
|---|---|
| *scheduler* | This parameter is unused. |
| *the_thread* | This parameter is unused.. |
| *node* | This parameter is unused. |
| *next_state* | This parameter is unused. |

Definition at line 39 of file schedulerdefaultaskforhelp.c.

#### 8.136.5.22 _Scheduler_Discard_idle_thread()

```
static __inline__ void _Scheduler_Discard_idle_thread (
            Scheduler_Context * context,
            Thread_Control * the_thread,
            Scheduler_Node * node,
            Scheduler_Release_idle_thread release_idle_thread )  [static]
```

Discard the idle thread from the scheduler node.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler context. |
| in,out | *the_thread* | The thread for the operation. |
| in,out | *node* | The scheduler node to discard the idle thread from. |
| | *release_idle_thread* | Method to release the idle thread from the context. |

Definition at line 1161 of file schedulerimpl.h.

#### 8.136.5.23 _Scheduler_Exchange_idle_thread()

```
static __inline__ void _Scheduler_Exchange_idle_thread (
            Scheduler_Node * needs_idle,
            Scheduler_Node * uses_idle,
            Thread_Control * idle )  [static]
```

Exchanges an idle thread from the scheduler node that uses it right now to another scheduler node.

**Parameters**

| | |
|---|---|
| *needs_idle* | The scheduler node that needs an idle thread. |
| *uses_idle* | The scheduler node that used the idle thread. |
| *idle* | The idle thread that is exchanged. |

Definition at line 1081 of file schedulerimpl.h.

#### 8.136.5.24 _Scheduler_Generic_block()

```
static __inline__ void _Scheduler_Generic_block (
            const Scheduler_Control * scheduler,
            Thread_Control * the_thread,
```

```
            Scheduler_Node * node,
            void(*)(const Scheduler_Control *, Thread_Control *, Scheduler_Node *) extract,
            void(*)(const Scheduler_Control *, Thread_Control *, bool) schedule )  [static]
```

Blocks the thread.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance. |
| *the_thread* | The thread to block. |
| *node* | The corresponding scheduler node. |
| *extract* | Method to extract the thread. |
| *schedule* | Method for scheduling threads. |

Definition at line 754 of file schedulerimpl.h.

### 8.136.5.25 _Scheduler_Get_affinity()

```
bool _Scheduler_Get_affinity (
            Thread_Control * the_thread,
            size_t cpusetsize,
            cpu_set_t * cpuset )
```

Copies the thread's scheduler's affinity to the given cpuset.

**Parameters**

| | | |
|---|---|---|
| | *the_thread* | The thread to get the affinity of its scheduler. |
| | *cpusetsize* | The size of *cpuset*. |
| out | *cpuset* | The cpuset that serves as destination for the copy operation |

**Return values**

| | |
|---|---|
| *true* | The copy operation was lossless. |
| *false* | The copy operation was not lossless |

### 8.136.5.26 _Scheduler_Get_by_CPU()

```
static __inline__ const Scheduler_Control* _Scheduler_Get_by_CPU (
            const Per_CPU_Control * cpu )  [static]
```

Gets the scheduler for the cpu.

**Parameters**

| | |
|---|---|
| *cpu* | The cpu control to get the scheduler of. |

**Returns**

The scheduler for the cpu.

Definition at line 99 of file schedulerimpl.h.

### 8.136.5.27 _Scheduler_Get_by_id()

```
static __inline__ const Scheduler_Control* _Scheduler_Get_by_id (
            Objects_Id id )  [static]
```

Gets the scheduler from the given object build id.

**Parameters**

| | |
|---|---|
| *id* | The object build id. |

**Returns**

The scheduler to the object id.

Definition at line 839 of file schedulerimpl.h.

### 8.136.5.28 _Scheduler_Get_context()

```
static __inline__ Scheduler_Context* _Scheduler_Get_context (
            const Scheduler_Control * scheduler )  [static]
```

Gets the context of the scheduler.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler to get the context of. |

**Returns**

The context of *scheduler*.

Definition at line 85 of file schedulerimpl.h.

### 8.136.5.29 _Scheduler_Get_index()

```
static __inline__ uint32_t _Scheduler_Get_index (
            const Scheduler_Control * scheduler )  [static]
```

Gets the index of the scheduler.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler to get the index of. |

**Returns**

The index of the given scheduler.

Definition at line 861 of file schedulerimpl.h.

### 8.136.5.30 _Scheduler_Get_index_by_id()

```
static __inline__ uint32_t _Scheduler_Get_index_by_id (
            Objects_Id id )  [static]
```

Gets the scheduler index from the given object build id.

**Parameters**

| | |
|---|---|
| *id* | The object build id. |

**Returns**

The scheduler index.

Definition at line 825 of file schedulerimpl.h.

### 8.136.5.31 _Scheduler_Get_processor_count()

```
static __inline__ uint32_t _Scheduler_Get_processor_count (
            const Scheduler_Control * scheduler )  [static]
```

Gets the number of processors of the scheduler.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance to get the number of processors of. |

**Returns**

The number of processors.

Definition at line 786 of file schedulerimpl.h.

#### 8.136.5.32 _Scheduler_Get_processors()

```
static __inline__ const Processor_mask* _Scheduler_Get_processors (
            const Scheduler_Control * scheduler )  [static]
```

Gets the processors of the scheduler.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler to get the processors of. |

**Returns**

The processors of the context of the given scheduler.

Definition at line 678 of file schedulerimpl.h.

#### 8.136.5.33 _Scheduler_Handler_initialization()

```
void _Scheduler_Handler_initialization (
            void  )
```

Initializes the scheduler to the policy chosen by the user.

This routine initializes the scheduler to the policy chosen by the user through confdefs, or to the priority scheduler with ready chains by default.

Definition at line 24 of file scheduler.c.

#### 8.136.5.34 _Scheduler_Has_processor_ownership()

```
static __inline__ bool _Scheduler_Has_processor_ownership (
            const Scheduler_Control * scheduler,
            uint32_t cpu_index )  [static]
```

Checks if the scheduler of the cpu with the given index is equal to the given scheduler.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler for the comparison. |
| *cpu_index* | The index of the cpu for the comparison. |

**Return values**

| | |
|---|---|
| *true* | The scheduler of the cpu is the given *scheduler*. |
| *false* | The scheduler of the cpu is not the given *scheduler*. |

Definition at line 650 of file schedulerimpl.h.

#### 8.136.5.35 _Scheduler_Is_non_preempt_mode_supported()

```
static __inline__ bool _Scheduler_Is_non_preempt_mode_supported (
            const Scheduler_Control * scheduler )  [static]
```

Indicate if the thread non-preempt mode is supported by the scheduler.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance. |

**Returns**

True if the non-preempt mode for threads is supported by the scheduler, otherwise false.

Definition at line 169 of file schedulerimpl.h.

#### 8.136.5.36 _Scheduler_Map_priority()

```
static __inline__ Priority_Control _Scheduler_Map_priority (
            const Scheduler_Control * scheduler,
            Priority_Control priority )  [static]
```

Maps a thread priority from the user domain to the scheduler domain.

Let M be the maximum scheduler priority. The mapping must be bijective in the closed interval [0, M], e.g. _↩ Scheduler_Unmap_priority( scheduler, _Scheduler_Map_priority( scheduler, p ) ) == p for all p in [0, M]. For other values the mapping is undefined.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance. |
| *priority* | The user domain thread priority. |

**Returns**

The corresponding thread priority of the scheduler domain is returned.

Definition at line 480 of file schedulerimpl.h.

#### 8.136.5.37 _Scheduler_Node_destroy()

```
static __inline__ void _Scheduler_Node_destroy (
            const Scheduler_Control * scheduler,
            Scheduler_Node * node )  [static]
```

Destroys a scheduler node.

The caller must ensure that _Scheduler_Node_destroy() will be called only after a corresponding _Scheduler_Node_initialize().

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| out | *node* | The scheduler node to destroy. |

Definition at line 541 of file schedulerimpl.h.

#### 8.136.5.38 _Scheduler_Node_do_initialize()

```
static __inline__ void _Scheduler_Node_do_initialize (
            const struct _Scheduler_Control * scheduler,
            Scheduler_Node * node,
            Thread_Control * the_thread,
            Priority_Control priority )  [static]
```

Initializes a node.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler for the initialization of *node*. |
| out | *node* | The node to initialize. |
| | *the_thread* | The thread for the initialization of *node*. |
| | *priority* | The priority value for *node*. |

Definition at line 91 of file schedulernodeimpl.h.

#### 8.136.5.39 _Scheduler_Node_get_idle()

```
static __inline__ Thread_Control* _Scheduler_Node_get_idle (
            const Scheduler_Node * node )  [static]
```

Gets the idle thread of the node.

**Parameters**

| | |
|---|---|
| *node* | The node to get the idle thread of. |

**Returns**

The idle thread of *node*.

Definition at line 234 of file schedulernodeimpl.h.

### 8.136.5.40 _Scheduler_Node_get_owner()

```
static __inline__ Thread_Control* _Scheduler_Node_get_owner (
            const Scheduler_Node * node )  [static]
```

Gets the owner of the node.

**Parameters**

| | |
|---|---|
| *node* | The node to get the owner of. |

**Returns**

The owner of the node.

Definition at line 135 of file schedulernodeimpl.h.

### 8.136.5.41 _Scheduler_Node_get_priority()

```
static __inline__ Priority_Control _Scheduler_Node_get_priority (
            Scheduler_Node * node )  [static]
```

Gets the priority of the node.

**Parameters**

| | |
|---|---|
| *node* | The node to get the priority of. |

**Returns**

The priority of the node.

Definition at line 149 of file schedulernodeimpl.h.

### 8.136.5.42 _Scheduler_Node_get_scheduler()

```
static __inline__ const Scheduler_Control* _Scheduler_Node_get_scheduler (
            const Scheduler_Node * node )  [static]
```

Gets the scheduler of the node.

**Parameters**

| | |
|---|---|
| *node* | The node to get the scheduler of. |

**Returns**

The scheduler of the node.

Definition at line 121 of file schedulernodeimpl.h.

### 8.136.5.43 _Scheduler_Node_get_user()

```
static __inline__ Thread_Control* _Scheduler_Node_get_user (
            const Scheduler_Node * node )  [static]
```

Gets the user of the node.

**Parameters**

| | |
|---|---|
| *node* | The node to get the user of. |

**Returns**

The user of the node.

Definition at line 206 of file schedulernodeimpl.h.

### 8.136.5.44 _Scheduler_Node_initialize()

```
static __inline__ void _Scheduler_Node_initialize (
            const Scheduler_Control * scheduler,
            Scheduler_Node * node,
            Thread_Control * the_thread,
            Priority_Control priority )  [static]
```

Initializes a scheduler node.

The scheduler node contains arbitrary data on function entry. The caller must ensure that _Scheduler_Node_destroy() will be called after a _Scheduler_Node_initialize() before the memory of the scheduler node is destroyed.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| out | *node* | The scheduler node to initialize. |
| | *the_thread* | The thread of the scheduler node to initialize. |
| | *priority* | The thread priority. |

Definition at line 517 of file schedulerimpl.h.

#### 8.136.5.45 _Scheduler_Node_set_priority()

```
static __inline__ void _Scheduler_Node_set_priority (
            Scheduler_Node * node,
            Priority_Control new_priority,
            bool prepend_it )  [static]
```

Sets the priority of the node.

**Parameters**

| | | |
|---|---|---|
| in,out | *node* | The node to set the priority of. |
| | *new_priority* | The new priority for *node*. |
| | *prepend_it* | Indicates whether the new priority should be prepended. |

Definition at line 178 of file schedulernodeimpl.h.

#### 8.136.5.46 _Scheduler_Node_set_user()

```
static __inline__ void _Scheduler_Node_set_user (
            Scheduler_Node * node,
            Thread_Control * user )  [static]
```

Sets the user of the node.

**Parameters**

| | | |
|---|---|---|
| out | *node* | The node to set the user of. |
| | *user* | The new user for *node*. |

Definition at line 219 of file schedulernodeimpl.h.

#### 8.136.5.47 _Scheduler_Priority_and_sticky_update()

```
static __inline__ void _Scheduler_Priority_and_sticky_update (
            Thread_Control * the_thread,
            int sticky_level_change )  [static]
```

Changes the sticky level of the home scheduler node and propagates a priority change of a thread to the scheduler.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread changing its priority or sticky level. |

**See also**

_Scheduler_Update_priority().

Definition at line 417 of file schedulerimpl.h.

### 8.136.5.48 _Scheduler_Release_critical()

```
static __inline__ void _Scheduler_Release_critical (
            const Scheduler_Control * scheduler,
            ISR_lock_Context * lock_context )  [static]
```

Releases the scheduler instance inside a critical section (interrupts disabled).

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance. |
| *lock_context* | The lock context used for _Scheduler_Acquire_critical(). |

Definition at line 143 of file schedulerimpl.h.

### 8.136.5.49 _Scheduler_Release_idle_thread()

```
static __inline__ Thread_Control* _Scheduler_Release_idle_thread (
            Scheduler_Context * context,
            Scheduler_Node * node,
            Scheduler_Release_idle_thread release_idle_thread )  [static]
```

Releases an idle thread using this scheduler node.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler instance context. |
| in,out | *node* | The node which may have an idle thread as user. |
| | *release_idle_thread* | Function to release an idle thread. |

**Return values**

| | |
|---|---|
| *idle* | The idle thread which used this node. |
| *NULL* | This node had no idle thread as an user. |

Definition at line 1054 of file schedulerimpl.h.

#### 8.136.5.50 _Scheduler_Release_job()

```
static __inline__ void _Scheduler_Release_job (
            Thread_Control * the_thread,
            Priority_Node * priority_node,
            uint64_t deadline,
            Thread_queue_Context * queue_context )  [static]
```

Releases a job of a thread with respect to the scheduler.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread. |
| *priority_node* | The priority node of the job. |
| *deadline* | The deadline in watchdog ticks since boot. |
| *queue_context* | The thread queue context to provide the set of threads for _Thread_Priority_update(). |

Definition at line 558 of file schedulerimpl.h.

#### 8.136.5.51 _Scheduler_Schedule()

```
static __inline__ void _Scheduler_Schedule (
            Thread_Control * the_thread )  [static]
```

General scheduling decision.

The preferred method to add a new scheduler is to define the jump table entries and add a case to the _Scheduler↩ _Initialize routine.

Generic scheduling implementations that rely on the ready queue only can be found in the _Scheduler_queue_XXX functions.

This kernel routine implements the scheduling decision logic for the scheduler. It does NOT dispatch.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread which state changed previously. |

Definition at line 224 of file schedulerimpl.h.

#### 8.136.5.52 _Scheduler_Set()

```
static __inline__ Status_Control _Scheduler_Set (
```

```
            const Scheduler_Control * new_scheduler,
            Thread_Control * the_thread,
            Priority_Control priority )  [static]
```

Sets a new scheduler.

**Parameters**

| | | |
|---|---|---|
| | *new_scheduler* | The new scheduler to set. |
| in,out | *the_thread* | The thread for the operations. |
| | *priority* | The initial priority for the thread with the new scheduler. |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The operation succeeded. |
| *STATUS_RESOURCE_IN_USE* | The thread's wait queue is not empty. |
| *STATUS_UNSATISFIED* | The new scheduler has no processors. |

Definition at line 1270 of file schedulerimpl.h.

### 8.136.5.53 _Scheduler_Set_affinity()

```
bool _Scheduler_Set_affinity (
            Thread_Control * the_thread,
            size_t cpusetsize,
            const cpu_set_t * cpuset )
```

Sets the thread's scheduler's affinity.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread to set the affinity of. |
| | *cpusetsize* | The size of *cpuset*. |
| | *cpuset* | The cpuset to set the affinity. |

**Return values**

| | |
|---|---|
| *true* | The operation succeeded. |
| *false* | The operation did not succeed. |

### 8.136.5.54 _Scheduler_Set_idle_thread()

```
static __inline__ void _Scheduler_Set_idle_thread (
            Scheduler_Node * node,
            Thread_Control * idle )  [static]
```

Sets the scheduler node's idle thread.

**Parameters**

| | | |
|---|---|---|
| in,out | *node* | The node to receive an idle thread. |
| | *idle* | The idle thread control for the operation. |

Definition at line 918 of file schedulerimpl.h.

### 8.136.5.55 _Scheduler_Start_idle()

```
static __inline__ void _Scheduler_Start_idle (
            const Scheduler_Control * scheduler,
            Thread_Control * the_thread,
            Per_CPU_Control * cpu )  [static]
```

Starts the idle thread for a particular processor.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| in,out | *the_thread* | The idle thread for the processor. |
| in,out | *cpu* | The processor for the idle thread. |

**See also**

_Thread_Create_idle().

Definition at line 631 of file schedulerimpl.h.

### 8.136.5.56 _Scheduler_Thread_change_state()

```
static __inline__ void _Scheduler_Thread_change_state (
            Thread_Control * the_thread,
            Thread_Scheduler_state new_state )  [static]
```

Changes the threads state to the given new state.

**Parameters**

| | | |
|---|---|---|
| out | *the_thread* | The thread to change the state of. |
| | *new_state* | The new state for *the_thread*. |

Definition at line 898 of file schedulerimpl.h.

### 8.136.5.57 _Scheduler_Tick()

```
static __inline__ void _Scheduler_Tick (
            const Per_CPU_Control * cpu )  [static]
```

Scheduler method invoked at each clock tick.

This method is invoked at each clock tick to allow the scheduler implementation to perform any activities required. For the scheduler which support standard RTEMS features, this includes time-slicing management.

**Parameters**

| | |
|---|---|
| *cpu* | The cpu control for the operation. |

Definition at line 612 of file schedulerimpl.h.

### 8.136.5.58 _Scheduler_Try_to_schedule_node()

```
static __inline__ Scheduler_Try_to_schedule_action _Scheduler_Try_to_schedule_node (
            Scheduler_Context * context,
            Scheduler_Node * node,
            const Thread_Control * idle,
            Scheduler_Get_idle_thread get_idle_thread )  [static]
```

Tries to schedule the scheduler node.

When a scheduler needs to schedule a node, it shall use this function to determine what it shall do with the node. The node replaces a victim node if it can be scheduled.

This function uses the state of the node and the scheduler state of the owner thread to determine what shall be done. Each scheduler maintains its nodes independent of other schedulers. This function ensures that a thread is scheduled by at most one scheduler. If a node requires an executing thread due to some locking protocol and the owner thread is already scheduled by another scheduler, then an idle thread shall be attached to the node.

**Parameters**

| | | |
|---|---|---|
| in,out | *context* | is the scheduler context. |
| in,out | *node* | is the node which could be scheduled. |
| | *idle* | is an idle thread used by the victim node or NULL. |
| | *get_idle_thread* | points to a function to get an idle thread. |

**Return values**

| | |
|---|---|
| *SCHEDULER_TRY_TO_SCHEDULE_DO_SCHED↩ ULE* | The node shall be scheduled. |
| *SCHEDULER_TRY_TO_SCHEDULE_DO_IDLE_E↩ XCHANGE* | The node shall be scheduled and the provided idle thread shall be attached to the node. This action is returned, if the node cannot use the owner thread and shall use an idle thread instead. In this case, the idle thread is provided by the victim node. |

**Return values**

| | |
|---|---|
| *SCHEDULER_TRY_TO_SCHEDULE_DO_BLOCK* | The node shall be blocked. This action is returned, if the owner thread is already scheduled by another scheduler. |

Definition at line 1001 of file schedulerimpl.h.

### 8.136.5.59 _Scheduler_Unblock()

```
static __inline__ void _Scheduler_Unblock (
            Thread_Control * the_thread )  [static]
```

Unblocks a thread with respect to the scheduler.

This operation must fetch the latest thread priority value for this scheduler instance and update its internal state if necessary.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread. |

**See also**

_Scheduler_Node_get_priority().

Definition at line 332 of file schedulerimpl.h.

### 8.136.5.60 _Scheduler_Unblock_node()

```
static __inline__ bool _Scheduler_Unblock_node (
            Scheduler_Context * context,
            Thread_Control * the_thread,
            Scheduler_Node * node,
            bool is_scheduled,
            Scheduler_Release_idle_thread release_idle_thread )  [static]
```

Unblocks this scheduler node.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler instance context. |
| in,out | *the_thread* | The thread which wants to get unblocked. |
| in,out | *node* | The node which wants to get unblocked. |
| | *is_scheduled* | This node is scheduled. |
| | *release_idle_thread* | Function to release an idle thread. |

**Return values**

| | |
|---|---|
| *true* | Continue with the unblocking operation. |
| *false* | Do not continue with the unblocking operation. |

Definition at line 1197 of file schedulerimpl.h.

### 8.136.5.61 _Scheduler_Unmap_priority()

```
static __inline__ Priority_Control _Scheduler_Unmap_priority (
            const Scheduler_Control * scheduler,
            Priority_Control priority )  [static]
```

Unmaps a thread priority from the scheduler domain to the user domain.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance. |
| *priority* | The scheduler domain thread priority. |

**Returns**

The corresponding thread priority of the user domain is returned.

Definition at line 496 of file schedulerimpl.h.

### 8.136.5.62 _Scheduler_Update_heir()

```
static __inline__ void _Scheduler_Update_heir (
            Thread_Control * new_heir,
            bool force_dispatch )  [static]
```

Updates the heir.

**Parameters**

| | | |
|---|---|---|
| in,out | *new_heir* | The new heir. |
| | *force_dispatch* | Indicates whether the dispatch happens also if the currently running thread is set as not preemptible. |

Definition at line 1235 of file schedulerimpl.h.

#### 8.136.5.63 _Scheduler_Update_priority()

```
static __inline__ void _Scheduler_Update_priority (
            Thread_Control * the_thread )  [static]
```

Propagates a priority change of a thread to the scheduler.

On uni-processor configurations, this operation must evaluate the thread state. In case the thread is not ready, then the priority update should be deferred to the next scheduler unblock operation.

The operation must update the heir and thread dispatch necessary variables in case the set of scheduled threads changes.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread changing its priority. |

**See also**

_Scheduler_Node_get_priority().

Definition at line 367 of file schedulerimpl.h.

#### 8.136.5.64 _Scheduler_Use_idle_thread()

```
static __inline__ Thread_Control* _Scheduler_Use_idle_thread (
            Scheduler_Context * context,
            Scheduler_Node * node,
            Per_CPU_Control * cpu,
            Scheduler_Get_idle_thread get_idle_thread )  [static]
```

Uses an idle thread for this scheduler node.

A thread whose home scheduler node has a sticky level greater than zero may use an idle thread in the home scheduler instance in the case it executes currently in another scheduler instance or in the case it is in a blocking state.

**Parameters**

| | | |
|---|---|---|
| | *context* | The scheduler instance context. |
| in,out | *node* | The node which wants to use the idle thread. |
| | *cpu* | The processor for the idle thread. |
| | *get_idle_thread* | Function to get an idle thread. |

Definition at line 945 of file schedulerimpl.h.

#### 8.136.5.65 _Scheduler_Yield()

```
static __inline__ void _Scheduler_Yield (
            Thread_Control * the_thread )  [static]
```

Scheduler yield with a particular thread.

This routine is invoked when a thread wishes to voluntarily transfer control of the processor to another thread.

**Parameters**

| | |
|---|---|
| *the_thread* | The yielding thread. |

Definition at line 245 of file schedulerimpl.h.

### 8.136.6 Variable Documentation

#### 8.136.6.1 _Scheduler_Count

```
const size_t _Scheduler_Count  [extern]
```

This constant contains the count of configured schedulers.

In SMP configurations, the constant is defined by <rtems/confdefs.h> through the count of entries of the #CON↩ FIGURE_SCHEDULER_TABLE_ENTRIES application configuration option.

In uniprocessor configurations, this is a compile time constant set to one. This is important so that the compiler can optimize some loops away.

**See also**

_Scheduler_Table.

#### 8.136.6.2 _Scheduler_Initial_assignments

```
const Scheduler_Assignment _Scheduler_Initial_assignments[]  [extern]
```

The scheduler assignments.

The length of this array must be equal to the maximum processors.

Application provided via <rtems/confdefs.h>.

**See also**

_Scheduler_Table and rtems_configuration_get_maximum_processors().

#### 8.136.6.3 _Scheduler_Node_size

```
const size_t _Scheduler_Node_size  [extern]
```

The size of a scheduler node.

This value is provided via <rtems/confdefs.h>.

#### 8.136.6.4 _Scheduler_Table

```
const Scheduler_Control _Scheduler_Table[]  [extern]
```

This table contains the configured schedulers.

The table is defined by <rtems/confdefs.h> through the #CONFIGURE_SCHEDULER_TABLE_ENTRIES application configuration option.

**See also**

_Scheduler_Count.

# 8.137 Score Timestamp

Score Timestamp.

## Files

- file timestamp.h

  *Helpers for Manipulating Timestamps.*
- file timestampimpl.h

  *Helpers for Manipulating Timestamps.*

## Typedefs

- typedef int64_t Timestamp_Control

## Functions

- static __inline__ void _Timestamp_Set (Timestamp_Control *_time, time_t _seconds, long _nanoseconds)

  *Sets timestamp to specified seconds and nanoseconds.*
- static __inline__ void _Timestamp_Set_to_zero (Timestamp_Control *_time)

  *Sets the timestamp to zero.*
- static __inline__ bool _Timestamp_Less_than (const Timestamp_Control *_lhs, const Timestamp_Control *_rhs)

  *Checks if the left hand side timestamp is less than the right one.*
- static __inline__ bool _Timestamp_Greater_than (const Timestamp_Control *_lhs, const Timestamp_Control *_rhs)

  *Checks if the left hand side timestamp is greater than the right one.*
- static __inline__ bool _Timestamp_Equal_to (const Timestamp_Control *_lhs, const Timestamp_Control *↩ _rhs)

  *Checks if the timestamps are equal.*
- static __inline__ void _Timestamp_Add_to (Timestamp_Control *_time, const Timestamp_Control *_add)

  *Adds two timestamps.*
- static __inline__ void _Timestamp_Subtract (const Timestamp_Control *_start, const Timestamp_Control *_end, Timestamp_Control *_result)

  *Subtracts two timestamps.*
- static __inline__ void _Timestamp_Divide (const Timestamp_Control *_lhs, const Timestamp_Control *_rhs, uint32_t *_ival_percentage, uint32_t *_fval_percentage)

  *Divides a timestamp by another timestamp.*
- static __inline__ time_t _Timestamp_Get_seconds (const Timestamp_Control *_time)

  *Gets seconds portion of timestamp.*
- static __inline__ uint32_t _Timestamp_Get_nanoseconds (const Timestamp_Control *_time)

  *Gets nanoseconds portion of timestamp.*
- static __inline__ uint64_t _Timestamp_Get_as_nanoseconds (const Timestamp_Control *_time)

  *Gets the timestamp as nanoseconds.*
- static __inline__ void _Timestamp_To_timespec (const Timestamp_Control *_timestamp, struct timespec *_timespec)

  *Converts timestamp to struct timespec.*
- static __inline__ void _Timestamp_To_timeval (const Timestamp_Control *_timestamp, struct timeval *_↩ timeval)

  *Converts timestamp to struct timeval.*

### 8.137.1 Detailed Description

Score Timestamp.

This handler encapsulates functionality related to manipulating SuperCore Timestamps. SuperCore Timestamps may be used to represent time of day, uptime, or intervals.

The key attribute of the SuperCore Timestamp handler is that it is a completely opaque handler. There can be multiple implementations of the required functionality and with a recompile, RTEMS can use any implementation. It is intended to be a simple wrapper.

This handler can be implemented as either struct timespec or unsigned64 bit numbers. The use of a wrapper class allows the the implementation of timestamps to change on a per architecture basis. This is an important option as the performance of this handler is critical.

### 8.137.2 Typedef Documentation

#### 8.137.2.1 Timestamp_Control

```
typedef int64_t Timestamp_Control
```

Define the Timestamp control type.

Definition at line 57 of file timestamp.h.

### 8.137.3 Function Documentation

#### 8.137.3.1 _Timestamp_Add_to()

```
static __inline__ void _Timestamp_Add_to (
            Timestamp_Control * _time,
            const Timestamp_Control * _add )  [static]
```

Adds two timestamps.

This routine adds two timestamps. The second argument is added to the first.

**Parameters**

| | | |
|---|---|---|
| in,out | *_time* | The base time to be added to. |
| | *_add* | points The timestamp to add to the first argument. |

Definition at line 147 of file timestampimpl.h.

### 8.137.3.2 _Timestamp_Divide()

```
static __inline__ void _Timestamp_Divide (
             const Timestamp_Control * _lhs,
             const Timestamp_Control * _rhs,
             uint32_t * _ival_percentage,
             uint32_t * _fval_percentage )  [static]
```

Divides a timestamp by another timestamp.

This routine divides a timestamp by another timestamp. The intended use is for calculating percentages to three decimal points.

**Parameters**

| | | |
|---|---|---|
| | *_lhs* | The left hand number. |
| | *_rhs* | The right hand number. |
| out | *_ival_percentage* | The integer portion of the average. |
| out | *_fval_percentage* | The thousandths of percentage. |

Definition at line 186 of file timestampimpl.h.

### 8.137.3.3 _Timestamp_Equal_to()

```
static __inline__ bool _Timestamp_Equal_to (
             const Timestamp_Control * _lhs,
             const Timestamp_Control * _rhs )  [static]
```

Checks if the timestamps are equal.

This method is the is equal to than operator for timestamps.

**Parameters**

| | |
|---|---|
| *_lhs* | The left hand side timestamp. |
| *_rhs* | The right hand side timestamp. |

**Return values**

| | |
|---|---|
| *true* | *_lhs* is equal to *_rhs* |
| *false* | *_lhs* is not equal to *_rhs*. |

Definition at line 130 of file timestampimpl.h.

### 8.137.3.4 _Timestamp_Get_as_nanoseconds()

```
static __inline__ uint64_t _Timestamp_Get_as_nanoseconds (
```

```
            const Timestamp_Control * _time )  [static]
```

Gets the timestamp as nanoseconds.

This method returns the timestamp as nanoseconds.

**Parameters**

| | |
|---|---|
| *_time* | The timestamp. |

**Returns**

The time in nanoseconds.

Definition at line 252 of file timestampimpl.h.

### 8.137.3.5 _Timestamp_Get_nanoseconds()

```
static __inline__ uint32_t _Timestamp_Get_nanoseconds (
            const Timestamp_Control * _time )  [static]
```

Gets nanoseconds portion of timestamp.

This method returns the nanoseconds portion of the specified timestamp.

**Parameters**

| | |
|---|---|
| *_time* | The timestamp. |

**Returns**

The nanoseconds portion of *_time*.

Definition at line 232 of file timestampimpl.h.

### 8.137.3.6 _Timestamp_Get_seconds()

```
static __inline__ time_t _Timestamp_Get_seconds (
            const Timestamp_Control * _time )  [static]
```

Gets seconds portion of timestamp.

This method returns the seconds portion of the specified timestamp.

**Parameters**

| | |
|---|---|
| *_time* | The timestamp. |

**Returns**

The seconds portion of *_time*.

Definition at line 216 of file timestampimpl.h.

### 8.137.3.7 _Timestamp_Greater_than()

```
static __inline__ bool _Timestamp_Greater_than (
            const Timestamp_Control * _lhs,
            const Timestamp_Control * _rhs )  [static]
```

Checks if the left hand side timestamp is greater than the right one.

This method is the greater than operator for timestamps.

**Parameters**

| | |
|---|---|
| *_lhs* | The left hand side timestamp. |
| *_rhs* | The right hand side timestamp. |

**Return values**

| | |
|---|---|
| *true* | *_lhs* is greater than the *_rhs*. |
| *false* | *_lhs* is less or equal than *_rhs*. |

Definition at line 110 of file timestampimpl.h.

### 8.137.3.8 _Timestamp_Less_than()

```
static __inline__ bool _Timestamp_Less_than (
            const Timestamp_Control * _lhs,
            const Timestamp_Control * _rhs )  [static]
```

Checks if the left hand side timestamp is less than the right one.

This method is the less than operator for timestamps.

**Parameters**

| | |
|---|---|
| *_lhs* | The left hand side timestamp. |
| *_rhs* | The right hand side timestamp. |

**Return values**

| | |
|---|---|
| *true* | *_lhs* is less than the *_rhs*. |
| *false* | *_lhs* is greater or equal than *rhs*. |

Definition at line 90 of file timestampimpl.h.

### 8.137.3.9 _Timestamp_Set()

```
static __inline__ void _Timestamp_Set (
            Timestamp_Control * _time,
            time_t _seconds,
            long _nanoseconds )  [static]
```

Sets timestamp to specified seconds and nanoseconds.

This method sets the timestamp to the specified *_seconds* and *_nanoseconds* value.

**Parameters**

| | | |
|---|---|---|
| out | *_time* | The timestamp instance to set. |
| | *_seconds* | The seconds portion of the timestamp. |
| | *_nanoseconds* | The nanoseconds portion of the timestamp. |

Definition at line 48 of file timestampimpl.h.

### 8.137.3.10 _Timestamp_Set_to_zero()

```
static __inline__ void _Timestamp_Set_to_zero (
            Timestamp_Control * _time )  [static]
```

Sets the timestamp to zero.

This method sets the timestamp to zero. value.

**Parameters**

| | | |
|---|---|---|
| out | *_time* | The timestamp instance to zero. |

Definition at line 71 of file timestampimpl.h.

### 8.137.3.11 _Timestamp_Subtract()

```
static __inline__ void _Timestamp_Subtract (
```

```
            const Timestamp_Control * _start,
            const Timestamp_Control * _end,
            Timestamp_Control * _result )  [static]
```

Subtracts two timestamps.

This routine subtracts two timestamps. *result* is set to *end* - *start*.

**Parameters**

| | | |
|---|---|---|
| | *_start* | The starting time. |
| | *_end* | The ending time. |
| out | *_result* | Contains the difference between starting and ending time after the method call. |

Definition at line 166 of file timestampimpl.h.

#### 8.137.3.12 _Timestamp_To_timespec()

```
static __inline__ void _Timestamp_To_timespec (
            const Timestamp_Control * _timestamp,
            struct timespec * _timespec )  [static]
```

Converts timestamp to struct timespec.

**Parameters**

| | | |
|---|---|---|
| | *_timestamp* | The timestamp. |
| out | *_timespec* | The timespec to be filled in by the method. |

Definition at line 269 of file timestampimpl.h.

#### 8.137.3.13 _Timestamp_To_timeval()

```
static __inline__ void _Timestamp_To_timeval (
            const Timestamp_Control * _timestamp,
            struct timeval * _timeval )  [static]
```

Converts timestamp to struct timeval.

**Parameters**

| | | |
|---|---|---|
| | *_timestamp* | The timestamp. |
| out | *_timeval* | The timeval to be filled in by the method. |

Definition at line 283 of file timestampimpl.h.

## 8.138 Semaphore Handler

Semaphore Handler.

### Files

- file coresem.h

  *Data Associated with the Counting Semaphore Handler.*
- file coresemimpl.h

  *Inlined Routines Associated with the SuperCore Semaphore.*
- file semaphoreimpl.h

  *Semaphore Implementation.*
- file coresem.c

  *Core Semaphore Initialize.*

### Classes

- struct CORE_semaphore_Control
- struct Sem_Control

### Macros

- #define **SEMAPHORE_TQ_OPERATIONS** &_Thread_queue_Operations_priority

### Functions

- void _CORE_semaphore_Initialize (CORE_semaphore_Control *the_semaphore, uint32_t initial_value)

  *Initializes the semaphore based on the parameters passed.*
- static __inline__ void _CORE_semaphore_Acquire_critical (CORE_semaphore_Control *the_semaphore, Thread_queue_Context *queue_context)

  *Acquires the semaphore critical.*
- static __inline__ void _CORE_semaphore_Release (CORE_semaphore_Control *the_semaphore, Thread_queue_Context *queue_context)

  *Releases the semaphore.*
- static __inline__ void _CORE_semaphore_Destroy (CORE_semaphore_Control *the_semaphore, const Thread_queue_Operations *operations, Thread_queue_Context *queue_context)

  *Destroys the semaphore.*
- static __inline__ Status_Control _CORE_semaphore_Surrender (CORE_semaphore_Control *the_↩ semaphore, const Thread_queue_Operations *operations, uint32_t maximum_count, Thread_queue_Context *queue_context)

  *Surrenders a unit to the semaphore.*
- static __inline__ uint32_t _CORE_semaphore_Get_count (const CORE_semaphore_Control *the_↩ semaphore)

  *Returns the current count associated with the semaphore.*
- static __inline__ Status_Control _CORE_semaphore_Seize (CORE_semaphore_Control *the_semaphore, const Thread_queue_Operations *operations, Thread_Control *executing, bool wait, Thread_queue_Context *queue_context)

  *Seizes the semaphore.*
- static Sem_Control * _Sem_Get (struct _Semaphore_Control *_sem)

*Gets the Sem_Control * of the semaphore.*

- static Thread_Control * _Sem_Queue_acquire_critical (Sem_Control *sem, Thread_queue_Context *queue_context)

  *Acquires the semaphore queue critical.*

- static void _Sem_Queue_release (Sem_Control *sem, ISR_Level level, Thread_queue_Context *queue_context)

  *Releases the semaphore queue.*

## 8.138.1 Detailed Description

Semaphore Handler.

This handler encapsulates functionality which provides the foundation Semaphore services used in all of the APIs supported by RTEMS.

## 8.138.2 Function Documentation

### 8.138.2.1 _CORE_semaphore_Acquire_critical()

```
static __inline__ void _CORE_semaphore_Acquire_critical (
              CORE_semaphore_Control * the_semaphore,
              Thread_queue_Context * queue_context )  [static]
```

Acquires the semaphore critical.

This routine acquires the semaphore.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_semaphore* | The semaphore to acquire. |
| | *queue_context* | The thread queue context. |

Definition at line 67 of file coresemimpl.h.

### 8.138.2.2 _CORE_semaphore_Destroy()

```
static __inline__ void _CORE_semaphore_Destroy (
              CORE_semaphore_Control * the_semaphore,
              const Thread_queue_Operations * operations,
              Thread_queue_Context * queue_context )  [static]
```

Destroys the semaphore.

This routine destroys the semaphore.

**Parameters**

| | | |
|---|---|---|
| out | *the_semaphore* | The semaphore to destroy. |
| | *operations* | The thread queue operations. |
| | *queue_context* | The thread queue context. |

Definition at line 100 of file coresemimpl.h.

#### 8.138.2.3 _CORE_semaphore_Get_count()

```
static __inline__ uint32_t _CORE_semaphore_Get_count (
              const CORE_semaphore_Control * the_semaphore )  [static]
```

Returns the current count associated with the semaphore.

**Parameters**

| | |
|---|---|
| *the_semaphore* | The semaphore to obtain the count of. |

**Returns**

the current count of this semaphore.

Definition at line 175 of file coresemimpl.h.

#### 8.138.2.4 _CORE_semaphore_Initialize()

```
void _CORE_semaphore_Initialize (
              CORE_semaphore_Control * the_semaphore,
              uint32_t initial_value )
```

Initializes the semaphore based on the parameters passed.

This package is the implementation of the CORE Semaphore Handler. This core object utilizes standard Dijkstra counting semaphores to provide synchronization and mutual exclusion capabilities.

This routine initializes the semaphore based on the parameters passed.

**Parameters**

| | | |
|---|---|---|
| out | *the_semaphore* | The semaphore to initialize. |
| | *initial_value* | The initial count of the semaphore. |

Definition at line 23 of file coresem.c.

### 8.138.2.5 _CORE_semaphore_Release()

```
static __inline__ void _CORE_semaphore_Release (
             CORE_semaphore_Control * the_semaphore,
             Thread_queue_Context * queue_context )  [static]
```

Releases the semaphore.

This routine releases the semaphore.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_semaphore* | The semaphore to release. |
| | *queue_context* | The thread queue context. |

Definition at line 83 of file coresemimpl.h.

### 8.138.2.6 _CORE_semaphore_Seize()

```
static __inline__ Status_Control _CORE_semaphore_Seize (
             CORE_semaphore_Control * the_semaphore,
             const Thread_queue_Operations * operations,
             Thread_Control * executing,
             bool wait,
             Thread_queue_Context * queue_context )  [static]
```

Seizes the semaphore.

This routine attempts to receive a unit from the_semaphore. If a unit is available or if the wait flag is false, then the routine returns. Otherwise, the calling task is blocked until a unit becomes available.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_semaphore* | The semaphore to obtain. |
| | *operations* | The thread queue operations. |
| | *executing* | The currently executing thread. |
| | *wait* | Indicates whether the calling thread is willing to wait. |
| | *queue_context* | is a temporary variable used to contain the ISR disable level cookie. |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The semaphore was successfully seized. |
| *STATUS_UNSATISFIED* | The semaphore is currently not free and the calling thread not willing to wait. |
| *STATUS_TIMEOUT* | A timeout occured. |

Definition at line 202 of file coresemimpl.h.

#### 8.138.2.7 _CORE_semaphore_Surrender()

```
static __inline__ Status_Control _CORE_semaphore_Surrender (
             CORE_semaphore_Control * the_semaphore,
             const Thread_queue_Operations * operations,
             uint32_t maximum_count,
             Thread_queue_Context * queue_context )  [static]
```

Surrenders a unit to the semaphore.

This routine frees a unit to the semaphore. If a task was blocked waiting for a unit from this semaphore, then that task will be readied and the unit given to that task. Otherwise, the unit will be returned to the semaphore.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_semaphore* | is the semaphore to surrender |
| | *operations* | The thread queue operations. |
| | *maximum_count* | The maximum number of units in the semaphore. |
| | *queue_context* | is a temporary variable used to contain the ISR disable level cookie. |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The unit was successfully freed to the semaphore. |
| *STATUS_MAXIMUM_COUNT_EXCEEDED* | The maximum number of units was exceeded. |

Definition at line 131 of file coresemimpl.h.

#### 8.138.2.8 _Sem_Get()

```
static Sem_Control* _Sem_Get (
             struct _Semaphore_Control * _sem )  [inline], [static]
```

Gets the Sem_Control * of the semaphore.

**Parameters**

| | |
|---|---|
| *sem* | The Semaphore_Control * to cast to Sem_Control *. |

**Returns**

*sem* cast to Sem_Control *.

Definition at line 55 of file semaphoreimpl.h.

### 8.138.2.9 _Sem_Queue_acquire_critical()

```
static Thread_Control* _Sem_Queue_acquire_critical (
            Sem_Control * sem,
            Thread_queue_Context * queue_context )  [inline], [static]
```

Acquires the semaphore queue critical.

This routine acquires the semaphore.

**Parameters**

| | | |
|---|---|---|
| in,out | *sem* | The semaphore to acquire the queue of. |
| | *queue_context* | The thread queue context. |

**Returns**

The executing thread.

Definition at line 70 of file semaphoreimpl.h.

### 8.138.2.10 _Sem_Queue_release()

```
static void _Sem_Queue_release (
            Sem_Control * sem,
            ISR_Level level,
            Thread_queue_Context * queue_context )  [inline], [static]
```

Releases the semaphore queue.

**Parameters**

| | | |
|---|---|---|
| in,out | *sem* | The semaphore to release the queue of. |
| | *level* | The interrupt level value to restore the interrupt status on the processor. |
| | *queue_context* | The thread queue context. |

Definition at line 94 of file semaphoreimpl.h.

## 8.139 Semaphore Manager

The Semaphore Manager utilizes standard Dijkstra counting semaphores to provide synchronization and mutual exclusion capabilities.

### Functions

- rtems_status_code rtems_semaphore_create (rtems_name name, uint32_t count, rtems_attribute attribute↩ _set, rtems_task_priority priority_ceiling, rtems_id *id)

  *Creates a semaphore with the specified properties and returns its identifier.*
- rtems_status_code rtems_semaphore_delete (rtems_id id)

  %
- rtems_status_code rtems_semaphore_flush (rtems_id id)

  %
- rtems_status_code rtems_semaphore_ident (rtems_name name, uint32_t node, rtems_id *id)

  *Identifies a semaphore object by the specified object name.*
- rtems_status_code rtems_semaphore_obtain (rtems_id id, rtems_option option_set, rtems_interval timeout)

  %
- rtems_status_code rtems_semaphore_release (rtems_id id)

  %
- rtems_status_code rtems_semaphore_set_priority (rtems_id semaphore_id, rtems_id scheduler_id, rtems_task_priority new_priority, rtems_task_priority *old_priority)

  %

### 8.139.1 Detailed Description

The Semaphore Manager utilizes standard Dijkstra counting semaphores to provide synchronization and mutual exclusion capabilities.

### 8.139.2 Function Documentation

#### 8.139.2.1 rtems_semaphore_create()

```
rtems_status_code rtems_semaphore_create (
            rtems_name name,
            uint32_t count,
            rtems_attribute attribute_set,
            rtems_task_priority priority_ceiling,
            rtems_id * id )
```

Creates a semaphore with the specified properties and returns its identifier.

This directive creates a semaphore which resides on the local node. The new semaphore has the user-defined name specified in `name` and the initial count specified in `count`. For control and maintenance of the semaphore, RTEMS allocates and initializes a SMCB. The RTEMS-assigned semaphore identifier is returned in `id`. This semaphore identifier is used with other semaphore related directives to access the semaphore.

The attribute set specified in `attribute_set` defines

- the scope of the semaphore (local or global),
- the discipline of the task wait queue used by the semaphore (FIFO or priority),
- the class of the semaphore (counting, binary, or simple binary), and
- the locking protocol of a binary semaphore (priority inheritance, priority ceiling or MrsP).

The attribute set is built through a *bitwise or* of the attribute constants described below. Not all combinations of attributes are allowed. Some attributes are mutually exclusive. If mutually exclusive attributes are combined, the behaviour is undefined.

The *scope of a semaphore* is either the local node only (local scope) or all nodes in a multiprocessing network (global scope). The scope is selected by the mutually exclusive RTEMS_LOCAL and RTEMS_GLOBAL attributes.

- The local scope is the default and can be emphasized through use of the RTEMS_LOCAL attribute.
- The global scope is selected by the RTEMS_GLOBAL attribute. In a single node system and the local and global scope are identical.

The *task wait queue discipline* is selected by the mutually exclusive RTEMS_FIFO and RTEMS_PRIORITY attributes.

- The FIFO discipline is the default and can be emphasized through use of the RTEMS_FIFO attribute.
- The priority discipline is selected by the RTEMS_PRIORITY attribute. Some locking protocols require the priority discipline.

The *semaphore class* is selected by the mutually exclusive RTEMS_COUNTING_SEMAPHORE, RTEMS_BINARY_SEMAPHORE, and RTEMS_SIMPLE_BINARY_SEMAPHORE attributes.

- Counting semaphores are the default and can be emphasized through use of the RTEMS_COUNTING_SEMAPHORE attribute.
- Binary semaphores are mutual exclusion (mutex) synchronization primitives which may have an owner. The count of a binary semaphore is restricted to 0 and 1. The binary semaphore class is selected by the RTEMS_BINARY_SEMAPHORE attribute.
- Simple binary semaphores have no owner. The count of a simple binary semaphore is restricted to 0 and 1. They may be used for task and interrupt synchronization. The simple binary semaphore class is selected by the RTEMS_SIMPLE_BINARY_SEMAPHORE attribute.

Binary semaphores may use a *locking protocol*. If a locking protocol is selected, then the scope shall be local and the priority task wait queue discipline shall be selected. The locking protocol is selected by the mutually exclusive RTEMS_INHERIT_PRIORITY, RTEMS_PRIORITY_CEILING, and RTEMS_MULTIPROCESSOR_RESOURCE_SHARING attributes.

- The default is to use no locking protocol.
- The RTEMS_INHERIT_PRIORITY attribute selects the priority inheritance locking protocol.
- The RTEMS_PRIORITY_CEILING attribute selects the priority ceiling locking protocol. For this locking protocol a priority ceiling shall be specified in `priority_ceiling`.

- The RTEMS_MULTIPROCESSOR_RESOURCE_SHARING attribute selects the MrsP locking protocol in SMP configurations, otherwise it selects the priority ceiling protocol. For this locking protocol a priority ceiling shall be specified in `priority_ceiling`. This priority is used to set the priority ceiling in all scheduler instances. This can be changed later with the rtems_semaphore_set_priority() directive using the returned semaphore identifier.

This directive may cause the calling task to be preempted due to an obtain and release of the object allocator mutex.

Semaphores should not be made global unless remote tasks must interact with the new semaphore. This is to avoid the system overhead incurred by the creation of a global semaphore. When a global semaphore is created, the semaphore's name and identifier must be transmitted to every node in the system for insertion in the local copy of the global object table.

The total number of global objects, including semaphores, is limited by the CONFIGURE_MP_MAXIMUM_GLOBAL_OBJECTS application configuration option.

It is not allowed to create an initially locked MrsP semaphore and the RTEMS_INVALID_NUMBER status code will be returned in SMP configurations in this case. This prevents lock order reversal problems with the allocator mutex.

**Parameters**

| | | |
|---|---|---|
| | *name* | is the object name of the new semaphore. |
| | *count* | is the initial count of the new semaphore. If the semaphore is a mutex, then a count of 0 will make the calling task the owner of the new mutex and a count of 1 will create a mutex without an owner. |
| | *attribute_set* | is the attribute set which defines the properties of the new semaphore. |
| | *priority_ceiling* | is the priority ceiling if the new semaphore is a binary semaphore with the priority ceiling or MrsP semaphore locking protocol as defined by the attribute set. |
| `out` | *id* | is the pointer to an object identifier variable. The object identifier of the new semaphore will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The priority_ceiling parameter was NULL. |
| *RTEMS_INVALID_NAME* | The semaphore name was invalid. |
| *RTEMS_INVALID_PRIORITY* | The priority ceiling was invalid. |
| *RTEMS_NOT_DEFINED* | The attribute set was invalid. |
| *RTEMS_TOO_MANY* | There was no inactive semaphore object available to create a new semaphore. The semaphore object maximum is defined by the CONFIGURE_MAXIMUM_SEMAPHORES application configuration option. |
| *RTEMS_TOO_MANY* | In multiprocessing configurations, there was no inactive global object available to create a new global semaphore. |

Definition at line 34 of file semcreate.c.

### 8.139.2.2 rtems_semaphore_delete()

```
rtems_status_code rtems_semaphore_delete (
            rtems_id id )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |

Definition at line 24 of file semdelete.c.

### 8.139.2.3 rtems_semaphore_flush()

```
rtems_status_code rtems_semaphore_flush (
            rtems_id id )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |

### 8.139.2.4 rtems_semaphore_ident()

```
rtems_status_code rtems_semaphore_ident (
            rtems_name name,
            uint32_t node,
            rtems_id * id )
```

Identifies a semaphore object by the specified object name.

This directive obtains the semaphore identifier associated with the semaphore name specified in `name`.

The node to search is specified in `node`. It shall be

- a valid node number,
- the constant RTEMS_SEARCH_ALL_NODES to search in all nodes,
- the constant RTEMS_SEARCH_LOCAL_NODE to search in the local node only, or
- the constant RTEMS_SEARCH_OTHER_NODES to search in all nodes except the local node.

If the semaphore name is not unique, then the semaphore identifier will match the first semaphore with that name in the search order. However, this semaphore identifier is not guaranteed to correspond to the desired semaphore. The semaphore identifier is used with other semaphore related directives to access the semaphore.

If node is RTEMS_SEARCH_ALL_NODES, all nodes are searched with the local node being searched first. All other nodes are searched with the lowest numbered node searched first.

If node is a valid node number which does not represent the local node, then only the semaphores exported by the designated node are searched.

This directive does not generate activity on remote nodes. It accesses only the local copy of the global object table.

**Parameters**

| | | |
|---|---|---|
| | *name* | is the object name to look up. |
| | *node* | is the node or node set to search for a matching object. |
| out | *id* | is the pointer to an object identifier variable. The object identifier of an object with the specified name will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The id parameter was NULL. |
| *RTEMS_INVALID_NAME* | The name parameter was 0. |
| *RTEMS_INVALID_NAME* | There was no object with the specified name on the specified nodes. |
| *RTEMS_INVALID_NODE* | In multiprocessing configurations, the specified node was invalid. |

Definition at line 44 of file semident.c.

#### 8.139.2.5 rtems_semaphore_obtain()

```
rtems_status_code rtems_semaphore_obtain (
            rtems_id id,
            rtems_option option_set,
            rtems_interval timeout )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *option_set* | % |
| *timeout* | % |

Definition at line 51 of file semobtain.c.

#### 8.139.2.6 rtems_semaphore_release()

```
rtems_status_code rtems_semaphore_release (
            rtems_id id )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |

Definition at line 28 of file semrelease.c.

#### 8.139.2.7 rtems_semaphore_set_priority()

```
rtems_status_code rtems_semaphore_set_priority (
             rtems_id semaphore_id,
             rtems_id scheduler_id,
             rtems_task_priority new_priority,
             rtems_task_priority * old_priority )
```

%

**Parameters**

| | |
|---|---|
| *semaphore↩ _id* | % |
| *scheduler_id* | % |
| *new_priority* | % |
| *old_priority* | % |

Definition at line 127 of file semsetpriority.c.

# 8.140 Semaphore Manager Implementation

## Files

- file semdata.h

  *Classic Semaphore Manager Data Structures.*
- file semimpl.h

  *Classic Semaphore Manager Implementation.*
- file semident.c

  *rtems_semaphore_ident() Implementation*

## Classes

- struct Semaphore_Control

## Macros

- #define SEMAPHORE_INFORMATION_DEFINE(max, scheduler_count)

  *Macro to define the objects information for the Classic Semaphore objects.*

## Enumerations

- enum Semaphore_Variant {
  **SEMAPHORE_VARIANT_MUTEX_INHERIT_PRIORITY**, **SEMAPHORE_VARIANT_MUTEX_PRIORITY_CEILING**, **SEMAPHORE_VARIANT_MUTEX_NO_PROTOCOL**, **SEMAPHORE_VARIANT_SIMPLE_BINARY**,
  **SEMAPHORE_VARIANT_COUNTING**, **SEMAPHORE_VARIANT_MRSP** }

  *Classic semaphore variants.*
- enum **Semaphore_Discipline** { **SEMAPHORE_DISCIPLINE_PRIORITY**, **SEMAPHORE_DISCIPLINE_FIFO** }

## Functions

- static __inline__ uintptr_t **_Semaphore_Get_flags** (const Semaphore_Control *the_semaphore)
- static __inline__ void **_Semaphore_Set_flags** (Semaphore_Control *the_semaphore, uintptr_t flags)
- static __inline__ Semaphore_Variant **_Semaphore_Get_variant** (uintptr_t flags)
- static __inline__ uintptr_t **_Semaphore_Set_variant** (uintptr_t flags, Semaphore_Variant variant)
- static __inline__ Semaphore_Discipline **_Semaphore_Get_discipline** (uintptr_t flags)
- static __inline__ uintptr_t **_Semaphore_Set_discipline** (uintptr_t flags, Semaphore_Discipline discipline)
- static __inline__ const Thread_queue_Operations * **_Semaphore_Get_operations** (uintptr_t flags)
- static __inline__ Semaphore_Control * _Semaphore_Allocate (void)

  *Allocates a semaphore control block from the inactive chain of free semaphore control blocks.*
- static __inline__ void _Semaphore_Free (Semaphore_Control *the_semaphore)

  *Frees a semaphore control block to the inactive chain of free semaphore control blocks.*
- static __inline__ Semaphore_Control * **_Semaphore_Get** (Objects_Id id, Thread_queue_Context *queue_context)

## Variables

- Objects_Information _Semaphore_Information

  *The Classic Semaphore objects information.*

## 8.140.1 Detailed Description

## 8.140.2 Macro Definition Documentation

### 8.140.2.1 SEMAPHORE_INFORMATION_DEFINE

```
#define SEMAPHORE_INFORMATION_DEFINE(
              max,
              scheduler_count )
```

**Value:**

```
  typedef union { \
    Objects_Control Object; \
    Semaphore_Control Semaphore; \
    struct { \
      Objects_Control Object; \
      MRSP_Control Control; \
      Priority_Control ceiling_priorities[ scheduler_count ]; \
    } MRSP; \
  } Semaphore_Configured_control; \
  OBJECTS_INFORMATION_DEFINE( \
    _Semaphore, \
    OBJECTS_CLASSIC_API, \
    OBJECTS_RTEMS_SEMAPHORES, \
    Semaphore_Configured_control, \
    max, \
    OBJECTS_NO_STRING_NAME, \
    _Semaphore_MP_Send_extract_proxy \
  )
```

Macro to define the objects information for the Classic Semaphore objects.

This macro should only be used by <rtems/confdefs.h>.

**Parameters**

| | |
|---|---|
| *max* | The configured object maximum (the OBJECTS_UNLIMITED_OBJECTS flag may be set). |
| *scheduler_count* | The configured scheduler count (only used in SMP configurations). |

Definition at line 116 of file semdata.h.

## 8.140.3 Enumeration Type Documentation

#### 8.140.3.1 Semaphore_Variant

```
enum Semaphore_Variant
```

Classic semaphore variants.

Must be in synchronization with Semaphore_Control::variant.

Definition at line 42 of file semimpl.h.

### 8.140.4 Function Documentation

#### 8.140.4.1 _Semaphore_Allocate()

```
static __inline__ Semaphore_Control* _Semaphore_Allocate (
            void  )  [static]
```

Allocates a semaphore control block from the inactive chain of free semaphore control blocks.

This function allocates a semaphore control block from the inactive chain of free semaphore control blocks.

Definition at line 144 of file semimpl.h.

#### 8.140.4.2 _Semaphore_Free()

```
static __inline__ void _Semaphore_Free (
            Semaphore_Control * the_semaphore )  [static]
```

Frees a semaphore control block to the inactive chain of free semaphore control blocks.

This routine frees a semaphore control block to the inactive chain of free semaphore control blocks.

Definition at line 156 of file semimpl.h.

## 8.141 Serial Mouse

Serial Mouse.

Serial Mouse.

## 8.142 Shared

This group contains the architecture-independent support modules shared across all BSPs.

### Modules

- BSP Interrupt Support

  *Generic BSP Interrupt Support.*
- Bootcard

  *Standard system startup.*
- Clock Support

  *Clock support.*
- Console Driver Support

  *Console Driver Support for Board Support Packages.*
- DEFAULT_INITIAL_EXTENSION Support

  *DEFAULT_INITIAL_EXTENSION Support Package.*
- GRLIB

  *Driver support for GRLIB IP Library.*

### 8.142.1 Detailed Description

This group contains the architecture-independent support modules shared across all BSPs.

## 8.143 Shared

Shared Support for SPARC Board Support Packages.

### Files

- file bsp_fatal_exit.c

  *ERC32/LEON2/LEON3 BSP specific exit handler.*

### 8.143.1 Detailed Description

Shared Support for SPARC Board Support Packages.

## 8.144 Signal Manager

The Signal Manager provides the capabilities required for asynchronous communication.

### Macros

- #define RTEMS_SIGNAL_0 0x00000001

  *This constant defines the bit in the signal set associated with signal 0.*
- #define RTEMS_SIGNAL_1 0x00000002

  *This constant defines the bit in the signal set associated with signal 1.*
- #define RTEMS_SIGNAL_10 0x00000400

  *This constant defines the bit in the signal set associated with signal 10.*
- #define RTEMS_SIGNAL_11 0x00000800

  *This constant defines the bit in the signal set associated with signal 11.*
- #define RTEMS_SIGNAL_12 0x00001000

  *This constant defines the bit in the signal set associated with signal 12.*
- #define RTEMS_SIGNAL_13 0x00002000

  *This constant defines the bit in the signal set associated with signal 13.*
- #define RTEMS_SIGNAL_14 0x00004000

  *This constant defines the bit in the signal set associated with signal 14.*
- #define RTEMS_SIGNAL_15 0x00008000

  *This constant defines the bit in the signal set associated with signal 15.*
- #define RTEMS_SIGNAL_16 0x00010000

  *This constant defines the bit in the signal set associated with signal 16.*
- #define RTEMS_SIGNAL_17 0x00020000

  *This constant defines the bit in the signal set associated with signal 17.*
- #define RTEMS_SIGNAL_18 0x00040000

  *This constant defines the bit in the signal set associated with signal 18.*
- #define RTEMS_SIGNAL_19 0x00080000

  *This constant defines the bit in the signal set associated with signal 19.*
- #define RTEMS_SIGNAL_2 0x00000004

  *This constant defines the bit in the signal set associated with signal 2.*
- #define RTEMS_SIGNAL_20 0x00100000

  *This constant defines the bit in the signal set associated with signal 20.*
- #define RTEMS_SIGNAL_21 0x00200000

  *This constant defines the bit in the signal set associated with signal 21.*
- #define RTEMS_SIGNAL_22 0x00400000

  *This constant defines the bit in the signal set associated with signal 22.*
- #define RTEMS_SIGNAL_23 0x00800000

  *This constant defines the bit in the signal set associated with signal 23.*
- #define RTEMS_SIGNAL_24 0x01000000

  *This constant defines the bit in the signal set associated with signal 24.*
- #define RTEMS_SIGNAL_25 0x02000000

  *This constant defines the bit in the signal set associated with signal 25.*
- #define RTEMS_SIGNAL_26 0x04000000

  *This constant defines the bit in the signal set associated with signal 26.*
- #define RTEMS_SIGNAL_27 0x08000000

  *This constant defines the bit in the signal set associated with signal 27.*
- #define RTEMS_SIGNAL_28 0x10000000

*This constant defines the bit in the signal set associated with signal 28.*

- #define RTEMS_SIGNAL_29 0x20000000

  *This constant defines the bit in the signal set associated with signal 29.*
- #define RTEMS_SIGNAL_3 0x00000008

  *This constant defines the bit in the signal set associated with signal 3.*
- #define RTEMS_SIGNAL_30 0x40000000

  *This constant defines the bit in the signal set associated with signal 30.*
- #define RTEMS_SIGNAL_31 0x80000000

  *This constant defines the bit in the signal set associated with signal 31.*
- #define RTEMS_SIGNAL_4 0x00000010

  *This constant defines the bit in the signal set associated with signal 4.*
- #define RTEMS_SIGNAL_5 0x00000020

  *This constant defines the bit in the signal set associated with signal 5.*
- #define RTEMS_SIGNAL_6 0x00000040

  *This constant defines the bit in the signal set associated with signal 6.*
- #define RTEMS_SIGNAL_7 0x00000080

  *This constant defines the bit in the signal set associated with signal 7.*
- #define RTEMS_SIGNAL_8 0x00000100

  *This constant defines the bit in the signal set associated with signal 8.*
- #define RTEMS_SIGNAL_9 0x00000200

  *This constant defines the bit in the signal set associated with signal 9.*

## Typedefs

- typedef void rtems_asr

  %
- typedef uint32_t rtems_signal_set

  %
- typedef rtems_asr(∗ rtems_asr_entry) (rtems_signal_set)

  %

## Functions

- rtems_status_code rtems_signal_catch (rtems_asr_entry asr_handler, rtems_mode mode_set)

  %
- rtems_status_code rtems_signal_send (rtems_id id, rtems_signal_set signal_set)

  %

### 8.144.1 Detailed Description

The Signal Manager provides the capabilities required for asynchronous communication.

### 8.144.2 Function Documentation

#### 8.144.2.1 rtems_signal_catch()

```
rtems_status_code rtems_signal_catch (
            rtems_asr_entry asr_handler,
            rtems_mode mode_set )
```

%

**Parameters**

| | |
|---|---|
| *asr_handler* | % |
| *mode_set* | % |

### 8.144.2.2 rtems_signal_send()

```
rtems_status_code rtems_signal_send (
            rtems_id id,
            rtems_signal_set signal_set )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *signal_set* | % |

# 8.145 Signals Implementation

## Files

- file signalimpl.h

  *Signals Implementation.*

## Functions

- void **_Signal_Action_handler** (Thread_Control *executing, Thread_Action *action, ISR_lock_Context *lock_context)

### 8.145.1 Detailed Description

## 8.146 Signed 16-Bit Integer Checks

Checks for signed 16-bit integers (int16_t).

### Macros

- #define **T_eq_i16**(a, e) T_flags_eq_int(a, e, 0)
- #define **T_assert_eq_i16**(a, e) T_flags_eq_int(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_i16**(a, e) T_flags_eq_int(a, e, T_CHECK_QUIET)
- #define **T_step_eq_i16**(s, a, e) T_flags_eq_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_i16**(s, a, e) T_flags_eq_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_i16**(a, e) T_flags_ne_int(a, e, 0)
- #define **T_assert_ne_i16**(a, e) T_flags_ne_int(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_i16**(a, e) T_flags_ne_int(a, e, T_CHECK_QUIET)
- #define **T_step_ne_i16**(s, a, e) T_flags_ne_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_i16**(s, a, e) T_flags_ne_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_i16**(a, e) T_flags_ge_int(a, e, 0)
- #define **T_assert_ge_i16**(a, e) T_flags_ge_int(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_i16**(a, e) T_flags_ge_int(a, e, T_CHECK_QUIET)
- #define **T_step_ge_i16**(s, a, e) T_flags_ge_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_i16**(s, a, e) T_flags_ge_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_i16**(a, e) T_flags_gt_int(a, e, 0)
- #define **T_assert_gt_i16**(a, e) T_flags_gt_int(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_i16**(a, e) T_flags_gt_int(a, e, T_CHECK_QUIET)
- #define **T_step_gt_i16**(s, a, e) T_flags_gt_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_i16**(s, a, e) T_flags_gt_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_i16**(a, e) T_flags_le_int(a, e, 0)
- #define **T_assert_le_i16**(a, e) T_flags_le_int(a, e, T_CHECK_STOP)
- #define **T_quiet_le_i16**(a, e) T_flags_le_int(a, e, T_CHECK_QUIET)
- #define **T_step_le_i16**(s, a, e) T_flags_le_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_i16**(s, a, e) T_flags_le_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_i16**(a, e) T_flags_lt_int(a, e, 0)
- #define **T_assert_lt_i16**(a, e) T_flags_lt_int(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_i16**(a, e) T_flags_lt_int(a, e, T_CHECK_QUIET)
- #define **T_step_lt_i16**(s, a, e) T_flags_lt_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_i16**(s, a, e) T_flags_lt_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.146.1 Detailed Description

Checks for signed 16-bit integers (int16_t).

## 8.147 Signed 32-Bit Integer Checks

Checks for signed 32-bit integers (int32_t).

### Macros

- #define **T_eq_i32**(a, e) T_flags_eq_long(a, e, 0)
- #define **T_assert_eq_i32**(a, e) T_flags_eq_long(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_i32**(a, e) T_flags_eq_long(a, e, T_CHECK_QUIET)
- #define **T_step_eq_i32**(s, a, e) T_flags_eq_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_i32**(s, a, e) T_flags_eq_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_i32**(a, e) T_flags_ne_long(a, e, 0)
- #define **T_assert_ne_i32**(a, e) T_flags_ne_long(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_i32**(a, e) T_flags_ne_long(a, e, T_CHECK_QUIET)
- #define **T_step_ne_i32**(s, a, e) T_flags_ne_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_i32**(s, a, e) T_flags_ne_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_i32**(a, e) T_flags_ge_long(a, e, 0)
- #define **T_assert_ge_i32**(a, e) T_flags_ge_long(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_i32**(a, e) T_flags_ge_long(a, e, T_CHECK_QUIET)
- #define **T_step_ge_i32**(s, a, e) T_flags_ge_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_i32**(s, a, e) T_flags_ge_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_i32**(a, e) T_flags_gt_long(a, e, 0)
- #define **T_assert_gt_i32**(a, e) T_flags_gt_long(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_i32**(a, e) T_flags_gt_long(a, e, T_CHECK_QUIET)
- #define **T_step_gt_i32**(s, a, e) T_flags_gt_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_i32**(s, a, e) T_flags_gt_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_i32**(a, e) T_flags_le_long(a, e, 0)
- #define **T_assert_le_i32**(a, e) T_flags_le_long(a, e, T_CHECK_STOP)
- #define **T_quiet_le_i32**(a, e) T_flags_le_long(a, e, T_CHECK_QUIET)
- #define **T_step_le_i32**(s, a, e) T_flags_le_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_i32**(s, a, e) T_flags_le_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_i32**(a, e) T_flags_lt_long(a, e, 0)
- #define **T_assert_lt_i32**(a, e) T_flags_lt_long(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_i32**(a, e) T_flags_lt_long(a, e, T_CHECK_QUIET)
- #define **T_step_lt_i32**(s, a, e) T_flags_lt_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_i32**(s, a, e) T_flags_lt_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.147.1 Detailed Description

Checks for signed 32-bit integers (int32_t).

## 8.148 Signed 64-Bit Integer Checks

Checks for signed 64-bit integers (int64_t).

### Macros

- #define **T_eq_i64**(a, e) T_flags_eq_ll(a, e, 0)
- #define **T_assert_eq_i64**(a, e) T_flags_eq_ll(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_i64**(a, e) T_flags_eq_ll(a, e, T_CHECK_QUIET)
- #define **T_step_eq_i64**(s, a, e) T_flags_eq_ll(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_i64**(s, a, e) T_flags_eq_ll(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_i64**(a, e) T_flags_ne_ll(a, e, 0)
- #define **T_assert_ne_i64**(a, e) T_flags_ne_ll(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_i64**(a, e) T_flags_ne_ll(a, e, T_CHECK_QUIET)
- #define **T_step_ne_i64**(s, a, e) T_flags_ne_ll(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_i64**(s, a, e) T_flags_ne_ll(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_i64**(a, e) T_flags_ge_ll(a, e, 0)
- #define **T_assert_ge_i64**(a, e) T_flags_ge_ll(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_i64**(a, e) T_flags_ge_ll(a, e, T_CHECK_QUIET)
- #define **T_step_ge_i64**(s, a, e) T_flags_ge_ll(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_i64**(s, a, e) T_flags_ge_ll(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_i64**(a, e) T_flags_gt_ll(a, e, 0)
- #define **T_assert_gt_i64**(a, e) T_flags_gt_ll(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_i64**(a, e) T_flags_gt_ll(a, e, T_CHECK_QUIET)
- #define **T_step_gt_i64**(s, a, e) T_flags_gt_ll(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_i64**(s, a, e) T_flags_gt_ll(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_i64**(a, e) T_flags_le_ll(a, e, 0)
- #define **T_assert_le_i64**(a, e) T_flags_le_ll(a, e, T_CHECK_STOP)
- #define **T_quiet_le_i64**(a, e) T_flags_le_ll(a, e, T_CHECK_QUIET)
- #define **T_step_le_i64**(s, a, e) T_flags_le_ll(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_i64**(s, a, e) T_flags_le_ll(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_i64**(a, e) T_flags_lt_ll(a, e, 0)
- #define **T_assert_lt_i64**(a, e) T_flags_lt_ll(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_i64**(a, e) T_flags_lt_ll(a, e, T_CHECK_QUIET)
- #define **T_step_lt_i64**(s, a, e) T_flags_lt_ll(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_i64**(s, a, e) T_flags_lt_ll(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.148.1 Detailed Description

Checks for signed 64-bit integers (int64_t).

## 8.149 Signed 8-Bit Integer Checks

Checks for signed 8-bit integers (int8_t).

### Macros

- #define **T_eq_i8**(a, e) T_flags_eq_int(a, e, 0)
- #define **T_assert_eq_i8**(a, e) T_flags_eq_int(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_i8**(a, e) T_flags_eq_int(a, e, T_CHECK_QUIET)
- #define **T_step_eq_i8**(s, a, e) T_flags_eq_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_i8**(s, a, e) T_flags_eq_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_i8**(a, e) T_flags_ne_int(a, e, 0)
- #define **T_assert_ne_i8**(a, e) T_flags_ne_int(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_i8**(a, e) T_flags_ne_int(a, e, T_CHECK_QUIET)
- #define **T_step_ne_i8**(s, a, e) T_flags_ne_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_i8**(s, a, e) T_flags_ne_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_i8**(a, e) T_flags_ge_int(a, e, 0)
- #define **T_assert_ge_i8**(a, e) T_flags_ge_int(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_i8**(a, e) T_flags_ge_int(a, e, T_CHECK_QUIET)
- #define **T_step_ge_i8**(s, a, e) T_flags_ge_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_i8**(s, a, e) T_flags_ge_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_i8**(a, e) T_flags_gt_int(a, e, 0)
- #define **T_assert_gt_i8**(a, e) T_flags_gt_int(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_i8**(a, e) T_flags_gt_int(a, e, T_CHECK_QUIET)
- #define **T_step_gt_i8**(s, a, e) T_flags_gt_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_i8**(s, a, e) T_flags_gt_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_i8**(a, e) T_flags_le_int(a, e, 0)
- #define **T_assert_le_i8**(a, e) T_flags_le_int(a, e, T_CHECK_STOP)
- #define **T_quiet_le_i8**(a, e) T_flags_le_int(a, e, T_CHECK_QUIET)
- #define **T_step_le_i8**(s, a, e) T_flags_le_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_i8**(s, a, e) T_flags_le_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_i8**(a, e) T_flags_lt_int(a, e, 0)
- #define **T_assert_lt_i8**(a, e) T_flags_lt_int(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_i8**(a, e) T_flags_lt_int(a, e, T_CHECK_QUIET)
- #define **T_step_lt_i8**(s, a, e) T_flags_lt_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_i8**(s, a, e) T_flags_lt_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.149.1 Detailed Description

Checks for signed 8-bit integers (int8_t).

## 8.150 Signed Character Checks

Checks for signed characters (signed char).

### Macros

- #define **T_eq_schar**(a, e) T_flags_eq_int(a, e, 0)
- #define **T_assert_eq_schar**(a, e) T_flags_eq_int(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_schar**(a, e) T_flags_eq_int(a, e, T_CHECK_QUIET)
- #define **T_step_eq_schar**(s, a, e) T_flags_eq_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_schar**(s, a, e) T_flags_eq_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_schar**(a, e) T_flags_ne_int(a, e, 0)
- #define **T_assert_ne_schar**(a, e) T_flags_ne_int(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_schar**(a, e) T_flags_ne_int(a, e, T_CHECK_QUIET)
- #define **T_step_ne_schar**(s, a, e) T_flags_ne_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_schar**(s, a, e) T_flags_ne_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_schar**(a, e) T_flags_ge_int(a, e, 0)
- #define **T_assert_ge_schar**(a, e) T_flags_ge_int(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_schar**(a, e) T_flags_ge_int(a, e, T_CHECK_QUIET)
- #define **T_step_ge_schar**(s, a, e) T_flags_ge_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_schar**(s, a, e) T_flags_ge_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_schar**(a, e) T_flags_gt_int(a, e, 0)
- #define **T_assert_gt_schar**(a, e) T_flags_gt_int(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_schar**(a, e) T_flags_gt_int(a, e, T_CHECK_QUIET)
- #define **T_step_gt_schar**(s, a, e) T_flags_gt_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_schar**(s, a, e) T_flags_gt_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_schar**(a, e) T_flags_le_int(a, e, 0)
- #define **T_assert_le_schar**(a, e) T_flags_le_int(a, e, T_CHECK_STOP)
- #define **T_quiet_le_schar**(a, e) T_flags_le_int(a, e, T_CHECK_QUIET)
- #define **T_step_le_schar**(s, a, e) T_flags_le_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_schar**(s, a, e) T_flags_le_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_schar**(a, e) T_flags_lt_int(a, e, 0)
- #define **T_assert_lt_schar**(a, e) T_flags_lt_int(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_schar**(a, e) T_flags_lt_int(a, e, T_CHECK_QUIET)
- #define **T_step_lt_schar**(s, a, e) T_flags_lt_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_schar**(s, a, e) T_flags_lt_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.150.1 Detailed Description

Checks for signed characters (signed char).

## 8.151 Signed Integer Checks

Checks for signed integers (int).

### Macros

- #define **T_eq_int**(a, e) T_flags_eq_int(a, e, 0)
- #define **T_assert_eq_int**(a, e) T_flags_eq_int(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_int**(a, e) T_flags_eq_int(a, e, T_CHECK_QUIET)
- #define **T_step_eq_int**(s, a, e) T_flags_eq_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_int**(s, a, e) T_flags_eq_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_int**(a, e) T_flags_ne_int(a, e, 0)
- #define **T_assert_ne_int**(a, e) T_flags_ne_int(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_int**(a, e) T_flags_ne_int(a, e, T_CHECK_QUIET)
- #define **T_step_ne_int**(s, a, e) T_flags_ne_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_int**(s, a, e) T_flags_ne_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_int**(a, e) T_flags_ge_int(a, e, 0)
- #define **T_assert_ge_int**(a, e) T_flags_ge_int(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_int**(a, e) T_flags_ge_int(a, e, T_CHECK_QUIET)
- #define **T_step_ge_int**(s, a, e) T_flags_ge_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_int**(s, a, e) T_flags_ge_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_int**(a, e) T_flags_gt_int(a, e, 0)
- #define **T_assert_gt_int**(a, e) T_flags_gt_int(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_int**(a, e) T_flags_gt_int(a, e, T_CHECK_QUIET)
- #define **T_step_gt_int**(s, a, e) T_flags_gt_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_int**(s, a, e) T_flags_gt_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_int**(a, e) T_flags_le_int(a, e, 0)
- #define **T_assert_le_int**(a, e) T_flags_le_int(a, e, T_CHECK_STOP)
- #define **T_quiet_le_int**(a, e) T_flags_le_int(a, e, T_CHECK_QUIET)
- #define **T_step_le_int**(s, a, e) T_flags_le_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_int**(s, a, e) T_flags_le_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_int**(a, e) T_flags_lt_int(a, e, 0)
- #define **T_assert_lt_int**(a, e) T_flags_lt_int(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_int**(a, e) T_flags_lt_int(a, e, T_CHECK_QUIET)
- #define **T_step_lt_int**(s, a, e) T_flags_lt_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_int**(s, a, e) T_flags_lt_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.151.1 Detailed Description

Checks for signed integers (int).

## 8.152 Signed Long Integer Checks

Checks for signed long integers (long).

### Macros

- #define **T_eq_long**(a, e) T_flags_eq_long(a, e, 0)
- #define **T_assert_eq_long**(a, e) T_flags_eq_long(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_long**(a, e) T_flags_eq_long(a, e, T_CHECK_QUIET)
- #define **T_step_eq_long**(s, a, e) T_flags_eq_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_long**(s, a, e) T_flags_eq_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_long**(a, e) T_flags_ne_long(a, e, 0)
- #define **T_assert_ne_long**(a, e) T_flags_ne_long(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_long**(a, e) T_flags_ne_long(a, e, T_CHECK_QUIET)
- #define **T_step_ne_long**(s, a, e) T_flags_ne_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_long**(s, a, e) T_flags_ne_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_long**(a, e) T_flags_ge_long(a, e, 0)
- #define **T_assert_ge_long**(a, e) T_flags_ge_long(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_long**(a, e) T_flags_ge_long(a, e, T_CHECK_QUIET)
- #define **T_step_ge_long**(s, a, e) T_flags_ge_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_long**(s, a, e) T_flags_ge_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_long**(a, e) T_flags_gt_long(a, e, 0)
- #define **T_assert_gt_long**(a, e) T_flags_gt_long(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_long**(a, e) T_flags_gt_long(a, e, T_CHECK_QUIET)
- #define **T_step_gt_long**(s, a, e) T_flags_gt_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_long**(s, a, e) T_flags_gt_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_long**(a, e) T_flags_le_long(a, e, 0)
- #define **T_assert_le_long**(a, e) T_flags_le_long(a, e, T_CHECK_STOP)
- #define **T_quiet_le_long**(a, e) T_flags_le_long(a, e, T_CHECK_QUIET)
- #define **T_step_le_long**(s, a, e) T_flags_le_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_long**(s, a, e) T_flags_le_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_long**(a, e) T_flags_lt_long(a, e, 0)
- #define **T_assert_lt_long**(a, e) T_flags_lt_long(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_long**(a, e) T_flags_lt_long(a, e, T_CHECK_QUIET)
- #define **T_step_lt_long**(s, a, e) T_flags_lt_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_long**(s, a, e) T_flags_lt_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_eq_ll**(a, e) T_flags_eq_ll(a, e, 0)
- #define **T_assert_eq_ll**(a, e) T_flags_eq_ll(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_ll**(a, e) T_flags_eq_ll(a, e, T_CHECK_QUIET)
- #define **T_step_eq_ll**(s, a, e) T_flags_eq_ll(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_ll**(s, a, e) T_flags_eq_ll(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_ll**(a, e) T_flags_ne_ll(a, e, 0)
- #define **T_assert_ne_ll**(a, e) T_flags_ne_ll(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_ll**(a, e) T_flags_ne_ll(a, e, T_CHECK_QUIET)
- #define **T_step_ne_ll**(s, a, e) T_flags_ne_ll(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_ll**(s, a, e) T_flags_ne_ll(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_ll**(a, e) T_flags_ge_ll(a, e, 0)
- #define **T_assert_ge_ll**(a, e) T_flags_ge_ll(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_ll**(a, e) T_flags_ge_ll(a, e, T_CHECK_QUIET)
- #define **T_step_ge_ll**(s, a, e) T_flags_ge_ll(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_ll**(s, a, e) T_flags_ge_ll(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_ll**(a, e) T_flags_gt_ll(a, e, 0)
- #define **T_assert_gt_ll**(a, e) T_flags_gt_ll(a, e, T_CHECK_STOP)

- #define **T_quiet_gt_ll**(a, e) T_flags_gt_ll(a, e, T_CHECK_QUIET)
- #define **T_step_gt_ll**(s, a, e) T_flags_gt_ll(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_ll**(s, a, e) T_flags_gt_ll(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_ll**(a, e) T_flags_le_ll(a, e, 0)
- #define **T_assert_le_ll**(a, e) T_flags_le_ll(a, e, T_CHECK_STOP)
- #define **T_quiet_le_ll**(a, e) T_flags_le_ll(a, e, T_CHECK_QUIET)
- #define **T_step_le_ll**(s, a, e) T_flags_le_ll(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_ll**(s, a, e) T_flags_le_ll(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_ll**(a, e) T_flags_lt_ll(a, e, 0)
- #define **T_assert_lt_ll**(a, e) T_flags_lt_ll(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_ll**(a, e) T_flags_lt_ll(a, e, T_CHECK_QUIET)
- #define **T_step_lt_ll**(s, a, e) T_flags_lt_ll(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_ll**(s, a, e) T_flags_lt_ll(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.152.1 Detailed Description

Checks for signed long integers (long).

Checks for signed long long integers (long long).

## 8.153 Signed Pointer Value Checks

Checks for signed pointer values (intptr_t).

### Macros

- #define **T_eq_iptr**(a, e) T_flags_eq_long(a, e, 0)
- #define **T_assert_eq_iptr**(a, e) T_flags_eq_long(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_iptr**(a, e) T_flags_eq_long(a, e, T_CHECK_QUIET)
- #define **T_step_eq_iptr**(s, a, e) T_flags_eq_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_iptr**(s, a, e) T_flags_eq_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_iptr**(a, e) T_flags_ne_long(a, e, 0)
- #define **T_assert_ne_iptr**(a, e) T_flags_ne_long(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_iptr**(a, e) T_flags_ne_long(a, e, T_CHECK_QUIET)
- #define **T_step_ne_iptr**(s, a, e) T_flags_ne_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_iptr**(s, a, e) T_flags_ne_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_iptr**(a, e) T_flags_ge_long(a, e, 0)
- #define **T_assert_ge_iptr**(a, e) T_flags_ge_long(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_iptr**(a, e) T_flags_ge_long(a, e, T_CHECK_QUIET)
- #define **T_step_ge_iptr**(s, a, e) T_flags_ge_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_iptr**(s, a, e) T_flags_ge_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_iptr**(a, e) T_flags_gt_long(a, e, 0)
- #define **T_assert_gt_iptr**(a, e) T_flags_gt_long(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_iptr**(a, e) T_flags_gt_long(a, e, T_CHECK_QUIET)
- #define **T_step_gt_iptr**(s, a, e) T_flags_gt_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_iptr**(s, a, e) T_flags_gt_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_iptr**(a, e) T_flags_le_long(a, e, 0)
- #define **T_assert_le_iptr**(a, e) T_flags_le_long(a, e, T_CHECK_STOP)
- #define **T_quiet_le_iptr**(a, e) T_flags_le_long(a, e, T_CHECK_QUIET)
- #define **T_step_le_iptr**(s, a, e) T_flags_le_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_iptr**(s, a, e) T_flags_le_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_iptr**(a, e) T_flags_lt_long(a, e, 0)
- #define **T_assert_lt_iptr**(a, e) T_flags_lt_long(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_iptr**(a, e) T_flags_lt_long(a, e, T_CHECK_QUIET)
- #define **T_step_lt_iptr**(s, a, e) T_flags_lt_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_iptr**(s, a, e) T_flags_lt_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.153.1 Detailed Description

Checks for signed pointer values (intptr_t).

## 8.154 Signed Short Integer Checks

Checks for signed short integers (short).

### Macros

- #define **T_eq_short**(a, e) T_flags_eq_int(a, e, 0)
- #define **T_assert_eq_short**(a, e) T_flags_eq_int(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_short**(a, e) T_flags_eq_int(a, e, T_CHECK_QUIET)
- #define **T_step_eq_short**(s, a, e) T_flags_eq_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_short**(s, a, e) T_flags_eq_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_short**(a, e) T_flags_ne_int(a, e, 0)
- #define **T_assert_ne_short**(a, e) T_flags_ne_int(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_short**(a, e) T_flags_ne_int(a, e, T_CHECK_QUIET)
- #define **T_step_ne_short**(s, a, e) T_flags_ne_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_short**(s, a, e) T_flags_ne_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_short**(a, e) T_flags_ge_int(a, e, 0)
- #define **T_assert_ge_short**(a, e) T_flags_ge_int(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_short**(a, e) T_flags_ge_int(a, e, T_CHECK_QUIET)
- #define **T_step_ge_short**(s, a, e) T_flags_ge_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_short**(s, a, e) T_flags_ge_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_short**(a, e) T_flags_gt_int(a, e, 0)
- #define **T_assert_gt_short**(a, e) T_flags_gt_int(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_short**(a, e) T_flags_gt_int(a, e, T_CHECK_QUIET)
- #define **T_step_gt_short**(s, a, e) T_flags_gt_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_short**(s, a, e) T_flags_gt_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_short**(a, e) T_flags_le_int(a, e, 0)
- #define **T_assert_le_short**(a, e) T_flags_le_int(a, e, T_CHECK_STOP)
- #define **T_quiet_le_short**(a, e) T_flags_le_int(a, e, T_CHECK_QUIET)
- #define **T_step_le_short**(s, a, e) T_flags_le_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_short**(s, a, e) T_flags_le_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_short**(a, e) T_flags_lt_int(a, e, 0)
- #define **T_assert_lt_short**(a, e) T_flags_lt_int(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_short**(a, e) T_flags_lt_int(a, e, T_CHECK_QUIET)
- #define **T_step_lt_short**(s, a, e) T_flags_lt_int(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_short**(s, a, e) T_flags_lt_int(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.154.1 Detailed Description

Checks for signed short integers (short).

## 8.155 Signed Size Checks

Checks for signed sizes (ssize_t).

### Macros

- #define **T_eq_ssz**(a, e) T_flags_eq_long(a, e, 0)
- #define **T_assert_eq_ssz**(a, e) T_flags_eq_long(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_ssz**(a, e) T_flags_eq_long(a, e, T_CHECK_QUIET)
- #define **T_step_eq_ssz**(s, a, e) T_flags_eq_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_ssz**(s, a, e) T_flags_eq_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_ssz**(a, e) T_flags_ne_long(a, e, 0)
- #define **T_assert_ne_ssz**(a, e) T_flags_ne_long(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_ssz**(a, e) T_flags_ne_long(a, e, T_CHECK_QUIET)
- #define **T_step_ne_ssz**(s, a, e) T_flags_ne_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_ssz**(s, a, e) T_flags_ne_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_ssz**(a, e) T_flags_ge_long(a, e, 0)
- #define **T_assert_ge_ssz**(a, e) T_flags_ge_long(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_ssz**(a, e) T_flags_ge_long(a, e, T_CHECK_QUIET)
- #define **T_step_ge_ssz**(s, a, e) T_flags_ge_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_ssz**(s, a, e) T_flags_ge_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_ssz**(a, e) T_flags_gt_long(a, e, 0)
- #define **T_assert_gt_ssz**(a, e) T_flags_gt_long(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_ssz**(a, e) T_flags_gt_long(a, e, T_CHECK_QUIET)
- #define **T_step_gt_ssz**(s, a, e) T_flags_gt_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_ssz**(s, a, e) T_flags_gt_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_ssz**(a, e) T_flags_le_long(a, e, 0)
- #define **T_assert_le_ssz**(a, e) T_flags_le_long(a, e, T_CHECK_STOP)
- #define **T_quiet_le_ssz**(a, e) T_flags_le_long(a, e, T_CHECK_QUIET)
- #define **T_step_le_ssz**(s, a, e) T_flags_le_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_ssz**(s, a, e) T_flags_le_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_ssz**(a, e) T_flags_lt_long(a, e, 0)
- #define **T_assert_lt_ssz**(a, e) T_flags_lt_long(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_ssz**(a, e) T_flags_lt_long(a, e, T_CHECK_QUIET)
- #define **T_step_lt_ssz**(s, a, e) T_flags_lt_long(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_ssz**(s, a, e) T_flags_lt_long(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.155.1 Detailed Description

Checks for signed sizes (ssize_t).

# 8.156 Simple Priority Scheduler

Simple Priority Scheduler.

## Files

- file schedulersimple.h

  *Manipulation of Threads Simple-Priority-Based Ready Queue.*
- file schedulersimpleimpl.h

  *Inlined Routines Associated with the Manipulation of the Priority-Based Scheduling Structures.*

## Classes

- struct Scheduler_simple_Context

  *Simple scheduler context.*

## Macros

- #define **SCHEDULER_SIMPLE_MAXIMUM_PRIORITY** 255
- #define SCHEDULER_SIMPLE_ENTRY_POINTS

## Functions

- void _Scheduler_simple_Initialize (const Scheduler_Control *scheduler)

  *Initializes simple scheduler.*
- void _Scheduler_simple_Schedule (const Scheduler_Control *scheduler, Thread_Control *the_thread)

  *Schedules threads.*
- void _Scheduler_simple_Yield (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Performs the yield of a thread.*
- void _Scheduler_simple_Block (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Blocks the thread.*
- void _Scheduler_simple_Unblock (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Unblocks the thread.*
- void _Scheduler_simple_Update_priority (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Updates the priority of the node.*
- static __inline__ Scheduler_simple_Context * _Scheduler_simple_Get_context (const Scheduler_Control *scheduler)

  *Gets context of the scheduler.*
- static __inline__ bool _Scheduler_simple_Priority_less_equal (const void *to_insert, const Chain_Node *next)

  *Checks if the priority is less or equal than the priority of the node.*
- static __inline__ void _Scheduler_simple_Insert (Chain_Control *chain, Thread_Control *to_insert, unsigned int insert_priority)

  *Inserts the thread control with the given priority into the chain.*
- static __inline__ void _Scheduler_simple_Extract (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Extracts the threads node.*
- static __inline__ void _Scheduler_simple_Schedule_body (const Scheduler_Control *scheduler, Thread_Control *the_thread, bool force_dispatch)

  *Scheduling decision logic.*

## 8.156.1 Detailed Description

Simple Priority Scheduler.

## 8.156.2 Macro Definition Documentation

### 8.156.2.1 SCHEDULER_SIMPLE_ENTRY_POINTS

```
#define SCHEDULER_SIMPLE_ENTRY_POINTS
```

**Value:**

```
{ \
  _Scheduler_simple_Initialize,         /* initialize entry point */ \
  _Scheduler_simple_Schedule,           /* schedule entry point */ \
  _Scheduler_simple_Yield,              /* yield entry point */ \
  _Scheduler_simple_Block,              /* block entry point */ \
  _Scheduler_simple_Unblock,            /* unblock entry point */ \
  _Scheduler_simple_Update_priority,    /* update priority entry point */ \
  _Scheduler_default_Map_priority,      /* map priority entry point */ \
  _Scheduler_default_Unmap_priority,    /* unmap priority entry point */ \
  SCHEDULER_OPERATION_DEFAULT_ASK_FOR_HELP \
  _Scheduler_default_Node_initialize,   /* node initialize entry point */ \
  _Scheduler_default_Node_destroy,      /* node destroy entry point */ \
  _Scheduler_default_Release_job,       /* new period of task */ \
  _Scheduler_default_Cancel_job,        /* cancel period of task */ \
  _Scheduler_default_Tick,              /* tick entry point */ \
  _Scheduler_default_Start_idle         /* start idle entry point */ \
  SCHEDULER_OPERATION_DEFAULT_GET_SET_AFFINITY \
}
```

Entry points for Scheduler Simple

Definition at line 45 of file schedulersimple.h.

## 8.156.3 Function Documentation

### 8.156.3.1 _Scheduler_simple_Block()

```
void _Scheduler_simple_Block (
            const Scheduler_Control * scheduler,
            Thread_Control * the_thread,
            Scheduler_Node * node )
```

Blocks the thread.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| in,out | *the_thread* | The thread to block. |
| in,out | *node* | The *thread's* scheduler node. |

### 8.156.3.2 _Scheduler_simple_Extract()

```
static __inline__ void _Scheduler_simple_Extract (
             const Scheduler_Control * scheduler,
             Thread_Control * the_thread,
             Scheduler_Node * node )  [static]
```

Extracts the threads node.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | This parameter is unused. |
| in,out | *the_thread* | The thread of which to extract the node out of its chain. |
| | *node* | This parameter is unused. |

Definition at line 102 of file schedulersimpleimpl.h.

### 8.156.3.3 _Scheduler_simple_Get_context()

```
static __inline__ Scheduler_simple_Context* _Scheduler_simple_Get_context (
             const Scheduler_Control * scheduler )  [static]
```

Gets context of the scheduler.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance to get the context of. |

**Returns**

The context of *scheduler*.

Definition at line 46 of file schedulersimpleimpl.h.

### 8.156.3.4 _Scheduler_simple_Initialize()

```
void _Scheduler_simple_Initialize (
             const Scheduler_Control * scheduler )
```

Initializes simple scheduler.

This routine initializes the simple scheduler.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler to be initialized. |

#### 8.156.3.5 _Scheduler_simple_Insert()

```
static __inline__ void _Scheduler_simple_Insert (
            Chain_Control * chain,
            Thread_Control * to_insert,
            unsigned int insert_priority )  [static]
```

Inserts the thread control with the given priority into the chain.

**Parameters**

| | | |
|---|---|---|
| in,out | *chain* | The chain to insert *to_insert* in. |
| in,out | *to_insert* | The node to insert into *chain*. |
| | *insert_priority* | The priority to insert *to_insert* with. |

Definition at line 81 of file schedulersimpleimpl.h.

#### 8.156.3.6 _Scheduler_simple_Priority_less_equal()

```
static __inline__ bool _Scheduler_simple_Priority_less_equal (
            const void * to_insert,
            const Chain_Node * next )  [static]
```

Checks if the priority is less or equal than the priority of the node.

**Parameters**

| | |
|---|---|
| *to_insert* | The priority to check whether it is less or equal than *next*. |
| *next* | The Chain node to compare the priority of. |

**Return values**

| | |
|---|---|
| *true* | *to_insert* is smaller or equal than the priority of *next*. |
| *false* | *to_insert* is greater than the priority of *next*. |

Definition at line 60 of file schedulersimpleimpl.h.

### 8.156.3.7 _Scheduler_simple_Schedule()

```
void _Scheduler_simple_Schedule (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread )
```

Schedules threads.

This routine sets the heir thread to be the next ready thread on the ready queue by getting the first node in the scheduler information.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance. |
| *the_thread* | causing the scheduling operation. |

### 8.156.3.8 _Scheduler_simple_Schedule_body()

```
static __inline__ void _Scheduler_simple_Schedule_body (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread,
              bool force_dispatch )  [static]
```

Scheduling decision logic.

This kernel routine implements scheduling decision logic for the simple scheduler.

**Parameters**

| | | |
|---|---|---|
| in,out | *scheduler* | The scheduler instance. |
| | *the_thread* | This parameter is unused. |
| | *force_dispatch* | Indicates whether the dispatch happens also if the currently executing thread is set as not preemptible. |

Definition at line 124 of file schedulersimpleimpl.h.

### 8.156.3.9 _Scheduler_simple_Unblock()

```
void _Scheduler_simple_Unblock (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread,
              Scheduler_Node * node )
```

Unblocks the thread.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| in,out | *the_thread* | The thread to unblock. |
| in,out | *node* | The *thread's* scheduler node. |

#### 8.156.3.10 _Scheduler_simple_Update_priority()

```
void _Scheduler_simple_Update_priority (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread,
              Scheduler_Node * node )
```

Updates the priority of the node.

**Parameters**

| | |
|---|---|
| *scheduler* | The scheduler instance. |
| *the_thread* | The thread for the operation. |
| *node* | The thread's scheduler node. |

#### 8.156.3.11 _Scheduler_simple_Yield()

```
void _Scheduler_simple_Yield (
              const Scheduler_Control * scheduler,
              Thread_Control * the_thread,
              Scheduler_Node * node )
```

Performs the yield of a thread.

**Parameters**

| | | |
|---|---|---|
| | *scheduler* | The scheduler instance. |
| in,out | *the_thread* | The thread that performed the yield operation. |
| | *node* | The scheduler node of *the_thread*. |

## 8.157 Size Checks

Checks for sizes (size_t).

### Macros

- #define **T_eq_sz**(a, e) T_flags_eq_ulong(a, e, 0)
- #define **T_assert_eq_sz**(a, e) T_flags_eq_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_sz**(a, e) T_flags_eq_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_eq_sz**(s, a, e) T_flags_eq_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_sz**(s, a, e) T_flags_eq_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_sz**(a, e) T_flags_ne_ulong(a, e, 0)
- #define **T_assert_ne_sz**(a, e) T_flags_ne_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_sz**(a, e) T_flags_ne_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_ne_sz**(s, a, e) T_flags_ne_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_sz**(s, a, e) T_flags_ne_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_sz**(a, e) T_flags_ge_ulong(a, e, 0)
- #define **T_assert_ge_sz**(a, e) T_flags_ge_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_sz**(a, e) T_flags_ge_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_ge_sz**(s, a, e) T_flags_ge_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_sz**(s, a, e) T_flags_ge_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_sz**(a, e) T_flags_gt_ulong(a, e, 0)
- #define **T_assert_gt_sz**(a, e) T_flags_gt_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_sz**(a, e) T_flags_gt_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_gt_sz**(s, a, e) T_flags_gt_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_sz**(s, a, e) T_flags_gt_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_sz**(a, e) T_flags_le_ulong(a, e, 0)
- #define **T_assert_le_sz**(a, e) T_flags_le_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_le_sz**(a, e) T_flags_le_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_le_sz**(s, a, e) T_flags_le_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_sz**(s, a, e) T_flags_le_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_sz**(a, e) T_flags_lt_ulong(a, e, 0)
- #define **T_assert_lt_sz**(a, e) T_flags_lt_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_sz**(a, e) T_flags_lt_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_lt_sz**(s, a, e) T_flags_lt_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_sz**(s, a, e) T_flags_lt_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.157.1 Detailed Description

Checks for sizes (size_t).

## 8.158 Stack Handler

Stack Handler.

### Files

- file stack.h

  *Information About the Thread Stack Handler.*
- file stackimpl.h

  *Inlined Routines from the Stack Handler.*
- file stackallocatorfreenothing.c

  *_Stack_Free_nothing() Implementation*
- file threadstackallocate.c

  *Stack Allocate Helper.*

### Classes

- struct Stack_Control

### Macros

- #define STACK_MINIMUM_SIZE CPU_STACK_MINIMUM_SIZE

### Typedefs

- typedef void(∗ Stack_Allocator_initialize) (size_t stack_space_size)

  *The stack allocator initialization handler.*
- typedef void ∗(∗ Stack_Allocator_allocate) (size_t stack_size)

  *Stack allocator allocate handler.*
- typedef void(∗ Stack_Allocator_free) (void ∗addr)

  *Stack allocator free handler.*

### Functions

- void _Stack_Allocator_do_initialize (void)

  *Do the stack allocator initialization during system initialize.*
- static __inline__ void _Stack_Initialize (Stack_Control ∗the_stack, void ∗starting_address, size_t size)

  *Initializes stack with the given starting address and size.*
- static __inline__ uint32_t _Stack_Minimum (void)

  *Returns the minimum stack size.*
- static __inline__ bool _Stack_Is_enough (size_t size, bool is_fp)

  *Checks if the size is enough for a valid stack area on this processor.*
- static __inline__ size_t _Stack_Ensure_minimum (size_t size)

  *Returns the appropriate stack size for the requested size.*
- static __inline__ size_t _Stack_Extend_size (size_t stack_size, bool is_fp)

  *Extend the stack size to account for additional data structures allocated in the stack area of a thread.*
- void ∗ _Stack_Allocate (size_t stack_size)

  *Allocate the requested stack space.*
- void _Stack_Free (void ∗stack_area)

  *Free the stack area allocated by _Stack_Allocate().*
- void _Stack_Free_nothing (void ∗stack_area)

  *This function does nothing.*

## Variables

- uint32_t rtems_minimum_stack_size

  *The minimum stack size.*
- const uintptr_t _Stack_Space_size

  *The configured stack space size.*
- const bool _Stack_Allocator_avoids_workspace

  *Indicates if the stack allocator avoids the workspace.*
- const Stack_Allocator_initialize _Stack_Allocator_initialize

  *The stack allocator initialization handler.*
- const Stack_Allocator_allocate _Stack_Allocator_allocate

  *The stack allocator allocate handler.*
- const Stack_Allocator_free _Stack_Allocator_free

  *The stack allocator free handler.*

### 8.158.1 Detailed Description

Stack Handler.

This handler encapsulates functionality which is used in the management of thread stacks.

### 8.158.2 Macro Definition Documentation

#### 8.158.2.1 STACK_MINIMUM_SIZE

```
#define STACK_MINIMUM_SIZE CPU_STACK_MINIMUM_SIZE
```

The following constant defines the minimum stack size which every thread must exceed.

Definition at line 48 of file stack.h.

### 8.158.3 Typedef Documentation

#### 8.158.3.1 Stack_Allocator_allocate

```
typedef void*( * Stack_Allocator_allocate) (size_t stack_size)
```

Stack allocator allocate handler.

**Parameters**

| | |
|---|---|
| *stack_size* | The size of the stack area to allocate in bytes. |

**Return values**

| | |
|---|---|
| *NULL* | Not enough memory. |
| *other* | Pointer to begin of stack area. |

Definition at line 75 of file stack.h.

#### 8.158.3.2 Stack_Allocator_free

```
typedef void( * Stack_Allocator_free) (void *addr)
```

Stack allocator free handler.

**Parameters**

| | |
|---|---|
| *]* | addr A pointer to previously allocated stack area or NULL. |

Definition at line 82 of file stack.h.

#### 8.158.3.3 Stack_Allocator_initialize

```
typedef void( * Stack_Allocator_initialize) (size_t stack_space_size)
```

The stack allocator initialization handler.

**Parameters**

| | |
|---|---|
| *stack_space_size* | The size of the stack space in bytes. |

Definition at line 65 of file stack.h.

### 8.158.4 Function Documentation

#### 8.158.4.1 _Stack_Allocate()

```
void* _Stack_Allocate (
            size_t stack_size )
```

Allocate the requested stack space.

**Parameters**

| | |
|---|---|
| *stack_size* | The stack space that is requested. |

**Return values**

| | |
|---|---|
| *stack_area* | The allocated stack area. |
| *NULL* | The allocation failed. |

Definition at line 25 of file threadstackallocate.c.

#### 8.158.4.2 _Stack_Allocator_do_initialize()

```
void _Stack_Allocator_do_initialize (
            void  )
```

Do the stack allocator initialization during system initialize.

This function is used to initialize application provided stack allocators.

#### 8.158.4.3 _Stack_Ensure_minimum()

```
static __inline__ size_t _Stack_Ensure_minimum (
            size_t size )  [static]
```

Returns the appropriate stack size for the requested size.

This function returns the appropriate stack size given the requested size. If the requested size is below the minimum, then the minimum configured stack size is returned.

**Parameters**

| | |
|---|---|
| *size* | The stack size to check. |

**Returns**

The appropriate stack size.

Definition at line 114 of file stackimpl.h.

#### 8.158.4.4 _Stack_Extend_size()

```
static __inline__ size_t _Stack_Extend_size (
            size_t stack_size,
            bool is_fp )  [static]
```

Extend the stack size to account for additional data structures allocated in the stack area of a thread.

**Parameters**

| | |
|---|---|
| *stack_size* | The stack size. |
| *is_fp* | Indicates if the stack is for a floating-point thread. |

**Returns**

The extended stack size.

Definition at line 132 of file stackimpl.h.

#### 8.158.4.5 _Stack_Free()

```
void _Stack_Free (
             void * stack_area )
```

Free the stack area allocated by _Stack_Allocate().

Do nothing if the stack area is NULL.

**Parameters**

| | |
|---|---|
| *stack_area* | The stack area to free, or NULL. |

#### 8.158.4.6 _Stack_Free_nothing()

```
void _Stack_Free_nothing (
             void * stack_area )
```

This function does nothing.

**Parameters**

| | |
|---|---|
| *stack_area* | is not used. |

Definition at line 42 of file stackallocatorfreenothing.c.

#### 8.158.4.7 _Stack_Initialize()

```
static __inline__ void _Stack_Initialize (
             Stack_Control * the_stack,
```

```
            void * starting_address,
            size_t size )  [static]
```

Initializes stack with the given starting address and size.

This routine initializes the_stack record to indicate that size bytes of memory starting at starting_address have been reserved for a stack.

**Parameters**

| | | |
|---|---|---|
| out | *the_stack* | The stack to initialize. |
| | *starting_address* | The starting_address for the new stack. |
| | *size* | The size of the stack in bytes. |

Definition at line 49 of file stackimpl.h.

#### 8.158.4.8 _Stack_Is_enough()

```
static __inline__ bool _Stack_Is_enough (
            size_t size,
            bool is_fp )  [static]
```

Checks if the size is enough for a valid stack area on this processor.

This function returns true if size bytes is enough memory for a valid stack area on this processor, and false otherwise.

**Parameters**

| | |
|---|---|
| *size* | The stack size to check. |
| *is_fp* | Indicates if the stack is for a floating-point thread. |

**Return values**

| | |
|---|---|
| *true* | *size* is large enough. |
| *false* | *size* is not large enough. |

Definition at line 84 of file stackimpl.h.

#### 8.158.4.9 _Stack_Minimum()

```
static __inline__ uint32_t _Stack_Minimum (
            void  )  [static]
```

Returns the minimum stack size.

This function returns the minimum stack size configured for this application.

**Returns**

The minimum stack size.

Definition at line 67 of file stackimpl.h.

## 8.158.5 Variable Documentation

### 8.158.5.1 _Stack_Allocator_allocate

```
const Stack_Allocator_allocate _Stack_Allocator_allocate  [extern]
```

The stack allocator allocate handler.

Application provided via <rtems/confdefs.h>.

Definition at line 37 of file stackallocator.c.

### 8.158.5.2 _Stack_Allocator_avoids_workspace

```
const bool _Stack_Allocator_avoids_workspace  [extern]
```

Indicates if the stack allocator avoids the workspace.

Application provided via <rtems/confdefs.h>.

Definition at line 35 of file stackallocator.c.

### 8.158.5.3 _Stack_Allocator_free

```
const Stack_Allocator_free _Stack_Allocator_free  [extern]
```

The stack allocator free handler.

Application provided via <rtems/confdefs.h>.

### 8.158.5.4 _Stack_Allocator_initialize

```
const Stack_Allocator_initialize _Stack_Allocator_initialize  [extern]
```

The stack allocator initialization handler.

Application provided via <rtems/confdefs.h>.

### 8.158.5.5 _Stack_Space_size

```
const uintptr_t _Stack_Space_size  [extern]
```

The configured stack space size.

Application provided via <rtems/confdefs.h>.

### 8.158.5.6 rtems_minimum_stack_size

```
uint32_t rtems_minimum_stack_size  [extern]
```

The minimum stack size.

Application provided via <rtems/confdefs.h>.

# 8.159 Standard C Library Support

RTEMS Specific Support for the Standard C Library.

## Modules

- Malloc Support

  *RTEMS extensions to the Malloc Family.*
- RTEMS Termios Device Support

  *RTEMS Termios Device Support Internal Data Structures.*

## Files

- file malloc_deferred.c

  *Malloc Deferred Support.*
- file print_printf.c

  *RTEMS Print Support.*
- file printk.c

  *Kernel Printf Function.*
- file printk_plugin.c

  *Plugin Printk.*
- file rtems_putc.c

  *RTEMS Output a Character.*
- file vprintk.c

  *Print Formatted Output.*

## Classes

- struct rtems_resource_rtems_api
- struct rtems_resource_posix_api
- struct rtems_resource_snapshot

## Macros

- #define **RTEMS_NEWLIB_EXTENSION**

## Functions

- void **malloc_dump** (void)
- bool malloc_walk (int source, bool printf_enabled)

  *Malloc walk.*
- void malloc_set_heap_pointer (Heap_Control *new_heap)

  *Set malloc heap pointer.*
- Heap_Control * malloc_get_heap_pointer (void)

  *Get malloc heap pointer.*
- size_t malloc_free_space (void)

  *Get free malloc information.*
- int malloc_info (Heap_Information_block *the_info)

  *Get malloc status information.*
- bool **newlib_create_hook** (rtems_tcb *current_task, rtems_tcb *creating_task)
- void **newlib_terminate_hook** (rtems_tcb *current_task)
- void rtems_resource_snapshot_take (rtems_resource_snapshot *snapshot)

  *Tasks a snapshot of the resource usage of the system.*
- bool rtems_resource_snapshot_equal (const rtems_resource_snapshot *a, const rtems_resource_snapshot *b)

  *Compares two resource snapshots for equality.*
- bool rtems_resource_snapshot_check (const rtems_resource_snapshot *snapshot)

  *Takes a new resource snapshot and checks that it is equal to the given resource snapshot.*

### 8.159.1 Detailed Description

RTEMS Specific Support for the Standard C Library.

### 8.159.2 Macro Definition Documentation

#### 8.159.2.1 RTEMS_NEWLIB_EXTENSION

```
#define RTEMS_NEWLIB_EXTENSION
```

**Value:**

```
{ \
  newlib_create_hook,     /* rtems_task_create  */ \
  0,                      /* rtems_task_start   */ \
  0,                      /* rtems_task_restart */ \
  0,                      /* rtems_task_delete  */ \
  0,                      /* task_switch  */ \
  0,                      /* task_begin   */ \
  0,                      /* task_exitted */ \
  0,                      /* fatal        */ \
  newlib_terminate_hook   /* thread terminate */ \
}
```

Definition at line 89 of file libcsupport.h.

### 8.159.3 Function Documentation

#### 8.159.3.1 malloc_free_space()

```
size_t malloc_free_space (
            void  )
```

Get free malloc information.

Find amount of free heap remaining

#### 8.159.3.2 malloc_get_heap_pointer()

```
Heap_Control* malloc_get_heap_pointer (
            void  )
```

Get malloc heap pointer.

This routine is primarily used for debugging.

#### 8.159.3.3 malloc_info()

```
int malloc_info (
            Heap_Information_block * the_info )
```

Get malloc status information.

Find amount of free heap remaining.

#### 8.159.3.4 malloc_set_heap_pointer()

```
void malloc_set_heap_pointer (
            Heap_Control * new_heap )
```

Set malloc heap pointer.

This routine is primarily used for debugging.

#### 8.159.3.5 rtems_resource_snapshot_check()

```
bool rtems_resource_snapshot_check (
            const rtems_resource_snapshot * snapshot )
```

Takes a new resource snapshot and checks that it is equal to the given resource snapshot.

**Parameters**

| | | |
|---|---|---|
| in | *snapshot* | The resource snapshot used for comparison with the new resource snapshot. |

**Return values**

| | |
|---|---|
| *true* | The resource snapshots are equal. |
| *false* | Otherwise. |

**See also**

rtems_resource_snapshot_take().

### 8.159.3.6 rtems_resource_snapshot_equal()

```
bool rtems_resource_snapshot_equal (
            const rtems_resource_snapshot * a,
            const rtems_resource_snapshot * b )
```

Compares two resource snapshots for equality.

**Parameters**

| | | |
|---|---|---|
| in | *a* | One resource snapshot. |
| in | *b* | Another resource snapshot. |

**Return values**

| | |
|---|---|
| *true* | The resource snapshots are equal. |
| *false* | Otherwise. |

**See also**

rtems_resource_snapshot_take().

### 8.159.3.7 rtems_resource_snapshot_take()

```
void rtems_resource_snapshot_take (
            rtems_resource_snapshot * snapshot )
```

Tasks a snapshot of the resource usage of the system.

**Parameters**

| | | |
|---|---|---|
| out | *snapshot* | The snapshot of used resources. |

### See also

rtems_resource_snapshot_equal() and rtems_resource_snapshot_check().

```
#include <assert.h>
#include <rtems/libcsupport.h>
void example(void)
{
  rtems_resource_snapshot before;
  test_setup();
  rtems_resource_snapshot_take(&before);
  test();
  assert(rtems_resource_snapshot_check(&before));
  test_cleanup();
}
```

## 8.160 String Checks

Checks for strings.

### Macros

- #define **T_eq_str**(a, e) T_flags_eq_str(a, e, 0)
- #define **T_assert_eq_str**(a, e) T_flags_eq_str(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_str**(a, e) T_flags_eq_str(a, e, T_CHECK_QUIET)
- #define **T_step_eq_str**(s, a, e) T_flags_eq_str(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_str**(s, a, e) T_flags_eq_str(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_str**(a, e) T_flags_ne_str(a, e, 0)
- #define **T_assert_ne_str**(a, e) T_flags_ne_str(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_str**(a, e) T_flags_ne_str(a, e, T_CHECK_QUIET)
- #define **T_step_ne_str**(s, a, e) T_flags_ne_str(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_str**(s, a, e) T_flags_ne_str(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_eq_nstr**(a, e, n) T_flags_eq_nstr(a, e, n, 0)
- #define **T_assert_eq_nstr**(a, e, n) T_flags_eq_nstr(a, e, n, T_CHECK_STOP)
- #define **T_quiet_eq_nstr**(a, e, n) T_flags_eq_nstr(a, e, n, T_CHECK_QUIET)
- #define **T_step_eq_nstr**(s, a, e, n) T_flags_eq_nstr(a, e, n, T_CHECK_STEP(s))
- #define **T_step_assert_eq_nstr**(s, a, e, n) T_flags_eq_nstr(a, e, n, T_CHECK_STEP(s) | T_CHECK_ST↩ OP)
- #define **T_ne_nstr**(a, e, n) T_flags_ne_nstr(a, e, n, 0)
- #define **T_assert_ne_nstr**(a, e, n) T_flags_ne_nstr(a, e, n, T_CHECK_STOP)
- #define **T_quiet_ne_nstr**(a, e, n) T_flags_ne_nstr(a, e, n, T_CHECK_QUIET)
- #define **T_step_ne_nstr**(s, a, e, n) T_flags_ne_nstr(a, e, n, T_CHECK_STEP(s))
- #define **T_step_assert_ne_nstr**(s, a, e, n) T_flags_ne_nstr(a, e, n, T_CHECK_STEP(s) | T_CHECK_ST↩ OP)

### 8.160.1 Detailed Description

Checks for strings.

# 8.161 SuperCore

Provides services for all APIs.

## Modules

- API Mutex Handler

  *Provides routines to ensure mutual exclusion on API level.*
- Address Handler

  *Support for Address Manipulation.*
- Assert Handler

  *Support for Assert Statements.*
- Atomic Operations

  *Support for atomic operations.*
- Atomic Operations CPU

  *Atomic Operations CPU API.*
- Barrier Handler

  *Functionality for Foundation Barrier Services.*
- Bitmap Priority Thread Routines

  *Bitmap Priority Thread Routines.*
- CPU Architecture Support

  *Provides CPU architecture dependent services.*
- Chain Handler

  *Provides Data Structures Chain Node and Chain Control.*
- Context Handler

  *Functionality for Abstraction of Thread Context Management.*
- Freechain Handler

  *The Freechain Handler.*
- Heap Handler

  *The Heap Handler provides a heap.*
- Helpers

  *Timespec Helpers.*
- ISR Handler

  *ISR Handler.*
- Internal Error Handler

  *Internal Error Handler.*
- MP Packet Handler

  *MP Packte Handler.*
- Memory Handler

  *Low level handler to provide memory areas for higher level memory handlers such as the Workspace Handler.*
- Message Queue Handler

  *Message Queue Handler.*
- Multiprocessor Resource Sharing Protocol Handler

  *Multiprocessor Resource Sharing Protocol (MrsP).*
- Mutex Handler

  *Mutex Handler.*
- Priority Handler

  *Priority Handler.*
- Processor Mask

*Processor Mask.*

- Profiling Support

  *Profiling support.*
- RTEMS Copyright Notice

  *Copyright Notice for RTEMS.*
- RTEMS Per CPU Information
- Red-Black Tree Handler

  *Red-Black Tree Handler.*
- SMP Barriers

  *The SMP barrier provides barrier synchronization for SMP systems at the lowest level.*
- SMP Locks

  *The SMP lock provides mutual exclusion for SMP systems at the lowest level.*
- SMP Support

  *SMP Support.*
- Scheduler Handler

  *This handler encapsulates functionality related to managing sets of threads that are ready for execution.*
- Score Timestamp

  *Score Timestamp.*
- Semaphore Handler

  *Semaphore Handler.*
- Stack Handler

  *Stack Handler.*
- System State Handler

  *Management of the internal system state of RTEMS.*
- Thread Handler

  *Thread Handler.*
- Thread Queue Handler

  *Thread Queue Handler.*
- Thread States

  *SuperCore Thread States.*
- Thread-Local Storage (TLS)

  *Thread-local storage (TLS) support.*
- Time of Day Handler

  *Time of Day Handler.*
- Timecounter Handler

  *Timecounter Handler.*
- User Extension Handler

  *The User Extension Handler provides invocation of application dependent routines at critical points in the life of each thread and the system as a whole.*
- Watchdog Handler

  *Watchdog Handler.*
- Workspace Handler

## Files

- file objectgetinfoid.c

  *Get Information of an Object from an ID.*
- file objectgetnameasstring.c

  *Extracts a Node from a Chain.*
- file objectnamespaceremove.c

*Removes Object from Namespace.*

- file objectnametoid.c

  *Object Name To Id.*

- file smp.c

  *SMP Support.*

### 8.161.1 Detailed Description

Provides services for all APIs.

## 8.162 Support Services

Items of this group should move to other groups.

### Macros

- #define RTEMS_MICROSECONDS_TO_TICKS(_us) ( ( _us ) / rtems_configuration_get_microseconds_per_tick() )

  *Returns the number of clock ticks for the specified microseconds value.*
- #define RTEMS_MILLISECONDS_TO_MICROSECONDS(_ms) ( ( _ms ) * 1000UL )

  *Returns the number of microseconds for the specified milliseconds value.*
- #define RTEMS_MILLISECONDS_TO_TICKS(_ms) RTEMS_MICROSECONDS_TO_TICKS( RTEMS_MILLISECONDS_TO_MICROSECONDS( _ms ) )

  *Returns the number of clock ticks for the specified milliseconds value.*

### Functions

- static bool rtems_is_name_valid (rtems_status_code name)

  *Returns true, if the specified object name is valid, otherwise returns false.*
- static void rtems_name_to_characters (rtems_name name, char *c1, char *c2, char *c3, char *c4)

  *Breaks the object name into the four component characters.*
- bool rtems_workspace_allocate (size_t bytes, void **pointer)

  %
- bool rtems_workspace_free (void *pointer)

  %
- bool rtems_workspace_get_information (Heap_Information_block *the_info)

  %
- void * rtems_workspace_greedy_allocate (const uintptr_t *block_sizes, size_t block_count)

  %
- void * rtems_workspace_greedy_allocate_all_except_largest (uintptr_t *allocatable_size)

  %
- void rtems_workspace_greedy_free (void *opaque)

  %

### 8.162.1 Detailed Description

Items of this group should move to other groups.

### 8.162.2 Macro Definition Documentation

#### 8.162.2.1 RTEMS_MICROSECONDS_TO_TICKS

```
#define RTEMS_MICROSECONDS_TO_TICKS(
              _us )   ( ( _us ) / rtems_configuration_get_microseconds_per_tick() )
```

Returns the number of clock ticks for the specified microseconds value.

The number of clock ticks per second is defined by the CONFIGURE_MICROSECONDS_PER_TICK application configuration option.

**Parameters**

| | |
|---|---|
| *_us* | is the microseconds value to convert to clock ticks. |

**Returns**

The number of clock ticks for the specified microseconds value is returned.

Definition at line 111 of file support.h.

### 8.162.2.2 RTEMS_MILLISECONDS_TO_MICROSECONDS

```
#define RTEMS_MILLISECONDS_TO_MICROSECONDS(
              _ms ) ( ( _ms ) * 1000UL )
```

Returns the number of microseconds for the specified milliseconds value.

**Parameters**

| | |
|---|---|
| *_ms* | is the milliseconds value to convert to microseconds. |

**Returns**

The number of microseconds for the specified milliseconds value is returned.

Definition at line 127 of file support.h.

### 8.162.2.3 RTEMS_MILLISECONDS_TO_TICKS

```
#define RTEMS_MILLISECONDS_TO_TICKS(
              _ms )  RTEMS_MICROSECONDS_TO_TICKS( RTEMS_MILLISECONDS_TO_MICROSECONDS( _ms ) )
```

Returns the number of clock ticks for the specified milliseconds value.

The number of clock ticks per second is defined by the CONFIGURE_MICROSECONDS_PER_TICK application configuration option.

**Parameters**

| | |
|---|---|
| *_ms* | is the milliseconds value to convert to clock ticks. |

**Returns**

The number of clock ticks for the specified milliseconds value is returned.

Definition at line 145 of file support.h.

## 8.162.3 Function Documentation

### 8.162.3.1 rtems_is_name_valid()

```
static bool rtems_is_name_valid (
            rtems_status_code name )  [inline], [static]
```

Returns true, if the specified object name is valid, otherwise returns false.

**Parameters**

| | |
|---|---|
| *name* | is the object name to check. |

**Return values**

| | |
|---|---|
| *true* | The specified object name is valid. |
| *false* | Otherwise. |

Definition at line 90 of file support.h.

### 8.162.3.2 rtems_name_to_characters()

```
static void rtems_name_to_characters (
            rtems_name name,
            char * c1,
            char * c2,
            char * c3,
            char * c4 )  [inline], [static]
```

Breaks the object name into the four component characters.

**Parameters**

| | | |
|---|---|---|
| | *name* | is the object name to break into four component characters. |
| out | *c1* | is the first character of the object name. |
| out | *c2* | is the second character of the object name. |
| out | *c3* | is the third character of the object name. |
| out | *c4* | is the fourth character of the object name. |

Definition at line 165 of file support.h.

### 8.162.3.3 rtems_workspace_allocate()

```
bool rtems_workspace_allocate (
            size_t bytes,
            void ** pointer )
```

%

**Parameters**

| | |
|---|---|
| *bytes* | % |
| *pointer* | % |

### 8.162.3.4 rtems_workspace_free()

```
bool rtems_workspace_free (
            void * pointer )
```

%

**Parameters**

| | |
|---|---|
| *pointer* | % |

### 8.162.3.5 rtems_workspace_get_information()

```
bool rtems_workspace_get_information (
            Heap_Information_block * the_info )
```

%

**Parameters**

| | |
|---|---|
| *the_info* | % |

### 8.162.3.6 rtems_workspace_greedy_allocate()

```
void* rtems_workspace_greedy_allocate (
            const uintptr_t * block_sizes,
            size_t block_count )
```

%

**Parameters**

| | |
|---|---|
| *block_sizes* | % |
| *block_count* | % |

#### 8.162.3.7 rtems_workspace_greedy_allocate_all_except_largest()

```
void* rtems_workspace_greedy_allocate_all_except_largest (
            uintptr_t * allocatable_size )
```

%

**Parameters**

| | |
|---|---|
| *allocatable_size* | % |

#### 8.162.3.8 rtems_workspace_greedy_free()

```
void rtems_workspace_greedy_free (
            void * opaque )
```

%

**Parameters**

| | |
|---|---|
| *opaque* | % |

# 8.163 System State Handler

Management of the internal system state of RTEMS.

## Files

- file sysstate.h

  *System State Handler API.*

## Macros

- #define **SYSTEM_STATE_CODES_FIRST** SYSTEM_STATE_BEFORE_INITIALIZATION
- #define **SYSTEM_STATE_CODES_LAST** SYSTEM_STATE_TERMINATED

## Enumerations

- enum System_state_Codes { SYSTEM_STATE_BEFORE_INITIALIZATION, SYSTEM_STATE_BEFORE_MULTITASKING, SYSTEM_STATE_UP, SYSTEM_STATE_TERMINATED }

  *System states.*

## Functions

- static __inline__ void _System_state_Set (System_state_Codes state)

  *Sets the current system state to the given state.*
- static __inline__ System_state_Codes _System_state_Get (void)

  *Gets the current system state.*
- static __inline__ bool _System_state_Is_before_initialization (System_state_Codes state)

  *Checks if the state is before initialization.*
- static __inline__ bool _System_state_Is_before_multitasking (System_state_Codes state)

  *Checks if the state is before multitasking.*
- static __inline__ bool _System_state_Is_up (System_state_Codes state)

  *Checks if the state is up.*
- static __inline__ bool _System_state_Is_terminated (System_state_Codes state)

  *Checks if the state is terminated.*

## Variables

- System_state_Codes **_System_state_Current**

## 8.163.1 Detailed Description

Management of the internal system state of RTEMS.

## 8.163.2 Enumeration Type Documentation

### 8.163.2.1 System_state_Codes

```
enum System_state_Codes
```

System states.

**Enumerator**

| | |
|---|---|
| SYSTEM_STATE_BEFORE_INITIALIZATION | The system is before the end of the first phase of initialization. |
| SYSTEM_STATE_BEFORE_MULTITASKING | The system is between end of the first phase of initialization but before multitasking is started. |
| SYSTEM_STATE_UP | The system is up and operating normally. |
| SYSTEM_STATE_TERMINATED | The system reached its terminal state. |

Definition at line 40 of file sysstate.h.

### 8.163.3 Function Documentation

#### 8.163.3.1 _System_state_Get()

```
static __inline__ System_state_Codes _System_state_Get (
            void  )  [static]
```

Gets the current system state.

**Returns**

The current system state.

Definition at line 90 of file sysstate.h.

#### 8.163.3.2 _System_state_Is_before_initialization()

```
static __inline__ bool _System_state_Is_before_initialization (
            System_state_Codes state )  [static]
```

Checks if the state is before initialization.

**Parameters**

| | |
|---|---|
| *state* | The state to check. |

**Return values**

| | |
|---|---|
| *true* | *state* is before initialization. |
| *false* | *state* is not before initialization. |

Definition at line 103 of file sysstate.h.

#### 8.163.3.3 _System_state_Is_before_multitasking()

```
static __inline__ bool _System_state_Is_before_multitasking (
            System_state_Codes state )  [static]
```

Checks if the state is before multitasking.

**Parameters**

| | |
|---|---|
| *state* | The state to check. |

**Return values**

| | |
|---|---|
| *true* | *state* is before multitasking. |
| *false* | *state* is not before multitasking. |

Definition at line 118 of file sysstate.h.

#### 8.163.3.4 _System_state_Is_terminated()

```
static __inline__ bool _System_state_Is_terminated (
            System_state_Codes state )  [static]
```

Checks if the state is terminated.

**Parameters**

| | |
|---|---|
| *state* | The state to check. |

**Return values**

| | |
|---|---|
| *true* | *state* is terminated. |
| *false* | *state* is not terminated. |

Definition at line 148 of file sysstate.h.

#### 8.163.3.5 _System_state_Is_up()

```
static __inline__ bool _System_state_Is_up (
            System_state_Codes state )  [static]
```

Checks if the state is up.

**Parameters**

| | |
|---|---|
| *state* | The state to check. |

**Return values**

| | |
|---|---|
| *true* | *state* is up. |
| *false* | *state* is not up. |

Definition at line 133 of file sysstate.h.

#### 8.163.3.6 _System_state_Set()

```
static __inline__ void _System_state_Set (
             System_state_Codes state )  [static]
```

Sets the current system state to the given state.

**Parameters**

| | |
|---|---|
| *state* | The state to set. |

Definition at line 78 of file sysstate.h.

# 8.164 Task Manager

The Task Manager provides a comprehensive set of directives to create, delete, and administer tasks.

## Classes

- struct rtems_task_config

  *This structure defines the configuration of a task constructed by rtems_task_construct().*
- struct rtems_initialization_tasks_table

  %

## Macros

- #define rtems_scheduler_get_processor() _SMP_Get_current_processor()

  *Returns the index of the current processor.*
- #define rtems_scheduler_get_processor_maximum() _SMP_Get_processor_maximum()

  *Returns the processor maximum supported by the system.*
- #define RTEMS_CURRENT_PRIORITY 0

  *This constant is passed to {set-priority:/name}() when the caller wants to obtain the current priority.*
- #define RTEMS_CONFIGURED_MINIMUM_STACK_SIZE 0

  %
- #define RTEMS_MINIMUM_PRIORITY 1

  %
- #define RTEMS_MINIMUM_STACK_SIZE STACK_MINIMUM_SIZE

  %
- #define RTEMS_NO_PRIORITY RTEMS_CURRENT_PRIORITY

  %
- #define RTEMS_SELF OBJECTS_ID_OF_SELF

  %
- #define RTEMS_TASK_STORAGE_ALIGNMENT CPU_HEAP_ALIGNMENT

  *This constant defines the recommended alignment of a task storage area in bytes.*
- #define RTEMS_TASK_STORAGE_SIZE(_size, _attributes)

  *Returns the recommended task storage area size for the specified size and task attributes.*
- #define RTEMS_YIELD_PROCESSOR WATCHDOG_NO_TIMEOUT

  %
- #define RTEMS_MAXIMUM_PRIORITY _RTEMS_Maximum_priority()

  %

## Typedefs

- typedef CPU_Uint32ptr rtems_task_argument

  *This type is used to represent task argument values.*
- typedef uint32_t rtems_task_priority

  %
- typedef void rtems_task

  %
- typedef rtems_task(∗ rtems_task_entry) (rtems_task_argument)

  *This type defines the entry point of an RTEMS task.*
- typedef struct _Thread_Control rtems_tcb

  %
- typedef bool(∗ rtems_task_visitor) (rtems_tcb ∗, void ∗)

  %

## Functions

- rtems_status_code rtems_scheduler_add_processor (rtems_id scheduler_id, uint32_t cpu_index)

  *Adds the processor to the set of processors owned by the scheduler instance.*
- rtems_status_code rtems_scheduler_get_processor_set (rtems_id scheduler_id, size_t cpusetsize, cpu_↩set_t *cpuset)

  *Gets the set of processors owned by the scheduler instance.*
- rtems_status_code rtems_scheduler_ident (rtems_name name, rtems_id *id)

  *Identifies a scheduler instance by its name.*
- rtems_status_code rtems_scheduler_ident_by_processor (uint32_t cpu_index, rtems_id *id)

  *Identifies a scheduler instance by a processor index.*
- rtems_status_code rtems_scheduler_ident_by_processor_set (size_t cpusetsize, const cpu_set_t *cpuset, rtems_id *id)

  *Identifies a scheduler instance by a processor set.*
- rtems_status_code rtems_scheduler_remove_processor (rtems_id scheduler_id, uint32_t cpu_index)

  *Removes a processor from set of processors owned by the scheduler instance.*
- rtems_status_code rtems_scheduler_get_maximum_priority (rtems_id scheduler_id, rtems_task_priority *priority)

  *Gets the maximum task priority of the scheduler instance.*
- rtems_status_code rtems_scheduler_map_priority_from_posix (rtems_id scheduler_id, int posix_priority, rtems_task_priority *priority)

  *Maps a POSIX thread priority to the corresponding Classic API task priority.*
- rtems_status_code rtems_scheduler_map_priority_to_posix (rtems_id scheduler_id, rtems_task_priority priority, int *posix_priority)

  *Maps a Classic API task priority to the corresponding POSIX thread priority.*
- rtems_status_code rtems_task_construct (const rtems_task_config *config, rtems_id *id)

  *Constructs a task from the specified the task configuration.*
- rtems_status_code rtems_task_create (rtems_name name, rtems_task_priority initial_priority, size_t stack↩_size, rtems_mode initial_modes, rtems_attribute attribute_set, rtems_id *id)

  *Creates a task object.*
- rtems_status_code rtems_task_delete (rtems_id id)

  %
- RTEMS_NO_RETURN void rtems_task_exit (void)

  %
- rtems_status_code rtems_task_get_affinity (rtems_id id, size_t cpusetsize, cpu_set_t *cpuset)

  %
- rtems_status_code rtems_task_get_priority (rtems_id task_id, rtems_id scheduler_id, rtems_task_priority *priority)

  %
- rtems_status_code rtems_task_get_scheduler (rtems_id task_id, rtems_id *scheduler_id)

  %
- rtems_status_code rtems_task_ident (rtems_name name, uint32_t node, rtems_id *id)

  *Identifies a task object by the specified object name.*
- rtems_status_code rtems_task_is_suspended (rtems_id id)

  %
- rtems_status_code rtems_task_mode (rtems_mode mode_set, rtems_mode mask, rtems_mode *previous↩_mode_set)

  %
- rtems_status_code rtems_task_restart (rtems_id id, rtems_task_argument argument)

  %
- rtems_status_code rtems_task_resume (rtems_id id)

  %

- rtems_id rtems_task_self (void)

  %
- rtems_status_code rtems_task_set_affinity (rtems_id id, size_t cpusetsize, const cpu_set_t *cpuset)

  %
- rtems_status_code rtems_task_set_priority (rtems_id id, rtems_task_priority new_priority, rtems_task_priority *old_priority)

  %
- rtems_status_code rtems_task_set_scheduler (rtems_id task_id, rtems_id scheduler_id, rtems_task_priority priority)

  %
- rtems_status_code rtems_task_start (rtems_id id, rtems_task_entry entry_point, rtems_task_argument argument)

  %
- rtems_status_code rtems_task_suspend (rtems_id id)

  %
- void rtems_task_iterate (rtems_task_visitor visitor, void *arg)

  %
- rtems_status_code rtems_task_wake_after (rtems_interval ticks)

  %
- rtems_status_code rtems_task_wake_when (rtems_time_of_day *time_buffer)

  %

### 8.164.1 Detailed Description

The Task Manager provides a comprehensive set of directives to create, delete, and administer tasks.

### 8.164.2 Task Definition

Many definitions of a task have been proposed in computer literature. Unfortunately, none of these definitions encompasses all facets of the concept in a manner which is operating system independent. Several of the more common definitions are provided to enable each user to select a definition which best matches their own experience and understanding of the task concept:

- a "dispatchable" unit.
- an entity to which the processor is allocated.
- an atomic unit of a real-time, multiprocessor system.
- single threads of execution which concurrently compete for resources.
- a sequence of closely related computations which can execute concurrently with other computational sequences.

From RTEMS' perspective, a task is the smallest thread of execution which can compete on its own for system resources. A task is manifested by the existence of a task control block (TCB).

### 8.164.3 Task Control Block

The Task Control Block (TCB) is an RTEMS defined data structure which contains all the information that is pertinent to the execution of a task. During system initialization, RTEMS reserves a TCB for each task configured. A TCB is allocated upon creation of the task and is returned to the TCB free list upon deletion of the task.

The TCB's elements are modified as a result of system calls made by the application in response to external and internal stimuli. TCBs are the only RTEMS internal data structure that can be accessed by an application via user extension routines. The TCB contains a task's name, ID, current priority, current and starting states, execution mode, TCB user extension pointer, scheduling control structures, as well as data required by a blocked task.

A task's context is stored in the TCB when a task switch occurs. When the task regains control of the processor, its context is restored from the TCB. When a task is restarted, the initial state of the task is restored from the starting context area in the task's TCB.

### 8.164.4 Task States

A task may exist in one of the following five states:

- executing - Currently scheduled to the CPU
- ready - May be scheduled to the CPU
- blocked - Unable to be scheduled to the CPU
- dormant - Created task that is not started
- non-existent - Uncreated or deleted task

An active task may occupy the executing, ready, blocked or dormant state, otherwise the task is considered non-existent. One or more tasks may be active in the system simultaneously. Multiple tasks communicate, synchronize, and compete for system resources with each other via system calls. The multiple tasks appear to execute in parallel, but actually each is dispatched to the CPU for periods of time determined by the RTEMS scheduling algorithm. The scheduling of a task is based on its current state and priority.

### 8.164.5 Task Priority

A task's priority determines its importance in relation to the other tasks executing on the same processor. RTEMS supports 255 levels of priority ranging from 1 to 255. The data type rtems_task_priority() is used to store task priorities.

Tasks of numerically smaller priority values are more important tasks than tasks of numerically larger priority values. For example, a task at priority level 5 is of higher privilege than a task at priority level 10. There is no limit to the number of tasks assigned to the same priority.

Each task has a priority associated with it at all times. The initial value of this priority is assigned at task creation time. The priority of a task may be changed at any subsequent time.

Priorities are used by the scheduler to determine which ready task will be allowed to execute. In general, the higher the logical priority of a task, the more likely it is to receive processor execution time.

### 8.164.6 Task Mode

A task's execution mode is a combination of the following four components:

- preemption
- ASR processing
- timeslicing
- interrupt level

It is used to modify RTEMS' scheduling process and to alter the execution environment of the task. The data type rtems_task_mode() is used to manage the task execution mode.

The preemption component allows a task to determine when control of the processor is relinquished. If preemption is disabled (RTEMS_NO_PREEMPT), the task will retain control of the processor as long as it is in the executing state – even if a higher priority task is made ready. If preemption is enabled (RTEMS_PREEMPT) and a higher priority task is made ready, then the processor will be taken away from the current task immediately and given to the higher priority task.

The timeslicing component is used by the RTEMS scheduler to determine how the processor is allocated to tasks of equal priority. If timeslicing is enabled (RTEMS_TIMESLICE), then RTEMS will limit the amount of time the task can execute before the processor is allocated to another ready task of equal priority. The length of the timeslice is application dependent and specified in the Configuration Table. If timeslicing is disabled (RTEMS_NO_TIMESLI↩CE), then the task will be allowed to execute until a task of higher priority is made ready. If RTEMS_NO_PREEMPT is selected, then the timeslicing component is ignored by the scheduler.

The asynchronous signal processing component is used to determine when received signals are to be processed by the task. If signal processing is enabled (RTEMS_ASR), then signals sent to the task will be processed the next time the task executes. If signal processing is disabled (RTEMS_NO_ASR), then all signals received by the task will remain posted until signal processing is enabled. This component affects only tasks which have established a routine to process asynchronous signals.

The interrupt level component is used to determine which interrupts will be enabled when the task is executing. RTEMS_INTERRUPT_LEVEL(n) specifies that the task will execute at interrupt level n.

- RTEMS_PREEMPT - enable preemption (default)
- RTEMS_NO_PREEMPT - disable preemption
- RTEMS_NO_TIMESLICE - disable timeslicing (default)
- RTEMS_TIMESLICE - enable timeslicing
- RTEMS_ASR - enable ASR processing (default)
- RTEMS_NO_ASR - disable ASR processing
- RTEMS_INTERRUPT_LEVEL(0) - enable all interrupts (default)
- RTEMS_INTERRUPT_LEVEL(n) - execute at interrupt level n

The set of default modes may be selected by specifying the RTEMS_DEFAULT_MODES constant.

### 8.164.7 Accessing Task Arguments

All RTEMS tasks are invoked with a single argument which is specified when they are started or restarted. The argument is commonly used to communicate startup information to the task. The simplest manner in which to define a task which accesses it argument is:

```
rtems_task user_task(
   rtems_task_argument argument
);
```

Application tasks requiring more information may view this single argument as an index into an array of parameter blocks.

### 8.164.8 Floating Point Considerations

Creating a task with the RTEMS_FLOATING_POINT attribute flag results in additional memory being allocated for the TCB to store the state of the numeric coprocessor during task switches. This additional memory is NOT allocated for RTEMS_NO_FLOATING_POINT tasks. Saving and restoring the context of a RTEMS_FLOATIN↩G_POINT task takes longer than that of a RTEMS_NO_FLOATING_POINT task because of the relatively large amount of time required for the numeric coprocessor to save or restore its computational state.

Since RTEMS was designed specifically for embedded military applications which are floating point intensive, the executive is optimized to avoid unnecessarily saving and restoring the state of the numeric coprocessor. The state of the numeric coprocessor is only saved when a RTEMS_FLOATING_POINT task is dispatched and that task was not the last task to utilize the coprocessor. In a system with only one RTEMS_FLOATING_POINT task, the state of the numeric coprocessor will never be saved or restored.

Although the overhead imposed by RTEMS_FLOATING_POINT tasks is minimal, some applications may wish to completely avoid the overhead associated with RTEMS_FLOATING_POINT tasks and still utilize a numeric coprocessor. By preventing a task from being preempted while performing a sequence of floating point operations, a RTEMS_NO_FLOATING_POINT task can utilize the numeric coprocessor without incurring the overhead of a RTEMS_FLOATING_POINT context switch. This approach also avoids the allocation of a floating point context area. However, if this approach is taken by the application designer, NO tasks should be created as RTEMS_F↩LOATING_POINT tasks. Otherwise, the floating point context will not be correctly maintained because RTEMS assumes that the state of the numeric coprocessor will not be altered by RTEMS_NO_FLOATING_POINT tasks.

If the supported processor type does not have hardware floating capabilities or a standard numeric coprocessor, RTEMS will not provide built-in support for hardware floating point on that processor. In this case, all tasks are considered RTEMS_NO_FLOATING_POINT whether created as RTEMS_FLOATING_POINT or RTEMS_NO↩_FLOATING_POINT tasks. A floating point emulation software library must be utilized for floating point operations.

On some processors, it is possible to disable the floating point unit dynamically. If this capability is supported by the target processor, then RTEMS will utilize this capability to enable the floating point unit only for tasks which are created with the RTEMS_FLOATING_POINT attribute. The consequence of a RTEMS_NO_FLOATING_POINT task attempting to access the floating point unit is CPU dependent but will generally result in an exception condition.

### 8.164.9 Per Task Variables

Per task variables are no longer available. In particular the rtems_task_variable_add(), rtems_task_variable_get() and rtems_task_variable_delete() functions are neither declared nor defined anymore. Use thread local storage or POSIX Keys instead.

### 8.164.10 Building a Task Attribute Set

In general, an attribute set is built by a bitwise OR of the desired components. The set of valid task attribute components is listed below:

- RTEMS_NO_FLOATING_POINT - does not use coprocessor (default)
- RTEMS_FLOATING_POINT - uses numeric coprocessor
- RTEMS_LOCAL - local task (default)
- RTEMS_GLOBAL - global task

Attribute values are specifically designed to be mutually exclusive, therefore bitwise OR and addition operations are equivalent as long as each attribute appears exactly once in the component list. A component listed as a default is not required to appear in the component list, although it is a good programming practice to specify default components. If all defaults are desired, then RTEMS_DEFAULT_ATTRIBUTES should be used. This example demonstrates the attribute_set parameter needed to create a local task which utilizes the numeric coprocessor. The attribute_set parameter could be `RTEMS_FLOATING_POINT` or `RTEMS_LOCAL | RTEMS_FLOATING_POINT`. The attribute_set parameter can be set to `RTEMS_FLOATING_POINT` because `RTEMS_LOCAL` is the default for all created tasks. If the task were global and used the numeric coprocessor, then the attribute_set parameter would be `RTEMS_GLOBAL | RTEMS_FLOATING_POINT`.

### 8.164.11 Building a Mode and Mask

In general, a mode and its corresponding mask is built by a bitwise OR of the desired components. The set of valid mode constants and each mode's corresponding mask constant is listed below:

| Mode Constant | Mask Constant | Description |
|---|---|---|
| RTEMS_PREEMPT | RTEMS_PREEMPT_MASK | enables preemption |
| RTEMS_NO_PREEMPT | RTEMS_PREEMPT_MASK | disables preemption |
| RTEMS_NO_TIMESLICE | RTEMS_TIMESLICE_MASK | disables timeslicing |
| RTEMS_TIMESLICE | RTEMS_TIMESLICE_MASK | enables timeslicing |
| RTEMS_ASR | RTEMS_ASR_MASK | enables ASR processing |
| RTEMS_NO_ASR | RTEMS_ASR_MASK | disables ASR processing |
| RTEMS_INTERRUPT_LEVEL(0) | RTEMS_INTERRUPT_MASK | enables all interrupts |
| RTEMS_INTERRUPT_LEVEL(n) | RTEMS_INTERRUPT_MASK | sets interrupts level n |

Mode values are specifically designed to be mutually exclusive, therefore bitwise OR and addition operations are equivalent as long as each mode appears exactly once in the component list. A mode component listed as a default is not required to appear in the mode component list, although it is a good programming practice to specify default components. If all defaults are desired, the mode RTEMS_DEFAULT_MODES and the mask RTEMS_ALL_MODE_MASKS should be used.

The following example demonstrates the mode and mask parameters used with the rtems_task_mode() directive to place a task at interrupt level 3 and make it non-preemptible. The mode should be set to `RTEMS_INTERRUPT_LEVEL(3) | RTEMS_NO_PREEMPT` to indicate the desired preemption mode and interrupt level, while the mask parameter should be set to `RTEMS_INTERRUPT_MASK | RTEMS_PREEMPT_MASK` to indicate that the calling task's interrupt level and preemption mode are being altered.

### 8.164.12 Macro Definition Documentation

#### 8.164.12.1 rtems_scheduler_get_processor

```
#define rtems_scheduler_get_processor( ) _SMP_Get_current_processor()
```

Returns the index of the current processor.

In uniprocessor configurations, this macro evaluates to a compile time constant of zero.

In SMP configurations, an architecture-specific method is used to obtain the index of the current processor in the system. The set of processor indices is the range of integers starting with zero up to rtems_scheduler_get_processor_maximum() minus one.

Outside of sections with disabled thread dispatching the current processor index may change after every instruction since the thread may migrate from one processor to another. Sections with disabled interrupts are sections with thread dispatching disabled.

**Returns**

The index of the current processor is returned.

Definition at line 140 of file tasks.h.

#### 8.164.12.2 rtems_scheduler_get_processor_maximum

```
#define rtems_scheduler_get_processor_maximum( ) _SMP_Get_processor_maximum()
```

Returns the processor maximum supported by the system.

In uniprocessor configurations, this macro evaluates to a compile time constant of one.

In SMP configurations, this macro returns the minimum of the processors (physically or virtually) available by the platform and the configured processor maximum. Not all processors in the range from processor index zero to the last processor index (which is the processor maximum minus one) may be configured to be used by a scheduler or may be online (online processors have a scheduler assigned).

**Returns**

The processor maximum supported by the system is returned.

Definition at line 161 of file tasks.h.

#### 8.164.12.3 RTEMS_TASK_STORAGE_ALIGNMENT

```
#define RTEMS_TASK_STORAGE_ALIGNMENT CPU_HEAP_ALIGNMENT
```

This constant defines the recommended alignment of a task storage area in bytes.

Use it with RTEMS_ALIGNED() to define the alignment of a statically allocated task storage area.

Definition at line 1086 of file tasks.h.

#### 8.164.12.4 RTEMS_TASK_STORAGE_SIZE

```
#define RTEMS_TASK_STORAGE_SIZE(
              _size,
              _attributes )
```

**Value:**

```
    ( ( _size ) + \
      ( ( ( _attributes ) & RTEMS_FLOATING_POINT ) != 0 ? \
        CONTEXT_FP_SIZE : 0 ) )
```

Returns the recommended task storage area size for the specified size and task attributes.

**Parameters**

| | |
|---|---|
| *_size* | is the size dedicated to the task stack and thread-local storage in bytes. |
| *_attributes* | is the attribute set of the task using the storage area. |

**Returns**

The recommended task storage area size calculated from the input parameters is returned.

Definition at line 1108 of file tasks.h.

## 8.164.13 Typedef Documentation

### 8.164.13.1 rtems_task_argument

```
typedef CPU_Uint32ptr rtems_task_argument
```

This type is used to represent task argument values.

The type is an architecture-specific unsigned integer type which is large enough to represent pointer values and 32-bit unsigned integers.

Definition at line 328 of file tasks.h.

## 8.164.14 Function Documentation

### 8.164.14.1 rtems_scheduler_add_processor()

```
rtems_status_code rtems_scheduler_add_processor (
            rtems_id scheduler_id,
            uint32_t cpu_index )
```

Adds the processor to the set of processors owned by the scheduler instance.

This directive shall be called from task context. It obtains and releases the objects allocator lock.

**Parameters**

| | |
|---|---|
| *scheduler↩ _id* | is the scheduler instance identifier. |
| *cpu_index* | is the index of the processor to add. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INCORRECT_STATE* | The processor was configured to be used by the application, however, it was not online. |
| *RTEMS_INVALID_ID* | The scheduler instance identifier was invalid. |
| *RTEMS_NOT_CONFIGURED* | The processor was not configured to be used by the application. |
| *RTEMS_RESOURCE_IN_USE* | The processor was already assigned to a scheduler instance. |

Definition at line 24 of file scheduleraddprocessor.c.

### 8.164.14.2 rtems_scheduler_get_maximum_priority()

```
rtems_status_code rtems_scheduler_get_maximum_priority (
            rtems_id scheduler_id,
            rtems_task_priority * priority )
```

Gets the maximum task priority of the scheduler instance.

**Parameters**

| | | |
|---|---|---|
| | *scheduler_id* | is the scheduler instance identifier. |
| out | *priority* | is the pointer to a task priority variable. The maximum priority of the scheduler instance will be stored in this variable, if the operation is successful. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The priority parameter was NULL. |
| *RTEMS_INVALID_ID* | The scheduler instance identifier was invalid. |

### 8.164.14.3 rtems_scheduler_get_processor_set()

```
rtems_status_code rtems_scheduler_get_processor_set (
            rtems_id scheduler_id,
            size_t cpusetsize,
            cpu_set_t * cpuset )
```

Gets the set of processors owned by the scheduler instance.

**Parameters**

| | | |
|---|---|---|
| | *scheduler_id* | is the scheduler instance identifier. |

**Parameters**

| | | |
|---|---|---|
| | *cpusetsize* | is the size of the referenced processor set variable in bytes. This value shall be positive. |
| out | *cpuset* | is the pointer to a processor set variable. The processor set of the scheduler instance will be stored in this variable, in case of a successful operation. A set bit in the processor set means that the corresponding processor is owned by the scheduler instance, otherwise the bit is cleared. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The cpuset parameter was NULL. |
| *RTEMS_INVALID_ID* | The scheduler instance identifier was invalid. |
| *RTEMS_INVALID_NUMBER* | The provided processor set was too small for the set of processors owned by the scheduler instance. |

Definition at line 22 of file schedulergetprocessorset.c.

### 8.164.14.4 rtems_scheduler_ident()

```
rtems_status_code rtems_scheduler_ident (
            rtems_name name,
            rtems_id * id )
```

Identifies a scheduler instance by its name.

The scheduler name is determined by the scheduler configuration.

**Parameters**

| | | |
|---|---|---|
| | *name* | is the scheduler name. |
| out | *id* | is the pointer to an object identifier variable. The identifier of the scheduler instance will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The id parameter was NULL. |
| *RTEMS_INVALID_NAME* | The scheduler name was invalid. |

Definition at line 22 of file scheduleridentbyprocessor.c.

### 8.164.14.5 rtems_scheduler_ident_by_processor()

```
rtems_status_code rtems_scheduler_ident_by_processor (
```

```
            uint32_t cpu_index,
            rtems_id * id )
```

Identifies a scheduler instance by a processor index.

**Parameters**

| | | |
|---|---|---|
| | *cpu_index* | is the processor index to identify the scheduler instance. |
| out | *id* | is the pointer to an object identifier variable. The identifier of the scheduler instance will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INCORRECT_STATE* | The processor index was valid, however, the corresponding processor was not owned by a scheduler instance. |
| *RTEMS_INVALID_ADDRESS* | The id parameter was NULL. |
| *RTEMS_INVALID_NAME* | The processor index was invalid. |

Definition at line 23 of file scheduleridentbyprocessor.c.

### 8.164.14.6 rtems_scheduler_ident_by_processor_set()

```
rtems_status_code rtems_scheduler_ident_by_processor_set (
            size_t cpusetsize,
            const cpu_set_t * cpuset,
            rtems_id * id )
```

Identifies a scheduler instance by a processor set.

The scheduler instance is selected according to the highest numbered online processor in the specified processor set.

**Parameters**

| | | |
|---|---|---|
| | *cpusetsize* | is the size of the referenced processor set variable in bytes. This value shall be positive. |
| | *cpuset* | is the pointer to a processor set variable. The referenced processor set will be used to identify the scheduler instance. |
| out | *id* | is the pointer to an object identifier variable. The identifier of the scheduler instance will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INCORRECT_STATE* | The processor set was valid, however, the highest numbered online processor in the processor set was not owned by a scheduler instance. |
| *RTEMS_INVALID_ADDRESS* | The id parameter was NULL. |
| *RTEMS_INVALID_NAME* | The processor set contained no online processor. |
| *RTEMS_INVALID_SIZE* | The processor set size was invalid. |

Definition at line 23 of file scheduleridentbyprocessorset.c.

### 8.164.14.7 rtems_scheduler_map_priority_from_posix()

```
rtems_status_code rtems_scheduler_map_priority_from_posix (
            rtems_id scheduler_id,
            int posix_priority,
            rtems_task_priority * priority )
```

Maps a POSIX thread priority to the corresponding Classic API task priority.

**Parameters**

| | | |
|---|---|---|
| | *scheduler_id* | is the scheduler instance identifier. |
| | *posix_priority* | is the POSIX thread priority to map. |
| out | *priority* | is the pointer to a Classic API task priority variable. The Classic API task priority value corresponding to the specified POSIX thread priority value will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The priority parameter was NULL. |
| *RTEMS_INVALID_ID* | The scheduler instance identifier was invalid. |
| *RTEMS_INVALID_PRIORITY* | The POSIX thread priority was invalid. |

### 8.164.14.8 rtems_scheduler_map_priority_to_posix()

```
rtems_status_code rtems_scheduler_map_priority_to_posix (
            rtems_id scheduler_id,
            rtems_task_priority priority,
            int * posix_priority )
```

Maps a Classic API task priority to the corresponding POSIX thread priority.

**Parameters**

| | | |
|---|---|---|
| | *scheduler_id* | is the scheduler instance identifier. |
| | *priority* | is the Classic API task priority to map. |
| out | *posix_priority* | is the pointer to a POSIX thread priority variable. The POSIX thread priority value corresponding to the specified Classic API task priority value will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |

**Return values**

| | |
|---|---|
| *RTEMS_INVALID_ADDRESS* | The posix_priority parameter was NULL. |
| *RTEMS_INVALID_ID* | The scheduler instance identifier was invalid. |
| *RTEMS_INVALID_PRIORITY* | The Classic API task priority was invalid. |

#### 8.164.14.9 rtems_scheduler_remove_processor()

```
rtems_status_code rtems_scheduler_remove_processor (
            rtems_id scheduler_id,
            uint32_t cpu_index )
```

Removes a processor from set of processors owned by the scheduler instance.

This directive shall be called from task context. It obtains and releases the objects allocator lock. Removing a processor from a scheduler instance is a complex operation that involves all tasks of the system.

**Parameters**

| | |
|---|---|
| *scheduler_id* | is the scheduler instance identifier. |
| *cpu_index* | is the index of the processor to remove. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ID* | The scheduler instance identifier was invalid. |
| *RTEMS_INVALID_NUMBER* | The processor was not owned by the specified scheduler instance. |
| *RTEMS_RESOURCE_IN_USE* | The set of processors owned by the specified scheduler instance would have been empty after the processor removal and there was at least one non-idle task that used this scheduler instance as its home scheduler instance. |

Definition at line 64 of file schedulerremoveprocessor.c.

#### 8.164.14.10 rtems_task_construct()

```
rtems_status_code rtems_task_construct (
            const rtems_task_config * config,
            rtems_id * id )
```

Constructs a task from the specified the task configuration.

In contrast to tasks created by rtems_task_create(), the tasks constructed by this directive use a user-provided task storage area. The task storage area contains the task stack, the thread-local storage, and the floating-point context on architectures with a separate floating-point context.

This directive is intended for applications which do not want to use the RTEMS Workspace and instead statically allocate all operating system resources. It is not recommended to use rtems_task_create() and rtems_task_construct() together in an application. It is also not recommended to use rtems_task_construct() for drivers or general purpose libraries. The reason for these recommendations is that the task configuration needs settings which can be only given with a through knowledge of the application resources.

An application based solely on static allocation can avoid any runtime memory allocators. This can simplify the application architecture as well as any analysis that may be required.

The stack space estimate done by <rtems/confdefs.h> assumes that all tasks are created by rtems_task_create(). The estimate can be adjusted to take user-provided task storage areas into account through the CONFIGURE_MINIMUM_TASKS_WIT application configuration option.

The CONFIGURE_MAXIMUM_TASKS should include tasks constructed by rtems_task_construct().

**Parameters**

| | | |
|---|---|---|
| | *config* | is the task configuration. |
| out | *id* | is the pointer to an object identifier variable. The identifier of the constructed task object will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The id parameter was NULL. |
| *RTEMS_INVALID_NAME* | The task name was invalid. |
| *RTEMS_INVALID_PRIORITY* | The initial task priority was invalid. |
| *RTEMS_INVALID_SIZE* | The thread-local storage size is greater than the maximum thread-local storage size specified in the task configuration. The thread-local storage size is determined by the thread-local variables used by the application and CONFIGURE_MAXIMUM_THREAD_LOCAL_STORAGE_SIZE. |
| *RTEMS_INVALID_SIZE* | The task storage area was too small to provide a task stack of the configured minimum size, see CONFIGURE_MINIMUM_TASK_STACK_SIZE. The task storage area contains the task stack, the thread-local storage, and the floating-point context on architectures with a separate floating-point context. |
| *RTEMS_TOO_MANY* | There was no inactive task object available to construct a task. |
| *RTEMS_TOO_MANY* | In multiprocessing configurations, there was no inactive global object available to construct a global task. |
| *RTEMS_UNSATISFIED* | One of the task create extensions failed during the task construction. |
| *RTEMS_UNSATISFIED* | In SMP configurations, the non-preemption mode was not supported. |
| *RTEMS_UNSATISFIED* | In SMP configurations, the interrupt level mode was not supported. |

Definition at line 73 of file taskconstruct.c.

### 8.164.14.11 rtems_task_create()

```
rtems_status_code rtems_task_create (
             rtems_name name,
             rtems_task_priority initial_priority,
             size_t stack_size,
```

```
            rtems_mode initial_modes,
            rtems_attribute attribute_set,
            rtems_id * id )
```

Creates a task object.

This directive creates a task which resides on the local node. It allocates and initializes a TCB, a stack, and an optional floating point context area. The mode parameter contains values which sets the task's initial execution mode. The RTEMS_FLOATING_POINT attribute should be specified if the created task is to use a numeric coprocessor. For performance reasons, it is recommended that tasks not using the numeric coprocessor should specify the RTEMS_NO_FLOATING_POINT attribute. If the RTEMS_GLOBAL attribute is specified, the task can be accessed from remote nodes. The task id, returned in id, is used in other task related directives to access the task. When created, a task is placed in the dormant state and can only be made ready to execute using the directive rtems_task_start().

**Parameters**

| | | |
|---|---|---|
| | *name* | is the user-defined task name. |
| | *initial_priority* | is the initial task priority. |
| | *stack_size* | is the task stack size in bytes. |
| | *initial_modes* | is the initial task mode. |
| | *attribute_set* | is the task attribute set. |
| out | *id* | is the pointer to an object identifier variable. The object identifier of the new task will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The id parameter was NULL. |
| *RTEMS_INVALID_NAME* | The task name was invalid. |
| *RTEMS_INVALID_PRIORITY* | The initial task priority was invalid. |
| *RTEMS_MP_NOT_CONFIGURED* | The multiprocessing support was not configured. |
| *RTEMS_TOO_MANY* | There was no inactive task object available to create a new task. |
| *RTEMS_TOO_MANY* | In multiprocessing configurations, there was no inactive global object available to create a new global task. |
| *RTEMS_UNSATISFIED* | There was not enough memory to allocate the task storage area. The task storage area contains the task stack, the thread-local storage, and the floating point context. |
| *RTEMS_UNSATISFIED* | One of the task create extensions failed to create the new task. |
| *RTEMS_UNSATISFIED* | In SMP configurations, the non-preemption mode was not supported. |
| *RTEMS_UNSATISFIED* | In SMP configurations, the interrupt level mode was not supported. |

### 8.164.14.12 rtems_task_delete()

```
rtems_status_code rtems_task_delete (
            rtems_id id )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |

Definition at line 24 of file taskdelete.c.

#### 8.164.14.13 rtems_task_get_affinity()

```
rtems_status_code rtems_task_get_affinity (
              rtems_id id,
              size_t cpusetsize,
              cpu_set_t * cpuset )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *cpusetsize* | % |
| *cpuset* | % |

Definition at line 26 of file taskgetaffinity.c.

#### 8.164.14.14 rtems_task_get_priority()

```
rtems_status_code rtems_task_get_priority (
              rtems_id task_id,
              rtems_id scheduler_id,
              rtems_task_priority * priority )
```

%

**Parameters**

| | |
|---|---|
| *task_id* | % |
| *scheduler↩ _id* | % |
| *priority* | % |

Definition at line 23 of file taskgetpriority.c.

#### 8.164.14.15 rtems_task_get_scheduler()

```
rtems_status_code rtems_task_get_scheduler (
```

```
            rtems_id task_id,
            rtems_id * scheduler_id )
```

%

**Parameters**

| | |
|---|---|
| *task_id* | % |
| *scheduler_id* | % |

Definition at line 22 of file taskgetscheduler.c.

#### 8.164.14.16 rtems_task_ident()

```
rtems_status_code rtems_task_ident (
            rtems_name name,
            uint32_t node,
            rtems_id * id )
```

Identifies a task object by the specified object name.

This directive obtains the task identifier associated with the task name specified in `name`.

A task may obtain its own identifier by specifying RTEMS_SELF for the name.

The node to search is specified in `node`. It shall be

- a valid node number,
- the constant RTEMS_SEARCH_ALL_NODES to search in all nodes,
- the constant RTEMS_SEARCH_LOCAL_NODE to search in the local node only, or
- the constant RTEMS_SEARCH_OTHER_NODES to search in all nodes except the local node.

If the task name is not unique, then the task identifier will match the first task with that name in the search order. However, this task identifier is not guaranteed to correspond to the desired task. The task identifier is used with other task related directives to access the task.

If node is RTEMS_SEARCH_ALL_NODES, all nodes are searched with the local node being searched first. All other nodes are searched with the lowest numbered node searched first.

If node is a valid node number which does not represent the local node, then only the tasks exported by the designated node are searched.

This directive does not generate activity on remote nodes. It accesses only the local copy of the global object table.

**Parameters**

| | | |
|---|---|---|
| | *name* | is the object name to look up. |
| | *node* | is the node or node set to search for a matching object. |
| `out` | *id* | is the pointer to an object identifier variable. The object identifier of an object with the |
| | | specified name will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The id parameter was NULL. |
| *RTEMS_INVALID_NAME* | There was no object with the specified name on the specified nodes. |
| *RTEMS_INVALID_NODE* | In multiprocessing configurations, the specified node was invalid. |

Definition at line 45 of file taskident.c.

### 8.164.14.17 rtems_task_is_suspended()

```
rtems_status_code rtems_task_is_suspended (
            rtems_id id )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |

Definition at line 25 of file taskissuspended.c.

### 8.164.14.18 rtems_task_iterate()

```
void rtems_task_iterate (
            rtems_task_visitor visitor,
            void * arg )
```

%

**Parameters**

| | |
|---|---|
| *visitor* | % |
| *arg* | % |

### 8.164.14.19 rtems_task_mode()

```
rtems_status_code rtems_task_mode (
            rtems_mode mode_set,
            rtems_mode mask,
            rtems_mode * previous_mode_set )
```

%

**Parameters**

| | |
|---|---|
| *mode_set* | % |
| *mask* | % |
| *previous_mode_set* | % |

#### 8.164.14.20 rtems_task_restart()

```
rtems_status_code rtems_task_restart (
            rtems_id id,
            rtems_task_argument argument )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *argument* | % |

Definition at line 24 of file taskrestart.c.

#### 8.164.14.21 rtems_task_resume()

```
rtems_status_code rtems_task_resume (
            rtems_id id )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |

Definition at line 24 of file taskresume.c.

#### 8.164.14.22 rtems_task_set_affinity()

```
rtems_status_code rtems_task_set_affinity (
            rtems_id id,
            size_t cpusetsize,
            const cpu_set_t * cpuset )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *cpusetsize* | % |
| *cpuset* | % |

Definition at line 26 of file tasksetaffinity.c.

#### 8.164.14.23 rtems_task_set_priority()

```
rtems_status_code rtems_task_set_priority (
             rtems_id id,
             rtems_task_priority new_priority,
             rtems_task_priority * old_priority )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *new_priority* | % |
| *old_priority* | % |

Definition at line 61 of file tasksetpriority.c.

#### 8.164.14.24 rtems_task_set_scheduler()

```
rtems_status_code rtems_task_set_scheduler (
             rtems_id task_id,
             rtems_id scheduler_id,
             rtems_task_priority priority )
```

%

**Parameters**

| | |
|---|---|
| *task_id* | % |
| *scheduler↩ _id* | % |
| *priority* | % |

Definition at line 23 of file tasksetscheduler.c.

#### 8.164.14.25 rtems_task_start()

```
rtems_status_code rtems_task_start (
            rtems_id id,
            rtems_task_entry entry_point,
            rtems_task_argument argument )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *entry_point* | % |
| *argument* | % |

Definition at line 25 of file taskstart.c.

#### 8.164.14.26 rtems_task_suspend()

```
rtems_status_code rtems_task_suspend (
            rtems_id id )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |

Definition at line 24 of file tasksuspend.c.

#### 8.164.14.27 rtems_task_wake_after()

```
rtems_status_code rtems_task_wake_after (
            rtems_interval ticks )
```

%

**Parameters**

| | |
|---|---|
| *ticks* | % |

Definition at line 25 of file taskwakeafter.c.

### 8.164.14.28 rtems_task_wake_when()

```
rtems_status_code rtems_task_wake_when (
            rtems_time_of_day * time_buffer )
```

%

**Parameters**

| | |
|---|---|
| *time_buffer* | % |

## 8.165 Task Modes

This group contains the Classic API task modes.

### Macros

- #define RTEMS_ALL_MODE_MASKS 0x0000ffff

  *This task mode constant is a bit mask with all mode bits set.*
- #define RTEMS_ASR 0x00000000

  *This task mode constant indicates that signal processing is disabled.*
- #define RTEMS_ASR_MASK 0x00000400

  *This mode constant corresponds to the signal enable/disable bit.*
- #define RTEMS_CURRENT_MODE 0

  *This task mode constant indicates that the current task mode of the executing task shall be returned by rtems_task_mode().*
- #define RTEMS_DEFAULT_MODES 0x00000000

  *This task mode constant represents the default mode set.*
- #define RTEMS_INTERRUPT_MASK CPU_MODES_INTERRUPT_MASK

  *This task mode constant corresponds to the interrupt enable/disable bits.*
- #define RTEMS_INTERRUPT_LEVEL(_interrupt_level) ( ( _interrupt_level ) & RTEMS_INTERRUPT_MASK )

  *Maps the interrupt level to the associated processor-dependent task mode interrupt level.*
- #define RTEMS_NO_ASR 0x00000400

  *This task mode constant indicates that signal processing is disabled.*
- #define RTEMS_NO_PREEMPT 0x00000100

  *This task mode constant indicates that preemption is disabled.*
- #define RTEMS_NO_TIMESLICE 0x00000000

  *This task mode constant indicates that timeslicing is disabled.*
- #define RTEMS_PREEMPT 0x00000000

  *This task mode constant indicates that preemption is enabled.*
- #define RTEMS_PREEMPT_MASK 0x00000100

  *This task mode constant corresponds to the preemption enable/disable bit.*
- #define RTEMS_TIMESLICE 0x00000200

  *This task mode constant indicates that timeslicing is enabled.*
- #define RTEMS_TIMESLICE_MASK 0x00000200

  *This task mode constant corresponds to the timeslice enable/disable bit.*

### Typedefs

- typedef uint32_t rtems_mode

  *This type represents a Classic API task mode set.*

### Functions

- rtems_mode rtems_interrupt_level_body (uint32_t level)

  *Maps the interrupt level to the associated processor-dependent task mode interrupt level.*

## Variables

- const uint32_t rtems_interrupt_mask

  *This task mode constant has the same value as RTEMS_INTERRUPT_MASK.*

## 8.165.1 Detailed Description

This group contains the Classic API task modes.

## 8.165.2 Macro Definition Documentation

### 8.165.2.1 RTEMS_INTERRUPT_LEVEL

```
#define RTEMS_INTERRUPT_LEVEL(
              _interrupt_level )  ( ( _interrupt_level ) & RTEMS_INTERRUPT_MASK )
```

Maps the interrupt level to the associated processor-dependent task mode interrupt level.

The Classic API supports 256 interrupt levels using the least significant eight bits of the mode set. On any particular processor variant, fewer than 256 levels may be supported. At least level 0 (all interrupts enabled) and level 1 (interrupts disabled, on most architectures) are supported.

**Parameters**

| | |
|---|---|
| *_interrupt_level* | is the interrupt level to map. |

**Returns**

null Returns the processor-dependent task mode interrupt level associated with the interrupt level.

Definition at line 145 of file modes.h.

## 8.165.3 Function Documentation

### 8.165.3.1 rtems_interrupt_level_body()

```
rtems_mode rtems_interrupt_level_body (
             uint32_t level )
```

Maps the interrupt level to the associated processor-dependent task mode interrupt level.

This function is used by bindings from languages other than C and C++.

**Parameters**

| | |
|---|---|
| *level* | is the interrupt level to map. |

**Returns**

Returns RTEMS_INTERRUPT_LEVEL() for the interrupt level.

## 8.165.4 Variable Documentation

### 8.165.4.1 rtems_interrupt_mask

```
const uint32_t rtems_interrupt_mask  [extern]
```

This task mode constant has the same value as RTEMS_INTERRUPT_MASK.

This task mode constant is used by bindings from languages other than C and C++.

# 8.166 Task Stack Allocator Configuration

## Macros

- #define CONFIGURE_TASK_STACK_ALLOCATOR

  *This configuration option is an initializer define.*
- #define CONFIGURE_TASK_STACK_ALLOCATOR_AVOIDS_WORK_SPACE

  *This configuration option is a boolean feature define.*
- #define CONFIGURE_TASK_STACK_ALLOCATOR_INIT

  *This configuration option is an initializer define.*
- #define CONFIGURE_TASK_STACK_DEALLOCATOR

  *This configuration option is an initializer define.*
- #define CONFIGURE_TASK_STACK_FROM_ALLOCATOR

  *This configuration option is an initializer define.*

## 8.166.1 Detailed Description

This section describes configuration options related to the task stack allocator. RTEMS allows the application or BSP to define its own allocation and deallocation methods for task stacks. This can be used to place task stacks in special areas of memory or to utilize a Memory Management Unit so that stack overflows are detected in hardware.

## 8.166.2 Macro Definition Documentation

### 8.166.2.1 CONFIGURE_TASK_STACK_ALLOCATOR

```
#define CONFIGURE_TASK_STACK_ALLOCATOR
```

This configuration option is an initializer define.

The value of this configuration option initializes the stack allocator allocate handler.

#### Default Value

The default value is `_Workspace_Allocate`, which indicates that task stacks will be allocated from the RTEMS Workspace.

#### Value Constraints

The value of this configuration option shall be defined to a valid function pointer of the type `void *( *allocate )( size_t )`.

#### Notes

A correctly configured system shall configure the following to be consistent:

- CONFIGURE_TASK_STACK_ALLOCATOR_INIT
- `CONFIGURE_TASK_STACK_ALLOCATOR`
- CONFIGURE_TASK_STACK_DEALLOCATOR

Definition at line 4490 of file appl-config.h.

### 8.166.2.2 CONFIGURE_TASK_STACK_ALLOCATOR_AVOIDS_WORK_SPACE

```
#define CONFIGURE_TASK_STACK_ALLOCATOR_AVOIDS_WORK_SPACE
```

This configuration option is a boolean feature define.

In case this configuration option is defined, then the system is informed that the task stack allocator does not use the RTEMS Workspace.

#### Default Configuration

If this configuration option is undefined, then the described feature is not enabled.

#### Notes

This configuration option may be used if a custom task stack allocator is configured, see CONFIGURE_TASK_STACK_ALLOCAT

Definition at line 4508 of file appl-config.h.

### 8.166.2.3 CONFIGURE_TASK_STACK_ALLOCATOR_INIT

```
#define CONFIGURE_TASK_STACK_ALLOCATOR_INIT
```

This configuration option is an initializer define.

The value of this configuration option initializes the stack allocator initialization handler.

#### Default Value

The default value is `NULL`.

#### Value Constraints

The value of this configuration option shall be defined to a valid function pointer of the type `void ( *initialize )( size_t )` or to `NULL`.

#### Notes

A correctly configured system shall configure the following to be consistent:

- `CONFIGURE_TASK_STACK_ALLOCATOR_INIT`
- CONFIGURE_TASK_STACK_ALLOCATOR
- CONFIGURE_TASK_STACK_DEALLOCATOR

Definition at line 4539 of file appl-config.h.

### 8.166.2.4 CONFIGURE_TASK_STACK_DEALLOCATOR

```
#define CONFIGURE_TASK_STACK_DEALLOCATOR
```

This configuration option is an initializer define.

The value of this configuration option initializes the stack allocator deallocate handler.

#### Default Value

The default value is `_Workspace_Free`, which indicates that task stacks will be allocated from the RTEMS Workspace.

#### Value Constraints

The value of this configuration option shall be defined to a valid function pointer of the type `void ( *deallocate )( void * )`.

#### Notes

A correctly configured system shall configure the following to be consistent:

- CONFIGURE_TASK_STACK_ALLOCATOR_INIT
- CONFIGURE_TASK_STACK_ALLOCATOR
- `CONFIGURE_TASK_STACK_DEALLOCATOR`

Definition at line 4569 of file appl-config.h.

### 8.166.2.5 CONFIGURE_TASK_STACK_FROM_ALLOCATOR

```
#define CONFIGURE_TASK_STACK_FROM_ALLOCATOR
```

This configuration option is an initializer define.

The value of this configuration option is used to calculate the task stack space size.

#### Default Value

The default value is a macro which supports the system heap allocator.

#### Value Constraints

The value of this configuration option shall be defined to a macro which accepts exactly one parameter and returns an unsigned integer. The parameter will be an allocation size and the macro shall return this size plus the overhead of the allocator to manage an allocation request for this size.

#### Notes

This configuration option may be used if a custom task stack allocator is configured, see CONFIGURE_TASK_STACK_ALLOCAT

Definition at line 4593 of file appl-config.h.

## 8.167 Termios

Termios.

### Files

- file termiosinitialize.c

  *Termios Initialization.*

### Classes

- struct rtems_termios_callbacks

### Typedefs

- typedef struct rtems_termios_callbacks **rtems_termios_callbacks**
- typedef rtems_termios_iproc_status_code(∗ rtems_termios_isig_handler) (unsigned char c, struct rtems_termios_tty ∗tty)

  *Type for ISIG (VINTR/VKILL) handler.*

### Enumerations

- enum rtems_termios_iproc_status_code { RTEMS_TERMIOS_IPROC_CONTINUE, RTEMS_TERMIOS_IPROC_INTERRUPT RTEMS_TERMIOS_IPROC_DONE }

  *The status code returned by all Termios input processing (iproc) functions and input signal (isig) handlers.*

### Functions

- static __inline__ void **rtems_termios_initialize** (void)
- rtems_status_code **rtems_termios_bufsize** (size_t cbufsize, size_t raw_input, size_t raw_output)
- rtems_termios_iproc_status_code rtems_termios_default_isig_handler (unsigned char c, struct rtems_termios_tty ∗tty)

  *Default handler for ISIG (VINTR/VKILL)*
- rtems_termios_iproc_status_code rtems_termios_posix_isig_handler (unsigned char c, struct rtems_termios_tty ∗tty)

  *POSIX handler for ISIG (VINTR/VKILL)*
- rtems_status_code rtems_termios_register_isig_handler (rtems_termios_isig_handler handler)

  *Register handler for ISIG (VINTR/VKILL)*
- int **rtems_termios_enqueue_raw_characters** (void ∗ttyp, const char ∗buf, int len)
- rtems_status_code **rtems_termios_open** (rtems_device_major_number major, rtems_device_minor_number minor, void ∗arg, const rtems_termios_callbacks ∗callbacks)
- rtems_status_code **rtems_termios_close** (void ∗arg)
- rtems_status_code **rtems_termios_read** (void ∗arg)
- rtems_status_code **rtems_termios_write** (void ∗arg)
- rtems_status_code **rtems_termios_ioctl** (void ∗arg)
- int **rtems_termios_dequeue_characters** (void ∗ttyp, int len)

## 8.167.1 Detailed Description

Termios.

## 8.167.2 Typedef Documentation

### 8.167.2.1 rtems_termios_isig_handler

```
typedef rtems_termios_iproc_status_code(* rtems_termios_isig_handler) (unsigned char c, struct rtems_termios_tty *tty)
```

Type for ISIG (VINTR/VKILL) handler.

This type defines the interface to a method which will process VKILL and VINTR characters. The user can register a handler to override the default behavior which is to take no special action on these characters. The POSIX required behavior is to generate a SIGINTR or SIGQUIT signal. This behavior is implemented if the user registers the rtems_termios_posix_isig_handler().

**Parameters**

| | |
|---|---|
| *c* | character that was input |
| *tty* | termios structure pointer for this input tty |

**Returns**

The value returned is according to that documented for rtems_termios_iproc_status_code.

Definition at line 1957 of file libio.h.

## 8.167.3 Enumeration Type Documentation

### 8.167.3.1 rtems_termios_iproc_status_code

```
enum rtems_termios_iproc_status_code
```

The status code returned by all Termios input processing (iproc) functions and input signal (isig) handlers.

**Enumerator**

| | |
|---|---|
| RTEMS_TERMIOS_IPROC_CONTINUE | This status indicates that the input processing can continue. Input signal handlers shall return this status if no signal was raised and the input processing can continue. |

**Enumerator**

| | |
|---|---|
| RTEMS_TERMIOS_IPROC_INTERRUPT | This status indicates that the input processing should stop due to a signal. This indicates that the character was processed and determined to be one that requires a signal to be raised (e.g. VINTR or VKILL). The tty must be in the right Termios mode for this to occur. There is no further processing of this character required and the pending read() operation should be interrupted. |
| RTEMS_TERMIOS_IPROC_DONE | This status indicates that the input processing is done. This status shall not be returned by input processing signal handlers. |

Definition at line 1912 of file libio.h.

## 8.167.4 Function Documentation

### 8.167.4.1 rtems_termios_default_isig_handler()

```
rtems_termios_iproc_status_code rtems_termios_default_isig_handler (
            unsigned char c,
            struct rtems_termios_tty * tty )
```

Default handler for ISIG (VINTR/VKILL)

This handler is installed by default by termios. It performs no actions on receiving the VINTR and VKILL characters. This is not POSIX conformant behavior but is the historical RTEMS behavior and avoid any default dependency on signal processing code.

**Parameters**

| | |
|---|---|
| *c* | character that was input |
| *tty* | termios structure pointer for this input tty |

The installed ISIG handler is only invoked if the character is (VKILL or VINTR) and ISIG is enabled. Thus the handler need only check the character and perform the appropriate action.

**Returns**

The value returned is according to that documented for all methods adhering to rtems_termios_isig_handler.

### 8.167.4.2 rtems_termios_posix_isig_handler()

```
rtems_termios_iproc_status_code rtems_termios_posix_isig_handler (
            unsigned char c,
            struct rtems_termios_tty * tty )
```

POSIX handler for ISIG (VINTR/VKILL)

This handler is installed by the used to obtain POSIX behavior upon receiving the VINTR and VKILL characters. The POSIX required behavior is to generate a SIGINTR or SIGQUIT signal if ISIG is enabled.

**Parameters**

| | |
|---|---|
| *c* | character that was input |
| *tty* | termios structure pointer for this input tty |

**Returns**

The value returned is according to that documented for all methods adhering to rtems_termios_isig_handler.

### 8.167.4.3 rtems_termios_register_isig_handler()

```
rtems_status_code rtems_termios_register_isig_handler (
            rtems_termios_isig_handler handler )
```

Register handler for ISIG (VINTR/VKILL)

This method is invoked to install an ISIG handler. This may be used to install rtems_termios_posix_isig_handler() to obtain POSIX behavior or a custom handler. The handler rtems_termios_default_isig_handler() is installed by default.

**Parameters**

| | |
|---|---|
| *handler* | entry point of the new ISIG handler |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | if successful or error code if unsuccessful. |

## 8.168 Test Suites

This group contains the test suites.

### Modules

- spec:/testsuites/validation-0

  *This general purpose validation test suite provides enough resources to run basic tests for all specified managers and functions.*
- spec:/testsuites/validation/profile

  *This test suite contains test cases which call in combination all functions included in the space profile.*

### 8.168.1 Detailed Description

This group contains the test suites.

# 8.169 Test Support

Test support functions.

## Classes

- struct rtems_test_parallel_context

  *Internal context for parallel job execution.*
- struct rtems_test_parallel_job

  *Basic parallel job description.*

## Macros

- #define RTEMS_TEST_INITIAL_EXTENSION { NULL, NULL, NULL, NULL, NULL, NULL, NULL, rtems_test_fatal_extension }

  *Initial extension for tests.*
- #define **TEST_STATE** RTEMS_TEST_STATE_PASS
- #define **RTEMS_TEST_PARALLEL_PROCESSOR_MAX** 32

## Typedefs

- typedef struct rtems_test_parallel_job **rtems_test_parallel_job**
- typedef void(∗ rtems_test_parallel_worker_setup) (rtems_test_parallel_context ∗ctx, size_t worker_index, rtems_id worker_id)

  *Worker task setup handler.*

## Enumerations

- enum RTEMS_TEST_STATE {
  **RTEMS_TEST_STATE_PASS**, **RTEMS_TEST_STATE_FAIL**, **RTEMS_TEST_STATE_USER_INPUT**, **R↩ TEMS_TEST_STATE_INDETERMINATE**,
  **RTEMS_TEST_STATE_BENCHMARK** }

  *Test states.*

## Functions

- void rtems_test_fatal_extension (rtems_fatal_source source, bool always_set_to_false, rtems_fatal_code code)

  *Fatal extension for tests.*
- int rtems_test_begin (const char ∗name, const RTEMS_TEST_STATE state)

  *Prints a begin of test message using printf().*
- int rtems_test_end (const char ∗name)

  *Prints an end of test message using printf().*
- RTEMS_NO_RETURN void rtems_test_exit (int status)

  *Exit the test without calling exit() since it closes stdin, etc and pulls in stdio code.*
- int rtems_test_printf (const char ∗format,...) RTEMS_PRINTFLIKE(1

  *Prints via the RTEMS printer.*
- static bool rtems_test_parallel_stop_job (const rtems_test_parallel_context ∗ctx)

*Indicates if a job body should stop its work loop.*

- static bool rtems_test_parallel_is_master_worker (size_t worker_index)

  *Indicates if a worker is the master worker.*
- static rtems_id rtems_test_parallel_get_task_id (const rtems_test_parallel_context *ctx, size_t worker_index)

  *Returns the task identifier for a worker.*
- void rtems_test_parallel (rtems_test_parallel_context *ctx, rtems_test_parallel_worker_setup worker_setup, const rtems_test_parallel_job *jobs, size_t job_count)

  *Runs a bunch of jobs in parallel on all processors of the system.*
- void rtems_test_busy_cpu_usage (time_t seconds, long nanoseconds)

  *Performs a busy loop for the specified seconds and nanoseconds based on the CPU usage of the executing thread.*
- RTEMS_NO_RETURN void rtems_test_run (rtems_task_argument arg, const RTEMS_TEST_STATE state)

  *Runs the test cases of the RTEMS Test Framework using a default configuration in the context of a task.*

## Variables

- const char rtems_test_name [ ]

  *Each test must define a test name string.*
- rtems_printer rtems_test_printer

  *Each test must define a printer.*

### 8.169.1 Detailed Description

Test support functions.

### 8.169.2 Typedef Documentation

#### 8.169.2.1 rtems_test_parallel_worker_setup

```
typedef void(* rtems_test_parallel_worker_setup) (rtems_test_parallel_context *ctx, size_↩
t worker_index, rtems_id worker_id)
```

Worker task setup handler.

Called during rtems_test_parallel() to optionally setup a worker task before it is started.

**Parameters**

| | | |
|---|---|---|
| in | *ctx* | The parallel context. |
| in | *worker_index* | The worker index. |
| in | *worker_id* | The worker task identifier. |

Definition at line 149 of file test-info.h.

## 8.169.3 Function Documentation

### 8.169.3.1 rtems_test_begin()

```
int rtems_test_begin (
            const char * name,
            const RTEMS_TEST_STATE state )
```

Prints a begin of test message using printf().

**Returns**

As specified by printf().

Definition at line 38 of file testbeginend.c.

### 8.169.3.2 rtems_test_busy_cpu_usage()

```
void rtems_test_busy_cpu_usage (
            time_t seconds,
            long nanoseconds )
```

Performs a busy loop for the specified seconds and nanoseconds based on the CPU usage of the executing thread.

This function continuously reads the CPU usage of the executing thread. This operation may lead to a scheduler instance lock contention in SMP configurations.

**Parameters**

| | | |
|---|---|---|
| in | *seconds* | The busy seconds. |
| in | *nanoseconds* | The busy nanoseconds. |

### 8.169.3.3 rtems_test_end()

```
int rtems_test_end (
            const char * name )
```

Prints an end of test message using printf().

**Returns**

As specified by printf().

Definition at line 75 of file testbeginend.c.

#### 8.169.3.4 rtems_test_parallel()

```
void rtems_test_parallel (
            rtems_test_parallel_context * ctx,
            rtems_test_parallel_worker_setup worker_setup,
            const rtems_test_parallel_job * jobs,
            size_t job_count )
```

Runs a bunch of jobs in parallel on all processors of the system.

The worker tasks inherit the priority of the executing task.

There are SMP barriers before and after the job body.

**Parameters**

| | | |
|---|---|---|
| in | *ctx* | The parallel context. |
| in | *worker_setup* | Optional handler to setup a worker task before it is started. |
| in | *jobs* | The table of jobs. |
| in | *job_count* | The count of jobs in the job table. |

#### 8.169.3.5 rtems_test_parallel_get_task_id()

```
static rtems_id rtems_test_parallel_get_task_id (
            const rtems_test_parallel_context * ctx,
            size_t worker_index )  [inline], [static]
```

Returns the task identifier for a worker.

**Parameters**

| | | |
|---|---|---|
| in | *ctx* | The parallel context. |
| in | *worker_index* | The worker index. |

**Returns**

The task identifier of the worker.

Definition at line 266 of file test-info.h.

#### 8.169.3.6 rtems_test_parallel_is_master_worker()

```
static bool rtems_test_parallel_is_master_worker (
            size_t worker_index )  [inline], [static]
```

Indicates if a worker is the master worker.

The master worker is the thread that called rtems_test_parallel().

**Parameters**

| | | |
|---|---|---|
| in | *worker_index* | The worker index. |

**Return values**

| | |
|---|---|
| *true* | This is the master worker. |
| *false* | Otherwise. |

Definition at line 253 of file test-info.h.

### 8.169.3.7 rtems_test_parallel_stop_job()

```
static bool rtems_test_parallel_stop_job (
              const rtems_test_parallel_context * ctx )  [inline], [static]
```

Indicates if a job body should stop its work loop.

**Parameters**

| | | |
|---|---|---|
| in | *ctx* | The parallel context. |

**Return values**

| | |
|---|---|
| *true* | The job body should stop its work loop and return to the caller. |
| *false* | Otherwise. |

Definition at line 236 of file test-info.h.

### 8.169.3.8 rtems_test_printf()

```
int rtems_test_printf (
              const char * format,
               ...  )
```

Prints via the RTEMS printer.

**Returns**

As specified by printf().

#### 8.169.3.9 rtems_test_run()

```
RTEMS_NO_RETURN void rtems_test_run (
              rtems_task_argument arg,
              const RTEMS_TEST_STATE state )
```

Runs the test cases of the RTEMS Test Framework using a default configuration in the context of a task.

The application must provide rtems_test_name.

**Parameters**

| | |
|---|---|
| *arg* | is the task argument. |
| *state* | is the test state. |

Definition at line 72 of file testrun.c.

# 8.170 Thread Handler

Thread Handler.

## Files

- file thread.h

  *Constants and Structures Related with the Thread Control Block.*
- file threaddispatch.h

  *Constants and Structures Related with Thread Dispatch.*
- file threadidledata.h

  *Constants for the idle threads.*
- file threadimpl.h

  *Inlined Routines from the Thread Handler.*
- file thread.c

  *Initialize Thread Handler.*
- file threadchangepriority.c

  *Changes the Priority of a Thread.*
- file threadclearstate.c

  *Clear Thread state.*
- file threadcreateidle.c

  *Create Idle Thread.*
- file threaddispatch.c

  *Dispatch Thread.*
- file threadget.c

  *Maps Thread IDs to TCB Pointer.*
- file threadhandler.c

  *Thread Handler.*
- file threadinitialize.c

  *Initialize Thread.*
- file threadloadenv.c

  *Initializes Enviroment for A Thread.*
- file threadrestart.c

  *Restart Thread.*
- file threadsetstate.c

  *Sets States for a Thread.*
- file threadstart.c

  *Initializes Thread and Executes it.*
- file threadstartmultitasking.c

  *Start Thread Multitasking.*
- file threadtimeout.c

  *Thread Wait Timeout.*
- file threadyield.c

  *Thread Yield.*

## Classes

- struct Thread_Entry_idle

  *Data for idle thread entry.*
- struct Thread_Entry_numeric

  *Data for thread entry with one numeric argument and no return value.*
- struct Thread_Entry_pointer

  *Data for thread entry with one pointer argument and a pointer return value.*
- struct Thread_Entry_information

  *Thread entry information.*
- struct Thread_Start_information
- struct Thread_Scheduler_control

  *Thread scheduler control.*
- union Thread_Wait_information_Object_argument_type

  *Union type to hold a pointer to an immutable or a mutable object.*
- struct Thread_Wait_information

  *Information required to manage a thread while it is blocked.*
- struct Thread_Timer_information

  *Information required to manage a thread timer.*
- struct Thread_Proxy_control
- struct Thread_Action

  *Thread action.*
- struct Thread_Keys_information

  *Per-thread information for POSIX Keys.*
- struct Thread_Action_control

  *Control block to manage thread actions.*
- struct Thread_Life_control

  *Thread life control.*
- struct Thread_Capture_control
- struct _Thread_Control
- struct Thread_Control_add_on

  *Thread control add-on.*
- struct Thread_Information

  *The thread object information.*
- struct Thread_Configuration

  *The configuration of a new thread to initialize.*
- struct Thread_Close_context

## Macros

- #define **RTEMS_SCORE_THREAD_ENABLE_EXHAUST_TIMESLICE**
- #define **RTEMS_SCORE_THREAD_ENABLE_SCHEDULER_CALLOUT**
- #define THREAD_API_FIRST THREAD_API_RTEMS
- #define THREAD_API_LAST THREAD_API_POSIX
- #define THREAD_DEFAULT_MAXIMUM_NAME_SIZE 16

  *The default maximum size of a thread name in characters (including the terminating '\0' character).*
- #define **THREAD_INFORMATION_DEFINE_ZERO**(name, api, cls)
- #define **THREAD_INFORMATION_DEFINE**(name, api, cls, max)
- #define RTEMS_SCORE_ROBUST_THREAD_DISPATCH

  *Enables a robust thread dispatch.*

- #define THREAD_OF_SCHEDULER_HELP_NODE(node) RTEMS_CONTAINER_OF( node, Thread_Control, Scheduler.Help_node )
- #define **THREAD_QUEUE_CONTEXT_OF_REQUEST**(node) RTEMS_CONTAINER_OF( node, Thread_queue_Context, Lock_context.Wait.Gate.Node )
- #define THREAD_WAIT_FLAGS_INITIAL 0x0U

  *The initial thread wait flags value set by _Thread_Initialize().*
- #define THREAD_WAIT_STATE_MASK 0xffU

  *Mask to get the thread wait state flags.*
- #define THREAD_WAIT_STATE_INTEND_TO_BLOCK 0x1U

  *Indicates that the thread begins with the blocking operation.*
- #define THREAD_WAIT_STATE_BLOCKED 0x2U

  *Indicates that the thread completed the blocking operation.*
- #define THREAD_WAIT_STATE_READY_AGAIN 0x4U

  *Indicates that a condition to end the thread wait occurred.*
- #define THREAD_WAIT_CLASS_MASK 0xff00U

  *Mask to get the thread wait class flags.*
- #define THREAD_WAIT_CLASS_EVENT 0x100U

  *Indicates that the thread waits for an event.*
- #define THREAD_WAIT_CLASS_SYSTEM_EVENT 0x200U

  *Indicates that the thread waits for a system event.*
- #define THREAD_WAIT_CLASS_OBJECT 0x400U

  *Indicates that the thread waits for an object.*
- #define THREAD_WAIT_CLASS_PERIOD 0x800U

  *Indicates that the thread waits for a period.*
- #define **THREAD_PIN_STEP** 2
- #define **THREAD_PIN_PREEMPTION** 1

## Typedefs

- typedef CPU_Uint32ptr Thread_Entry_numeric_type

  *Type of the numeric argument of a thread entry function with at least one numeric argument.*
- typedef void(* Thread_CPU_budget_algorithm_callout) (Thread_Control *)
- typedef unsigned int Thread_Wait_flags

  *This type is able to contain several flags used to control the wait class and state of a thread.*
- typedef struct Thread_Action **Thread_Action**
- typedef void(* Thread_Action_handler) (Thread_Control *the_thread, Thread_Action *action, ISR_lock_Context *lock_context)

  *Thread action handler.*
- typedef void(* **rtems_per_thread_routine**) (Thread_Control *)
- typedef struct Thread_Configured_control Thread_Configured_control

  *The configured thread control block.*
- typedef struct Thread_queue_Configured_heads Thread_queue_Configured_heads

  *The configured thread queue heads.*
- typedef void *(* Thread_Idle_body) (uintptr_t)

  *The idle thread body type.*
- typedef bool(* **Thread_Visitor**) (Thread_Control *the_thread, void *arg)

## Enumerations

- enum Thread_CPU_budget_algorithms { **THREAD_CPU_BUDGET_ALGORITHM_NONE**, **THREAD_CPU_BUDGET_ALGORITHM_RESET_TIMESLICE**, **THREAD_CPU_BUDGET_ALGORITHM_EXHAUST_TIMESLICE**, **THREAD_CPU_BUDGET_ALGORITHM_CALLOUT** }
- enum Thread_Scheduler_state { THREAD_SCHEDULER_BLOCKED, THREAD_SCHEDULER_SCHEDULED, THREAD_SCHEDULER_READY }

  *The thread state with respect to the scheduler.*
- enum Thread_APIs { THREAD_API_RTEMS, THREAD_API_POSIX }
- enum Thread_Life_state {
  **THREAD_LIFE_PROTECTED** = 0x1, **THREAD_LIFE_RESTARTING** = 0x2, **THREAD_LIFE_TERMINATING** = 0x4, **THREAD_LIFE_CHANGE_DEFERRED** = 0x8,
  **THREAD_LIFE_DETACHED** = 0x10 }

  *Thread life states.*

## Functions

- void rtems_iterate_over_all_threads (rtems_per_thread_routine routine) RTEMS_DEPRECATED

  *Deprecated, use rtems_task_iterate() instead.*
- Objects_Control * _Thread_Allocate_unlimited (Objects_Information *information)

  *Return an inactive thread object or NULL.*
- static __inline__ bool _Thread_Dispatch_is_enabled (void)

  *Indicates if the executing thread is inside a thread dispatch critical section.*
- static __inline__ uint32_t _Thread_Dispatch_get_disable_level (void)

  *Gets thread dispatch disable level.*
- static __inline__ void _Thread_Dispatch_initialization (void)

  *Thread dispatch initialization.*
- void _Thread_Dispatch (void)

  *Performs a thread dispatch if necessary.*
- void _Thread_Dispatch_direct (Per_CPU_Control *cpu_self)

  *Directly do a thread dispatch.*
- void _Thread_Do_dispatch (Per_CPU_Control *cpu_self, ISR_Level level)

  *Performs a thread dispatch on the current processor.*
- static __inline__ Per_CPU_Control * _Thread_Dispatch_disable_with_CPU (Per_CPU_Control *cpu_self, const ISR_lock_Context *lock_context)

  *Disables thread dispatching inside a critical section (interrupts disabled) with the current processor.*
- static __inline__ Per_CPU_Control * _Thread_Dispatch_disable_critical (const ISR_lock_Context *lock_context)

  *Disables thread dispatching inside a critical section (interrupts disabled).*
- static __inline__ Per_CPU_Control * _Thread_Dispatch_disable (void)

  *Disables thread dispatching.*
- void _Thread_Dispatch_enable (Per_CPU_Control *cpu_self)

  *Enables thread dispatching.*
- static __inline__ void _Thread_Dispatch_unnest (Per_CPU_Control *cpu_self)

  *Unnests thread dispatching.*
- static __inline__ void _Thread_Dispatch_request (Per_CPU_Control *cpu_self, Per_CPU_Control *cpu_target)

  *Requests a thread dispatch on the target processor.*
- void _Thread_Iterate (Thread_Visitor visitor, void *arg)

  *Calls the visitor with all threads and the given argument until it is done.*
- void _Thread_Initialize_information (Thread_Information *information)

*Initializes the thread information.*

- void _Thread_Handler_initialization (void)

  *Initializes thread handler.*

- void _Thread_Create_idle (void)

  *Creates idle thread.*

- RTEMS_NO_RETURN void _Thread_Start_multitasking (void)

  *Starts thread multitasking.*

- bool _Thread_Initialize (Thread_Information *information, Thread_Control *the_thread, const Thread_Configuration *config)

  *Initializes thread.*

- bool _Thread_Start (Thread_Control *the_thread, const Thread_Entry_information *entry, ISR_lock_Context *lock_context)

  *Starts the specified thread.*

- RTEMS_NO_RETURN void _Thread_Restart_self (Thread_Control *executing, const Thread_Entry_information *entry, ISR_lock_Context *lock_context)

  *Restarts the currently executing thread.*

- bool _Thread_Restart_other (Thread_Control *the_thread, const Thread_Entry_information *entry, ISR_lock_Context *lock_context)

  *Restarts the thread.*

- void _Thread_Yield (Thread_Control *executing)

  *Yields the currently executing thread.*

- Thread_Life_state _Thread_Change_life (Thread_Life_state clear, Thread_Life_state set, Thread_Life_state ignore)

  *Changes the currently executing thread to a new state with the sets.*

- Thread_Life_state _Thread_Set_life_protection (Thread_Life_state state)

  *Set the thread to life protected.*

- void _Thread_Kill_zombies (void)

  *Kills all zombie threads in the system.*

- void _Thread_Exit (Thread_Control *executing, Thread_Life_state set, void *exit_value)

  *Exits the currently executing thread.*

- void _Thread_Join (Thread_Control *the_thread, States_Control waiting_for_join, Thread_Control *executing, Thread_queue_Context *queue_context)

  *Joins the currently executing thread with the given thread to wait for.*

- void _Thread_Cancel (Thread_Control *the_thread, Thread_Control *executing, void *exit_value)

  *Cancels the thread.*

- void _Thread_Close (Thread_Control *the_thread, Thread_Control *executing, Thread_Close_context *context)

  *Closes the thread.*

- static __inline__ bool _Thread_Is_ready (const Thread_Control *the_thread)

  *Checks if the thread is ready.*

- States_Control _Thread_Clear_state_locked (Thread_Control *the_thread, States_Control state)

  *Clears the specified thread state without locking the lock context.*

- States_Control _Thread_Clear_state (Thread_Control *the_thread, States_Control state)

  *Clears the specified thread state.*

- States_Control _Thread_Set_state_locked (Thread_Control *the_thread, States_Control state)

  *Sets the specified thread state without locking the lock context.*

- States_Control _Thread_Set_state (Thread_Control *the_thread, States_Control state)

  *Sets the specified thread state.*

- void _Thread_Load_environment (Thread_Control *the_thread)

  *Initializes enviroment for a thread.*

- void _Thread_Entry_adaptor_idle (Thread_Control *executing)

  *Calls the start kinds idle entry of the thread.*

- void _Thread_Entry_adaptor_numeric (Thread_Control *executing)

  *Calls the start kinds numeric entry of the thread.*
- void _Thread_Entry_adaptor_pointer (Thread_Control *executing)

  *Calls the start kinds pointer entry of the thread.*
- void _Thread_Handler (void)

  *Wrapper function for all threads.*
- static __inline__ void _Thread_State_acquire_critical (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Acquires the lock context in a critical section.*
- static __inline__ void _Thread_State_acquire (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Disables interrupts and acquires the lock_context.*
- static __inline__ Thread_Control * _Thread_State_acquire_for_executing (ISR_lock_Context *lock_context)

  *Disables interrupts and acquires the lock context for the currently executing thread.*
- static __inline__ void _Thread_State_release_critical (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Release the lock context in a critical section.*
- static __inline__ void _Thread_State_release (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Releases the lock context and enables interrupts.*
- void _Thread_Priority_perform_actions (Thread_Control *start_of_path, Thread_queue_Context *queue_context)

  *Checks if the thread is owner of the lock of the join queue.*
- void _Thread_Priority_add (Thread_Control *the_thread, Priority_Node *priority_node, Thread_queue_Context *queue_context)

  *Adds the specified thread priority node to the corresponding thread priority aggregation.*
- void _Thread_Priority_remove (Thread_Control *the_thread, Priority_Node *priority_node, Thread_queue_Context *queue_context)

  *Removes the specified thread priority node from the corresponding thread priority aggregation.*
- void _Thread_Priority_changed (Thread_Control *the_thread, Priority_Node *priority_node, bool prepend_it, Thread_queue_Context *queue_context)

  *Propagates a thread priority value change in the specified thread priority node to the corresponding thread priority aggregation.*
- static __inline__ void _Thread_Priority_change (Thread_Control *the_thread, Priority_Node *priority_node, Priority_Control new_priority, bool prepend_it, Thread_queue_Context *queue_context)

  *Changes the thread priority value of the specified thread priority node in the corresponding thread priority aggregation.*
- void _Thread_Priority_replace (Thread_Control *the_thread, Priority_Node *victim_node, Priority_Node *replacement_node)

  *Replaces the victim priority node with the replacement priority node in the corresponding thread priority aggregation.*
- void _Thread_Priority_update (Thread_queue_Context *queue_context)

  *Updates the priority of all threads in the set.*
- void _Thread_Priority_and_sticky_update (Thread_Control *the_thread, int sticky_level_change)

  *Updates the priority of the thread and changes it sticky level.*
- static __inline__ bool _Thread_Priority_less_than (Priority_Control left, Priority_Control right)

  *Checks if the left thread priority is less than the right thread priority in the intuitive sense of priority.*
- static __inline__ Priority_Control _Thread_Priority_highest (Priority_Control left, Priority_Control right)

  *Returns the highest priority of the left and right thread priorities in the intuitive sense of priority.*
- static __inline__ Objects_Information * _Thread_Get_objects_information (Objects_Id id)

  *Gets object information for the object id.*
- Thread_Control * _Thread_Get (Objects_Id id, ISR_lock_Context *lock_context)

  *Gets a thread by its identifier.*
- static __inline__ Per_CPU_Control * _Thread_Get_CPU (const Thread_Control *thread)

*Gets the cpu of the thread's scheduler.*

- static __inline__ void _Thread_Set_CPU (Thread_Control *thread, Per_CPU_Control *cpu)

  *Sets the cpu of the thread's scheduler.*
- static __inline__ bool _Thread_Is_executing (const Thread_Control *the_thread)

  *Checks if the thread is the currently executing thread.*
- static __inline__ bool _Thread_Is_executing_on_a_processor (const Thread_Control *the_thread)

  *Checks if the thread executes currently on some processor in the system.*
- static __inline__ bool _Thread_Is_heir (const Thread_Control *the_thread)

  *Checks if the thread is the heir.*
- static __inline__ void _Thread_Unblock (Thread_Control *the_thread)

  *Unblocks the thread.*
- static __inline__ void _Thread_Save_fp (Thread_Control *executing)

  *Checks if the floating point context of the thread is currently loaded in the floating point unit.*
- static __inline__ void _Thread_Restore_fp (Thread_Control *executing)

  *Restores the executing thread's floating point area.*
- static __inline__ bool _Thread_Is_context_switch_necessary (void)

  *Deallocates the currently loaded floating point context.*
- static __inline__ uint32_t _Thread_Get_maximum_internal_threads (void)

  *Gets the maximum number of internal threads.*
- static __inline__ Thread_Control * _Thread_Internal_allocate (void)

  *Allocates an internal thread and returns it.*
- static __inline__ Thread_Control * _Thread_Get_heir_and_make_it_executing (Per_CPU_Control *cpu_self)

  *Gets the heir of the processor and makes it executing.*
- static __inline__ void _Thread_Update_CPU_time_used (Thread_Control *the_thread, Per_CPU_Control *cpu)

  *Updates the cpu time used of the thread.*
- static __inline__ void _Thread_Dispatch_update_heir (Per_CPU_Control *cpu_self, Per_CPU_Control *cpu_for_heir, Thread_Control *heir)

  *Updates the used cpu time for the heir and dispatches a new heir.*
- void _Thread_Get_CPU_time_used (Thread_Control *the_thread, Timestamp_Control *cpu_time_used)

  *Gets the used cpu time of the thread and stores it in the given Timestamp_Control.*
- static __inline__ void _Thread_Action_control_initialize (Thread_Action_control *action_control)

  *Initializes the control chain of the action control.*
- static __inline__ void _Thread_Action_initialize (Thread_Action *action)

  *Initializes the Thread action.*
- static __inline__ void _Thread_Add_post_switch_action (Thread_Control *the_thread, Thread_Action *action, Thread_Action_handler handler)

  *Adds a post switch action to the thread with the given handler.*
- static __inline__ bool _Thread_Is_life_restarting (Thread_Life_state life_state)

  *Checks if the thread life state is restarting.*
- static __inline__ bool _Thread_Is_life_terminating (Thread_Life_state life_state)

  *Checks if the thread life state is terminating.*
- static __inline__ bool _Thread_Is_life_change_allowed (Thread_Life_state life_state)

  *Checks if the thread life state allos life change.*
- static __inline__ bool _Thread_Is_life_changing (Thread_Life_state life_state)

  *Checks if the thread life state is life changing.*
- static __inline__ bool _Thread_Is_joinable (const Thread_Control *the_thread)

  *Checks if the thread is joinable.*
- static __inline__ void _Thread_Resource_count_increment (Thread_Control *the_thread)

  *Increments the thread's resource count.*
- static __inline__ void _Thread_Resource_count_decrement (Thread_Control *the_thread)

*Decrements the thread's resource count.*

- static __inline__ void _Thread_Scheduler_cancel_need_for_help (Thread_Control *the_thread, Per_CPU_Control *cpu)

  *Cancels the thread's need for help.*

- static __inline__ const Scheduler_Control * _Thread_Scheduler_get_home (const Thread_Control *the_thread)

  *Gets the home scheduler of the thread.*

- static __inline__ Scheduler_Node * _Thread_Scheduler_get_home_node (const Thread_Control *the_thread)

  *Gets the scheduler's home node.*

- static __inline__ Scheduler_Node * _Thread_Scheduler_get_node_by_index (const Thread_Control *the_thread, size_t scheduler_index)

  *Gets the thread's scheduler node by index.*

- static __inline__ void _Thread_Scheduler_acquire_critical (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Acquires the lock context in a critical section.*

- static __inline__ void _Thread_Scheduler_release_critical (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Releases the lock context in a critical section.*

- void _Thread_Scheduler_process_requests (Thread_Control *the_thread)

  *Process the thread's scheduler requests.*

- static __inline__ void _Thread_Scheduler_add_request (Thread_Control *the_thread, Scheduler_Node *scheduler_node, Scheduler_Node_request request)

  *Add a scheduler request to the thread.*

- static __inline__ void _Thread_Scheduler_add_wait_node (Thread_Control *the_thread, Scheduler_Node *scheduler_node)

  *Adds a wait node to the thread and adds a corresponding request to the thread.*

- static __inline__ void _Thread_Scheduler_remove_wait_node (Thread_Control *the_thread, Scheduler_Node *scheduler_node)

  *Remove a wait node from the thread and add a corresponding request to it.*

- static __inline__ Priority_Control _Thread_Get_priority (const Thread_Control *the_thread)

  *Returns the priority of the thread.*

- static __inline__ Priority_Control _Thread_Get_unmapped_priority (const Thread_Control *the_thread)

  *Returns the unmapped priority of the thread.*

- static __inline__ Priority_Control _Thread_Get_unmapped_real_priority (const Thread_Control *the_thread)

  *Returns the unmapped real priority of the thread.*

- static __inline__ void _Thread_Wait_acquire_default_critical (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Acquires the thread wait default lock inside a critical section (interrupts disabled).*

- static __inline__ Thread_Control * _Thread_Wait_acquire_default_for_executing (ISR_lock_Context *lock_context)

  *Acquires the thread wait default lock and returns the executing thread.*

- static __inline__ void _Thread_Wait_acquire_default (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Acquires the thread wait default lock and disables interrupts.*

- static __inline__ void _Thread_Wait_release_default_critical (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Releases the thread wait default lock inside a critical section (interrupts disabled).*

- static __inline__ void _Thread_Wait_release_default (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Releases the thread wait default lock and restores the previous interrupt status.*

- static __inline__ void _Thread_Wait_remove_request_locked (Thread_Control *the_thread, Thread_queue_Lock_context *queue_lock_context)

*Removes the first pending wait lock request.*

- static __inline__ void _Thread_Wait_acquire_queue_critical (Thread_queue_Queue *queue, Thread_queue_Lock_context *queue_lock_context)

  *Acquires the wait queue inside a critical section.*
- static __inline__ void _Thread_Wait_release_queue_critical (Thread_queue_Queue *queue, Thread_queue_Lock_context *queue_lock_context)

  *Releases the wait queue inside a critical section.*
- static __inline__ void _Thread_Wait_acquire_critical (Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Acquires the thread wait lock inside a critical section (interrupts disabled).*
- static __inline__ void _Thread_Wait_acquire (Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Acquires the thread wait default lock and disables interrupts.*
- static __inline__ void _Thread_Wait_release_critical (Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Releases the thread wait lock inside a critical section (interrupts disabled).*
- static __inline__ void _Thread_Wait_release (Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Releases the thread wait lock and restores the previous interrupt status.*
- static __inline__ void _Thread_Wait_claim (Thread_Control *the_thread, Thread_queue_Queue *queue)

  *Claims the thread wait queue.*
- static __inline__ void _Thread_Wait_claim_finalize (Thread_Control *the_thread, const Thread_queue_Operations *operations)

  *Finalizes the thread wait queue claim via registration of the corresponding thread queue operations.*
- static __inline__ void _Thread_Wait_remove_request (Thread_Control *the_thread, Thread_queue_Lock_context *queue_lock_context)

  *Removes a thread wait lock request.*
- static __inline__ void _Thread_Wait_restore_default (Thread_Control *the_thread)

  *Restores the default thread wait queue and operations.*
- static __inline__ void _Thread_Wait_tranquilize (Thread_Control *the_thread)

  *Tranquilizes the thread after a wait on a thread queue.*
- static __inline__ void _Thread_Wait_cancel (Thread_Control *the_thread, Thread_queue_Context *queue↩ _context)

  *Cancels a thread wait on a thread queue.*
- static __inline__ void _Thread_Wait_flags_set (Thread_Control *the_thread, Thread_Wait_flags flags)

  *Sets the thread's wait flags.*
- static __inline__ Thread_Wait_flags _Thread_Wait_flags_get (const Thread_Control *the_thread)

  *Gets the thread's wait flags according to the ATOMIC_ORDER_RELAXED.*
- static __inline__ Thread_Wait_flags _Thread_Wait_flags_get_acquire (const Thread_Control *the_thread)

  *Gets the thread's wait flags according to the ATOMIC_ORDER_ACQUIRE.*
- static __inline__ bool _Thread_Wait_flags_try_change_release (Thread_Control *the_thread, Thread_Wait_flags expected_flags, Thread_Wait_flags desired_flags)

  *Tries to change the thread wait flags with release semantics in case of success.*
- static __inline__ bool _Thread_Wait_flags_try_change_acquire (Thread_Control *the_thread, Thread_Wait_flags expected_flags, Thread_Wait_flags desired_flags)

  *Tries to change the thread wait flags with acquire semantics.*
- Objects_Id _Thread_Wait_get_id (const Thread_Control *the_thread)

  *Returns the object identifier of the object containing the current thread wait queue.*
- static __inline__ Status_Control _Thread_Wait_get_status (const Thread_Control *the_thread)

  *Get the status of the wait return code of the thread.*
- void _Thread_Continue (Thread_Control *the_thread, Status_Control status)

  *Cancels a blocking operation so that the thread can continue its execution.*

- void _Thread_Timeout (Watchdog_Control *the_watchdog)

  *General purpose thread wait timeout.*
- static __inline__ void _Thread_Timer_initialize (Thread_Timer_information *timer, Per_CPU_Control *cpu)

  *Initializes the thread timer.*
- static __inline__ void _Thread_Add_timeout_ticks (Thread_Control *the_thread, Per_CPU_Control *cpu, Watchdog_Interval ticks)

  *Adds timeout ticks to the thread.*
- static __inline__ void _Thread_Timer_insert_realtime (Thread_Control *the_thread, Per_CPU_Control *cpu, Watchdog_Service_routine_entry routine, uint64_t expire)

  *Inserts the cpu's watchdog realtime into the thread's timer.*
- static __inline__ void _Thread_Timer_remove (Thread_Control *the_thread)

  *Remove the watchdog timer from the thread.*
- static __inline__ void _Thread_Remove_timer_and_unblock (Thread_Control *the_thread, Thread_queue_Queue *queue)

  *Remove the watchdog timer from the thread and unblock if necessary.*
- Status_Control _Thread_Set_name (Thread_Control *the_thread, const char *name)

  *Sets the name of the thread.*
- size_t _Thread_Get_name (const Thread_Control *the_thread, char *buffer, size_t buffer_size)

  *Gets the name of the thread.*
- void _Thread_Do_unpin (Thread_Control *executing, Per_CPU_Control *cpu_self)

  *Unpins the thread.*
- static __inline__ void _Thread_Pin (Thread_Control *executing)

  *Pin the executing thread.*
- static __inline__ void _Thread_Unpin (Thread_Control *executing, Per_CPU_Control *cpu_self)

  *Unpins the thread.*

## Variables

- const Thread_Control_add_on _Thread_Control_add_ons [ ]

  *Thread control add-ons.*
- const size_t _Thread_Control_add_on_count

  *Thread control add-on count.*
- const size_t _Thread_Initial_thread_count

  *Count of configured threads.*
- const size_t _Thread_Maximum_name_size

  *Maximum size of a thread name in characters (including the terminating '\0' character).*
- const size_t _Thread_Maximum_TLS_size

  *If this constant is greater than zero, then it defines the maximum thread-local storage size, otherwise the thread-local storage size is defined by the linker depending on the thread-local storage objects used by the application in the statically-linked executable.*
- const size_t _Thread_queue_Heads_size

  *Size of the thread queue heads of a particular application.*
- Thread_Information _Thread_Information

  *The internal thread objects information.*
- char _Thread_Idle_stacks [ ]

  *The idle thread stacks.*
- const size_t _Thread_Idle_stack_size

  *The idle thread stack size in bytes.*
- const Thread_Idle_body _Thread_Idle_body

  *The idle thread body.*
- void * rtems_ada_self
- Objects_Id _Thread_Global_constructor

  *Object identifier of the global constructor thread.*

## 8.170.1 Detailed Description

Thread Handler.

This handler encapsulates functionality related to the management of threads. This includes the creation, deletion, and scheduling of threads.

The following variables are maintained as part of the per cpu data structure.

- Idle thread pointer
- Executing thread pointer
- Heir thread pointer

## 8.170.2 Macro Definition Documentation

### 8.170.2.1 RTEMS_SCORE_ROBUST_THREAD_DISPATCH

```
#define RTEMS_SCORE_ROBUST_THREAD_DISPATCH
```

Enables a robust thread dispatch.

On each change of the thread dispatch disable level from one to zero the interrupt status is checked. In case interrupts are disabled and SMP is enabled or the CPU port needs it, then the system terminates with the fatal internal error INTERNAL_ERROR_BAD_THREAD_DISPATCH_ENVIRONMENT.

Definition at line 44 of file threaddispatch.h.

### 8.170.2.2 THREAD_API_FIRST

```
#define THREAD_API_FIRST THREAD_API_RTEMS
```

This macro defines the first API which has threads.

Definition at line 590 of file thread.h.

### 8.170.2.3 THREAD_API_LAST

```
#define THREAD_API_LAST THREAD_API_POSIX
```

This macro defines the last API which has threads.

Definition at line 593 of file thread.h.

#### 8.170.2.4 THREAD_DEFAULT_MAXIMUM_NAME_SIZE

```
#define THREAD_DEFAULT_MAXIMUM_NAME_SIZE 16
```

The default maximum size of a thread name in characters (including the terminating '\0' character).

This is the default value for the application configuration option CONFIGURE_MAXIMUM_THREAD_NAME_SIZE.

Definition at line 946 of file thread.h.

#### 8.170.2.5 THREAD_INFORMATION_DEFINE

```
#define THREAD_INFORMATION_DEFINE(
                name,
                api,
                cls,
                max )
```

**Value:**

```
static Objects_Control * \
name##_Local_table[ _Objects_Maximum_per_allocation( max ) ]; \
static Thread_Configured_control \
name##_Objects[ _Objects_Maximum_per_allocation( max ) ]; \
static Thread_queue_Configured_heads \
name##_Heads[ _Objects_Maximum_per_allocation( max ) ]; \
Thread_Information name##_Information = { \
  { \
    _Objects_Build_id( api, cls, 1, _Objects_Maximum_per_allocation( max ) ), \
    name##_Local_table, \
    _Objects_Is_unlimited( max ) ? \
      _Thread_Allocate_unlimited : _Objects_Allocate_static, \
    _Objects_Is_unlimited( max ) ? \
      _Objects_Free_unlimited : _Objects_Free_static, \
    0, \
    _Objects_Is_unlimited( max ) ? _Objects_Maximum_per_allocation( max ) : 0, \
    sizeof( Thread_Configured_control ), \
    OBJECTS_NO_STRING_NAME, \
    CHAIN_INITIALIZER_EMPTY( name##_Information.Objects.Inactive ), \
    NULL, \
    NULL, \
    &name##_Objects[ 0 ].Control.Object \
    OBJECTS_INFORMATION_MP( name##_Information.Objects, NULL ) \
  }, { \
    &name##_Heads[ 0 ] \
  } \
}
```

Definition at line 1060 of file thread.h.

#### 8.170.2.6 THREAD_INFORMATION_DEFINE_ZERO

```
#define THREAD_INFORMATION_DEFINE_ZERO(
                name,
                api,
                cls )
```

**Value:**

```
Thread_Information name##_Information = { \
  { \
    _Objects_Build_id( api, cls, 1, 0 ), \
    NULL, \
```

```
    _Objects_Allocate_none, \
    NULL, \
    0, \
    0, \
    0, \
    0, \
    CHAIN_INITIALIZER_EMPTY( name##_Information.Objects.Inactive ), \
    NULL, \
    NULL, \
    NULL \
    OBJECTS_INFORMATION_MP( name##_Information.Objects, NULL ), \
  }, { \
    NULL \
  } \
}
```

Definition at line 1031 of file thread.h.

#### 8.170.2.7 THREAD_OF_SCHEDULER_HELP_NODE

```
#define THREAD_OF_SCHEDULER_HELP_NODE(
              node )  RTEMS_CONTAINER_OF( node, Thread_Control, Scheduler.Help_node )
```

The following points to the thread whose floating point context is currently loaded.

Definition at line 76 of file threadimpl.h.

#### 8.170.2.8 THREAD_WAIT_STATE_INTEND_TO_BLOCK

```
#define THREAD_WAIT_STATE_INTEND_TO_BLOCK 0x1U
```

Indicates that the thread begins with the blocking operation.

A blocking operation consists of an optional watchdog initialization and the setting of the appropriate thread blocking state with the corresponding scheduler block operation.

Definition at line 2136 of file threadimpl.h.

#### 8.170.2.9 THREAD_WAIT_STATE_READY_AGAIN

```
#define THREAD_WAIT_STATE_READY_AGAIN 0x4U
```

Indicates that a condition to end the thread wait occurred.

This could be a timeout, a signal, an event or a resource availability.

Definition at line 2148 of file threadimpl.h.

### 8.170.3 Typedef Documentation

#### 8.170.3.1 Thread_Action_handler

```
typedef void( * Thread_Action_handler) (Thread_Control *the_thread, Thread_Action *action,
ISR_lock_Context *lock_context)
```

Thread action handler.

The thread action handler will be called with interrupts disabled and a corresponding lock acquired, e.g. _Thread_State_acquire(). The handler must release the corresponding lock, e.g. _Thread_State_release(). So, the corresponding lock may be used to protect private data used by the particular action.

Since the action is passed to the handler additional data may be accessed via RTEMS_CONTAINER_OF().

**Parameters**

| | | |
|---|---|---|
| in | *the_thread* | The thread performing the action. |
| in | *action* | The thread action. |
| in | *lock_context* | The lock context to use for the lock release. |

Definition at line 613 of file thread.h.

#### 8.170.3.2 Thread_Configured_control

```
typedef struct Thread_Configured_control Thread_Configured_control
```

The configured thread control block.

This type is defined in <rtems/confdefs.h> and depends on the application configuration.

Definition at line 964 of file thread.h.

#### 8.170.3.3 Thread_CPU_budget_algorithm_callout

```
typedef void(* Thread_CPU_budget_algorithm_callout) (Thread_Control *)
```

This defines thes the entry point for the thread specific timeslice budget management algorithm.

Definition at line 172 of file thread.h.

#### 8.170.3.4 Thread_Entry_numeric_type

```
typedef CPU_Uint32ptr Thread_Entry_numeric_type
```

Type of the numeric argument of a thread entry function with at least one numeric argument.

This numeric argument type designates an unsigned integer type with the property that any valid pointer to void can be converted to this type and then converted back to a pointer to void. The result will compare equal to the original pointer.

Definition at line 101 of file thread.h.

#### 8.170.3.5 Thread_queue_Configured_heads

```
typedef struct Thread_queue_Configured_heads Thread_queue_Configured_heads
```

The configured thread queue heads.

In SMP configurations, this type is defined in <rtems/confdefs.h> and depends on the application configuration.

Definition at line 964 of file thread.h.

#### 8.170.3.6 Thread_Wait_flags

```
typedef unsigned int Thread_Wait_flags
```

This type is able to contain several flags used to control the wait class and state of a thread.

The mutually exclusive wait class flags are

- THREAD_WAIT_CLASS_EVENT,
- THREAD_WAIT_CLASS_SYSTEM_EVENT, and
- THREAD_WAIT_CLASS_OBJECT.

The mutually exclusive wait state flags are

- THREAD_WAIT_STATE_INTEND_TO_BLOCK,
- THREAD_WAIT_STATE_BLOCKED, and
- THREAD_WAIT_STATE_READY_AGAIN.

Definition at line 383 of file thread.h.

### 8.170.4 Enumeration Type Documentation

#### 8.170.4.1 Thread_APIs

```
enum Thread_APIs
```

The following record defines the control block used to manage each thread.

**Note**

It is critical that proxies and threads have identical memory images for the shared part.

**Enumerator**

| | |
|---|---|
| THREAD_API_RTEMS | This value is for the Classic RTEMS API. |
| THREAD_API_POSIX | This value is for the POSIX API. |

Definition at line 582 of file thread.h.

#### 8.170.4.2 Thread_CPU_budget_algorithms

`enum Thread_CPU_budget_algorithms`

The following lists the algorithms used to manage the thread cpu budget.

Reset Timeslice: At each context switch, reset the time quantum. Exhaust Timeslice: Only reset the quantum once it is consumed. Callout: Execute routine when budget is consumed.

Definition at line 158 of file thread.h.

#### 8.170.4.3 Thread_Life_state

`enum Thread_Life_state`

Thread life states.

The thread life states are orthogonal to the thread states used for synchronization primitives and blocking operations. They reflect the state changes triggered with thread restart and delete requests.

The individual state values must be a power of two to allow use of bit operations to manipulate and evaluate the thread life state.

Definition at line 672 of file thread.h.

#### 8.170.4.4 Thread_Scheduler_state

`enum Thread_Scheduler_state`

The thread state with respect to the scheduler.

**Enumerator**

| | |
|---|---|
| THREAD_SCHEDULER_BLOCKED | This thread is blocked with respect to the scheduler. This thread uses no scheduler nodes. |
| THREAD_SCHEDULER_SCHEDULED | This thread is scheduled with respect to the scheduler. This thread executes using one of its scheduler nodes. This could be its own scheduler node or in case it owns resources taking part in the scheduler helping protocol a scheduler node of another thread. |
| THREAD_SCHEDULER_READY | This thread is ready with respect to the scheduler. None of the scheduler nodes of this thread is scheduled. |

Definition at line 214 of file thread.h.

## 8.170.5 Function Documentation

### 8.170.5.1 _Thread_Action_control_initialize()

```
static __inline__ void _Thread_Action_control_initialize (
            Thread_Action_control * action_control )  [static]
```

Initializes the control chain of the action control.

**Parameters**

| | | |
|---|---|---|
| out | *action_control* | The action control to initialize. |

Definition at line 1193 of file threadimpl.h.

### 8.170.5.2 _Thread_Action_initialize()

```
static __inline__ void _Thread_Action_initialize (
            Thread_Action * action )  [static]
```

Initializes the Thread action.

**Parameters**

| | | |
|---|---|---|
| out | *action* | The Thread_Action to initialize. |

Definition at line 1205 of file threadimpl.h.

### 8.170.5.3 _Thread_Add_post_switch_action()

```
static __inline__ void _Thread_Add_post_switch_action (
            Thread_Control * the_thread,
            Thread_Action * action,
            Thread_Action_handler handler )  [static]
```

Adds a post switch action to the thread with the given handler.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |
| in,out | *action* | The action to add. |
| | *handler* | The handler for the action. |

Definition at line 1219 of file threadimpl.h.

#### 8.170.5.4 _Thread_Add_timeout_ticks()

```
static __inline__ void _Thread_Add_timeout_ticks (
             Thread_Control * the_thread,
             Per_CPU_Control * cpu,
             Watchdog_Interval ticks )  [static]
```

Adds timeout ticks to the thread.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread to add the timeout ticks to. |
| | *cpu* | The cpu for the operation. |
| | *ticks* | The ticks to add to the timeout ticks. |

Definition at line 2390 of file threadimpl.h.

#### 8.170.5.5 _Thread_Allocate_unlimited()

```
Objects_Control* _Thread_Allocate_unlimited (
             Objects_Information * information )
```

Return an inactive thread object or NULL.

**Return values**

| | |
|---|---|
| *NULL* | No inactive object is available. |
| *object* | An inactive object. |

#### 8.170.5.6 _Thread_Cancel()

```
void _Thread_Cancel (
             Thread_Control * the_thread,
             Thread_Control * executing,
             void * exit_value )
```

Cancels the thread.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread to cancel. |
| | *executing* | The currently executing thread. |
| | *exit_value* | The exit value for the thread. |

Definition at line 459 of file threadrestart.c.

#### 8.170.5.7 _Thread_Change_life()

```
Thread_Life_state _Thread_Change_life (
            Thread_Life_state clear,
            Thread_Life_state set,
            Thread_Life_state ignore )
```

Changes the currently executing thread to a new state with the sets.

**Parameters**

| | |
|---|---|
| *clear* | States to clear. |
| *set* | States to set. |
| *ignore* | States to ignore. |

**Returns**

The previous state the thread was in.

Definition at line 658 of file threadrestart.c.

#### 8.170.5.8 _Thread_Clear_state()

```
States_Control _Thread_Clear_state (
            Thread_Control * the_thread,
            States_Control state )
```

Clears the specified thread state.

In the case the previous state is a non-ready state and the next state is the ready state, then the thread is unblocked by the scheduler.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |
| | *state* | The state to clear. It must not be zero. |

**Returns**

The previous state.

Definition at line 51 of file threadclearstate.c.

### 8.170.5.9 _Thread_Clear_state_locked()

```
States_Control _Thread_Clear_state_locked (
            Thread_Control * the_thread,
            States_Control state )
```

Clears the specified thread state without locking the lock context.

In the case the previous state is a non-ready state and the next state is the ready state, then the thread is unblocked by the scheduler.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |
| | *state* | The state to clear. It must not be zero. |

**Returns**

The thread's previous state.

Definition at line 25 of file threadclearstate.c.

### 8.170.5.10 _Thread_Close()

```
void _Thread_Close (
            Thread_Control * the_thread,
            Thread_Control * executing,
            Thread_Close_context * context )
```

Closes the thread.

Closes the thread object and starts the thread termination sequence. In case the executing thread is not terminated, then this function waits until the terminating thread reached the zombie state.

**Parameters**

| | | |
|---|---|---|
| | *the_thread* | The thread to close. |
| | *executing* | The currently executing thread. |
| in,out | *context* | The thread close context. |

Definition at line 518 of file threadrestart.c.

### 8.170.5.11 _Thread_Continue()

```
void _Thread_Continue (
            Thread_Control * the_thread,
            Status_Control status )
```

Cancels a blocking operation so that the thread can continue its execution.

In case this function actually cancelled the blocking operation, then the thread wait return code is set to the specified status.

A specialization of this function is _Thread_Timeout().

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |
| | *status* | The thread wait status. |

Definition at line 25 of file threadtimeout.c.

#### 8.170.5.12 _Thread_Create_idle()

```
void _Thread_Create_idle (
             void  )
```

Creates idle thread.

This routine creates the idle thread.

**Warning**

No thread should be created before this one.

Definition at line 93 of file threadcreateidle.c.

#### 8.170.5.13 _Thread_Dispatch()

```
void _Thread_Dispatch (
             void  )
```

Performs a thread dispatch if necessary.

This routine is responsible for transferring control of the processor from the executing thread to the heir thread. Once the heir is running an attempt is made to run the pending post-switch thread actions.

As part of this process, it is responsible for the following actions

- update timing information of the executing thread,
- save the context of the executing thread,
- invokation of the thread switch user extensions,
- restore the context of the heir thread, and
- run of pending post-switch thread actions of the resulting executing thread.

On entry the thread dispatch level must be equal to zero.

Definition at line 331 of file threaddispatch.c.

### 8.170.5.14 _Thread_Dispatch_direct()

```
void _Thread_Dispatch_direct (
            Per_CPU_Control * cpu_self )
```

Directly do a thread dispatch.

Must be called with a thread dispatch disable level of one, otherwise the INTERNAL_ERROR_BAD_THREAD_D↩ ISPATCH_DISABLE_LEVEL will occur. This function is useful for operations which synchronously block, e.g. self restart, self deletion, yield, sleep.

**Parameters**

| | |
|---|---|
| *cpu_self* | The current processor. |

**See also**

_Thread_Dispatch().

Definition at line 350 of file threaddispatch.c.

### 8.170.5.15 _Thread_Dispatch_disable()

```
static __inline__ Per_CPU_Control* _Thread_Dispatch_disable (
            void  )  [static]
```

Disables thread dispatching.

**Returns**

The current processor.

Definition at line 191 of file threaddispatch.h.

### 8.170.5.16 _Thread_Dispatch_disable_critical()

```
static __inline__ Per_CPU_Control* _Thread_Dispatch_disable_critical (
            const ISR_lock_Context * lock_context )  [static]
```

Disables thread dispatching inside a critical section (interrupts disabled).

**Parameters**

| | |
|---|---|
| *lock_context* | The lock context of the corresponding _ISR_lock_ISR_disable() that started the critical section. |

**Returns**

The current processor.

Definition at line 179 of file threaddispatch.h.

#### 8.170.5.17 _Thread_Dispatch_disable_with_CPU()

```
static __inline__ Per_CPU_Control* _Thread_Dispatch_disable_with_CPU (
            Per_CPU_Control * cpu_self,
            const ISR_lock_Context * lock_context )  [static]
```

Disables thread dispatching inside a critical section (interrupts disabled) with the current processor.

**Parameters**

| | |
|---|---|
| *cpu_self* | The current processor. |
| *lock_context* | The lock context of the corresponding _ISR_lock_ISR_disable() that started the critical section. |

**Returns**

The current processor.

Definition at line 152 of file threaddispatch.h.

#### 8.170.5.18 _Thread_Dispatch_enable()

```
void _Thread_Dispatch_enable (
            Per_CPU_Control * cpu_self )
```

Enables thread dispatching.

May perform a thread dispatch if necessary as a side-effect.

**Parameters**

| | | |
|---|---|---|
| in,out | *cpu_self* | The current processor. |

Definition at line 362 of file threaddispatch.c.

#### 8.170.5.19 _Thread_Dispatch_get_disable_level()

```
static __inline__ uint32_t _Thread_Dispatch_get_disable_level (
            void  )  [static]
```

Gets thread dispatch disable level.

**Returns**

The value of the thread dispatch level.

Definition at line 79 of file threaddispatch.h.

### 8.170.5.20 _Thread_Dispatch_initialization()

```
static __inline__ void _Thread_Dispatch_initialization (
            void  )  [static]
```

Thread dispatch initialization.

This routine initializes the thread dispatching subsystem.

Definition at line 89 of file threaddispatch.h.

### 8.170.5.21 _Thread_Dispatch_is_enabled()

```
static __inline__ bool _Thread_Dispatch_is_enabled (
            void  )  [static]
```

Indicates if the executing thread is inside a thread dispatch critical section.

**Return values**

| | |
|---|---|
| *true* | Thread dispatching is enabled. |
| *false* | The executing thread is inside a thread dispatch critical section and dispatching is not allowed. |

Definition at line 55 of file threaddispatch.h.

### 8.170.5.22 _Thread_Dispatch_request()

```
static __inline__ void _Thread_Dispatch_request (
            Per_CPU_Control * cpu_self,
            Per_CPU_Control * cpu_target )  [static]
```

Requests a thread dispatch on the target processor.

**Parameters**

| | | |
|---|---|---|
| in,out | *cpu_self* | The current processor. |
| in,out | *cpu_target* | The target processor to request a thread dispatch. |

Definition at line 235 of file threaddispatch.h.

#### 8.170.5.23 _Thread_Dispatch_unnest()

```
static __inline__ void _Thread_Dispatch_unnest (
            Per_CPU_Control * cpu_self )  [static]
```

Unnests thread dispatching.

**Parameters**

| in,out | *cpu_self* | The current processor. |
|---|---|---|

Definition at line 223 of file threaddispatch.h.

#### 8.170.5.24 _Thread_Dispatch_update_heir()

```
static __inline__ void _Thread_Dispatch_update_heir (
            Per_CPU_Control * cpu_self,
            Per_CPU_Control * cpu_for_heir,
            Thread_Control * heir )  [static]
```

Updates the used cpu time for the heir and dispatches a new heir.

**Parameters**

| in,out | *cpu_self* | The current processor. |
|---|---|---|
| in,out | *cpu_for_heir* | The processor to do a dispatch on. |
| | *heir* | The new heir for *cpu_for_heir*. |

Definition at line 1162 of file threadimpl.h.

#### 8.170.5.25 _Thread_Do_dispatch()

```
void _Thread_Do_dispatch (
            Per_CPU_Control * cpu_self,
            ISR_Level level )
```

Performs a thread dispatch on the current processor.

On entry the thread dispatch disable level must be equal to one and interrupts must be disabled.

This function assumes that a thread dispatch is necessary.

**Parameters**

| | |
|---|---|
| *cpu_self* | The current processor. |
| *level* | The previous interrupt level. |

**See also**

_Thread_Dispatch().

Definition at line 259 of file threaddispatch.c.

### 8.170.5.26 _Thread_Do_unpin()

```
void _Thread_Do_unpin (
              Thread_Control * executing,
              Per_CPU_Control * cpu_self )
```

Unpins the thread.

**Parameters**

| | |
|---|---|
| *executing* | The currently executing thread. |
| *cpu_self* | The cpu for the operation. |

Definition at line 15 of file threadunpin.c.

### 8.170.5.27 _Thread_Entry_adaptor_idle()

```
void _Thread_Entry_adaptor_idle (
              Thread_Control * executing )
```

Calls the start kinds idle entry of the thread.

**Parameters**

| | |
|---|---|
| *executing* | The currently executing thread. |

Definition at line 21 of file threadentryadaptoridle.c.

### 8.170.5.28 _Thread_Entry_adaptor_numeric()

```
void _Thread_Entry_adaptor_numeric (
              Thread_Control * executing )
```

Calls the start kinds numeric entry of the thread.

**Parameters**

| | |
|---|---|
| *executing* | The currently executing thread. |

Definition at line 21 of file threadentryadaptornumeric.c.

#### 8.170.5.29 _Thread_Entry_adaptor_pointer()

```
void _Thread_Entry_adaptor_pointer (
              Thread_Control * executing )
```

Calls the start kinds pointer entry of the thread.

Stores the return value in the Wait.return_argument of the thread.

**Parameters**

| | |
|---|---|
| *executing* | The currently executing thread. |

#### 8.170.5.30 _Thread_Exit()

```
void _Thread_Exit (
              Thread_Control * executing,
              Thread_Life_state set,
              void * exit_value )
```

Exits the currently executing thread.

**Parameters**

| | | |
|---|---|---|
| in,out | *executing* | The currently executing thread. |
| | *set* | The states to set. |
| out | *exit_value* | Contains the exit value of the thread. |

Definition at line 541 of file threadrestart.c.

#### 8.170.5.31 _Thread_Get()

```
Thread_Control* _Thread_Get (
              Objects_Id id,
              ISR_lock_Context * lock_context )
```

Gets a thread by its identifier.

**See also**

_Objects_Get().

**Parameters**

| | |
|---|---|
| *id* | The id of the thread. |
| *lock_context* | The lock context. |

Definition at line 24 of file threadget.c.

### 8.170.5.32 _Thread_Get_CPU()

```
static __inline__ Per_CPU_Control* _Thread_Get_CPU (
              const Thread_Control * thread )  [static]
```

Gets the cpu of the thread's scheduler.

**Parameters**

| | |
|---|---|
| *thread* | The thread. |

**Returns**

The cpu of the thread's scheduler.

Definition at line 867 of file threadimpl.h.

### 8.170.5.33 _Thread_Get_CPU_time_used()

```
void _Thread_Get_CPU_time_used (
              Thread_Control * the_thread,
              Timestamp_Control * cpu_time_used )
```

Gets the used cpu time of the thread and stores it in the given Timestamp_Control.

**Parameters**

| | | |
|---|---|---|
| | *the_thread* | The thread to get the used cpu time of. |
| out | *cpu_time_used* | Stores the used cpu time of *the_thread*. |

Definition at line 31 of file threadgetcputimeused.c.

#### 8.170.5.34 _Thread_Get_heir_and_make_it_executing()

```
static __inline__ Thread_Control* _Thread_Get_heir_and_make_it_executing (
            Per_CPU_Control * cpu_self )  [static]
```

Gets the heir of the processor and makes it executing.

Must be called with interrupts disabled. The thread dispatch necessary indicator is cleared as a side-effect.

**Parameters**

| | | |
|---|---|---|
| in,out | *cpu_self* | The processor to get the heir of. |

**Returns**

The heir thread.

**See also**

_Thread_Dispatch(), _Thread_Start_multitasking() and _Thread_Dispatch_update_heir().

Definition at line 1120 of file threadimpl.h.

#### 8.170.5.35 _Thread_Get_maximum_internal_threads()

```
static __inline__ uint32_t _Thread_Get_maximum_internal_threads (
            void  )  [static]
```

Gets the maximum number of internal threads.

**Returns**

The maximum number of internal threads.

Definition at line 1079 of file threadimpl.h.

#### 8.170.5.36 _Thread_Get_name()

```
size_t _Thread_Get_name (
            const Thread_Control * the_thread,
            char * buffer,
            size_t buffer_size )
```

Gets the name of the thread.

**Parameters**

| | | |
|---|---|---|
| | *the_thread* | The thread to get the name of. |
| out | *buffer* | Contains the thread's name. |
| | *buffer_size* | The size of *buffer*. |

**Returns**

The number of bytes copied to *buffer*.

### 8.170.5.37 _Thread_Get_objects_information()

```
static __inline__ Objects_Information* _Thread_Get_objects_information (
            Objects_Id id )  [static]
```

Gets object information for the object id.

**Parameters**

| | |
|---|---|
| *id* | The id of the object information. |

**Return values**

| | |
|---|---|
| *pointer* | The object information for this id. |
| *NULL* | The object id is not valid. |

Definition at line 826 of file threadimpl.h.

### 8.170.5.38 _Thread_Get_priority()

```
static __inline__ Priority_Control _Thread_Get_priority (
            const Thread_Control * the_thread )  [static]
```

Returns the priority of the thread.

Returns the user API and thread wait information relevant thread priority. This includes temporary thread priority adjustments due to locking protocols, a job release or the POSIX sporadic server for example.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread of which to get the priority. |

**Returns**

The priority of the thread.

Definition at line 1595 of file threadimpl.h.

#### 8.170.5.39 _Thread_Get_unmapped_priority()

```
static __inline__ Priority_Control _Thread_Get_unmapped_priority (
            const Thread_Control * the_thread )  [static]
```

Returns the unmapped priority of the thread.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread of which to get the unmapped priority. |

**Returns**

The unmapped priority of the thread.

Definition at line 1612 of file threadimpl.h.

#### 8.170.5.40 _Thread_Get_unmapped_real_priority()

```
static __inline__ Priority_Control _Thread_Get_unmapped_real_priority (
            const Thread_Control * the_thread )  [static]
```

Returns the unmapped real priority of the thread.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread of which to get the unmapped real priority. |

**Returns**

The unmapped real priority of the thread.

Definition at line 1626 of file threadimpl.h.

#### 8.170.5.41 _Thread_Handler()

```
void _Thread_Handler (
            void  )
```

Wrapper function for all threads.

This routine is the wrapper function for all threads. It is the starting point for all threads. The user provided thread entry point is invoked by this routine. Operations which must be performed immediately before and after the user's thread executes are found here.

**Note**

On entry, it is assumed all interrupts are blocked and that this routine needs to set the initial isr level. This may or may not actually be needed by the context switch routine and as a result interrupts may already be at there proper level. Either way, setting the initial isr level properly here is safe.

Definition at line 76 of file threadhandler.c.

#### 8.170.5.42 _Thread_Handler_initialization()

```
void _Thread_Handler_initialization (
             void  )
```

Initializes thread handler.

This routine performs the initialization necessary for this handler.

Definition at line 58 of file thread.c.

#### 8.170.5.43 _Thread_Initialize()

```
bool _Thread_Initialize (
             Thread_Information * information,
             Thread_Control * the_thread,
             const Thread_Configuration * config )
```

Initializes thread.

This routine initializes the specified the thread. It allocates all memory associated with this thread. It completes by adding the thread to the local object table so operations on this thread id are allowed.

**Note**

If stack_area is NULL, it is allocated from the workspace.

If the stack is allocated from the workspace, then it is guaranteed to be of at least minimum size.

**Parameters**

| | |
|---|---|
| *information* | The thread information. |
| *the_thread* | The thread to initialize. |
| *config* | The configuration of the thread to initialize. |

**Return values**

| | |
|---|---|
| *true* | The thread initialization was successful. |
| *false* | The thread initialization failed. |

Definition at line 30 of file threadinitialize.c.

#### 8.170.5.44 _Thread_Initialize_information()

```
void _Thread_Initialize_information (
             Thread_Information * information )
```

Initializes the thread information.

**Parameters**

| | | |
|---|---|---|
| out | *information* | Information to initialize. |

Definition at line 46 of file thread.c.

#### 8.170.5.45 _Thread_Internal_allocate()

```
static __inline__ Thread_Control* _Thread_Internal_allocate (
             void  )  [static]
```

Allocates an internal thread and returns it.

**Return values**

| | |
|---|---|
| *pointer* | Pointer to the allocated Thread_Control. |
| *NULL* | The operation failed. |

Definition at line 1101 of file threadimpl.h.

#### 8.170.5.46 _Thread_Is_context_switch_necessary()

```
static __inline__ bool _Thread_Is_context_switch_necessary (
             void  )  [static]
```

Deallocates the currently loaded floating point context.

This routine is invoked when the currently loaded floating point context is now longer associated with an active thread.

Checks if dispatching is disabled.

This function returns true if dispatching is disabled, and false otherwise.

**Return values**

| | |
|---|---|
| *true* | Dispatching is disabled. |
| *false* | Dispatching is enabled. |

Definition at line 1069 of file threadimpl.h.

### 8.170.5.47 _Thread_Is_executing()

```
static __inline__ bool _Thread_Is_executing (
             const Thread_Control * the_thread )  [static]
```

Checks if the thread is the currently executing thread.

This function returns true if the_thread is the currently executing thread, and false otherwise.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to verify if it is the currently executing thread. |

**Return values**

| | |
|---|---|
| *true* | *the_thread* is the currently executing one. |
| *false* | *the_thread* is not the currently executing one. |

Definition at line 910 of file threadimpl.h.

### 8.170.5.48 _Thread_Is_executing_on_a_processor()

```
static __inline__ bool _Thread_Is_executing_on_a_processor (
             const Thread_Control * the_thread )  [static]
```

Checks if the thread executes currently on some processor in the system.

Do not confuse this with _Thread_Is_executing() which checks only the current processor.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread for the verification. |

**Return values**

| | |
|---|---|
| *true* | *the_thread* is the currently executing one. |
| *false* | *the_thread* is not the currently executing one. |

Definition at line 930 of file threadimpl.h.

#### 8.170.5.49 _Thread_Is_heir()

```
static __inline__ bool _Thread_Is_heir (
            const Thread_Control * the_thread )  [static]
```

Checks if the thread is the heir.

This function returns true if the_thread is the heir thread, and false otherwise.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread for the verification. |

**Return values**

| | |
|---|---|
| *true* | *the_thread* is the heir. |
| *false* | *the_thread* is not the heir. |

Definition at line 949 of file threadimpl.h.

#### 8.170.5.50 _Thread_Is_joinable()

```
static __inline__ bool _Thread_Is_joinable (
            const Thread_Control * the_thread )  [static]
```

Checks if the thread is joinable.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread for the verification. |

**Return values**

| | |
|---|---|
| *true* | *life_state* is joinable. |
| *false* | *life_state* is not joinable. |

Definition at line 1311 of file threadimpl.h.

#### 8.170.5.51 _Thread_Is_life_change_allowed()

```
static __inline__ bool _Thread_Is_life_change_allowed (
            Thread_Life_state life_state )  [static]
```

Checks if the thread life state allos life change.

**Parameters**

| | |
|---|---|
| *life_state* | The thread life state for the verification. |

**Return values**

| | |
|---|---|
| *true* | *life_state* allows life change. |
| *false* | *life_state* does not allow life change. |

Definition at line 1279 of file threadimpl.h.

### 8.170.5.52 _Thread_Is_life_changing()

```
static __inline__ bool _Thread_Is_life_changing (
             Thread_Life_state life_state )  [static]
```

Checks if the thread life state is life changing.

**Parameters**

| | |
|---|---|
| *life_state* | The thread life state for the verification. |

**Return values**

| | |
|---|---|
| *true* | *life_state* is life changing. |
| *false* | *life_state* is not life changing. |

Definition at line 1295 of file threadimpl.h.

### 8.170.5.53 _Thread_Is_life_restarting()

```
static __inline__ bool _Thread_Is_life_restarting (
             Thread_Life_state life_state )  [static]
```

Checks if the thread life state is restarting.

**Parameters**

| | |
|---|---|
| *life_state* | The thread life state for the verification. |

**Return values**

| | |
|---|---|
| *true* | *life_state* is restarting. |
| *false* | *life_state* is not restarting. |

Definition at line 1249 of file threadimpl.h.

#### 8.170.5.54 _Thread_Is_life_terminating()

```
static __inline__ bool _Thread_Is_life_terminating (
            Thread_Life_state life_state )  [static]
```

Checks if the thread life state is terminating.

**Parameters**

| | |
|---|---|
| *life_state* | The thread life state for the verification. |

**Return values**

| | |
|---|---|
| *true* | *life_state* is terminating. |
| *false* | *life_state* is not terminating. |

Definition at line 1264 of file threadimpl.h.

#### 8.170.5.55 _Thread_Is_ready()

```
static __inline__ bool _Thread_Is_ready (
            const Thread_Control * the_thread )  [static]
```

Checks if the thread is ready.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to check if it is ready. |

**Return values**

| | |
|---|---|
| *true* | The thread is currently in the ready state. |
| *false* | The thread is currently not ready. |

Definition at line 401 of file threadimpl.h.

#### 8.170.5.56 _Thread_Iterate()

```
void _Thread_Iterate (
             Thread_Visitor visitor,
             void * arg )
```

Calls the visitor with all threads and the given argument until it is done.

**Parameters**

| | |
|---|---|
| *visitor* | Function that gets a thread and *arg* as parameters and returns if it is done. |
| *arg* | Parameter for *visitor* |

Definition at line 21 of file threaditerate.c.

#### 8.170.5.57 _Thread_Join()

```
void _Thread_Join (
             Thread_Control * the_thread,
             States_Control waiting_for_join,
             Thread_Control * executing,
             Thread_queue_Context * queue_context )
```

Joins the currently executing thread with the given thread to wait for.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread to wait for. |
| | *waiting_for_join* | The states control for the join. |
| in,out | *executing* | The currently executing thread. |
| | *queue_context* | The thread queue context. |

Definition at line 430 of file threadrestart.c.

#### 8.170.5.58 _Thread_Kill_zombies()

```
void _Thread_Kill_zombies (
             void  )
```

Kills all zombie threads in the system.

Threads change into the zombie state as the last step in the thread termination sequence right before a context switch to the heir thread is initiated. Since the thread stack is still in use during this phase we have to postpone the thread stack reclamation until this point. On SMP configurations we may have to busy wait for context switch completion here.

Definition at line 217 of file threadrestart.c.

#### 8.170.5.59 _Thread_Load_environment()

```
void _Thread_Load_environment (
            Thread_Control * the_thread )
```

Initializes enviroment for a thread.

This routine initializes the context of *the_thread* to its appropriate starting state.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The pointer to the thread control block. |

Definition at line 24 of file threadloadenv.c.

#### 8.170.5.60 _Thread_Pin()

```
static __inline__ void _Thread_Pin (
            Thread_Control * executing )  [static]
```

Pin the executing thread.

**Parameters**

| | |
|---|---|
| *executing* | The currently executing thread. |

Definition at line 2538 of file threadimpl.h.

#### 8.170.5.61 _Thread_Priority_add()

```
void _Thread_Priority_add (
            Thread_Control * the_thread,
            Priority_Node * priority_node,
            Thread_queue_Context * queue_context )
```

Adds the specified thread priority node to the corresponding thread priority aggregation.

The caller must be the owner of the thread wait lock.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread. |
| *priority_node* | The thread priority node to add. |
| *queue_context* | The thread queue context to return an updated set of threads for _Thread_Priority_update(). The thread queue context must be initialized via _Thread_queue_Context_clear_priority_updates() before a call of this function. |

**See also**

_Thread_Wait_acquire().

Definition at line 277 of file threadchangepriority.c.

### 8.170.5.62 _Thread_Priority_and_sticky_update()

```
void _Thread_Priority_and_sticky_update (
              Thread_Control * the_thread,
              int sticky_level_change )
```

Updates the priority of the thread and changes it sticky level.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread. |
| *sticky_level_change* | The new value for the sticky level. |

Definition at line 362 of file threadchangepriority.c.

### 8.170.5.63 _Thread_Priority_change()

```
static __inline__ void _Thread_Priority_change (
              Thread_Control * the_thread,
              Priority_Node * priority_node,
              Priority_Control new_priority,
              bool prepend_it,
              Thread_queue_Context * queue_context )  [static]
```

Changes the thread priority value of the specified thread priority node in the corresponding thread priority aggregation.

The caller must be the owner of the thread wait lock.

**Parameters**

| | | |
|---|---|---|
| | *the_thread* | The thread. |
| out | *priority_node* | The thread priority node to change. |
| | *new_priority* | The new thread priority value of the thread priority node to change. |
| | *prepend_it* | In case this is true, then the thread is prepended to its priority group in its home scheduler instance, otherwise it is appended. |
| | *queue_context* | The thread queue context to return an updated set of threads for _Thread_Priority_update(). The thread queue context must be initialized via _Thread_queue_Context_clear_priority_updates() before a call of this function. |

**See also**

_Thread_Wait_acquire().

Definition at line 722 of file threadimpl.h.

#### 8.170.5.64 _Thread_Priority_changed()

```
void _Thread_Priority_changed (
              Thread_Control * the_thread,
              Priority_Node * priority_node,
              bool prepend_it,
              Thread_queue_Context * queue_context )
```

Propagates a thread priority value change in the specified thread priority node to the corresponding thread priority aggregation.

The caller must be the owner of the thread wait lock.

**Parameters**

| | | |
|---|---|---|
| | *the_thread* | The thread. |
| out | *priority_node* | The thread priority node to change. |
| | *prepend_it* | In case this is true, then the thread is prepended to its priority group in its home scheduler instance, otherwise it is appended. |
| | *queue_context* | The thread queue context to return an updated set of threads for _Thread_Priority_update(). The thread queue context must be initialized via _Thread_queue_Context_clear_priority_updates() before a call of this function. |

**See also**

_Thread_Wait_acquire().

Definition at line 307 of file threadchangepriority.c.

#### 8.170.5.65 _Thread_Priority_highest()

```
static __inline__ Priority_Control _Thread_Priority_highest (
              Priority_Control left,
              Priority_Control right )  [static]
```

Returns the highest priority of the left and right thread priorities in the intuitive sense of priority.

**Parameters**

| | |
|---|---|
| *left* | The left thread priority. |
| *right* | The right thread priority. |

**Returns**

The highest priority in the intuitive sense of priority.

Definition at line 810 of file threadimpl.h.

### 8.170.5.66 _Thread_Priority_less_than()

```
static __inline__ bool _Thread_Priority_less_than (
              Priority_Control left,
              Priority_Control right )  [static]
```

Checks if the left thread priority is less than the right thread priority in the intuitive sense of priority.

**Parameters**

| | |
|---|---|
| *left* | The left thread priority. |
| *right* | The right thread priority. |

**Return values**

| | |
|---|---|
| *true* | The left priority is less in the intuitive sense. |
| *false* | The left priority is greater or equal in the intuitive sense. |

Definition at line 793 of file threadimpl.h.

### 8.170.5.67 _Thread_Priority_perform_actions()

```
void _Thread_Priority_perform_actions (
              Thread_Control * start_of_path,
              Thread_queue_Context * queue_context )
```

Checks if the thread is owner of the lock of the join queue.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread for the verification. |

**Return values**

| | |
|---|---|
| *true* | The thread is owner of the lock of the join queue. |
| *false* | The thread is not owner of the lock of the join queue. |

Performs the priority actions specified by the thread queue context along the thread queue path.

The caller must be the owner of the thread wait lock.

**Parameters**

| | |
|---|---|
| *start_of_path* | The start thread of the thread queue path. |
| *queue_context* | The thread queue context specifying the thread queue path and initial thread priority actions. |

**See also**

_Thread_queue_Path_acquire_critical().

Definition at line 182 of file threadchangepriority.c.

### 8.170.5.68 _Thread_Priority_remove()

```
void _Thread_Priority_remove (
              Thread_Control * the_thread,
              Priority_Node * priority_node,
              Thread_queue_Context * queue_context )
```

Removes the specified thread priority node from the corresponding thread priority aggregation.

The caller must be the owner of the thread wait lock.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread. |
| *priority_node* | The thread priority node to remove. |
| *queue_context* | The thread queue context to return an updated set of threads for _Thread_Priority_update(). The thread queue context must be initialized via _Thread_queue_Context_clear_priority_updates() before a call of this function. |

**See also**

_Thread_Wait_acquire().

Definition at line 292 of file threadchangepriority.c.

### 8.170.5.69 _Thread_Priority_replace()

```
void _Thread_Priority_replace (
              Thread_Control * the_thread,
              Priority_Node * victim_node,
              Priority_Node * replacement_node )
```

Replaces the victim priority node with the replacement priority node in the corresponding thread priority aggregation.

The caller must be the owner of the thread wait lock.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread. |
| *victim_node* | The victim thread priority node. |
| *replacement_node* | The replacement thread priority node. |

**See also**

_Thread_Wait_acquire().

Definition at line 323 of file threadchangepriority.c.

### 8.170.5.70 _Thread_Priority_update()

```
void _Thread_Priority_update (
              Thread_queue_Context * queue_context )
```

Updates the priority of all threads in the set.

**Parameters**

| | |
|---|---|
| *queue_context* | The thread queue context to return an updated set of threads for _Thread_Priority_update(). The thread queue context must be initialized via _Thread_queue_Context_clear_priority_updates() before a call of this function. |

**See also**

_Thread_Priority_add(), _Thread_Priority_change(), _Thread_Priority_changed() and _Thread_Priority_remove().

Definition at line 339 of file threadchangepriority.c.

### 8.170.5.71 _Thread_Remove_timer_and_unblock()

```
static __inline__ void _Thread_Remove_timer_and_unblock (
              Thread_Control * the_thread,
              Thread_queue_Queue * queue )  [static]
```

Remove the watchdog timer from the thread and unblock if necessary.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread to remove the watchdog from and unblock if necessary. |
| | *queue* | The thread queue. |

Definition at line 2467 of file threadimpl.h.

#### 8.170.5.72 _Thread_Resource_count_decrement()

```
static __inline__ void _Thread_Resource_count_decrement (
              Thread_Control * the_thread )  [static]
```

Decrements the thread's resource count.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread to decrement the resource count of. |

Definition at line 1340 of file threadimpl.h.

#### 8.170.5.73 _Thread_Resource_count_increment()

```
static __inline__ void _Thread_Resource_count_increment (
              Thread_Control * the_thread )  [static]
```

Increments the thread's resource count.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread to increase the resource count of. |

Definition at line 1324 of file threadimpl.h.

#### 8.170.5.74 _Thread_Restart_other()

```
bool _Thread_Restart_other (
              Thread_Control * the_thread,
              const Thread_Entry_information * entry,
              ISR_lock_Context * lock_context )
```

Restarts the thread.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread to restart. |
| | *entry* | The start entry information for *the_thread*. |
| | *lock_context* | The lock context. |

**Return values**

| | |
|---|---|
| *true* | The operation was successful. |
| *false* | The operation failed. |

Definition at line 568 of file threadrestart.c.

### 8.170.5.75 _Thread_Restart_self()

```
RTEMS_NO_RETURN void _Thread_Restart_self (
            Thread_Control * executing,
            const Thread_Entry_information * entry,
            ISR_lock_Context * lock_context )
```

Restarts the currently executing thread.

**Parameters**

| | | |
|---|---|---|
| in,out | *executing* | The currently executing thread. |
| | *entry* | The start entry information for *executing*. |
| | *lock_context* | The lock context. |

Definition at line 611 of file threadrestart.c.

### 8.170.5.76 _Thread_Restore_fp()

```
static __inline__ void _Thread_Restore_fp (
            Thread_Control * executing )  [static]
```

Restores the executing thread's floating point area.

**Parameters**

| | |
|---|---|
| *executing* | The currently executing thread. |

Definition at line 1029 of file threadimpl.h.

### 8.170.5.77 _Thread_Save_fp()

```
static __inline__ void _Thread_Save_fp (
            Thread_Control * executing )  [static]
```

Checks if the floating point context of the thread is currently loaded in the floating point unit.

This function returns true if the floating point context of the_thread is currently loaded in the floating point unit, and false otherwise.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread for the verification. |

**Return values**

| | |
|---|---|
| *true* | The floating point context of *the_thread* is currently loaded in the floating point unit. |
| *false* | The floating point context of *the_thread* is currently not loaded in the floating point unit. |

Saves the executing thread's floating point area.

**Parameters**

| | |
|---|---|
| *executing* | The currently executing thread. |

Definition at line 1014 of file threadimpl.h.

### 8.170.5.78 _Thread_Scheduler_acquire_critical()

```
static __inline__ void _Thread_Scheduler_acquire_critical (
             Thread_Control * the_thread,
             ISR_lock_Context * lock_context )  [static]
```

Acquires the lock context in a critical section.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to acquire the lock context. |
| *lock_context* | The lock context. |

Definition at line 1468 of file threadimpl.h.

### 8.170.5.79 _Thread_Scheduler_add_request()

```
static __inline__ void _Thread_Scheduler_add_request (
             Thread_Control * the_thread,
             Scheduler_Node * scheduler_node,
             Scheduler_Node_request request )  [static]
```

Add a scheduler request to the thread.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread to add a scheduler request to. |
| in,out | *scheduler_node* | The scheduler node for the request. |
| | *request* | The request to add. |

Definition at line 1504 of file threadimpl.h.

#### 8.170.5.80 _Thread_Scheduler_add_wait_node()

```
static __inline__ void _Thread_Scheduler_add_wait_node (
              Thread_Control * the_thread,
              Scheduler_Node * scheduler_node )  [static]
```

Adds a wait node to the thread and adds a corresponding request to the thread.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread to add the wait node to. |
| | *scheduler_node* | The scheduler node which provides the wait node. |

Definition at line 1547 of file threadimpl.h.

#### 8.170.5.81 _Thread_Scheduler_cancel_need_for_help()

```
static __inline__ void _Thread_Scheduler_cancel_need_for_help (
              Thread_Control * the_thread,
              Per_CPU_Control * cpu )  [static]
```

Cancels the thread's need for help.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to cancel the help request of. |
| *cpu* | The cpu to get the lock context of in order to cancel the help request. |

Definition at line 1379 of file threadimpl.h.

#### 8.170.5.82 _Thread_Scheduler_get_home()

```
static __inline__ const Scheduler_Control* _Thread_Scheduler_get_home (
              const Thread_Control * the_thread )  [static]
```

Gets the home scheduler of the thread.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to get the home scheduler of. |

**Returns**

The thread's home scheduler.

Definition at line 1404 of file threadimpl.h.

### 8.170.5.83 _Thread_Scheduler_get_home_node()

```
static __inline__ Scheduler_Node* _Thread_Scheduler_get_home_node (
              const Thread_Control * the_thread )  [static]
```

Gets the scheduler's home node.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to get the home node of. |

**Returns**

The thread's home node.

Definition at line 1423 of file threadimpl.h.

### 8.170.5.84 _Thread_Scheduler_get_node_by_index()

```
static __inline__ Scheduler_Node* _Thread_Scheduler_get_node_by_index (
              const Thread_Control * the_thread,
              size_t scheduler_index )  [static]
```

Gets the thread's scheduler node by index.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread of which to get a scheduler node. |
| *scheduler_index* | The index of the desired scheduler node. |

**Returns**

The scheduler node with the specified index.

Definition at line 1445 of file threadimpl.h.

#### 8.170.5.85 _Thread_Scheduler_process_requests()

```
void _Thread_Scheduler_process_requests (
            Thread_Control * the_thread )
```

Process the thread's scheduler requests.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread for the operation. |

Definition at line 23 of file threadscheduler.c.

#### 8.170.5.86 _Thread_Scheduler_release_critical()

```
static __inline__ void _Thread_Scheduler_release_critical (
            Thread_Control * the_thread,
            ISR_lock_Context * lock_context )  [static]
```

Releases the lock context in a critical section.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to release the lock context. |
| *lock_context* | The lock context. |

Definition at line 1482 of file threadimpl.h.

#### 8.170.5.87 _Thread_Scheduler_remove_wait_node()

```
static __inline__ void _Thread_Scheduler_remove_wait_node (
            Thread_Control * the_thread,
            Scheduler_Node * scheduler_node )  [static]
```

Remove a wait node from the thread and add a corresponding request to it.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to add the request to remove a wait node. |
| *scheduler_node* | The scheduler node to remove a wait node from. |

Definition at line 1570 of file threadimpl.h.

### 8.170.5.88 _Thread_Set_CPU()

```
static __inline__ void _Thread_Set_CPU (
            Thread_Control * thread,
            Per_CPU_Control * cpu )  [static]
```

Sets the cpu of the thread's scheduler.

**Parameters**

| | | |
|---|---|---|
| out | *thread* | The thread. |
| | *cpu* | The cpu to set. |

Definition at line 886 of file threadimpl.h.

### 8.170.5.89 _Thread_Set_life_protection()

```
Thread_Life_state _Thread_Set_life_protection (
            Thread_Life_state state )
```

Set the thread to life protected.

Calls _Thread_Change_life with the given state AND THREAD_LIFE_PROTECTED to set and THREAD_LIFE_↩ PROTECTED to clear.

**Parameters**

| | |
|---|---|
| *state* | The states to set. |

**Returns**

The previous state the thread was in.

Definition at line 680 of file threadrestart.c.

### 8.170.5.90 _Thread_Set_name()

```
Status_Control _Thread_Set_name (
            Thread_Control * the_thread,
            const char * name )
```

Sets the name of the thread.

**Parameters**

| | | |
|---|---|---|
| out | *the_thread* | The thread to change the name of. |
| | *name* | The new name for the thread. |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | The operation succeeded. |
| *STATUS_RESULT_TOO_LARGE* | The name was too long. |

### 8.170.5.91 _Thread_Set_state()

```
States_Control _Thread_Set_state (
            Thread_Control * the_thread,
            States_Control state )
```

Sets the specified thread state.

In the case the previous state is the ready state, then the thread is blocked by the scheduler.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |
| | *state* | The state to set. It must not be zero. |

**Returns**

The previous state.

Definition at line 50 of file threadsetstate.c.

### 8.170.5.92 _Thread_Set_state_locked()

```
States_Control _Thread_Set_state_locked (
            Thread_Control * the_thread,
            States_Control state )
```

Sets the specified thread state without locking the lock context.

In the case the previous state is the ready state, then the thread is blocked by the scheduler.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |
| | *state* | The state to set. It must not be zero. |

**Returns**

The previous state.

Definition at line 28 of file threadsetstate.c.

### 8.170.5.93 _Thread_Start()

```
bool _Thread_Start (
              Thread_Control * the_thread,
              const Thread_Entry_information * entry,
              ISR_lock_Context * lock_context )
```

Starts the specified thread.

If the thread is not in the dormant state, the routine returns with a value of false and performs no actions except enabling interrupts as indicated by the ISR lock context.

Otherwise, this routine initializes the executable information for the thread and makes it ready to execute. After the call of this routine, the thread competes with all other ready threads for CPU time.

Then the routine enables the local interrupts as indicated by the ISR lock context.

Then the thread start user extensions are called with thread dispatching disabled and interrupts enabled after making the thread ready. Please note that in SMP configurations, the thread switch and begin user extensions may be called in parallel on another processor.

Then thread dispatching is enabled and other threads may execute before the routine returns.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | is the thread to start. |
| | *entry* | is the thread entry information. |
| in,out | *is* | the ISR lock context which shall be used to disable the local interrupts before the call of this routine. |

**Return values**

| | |
|---|---|
| *true* | The thread was in the dormant state and was sucessefully started. |
| *false* | Otherwise. |

Definition at line 26 of file threadstart.c.

### 8.170.5.94 _Thread_Start_multitasking()

```
RTEMS_NO_RETURN void _Thread_Start_multitasking (
              void  )
```

Starts thread multitasking.

This routine initiates multitasking. It is invoked only as part of initialization and its invocation is the last act of the non-multitasking part of the system initialization.

Definition at line 24 of file threadstartmultitasking.c.

#### 8.170.5.95 _Thread_State_acquire()

```
static __inline__ void _Thread_State_acquire (
             Thread_Control * the_thread,
             ISR_lock_Context * lock_context )  [static]
```

Disables interrupts and acquires the lock_context.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to acquire the lock context. |
| *lock_context* | The lock context. |

Definition at line 542 of file threadimpl.h.

#### 8.170.5.96 _Thread_State_acquire_critical()

```
static __inline__ void _Thread_State_acquire_critical (
             Thread_Control * the_thread,
             ISR_lock_Context * lock_context )  [static]
```

Acquires the lock context in a critical section.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to acquire the lock context. |
| *lock_context* | The lock context. |

Definition at line 528 of file threadimpl.h.

#### 8.170.5.97 _Thread_State_acquire_for_executing()

```
static __inline__ Thread_Control* _Thread_State_acquire_for_executing (
             ISR_lock_Context * lock_context )  [static]
```

Disables interrupts and acquires the lock context for the currently executing thread.

**Parameters**

| | |
|---|---|
| *lock_context* | The lock context. |

**Returns**

The currently executing thread.

Definition at line 559 of file threadimpl.h.

### 8.170.5.98 _Thread_State_release()

```
static __inline__ void _Thread_State_release (
            Thread_Control * the_thread,
            ISR_lock_Context * lock_context )  [static]
```

Releases the lock context and enables interrupts.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread to release the lock context. |
| out | *lock_context* | The lock context. |

Definition at line 592 of file threadimpl.h.

### 8.170.5.99 _Thread_State_release_critical()

```
static __inline__ void _Thread_State_release_critical (
            Thread_Control * the_thread,
            ISR_lock_Context * lock_context )  [static]
```

Release the lock context in a critical section.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to release the lock context. |
| *lock_context* | The lock context. |

Definition at line 578 of file threadimpl.h.

### 8.170.5.100 _Thread_Timeout()

```
void _Thread_Timeout (
            Watchdog_Control * the_watchdog )
```

General purpose thread wait timeout.

**Parameters**

| | |
|---|---|
| *the_watchdog* | The thread timer watchdog. |

Definition at line 83 of file threadtimeout.c.

#### 8.170.5.101 _Thread_Timer_initialize()

```
static __inline__ void _Thread_Timer_initialize (
            Thread_Timer_information * timer,
            Per_CPU_Control * cpu )  [static]
```

Initializes the thread timer.

**Parameters**

| | | |
|---|---|---|
| in,out | *timer* | The timer to initialize. |
| | *cpu* | The cpu for the operation. |

Definition at line 2373 of file threadimpl.h.

#### 8.170.5.102 _Thread_Timer_insert_realtime()

```
static __inline__ void _Thread_Timer_insert_realtime (
            Thread_Control * the_thread,
            Per_CPU_Control * cpu,
            Watchdog_Service_routine_entry routine,
            uint64_t expire )  [static]
```

Inserts the cpu's watchdog realtime into the thread's timer.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | for the operation. |
| | *cpu* | The cpu to get the watchdog header from. |
| | *routine* | The watchdog routine for the thread. |
| | *expire* | Expiration for the watchdog. |

Definition at line 2416 of file threadimpl.h.

#### 8.170.5.103 _Thread_Timer_remove()

```
static __inline__ void _Thread_Timer_remove (
            Thread_Control * the_thread )  [static]
```

Remove the watchdog timer from the thread.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread to remove the watchdog from. |

Definition at line 2441 of file threadimpl.h.

#### 8.170.5.104 _Thread_Unblock()

```
static __inline__ void _Thread_Unblock (
            Thread_Control * the_thread )  [static]
```

Unblocks the thread.

This routine clears any blocking state for the_thread. It performs any necessary scheduling operations including the selection of a new heir thread.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread to unblock. |

Definition at line 965 of file threadimpl.h.

#### 8.170.5.105 _Thread_Unpin()

```
static __inline__ void _Thread_Unpin (
            Thread_Control * executing,
            Per_CPU_Control * cpu_self )  [static]
```

Unpins the thread.

**Parameters**

| | |
|---|---|
| *executing* | The currently executing thread. |
| *cpu_self* | The cpu for the operation. |

Definition at line 2555 of file threadimpl.h.

#### 8.170.5.106 _Thread_Update_CPU_time_used()

```
static __inline__ void _Thread_Update_CPU_time_used (
            Thread_Control * the_thread,
            Per_CPU_Control * cpu )  [static]
```

Updates the cpu time used of the thread.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread to add additional cpu time that is used. |
| | *cpu* | The cpu. |

Definition at line 1140 of file threadimpl.h.

#### 8.170.5.107 _Thread_Wait_acquire()

```
static __inline__ void _Thread_Wait_acquire (
            Thread_Control * the_thread,
            Thread_queue_Context * queue_context )  [static]
```

Acquires the thread wait default lock and disables interrupts.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |
| in,out | *queue_context* | The thread queue context for the corresponding _Thread_Wait_release(). |

Definition at line 1853 of file threadimpl.h.

#### 8.170.5.108 _Thread_Wait_acquire_critical()

```
static __inline__ void _Thread_Wait_acquire_critical (
            Thread_Control * the_thread,
            Thread_queue_Context * queue_context )  [static]
```

Acquires the thread wait lock inside a critical section (interrupts disabled).

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |
| in,out | *queue_context* | The thread queue context for the corresponding _Thread_Wait_release_critical(). |

Definition at line 1797 of file threadimpl.h.

#### 8.170.5.109 _Thread_Wait_acquire_default()

```
static __inline__ void _Thread_Wait_acquire_default (
             Thread_Control * the_thread,
             ISR_lock_Context * lock_context )  [static]
```

Acquires the thread wait default lock and disables interrupts.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |
| out | *lock_context* | The lock context used for the corresponding lock release. |

**See also**

_Thread_Wait_release_default().

Definition at line 1684 of file threadimpl.h.

#### 8.170.5.110 _Thread_Wait_acquire_default_critical()

```
static __inline__ void _Thread_Wait_acquire_default_critical (
             Thread_Control * the_thread,
             ISR_lock_Context * lock_context )  [static]
```

Acquires the thread wait default lock inside a critical section (interrupts disabled).

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |
| | *lock_context* | The lock context used for the corresponding lock release. |

**See also**

_Thread_Wait_release_default_critical().

Definition at line 1643 of file threadimpl.h.

#### 8.170.5.111 _Thread_Wait_acquire_default_for_executing()

```
static __inline__ Thread_Control* _Thread_Wait_acquire_default_for_executing (
             ISR_lock_Context * lock_context )  [static]
```

Acquires the thread wait default lock and returns the executing thread.

**Parameters**

| | |
|---|---|
| *lock_context* | The lock context used for the corresponding lock release. |

**Returns**

The executing thread.

**See also**

_Thread_Wait_release_default().

Definition at line 1662 of file threadimpl.h.

#### 8.170.5.112 _Thread_Wait_acquire_queue_critical()

```
static __inline__ void _Thread_Wait_acquire_queue_critical (
            Thread_queue_Queue * queue,
            Thread_queue_Lock_context * queue_lock_context )  [static]
```

Acquires the wait queue inside a critical section.

**Parameters**

| | |
|---|---|
| *queue* | The queue that acquires. |
| *queue_lock_context* | The queue lock context. |

Definition at line 1759 of file threadimpl.h.

#### 8.170.5.113 _Thread_Wait_cancel()

```
static __inline__ void _Thread_Wait_cancel (
            Thread_Control * the_thread,
            Thread_queue_Context * queue_context )  [static]
```

Cancels a thread wait on a thread queue.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |
| | *queue_context* | The thread queue context used for corresponding _Thread_Wait_acquire(). |

Definition at line 2091 of file threadimpl.h.

### 8.170.5.114 _Thread_Wait_claim()

```
static __inline__ void _Thread_Wait_claim (
            Thread_Control * the_thread,
            Thread_queue_Queue * queue )  [static]
```

Claims the thread wait queue.

The caller must not be the owner of the default thread wait lock. The caller must be the owner of the corresponding thread queue lock. The registration of the corresponding thread queue operations is deferred and done after the deadlock detection. This is crucial to support timeouts on SMP configurations.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |
| in,out | *queue* | The new thread queue. |

**See also**

_Thread_Wait_claim_finalize() and _Thread_Wait_restore_default().

Definition at line 1937 of file threadimpl.h.

### 8.170.5.115 _Thread_Wait_claim_finalize()

```
static __inline__ void _Thread_Wait_claim_finalize (
            Thread_Control * the_thread,
            const Thread_queue_Operations * operations )  [static]
```

Finalizes the thread wait queue claim via registration of the corresponding thread queue operations.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |
| | *operations* | The corresponding thread queue operations. |

Definition at line 1966 of file threadimpl.h.

### 8.170.5.116 _Thread_Wait_flags_get()

```
static __inline__ Thread_Wait_flags _Thread_Wait_flags_get (
            const Thread_Control * the_thread )  [static]
```

Gets the thread's wait flags according to the ATOMIC_ORDER_RELAXED.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to get the wait flags of. |

**Returns**

The thread's wait flags.

Definition at line 2200 of file threadimpl.h.

#### 8.170.5.117 _Thread_Wait_flags_get_acquire()

```
static __inline__ Thread_Wait_flags _Thread_Wait_flags_get_acquire (
              const Thread_Control * the_thread )  [static]
```

Gets the thread's wait flags according to the ATOMIC_ORDER_ACQUIRE.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to get the wait flags of. |

**Returns**

The thread's wait flags.

Definition at line 2218 of file threadimpl.h.

#### 8.170.5.118 _Thread_Wait_flags_set()

```
static __inline__ void _Thread_Wait_flags_set (
              Thread_Control * the_thread,
              Thread_Wait_flags flags )  [static]
```

Sets the thread's wait flags.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread to set the wait flags of. |
| | *flags* | The flags to set. |

Definition at line 2181 of file threadimpl.h.

#### 8.170.5.119 _Thread_Wait_flags_try_change_acquire()

```
static __inline__ bool _Thread_Wait_flags_try_change_acquire (
              Thread_Control * the_thread,
              Thread_Wait_flags expected_flags,
              Thread_Wait_flags desired_flags )  [static]
```

Tries to change the thread wait flags with acquire semantics.

In case the wait flags are equal to the expected wait flags, then the wait flags are set to the desired wait flags.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread. |
| *expected_flags* | The expected wait flags. |
| *desired_flags* | The desired wait flags. |

**Return values**

| | |
|---|---|
| *true* | The wait flags were equal to the expected wait flags. |
| *false* | The wait flags were not equal to the expected wait flags. |

Definition at line 2285 of file threadimpl.h.

#### 8.170.5.120 _Thread_Wait_flags_try_change_release()

```
static __inline__ bool _Thread_Wait_flags_try_change_release (
              Thread_Control * the_thread,
              Thread_Wait_flags expected_flags,
              Thread_Wait_flags desired_flags )  [static]
```

Tries to change the thread wait flags with release semantics in case of success.

Must be called inside a critical section (interrupts disabled).

In case the wait flags are equal to the expected wait flags, then the wait flags are set to the desired wait flags.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread. |
| *expected_flags* | The expected wait flags. |
| *desired_flags* | The desired wait flags. |

**Return values**

| | |
|---|---|
| *true* | The wait flags were equal to the expected wait flags. |
| *false* | The wait flags were not equal to the expected wait flags. |

Definition at line 2245 of file threadimpl.h.

#### 8.170.5.121 _Thread_Wait_get_id()

```
Objects_Id _Thread_Wait_get_id (
              const Thread_Control * the_thread )
```

Returns the object identifier of the object containing the current thread wait queue.

This function may be used for debug and system information purposes. The caller must be the owner of the thread lock.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread. |

**Return values**

| | |
|---|---|
| *0* | The thread waits on no thread queue currently, the thread wait queue is not contained in an object, or the current thread state provides insufficient information, e.g. the thread is in the middle of a blocking operation. |
| *other* | The object identifier of the object containing the thread wait queue. |

#### 8.170.5.122 _Thread_Wait_get_status()

```
static __inline__ Status_Control _Thread_Wait_get_status (
              const Thread_Control * the_thread )  [static]
```

Get the status of the wait return code of the thread.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to get the status of the wait return code of. |

Definition at line 2339 of file threadimpl.h.

#### 8.170.5.123 _Thread_Wait_release()

```
static __inline__ void _Thread_Wait_release (
              Thread_Control * the_thread,
              Thread_queue_Context * queue_context )  [static]
```

Releases the thread wait lock and restores the previous interrupt status.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |
| in,out | *queue_context* | The thread queue context used for corresponding _Thread_Wait_acquire(). |

Definition at line 1914 of file threadimpl.h.

#### 8.170.5.124 _Thread_Wait_release_critical()

```
static __inline__ void _Thread_Wait_release_critical (
            Thread_Control * the_thread,
            Thread_queue_Context * queue_context )  [static]
```

Releases the thread wait lock inside a critical section (interrupts disabled).

The previous interrupt status is not restored.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |
| in,out | *queue_context* | The thread queue context used for corresponding _Thread_Wait_acquire_critical(). |

Definition at line 1872 of file threadimpl.h.

#### 8.170.5.125 _Thread_Wait_release_default()

```
static __inline__ void _Thread_Wait_release_default (
            Thread_Control * the_thread,
            ISR_lock_Context * lock_context )  [static]
```

Releases the thread wait default lock and restores the previous interrupt status.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |
| out | *lock_context* | The lock context used for the corresponding lock acquire. |

Definition at line 1719 of file threadimpl.h.

#### 8.170.5.126 _Thread_Wait_release_default_critical()

```
static __inline__ void _Thread_Wait_release_default_critical (
            Thread_Control * the_thread,
            ISR_lock_Context * lock_context )  [static]
```

Releases the thread wait default lock inside a critical section (interrupts disabled).

The previous interrupt status is not restored.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |
| | *lock_context* | The lock context used for the corresponding lock acquire. |

Definition at line 1703 of file threadimpl.h.

### 8.170.5.127 _Thread_Wait_release_queue_critical()

```
static __inline__ void _Thread_Wait_release_queue_critical (
            Thread_queue_Queue * queue,
            Thread_queue_Lock_context * queue_lock_context )  [static]
```

Releases the wait queue inside a critical section.

**Parameters**

| | |
|---|---|
| *queue* | The queue that releases. |
| *queue_lock_context* | The queue lock context. |

Definition at line 1777 of file threadimpl.h.

### 8.170.5.128 _Thread_Wait_remove_request()

```
static __inline__ void _Thread_Wait_remove_request (
            Thread_Control * the_thread,
            Thread_queue_Lock_context * queue_lock_context )  [static]
```

Removes a thread wait lock request.

On SMP configurations, removes a thread wait lock request.

On other configurations, this function does nothing.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |
| in,out | *queue_lock_context* | The thread queue lock context used for corresponding _Thread_Wait_acquire(). |

Definition at line 1985 of file threadimpl.h.

#### 8.170.5.129 _Thread_Wait_remove_request_locked()

```
static __inline__ void _Thread_Wait_remove_request_locked (
             Thread_Control * the_thread,
             Thread_queue_Lock_context * queue_lock_context )  [static]
```

Removes the first pending wait lock request.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to remove the request from. |
| *queue_lock_context* | The queue lock context. |

Definition at line 1738 of file threadimpl.h.

#### 8.170.5.130 _Thread_Wait_restore_default()

```
static __inline__ void _Thread_Wait_restore_default (
             Thread_Control * the_thread )  [static]
```

Restores the default thread wait queue and operations.

The caller must be the owner of the current thread wait queue lock.

On SMP configurations, the pending requests are updated to use the stale thread queue operations.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread. |

**See also**

_Thread_Wait_claim().

Definition at line 2014 of file threadimpl.h.

#### 8.170.5.131 _Thread_Wait_tranquilize()

```
static __inline__ void _Thread_Wait_tranquilize (
             Thread_Control * the_thread )  [static]
```

Tranquilizes the thread after a wait on a thread queue.

After the violent blocking procedure this function makes the thread calm and peaceful again so that it can carry out its normal work.

On SMP configurations, ensures that all pending thread wait lock requests completed before the thread is able to begin a new thread wait procedure.

On other configurations, this function does nothing.

It must be called after a _Thread_Wait_claim() exactly once

- after the corresponding thread queue lock was released, and
- the default wait state is restored or some other processor is about to do this.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread. |

Definition at line 2073 of file threadimpl.h.

#### 8.170.5.132 _Thread_Yield()

```
void _Thread_Yield (
              Thread_Control * executing )
```

Yields the currently executing thread.

**Parameters**

| | | |
|---|---|---|
| in,out | *executing* | The thread that performs a yield. |

Definition at line 30 of file threadyield.c.

#### 8.170.5.133 rtems_iterate_over_all_threads()

```
void rtems_iterate_over_all_threads (
              rtems_per_thread_routine routine )
```

Deprecated, use rtems_task_iterate() instead.

Use rtems_task_iterate() instead.

### 8.170.6 Variable Documentation

#### 8.170.6.1 _Thread_Control_add_on_count

```
const size_t _Thread_Control_add_on_count  [extern]
```

Thread control add-on count.

Count of entries in _Thread_Control_add_ons.

This value is provided via <rtems/confdefs.h>.

### 8.170.6.2 _Thread_Control_add_ons

```
const Thread_Control_add_on _Thread_Control_add_ons[] [extern]
```

Thread control add-ons.

The thread control block contains fields that point to application configuration dependent memory areas, like the scheduler information, the API control blocks, the user extension context table, and the Newlib re-entrancy support. Account for these areas in the configuration and avoid extra workspace allocations for these areas.

This array is provided via <rtems/confdefs.h>.

**See also**

_Thread_Control_add_on_count.

### 8.170.6.3 _Thread_Global_constructor

```
Objects_Id _Thread_Global_constructor [extern]
```

Object identifier of the global constructor thread.

This variable is set by _RTEMS_tasks_Initialize_user_tasks_body() or _POSIX_Threads_Initialize_user_threads↩ _body().

It is consumed by _Thread_Handler().

Definition at line 53 of file threadhandler.c.

### 8.170.6.4 _Thread_Idle_body

```
const Thread_Idle_body _Thread_Idle_body [extern]
```

The idle thread body.

This constant is defined by the application configuration via <rtems/confdefs.h>.

Definition at line 35 of file threadidledefault.c.

### 8.170.6.5 _Thread_Idle_stack_size

```
const size_t _Thread_Idle_stack_size [extern]
```

The idle thread stack size in bytes.

This constant is defined by the application configuration via <rtems/confdefs.h>.

#### 8.170.6.6 _Thread_Idle_stacks

```
char _Thread_Idle_stacks[]  [extern]
```

The idle thread stacks.

Provided by the application via <rtems/confdefs.h>.

#### 8.170.6.7 _Thread_Initial_thread_count

```
const size_t _Thread_Initial_thread_count  [extern]
```

Count of configured threads.

This value is provided via <rtems/confdefs.h>.

#### 8.170.6.8 _Thread_Maximum_name_size

```
const size_t _Thread_Maximum_name_size  [extern]
```

Maximum size of a thread name in characters (including the terminating '\0' character).

This value is provided via <rtems/confdefs.h>.

#### 8.170.6.9 _Thread_Maximum_TLS_size

```
const size_t _Thread_Maximum_TLS_size  [extern]
```

If this constant is greater than zero, then it defines the maximum thread-local storage size, otherwise the thread-local storage size is defined by the linker depending on the thread-local storage objects used by the application in the statically-linked executable.

This value is provided via <rtems/confdefs.h>.

#### 8.170.6.10 _Thread_queue_Heads_size

```
const size_t _Thread_queue_Heads_size  [extern]
```

Size of the thread queue heads of a particular application.

In SMP configurations, this value is provided via <rtems/confdefs.h>.

#### 8.170.6.11 rtems_ada_self

```
void* rtems_ada_self  [extern]
```

Self for the GNU Ada Run-Time

# 8.171 Thread Queue Handler

Thread Queue Handler.

## Files

- file threadq.h

  *Constants and Structures Needed to Declare a Thread Queue.*
- file threadqimpl.h

  *Constants and Structures Associated with the Manipulation of Objects.*

## Classes

- struct Thread_queue_Gate

  *The thread queue gate is an SMP synchronization means.*
- struct Thread_queue_Lock_context
- struct Thread_queue_Link

  *A thread queue link from one thread to another specified by the thread queue owner and thread wait queue relationships.*
- struct Thread_queue_Context

  *Thread queue context for the thread queue methods.*
- struct Thread_queue_Priority_queue

  *Thread priority queue.*
- struct _Thread_queue_Heads

  *Thread queue heads.*
- struct Thread_queue_Queue
- struct Thread_queue_Operations

  *Thread queue operations.*
- struct Thread_queue_Control
- struct Thread_queue_Syslock_queue

  *Thread queue with a layout compatible to struct _Thread_queue_Queue defined in Newlib <sys/lock.h>.*
- struct Thread_queue_Object

  *Helper structure to ensure that all objects containing a thread queue have the right layout.*

## Macros

- #define **THREAD_QUEUE_LINK_OF_PATH_NODE**(node) RTEMS_CONTAINER_OF( node, Thread_queue_Link, Path_node );
- #define **_Thread_queue_Context_ISR_disable**(queue_context, level)
- #define _Thread_queue_Context_set_MP_callout(queue_context, mp_callout)

  *Sets the MP callout in the thread queue context.*
- #define **_Thread_queue_Queue_acquire_critical**(queue, lock_stats, lock_context) _Thread_queue_Queue_do_acquire_criti queue, lock_context )
- #define _Thread_queue_Dequeue(the_thread_queue, operations, mp_callout)

  *Gets a pointer to a thread waiting on the_thread_queue.*
- #define **THREAD_QUEUE_INITIALIZER**(_name)
- #define **THREAD_QUEUE_OBJECT_ASSERT**(object_type, wait_queue_member, msg)
- #define **THREAD_QUEUE_QUEUE_TO_OBJECT**(queue)

## Typedefs

- typedef struct _Thread_Control **Thread_Control**
- typedef struct Thread_queue_Context **Thread_queue_Context**
- typedef struct Thread_queue_Queue **Thread_queue_Queue**
- typedef struct Thread_queue_Operations **Thread_queue_Operations**
- typedef void(* Thread_queue_Enqueue_callout) (Thread_queue_Queue *queue, Thread_Control *the_thread, struct Per_CPU_Control *cpu_self, Thread_queue_Context *queue_context)

  *Thread queue enqueue callout.*
- typedef void(* Thread_queue_Deadlock_callout) (Thread_Control *the_thread)

  *Thread queue deadlock callout.*
- typedef struct _Thread_queue_Heads Thread_queue_Heads

  *Thread queue heads.*
- typedef void(* Thread_queue_Priority_actions_operation) (Thread_queue_Queue *queue, Priority_Actions *priority_actions)

  *Thread queue action operation.*
- typedef void(* Thread_queue_Enqueue_operation) (Thread_queue_Queue *queue, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Thread queue enqueue operation.*
- typedef void(* Thread_queue_Extract_operation) (Thread_queue_Queue *queue, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Thread queue extract operation.*
- typedef Thread_Control *(* Thread_queue_Surrender_operation) (Thread_queue_Queue *queue, Thread_queue_Heads *heads, Thread_Control *previous_owner, Thread_queue_Context *queue_context)

  *Thread queue surrender operation.*
- typedef Thread_Control *(* Thread_queue_First_operation) (Thread_queue_Heads *heads)

  *Thread queue first operation.*
- typedef Thread_Control *(* Thread_queue_Flush_filter) (Thread_Control *the_thread, Thread_queue_Queue *queue, Thread_queue_Context *queue_context)

  *Thread queue flush filter function.*

## Functions

- void _Thread_queue_Enqueue_do_nothing_extra (Thread_queue_Queue *queue, Thread_Control *the_thread, Per_CPU_Control *cpu_self, Thread_queue_Context *queue_context)

  *Does nothing.*
- void _Thread_queue_Add_timeout_ticks (Thread_queue_Queue *queue, Thread_Control *the_thread, Per_CPU_Control *cpu_self, Thread_queue_Context *queue_context)

  *Adds timeout ticks of the queue to the thread.*
- void _Thread_queue_Add_timeout_monotonic_timespec (Thread_queue_Queue *queue, Thread_Control *the_thread, Per_CPU_Control *cpu_self, Thread_queue_Context *queue_context)

  *Adds a monotonic timespec to the thread and sets the watchdog header to monotonic.*
- void _Thread_queue_Add_timeout_realtime_timespec (Thread_queue_Queue *queue, Thread_Control *the_thread, Per_CPU_Control *cpu_self, Thread_queue_Context *queue_context)

  *Adds a monotonic timespec to the thread and sets the watchdog header to realtime.*
- void _Thread_queue_Deadlock_status (Thread_Control *the_thread)

  *Sets the thread wait return code to STATUS_DEADLOCK.*
- void _Thread_queue_Deadlock_fatal (Thread_Control *the_thread)

  *Results in an INTERNAL_ERROR_THREAD_QUEUE_DEADLOCK fatal error.*
- static __inline__ void _Thread_queue_Context_initialize (Thread_queue_Context *queue_context)

  *Initializes a thread queue context.*

- static __inline__ void _Thread_queue_Context_set_thread_state (Thread_queue_Context *queue_context, States_Control thread_state)

  *Sets the thread state for the thread to enqueue in the thread queue context.*
- static __inline__ void _Thread_queue_Context_set_timeout_ticks (Thread_queue_Context *queue_context, Watchdog_Interval ticks)

  *Sets the timeout ticks in the thread queue context.*
- static __inline__ void _Thread_queue_Context_set_timeout_argument (Thread_queue_Context *queue_context, const void *arg)

  *Sets the timeout argument in the thread queue context.*
- static __inline__ void _Thread_queue_Context_set_enqueue_callout (Thread_queue_Context *queue_context, Thread_queue_Enqueue_callout enqueue_callout)

  *Sets the enqueue callout in the thread queue context.*
- static __inline__ void _Thread_queue_Context_set_enqueue_do_nothing_extra (Thread_queue_Context *queue_context)

  *Sets the do nothing enqueue callout in the thread queue context.*
- static __inline__ void _Thread_queue_Context_set_enqueue_timeout_ticks (Thread_queue_Context *queue_context, Watchdog_Interval ticks)

  *Sets the enqueue callout to add a relative monotonic timeout in ticks.*
- static __inline__ void _Thread_queue_Context_set_enqueue_timeout_monotonic_timespec (Thread_queue_Context *queue_context, const struct timespec *abstime)

  *Sets the enqueue callout to add an absolute monotonic timeout in timespec format.*
- static __inline__ void _Thread_queue_Context_set_enqueue_timeout_realtime_timespec (Thread_queue_Context *queue_context, const struct timespec *abstime)

  *Sets the enqueue callout to add an absolute realtime timeout in timespec format.*
- static __inline__ void _Thread_queue_Context_set_deadlock_callout (Thread_queue_Context *queue_context, Thread_queue_Deadlock_callout deadlock_callout)

  *Sets the deadlock callout in the thread queue context.*
- static __inline__ void _Thread_queue_Context_clear_priority_updates (Thread_queue_Context *queue_context)

  *Clears the priority update count of the thread queue context.*
- static __inline__ size_t _Thread_queue_Context_save_priority_updates (Thread_queue_Context *queue_context)

  *Returns the priority update count of the thread queue context.*
- static __inline__ void _Thread_queue_Context_restore_priority_updates (Thread_queue_Context *queue_context, size_t update_count)

  *Sets the priority update count of the thread queue context.*
- static __inline__ void _Thread_queue_Context_add_priority_update (Thread_queue_Context *queue_context, Thread_Control *the_thread)

  *Adds a priority update of the thread to the thread queue context.*
- static __inline__ void _Thread_queue_Context_set_ISR_level (Thread_queue_Context *queue_context, ISR_Level level)

  *Sets the thread queue context ISR level.*
- static __inline__ Per_CPU_Control * _Thread_queue_Dispatch_disable (Thread_queue_Context *queue_context)

  *Disables dispatching in a critical section.*
- static __inline__ void _Thread_queue_Gate_close (Thread_queue_Gate *gate)

  *Closes the gate.*
- static __inline__ void _Thread_queue_Gate_add (Chain_Control *chain, Thread_queue_Gate *gate)

  *Adds the gate to the chain.*
- static __inline__ void _Thread_queue_Gate_open (Thread_queue_Gate *gate)

  *Opens the gate.*
- static __inline__ void _Thread_queue_Gate_wait (Thread_queue_Gate *gate)

  *Waits on a gate to open.*

- static __inline__ void _Thread_queue_Heads_initialize (Thread_queue_Heads *heads)

  *Initializes the thread queue heads.*
- static __inline__ void _Thread_queue_Queue_initialize (Thread_queue_Queue *queue, const char *name)

  *Initializes the thread queue queue with the given name.*
- static __inline__ void _Thread_queue_Queue_do_acquire_critical (Thread_queue_Queue *queue, ISR_lock_Context *lock_context)

  *Acquires the thread queue queue in a critical section.*
- static __inline__ void _Thread_queue_Queue_release_critical (Thread_queue_Queue *queue, ISR_lock_Context *lock_context)

  *Releases the thread queue queue in a critical section.*
- static __inline__ void _Thread_queue_Queue_release (Thread_queue_Queue *queue, ISR_lock_Context *lock_context)

  *Releases the thread queue queue and enables interrupts.*
- size_t _Thread_queue_Queue_get_name_and_id (const Thread_queue_Queue *queue, char *buffer, size_t buffer_size, Objects_Id *id)

  *Copies the thread queue name to the specified buffer.*
- void _Thread_queue_Do_acquire_critical (Thread_queue_Control *the_thread_queue, ISR_lock_Context *lock_context)

  *Acquires the thread queue control in a critical section.*
- static __inline__ void _Thread_queue_Acquire_critical (Thread_queue_Control *the_thread_queue, Thread_queue_Context *queue_context)

  *Acquires the thread queue control in a critical section.*
- void _Thread_queue_Acquire (Thread_queue_Control *the_thread_queue, Thread_queue_Context *queue_context)

  *Acquires the thread queue control in a critical section.*
- void _Thread_queue_Do_release_critical (Thread_queue_Control *the_thread_queue, ISR_lock_Context *lock_context)

  *Checks if the thread queue control is the owner of the lock.*
- static __inline__ void _Thread_queue_Release_critical (Thread_queue_Control *the_thread_queue, Thread_queue_Context *queue_context)

  *Releases the thread queue control in a critical section.*
- void _Thread_queue_Release (Thread_queue_Control *the_thread_queue, Thread_queue_Context *queue_context)

  *Releases the thread queue control and enables interrupts.*
- Thread_Control * _Thread_queue_Do_dequeue (Thread_queue_Control *the_thread_queue, const Thread_queue_Operations *operations)

  *Dequeues the first thread waiting on the thread queue and returns it.*
- void _Thread_queue_Enqueue (Thread_queue_Queue *queue, const Thread_queue_Operations *operations, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Blocks the thread and places it on the thread queue.*
- Status_Control _Thread_queue_Enqueue_sticky (Thread_queue_Queue *queue, const Thread_queue_Operations *operations, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Enqueues the thread on the thread queue and busy waits for dequeue.*
- bool _Thread_queue_Extract_locked (Thread_queue_Queue *queue, const Thread_queue_Operations *operations, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Extracts the thread from the thread queue, restores the default wait operations and restores the default thread lock.*
- void _Thread_queue_Unblock_critical (bool unblock, Thread_queue_Queue *queue, Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Unblocks the thread which was on the thread queue before.*
- void _Thread_queue_Extract_critical (Thread_queue_Queue *queue, const Thread_queue_Operations *operations, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Extracts the thread from the thread queue and unblocks it.*
- void _Thread_queue_Extract (Thread_Control *the_thread)

*Extracts thread from thread queue.*

- void _Thread_queue_Extract_with_proxy (Thread_Control *the_thread)

  *Extracts the_thread from the_thread_queue.*

- void _Thread_queue_Surrender (Thread_queue_Queue *queue, Thread_queue_Heads *heads, Thread_Control *previous_owner, Thread_queue_Context *queue_context, const Thread_queue_Operations *operations)

  *Surrenders the thread queue previously owned by the thread to the first enqueued thread.*

- void _Thread_queue_Surrender_sticky (Thread_queue_Queue *queue, Thread_queue_Heads *heads, Thread_Control *previous_owner, Thread_queue_Context *queue_context, const Thread_queue_Operations *operations)

  *Surrenders the thread queue previously owned by the thread to the first enqueued thread.*

- static __inline__ bool _Thread_queue_Is_empty (const Thread_queue_Queue *queue)

  *Checks if the thread queue queue is empty.*

- static __inline__ Thread_Control * _Thread_queue_First_locked (Thread_queue_Control *the_thread_queue, const Thread_queue_Operations *operations)

  *Returns the first thread on the thread queue if it exists, otherwise NULL.*

- Thread_Control * _Thread_queue_First (Thread_queue_Control *the_thread_queue, const Thread_queue_Operations *operations)

  *Returns the first thread on the thread queue if it exists, otherwise NULL.*

- Thread_Control * _Thread_queue_Flush_default_filter (Thread_Control *the_thread, Thread_queue_Queue *queue, Thread_queue_Context *queue_context)

  *Default thread queue flush filter function.*

- Thread_Control * _Thread_queue_Flush_status_unavailable (Thread_Control *the_thread, Thread_queue_Queue *queue, Thread_queue_Context *queue_context)

  *Status unavailable thread queue flush filter function.*

- Thread_Control * _Thread_queue_Flush_status_object_was_deleted (Thread_Control *the_thread, Thread_queue_Queue *queue, Thread_queue_Context *queue_context)

  *Status object was deleted thread queue flush filter function.*

- size_t _Thread_queue_Flush_critical (Thread_queue_Queue *queue, const Thread_queue_Operations *operations, Thread_queue_Flush_filter filter, Thread_queue_Context *queue_context)

  *Unblocks all threads enqueued on the thread queue.*

- void _Thread_queue_Initialize (Thread_queue_Control *the_thread_queue, const char *name)

  *Initializes the thread queue control to the given name.*

- static __inline__ void _Thread_queue_Destroy (Thread_queue_Control *the_thread_queue)

  *Destroys the thread queue.*

- bool _Thread_queue_Path_acquire_critical (Thread_queue_Queue *queue, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Does nothing.*

- void _Thread_queue_Path_release_critical (Thread_queue_Context *queue_context)

  *Releases the thread queue path in a critical section.*

- void _Thread_queue_Object_initialize (Thread_queue_Control *the_thread_queue)

  *Initializes a thread queue embedded in an object with identifier.*

## Variables

- const Thread_queue_Operations **_Thread_queue_Operations_default**
- const Thread_queue_Operations **_Thread_queue_Operations_FIFO**
- const Thread_queue_Operations **_Thread_queue_Operations_priority**
- const Thread_queue_Operations **_Thread_queue_Operations_priority_inherit**
- const char _Thread_queue_Object_name [ ]

  *The special thread queue name to indicated that the thread queue is embedded in an object with identifier.*

### 8.171.1 Detailed Description

Thread Queue Handler.

This handler provides the capability to have threads block in ordered sets. The sets may be ordered using the FIFO or priority discipline.

### 8.171.2 Macro Definition Documentation

#### 8.171.2.1 _Thread_queue_Context_ISR_disable

```
#define _Thread_queue_Context_ISR_disable(
              queue_context,
              level )
```

**Value:**

```
  do { \
    _ISR_Local_disable( level ); \
    _ISR_lock_ISR_disable_profile( \
      &( queue_context )->Lock_context.Lock_context \
    ) \
  } while ( 0 )
```

Definition at line 397 of file threadqimpl.h.

#### 8.171.2.2 _Thread_queue_Context_set_MP_callout

```
#define _Thread_queue_Context_set_MP_callout(
              queue_context,
              mp_callout )
```

**Value:**

```
  do { \
    (void) queue_context; \
  } while ( 0 )
```

Sets the MP callout in the thread queue context.

**Parameters**

| | | |
|---|---|---|
| out | *queue_context* | The thread queue context. |
| | *mp_callout* | Callout to unblock the thread in case it is actually a thread proxy. This parameter is only used on multiprocessing configurations. Used by thread queue extract and unblock methods for objects with multiprocessing (MP) support. |

Definition at line 456 of file threadqimpl.h.

#### 8.171.2.3 _Thread_queue_Dequeue

```
#define _Thread_queue_Dequeue(
                the_thread_queue,
                operations,
                mp_callout )
```

**Value:**

```
    _Thread_queue_Do_dequeue( \
      the_thread_queue, \
      operations \
    )
```

Gets a pointer to a thread waiting on the_thread_queue.

This function returns a pointer to a thread waiting on the_thread_queue. The selection of this thread is based on the discipline of the_thread_queue. If no threads are waiting on the_thread_queue, then NULL is returned.

- INTERRUPT LATENCY:
  - single case

Definition at line 838 of file threadqimpl.h.

#### 8.171.2.4 THREAD_QUEUE_INITIALIZER

```
#define THREAD_QUEUE_INITIALIZER(
                _name )
```

**Value:**

```
    { \
      .Queue = { \
        .Lock = SMP_TICKET_LOCK_INITIALIZER, \
        .heads = NULL, \
        .owner = NULL, \
        .name = _name \
      } \
    }
```

Definition at line 1353 of file threadqimpl.h.

#### 8.171.2.5 THREAD_QUEUE_OBJECT_ASSERT

```
#define THREAD_QUEUE_OBJECT_ASSERT(
                object_type,
                wait_queue_member,
                msg )
```

**Value:**

```
  RTEMS_STATIC_ASSERT( \
    offsetof( object_type, wait_queue_member ) \
      == offsetof( Thread_queue_Object, Wait_queue ) \
    && RTEMS_HAVE_MEMBER_SAME_TYPE( \
      object_type, \
      wait_queue_member, \
      Thread_queue_Object, \
      Wait_queue \
    ), \
    msg \
  )
```

Definition at line 1450 of file threadqimpl.h.

#### 8.171.2.6 THREAD_QUEUE_QUEUE_TO_OBJECT

```
#define THREAD_QUEUE_QUEUE_TO_OBJECT(
              queue )
```

**Value:**

```
  RTEMS_CONTAINER_OF( \
    queue, \
    Thread_queue_Object, \
    Wait_queue.Queue \
  )
```

Definition at line 1463 of file threadqimpl.h.

### 8.171.3 Typedef Documentation

#### 8.171.3.1 Thread_queue_Deadlock_callout

```
typedef void( * Thread_queue_Deadlock_callout) (Thread_Control *the_thread)
```

Thread queue deadlock callout.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread that detected the deadlock. |

**See also**

_Thread_queue_Context_set_deadlock_callout().

Definition at line 86 of file threadq.h.

#### 8.171.3.2 Thread_queue_Enqueue_callout

```
typedef void( * Thread_queue_Enqueue_callout) (Thread_queue_Queue *queue, Thread_Control *the↩
_thread, struct Per_CPU_Control *cpu_self, Thread_queue_Context *queue_context)
```

Thread queue enqueue callout.

**Parameters**

| | | |
|---|---|---|
| in | *queue* | The actual thread queue. |
| in | *the_thread* | The thread to enqueue. |
| in | *cpu_self* | The current processor. |
| in | *queue_context* | The thread queue context of the lock acquire. |

**See also**

_Thread_queue_Context_set_enqueue_callout().

Definition at line 72 of file threadq.h.

#### 8.171.3.3 Thread_queue_Enqueue_operation

```
typedef void( * Thread_queue_Enqueue_operation) (Thread_queue_Queue *queue, Thread_Control
*the_thread, Thread_queue_Context *queue_context)
```

Thread queue enqueue operation.

A potential thread to update the priority due to priority inheritance is returned via the thread queue context. This thread is handed over to _Thread_Priority_update().

**Parameters**

| | | |
|---|---|---|
| in | *queue* | The actual thread queue. |
| in | *the_thread* | The thread to enqueue on the queue. |

Definition at line 463 of file threadq.h.

#### 8.171.3.4 Thread_queue_Extract_operation

```
typedef void( * Thread_queue_Extract_operation) (Thread_queue_Queue *queue, Thread_Control
*the_thread, Thread_queue_Context *queue_context)
```

Thread queue extract operation.

**Parameters**

| | | |
|---|---|---|
| in | *queue* | The actual thread queue. |
| in | *the_thread* | The thread to extract from the thread queue. |

Definition at line 475 of file threadq.h.

#### 8.171.3.5 Thread_queue_First_operation

```
typedef Thread_Control*( * Thread_queue_First_operation) (Thread_queue_Heads *heads)
```

Thread queue first operation.

**Parameters**

| | | |
|---|---|---|
| in | *heads* | The thread queue heads. |

**Return values**

| | |
|---|---|
| *NULL* | No thread is present on the thread queue. |
| *first* | The first thread of the thread queue according to the insert order. This thread remains on the thread queue. |

Definition at line 508 of file threadq.h.

### 8.171.3.6 Thread_queue_Flush_filter

```
typedef Thread_Control*( * Thread_queue_Flush_filter) (Thread_Control *the_thread, Thread_queue_Queue
*queue, Thread_queue_Context *queue_context)
```

Thread queue flush filter function.

Called under protection of the thread queue lock by _Thread_queue_Flush_critical() to optionally alter the thread wait information and control the iteration.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to extract. This is the first parameter to optimize for architectures that use the same register for the first parameter and the return value. |
| *queue* | The actual thread queue. |
| *queue_context* | The thread queue context of the lock acquire. May be used to pass additional data to the filter function via an overlay structure. The filter function should not release or acquire the thread queue lock. |

**Return values**

| | |
|---|---|
| *the_thread* | Extract this thread. |
| *NULL* | Do not extract this thread and stop the thread queue flush operation. Threads that are already extracted will complete the flush operation. |

Definition at line 1223 of file threadqimpl.h.

### 8.171.3.7 Thread_queue_Heads

```
typedef struct _Thread_queue_Heads Thread_queue_Heads
```

Thread queue heads.

Each thread is equipped with spare thread queue heads in case it is not enqueued on a thread queue. The first thread enqueued on a thread queue will give its spare thread queue heads to that thread queue. The threads arriving at the queue will add their thread queue heads to the free chain of the queue heads provided by the first thread enqueued. Once a thread is dequeued it use the free chain to get new spare thread queue heads.

Uses a leading underscore in the structure name to allow forward declarations in standard header files provided by Newlib and GCC.

#### 8.171.3.8 Thread_queue_Priority_actions_operation

```
typedef void( * Thread_queue_Priority_actions_operation) (Thread_queue_Queue *queue, Priority_Actions
*priority_actions)
```

Thread queue action operation.

**Parameters**

| | | |
|---|---|---|
| in | *queue* | The actual thread queue. |
| in | *the_thread* | The thread. |
| in | *queue_context* | The thread queue context providing the thread queue action set to perform. Returns the thread queue action set to perform on the thread queue owner or the empty set in case there is nothing to do. |

Definition at line 448 of file threadq.h.

#### 8.171.3.9 Thread_queue_Surrender_operation

```
typedef Thread_Control*( * Thread_queue_Surrender_operation) (Thread_queue_Queue *queue, Thread_queue_Heads
*heads, Thread_Control *previous_owner, Thread_queue_Context *queue_context)
```

Thread queue surrender operation.

This operation must dequeue and return the first thread on the queue.

**Parameters**

| | | |
|---|---|---|
| in | *queue* | The actual thread queue. |
| in | *heads* | The thread queue heads. It must not be NULL. |
| in | *previous_owner* | The previous owner of the thread queue. |

**Returns**

The previous first thread on the queue.

Definition at line 492 of file threadq.h.

### 8.171.4 Function Documentation

#### 8.171.4.1 _Thread_queue_Acquire()

```
void _Thread_queue_Acquire (
            Thread_queue_Control * the_thread_queue,
            Thread_queue_Context * queue_context )
```

Acquires the thread queue control in a critical section.

**Parameters**

| | | |
|---|---|---|
| | *the_thread_queue* | The thread queue control to acquire. |
| out | *queue_context* | The thread queue context. |

Definition at line 87 of file threadq.c.

#### 8.171.4.2 _Thread_queue_Acquire_critical()

```
static __inline__ void _Thread_queue_Acquire_critical (
            Thread_queue_Control * the_thread_queue,
            Thread_queue_Context * queue_context )  [static]
```

Acquires the thread queue control in a critical section.

**Parameters**

| | | |
|---|---|---|
| | *the_thread_queue* | The thread queue control to acquire. |
| out | *lock_context* | The interrupt lock context. |

Definition at line 681 of file threadqimpl.h.

#### 8.171.4.3 _Thread_queue_Add_timeout_monotonic_timespec()

```
void _Thread_queue_Add_timeout_monotonic_timespec (
            Thread_queue_Queue * queue,
            Thread_Control * the_thread,
            Per_CPU_Control * cpu_self,
            Thread_queue_Context * queue_context )
```

Adds a monotonic timespec to the thread and sets the watchdog header to monotonic.

**Parameters**

| | | |
|---|---|---|
| | *queue* | This parameter is unused. |
| in,out | *the_thread* | The thread to add the timeout and set watchdog header to monotonic. |
| | *cpu_self* | The cpu to get the monotonic watchdog header from. |
| | *queue_context* | The thread queue context. |

Definition at line 90 of file threadqtimeout.c.

#### 8.171.4.4 _Thread_queue_Add_timeout_realtime_timespec()

```
void _Thread_queue_Add_timeout_realtime_timespec (
            Thread_queue_Queue * queue,
            Thread_Control * the_thread,
            Per_CPU_Control * cpu_self,
            Thread_queue_Context * queue_context )
```

Adds a monotonic timespec to the thread and sets the watchdog header to realtime.

**Parameters**

| | | |
|---|---|---|
| | *queue* | This parameter is unused. |
| in,out | *the_thread* | The thread to add the timeout and set watchdog header to realtime. |
| | *cpu_self* | The cpu to get the realtime watchdog header from. |
| | *queue_context* | The thread queue context. |

Definition at line 110 of file threadqtimeout.c.

#### 8.171.4.5 _Thread_queue_Add_timeout_ticks()

```
void _Thread_queue_Add_timeout_ticks (
            Thread_queue_Queue * queue,
            Thread_Control * the_thread,
            Per_CPU_Control * cpu_self,
            Thread_queue_Context * queue_context )
```

Adds timeout ticks of the queue to the thread.

**Parameters**

| | | |
|---|---|---|
| | *queue* | This parameter is unused. |
| in,out | *the_thread* | The thread to add timeout ticks to. |
| | *cpu_self* | The cpu for the operation. |
| | *queue_context* | The thread queue context. |

Definition at line 17 of file threadqtimeout.c.

#### 8.171.4.6 _Thread_queue_Context_add_priority_update()

```
static __inline__ void _Thread_queue_Context_add_priority_update (
            Thread_queue_Context * queue_context,
            Thread_Control * the_thread )  [static]
```

Adds a priority update of the thread to the thread queue context.

**Parameters**

| | | |
|---|---|---|
| in,out | *queue_context* | The thread queue context to increase the priority update count of and set the_thread in its Priority update array. |
| | *the_thread* | The thread for the priority update. |

Definition at line 383 of file threadqimpl.h.

#### 8.171.4.7 _Thread_queue_Context_clear_priority_updates()

```
static __inline__ void _Thread_queue_Context_clear_priority_updates (
            Thread_queue_Context * queue_context )  [static]
```

Clears the priority update count of the thread queue context.

**Parameters**

| | | |
|---|---|---|
| out | *queue_context* | The thread queue context to clear the priority update count. |

Definition at line 338 of file threadqimpl.h.

#### 8.171.4.8 _Thread_queue_Context_initialize()

```
static __inline__ void _Thread_queue_Context_initialize (
            Thread_queue_Context * queue_context )  [static]
```

Initializes a thread queue context.

**Parameters**

| | | |
|---|---|---|
| out | *queue_context* | The thread queue context to initialize. |

Definition at line 152 of file threadqimpl.h.

#### 8.171.4.9 _Thread_queue_Context_restore_priority_updates()

```
static __inline__ void _Thread_queue_Context_restore_priority_updates (
            Thread_queue_Context * queue_context,
            size_t update_count )  [static]
```

Sets the priority update count of the thread queue context.

**Parameters**

| | | |
|---|---|---|
| out | *queue_context* | The thread queue context to set the priority update count of. |
| | *update_count* | The priority update count. |

Definition at line 367 of file threadqimpl.h.

#### 8.171.4.10 _Thread_queue_Context_save_priority_updates()

```
static __inline__ size_t _Thread_queue_Context_save_priority_updates (
            Thread_queue_Context * queue_context )  [static]
```

Returns the priority update count of the thread queue context.

**Parameters**

| | |
|---|---|
| *queue_context* | The thread queue context to get the priority update count of. |

**Returns**

The priority update count of *queue_context*.

Definition at line 353 of file threadqimpl.h.

#### 8.171.4.11 _Thread_queue_Context_set_deadlock_callout()

```
static __inline__ void _Thread_queue_Context_set_deadlock_callout (
            Thread_queue_Context * queue_context,
            Thread_queue_Deadlock_callout deadlock_callout )  [static]
```

Sets the deadlock callout in the thread queue context.

A deadlock callout must be provided for _Thread_queue_Enqueue() operations that operate on thread queues which may have an owner, e.g. mutex objects. Available deadlock callouts are _Thread_queue_Deadlock_status() and _Thread_queue_Deadlock_fatal().

**Parameters**

| | | |
|---|---|---|
| out | *queue_context* | The thread queue context. |
| | *deadlock_callout* | The deadlock callout. |

**See also**

_Thread_queue_Enqueue().

Definition at line 324 of file threadqimpl.h.

#### 8.171.4.12 _Thread_queue_Context_set_enqueue_callout()

```
static __inline__ void _Thread_queue_Context_set_enqueue_callout (
            Thread_queue_Context * queue_context,
            Thread_queue_Enqueue_callout enqueue_callout )  [static]
```

Sets the enqueue callout in the thread queue context.

**Parameters**

| | | |
|---|---|---|
| out | *queue_context* | The thread queue context. |
| | *enqueue_callout* | The enqueue callout. |

**See also**

_Thread_queue_Enqueue().

Definition at line 229 of file threadqimpl.h.

#### 8.171.4.13 _Thread_queue_Context_set_enqueue_do_nothing_extra()

```
static __inline__ void _Thread_queue_Context_set_enqueue_do_nothing_extra (
            Thread_queue_Context * queue_context )  [static]
```

Sets the do nothing enqueue callout in the thread queue context.

**Parameters**

| | | |
|---|---|---|
| out | *queue_context* | The thread queue context. |

**See also**

_Thread_queue_Enqueue().

Definition at line 245 of file threadqimpl.h.

#### 8.171.4.14 _Thread_queue_Context_set_enqueue_timeout_monotonic_timespec()

```
static __inline__ void _Thread_queue_Context_set_enqueue_timeout_monotonic_timespec (
            Thread_queue_Context * queue_context,
            const struct timespec * abstime )  [static]
```

Sets the enqueue callout to add an absolute monotonic timeout in timespec format.

**Parameters**

| | | |
|---|---|---|
| out | *queue_context* | The thread queue context. |
| | *abstime* | The absolute monotonic timeout. |

**See also**

_Thread_queue_Enqueue().

Definition at line 281 of file threadqimpl.h.

#### 8.171.4.15 _Thread_queue_Context_set_enqueue_timeout_realtime_timespec()

```
static __inline__ void _Thread_queue_Context_set_enqueue_timeout_realtime_timespec (
            Thread_queue_Context * queue_context,
            const struct timespec * abstime )  [static]
```

Sets the enqueue callout to add an absolute realtime timeout in timespec format.

**Parameters**

| | | |
|---|---|---|
| out | *queue_context* | The thread queue context. |
| | *abstime* | The absolute realtime timeout. |

**See also**

_Thread_queue_Enqueue().

Definition at line 301 of file threadqimpl.h.

#### 8.171.4.16 _Thread_queue_Context_set_enqueue_timeout_ticks()

```
static __inline__ void _Thread_queue_Context_set_enqueue_timeout_ticks (
            Thread_queue_Context * queue_context,
            Watchdog_Interval ticks )  [static]
```

Sets the enqueue callout to add a relative monotonic timeout in ticks.

**Parameters**

| | | |
|---|---|---|
| out | *queue_context* | The thread queue context. |
| | *ticks* | The timeout in ticks. |

**See also**

_Thread_queue_Enqueue().

Definition at line 262 of file threadqimpl.h.

#### 8.171.4.17 _Thread_queue_Context_set_ISR_level()

```
static __inline__ void _Thread_queue_Context_set_ISR_level (
            Thread_queue_Context * queue_context,
            ISR_Level level )  [static]
```

Sets the thread queue context ISR level.

**Parameters**

| | | |
|---|---|---|
| out | *queue_context* | The thread queue context to set the ISR level of. |
| | *level* | The ISR level to set *queue_context* to. |

Definition at line 411 of file threadqimpl.h.

#### 8.171.4.18 _Thread_queue_Context_set_thread_state()

```
static __inline__ void _Thread_queue_Context_set_thread_state (
            Thread_queue_Context * queue_context,
            States_Control thread_state )  [static]
```

Sets the thread state for the thread to enqueue in the thread queue context.

**Parameters**

| | | |
|---|---|---|
| out | *queue_context* | The thread queue context. |
| | *state* | The thread state. |

**See also**

_Thread_queue_Enqueue().

Definition at line 178 of file threadqimpl.h.

#### 8.171.4.19 _Thread_queue_Context_set_timeout_argument()

```
static __inline__ void _Thread_queue_Context_set_timeout_argument (
            Thread_queue_Context * queue_context,
            const void * arg )  [static]
```

Sets the timeout argument in the thread queue context.

**Parameters**

| | | |
|---|---|---|
| out | *queue_context* | The thread queue context. |
| | *arg* | The timeout argument. |

**See also**

_Thread_queue_Enqueue().

Definition at line 212 of file threadqimpl.h.

#### 8.171.4.20 _Thread_queue_Context_set_timeout_ticks()

```
static __inline__ void _Thread_queue_Context_set_timeout_ticks (
            Thread_queue_Context * queue_context,
            Watchdog_Interval ticks )  [static]
```

Sets the timeout ticks in the thread queue context.

**Parameters**

| | | |
|---|---|---|
| out | *queue_context* | The thread queue context. |
| | *ticks* | The timeout in ticks. |

**See also**

_Thread_queue_Enqueue().

Definition at line 195 of file threadqimpl.h.

#### 8.171.4.21 _Thread_queue_Deadlock_fatal()

```
void _Thread_queue_Deadlock_fatal (
            Thread_Control * the_thread )
```

Results in an INTERNAL_ERROR_THREAD_QUEUE_DEADLOCK fatal error.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread for the operation. |

Definition at line 375 of file threadqenqueue.c.

#### 8.171.4.22 _Thread_queue_Deadlock_status()

```
void _Thread_queue_Deadlock_status (
            Thread_Control * the_thread )
```

Sets the thread wait return code to STATUS_DEADLOCK.

**Parameters**

| | | |
|---|---|---|
| out | *the_thread* | The thread to set the wait return code to STATUS_DEADLOCK. |

Definition at line 370 of file threadqenqueue.c.

#### 8.171.4.23 _Thread_queue_Destroy()

```
static __inline__ void _Thread_queue_Destroy (
            Thread_queue_Control * the_thread_queue )  [static]
```

Destroys the thread queue.

**Parameters**

| | | |
|---|---|---|
| out | *the_thread_queue* | The thread queue to destroy. |

Definition at line 1378 of file threadqimpl.h.

#### 8.171.4.24 _Thread_queue_Dispatch_disable()

```
static __inline__ Per_CPU_Control* _Thread_queue_Dispatch_disable (
            Thread_queue_Context * queue_context )  [static]
```

Disables dispatching in a critical section.

**Parameters**

| | |
|---|---|
| *queue_context* | The thread queue context to get the lock context from. |

**Returns**

The current processor.

Definition at line 429 of file threadqimpl.h.

#### 8.171.4.25 _Thread_queue_Do_acquire_critical()

```
void _Thread_queue_Do_acquire_critical (
              Thread_queue_Control * the_thread_queue,
              ISR_lock_Context * lock_context )
```

Acquires the thread queue control in a critical section.

**Parameters**

| | | |
|---|---|---|
| | *the_thread_queue* | The thread queue control to acquire. |
| out | *lock_context* | The interrupt lock context. |

Definition at line 72 of file threadq.c.

#### 8.171.4.26 _Thread_queue_Do_dequeue()

```
Thread_Control* _Thread_queue_Do_dequeue (
              Thread_queue_Control * the_thread_queue,
              const Thread_queue_Operations * operations )
```

Dequeues the first thread waiting on the thread queue and returns it.

**Parameters**

| | |
|---|---|
| *the_thread_queue* | The thread queue for the operation. |
| *operations* | The thread queue operations. |

**Returns**

The first locked thread.

Definition at line 742 of file threadqenqueue.c.

#### 8.171.4.27 _Thread_queue_Do_release_critical()

```
void _Thread_queue_Do_release_critical (
              Thread_queue_Control * the_thread_queue,
              ISR_lock_Context * lock_context )
```

Checks if the thread queue control is the owner of the lock.

**Parameters**

| | |
|---|---|
| *the_thread_queue* | The thread queue control for the verification. |

**Return values**

| | |
|---|---|
| *true* | The thread queue control is the owner of the lock. |
| *false* | The thread queue control is not the owner of the lock. |

Releases the thread queue control in a critical section.

**Parameters**

| | | |
|---|---|---|
| | *the_thread_queue* | The thread queue control to release. |
| out | *lock_context* | The interrupt lock context. |

Definition at line 103 of file threadq.c.

### 8.171.4.28 _Thread_queue_Enqueue()

```
void _Thread_queue_Enqueue (
              Thread_queue_Queue * queue,
              const Thread_queue_Operations * operations,
              Thread_Control * the_thread,
              Thread_queue_Context * queue_context )
```

Blocks the thread and places it on the thread queue.

This enqueues the thread on the thread queue, blocks the thread, and optionally starts the thread timer in case the timeout discipline is not WATCHDOG_NO_TIMEOUT. Timeout discipline and value are in the queue_context.

The caller must be the owner of the thread queue lock. This function will release the thread queue lock and register it as the new thread lock. Thread dispatching is disabled before the thread queue lock is released. Thread dispatching is enabled once the sequence to block the thread is complete. The operation to enqueue the thread on the queue is protected by the thread queue lock. This makes it possible to use the thread queue lock to protect the state of objects embedding the thread queue and directly enter _Thread_queue_Enqueue() in case the thread must block.

The thread queue context must be set up with the following functions, otherwise the behaviour is unpredictable

- _Thread_queue_Context_set_thread_state(),
- _Thread_queue_Context_set_enqueue_callout() or _Thread_queue_Context_set_enqueue_do_nothing_extra() or _Thread_queue_Context_set_enqueue_timeout_ticks() or _Thread_queue_Context_set_enqueue_timeout_monotonic_time
  or _Thread_queue_Context_set_enqueue_timeout_realtime_timespec(),
- _Thread_queue_Context_set_deadlock_callout().

```
#include <rtems/score/threadqimpl.h>
#include <rtems/score/statesimpl.h>
#define MUTEX_TQ_OPERATIONS &_Thread_queue_Operations_priority
typedef struct {
  Thread_queue_Control Queue;
} Mutex;
void _Mutex_Obtain( Mutex *mutex )
{
  Thread_queue_Context  queue_context;
  Thread_Control       *executing;
  _Thread_queue_Context_initialize( &queue_context );
  _Thread_queue_Acquire( &mutex->Queue, queue_context );
  executing = _Thread_Executing;
  if ( mutex->Queue.owner == NULL ) {
```

```
    mutex->Queue.owner = executing;
    _Thread_queue_Release( &mutex->Queue, queue_context );
  } else {
    _Thread_queue_Context_set_thread_state(
      &queue_context,
      STATES_WAITING_FOR_MUTEX
    );
    _Thread_queue_Context_set_enqueue_do_nothing_extra( &queue_context );
    _Thread_queue_Context_set_deadlock_callout(
      queue_context,
      _Thread_queue_Deadlock_fatal
    );
    _Thread_queue_Enqueue(
      &mutex->Queue.Queue,
      MUTEX_TQ_OPERATIONS,
      executing,
      &queue_context
    );
  }
}
```

**Parameters**

| | | |
|---|---|---|
| | *queue* | The actual thread queue. |
| | *operations* | The thread queue operations. |
| in,out | *the_thread* | The thread to enqueue. |
| in,out | *queue_context* | The thread queue context of the lock acquire. |

Definition at line 380 of file threadqenqueue.c.

### 8.171.4.29 _Thread_queue_Enqueue_do_nothing_extra()

```
void _Thread_queue_Enqueue_do_nothing_extra (
              Thread_queue_Queue * queue,
              Thread_Control * the_thread,
              Per_CPU_Control * cpu_self,
              Thread_queue_Context * queue_context )
```

Does nothing.

**Parameters**

| | |
|---|---|
| *queue* | This parameter is unused. |
| *the_thread* | This parameter is unused. |
| *cpu_self* | This parameter is unused. |
| *queue_context* | This parameter is unused. |

Definition at line 360 of file threadqenqueue.c.

### 8.171.4.30 _Thread_queue_Enqueue_sticky()

```
Status_Control _Thread_queue_Enqueue_sticky (
              Thread_queue_Queue * queue,
```

```
            const Thread_queue_Operations * operations,
            Thread_Control * the_thread,
            Thread_queue_Context * queue_context )
```

Enqueues the thread on the thread queue and busy waits for dequeue.

Optionally starts the thread timer in case the timeout discipline is not WATCHDOG_NO_TIMEOUT. Timeout discipline and value are in the queue_context.

The caller must be the owner of the thread queue lock. This function will release the thread queue lock and register it as the new thread lock.

The thread priorities of the owner and the are updated with respect to the scheduler. The sticky level of the thread is incremented. A thread dispatch is performed if necessary.

Afterwards, the thread busy waits on the thread wait flags until a timeout occurs or the thread queue is surrendered to this thread. So, it sticks to the processor instead of blocking with respect to the scheduler.

**Parameters**

| | | |
|---|---|---|
| | *queue* | The actual thread queue. |
| | *operations* | The thread queue operations. |
| in,out | *the_thread* | The thread to enqueue. |
| in,out | *queue_context* | The thread queue context of the lock acquire. |

Definition at line 456 of file threadqenqueue.c.

#### 8.171.4.31 _Thread_queue_Extract()

```
void _Thread_queue_Extract (
            Thread_Control * the_thread )
```

Extracts thread from thread queue.

This routine removes *the_thread* its thread queue and cancels any timeouts associated with this blocking.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The pointer to a thread control block that is to be removed |

Definition at line 624 of file threadqenqueue.c.

#### 8.171.4.32 _Thread_queue_Extract_critical()

```
void _Thread_queue_Extract_critical (
            Thread_queue_Queue * queue,
            const Thread_queue_Operations * operations,
```

```
              Thread_Control * the_thread,
              Thread_queue_Context * queue_context )
```

Extracts the thread from the thread queue and unblocks it.

The caller must be the owner of the thread queue lock. This function will release the thread queue lock and restore the default thread lock. Thread dispatching is disabled before the thread queue lock is released and an unblock is necessary. Thread dispatching is enabled once the sequence to unblock the thread is complete. This makes it possible to use the thread queue lock to protect the state of objects embedding the thread queue and directly enter _Thread_queue_Extract_critical() to finalize an operation in case a waiting thread exists.

```
#include <rtems/score/threadqimpl.h>
typedef struct {
  Thread_queue_Control  Queue;
  Thread_Control       *owner;
} Mutex;
void _Mutex_Release( Mutex *mutex )
{
  Thread_queue_Context  queue_context;
  Thread_Control       *first;
  _Thread_queue_Context_initialize( &queue_context, NULL );
  _Thread_queue_Acquire( &mutex->Queue, queue_context );
  first = _Thread_queue_First_locked( &mutex->Queue );
  mutex->owner = first;
  if ( first != NULL ) {
    _Thread_queue_Extract_critical(
      &mutex->Queue.Queue,
      mutex->Queue.operations,
      first,
      &queue_context
  );
}
```

**Parameters**

| | | |
|---|---|---|
| | *queue* | The actual thread queue. |
| | *operations* | The thread queue operations. |
| in,out | *the_thread* | The thread to extract. |
| in,out | *queue_context* | The thread queue context of the lock acquire. |

Definition at line 600 of file threadqenqueue.c.

### 8.171.4.33 _Thread_queue_Extract_locked()

```
bool _Thread_queue_Extract_locked (
              Thread_queue_Queue * queue,
              const Thread_queue_Operations * operations,
              Thread_Control * the_thread,
              Thread_queue_Context * queue_context )
```

Extracts the thread from the thread queue, restores the default wait operations and restores the default thread lock.

The caller must be the owner of the thread queue lock. The thread queue lock is not released.

**Parameters**

| | | |
|---|---|---|
| | *queue* | The actual thread queue. |
| | *operations* | The thread queue operations. |
| in,out | *the_thread* | The thread to extract. |
| in,out | *queue_context* | The thread queue context. |

**Returns**

Returns the unblock indicator for _Thread_queue_Unblock_critical(). True indicates, that this thread must be unblocked by the scheduler later in _Thread_queue_Unblock_critical(), and false otherwise. In case false is returned, then the thread queue enqueue procedure was interrupted. Thus it will unblock itself and the thread wait information is no longer accessible, since this thread may already block on another resource in an SMP configuration.

Definition at line 565 of file threadqenqueue.c.

#### 8.171.4.34 _Thread_queue_Extract_with_proxy()

```
void _Thread_queue_Extract_with_proxy (
            Thread_Control * the_thread )
```

Extracts the_thread from the_thread_queue.

This routine extracts the_thread from the_thread_queue and ensures that if there is a proxy for this task on another node, it is also dealt with.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The pointer to a thread control block that is to be removed |

Extracts the_thread from the_thread_queue.

This routine extracts the_thread from the_thread_queue and ensures that if there is a proxy for this task on another node, it is also dealt with. A proxy is a data data that is on the thread queue on the remote node and acts as a proxy for the local thread. If the local thread was waiting on a remote operation, then the remote side of the operation must be cleaned up.

Definition at line 30 of file threadqextractwithproxy.c.

#### 8.171.4.35 _Thread_queue_First()

```
Thread_Control* _Thread_queue_First (
            Thread_queue_Control * the_thread_queue,
            const Thread_queue_Operations * operations )
```

Returns the first thread on the thread queue if it exists, otherwise NULL.

**Parameters**

| | |
|---|---|
| *the_thread_queue* | The thread queue. |

**Return values**

| | |
|---|---|
| *first* | The first thread on the thread queue according to the enqueue order. |

**Return values**

| | |
|---|---|
| *NULL* | No thread is present on the thread queue. |

### 8.171.4.36 _Thread_queue_First_locked()

```
static __inline__ Thread_Control* _Thread_queue_First_locked (
            Thread_queue_Control * the_thread_queue,
            const Thread_queue_Operations * operations )  [static]
```

Returns the first thread on the thread queue if it exists, otherwise `NULL`.

The caller must be the owner of the thread queue lock. The thread queue lock is not released.

**Parameters**

| | |
|---|---|
| *the_thread_queue* | The thread queue. |
| *operations* | The thread queue operations. |

**Return values**

| | |
|---|---|
| *first* | The first thread on the thread queue according to the enqueue order. |
| *NULL* | No thread is present on the thread queue. |

Definition at line 1173 of file threadqimpl.h.

### 8.171.4.37 _Thread_queue_Flush_critical()

```
size_t _Thread_queue_Flush_critical (
            Thread_queue_Queue * queue,
            const Thread_queue_Operations * operations,
            Thread_queue_Flush_filter filter,
            Thread_queue_Context * queue_context )
```

Unblocks all threads enqueued on the thread queue.

This function iteratively extracts the first enqueued thread of the thread queue until the thread queue is empty or the filter function indicates a stop. The thread timers of the extracted threads are cancelled. The extracted threads are unblocked.

**Parameters**

| | |
|---|---|
| *queue* | The actual thread queue. |
| *operations* | The thread queue operations. |

**Parameters**

| | |
|---|---|
| *filter* | The filter functions is called for each thread to extract from the thread queue. It may be used to alter the thread under protection of the thread queue lock, for example to set the thread wait return code. The return value of the filter function controls if the thread queue flush operation should stop or continue. |
| *queue_context* | The thread queue context of the lock acquire. May be used to pass additional data to the filter function via an overlay structure. The filter function should not release or acquire the thread queue lock. |

**Returns**

The count of extracted threads.

Definition at line 63 of file threadqflush.c.

### 8.171.4.38 _Thread_queue_Flush_default_filter()

```
Thread_Control* _Thread_queue_Flush_default_filter (
              Thread_Control * the_thread,
              Thread_queue_Queue * queue,
              Thread_queue_Context * queue_context )
```

Default thread queue flush filter function.

**Parameters**

| | |
|---|---|
| *the_thread* | The thread to extract. |
| *queue* | This parameter is unused. |
| *queue_context* | This parameter is unused. |

**Return values**

| | |
|---|---|
| *the_thread* | Extract this thread. |

Definition at line 26 of file threadqflush.c.

### 8.171.4.39 _Thread_queue_Flush_status_object_was_deleted()

```
Thread_Control* _Thread_queue_Flush_status_object_was_deleted (
              Thread_Control * the_thread,
              Thread_queue_Queue * queue,
              Thread_queue_Context * queue_context )
```

Status object was deleted thread queue flush filter function.

Sets the thread wait return code of the thread to STATUS_OBJECT_WAS_DELETED

**Parameters**

| | | |
|---|---|---|
| out | *the_thread* | The thread to extract. |
| | *queue* | This parameter is unused. |
| | *queue_context* | This parameter is unused. |

**Return values**

| | |
|---|---|
| *the_thread* | Extract this thread. |

Definition at line 37 of file threadqflush.c.

#### 8.171.4.40 _Thread_queue_Flush_status_unavailable()

```
Thread_Control* _Thread_queue_Flush_status_unavailable (
            Thread_Control * the_thread,
            Thread_queue_Queue * queue,
            Thread_queue_Context * queue_context )
```

Status unavailable thread queue flush filter function.

Sets the thread wait return code of the thread to STATUS_UNAVAILABLE.

**Parameters**

| | | |
|---|---|---|
| out | *the_thread* | The thread to extract. |
| | *queue* | This parameter is unused. |
| | *queue_context* | This parameter is unused. |

**Return values**

| | |
|---|---|
| *the_thread* | Extract this thread. |

Definition at line 50 of file threadqflush.c.

#### 8.171.4.41 _Thread_queue_Gate_add()

```
static __inline__ void _Thread_queue_Gate_add (
            Chain_Control * chain,
            Thread_queue_Gate * gate )  [static]
```

Adds the gate to the chain.

**Parameters**

| | | |
|---|---|---|
| in,out | *chain* | The chain to add the gate to. |
| | *gate* | The gate to add to the chain. |

Definition at line 481 of file threadqimpl.h.

#### 8.171.4.42 _Thread_queue_Gate_close()

```
static __inline__ void _Thread_queue_Gate_close (
            Thread_queue_Gate * gate )  [static]
```

Closes the gate.

**Parameters**

| | | |
|---|---|---|
| out | *gate* | The gate to close. |

Definition at line 468 of file threadqimpl.h.

#### 8.171.4.43 _Thread_queue_Gate_open()

```
static __inline__ void _Thread_queue_Gate_open (
            Thread_queue_Gate * gate )  [static]
```

Opens the gate.

**Parameters**

| | | |
|---|---|---|
| out | *gate* | The gate to open. |

Definition at line 494 of file threadqimpl.h.

#### 8.171.4.44 _Thread_queue_Gate_wait()

```
static __inline__ void _Thread_queue_Gate_wait (
            Thread_queue_Gate * gate )  [static]
```

Waits on a gate to open.

Performs busy waiting.

**Parameters**

| | |
|---|---|
| *gate* | The gate to wait for. |

Definition at line 508 of file threadqimpl.h.

#### 8.171.4.45 _Thread_queue_Heads_initialize()

```
static __inline__ void _Thread_queue_Heads_initialize (
            Thread_queue_Heads * heads )  [static]
```

Initializes the thread queue heads.

**Parameters**

| | | |
|---|---|---|
| out | *heads* | The thread queue heads to initialize. |

Definition at line 523 of file threadqimpl.h.

#### 8.171.4.46 _Thread_queue_Initialize()

```
void _Thread_queue_Initialize (
            Thread_queue_Control * the_thread_queue,
            const char * name )
```

Initializes the thread queue control to the given name.

**Parameters**

| | | |
|---|---|---|
| out | *the_thread_queue* | The thread queue control to initialize. |
| | *name* | The name for *the_thread_queue*. |

Definition at line 137 of file threadq.c.

#### 8.171.4.47 _Thread_queue_Is_empty()

```
static __inline__ bool _Thread_queue_Is_empty (
            const Thread_queue_Queue * queue )  [static]
```

Checks if the thread queue queue is empty.

**Parameters**

| | |
|---|---|
| *queue* | The thread queue for the verification. |

**Return values**

| | |
|---|---|
| *true* | *queue* is empty. |
| *false* | *queue* is not empty. |

Definition at line 1152 of file threadqimpl.h.

#### 8.171.4.48 _Thread_queue_Object_initialize()

```
void _Thread_queue_Object_initialize (
              Thread_queue_Control * the_thread_queue )
```

Initializes a thread queue embedded in an object with identifier.

The object must have the layout specified by Thread_queue_Object. It should be ensured with the THREAD_QU↩EUE_OBJECT_ASSERT() static assertion.

**Parameters**

| | | |
|---|---|---|
| out | *the_thread_queue* | The thread queue. |

Definition at line 148 of file threadq.c.

#### 8.171.4.49 _Thread_queue_Path_acquire_critical()

```
bool _Thread_queue_Path_acquire_critical (
              Thread_queue_Queue * queue,
              Thread_Control * the_thread,
              Thread_queue_Context * queue_context )
```

Does nothing.

**Parameters**

| | |
|---|---|
| *the_proxy* | This parameter is unused. |
| *mp_id* | This parameter is unused. |

Acquires the thread queue path in a critical section.

**Parameters**

| | |
|---|---|
| *queue* | The thread queue queue. |
| *the_thread* | The thread for the operation. |
| *queue_context* | The thread queue context. |

**Return values**

| | |
|---|---|
| *true* | The operation was successful. |
| *false* | The operation failed. |

Definition at line 255 of file threadqenqueue.c.

#### 8.171.4.50 _Thread_queue_Path_release_critical()

```
void _Thread_queue_Path_release_critical (
            Thread_queue_Context * queue_context )
```

Releases the thread queue path in a critical section.

**Parameters**

| | |
|---|---|
| *queue_context* | The thread queue context. |

Definition at line 170 of file threadqenqueue.c.

#### 8.171.4.51 _Thread_queue_Queue_do_acquire_critical()

```
static __inline__ void _Thread_queue_Queue_do_acquire_critical (
            Thread_queue_Queue * queue,
            ISR_lock_Context * lock_context )  [static]
```

Acquires the thread queue queue in a critical section.

**Parameters**

| | | |
|---|---|---|
| | *queue* | The thread queue queue to acquire in a critical section. |
| | *lock_stats* | The lock statistics. |
| out | *lock_context* | The interrupt lock context. |

Definition at line 567 of file threadqimpl.h.

#### 8.171.4.52 _Thread_queue_Queue_get_name_and_id()

```
size_t _Thread_queue_Queue_get_name_and_id (
              const Thread_queue_Queue * queue,
              char * buffer,
              size_t buffer_size,
              Objects_Id * id )
```

Copies the thread queue name to the specified buffer.

**Parameters**

| | | |
|---|---|---|
| | *queue* | The actual thread queue. |
| out | *buffer* | The buffer for the thread queue name copy. |
| | *buffer_size* | The buffer size in characters. |
| | *id* | The object identifier in case the thread queue is embedded in an object with identifier, otherwise it is set to 0. |

**Returns**

The length of the thread queue name. May be greater than or equal to the buffer size if truncation occurred.

Definition at line 163 of file threadq.c.

#### 8.171.4.53 _Thread_queue_Queue_initialize()

```
static __inline__ void _Thread_queue_Queue_initialize (
              Thread_queue_Queue * queue,
              const char * name )  [static]
```

Initializes the thread queue queue with the given name.

**Parameters**

| | | |
|---|---|---|
| out | *queue* | The thread queue queue to initialize. |
| | *name* | The name for the *queue*. |

Definition at line 547 of file threadqimpl.h.

#### 8.171.4.54 _Thread_queue_Queue_release()

```
static __inline__ void _Thread_queue_Queue_release (
              Thread_queue_Queue * queue,
              ISR_lock_Context * lock_context )  [static]
```

Releases the thread queue queue and enables interrupts.

**Parameters**

| | | |
|---|---|---|
| | *queue* | The thread queue queue to release. |
| out | *lock_context* | The interrupt lock context to enable interrupts. |

Definition at line 625 of file threadqimpl.h.

#### 8.171.4.55 _Thread_queue_Queue_release_critical()

```
static __inline__ void _Thread_queue_Queue_release_critical (
            Thread_queue_Queue * queue,
            ISR_lock_Context * lock_context )  [static]
```

Releases the thread queue queue in a critical section.

**Parameters**

| | | |
|---|---|---|
| | *queue* | The thread queue queue to release in a critical section. |
| out | *lock_context* | The interrupt lock context. |

Definition at line 603 of file threadqimpl.h.

#### 8.171.4.56 _Thread_queue_Release()

```
void _Thread_queue_Release (
            Thread_queue_Control * the_thread_queue,
            Thread_queue_Context * queue_context )
```

Releases the thread queue control and enables interrupts.

**Parameters**

| | | |
|---|---|---|
| | *the_thread_queue* | The thread queue control to release. |
| out | *queue_context* | The thread queue context. |

Definition at line 118 of file threadq.c.

#### 8.171.4.57 _Thread_queue_Release_critical()

```
static __inline__ void _Thread_queue_Release_critical (
            Thread_queue_Control * the_thread_queue,
            Thread_queue_Context * queue_context )  [static]
```

Releases the thread queue control in a critical section.

**Parameters**

| | | |
|---|---|---|
| | *the_thread_queue* | The thread queue control to release. |
| out | *queue_context* | The thread queue context. |

Definition at line 764 of file threadqimpl.h.

### 8.171.4.58 _Thread_queue_Surrender()

```
void _Thread_queue_Surrender (
              Thread_queue_Queue * queue,
              Thread_queue_Heads * heads,
              Thread_Control * previous_owner,
              Thread_queue_Context * queue_context,
              const Thread_queue_Operations * operations )
```

Surrenders the thread queue previously owned by the thread to the first enqueued thread.

The owner of the thread queue must be set to NULL by the caller.

This function releases the thread queue lock. In addition it performs a thread dispatch if necessary.

**Parameters**

| | | |
|---|---|---|
| in,out | *queue* | The actual thread queue. |
| | *heads* | The thread queue heads. It must not be NULL. |
| | *previous_owner* | The previous owner thread surrendering the thread queue. |
| | *queue_context* | The thread queue context of the lock acquire. |
| | *operations* | The thread queue operations. |

Definition at line 660 of file threadqenqueue.c.

### 8.171.4.59 _Thread_queue_Surrender_sticky()

```
void _Thread_queue_Surrender_sticky (
              Thread_queue_Queue * queue,
              Thread_queue_Heads * heads,
              Thread_Control * previous_owner,
              Thread_queue_Context * queue_context,
              const Thread_queue_Operations * operations )
```

Surrenders the thread queue previously owned by the thread to the first enqueued thread.

The owner of the thread queue must be set to NULL by the caller.

The caller must be the owner of the thread queue lock. This function will release the thread queue.

The thread priorities of the previous owner and the new owner are updated. The sticky level of the previous owner is decremented. A thread dispatch is performed if necessary.

**Parameters**

| | | |
|---|---|---|
| in,out | *queue* | The actual thread queue. |
| | *heads* | The thread queue heads. It must not be NULL. |
| in,out | *previous_owner* | The previous owner thread surrendering the thread queue. |
| | *queue_context* | The thread queue context of the lock acquire. |
| | *operations* | The thread queue operations. |

Definition at line 708 of file threadqenqueue.c.

#### 8.171.4.60 _Thread_queue_Unblock_critical()

```
void _Thread_queue_Unblock_critical (
              bool unblock,
              Thread_queue_Queue * queue,
              Thread_Control * the_thread,
              ISR_lock_Context * lock_context )
```

Unblocks the thread which was on the thread queue before.

The caller must be the owner of the thread queue lock. This function will release the thread queue lock. Thread dispatching is disabled before the thread queue lock is released and an unblock is necessary. Thread dispatching is enabled once the sequence to unblock the thread is complete.

**Parameters**

| | | |
|---|---|---|
| | *unblock* | The unblock indicator returned by _Thread_queue_Extract_locked(). |
| | *queue* | The actual thread queue. |
| in,out | *the_thread* | The thread to extract. |
| in,out | *lock_context* | The lock context of the lock acquire. |

Definition at line 579 of file threadqenqueue.c.

### 8.171.5 Variable Documentation

#### 8.171.5.1 _Thread_queue_Object_name

```
const char _Thread_queue_Object_name[]  [extern]
```

The special thread queue name to indicated that the thread queue is embedded in an object with identifier.

**See also**

_Thread_queue_Object_initialize().

Definition at line 135 of file threadq.c.

# 8.172 Thread States

SuperCore Thread States.

## Files

- file states.h

  *Thread Execution State Information.*
- file statesimpl.h

  *Inlined Routines Associated with Thread State Information.*

## Macros

- #define STATES_READY 0x00000000
- #define STATES_WAITING_FOR_MUTEX 0x00000001
- #define STATES_WAITING_FOR_SEMAPHORE 0x00000002
- #define STATES_WAITING_FOR_EVENT 0x00000004
- #define STATES_WAITING_FOR_SYSTEM_EVENT 0x00000008
- #define STATES_WAITING_FOR_MESSAGE 0x00000010
- #define STATES_WAITING_FOR_CONDITION_VARIABLE 0x00000020
- #define STATES_WAITING_FOR_FUTEX 0x00000040
- #define STATES_WAITING_FOR_BSD_WAKEUP 0x00000080
- #define STATES_WAITING_FOR_TIME 0x00000100

  *This macro corresponds to a task which is waiting for a relative or absolute timeout.*
- #define STATES_WAITING_FOR_PERIOD 0x00000200
- #define STATES_WAITING_FOR_SIGNAL 0x00000400
- #define STATES_WAITING_FOR_BARRIER 0x00000800
- #define STATES_WAITING_FOR_RWLOCK 0x00001000
- #define STATES_WAITING_FOR_JOIN_AT_EXIT 0x00002000
- #define STATES_WAITING_FOR_JOIN 0x00004000
- #define STATES_SUSPENDED 0x00008000
- #define STATES_WAITING_FOR_SEGMENT 0x00010000
- #define STATES_LIFE_IS_CHANGING 0x00020000
- #define STATES_DEBUGGER 0x08000000
- #define STATES_INTERRUPTIBLE_BY_SIGNAL 0x10000000
- #define STATES_WAITING_FOR_RPC_REPLY 0x20000000
- #define STATES_ZOMBIE 0x40000000
- #define STATES_DORMANT 0x80000000
- #define STATES_LOCALLY_BLOCKED
- #define STATES_BLOCKED
- #define STATES_ALL_SET 0xffffffff

## Typedefs

- typedef uint32_t States_Control

## Functions

- static __inline__ States_Control _States_Set (States_Control states_to_set, States_Control current_state)

  *Returns the result of setting the states to set to the given state.*
- static __inline__ States_Control _States_Clear (States_Control states_to_clear, States_Control current_↩ state)

  *Clears the states into the passed current state and returns the result.*
- static __inline__ bool _States_Is_ready (States_Control the_states)

  *Checks if the state is ready.*
- static __inline__ bool _States_Is_dormant (States_Control the_states)

  *Checks if DORMANT state is set.*
- static __inline__ bool _States_Is_suspended (States_Control the_states)

  *Checks if SUSPENDED state is set.*
- static __inline__ bool _States_Is_waiting_for_rpc_reply (States_Control the_states)

  *Checks if WAITING_FOR_TIME state is set.*
- static __inline__ bool _States_Is_waiting_for_join_at_exit (States_Control the_states)

  *Checks if WAITING_FOR_JOIN_AT_EXIT state is set.*
- static __inline__ bool _States_Is_interruptible_by_signal (States_Control the_states)

  *Checks if the state is set to be interruptible.*
- static __inline__ bool _States_Is_locally_blocked (States_Control the_states)

  *Checks if the state is blocked waiting on a local resource.*

### 8.172.1 Detailed Description

SuperCore Thread States.

This handler encapsulates functionality which relates to the management of the state bitmap associated with each thread.

### 8.172.2 Macro Definition Documentation

#### 8.172.2.1 STATES_ALL_SET

```
#define STATES_ALL_SET 0xffffffff
```

All state bits set to one (provided for _Thread_Start())

Definition at line 144 of file statesimpl.h.

#### 8.172.2.2 STATES_BLOCKED

```
#define STATES_BLOCKED
```

**Value:**

```
( STATES_LOCALLY_BLOCKED          | \
STATES_WAITING_FOR_TIME         | \
STATES_WAITING_FOR_PERIOD       | \
STATES_WAITING_FOR_EVENT        | \
STATES_WAITING_FOR_RPC_REPLY    | \
STATES_WAITING_FOR_SYSTEM_EVENT | \
STATES_INTERRUPTIBLE_BY_SIGNAL )
```

This macro corresponds to a task waiting which is blocked.

Definition at line 135 of file statesimpl.h.

#### 8.172.2.3 STATES_DEBUGGER

```
#define STATES_DEBUGGER 0x08000000
```

This macro corresponds to a task being held by the debugger.

Definition at line 105 of file statesimpl.h.

#### 8.172.2.4 STATES_DORMANT

```
#define STATES_DORMANT 0x80000000
```

This macro corresponds to a task being created but not yet started.

Definition at line 119 of file statesimpl.h.

#### 8.172.2.5 STATES_INTERRUPTIBLE_BY_SIGNAL

```
#define STATES_INTERRUPTIBLE_BY_SIGNAL 0x10000000
```

This macro corresponds to a task which is in an interruptible blocking state.

Definition at line 110 of file statesimpl.h.

#### 8.172.2.6 STATES_LIFE_IS_CHANGING

```
#define STATES_LIFE_IS_CHANGING 0x00020000
```

This macro corresponds to a task those life is changing.

Definition at line 102 of file statesimpl.h.

#### 8.172.2.7 STATES_LOCALLY_BLOCKED

```
#define STATES_LOCALLY_BLOCKED
```

**Value:**

```
( STATES_WAITING_FOR_SEGMENT            | \
STATES_WAITING_FOR_MESSAGE            | \
STATES_WAITING_FOR_SEMAPHORE          | \
STATES_WAITING_FOR_MUTEX              | \
STATES_WAITING_FOR_CONDITION_VARIABLE | \
STATES_WAITING_FOR_JOIN               | \
STATES_WAITING_FOR_SIGNAL             | \
STATES_WAITING_FOR_BARRIER            | \
STATES_WAITING_FOR_BSD_WAKEUP         | \
STATES_WAITING_FOR_FUTEX              | \
STATES_WAITING_FOR_RWLOCK            )
```

This macro corresponds to a task waiting for a local object operation.

Definition at line 122 of file statesimpl.h.

#### 8.172.2.8 STATES_READY

```
#define STATES_READY 0x00000000
```

This macro corresponds to a task being ready.

Definition at line 45 of file statesimpl.h.

#### 8.172.2.9 STATES_SUSPENDED

```
#define STATES_SUSPENDED 0x00008000
```

This macro corresponds to a task being suspended.

Definition at line 96 of file statesimpl.h.

#### 8.172.2.10 STATES_WAITING_FOR_BARRIER

```
#define STATES_WAITING_FOR_BARRIER 0x00000800
```

This macro corresponds to a task waiting for a barrier.

Definition at line 84 of file statesimpl.h.

#### 8.172.2.11 STATES_WAITING_FOR_BSD_WAKEUP

```
#define STATES_WAITING_FOR_BSD_WAKEUP 0x00000080
```

This macro corresponds to a task waiting for BSD wakeup.

Definition at line 69 of file statesimpl.h.

#### 8.172.2.12 STATES_WAITING_FOR_CONDITION_VARIABLE

```
#define STATES_WAITING_FOR_CONDITION_VARIABLE 0x00000020
```

This macro corresponds to a task waiting for a condition variable.

Definition at line 63 of file statesimpl.h.

#### 8.172.2.13 STATES_WAITING_FOR_EVENT

```
#define STATES_WAITING_FOR_EVENT 0x00000004
```

This macro corresponds to a task waiting for an event.

Definition at line 54 of file statesimpl.h.

#### 8.172.2.14 STATES_WAITING_FOR_FUTEX

```
#define STATES_WAITING_FOR_FUTEX 0x00000040
```

This macro corresponds to a task waiting for a futex.

Definition at line 66 of file statesimpl.h.

#### 8.172.2.15 STATES_WAITING_FOR_JOIN

```
#define STATES_WAITING_FOR_JOIN 0x00004000
```

This macro corresponds to a task waiting for a join.

Definition at line 93 of file statesimpl.h.

#### 8.172.2.16 STATES_WAITING_FOR_JOIN_AT_EXIT

```
#define STATES_WAITING_FOR_JOIN_AT_EXIT 0x00002000
```

This macro corresponds to a task waiting for a join while exiting.

Definition at line 90 of file statesimpl.h.

#### 8.172.2.17 STATES_WAITING_FOR_MESSAGE

```
#define STATES_WAITING_FOR_MESSAGE 0x00000010
```

This macro corresponds to a task waiting for a message.

Definition at line 60 of file statesimpl.h.

#### 8.172.2.18 STATES_WAITING_FOR_MUTEX

```
#define STATES_WAITING_FOR_MUTEX 0x00000001
```

This macro corresponds to a task waiting for a mutex.

Definition at line 48 of file statesimpl.h.

#### 8.172.2.19 STATES_WAITING_FOR_PERIOD

```
#define STATES_WAITING_FOR_PERIOD 0x00000200
```

This macro corresponds to a task waiting for a period.

Definition at line 78 of file statesimpl.h.

#### 8.172.2.20 STATES_WAITING_FOR_RPC_REPLY

```
#define STATES_WAITING_FOR_RPC_REPLY 0x20000000
```

This macro corresponds to a task waiting for a reply to an MPCI request.

Definition at line 113 of file statesimpl.h.

#### 8.172.2.21 STATES_WAITING_FOR_RWLOCK

```
#define STATES_WAITING_FOR_RWLOCK 0x00001000
```

This macro corresponds to a task waiting for a RWLock.

Definition at line 87 of file statesimpl.h.

#### 8.172.2.22 STATES_WAITING_FOR_SEGMENT

```
#define STATES_WAITING_FOR_SEGMENT 0x00010000
```

This macro corresponds to a task waiting for a fixed size segment.

Definition at line 99 of file statesimpl.h.

#### 8.172.2.23 STATES_WAITING_FOR_SEMAPHORE

```
#define STATES_WAITING_FOR_SEMAPHORE 0x00000002
```

This macro corresponds to a task waiting for a semaphore.

Definition at line 51 of file statesimpl.h.

#### 8.172.2.24 STATES_WAITING_FOR_SIGNAL

```
#define STATES_WAITING_FOR_SIGNAL 0x00000400
```

This macro corresponds to a task waiting for a signal.

Definition at line 81 of file statesimpl.h.

#### 8.172.2.25 STATES_WAITING_FOR_SYSTEM_EVENT

```
#define STATES_WAITING_FOR_SYSTEM_EVENT 0x00000008
```

This macro corresponds to a task waiting for a system event.

Definition at line 57 of file statesimpl.h.

#### 8.172.2.26 STATES_ZOMBIE

```
#define STATES_ZOMBIE 0x40000000
```

This macro corresponds to a task being a zombie.

Definition at line 116 of file statesimpl.h.

### 8.172.3 Typedef Documentation

#### 8.172.3.1 States_Control

```
typedef uint32_t States_Control
```

The following type defines the control block used to manage a thread's state.

Definition at line 46 of file states.h.

### 8.172.4 Function Documentation

#### 8.172.4.1 _States_Clear()

```
static __inline__ States_Control _States_Clear (
            States_Control states_to_clear,
            States_Control current_state )  [static]
```

Clears the states into the passed current state and returns the result.

This function clears the given *states_to_clear* into the *current_state* passed in. The result is returned to the user in *current_state*.

**Parameters**

| | |
|---|---|
| *states_to_clear* | The state bits to clear. |
| *current_state* | The state set to remove them from. |

**Returns**

This method returns the updated states value.

Definition at line 176 of file statesimpl.h.

### 8.172.4.2 _States_Is_dormant()

```
static __inline__ bool _States_Is_dormant (
            States_Control the_states )  [static]
```

Checks if DORMANT state is set.

This function returns true if the DORMANT state is set in *the_states*, and false otherwise.

**Parameters**

| | |
|---|---|
| *the_states* | The task state set to test. |

**Return values**

| | |
|---|---|
| *true* | DORMANT state is set in *the_states*. |
| *false* | DORMANT state is not set in *the_states*. |

Definition at line 213 of file statesimpl.h.

### 8.172.4.3 _States_Is_interruptible_by_signal()

```
static __inline__ bool _States_Is_interruptible_by_signal (
            States_Control the_states )  [static]
```

Checks if the state is set to be interruptible.

This function returns true if the task's state is set in way that allows it to be interrupted by a signal.

**Parameters**

| | |
|---|---|
| *the_states* | The task state set to test. |

**Return values**

| | |
|---|---|
| *true* | *the_states* is interruptible. |
| *false* | *the_states* is not interruptible. |

Definition at line 285 of file statesimpl.h.

#### 8.172.4.4 _States_Is_locally_blocked()

```
static __inline__ bool _States_Is_locally_blocked (
            States_Control the_states )  [static]
```

Checks if the state is blocked waiting on a local resource.

This function returns true if one of the states which indicates that a task is blocked waiting for a local resource is set in the_states, and false otherwise.

**Parameters**

| | |
|---|---|
| *the_states* | The task state set to test. |

**Return values**

| | |
|---|---|
| *true* | The state indicates that the task is blocked waiting on a local resource. |
| *false* | The state indicates that the task is not blocked waiting on a local resource. |

Definition at line 306 of file statesimpl.h.

#### 8.172.4.5 _States_Is_ready()

```
static __inline__ bool _States_Is_ready (
            States_Control the_states )  [static]
```

Checks if the state is ready.

This function returns true if the_states indicates that the state is READY, and false otherwise.

**Parameters**

| | |
|---|---|
| *the_states* | The task state set to test. |

**Return values**

| | |
|---|---|
| *true* | The state is ready. |
| *false* | The state is not ready. |

Definition at line 195 of file statesimpl.h.

#### 8.172.4.6 _States_Is_suspended()

```
static __inline__ bool _States_Is_suspended (
            States_Control the_states )  [static]
```

Checks if SUSPENDED state is set.

This function returns true if the SUSPENDED state is set in *the_states*, and false otherwise.

**Parameters**

| | |
|---|---|
| *the_states* | The task state set to test. |

**Return values**

| | |
|---|---|
| *true* | SUSPENDED state is set in *the_states*. |
| *false* | SUSPENDED state is not set in *the_states*. |

Definition at line 231 of file statesimpl.h.

#### 8.172.4.7 _States_Is_waiting_for_join_at_exit()

```
static __inline__ bool _States_Is_waiting_for_join_at_exit (
            States_Control the_states )  [static]
```

Checks if WAITING_FOR_JOIN_AT_EXIT state is set.

This function returns true if the WAITING_FOR_JOIN_AT_EXIT state is set in *the_states*, and false otherwise.

**Parameters**

| | |
|---|---|
| *the_states* | The task state set to test. |

**Return values**

| | |
|---|---|
| *true* | WAITING_FOR_JOIN_AT_EXIT state is set in *the_states*. |
| *false* | WAITING_FOR_JOIN_AT_EXIT state is not set in *the_states*. |

Definition at line 267 of file statesimpl.h.

#### 8.172.4.8 _States_Is_waiting_for_rpc_reply()

```
static __inline__ bool _States_Is_waiting_for_rpc_reply (
            States_Control the_states )  [static]
```

Checks if WAITING_FOR_TIME state is set.

This function returns true if the WAITING_FOR_TIME state is set in *the_states*, and false otherwise.

**Parameters**

| | |
|---|---|
| *the_states* | The task state set to test. |

**Return values**

| | |
|---|---|
| *true* | WAITING_FOR_TIME state is set in *the_states*. |
| *false* | WAITING_FOR_TIME state is not set in *the_states*. |

Definition at line 249 of file statesimpl.h.

#### 8.172.4.9 _States_Set()

```
static __inline__ States_Control _States_Set (
             States_Control states_to_set,
             States_Control current_state )  [static]
```

Returns the result of setting the states to set to the given state.

This function sets the given *states_to_set* into the *current_state* passed in. The result is returned to the user in *current_state*.

**Parameters**

| | |
|---|---|
| *states_to_set* | The state bits to set. |
| *current_state* | The state set to add them to. |

**Returns**

This method returns the updated states value.

Definition at line 157 of file statesimpl.h.

# 8.173 Thread-Local Storage (TLS)

Thread-local storage (TLS) support.

## Files

- file tls.h

  *Thread-Local Storage (TLS)*

## Classes

- struct TLS_Dynamic_thread_vector
- struct TLS_Thread_control_block
- struct TLS_Index

## Typedefs

- typedef struct TLS_Thread_control_block **TLS_Thread_control_block**

## Functions

- static uintptr_t _TLS_Get_size (void)

  *Gets the TLS size.*
- static uintptr_t _TLS_Heap_align_up (uintptr_t val)

  *Returns the value aligned up to the heap alignment.*
- static uintptr_t _TLS_Get_thread_control_block_area_size (uintptr_t alignment)

  *Returns the size of the thread control block area size for this alignment, or the minimum size if alignment is too small.*
- uintptr_t _TLS_Get_allocation_size (void)

  *Return the TLS area allocation size.*
- static void * _TLS_Copy_and_clear (void *tls_area)

  *Copies TLS size bytes from the address tls_area and returns a pointer to the start of the area after clearing it.*
- static void * _TLS_Initialize (void *tls_block, TLS_Thread_control_block *tcb, TLS_Dynamic_thread_vector *dtv)

  *Initializes the dynamic thread vector.*
- static void * _TLS_TCB_at_area_begin_initialize (void *tls_area)

  *Initializes a dynamic thread vector beginning at the given starting address.*
- static void * _TLS_TCB_before_TLS_block_initialize (void *tls_area)

  *Initializes a dynamic thread vector with the area before a given starting address as thread control block.*
- static void * _TLS_TCB_after_TLS_block_initialize (void *tls_area)

  *Initializes a dynamic thread vector with the area after a given starting address as thread control block.*

## Variables

- char **_TLS_Data_begin** [ ]
- char **_TLS_Data_end** [ ]
- char **_TLS_Data_size** [ ]
- char **_TLS_BSS_begin** [ ]
- char **_TLS_BSS_end** [ ]
- char **_TLS_BSS_size** [ ]
- char **_TLS_Size** [ ]
- char _TLS_Alignment [ ]

  *The TLS section alignment.*

### 8.173.1 Detailed Description

Thread-local storage (TLS) support.

Variants I and II are according to Ulrich Drepper, "ELF Handling For Thread-Local Storage".

### 8.173.2 Function Documentation

#### 8.173.2.1 _TLS_Copy_and_clear()

```
static void* _TLS_Copy_and_clear (
             void * tls_area )  [inline], [static]
```

Copies TLS size bytes from the address tls_area and returns a pointer to the start of the area after clearing it.

**Parameters**

| | |
|---|---|
| *tls_area* | The starting address of the area to clear. |

**Returns**

The pointer to the beginning of the cleared section.

Definition at line 167 of file tls.h.

#### 8.173.2.2 _TLS_Get_allocation_size()

```
uintptr_t _TLS_Get_allocation_size (
             void  )
```

Return the TLS area allocation size.

**Returns**

The TLS area allocation size.

Definition at line 38 of file tlsallocsize.c.

#### 8.173.2.3 _TLS_Get_size()

```
static uintptr_t _TLS_Get_size (
            void  )  [inline], [static]
```

Gets the TLS size.

**Returns**

The TLS size.

Definition at line 108 of file tls.h.

#### 8.173.2.4 _TLS_Get_thread_control_block_area_size()

```
static uintptr_t _TLS_Get_thread_control_block_area_size (
            uintptr_t alignment )  [inline], [static]
```

Returns the size of the thread control block area size for this alignment, or the minimum size if alignment is too small.

**Parameters**

| | |
|---|---|
| *alignment* | The alignment for the operation. |

**Returns**

The size of the thread control block area.

Definition at line 144 of file tls.h.

#### 8.173.2.5 _TLS_Heap_align_up()

```
static uintptr_t _TLS_Heap_align_up (
            uintptr_t val )  [inline], [static]
```

Returns the value aligned up to the heap alignment.

**Parameters**

| | |
|---|---|
| *val* | The value to align. |

**Returns**

The value aligned to the heap alignment.

Definition at line 129 of file tls.h.

#### 8.173.2.6 _TLS_Initialize()

```
static void* _TLS_Initialize (
             void * tls_block,
             TLS_Thread_control_block * tcb,
             TLS_Dynamic_thread_vector * dtv )  [inline], [static]
```

Initializes the dynamic thread vector.

**Parameters**

| | | |
|---|---|---|
| | *tls_block* | The tls block for *dtv*. |
| | *tcb* | The thread control block for *dtv*. |
| out | *dtv* | The dynamic thread vector to initialize. |

**Returns**

Pointer to an area that was copied and cleared from tls_block onwards (

**See also**

_TLS_Copy_and_clear).

Definition at line 196 of file tls.h.

#### 8.173.2.7 _TLS_TCB_after_TLS_block_initialize()

```
static void* _TLS_TCB_after_TLS_block_initialize (
             void * tls_area )  [inline], [static]
```

Initializes a dynamic thread vector with the area after a given starting address as thread control block.

Use Variant II

**Parameters**

| | |
|---|---|
| *tls_area* | The tls area for the initialization. |

**Returns**

Pointer to an area that was copied and cleared from tls_block onwards (

**See also**

_TLS_Copy_and_clear).

Definition at line 272 of file tls.h.

#### 8.173.2.8 _TLS_TCB_at_area_begin_initialize()

```
static void* _TLS_TCB_at_area_begin_initialize (
             void * tls_area )  [inline], [static]
```

Initializes a dynamic thread vector beginning at the given starting address.

Use Variant I, TLS offsets emitted by linker takes the TCB into account.

**Parameters**

| | |
|---|---|
| *tls_area* | The tls area for the initialization. |

**Returns**

Pointer to an area that was copied and cleared from tls_block onwards (

**See also**

_TLS_Copy_and_clear).

Definition at line 225 of file tls.h.

#### 8.173.2.9 _TLS_TCB_before_TLS_block_initialize()

```
static void* _TLS_TCB_before_TLS_block_initialize (
             void * tls_area )  [inline], [static]
```

Initializes a dynamic thread vector with the area before a given starting address as thread control block.

Use Variant I, TLS offsets emitted by linker neglects the TCB.

**Parameters**

| | |
|---|---|
| *tls_area* | The tls area for the initialization. |

**Returns**

Pointer to an area that was copied and cleared from tls_block onwards (

**See also**

_TLS_Copy_and_clear).

Definition at line 248 of file tls.h.

## 8.173.3 Variable Documentation

### 8.173.3.1 _TLS_Alignment

```
char _TLS_Alignment[]  [extern]
```

The TLS section alignment.

This symbol is provided by the linker command file as the maximum alignment of the .tdata and .tbss sections. The linker ensures that the first TLS output section is aligned to the maximum alignment of all TLS output sections, see function _bfd_elf_tls_setup() in bfd/elflink.c of the GNU Binutils sources. The linker command file must take into account the case that the .tdata section is empty and the .tbss section is non-empty.

## 8.174 Time Services

Time service functions.

### Typedefs

- typedef CPU_Counter_ticks **T_ticks**
- typedef uint64_t **T_time**
- typedef char **T_time_string**[24]

### Functions

- const char * **T_time_to_string_ns** (T_time, T_time_string)
- const char * **T_time_to_string_us** (T_time, T_time_string)
- const char * **T_time_to_string_ms** (T_time, T_time_string)
- const char * **T_time_to_string_s** (T_time, T_time_string)
- const char * **T_ticks_to_string_ns** (T_ticks, T_time_string)
- const char * **T_ticks_to_string_us** (T_ticks, T_time_string)
- const char * **T_ticks_to_string_ms** (T_ticks, T_time_string)
- const char * **T_ticks_to_string_s** (T_ticks, T_time_string)
- T_time **T_ticks_to_time** (T_ticks)
- T_ticks **T_time_to_ticks** (T_time)
- T_time **T_seconds_and_nanoseconds_to_time** (uint32_t, uint32_t)
- void **T_time_to_seconds_and_nanoseconds** (T_time, uint32_t *, uint32_t *)
- T_time **T_now** (void)
- static T_ticks **T_tick** (void)
- T_time **T_now_clock** (void)
- T_time **T_now_dummy** (void)
- T_time **T_now_tick** (void)
- T_time **T_case_begin_time** (void)

### 8.174.1 Detailed Description

Time service functions.

## 8.175 Time Test 27 Support

Time Test 27 Support.

Time Test 27 Support.

# 8.176 Time of Day Handler

Time of Day Handler.

## Modules

- Time of Day Handler Action Hooks

  *Time of Day Handler Action Hooks.*

## Files

- file todimpl.h

  *Time of Day Handler API.*
- file coretod.c

  *Initializes the Time of Day Handler.*

## Classes

- struct TOD_Control

  *TOD control.*

## Macros

- #define TOD_SECONDS_PER_MINUTE (uint32_t)60
- #define TOD_MINUTES_PER_HOUR (uint32_t)60
- #define TOD_MONTHS_PER_YEAR (uint32_t)12
- #define TOD_DAYS_PER_YEAR (uint32_t)365
- #define TOD_HOURS_PER_DAY (uint32_t)24
- #define TOD_SECONDS_PER_DAY
- #define TOD_SECONDS_PER_NON_LEAP_YEAR (365 ∗ TOD_SECONDS_PER_DAY)
- #define TOD_MILLISECONDS_PER_SECOND (uint32_t)1000
- #define TOD_MICROSECONDS_PER_SECOND (uint32_t)1000000
- #define TOD_NANOSECONDS_PER_SECOND (uint32_t)1000000000
- #define TOD_NANOSECONDS_PER_MICROSECOND (uint32_t)1000
- #define TOD_TICKS_PER_SECOND TOD_TICKS_PER_SECOND_method()

  *Gets number of ticks in a second.*

## Functions

- void _TOD_Lock (void)

  *Locks the time of day mutex.*
- void _TOD_Unlock (void)

  *Unlocks the time of day mutex.*
- static void _TOD_Acquire (ISR_lock_Context *lock_context)

  *Checks if api mutex is owner of the time of day mutex.*
- static void _TOD_Release (ISR_lock_Context *lock_context)

  *Releases the lock context for the timecounter.*
- Status_Control _TOD_Set (const struct timespec *tod, ISR_lock_Context *lock_context)

  *Sets the time of day.*
- static void _TOD_Get (struct timespec *tod)

  *Gets the current time in the timespec format.*
- static void _TOD_Get_uptime (Timestamp_Control *time)

  *Gets the system uptime with potential accuracy to the nanosecond.*
- static void _TOD_Get_zero_based_uptime (Timestamp_Control *time)

  *Gets the system uptime with potential accuracy to the nanosecond.*
- static void _TOD_Get_zero_based_uptime_as_timespec (struct timespec *time)

  *Gets the system uptime with potential accuracy to the nanosecond.*
- static uint32_t _TOD_Seconds_since_epoch (void)

  *Returns Number of seconds Since RTEMS epoch.*
- uint32_t TOD_TICKS_PER_SECOND_method (void)

  *Gets number of ticks in a second.*
- static __inline__ void _TOD_Get_timeval (struct timeval *time)

  *This routine returns a timeval based upon the internal timespec format TOD.*
- void _TOD_Adjust (const struct timespec *delta)

  *Adjusts the Time of Time.*
- static __inline__ bool _TOD_Is_set (void)

  *Check if the TOD is Set.*

## Variables

- TOD_Control _TOD

  *TOD Management information.*

### 8.176.1 Detailed Description

Time of Day Handler.

The following constants are related to the time of day and are independent of RTEMS.

This handler encapsulates functionality used to manage time of day.

### 8.176.2 Macro Definition Documentation

#### 8.176.2.1 TOD_DAYS_PER_YEAR

```
#define TOD_DAYS_PER_YEAR (uint32_t)365
```

This constant represents the number of days in a non-leap year.

Definition at line 64 of file todimpl.h.

#### 8.176.2.2 TOD_HOURS_PER_DAY

```
#define TOD_HOURS_PER_DAY (uint32_t)24
```

This constant represents the number of hours per day.

Definition at line 69 of file todimpl.h.

#### 8.176.2.3 TOD_MICROSECONDS_PER_SECOND

```
#define TOD_MICROSECONDS_PER_SECOND (uint32_t)1000000
```

This constant represents the number of microseconds in a second.

Definition at line 92 of file todimpl.h.

#### 8.176.2.4 TOD_MILLISECONDS_PER_SECOND

```
#define TOD_MILLISECONDS_PER_SECOND (uint32_t)1000
```

This constant represents the number of millisecond in a second.

Definition at line 87 of file todimpl.h.

#### 8.176.2.5 TOD_MINUTES_PER_HOUR

```
#define TOD_MINUTES_PER_HOUR (uint32_t)60
```

This constant represents the number of minutes per hour.

Definition at line 54 of file todimpl.h.

#### 8.176.2.6 TOD_MONTHS_PER_YEAR

```
#define TOD_MONTHS_PER_YEAR (uint32_t)12
```

This constant represents the number of months in a year.

Definition at line 59 of file todimpl.h.

#### 8.176.2.7 TOD_NANOSECONDS_PER_MICROSECOND

```
#define TOD_NANOSECONDS_PER_MICROSECOND (uint32_t)1000
```

This constant represents the number of nanoseconds in a mircosecond.

Definition at line 102 of file todimpl.h.

#### 8.176.2.8 TOD_NANOSECONDS_PER_SECOND

```
#define TOD_NANOSECONDS_PER_SECOND (uint32_t)1000000000
```

This constant represents the number of nanoseconds in a second.

Definition at line 97 of file todimpl.h.

#### 8.176.2.9 TOD_SECONDS_PER_DAY

```
#define TOD_SECONDS_PER_DAY
```

**Value:**

```
(uint32_t) (TOD_SECONDS_PER_MINUTE * \
TOD_MINUTES_PER_HOUR   * \
TOD_HOURS_PER_DAY)
```

This constant represents the number of seconds in a day which does not include a leap second.

Definition at line 75 of file todimpl.h.

#### 8.176.2.10 TOD_SECONDS_PER_MINUTE

```
#define TOD_SECONDS_PER_MINUTE (uint32_t)60
```

This constant represents the number of seconds in a minute.

Definition at line 49 of file todimpl.h.

#### 8.176.2.11 TOD_SECONDS_PER_NON_LEAP_YEAR

```
#define TOD_SECONDS_PER_NON_LEAP_YEAR (365 * TOD_SECONDS_PER_DAY)
```

This constant represents the number of seconds in a non-leap year.

Definition at line 82 of file todimpl.h.

#### 8.176.2.12 TOD_TICKS_PER_SECOND

```
#define TOD_TICKS_PER_SECOND TOD_TICKS_PER_SECOND_method()
```

Gets number of ticks in a second.

This method exists to hide the fact that TOD_TICKS_PER_SECOND can not be implemented as a macro in a .h file due to visibility issues. The Configuration Table is not available to SuperCore .h files but is available to their .c files.

Definition at line 300 of file todimpl.h.

### 8.176.3 Function Documentation

#### 8.176.3.1 _TOD_Acquire()

```
static void _TOD_Acquire (
            ISR_lock_Context * lock_context )  [inline], [static]
```

Checks if api mutex is owner of the time of day mutex.

**Return values**

| | |
|---|---|
| *true* | It is owner of the time of day mutex. |
| *false* | It is not owner of the time of day mutex. |

Acquires the lock context for the timecounter.

**Parameters**

| | |
|---|---|
| *lock_context* | The lock to acquire. |

Definition at line 175 of file todimpl.h.

#### 8.176.3.2 _TOD_Adjust()

```
void _TOD_Adjust (
             const struct timespec * delta )
```

Adjusts the Time of Time.

This method is used to adjust the current time of day by the specified amount.

**Parameters**

| | |
|---|---|
| *delta* | is the amount to adjust. |

#### 8.176.3.3 _TOD_Get()

```
static void _TOD_Get (
             struct timespec * tod )  [inline], [static]
```

Gets the current time in the timespec format.

**Parameters**

| | | |
|---|---|---|
| out | *time* | The value gathered by the request. |

Definition at line 214 of file todimpl.h.

#### 8.176.3.4 _TOD_Get_timeval()

```
static __inline__ void _TOD_Get_timeval (
             struct timeval * time )  [static]
```

This routine returns a timeval based upon the internal timespec format TOD.

**Parameters**

| | | |
|---|---|---|
| out | *time* | The timeval to be filled in by the method. |

Definition at line 308 of file todimpl.h.

#### 8.176.3.5 _TOD_Get_uptime()

```
static void _TOD_Get_uptime (
             Timestamp_Control * time )  [inline], [static]
```

Gets the system uptime with potential accuracy to the nanosecond.

This routine returns the system uptime with potential accuracy to the nanosecond.

The initial uptime value is undefined.

**Parameters**

| | | |
|---|---|---|
| out | *time* | Is a pointer to the uptime after the method call. |

Definition at line 231 of file todimpl.h.

#### 8.176.3.6 _TOD_Get_zero_based_uptime()

```
static void _TOD_Get_zero_based_uptime (
            Timestamp_Control * time )  [inline], [static]
```

Gets the system uptime with potential accuracy to the nanosecond.

The initial uptime value is zero.

**Parameters**

| | | |
|---|---|---|
| out | *time* | Is a pointer to the uptime after the method call. |

Definition at line 245 of file todimpl.h.

#### 8.176.3.7 _TOD_Get_zero_based_uptime_as_timespec()

```
static void _TOD_Get_zero_based_uptime_as_timespec (
            struct timespec * time )  [inline], [static]
```

Gets the system uptime with potential accuracy to the nanosecond.

The initial uptime value is zero.

**Parameters**

| | | |
|---|---|---|
| out | *time* | Is a pointer to the uptime after the method call. |

Definition at line 259 of file todimpl.h.

#### 8.176.3.8 _TOD_Is_set()

```
static __inline__ bool _TOD_Is_set (
             void  )  [static]
```

Check if the TOD is Set.

**Return values**

| | |
|---|---|
| *true* | The time is set. |
| *false* | The time is not set. |

Definition at line 333 of file todimpl.h.

#### 8.176.3.9 _TOD_Release()

```
static void _TOD_Release (
             ISR_lock_Context * lock_context )  [inline], [static]
```

Releases the lock context for the timecounter.

**Parameters**

| | |
|---|---|
| *lock_context* | The lock to release. |

Definition at line 185 of file todimpl.h.

#### 8.176.3.10 _TOD_Seconds_since_epoch()

```
static uint32_t _TOD_Seconds_since_epoch (
             void  )  [inline], [static]
```

Returns Number of seconds Since RTEMS epoch.

The following contains the number of seconds from 00:00:00 January 1, TOD_BASE_YEAR until the current time of day.

**Returns**

The number of seconds since RTEMS epoch.

Definition at line 275 of file todimpl.h.

#### 8.176.3.11 _TOD_Set()

```
Status_Control _TOD_Set (
              const struct timespec * tod,
              ISR_lock_Context * lock_context )
```

Sets the time of day.

The caller must be the owner of the TOD lock.

**Parameters**

| | |
|---|---|
| *tod* | The new time of day in timespec format representing the time since UNIX Epoch. |
| *lock_context* | The ISR lock context used for the corresponding _TOD_Acquire(). The caller must be the owner of the TOD lock. This function will release the TOD lock. |

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | Successful operation. |
| *other* | Some error occurred. |

### 8.176.3.12 TOD_TICKS_PER_SECOND_method()

```
uint32_t TOD_TICKS_PER_SECOND_method (
             void  )
```

Gets number of ticks in a second.

This method returns the number of ticks in a second.

**Note**

If the clock tick value does not multiply evenly into a second then this number of ticks will be slightly shorter than a second.

**Returns**

The number of ticks in a second.

# 8.177 Time of Day Handler Action Hooks

Time of Day Handler Action Hooks.

## Classes

- struct TOD_Hook

  *Structure to manage each TOD action hook.*

## Typedefs

- typedef struct TOD_Hook TOD_Hook

  *Structure to manage each TOD action hook.*

## Enumerations

- enum TOD_Action { TOD_ACTION_SET_CLOCK }

  *Possible actions where a registered hook could be invoked.*

## Functions

- void _TOD_Hook_Register (TOD_Hook *hook)

  *Add a TOD Action Hook.*
- void _TOD_Hook_Unregister (TOD_Hook *hook)

  *Remove a TOD Action Hook.*
- Status_Control _TOD_Hook_Run (TOD_Action action, const struct timespec *tod)

  *Run the TOD Action Hooks.*

## Variables

- Chain_Control _TOD_Hooks

  *Set of registered methods for TOD Actions.*

## 8.177.1 Detailed Description

Time of Day Handler Action Hooks.

The following support registering a hook which is invoked when the TOD is set. These can be used by a paravirtualized BSP to mirror time changes to the hosting environment or a regular BSP to program a real-time clock when the RTEMS TOD is set.

## 8.177.2 Enumeration Type Documentation

### 8.177.2.1 TOD_Action

```
enum TOD_Action
```

Possible actions where a registered hook could be invoked.

**Enumerator**

| | |
|---|---|
| TOD_ACTION_SET_CLOCK | Constant to indicate the TOD is being set. |

Definition at line 359 of file todimpl.h.

## 8.177.3 Function Documentation

### 8.177.3.1 _TOD_Hook_Register()

```
void _TOD_Hook_Register (
              TOD_Hook * hook )
```

Add a TOD Action Hook.

This method is used to add a hook to the TOD action set.

hook is the action hook to register.

### 8.177.3.2 _TOD_Hook_Run()

```
Status_Control _TOD_Hook_Run (
              TOD_Action action,
              const struct timespec * tod )
```

Run the TOD Action Hooks.

This method is used to invoke the set of TOD action hooks.

action The action which triggered this run.

tod The current time of day.

**Return values**

| | |
|---|---|
| *STATUS_SUCCESSFUL* | Successful operation. |
| *other* | Some error occurred. |

### 8.177.3.3 _TOD_Hook_Unregister()

```
void _TOD_Hook_Unregister (
              TOD_Hook * hook )
```

Remove a TOD Action Hook.

This method is used to remove a hook from the TOD action set.

hook is the action hook to unregister.

## 8.178 Timecounter Handler

Timecounter Handler.

### Files

- file timecounter.h

  *Timecounter API.*
- file timecounterimpl.h

  *Timecounter Implementation.*

### Macros

- #define _Timecounter_Acquire(lock_context) _ISR_lock_ISR_disable_and_acquire( &_Timecounter_Lock, lock_context )

  *Lock to protect the timecounter mechanic.*
- #define _Timecounter_Release(lock_context) _ISR_lock_Release_and_ISR_enable(&_Timecounter_Lock, lock_context)

  *Releases the timecounter lock.*

### Functions

- void _Timecounter_Bintime (struct bintime *bt)

  *Returns the wall clock time in the bintime format.*
- void _Timecounter_Nanotime (struct timespec *ts)

  *Returns the wall clock time in the timespec format.*
- void _Timecounter_Microtime (struct timeval *tv)

  *Returns the wall clock time in the timeval format.*
- void _Timecounter_Binuptime (struct bintime *bt)

  *Returns the uptime in the bintime format.*
- int64_t _Timecounter_Sbinuptime (void)

  *Returns the uptime in the sbintime_t format.*
- void _Timecounter_Nanouptime (struct timespec *ts)

  *Returns the uptime in the timespec format.*
- void _Timecounter_Microuptime (struct timeval *tv)

  *Returns the uptime in the timeval format.*
- void _Timecounter_Getbintime (struct bintime *bt)

  *Returns the wall clock time in the bintime format.*
- void _Timecounter_Getnanotime (struct timespec *ts)

  *Returns the wall clock time in the timespec format.*
- void _Timecounter_Getmicrotime (struct timeval *tv)

  *Returns the wall clock time in the timeval format.*
- void _Timecounter_Getbinuptime (struct bintime *bt)

  *Returns the uptime in the bintime format.*
- void _Timecounter_Getnanouptime (struct timespec *ts)

  *Returns the uptime in the timespec format.*
- void _Timecounter_Getmicrouptime (struct timeval *tv)

  *Returns the uptime in the timeval format.*

- void _Timecounter_Getboottime (struct timeval *tv)

  *Returns the boot time in the timeval format.*
- void _Timecounter_Getboottimebin (struct bintime *bt)

  *Returns the boot time in the bintime format.*
- void _Timecounter_Install (struct timecounter *tc)

  *Installs the timecounter.*
- void _Timecounter_Tick (void)

  *Performs a timecounter tick.*
- void _Timecounter_Tick_simple (uint32_t delta, uint32_t offset, ISR_lock_Context *lock_context)

  *Performs a simple timecounter tick.*
- void _Timecounter_Set_clock (const struct bintime *bt, ISR_lock_Context *lock_context)

  *Sets the timecounter clock to the given value.*

## Variables

- volatile time_t _Timecounter_Time_second

  *The wall clock time in seconds.*
- volatile int32_t _Timecounter_Time_uptime

  *The uptime in seconds.*
- struct timecounter * _Timecounter

  *The current timecounter.*

### 8.178.1 Detailed Description

Timecounter Handler.

### 8.178.2 Macro Definition Documentation

#### 8.178.2.1 _Timecounter_Acquire

```
#define _Timecounter_Acquire(
              lock_context )  _ISR_lock_ISR_disable_and_acquire( &_Timecounter_Lock, lock_↩
context )
```

Lock to protect the timecounter mechanic.

Acquires the timecounter lock.

**Parameters**

| | |
|---|---|
| *lock_context* | The lock context. |

See _Timecounter_Tick_simple().

Definition at line 201 of file timecounter.h.

#### 8.178.2.2 _Timecounter_Release

```
#define _Timecounter_Release(
              lock_context )  _ISR_lock_Release_and_ISR_enable(&_Timecounter_Lock, lock_↩
context)
```

Releases the timecounter lock.

**Parameters**

| | |
|---|---|
| *lock_context* | The lock context. |

See _Timecounter_Tick_simple().

Definition at line 211 of file timecounter.h.

### 8.178.3 Function Documentation

#### 8.178.3.1 _Timecounter_Bintime()

```
void _Timecounter_Bintime (
              struct bintime * bt )
```

Returns the wall clock time in the bintime format.

**Parameters**

| | | |
|---|---|---|
| out | *bt* | Returns the wall clock time. |

#### 8.178.3.2 _Timecounter_Binuptime()

```
void _Timecounter_Binuptime (
              struct bintime * bt )
```

Returns the uptime in the bintime format.

**Parameters**

| | | |
|---|---|---|
| out | *bt* | Returns the uptime. |

#### 8.178.3.3 _Timecounter_Getbintime()

```
void _Timecounter_Getbintime (
              struct bintime * bt )
```

Returns the wall clock time in the bintime format.

This function obtains the time with a lower overhead and lower accuracy compared to the _Timecounter_Bintime() variant.

**Parameters**

| | | |
|---|---|---|
| out | *ts* | Returns the wall clock time. |

#### 8.178.3.4 _Timecounter_Getbinuptime()

```
void _Timecounter_Getbinuptime (
              struct bintime * bt )
```

Returns the uptime in the bintime format.

This function obtains the time with a lower overhead and lower accuracy compared to the _Timecounter_Binuptime() variant.

**Parameters**

| | | |
|---|---|---|
| out | *ts* | Returns the uptime. |

#### 8.178.3.5 _Timecounter_Getboottime()

```
void _Timecounter_Getboottime (
              struct timeval * tv )
```

Returns the boot time in the timeval format.

**Parameters**

| | | |
|---|---|---|
| out | *tv* | Returns the boot time. |

#### 8.178.3.6 _Timecounter_Getboottimebin()

```
void _Timecounter_Getboottimebin (
              struct bintime * bt )
```

Returns the boot time in the bintime format.

**Parameters**

| | | |
|---|---|---|
| out | *tv* | Returns the boot time. |

### 8.178.3.7 _Timecounter_Getmicrotime()

```
void _Timecounter_Getmicrotime (
            struct timeval * tv )
```

Returns the wall clock time in the timeval format.

This function obtains the time with a lower overhead and lower accuracy compared to the _Timecounter_Microtime() variant.

**Parameters**

| | | |
|---|---|---|
| out | *tv* | Returns the wall clock time. |

**See also**

_Timecounter_Getbintime().

### 8.178.3.8 _Timecounter_Getmicrouptime()

```
void _Timecounter_Getmicrouptime (
            struct timeval * tv )
```

Returns the uptime in the timeval format.

This function obtains the time with a lower overhead and lower accuracy compared to the _Timecounter_Microuptime() variant.

**Parameters**

| | | |
|---|---|---|
| out | *tv* | Returns the uptime. |

### 8.178.3.9 _Timecounter_Getnanotime()

```
void _Timecounter_Getnanotime (
            struct timespec * ts )
```

Returns the wall clock time in the timespec format.

This function obtains the time with a lower overhead and lower accuracy compared to the _Timecounter_Nanotime() variant.

**Parameters**

| | | |
|---|---|---|
| out | *ts* | Returns the wall clock time. |

**See also**

_Timecounter_Getbintime().

### 8.178.3.10 _Timecounter_Getnanouptime()

```
void _Timecounter_Getnanouptime (
            struct timespec * ts )
```

Returns the uptime in the timespec format.

This function obtains the time with a lower overhead and lower accuracy compared to the _Timecounter_Nanouptime() variant.

**Parameters**

| | | |
|---|---|---|
| out | *ts* | Returns the uptime. |

### 8.178.3.11 _Timecounter_Install()

```
void _Timecounter_Install (
            struct timecounter * tc )
```

Installs the timecounter.

The timecounter structure must contain valid values in the fields tc_get_timecount, tc_counter_mask, tc_frequency and tc_quality. All other fields must be zero initialized.

**Parameters**

| | |
|---|---|
| *tc* | The timecounter. |

### 8.178.3.12 _Timecounter_Microtime()

```
void _Timecounter_Microtime (
            struct timeval * tv )
```

Returns the wall clock time in the timeval format.

**Parameters**

| | | |
|---|---|---|
| out | *tv* | Returns the wall clock time. |

#### 8.178.3.13 _Timecounter_Microuptime()

```
void _Timecounter_Microuptime (
            struct timeval * tv )
```

Returns the uptime in the timeval format.

**Parameters**

| | | |
|---|---|---|
| out | *tv* | Returns the uptime. |

#### 8.178.3.14 _Timecounter_Nanotime()

```
void _Timecounter_Nanotime (
            struct timespec * ts )
```

Returns the wall clock time in the timespec format.

**Parameters**

| | | |
|---|---|---|
| out | *ts* | Returns the wall clock time. |

#### 8.178.3.15 _Timecounter_Nanouptime()

```
void _Timecounter_Nanouptime (
            struct timespec * ts )
```

Returns the uptime in the timespec format.

**Parameters**

| | | |
|---|---|---|
| out | *ts* | Returns the uptime. |

#### 8.178.3.16 _Timecounter_Sbinuptime()

```
int64_t _Timecounter_Sbinuptime (
              void  )
```

Returns the uptime in the sbintime_t format.

**Returns**

Returns the uptime.

Definition at line 487 of file kern_tc.c.

#### 8.178.3.17 _Timecounter_Set_clock()

```
void _Timecounter_Set_clock (
              const struct bintime * bt,
              ISR_lock_Context * lock_context )
```

Sets the timecounter clock to the given value.

**Parameters**

| | |
|---|---|
| *bt* | The value to set the clock to. |
| *lock_context* | The interrupt lock context. |

Definition at line 1454 of file kern_tc.c.

#### 8.178.3.18 _Timecounter_Tick_simple()

```
void _Timecounter_Tick_simple (
              uint32_t delta,
              uint32_t offset,
              ISR_lock_Context * lock_context )
```

Performs a simple timecounter tick.

This is a special purpose tick function for simple timecounter to support legacy clock drivers.

**Parameters**

| | |
|---|---|
| *delta* | The time in timecounter ticks elapsed since the last call to _Timecounter_Tick_simple(). |
| *offset* | The current value of the timecounter. |
| *]* | lock_context The lock context of the corresponding _Timecounter_Acquire(). |

Definition at line 2142 of file kern_tc.c.

## 8.178.4 Variable Documentation

### 8.178.4.1 _Timecounter_Time_uptime

```
volatile int32_t _Timecounter_Time_uptime  [extern]
```

The uptime in seconds.

For compatibility with the FreeBSD network stack the initial value is one second.

# 8.179 Timecounter Support

## Files

- file timecounter.h

  *Timecounter API.*

## Classes

- struct rtems_timecounter_simple

  *Simple timecounter to support legacy clock drivers.*

## Macros

- #define RTEMS_TIMECOUNTER_QUALITY_CLOCK_DRIVER 100

  *Timecounter quality for the clock drivers.*

## Typedefs

- typedef void rtems_timecounter_simple_at_tick(rtems_timecounter_simple *tc)

  *At tick handling done under protection of the timecounter lock.*
- typedef uint32_t rtems_timecounter_simple_get(rtems_timecounter_simple *tc)

  *Returns the current value of a simple timecounter.*
- typedef bool rtems_timecounter_simple_is_pending(rtems_timecounter_simple *tc)

  *Returns true if the interrupt of a simple timecounter is pending, and false otherwise.*

## Functions

- static __inline__ void rtems_timecounter_install (struct timecounter *tc)

  *Installs the timecounter.*
- static __inline__ void rtems_timecounter_tick (void)

  *Performs a timecounter tick.*
- void rtems_timecounter_simple_install (rtems_timecounter_simple *tc, uint32_t counter_frequency_in_hz, uint32_t counter_ticks_per_clock_tick, timecounter_get_t *get_timecount)

  *Initializes and installs a simple timecounter.*
- static __inline__ uint32_t rtems_timecounter_simple_scale (const rtems_timecounter_simple *tc, uint32_t value)

  *Maps a simple timecounter value into its binary frequency domain.*
- static __inline__ void rtems_timecounter_simple_downcounter_tick (rtems_timecounter_simple *tc, rtems_timecounter_simple_get get, rtems_timecounter_simple_at_tick at_tick)

  *Performs a simple timecounter tick for downcounters.*
- static __inline__ void rtems_timecounter_simple_upcounter_tick (rtems_timecounter_simple *tc, rtems_timecounter_simple_get get, rtems_timecounter_simple_at_tick at_tick)

  *Performs a simple timecounter tick for upcounters.*
- static __inline__ uint32_t rtems_timecounter_simple_downcounter_get (struct timecounter *tc_base, rtems_timecounter_simple_get get, rtems_timecounter_simple_is_pending is_pending)

  *Gets the simple timecounter value mapped to its binary frequency domain for downcounters.*
- static __inline__ uint32_t rtems_timecounter_simple_upcounter_get (struct timecounter *tc_base, rtems_timecounter_simple_get get, rtems_timecounter_simple_is_pending is_pending)

  *Gets the simple timecounter value mapped to its binary frequency domain for upcounters.*

## 8.179.1 Detailed Description

## 8.179.2 Macro Definition Documentation

### 8.179.2.1 RTEMS_TIMECOUNTER_QUALITY_CLOCK_DRIVER

```
#define RTEMS_TIMECOUNTER_QUALITY_CLOCK_DRIVER 100
```

Timecounter quality for the clock drivers.

Timecounter with higher quality value are used in favour of those with lower quality value.

Definition at line 47 of file timecounter.h.

## 8.179.3 Function Documentation

### 8.179.3.1 rtems_timecounter_install()

```
static __inline__ void rtems_timecounter_install (
            struct timecounter * tc )  [static]
```

Installs the timecounter.

The timecounter structure must contain valid values in the fields tc_get_timecount, tc_counter_mask, tc_frequency and tc_quality. All other fields must be zero initialized.

**Parameters**

| | |
|---|---|
| *tc* | The timecounter. |

Below is an exemplary code snippet that shows the adjustable parameters and the following call of the install routine.

```
struct timecounter tc;
uint32_t get_timecount( struct timecounter *tc )
{
  return some_free_running_counter;
}
void install( void )
{
  tc.tc_get_timecount = get_timecount;
  tc.tc_counter_mask = 0xffffffff;
  tc.tc_frequency = 123456;
  tc.tc_quality = RTEMS_TIMECOUNTER_QUALITY_CLOCK_DRIVER;
  rtems_timecounter_install( &tc );
}
```

Definition at line 73 of file timecounter.h.

### 8.179.3.2 rtems_timecounter_simple_downcounter_get()

```
static __inline__ uint32_t rtems_timecounter_simple_downcounter_get (
            struct timecounter * tc_base,
            rtems_timecounter_simple_get get,
            rtems_timecounter_simple_is_pending is_pending )  [static]
```

Gets the simple timecounter value mapped to its binary frequency domain for downcounters.

**Parameters**

| | | |
|---|---|---|
| in | *tc* | The simple timecounter. |
| in | *get* | The method to get the value of the simple timecounter. |
| in | *is_pending* | The method which indicates if the interrupt of the simple timecounter is pending. |

Definition at line 278 of file timecounter.h.

### 8.179.3.3 rtems_timecounter_simple_downcounter_tick()

```
static __inline__ void rtems_timecounter_simple_downcounter_tick (
            rtems_timecounter_simple * tc,
            rtems_timecounter_simple_get get,
            rtems_timecounter_simple_at_tick at_tick )  [static]
```

Performs a simple timecounter tick for downcounters.

**Parameters**

| | | |
|---|---|---|
| in | *tc* | The simple timecounter. |
| in | *get* | The method to get the value of the simple timecounter. |
| in | *at_tick* | The method to perform work under timecounter lock protection at this tick, e.g. clear a pending flag. |

Definition at line 222 of file timecounter.h.

### 8.179.3.4 rtems_timecounter_simple_install()

```
void rtems_timecounter_simple_install (
            rtems_timecounter_simple * tc,
            uint32_t counter_frequency_in_hz,
            uint32_t counter_ticks_per_clock_tick,
            timecounter_get_t * get_timecount )
```

Initializes and installs a simple timecounter.

A simple timecounter can be used if the hardware provides no free running counter. A periodic hardware counter must be provided. The counter period must be synchronous to the clock tick. The counter ticks per clock tick is scaled up to the next power of two.

**Parameters**

| | | |
|---|---|---|
| in | *tc* | Zero initialized simple timecounter. |
| in | *counter_frequency_in_hz* | The hardware counter frequency in Hz. |
| in | *counter_ticks_per_clock_tick* | The hardware counter ticks per clock tick. |
| in | *get_timecount* | The method to get the current time count. |

```
#include <rtems/timecounter.h>
static rtems_timecounter_simple some_tc;
static uint32_t some_tc_get( rtems_timecounter_simple *tc )
{
  return some.value;
}
static bool some_tc_is_pending( rtems_timecounter_simple *tc )
{
  return some.is_pending;
}
static uint32_t some_tc_get_timecount( struct timecounter *tc )
{
  return rtems_timecounter_simple_downcounter_get(
    tc,
    some_tc_get,
    some_tc_is_pending
  );
}
static void some_tc_tick( void )
{
  rtems_timecounter_simple_downcounter_tick( &some_tc, some_tc_get );
}
void some_tc_init( void )
{
  uint64_t us_per_tick;
  uint32_t counter_frequency_in_hz;
  uint32_t counter_ticks_per_clock_tick;
  us_per_tick = rtems_configuration_get_microseconds_per_tick();
  counter_frequency_in_hz = some_tc_get_frequency();
  counter_ticks_per_clock_tick =
    (uint32_t) ( counter_frequency_in_hz * us_per_tick ) / 1000000;
  some_tc_init_hardware( counter_ticks_per_clock_tick );
  some_tc_init_clock_tick_interrupt( some_tc_tick );
  rtems_timecounter_simple_install(
    &some_tc,
    counter_frequency_in_hz,
    counter_ticks_per_clock_tick,
    some_tc_get_timecount
  );
}
```

**See also**

rtems_timecounter_simple_downcounter_get(), rtems_timecounter_simple_downcounter_tick(), rtems_timecounter_simple_upc and rtems_timecounter_simple_upcounter_tick().

### 8.179.3.5 rtems_timecounter_simple_scale()

```
static __inline__ uint32_t rtems_timecounter_simple_scale (
             const rtems_timecounter_simple * tc,
             uint32_t value )  [static]
```

Maps a simple timecounter value into its binary frequency domain.

**Parameters**

| | | |
|---|---|---|
| in | *tc* | The simple timecounter. |
| in | *value* | The value of the simple timecounter. |

**Returns**

The scaled value.

Definition at line 206 of file timecounter.h.

### 8.179.3.6 rtems_timecounter_simple_upcounter_get()

```
static __inline__ uint32_t rtems_timecounter_simple_upcounter_get (
            struct timecounter * tc_base,
            rtems_timecounter_simple_get get,
            rtems_timecounter_simple_is_pending is_pending )  [static]
```

Gets the simple timecounter value mapped to its binary frequency domain for upcounters.

**Parameters**

| | | |
|---|---|---|
| in | *tc* | The simple timecounter. |
| in | *get* | The method to get the value of the simple timecounter. |
| in | *is_pending* | The method which indicates if the interrupt of the simple timecounter is pending. |

Definition at line 309 of file timecounter.h.

### 8.179.3.7 rtems_timecounter_simple_upcounter_tick()

```
static __inline__ void rtems_timecounter_simple_upcounter_tick (
            rtems_timecounter_simple * tc,
            rtems_timecounter_simple_get get,
            rtems_timecounter_simple_at_tick at_tick )  [static]
```

Performs a simple timecounter tick for upcounters.

**Parameters**

| | | |
|---|---|---|
| in | *tc* | The simple timecounter. |
| in | *get* | The method to get the value of the simple timecounter. |
| in | *at_tick* | The method to perform work under timecounter lock protection at this tick, e.g. clear a pending flag. |

Definition at line 251 of file timecounter.h.

### 8.179.3.8 rtems_timecounter_tick()

```
static __inline__ void rtems_timecounter_tick (
            void  )  [static]
```

Performs a timecounter tick.

Definition at line 83 of file timecounter.h.

# 8.180 Timer Manager

The Timer Manager provides support for timer facilities.

## Classes

- struct rtems_timer_information
  %

## Macros

- #define TIMER_CLASS_BIT_NOT_DORMANT 0x4
  %
- #define TIMER_CLASS_BIT_ON_TASK 0x2
  %
- #define TIMER_CLASS_BIT_TIME_OF_DAY 0x1
  %
- #define RTEMS_TIMER_SERVER_DEFAULT_PRIORITY ( (rtems_task_priority) -1 )
  %

## Typedefs

- typedef void rtems_timer_service_routine
  %
- typedef rtems_timer_service_routine(* rtems_timer_service_routine_entry) (rtems_id, void *)
  %

## Enumerations

- enum Timer_Classes {
  TIMER_DORMANT, TIMER_INTERVAL = TIMER_CLASS_BIT_NOT_DORMANT, TIMER_INTERVAL_ON_TASK,
  TIMER_TIME_OF_DAY,
  TIMER_TIME_OF_DAY_ON_TASK }
  %

## Functions

- rtems_status_code rtems_timer_cancel (rtems_id id)
  %
- rtems_status_code rtems_timer_create (rtems_name name, rtems_id *id)
  %
- rtems_status_code rtems_timer_delete (rtems_id id)
  %
- rtems_status_code rtems_timer_ident (rtems_name name, rtems_id *id)
  *Identifies a timer object by the specified object name.*
- rtems_status_code rtems_timer_get_information (rtems_id id, rtems_timer_information *the_info)
  %

- rtems_status_code rtems_timer_initiate_server (rtems_task_priority priority, size_t stack_size, rtems_attribute attribute_set)

  %
- rtems_status_code rtems_timer_reset (rtems_id id)

  %
- rtems_status_code rtems_timer_fire_after (rtems_id id, rtems_interval ticks, rtems_timer_service_routine_entry routine, void *user_data)

  %
- rtems_status_code rtems_timer_fire_when (rtems_id id, rtems_time_of_day *wall_time, rtems_timer_service_routine_entry routine, void *user_data)

  %
- rtems_status_code rtems_timer_server_fire_after (rtems_id id, rtems_interval ticks, rtems_timer_service_routine_entry routine, void *user_data)

  %
- rtems_status_code rtems_timer_server_fire_when (rtems_id id, rtems_time_of_day *wall_time, rtems_timer_service_routine_e routine, void *user_data)

  %

## 8.180.1 Detailed Description

The Timer Manager provides support for timer facilities.

## 8.180.2 Enumeration Type Documentation

### 8.180.2.1 Timer_Classes

```
enum Timer_Classes
```

%

**Enumerator**

| | |
|---|---|
| TIMER_DORMANT | % |
| TIMER_INTERVAL | % |
| TIMER_INTERVAL_ON_TASK | % |
| TIMER_TIME_OF_DAY | % |
| TIMER_TIME_OF_DAY_ON_TASK | % |

Definition at line 120 of file timer.h.

## 8.180.3 Function Documentation

#### 8.180.3.1 rtems_timer_cancel()

```
rtems_status_code rtems_timer_cancel (
            rtems_id id )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |

Definition at line 19 of file timercancel.c.

#### 8.180.3.2 rtems_timer_create()

```
rtems_status_code rtems_timer_create (
            rtems_name name,
            rtems_id * id )
```

%

**Parameters**

| | |
|---|---|
| *name* | % |
| *id* | % |

Definition at line 185 of file timercreate.c.

#### 8.180.3.3 rtems_timer_delete()

```
rtems_status_code rtems_timer_delete (
            rtems_id id )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |

Definition at line 23 of file timerdelete.c.

#### 8.180.3.4 rtems_timer_fire_after()

```
rtems_status_code rtems_timer_fire_after (
            rtems_id id,
```

```
            rtems_interval ticks,
            rtems_timer_service_routine_entry routine,
            void * user_data )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *ticks* | % |
| *routine* | % |
| *user_data* | % |

Definition at line 23 of file timerfireafter.c.

#### 8.180.3.5 rtems_timer_fire_when()

```
rtems_status_code rtems_timer_fire_when (
            rtems_id id,
            rtems_time_of_day * wall_time,
            rtems_timer_service_routine_entry routine,
            void * user_data )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *wall_time* | % |
| *routine* | % |
| *user_data* | % |

#### 8.180.3.6 rtems_timer_get_information()

```
rtems_status_code rtems_timer_get_information (
            rtems_id id,
            rtems_timer_information * the_info )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *the_info* | % |

### 8.180.3.7 rtems_timer_ident()

```
rtems_status_code rtems_timer_ident (
            rtems_name name,
            rtems_id * id )
```

Identifies a timer object by the specified object name.

This directive obtains the timer identifier associated with the timer name specified in `name`.

If the timer name is not unique, then the timer identifier will match the first timer with that name in the search order. However, this timer identifier is not guaranteed to correspond to the desired timer. The timer identifier is used with other timer related directives to access the timer.

The objects are searched from lowest to the highest index. Only the local node is searched.

**Parameters**

| | | |
|---|---|---|
| | *name* | is the object name to look up. |
| `out` | *id* | is the pointer to an object identifier variable. The object identifier of an object with the specified name will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The id parameter was NULL. |
| *RTEMS_INVALID_NAME* | The name parameter was 0. |
| *RTEMS_INVALID_NAME* | There was no object with the specified name on the local node. |

Definition at line 44 of file timerident.c.

### 8.180.3.8 rtems_timer_initiate_server()

```
rtems_status_code rtems_timer_initiate_server (
            rtems_task_priority priority,
            size_t stack_size,
            rtems_attribute attribute_set )
```

%

**Parameters**

| | |
|---|---|
| *priority* | % |
| *stack_size* | % |
| *attribute_set* | % |

#### 8.180.3.9 rtems_timer_reset()

```
rtems_status_code rtems_timer_reset (
            rtems_id id )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |

Definition at line 24 of file timerreset.c.

#### 8.180.3.10 rtems_timer_server_fire_after()

```
rtems_status_code rtems_timer_server_fire_after (
            rtems_id id,
            rtems_interval ticks,
            rtems_timer_service_routine_entry routine,
            void * user_data )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *ticks* | % |
| *routine* | % |
| *user_data* | % |

#### 8.180.3.11 rtems_timer_server_fire_when()

```
rtems_status_code rtems_timer_server_fire_when (
            rtems_id id,
            rtems_time_of_day * wall_time,
            rtems_timer_service_routine_entry routine,
            void * user_data )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |
| *wall_time* | % |
| *routine* | % |
| *user_data* | % |

## 8.181 Tracing

Tracing.

Tracing.

## 8.182 UART

Driver interface for APBUART.

### Files

- file apbuart.h

### Macros

- #define **APBUART_CTRL_RE** 0x1
- #define **APBUART_CTRL_TE** 0x2
- #define **APBUART_CTRL_RI** 0x4
- #define **APBUART_CTRL_TI** 0x8
- #define **APBUART_CTRL_PS** 0x10
- #define **APBUART_CTRL_PE** 0x20
- #define **APBUART_CTRL_FL** 0x40
- #define **APBUART_CTRL_LB** 0x80
- #define **APBUART_CTRL_EC** 0x100
- #define **APBUART_CTRL_TF** 0x200
- #define **APBUART_CTRL_RF** 0x400
- #define **APBUART_CTRL_DB** 0x800
- #define **APBUART_CTRL_BI** 0x1000
- #define **APBUART_CTRL_DI** 0x2000
- #define **APBUART_CTRL_FA** 0x80000000
- #define **APBUART_STATUS_DR** 0x1
- #define **APBUART_STATUS_TS** 0x2
- #define **APBUART_STATUS_TE** 0x4
- #define **APBUART_STATUS_BR** 0x8
- #define **APBUART_STATUS_OV** 0x10
- #define **APBUART_STATUS_PE** 0x20
- #define **APBUART_STATUS_FE** 0x40
- #define **APBUART_STATUS_ERR** 0x78
- #define **APBUART_STATUS_TH** 0x80
- #define **APBUART_STATUS_RH** 0x100
- #define **APBUART_STATUS_TF** 0x200
- #define **APBUART_STATUS_RF** 0x400

### Functions

- void **apbuart_outbyte_polled** (struct apbuart_regs *regs, unsigned char ch, int do_cr_on_newline, int wait↩ _sent)
- int **apbuart_inbyte_nonblocking** (struct apbuart_regs *regs)

### 8.182.1 Detailed Description

Driver interface for APBUART.

# 8.183 Unsigned 16-Bit Integer Checks

Checks for unsigned 16-bit integers (uint16_t).

## Macros

- #define **T_eq_u16**(a, e) T_flags_eq_uint(a, e, 0)
- #define **T_assert_eq_u16**(a, e) T_flags_eq_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_u16**(a, e) T_flags_eq_uint(a, e, T_CHECK_QUIET)
- #define **T_step_eq_u16**(s, a, e) T_flags_eq_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_u16**(s, a, e) T_flags_eq_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_u16**(a, e) T_flags_ne_uint(a, e, 0)
- #define **T_assert_ne_u16**(a, e) T_flags_ne_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_u16**(a, e) T_flags_ne_uint(a, e, T_CHECK_QUIET)
- #define **T_step_ne_u16**(s, a, e) T_flags_ne_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_u16**(s, a, e) T_flags_ne_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_u16**(a, e) T_flags_ge_uint(a, e, 0)
- #define **T_assert_ge_u16**(a, e) T_flags_ge_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_u16**(a, e) T_flags_ge_uint(a, e, T_CHECK_QUIET)
- #define **T_step_ge_u16**(s, a, e) T_flags_ge_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_u16**(s, a, e) T_flags_ge_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_u16**(a, e) T_flags_gt_uint(a, e, 0)
- #define **T_assert_gt_u16**(a, e) T_flags_gt_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_u16**(a, e) T_flags_gt_uint(a, e, T_CHECK_QUIET)
- #define **T_step_gt_u16**(s, a, e) T_flags_gt_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_u16**(s, a, e) T_flags_gt_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_u16**(a, e) T_flags_le_uint(a, e, 0)
- #define **T_assert_le_u16**(a, e) T_flags_le_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_le_u16**(a, e) T_flags_le_uint(a, e, T_CHECK_QUIET)
- #define **T_step_le_u16**(s, a, e) T_flags_le_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_u16**(s, a, e) T_flags_le_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_u16**(a, e) T_flags_lt_uint(a, e, 0)
- #define **T_assert_lt_u16**(a, e) T_flags_lt_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_u16**(a, e) T_flags_lt_uint(a, e, T_CHECK_QUIET)
- #define **T_step_lt_u16**(s, a, e) T_flags_lt_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_u16**(s, a, e) T_flags_lt_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.183.1 Detailed Description

Checks for unsigned 16-bit integers (uint16_t).

## 8.184 Unsigned 32-Bit Integer Checks

Checks for unsigned 32-bit integers (uint32_t).

### Macros

- #define **T_eq_u32**(a, e) T_flags_eq_ulong(a, e, 0)
- #define **T_assert_eq_u32**(a, e) T_flags_eq_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_u32**(a, e) T_flags_eq_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_eq_u32**(s, a, e) T_flags_eq_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_u32**(s, a, e) T_flags_eq_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_u32**(a, e) T_flags_ne_ulong(a, e, 0)
- #define **T_assert_ne_u32**(a, e) T_flags_ne_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_u32**(a, e) T_flags_ne_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_ne_u32**(s, a, e) T_flags_ne_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_u32**(s, a, e) T_flags_ne_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_u32**(a, e) T_flags_ge_ulong(a, e, 0)
- #define **T_assert_ge_u32**(a, e) T_flags_ge_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_u32**(a, e) T_flags_ge_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_ge_u32**(s, a, e) T_flags_ge_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_u32**(s, a, e) T_flags_ge_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_u32**(a, e) T_flags_gt_ulong(a, e, 0)
- #define **T_assert_gt_u32**(a, e) T_flags_gt_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_u32**(a, e) T_flags_gt_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_gt_u32**(s, a, e) T_flags_gt_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_u32**(s, a, e) T_flags_gt_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_u32**(a, e) T_flags_le_ulong(a, e, 0)
- #define **T_assert_le_u32**(a, e) T_flags_le_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_le_u32**(a, e) T_flags_le_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_le_u32**(s, a, e) T_flags_le_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_u32**(s, a, e) T_flags_le_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_u32**(a, e) T_flags_lt_ulong(a, e, 0)
- #define **T_assert_lt_u32**(a, e) T_flags_lt_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_u32**(a, e) T_flags_lt_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_lt_u32**(s, a, e) T_flags_lt_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_u32**(s, a, e) T_flags_lt_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.184.1 Detailed Description

Checks for unsigned 32-bit integers (uint32_t).

# 8.185 Unsigned 64-Bit Integer Checks

Checks for unsigned 64-bit integers (uint64_t).

## Macros

- #define **T_eq_u64**(a, e) T_flags_eq_ull(a, e, 0)
- #define **T_assert_eq_u64**(a, e) T_flags_eq_ull(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_u64**(a, e) T_flags_eq_ull(a, e, T_CHECK_QUIET)
- #define **T_step_eq_u64**(s, a, e) T_flags_eq_ull(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_u64**(s, a, e) T_flags_eq_ull(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_u64**(a, e) T_flags_ne_ull(a, e, 0)
- #define **T_assert_ne_u64**(a, e) T_flags_ne_ull(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_u64**(a, e) T_flags_ne_ull(a, e, T_CHECK_QUIET)
- #define **T_step_ne_u64**(s, a, e) T_flags_ne_ull(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_u64**(s, a, e) T_flags_ne_ull(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_u64**(a, e) T_flags_ge_ull(a, e, 0)
- #define **T_assert_ge_u64**(a, e) T_flags_ge_ull(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_u64**(a, e) T_flags_ge_ull(a, e, T_CHECK_QUIET)
- #define **T_step_ge_u64**(s, a, e) T_flags_ge_ull(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_u64**(s, a, e) T_flags_ge_ull(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_u64**(a, e) T_flags_gt_ull(a, e, 0)
- #define **T_assert_gt_u64**(a, e) T_flags_gt_ull(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_u64**(a, e) T_flags_gt_ull(a, e, T_CHECK_QUIET)
- #define **T_step_gt_u64**(s, a, e) T_flags_gt_ull(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_u64**(s, a, e) T_flags_gt_ull(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_u64**(a, e) T_flags_le_ull(a, e, 0)
- #define **T_assert_le_u64**(a, e) T_flags_le_ull(a, e, T_CHECK_STOP)
- #define **T_quiet_le_u64**(a, e) T_flags_le_ull(a, e, T_CHECK_QUIET)
- #define **T_step_le_u64**(s, a, e) T_flags_le_ull(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_u64**(s, a, e) T_flags_le_ull(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_u64**(a, e) T_flags_lt_ull(a, e, 0)
- #define **T_assert_lt_u64**(a, e) T_flags_lt_ull(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_u64**(a, e) T_flags_lt_ull(a, e, T_CHECK_QUIET)
- #define **T_step_lt_u64**(s, a, e) T_flags_lt_ull(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_u64**(s, a, e) T_flags_lt_ull(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.185.1 Detailed Description

Checks for unsigned 64-bit integers (uint64_t).

## 8.186 Unsigned 8-Bit Integer Checks

Checks for unsigned 8-bit integers (uint8_t).

### Macros

- #define **T_eq_u8**(a, e) T_flags_eq_uint(a, e, 0)
- #define **T_assert_eq_u8**(a, e) T_flags_eq_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_u8**(a, e) T_flags_eq_uint(a, e, T_CHECK_QUIET)
- #define **T_step_eq_u8**(s, a, e) T_flags_eq_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_u8**(s, a, e) T_flags_eq_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_u8**(a, e) T_flags_ne_uint(a, e, 0)
- #define **T_assert_ne_u8**(a, e) T_flags_ne_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_u8**(a, e) T_flags_ne_uint(a, e, T_CHECK_QUIET)
- #define **T_step_ne_u8**(s, a, e) T_flags_ne_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_u8**(s, a, e) T_flags_ne_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_u8**(a, e) T_flags_ge_uint(a, e, 0)
- #define **T_assert_ge_u8**(a, e) T_flags_ge_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_u8**(a, e) T_flags_ge_uint(a, e, T_CHECK_QUIET)
- #define **T_step_ge_u8**(s, a, e) T_flags_ge_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_u8**(s, a, e) T_flags_ge_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_u8**(a, e) T_flags_gt_uint(a, e, 0)
- #define **T_assert_gt_u8**(a, e) T_flags_gt_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_u8**(a, e) T_flags_gt_uint(a, e, T_CHECK_QUIET)
- #define **T_step_gt_u8**(s, a, e) T_flags_gt_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_u8**(s, a, e) T_flags_gt_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_u8**(a, e) T_flags_le_uint(a, e, 0)
- #define **T_assert_le_u8**(a, e) T_flags_le_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_le_u8**(a, e) T_flags_le_uint(a, e, T_CHECK_QUIET)
- #define **T_step_le_u8**(s, a, e) T_flags_le_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_u8**(s, a, e) T_flags_le_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_u8**(a, e) T_flags_lt_uint(a, e, 0)
- #define **T_assert_lt_u8**(a, e) T_flags_lt_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_u8**(a, e) T_flags_lt_uint(a, e, T_CHECK_QUIET)
- #define **T_step_lt_u8**(s, a, e) T_flags_lt_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_u8**(s, a, e) T_flags_lt_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.186.1 Detailed Description

Checks for unsigned 8-bit integers (uint8_t).

## 8.187 Unsigned Character Checks

Checks for unsigned characters (unsigned char).

### Macros

- #define **T_eq_uchar**(a, e) T_flags_eq_uint(a, e, 0)
- #define **T_assert_eq_uchar**(a, e) T_flags_eq_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_uchar**(a, e) T_flags_eq_uint(a, e, T_CHECK_QUIET)
- #define **T_step_eq_uchar**(s, a, e) T_flags_eq_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_uchar**(s, a, e) T_flags_eq_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_uchar**(a, e) T_flags_ne_uint(a, e, 0)
- #define **T_assert_ne_uchar**(a, e) T_flags_ne_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_uchar**(a, e) T_flags_ne_uint(a, e, T_CHECK_QUIET)
- #define **T_step_ne_uchar**(s, a, e) T_flags_ne_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_uchar**(s, a, e) T_flags_ne_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_uchar**(a, e) T_flags_ge_uint(a, e, 0)
- #define **T_assert_ge_uchar**(a, e) T_flags_ge_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_uchar**(a, e) T_flags_ge_uint(a, e, T_CHECK_QUIET)
- #define **T_step_ge_uchar**(s, a, e) T_flags_ge_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_uchar**(s, a, e) T_flags_ge_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_uchar**(a, e) T_flags_gt_uint(a, e, 0)
- #define **T_assert_gt_uchar**(a, e) T_flags_gt_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_uchar**(a, e) T_flags_gt_uint(a, e, T_CHECK_QUIET)
- #define **T_step_gt_uchar**(s, a, e) T_flags_gt_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_uchar**(s, a, e) T_flags_gt_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_uchar**(a, e) T_flags_le_uint(a, e, 0)
- #define **T_assert_le_uchar**(a, e) T_flags_le_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_le_uchar**(a, e) T_flags_le_uint(a, e, T_CHECK_QUIET)
- #define **T_step_le_uchar**(s, a, e) T_flags_le_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_uchar**(s, a, e) T_flags_le_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_uchar**(a, e) T_flags_lt_uint(a, e, 0)
- #define **T_assert_lt_uchar**(a, e) T_flags_lt_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_uchar**(a, e) T_flags_lt_uint(a, e, T_CHECK_QUIET)
- #define **T_step_lt_uchar**(s, a, e) T_flags_lt_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_uchar**(s, a, e) T_flags_lt_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.187.1 Detailed Description

Checks for unsigned characters (unsigned char).

## 8.188 Unsigned Integer Checks

Checks for unsigned integers (unsigned int).

### Macros

- #define **T_eq_uint**(a, e) T_flags_eq_uint(a, e, 0)
- #define **T_assert_eq_uint**(a, e) T_flags_eq_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_uint**(a, e) T_flags_eq_uint(a, e, T_CHECK_QUIET)
- #define **T_step_eq_uint**(s, a, e) T_flags_eq_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_uint**(s, a, e) T_flags_eq_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_uint**(a, e) T_flags_ne_uint(a, e, 0)
- #define **T_assert_ne_uint**(a, e) T_flags_ne_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_uint**(a, e) T_flags_ne_uint(a, e, T_CHECK_QUIET)
- #define **T_step_ne_uint**(s, a, e) T_flags_ne_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_uint**(s, a, e) T_flags_ne_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_uint**(a, e) T_flags_ge_uint(a, e, 0)
- #define **T_assert_ge_uint**(a, e) T_flags_ge_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_uint**(a, e) T_flags_ge_uint(a, e, T_CHECK_QUIET)
- #define **T_step_ge_uint**(s, a, e) T_flags_ge_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_uint**(s, a, e) T_flags_ge_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_uint**(a, e) T_flags_gt_uint(a, e, 0)
- #define **T_assert_gt_uint**(a, e) T_flags_gt_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_uint**(a, e) T_flags_gt_uint(a, e, T_CHECK_QUIET)
- #define **T_step_gt_uint**(s, a, e) T_flags_gt_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_uint**(s, a, e) T_flags_gt_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_uint**(a, e) T_flags_le_uint(a, e, 0)
- #define **T_assert_le_uint**(a, e) T_flags_le_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_le_uint**(a, e) T_flags_le_uint(a, e, T_CHECK_QUIET)
- #define **T_step_le_uint**(s, a, e) T_flags_le_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_uint**(s, a, e) T_flags_le_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_uint**(a, e) T_flags_lt_uint(a, e, 0)
- #define **T_assert_lt_uint**(a, e) T_flags_lt_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_uint**(a, e) T_flags_lt_uint(a, e, T_CHECK_QUIET)
- #define **T_step_lt_uint**(s, a, e) T_flags_lt_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_uint**(s, a, e) T_flags_lt_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.188.1 Detailed Description

Checks for unsigned integers (unsigned int).

# 8.189 Unsigned Long Integer Checks

Checks for unsigned long integers (unsigned long).

## Macros

- #define **T_eq_ulong**(a, e) T_flags_eq_ulong(a, e, 0)
- #define **T_assert_eq_ulong**(a, e) T_flags_eq_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_ulong**(a, e) T_flags_eq_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_eq_ulong**(s, a, e) T_flags_eq_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_ulong**(s, a, e) T_flags_eq_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_ulong**(a, e) T_flags_ne_ulong(a, e, 0)
- #define **T_assert_ne_ulong**(a, e) T_flags_ne_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_ulong**(a, e) T_flags_ne_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_ne_ulong**(s, a, e) T_flags_ne_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_ulong**(s, a, e) T_flags_ne_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_ulong**(a, e) T_flags_ge_ulong(a, e, 0)
- #define **T_assert_ge_ulong**(a, e) T_flags_ge_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_ulong**(a, e) T_flags_ge_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_ge_ulong**(s, a, e) T_flags_ge_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_ulong**(s, a, e) T_flags_ge_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_ulong**(a, e) T_flags_gt_ulong(a, e, 0)
- #define **T_assert_gt_ulong**(a, e) T_flags_gt_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_ulong**(a, e) T_flags_gt_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_gt_ulong**(s, a, e) T_flags_gt_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_ulong**(s, a, e) T_flags_gt_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_ulong**(a, e) T_flags_le_ulong(a, e, 0)
- #define **T_assert_le_ulong**(a, e) T_flags_le_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_le_ulong**(a, e) T_flags_le_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_le_ulong**(s, a, e) T_flags_le_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_ulong**(s, a, e) T_flags_le_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_ulong**(a, e) T_flags_lt_ulong(a, e, 0)
- #define **T_assert_lt_ulong**(a, e) T_flags_lt_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_ulong**(a, e) T_flags_lt_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_lt_ulong**(s, a, e) T_flags_lt_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_ulong**(s, a, e) T_flags_lt_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.189.1 Detailed Description

Checks for unsigned long integers (unsigned long).

## 8.190 Unsigned Long Long Integer Checks

Checks for unsigned long long integers (unsigned long long).

### Macros

- #define **T_eq_ull**(a, e) T_flags_eq_ull(a, e, 0)
- #define **T_assert_eq_ull**(a, e) T_flags_eq_ull(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_ull**(a, e) T_flags_eq_ull(a, e, T_CHECK_QUIET)
- #define **T_step_eq_ull**(s, a, e) T_flags_eq_ull(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_ull**(s, a, e) T_flags_eq_ull(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_ull**(a, e) T_flags_ne_ull(a, e, 0)
- #define **T_assert_ne_ull**(a, e) T_flags_ne_ull(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_ull**(a, e) T_flags_ne_ull(a, e, T_CHECK_QUIET)
- #define **T_step_ne_ull**(s, a, e) T_flags_ne_ull(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_ull**(s, a, e) T_flags_ne_ull(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_ull**(a, e) T_flags_ge_ull(a, e, 0)
- #define **T_assert_ge_ull**(a, e) T_flags_ge_ull(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_ull**(a, e) T_flags_ge_ull(a, e, T_CHECK_QUIET)
- #define **T_step_ge_ull**(s, a, e) T_flags_ge_ull(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_ull**(s, a, e) T_flags_ge_ull(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_ull**(a, e) T_flags_gt_ull(a, e, 0)
- #define **T_assert_gt_ull**(a, e) T_flags_gt_ull(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_ull**(a, e) T_flags_gt_ull(a, e, T_CHECK_QUIET)
- #define **T_step_gt_ull**(s, a, e) T_flags_gt_ull(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_ull**(s, a, e) T_flags_gt_ull(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_ull**(a, e) T_flags_le_ull(a, e, 0)
- #define **T_assert_le_ull**(a, e) T_flags_le_ull(a, e, T_CHECK_STOP)
- #define **T_quiet_le_ull**(a, e) T_flags_le_ull(a, e, T_CHECK_QUIET)
- #define **T_step_le_ull**(s, a, e) T_flags_le_ull(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_ull**(s, a, e) T_flags_le_ull(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_ull**(a, e) T_flags_lt_ull(a, e, 0)
- #define **T_assert_lt_ull**(a, e) T_flags_lt_ull(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_ull**(a, e) T_flags_lt_ull(a, e, T_CHECK_QUIET)
- #define **T_step_lt_ull**(s, a, e) T_flags_lt_ull(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_ull**(s, a, e) T_flags_lt_ull(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.190.1 Detailed Description

Checks for unsigned long long integers (unsigned long long).

## 8.191 Unsigned Pointer Value Checks

Checks for unsigned pointer values (uintptr_t).

### Macros

- #define **T_eq_uptr**(a, e) T_flags_eq_ulong(a, e, 0)
- #define **T_assert_eq_uptr**(a, e) T_flags_eq_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_uptr**(a, e) T_flags_eq_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_eq_uptr**(s, a, e) T_flags_eq_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_uptr**(s, a, e) T_flags_eq_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_uptr**(a, e) T_flags_ne_ulong(a, e, 0)
- #define **T_assert_ne_uptr**(a, e) T_flags_ne_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_uptr**(a, e) T_flags_ne_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_ne_uptr**(s, a, e) T_flags_ne_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_uptr**(s, a, e) T_flags_ne_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_uptr**(a, e) T_flags_ge_ulong(a, e, 0)
- #define **T_assert_ge_uptr**(a, e) T_flags_ge_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_uptr**(a, e) T_flags_ge_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_ge_uptr**(s, a, e) T_flags_ge_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_uptr**(s, a, e) T_flags_ge_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_uptr**(a, e) T_flags_gt_ulong(a, e, 0)
- #define **T_assert_gt_uptr**(a, e) T_flags_gt_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_uptr**(a, e) T_flags_gt_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_gt_uptr**(s, a, e) T_flags_gt_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_uptr**(s, a, e) T_flags_gt_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_uptr**(a, e) T_flags_le_ulong(a, e, 0)
- #define **T_assert_le_uptr**(a, e) T_flags_le_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_le_uptr**(a, e) T_flags_le_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_le_uptr**(s, a, e) T_flags_le_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_uptr**(s, a, e) T_flags_le_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_uptr**(a, e) T_flags_lt_ulong(a, e, 0)
- #define **T_assert_lt_uptr**(a, e) T_flags_lt_ulong(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_uptr**(a, e) T_flags_lt_ulong(a, e, T_CHECK_QUIET)
- #define **T_step_lt_uptr**(s, a, e) T_flags_lt_ulong(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_uptr**(s, a, e) T_flags_lt_ulong(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.191.1 Detailed Description

Checks for unsigned pointer values (uintptr_t).

## 8.192 Unsigned Short Integer Checks

Checks for unsigned short integers (unsigned short).

### Macros

- #define **T_eq_ushort**(a, e) T_flags_eq_uint(a, e, 0)
- #define **T_assert_eq_ushort**(a, e) T_flags_eq_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_eq_ushort**(a, e) T_flags_eq_uint(a, e, T_CHECK_QUIET)
- #define **T_step_eq_ushort**(s, a, e) T_flags_eq_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_eq_ushort**(s, a, e) T_flags_eq_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ne_ushort**(a, e) T_flags_ne_uint(a, e, 0)
- #define **T_assert_ne_ushort**(a, e) T_flags_ne_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_ne_ushort**(a, e) T_flags_ne_uint(a, e, T_CHECK_QUIET)
- #define **T_step_ne_ushort**(s, a, e) T_flags_ne_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ne_ushort**(s, a, e) T_flags_ne_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_ge_ushort**(a, e) T_flags_ge_uint(a, e, 0)
- #define **T_assert_ge_ushort**(a, e) T_flags_ge_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_ge_ushort**(a, e) T_flags_ge_uint(a, e, T_CHECK_QUIET)
- #define **T_step_ge_ushort**(s, a, e) T_flags_ge_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_ge_ushort**(s, a, e) T_flags_ge_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_gt_ushort**(a, e) T_flags_gt_uint(a, e, 0)
- #define **T_assert_gt_ushort**(a, e) T_flags_gt_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_gt_ushort**(a, e) T_flags_gt_uint(a, e, T_CHECK_QUIET)
- #define **T_step_gt_ushort**(s, a, e) T_flags_gt_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_gt_ushort**(s, a, e) T_flags_gt_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_le_ushort**(a, e) T_flags_le_uint(a, e, 0)
- #define **T_assert_le_ushort**(a, e) T_flags_le_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_le_ushort**(a, e) T_flags_le_uint(a, e, T_CHECK_QUIET)
- #define **T_step_le_ushort**(s, a, e) T_flags_le_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_le_ushort**(s, a, e) T_flags_le_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)
- #define **T_lt_ushort**(a, e) T_flags_lt_uint(a, e, 0)
- #define **T_assert_lt_ushort**(a, e) T_flags_lt_uint(a, e, T_CHECK_STOP)
- #define **T_quiet_lt_ushort**(a, e) T_flags_lt_uint(a, e, T_CHECK_QUIET)
- #define **T_step_lt_ushort**(s, a, e) T_flags_lt_uint(a, e, T_CHECK_STEP(s))
- #define **T_step_assert_lt_ushort**(s, a, e) T_flags_lt_uint(a, e, T_CHECK_STEP(s) | T_CHECK_STOP)

### 8.192.1 Detailed Description

Checks for unsigned short integers (unsigned short).

# 8.193 User Extension Handler

The User Extension Handler provides invocation of application dependent routines at critical points in the life of each thread and the system as a whole.

## Files

- file userext.h

  *User Extension Handler API.*
- file userextdata.h

  *User Extension Handler Data Structures.*
- file userextimpl.h

  *User Extension Handler API.*
- file userext.c

  *User Extension Handler implementation.*
- file userextaddset.c

  *User Extension Handler implementation.*
- file userextiterate.c

  *User Extension Iteration Helpers.*
- file userextremoveset.c

  *User Extension Handler implementation.*

## Classes

- struct User_extensions_Table

  *User extension table.*
- struct User_extensions_Switch_control

  *Manages the switch callouts.*
- struct User_extensions_Control

  *Manages each user extension set.*
- struct User_extensions_Iterator

  *Chain iterator for dynamic user extensions.*
- struct User_extensions_List
- struct User_extensions_Thread_create_context
- struct User_extensions_Fatal_context

## Typedefs

- typedef bool(∗ User_extensions_thread_create_extension) (struct _Thread_Control ∗executing, struct _Thread_Control ∗created)

  *Task create extension.*
- typedef void(∗ User_extensions_thread_delete_extension) (struct _Thread_Control ∗executing, struct _Thread_Control ∗deleted)

  *Task delete extension.*
- typedef void(∗ User_extensions_thread_start_extension) (struct _Thread_Control ∗executing, struct _Thread_Control ∗started)

  *Task start extension.*
- typedef void(∗ User_extensions_thread_restart_extension) (struct _Thread_Control ∗executing, struct _Thread_Control ∗restarted)

*Task restart extension.*

- typedef void(∗ User_extensions_thread_switch_extension) (struct _Thread_Control ∗executing, struct _Thread_Control ∗heir)

  *Task switch extension.*
- typedef void(∗ User_extensions_thread_begin_extension) (struct _Thread_Control ∗executing)

  *Task begin extension.*
- typedef void(∗ User_extensions_thread_exitted_extension) (struct _Thread_Control ∗executing)

  *Task exitted extension.*
- typedef void(∗ User_extensions_fatal_extension) (Internal_errors_Source source, bool always_set_to_false, Internal_errors_t code)

  *Fatal error extension.*
- typedef void(∗ User_extensions_thread_terminate_extension) (struct _Thread_Control ∗terminated)

  *Task termination extension.*
- typedef struct User_extensions_Iterator User_extensions_Iterator

  *Chain iterator for dynamic user extensions.*

## Variables

- const size_t _User_extensions_Initial_count

  *The count of initial user extensions.*
- const User_extensions_Table _User_extensions_Initial_extensions [ ]

  *The table of initial user extensions.*
- User_extensions_Switch_control _User_extensions_Initial_switch_controls [ ]

  *A spare switch control for each initial user extension.*
- User_extensions_List _User_extensions_List

  *List of active extensions.*
- Chain_Control _User_extensions_Switches_list

  *List of active task switch extensions.*

## Extension Maintainance

- typedef void(∗ User_extensions_Visitor) (Thread_Control ∗executing, void ∗arg, const User_extensions_Table ∗callouts)

  *User extension visitor.*
- void _User_extensions_Handler_initialization (void)

  *Initializes the user extensions handler.*
- void _User_extensions_Add_set (User_extensions_Control ∗extension)

  *Adds a user extension.*
- static __inline__ void _User_extensions_Add_API_set (User_extensions_Control ∗extension)

  *Adds a user extension.*
- static __inline__ void _User_extensions_Add_set_with_table (User_extensions_Control ∗extension, const User_extensions_Table ∗extension_table)

  *Adds a user extension with the given extension table as callouts.*
- void _User_extensions_Remove_set (User_extensions_Control ∗extension)

  *Removes a user extension.*
- void _User_extensions_Thread_create_visitor (Thread_Control ∗executing, void ∗arg, const User_extensions_Table ∗callouts)

  *Creates a visitor.*
- void _User_extensions_Thread_delete_visitor (Thread_Control ∗executing, void ∗arg, const User_extensions_Table ∗callouts)

*Deletes a visitor.*

- void _User_extensions_Thread_start_visitor (Thread_Control *executing, void *arg, const User_extensions_Table *callouts)

  *Starts a visitor.*
- void _User_extensions_Thread_restart_visitor (Thread_Control *executing, void *arg, const User_extensions_Table *callouts)

  *Restarts a visitor.*
- void _User_extensions_Thread_begin_visitor (Thread_Control *executing, void *arg, const User_extensions_Table *callouts)

  *Calls the begin function of the thread callout for the visitor.*
- void _User_extensions_Thread_exitted_visitor (Thread_Control *executing, void *arg, const User_extensions_Table *callouts)

  *Calls the exitted function of the thread callout for the visitor.*
- void _User_extensions_Fatal_visitor (Thread_Control *executing, void *arg, const User_extensions_Table *callouts)

  *Calls the fatal function of the thread callout for the visitor.*
- void _User_extensions_Thread_terminate_visitor (Thread_Control *executing, void *arg, const User_extensions_Table *callouts)

  *Terminates a visitor.*
- void _User_extensions_Iterate (void *arg, User_extensions_Visitor visitor, Chain_Iterator_direction direction)

  *Iterates through all user extensions and calls the visitor for each.*

## Extension Callout Dispatcher

- static bool _User_extensions_Thread_create (Thread_Control *created)

  *Creates a thread.*
- static void _User_extensions_Thread_delete (Thread_Control *deleted)

  *Deletes a thread.*
- static void _User_extensions_Thread_start (Thread_Control *started)

  *Starts a thread.*
- static void _User_extensions_Thread_restart (Thread_Control *restarted)

  *Restarts a thread.*
- static void _User_extensions_Thread_begin (Thread_Control *executing)

  *Begins a thread.*
- static void _User_extensions_Thread_switch (Thread_Control *executing, Thread_Control *heir)

  *Switches the thread from the executing to the heir.*
- static void _User_extensions_Thread_exitted (Thread_Control *executing)

  *A user extension thread exitted.*
- static void _User_extensions_Fatal (Internal_errors_Source source, Internal_errors_t error)

  *Forwards all visitors that there was a fatal failure.*
- static void _User_extensions_Thread_terminate (Thread_Control *executing)

  *Terminates the executing thread.*
- static void _User_extensions_Acquire (ISR_lock_Context *lock_context)

  *Disables interrupts and acquires the lock context.*
- static void _User_extensions_Release (ISR_lock_Context *lock_context)

  *Releases the lock context and enables interrupts.*
- static void _User_extensions_Destroy_iterators (Thread_Control *the_thread)

  *Destroys all user extension iterators of a thread.*

### 8.193.1 Detailed Description

The User Extension Handler provides invocation of application dependent routines at critical points in the life of each thread and the system as a whole.

### 8.193.2 Typedef Documentation

#### 8.193.2.1 User_extensions_fatal_extension

```
typedef void( * User_extensions_fatal_extension) (Internal_errors_Source source, bool always_↩
set_to_false, Internal_errors_t code)
```

Fatal error extension.

It corresponds to _Terminate() (used by the rtems_fatal() directive).

This extension should not call any RTEMS directives.

**Parameters**

| | | |
|---|---|---|
| in | *source* | The fatal source indicating the subsystem the fatal condition originated in. |
| in | *always_set_to_false* | This parameter is always set to false and provided only for backward compatibility reasons. |
| in | *code* | The fatal error code. This value must be interpreted with respect to the source. |

Definition at line 201 of file userext.h.

#### 8.193.2.2 User_extensions_Iterator

```
typedef struct User_extensions_Iterator User_extensions_Iterator
```

Chain iterator for dynamic user extensions.

Since user extensions may delete or restart the executing thread, we must clean up registered iterators.

**See also**

_User_extensions_Iterate(), _User_extensions_Destroy_iterators() and Thread_Control::last_user_extensions_iterator.

#### 8.193.2.3 User_extensions_thread_begin_extension

```
typedef void( * User_extensions_thread_begin_extension) (struct _Thread_Control *executing)
```

Task begin extension.

It corresponds to _Thread_Handler().

Thread dispatching is disabled. The executing thread is not the holder of the allocator mutex.

**Parameters**

| | | |
|---|---|---|
| in | *executing* | The executing thread. |

Definition at line 169 of file userext.h.

### 8.193.2.4 User_extensions_thread_create_extension

```
typedef bool( * User_extensions_thread_create_extension) (struct _Thread_Control *executing,
struct _Thread_Control *created)
```

Task create extension.

It corresponds to _Thread_Initialize() (used by the rtems_task_create() directive and pthread_create()).

It is invoked after the new thread has been completely initialized, but before it is placed on a ready chain.

Thread dispatching may be disabled. This depends on the context of the _Thread_Initialize() call. Thread dispatch is disabled during the creation of the idle thread and the initialization threads. It can be considered as an invalid API usage, if the application calls _Thread_Initialize() with disabled thread dispatching. Disabled thread dispatching is different from disabled preemption.

It can be assumed that the executing thread locked the allocator mutex. The only exception is the creation of the idle thread. In this case the allocator mutex is not locked. Since the allocator mutex allows nesting the normal memory allocation routines can be used.

**Parameters**

| | | |
|---|---|---|
| in | *executing* | The executing thread. |
| in | *created* | The created thread. |

**Return values**

| | |
|---|---|
| *true* | Successful operation. |
| *false* | A thread create user extension will frequently attempt to allocate resources. If this allocation fails, then the extension should return *false* and the entire thread create operation will fail. |

Definition at line 69 of file userext.h.

### 8.193.2.5 User_extensions_thread_delete_extension

```
typedef void( * User_extensions_thread_delete_extension) (struct _Thread_Control *executing,
struct _Thread_Control *deleted)
```

Task delete extension.

It corresponds to _Thread_Close() (used by the rtems_task_delete() directive, pthread_exit() and pthread_cancel()).

It is invoked before all resources of the thread are deleted. The executing and deleted arguments are never equal.

Thread dispatching is enabled. The executing thread locked the allocator mutex.

**Parameters**

| | | |
|---|---|---|
| in | *executing* | The executing thread. |
| in | *deleted* | The deleted thread. |

Definition at line 89 of file userext.h.

#### 8.193.2.6 User_extensions_thread_exitted_extension

```
typedef void( * User_extensions_thread_exitted_extension) (struct _Thread_Control *executing)
```

Task exitted extension.

It corresponds to _Thread_Handler() after a return of the entry function.

Thread dispatching is disabled. The state of the allocator mutex is arbitrary.

**Parameters**

| | | |
|---|---|---|
| in | *executing* | The executing thread. |

Definition at line 183 of file userext.h.

#### 8.193.2.7 User_extensions_thread_restart_extension

```
typedef void( * User_extensions_thread_restart_extension) (struct _Thread_Control *executing,
struct _Thread_Control *restarted)
```

Task restart extension.

It corresponds to _Thread_Restart() (used by the rtems_task_restart() directive).

It is invoked in the context of the restarted thread right before the execution context is reloaded. The executing and restarted arguments are always equal. The thread stack reflects the previous execution context.

Thread dispatching is enabled. The thread is not the holder of the allocator mutex. The thread life is protected. Thread restart and delete requests issued by restart extensions lead to recursion.

**Parameters**

| | | |
|---|---|---|
| in | *executing* | The executing thread. |
| in | *restarted* | The executing thread. Yes, the executing thread. |

Definition at line 131 of file userext.h.

#### 8.193.2.8 User_extensions_thread_start_extension

```
typedef void( * User_extensions_thread_start_extension) (struct _Thread_Control *executing,
struct _Thread_Control *started)
```

Task start extension.

It corresponds to _Thread_Start() (used by the rtems_task_start() directive).

It is invoked after the environment of the thread has been loaded and the thread has been made ready.

Thread dispatching is disabled. The executing thread is not the holder of the allocator mutex.

**Parameters**

| | | |
|---|---|---|
| in | *executing* | The executing thread. |
| in | *started* | The started thread. |

Definition at line 109 of file userext.h.

#### 8.193.2.9 User_extensions_thread_switch_extension

```
typedef void( * User_extensions_thread_switch_extension) (struct _Thread_Control *executing,
struct _Thread_Control *heir)
```

Task switch extension.

It corresponds to _Thread_Dispatch().

It is invoked before the context switch from the executing to the heir thread.

Thread dispatching is disabled. The state of the allocator mutex is arbitrary. Interrupts are disabled and the per-CPU lock is acquired on SMP configurations.

The context switches initiated through _Thread_Start_multitasking() are not covered by this extension.

**Parameters**

| | | |
|---|---|---|
| in | *executing* | The executing thread. |
| in | *heir* | The heir thread. |

Definition at line 154 of file userext.h.

#### 8.193.2.10 User_extensions_thread_terminate_extension

```
typedef void( * User_extensions_thread_terminate_extension) (struct _Thread_Control *terminated)
```

Task termination extension.

This extension is invoked by _Thread_Life_action_handler() in case a termination request is recognized.

It is invoked in the context of the terminated thread right before the thread dispatch to the heir thread. The POSIX cleanup and key destructors execute in this context.

Thread dispatching is enabled. The thread is not the holder of the allocator mutex. The thread life is protected. Thread restart and delete requests issued by terminate extensions lead to recursion.

**Parameters**

| | | |
|---|---|---|
| in | *terminated* | The terminated thread. |

Definition at line 223 of file userext.h.

#### 8.193.2.11 User_extensions_Visitor

```
typedef void(* User_extensions_Visitor) (Thread_Control *executing, void *arg, const User_extensions_Table
*callouts)
```

User extension visitor.

**Parameters**

| | | |
|---|---|---|
| in,out | *executing* | The currently executing thread. |
| in,out | *arg* | The argument passed to _User_extensions_Iterate(). |
| in | *callouts* | The current callouts. |

Definition at line 143 of file userextimpl.h.

### 8.193.3 Function Documentation

#### 8.193.3.1 _User_extensions_Acquire()

```
static void _User_extensions_Acquire (
            ISR_lock_Context * lock_context )  [inline], [static]
```

Disables interrupts and acquires the lock context.

**Parameters**

| | |
|---|---|
| *lock_context* | The lock context to acquire. |

Definition at line 473 of file userextimpl.h.

#### 8.193.3.2 _User_extensions_Add_API_set()

```
static __inline__ void _User_extensions_Add_API_set (
            User_extensions_Control * extension )  [static]
```

Adds a user extension.

**Parameters**

| | |
|---|---|
| *extension* | The user extension to add. |

Definition at line 104 of file userextimpl.h.

#### 8.193.3.3 _User_extensions_Add_set()

```
void _User_extensions_Add_set (
            User_extensions_Control * extension )
```

Adds a user extension.

**Parameters**

| | |
|---|---|
| *extension* | The user extension to add. |

Definition at line 45 of file userextaddset.c.

#### 8.193.3.4 _User_extensions_Add_set_with_table()

```
static __inline__ void _User_extensions_Add_set_with_table (
            User_extensions_Control * extension,
            const User_extensions_Table * extension_table )  [static]
```

Adds a user extension with the given extension table as callouts.

**Parameters**

| | | |
|---|---|---|
| in,out | *extension* | The user extension to add. |
| | *extension_table* | Is set as callouts for *extension*. |

Definition at line 117 of file userextimpl.h.

#### 8.193.3.5 _User_extensions_Destroy_iterators()

```
static void _User_extensions_Destroy_iterators (
              Thread_Control * the_thread )  [inline], [static]
```

Destroys all user extension iterators of a thread.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_thread* | The thread to destroy all user extension iterators of. |

Definition at line 500 of file userextimpl.h.

#### 8.193.3.6 _User_extensions_Fatal()

```
static void _User_extensions_Fatal (
              Internal_errors_Source source,
              Internal_errors_t error )  [inline], [static]
```

Forwards all visitors that there was a fatal failure.

**Parameters**

| | |
|---|---|
| *source* | The error source. |
| *error* | The error. |

Definition at line 438 of file userextimpl.h.

#### 8.193.3.7 _User_extensions_Fatal_visitor()

```
void _User_extensions_Fatal_visitor (
              Thread_Control * executing,
              void * arg,
              const User_extensions_Table * callouts )
```

Calls the fatal function of the thread callout for the visitor.

**Parameters**

| | |
|---|---|
| *executing* | The currently executing thread. |
| *arg* | Is used as the user extension fatal context. |
| *callouts* | The user extension table for the operation. |

Definition at line 120 of file userextiterate.c.

#### 8.193.3.8 _User_extensions_Iterate()

```
void _User_extensions_Iterate (
              void * arg,
              User_extensions_Visitor visitor,
              Chain_Iterator_direction direction )
```

Iterates through all user extensions and calls the visitor for each.

**Parameters**

| | | |
|---|---|---|
| in,out | *arg* | The argument passed to the visitor. |
| | *visitor* | The visitor for each extension. |
| | *direction* | The iteration direction for dynamic extensions. |

Definition at line 149 of file userextiterate.c.

#### 8.193.3.9 _User_extensions_Release()

```
static void _User_extensions_Release (
              ISR_lock_Context * lock_context )  [inline], [static]
```

Releases the lock context and enables interrupts.

**Parameters**

| | |
|---|---|
| *lock_context* | The lock context to release. |

Definition at line 486 of file userextimpl.h.

#### 8.193.3.10 _User_extensions_Remove_set()

```
void _User_extensions_Remove_set (
              User_extensions_Control * extension )
```

Removes a user extension.

**Parameters**

| | |
|---|---|
| *extension* | The user extension to remove. |

Definition at line 25 of file userextremoveset.c.

### 8.193.3.11 _User_extensions_Thread_begin()

```
static void _User_extensions_Thread_begin (
            Thread_Control * executing )  [inline], [static]
```

Begins a thread.

**Parameters**

| | |
|---|---|
| *created* | The thread to begin. |

Definition at line 351 of file userextimpl.h.

### 8.193.3.12 _User_extensions_Thread_begin_visitor()

```
void _User_extensions_Thread_begin_visitor (
            Thread_Control * executing,
            void * arg,
            const User_extensions_Table * callouts )
```

Calls the begin function of the thread callout for the visitor.

**Parameters**

| | |
|---|---|
| *executing* | The currently executing thread. |
| *arg* | This parameter is unused. |
| *callouts* | The user extension table for the operation. |

Definition at line 94 of file userextiterate.c.

### 8.193.3.13 _User_extensions_Thread_create()

```
static bool _User_extensions_Thread_create (
            Thread_Control * created )  [inline], [static]
```

Creates a thread.

**Parameters**

| | | |
|---|---|---|
| out | *created* | The thread to create. |

**Return values**

| | |
|---|---|
| *true* | The operation succeeded. |
| *false* | The operation failed. |

Definition at line 291 of file userextimpl.h.

#### 8.193.3.14 _User_extensions_Thread_create_visitor()

```
void _User_extensions_Thread_create_visitor (
            Thread_Control * executing,
            void * arg,
            const User_extensions_Table * callouts )
```

Creates a visitor.

**Parameters**

| | | |
|---|---|---|
| | *executing* | The currently executing thread. |
| in, out | *arg* | Is used as the thread create context for the operation. |
| | *callouts* | The user extension table for the operation. |

Definition at line 40 of file userextiterate.c.

#### 8.193.3.15 _User_extensions_Thread_delete()

```
static void _User_extensions_Thread_delete (
            Thread_Control * deleted )  [inline], [static]
```

Deletes a thread.

**Parameters**

| | | |
|---|---|---|
| out | *created* | The thread to delete. |

Definition at line 309 of file userextimpl.h.

#### 8.193.3.16 _User_extensions_Thread_delete_visitor()

```
void _User_extensions_Thread_delete_visitor (
            Thread_Control * executing,
            void * arg,
            const User_extensions_Table * callouts )
```

Deletes a visitor.

**Parameters**

| | | |
|---|---|---|
| | *executing* | The currently executing thread. |
| in, out | *arg* | Parameter for the callout. |
| | *callouts* | The user extension table for the operation. |

Definition at line 55 of file userextiterate.c.

#### 8.193.3.17 _User_extensions_Thread_exitted()

```
static void _User_extensions_Thread_exitted (
              Thread_Control * executing )  [inline], [static]
```

A user extension thread exitted.

**Parameters**

| | |
|---|---|
| *created* | The thread. |

Definition at line 423 of file userextimpl.h.

#### 8.193.3.18 _User_extensions_Thread_exitted_visitor()

```
void _User_extensions_Thread_exitted_visitor (
              Thread_Control * executing,
              void * arg,
              const User_extensions_Table * callouts )
```

Calls the exitted function of the thread callout for the visitor.

**Parameters**

| | |
|---|---|
| *executing* | The currently executing thread. |
| *arg* | This parameter is unused. |
| *callouts* | The user extension table for the operation. |

Definition at line 107 of file userextiterate.c.

#### 8.193.3.19 _User_extensions_Thread_restart()

```
static void _User_extensions_Thread_restart (
              Thread_Control * restarted )  [inline], [static]
```

Restarts a thread.

**Parameters**

| | |
|---|---|
| *created* | The thread to restart. |

Definition at line 337 of file userextimpl.h.

#### 8.193.3.20 _User_extensions_Thread_restart_visitor()

```
void _User_extensions_Thread_restart_visitor (
            Thread_Control * executing,
            void * arg,
            const User_extensions_Table * callouts )
```

Restarts a visitor.

**Parameters**

| | |
|---|---|
| *executing* | The currently executing thread. |
| *arg* | Parameter for the callout. |
| *callouts* | The user extension table for the operation. |

Definition at line 81 of file userextiterate.c.

#### 8.193.3.21 _User_extensions_Thread_start()

```
static void _User_extensions_Thread_start (
            Thread_Control * started )  [inline], [static]
```

Starts a thread.

**Parameters**

| | |
|---|---|
| *created* | The thread to start. |

Definition at line 323 of file userextimpl.h.

#### 8.193.3.22 _User_extensions_Thread_start_visitor()

```
void _User_extensions_Thread_start_visitor (
            Thread_Control * executing,
            void * arg,
            const User_extensions_Table * callouts )
```

Starts a visitor.

**Parameters**

| | |
|---|---|
| *executing* | The currently executing thread. |
| *arg* | Parameter for the callout. |
| *callouts* | The user extension table for the operation. |

Definition at line 68 of file userextiterate.c.

#### 8.193.3.23 _User_extensions_Thread_switch()

```
static void _User_extensions_Thread_switch (
              Thread_Control * executing,
              Thread_Control * heir )  [inline], [static]
```

Switches the thread from the executing to the heir.

**Parameters**

| | |
|---|---|
| *executing* | The currently executing thread. |
| *heir* | The thread that will switch with *executing*. |

Definition at line 366 of file userextimpl.h.

#### 8.193.3.24 _User_extensions_Thread_terminate()

```
static void _User_extensions_Thread_terminate (
              Thread_Control * executing )  [inline], [static]
```

Terminates the executing thread.

**Parameters**

| | |
|---|---|
| *executing* | The currently executing thread. |

Definition at line 457 of file userextimpl.h.

#### 8.193.3.25 _User_extensions_Thread_terminate_visitor()

```
void _User_extensions_Thread_terminate_visitor (
              Thread_Control * executing,
              void * arg,
              const User_extensions_Table * callouts )
```

Terminates a visitor.

**Parameters**

| | |
|---|---|
| *executing* | The currently executing thread. |
| *arg* | This parameter is unused. |
| *callouts* | The user extension table for the operation. |

Definition at line 135 of file userextiterate.c.

## 8.193.4 Variable Documentation

### 8.193.4.1 _User_extensions_Initial_count

```
const size_t _User_extensions_Initial_count  [extern]
```

The count of initial user extensions.

Application provided via <rtems/confdefs.h>.

### 8.193.4.2 _User_extensions_Initial_extensions

```
const User_extensions_Table _User_extensions_Initial_extensions[]  [extern]
```

The table of initial user extensions.

Application provided via <rtems/confdefs.h>.

### 8.193.4.3 _User_extensions_Initial_switch_controls

```
User_extensions_Switch_control _User_extensions_Initial_switch_controls[]  [extern]
```

A spare switch control for each initial user extension.

Application provided via <rtems/confdefs.h>.

# 8.194 User Extensions Implementation

## Files

- file extensiondata.h

  *Classic User Extensions Data Structures.*
- file extensionimpl.h

  *Classic User Extensions Implementation.*
- file extensionident.c

  *rtems_extension_ident() Implementation*

## Classes

- struct Extension_Control

## Macros

- #define EXTENSION_INFORMATION_DEFINE(max)

  *Macro to define the objects information for the Classic Extensions objects.*

## Functions

- static __inline__ Extension_Control * **_Extension_Allocate** (void)
- static __inline__ void **_Extension_Free** (Extension_Control *the_extension)
- static __inline__ Extension_Control * **_Extension_Get** (Objects_Id id)

## Variables

- Objects_Information _Extension_Information

  *The Classic Extensions objects information.*

## 8.194.1 Detailed Description

## 8.194.2 Macro Definition Documentation

### 8.194.2.1 EXTENSION_INFORMATION_DEFINE

```
#define EXTENSION_INFORMATION_DEFINE(
              max )
```

**Value:**

```
  OBJECTS_INFORMATION_DEFINE( \
    _Extension, \
    OBJECTS_CLASSIC_API, \
    OBJECTS_RTEMS_EXTENSIONS, \
    Extension_Control, \
    max, \
    OBJECTS_NO_STRING_NAME, \
    NULL \
  )
```

Macro to define the objects information for the Classic Extensions objects.

This macro should only be used by <rtems/confdefs.h>.

**Parameters**

| | |
|---|---|
| *max* | The configured object maximum (the OBJECTS_UNLIMITED_OBJECTS flag may be set). |

Definition at line 54 of file extensiondata.h.

# 8.195 User Extensions Manager

The User Extensions Manager allows the application developer to augment the executive by allowing them to supply extension routines which are invoked at critical system events.

## Typedefs

- typedef User_extensions_fatal_extension rtems_fatal_extension
  %
- typedef Internal_errors_t rtems_fatal_code
  %
- typedef Internal_errors_Source rtems_fatal_source
  %
- typedef User_extensions_Table rtems_extensions_table
  %
- typedef User_extensions_thread_begin_extension rtems_task_begin_extension
  %
- typedef User_extensions_thread_create_extension rtems_task_create_extension
  %
- typedef User_extensions_thread_delete_extension rtems_task_delete_extension
  %
- typedef User_extensions_thread_exitted_extension rtems_task_exitted_extension
  %
- typedef User_extensions_thread_restart_extension rtems_task_restart_extension
  %
- typedef User_extensions_thread_start_extension rtems_task_start_extension
  %
- typedef User_extensions_thread_switch_extension rtems_task_switch_extension
  %
- typedef User_extensions_thread_terminate_extension rtems_task_terminate_extension
  %

## Functions

- rtems_status_code rtems_extension_delete (rtems_id id)
  %
- rtems_status_code rtems_extension_ident (rtems_name name, rtems_id *id)
  *Identifies an extension set object by the specified object name.*
- rtems_status_code rtems_extension_create (rtems_name name, const rtems_extensions_table *extension↩
  _table, rtems_id *id)
  %

### 8.195.1 Detailed Description

The User Extensions Manager allows the application developer to augment the executive by allowing them to supply extension routines which are invoked at critical system events.

## 8.195.2 Function Documentation

### 8.195.2.1 rtems_extension_create()

```
rtems_status_code rtems_extension_create (
            rtems_name name,
            const rtems_extensions_table * extension_table,
            rtems_id * id )
```

%

**Parameters**

| | |
|---|---|
| *name* | % |
| *extension_table* | % |
| *id* | % |

Definition at line 27 of file extensioncreate.c.

### 8.195.2.2 rtems_extension_delete()

```
rtems_status_code rtems_extension_delete (
            rtems_id id )
```

%

**Parameters**

| | |
|---|---|
| *id* | % |

Definition at line 25 of file extensiondelete.c.

### 8.195.2.3 rtems_extension_ident()

```
rtems_status_code rtems_extension_ident (
            rtems_name name,
            rtems_id * id )
```

Identifies an extension set object by the specified object name.

This directive obtains the extension set identifier associated with the extension set name specified in `name`.

If the extension set name is not unique, then the extension set identifier will match the first extension set with that name in the search order. However, this extension set identifier is not guaranteed to correspond to the desired extension set. The extension set identifier is used with other extension related directives to access the extension set.

The objects are searched from lowest to the highest index. Only the local node is searched.

**Parameters**

| | | |
|---|---|---|
| | *name* | is the object name to look up. |
| out | *id* | is the pointer to an object identifier variable. The object identifier of an object with the specified name will be stored in this variable, in case of a successful operation. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | The requested operation was successful. |
| *RTEMS_INVALID_ADDRESS* | The id parameter was NULL. |
| *RTEMS_INVALID_NAME* | The name parameter was 0. |
| *RTEMS_INVALID_NAME* | There was no object with the specified name on the local node. |

Definition at line 44 of file extensionident.c.

## 8.196 Version

The Version API provides functions to return the version or parts of the version of RTEMS you are using.

### Files

- file version.c

  *Creates the version strings from the various pieces of version information. The main version number is part of the build system and is stamped into rtems/score/cpuopts.h. The version control key string is extracted from the version control tool when the code is being built and is updated if it has changed. The key may indicate there are local modification.*

### Functions

- const char * rtems_version (void)

  *Returns the version string.*
- int rtems_version_major (void)

  *Returns the version's major number.*
- int rtems_version_minor (void)

  *Returns the version's minor number.*
- int rtems_version_revision (void)

  *Returns the version's revision number.*
- const char * rtems_version_control_key (void)

  *Returns the version control key for the current version of code that has been built.*
- static bool rtems_version_control_key_is_valid (const char *key)

  *Returns true, if the version control key is valid, otherwise false.*
- const char * rtems_board_support_package (void)

  *Returns the board support package name.*

### 8.196.1 Detailed Description

The Version API provides functions to return the version or parts of the version of RTEMS you are using.

### 8.196.2 Function Documentation

#### 8.196.2.1 rtems_board_support_package()

```
const char* rtems_board_support_package (
            void  )
```

Returns the board support package name.

**Returns**

The board support package name.

Definition at line 33 of file rtems-version.c.

### 8.196.2.2 rtems_version()

```
const char* rtems_version (
            void  )
```

Returns the version string.

**Return values**

| | |
|---|---|
| *text* | The version as a string. |

Definition at line 32 of file version.c.

### 8.196.2.3 rtems_version_control_key()

```
const char* rtems_version_control_key (
            void  )
```

Returns the version control key for the current version of code that has been built.

The key is specific to the version control system being used and allows the built version to be identified.

Use rtems_version_control_key_is_valid() to check if the version control key is valid.

**Returns**

The version control key.

Definition at line 56 of file version.c.

### 8.196.2.4 rtems_version_control_key_is_valid()

```
static bool rtems_version_control_key_is_valid (
            const char * key )  [inline], [static]
```

Returns true, if the version control key is valid, otherwise false.

**Return values**

| | |
|---|---|
| *true* | The version control key is valid. |
| *false* | Otherwise. |

Definition at line 83 of file version.h.

#### 8.196.2.5 rtems_version_major()

```
int rtems_version_major (
            void  )
```

Returns the version's major number.

**Return values**

| | |
|---|---|
| *int* | The version's major number. |

Definition at line 41 of file version.c.

#### 8.196.2.6 rtems_version_minor()

```
int rtems_version_minor (
            void  )
```

Returns the version's minor number.

**Return values**

| | |
|---|---|
| *int* | The version's minor number. |

Definition at line 46 of file version.c.

#### 8.196.2.7 rtems_version_revision()

```
int rtems_version_revision (
            void  )
```

Returns the version's revision number.

**Return values**

| | |
|---|---|
| *int* | The version's revision number. |

Definition at line 51 of file version.c.

# 8.197 Watchdog Handler

Watchdog Handler.

## Files

- file watchdog.h

  *Constants and Structures Associated with Watchdog Timers.*
- file watchdogimpl.h

  *Inlined Routines in the Watchdog Handler.*
- file watchdogticks.h

  *Constants for the watchdog ticks.*
- file watchdoginsert.c

  *Watchdog Insert.*
- file watchdogremove.c

  *Remove Watchdog.*
- file watchdogtickssinceboot.c

  *Watchdog Ticks Since Boot.*

## Classes

- struct Watchdog_Header

  *The watchdog header to manage scheduled watchdogs.*
- struct Watchdog_Control

  *The control block used to manage each watchdog timer.*

## Macros

- #define WATCHDOG_INITIALIZER(routine)

  *Watchdog initializer for static initialization.*
- #define **_Watchdog_Tickle**(header, first, now, lock, lock_context) _Watchdog_Do_tickle( header, first, now, lock, lock_context )
- #define WATCHDOG_MAXIMUM_TICKS UINT64_MAX

  *The maximum watchdog ticks value for the far future.*
- #define **WATCHDOG_NANOSECONDS_PER_SECOND** 1000000000
- #define WATCHDOG_BITS_FOR_1E9_NANOSECONDS 30

  *The bits necessary to store 1000000000 (= WATCHDOG_NANOSECONDS_PER_SECOND) nanoseconds.*
- #define WATCHDOG_MAX_SECONDS 0x3fffffff

  *The maximum number of seconds representable in the nanoseconds watchdog format.*
- #define WATCHDOG_NO_TIMEOUT 0

  *Special watchdog ticks value to indicate an infinite wait.*
- #define WATCHDOG_TICKS_PER_TIMESLICE_DEFAULT 50

  *Default value for the watchdog ticks per timeslice.*

## Typedefs

- typedef struct Watchdog_Control **Watchdog_Control**
- typedef void Watchdog_Service_routine

  *Return type from a Watchdog Service Routine.*
- typedef Watchdog_Service_routine(∗ Watchdog_Service_routine_entry) (Watchdog_Control ∗)

  *Pointer to a watchdog service routine.*
- typedef uint32_t Watchdog_Interval

  *Type is used to specify the length of intervals.*

## Enumerations

- enum Watchdog_State { WATCHDOG_SCHEDULED_BLACK, WATCHDOG_SCHEDULED_RED, WATCHDOG_INACTIVE, WATCHDOG_PENDING }

  *Watchdog states.*

## Functions

- static __inline__ void _Watchdog_Header_initialize (Watchdog_Header ∗header)

  *Initializes the watchdog header.*
- static __inline__ Watchdog_Control ∗ _Watchdog_Header_first (const Watchdog_Header ∗header)

  *Returns the first of the watchdog header.*
- static __inline__ void _Watchdog_Header_destroy (Watchdog_Header ∗header)

  *Destroys the watchdog header.*
- void _Watchdog_Tick (struct Per_CPU_Control ∗cpu)

  *Performs a watchdog tick.*
- static __inline__ Watchdog_State _Watchdog_Get_state (const Watchdog_Control ∗the_watchdog)

  *Gets the state of the watchdog.*
- static __inline__ void _Watchdog_Set_state (Watchdog_Control ∗the_watchdog, Watchdog_State state)

  *Sets the state of the watchdog.*
- static __inline__ Per_CPU_Control ∗ _Watchdog_Get_CPU (const Watchdog_Control ∗the_watchdog)

  *Gets the watchdog's cpu.*
- static __inline__ void _Watchdog_Set_CPU (Watchdog_Control ∗the_watchdog, Per_CPU_Control ∗cpu)

  *Sets the cpu for the watchdog.*
- static __inline__ void _Watchdog_Preinitialize (Watchdog_Control ∗the_watchdog, Per_CPU_Control ∗cpu)

  *Pre-initializes a watchdog.*
- static __inline__ void _Watchdog_Initialize (Watchdog_Control ∗the_watchdog, Watchdog_Service_routine_entry routine)

  *Initializes a watchdog with a new service routine.*
- void _Watchdog_Do_tickle (Watchdog_Header ∗header, Watchdog_Control ∗first, uint64_t now, ISR_lock_Control ∗lock, ISR_lock_Context ∗lock_context)

  *Calls the routine of each not expired watchdog control node.*
- void _Watchdog_Insert (Watchdog_Header ∗header, Watchdog_Control ∗the_watchdog, uint64_t expire)

  *Inserts a watchdog into the set of scheduled watchdogs according to the specified expiration time.*
- void _Watchdog_Remove (Watchdog_Header ∗header, Watchdog_Control ∗the_watchdog)

  *In the case the watchdog is scheduled, then it is removed from the set of scheduled watchdogs.*
- static __inline__ uint64_t _Watchdog_Cancel (Watchdog_Header ∗header, Watchdog_Control ∗the_↩ watchdog, uint64_t now)

  *In the case the watchdog is scheduled, then it is removed from the set of scheduled watchdogs.*
- static __inline__ bool _Watchdog_Is_scheduled (const Watchdog_Control ∗the_watchdog)

*Checks if the watchdog is scheduled.*

- static __inline__ void _Watchdog_Next_first (Watchdog_Header *header, Watchdog_Control *the_↩ watchdog)

  *Sets the first node of the header.*

- static __inline__ bool _Watchdog_Is_valid_timespec (const struct timespec *ts)

  *Checks if the timespec is a valid timespec for a watchdog.*

- static __inline__ bool _Watchdog_Is_valid_interval_timespec (const struct timespec *ts)

  *Checks if the timespec is a valid interval timespec for a watchdog.*

- static __inline__ const struct timespec * _Watchdog_Future_timespec (struct timespec *now, const struct timespec *delta)

  *Adds the delta timespec to the current time if the delta is a valid interval timespec.*

- static __inline__ bool _Watchdog_Is_far_future_timespec (const struct timespec *ts)

  *Checks if the timespec is too far in the future.*

- static __inline__ uint64_t _Watchdog_Ticks_from_seconds (uint32_t seconds)

  *Converts the seconds to ticks.*

- static __inline__ uint64_t _Watchdog_Ticks_from_timespec (const struct timespec *ts)

  *Converts the timespec in ticks.*

- static __inline__ uint64_t _Watchdog_Ticks_from_sbintime (int64_t sbt)

  *Converts the sbintime in ticks.*

- static __inline__ void _Watchdog_Per_CPU_acquire_critical (Per_CPU_Control *cpu, ISR_lock_Context *lock_context)

  *Acquires the per cpu watchdog lock in a critical section.*

- static __inline__ void _Watchdog_Per_CPU_release_critical (Per_CPU_Control *cpu, ISR_lock_Context *lock_context)

  *Releases the per cpu watchdog lock in a critical section.*

- static __inline__ uint64_t _Watchdog_Per_CPU_insert_ticks (Watchdog_Control *the_watchdog, Per_CPU_Control *cpu, Watchdog_Interval ticks)

  *Sets the watchdog's cpu to the given instance and sets its expiration time to the watchdog expiration time of the cpu plus the ticks.*

- static __inline__ uint64_t _Watchdog_Per_CPU_insert (Watchdog_Control *the_watchdog, Per_CPU_Control *cpu, Watchdog_Header *header, uint64_t expire)

  *Sets the watchdog's cpu and inserts it with the given expiration time in the scheduled watchdogs.*

- static __inline__ void _Watchdog_Per_CPU_remove (Watchdog_Control *the_watchdog, Per_CPU_Control *cpu, Watchdog_Header *header)

  *Removes the watchdog from the cpu and the scheduled watchdogs.*

- static __inline__ void _Watchdog_Per_CPU_remove_ticks (Watchdog_Control *the_watchdog)

  *Removes the watchdog from the cpu and the scheduled watchdogs.*

## Variables

- volatile Watchdog_Interval _Watchdog_Ticks_since_boot

  *The watchdog ticks counter.*

- const uint32_t _Watchdog_Microseconds_per_tick

  *The watchdog microseconds per tick.*

- const uint32_t _Watchdog_Nanoseconds_per_tick

  *The watchdog nanoseconds per tick.*

- const uint32_t _Watchdog_Ticks_per_second

  *The watchdog ticks per second.*

- const uint32_t _Watchdog_Ticks_per_timeslice

  *The watchdog ticks per timeslice.*

### 8.197.1 Detailed Description

Watchdog Handler.

This handler encapsulates functionality related to the scheduling of watchdog functions to be called at specific times in the future.

**Note**

This handler does not have anything to do with hardware watchdog timers.

### 8.197.2 Macro Definition Documentation

#### 8.197.2.1 WATCHDOG_BITS_FOR_1E9_NANOSECONDS

```
#define WATCHDOG_BITS_FOR_1E9_NANOSECONDS 30
```

The bits necessary to store 1000000000 (= WATCHDOG_NANOSECONDS_PER_SECOND) nanoseconds.

The expiration time is an unsigned 64-bit integer. To store nanoseconds timeouts we use 30 bits (2∗∗30 == 1073741824) for the nanoseconds and 34 bits for the seconds since UNIX Epoch. This leads to a year 2514 problem.

Definition at line 401 of file watchdogimpl.h.

#### 8.197.2.2 WATCHDOG_INITIALIZER

```
#define WATCHDOG_INITIALIZER(
              routine )
```

**Value:**

```
{ \
  { { { NULL, NULL, NULL, WATCHDOG_INACTIVE } } }, \
  &_Per_CPU_Information[ 0 ].per_cpu, \
  ( routine ), \
  0 \
}
```

Watchdog initializer for static initialization.

The processor of this watchdog is set to processor with index zero.

**See also**

_Watchdog_Preinitialize().

Definition at line 79 of file watchdogimpl.h.

#### 8.197.2.3 WATCHDOG_MAX_SECONDS

```
#define WATCHDOG_MAX_SECONDS 0x3fffffff
```

The maximum number of seconds representable in the nanoseconds watchdog format.

We have 2**34 bits for the seconds part.

Definition at line 409 of file watchdogimpl.h.

### 8.197.3 Typedef Documentation

#### 8.197.3.1 Watchdog_Interval

```
typedef uint32_t Watchdog_Interval
```

Type is used to specify the length of intervals.

This type is used to specify the length of intervals.

Definition at line 38 of file watchdogticks.h.

#### 8.197.3.2 Watchdog_Service_routine

```
typedef void Watchdog_Service_routine
```

Return type from a Watchdog Service Routine.

This type defines the return type from a Watchdog Service Routine.

Definition at line 58 of file watchdog.h.

#### 8.197.3.3 Watchdog_Service_routine_entry

```
typedef Watchdog_Service_routine( * Watchdog_Service_routine_entry) (Watchdog_Control *)
```

Pointer to a watchdog service routine.

This type define a pointer to a watchdog service routine.

Definition at line 65 of file watchdog.h.

### 8.197.4 Enumeration Type Documentation

#### 8.197.4.1 Watchdog_State

```
enum Watchdog_State
```

Watchdog states.

**Enumerator**

| | |
|---|---|
| WATCHDOG_SCHEDULED_BLACK | The watchdog is scheduled and a black node in the red-black tree. |
| WATCHDOG_SCHEDULED_RED | The watchdog is scheduled and a red node in the red-black tree. |
| WATCHDOG_INACTIVE | The watchdog is inactive. |
| WATCHDOG_PENDING | The watchdog is on a chain of pending watchdogs. This state is used by the timer server for example. |

Definition at line 47 of file watchdogimpl.h.

## 8.197.5 Function Documentation

### 8.197.5.1 _Watchdog_Cancel()

```
static __inline__ uint64_t _Watchdog_Cancel (
            Watchdog_Header * header,
            Watchdog_Control * the_watchdog,
            uint64_t now )  [static]
```

In the case the watchdog is scheduled, then it is removed from the set of scheduled watchdogs.

The watchdog must be initialized before this call.

**Parameters**

| | | |
|---|---|---|
| in,out | *header* | The scheduled watchdogs. |
| in,out | *the_watchdog* | The watchdog to remove. |
| | *now* | The current time. |

**Return values**

| | |
|---|---|
| *other* | The difference of the now and expiration time. |
| *0* | The now time is greater than or equal to the expiration time of the watchdog. |

Definition at line 318 of file watchdogimpl.h.

### 8.197.5.2 _Watchdog_Do_tickle()

```
void _Watchdog_Do_tickle (
            Watchdog_Header * header,
            Watchdog_Control * first,
            uint64_t now,
            ISR_lock_Control * lock,
            ISR_lock_Context * lock_context )
```

Calls the routine of each not expired watchdog control node.

**Parameters**

| | |
|---|---|
| *header* | The watchdog header. |
| *first* | The first watchdog control node. |
| *now* | The current time to check the expiration time against. |
| *lock* | The lock that is released before calling the routine and then acquired after the call. |
| *lock_context* | The lock context for the release before calling the routine and for the acquire after. |

Definition at line 24 of file watchdogtick.c.

### 8.197.5.3 _Watchdog_Future_timespec()

```
static __inline__ const struct timespec* _Watchdog_Future_timespec (
            struct timespec * now,
            const struct timespec * delta )  [static]
```

Adds the delta timespec to the current time if the delta is a valid interval timespec.

**Parameters**

| | | |
|---|---|---|
| in,out | *now* | The current time. |
| | *delta* | The delta timespec for the addition. |

**Return values**

| | |
|---|---|
| *pointer* | Pointer to the now timespec. |
| *NULL* | *delta* is not a valid interval timespec. |

Definition at line 452 of file watchdogimpl.h.

### 8.197.5.4 _Watchdog_Get_CPU()

```
static __inline__ Per_CPU_Control* _Watchdog_Get_CPU (
            const Watchdog_Control * the_watchdog )  [static]
```

Gets the watchdog's cpu.

**Parameters**

| | |
|---|---|
| *the_watchdog* | The watchdog to get the cpu of. |

**Returns**

The cpu of the watchdog.

Definition at line 177 of file watchdogimpl.h.

#### 8.197.5.5 _Watchdog_Get_state()

```
static __inline__ Watchdog_State _Watchdog_Get_state (
            const Watchdog_Control * the_watchdog )  [static]
```

Gets the state of the watchdog.

**Parameters**

| | |
|---|---|
| *the_watchdog* | The watchdog to get the state of. |

**Returns**

The RB_COLOR of *the_watchdog*.

Definition at line 149 of file watchdogimpl.h.

#### 8.197.5.6 _Watchdog_Header_destroy()

```
static __inline__ void _Watchdog_Header_destroy (
            Watchdog_Header * header )  [static]
```

Destroys the watchdog header.

**Parameters**

| | |
|---|---|
| *header* | The watchdog header to destroy. |

Definition at line 127 of file watchdogimpl.h.

#### 8.197.5.7 _Watchdog_Header_first()

```
static __inline__ Watchdog_Control* _Watchdog_Header_first (
            const Watchdog_Header * header )  [static]
```

Returns the first of the watchdog header.

**Parameters**

| | |
|---|---|
| *header* | The watchdog header to remove the first of. |

**Returns**

The first of *header*.

Definition at line 115 of file watchdogimpl.h.

### 8.197.5.8 _Watchdog_Header_initialize()

```
static __inline__ void _Watchdog_Header_initialize (
            Watchdog_Header * header )  [static]
```

Initializes the watchdog header.

**Parameters**

| | | |
|---|---|---|
| out | *header* | The header to initialize. |

Definition at line 100 of file watchdogimpl.h.

### 8.197.5.9 _Watchdog_Initialize()

```
static __inline__ void _Watchdog_Initialize (
            Watchdog_Control * the_watchdog,
            Watchdog_Service_routine_entry routine )  [static]
```

Initializes a watchdog with a new service routine.

The watchdog must be inactive.

**Parameters**

| | | |
|---|---|---|
| out | *the_watchdog* | The watchdog to initialize. |
| | *routing* | The service routine for *the_watchdog*. |

Definition at line 236 of file watchdogimpl.h.

### 8.197.5.10 _Watchdog_Insert()

```
void _Watchdog_Insert (
            Watchdog_Header * header,
            Watchdog_Control * the_watchdog,
            uint64_t expire )
```

Inserts a watchdog into the set of scheduled watchdogs according to the specified expiration time.

The watchdog must be inactive.

**Parameters**

| | | |
|---|---|---|
| in,out | *header* | The set of scheduler watchdogs to insert into. |
| in,out | *the_watchdog* | The watchdog to insert. |
| | *expire* | The expiration time for the watchdog. |

Definition at line 29 of file watchdoginsert.c.

### 8.197.5.11 _Watchdog_Is_far_future_timespec()

```
static __inline__ bool _Watchdog_Is_far_future_timespec (
            const struct timespec * ts )  [static]
```

Checks if the timespec is too far in the future.

**Parameters**

| | |
|---|---|
| *ts* | The timespec for the verification. |

**Return values**

| | |
|---|---|
| *true* | *ts* is too far in the future. |
| *false* | *ts* is not too far in the future. |

Definition at line 490 of file watchdogimpl.h.

### 8.197.5.12 _Watchdog_Is_scheduled()

```
static __inline__ bool _Watchdog_Is_scheduled (
            const Watchdog_Control * the_watchdog )  [static]
```

Checks if the watchdog is scheduled.

**Parameters**

| | |
|---|---|
| *the_watchdog* | The watchdog for the verification. |

**Return values**

| | |
|---|---|
| *true* | The watchdog is scheduled. |
| *false* | The watchdog is inactive. |

Definition at line 348 of file watchdogimpl.h.

### 8.197.5.13 _Watchdog_Is_valid_interval_timespec()

```
static __inline__ bool _Watchdog_Is_valid_interval_timespec (
            const struct timespec * ts )  [static]
```

Checks if the timespec is a valid interval timespec for a watchdog.

**Parameters**

| | |
|---|---|
| *ts* | The timespec for the verification. |

**Return values**

| | |
|---|---|
| *true* | The timespec is a valid interval timespec. |
| *false* | The timespec is invalid. |

Definition at line 435 of file watchdogimpl.h.

### 8.197.5.14 _Watchdog_Is_valid_timespec()

```
static __inline__ bool _Watchdog_Is_valid_timespec (
            const struct timespec * ts )  [static]
```

Checks if the timespec is a valid timespec for a watchdog.

**Parameters**

| | |
|---|---|
| *ts* | The timespec for the verification. |

**Return values**

| | |
|---|---|
| *true* | The timespec is a valid timespec. |
| *false* | The timespec is invalid. |

Definition at line 419 of file watchdogimpl.h.

### 8.197.5.15 _Watchdog_Next_first()

```
static __inline__ void _Watchdog_Next_first (
            Watchdog_Header * header,
            Watchdog_Control * the_watchdog )  [static]
```

Sets the first node of the header.

Sets the first node of the header to either the leftmost child node of the watchdog control node, or if not present sets it to the right child node of the watchdog control node. if both are not present, the new first node is the parent node of the current first node.

**Parameters**

| | | |
|---|---|---|
| in,out | *header* | The watchdog header. |
| | *the_watchdog* | The watchdog control node for the operation. |

Definition at line 366 of file watchdogimpl.h.

#### 8.197.5.16 _Watchdog_Per_CPU_acquire_critical()

```
static __inline__ void _Watchdog_Per_CPU_acquire_critical (
            Per_CPU_Control * cpu,
            ISR_lock_Context * lock_context )  [static]
```

Acquires the per cpu watchdog lock in a critical section.

**Parameters**

| | |
|---|---|
| *cpu* | The cpu to acquire the watchdog lock of. |
| *lock_context* | The lock context. |

Definition at line 561 of file watchdogimpl.h.

#### 8.197.5.17 _Watchdog_Per_CPU_insert()

```
static __inline__ uint64_t _Watchdog_Per_CPU_insert (
            Watchdog_Control * the_watchdog,
            Per_CPU_Control * cpu,
            Watchdog_Header * header,
            uint64_t expire )  [static]
```

Sets the watchdog's cpu and inserts it with the given expiration time in the scheduled watchdogs.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_watchdog* | The watchdog to set cpu and expiration time of. |
| | *cpu* | The cpu for the operation. |
| in,out | *header* | The scheduled watchdogs. |
| | *expire* | The expiration time for the watchdog. |

**Returns**

The expiration time of the watchdog.

Definition at line 625 of file watchdogimpl.h.

### 8.197.5.18 _Watchdog_Per_CPU_insert_ticks()

```
static __inline__ uint64_t _Watchdog_Per_CPU_insert_ticks (
             Watchdog_Control * the_watchdog,
             Per_CPU_Control * cpu,
             Watchdog_Interval ticks )  [static]
```

Sets the watchdog's cpu to the given instance and sets its expiration time to the watchdog expiration time of the cpu plus the ticks.

**Parameters**

| in,out | | |
|---|---|---|
| | *the_watchdog* | The watchdog to set the cpu and expiration time of. |
| | *cpu* | The cpu for the watchdog. |
| | *ticks* | The ticks to add to the expiration time. |

**Returns**

The new expiration time of the watchdog.

Definition at line 593 of file watchdogimpl.h.

### 8.197.5.19 _Watchdog_Per_CPU_release_critical()

```
static __inline__ void _Watchdog_Per_CPU_release_critical (
             Per_CPU_Control * cpu,
             ISR_lock_Context * lock_context )  [static]
```

Releases the per cpu watchdog lock in a critical section.

**Parameters**

| | |
|---|---|
| *cpu* | The cpu to release the watchdog lock of. |
| *lock_context* | The lock context. |

Definition at line 575 of file watchdogimpl.h.

#### 8.197.5.20 _Watchdog_Per_CPU_remove()

```
static __inline__ void _Watchdog_Per_CPU_remove (
            Watchdog_Control * the_watchdog,
            Per_CPU_Control * cpu,
            Watchdog_Header * header )  [static]
```

Removes the watchdog from the cpu and the scheduled watchdogs.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_watchdog* | The watchdog to remove. |
| | *cpu* | The cpu to remove the watchdog from. |
| in,out | *The* | scheduled watchdogs. |

Definition at line 649 of file watchdogimpl.h.

#### 8.197.5.21 _Watchdog_Per_CPU_remove_ticks()

```
static __inline__ void _Watchdog_Per_CPU_remove_ticks (
            Watchdog_Control * the_watchdog )  [static]
```

Removes the watchdog from the cpu and the scheduled watchdogs.

**Parameters**

| | | |
|---|---|---|
| in,out | *the_watchdog* | The watchdog to remove. |

Definition at line 670 of file watchdogimpl.h.

#### 8.197.5.22 _Watchdog_Preinitialize()

```
static __inline__ void _Watchdog_Preinitialize (
            Watchdog_Control * the_watchdog,
            Per_CPU_Control * cpu )  [static]
```

Pre-initializes a watchdog.

This routine must be called before a watchdog is used in any way. The exception are statically initialized watchdogs via WATCHDOG_INITIALIZER().

**Parameters**

| | | |
|---|---|---|
| out | *the_watchdog* | The uninitialized watchdog. |

Definition at line 214 of file watchdogimpl.h.

#### 8.197.5.23 _Watchdog_Remove()

```
void _Watchdog_Remove (
            Watchdog_Header * header,
            Watchdog_Control * the_watchdog )
```

In the case the watchdog is scheduled, then it is removed from the set of scheduled watchdogs.

The watchdog must be initialized before this call.

**Parameters**

| | | |
|---|---|---|
| in,out | *header* | The scheduled watchdogs. |
| in,out | *the_watchdog* | The watchdog to remove. |

Definition at line 29 of file watchdogremove.c.

#### 8.197.5.24 _Watchdog_Set_CPU()

```
static __inline__ void _Watchdog_Set_CPU (
            Watchdog_Control * the_watchdog,
            Per_CPU_Control * cpu )  [static]
```

Sets the cpu for the watchdog.

**Parameters**

| | | |
|---|---|---|
| out | *the_watchdog* | The watchdog to set the cpu of. |
| | *cpu* | The cpu to be set as *the_watchdog's* cpu. |

Definition at line 194 of file watchdogimpl.h.

#### 8.197.5.25 _Watchdog_Set_state()

```
static __inline__ void _Watchdog_Set_state (
            Watchdog_Control * the_watchdog,
            Watchdog_State state )  [static]
```

Sets the state of the watchdog.

**Parameters**

| | | |
|---|---|---|
| out | *the_watchdog* | The watchdog to set the state of. |
| | *state* | The state to set the watchdog to. |

Definition at line 162 of file watchdogimpl.h.

### 8.197.5.26 _Watchdog_Tick()

```
void _Watchdog_Tick (
            struct Per_CPU_Control * cpu )
```

Performs a watchdog tick.

**Parameters**

| | |
|---|---|
| *cpu* | The processor for this watchdog tick. |

Definition at line 54 of file watchdogtick.c.

### 8.197.5.27 _Watchdog_Ticks_from_sbintime()

```
static __inline__ uint64_t _Watchdog_Ticks_from_sbintime (
            int64_t sbt )  [static]
```

Converts the sbintime in ticks.

**Parameters**

| | |
|---|---|
| *sbt* | The sbintime to convert to ticks. |

**Returns**

*sbt* converted to ticks.

Definition at line 546 of file watchdogimpl.h.

### 8.197.5.28 _Watchdog_Ticks_from_seconds()

```
static __inline__ uint64_t _Watchdog_Ticks_from_seconds (
            uint32_t seconds )  [static]
```

Converts the seconds to ticks.

**Parameters**

| | |
|---|---|
| *seconds* | The seconds to convert to ticks. |

**Returns**

*seconds* converted to ticks.

Definition at line 504 of file watchdogimpl.h.

#### 8.197.5.29 _Watchdog_Ticks_from_timespec()

```
static __inline__ uint64_t _Watchdog_Ticks_from_timespec (
            const struct timespec * ts )  [static]
```

Converts the timespec in ticks.

**Parameters**

| | |
|---|---|
| *ts* | The timespec to convert to ticks. |

**Returns**

*ts* converted to ticks.

Definition at line 522 of file watchdogimpl.h.

### 8.197.6 Variable Documentation

#### 8.197.6.1 _Watchdog_Microseconds_per_tick

```
const uint32_t _Watchdog_Microseconds_per_tick  [extern]
```

The watchdog microseconds per tick.

This constant is defined by the application configuration via <rtems/confdefs.h>.

#### 8.197.6.2 _Watchdog_Nanoseconds_per_tick

```
const uint32_t _Watchdog_Nanoseconds_per_tick  [extern]
```

The watchdog nanoseconds per tick.

This constant is defined by the application configuration via <rtems/confdefs.h>.

#### 8.197.6.3 _Watchdog_Ticks_per_second

```
const uint32_t _Watchdog_Ticks_per_second  [extern]
```

The watchdog ticks per second.

This constant is defined by the application configuration via <rtems/confdefs.h>.

#### 8.197.6.4 _Watchdog_Ticks_per_timeslice

```
const uint32_t _Watchdog_Ticks_per_timeslice  [extern]
```

The watchdog ticks per timeslice.

This constant is defined by the application configuration via <rtems/confdefs.h>.

Definition at line 34 of file watchdogtimeslicedefault.c.

#### 8.197.6.5 _Watchdog_Ticks_since_boot

```
volatile Watchdog_Interval _Watchdog_Ticks_since_boot  [extern]
```

The watchdog ticks counter.

With a 1ms watchdog tick, this counter overflows after 50 days since boot.

Definition at line 29 of file watchdogtickssinceboot.c.

# 8.198 Workspace Handler

## Files

- file wkspaceinitone.c

  *_Workspace_Handler_initialization() Implementation*
- file wkspace.h

  *Information Related to the RAM Workspace.*
- file wkspacedata.h

  *Constants defined by the application configuration for the idle threads.*
- file wkspaceinitone.h

  *Workspace Handler Initialization API.*
- file wkspace.c

  *Workspace Handler System Initialization.*
- file wkspaceallocate.c

  *_Workspace_Allocate() Implementation*
- file wkspacemallocinitdefault.c

  *This source file provides the default definition of _Workspace_Malloc_initializer.*
- file wkspacemallocinitunified.c

  *This source file provides the implementation of _Workspace_Malloc_initialize_unified().*

## Functions

- void _Workspace_Handler_initialization (void)

  *Initializes the workspace handler.*
- void * _Workspace_Allocate (size_t size)

  *Allocates a memory block of the specified size from the workspace.*
- void _Workspace_Free (void *block)

  *Frees memory to the workspace.*
- char * _Workspace_String_duplicate (const char *string, size_t len)

  *Duplicates string with memory from the workspace.*
- struct Heap_Control * _Workspace_Malloc_initialize_separate (void)

  *Initializes the C Program Heap separated from the RTEMS Workspace.*
- struct Heap_Control * _Workspace_Malloc_initialize_unified (void)

  *Initializes the C Program Heap so that it is unified with the RTEMS Workspace.*
- static __inline__ void **_Workspace_Initialize_with_one_area** (void)

## Variables

- Heap_Control _Workspace_Area

  *Executive workspace control.*
- const uintptr_t _Workspace_Size

  *The workspace size in bytes.*
- const bool _Workspace_Is_unified

  *Indicates if the workspace and C program heap are unified.*
- struct Heap_Control *(*const _Workspace_Malloc_initializer )(void)

  *This constant provides the C Program Heap initialization handler.*

## 8.198.1 Detailed Description

This handler encapsulates functionality related to the management of the RTEMS Executive Workspace.

## 8.198.2 Function Documentation

### 8.198.2.1 _Workspace_Allocate()

```
void* _Workspace_Allocate (
            size_t size )
```

Allocates a memory block of the specified size from the workspace.

**Parameters**

| | |
|---|---|
| *size* | The size of the memory block. |

**Return values**

| | |
|---|---|
| *pointer* | The pointer to the memory block. The pointer is at least aligned by CPU_HEAP_ALIGNMENT. |
| *NULL* | No memory block with the requested size is available in the workspace. |

Definition at line 43 of file wkspaceallocate.c.

### 8.198.2.2 _Workspace_Free()

```
void _Workspace_Free (
            void * block )
```

Frees memory to the workspace.

This function frees the specified block of memory.

**Parameters**

| | |
|---|---|
| *block* | The memory to free. |

**Note**

If *block* is equal to NULL, then the request is ignored. This allows the caller to not worry about whether or not a pointer is NULL.

### 8.198.2.3 _Workspace_Handler_initialization()

```
void _Workspace_Handler_initialization (
             void  )
```

Initializes the workspace handler.

This routine performs the initialization necessary for this handler.

Definition at line 42 of file wkspaceinitone.c.

### 8.198.2.4 _Workspace_Malloc_initialize_separate()

```
struct Heap_Control* _Workspace_Malloc_initialize_separate (
             void  )
```

Initializes the C Program Heap separated from the RTEMS Workspace.

**Returns**

Returns the heap control used for the C Program Heap.

Definition at line 45 of file mallocinitone.c.

### 8.198.2.5 _Workspace_Malloc_initialize_unified()

```
struct Heap_Control* _Workspace_Malloc_initialize_unified (
             void  )
```

Initializes the C Program Heap so that it is unified with the RTEMS Workspace.

**Returns**

Returns the heap control used for the C Program Heap.

Definition at line 44 of file wkspacemallocinitunified.c.

### 8.198.2.6 _Workspace_String_duplicate()

```
char* _Workspace_String_duplicate (
             const char * string,
             size_t len )
```

Duplicates string with memory from the workspace.

**Parameters**

| | |
|---|---|
| *string* | The pointer to a zero terminated string. |
| *len* | The length of the string (equal to strlen(string)). |

**Return values**

| | |
|---|---|
| *other* | Duplicated string. |
| *NULL* | Not enough memory. |

## 8.198.3 Variable Documentation

### 8.198.3.1 _Workspace_Area

```
Heap_Control _Workspace_Area  [extern]
```

Executive workspace control.

This is the heap control structure used to manage the RTEMS Executive Workspace.

Definition at line 43 of file wkspace.c.

### 8.198.3.2 _Workspace_Is_unified

```
const bool _Workspace_Is_unified  [extern]
```

Indicates if the workspace and C program heap are unified.

This constant is defined by the application configuration via <rtems/confdefs.h>.

Definition at line 34 of file wkspaceisunifieddefault.c.

### 8.198.3.3 _Workspace_Malloc_initializer

```
struct Heap_Control*( * const _Workspace_Malloc_initializer) (void)  [extern]
```

This constant provides the C Program Heap initialization handler.

This constant is defined by the application configuration option CONFIGURE_UNIFIED_WORK_AREAS via <rtems/confdefs.h> or a default configuration.

Definition at line 1 of file wkspacemallocinitdefault.c.

### 8.198.3.4 _Workspace_Size

```
const uintptr_t _Workspace_Size  [extern]
```

The workspace size in bytes.

This constant is defined by the application configuration via <rtems/confdefs.h>.

## 8.199 libfs

libfs

libfs

## 8.200 nfs Client

nfs Client

nfs Client

# 8.201 spec:/rtems/attr/val/attr

Tests the attribute constants of the Classic API.

## Files

- file tc-attr.c

## Functions

- static bool **IsPowerOfTwo** (rtems_attribute attribute)
- static int **PopCount** (rtems_attribute attributes)
- void **T_case_body_RtemsAttrValAttr** (void)

### 8.201.1 Detailed Description

Tests the attribute constants of the Classic API.

This test case performs the following actions:

- Validate the non-default attribute constants.
  - Check that RTEMS_BARRIER_AUTOMATIC_RELEASE is a power of two.
  - Check that RTEMS_BINARY_SEMAPHORE is a power of two.
  - Check that RTEMS_FLOATING_POINT is a power of two.
  - Check that RTEMS_GLOBAL is a power of two.
  - Check that RTEMS_INHERIT_PRIORITY is a power of two.
  - Check that RTEMS_MULTIPROCESSOR_RESOURCE_SHARING is a power of two.
  - Check that RTEMS_PRIORITY is a power of two.
  - Check that RTEMS_PRIORITY_CEILING is a power of two.
  - Check that RTEMS_SIMPLE_BINARY_SEMAPHORE is a power of two.
  - Check that RTEMS_SYSTEM_TASK is a power of two.
- Validate the default attribute constants.
  - Check that RTEMS_APPLICATION_TASK is equal to zero.
  - Check that RTEMS_BARRIER_MANUAL_RELEASE is equal to zero.
  - Check that RTEMS_COUNTING_SEMAPHORE is equal to zero.
  - Check that RTEMS_DEFAULT_ATTRIBUTES is equal to zero.
  - Check that RTEMS_FIFO is equal to zero.
  - Check that RTEMS_LOCAL is equal to zero.
  - Check that RTEMS_NO_FLOATING_POINT is equal to zero.
  - Check that RTEMS_NO_INHERIT_PRIORITY is equal to zero.
  - Check that RTEMS_NO_MULTIPROCESSOR_RESOURCE_SHARING is equal to zero.
  - Check that RTEMS_NO_PRIORITY_CEILING is equal to zero.
- Calculate the bitwise or of all non-default attribute constants.
  - Check that the count of set bits in the calculated value is equal to the count of non-default attribute constants. Since each non-default attribute constant is a power of two, this proves that each constant has a unique value.
- Calculate the bitwise or of the RTEMS_BINARY_SEMAPHORE, RTEMS_COUNTING_SEMAPHORE, and RTEMS_SIMPLE_BINARY_SEMAPHORE attribute constants.
  - Check that the calculated value is equal to RTEMS_SEMAPHORE_CLASS.

## 8.202 spec:/rtems/barrier/val/ident

Test the rtems_barrier_ident directive.

### Files

- file tc-barrier-ident.c

### Functions

- static rtems_status_code **ClassicBarrierIdentAction** (rtems_name name, rtems_id *id)
- void **T_case_body_RtemsBarrierValIdent** (void)

### 8.202.1 Detailed Description

Test the rtems_barrier_ident directive.

This test case performs the following actions:

- Run the generic object identification tests for Classic API partition class objects defined by /rtems/req/ident-local.

# 8.203 spec:/rtems/event/req/send-receive

## Files

- file tr-event-send-receive.c
- file tr-event-send-receive.h

## Classes

- struct RtemsEventReqSendReceive_Context

  *Test context for spec:/rtems/event/req/send-receive test case.*

## Macros

- #define **INPUT_EVENTS** ( RTEMS_EVENT_5 | RTEMS_EVENT_23 )
- #define **WORKER_ATTRIBUTES** RTEMS_DEFAULT_ATTRIBUTES
- #define **MAX_TLS_SIZE** RTEMS_ALIGN_UP( 64, RTEMS_TASK_STORAGE_ALIGNMENT )

## Typedefs

- typedef RtemsEventReqSendReceive_Context **Context**

## Enumerations

- enum **Priorities** { **PRIO_HIGH** = 1, **PRIO_NORMAL**, **PRIO_LOW**, **PRIO_OTHER** }
- enum **SenderTypes** { **SENDER_NONE**, **SENDER_SELF**, **SENDER_SELF_2**, **SENDER_WORKER**, **SENDER_INTERRUPT** }
- enum **ReceiveTypes** { **RECEIVE_SKIP**, **RECEIVE_NORMAL**, **RECEIVE_INTERRUPT** }
- enum **ReceiveConditionStates** { **RECEIVE_COND_UNKNOWN**, **RECEIVE_COND_SATSIFIED**, **RECEIVE_COND_UNSATISFIED** }
- enum **RtemsEventReqSendReceive_Pre_Id** { **RtemsEventReqSendReceive_Pre_Id_InvId**, **RtemsEventReqSendReceive_Pre_Id_Task**, **RtemsEventReqSendReceive_Pre_Id_NA** }
- enum **RtemsEventReqSendReceive_Pre_Send** { **RtemsEventReqSendReceive_Pre_Send_Zero**, **RtemsEventReqSendReceive_Pre_Send_Unrelated**, **RtemsEventReqSendReceive_Pre_Send_Any**, **RtemsEventReqSendReceive_Pre_Send_All**, **RtemsEventReqSendReceive_Pre_Send_MixedAny**, **RtemsEventReqSendReceive_Pre_Send_MixedAll**, **RtemsEventReqSendReceive_Pre_Send_NA** }
- enum **RtemsEventReqSendReceive_Pre_ReceiverState** { **RtemsEventReqSendReceive_Pre_ReceiverState_NotWaiting**, **RtemsEventReqSendReceive_Pre_ReceiverState_Poll**, **RtemsEventReqSendReceive_Pre_ReceiverState_Timeout**, **RtemsEventReqSendReceive_Pre_ReceiverState_Lower**, **RtemsEventReqSendReceive_Pre_ReceiverState_Equal**, **RtemsEventReqSendReceive_Pre_ReceiverState_Higher**, **RtemsEventReqSendReceive_Pre_ReceiverState_Other**, **RtemsEventReqSendReceive_Pre_ReceiverState_Intend**, **RtemsEventReqSendReceive_Pre_ReceiverState_NA** }
- enum **RtemsEventReqSendReceive_Pre_Satisfy** { **RtemsEventReqSendReceive_Pre_Satisfy_All**, **RtemsEventReqSendReceive_Pre_Satisfy_Any**, **RtemsEventReqSendReceive_Pre_Satisfy_NA** }
- enum **RtemsEventReqSendReceive_Post_SendStatus** { **RtemsEventReqSendReceive_Post_SendStatus_Ok**, **RtemsEventReqSendReceive_Post_SendStatus_InvId**, **RtemsEventReqSendReceive_Post_SendStatus_NA** }

- enum **RtemsEventReqSendReceive_Post_ReceiveStatus** { **RtemsEventReqSendReceive_Post_ReceiveStatus_None**, **RtemsEventReqSendReceive_Post_ReceiveStatus_Pending**, **RtemsEventReqSendReceive_Post_ReceiveStatus_Timeout**, **RtemsEventReqSendReceive_Post_ReceiveStatus_Satisfied**, **RtemsEventReqSendReceive_Post_ReceiveStatus_Unsatisfied**, **RtemsEventReqSendReceive_Post_ReceiveStatus_Blocked**, **RtemsEventReqSendReceive_Post_ReceiveStatus_NA** }
- enum **RtemsEventReqSendReceive_Post_SenderPreemption** { **RtemsEventReqSendReceive_Post_SenderPreemption_No**, **RtemsEventReqSendReceive_Post_SenderPreemption_Yes**, **RtemsEventReqSendReceive_Post_SenderPreemption_NA** }

## Functions

- **RTEMS_ALIGNED** (RTEMS_TASK_STORAGE_ALIGNMENT)
- static rtems_id **CreateWakeupSema** (void)
- static void **DeleteWakeupSema** (rtems_id id)
- static void **Wait** (rtems_id id)
- static void **Wakeup** (rtems_id id)
- static bool **BlockedForEvent** (Context *ctx, Thread_Wait_flags flags)
- static bool **IntendsToBlockForEvent** (Context *ctx, Thread_Wait_flags flags)
- static bool **EventReadyAgain** (Context *ctx, Thread_Wait_flags flags)
- static bool **IsSatisfiedFlags** (Context *ctx)
- static bool **IsSatisfiedState** (Context *ctx)
- static void **SendAction** (Context *ctx)
- static void **Send** (Context *ctx, bool(*is_satsified)(Context *))
- static void **Worker** (rtems_task_argument arg)
- static rtems_event_set **GetPendingEvents** (Context *ctx)
- static void **RtemsEventReqSendReceive_Cleanup** (Context *ctx)
- static void **InterruptPrepare** (void *arg)
- static void **InterruptAction** (void *arg)
- static void **InterruptContinue** (Context *ctx)
- static T_interrupt_test_state **Interrupt** (void *arg)
- static void **RtemsEventReqSendReceive_Pre_Id_Prepare** (RtemsEventReqSendReceive_Context *ctx, RtemsEventReqSendReceive_Pre_Id state)
- static void **RtemsEventReqSendReceive_Pre_Send_Prepare** (RtemsEventReqSendReceive_Context *ctx, RtemsEventReqSendReceive_Pre_Send state)
- static void **RtemsEventReqSendReceive_Pre_ReceiverState_Prepare** (RtemsEventReqSendReceive_Context *ctx, RtemsEventReqSendReceive_Pre_ReceiverState state)
- static void **RtemsEventReqSendReceive_Pre_Satisfy_Prepare** (RtemsEventReqSendReceive_Context *ctx, RtemsEventReqSendReceive_Pre_Satisfy state)
- static void **RtemsEventReqSendReceive_Post_SendStatus_Check** (RtemsEventReqSendReceive_Context *ctx, RtemsEventReqSendReceive_Post_SendStatus state)
- static void **RtemsEventReqSendReceive_Post_ReceiveStatus_Check** (RtemsEventReqSendReceive_Context *ctx, RtemsEventReqSendReceive_Post_ReceiveStatus state)
- static void **RtemsEventReqSendReceive_Post_SenderPreemption_Check** (RtemsEventReqSendReceive_Context *ctx, RtemsEventReqSendReceive_Post_SenderPreemption state)
- static void **RtemsEventReqSendReceive_Setup** (RtemsEventReqSendReceive_Context *ctx)
- static void **RtemsEventReqSendReceive_Setup_Wrap** (void *arg)
- static void **RtemsEventReqSendReceive_Teardown** (RtemsEventReqSendReceive_Context *ctx)
- static void **RtemsEventReqSendReceive_Teardown_Wrap** (void *arg)
- static size_t **RtemsEventReqSendReceive_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsEventReqSendReceive_Prepare** (RtemsEventReqSendReceive_Context *ctx)
- static void **RtemsEventReqSendReceive_Action** (RtemsEventReqSendReceive_Context *ctx)
- void RtemsEventReqSendReceive_Run (rtems_status_code(*send)(rtems_id, rtems_event_set), rtems_status_code(*receive)(rtems_option, rtems_interval, rtems_event_set *), rtems_event_set( *get_pending_events)(Thread_Control *), unsigned int wait_class, int waiting_for_event)

  *Runs the parameterized test case.*

## Variables

- static RtemsEventReqSendReceive_Context **RtemsEventReqSendReceive_Instance**
- static const char *const **RtemsEventReqSendReceive_PreDesc_Id** [ ]
- static const char *const **RtemsEventReqSendReceive_PreDesc_Send** [ ]
- static const char *const **RtemsEventReqSendReceive_PreDesc_ReceiverState** [ ]
- static const char *const **RtemsEventReqSendReceive_PreDesc_Satisfy** [ ]
- static const char *const *const **RtemsEventReqSendReceive_PreDesc** [ ]
- static const T_interrupt_test_config **InterruptConfig**
- static T_fixture **RtemsEventReqSendReceive_Fixture**
- static const uint8_t **RtemsEventReqSendReceive_TransitionMap** [ ][3]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1
  uint16_t **Pre_Id_NA**: 1
  uint16_t **Pre_Name_NA**: 1
  uint16_t **Pre_Prio_NA**: 1
  uint16_t **Pre_Tasks_NA**: 1
  uint16_t **Pre_TLS_NA**: 1
  uint16_t **Pre_Stack_NA**: 1
  uint16_t **Pre_Ext_NA**: 1
  uint16_t **Pre_Preempt_NA**: 1
  uint8_t **Pre_Pre_NA**: 1
  uint8_t **Pre_Send_NA**: 1
  uint8_t **Pre_ReceiverState_NA**: 1
  uint8_t **Pre_Satisfy_NA**: 1
  uint8_t **Pre_Node_NA**: 1
  } **RtemsEventReqSendReceive_TransitionInfo** [ ]
- static T_fixture_node **RtemsEventReqSendReceive_Node**

### 8.203.1 Detailed Description

### 8.203.2 Function Documentation

#### 8.203.2.1 RtemsEventReqSendReceive_Run()

```
void RtemsEventReqSendReceive_Run (
             rtems_status_code(*)(rtems_id, rtems_event_set) send,
             rtems_status_code(*)(rtems_event_set, rtems_option, rtems_interval, rtems_event_set
*) receive,
             rtems_event_set(*)(Thread_Control *) get_pending_events,
             unsigned int wait_class,
             int waiting_for_event )
```

Runs the parameterized test case.

**Parameters**

| | |
|---|---|
| *send* | is the event send handler. |
| *receive* | is the event receive handler. |
| *get_pending_events* | is the get pending events handler. |
| *wait_class* | is the thread wait class. |
| *waiting_for_event* | is the thread waiting for event state. |

Definition at line 2300 of file tr-event-send-receive.c.

### 8.203.3 Variable Documentation

#### 8.203.3.1 InterruptConfig

```
const T_interrupt_test_config InterruptConfig  [static]
```

**Initial value:**

```
= {
  .prepare = InterruptPrepare,
  .action = InterruptAction,
  .interrupt = Interrupt,
  .max_iteration_count = 10000
}
```

Definition at line 568 of file tr-event-send-receive.c.

#### 8.203.3.2 RtemsEventReqSendReceive_Fixture

```
T_fixture RtemsEventReqSendReceive_Fixture  [static]
```

**Initial value:**

```
= {
  .setup = RtemsEventReqSendReceive_Setup_Wrap,
  .stop = NULL,
  .teardown = RtemsEventReqSendReceive_Teardown_Wrap,
  .scope = RtemsEventReqSendReceive_Scope,
  .initial_context = &RtemsEventReqSendReceive_Instance
}
```

Definition at line 924 of file tr-event-send-receive.c.

#### 8.203.3.3 RtemsEventReqSendReceive_PreDesc

```
const char* const* const RtemsEventReqSendReceive_PreDesc[]  [static]
```

**Initial value:**

```
= {
  RtemsEventReqSendReceive_PreDesc_Id,
  RtemsEventReqSendReceive_PreDesc_Send,
  RtemsEventReqSendReceive_PreDesc_ReceiverState,
  RtemsEventReqSendReceive_PreDesc_Satisfy,
  NULL
}
```

Definition at line 286 of file tr-event-send-receive.c.

#### 8.203.3.4 RtemsEventReqSendReceive_PreDesc_Id

```
const char* const RtemsEventReqSendReceive_PreDesc_Id[]  [static]
```

**Initial value:**

```
= {
  "InvId",
  "Task",
  "NA"
}
```

Definition at line 252 of file tr-event-send-receive.c.

#### 8.203.3.5 RtemsEventReqSendReceive_PreDesc_ReceiverState

```
const char* const RtemsEventReqSendReceive_PreDesc_ReceiverState[]  [static]
```

**Initial value:**

```
= {
  "NotWaiting",
  "Poll",
  "Timeout",
  "Lower",
  "Equal",
  "Higher",
  "Other",
  "Intend",
  "NA"
}
```

Definition at line 268 of file tr-event-send-receive.c.

#### 8.203.3.6 RtemsEventReqSendReceive_PreDesc_Satisfy

```
const char* const RtemsEventReqSendReceive_PreDesc_Satisfy[]  [static]
```

**Initial value:**

```
= {
  "All",
  "Any",
  "NA"
}
```

Definition at line 280 of file tr-event-send-receive.c.

#### 8.203.3.7 RtemsEventReqSendReceive_PreDesc_Send

```
const char* const RtemsEventReqSendReceive_PreDesc_Send[]  [static]
```

**Initial value:**

```
= {
  "Zero",
  "Unrelated",
  "Any",
  "All",
  "MixedAny",
  "MixedAll",
  "NA"
}
```

Definition at line 258 of file tr-event-send-receive.c.

# 8.204 spec:/rtems/event/val/event-constant

Tests an event constant and number of the Event Manager using the Classic and system event sets of the executing task.

## Files

- file tr-event-constant.c
- file tr-event-constant.h

## Functions

- static void **RtemsEventValEventConstant_Wrap** (rtems_event_set event, int number)
- void RtemsEventValEventConstant_Run (rtems_event_set event, int number)

  *Runs the parameterized test case.*

## Variables

- static T_fixture_node **RtemsEventValEventConstant_Node**

## 8.204.1 Detailed Description

Tests an event constant and number of the Event Manager using the Classic and system event sets of the executing task.

This test case performs the following actions:

- Validate the event constant.
  - Check that the event constant is equal to the event number bit in the event set.
  - Check that the event number bit of the event constant is not set in RTEMS_PENDING_EVENTS.
- Get all pending events of the Classic event set of the executing task.
  - Check that the directive call was successful.
  - Check that there were no pending events.
- Get all pending events of the system event set of the executing task.
  - Check that the directive call was successful.
  - Check that there were no pending events.
- Receive all pending events of the Classic event set of the executing task.
  - Check that the directive call was unsatisfied.
  - Check that there were no events received.
- Receive all pending events of the system event set of the executing task.
  - Check that the directive call was unsatisfied.
  - Check that there were no events received.
- Send the event to the Classic event set of the executing task.

    - Check that the directive call was successful.

- Get all pending events of the Classic event set of the executing task.
    - Check that the directive call was successful.
    - Check that the pending event is equal to the event sent by a previous action.
- Get all pending events of the system event set of the executing task.
    - Check that the directive call was successful.
    - Check that there were no pending events.
- Receive any event of the Classic event set of the executing task.
    - Check that the directive call was successful.
    - Check that the received event is equal to the event sent by a previous action.
- Receive any event of the system event set of the executing task.
    - Check that the directive call was unsatisfied.
    - Check that the no events were received.
- Send the event to the Classic event set of the executing task.
    - Check that the directive call was successful.
- Get all pending events of the Classic event set of the executing task.
    - Check that the directive call was successful.
    - Check that there were no pending events.
- Get all pending events of the system event set of the executing task.
    - Check that the directive call was successful.
    - Check that the pending event is equal to the event sent by a previous action.
- Receive any event of the Classic event set of the executing task.
    - Check that the directive call was unsatisfied.
    - Check that the no events were received.
- Receive any event of the system event set of the executing task.
    - Check that the directive call was successful.
    - Check that the received event is equal to the event sent by a previous action.
- Get all pending events of the Classic event set of the executing task.
    - Check that the directive call was successful.
    - Check that there were no pending events.
- Get all pending events of the system event set of the executing task.
    - Check that the directive call was successful.
    - Check that there were no pending events.

### 8.204.2 Function Documentation

#### 8.204.2.1 RtemsEventValEventConstant_Run()

```
void RtemsEventValEventConstant_Run (
              rtems_event_set event,
              int number )
```

Runs the parameterized test case.

**Parameters**

| | |
|---|---|
| *event* | is the event constant. |
| *number* | is the event number. |

Definition at line 339 of file tr-event-constant.c.

# 8.205 spec:/rtems/event/val/events

Tests the Event Manager API.

## Files

- file tc-events.c

## Functions

- void **T_case_body_RtemsEventValEvents** (void)

## Variables

- static const rtems_event_set **events** [ ]

### 8.205.1 Detailed Description

Tests the Event Manager API.

This test case performs the following actions:

- Run the event constant and number test for all 32 event constants.
- Calculate the value of a bitwise or of all 32 event constants.
  - Check that the value is equal to RTEMS_ALL_EVENTS.
- Validate the Event Manager directive options.
  - Check that RTEMS_EVENT_ALL is equal to zero.
  - Check that RTEMS_EVENT_ANY is a power of two.

## 8.206 spec:/rtems/event/val/send-receive

Tests the rtems_event_send and rtems_event_receive directives.

### Files

- file tc-event-send-receive.c

### Functions

- static rtems_status_code **EventSend** (rtems_id id, rtems_event_set event_in)
- static rtems_status_code **EventReceive** (rtems_id event_in, rtems_option option_set, rtems_interval ticks, rtems_event_set *event_out)
- static rtems_event_set **GetPendingEvents** (Thread_Control *thread)
- void **T_case_body_RtemsEventValSendReceive** (void)

### 8.206.1 Detailed Description

Tests the rtems_event_send and rtems_event_receive directives.

This test case performs the following actions:

- Run the event send and receive tests for the application event set defined by /rtems/event/req/send-receive.

# 8.207 spec:/rtems/event/val/system-send-receive

Tests the rtems_event_system_send and rtems_event_system_receive directives.

## Files

- file tc-event-send-receive.c

## Functions

- static rtems_status_code **EventSystemSend** (rtems_id id, rtems_event_set event_in)
- static rtems_status_code **EventSystemReceive** (rtems_id event_in, rtems_option option_set, rtems_interval ticks, rtems_event_set *event_out)
- static rtems_event_set **GetPendingSystemEvents** (Thread_Control *thread)
- void **T_case_body_RtemsEventValSystemSendReceive** (void)

### 8.207.1 Detailed Description

Tests the rtems_event_system_send and rtems_event_system_receive directives.

This test case performs the following actions:

- Run the event send and receive tests for the system event set defined by /rtems/event/req/send-receive.

## 8.208 spec:/rtems/message/req/construct-errors

### Files

- file tc-message-construct-errors.c

### Classes

- struct RtemsMessageReqConstructErrors_Context

  *Test context for spec:/rtems/message/req/construct-errors test case.*

### Macros

- #define **MAX_MESSAGE_QUEUES** 4
- #define **MAX_PENDING_MESSAGES** 1
- #define **MAX_MESSAGE_SIZE** 1

### Enumerations

- enum **RtemsMessageReqConstructErrors_Pre_Id** { **RtemsMessageReqConstructErrors_Pre_Id_Id**, **RtemsMessageReqConstructErrors_Pre_Id_Null**, **RtemsMessageReqConstructErrors_Pre_Id_NA** }
- enum **RtemsMessageReqConstructErrors_Pre_Name** { **RtemsMessageReqConstructErrors_Pre_Name_Valid**, **RtemsMessageReqConstructErrors_Pre_Name_Invalid**, **RtemsMessageReqConstructErrors_Pre_Name_NA** }
- enum **RtemsMessageReqConstructErrors_Pre_MaxPending** { **RtemsMessageReqConstructErrors_Pre_MaxPending_Valid**, **RtemsMessageReqConstructErrors_Pre_MaxPending_Zero**, **RtemsMessageReqConstructErrors_Pre_MaxPending_Big**, **RtemsMessageReqConstructErrors_Pre_MaxPending_NA** }
- enum **RtemsMessageReqConstructErrors_Pre_MaxSize** { **RtemsMessageReqConstructErrors_Pre_MaxSize_Valid**, **RtemsMessageReqConstructErrors_Pre_MaxSize_Zero**, **RtemsMessageReqConstructErrors_Pre_MaxSize_Big**, **RtemsMessageReqConstructErrors_Pre_MaxSize_NA** }
- enum **RtemsMessageReqConstructErrors_Pre_Queues** { **RtemsMessageReqConstructErrors_Pre_Queues_Avail**, **RtemsMessageReqConstructErrors_Pre_Queues_None**, **RtemsMessageReqConstructErrors_Pre_Queues_NA** }
- enum **RtemsMessageReqConstructErrors_Pre_Area** { **RtemsMessageReqConstructErrors_Pre_Area_Valid**, **RtemsMessageReqConstructErrors_Pre_Area_Null**, **RtemsMessageReqConstructErrors_Pre_Area_NA** }
- enum **RtemsMessageReqConstructErrors_Pre_AreaSize** { **RtemsMessageReqConstructErrors_Pre_AreaSize_Valid**, **RtemsMessageReqConstructErrors_Pre_AreaSize_Invalid**, **RtemsMessageReqConstructErrors_Pre_AreaSize_NA** }
- enum **RtemsMessageReqConstructErrors_Post_Status** { **RtemsMessageReqConstructErrors_Post_Status_Ok**, **RtemsMessageReqConstructErrors_Post_Status_InvAddress**, **RtemsMessageReqConstructErrors_Post_Status_InvName**, **RtemsMessageReqConstructErrors_Post_Status_InvNumber**, **RtemsMessageReqConstructErrors_Post_Status_InvSize**, **RtemsMessageReqConstructErrors_Post_Status_TooMany**, **RtemsMessageReqConstructErrors_Post_Status_Unsatisfied**, **RtemsMessageReqConstructErrors_Post_Status_NA** }

## Functions

- static **RTEMS_MESSAGE_QUEUE_BUFFER** (static RTEMS_MESSAGE_QUEUE_BUFFER MAX_MES↩ SAGE_SIZE)
- static void **RtemsMessageReqConstructErrors_Pre_Name_Prepare** (RtemsMessageReqConstructErrors_Context ∗ctx, RtemsMessageReqConstructErrors_Pre_Name state)
- static void **RtemsMessageReqConstructErrors_Pre_MaxPending_Prepare** (RtemsMessageReqConstructErrors_Context ∗ctx, RtemsMessageReqConstructErrors_Pre_MaxPending state)
- static void **RtemsMessageReqConstructErrors_Pre_MaxSize_Prepare** (RtemsMessageReqConstructErrors_Context ∗ctx, RtemsMessageReqConstructErrors_Pre_MaxSize state)
- static void **RtemsMessageReqConstructErrors_Pre_Queues_Prepare** (RtemsMessageReqConstructErrors_Context ∗ctx, RtemsMessageReqConstructErrors_Pre_Queues state)
- static void **RtemsMessageReqConstructErrors_Pre_Area_Prepare** (RtemsMessageReqConstructErrors_Context ∗ctx, RtemsMessageReqConstructErrors_Pre_Area state)
- static void **RtemsMessageReqConstructErrors_Pre_AreaSize_Prepare** (RtemsMessageReqConstructErrors_Context ∗ctx, RtemsMessageReqConstructErrors_Pre_AreaSize state)
- static void **RtemsMessageReqConstructErrors_Post_Status_Check** (RtemsMessageReqConstructErrors_Context ∗ctx, RtemsMessageReqConstructErrors_Post_Status state)
- static void **RtemsMessageReqConstructErrors_Setup** (RtemsMessageReqConstructErrors_Context ∗ctx)
- static void **RtemsMessageReqConstructErrors_Setup_Wrap** (void ∗arg)
- static size_t **RtemsMessageReqConstructErrors_Scope** (void ∗arg, char ∗buf, size_t n)
- static void **RtemsMessageReqConstructErrors_Prepare** (RtemsMessageReqConstructErrors_Context ∗ctx)
- static void **RtemsMessageReqConstructErrors_Action** (RtemsMessageReqConstructErrors_Context ∗ctx)
- static void **RtemsMessageReqConstructErrors_Cleanup** (RtemsMessageReqConstructErrors_Context ∗ctx)
- **T_TEST_CASE_FIXTURE** (RtemsMessageReqConstructErrors, &RtemsMessageReqConstructErrors_↩ Fixture)

## Variables

- static RtemsMessageReqConstructErrors_Context **RtemsMessageReqConstructErrors_Instance**
- static const char ∗const **RtemsMessageReqConstructErrors_PreDesc_Id** [ ]
- static const char ∗const **RtemsMessageReqConstructErrors_PreDesc_Name** [ ]
- static const char ∗const **RtemsMessageReqConstructErrors_PreDesc_MaxPending** [ ]
- static const char ∗const **RtemsMessageReqConstructErrors_PreDesc_MaxSize** [ ]
- static const char ∗const **RtemsMessageReqConstructErrors_PreDesc_Queues** [ ]
- static const char ∗const **RtemsMessageReqConstructErrors_PreDesc_Area** [ ]
- static const char ∗const **RtemsMessageReqConstructErrors_PreDesc_AreaSize** [ ]
- static const char ∗const ∗const **RtemsMessageReqConstructErrors_PreDesc** [ ]
- static T_fixture **RtemsMessageReqConstructErrors_Fixture**
- static const uint8_t **RtemsMessageReqConstructErrors_TransitionMap** [ ][1]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1

uint8_t **Pre_Size_NA**: 1
uint8_t **Pre_Parts_NA**: 1
uint8_t **Pre_InUse_NA**: 1
uint8_t **Pre_Buf_NA**: 1
uint8_t **Pre_Avail_NA**: 1
uint16_t **Skip**: 1
uint16_t **Pre_Id_NA**: 1
uint16_t **Pre_Name_NA**: 1
uint16_t **Pre_Prio_NA**: 1
uint16_t **Pre_Tasks_NA**: 1
uint16_t **Pre_TLS_NA**: 1
uint16_t **Pre_Stack_NA**: 1
uint16_t **Pre_Ext_NA**: 1
uint16_t **Pre_Preempt_NA**: 1
uint8_t **Pre_Pre_NA**: 1
uint8_t **Pre_Send_NA**: 1
uint8_t **Pre_ReceiverState_NA**: 1
uint8_t **Pre_Satisfy_NA**: 1
uint8_t **Pre_Node_NA**: 1
} **RtemsMessageReqConstructErrors_TransitionInfo** [ ]

## 8.208.1 Detailed Description

## 8.208.2 Variable Documentation

### 8.208.2.1 RtemsMessageReqConstructErrors_Fixture

```
T_fixture RtemsMessageReqConstructErrors_Fixture  [static]
```

**Initial value:**

```
= {
  .setup = RtemsMessageReqConstructErrors_Setup_Wrap,
  .stop = NULL,
  .teardown = NULL,
  .scope = RtemsMessageReqConstructErrors_Scope,
  .initial_context = &RtemsMessageReqConstructErrors_Instance
}
```

Definition at line 509 of file tc-message-construct-errors.c.

### 8.208.2.2 RtemsMessageReqConstructErrors_PreDesc

```
const char* const* const RtemsMessageReqConstructErrors_PreDesc[]  [static]
```

**Initial value:**

```
= {
  RtemsMessageReqConstructErrors_PreDesc_Id,
  RtemsMessageReqConstructErrors_PreDesc_Name,
  RtemsMessageReqConstructErrors_PreDesc_MaxPending,
  RtemsMessageReqConstructErrors_PreDesc_MaxSize,
  RtemsMessageReqConstructErrors_PreDesc_Queues,
  RtemsMessageReqConstructErrors_PreDesc_Area,
  RtemsMessageReqConstructErrors_PreDesc_AreaSize,
  NULL
}
```

Definition at line 196 of file tc-message-construct-errors.c.

#### 8.208.2.3 RtemsMessageReqConstructErrors_PreDesc_Area

```
const char* const RtemsMessageReqConstructErrors_PreDesc_Area[]  [static]
```

**Initial value:**

```
= {
  "Valid",
  "Null",
  "NA"
}
```

Definition at line 184 of file tc-message-construct-errors.c.

#### 8.208.2.4 RtemsMessageReqConstructErrors_PreDesc_AreaSize

```
const char* const RtemsMessageReqConstructErrors_PreDesc_AreaSize[]  [static]
```

**Initial value:**

```
= {
  "Valid",
  "Invalid",
  "NA"
}
```

Definition at line 190 of file tc-message-construct-errors.c.

#### 8.208.2.5 RtemsMessageReqConstructErrors_PreDesc_Id

```
const char* const RtemsMessageReqConstructErrors_PreDesc_Id[]  [static]
```

**Initial value:**

```
= {
  "Id",
  "Null",
  "NA"
}
```

Definition at line 152 of file tc-message-construct-errors.c.

#### 8.208.2.6 RtemsMessageReqConstructErrors_PreDesc_MaxPending

```
const char* const RtemsMessageReqConstructErrors_PreDesc_MaxPending[]  [static]
```

**Initial value:**

```
= {
  "Valid",
  "Zero",
  "Big",
  "NA"
}
```

Definition at line 164 of file tc-message-construct-errors.c.

#### 8.208.2.7 RtemsMessageReqConstructErrors_PreDesc_MaxSize

```
const char* const RtemsMessageReqConstructErrors_PreDesc_MaxSize[]  [static]
```

**Initial value:**

```
= {
  "Valid",
  "Zero",
  "Big",
  "NA"
}
```

Definition at line 171 of file tc-message-construct-errors.c.

#### 8.208.2.8 RtemsMessageReqConstructErrors_PreDesc_Name

```
const char* const RtemsMessageReqConstructErrors_PreDesc_Name[]  [static]
```

**Initial value:**

```
= {
  "Valid",
  "Invalid",
  "NA"
}
```

Definition at line 158 of file tc-message-construct-errors.c.

#### 8.208.2.9 RtemsMessageReqConstructErrors_PreDesc_Queues

```
const char* const RtemsMessageReqConstructErrors_PreDesc_Queues[]  [static]
```

**Initial value:**

```
= {
  "Avail",
  "None",
  "NA"
}
```

Definition at line 178 of file tc-message-construct-errors.c.

# 8.209 spec:/rtems/message/val/ident

Test the rtems_message_queue_ident directive.

## Files

- file tc-message-ident.c

## Functions

- static **RTEMS_MESSAGE_QUEUE_BUFFER** (1)
- static rtems_status_code **ClassicMessageIdentAction** (rtems_name name, uint32_t node, rtems_id *id)
- void **T_case_body_RtemsMessageValIdent** (void)

### 8.209.1 Detailed Description

Test the rtems_message_queue_ident directive.

This test case performs the following actions:

- Run the generic object identification tests for Classic API message queue class objects defined by /rtems/req/ident.

## 8.210 spec:/rtems/mode/val/modes

Tests the task mode constants and function-like macros of the Classic API.

### Files

- file tc-modes.c

### Functions

- static bool **IsPowerOfTwo** (rtems_mode mode)
- static int **PopCount** (rtems_mode modes)
- void **T_case_body_RtemsModeValModes** (void)

### 8.210.1 Detailed Description

Tests the task mode constants and function-like macros of the Classic API.

This test case performs the following actions:

- Validate the non-default task mode constants.
  - Check that RTEMS_NO_ASR is a power of two representable as an integer of type rtems_mode.
  - Check that RTEMS_NO_PREEMPT is a power of two representable as an integer of type rtems_mode.
  - Check that RTEMS_TIMESLICE is a power of two representable as an integer of type rtems_mode.
- Validate the default task mode constants.
  - Check that RTEMS_ASR is equal to zero.
  - Check that RTEMS_DEFAULT_MODES is equal to zero.
  - Check that RTEMS_NO_TIMESLICE is equal to zero.
  - Check that RTEMS_PREEMPT is equal to zero.
- Validate RTEMS_ALL_MODE_MASKS.
  - Check that the bitwise and of RTEMS_ASR_MASK and RTEMS_ALL_MODE_MASKS is equal to R↩ TEMS_ASR_MASK.
  - Check that the bitwise and of RTEMS_PREEMPT_MASK and RTEMS_ALL_MODE_MASKS is equal to RTEMS_PREEMPT_MASK.
  - Check that the bitwise and of RTEMS_TIMESLICE_MASK and RTEMS_ALL_MODE_MASKS is equal to RTEMS_TIMESLICE_MASK.
  - Check that the bitwise and of RTEMS_INTERRUPT_MASK and RTEMS_ALL_MODE_MASKS is equal to RTEMS_INTERRUPT_MASK.
- Validate the task mode mask constants except RTEMS_INTERRUPT_MASK.
  - Check that RTEMS_ASR_MASK is a power of two representable as an integer of type rtems_mode.
  - Check that RTEMS_PREEMPT_MASK is a power of two representable as an integer of type rtems_↩ mode.
  - Check that RTEMS_TIMESLICE_MASK is a power of two representable as an integer of type rtems↩ _mode.
- Calculate the bitwise or of all task mode mask constants and 0xff.
  - Check that the count of set bits in the calculated value is equal to the count of task mode mask constants except RTEMS_INTERRUPT_MASK plus eight. Since each task mode mask constants except RTEM↩ S_INTERRUPT_MASK is a power of two and the bitwise and of 0xff and RTEMS_INTERRUPT_MASK is equal to RTEMS_INTERRUPT_MASK this proves that each constant and 0xff has a unique value.

# 8.211 spec:/rtems/option/val/options

Tests the option constants of the Classic API.

## Files

- file tc-options.c

## Functions

- static bool **IsPowerOfTwo** (rtems_option option)
- static int **PopCount** (rtems_option options)
- void **T_case_body_RtemsOptionValOptions** (void)

### 8.211.1 Detailed Description

Tests the option constants of the Classic API.

This test case performs the following actions:

- Validate the non-default option constants.
  - Check that RTEMS_EVENT_ANY is a power of two.
  - Check that RTEMS_NO_WAIT is a power of two.
- Validate the default option constants.
  - Check that RTEMS_DEFAULT_OPTIONS is equal to zero.
  - Check that RTEMS_EVENT_ALL is equal to zero.
  - Check that RTEMS_WAIT is equal to zero.
- Calculate the bitwise or of all non-default option constants.
  - Check that the count of set bits in the calculated value is equal to the count of non-default option constants. Since each non-default option constant is a power of two, this proves that each constant has a unique value.

## 8.212 spec:/rtems/part/req/create

### Files

- file tc-part-create.c

### Classes

- struct RtemsPartReqCreate_Context

  *Test context for spec:/rtems/part/req/create test case.*

### Macros

- #define **PART_NAME** rtems_build_name( 'N', 'A', 'M', 'E' )
- #define **MAX_PARTITIONS** 4
- #define **BUFFER_COUNT** 2
- #define **BUFFER_SIZE** ( 2 * sizeof( void * ) )

### Enumerations

- enum **RtemsPartReqCreate_Pre_Id** { **RtemsPartReqCreate_Pre_Id_Id**, **RtemsPartReqCreate_Pre_Id_Null**, **RtemsPartReqCreate_Pre_Id_NA** }
- enum **RtemsPartReqCreate_Pre_Name** { **RtemsPartReqCreate_Pre_Name_Valid**, **RtemsPartReqCreate_Pre_Name_Invalid**, **RtemsPartReqCreate_Pre_Name_NA** }
- enum **RtemsPartReqCreate_Pre_Start** { **RtemsPartReqCreate_Pre_Start_Valid**, **RtemsPartReqCreate_Pre_Start_Null**, **RtemsPartReqCreate_Pre_Start_BadAlign**, **RtemsPartReqCreate_Pre_Start_NA** }
- enum **RtemsPartReqCreate_Pre_Length** { **RtemsPartReqCreate_Pre_Length_Valid**, **RtemsPartReqCreate_Pre_Length_Zero**, **RtemsPartReqCreate_Pre_Length_Invalid**, **RtemsPartReqCreate_Pre_Length_NA** }
- enum **RtemsPartReqCreate_Pre_Size** { **RtemsPartReqCreate_Pre_Size_Valid**, **RtemsPartReqCreate_Pre_Size_Zero**, **RtemsPartReqCreate_Pre_Size_Small**, **RtemsPartReqCreate_Pre_Size_NA** }
- enum **RtemsPartReqCreate_Pre_Parts** { **RtemsPartReqCreate_Pre_Parts_Avail**, **RtemsPartReqCreate_Pre_Parts_None**, **RtemsPartReqCreate_Pre_Parts_NA** }
- enum **RtemsPartReqCreate_Post_Status** { **RtemsPartReqCreate_Post_Status_Ok**, **RtemsPartReqCreate_Post_Status_InvAddress**, **RtemsPartReqCreate_Post_Status_InvName**, **RtemsPartReqCreate_Post_Status_InvNumber**, **RtemsPartReqCreate_Post_Status_InvSize**, **RtemsPartReqCreate_Post_Status_TooMany**, **RtemsPartReqCreate_Post_Status_NA** }

### Functions

- static **RTEMS_ALIGNED** (static RTEMS_ALIGNED RTEMS_PARTITION_ALIGNMENT)
- static void **RtemsPartReqCreate_Pre_Name_Prepare** (RtemsPartReqCreate_Context *ctx, RtemsPartReqCreate_Pre_Name state)
- static void **RtemsPartReqCreate_Pre_Start_Prepare** (RtemsPartReqCreate_Context *ctx, RtemsPartReqCreate_Pre_Start state)
- static void **RtemsPartReqCreate_Pre_Length_Prepare** (RtemsPartReqCreate_Context *ctx, RtemsPartReqCreate_Pre_Length state)
- static void **RtemsPartReqCreate_Pre_Size_Prepare** (RtemsPartReqCreate_Context *ctx, RtemsPartReqCreate_Pre_Size state)

- static void **RtemsPartReqCreate_Pre_Parts_Prepare** (RtemsPartReqCreate_Context *ctx, RtemsPart↩ ReqCreate_Pre_Parts state)
- static void **RtemsPartReqCreate_Post_Status_Check** (RtemsPartReqCreate_Context *ctx, RtemsPart↩ ReqCreate_Post_Status state)
- static void **RtemsPartReqCreate_Setup** (RtemsPartReqCreate_Context *ctx)
- static void **RtemsPartReqCreate_Setup_Wrap** (void *arg)
- static size_t **RtemsPartReqCreate_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsPartReqCreate_Prepare** (RtemsPartReqCreate_Context *ctx)
- static void **RtemsPartReqCreate_Action** (RtemsPartReqCreate_Context *ctx)
- static void **RtemsPartReqCreate_Cleanup** (RtemsPartReqCreate_Context *ctx)
- **T_TEST_CASE_FIXTURE** (RtemsPartReqCreate, &RtemsPartReqCreate_Fixture)

## Variables

- static RtemsPartReqCreate_Context **RtemsPartReqCreate_Instance**
- static const char *const **RtemsPartReqCreate_PreDesc_Id** [ ]
- static const char *const **RtemsPartReqCreate_PreDesc_Name** [ ]
- static const char *const **RtemsPartReqCreate_PreDesc_Start** [ ]
- static const char *const **RtemsPartReqCreate_PreDesc_Length** [ ]
- static const char *const **RtemsPartReqCreate_PreDesc_Size** [ ]
- static const char *const **RtemsPartReqCreate_PreDesc_Parts** [ ]
- static const char *const *const **RtemsPartReqCreate_PreDesc** [ ]
- static T_fixture **RtemsPartReqCreate_Fixture**
- static const uint8_t **RtemsPartReqCreate_TransitionMap** [ ][1]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1
  uint16_t **Pre_Id_NA**: 1
  uint16_t **Pre_Name_NA**: 1
  uint16_t **Pre_Prio_NA**: 1
  uint16_t **Pre_Tasks_NA**: 1
  uint16_t **Pre_TLS_NA**: 1
  uint16_t **Pre_Stack_NA**: 1
  uint16_t **Pre_Ext_NA**: 1
  uint16_t **Pre_Preempt_NA**: 1
  uint8_t **Pre_Pre_NA**: 1
  uint8_t **Pre_Send_NA**: 1
  uint8_t **Pre_ReceiverState_NA**: 1
  uint8_t **Pre_Satisfy_NA**: 1
  uint8_t **Pre_Node_NA**: 1
  } **RtemsPartReqCreate_TransitionInfo** [ ]

### 8.212.1 Detailed Description

### 8.212.2 Variable Documentation

#### 8.212.2.1 RtemsPartReqCreate_Fixture

```
T_fixture RtemsPartReqCreate_Fixture  [static]
```

**Initial value:**

```
= {
  .setup = RtemsPartReqCreate_Setup_Wrap,
  .stop = NULL,
  .teardown = NULL,
  .scope = RtemsPartReqCreate_Scope,
  .initial_context = &RtemsPartReqCreate_Instance
}
```

Definition at line 494 of file tc-part-create.c.

#### 8.212.2.2 RtemsPartReqCreate_PreDesc

```
const char* const* const RtemsPartReqCreate_PreDesc[]  [static]
```

**Initial value:**

```
= {
  RtemsPartReqCreate_PreDesc_Id,
  RtemsPartReqCreate_PreDesc_Name,
  RtemsPartReqCreate_PreDesc_Start,
  RtemsPartReqCreate_PreDesc_Length,
  RtemsPartReqCreate_PreDesc_Size,
  RtemsPartReqCreate_PreDesc_Parts,
  NULL
}
```

Definition at line 192 of file tc-part-create.c.

#### 8.212.2.3 RtemsPartReqCreate_PreDesc_Id

```
const char* const RtemsPartReqCreate_PreDesc_Id[]  [static]
```

**Initial value:**

```
= {
  "Id",
  "Null",
  "NA"
}
```

Definition at line 153 of file tc-part-create.c.

#### 8.212.2.4 RtemsPartReqCreate_PreDesc_Length

```
const char* const RtemsPartReqCreate_PreDesc_Length[]  [static]
```

**Initial value:**

```
= {
  "Valid",
  "Zero",
  "Invalid",
  "NA"
}
```

Definition at line 172 of file tc-part-create.c.

#### 8.212.2.5 RtemsPartReqCreate_PreDesc_Name

```
const char* const RtemsPartReqCreate_PreDesc_Name[]  [static]
```

**Initial value:**

```
= {
  "Valid",
  "Invalid",
  "NA"
}
```

Definition at line 159 of file tc-part-create.c.

#### 8.212.2.6 RtemsPartReqCreate_PreDesc_Parts

```
const char* const RtemsPartReqCreate_PreDesc_Parts[]  [static]
```

**Initial value:**

```
= {
  "Avail",
  "None",
  "NA"
}
```

Definition at line 186 of file tc-part-create.c.

#### 8.212.2.7 RtemsPartReqCreate_PreDesc_Size

```
const char* const RtemsPartReqCreate_PreDesc_Size[]  [static]
```

**Initial value:**

```
= {
  "Valid",
  "Zero",
  "Small",
  "NA"
}
```

Definition at line 179 of file tc-part-create.c.

#### 8.212.2.8 RtemsPartReqCreate_PreDesc_Start

```
const char* const RtemsPartReqCreate_PreDesc_Start[]  [static]
```

**Initial value:**

```
= {
  "Valid",
  "Null",
  "BadAlign",
  "NA"
}
```

Definition at line 165 of file tc-part-create.c.

## 8.213 spec:/rtems/part/req/delete

### Files

- file tc-part-delete.c

### Classes

- struct RtemsPartReqDelete_Context

  *Test context for spec:/rtems/part/req/delete test case.*

### Macros

- #define **PART_NAME** rtems_build_name( 'N', 'A', 'M', 'E' )
- #define **BUFFER_COUNT** 1
- #define **BUFFER_SIZE** ( 2 * sizeof( void * ) )

### Enumerations

- enum **RtemsPartReqDelete_Pre_Id** { **RtemsPartReqDelete_Pre_Id_Id**, **RtemsPartReqDelete_Pre_Id_Invalid**, **RtemsPartReqDelete_Pre_Id_NA** }
- enum **RtemsPartReqDelete_Pre_InUse** { **RtemsPartReqDelete_Pre_InUse_Yes**, **RtemsPartReqDelete_Pre_InUse_No**, **RtemsPartReqDelete_Pre_InUse_NA** }
- enum **RtemsPartReqDelete_Post_Status** { **RtemsPartReqDelete_Post_Status_Ok**, **RtemsPartReqDelete_Post_Status_InvId**, **RtemsPartReqDelete_Post_Status_InUse**, **RtemsPartReqDelete_Post_Status_NA** }

### Functions

- static **RTEMS_ALIGNED** (RTEMS_PARTITION_ALIGNMENT)
- static void **RtemsPartReqDelete_Pre_InUse_Prepare** (RtemsPartReqDelete_Context *ctx, RtemsPartReqDelete_Pre_InUse state)
- static void **RtemsPartReqDelete_Post_Status_Check** (RtemsPartReqDelete_Context *ctx, RtemsPartReqDelete_Post_Status state)
- static size_t **RtemsPartReqDelete_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsPartReqDelete_Prepare** (RtemsPartReqDelete_Context *ctx)
- static void **RtemsPartReqDelete_Action** (RtemsPartReqDelete_Context *ctx)
- static void **RtemsPartReqDelete_Cleanup** (RtemsPartReqDelete_Context *ctx)
- **T_TEST_CASE_FIXTURE** (RtemsPartReqDelete, &RtemsPartReqDelete_Fixture)

## Variables

- static RtemsPartReqDelete_Context **RtemsPartReqDelete_Instance**
- static const char *const **RtemsPartReqDelete_PreDesc_Id** [ ]
- static const char *const **RtemsPartReqDelete_PreDesc_InUse** [ ]
- static const char *const *const **RtemsPartReqDelete_PreDesc** [ ]
- static T_fixture **RtemsPartReqDelete_Fixture**
- static const uint8_t **RtemsPartReqDelete_TransitionMap** [ ][1]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1
  uint16_t **Pre_Id_NA**: 1
  uint16_t **Pre_Name_NA**: 1
  uint16_t **Pre_Prio_NA**: 1
  uint16_t **Pre_Tasks_NA**: 1
  uint16_t **Pre_TLS_NA**: 1
  uint16_t **Pre_Stack_NA**: 1
  uint16_t **Pre_Ext_NA**: 1
  uint16_t **Pre_Preempt_NA**: 1
  uint8_t **Pre_Pre_NA**: 1
  uint8_t **Pre_Send_NA**: 1
  uint8_t **Pre_ReceiverState_NA**: 1
  uint8_t **Pre_Satisfy_NA**: 1
  uint8_t **Pre_Node_NA**: 1
  } **RtemsPartReqDelete_TransitionInfo** [ ]

## 8.213.1 Detailed Description

## 8.213.2 Variable Documentation

### 8.213.2.1 RtemsPartReqDelete_Fixture

```
T_fixture RtemsPartReqDelete_Fixture  [static]
```

**Initial value:**

```
= {
  .setup = NULL,
  .stop = NULL,
```

```
  .teardown = NULL,
  .scope = RtemsPartReqDelete_Scope,
  .initial_context = &RtemsPartReqDelete_Instance
}
```

Definition at line 242 of file tc-part-delete.c.

#### 8.213.2.2 RtemsPartReqDelete_PreDesc

```
const char* const* const RtemsPartReqDelete_PreDesc[]  [static]
```

**Initial value:**

```
= {
  RtemsPartReqDelete_PreDesc_Id,
  RtemsPartReqDelete_PreDesc_InUse,
  NULL
}
```

Definition at line 122 of file tc-part-delete.c.

#### 8.213.2.3 RtemsPartReqDelete_PreDesc_Id

```
const char* const RtemsPartReqDelete_PreDesc_Id[]  [static]
```

**Initial value:**

```
= {
  "Id",
  "Invalid",
  "NA"
}
```

Definition at line 110 of file tc-part-delete.c.

#### 8.213.2.4 RtemsPartReqDelete_PreDesc_InUse

```
const char* const RtemsPartReqDelete_PreDesc_InUse[]  [static]
```

**Initial value:**

```
= {
  "Yes",
  "No",
  "NA"
}
```

Definition at line 116 of file tc-part-delete.c.

#### 8.213.2.5 RtemsPartReqDelete_TransitionInfo

```
const { ...  } RtemsPartReqDelete_TransitionInfo[]  [static]
```

**Initial value:**

```
= {
  {
    0, 0, 0
  }, {
    0, 0, 0
  }, {
    0, 0, 0
  }, {
    0, 0, 0
  }
}
```

#### 8.213.2.6 RtemsPartReqDelete_TransitionMap

```
const uint8_t RtemsPartReqDelete_TransitionMap[][1]  [static]
```

**Initial value:**

```
= {
  {
    RtemsPartReqDelete_Post_Status_InUse
  }, {
    RtemsPartReqDelete_Post_Status_Ok
  }, {
    RtemsPartReqDelete_Post_Status_InvId
  }, {
    RtemsPartReqDelete_Post_Status_InvId
  }
}
```

Definition at line 250 of file tc-part-delete.c.

## 8.214 spec:/rtems/part/req/get-buffer

### Files

- file tc-part-get.c

### Classes

- struct RtemsPartReqGetBuffer_Context

  *Test context for spec:/rtems/part/req/get-buffer test case.*

### Macros

- #define **BUFFER_COUNT** 1
- #define **BUFFER_SIZE** ( 2 * sizeof( void * ) )

### Enumerations

- enum **RtemsPartReqGetBuffer_Pre_Id** { **RtemsPartReqGetBuffer_Pre_Id_Id**, **RtemsPartReqGetBuffer_Pre_Id_Invalid**, **RtemsPartReqGetBuffer_Pre_Id_NA** }
- enum **RtemsPartReqGetBuffer_Pre_Buf** { **RtemsPartReqGetBuffer_Pre_Buf_Valid**, **RtemsPartReqGetBuffer_Pre_Buf_Null**, **RtemsPartReqGetBuffer_Pre_Buf_NA** }
- enum **RtemsPartReqGetBuffer_Pre_Avail** { **RtemsPartReqGetBuffer_Pre_Avail_Yes**, **RtemsPartReqGetBuffer_Pre_Avail_No**, **RtemsPartReqGetBuffer_Pre_Avail_NA** }
- enum **RtemsPartReqGetBuffer_Post_Status** { **RtemsPartReqGetBuffer_Post_Status_Ok**, **RtemsPartReqGetBuffer_Post_Status_InvId**, **RtemsPartReqGetBuffer_Post_Status_InvAddr**, **RtemsPartReqGetBuffer_Post_Status_Unsatisfied**, **RtemsPartReqGetBuffer_Post_Status_NA** }

### Functions

- static **RTEMS_ALIGNED** (RTEMS_PARTITION_ALIGNMENT)
- static void **RtemsPartReqGetBuffer_Pre_Buf_Prepare** (RtemsPartReqGetBuffer_Context *ctx, RtemsPartReqGetBuffer_Pre_Buf state)
- static void **RtemsPartReqGetBuffer_Pre_Avail_Prepare** (RtemsPartReqGetBuffer_Context *ctx, RtemsPartReqGetBuffer_Pre_Avail state)
- static void **RtemsPartReqGetBuffer_Post_Status_Check** (RtemsPartReqGetBuffer_Context *ctx, RtemsPartReqGetBuffer_Post_Status state)
- static void **RtemsPartReqGetBuffer_Setup** (RtemsPartReqGetBuffer_Context *ctx)
- static void **RtemsPartReqGetBuffer_Setup_Wrap** (void *arg)
- static void **RtemsPartReqGetBuffer_Teardown** (RtemsPartReqGetBuffer_Context *ctx)
- static void **RtemsPartReqGetBuffer_Teardown_Wrap** (void *arg)
- static size_t **RtemsPartReqGetBuffer_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsPartReqGetBuffer_Prepare** (RtemsPartReqGetBuffer_Context *ctx)
- static void **RtemsPartReqGetBuffer_Action** (RtemsPartReqGetBuffer_Context *ctx)
- static void **RtemsPartReqGetBuffer_Cleanup** (RtemsPartReqGetBuffer_Context *ctx)
- **T_TEST_CASE_FIXTURE** (RtemsPartReqGetBuffer, &RtemsPartReqGetBuffer_Fixture)

## Variables

- static RtemsPartReqGetBuffer_Context **RtemsPartReqGetBuffer_Instance**
- static const char *const **RtemsPartReqGetBuffer_PreDesc_Id** [ ]
- static const char *const **RtemsPartReqGetBuffer_PreDesc_Buf** [ ]
- static const char *const **RtemsPartReqGetBuffer_PreDesc_Avail** [ ]
- static const char *const *const **RtemsPartReqGetBuffer_PreDesc** [ ]
- static T_fixture **RtemsPartReqGetBuffer_Fixture**
- static const uint8_t **RtemsPartReqGetBuffer_TransitionMap** [ ][1]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1
  uint16_t **Pre_Id_NA**: 1
  uint16_t **Pre_Name_NA**: 1
  uint16_t **Pre_Prio_NA**: 1
  uint16_t **Pre_Tasks_NA**: 1
  uint16_t **Pre_TLS_NA**: 1
  uint16_t **Pre_Stack_NA**: 1
  uint16_t **Pre_Ext_NA**: 1
  uint16_t **Pre_Preempt_NA**: 1
  uint8_t **Pre_Pre_NA**: 1
  uint8_t **Pre_Send_NA**: 1
  uint8_t **Pre_ReceiverState_NA**: 1
  uint8_t **Pre_Satisfy_NA**: 1
  uint8_t **Pre_Node_NA**: 1
  } **RtemsPartReqGetBuffer_TransitionInfo** [ ]

### 8.214.1 Detailed Description

### 8.214.2 Variable Documentation

#### 8.214.2.1 RtemsPartReqGetBuffer_Fixture

```
T_fixture RtemsPartReqGetBuffer_Fixture  [static]
```

**Initial value:**

```
= {
```

```
  .setup = RtemsPartReqGetBuffer_Setup_Wrap,
  .stop = NULL,
  .teardown = RtemsPartReqGetBuffer_Teardown_Wrap,
  .scope = RtemsPartReqGetBuffer_Scope,
  .initial_context = &RtemsPartReqGetBuffer_Instance
}
```

Definition at line 312 of file tc-part-get.c.

#### 8.214.2.2 RtemsPartReqGetBuffer_PreDesc

```
const char* const* const RtemsPartReqGetBuffer_PreDesc[]  [static]
```

**Initial value:**

```
= {
  RtemsPartReqGetBuffer_PreDesc_Id,
  RtemsPartReqGetBuffer_PreDesc_Buf,
  RtemsPartReqGetBuffer_PreDesc_Avail,
  NULL
}
```

Definition at line 139 of file tc-part-get.c.

#### 8.214.2.3 RtemsPartReqGetBuffer_PreDesc_Avail

```
const char* const RtemsPartReqGetBuffer_PreDesc_Avail[]  [static]
```

**Initial value:**

```
= {
  "Yes",
  "No",
  "NA"
}
```

Definition at line 133 of file tc-part-get.c.

#### 8.214.2.4 RtemsPartReqGetBuffer_PreDesc_Buf

```
const char* const RtemsPartReqGetBuffer_PreDesc_Buf[]  [static]
```

**Initial value:**

```
= {
  "Valid",
  "Null",
  "NA"
}
```

Definition at line 127 of file tc-part-get.c.

#### 8.214.2.5 RtemsPartReqGetBuffer_PreDesc_Id

```
const char* const RtemsPartReqGetBuffer_PreDesc_Id[]  [static]
```

**Initial value:**

```
= {
  "Id",
  "Invalid",
  "NA"
}
```

Definition at line 121 of file tc-part-get.c.

#### 8.214.2.6 RtemsPartReqGetBuffer_TransitionInfo

```
const { ...  } RtemsPartReqGetBuffer_TransitionInfo[]  [static]
```

**Initial value:**

```
= {
  {
    0, 0, 0, 0
  }, {
    0, 0, 0, 0
  }, {
    0, 0, 0, 0
  }, {
    0, 0, 0, 0
  }, {
    0, 0, 0, 0
  }, {
    0, 0, 0, 0
  }, {
    0, 0, 0, 0
  }, {
    0, 0, 0, 0
  }
}
```

#### 8.214.2.7 RtemsPartReqGetBuffer_TransitionMap

```
const uint8_t RtemsPartReqGetBuffer_TransitionMap[][1]  [static]
```

**Initial value:**

```
= {
  {
    RtemsPartReqGetBuffer_Post_Status_Ok
  }, {
    RtemsPartReqGetBuffer_Post_Status_Unsatisfied
  }, {
    RtemsPartReqGetBuffer_Post_Status_InvAddr
  }, {
    RtemsPartReqGetBuffer_Post_Status_InvAddr
  }, {
    RtemsPartReqGetBuffer_Post_Status_InvId
  }, {
    RtemsPartReqGetBuffer_Post_Status_InvId
  }, {
    RtemsPartReqGetBuffer_Post_Status_InvAddr
  }, {
    RtemsPartReqGetBuffer_Post_Status_InvAddr
  }
}
```

Definition at line 320 of file tc-part-get.c.

## 8.215 spec:/rtems/part/req/return-buffer

### Files

- file tc-part-return.c

### Classes

- struct RtemsPartReqReturnBuffer_Context

  *Test context for spec:/rtems/part/req/return-buffer test case.*

### Macros

- #define **BUFFER_COUNT** 1
- #define **BUFFER_SIZE** ( 2 * sizeof( void * ) )

### Enumerations

- enum **RtemsPartReqReturnBuffer_Pre_Id** { **RtemsPartReqReturnBuffer_Pre_Id_Id**, **RtemsPartReqReturnBuffer_Pre_Id_Invalid**, **RtemsPartReqReturnBuffer_Pre_Id_NA** }
- enum **RtemsPartReqReturnBuffer_Pre_Buf** { **RtemsPartReqReturnBuffer_Pre_Buf_Valid**, **RtemsPartReqReturnBuffer_Pre_Buf_Invalid**, **RtemsPartReqReturnBuffer_Pre_Buf_NA** }
- enum **RtemsPartReqReturnBuffer_Post_Status** { **RtemsPartReqReturnBuffer_Post_Status_Ok**, **RtemsPartReqReturnBuffer_Post_Status_Invld**, **RtemsPartReqReturnBuffer_Post_Status_InvAddr**, **RtemsPartReqReturnBuffer_Post_Status_NA** }

### Functions

- static **RTEMS_ALIGNED** (RTEMS_PARTITION_ALIGNMENT)
- static void **RtemsPartReqReturnBuffer_Pre_Buf_Prepare** (RtemsPartReqReturnBuffer_Context *ctx, RtemsPartReqReturnBuffer_Pre_Buf state)
- static void **RtemsPartReqReturnBuffer_Post_Status_Check** (RtemsPartReqReturnBuffer_Context *ctx, RtemsPartReqReturnBuffer_Post_Status state)
- static void **RtemsPartReqReturnBuffer_Setup** (RtemsPartReqReturnBuffer_Context *ctx)
- static void **RtemsPartReqReturnBuffer_Setup_Wrap** (void *arg)
- static void **RtemsPartReqReturnBuffer_Teardown** (RtemsPartReqReturnBuffer_Context *ctx)
- static void **RtemsPartReqReturnBuffer_Teardown_Wrap** (void *arg)
- static size_t **RtemsPartReqReturnBuffer_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsPartReqReturnBuffer_Action** (RtemsPartReqReturnBuffer_Context *ctx)
- **T_TEST_CASE_FIXTURE** (RtemsPartReqReturnBuffer, &RtemsPartReqReturnBuffer_Fixture)

## Variables

- static RtemsPartReqReturnBuffer_Context **RtemsPartReqReturnBuffer_Instance**
- static const char *const **RtemsPartReqReturnBuffer_PreDesc_Id** [ ]
- static const char *const **RtemsPartReqReturnBuffer_PreDesc_Buf** [ ]
- static const char *const *const **RtemsPartReqReturnBuffer_PreDesc** [ ]
- static T_fixture **RtemsPartReqReturnBuffer_Fixture**
- static const uint8_t **RtemsPartReqReturnBuffer_TransitionMap** [ ][1]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1
  uint16_t **Pre_Id_NA**: 1
  uint16_t **Pre_Name_NA**: 1
  uint16_t **Pre_Prio_NA**: 1
  uint16_t **Pre_Tasks_NA**: 1
  uint16_t **Pre_TLS_NA**: 1
  uint16_t **Pre_Stack_NA**: 1
  uint16_t **Pre_Ext_NA**: 1
  uint16_t **Pre_Preempt_NA**: 1
  uint8_t **Pre_Pre_NA**: 1
  uint8_t **Pre_Send_NA**: 1
  uint8_t **Pre_ReceiverState_NA**: 1
  uint8_t **Pre_Satisfy_NA**: 1
  uint8_t **Pre_Node_NA**: 1
  } **RtemsPartReqReturnBuffer_TransitionInfo** [ ]

## 8.215.1 Detailed Description

## 8.215.2 Variable Documentation

### 8.215.2.1 RtemsPartReqReturnBuffer_Fixture

```
T_fixture RtemsPartReqReturnBuffer_Fixture  [static]
```

**Initial value:**

```
= {
  .setup = RtemsPartReqReturnBuffer_Setup_Wrap,
  .stop = NULL,
```

```
  .teardown = RtemsPartReqReturnBuffer_Teardown_Wrap,
  .scope = RtemsPartReqReturnBuffer_Scope,
  .initial_context = &RtemsPartReqReturnBuffer_Instance
}
```

Definition at line 284 of file tc-part-return.c.

#### 8.215.2.2 RtemsPartReqReturnBuffer_PreDesc

```
const char* const* const RtemsPartReqReturnBuffer_PreDesc[]  [static]
```

**Initial value:**

```
= {
  RtemsPartReqReturnBuffer_PreDesc_Id,
  RtemsPartReqReturnBuffer_PreDesc_Buf,
  NULL
}
```

Definition at line 125 of file tc-part-return.c.

#### 8.215.2.3 RtemsPartReqReturnBuffer_PreDesc_Buf

```
const char* const RtemsPartReqReturnBuffer_PreDesc_Buf[]  [static]
```

**Initial value:**

```
= {
  "Valid",
  "Invalid",
  "NA"
}
```

Definition at line 119 of file tc-part-return.c.

#### 8.215.2.4 RtemsPartReqReturnBuffer_PreDesc_Id

```
const char* const RtemsPartReqReturnBuffer_PreDesc_Id[]  [static]
```

**Initial value:**

```
= {
  "Id",
  "Invalid",
  "NA"
}
```

Definition at line 113 of file tc-part-return.c.

#### 8.215.2.5 RtemsPartReqReturnBuffer_TransitionInfo

```
const { ...  } RtemsPartReqReturnBuffer_TransitionInfo[]  [static]
```

**Initial value:**

```
= {
  {
    0, 0, 0
  }, {
    0, 0, 0
  }, {
    0, 0, 0
  }, {
    0, 0, 0
  }
}
```

#### 8.215.2.6 RtemsPartReqReturnBuffer_TransitionMap

```
const uint8_t RtemsPartReqReturnBuffer_TransitionMap[][1]  [static]
```

**Initial value:**

```
= {
  {
    RtemsPartReqReturnBuffer_Post_Status_Ok
  }, {
    RtemsPartReqReturnBuffer_Post_Status_InvAddr
  }, {
    RtemsPartReqReturnBuffer_Post_Status_InvId
  }, {
    RtemsPartReqReturnBuffer_Post_Status_InvId
  }
}
```

Definition at line 292 of file tc-part-return.c.

## 8.216 spec:/rtems/part/val/ident

Test the rtems_partition_ident directive.

### Files

- file tc-part-ident.c

### Functions

- static rtems_status_code **ClassicPartIdentAction** (rtems_name name, uint32_t node, rtems_id *id)
- void **T_case_body_RtemsPartValIdent** (void)

### 8.216.1 Detailed Description

Test the rtems_partition_ident directive.

This test case performs the following actions:

- Run the generic object identification tests for Classic API partition class objects defined by /rtems/req/ident.

# 8.217 spec:/rtems/part/val/part

Validates some functional requirements of the Partition Manager.

## Files

- file tc-part.c

## Functions

- void **T_case_body_RtemsPartValPart** (void)

### 8.217.1 Detailed Description

Validates some functional requirements of the Partition Manager.

This test case performs the following actions:

- Create a partition with a buffer area length which is greater than three times the buffer size and less than four times the buffer size.
  - Check that exactly three buffers can be obtained from the partition for use in parallel.
  - Return the three buffers in use to the partition and check that they can be obtained from the partition for use in parallel in FIFO order.

## 8.218 spec:/rtems/ratemon/val/ident

Test the rtems_rate_monotonic_ident directive.

### Files

- file tc-ratemon-ident.c

### Functions

- static rtems_status_code **ClassicRatemonIdentAction** (rtems_name name, rtems_id *id)
- void **T_case_body_RtemsRatemonValIdent** (void)

### 8.218.1 Detailed Description

Test the rtems_rate_monotonic_ident directive.

This test case performs the following actions:

- Run the generic object identification tests for Classic API rate monotonic class objects defined by /rtems/req/ident-local.

# 8.219 spec:/rtems/req/ident

## Files

- file tr-object-ident.c
- file tr-object-ident.h

## Classes

- struct RtemsReqIdent_Context

  *Test context for spec:/rtems/req/ident test case.*

## Macros

- #define **ClassicObjectIdentName** rtems_build_name( 'I', 'D', 'N', 'T' )

## Enumerations

- enum **RtemsReqIdent_Pre_Name** { **RtemsReqIdent_Pre_Name_Invalid**, **RtemsReqIdent_Pre_Name_Valid**, **RtemsReqIdent_Pre_Name_NA** }
- enum **RtemsReqIdent_Pre_Node** { **RtemsReqIdent_Pre_Node_Local**, **RtemsReqIdent_Pre_Node_Remote**, **RtemsReqIdent_Pre_Node_Invalid**, **RtemsReqIdent_Pre_Node_SearchAll**, **RtemsReqIdent_Pre_Node_SearchOther**, **RtemsReqIdent_Pre_Node_SearchLocal**, **RtemsReqIdent_Pre_Node_NA** }
- enum **RtemsReqIdent_Pre_Id** { **RtemsReqIdent_Pre_Id_NullPtr**, **RtemsReqIdent_Pre_Id_Valid**, **RtemsReqIdent_Pre_Id_NA** }
- enum **RtemsReqIdent_Post_Status** { **RtemsReqIdent_Post_Status_Ok**, **RtemsReqIdent_Post_Status_InvAddr**, **RtemsReqIdent_Post_Status_InvName**, **RtemsReqIdent_Post_Status_InvNode**, **RtemsReqIdent_Post_Status_NA** }
- enum **RtemsReqIdent_Post_Id** { **RtemsReqIdent_Post_Id_Nop**, **RtemsReqIdent_Post_Id_NullPtr**, **RtemsReqIdent_Post_Id_LocalObj**, **RtemsReqIdent_Post_Id_RemoteObj**, **RtemsReqIdent_Post_Id_NA** }

## Functions

- static void **RtemsReqIdent_Pre_Name_Prepare** (RtemsReqIdent_Context *ctx, RtemsReqIdent_Pre_Name state)
- static void **RtemsReqIdent_Pre_Node_Prepare** (RtemsReqIdent_Context *ctx, RtemsReqIdent_Pre_Node state)
- static void **RtemsReqIdent_Pre_Id_Prepare** (RtemsReqIdent_Context *ctx, RtemsReqIdent_Pre_Id state)
- static void **RtemsReqIdent_Post_Status_Check** (RtemsReqIdent_Context *ctx, RtemsReqIdent_Post_Status state)
- static void **RtemsReqIdent_Post_Id_Check** (RtemsReqIdent_Context *ctx, RtemsReqIdent_Post_Id state)
- static size_t **RtemsReqIdent_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsReqIdent_Action** (RtemsReqIdent_Context *ctx)
- void RtemsReqIdent_Run (rtems_id id_local_object, rtems_status_code(*action)(rtems_name, uint32_t, rtems_id *))

  *Runs the parameterized test case.*

## Variables

- static RtemsReqIdent_Context **RtemsReqIdent_Instance**
- static const char *const **RtemsReqIdent_PreDesc_Name** [ ]
- static const char *const **RtemsReqIdent_PreDesc_Node** [ ]
- static const char *const **RtemsReqIdent_PreDesc_Id** [ ]
- static const char *const *const **RtemsReqIdent_PreDesc** [ ]
- static T_fixture **RtemsReqIdent_Fixture**
- static const uint8_t **RtemsReqIdent_TransitionMap** [ ][2]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1
  uint16_t **Pre_Id_NA**: 1
  uint16_t **Pre_Name_NA**: 1
  uint16_t **Pre_Prio_NA**: 1
  uint16_t **Pre_Tasks_NA**: 1
  uint16_t **Pre_TLS_NA**: 1
  uint16_t **Pre_Stack_NA**: 1
  uint16_t **Pre_Ext_NA**: 1
  uint16_t **Pre_Preempt_NA**: 1
  uint8_t **Pre_Pre_NA**: 1
  uint8_t **Pre_Send_NA**: 1
  uint8_t **Pre_ReceiverState_NA**: 1
  uint8_t **Pre_Satisfy_NA**: 1
  uint8_t **Pre_Node_NA**: 1
  } **RtemsReqIdent_TransitionInfo** [ ]
- static T_fixture_node **RtemsReqIdent_Node**

## 8.219.1 Detailed Description

## 8.219.2 Function Documentation

### 8.219.2.1 RtemsReqIdent_Run()

```
void RtemsReqIdent_Run (
            rtems_id id_local_object,
            rtems_status_code(*)(rtems_name, uint32_t, rtems_id *) action )
```

Runs the parameterized test case.

**Parameters**

| | |
|---|---|
| *id_local_object* | is the identifier of an active object of the class under test with the name ClassicObjectIdentName. |
| *action* | is the action handler. |

Definition at line 464 of file tr-object-ident.c.

## 8.219.3 Variable Documentation

### 8.219.3.1 RtemsReqIdent_Fixture

```
T_fixture RtemsReqIdent_Fixture  [static]
```

**Initial value:**

```
= {
  .setup = NULL,
  .stop = NULL,
  .teardown = NULL,
  .scope = RtemsReqIdent_Scope,
  .initial_context = &RtemsReqIdent_Instance
}
```

Definition at line 298 of file tr-object-ident.c.

### 8.219.3.2 RtemsReqIdent_PreDesc

```
const char* const* const RtemsReqIdent_PreDesc[]  [static]
```

**Initial value:**

```
= {
  RtemsReqIdent_PreDesc_Name,
  RtemsReqIdent_PreDesc_Node,
  RtemsReqIdent_PreDesc_Id,
  NULL
}
```

Definition at line 129 of file tr-object-ident.c.

### 8.219.3.3 RtemsReqIdent_PreDesc_Id

```
const char* const RtemsReqIdent_PreDesc_Id[]  [static]
```

**Initial value:**

```
= {
  "NullPtr",
  "Valid",
  "NA"
}
```

Definition at line 123 of file tr-object-ident.c.

#### 8.219.3.4 RtemsReqIdent_PreDesc_Name

```
const char* const RtemsReqIdent_PreDesc_Name[]  [static]
```

**Initial value:**

```
= {
  "Invalid",
  "Valid",
  "NA"
}
```

Definition at line 107 of file tr-object-ident.c.

#### 8.219.3.5 RtemsReqIdent_PreDesc_Node

```
const char* const RtemsReqIdent_PreDesc_Node[]  [static]
```

**Initial value:**

```
= {
  "Local",
  "Remote",
  "Invalid",
  "SearchAll",
  "SearchOther",
  "SearchLocal",
  "NA"
}
```

Definition at line 113 of file tr-object-ident.c.

# 8.220 spec:/rtems/req/ident-local

## Files

- file tr-object-ident-local.c
- file tr-object-ident-local.h

## Classes

- struct RtemsReqIdentLocal_Context

  *Test context for spec:/rtems/req/ident-local test case.*

## Macros

- #define **ClassicObjectLocalIdentName** rtems_build_name( 'I', 'D', 'N', 'T' )

## Enumerations

- enum **RtemsReqIdentLocal_Pre_Name** { **RtemsReqIdentLocal_Pre_Name_Invalid**, **RtemsReqIdentLocal_Pre_Name_Valid**, **RtemsReqIdentLocal_Pre_Name_NA** }
- enum **RtemsReqIdentLocal_Pre_Id** { **RtemsReqIdentLocal_Pre_Id_NullPtr**, **RtemsReqIdentLocal_Pre_Id_Valid**, **RtemsReqIdentLocal_Pre_Id_NA** }
- enum **RtemsReqIdentLocal_Post_Status** { **RtemsReqIdentLocal_Post_Status_Ok**, **RtemsReqIdentLocal_Post_Status_InvAddr**, **RtemsReqIdentLocal_Post_Status_InvName**, **RtemsReqIdentLocal_Post_Status_NA** }
- enum **RtemsReqIdentLocal_Post_Id** { **RtemsReqIdentLocal_Post_Id_Nop**, **RtemsReqIdentLocal_Post_Id_NullPtr**, **RtemsReqIdentLocal_Post_Id_Id**, **RtemsReqIdentLocal_Post_Id_NA** }

## Functions

- static void **RtemsReqIdentLocal_Pre_Name_Prepare** (RtemsReqIdentLocal_Context *ctx, RtemsReqIdentLocal_Pre_Name state)
- static void **RtemsReqIdentLocal_Pre_Id_Prepare** (RtemsReqIdentLocal_Context *ctx, RtemsReqIdentLocal_Pre_Id state)
- static void **RtemsReqIdentLocal_Post_Status_Check** (RtemsReqIdentLocal_Context *ctx, RtemsReqIdentLocal_Post_Status state)
- static void **RtemsReqIdentLocal_Post_Id_Check** (RtemsReqIdentLocal_Context *ctx, RtemsReqIdentLocal_Post_Id state)
- static size_t **RtemsReqIdentLocal_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsReqIdentLocal_Action** (RtemsReqIdentLocal_Context *ctx)
- void RtemsReqIdentLocal_Run (rtems_id id_local_object, rtems_status_code(*action)(rtems_name, rtems_id *))

  *Runs the parameterized test case.*

## Variables

- static RtemsReqIdentLocal_Context **RtemsReqIdentLocal_Instance**
- static const char *const **RtemsReqIdentLocal_PreDesc_Name** [ ]
- static const char *const **RtemsReqIdentLocal_PreDesc_Id** [ ]
- static const char *const *const **RtemsReqIdentLocal_PreDesc** [ ]
- static T_fixture **RtemsReqIdentLocal_Fixture**
- static const uint8_t **RtemsReqIdentLocal_TransitionMap** [ ][2]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1
  uint16_t **Pre_Id_NA**: 1
  uint16_t **Pre_Name_NA**: 1
  uint16_t **Pre_Prio_NA**: 1
  uint16_t **Pre_Tasks_NA**: 1
  uint16_t **Pre_TLS_NA**: 1
  uint16_t **Pre_Stack_NA**: 1
  uint16_t **Pre_Ext_NA**: 1
  uint16_t **Pre_Preempt_NA**: 1
  uint8_t **Pre_Pre_NA**: 1
  uint8_t **Pre_Send_NA**: 1
  uint8_t **Pre_ReceiverState_NA**: 1
  uint8_t **Pre_Satisfy_NA**: 1
  uint8_t **Pre_Node_NA**: 1
  } **RtemsReqIdentLocal_TransitionInfo** [ ]
- static T_fixture_node **RtemsReqIdentLocal_Node**

### 8.220.1 Detailed Description

### 8.220.2 Function Documentation

#### 8.220.2.1 RtemsReqIdentLocal_Run()

```
void RtemsReqIdentLocal_Run (
            rtems_id id_local_object,
            rtems_status_code(*)(rtems_name, rtems_id *) action )
```

Runs the parameterized test case.

**Parameters**

| | |
|---|---|
| *id_local_object* | is the identifier of an active object of the class under test with the name ClassicObjectLocalIdentName. |
| *action* | is the action handler. |

Definition at line 278 of file tr-object-ident-local.c.

## 8.220.3 Variable Documentation

### 8.220.3.1 RtemsReqIdentLocal_Fixture

```
T_fixture RtemsReqIdentLocal_Fixture  [static]
```

**Initial value:**

```
= {
  .setup = NULL,
  .stop = NULL,
  .teardown = NULL,
  .scope = RtemsReqIdentLocal_Scope,
  .initial_context = &RtemsReqIdentLocal_Instance
}
```

Definition at line 231 of file tr-object-ident-local.c.

### 8.220.3.2 RtemsReqIdentLocal_PreDesc

```
const char* const* const RtemsReqIdentLocal_PreDesc[]  [static]
```

**Initial value:**

```
= {
  RtemsReqIdentLocal_PreDesc_Name,
  RtemsReqIdentLocal_PreDesc_Id,
  NULL
}
```

Definition at line 115 of file tr-object-ident-local.c.

### 8.220.3.3 RtemsReqIdentLocal_PreDesc_Id

```
const char* const RtemsReqIdentLocal_PreDesc_Id[]  [static]
```

**Initial value:**

```
= {
  "NullPtr",
  "Valid",
  "NA"
}
```

Definition at line 109 of file tr-object-ident-local.c.

#### 8.220.3.4 RtemsReqIdentLocal_PreDesc_Name

```
const char* const RtemsReqIdentLocal_PreDesc_Name[]  [static]
```

**Initial value:**

```
= {
  "Invalid",
  "Valid",
  "NA"
}
```

Definition at line 103 of file tr-object-ident-local.c.

#### 8.220.3.5 RtemsReqIdentLocal_TransitionInfo

```
const { ...  } RtemsReqIdentLocal_TransitionInfo[]  [static]
```

**Initial value:**

```
= {
  {
    0, 0, 0
  }, {
    0, 0, 0
  }, {
    0, 0, 0
  }, {
    0, 0, 0
  }
}
```

#### 8.220.3.6 RtemsReqIdentLocal_TransitionMap

```
const uint8_t RtemsReqIdentLocal_TransitionMap[][2]  [static]
```

**Initial value:**

```
= {
  {
    RtemsReqIdentLocal_Post_Status_InvAddr,
    RtemsReqIdentLocal_Post_Id_NullPtr
  }, {
    RtemsReqIdentLocal_Post_Status_InvName,
    RtemsReqIdentLocal_Post_Id_Nop
  }, {
    RtemsReqIdentLocal_Post_Status_InvAddr,
    RtemsReqIdentLocal_Post_Id_NullPtr
  }, {
    RtemsReqIdentLocal_Post_Status_Ok,
    RtemsReqIdentLocal_Post_Id_Id
  }
}
```

Definition at line 239 of file tr-object-ident-local.c.

# 8.221 spec:/rtems/sem/val/ident

Test the rtems_semaphore_ident directive.

## Files

- file tc-sem-ident.c

## Functions

- static rtems_status_code **ClassicSemIdentAction** (rtems_name name, uint32_t node, rtems_id *id)
- void **T_case_body_RtemsSemValIdent** (void)

### 8.221.1 Detailed Description

Test the rtems_semaphore_ident directive.

This test case performs the following actions:

- Run the generic object identification tests for Classic API semaphore class objects defined by /rtems/req/ident.

## 8.222 spec:/rtems/task/req/construct-errors

### Files

- file tc-task-construct-errors.c

### Classes

- struct RtemsTaskReqConstructErrors_Context

  *Test context for spec:/rtems/task/req/construct-errors test case.*

### Macros

- #define **MAX_TLS_SIZE** RTEMS_ALIGN_UP( 128, RTEMS_TASK_STORAGE_ALIGNMENT )

### Enumerations

- enum **RtemsTaskReqConstructErrors_Pre_Id** { **RtemsTaskReqConstructErrors_Pre_Id_Id**, **RtemsTaskReqConstructErrors_Pre_Id_Null**, **RtemsTaskReqConstructErrors_Pre_Id_NA** }
- enum **RtemsTaskReqConstructErrors_Pre_Name** { **RtemsTaskReqConstructErrors_Pre_Name_Valid**, **RtemsTaskReqConstructErrors_Pre_Name_Inv**, **RtemsTaskReqConstructErrors_Pre_Name_NA** }
- enum **RtemsTaskReqConstructErrors_Pre_Prio** { **RtemsTaskReqConstructErrors_Pre_Prio_Valid**, **RtemsTaskReqConstructErrors_Pre_Prio_Zero**, **RtemsTaskReqConstructErrors_Pre_Prio_Inv**, **RtemsTaskReqConstructErrors_Pre_Prio_NA** }
- enum **RtemsTaskReqConstructErrors_Pre_Tasks** { **RtemsTaskReqConstructErrors_Pre_Tasks_Avail**, **RtemsTaskReqConstructErrors_Pre_Tasks_None**, **RtemsTaskReqConstructErrors_Pre_Tasks_NA** }
- enum **RtemsTaskReqConstructErrors_Pre_TLS** { **RtemsTaskReqConstructErrors_Pre_TLS_Enough**, **RtemsTaskReqConstructErrors_Pre_TLS_Small**, **RtemsTaskReqConstructErrors_Pre_TLS_NA** }
- enum **RtemsTaskReqConstructErrors_Pre_Stack** { **RtemsTaskReqConstructErrors_Pre_Stack_Enough**, **RtemsTaskReqConstructErrors_Pre_Stack_Small**, **RtemsTaskReqConstructErrors_Pre_Stack_NA** }
- enum **RtemsTaskReqConstructErrors_Pre_Ext** { **RtemsTaskReqConstructErrors_Pre_Ext_Ok**, **RtemsTaskReqConstructErrors_Pre_Ext_Err**, **RtemsTaskReqConstructErrors_Pre_Ext_NA** }
- enum **RtemsTaskReqConstructErrors_Pre_Preempt** { **RtemsTaskReqConstructErrors_Pre_Preempt_Yes**, **RtemsTaskReqConstructErrors_Pre_Preempt_No**, **RtemsTaskReqConstructErrors_Pre_Preempt_NA** }
- enum **RtemsTaskReqConstructErrors_Post_Status** { **RtemsTaskReqConstructErrors_Post_Status_Ok**, **RtemsTaskReqConstructErrors_Post_Status_InvAddress**, **RtemsTaskReqConstructErrors_Post_Status_InvName**, **RtemsTaskReqConstructErrors_Post_Status_InvPrio**, **RtemsTaskReqConstructErrors_Post_Status_InvSize**, **RtemsTaskReqConstructErrors_Post_Status_TooMany**, **RtemsTaskReqConstructErrors_Post_Status_Unsatisfied**, **RtemsTaskReqConstructErrors_Post_Status_NA** }

## Functions

- **RTEMS_ALIGNED** (RTEMS_TASK_STORAGE_ALIGNMENT)
- static bool **ThreadCreate** (rtems_tcb *executing, rtems_tcb *created)
- static void **RtemsTaskReqConstructErrors_Pre_Id_Prepare** (RtemsTaskReqConstructErrors_Context *ctx, RtemsTaskReqConstructErrors_Pre_Id state)
- static void **RtemsTaskReqConstructErrors_Pre_Name_Prepare** (RtemsTaskReqConstructErrors_Context *ctx, RtemsTaskReqConstructErrors_Pre_Name state)
- static void **RtemsTaskReqConstructErrors_Pre_Prio_Prepare** (RtemsTaskReqConstructErrors_Context *ctx, RtemsTaskReqConstructErrors_Pre_Prio state)
- static void **RtemsTaskReqConstructErrors_Pre_Tasks_Prepare** (RtemsTaskReqConstructErrors_Context *ctx, RtemsTaskReqConstructErrors_Pre_Tasks state)
- static void **RtemsTaskReqConstructErrors_Pre_TLS_Prepare** (RtemsTaskReqConstructErrors_Context *ctx, RtemsTaskReqConstructErrors_Pre_TLS state)
- static void **RtemsTaskReqConstructErrors_Pre_Stack_Prepare** (RtemsTaskReqConstructErrors_Context *ctx, RtemsTaskReqConstructErrors_Pre_Stack state)
- static void **RtemsTaskReqConstructErrors_Pre_Ext_Prepare** (RtemsTaskReqConstructErrors_Context *ctx, RtemsTaskReqConstructErrors_Pre_Ext state)
- static void **RtemsTaskReqConstructErrors_Pre_Preempt_Prepare** (RtemsTaskReqConstructErrors_Context *ctx, RtemsTaskReqConstructErrors_Pre_Preempt state)
- static void **RtemsTaskReqConstructErrors_Post_Status_Check** (RtemsTaskReqConstructErrors_Context *ctx, RtemsTaskReqConstructErrors_Post_Status state)
- static void **RtemsTaskReqConstructErrors_Setup** (RtemsTaskReqConstructErrors_Context *ctx)
- static void **RtemsTaskReqConstructErrors_Setup_Wrap** (void *arg)
- static void **RtemsTaskReqConstructErrors_Teardown** (RtemsTaskReqConstructErrors_Context *ctx)
- static void **RtemsTaskReqConstructErrors_Teardown_Wrap** (void *arg)
- static size_t **RtemsTaskReqConstructErrors_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsTaskReqConstructErrors_Prepare** (RtemsTaskReqConstructErrors_Context *ctx)
- static void **RtemsTaskReqConstructErrors_Action** (RtemsTaskReqConstructErrors_Context *ctx)
- static void **RtemsTaskReqConstructErrors_Cleanup** (RtemsTaskReqConstructErrors_Context *ctx)
- **T_TEST_CASE_FIXTURE** (RtemsTaskReqConstructErrors, &RtemsTaskReqConstructErrors_Fixture)

## Variables

- static RtemsTaskReqConstructErrors_Context **RtemsTaskReqConstructErrors_Instance**
- static const char *const **RtemsTaskReqConstructErrors_PreDesc_Id** [ ]
- static const char *const **RtemsTaskReqConstructErrors_PreDesc_Name** [ ]
- static const char *const **RtemsTaskReqConstructErrors_PreDesc_Prio** [ ]
- static const char *const **RtemsTaskReqConstructErrors_PreDesc_Tasks** [ ]
- static const char *const **RtemsTaskReqConstructErrors_PreDesc_TLS** [ ]
- static const char *const **RtemsTaskReqConstructErrors_PreDesc_Stack** [ ]
- static const char *const **RtemsTaskReqConstructErrors_PreDesc_Ext** [ ]
- static const char *const **RtemsTaskReqConstructErrors_PreDesc_Preempt** [ ]
- static const char *const *const **RtemsTaskReqConstructErrors_PreDesc** [ ]
- static _Thread_local int **tls_variable**
- static const rtems_extensions_table **extensions**
- static T_fixture **RtemsTaskReqConstructErrors_Fixture**
- static const uint8_t **RtemsTaskReqConstructErrors_TransitionMap** [ ][1]
-

struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1
  uint16_t **Pre_Id_NA**: 1
  uint16_t **Pre_Name_NA**: 1
  uint16_t **Pre_Prio_NA**: 1
  uint16_t **Pre_Tasks_NA**: 1
  uint16_t **Pre_TLS_NA**: 1
  uint16_t **Pre_Stack_NA**: 1
  uint16_t **Pre_Ext_NA**: 1
  uint16_t **Pre_Preempt_NA**: 1
  uint8_t **Pre_Pre_NA**: 1
  uint8_t **Pre_Send_NA**: 1
  uint8_t **Pre_ReceiverState_NA**: 1
  uint8_t **Pre_Satisfy_NA**: 1
  uint8_t **Pre_Node_NA**: 1
} **RtemsTaskReqConstructErrors_TransitionInfo** [ ]

## 8.222.1 Detailed Description

## 8.222.2 Variable Documentation

### 8.222.2.1 extensions

```
const rtems_extensions_table extensions  [static]
```

**Initial value:**

```
= {
  .thread_create = ThreadCreate
}
```

Definition at line 253 of file tc-task-construct-errors.c.

#### 8.222.2.2 RtemsTaskReqConstructErrors_Fixture

```
T_fixture RtemsTaskReqConstructErrors_Fixture  [static]
```

**Initial value:**

```
= {
  .setup = RtemsTaskReqConstructErrors_Setup_Wrap,
  .stop = NULL,
  .teardown = RtemsTaskReqConstructErrors_Teardown_Wrap,
  .scope = RtemsTaskReqConstructErrors_Scope,
  .initial_context = &RtemsTaskReqConstructErrors_Instance
}
```

Definition at line 589 of file tc-task-construct-errors.c.

#### 8.222.2.3 RtemsTaskReqConstructErrors_PreDesc

```
const char* const* const RtemsTaskReqConstructErrors_PreDesc[]  [static]
```

**Initial value:**

```
= {
  RtemsTaskReqConstructErrors_PreDesc_Id,
  RtemsTaskReqConstructErrors_PreDesc_Name,
  RtemsTaskReqConstructErrors_PreDesc_Prio,
  RtemsTaskReqConstructErrors_PreDesc_Tasks,
  RtemsTaskReqConstructErrors_PreDesc_TLS,
  RtemsTaskReqConstructErrors_PreDesc_Stack,
  RtemsTaskReqConstructErrors_PreDesc_Ext,
  RtemsTaskReqConstructErrors_PreDesc_Preempt,
  NULL
}
```

Definition at line 212 of file tc-task-construct-errors.c.

#### 8.222.2.4 RtemsTaskReqConstructErrors_PreDesc_Ext

```
const char* const RtemsTaskReqConstructErrors_PreDesc_Ext[]  [static]
```

**Initial value:**

```
= {
  "Ok",
  "Err",
  "NA"
}
```

Definition at line 200 of file tc-task-construct-errors.c.

#### 8.222.2.5 RtemsTaskReqConstructErrors_PreDesc_Id

```
const char* const RtemsTaskReqConstructErrors_PreDesc_Id[]  [static]
```

**Initial value:**

```
= {
  "Id",
  "Null",
  "NA"
}
```

Definition at line 163 of file tc-task-construct-errors.c.

#### 8.222.2.6 RtemsTaskReqConstructErrors_PreDesc_Name

```
const char* const RtemsTaskReqConstructErrors_PreDesc_Name[]  [static]
```

**Initial value:**

```
= {
  "Valid",
  "Inv",
  "NA"
}
```

Definition at line 169 of file tc-task-construct-errors.c.

#### 8.222.2.7 RtemsTaskReqConstructErrors_PreDesc_Preempt

```
const char* const RtemsTaskReqConstructErrors_PreDesc_Preempt[]  [static]
```

**Initial value:**

```
= {
  "Yes",
  "No",
  "NA"
}
```

Definition at line 206 of file tc-task-construct-errors.c.

#### 8.222.2.8 RtemsTaskReqConstructErrors_PreDesc_Prio

```
const char* const RtemsTaskReqConstructErrors_PreDesc_Prio[]  [static]
```

**Initial value:**

```
= {
  "Valid",
  "Zero",
  "Inv",
  "NA"
}
```

Definition at line 175 of file tc-task-construct-errors.c.

#### 8.222.2.9 RtemsTaskReqConstructErrors_PreDesc_Stack

```
const char* const RtemsTaskReqConstructErrors_PreDesc_Stack[]  [static]
```

**Initial value:**

```
= {
  "Enough",
  "Small",
  "NA"
}
```

Definition at line 194 of file tc-task-construct-errors.c.

#### 8.222.2.10 RtemsTaskReqConstructErrors_PreDesc_Tasks

```
const char* const RtemsTaskReqConstructErrors_PreDesc_Tasks[]  [static]
```

**Initial value:**

```
= {
  "Avail",
  "None",
  "NA"
}
```

Definition at line 182 of file tc-task-construct-errors.c.

#### 8.222.2.11 RtemsTaskReqConstructErrors_PreDesc_TLS

```
const char* const RtemsTaskReqConstructErrors_PreDesc_TLS[]  [static]
```

**Initial value:**

```
= {
  "Enough",
  "Small",
  "NA"
}
```

Definition at line 188 of file tc-task-construct-errors.c.

## 8.223 spec:/rtems/task/req/ident

### Files

- file tc-task-ident.c

### Classes

- struct RtemsTaskReqIdent_Context

  *Test context for spec:/rtems/task/req/ident test case.*

### Macros

- #define **TASK_ATTRIBUTES** RTEMS_DEFAULT_ATTRIBUTES
- #define **MAX_TLS_SIZE** RTEMS_ALIGN_UP( 64, RTEMS_TASK_STORAGE_ALIGNMENT )

### Enumerations

- enum **RtemsTaskReqIdent_Pre_Pre** { **RtemsTaskReqIdent_Pre_Pre_Self**, **RtemsTaskReqIdent_Pre_↩ Pre_Generic**, **RtemsTaskReqIdent_Pre_Pre_NA** }
- enum **RtemsTaskReqIdent_Post_Post** { **RtemsTaskReqIdent_Post_Post_OkAndSelfId**, **RtemsTask↩ ReqIdent_Post_Post_Generic**, **RtemsTaskReqIdent_Post_Post_NA** }

### Functions

- static rtems_status_code **ClassicTaskIdentAction** (rtems_name name, uint32_t node, rtems_id *id)
- static void **RtemsTaskReqIdent_Pre_Pre_Prepare** (RtemsTaskReqIdent_Context *ctx, RtemsTaskReq↩ Ident_Pre_Pre state)
- static void **RtemsTaskReqIdent_Post_Post_Check** (RtemsTaskReqIdent_Context *ctx, RtemsTaskReq↩ Ident_Post_Post state)
- static void **RtemsTaskReqIdent_Setup** (RtemsTaskReqIdent_Context *ctx)
- static void **RtemsTaskReqIdent_Setup_Wrap** (void *arg)
- static void **RtemsTaskReqIdent_Teardown** (RtemsTaskReqIdent_Context *ctx)
- static void **RtemsTaskReqIdent_Teardown_Wrap** (void *arg)
- static size_t **RtemsTaskReqIdent_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsTaskReqIdent_Action** (RtemsTaskReqIdent_Context *ctx)
- **T_TEST_CASE_FIXTURE** (RtemsTaskReqIdent, &RtemsTaskReqIdent_Fixture)

## Variables

- static RtemsTaskReqIdent_Context **RtemsTaskReqIdent_Instance**
- static const char *const **RtemsTaskReqIdent_PreDesc_Pre** [ ]
- static const char *const *const **RtemsTaskReqIdent_PreDesc** [ ]
- static char **ClassicTaskIdentStorage** [RTEMS_TASK_STORAGE_SIZE(MAX_TLS_SIZE+RTEMS_MINIMUM_STACK_SIZE_
  TASK_ATTRIBUTES)]
- static const rtems_task_config **ClassicTaskIdentConfig**
- static T_fixture **RtemsTaskReqIdent_Fixture**
- static const uint8_t **RtemsTaskReqIdent_TransitionMap** [ ][1]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1
  uint16_t **Pre_Id_NA**: 1
  uint16_t **Pre_Name_NA**: 1
  uint16_t **Pre_Prio_NA**: 1
  uint16_t **Pre_Tasks_NA**: 1
  uint16_t **Pre_TLS_NA**: 1
  uint16_t **Pre_Stack_NA**: 1
  uint16_t **Pre_Ext_NA**: 1
  uint16_t **Pre_Preempt_NA**: 1
  uint8_t **Pre_Pre_NA**: 1
  uint8_t **Pre_Send_NA**: 1
  uint8_t **Pre_ReceiverState_NA**: 1
  uint8_t **Pre_Satisfy_NA**: 1
  uint8_t **Pre_Node_NA**: 1
  } **RtemsTaskReqIdent_TransitionInfo** [ ]

## 8.223.1 Detailed Description

## 8.223.2 Variable Documentation

### 8.223.2.1 ClassicTaskIdentConfig

```
const rtems_task_config ClassicTaskIdentConfig  [static]
```

**Initial value:**
```
= {
  .name = ClassicObjectIdentName,
  .initial_priority = 1,
  .storage_area = ClassicTaskIdentStorage,
  .storage_size = sizeof( ClassicTaskIdentStorage ),
  .maximum_thread_local_storage_size = MAX_TLS_SIZE,
  .initial_modes = RTEMS_DEFAULT_MODES,
  .attributes = TASK_ATTRIBUTES
}
```

Definition at line 135 of file tc-task-ident.c.

#### 8.223.2.2 RtemsTaskReqIdent_Fixture

```
T_fixture RtemsTaskReqIdent_Fixture  [static]
```

**Initial value:**
```
= {
  .setup = RtemsTaskReqIdent_Setup_Wrap,
  .stop = NULL,
  .teardown = RtemsTaskReqIdent_Teardown_Wrap,
  .scope = RtemsTaskReqIdent_Scope,
  .initial_context = &RtemsTaskReqIdent_Instance
}
```

Definition at line 243 of file tc-task-ident.c.

#### 8.223.2.3 RtemsTaskReqIdent_PreDesc

```
const char* const* const RtemsTaskReqIdent_PreDesc[]  [static]
```

**Initial value:**
```
= {
  RtemsTaskReqIdent_PreDesc_Pre,
  NULL
}
```

Definition at line 109 of file tc-task-ident.c.

#### 8.223.2.4 RtemsTaskReqIdent_PreDesc_Pre

```
const char* const RtemsTaskReqIdent_PreDesc_Pre[]  [static]
```

**Initial value:**
```
= {
  "Self",
  "Generic",
  "NA"
}
```

Definition at line 103 of file tc-task-ident.c.

#### 8.223.2.5 RtemsTaskReqIdent_TransitionInfo

```
const { ...  } RtemsTaskReqIdent_TransitionInfo[]  [static]
```

**Initial value:**

```
= {
  {
    0, 0
  }, {
    0, 0
  }
}
```

#### 8.223.2.6 RtemsTaskReqIdent_TransitionMap

```
const uint8_t RtemsTaskReqIdent_TransitionMap[][1]  [static]
```

**Initial value:**

```
= {
  {
    RtemsTaskReqIdent_Post_Post_OkAndSelfId
  }, {
    RtemsTaskReqIdent_Post_Post_Generic
  }
}
```

Definition at line 251 of file tc-task-ident.c.

## 8.224 spec:/rtems/timer/val/ident

Test the rtems_timer_ident directive.

### Files

- file tc-timer-ident.c

### Functions

- static rtems_status_code **ClassicTimerIdentAction** (rtems_name name, rtems_id *id)
- void **T_case_body_RtemsTimerValIdent** (void)

### 8.224.1 Detailed Description

Test the rtems_timer_ident directive.

This test case performs the following actions:

- Run the generic object identification tests for Classic API timer class objects defined by /rtems/req/ident-local.

# 8.225 spec:/rtems/userext/val/ident

Test the rtems_extension_ident directive.

## Files

- file tc-userext-ident.c

## Functions

- static rtems_status_code **ClassicUserExtIdentAction** (rtems_name name, rtems_id *id)
- void **T_case_body_RtemsUserextValIdent** (void)

### 8.225.1 Detailed Description

Test the rtems_extension_ident directive.

This test case performs the following actions:

- Run the generic object identification tests for Classic API user extension class objects defined by /rtems/req/ident-local.

## 8.226 spec:/testsuites/validation-0

This general purpose validation test suite provides enough resources to run basic tests for all specified managers and functions.

### Modules

- spec:/rtems/attr/val/attr

  *Tests the attribute constants of the Classic API.*
- spec:/rtems/barrier/val/ident

  *Test the rtems_barrier_ident directive.*
- spec:/rtems/event/req/send-receive
- spec:/rtems/event/val/event-constant

  *Tests an event constant and number of the Event Manager using the Classic and system event sets of the executing task.*
- spec:/rtems/event/val/events

  *Tests the Event Manager API.*
- spec:/rtems/event/val/send-receive

  *Tests the rtems_event_send and rtems_event_receive directives.*
- spec:/rtems/event/val/system-send-receive

  *Tests the rtems_event_system_send and rtems_event_system_receive directives.*
- spec:/rtems/message/req/construct-errors
- spec:/rtems/message/val/ident

  *Test the rtems_message_queue_ident directive.*
- spec:/rtems/mode/val/modes

  *Tests the task mode constants and function-like macros of the Classic API.*
- spec:/rtems/option/val/options

  *Tests the option constants of the Classic API.*
- spec:/rtems/part/req/create
- spec:/rtems/part/req/delete
- spec:/rtems/part/req/get-buffer
- spec:/rtems/part/req/return-buffer
- spec:/rtems/part/val/ident

  *Test the rtems_partition_ident directive.*
- spec:/rtems/part/val/part

  *Validates some functional requirements of the Partition Manager.*
- spec:/rtems/ratemon/val/ident

  *Test the rtems_rate_monotonic_ident directive.*
- spec:/rtems/req/ident
- spec:/rtems/req/ident-local
- spec:/rtems/sem/val/ident

  *Test the rtems_semaphore_ident directive.*
- spec:/rtems/task/req/construct-errors
- spec:/rtems/task/req/ident
- spec:/rtems/timer/val/ident

  *Test the rtems_timer_ident directive.*
- spec:/rtems/userext/val/ident

  *Test the rtems_extension_ident directive.*

## Files

- file ts-validation-0.c

## Macros

- #define **MAX_TLS_SIZE** RTEMS_ALIGN_UP( 64, RTEMS_TASK_STORAGE_ALIGNMENT )
- #define **ATTRIBUTES** RTEMS_FLOATING_POINT
- #define **CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER**
- #define **CONFIGURE_MAXIMUM_PROCESSORS** 4
- #define **CONFIGURE_MAXIMUM_BARRIERS** 3
- #define **CONFIGURE_MAXIMUM_MESSAGE_QUEUES** 3
- #define **CONFIGURE_MAXIMUM_PARTITIONS** 3
- #define **CONFIGURE_MAXIMUM_PERIODS** 3
- #define **CONFIGURE_MAXIMUM_SEMAPHORES** 3
- #define **CONFIGURE_MAXIMUM_TASKS** 3
- #define **CONFIGURE_MINIMUM_TASKS_WITH_USER_PROVIDED_STORAGE** CONFIGURE_MAXIMUM_TASKS
- #define **CONFIGURE_MAXIMUM_TIMERS** 3
- #define **CONFIGURE_MAXIMUM_USER_EXTENSIONS** 3
- #define **CONFIGURE_MICROSECONDS_PER_TICK** 1000
- #define **CONFIGURE_MAXIMUM_FILE_DESCRIPTORS** 0
- #define **CONFIGURE_DISABLE_NEWLIB_REENTRANCY**
- #define **CONFIGURE_APPLICATION_DISABLE_FILESYSTEM**
- #define **CONFIGURE_IDLE_TASK_INITIALIZES_APPLICATION**
- #define **CONFIGURE_IDLE_TASK_BODY** _CPU_Thread_Idle_body
- #define **CONFIGURE_SCHEDULER_EDF_SMP**
- #define **CONFIGURE_SCHEDULER_TABLE_ENTRIES**
- #define **CONFIGURE_SCHEDULER_ASSIGNMENTS**
- #define **CONFIGURE_INIT**

## Enumerations

- enum {
  **_Sysinit_bsp_start** = 0x00030080, **_Sysinit_bsp_debug_uart_init** = 0x00030003, **_Sysinit_amba_↩ initialize** = 0x00030001, **_Sysinit_leon3_cpu_index_init** = 0x00030000,
  **_Sysinit_leon3_counter_initialize** = 0x00040000, **_Sysinit_bsp_memory_initialize** = 0x00018080, **_↩ Sysinit__Malloc_Initialize** = 0x00028080, **_Sysinit__Barrier_Manager_initialization** = 0x00140080,
  **_Sysinit__Message_queue_Manager_initialization** = 0x000e0080, **_Sysinit__Partition_Manager_↩ initialization** = 0x00100080, **_Sysinit__Rate_monotonic_Manager_initialization** = 0x00130080, **_↩ Sysinit__Semaphore_Manager_initialization** = 0x000f0080,
  **_Sysinit__RTEMS_tasks_Manager_initialization** = 0x000a0080, **_Sysinit__Timer_Manager_↩ initialization** = 0x000b0080, **_Sysinit_rtems_initialize_data_structures** = 0x00070080, **_Sysinit__↩ Extension_Manager_initialization** = 0x00090080,
  **_Sysinit__Per_CPU_Data_initialize** = 0x00022080, **_Sysinit__Workspace_Handler_initialization** = 0x00026080, **_Sysinit_init_task** = 0x00290080, **_Sysinit_init_runner_task** = 0x00290080 }

## Functions

- static void **runner_task** (rtems_task_argument arg)
- static void **init_runner_task** (void)
- **RTEMS_SCHEDULER_EDF_SMP** (a)
- **RTEMS_SCHEDULER_EDF_SMP** (b)
- **RTEMS_SCHEDULER_EDF_SMP** (c)

## Variables

- const char rtems_test_name [ ] = "Validation0"

  *Each test must define a test name string.*
- static char **buffer** [512]
- static const T_action **actions** [ ]
- static const T_config **test_config**
- static char **runner_task_storage** [RTEMS_TASK_STORAGE_SIZE(MAX_TLS_SIZE+RTEMS_MINIMUM_STACK_SIZE, ATTRIBUTES)]
- static const rtems_task_config **runner_task_config**
- rtems_sysinit_item const **_Linker_set__Sysinit_init_runner_task** = { init_runner_task }

### 8.226.1 Detailed Description

This general purpose validation test suite provides enough resources to run basic tests for all specified managers and functions.

In SMP configurations, up to three scheduler instances using the SMP EDF scheduler are provided using up to four processors.

### 8.226.2 Macro Definition Documentation

#### 8.226.2.1 CONFIGURE_SCHEDULER_ASSIGNMENTS

```
#define CONFIGURE_SCHEDULER_ASSIGNMENTS
```

**Value:**

```
  RTEMS_SCHEDULER_ASSIGN(0, RTEMS_SCHEDULER_ASSIGN_PROCESSOR_MANDATORY), \
  RTEMS_SCHEDULER_ASSIGN(1, RTEMS_SCHEDULER_ASSIGN_PROCESSOR_OPTIONAL), \
  RTEMS_SCHEDULER_ASSIGN(2, RTEMS_SCHEDULER_ASSIGN_PROCESSOR_OPTIONAL), \
  RTEMS_SCHEDULER_ASSIGN(2, RTEMS_SCHEDULER_ASSIGN_PROCESSOR_OPTIONAL)
```

Definition at line 215 of file ts-validation-0.c.

#### 8.226.2.2 CONFIGURE_SCHEDULER_TABLE_ENTRIES

```
#define CONFIGURE_SCHEDULER_TABLE_ENTRIES
```

**Value:**

```
  RTEMS_SCHEDULER_TABLE_EDF_SMP(a, rtems_build_name('A', ' ', ' ', ' ')), \
  RTEMS_SCHEDULER_TABLE_EDF_SMP(b, rtems_build_name('B', ' ', ' ', ' ')), \
  RTEMS_SCHEDULER_TABLE_EDF_SMP(c, rtems_build_name('C', ' ', ' ', ' '))
```

Definition at line 210 of file ts-validation-0.c.

### 8.226.3 Variable Documentation

#### 8.226.3.1 actions

```
const T_action actions[]  [static]
```

**Initial value:**

```
= {
  T_report_hash_sha256,
  T_check_task_context,
  T_check_rtems_barriers,
  T_check_rtems_extensions,
  T_check_rtems_message_queues,
  T_check_rtems_partitions,
  T_check_rtems_periods,
  T_check_rtems_semaphores,
  T_check_rtems_tasks,
  T_check_rtems_timers
}
```

Definition at line 78 of file ts-validation-0.c.

#### 8.226.3.2 runner_task_config

```
const rtems_task_config runner_task_config  [static]
```

**Initial value:**

```
= {
  .name = rtems_build_name( 'R', 'U', 'N', ' ' ),
  .initial_priority = 1,
  .storage_area = runner_task_storage,
  .storage_size = sizeof( runner_task_storage ),
  .maximum_thread_local_storage_size = MAX_TLS_SIZE,
  .initial_modes = RTEMS_DEFAULT_MODES,
  .attributes = ATTRIBUTES
}
```

Definition at line 131 of file ts-validation-0.c.

#### 8.226.3.3 test_config

```
const T_config test_config  [static]
```

**Initial value:**

```
= {
  .name = rtems_test_name,
  .buf = buffer,
  .buf_size = sizeof( buffer ),
  .putchar = rtems_put_char,
  .verbosity = RTEMS_TEST_VERBOSITY,
  .now = T_now_clock,
  .action_count = T_ARRAY_SIZE( actions ),
  .actions = actions
}
```

Definition at line 91 of file ts-validation-0.c.

## 8.227 spec:/testsuites/validation/c-library

This test case calls functions of the C Library which are included in the space profile.

### Files

- file tc-space-profile.c

### Functions

- void **T_case_body_TestsuitesValidationCLibrary** (void)

### 8.227.1 Detailed Description

This test case calls functions of the C Library which are included in the space profile.

This test case performs the following actions:

- Allocate four bytes with an alignment of 128 bytes with aligned_alloc().
  - Check that the returned pointer is not NULL.
  - Check that the returned pointer is aligned by 128 bytes.
- Allocate four bytes with rtems_malloc().
  - Check that the returned pointer is not NULL.
- Set an integer variable to one and then to zero with memset().
  - Check that the integer variable is equal to zero.
- Set a source integer variable to one, set a destination integer variable to two, and copy the source to the destination variable through memcpy().
  - Check that the destination integer variable is equal to one.

# 8.228 spec:/testsuites/validation/classic-barrier

This test case calls functions of the Barrier Manager which are included in the space profile.

## Files

- file tc-space-profile.c

## Functions

- void **T_case_body_TestsuitesValidationClassicBarrier** (void)

### 8.228.1 Detailed Description

This test case calls functions of the Barrier Manager which are included in the space profile.

This test case performs the following actions:

- Set an object identifier to an invalid value and create an automatic release barrier object for one task.
  - Check that the barrier creation was successful.
- Set a second object identifier to an invalid value and identify a barrier object by its name.
  - Check that the barrier identification by name was successful.
  - Check that the second identifier is equal to the one returned by the barrier creation.
- Set the count of released tasks to one and release the barrier.
  - Check that the barrier release was successful.
  - Check that the count of released tasks is zero.
- Wait with an infinite timeout for the barrier.
  - Check that the barrier wait was successful.

## 8.229 spec:/testsuites/validation/profile

This test suite contains test cases which call in combination all functions included in the space profile.

### Modules

- spec:/testsuites/validation/c-library

  *This test case calls functions of the C Library which are included in the space profile.*
- spec:/testsuites/validation/classic-barrier

  *This test case calls functions of the Barrier Manager which are included in the space profile.*

### Files

- file ts-space-profile.c

### Macros

- #define **NAME** rtems_build_name('N', 'A', 'M', 'E')
- #define **INIT_TASK_ATTRIBUTES** RTEMS_DEFAULT_ATTRIBUTES
- #define **MAX_TLS_SIZE** RTEMS_ALIGN_UP(64, RTEMS_TASK_STORAGE_ALIGNMENT)
- #define **CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER**
- #define **CONFIGURE_MAXIMUM_PROCESSORS** 4
- #define **CONFIGURE_MAXIMUM_BARRIERS** 1
- #define **CONFIGURE_MAXIMUM_MESSAGE_QUEUES** 1
- #define **CONFIGURE_MAXIMUM_PARTITIONS** 1
- #define **CONFIGURE_MAXIMUM_PERIODS** 1
- #define **CONFIGURE_MAXIMUM_SEMAPHORES** 1
- #define **CONFIGURE_MAXIMUM_TASKS** 1
- #define **CONFIGURE_MAXIMUM_TIMERS** 1
- #define **CONFIGURE_MAXIMUM_USER_EXTENSIONS** 1
- #define **CONFIGURE_MESSAGE_BUFFER_MEMORY** 1
- #define **CONFIGURE_MICROSECONDS_PER_TICK** 10000
- #define **CONFIGURE_SCHEDULER_NAME** NAME
- #define **CONFIGURE_INITIAL_EXTENSIONS** { .fatal = fatal_extension }
- #define **CONFIGURE_MAXIMUM_FILE_DESCRIPTORS** 0
- #define **CONFIGURE_DISABLE_NEWLIB_REENTRANCY**
- #define **CONFIGURE_APPLICATION_DISABLE_FILESYSTEM**
- #define **CONFIGURE_IDLE_TASK_INITIALIZES_APPLICATION**
- #define **CONFIGURE_IDLE_TASK_BODY** _CPU_Thread_Idle_body
- #define **CONFIGURE_INIT**

### Enumerations

- enum {
  **_Sysinit_bsp_start** = 0x00030080, **_Sysinit_bsp_debug_uart_init** = 0x00030003, **_Sysinit_amba_↩ initialize** = 0x00030001, **_Sysinit_leon3_cpu_index_init** = 0x00030000,
  **_Sysinit_leon3_counter_initialize** = 0x00040000, **_Sysinit_bsp_memory_initialize** = 0x00018080, _↩ **Sysinit__Malloc_Initialize** = 0x00028080, **_Sysinit__Barrier_Manager_initialization** = 0x00140080,
  **_Sysinit__Message_queue_Manager_initialization** = 0x000e0080, **_Sysinit__Partition_Manager_↩ initialization** = 0x00100080, **_Sysinit__Rate_monotonic_Manager_initialization** = 0x00130080, _↩ **Sysinit__Semaphore_Manager_initialization** = 0x000f0080,
  **_Sysinit__RTEMS_tasks_Manager_initialization** = 0x000a0080, **_Sysinit__Timer_Manager_↩ initialization** = 0x000b0080, **_Sysinit_rtems_initialize_data_structures** = 0x00070080, **_Sysinit__↩ Extension_Manager_initialization** = 0x00090080,
  **_Sysinit__Per_CPU_Data_initialize** = 0x00022080, **_Sysinit__Workspace_Handler_initialization** = 0x00026080, **_Sysinit_init_task** = 0x00290080, **_Sysinit_init_runner_task** = 0x00290080 }

## Functions

- static void **fatal_extension** (rtems_fatal_source source, bool always_set_to_false, rtems_fatal_code error)
- static void **Init** (rtems_task_argument arg)
- **RTEMS_ALIGNED** (RTEMS_TASK_STORAGE_ALIGNMENT)
- static void **init_task** (void)

## Variables

- static const T_action **actions** [ ]
- static const T_config **test_config**
- static const rtems_task_config **task_config**
- rtems_sysinit_item const **_Linker_set__Sysinit_init_task** = { init_task }

### 8.229.1 Detailed Description

This test suite contains test cases which call in combination all functions included in the space profile.

### 8.229.2 Variable Documentation

#### 8.229.2.1 actions

```
const T_action actions[]  [static]
```

**Initial value:**

```
= {
  T_report_hash_sha256,
  check_task_context,
  T_check_rtems_barriers,
  T_check_rtems_extensions,
  T_check_rtems_message_queues,
  T_check_rtems_partitions,
  T_check_rtems_periods,
  T_check_rtems_semaphores,
  T_check_rtems_tasks,
  T_check_rtems_timers
}
```

Definition at line 119 of file ts-space-profile.c.

#### 8.229.2.2 task_config

```
const rtems_task_config task_config  [static]
```

**Initial value:**

```
= {
  .name = NAME,
  .initial_priority = 1,
  .storage_area = init_task_storage,
  .storage_size = sizeof(init_task_storage),
  .maximum_thread_local_storage_size = MAX_TLS_SIZE,
  .initial_modes = RTEMS_DEFAULT_MODES,
  .attributes = INIT_TASK_ATTRIBUTES
}
```

Definition at line 143 of file ts-space-profile.c.

#### 8.229.2.3 test_config

```
const T_config test_config  [static]
```

**Initial value:**

```
= {
  .name = "SpaceProfile",
  .buf = buffer,
  .buf_size = sizeof(buffer),
  .putchar = rtems_put_char,
  .verbosity = T_VERBOSE,
  .now = T_now_clock,
  .action_count = T_ARRAY_SIZE(actions),
  .actions = actions
}
```

Definition at line 132 of file ts-space-profile.c.

# Chapter 9

# Class Documentation

## 9.1 __rtems_dev_t Union Reference

### Public Attributes

- dev_t **device**
- struct {
  rtems_device_major_number **major**
  rtems_device_minor_number **minor**
  } **__overlay**

### 9.1.1 Detailed Description

Definition at line 1492 of file libio.h.

The documentation for this union was generated from the following file:

- cpukit/include/rtems/libio.h

## 9.2 _rtems_filesystem_file_handlers_r Struct Reference

File system node operations table.

```
#include <libio.h>
```

### Public Attributes

- rtems_filesystem_open_t **open_h**
- rtems_filesystem_close_t **close_h**
- rtems_filesystem_read_t **read_h**
- rtems_filesystem_write_t **write_h**
- rtems_filesystem_ioctl_t **ioctl_h**
- rtems_filesystem_lseek_t **lseek_h**
- rtems_filesystem_fstat_t **fstat_h**
- rtems_filesystem_ftruncate_t **ftruncate_h**
- rtems_filesystem_fsync_t **fsync_h**
- rtems_filesystem_fdatasync_t **fdatasync_h**
- rtems_filesystem_fcntl_t **fcntl_h**
- rtems_filesystem_poll_t **poll_h**
- rtems_filesystem_kqfilter_t **kqfilter_h**
- rtems_filesystem_readv_t **readv_h**
- rtems_filesystem_writev_t **writev_h**
- rtems_filesystem_mmap_t **mmap_h**

#### 9.2.1 Detailed Description

File system node operations table.

Definition at line 1005 of file libio.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/libio.h

## 9.3 _rtems_filesystem_operations_table Struct Reference

File system operations table.

`#include <libio.h>`

### Public Attributes

- rtems_filesystem_mt_entry_lock_t **lock_h**
- rtems_filesystem_mt_entry_unlock_t **unlock_h**
- rtems_filesystem_eval_path_t **eval_path_h**
- rtems_filesystem_link_t **link_h**
- rtems_filesystem_are_nodes_equal_t **are_nodes_equal_h**
- rtems_filesystem_mknod_t **mknod_h**
- rtems_filesystem_rmnod_t **rmnod_h**
- rtems_filesystem_fchmod_t **fchmod_h**
- rtems_filesystem_chown_t **chown_h**
- rtems_filesystem_clonenode_t **clonenod_h**
- rtems_filesystem_freenode_t **freenod_h**
- rtems_filesystem_mount_t **mount_h**
- rtems_filesystem_unmount_t **unmount_h**
- rtems_filesystem_fsunmount_me_t **fsunmount_me_h**
- rtems_filesystem_utime_t **utime_h**
- rtems_filesystem_symlink_t **symlink_h**
- rtems_filesystem_readlink_t **readlink_h**
- rtems_filesystem_rename_t **rename_h**
- rtems_filesystem_statvfs_t **statvfs_h**

### 9.3.1 Detailed Description

File system operations table.

Definition at line 472 of file libio.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/libio.h

## 9.4 _Scheduler_Control Struct Reference

Scheduler control.

`#include <scheduler.h>`

### Public Attributes

- Scheduler_Context * context
  *Reference to a statically allocated scheduler context.*
- Scheduler_Operations Operations
  *The scheduler operations.*
- Priority_Control maximum_priority
  *The maximum priority value of this scheduler.*
- uint32_t name
  *The scheduler name.*
- bool is_non_preempt_mode_supported
  *True if the non-preempt mode for threads is supported by the scheduler, otherwise false.*

### 9.4.1 Detailed Description

Scheduler control.

Definition at line 264 of file scheduler.h.

### 9.4.2 Member Data Documentation

#### 9.4.2.1 maximum_priority

`Priority_Control _Scheduler_Control::maximum_priority`

The maximum priority value of this scheduler.

It defines the lowest (least important) thread priority for this scheduler. For example the idle threads have this priority.

Definition at line 281 of file scheduler.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/scheduler.h

## 9.5 _Thread_Control Struct Reference

`#include <thread.h>`

### Public Attributes

- Objects_Control Object
- Thread_queue_Control Join_queue

  *Thread queue for thread join operations and multi-purpose lock.*
- States_Control current_state
- Priority_Node Real_priority

  *The base priority of this thread in its home scheduler instance.*
- Thread_Scheduler_control Scheduler

  *Scheduler related control.*
- Thread_Wait_information Wait
- Thread_Timer_information Timer
- bool is_idle
- bool is_preemptible
- bool is_fp
- bool was_created_with_inherited_scheduler

  *True, if the thread was created with an inherited scheduler (PTHREAD_INHERIT_SCHED), and false otherwise.*
- uint32_t cpu_time_budget
- Thread_CPU_budget_algorithms budget_algorithm
- Thread_CPU_budget_algorithm_callout budget_callout
- Timestamp_Control cpu_time_used
- Thread_Start_information Start
- Thread_Action_control **Post_switch_actions**
- Context_Control Registers
- struct _reent * libc_reent
- void * API_Extensions [THREAD_API_LAST+1]
- Thread_Keys_information Keys

  *The POSIX Keys information.*
- Thread_Life_control Life

  *Thread life-cycle control.*
- Thread_Capture_control **Capture**
- struct rtems_user_env_t * user_environment

  *Pointer to an optional thread-specific POSIX user environment.*
- struct _pthread_cleanup_context * last_cleanup_context

  *LIFO list of POSIX cleanup contexts.*
- struct User_extensions_Iterator * last_user_extensions_iterator

  *LIFO list of user extensions iterators.*
- void * extensions [RTEMS_ZERO_LENGTH_ARRAY]

  *Variable length array of user extension pointers.*

### 9.5.1 Detailed Description

This structure defines the Thread Control Block (TCB).

Uses a leading underscore in the structure name to allow forward declarations in standard header files provided by Newlib and GCC.

In case the second member changes (currently Join_queue), then the memset() in _Thread_Initialize() must be adjusted.

Definition at line 725 of file thread.h.

## 9.5.2 Member Data Documentation

### 9.5.2.1 API_Extensions

```
void* _Thread_Control::API_Extensions[THREAD_API_LAST+1]
```

This array contains the API extension area pointers.

Definition at line 840 of file thread.h.

### 9.5.2.2 budget_algorithm

```
Thread_CPU_budget_algorithms _Thread_Control::budget_algorithm
```

This field is the algorithm used to manage this thread's time quantum. The algorithm may be specified as none which case, no limit is in place.

Definition at line 814 of file thread.h.

### 9.5.2.3 budget_callout

```
Thread_CPU_budget_algorithm_callout _Thread_Control::budget_callout
```

This field is the method invoked with the budgeted time is consumed.

Definition at line 816 of file thread.h.

### 9.5.2.4 cpu_time_budget

```
uint32_t _Thread_Control::cpu_time_budget
```

This field is the length of the time quantum that this thread is allowed to consume. The algorithm used to manage limits on CPU usage is specified by budget_algorithm.

Definition at line 809 of file thread.h.

#### 9.5.2.5 cpu_time_used

```
Timestamp_Control _Thread_Control::cpu_time_used
```

This field is the amount of CPU time consumed by this thread since it was created.

Definition at line 820 of file thread.h.

#### 9.5.2.6 current_state

```
States_Control _Thread_Control::current_state
```

This field is the current execution state of this thread.

Definition at line 749 of file thread.h.

#### 9.5.2.7 extensions

```
void* _Thread_Control::extensions[RTEMS_ZERO_LENGTH_ARRAY]
```

Variable length array of user extension pointers.

The length is defined by the application via <rtems/confdefs.h>.

Definition at line 876 of file thread.h.

#### 9.5.2.8 is_fp

```
bool _Thread_Control::is_fp
```

This field is true if the thread uses the floating point unit.

Definition at line 797 of file thread.h.

#### 9.5.2.9 is_idle

```
bool _Thread_Control::is_idle
```

This field is true if the thread is an idle thread.

Definition at line 789 of file thread.h.

#### 9.5.2.10 is_preemptible

```
bool _Thread_Control::is_preemptible
```

This field is true if the thread is preemptible.

Definition at line 795 of file thread.h.

#### 9.5.2.11 Join_queue

```
Thread_queue_Control _Thread_Control::Join_queue
```

Thread queue for thread join operations and multi-purpose lock.

The lock of this thread queue is used for various purposes. It protects the following fields

- RTEMS_API_Control::Signal,
- Thread_Control::budget_algorithm,
- Thread_Control::budget_callout,
- Thread_Control::cpu_time_budget,
- Thread_Control::current_state,
- Thread_Control::Post_switch_actions,
- Thread_Control::Scheduler::control, and
- Thread_Control::Scheduler::own_control.

**See also**

_Thread_State_acquire().

Definition at line 746 of file thread.h.

#### 9.5.2.12 libc_reent

```
struct _reent* _Thread_Control::libc_reent
```

This field points to the newlib reentrancy structure for this thread.

Definition at line 838 of file thread.h.

#### 9.5.2.13 Life

Thread_Life_control _Thread_Control::Life

Thread life-cycle control.

Control state changes triggered by thread restart and delete requests.

Definition at line 852 of file thread.h.

#### 9.5.2.14 Object

Objects_Control _Thread_Control::Object

This field is the object management structure for each thread.

Definition at line 727 of file thread.h.

#### 9.5.2.15 Registers

Context_Control _Thread_Control::Registers

This field contains the context of this thread.

Definition at line 830 of file thread.h.

#### 9.5.2.16 Start

Thread_Start_information _Thread_Control::Start

This field contains information about the starting state of this thread.

Definition at line 825 of file thread.h.

#### 9.5.2.17 Timer

Thread_Timer_information _Thread_Control::Timer

This field is the Watchdog used to manage thread delays and timeouts.

Definition at line 769 of file thread.h.

#### 9.5.2.18 Wait

```
Thread_Wait_information _Thread_Control::Wait
```

This field is the blocking information for this thread.

Definition at line 767 of file thread.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/thread.h

## 9.6 _Thread_queue_Heads Struct Reference

Thread queue heads.

```
#include <threadq.h>
```

### Public Attributes

- union {
  Chain_Control Fifo
  *This is the FIFO discipline list.*
  } Heads

- Chain_Control Free_chain
  *A chain with free thread queue heads providing the spare thread queue heads for a thread once it is dequeued.*
- Chain_Node Free_node
  *A chain node to add these thread queue heads to the free chain of the thread queue heads dedicated to the thread queue of an object.*
- Thread_queue_Priority_queue Priority [RTEMS_ZERO_LENGTH_ARRAY]
  *One priority queue per scheduler instance.*

### 9.6.1 Detailed Description

Thread queue heads.

Each thread is equipped with spare thread queue heads in case it is not enqueued on a thread queue. The first thread enqueued on a thread queue will give its spare thread queue heads to that thread queue. The threads arriving at the queue will add their thread queue heads to the free chain of the queue heads provided by the first thread enqueued. Once a thread is dequeued it use the free chain to get new spare thread queue heads.

Uses a leading underscore in the structure name to allow forward declarations in standard header files provided by Newlib and GCC.

Definition at line 360 of file threadq.h.

### 9.6.2 Member Data Documentation

#### 9.6.2.1 Fifo

```
Chain_Control _Thread_queue_Heads::Fifo
```

This is the FIFO discipline list.

On SMP configurations this FIFO is used to enqueue the per scheduler instance priority queues of this structure. This ensures FIFO fairness among the highest priority thread of each scheduler instance.

Definition at line 372 of file threadq.h.

#### 9.6.2.2 Heads

```
union { ... } _Thread_queue_Heads::Heads
```

This union contains the data structures used to manage the blocked set of tasks which varies based upon the discipline.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/threadq.h

## 9.7 ambapp_ahb_bus Struct Reference

### Public Attributes

- unsigned int **ioarea**
- unsigned int **freq_hz**
- struct ambapp_dev * **bridge**
- struct ambapp_dev * **dev**

### 9.7.1 Detailed Description

Definition at line 98 of file ambapp.h.

The documentation for this struct was generated from the following file:

- bsps/include/grlib/ambapp.h

## 9.8 ambapp_ahb_info Struct Reference

### Public Attributes

- struct ambapp_common_info **common**
- unsigned int **start** [4]
- unsigned int **mask** [4]
- char **type** [4]
- unsigned int **custom** [3]

### 9.8.1 Detailed Description

Definition at line 62 of file ambapp.h.

The documentation for this struct was generated from the following file:

- bsps/include/grlib/ambapp.h

## 9.9 ambapp_apb_info Struct Reference

### Public Attributes

- struct ambapp_common_info **common**
- unsigned int **start**
- unsigned int **mask**

### 9.9.1 Detailed Description

Definition at line 54 of file ambapp.h.

The documentation for this struct was generated from the following file:

- bsps/include/grlib/ambapp.h

## 9.10 ambapp_bus Struct Reference

### Public Attributes

- struct ambapp_dev * **root**
- struct ambapp_mmap * **mmaps**
- struct ambapp_ahb_bus **ahbs** [AHB_BUS_MAX]

### 9.10.1 Detailed Description

Definition at line 112 of file ambapp.h.

The documentation for this struct was generated from the following file:

- bsps/include/grlib/ambapp.h

## 9.11 ambapp_common_info Struct Reference

### Public Attributes

- unsigned char **irq**
- unsigned char **ver**
- unsigned char **ahbidx**

### 9.11.1 Detailed Description

Definition at line 48 of file ambapp.h.

The documentation for this struct was generated from the following file:

- bsps/include/grlib/ambapp.h

## 9.12 ambapp_core Struct Reference

### Public Attributes

- char **irq**
- unsigned char **vendor**
- unsigned short **device**
- int **index**
- struct ambapp_ahb_info * **ahb_mst**
- struct ambapp_ahb_info * **ahb_slv**
- struct ambapp_apb_info * **apb_slv**

### 9.12.1 Detailed Description

Definition at line 88 of file ambapp.h.

The documentation for this struct was generated from the following file:

- bsps/include/grlib/ambapp.h

## 9.13 ambapp_dev Struct Reference

### Public Attributes

- struct ambapp_dev * **next**
- struct ambapp_dev * **prev**
- struct ambapp_dev * **children**
- void * **owner**
- unsigned char **dev_type**
- unsigned char **vendor**
- unsigned short **device**
- struct ambapp_common_info **devinfo** [0]

### 9.13.1 Detailed Description

Definition at line 73 of file ambapp.h.

The documentation for this struct was generated from the following file:

- bsps/include/grlib/ambapp.h

## 9.14 ambapp_mmap Struct Reference

### Public Attributes

- unsigned int **size**
- unsigned int **local_adr**
- unsigned int **remote_adr**

### 9.14.1 Detailed Description

Definition at line 105 of file ambapp.h.

The documentation for this struct was generated from the following file:

- bsps/include/grlib/ambapp.h

## 9.15 ambapp_pnp_ahb Struct Reference

### Public Attributes

- const unsigned int **id**
- const unsigned int **custom** [3]
- const unsigned int **mbar** [4]

### 9.15.1 Detailed Description

Definition at line 147 of file ambapp.h.

The documentation for this struct was generated from the following file:

- bsps/include/grlib/ambapp.h

## 9.16 ambapp_pnp_apb Struct Reference

### Public Attributes

- const unsigned int **id**
- const unsigned int **iobar**

### 9.16.1 Detailed Description

Definition at line 153 of file ambapp.h.

The documentation for this struct was generated from the following file:

- bsps/include/grlib/ambapp.h

## 9.17 apbuart_regs Struct Reference

### Public Attributes

- volatile unsigned int **data**
- volatile unsigned int **status**
- volatile unsigned int **ctrl**
- volatile unsigned int **scaler**

### 9.17.1 Detailed Description

Definition at line 41 of file grlib.h.

The documentation for this struct was generated from the following file:

- bsps/include/grlib/grlib.h

## 9.18 API_Mutex_Control Struct Reference

Control block used to manage each API mutex.

`#include <apimutex.h>`

### Public Attributes

- struct _Mutex_recursive_Control Mutex
- Thread_Life_state previous_thread_life_state

  *The thread life protection state before the outer-most mutex obtain.*

### 9.18.1 Detailed Description

Control block used to manage each API mutex.

Definition at line 42 of file apimutex.h.

### 9.18.2 Member Data Documentation

#### 9.18.2.1 Mutex

`struct _Mutex_recursive_Control API_Mutex_Control::Mutex`

A recursive mutex.

Definition at line 102 of file apimutex.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/apimutex.h

# 9.19 ASR_Information Struct Reference

`#include <asrdata.h>`

## Public Attributes

- bool is_enabled
- rtems_asr_entry handler
- rtems_mode mode_set
- rtems_signal_set signals_posted
- rtems_signal_set signals_pending
- uint32_t nest_level

## 9.19.1 Detailed Description

The following defines the control structure used to manage signals. Each thread has a copy of this record.

Definition at line 36 of file asrdata.h.

## 9.19.2 Member Data Documentation

### 9.19.2.1 handler

`rtems_asr_entry ASR_Information::handler`

This field indicates if address of the signal handler function.

Definition at line 40 of file asrdata.h.

### 9.19.2.2 is_enabled

`bool ASR_Information::is_enabled`

This field indicates if are ASRs enabled currently.

Definition at line 38 of file asrdata.h.

#### 9.19.2.3 mode_set

```
rtems_mode ASR_Information::mode_set
```

This field indicates if the task mode the signal will run with.

Definition at line 42 of file asrdata.h.

#### 9.19.2.4 nest_level

```
uint32_t ASR_Information::nest_level
```

This field indicates if nest level of signals being processed

Definition at line 48 of file asrdata.h.

#### 9.19.2.5 signals_pending

```
rtems_signal_set ASR_Information::signals_pending
```

This field indicates the signal set that is pending.

Definition at line 46 of file asrdata.h.

#### 9.19.2.6 signals_posted

```
rtems_signal_set ASR_Information::signals_posted
```

This field indicates the signal set that is posted.

Definition at line 44 of file asrdata.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/asrdata.h

## 9.20 Barrier_Control Struct Reference

```
#include <barrierdata.h>
```

## Public Attributes

- Objects_Control Object
- CORE_barrier_Control Barrier
- rtems_attribute attribute_set

### 9.20.1 Detailed Description

This type defines the control block used to manage each barrier.

Definition at line 38 of file barrierdata.h.

### 9.20.2 Member Data Documentation

#### 9.20.2.1 attribute_set

```
rtems_attribute Barrier_Control::attribute_set
```

This is used to specify the attributes of a barrier.

Definition at line 44 of file barrierdata.h.

#### 9.20.2.2 Barrier

```
CORE_barrier_Control Barrier_Control::Barrier
```

This is used to implement the barrier.

Definition at line 42 of file barrierdata.h.

#### 9.20.2.3 Object

```
Objects_Control Barrier_Control::Object
```

This is used to manage a barrier as an object.

Definition at line 40 of file barrierdata.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/barrierdata.h

## 9.21 bsp_interrupt_handler_entry Struct Reference

### Public Attributes

- rtems_interrupt_handler **handler**
- void * **arg**
- const char * **info**
- struct bsp_interrupt_handler_entry * **next**

### 9.21.1 Detailed Description

Definition at line 77 of file irq-generic.h.

The documentation for this struct was generated from the following file:

- bsps/include/bsp/irq-generic.h

## 9.22 Chain_Control Struct Reference

`#include <chain.h>`

### Public Attributes

- struct {
  Chain_Node **Node**
  Chain_Node * **fill**
  } **Head**

- struct {
  Chain_Node * **fill**
  Chain_Node **Node**
  } **Tail**

### 9.22.1 Detailed Description

This is used to manage a chain. A chain consists of a doubly linked list of zero or more nodes.

**Note**

This implementation does not require special checks for manipulating the first and last elements on the chain. To accomplish this the *Chain_Control* structure is treated as two overlapping Chain_Node structures.

Definition at line 86 of file chain.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/chain.h

## 9.23 Chain_Iterator Struct Reference

A chain iterator which is updated during node extraction if it is properly registered.

`#include <chainimpl.h>`

### Public Attributes

- Chain_Node Registry_node
  *Node for registration.*
- Chain_Iterator_direction direction
  *The direction of this iterator.*
- Chain_Node * position
  *The current position of this iterator.*

### 9.23.1 Detailed Description

A chain iterator which is updated during node extraction if it is properly registered.

**See also**

_Chain_Iterator_initialize().

Definition at line 885 of file chainimpl.h.

### 9.23.2 Member Data Documentation

#### 9.23.2.1 direction

`Chain_Iterator_direction Chain_Iterator::direction`

The direction of this iterator.

Immutable after initialization via _Chain_Iterator_initialize().

Definition at line 898 of file chainimpl.h.

#### 9.23.2.2 position

```
Chain_Node* Chain_Iterator::position
```

The current position of this iterator.

The position is initialized via _Chain_Iterator_initialize(). It must be explicitly set after one valid iteration step, e.g. in case a next node in the iterator direction existed. It is updated through the registration in case a node is extracted via _Chain_Iterator_registry_update().

Definition at line 908 of file chainimpl.h.

#### 9.23.2.3 Registry_node

```
Chain_Node Chain_Iterator::Registry_node
```

Node for registration.

Used during _Chain_Iterator_initialize() and _Chain_Iterator_destroy().

Definition at line 891 of file chainimpl.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/chainimpl.h

## 9.24 Chain_Iterator_registry Struct Reference

A registry for chain iterators.

```
#include <chainimpl.h>
```

### Public Attributes

- Chain_Control **Iterators**

### 9.24.1 Detailed Description

A registry for chain iterators.

Should be attached to a chain control to enable safe iteration through a chain in case of concurrent node extractions.

Definition at line 917 of file chainimpl.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/chainimpl.h

# 9.25 Chain_Node_struct Struct Reference

`#include <chain.h>`

## Public Attributes

- Chain_Node * next
- Chain_Node * previous

### 9.25.1 Detailed Description

This is used to manage each element (node) which is placed on a chain.

**Note**

Typically, a more complicated structure will use the chain package. The more complicated structure will include a chain node as the first element in its control structure. It will then call the chain package with a pointer to that node element. The node pointer and the higher level structure start at the same address so the user can cast the pointers back and forth.

Definition at line 68 of file chain.h.

### 9.25.2 Member Data Documentation

#### 9.25.2.1 next

```
Chain_Node* Chain_Node_struct::next
```

This points to the node after this one on this chain.

Definition at line 70 of file chain.h.

#### 9.25.2.2 previous

```
Chain_Node* Chain_Node_struct::previous
```

This points to the node immediate prior to this one on this chain.

Definition at line 72 of file chain.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/chain.h

## 9.26 Context_Control Struct Reference

SPARC basic context.

`#include <cpu.h>`

### Public Attributes

- uint32_t g5
- uint32_t g7
- double l0_and_l1
- uint32_t l2
- uint32_t l3
- uint32_t l4
- uint32_t l5
- uint32_t l6
- uint32_t l7
- uint32_t i0
- uint32_t i1
- uint32_t i2
- uint32_t i3
- uint32_t i4
- uint32_t i5
- uint32_t i6_fp
- uint32_t i7
- uint32_t o6_sp
- uint32_t o7
- uint32_t psr
- uint32_t isr_dispatch_disable
- volatile uint32_t **is_executing**

### 9.26.1 Detailed Description

SPARC basic context.

This structure defines the non-volatile integer and processor state context for the SPARC architecture according to "SYSTEM V APPLICATION BINARY INTERFACE - SPARC Processor Supplement", Third Edition.

The registers g2 through g4 are reserved for applications. GCC uses them as volatile registers by default. So they are treated like volatile registers in RTEMS as well.

The register g6 contains the per-CPU control of the current processor. It is an invariant of the processor context. This register must not be saved and restored during context switches or interrupt services.

Definition at line 318 of file cpu.h.

### 9.26.2 Member Data Documentation

#### 9.26.2.1 g5

```
uint32_t Context_Control::g5
```

This will contain the contents of the g5 register.

Definition at line 320 of file cpu.h.

#### 9.26.2.2 g7

```
uint32_t Context_Control::g7
```

This will contain the contents of the g7 register.

Definition at line 322 of file cpu.h.

#### 9.26.2.3 i0

```
uint32_t Context_Control::i0
```

This will contain the contents of the i0 register.

Definition at line 346 of file cpu.h.

#### 9.26.2.4 i1

```
uint32_t Context_Control::i1
```

This will contain the contents of the i1 register.

Definition at line 348 of file cpu.h.

#### 9.26.2.5 i2

```
uint32_t Context_Control::i2
```

This will contain the contents of the i2 register.

Definition at line 350 of file cpu.h.

#### 9.26.2.6 i3

```
uint32_t Context_Control::i3
```

This will contain the contents of the i3 register.

Definition at line 352 of file cpu.h.

#### 9.26.2.7 i4

```
uint32_t Context_Control::i4
```

This will contain the contents of the i4 register.

Definition at line 354 of file cpu.h.

#### 9.26.2.8 i5

```
uint32_t Context_Control::i5
```

This will contain the contents of the i5 register.

Definition at line 356 of file cpu.h.

#### 9.26.2.9 i6_fp

```
uint32_t Context_Control::i6_fp
```

This will contain the contents of the i6 (e.g. frame pointer) register.

Definition at line 358 of file cpu.h.

#### 9.26.2.10 i7

```
uint32_t Context_Control::i7
```

This will contain the contents of the i7 register.

Definition at line 360 of file cpu.h.

#### 9.26.2.11 isr_dispatch_disable

```
uint32_t Context_Control::isr_dispatch_disable
```

This field is used to prevent heavy nesting of calls to _Thread_Dispatch on an interrupted task's stack. This is problematic on the slower SPARC CPU models at high interrupt rates.

Definition at line 377 of file cpu.h.

#### 9.26.2.12 l0_and_l1

```
double Context_Control::l0_and_l1
```

This will contain the contents of the l0 and l1 registers.

Using a double l0_and_l1 will put everything in this structure on a double word boundary which allows us to use double word loads and stores safely in the context switch.

Definition at line 331 of file cpu.h.

#### 9.26.2.13 l2

```
uint32_t Context_Control::l2
```

This will contain the contents of the l2 register.

Definition at line 333 of file cpu.h.

#### 9.26.2.14 l3

```
uint32_t Context_Control::l3
```

This will contain the contents of the l3 register.

Definition at line 335 of file cpu.h.

#### 9.26.2.15 l4

```
uint32_t Context_Control::l4
```

This will contain the contents of the l4 register.

Definition at line 337 of file cpu.h.

#### 9.26.2.16 l5

```
uint32_t Context_Control::l5
```

This will contain the contents of the l5 registeer.

Definition at line 339 of file cpu.h.

#### 9.26.2.17 l6

```
uint32_t Context_Control::l6
```

This will contain the contents of the l6 register.

Definition at line 341 of file cpu.h.

#### 9.26.2.18 l7

```
uint32_t Context_Control::l7
```

This will contain the contents of the l7 register.

Definition at line 343 of file cpu.h.

#### 9.26.2.19 o6_sp

```
uint32_t Context_Control::o6_sp
```

This will contain the contents of the o6 (e.g. frame pointer) register.

Definition at line 363 of file cpu.h.

#### 9.26.2.20 o7

```
uint32_t Context_Control::o7
```

This will contain the contents of the o7 (e.g. address of CALL instruction) register.

Definition at line 368 of file cpu.h.

#### 9.26.2.21 psr

```
uint32_t Context_Control::psr
```

This will contain the contents of the processor status register.

Definition at line 371 of file cpu.h.

The documentation for this struct was generated from the following file:

- cpukit/score/cpu/sparc/include/rtems/score/cpu.h

## 9.27 Context_Control_fp Struct Reference

SPARC basic context.

```
#include <cpu.h>
```

### Public Attributes

- double f0_f1
- double f2_f3
- double f4_f5
- double f6_f7
- double f8_f9
- double f10_f11
- double f12_f13
- double f14_f15
- double f16_f17
- double f18_f19
- double f20_f21
- double f22_f23
- double f24_f25
- double f26_f27
- double f28_f29
- double f30_f31
- uint32_t fsr

### 9.27.1 Detailed Description

SPARC basic context.

This structure defines floating point context area.

Definition at line 478 of file cpu.h.

### 9.27.2 Member Data Documentation

#### 9.27.2.1 f0_f1

```
double Context_Control_fp::f0_f1
```

This will contain the contents of the f0 and f1 register.

Definition at line 480 of file cpu.h.

#### 9.27.2.2 f10_f11

```
double Context_Control_fp::f10_f11
```

This will contain the contents of the f10 and f11 register.

Definition at line 490 of file cpu.h.

#### 9.27.2.3 f12_f13

```
double Context_Control_fp::f12_f13
```

This will contain the contents of the f12 and f13 register.

Definition at line 492 of file cpu.h.

#### 9.27.2.4 f14_f15

```
double Context_Control_fp::f14_f15
```

This will contain the contents of the f14 and f15 register.

Definition at line 494 of file cpu.h.

#### 9.27.2.5 f16_f17

```
double Context_Control_fp::f16_f17
```

This will contain the contents of the f16 and f17 register.

Definition at line 496 of file cpu.h.

#### 9.27.2.6 f18_f19

```
double Context_Control_fp::f18_f19
```

This will contain the contents of the f18 and f19 register.

Definition at line 498 of file cpu.h.

#### 9.27.2.7 f20_f21

```
double Context_Control_fp::f20_f21
```

This will contain the contents of the f20 and f21 register.

Definition at line 500 of file cpu.h.

#### 9.27.2.8 f22_f23

```
double Context_Control_fp::f22_f23
```

This will contain the contents of the f22 and f23 register.

Definition at line 502 of file cpu.h.

#### 9.27.2.9 f24_f25

```
double Context_Control_fp::f24_f25
```

This will contain the contents of the f24 and f25 register.

Definition at line 504 of file cpu.h.

#### 9.27.2.10 f26_f27

```
double Context_Control_fp::f26_f27
```

This will contain the contents of the f26 and f27 register.

Definition at line 506 of file cpu.h.

#### 9.27.2.11 f28_f29

```
double Context_Control_fp::f28_f29
```

This will contain the contents of the f28 and f29 register.

Definition at line 508 of file cpu.h.

#### 9.27.2.12 f2_f3

```
double Context_Control_fp::f2_f3
```

This will contain the contents of the f2 and f3 register.

Definition at line 482 of file cpu.h.

#### 9.27.2.13 f30_f31

```
double Context_Control_fp::f30_f31
```

This will contain the contents of the f30 and f31 register.

Definition at line 510 of file cpu.h.

#### 9.27.2.14 f4_f5

```
double Context_Control_fp::f4_f5
```

This will contain the contents of the f4 and f5 register.

Definition at line 484 of file cpu.h.

#### 9.27.2.15 f6_f7

```
double Context_Control_fp::f6_f7
```

This will contain the contents of the f6 and f7 register.

Definition at line 486 of file cpu.h.

#### 9.27.2.16 f8_f9

```
double Context_Control_fp::f8_f9
```

This will contain the contents of the f8 and f9 register.

Definition at line 488 of file cpu.h.

#### 9.27.2.17 fsr

```
uint32_t Context_Control_fp::fsr
```

This will contain the contents of the floating point status register.

Definition at line 512 of file cpu.h.

The documentation for this struct was generated from the following file:

- cpukit/score/cpu/sparc/include/rtems/score/cpu.h

## 9.28 CORE_barrier_Attributes Struct Reference

```
#include <corebarrier.h>
```

### Public Attributes

- CORE_barrier_Disciplines discipline
- uint32_t maximum_count

### 9.28.1 Detailed Description

The following defines the control block used to manage the attributes of each barrier.

Definition at line 61 of file corebarrier.h.

### 9.28.2 Member Data Documentation

#### 9.28.2.1 discipline

```
CORE_barrier_Disciplines CORE_barrier_Attributes::discipline
```

This field indicates whether the barrier is automatic or manual.

Definition at line 64 of file corebarrier.h.

#### 9.28.2.2 maximum_count

```
uint32_t CORE_barrier_Attributes::maximum_count
```

This element indicates the number of threads which must arrive at the barrier to trip the automatic release.

Definition at line 68 of file corebarrier.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/corebarrier.h

## 9.29 CORE_barrier_Control Struct Reference

```
#include <corebarrier.h>
```

### Public Attributes

- Thread_queue_Control Wait_queue
- CORE_barrier_Attributes Attributes
- uint32_t number_of_waiting_threads

### 9.29.1 Detailed Description

The following defines the control block used to manage each barrier.

Definition at line 75 of file corebarrier.h.

### 9.29.2 Member Data Documentation

#### 9.29.2.1 Attributes

```
CORE_barrier_Attributes CORE_barrier_Control::Attributes
```

This element is the set of attributes which define this instance's behavior.

Definition at line 83 of file corebarrier.h.

#### 9.29.2.2 number_of_waiting_threads

```
uint32_t CORE_barrier_Control::number_of_waiting_threads
```

This element contains the current number of thread waiting for this barrier to be released.

Definition at line 86 of file corebarrier.h.

#### 9.29.2.3 Wait_queue

```
Thread_queue_Control CORE_barrier_Control::Wait_queue
```

This field is the Waiting Queue used to manage the set of tasks which are blocked waiting for the barrier to be released.

Definition at line 79 of file corebarrier.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/corebarrier.h

## 9.30 CORE_ceiling_mutex_Control Struct Reference

The recursive mutex control with priority ceiling protocol support.

```
#include <coremutex.h>
```

### Public Attributes

- CORE_recursive_mutex_Control Recursive
  *The plain recursive mutex.*
- Priority_Node Priority_ceiling
  *The priority ceiling node for the mutex owner.*
- const struct _Scheduler_Control * scheduler
  *The scheduler instance for this priority ceiling mutex.*

### 9.30.1 Detailed Description

The recursive mutex control with priority ceiling protocol support.

Definition at line 83 of file coremutex.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/coremutex.h

## 9.31 CORE_message_queue_Buffer Struct Reference

The structure is used to organize message buffers of a message queue.

`#include <coremsgbuffer.h>`

### Public Attributes

- Chain_Node Node

  *This member is used to enqueue the buffer in the pending or free buffer queue of a message queue.*
- size_t size

  *This member defines the size of this message.*
- int priority

  *This member defines the priority of this message.*
- size_t buffer [RTEMS_ZERO_LENGTH_ARRAY]

  *This member contains the actual message.*

### 9.31.1 Detailed Description

The structure is used to organize message buffers of a message queue.

Definition at line 63 of file coremsgbuffer.h.

### 9.31.2 Member Data Documentation

#### 9.31.2.1 buffer

`size_t CORE_message_queue_Buffer::buffer[RTEMS_ZERO_LENGTH_ARRAY]`

This member contains the actual message.

This is a zero-length array since the maximum message size is defined by the user. Use a size_t array to make sure that the member offset is at the structure end. This enables a more efficient memcpy() on 64-bit targets and makes it easier to inspect the message buffers with a debugger.

Definition at line 87 of file coremsgbuffer.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/coremsgbuffer.h

## 9.32 CORE_message_queue_Control Struct Reference

Control block used to manage each message queue.

`#include <coremsg.h>`

## Public Attributes

- Thread_queue_Control Wait_queue
- const Thread_queue_Operations * operations

  *The thread queue operations according to the blocking discipline.*
- uint32_t maximum_pending_messages
- uint32_t number_of_pending_messages
- size_t maximum_message_size
- Chain_Control Pending_messages
- CORE_message_queue_Buffer * message_buffers
- void(* free_message_buffers )(void *)

  *This member contains the optional message buffer storage area free handler.*
- Chain_Control Inactive_messages

### 9.32.1 Detailed Description

Control block used to manage each message queue.

The following defines the control block used to manage each Message Queue.

Definition at line 96 of file coremsg.h.

### 9.32.2 Member Data Documentation

#### 9.32.2.1 free_message_buffers

```
void( * CORE_message_queue_Control::free_message_buffers) (void *)
```

This member contains the optional message buffer storage area free handler.

It may be NULL. In this case no action is performed to free the message buffer storage area.

Definition at line 135 of file coremsg.h.

#### 9.32.2.2 Inactive_messages

```
Chain_Control CORE_message_queue_Control::Inactive_messages
```

This chain is the set of inactive messages. A message is inactive when it does not contain a pending message.

Definition at line 146 of file coremsg.h.

#### 9.32.2.3 maximum_message_size

```
size_t CORE_message_queue_Control::maximum_message_size
```

This is the size in bytes of the largest message which may be sent via this queue.

Definition at line 117 of file coremsg.h.

#### 9.32.2.4 maximum_pending_messages

```
uint32_t CORE_message_queue_Control::maximum_pending_messages
```

This element is maximum number of messages which may be pending at any given time.

Definition at line 110 of file coremsg.h.

#### 9.32.2.5 message_buffers

```
CORE_message_queue_Buffer* CORE_message_queue_Control::message_buffers
```

This is the address of the memory allocated for message buffers. It is allocated are part of message queue initialization and freed as part of destroying it.

Definition at line 126 of file coremsg.h.

#### 9.32.2.6 number_of_pending_messages

```
uint32_t CORE_message_queue_Control::number_of_pending_messages
```

This element is the number of messages which are currently pending.

Definition at line 113 of file coremsg.h.

#### 9.32.2.7 Pending_messages

```
Chain_Control CORE_message_queue_Control::Pending_messages
```

This chain is the set of pending messages. It may be ordered by message priority or in FIFO order.

Definition at line 121 of file coremsg.h.

#### 9.32.2.8 Wait_queue

```
Thread_queue_Control CORE_message_queue_Control::Wait_queue
```

This field is the Waiting Queue used to manage the set of tasks which are blocked waiting to receive a message from this queue.

Definition at line 100 of file coremsg.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/coremsg.h

## 9.33 CORE_mutex_Control Struct Reference

Control block used to manage each mutex.

```
#include <coremutex.h>
```

### Public Attributes

- Thread_queue_Control Wait_queue
  *The thread queue of this mutex.*

### 9.33.1 Detailed Description

Control block used to manage each mutex.

The following defines the control block used to manage each mutex.

Definition at line 56 of file coremutex.h.

### 9.33.2 Member Data Documentation

#### 9.33.2.1 Wait_queue

```
Thread_queue_Control CORE_mutex_Control::Wait_queue
```

The thread queue of this mutex.

The owner of the thread queue indicates the mutex owner.

Definition at line 62 of file coremutex.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/coremutex.h

## 9.34 CORE_recursive_mutex_Control Struct Reference

The recursive mutex control.

`#include <coremutex.h>`

### Public Attributes

- CORE_mutex_Control Mutex
  *The plain non-recursive mutex.*
- unsigned int nest_level
  *The nest level in case of a recursive seize.*

### 9.34.1 Detailed Description

The recursive mutex control.

Definition at line 68 of file coremutex.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/coremutex.h

## 9.35 CORE_semaphore_Control Struct Reference

`#include <coresem.h>`

### Public Attributes

- Thread_queue_Control Wait_queue
- uint32_t count

### 9.35.1 Detailed Description

The following defines the control block used to manage each counting semaphore.

Definition at line 48 of file coresem.h.

### 9.35.2 Member Data Documentation

#### 9.35.2.1 count

```
uint32_t CORE_semaphore_Control::count
```

This element contains the current count of this semaphore.

Definition at line 55 of file coresem.h.

#### 9.35.2.2 Wait_queue

```
Thread_queue_Control CORE_semaphore_Control::Wait_queue
```

This field is the Waiting Queue used to manage the set of tasks which are blocked waiting to obtain the semaphore.

Definition at line 52 of file coresem.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/coresem.h

## 9.36 CPU_Exception_frame Struct Reference

### Public Attributes

- uint32_t **trap**
- CPU_Interrupt_frame * **isf**

### 9.36.1 Detailed Description

Definition at line 1026 of file cpu.h.

The documentation for this struct was generated from the following file:

- cpukit/score/cpu/sparc/include/rtems/score/cpu.h

## 9.37 CPU_Interrupt_frame Struct Reference

Interrupt stack frame (ISF).

```
#include <cpu.h>
```

## Public Attributes

- SPARC_Minimum_stack_frame Stack_frame
- uint32_t psr
- uint32_t pc
- uint32_t npc
- uint32_t g1
- uint32_t g2
- uint32_t g3
- uint32_t g4
- uint32_t g5
- uint32_t reserved_for_alignment
- uint32_t g7
- uint32_t i0
- uint32_t i1
- uint32_t i2
- uint32_t i3
- uint32_t i4
- uint32_t i5
- uint32_t i6_fp
- uint32_t i7
- uint32_t y
- uint32_t tpc

### 9.37.1 Detailed Description

Interrupt stack frame (ISF).

Context saved on stack for an interrupt.

NOTE: The PSR, PC, and NPC are only saved in this structure for the benefit of the user's handler.

Definition at line 571 of file cpu.h.

### 9.37.2 Member Data Documentation

#### 9.37.2.1 g1

```
uint32_t CPU_Interrupt_frame::g1
```

This is the offset of the g1 register on an ISF.

Definition at line 581 of file cpu.h.

#### 9.37.2.2 g2

```
uint32_t CPU_Interrupt_frame::g2
```

This is the offset of the g2 register on an ISF.

Definition at line 583 of file cpu.h.

#### 9.37.2.3 g3

```
uint32_t CPU_Interrupt_frame::g3
```

This is the offset of the g3 register on an ISF.

Definition at line 585 of file cpu.h.

#### 9.37.2.4 g4

```
uint32_t CPU_Interrupt_frame::g4
```

This is the offset of the g4 register on an ISF.

Definition at line 587 of file cpu.h.

#### 9.37.2.5 g5

```
uint32_t CPU_Interrupt_frame::g5
```

This is the offset of the g5 register on an ISF.

Definition at line 589 of file cpu.h.

#### 9.37.2.6 g7

```
uint32_t CPU_Interrupt_frame::g7
```

This is the offset of the g7 register on an ISF.

Definition at line 593 of file cpu.h.

#### 9.37.2.7 i0

```
uint32_t CPU_Interrupt_frame::i0
```

This is the offset of the i0 register on an ISF.

Definition at line 595 of file cpu.h.

#### 9.37.2.8 i1

```
uint32_t CPU_Interrupt_frame::i1
```

This is the offset of the i1 register on an ISF.

Definition at line 597 of file cpu.h.

#### 9.37.2.9 i2

```
uint32_t CPU_Interrupt_frame::i2
```

This is the offset of the i2 register on an ISF.

Definition at line 599 of file cpu.h.

#### 9.37.2.10 i3

```
uint32_t CPU_Interrupt_frame::i3
```

This is the offset of the i3 register on an ISF.

Definition at line 601 of file cpu.h.

#### 9.37.2.11 i4

```
uint32_t CPU_Interrupt_frame::i4
```

This is the offset of the i4 register on an ISF.

Definition at line 603 of file cpu.h.

#### 9.37.2.12 i5

```
uint32_t CPU_Interrupt_frame::i5
```

This is the offset of the i5 register on an ISF.

Definition at line 605 of file cpu.h.

#### 9.37.2.13 i6_fp

```
uint32_t CPU_Interrupt_frame::i6_fp
```

This is the offset of the i6 register on an ISF.

Definition at line 607 of file cpu.h.

#### 9.37.2.14 i7

```
uint32_t CPU_Interrupt_frame::i7
```

This is the offset of the i7 register on an ISF.

Definition at line 609 of file cpu.h.

#### 9.37.2.15 npc

```
uint32_t CPU_Interrupt_frame::npc
```

This is the offset of the XXX on an ISF.

Definition at line 579 of file cpu.h.

#### 9.37.2.16 pc

```
uint32_t CPU_Interrupt_frame::pc
```

This is the offset of the XXX on an ISF.

Definition at line 577 of file cpu.h.

#### 9.37.2.17 psr

```
uint32_t CPU_Interrupt_frame::psr
```

This is the offset of the PSR on an ISF.

Definition at line 575 of file cpu.h.

#### 9.37.2.18 reserved_for_alignment

```
uint32_t CPU_Interrupt_frame::reserved_for_alignment
```

This is the offset is reserved for alignment on an ISF.

Definition at line 591 of file cpu.h.

#### 9.37.2.19 Stack_frame

```
SPARC_Minimum_stack_frame CPU_Interrupt_frame::Stack_frame
```

On an interrupt, we must save the minimum stack frame.

Definition at line 573 of file cpu.h.

#### 9.37.2.20 tpc

```
uint32_t CPU_Interrupt_frame::tpc
```

This is the offset of the tpc register on an ISF.

Definition at line 613 of file cpu.h.

#### 9.37.2.21 y

```
uint32_t CPU_Interrupt_frame::y
```

This is the offset of the y register on an ISF.

Definition at line 611 of file cpu.h.

The documentation for this struct was generated from the following file:

- cpukit/score/cpu/sparc/include/rtems/score/cpu.h

## 9.38 CPU_Per_CPU_control Struct Reference

### 9.38.1 Detailed Description

Definition at line 126 of file cpuimpl.h.

The documentation for this struct was generated from the following file:

- cpukit/score/cpu/sparc/include/rtems/score/cpuimpl.h

## 9.39 CPU_Trap_table_entry Struct Reference

`#include <cpu.h>`

### Public Attributes

- uint32_t mov_psr_l0
- uint32_t sethi_of_handler_to_l4
- uint32_t jmp_to_low_of_handler_plus_l4
- uint32_t mov_vector_l3

### 9.39.1 Detailed Description

The following type defines an entry in the SPARC's trap table.

NOTE: The instructions chosen are RTEMS dependent although one is obligated to use two of the four instructions to perform a long jump. The other instructions load one register with the trap type (a.k.a. vector) and another with the psr.

Definition at line 627 of file cpu.h.

### 9.39.2 Member Data Documentation

#### 9.39.2.1 jmp_to_low_of_handler_plus_l4

`uint32_t CPU_Trap_table_entry::jmp_to_low_of_handler_plus_l4`

This will contain a "jmp %l4 + %lo(_handler)" instruction.

Definition at line 633 of file cpu.h.

#### 9.39.2.2 mov_psr_l0

`uint32_t CPU_Trap_table_entry::mov_psr_l0`

This will contain a "mov %psr, %l0" instruction.

Definition at line 629 of file cpu.h.

#### 9.39.2.3 mov_vector_l3

`uint32_t CPU_Trap_table_entry::mov_vector_l3`

This will contain a " mov _vector, %l3" instruction.

Definition at line 635 of file cpu.h.

#### 9.39.2.4 sethi_of_handler_to_l4

`uint32_t CPU_Trap_table_entry::sethi_of_handler_to_l4`

This will contain a "sethi %hi(_handler), %l4" instruction.

Definition at line 631 of file cpu.h.

The documentation for this struct was generated from the following file:

- cpukit/score/cpu/sparc/include/rtems/score/cpu.h

## 9.40 Dual_ported_memory_Control Struct Reference

`#include <dpmemdata.h>`

### Public Attributes

- Objects_Control Object
- void * internal_base
- void * external_base
- uint32_t length

### 9.40.1 Detailed Description

The following structure defines the port control block. Each port has a control block associated with it. This control block contains all information required to support the port related operations.

Definition at line 38 of file dpmemdata.h.

## 9.40.2 Member Data Documentation

### 9.40.2.1 external_base

```
void* Dual_ported_memory_Control::external_base
```

This field is the base external address of the port.

Definition at line 44 of file dpmemdata.h.

### 9.40.2.2 internal_base

```
void* Dual_ported_memory_Control::internal_base
```

This field is the base internal address of the port.

Definition at line 42 of file dpmemdata.h.

### 9.40.2.3 length

```
uint32_t Dual_ported_memory_Control::length
```

This field is the length of dual-ported area of the port.

Definition at line 46 of file dpmemdata.h.

### 9.40.2.4 Object

```
Objects_Control Dual_ported_memory_Control::Object
```

This field is the object management portion of a Port instance.

Definition at line 40 of file dpmemdata.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/dpmemdata.h

## 9.41 Event_Control Struct Reference

This structure is used to manage a set of events.

`#include <eventdata.h>`

### Public Attributes

- rtems_event_set pending_events
  *The member contains the pending events.*

### 9.41.1 Detailed Description

This structure is used to manage a set of events.

Definition at line 36 of file eventdata.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/eventdata.h

## 9.42 Extension_Control Struct Reference

### Public Attributes

- Objects_Control **Object**
- User_extensions_Control **Extension**

### 9.42.1 Detailed Description

Definition at line 35 of file extensiondata.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/extensiondata.h

## 9.43 ffclock_estimate Struct Reference

### Public Attributes

- struct bintime **update_time**
- ffcounter **update_ffcount**
- ffcounter **leapsec_next**
- uint64_t **period**
- uint32_t **errb_abs**
- uint32_t **errb_rate**
- uint32_t **status**
- int16_t **leapsec_total**
- int8_t **leapsec**

### 9.43.1 Detailed Description

Definition at line 43 of file timeffc.h.

The documentation for this struct was generated from the following file:

- cpukit/include/sys/timeffc.h

## 9.44 Freechain_Control Struct Reference

The freechain control.

`#include <freechain.h>`

### Public Attributes

- Chain_Control Free
  *Chain of free nodes.*

### 9.44.1 Detailed Description

The freechain control.

Definition at line 42 of file freechain.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/freechain.h

## 9.45 gptimer_regs Struct Reference

### Public Attributes

- volatile unsigned int **scaler_value**
- volatile unsigned int **scaler_reload**
- volatile unsigned int **cfg**
- volatile unsigned int **notused**
- struct gptimer_timer_regs **timer** [7]

### 9.45.1 Detailed Description

Definition at line 110 of file grlib.h.

The documentation for this struct was generated from the following file:

- bsps/include/grlib/grlib.h

## 9.46 gptimer_timer_regs Struct Reference

### Public Attributes

- volatile unsigned int **value**
- volatile unsigned int **reload**
- volatile unsigned int **ctrl**
- volatile unsigned int **notused**

### 9.46.1 Detailed Description

Definition at line 94 of file grlib.h.

The documentation for this struct was generated from the following file:

- bsps/include/grlib/grlib.h

## 9.47 grgpio_regs Struct Reference

### Public Attributes

- volatile unsigned int **data**
- volatile unsigned int **output**
- volatile unsigned int **dir**
- volatile unsigned int **imask**
- volatile unsigned int **ipol**
- volatile unsigned int **iedge**
- volatile unsigned int **bypass**
- volatile unsigned int **cap**
- volatile unsigned int **irqmap** [4]
- volatile unsigned int **res_30**
- volatile unsigned int **res_34**
- volatile unsigned int **res_38**
- volatile unsigned int **res_3C**
- volatile unsigned int **iavail**
- volatile unsigned int **iflag**
- volatile unsigned int **res_48**
- volatile unsigned int **pulse**
- volatile unsigned int **res_50**
- volatile unsigned int **output_or**
- volatile unsigned int **dir_or**
- volatile unsigned int **imask_or**
- volatile unsigned int **res_60**
- volatile unsigned int **output_and**
- volatile unsigned int **dir_and**
- volatile unsigned int **imask_and**
- volatile unsigned int **res_70**
- volatile unsigned int **output_xor**
- volatile unsigned int **dir_xor**
- volatile unsigned int **imask_xor**

### 9.47.1 Detailed Description

Definition at line 119 of file grlib.h.

The documentation for this struct was generated from the following file:

- bsps/include/grlib/grlib.h

## 9.48 Heap_Area Struct Reference

Heap area structure for table based heap initialization and extension.

`#include <heap.h>`

### Public Attributes

- void * **begin**
- uintptr_t **size**

### 9.48.1 Detailed Description

Heap area structure for table based heap initialization and extension.

**See also**

Heap_Initialization_or_extend_handler.

Definition at line 338 of file heap.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/heap.h

## 9.49 Heap_Block Struct Reference

Description for free or used blocks.

`#include <heap.h>`

### Public Attributes

- uintptr_t prev_size

  *Size of the previous block or part of the allocated area of the previous block.*
- uintptr_t size_and_flag

  *Contains the size of the current block and a flag which indicates if the previous block is free or used.*
- Heap_Block * next

  *Pointer to the next free block or part of the allocated area.*
- Heap_Block * prev

  *Pointer to the previous free block or part of the allocated area.*

### 9.49.1 Detailed Description

Description for free or used blocks.

Definition at line 256 of file heap.h.

### 9.49.2 Member Data Documentation

#### 9.49.2.1 next

```
Heap_Block* Heap_Block::next
```

Pointer to the next free block or part of the allocated area.

This field is page size aligned and begins of the allocated area in case the block is used.

This field is only valid if the block is free and thus part of the free block list.

Definition at line 304 of file heap.h.

#### 9.49.2.2 prev

```
Heap_Block* Heap_Block::prev
```

Pointer to the previous free block or part of the allocated area.

This field is only valid if the block is free and thus part of the free block list.

Definition at line 312 of file heap.h.

#### 9.49.2.3 prev_size

```
uintptr_t Heap_Block::prev_size
```

Size of the previous block or part of the allocated area of the previous block.

This field is only valid if the previous block is free. This case is indicated by a cleared `HEAP_PREV_BLOCK_USED` flag in the *size_and_flag* field of the current block.

In a used block only the *size_and_flag* field needs to be valid. The *prev_size* field of the current block is maintained by the previous block. The current block can use the *prev_size* field in the next block for allocation.

Definition at line 270 of file heap.h.

#### 9.49.2.4 size_and_flag

```
uintptr_t Heap_Block::size_and_flag
```

Contains the size of the current block and a flag which indicates if the previous block is free or used.

If the flag `HEAP_PREV_BLOCK_USED` is set, then the previous block is used, otherwise the previous block is free. A used previous block may claim the *prev_size* field for allocation. This trick allows to decrease the overhead in the used blocks by the size of the *prev_size* field. As sizes are required to be multiples of two, the least significant bits would be always zero. We use this bit to store the flag.

This field is always valid.

Definition at line 289 of file heap.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/heap.h

## 9.50 Heap_Control Struct Reference

Control block used to manage a heap.

```
#include <heap.h>
```

### Public Attributes

- Heap_Block **free_list**
- uintptr_t **page_size**
- uintptr_t **min_block_size**
- uintptr_t **area_begin**
- uintptr_t **area_end**
- Heap_Block * **first_block**
- Heap_Block * **last_block**
- Heap_Statistics **stats**

### 9.50.1 Detailed Description

Control block used to manage a heap.

Definition at line 318 of file heap.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/heap.h

## 9.51 Heap_Error_context Struct Reference

Context of a heap error.

`#include <heap.h>`

### Public Attributes

- Heap_Control * heap
  *The heap of the block.*
- Heap_Block * block
  *The heap block causing the error.*
- Heap_Error_reason reason
  *The heap error reason.*

### 9.51.1 Detailed Description

Context of a heap error.

**See also**

_Heap_Protection_block_error().

Definition at line 176 of file heap.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/heap.h

## 9.52 Heap_Information Struct Reference

Information about blocks.

`#include <heapinfo.h>`

### Public Attributes

- uintptr_t number
  *Number of blocks of this type.*
- uintptr_t largest
  *Largest block of this type.*
- uintptr_t total
  *Total size of the blocks of this type.*

### 9.52.1 Detailed Description

Information about blocks.

Definition at line 125 of file heapinfo.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/heapinfo.h

## 9.53 Heap_Information_block Struct Reference

Information block returned by _Heap_Get_information().

```
#include <heapinfo.h>
```

### Public Attributes

- Heap_Information **Free**
- Heap_Information **Used**
- Heap_Statistics **Stats**

### 9.53.1 Detailed Description

Information block returned by _Heap_Get_information().

Definition at line 145 of file heapinfo.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/heapinfo.h

## 9.54 Heap_Statistics Struct Reference

Run-time heap statistics.

```
#include <heapinfo.h>
```

## Public Attributes

- uint64_t lifetime_allocated

  *Lifetime number of bytes allocated from this heap.*
- uint64_t lifetime_freed

  *Lifetime number of bytes freed to this heap.*
- uintptr_t size

  *Size of the allocatable area in bytes.*
- uintptr_t free_size

  *Current free size in bytes.*
- uintptr_t min_free_size

  *Minimum free size ever in bytes.*
- uint32_t free_blocks

  *Current number of free blocks.*
- uint32_t max_free_blocks

  *Maximum number of free blocks ever.*
- uint32_t used_blocks

  *Current number of used blocks.*
- uint32_t max_search

  *Maximum number of blocks searched ever.*
- uint32_t searches

  *Total number of searches.*
- uint32_t allocs

  *Total number of successful allocations.*
- uint32_t failed_allocs

  *Total number of failed allocations.*
- uint32_t frees

  *Total number of successful frees.*
- uint32_t resizes

  *Total number of successful resizes.*

### 9.54.1 Detailed Description

Run-time heap statistics.

The value *searches* / *allocs* gives the mean number of searches per allocation, while *max_search* gives maximum number of searches ever performed on a single allocation call.

Definition at line 40 of file heapinfo.h.

### 9.54.2 Member Data Documentation

#### 9.54.2.1 free_size

```
uintptr_t Heap_Statistics::free_size
```

Current free size in bytes.

This value is an integral multiple of the page size.

Definition at line 67 of file heapinfo.h.

#### 9.54.2.2 lifetime_allocated

```
uint64_t Heap_Statistics::lifetime_allocated
```

Lifetime number of bytes allocated from this heap.

This value is an integral multiple of the page size.

Definition at line 46 of file heapinfo.h.

#### 9.54.2.3 lifetime_freed

```
uint64_t Heap_Statistics::lifetime_freed
```

Lifetime number of bytes freed to this heap.

This value is an integral multiple of the page size.

Definition at line 53 of file heapinfo.h.

#### 9.54.2.4 min_free_size

```
uintptr_t Heap_Statistics::min_free_size
```

Minimum free size ever in bytes.

This value is an integral multiple of the page size.

Definition at line 74 of file heapinfo.h.

#### 9.54.2.5 size

```
uintptr_t Heap_Statistics::size
```

Size of the allocatable area in bytes.

This value is an integral multiple of the page size.

Definition at line 60 of file heapinfo.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/heapinfo.h

## 9.55 Internal_errors_Information Struct Reference

```
#include <interr.h>
```

### Public Attributes

- Internal_errors_Source the_source
- Internal_errors_t the_error

### 9.55.1 Detailed Description

This type holds the fatal error information.

Definition at line 215 of file interr.h.

### 9.55.2 Member Data Documentation

#### 9.55.2.1 the_error

```
Internal_errors_t Internal_errors_Information::the_error
```

This is the error code.

Definition at line 219 of file interr.h.

#### 9.55.2.2 the_source

```
Internal_errors_Source Internal_errors_Information::the_source
```

This is the source of the error.

Definition at line 217 of file interr.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/interr.h

## 9.56 irqmp_regs Struct Reference

### Public Attributes

- volatile unsigned int **ilevel**
- volatile unsigned int **ipend**
- volatile unsigned int **iforce**
- volatile unsigned int **iclear**
- volatile unsigned int **mpstat**
- volatile unsigned int **bcast**
- volatile unsigned int **notused02**
- volatile unsigned int **wdgctrl**
- volatile unsigned int **ampctrl**
- volatile unsigned int **icsel** [2]
- volatile unsigned int **notused13**
- volatile unsigned int **notused20**
- volatile unsigned int **notused21**
- volatile unsigned int **notused22**
- volatile unsigned int **notused23**
- volatile unsigned int **mask** [16]
- volatile unsigned int **force** [16]
- volatile unsigned int **intid** [16]
- volatile struct irqmp_timestamp_regs **timestamp** [16]
- volatile unsigned int **resetaddr** [4]
- volatile unsigned int **resv0** [12]
- volatile unsigned int **pboot**
- volatile unsigned int **resv1** [47]
- volatile unsigned int **irqmap** [8]
- volatile unsigned int **resv2** [824]

### 9.56.1 Detailed Description

Definition at line 64 of file grlib.h.

The documentation for this struct was generated from the following file:

- bsps/include/grlib/grlib.h

## 9.57 irqmp_timestamp_regs Struct Reference

### Public Attributes

- volatile unsigned int **counter**
- volatile unsigned int **control**
- volatile unsigned int **assertion**
- volatile unsigned int **ack**

### 9.57.1 Detailed Description

Definition at line 49 of file grlib.h.

The documentation for this struct was generated from the following file:

- bsps/include/grlib/grlib.h

## 9.58 ISR_lock_Context Struct Reference

Local ISR lock context for acquire and release pairs.

`#include <isrlock.h>`

### Public Attributes

- SMP_lock_Context **Lock_context**

### 9.58.1 Detailed Description

Local ISR lock context for acquire and release pairs.

Definition at line 65 of file isrlock.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/isrlock.h

## 9.59 ISR_lock_Control Struct Reference

ISR lock control.

`#include <isrlock.h>`

### Public Attributes

- SMP_lock_Control **Lock**

### 9.59.1 Detailed Description

ISR lock control.

**Warning**

Empty structures are implementation-defined in C. GCC gives them a size of zero. In C++ empty structures have a non-zero size.

Definition at line 56 of file isrlock.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/isrlock.h

## 9.60 I2c_regs Struct Reference

### Public Attributes

- volatile unsigned int **control**
- volatile unsigned int **status**
- volatile unsigned int **flush_mem_addr**
- volatile unsigned int **flush_set_index**
- volatile unsigned int **access_counter**
- volatile unsigned int **hit_counter**
- volatile unsigned int **bus_cycle_counter**
- volatile unsigned int **bus_usage_counter**
- volatile unsigned int **error_status_control**
- volatile unsigned int **error_addr**
- volatile unsigned int **tag_check_bit**
- volatile unsigned int **data_check_bit**
- volatile unsigned int **scrub_control_status**
- volatile unsigned int **scrub_delay**
- volatile unsigned int **error_injection**
- volatile unsigned int **access_control**
- volatile unsigned int **reserved_40** [16]
- volatile unsigned int **mtrr** [32]
- volatile unsigned int **reserved_100** [131008]
- volatile unsigned int **diag_iface_tag** [16384]
- volatile unsigned int **reserved_90000** [376832]
- volatile unsigned int **diag_iface_data** [524288]

### 9.60.1 Detailed Description

Definition at line 152 of file grlib.h.

The documentation for this struct was generated from the following file:

- bsps/include/grlib/grlib.h

## 9.61 load_context Struct Reference

### Public Attributes

- struct T_measure_runtime_context * **master**
- rtems_id **id**
- volatile unsigned int * **chunk**

### 9.61.1 Detailed Description

Definition at line 39 of file t-test-rtems-measure.c.

The documentation for this struct was generated from the following file:

- cpukit/libtest/t-test-rtems-measure.c

## 9.62 mctrl_regs Struct Reference

### Public Attributes

- unsigned int **mcfg1**
- unsigned int **mcfg2**
- unsigned int **mcfg3**

### 9.62.1 Detailed Description

Definition at line 34 of file grlib.h.

The documentation for this struct was generated from the following file:

- bsps/include/grlib/grlib.h

## 9.63 Memory_Area Struct Reference

The memory area description.

`#include <memory.h>`

### Public Attributes

- const void * begin

  *A pointer to the begin of the memory area.*
- void * free

  *A pointer to the begin of the free area of the memory area.*
- const void * end

  *A pointer to the end of the memory area.*

### 9.63.1 Detailed Description

The memory area description.

Definition at line 60 of file memory.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/memory.h

## 9.64 Memory_Information Struct Reference

The memory information.

`#include <memory.h>`

### Public Attributes

- size_t count

  *The count of memory areas.*
- Memory_Area * areas

  *The memory area table.*

### 9.64.1 Detailed Description

The memory information.

Definition at line 80 of file memory.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/memory.h

## 9.65 Message_queue_Control Struct Reference

`#include <messagedata.h>`

### Public Attributes

- Objects_Control Object
- CORE_message_queue_Control message_queue

## 9.65.1 Detailed Description

The following records define the control block used to manage each message queue.

Definition at line 38 of file messagedata.h.

## 9.65.2 Member Data Documentation

### 9.65.2.1 message_queue

```
CORE_message_queue_Control Message_queue_Control::message_queue
```

This field is the instance of the SuperCore Message Queue.

Definition at line 42 of file messagedata.h.

### 9.65.2.2 Object

```
Objects_Control Message_queue_Control::Object
```

This field is the inherited object characteristics.

Definition at line 40 of file messagedata.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/messagedata.h

# 9.66 MP_packet_Prefix Struct Reference

```
#include <mppkt.h>
```

## Public Attributes

- MP_packet_Classes the_class
- Objects_Id id
- Objects_Id source_tid
- uint32_t source_priority
- uint32_t return_code
- uint32_t length
- uint32_t to_convert
- Watchdog_Interval timeout

### 9.66.1 Detailed Description

The following record contains the prefix for every packet passed between nodes in an MP system.

**Note**

This structure is padded to ensure that anything following it is on a 16 byte boundary. This is the most stringent structure alignment rule encountered yet.

Definition at line 85 of file mppkt.h.

### 9.66.2 Member Data Documentation

#### 9.66.2.1 id

```
Objects_Id MP_packet_Prefix::id
```

This field is the id of the object to be acted upon.

Definition at line 89 of file mppkt.h.

#### 9.66.2.2 length

```
uint32_t MP_packet_Prefix::length
```

This field is the length of the data following the prefix.

Definition at line 97 of file mppkt.h.

#### 9.66.2.3 return_code

```
uint32_t MP_packet_Prefix::return_code
```

This field is where the status of the operation will be returned.

Definition at line 95 of file mppkt.h.

#### 9.66.2.4 source_priority

```
uint32_t MP_packet_Prefix::source_priority
```

This field is the priority of the originating thread.

Definition at line 93 of file mppkt.h.

#### 9.66.2.5 source_tid

```
Objects_Id MP_packet_Prefix::source_tid
```

This field is the ID of the originating thread.

Definition at line 91 of file mppkt.h.

#### 9.66.2.6 the_class

```
MP_packet_Classes MP_packet_Prefix::the_class
```

This field indicates the API class of the operation being performed.

Definition at line 87 of file mppkt.h.

#### 9.66.2.7 timeout

```
Watchdog_Interval MP_packet_Prefix::timeout
```

This field is the requested timeout for this operation.

Definition at line 101 of file mppkt.h.

#### 9.66.2.8 to_convert

```
uint32_t MP_packet_Prefix::to_convert
```

This field is the length of the data which required network conversion.

Definition at line 99 of file mppkt.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/mppkt.h

## 9.67 MRSP_Control Struct Reference

MrsP control block.

```
#include <mrsp.h>
```

### Public Attributes

- Thread_queue_Control Wait_queue

  *The thread queue to manage ownership and waiting threads.*
- Priority_Node Ceiling_priority

  *The ceiling priority used by the owner thread.*
- Priority_Control ceiling_priorities [RTEMS_ZERO_LENGTH_ARRAY]

  *One ceiling priority per scheduler instance.*

### 9.67.1 Detailed Description

MrsP control block.

Definition at line 62 of file mrsp.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/mrsp.h

## 9.68 Mutex_Control Struct Reference

### Public Attributes

- Thread_queue_Syslock_queue **Queue**

### 9.68.1 Detailed Description

Definition at line 38 of file muteximpl.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/muteximpl.h

## 9.69 Mutex_recursive_Control Struct Reference

### Public Attributes

- Mutex_Control **Mutex**
- unsigned int **nest_level**

### 9.69.1 Detailed Description

Definition at line 42 of file muteximpl.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/muteximpl.h

## 9.70 ntp_fp Struct Reference

### Public Attributes

- unsigned int **integral**
- unsigned int **fractional**

### 9.70.1 Detailed Description

Definition at line 34 of file timepps.h.

The documentation for this struct was generated from the following file:

- cpukit/include/sys/timepps.h

## 9.71 ntptimeval Struct Reference

### Public Attributes

- struct timespec **time**
- long **maxerror**
- long **esterror**
- long **tai**
- int **time_state**

### 9.71.1 Detailed Description

Definition at line 114 of file timex.h.

The documentation for this struct was generated from the following file:

- cpukit/include/sys/timex.h

## 9.72 Objects_Control Struct Reference

`#include <objectdata.h>`

### Public Attributes

- Chain_Node Node
- Objects_Id id
- Objects_Name name

### 9.72.1 Detailed Description

The following defines the Object Control Block used to manage each object local to this node.

Definition at line 39 of file objectdata.h.

### 9.72.2 Member Data Documentation

#### 9.72.2.1 id

`Objects_Id Objects_Control::id`

This is the object's ID.

Definition at line 43 of file objectdata.h.

#### 9.72.2.2 name

`Objects_Name Objects_Control::name`

This is the object's name.

Definition at line 45 of file objectdata.h.

#### 9.72.2.3 Node

```
Chain_Node Objects_Control::Node
```

This is the chain node portion of an object.

Definition at line 41 of file objectdata.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/objectdata.h

## 9.73 Objects_Information Struct Reference

The information structure used to manage each API class of objects.

```
#include <objectdata.h>
```

### Public Attributes

- Objects_Id maximum_id

  *This is the maximum valid ID of this object API class.*
- Objects_Control ** local_table

  *This points to the table of local object control blocks.*
- Objects_Control *(* allocate )(Objects_Information *)

  *Allocate an object.*
- void(* deallocate )(Objects_Information *, Objects_Control *)

  *Free an object.*
- Objects_Maximum inactive

  *This is the number of object control blocks on the inactive chain.*
- Objects_Maximum objects_per_block

  *This is the number of object control blocks in an allocation block.*
- uint16_t object_size

  *This is the size in bytes of each object control block.*
- uint16_t name_length

  *This is the maximum length of names.*
- Chain_Control Inactive

  *This is the chain of inactive object control blocks.*
- Objects_Maximum * inactive_per_block

  *This is the number of inactive object control blocks per allocation block.*
- Objects_Control ** object_blocks

  *This is a table to allocation blocks of object control blocks.*
- Objects_Control * initial_objects

  *This points to the object control blocks initially available.*

### 9.73.1 Detailed Description

The information structure used to manage each API class of objects.

If objects for the API class are configured, an instance of this structure is statically allocated and pre-initialized by OBJECTS_INFORMATION_DEFINE() through <rtems/confdefs.h>. The RTEMS library contains a statically allocated and pre-initialized instance for each API class providing zero objects, see OBJECTS_INFORMATION_DEFINE_ZERO().

Definition at line 176 of file objectdata.h.

### 9.73.2 Member Data Documentation

#### 9.73.2.1 allocate

```
Objects_Control*( * Objects_Information::allocate) (Objects_Information *)
```

Allocate an object.

**See also**

_Objects_Allocate_none(), _Objects_Allocate_static(), and _Objects_Allocate_unlimited().

Definition at line 203 of file objectdata.h.

#### 9.73.2.2 deallocate

```
void( * Objects_Information::deallocate) (Objects_Information *, Objects_Control *)
```

Free an object.

In case _Objects_Allocate_none() is used, then this may be the NULL pointer.

**See also**

_Objects_Free_static(), and _Objects_Free_unlimited().

Definition at line 213 of file objectdata.h.

#### 9.73.2.3 inactive

```
Objects_Maximum Objects_Information::inactive
```

This is the number of object control blocks on the inactive chain.

This member is only used if unlimited objects are configured for this API class. It is used to trigger calls to _Objects_Shrink_information() in _Objects_Free().

Definition at line 222 of file objectdata.h.

#### 9.73.2.4 Inactive

```
Chain_Control Objects_Information::Inactive
```

This is the chain of inactive object control blocks.

This member is statically initialized to an empty chain. The _Objects_Initialize_information() will populate this chain with the object control blocks initially configured.

Definition at line 256 of file objectdata.h.

#### 9.73.2.5 inactive_per_block

```
Objects_Maximum* Objects_Information::inactive_per_block
```

This is the number of inactive object control blocks per allocation block.

It is only used if unlimited objects are configured for this API class.

Definition at line 264 of file objectdata.h.

#### 9.73.2.6 initial_objects

```
Objects_Control* Objects_Information::initial_objects
```

This points to the object control blocks initially available.

This member is statically initialized and read-only. In case objects for this API class are configured, it points to a statically allocated table of object control blocks defined by <rtems/confdefs.h>, otherwise this member is NULL.

Definition at line 282 of file objectdata.h.

#### 9.73.2.7 local_table

```
Objects_Control** Objects_Information::local_table
```

This points to the table of local object control blocks.

This member is statically initialized. In case objects for this API class are configured, it initially points to a statically allocated table defined by <rtems/confdefs.h>. _Objects_Extend_information() may replace the table with a larger one on demand.

Definition at line 195 of file objectdata.h.

#### 9.73.2.8 maximum_id

```
Objects_Id Objects_Information::maximum_id
```

This is the maximum valid ID of this object API class.

This member is statically initialized and provides also the object API, class and multiprocessing node information.

It is used by _Objects_Get() to validate an object ID.

Definition at line 185 of file objectdata.h.

#### 9.73.2.9 name_length

```
uint16_t Objects_Information::name_length
```

This is the maximum length of names.

This member is statically initialized and read-only. A length of zero indicates that this API class has a no string name (OBJECTS_NO_STRING_NAME).

Definition at line 247 of file objectdata.h.

#### 9.73.2.10 object_blocks

```
Objects_Control** Objects_Information::object_blocks
```

This is a table to allocation blocks of object control blocks.

It is only used if unlimited objects are configured for this API class. The object control blocks extend and shrink by these allocation blocks.

Definition at line 272 of file objectdata.h.

#### 9.73.2.11 object_size

```
uint16_t Objects_Information::object_size
```

This is the size in bytes of each object control block.

This member is statically initialized and read-only.

Definition at line 238 of file objectdata.h.

#### 9.73.2.12 objects_per_block

```
Objects_Maximum Objects_Information::objects_per_block
```

This is the number of object control blocks in an allocation block.

This member is statically initialized and read-only. It is only used if unlimited objects are configured for this API class. It defines the count of object control blocks used to extend and shrink this API class.

Definition at line 231 of file objectdata.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/objectdata.h

## 9.74 Objects_Name Union Reference

```
#include <object.h>
```

### Public Attributes

- const char * name_p
- uint32_t name_u32

### 9.74.1 Detailed Description

The following type defines the control block used to manage object names.

Definition at line 64 of file object.h.

### 9.74.2 Member Data Documentation

#### 9.74.2.1 name_p

```
const char* Objects_Name::name_p
```

This is a pointer to a string name.

Definition at line 66 of file object.h.

#### 9.74.2.2 name_u32

```
uint32_t Objects_Name::name_u32
```

This is the actual 32-bit "raw" integer name.

Definition at line 68 of file object.h.

The documentation for this union was generated from the following file:

- cpukit/include/rtems/score/object.h

## 9.75 Partition_Control Struct Reference

This structure is the Partition Control Block (PTCB).

```
#include <partdata.h>
```

### Public Attributes

- Objects_Control Object
  *This member turns the PTCB into an object.*
- ISR_lock_Control Lock
  *This lock protects the chain of free buffers.*
- void * starting_address
  *This member contains the physical starting address of the buffer area.*
- uintptr_t length
  *This member contains the size of the buffer area in bytes.*
- size_t buffer_size
  *This member contains the size of each buffer in bytes.*
- rtems_attribute attribute_set
  *This member contains the attribute set provided at create time.*
- uintptr_t number_of_used_blocks
  *This member contains the count of allocated buffers.*
- Chain_Control Memory
  *This chain is used to manage unallocated buffers.*

### 9.75.1 Detailed Description

This structure is the Partition Control Block (PTCB).

Definition at line 38 of file partdata.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/partdata.h

## 9.76 Per_CPU_Control Struct Reference

Per CPU Core Structure.

`#include <percpu.h>`

### Public Attributes

- void * interrupt_stack_low
  *The interrupt stack low address for this processor.*
- void * interrupt_stack_high
  *The interrupt stack high address for this processor.*
- uint32_t isr_nest_level
- uint32_t isr_dispatch_disable
  *Indicates if an ISR thread dispatch is disabled.*
- volatile uint32_t thread_dispatch_disable_level
  *The thread dispatch critical section nesting counter which is used to prevent context switches at inopportune moments.*
- volatile bool dispatch_necessary
  *This is set to true when this processor needs to run the thread dispatcher.*
- bool **reserved_for_executing_alignment** [3]
- struct _Thread_Control * executing
  *This is the thread executing on this processor.*
- struct _Thread_Control * heir
  *This is the heir thread for this processor.*
- CPU_Interrupt_frame **Interrupt_frame**
- Timestamp_Control cpu_usage_timestamp
  *The CPU usage timestamp contains the time point of the last heir thread change or last CPU usage update of the executing thread of this processor.*
- struct {
  uint64_t ticks
  *Protects all watchdog operations on this processor.*
  Watchdog_Header Header [PER_CPU_WATCHDOG_COUNT]
  *Header for watchdogs.*
  } Watchdog

  *Watchdog state for this processor.*
- ISR_lock_Control Lock
  *This lock protects some members of this structure.*
- ISR_lock_Context Lock_context

*Lock context used to acquire all per-CPU locks.*

- Chain_Control Threads_in_need_for_help

  *Chain of threads in need for help.*

- Atomic_Ulong message

  *Bit field for SMP messages.*

- struct {
  const struct _Scheduler_Control * control
  *The scheduler control of the scheduler owning this processor.*
  const struct Scheduler_Context * context
  *The scheduler context of the scheduler owning this processor.*
  struct _Thread_Control * idle_if_online_and_unused
  *The idle thread for this processor in case it is online and currently not used by a scheduler instance.*
  } **Scheduler**

- struct _Thread_Control * ancestor

  *The ancestor of the executing thread.*

- char * data

  *Begin of the per-CPU data area.*

- Per_CPU_State state

  *Indicates the current state of the CPU.*

- struct {
  ISR_lock_Control Lock
  *Lock to protect the FIFO list of jobs to be performed by this processor.*
  struct Per_CPU_Job * head
  *Head of the FIFO list of jobs to be performed by this processor.*
  struct Per_CPU_Job ** tail
  *Tail of the FIFO list of jobs to be performed by this processor.*
  } Jobs

  *FIFO list of jobs to be performed by this processor.*

- bool online

  *Indicates if the processor has been successfully started via _CPU_SMP_Start_processor().*

- bool boot

  *Indicates if the processor is the one that performed the initial system initialization.*

- struct Record_Control * **record**
- Per_CPU_Stats **Stats**

## 9.76.1 Detailed Description

Per CPU Core Structure.

This structure is used to hold per core state information.

Definition at line 347 of file percpu.h.

## 9.76.2 Member Data Documentation

#### 9.76.2.1 ancestor

```
struct _Thread_Control* Per_CPU_Control::ancestor
```

The ancestor of the executing thread.

This member is used by _User_extensions_Thread_switch().

Definition at line 536 of file percpu.h.

#### 9.76.2.2 context

```
const struct Scheduler_Context* Per_CPU_Control::context
```

The scheduler context of the scheduler owning this processor.

This pointer is NULL in case this processor is currently not used by a scheduler instance.

Definition at line 522 of file percpu.h.

#### 9.76.2.3 control

```
const struct _Scheduler_Control* Per_CPU_Control::control
```

The scheduler control of the scheduler owning this processor.

This pointer is NULL in case this processor is currently not used by a scheduler instance.

Definition at line 514 of file percpu.h.

#### 9.76.2.4 cpu_usage_timestamp

```
Timestamp_Control Per_CPU_Control::cpu_usage_timestamp
```

The CPU usage timestamp contains the time point of the last heir thread change or last CPU usage update of the executing thread of this processor.

Protected by the scheduler lock.

**See also**

_Scheduler_Update_heir(), _Thread_Dispatch_update_heir() and _Thread_Get_CPU_time_used().

Definition at line 452 of file percpu.h.

#### 9.76.2.5 data

```
char* Per_CPU_Control::data
```

Begin of the per-CPU data area.

Contains items defined via PER_CPU_DATA_ITEM().

Definition at line 543 of file percpu.h.

#### 9.76.2.6 dispatch_necessary

```
volatile bool Per_CPU_Control::dispatch_necessary
```

This is set to true when this processor needs to run the thread dispatcher.

It is volatile since interrupts may alter this flag.

This member is not protected by a lock and must be accessed only by this processor. Code (e.g. scheduler and post-switch action requests) running on another processors must use an inter-processor interrupt to set the thread dispatch necessary indicator to true.

**See also**

_Thread_Get_heir_and_make_it_executing().

Definition at line 400 of file percpu.h.

#### 9.76.2.7 executing

```
struct _Thread_Control* Per_CPU_Control::executing
```

This is the thread executing on this processor.

This member is not protected by a lock. The only writer is this processor.

On SMP configurations a thread may be registered as executing on more than one processor in case a thread migration is in progress. On SMP configurations use _Thread_Is_executing_on_a_processor() to figure out if a thread context is executing on a processor.

Definition at line 420 of file percpu.h.

#### 9.76.2.8 head

```
struct Per_CPU_Job* Per_CPU_Control::head
```

Head of the FIFO list of jobs to be performed by this processor.

This member is protected by the Per_CPU_Control::Jobs::Lock lock.

Definition at line 572 of file percpu.h.

#### 9.76.2.9 Header

```
Watchdog_Header Per_CPU_Control::Header[PER_CPU_WATCHDOG_COUNT]
```

Header for watchdogs.

**See also**

Per_CPU_Watchdog_index.

Definition at line 474 of file percpu.h.

#### 9.76.2.10 heir

```
struct _Thread_Control* Per_CPU_Control::heir
```

This is the heir thread for this processor.

This member is not protected by a lock. The only writer after multitasking start is the scheduler owning this processor. It is assumed that stores to pointers are atomic on all supported SMP architectures. The CPU port specific code (inter-processor interrupt handling and _CPU_SMP_Send_interrupt()) must guarantee that this processor observes the last value written.

A thread can be a heir on at most one processor in the system.

**See also**

_Thread_Get_heir_and_make_it_executing().

Definition at line 436 of file percpu.h.

#### 9.76.2.11 isr_dispatch_disable

```
uint32_t Per_CPU_Control::isr_dispatch_disable
```

Indicates if an ISR thread dispatch is disabled.

This flag is context switched with each thread. It indicates that this thread has an interrupt stack frame on its stack. By using this flag, we can avoid nesting more interrupt dispatching attempts on a previously interrupted thread's stack.

Definition at line 379 of file percpu.h.

#### 9.76.2.12 isr_nest_level

```
uint32_t Per_CPU_Control::isr_nest_level
```

This contains the current interrupt nesting level on this CPU.

Definition at line 369 of file percpu.h.

#### 9.76.2.13 Jobs

```
struct { ... } Per_CPU_Control::Jobs
```

FIFO list of jobs to be performed by this processor.

**See also**

_SMP_Multicast_action().

#### 9.76.2.14 Lock

```
ISR_lock_Control Per_CPU_Control::Lock
```

This lock protects some members of this structure.

Lock to protect the FIFO list of jobs to be performed by this processor.

Definition at line 481 of file percpu.h.

#### 9.76.2.15 Lock_context

```
ISR_lock_Context Per_CPU_Control::Lock_context
```

Lock context used to acquire all per-CPU locks.

This member is protected by the Per_CPU_Control::Lock lock.

**See also**

_Per_CPU_Acquire_all().

Definition at line 490 of file percpu.h.

#### 9.76.2.16 message

```
Atomic_Ulong Per_CPU_Control::message
```

Bit field for SMP messages.

This member is not protected locks. Atomic operations are used to set and get the message bits.

Definition at line 505 of file percpu.h.

#### 9.76.2.17 state

```
Per_CPU_State Per_CPU_Control::state
```

Indicates the current state of the CPU.

This member is protected by the _Per_CPU_State_lock lock.

**See also**

_Per_CPU_State_change().

Definition at line 552 of file percpu.h.

#### 9.76.2.18 tail

```
struct Per_CPU_Job** Per_CPU_Control::tail
```

Tail of the FIFO list of jobs to be performed by this processor.

This member is only valid if the head is not NULL.

This member is protected by the Per_CPU_Control::Jobs::Lock lock.

Definition at line 582 of file percpu.h.

#### 9.76.2.19 Threads_in_need_for_help

```
Chain_Control Per_CPU_Control::Threads_in_need_for_help
```

Chain of threads in need for help.

This member is protected by the Per_CPU_Control::Lock lock.

Definition at line 497 of file percpu.h.

#### 9.76.2.20 ticks

```
uint64_t Per_CPU_Control::ticks
```

Protects all watchdog operations on this processor.

Watchdog ticks on this processor used for monotonic clock watchdogs.

Definition at line 467 of file percpu.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/percpu.h

## 9.77 Per_CPU_Control_envelope Struct Reference

### Public Attributes

- Per_CPU_Control **per_cpu**
- char **unused_space_for_cache_line_alignment** [PER_CPU_CONTROL_SIZE - sizeof(Per_CPU_Control)]

### 9.77.1 Detailed Description

Definition at line 604 of file percpu.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/percpu.h

## 9.78 Per_CPU_Job Struct Reference

A per-processor job.

```
#include <percpu.h>
```

### Public Attributes

- union {
  struct Per_CPU_Job * next
  *The next job in the corresponding per-processor job list.*
  Atomic_Ulong done
  *Indication if the job is done.*
  };

- const Per_CPU_Job_context * context
  *Pointer to the job context to get the handler and argument.*

## 9.78.1 Detailed Description

A per-processor job.

This structure must be as small as possible due to stack space constraints in _SMP_Multicast_action().

Definition at line 209 of file percpu.h.

## 9.78.2 Member Data Documentation

### 9.78.2.1 done

```
Atomic_Ulong Per_CPU_Job::done
```

Indication if the job is done.

A job is done if this member has the value PER_CPU_JOB_DONE. This assumes that PER_CPU_JOB_DONE is not a valid pointer value.

Definition at line 222 of file percpu.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/percpu.h

# 9.79 Per_CPU_Job_context Struct Reference

Context for per-processor jobs.

```
#include <percpu.h>
```

### Public Attributes

- Per_CPU_Job_handler handler

  *The job handler.*
- void * arg

  *The job handler argument.*

### 9.79.1 Detailed Description

Context for per-processor jobs.

This is separate from Per_CPU_Job to save stack memory in _SMP_Multicast_action().

Definition at line 184 of file percpu.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/percpu.h

## 9.80 Per_CPU_Stats Struct Reference

Per-CPU statistics.

```
#include <percpu.h>
```

### 9.80.1 Detailed Description

Per-CPU statistics.

Definition at line 236 of file percpu.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/percpu.h

## 9.81 POSIX_Spinlock_Control Struct Reference

### Public Attributes

- SMP_ticket_lock_Control **Lock**
- ISR_Level **interrupt_state**

### 9.81.1 Detailed Description

Definition at line 37 of file spinlockimpl.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/posix/spinlockimpl.h

## 9.82 pps_fetch_args Struct Reference

### Public Attributes

- int **tsformat**
- pps_info_t **pps_info_buf**
- struct timespec **timeout**

### 9.82.1 Detailed Description

Definition at line 107 of file timepps.h.

The documentation for this struct was generated from the following file:

- cpukit/include/sys/timepps.h

## 9.83 pps_fetch_ffc_args Struct Reference

### Public Attributes

- int **tsformat**
- pps_info_ffc_t **pps_info_buf_ffc**
- struct timespec **timeout**

### 9.83.1 Detailed Description

Definition at line 113 of file timepps.h.

The documentation for this struct was generated from the following file:

- cpukit/include/sys/timepps.h

## 9.84 pps_info_ffc_t Struct Reference

### Public Attributes

- pps_seq_t **assert_sequence**
- pps_seq_t **clear_sequence**
- pps_timeu_t **assert_tu**
- pps_timeu_t **clear_tu**
- ffcounter **assert_ffcount**
- ffcounter **clear_ffcount**
- int **current_mode**

### 9.84.1 Detailed Description

Definition at line 53 of file timepps.h.

The documentation for this struct was generated from the following file:

- cpukit/include/sys/timepps.h

## 9.85 pps_info_t Struct Reference

### Public Attributes

- pps_seq_t **assert_sequence**
- pps_seq_t **clear_sequence**
- pps_timeu_t **assert_tu**
- pps_timeu_t **clear_tu**
- int **current_mode**

### 9.85.1 Detailed Description

Definition at line 45 of file timepps.h.

The documentation for this struct was generated from the following file:

- cpukit/include/sys/timepps.h

## 9.86 pps_kcbind_args Struct Reference

### Public Attributes

- int **kernel_consumer**
- int **edge**
- int **tsformat**

### 9.86.1 Detailed Description

Definition at line 119 of file timepps.h.

The documentation for this struct was generated from the following file:

- cpukit/include/sys/timepps.h

## 9.87 pps_params_t Struct Reference

### Public Attributes

- int **api_version**
- int **mode**
- pps_timeu_t **assert_off_tu**
- pps_timeu_t **clear_off_tu**

### 9.87.1 Detailed Description

Definition at line 69 of file timepps.h.

The documentation for this struct was generated from the following file:

- cpukit/include/sys/timepps.h

## 9.88 pps_timeu Union Reference

### Public Attributes

- struct timespec **tspec**
- ntp_fp_t **ntpfp**
- unsigned long **longpad** [3]

### 9.88.1 Detailed Description

Definition at line 39 of file timepps.h.

The documentation for this union was generated from the following file:

- cpukit/include/sys/timepps.h

## 9.89 Priority_Actions Struct Reference

A list of priority actions.

`#include <priority.h>`

### Public Attributes

- Priority_Aggregation * actions
  *The first action of a priority action list.*

### 9.89.1 Detailed Description

A list of priority actions.

Actions are only added to the list. The action lists reside on the stack and have a short life-time. They are moved, processed or destroyed as a whole.

Definition at line 193 of file priority.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/priority.h

## 9.90 Priority_Aggregation Struct Reference

The priority aggregation.

`#include <priority.h>`

### Public Attributes

- Priority_Node Node
  *This priority node reflects the overall priority of the aggregation.*
- RBTree_Control Contributors
  *A red-black tree to contain priority nodes contributing to the overall priority of this priority aggregation.*
- const struct _Scheduler_Control * scheduler
  *The scheduler instance of this priority aggregation.*
- struct {
  Priority_Aggregation * next
  *The next priority aggregation in the action list.*
  Priority_Node * node
  *The priority node of the action.*
  Priority_Action_type type
  *The type of the action.*
  } Action

  *A priority action block to manage priority node additions, changes and removals.*

### 9.90.1 Detailed Description

The priority aggregation.

This structure serves two purposes. Firstly, it provides a place to register priority nodes and reflects the overall priority of its contributors. Secondly, it provides an action block to signal addition, change and removal of a priority node.

Definition at line 133 of file priority.h.

### 9.90.2 Member Data Documentation

#### 9.90.2.1 Node

```
Priority_Node Priority_Aggregation::Node
```

This priority node reflects the overall priority of the aggregation.

The overall priority of the aggregation is the minimum priority of the priority nodes in the contributors tree.

This priority node may be used to add this aggregation to another aggregation to build up a recursive priority scheme.

In case priority nodes of the contributors tree are added, changed or removed the priority of this node may change. To signal such changes to a priority aggregation the action block may be used.

Definition at line 147 of file priority.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/priority.h

## 9.91 Priority_bit_map_Control Struct Reference

### Public Attributes

- Priority_bit_map_Word major_bit_map

  *Each sixteen bit entry in this word is associated with one of the sixteen entries in the bit map.*
- Priority_bit_map_Word bit_map [16]

  *Each bit in the bit map indicates whether or not there are threads ready at a particular priority.*

### 9.91.1 Detailed Description

Definition at line 42 of file prioritybitmap.h.

### 9.91.2 Member Data Documentation

#### 9.91.2.1 bit_map

```
Priority_bit_map_Word Priority_bit_map_Control::bit_map[16]
```

Each bit in the bit map indicates whether or not there are threads ready at a particular priority.

The mapping of individual priority levels to particular bits is processor dependent as is the value of each bit used to indicate that threads are ready at that priority.

Definition at line 57 of file prioritybitmap.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/prioritybitmap.h

## 9.92 Priority_bit_map_Information Struct Reference

```
#include <prioritybitmap.h>
```

### Public Attributes

- Priority_bit_map_Word * minor
- Priority_bit_map_Word ready_major
- Priority_bit_map_Word ready_minor
- Priority_bit_map_Word block_major
- Priority_bit_map_Word block_minor

### 9.92.1 Detailed Description

The following record defines the information associated with each thread to manage its interaction with the priority bit maps.

Definition at line 64 of file prioritybitmap.h.

### 9.92.2 Member Data Documentation

#### 9.92.2.1 block_major

```
Priority_bit_map_Word Priority_bit_map_Information::block_major
```

This is the priority bit map block mask.

Definition at line 72 of file prioritybitmap.h.

#### 9.92.2.2 block_minor

```
Priority_bit_map_Word Priority_bit_map_Information::block_minor
```

This is the priority bit map block mask.

Definition at line 74 of file prioritybitmap.h.

#### 9.92.2.3 minor

```
Priority_bit_map_Word* Priority_bit_map_Information::minor
```

This is the address of minor bit map slot.

Definition at line 66 of file prioritybitmap.h.

#### 9.92.2.4 ready_major

```
Priority_bit_map_Word Priority_bit_map_Information::ready_major
```

This is the priority bit map ready mask.

Definition at line 68 of file prioritybitmap.h.

#### 9.92.2.5 ready_minor

```
Priority_bit_map_Word Priority_bit_map_Information::ready_minor
```

This is the priority bit map ready mask.

Definition at line 70 of file prioritybitmap.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/prioritybitmap.h

## 9.93 Priority_Node Struct Reference

The priority node to build up a priority aggregation.

```
#include <priority.h>
```

### Public Attributes

- union {
  Chain_Node **Chain**
  RBTree_Node **RBTree**
  } Node

  *Node component for a chain or red-black tree.*
- Priority_Control priority

  *The priority value of this node.*

### 9.93.1 Detailed Description

The priority node to build up a priority aggregation.

Definition at line 98 of file priority.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/priority.h

## 9.94 Rate_monotonic_Control Struct Reference

The following structure defines the control block used to manage each period.

`#include <ratemondata.h>`

### Public Attributes

- Objects_Control Object
- Watchdog_Control Timer

  *Protects the rate monotonic period state.*
- rtems_rate_monotonic_period_states state
- Priority_Node Priority

  *A priority node for use by the scheduler job release and cancel operations.*
- uint32_t next_length
- Thread_Control * owner
- Timestamp_Control cpu_usage_period_initiated
- Timestamp_Control time_period_initiated
- Rate_monotonic_Statistics Statistics
- uint32_t postponed_jobs
- uint64_t latest_deadline

### 9.94.1 Detailed Description

The following structure defines the control block used to manage each period.

State changes are protected by the default thread lock of the owner thread. The owner thread is the thread that created the period object. The owner thread field is immutable after object creation.

Definition at line 69 of file ratemondata.h.

## 9.94.2 Member Data Documentation

### 9.94.2.1 cpu_usage_period_initiated

```
Timestamp_Control Rate_monotonic_Control::cpu_usage_period_initiated
```

This field contains the cpu usage value of the owning thread when the period was initiated. It is used to compute the period's statistics.

Definition at line 107 of file ratemondata.h.

### 9.94.2.2 latest_deadline

```
uint64_t Rate_monotonic_Control::latest_deadline
```

This field contains the tick of the latest deadline decided by the period watchdog.

Definition at line 130 of file ratemondata.h.

### 9.94.2.3 next_length

```
uint32_t Rate_monotonic_Control::next_length
```

This field contains the length of the next period to be executed.

Definition at line 94 of file ratemondata.h.

### 9.94.2.4 Object

```
Objects_Control Rate_monotonic_Control::Object
```

This field is the object management portion of a Period instance.

Definition at line 71 of file ratemondata.h.

#### 9.94.2.5 owner

```
Thread_Control* Rate_monotonic_Control::owner
```

This field contains a pointer to the TCB for the thread which owns and uses this period instance.

Definition at line 100 of file ratemondata.h.

#### 9.94.2.6 postponed_jobs

```
uint32_t Rate_monotonic_Control::postponed_jobs
```

This field contains the number of postponed jobs. When the watchdog timeout, this variable will be increased immediately.

Definition at line 124 of file ratemondata.h.

#### 9.94.2.7 state

```
rtems_rate_monotonic_period_states Rate_monotonic_Control::state
```

This field indicates the current state of the period.

Definition at line 82 of file ratemondata.h.

#### 9.94.2.8 Statistics

```
Rate_monotonic_Statistics Rate_monotonic_Control::Statistics
```

This field contains the statistics maintained for the period.

Definition at line 118 of file ratemondata.h.

#### 9.94.2.9 time_period_initiated

```
Timestamp_Control Rate_monotonic_Control::time_period_initiated
```

This field contains the wall time value when the period was initiated. It is used to compute the period's statistics.

Definition at line 113 of file ratemondata.h.

#### 9.94.2.10 Timer

```
Watchdog_Control Rate_monotonic_Control::Timer
```

Protects the rate monotonic period state.

This is the timer used to provide the unblocking mechanism.

Definition at line 79 of file ratemondata.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/ratemondata.h

## 9.95 Rate_monotonic_Statistics Struct Reference

```
#include <ratemondata.h>
```

### Public Attributes

- uint32_t count
- uint32_t missed_count
- Timestamp_Control min_cpu_time
- Timestamp_Control max_cpu_time
- Timestamp_Control total_cpu_time
- Timestamp_Control min_wall_time
- Timestamp_Control max_wall_time
- Timestamp_Control total_wall_time

### 9.95.1 Detailed Description

The following defines the INTERNAL data structure that has the statistics kept on each period instance.

Definition at line 40 of file ratemondata.h.

### 9.95.2 Member Data Documentation

#### 9.95.2.1 count

```
uint32_t Rate_monotonic_Statistics::count
```

This field contains the number of periods executed.

Definition at line 42 of file ratemondata.h.

#### 9.95.2.2 max_cpu_time

```
Timestamp_Control Rate_monotonic_Statistics::max_cpu_time
```

This field contains the highest amount of CPU time used in a period.

Definition at line 49 of file ratemondata.h.

#### 9.95.2.3 max_wall_time

```
Timestamp_Control Rate_monotonic_Statistics::max_wall_time
```

This field contains the highest amount of wall time used in a period.

Definition at line 56 of file ratemondata.h.

#### 9.95.2.4 min_cpu_time

```
Timestamp_Control Rate_monotonic_Statistics::min_cpu_time
```

This field contains the least amount of CPU time used in a period.

Definition at line 47 of file ratemondata.h.

#### 9.95.2.5 min_wall_time

```
Timestamp_Control Rate_monotonic_Statistics::min_wall_time
```

This field contains the least amount of wall time used in a period.

Definition at line 54 of file ratemondata.h.

#### 9.95.2.6 missed_count

```
uint32_t Rate_monotonic_Statistics::missed_count
```

This field contains the number of periods missed.

Definition at line 44 of file ratemondata.h.

#### 9.95.2.7 total_cpu_time

```
Timestamp_Control Rate_monotonic_Statistics::total_cpu_time
```

This field contains the total amount of wall time used in a period.

Definition at line 51 of file ratemondata.h.

#### 9.95.2.8 total_wall_time

```
Timestamp_Control Rate_monotonic_Statistics::total_wall_time
```

This field contains the total amount of CPU time used in a period.

Definition at line 58 of file ratemondata.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/ratemondata.h

## 9.96 RBTree_Node Struct Reference

Red-black tree node.

```
#include <rbtree.h>
```

### Public Member Functions

- **RB_ENTRY** (RBTree_Node) Node

### 9.96.1 Detailed Description

Red-black tree node.

This is used to manage each node (element) which is placed on a red-black tree.

Definition at line 55 of file rbtree.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/rbtree.h

## 9.97 Region_Control Struct Reference

```
#include <regiondata.h>
```

### Public Attributes

- Objects_Control **Object**
- Thread_queue_Control **Wait_queue**
- const Thread_queue_Operations * **wait_operations**
- uintptr_t **maximum_segment_size**
- rtems_attribute **attribute_set**
- Heap_Control **Memory**

### 9.97.1 Detailed Description

The following records define the control block used to manage each region.

Definition at line 40 of file regiondata.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/regiondata.h

## 9.98 rtems_api_configuration_table Struct Reference

This structure contains a summary of the Classic API configuration.

`#include <config.h>`

### Public Attributes

- uint32_t maximum_tasks

  *This member contains the maximum number of Classic API Tasks configured for this application.*
- bool notepads_enabled

  *This member is true, if the Classic API Notepads are enabled, otherwise it is false.*
- uint32_t maximum_timers

  *This member contains the maximum number of Classic API Timers configured for this application.*
- uint32_t maximum_semaphores

  *This member contains the maximum number of Classic API Semaphores configured for this application.*
- uint32_t maximum_message_queues

  *This member contains the maximum number of Classic API Message Queues configured for this application.*
- uint32_t maximum_partitions

  *This member contains the maximum number of Classic API Partitions configured for this application.*
- uint32_t maximum_regions

  *This member contains the maximum number of Classic API Regions configured for this application.*
- uint32_t maximum_ports

  *This member contains the maximum number of Classic API Dual-Ported Memories configured for this application.*
- uint32_t maximum_periods

  *This member contains the maximum number of Classic API Rate Monotonic Periods configured for this application.*
- uint32_t maximum_barriers

  *This member contains the maximum number of Classic API Barriers configured for this application.*
- uint32_t number_of_initialization_tasks

  *This member contains the number of Classic API Initialization Tasks configured for this application.*
- const rtems_initialization_tasks_table * User_initialization_tasks_table

  *This member contains the pointer to Classic API Initialization Tasks Table of this application.*

### 9.98.1 Detailed Description

This structure contains a summary of the Classic API configuration.

Use rtems_configuration_get_rtems_api_configuration() to get the configuration table.

Definition at line 73 of file config.h.

### 9.98.2 Member Data Documentation

#### 9.98.2.1 maximum_barriers

```
uint32_t rtems_api_configuration_table::maximum_barriers
```

This member contains the maximum number of Classic API Barriers configured for this application.

See CONFIGURE_MAXIMUM_BARRIERS.

Definition at line 150 of file config.h.

#### 9.98.2.2 maximum_message_queues

```
uint32_t rtems_api_configuration_table::maximum_message_queues
```

This member contains the maximum number of Classic API Message Queues configured for this application.

See CONFIGURE_MAXIMUM_MESSAGE_QUEUES.

Definition at line 110 of file config.h.

#### 9.98.2.3 maximum_partitions

```
uint32_t rtems_api_configuration_table::maximum_partitions
```

This member contains the maximum number of Classic API Partitions configured for this application.

See CONFIGURE_MAXIMUM_PARTITIONS.

Definition at line 118 of file config.h.

#### 9.98.2.4 maximum_periods

```
uint32_t rtems_api_configuration_table::maximum_periods
```

This member contains the maximum number of Classic API Rate Monotonic Periods configured for this application.

See CONFIGURE_MAXIMUM_PERIODS.

Definition at line 142 of file config.h.

#### 9.98.2.5 maximum_ports

```
uint32_t rtems_api_configuration_table::maximum_ports
```

This member contains the maximum number of Classic API Dual-Ported Memories configured for this application.

See CONFIGURE_MAXIMUM_PORTS.

Definition at line 134 of file config.h.

#### 9.98.2.6 maximum_regions

```
uint32_t rtems_api_configuration_table::maximum_regions
```

This member contains the maximum number of Classic API Regions configured for this application.

See CONFIGURE_MAXIMUM_REGIONS.

Definition at line 126 of file config.h.

#### 9.98.2.7 maximum_semaphores

```
uint32_t rtems_api_configuration_table::maximum_semaphores
```

This member contains the maximum number of Classic API Semaphores configured for this application.

See CONFIGURE_MAXIMUM_SEMAPHORES.

Definition at line 102 of file config.h.

#### 9.98.2.8 maximum_tasks

```
uint32_t rtems_api_configuration_table::maximum_tasks
```

This member contains the maximum number of Classic API Tasks configured for this application.

See CONFIGURE_MAXIMUM_TASKS.

Definition at line 80 of file config.h.

#### 9.98.2.9 maximum_timers

```
uint32_t rtems_api_configuration_table::maximum_timers
```

This member contains the maximum number of Classic API Timers configured for this application.

See CONFIGURE_MAXIMUM_TIMERS.

Definition at line 94 of file config.h.

#### 9.98.2.10 number_of_initialization_tasks

```
uint32_t rtems_api_configuration_table::number_of_initialization_tasks
```

This member contains the number of Classic API Initialization Tasks configured for this application.

See CONFIGURE_RTEMS_INIT_TASKS_TABLE.

Definition at line 158 of file config.h.

#### 9.98.2.11 User_initialization_tasks_table

```
const rtems_initialization_tasks_table* rtems_api_configuration_table::User_initialization_↩
tasks_table
```

This member contains the pointer to Classic API Initialization Tasks Table of this application.

See CONFIGURE_RTEMS_INIT_TASKS_TABLE.

Definition at line 166 of file config.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/config.h

# 9.99 RTEMS_API_Control Struct Reference

`#include <tasksdata.h>`

## Public Attributes

- Event_Control Event
- Event_Control System_event
- ASR_Information Signal
- Thread_Action Signal_action

  *Signal post-switch action in case signals are pending.*

## 9.99.1 Detailed Description

This is the API specific information required by each thread for the RTEMS API to function correctly.

Definition at line 41 of file tasksdata.h.

## 9.99.2 Member Data Documentation

### 9.99.2.1 Event

`Event_Control RTEMS_API_Control::Event`

This field contains the event control for this task.

Definition at line 43 of file tasksdata.h.

### 9.99.2.2 Signal

`ASR_Information RTEMS_API_Control::Signal`

This field contains the Classic API Signal information for this task.

Definition at line 47 of file tasksdata.h.

#### 9.99.2.3 System_event

```
Event_Control RTEMS_API_Control::System_event
```

This field contains the system event control for this task.

Definition at line 45 of file tasksdata.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/tasksdata.h

## 9.100 rtems_assert_context Struct Reference

%

```
#include <fatal.h>
```

### Public Attributes

- const char * file
  *This member is.*
- int line
  *This member is.*
- const char * function
  *This member is.*
- const char * failed_expression
  *This member is.*

### 9.100.1 Detailed Description

%

Definition at line 83 of file fatal.h.

### 9.100.2 Member Data Documentation

#### 9.100.2.1 failed_expression

```
const char* rtems_assert_context::failed_expression
```

This member is.

%

Definition at line 110 of file fatal.h.

#### 9.100.2.2 file

```
const char* rtems_assert_context::file
```

This member is.

%

Definition at line 89 of file fatal.h.

#### 9.100.2.3 function

```
const char* rtems_assert_context::function
```

This member is.

%

Definition at line 103 of file fatal.h.

#### 9.100.2.4 line

```
int rtems_assert_context::line
```

This member is.

%

Definition at line 96 of file fatal.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/fatal.h

## 9.101 rtems_assoc_32_pair Struct Reference

### Public Attributes

- uint32_t **bits**
- const char * **name**

### 9.101.1 Detailed Description

Definition at line 159 of file assoc.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/assoc.h

## 9.102 rtems_assoc_t Struct Reference

### Public Attributes

- const char * **name**
- uint32_t **local_value**
- uint32_t **remote_value**

### 9.102.1 Detailed Description

Definition at line 31 of file assoc.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/assoc.h

## 9.103 rtems_binary_semaphore Struct Reference

### Public Attributes

- struct _Semaphore_Control **Semaphore**

### 9.103.1 Detailed Description

Definition at line 221 of file thread.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/thread.h

## 9.104 rtems_driver_address_table Struct Reference

This structure contains the device driver entries.

```
#include <io.h>
```

### Public Attributes

- rtems_device_driver_entry initialization_entry
  *This member is the device driver initialization entry.*
- rtems_device_driver_entry open_entry
  *This member is the device driver open entry.*
- rtems_device_driver_entry close_entry
  *This member is the device driver close entry.*
- rtems_device_driver_entry read_entry
  *This member is the device driver read entry.*
- rtems_device_driver_entry write_entry
  *This member is the device driver write entry.*
- rtems_device_driver_entry control_entry
  *This member is the device driver control entry.*

### 9.104.1 Detailed Description

This structure contains the device driver entries.

This structure is used to register a device driver via rtems_io_register_driver().

Definition at line 132 of file io.h.

### 9.104.2 Member Data Documentation

#### 9.104.2.1 close_entry

```
rtems_device_driver_entry rtems_driver_address_table::close_entry
```

This member is the device driver close entry.

This entry is called by rtems_io_close().

Definition at line 152 of file io.h.

#### 9.104.2.2 control_entry

```
rtems_device_driver_entry rtems_driver_address_table::control_entry
```

This member is the device driver control entry.

This entry is called by rtems_io_control().

Definition at line 173 of file io.h.

#### 9.104.2.3 initialization_entry

```
rtems_device_driver_entry rtems_driver_address_table::initialization_entry
```

This member is the device driver initialization entry.

This entry is called by rtems_io_initialize().

Definition at line 138 of file io.h.

#### 9.104.2.4 open_entry

`rtems_device_driver_entry rtems_driver_address_table::open_entry`

This member is the device driver open entry.

This entry is called by rtems_io_open().

Definition at line 145 of file io.h.

#### 9.104.2.5 read_entry

`rtems_device_driver_entry rtems_driver_address_table::read_entry`

This member is the device driver read entry.

This entry is called by rtems_io_read().

Definition at line 159 of file io.h.

#### 9.104.2.6 write_entry

`rtems_device_driver_entry rtems_driver_address_table::write_entry`

This member is the device driver write entry.

This entry is called by rtems_io_write().

Definition at line 166 of file io.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/io.h

## 9.105 rtems_filesystem_eval_path_context_t Struct Reference

Path evaluation context.

`#include <libio.h>`

## Public Attributes

- const char * path
- size_t pathlen
- const char * token
- size_t tokenlen
- int flags
- int recursionlevel
- rtems_filesystem_location_info_t currentloc
- rtems_filesystem_global_location_t * rootloc
- rtems_filesystem_global_location_t * startloc

### 9.105.1 Detailed Description

Path evaluation context.

Definition at line 84 of file libio.h.

### 9.105.2 Member Data Documentation

#### 9.105.2.1 currentloc

```
rtems_filesystem_location_info_t rtems_filesystem_eval_path_context_t::currentloc
```

This is the current file system location of the evaluation process. Tokens are evaluated with respect to the current location. The token interpretation may change the current location. The purpose of the path evaluation is to change the start location into a final current location according to the path.

Definition at line 135 of file libio.h.

#### 9.105.2.2 flags

```
int rtems_filesystem_eval_path_context_t::flags
```

The path evaluation is controlled by the following flags

- RTEMS_FS_PERMS_READ,
- RTEMS_FS_PERMS_WRITE,
- RTEMS_FS_PERMS_EXEC,
- RTEMS_FS_FOLLOW_HARD_LINK,
- RTEMS_FS_FOLLOW_SYM_LINK,
- RTEMS_FS_MAKE,
- RTEMS_FS_EXCLUSIVE,
- RTEMS_FS_ACCEPT_RESIDUAL_DELIMITERS, and
- RTEMS_FS_REJECT_TERMINAL_DOT.

Definition at line 119 of file libio.h.

#### 9.105.2.3 path

```
const char* rtems_filesystem_eval_path_context_t::path
```

The contents of the remaining path to be evaluated.

Definition at line 88 of file libio.h.

#### 9.105.2.4 pathlen

```
size_t rtems_filesystem_eval_path_context_t::pathlen
```

The length of the remaining path to be evaluated.

Definition at line 93 of file libio.h.

#### 9.105.2.5 recursionlevel

```
int rtems_filesystem_eval_path_context_t::recursionlevel
```

Symbolic link evaluation is a recursive operation. This field helps to limit the recursion level and thus prevents a stack overflow. The recursion level is limited by RTEMS_FILESYSTEM_SYMLOOP_MAX.

Definition at line 126 of file libio.h.

#### 9.105.2.6 rootloc

```
rtems_filesystem_global_location_t* rtems_filesystem_eval_path_context_t::rootloc
```

The location of the root directory of the user environment during the evaluation start.

Definition at line 141 of file libio.h.

#### 9.105.2.7 startloc

```
rtems_filesystem_global_location_t* rtems_filesystem_eval_path_context_t::startloc
```

The start location of the evaluation process. The start location my change during symbolic link evaluation.

Definition at line 147 of file libio.h.

#### 9.105.2.8 token

```
const char* rtems_filesystem_eval_path_context_t::token
```

The contents of the token to be evaluated with respect to the current location.

Definition at line 99 of file libio.h.

#### 9.105.2.9 tokenlen

```
size_t rtems_filesystem_eval_path_context_t::tokenlen
```

The length of the token to be evaluated with respect to the current location.

Definition at line 105 of file libio.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/libio.h

## 9.106 rtems_filesystem_global_location_t Struct Reference

Global file system location.

`#include <fs.h>`

### Public Attributes

- rtems_filesystem_location_info_t **location**
- int **reference_count**
- struct rtems_filesystem_global_location_t * deferred_released_next
- int deferred_released_count

### 9.106.1 Detailed Description

Global file system location.

The global file system locations are used for

- the mount point location in the mount table entry,
- the file system root location in the mount table entry,
- the root directory location in the user environment, and
- the current directory location in the user environment.

During the path evaluation global start locations are obtained to ensure that the current file system will be not unmounted in the meantime.

To support a release within critical sections of the operating system a deferred release is supported. This is similar to malloc() and free().

**See also**

rtems_filesystem_global_location_obtain() and rtems_filesystem_global_location_release().

Definition at line 81 of file fs.h.

### 9.106.2 Member Data Documentation

#### 9.106.2.1 deferred_released_count

```
int rtems_filesystem_global_location_t::deferred_released_count
```

A release within a critical section can happen multiple times. This field counts the deferred releases.

Definition at line 96 of file fs.h.

#### 9.106.2.2 deferred_released_next

```
struct rtems_filesystem_global_location_t* rtems_filesystem_global_location_t::deferred_↩
released_next
```

A release within a critical section of the operating system will add this location to a list of deferred released locations. This list is processed in the next rtems_filesystem_global_location_obtain() in FIFO order.

Definition at line 90 of file fs.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/fs.h

## 9.107 rtems_filesystem_limits_and_options_t Struct Reference

Contain file system specific information which is required to support fpathconf().

```
#include <libio.h>
```

### Public Attributes

- int **link_max**
- int **max_canon**
- int **max_input**
- int **name_max**
- int **path_max**
- int **pipe_buf**
- int **posix_async_io**
- int **posix_chown_restrictions**
- int **posix_no_trunc**
- int **posix_prio_io**
- int **posix_sync_io**
- int **posix_vdisable**

### 9.107.1 Detailed Description

Contain file system specific information which is required to support fpathconf().

Definition at line 1291 of file libio.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/libio.h

## 9.108 rtems_filesystem_location_info_tt Struct Reference

File system location.

`#include <fs.h>`

### Public Attributes

- rtems_chain_node **mt_entry_node**
- void * **node_access**
- void * **node_access_2**
- const rtems_filesystem_file_handlers_r * **handlers**
- rtems_filesystem_mount_table_entry_t * **mt_entry**

### 9.108.1 Detailed Description

File system location.

Definition at line 53 of file fs.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/fs.h

## 9.109 rtems_filesystem_mount_configuration Struct Reference

### Public Attributes

- const char * **source**
- const char * **target**
- const char * **filesystemtype**
- rtems_filesystem_options_t **options**
- const void * **data**

### 9.109.1 Detailed Description

Definition at line 1857 of file libio.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/libio.h

## 9.110 rtems_filesystem_mount_table_entry_tt Struct Reference

Mount table entry.

`#include <libio.h>`

### Public Attributes

- rtems_chain_node **mt_node**
- void * **fs_info**
- const rtems_filesystem_operations_table * **ops**
- const void * **immutable_fs_info**
- rtems_chain_control **location_chain**
- rtems_filesystem_global_location_t * **mt_point_node**
- rtems_filesystem_global_location_t * **mt_fs_root**
- bool **mounted**
- bool **writeable**
- const rtems_filesystem_limits_and_options_t * **pathconf_limits_and_options**
- const char * **target**
- const char * **type**
- char * **dev**
- rtems_id unmount_task

### 9.110.1 Detailed Description

Mount table entry.

Definition at line 1606 of file libio.h.

### 9.110.2 Member Data Documentation

#### 9.110.2.1 unmount_task

```
rtems_id rtems_filesystem_mount_table_entry_tt::unmount_task
```

The task that initiated the unmount process. After unmount process completion this task will be notified via the transient event.

**See also**

ClassicEventTransient.

Definition at line 1643 of file libio.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/libio.h

## 9.111 rtems_filesystem_table_t Struct Reference

File system table entry.

```
#include <libio.h>
```

### Public Attributes

- const char * **type**
- rtems_filesystem_fsmount_me_t **mount_h**

### 9.111.1 Detailed Description

File system table entry.

Definition at line 1658 of file libio.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/libio.h

## 9.112 rtems_initialization_tasks_table Struct Reference

%

```
#include <tasks.h>
```

## Public Attributes

- rtems_name name
  %
- size_t stack_size
  %
- rtems_task_priority initial_priority
  %
- rtems_attribute attribute_set
  %
- rtems_task_entry entry_point
  %
- rtems_mode mode_set
  %
- rtems_task_argument argument
  %

### 9.112.1 Detailed Description

%

Definition at line 846 of file tasks.h.

### 9.112.2 Member Data Documentation

#### 9.112.2.1 argument

```
rtems_task_argument rtems_initialization_tasks_table::argument
```

%

%

Definition at line 894 of file tasks.h.

#### 9.112.2.2 attribute_set

```
rtems_attribute rtems_initialization_tasks_table::attribute_set
```

%

%

Definition at line 873 of file tasks.h.

#### 9.112.2.3 entry_point

```
rtems_task_entry rtems_initialization_tasks_table::entry_point
```

%

%

Definition at line 880 of file tasks.h.

#### 9.112.2.4 initial_priority

```
rtems_task_priority rtems_initialization_tasks_table::initial_priority
```

%

%

Definition at line 866 of file tasks.h.

#### 9.112.2.5 mode_set

```
rtems_mode rtems_initialization_tasks_table::mode_set
```

%

%

Definition at line 887 of file tasks.h.

#### 9.112.2.6 name

```
rtems_name rtems_initialization_tasks_table::name
```

%

%

Definition at line 852 of file tasks.h.

#### 9.112.2.7 stack_size

```
size_t rtems_initialization_tasks_table::stack_size
```

%

%

Definition at line 859 of file tasks.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/tasks.h

## 9.113 rtems_interrupt_server_action Struct Reference

An interrupt server action.

```
#include <irq-extension.h>
```

### Public Attributes

- struct rtems_interrupt_server_action * **next**
- rtems_interrupt_handler **handler**
- void * **arg**

### 9.113.1 Detailed Description

An interrupt server action.

This structure must be treated as an opaque data type. Members must not be accessed directly.

**See also**

rtems_interrupt_server_action_prepend().

Definition at line 245 of file irq-extension.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/irq-extension.h

## 9.114 rtems_interrupt_server_config Struct Reference

An interrupt server configuration.

```
#include <irq-extension.h>
```

## Public Attributes

- rtems_name name
  *The task name of the interrupt server.*
- rtems_task_priority priority
  *The initial task priority of the interrupt server.*
- void * storage_area
  *The task storage area of the interrupt server.*
- size_t storage_size
  *The task storage size of the interrupt server.*
- rtems_mode modes
  *The initial task modes of the interrupt server.*
- rtems_attribute attributes
  *The task attributes of the interrupt server.*
- void(* destroy )(rtems_interrupt_server_control *)
  *An optional handler to destroy the interrupt server control handed over to rtems_interrupt_server_create().*

### 9.114.1 Detailed Description

An interrupt server configuration.

**See also**

rtems_interrupt_server_create()

Definition at line 279 of file irq-extension.h.

### 9.114.2 Member Data Documentation

#### 9.114.2.1 destroy

```
void( * rtems_interrupt_server_config::destroy) (rtems_interrupt_server_control *)
```

An optional handler to destroy the interrupt server control handed over to rtems_interrupt_server_create().

This handler is called in the context of the interrupt server to be deleted, see also rtems_interrupt_server_delete().

Definition at line 323 of file irq-extension.h.

#### 9.114.2.2 storage_area

```
void* rtems_interrupt_server_config::storage_area
```

The task storage area of the interrupt server.

It shall be NULL for interrupt servers created by rtems_interrupt_server_create().

Definition at line 296 of file irq-extension.h.

#### 9.114.2.3 storage_size

```
size_t rtems_interrupt_server_config::storage_size
```

The task storage size of the interrupt server.

For interrupt servers created by rtems_interrupt_server_create() this is the task stack size.

Definition at line 304 of file irq-extension.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/irq-extension.h

## 9.115 rtems_interrupt_server_control Struct Reference

An interrupt server control.

```
#include <irq-extension.h>
```

### Public Attributes

- rtems_chain_control **entries**
- rtems_id **server**
- unsigned long **errors**
- uint32_t **index**
- rtems_chain_node **node**
- void(∗ **destroy** )(struct rtems_interrupt_server_control ∗)

### 9.115.1 Detailed Description

An interrupt server control.

This structure must be treated as an opaque data type. Members must not be accessed directly.

**See also**

rtems_interrupt_server_create()

Definition at line 264 of file irq-extension.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/irq-extension.h

## 9.116 rtems_interrupt_server_entry Struct Reference

An interrupt server entry.

`#include <irq-extension.h>`

### Public Attributes

- rtems_chain_node **node**
- void * **server**
- rtems_vector_number **vector**
- rtems_interrupt_server_action * **actions**

### 9.116.1 Detailed Description

An interrupt server entry.

This structure must be treated as an opaque data type. Members must not be accessed directly.

**See also**

rtems_interrupt_server_entry_initialize(), rtems_interrupt_server_action_prepend(), rtems_interrupt_server_entry_submit(), and rtems_interrupt_server_entry_destroy().

Definition at line 337 of file irq-extension.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/irq-extension.h

## 9.117 rtems_interrupt_server_request Struct Reference

An interrupt server request.

`#include <irq-extension.h>`

### Public Attributes

- rtems_interrupt_server_entry **entry**
- rtems_interrupt_server_action **action**

### 9.117.1 Detailed Description

An interrupt server request.

This structure must be treated as an opaque data type. Members must not be accessed directly.

**See also**

rtems_interrupt_server_request_initialize(), rtems_interrupt_server_request_set_vector(), rtems_interrupt_server_request_subm
and rtems_interrupt_server_request_destroy().

Definition at line 355 of file irq-extension.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/irq-extension.h

## 9.118 rtems_libio_ioctl_args_t Struct Reference

Parameter block for ioctl.

`#include <libio.h>`

### Public Attributes

- rtems_libio_t * **iop**
- ioctl_command_t **command**
- void * **buffer**
- int **ioctl_return**

### 9.118.1 Detailed Description

Parameter block for ioctl.

Definition at line 1357 of file libio.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/libio.h

## 9.119 rtems_libio_open_close_args_t Struct Reference

Parameter block for open/close.

`#include <libio.h>`

### Public Attributes

- rtems_libio_t * **iop**
- uint32_t **flags**
- uint32_t **mode**

### 9.119.1 Detailed Description

Parameter block for open/close.

Definition at line 1348 of file libio.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/libio.h

## 9.120 rtems_libio_rw_args_t Struct Reference

Paramameter block for read/write.

```
#include <libio.h>
```

### Public Attributes

- rtems_libio_t * **iop**
- off_t **offset**
- char * **buffer**
- uint32_t **count**
- uint32_t **flags**
- uint32_t **bytes_moved**

### 9.120.1 Detailed Description

Paramameter block for read/write.

It must include 'offset' instead of using iop's offset since we can have multiple outstanding i/o's on a device.

Definition at line 1336 of file libio.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/libio.h

## 9.121 rtems_libio_tt Struct Reference

An open file data structure.

```
#include <libio.h>
```

### Public Attributes

- Atomic_Uint **flags**
- off_t **offset**
- rtems_filesystem_location_info_t **pathinfo**
- uint32_t **data0**
- void * **data1**

### 9.121.1 Detailed Description

An open file data structure.

It will be indexed by 'fd'.

**Todo** Should really have a separate per/file data structure that this points to (eg: offset, driver, pathname should be in that)

Definition at line 1322 of file libio.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/libio.h

## 9.122 rtems_message_queue_config Struct Reference

This structure defines the configuration of a message queue constructed by rtems_message_queue_construct().

```
#include <message.h>
```

### Public Attributes

- rtems_name name

  *This member defines the name of the message queue.*
- uint32_t maximum_pending_messages

  *This member defines the maximum number of pending messages supported by the message queue.*
- size_t maximum_message_size

  *This member defines the maximum message size supported by the message queue.*
- void * storage_area

  *This member shall point to the message buffer storage area begin.*
- size_t storage_size

  *This member defines size of the message buffer storage area in bytes.*
- void(* storage_free )(void *)

  *This member defines the optional handler to free the message buffer storage area.*
- rtems_attribute attributes

  *This member defines the attributes of the message queue.*

### 9.122.1 Detailed Description

This structure defines the configuration of a message queue constructed by rtems_message_queue_construct().

Definition at line 126 of file message.h.

### 9.122.2 Member Data Documentation

#### 9.122.2.1 storage_area

```
void* rtems_message_queue_config::storage_area
```

This member shall point to the message buffer storage area begin.

The message buffer storage area for the message queue shall be an array of the type defined by RTEMS_MESSAGE_QUEUE_BUFFER() with a maximum message size equal to the maximum message size of this configuration.

Definition at line 151 of file message.h.

#### 9.122.2.2 storage_free

```
void( * rtems_message_queue_config::storage_free) (void *)
```

This member defines the optional handler to free the message buffer storage area.

It is called when the message queue is deleted. It is called from task context under protection of the object allocator lock. It is allowed to call free() in this handler. If handler is NULL, then no action will be performed.

Definition at line 167 of file message.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/message.h

## 9.123 rtems_object_api_class_information Struct Reference

%

```
#include <object.h>
```

## Public Attributes

- rtems_id minimum_id
  *This member is.*
- rtems_id maximum_id
  *This member is.*
- uint32_t maximum
  *This member is.*
- bool auto_extend
  *This member is.*
- uint32_t unallocated
  *This member is.*

### 9.123.1 Detailed Description

%

Definition at line 85 of file object.h.

### 9.123.2 Member Data Documentation

#### 9.123.2.1 auto_extend

```
bool rtems_object_api_class_information::auto_extend
```

This member is.

%

Definition at line 112 of file object.h.

#### 9.123.2.2 maximum

```
uint32_t rtems_object_api_class_information::maximum
```

This member is.

%

Definition at line 105 of file object.h.

#### 9.123.2.3 maximum_id

```
rtems_id rtems_object_api_class_information::maximum_id
```

This member is.

%

Definition at line 98 of file object.h.

#### 9.123.2.4 minimum_id

```
rtems_id rtems_object_api_class_information::minimum_id
```

This member is.

%

Definition at line 91 of file object.h.

#### 9.123.2.5 unallocated

```
uint32_t rtems_object_api_class_information::unallocated
```

This member is.

%

Definition at line 119 of file object.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/object.h

## 9.124 rtems_printer Struct Reference

```
#include <printer.h>
```

### Public Attributes

- void * **context**
- rtems_print_printer **printer**

### 9.124.1 Detailed Description

Type definition for the printer structure used to access the kernel print support.

Definition at line 55 of file printer.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/printer.h

## 9.125 rtems_printer_task_context Struct Reference

### Public Attributes

- rtems_id **task**
- rtems_chain_control **free_buffers**
- rtems_chain_control **todo_buffers**
- size_t **task_stack_size**
- rtems_task_priority **task_priority**
- int **fd**
- void * **buffer_table**
- size_t **buffer_count**
- size_t **buffer_size**

### 9.125.1 Detailed Description

Definition at line 116 of file printer.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/printer.h

## 9.126 rtems_rate_monotonic_period_statistics Struct Reference

%

```
#include <ratemon.h>
```

## Public Attributes

- uint32_t count
  *This member is.*
- uint32_t missed_count
  *This member is.*
- struct timespec min_cpu_time
  *This member is.*
- struct timespec max_cpu_time
  *This member is.*
- struct timespec total_cpu_time
  *This member is.*
- struct timespec min_wall_time
  *This member is.*
- struct timespec max_wall_time
  *This member is.*
- struct timespec total_wall_time
  *This member is.*

### 9.126.1 Detailed Description

%

Definition at line 196 of file ratemon.h.

### 9.126.2 Member Data Documentation

#### 9.126.2.1 count

```
uint32_t rtems_rate_monotonic_period_statistics::count
```

This member is.

%

Definition at line 202 of file ratemon.h.

#### 9.126.2.2 max_cpu_time

```
struct timespec rtems_rate_monotonic_period_statistics::max_cpu_time
```

This member is.

%

Definition at line 209 of file ratemon.h.

#### 9.126.2.3 max_wall_time

```
struct timespec rtems_rate_monotonic_period_statistics::max_wall_time
```

This member is.

%

Definition at line 209 of file ratemon.h.

#### 9.126.2.4 min_cpu_time

```
struct timespec rtems_rate_monotonic_period_statistics::min_cpu_time
```

This member is.

%

Definition at line 209 of file ratemon.h.

#### 9.126.2.5 min_wall_time

```
struct timespec rtems_rate_monotonic_period_statistics::min_wall_time
```

This member is.

%

Definition at line 209 of file ratemon.h.

#### 9.126.2.6 missed_count

```
uint32_t rtems_rate_monotonic_period_statistics::missed_count
```

This member is.

%

Definition at line 209 of file ratemon.h.

#### 9.126.2.7 total_cpu_time

```
struct timespec rtems_rate_monotonic_period_statistics::total_cpu_time
```

This member is.

%

Definition at line 209 of file ratemon.h.

#### 9.126.2.8 total_wall_time

```
struct timespec rtems_rate_monotonic_period_statistics::total_wall_time
```

This member is.

%

Definition at line 209 of file ratemon.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/ratemon.h

## 9.127 rtems_rate_monotonic_period_status Struct Reference

%

```
#include <ratemon.h>
```

### Public Attributes

- rtems_id owner
  *This member is.*
- rtems_rate_monotonic_period_states state
  *This member is.*
- struct timespec since_last_period
  *This member is.*
- struct timespec executed_since_last_period
  *This member is.*
- uint32_t postponed_jobs_count
  *This member is.*

### 9.127.1 Detailed Description

%

Definition at line 277 of file ratemon.h.

## 9.127.2 Member Data Documentation

### 9.127.2.1 executed_since_last_period

```
struct timespec rtems_rate_monotonic_period_status::executed_since_last_period
```

This member is.

%

Definition at line 290 of file ratemon.h.

### 9.127.2.2 owner

```
rtems_id rtems_rate_monotonic_period_status::owner
```

This member is.

%

Definition at line 283 of file ratemon.h.

### 9.127.2.3 postponed_jobs_count

```
uint32_t rtems_rate_monotonic_period_status::postponed_jobs_count
```

This member is.

%

Definition at line 311 of file ratemon.h.

### 9.127.2.4 since_last_period

```
struct timespec rtems_rate_monotonic_period_status::since_last_period
```

This member is.

%

Definition at line 290 of file ratemon.h.

#### 9.127.2.5 state

`rtems_rate_monotonic_period_states rtems_rate_monotonic_period_status::state`

This member is.

%

Definition at line 290 of file ratemon.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/ratemon.h

## 9.128 rtems_resource_posix_api Struct Reference

### Public Attributes

- uint32_t **active_message_queues**
- uint32_t **active_semaphores**
- uint32_t **active_threads**
- uint32_t **active_timers**

### 9.128.1 Detailed Description

Definition at line 115 of file libcsupport.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/libcsupport.h

## 9.129 rtems_resource_rtems_api Struct Reference

### Public Attributes

- uint32_t **active_barriers**
- uint32_t **active_extensions**
- uint32_t **active_message_queues**
- uint32_t **active_partitions**
- uint32_t **active_periods**
- uint32_t **active_ports**
- uint32_t **active_regions**
- uint32_t **active_semaphores**
- uint32_t **active_tasks**
- uint32_t **active_timers**

### 9.129.1 Detailed Description

Definition at line 102 of file libcsupport.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/libcsupport.h

## 9.130 rtems_resource_snapshot Struct Reference

### Public Attributes

- Heap_Information_block **workspace_info**
- Heap_Information_block **heap_info**
- uint32_t **active_posix_key_value_pairs**
- uint32_t **active_posix_keys**
- rtems_resource_rtems_api **rtems_api**
- rtems_resource_posix_api **posix_api**
- int **open_files**

### 9.130.1 Detailed Description

Definition at line 122 of file libcsupport.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/libcsupport.h

## 9.131 rtems_sysinit_item Struct Reference

### Public Attributes

- rtems_sysinit_handler **handler**

### 9.131.1 Detailed Description

Definition at line 117 of file sysinit.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/sysinit.h

# 9.132 rtems_task_config Struct Reference

This structure defines the configuration of a task constructed by rtems_task_construct().

`#include <tasks.h>`

## Public Attributes

- rtems_name name
  *This member defines the name of the task.*
- rtems_task_priority initial_priority
  *This member defines the initial priority of the task.*
- void * storage_area
  *This member shall point to the task storage area begin.*
- size_t storage_size
  *This member defines size of the task storage area in bytes.*
- size_t maximum_thread_local_storage_size
  *This member defines the maximum thread-local storage size supported by the task storage area.*
- void(* storage_free )(void *)
  *This member defines the optional handler to free the task storage area.*
- rtems_mode initial_modes
  *This member defines the initial modes of the task.*
- rtems_attribute attributes
  *This member defines the attributes of the task.*

## 9.132.1 Detailed Description

This structure defines the configuration of a task constructed by rtems_task_construct().

Definition at line 443 of file tasks.h.

## 9.132.2 Member Data Documentation

### 9.132.2.1 maximum_thread_local_storage_size

`size_t rtems_task_config::maximum_thread_local_storage_size`

This member defines the maximum thread-local storage size supported by the task storage area.

Use RTEMS_ALIGN_UP() and RTEMS_TASK_STORAGE_ALIGNMENT to adjust the size to meet the minimum alignment requirement of a thread-local storage area used to construct a task.

If the value is less than the actual thread-local storage size, then the task construction by rtems_task_construct() fails.

If the is less than the task storage area size, then the task construction by rtems_task_construct() fails.

The actual thread-local storage size is determined when the application executable is linked. The `rtems-exeinfo` command line tool included in the RTEMS Tools can be used to obtain the thread-local storage size and alignment of an application executable.

The application may configure the maximum thread-local storage size for all threads explicitly through the CONFIGURE_MAXIMUM_THREAD_LOCAL_STORAGE_SIZE configuration option.

Definition at line 499 of file tasks.h.

#### 9.132.2.2 storage_area

```
void* rtems_task_config::storage_area
```

This member shall point to the task storage area begin.

The task storage area will contain the task stack, the thread-local storage, and the floating-point context on architectures with a separate floating-point context.

The task storage area begin address and size should be aligned by RTEMS_TASK_STORAGE_ALIGNMENT. To avoid memory waste, use RTEMS_ALIGNED() and RTEMS_TASK_STORAGE_ALIGNMENT to enforce the recommended alignment of a statically allocated task storage area.

Definition at line 466 of file tasks.h.

#### 9.132.2.3 storage_free

```
void( * rtems_task_config::storage_free) (void *)
```

This member defines the optional handler to free the task storage area.

It is called on exactly two mutually exclusive occasions. Firstly, when the task construction aborts due to a failed task create extension, or secondly, when the task is deleted. It is called from task context under protection of the object allocator lock. It is allowed to call free() in this handler. If handler is NULL, then no action will be performed.

Definition at line 511 of file tasks.h.

#### 9.132.2.4 storage_size

```
size_t rtems_task_config::storage_size
```

This member defines size of the task storage area in bytes.

Use the RTEMS_TASK_STORAGE_SIZE() macro to determine the recommended task storage area size.

Definition at line 474 of file tasks.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/tasks.h

## 9.133 rtems_termios_callbacks Struct Reference

### Public Attributes

- int(* **firstOpen** )(int major, int minor, void *arg)
- int(* **lastClose** )(int major, int minor, void *arg)
- int(* **pollRead** )(int minor)
- ssize_t(* **write** )(int minor, const char *buf, size_t len)
- int(* **setAttributes** )(int minor, const struct termios *t)
- int(* **stopRemoteTx** )(int minor)
- int(* **startRemoteTx** )(int minor)
- int **outputUsesInterrupts**

### 9.133.1 Detailed Description

Definition at line 1881 of file libio.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/libio.h

## 9.134 rtems_termios_device_context Struct Reference

Termios device context.

`#include <termiostypes.h>`

### Public Attributes

- union {
  rtems_interrupt_lock **interrupt**
  rtems_mutex **mutex**
  } **lock**
- void(∗ **lock_acquire** )(struct rtems_termios_device_context ∗, rtems_interrupt_lock_context ∗)
- void(∗ **lock_release** )(struct rtems_termios_device_context ∗, rtems_interrupt_lock_context ∗)

### 9.134.1 Detailed Description

Termios device context.

**See also**

RTEMS_TERMIOS_DEVICE_CONTEXT_INITIALIZER(), rtems_termios_device_context_initialize() and rtems_termios_device_install().

Definition at line 75 of file termiostypes.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/termiostypes.h

## 9.135 rtems_termios_device_flow Struct Reference

Termios device flow control handler.

`#include <termiostypes.h>`

## Public Attributes

- void(∗ stop_remote_tx )(rtems_termios_device_context ∗context)

  *Indicate to stop remote transmitter.*
- void(∗ start_remote_tx )(rtems_termios_device_context ∗context)

  *Indicate to start remote transmitter.*

### 9.135.1 Detailed Description

Termios device flow control handler.

**See also**

rtems_termios_device_install().

Definition at line 248 of file termiostypes.h.

### 9.135.2 Member Data Documentation

#### 9.135.2.1 start_remote_tx

```
void(* rtems_termios_device_flow::start_remote_tx) (rtems_termios_device_context *context)
```

Indicate to start remote transmitter.

**Parameters**

| | | |
|---|---|---|
| in | *context* | The Termios device context. |

Definition at line 261 of file termiostypes.h.

#### 9.135.2.2 stop_remote_tx

```
void(* rtems_termios_device_flow::stop_remote_tx) (rtems_termios_device_context *context)
```

Indicate to stop remote transmitter.

**Parameters**

| | | |
|---|---|---|
| in | *context* | The Termios device context. |

Definition at line 254 of file termiostypes.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/termiostypes.h

## 9.136 rtems_termios_device_handler Struct Reference

Termios device handler.

`#include <termiostypes.h>`

### Public Attributes

- bool(* first_open )(struct rtems_termios_tty *tty, rtems_termios_device_context *context, struct termios *term, rtems_libio_open_close_args_t *args)
  *First open of this device.*
- void(* last_close )(struct rtems_termios_tty *tty, rtems_termios_device_context *context, rtems_libio_open_close_args_t *args)
  *Last close of this device.*
- int(* poll_read )(rtems_termios_device_context *context)
  *Polled read.*
- void(* write )(rtems_termios_device_context *context, const char *buf, size_t len)
  *Polled write in case mode is TERMIOS_POLLED or write support otherwise.*
- bool(* set_attributes )(rtems_termios_device_context *context, const struct termios *term)
  *Set attributes after a Termios settings change.*
- int(* ioctl )(rtems_termios_device_context *context, ioctl_command_t request, void *buffer)
  *IO control handler.*
- rtems_termios_device_mode mode
  *Termios device mode.*

### 9.136.1 Detailed Description

Termios device handler.

**See also**

rtems_termios_device_install().

Definition at line 141 of file termiostypes.h.

### 9.136.2 Member Data Documentation

#### 9.136.2.1 first_open

```
bool(* rtems_termios_device_handler::first_open) (struct rtems_termios_tty *tty, rtems_termios_device_context
*context, struct termios *term, rtems_libio_open_close_args_t *args)
```

First open of this device.

**Parameters**

| | | |
|---|---|---|
| in | *tty* | The Termios control. This parameter may be passed to interrupt service routines since it must be provided for the rtems_termios_enqueue_raw_characters() and rtems_termios_dequeue_characters() functions. |
| in | *context* | The Termios device context. |
| in | *term* | The current Termios attributes. |
| in | *args* | The open/close arguments. This is parameter provided to support legacy drivers. It must not be used by new drivers. |

**Return values**

| | |
|---|---|
| *true* | Successful operation. |
| *false* | Cannot open device. |

**See also**

rtems_termios_get_device_context() and rtems_termios_set_best_baud().

Definition at line 159 of file termiostypes.h.

### 9.136.2.2 ioctl

```
int(* rtems_termios_device_handler::ioctl) (rtems_termios_device_context *context, ioctl_↩
command_t request, void *buffer)
```

IO control handler.

Invoked in case the Termios layer cannot deal with the IO request.

**Parameters**

| | | |
|---|---|---|
| in | *context* | The Termios device context. |
| in | *request* | The IO control request. |
| in | *buffer* | The IO control buffer. |

Definition at line 231 of file termiostypes.h.

### 9.136.2.3 last_close

```
void(* rtems_termios_device_handler::last_close) (struct rtems_termios_tty *tty, rtems_termios_device_context
*context, rtems_libio_open_close_args_t *args)
```

Last close of this device.

**Parameters**

| | | |
|---|---|---|
| in | *tty* | The Termios control. |
| in | *context* | The Termios device context. |
| in | *args* | The open/close arguments. This is parameter provided to support legacy drivers. It must not be used by new drivers. |

Definition at line 174 of file termiostypes.h.

### 9.136.2.4 poll_read

```
int(* rtems_termios_device_handler::poll_read) (rtems_termios_device_context *context)
```

Polled read.

In case mode is TERMIOS_IRQ_DRIVEN, TERMIOS_IRQ_SERVER_DRIVEN or TERMIOS_TASK_DRIVEN, then data is received via rtems_termios_enqueue_raw_characters().

**Parameters**

| | | |
|---|---|---|
| in | *context* | The Termios device context. |

**Return values**

| | |
|---|---|
| *char* | The received data encoded as unsigned char. |
| *-1* | No data currently available. |

Definition at line 192 of file termiostypes.h.

### 9.136.2.5 set_attributes

```
bool(* rtems_termios_device_handler::set_attributes) (rtems_termios_device_context *context,
const struct termios *term)
```

Set attributes after a Termios settings change.

**Parameters**

| | | |
|---|---|---|
| in | *context* | The Termios device context. |
| in | *term* | The new Termios attributes. |

**Return values**

| | |
|---|---|
| *true* | Successful operation. |
| *false* | Invalid attributes. |

Definition at line 217 of file termiostypes.h.

#### 9.136.2.6 write

```
void(* rtems_termios_device_handler::write) (rtems_termios_device_context *context, const char
*buf, size_t len)
```

Polled write in case mode is TERMIOS_POLLED or write support otherwise.

**Parameters**

| | | |
|---|---|---|
| in | *context* | The Termios device context. |
| in | *buf* | The output buffer. |
| in | *len* | The output buffer length in characters. |

Definition at line 202 of file termiostypes.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/termiostypes.h

## 9.137 rtems_termios_device_node Struct Reference

Termios device node for installed devices.

```
#include <termiostypes.h>
```

### Public Attributes

- rtems_chain_node **node**
- rtems_device_major_number **major**
- rtems_device_minor_number **minor**
- const rtems_termios_device_handler * **handler**
- const rtems_termios_device_flow * **flow**
- rtems_termios_device_context * **context**
- struct rtems_termios_tty * **tty**

### 9.137.1 Detailed Description

Termios device node for installed devices.

**See also**

rtems_termios_device_install().

Definition at line 269 of file termiostypes.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/termiostypes.h

## 9.138 rtems_termios_linesw Struct Reference

### Public Attributes

- int(∗ **l_open** )(struct rtems_termios_tty ∗tp)
- int(∗ **l_close** )(struct rtems_termios_tty ∗tp)
- int(∗ **l_read** )(struct rtems_termios_tty ∗tp, rtems_libio_rw_args_t ∗args)
- int(∗ **l_write** )(struct rtems_termios_tty ∗tp, rtems_libio_rw_args_t ∗args)
- int(∗ **l_rint** )(int c, struct rtems_termios_tty ∗tp)
- int(∗ **l_start** )(struct rtems_termios_tty ∗tp)
- int(∗ **l_ioctl** )(struct rtems_termios_tty ∗tp, rtems_libio_ioctl_args_t ∗args)
- int(∗ **l_modem** )(struct rtems_termios_tty ∗tp, int flags)

### 9.138.1 Detailed Description

Definition at line 479 of file termiostypes.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/termiostypes.h

## 9.139 rtems_termios_rawbuf Struct Reference

### Public Attributes

- char ∗ **theBuf**
- volatile unsigned int **Head**
- volatile unsigned int **Tail**
- volatile unsigned int **Size**
- rtems_binary_semaphore **Semaphore**

### 9.139.1 Detailed Description

Definition at line 51 of file termiostypes.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/termiostypes.h

## 9.140 rtems_termios_tty Struct Reference

### Public Types

- enum { **rob_idle**, **rob_busy**, **rob_wait** }

## Public Attributes

- struct rtems_termios_tty * **forw**
- struct rtems_termios_tty * **back**
- int **refcount**
- rtems_device_major_number **major**
- rtems_device_minor_number **minor**
- rtems_mutex **isem**
- rtems_mutex **osem**
- char * **cbuf**
- int **ccount**
- int **cindex**
- int **column**
- int **read_start_column**
- struct termios **termios**
- rtems_interval **vtimeTicks**
- struct rtems_termios_rawbuf **rawInBuf**
- bool **rawInBufSemaphoreWait**
- rtems_interval **rawInBufSemaphoreTimeout**
- rtems_interval **rawInBufSemaphoreFirstTimeout**
- unsigned int **rawInBufDropped**
- struct rtems_termios_rawbuf **rawOutBuf**
- int **t_dqlen**
- enum rtems_termios_tty:: { ... } **rawOutBufState**
- rtems_termios_callbacks **device**
- rtems_termios_device_context legacy_device_context

  *Context for legacy devices using the callbacks.*
- rtems_termios_device_handler handler

  *The device handler.*
- rtems_termios_device_flow flow

  *The device flow control handler.*
- volatile unsigned int **flow_ctrl**
- unsigned int **lowwater**
- unsigned int **highwater**
- rtems_id **rxTaskId**
- rtems_id **txTaskId**
- int **t_line**
- void * **t_sc**
- struct ttywakeup **tty_snd**
- struct ttywakeup **tty_rcv**
- bool **tty_rcvwakeup**
- rtems_termios_device_node * device_node

  *Corresponding device node.*
- rtems_termios_device_context * device_context

  *Context for device driver.*

### 9.140.1 Detailed Description

Definition at line 283 of file termiostypes.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/termiostypes.h

## 9.141 rtems_test_parallel_context Struct Reference

Internal context for parallel job execution.

`#include <test-info.h>`

### Public Attributes

- Atomic_Ulong **stop**
- SMP_barrier_Control **barrier**
- size_t **worker_count**
- rtems_id **worker_ids** [RTEMS_TEST_PARALLEL_PROCESSOR_MAX]
- rtems_id **stop_worker_timer_id**
- const struct rtems_test_parallel_job * **jobs**
- size_t **job_count**

### 9.141.1 Detailed Description

Internal context for parallel job execution.

Definition at line 129 of file test-info.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/test-info.h

## 9.142 rtems_test_parallel_job Struct Reference

Basic parallel job description.

`#include <test-info.h>`

### Public Attributes

- rtems_interval(* init )(rtems_test_parallel_context *ctx, void *arg, size_t active_workers)
  *Job initialization handler.*
- void(* body )(rtems_test_parallel_context *ctx, void *arg, size_t active_workers, size_t worker_index)
  *Job body handler.*
- void(* fini )(rtems_test_parallel_context *ctx, void *arg, size_t active_workers)
  *Job finalization handler.*
- void * arg
  *Job specific argument.*
- bool cascade
  *Job cascading flag.*

### 9.142.1 Detailed Description

Basic parallel job description.

Definition at line 158 of file test-info.h.

### 9.142.2 Member Data Documentation

#### 9.142.2.1 body

```
void(* rtems_test_parallel_job::body) (rtems_test_parallel_context *ctx, void *arg, size_↩
t active_workers, size_t worker_index)
```

Job body handler.

**Parameters**

| | | |
|---|---|---|
| in | *ctx* | The parallel context. |
| in | *arg* | The user specified argument. |
| in | *active_workers* | Count of active workers. Depends on the cascade option. |
| in | *worker_index* | The worker index. It ranges from 0 to the processor count minus one. |

Definition at line 189 of file test-info.h.

#### 9.142.2.2 cascade

```
bool rtems_test_parallel_job::cascade
```

Job cascading flag.

This flag indicates whether the job should be executed in a cascaded manner (the job is executed on one processor first, two processors afterwards and incremented step by step until all processors are used).

Definition at line 225 of file test-info.h.

#### 9.142.2.3 fini

```
void(* rtems_test_parallel_job::fini) (rtems_test_parallel_context *ctx, void *arg, size_↩
t active_workers)
```

Job finalization handler.

This handler executes only in the context of the master worker after the job body handler.

**Parameters**

| | | |
|---|---|---|
| in | *ctx* | The parallel context. |
| in | *arg* | The user specified argument. |
| in | *active_workers* | Count of active workers. Depends on the cascade option. |

Definition at line 207 of file test-info.h.

#### 9.142.2.4 init

```
rtems_interval(* rtems_test_parallel_job::init) (rtems_test_parallel_context *ctx, void *arg,
size_t active_workers)
```

Job initialization handler.

This handler executes only in the context of the master worker before the job body handler.

**Parameters**

| | | |
|---|---|---|
| in | *ctx* | The parallel context. |
| in | *arg* | The user specified argument. |
| in | *active_workers* | Count of active workers. Depends on the cascade option. |

**Returns**

The desired job body execution time in clock ticks. See rtems_test_parallel_stop_job().

Definition at line 173 of file test-info.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/test-info.h

## 9.143 rtems_time_of_day Struct Reference

This type represents Classic API calendar times.

```
#include <types.h>
```

### Public Attributes

- uint32_t year

  *This member contains the year A.D.*
- uint32_t month

  *This member contains the month of the year with values from 1 to 12.*
- uint32_t day

  *This member contains the day of the month with values from 1 to 31.*
- uint32_t hour

  *This member contains the hour of the day with values from 0 to 23.*
- uint32_t minute

  *This member contains the minute of the hour with values from 0 to 59.*
- uint32_t second

  *This member contains the second of the minute with values from 0 to 59.*
- uint32_t ticks

  *This member contains the clock tick of the second with values from 0 to rtems_clock_get_ticks_per_second() minus one.*

### 9.143.1 Detailed Description

This type represents Classic API calendar times.

Definition at line 251 of file types.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/types.h

## 9.144 rtems_timecounter_simple Struct Reference

Simple timecounter to support legacy clock drivers.

`#include <timecounter.h>`

### Public Attributes

- struct timecounter **tc**
- uint64_t **scaler**
- uint32_t **real_interval**
- uint32_t **binary_interval**

### 9.144.1 Detailed Description

Simple timecounter to support legacy clock drivers.

Definition at line 91 of file timecounter.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/timecounter.h

# 9.145 rtems_timer_information Struct Reference

%

`#include <timer.h>`

## Public Attributes

- Timer_Classes the_class

  *This member is.*
- Watchdog_Interval initial

  *This member is.*
- Watchdog_Interval start_time

  *This member is.*
- Watchdog_Interval stop_time

  *This member is.*

## 9.145.1 Detailed Description

%

Definition at line 217 of file timer.h.

## 9.145.2 Member Data Documentation

### 9.145.2.1 initial

`Watchdog_Interval rtems_timer_information::initial`

This member is.

%

Definition at line 230 of file timer.h.

### 9.145.2.2 start_time

`Watchdog_Interval rtems_timer_information::start_time`

This member is.

%

Definition at line 237 of file timer.h.

#### 9.145.2.3 stop_time

```
Watchdog_Interval rtems_timer_information::stop_time
```

This member is.

%

Definition at line 244 of file timer.h.

#### 9.145.2.4 the_class

```
Timer_Classes rtems_timer_information::the_class
```

This member is.

%

Definition at line 223 of file timer.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/timer.h

## 9.146 RtemsEventReqSendReceive_Context Struct Reference

Test context for spec:/rtems/event/req/send-receive test case.

### Public Attributes

- SenderTypes sender_type

  *This member defines the sender type to perform the event send action.*
- Priorities sender_prio

  *This member defines the sender task priority.*
- rtems_id receiver_id

  *This member defines the receiver ID used for the event send action.*
- rtems_event_set events_to_send

  *This member defines the events to send for the event send action.*
- rtems_status_code send_status

  *This member contains the status of the event send action.*
- ReceiveTypes receive_type

  *This member contains the scheduler ID of the runner task.*
- rtems_option receive_option_set

  *This member defines the option set used for the event receive action.*
- rtems_interval receive_timeout

  *This member defines the timeout used for the event receive action.*
- rtems_event_set received_events

*This member contains the events received by the event receive action.*

- rtems_status_code receive_status

  *This member contains the status of the event receive action.*

- ReceiveConditionStates receive_condition_state

  *This member contains the event conditon state of the receiver task after the event send action.*

- rtems_event_set unsatisfied_pending

  *This member contains the pending events after an event send action which did not satsify the event condition of the receiver.*

- Thread_Control * runner_thread

  *This member contains the TCB of the runner task.*

- rtems_id runner_id

  *This member contains the ID of the runner task.*

- rtems_id worker_id

  *This member contains the task ID of the worker task.*

- rtems_id worker_wakeup

  *This member contains the ID of the semaphore used to wake up the worker task.*

- rtems_id runner_wakeup

  *This member contains the ID of the semaphore used to wake up the runner task.*

- rtems_id runner_sched

  *This member contains the scheduler ID of scheduler used by the runner task.*

- rtems_id other_sched

  *This member contains the scheduler ID of another scheduler which is not used by the runner task.*

- T_thread_switch_log_4 thread_switch_log

  *This member contains the thread switch log.*

- rtems_status_code(* send )(rtems_id, rtems_event_set)

  *This member contains a copy of the corresponding RtemsEventReqSendReceive_Run() parameter.*

- rtems_status_code(* receive )(rtems_event_set, rtems_option, rtems_interval, rtems_event_set *)

  *This member contains a copy of the corresponding RtemsEventReqSendReceive_Run() parameter.*

- rtems_event_set(* get_pending_events )(Thread_Control *)

  *This member contains a copy of the corresponding RtemsEventReqSendReceive_Run() parameter.*

- unsigned int wait_class

  *This member contains a copy of the corresponding RtemsEventReqSendReceive_Run() parameter.*

- int waiting_for_event

  *This member contains a copy of the corresponding RtemsEventReqSendReceive_Run() parameter.*

- size_t pcs [4]

  *This member defines the pre-condition states for the next action.*

- bool in_action_loop

  *This member indicates if the test action loop is currently executed.*

## 9.146.1 Detailed Description

Test context for spec:/rtems/event/req/send-receive test case.

Definition at line 97 of file tr-event-send-receive.c.

The documentation for this struct was generated from the following file:

- testsuites/validation/tr-event-send-receive.c

## 9.147 RtemsMessageReqConstructErrors_Context Struct Reference

Test context for spec:/rtems/message/req/construct-errors test case.

### Public Attributes

- rtems_status_code **status**
- rtems_message_queue_config **config**
- rtems_id * **id**
- rtems_id **id_value**
- Chain_Control **message_queues**
- size_t pcs [7]

  *This member defines the pre-condition states for the next action.*
- bool in_action_loop

  *This member indicates if the test action loop is currently executed.*

### 9.147.1 Detailed Description

Test context for spec:/rtems/message/req/construct-errors test case.

Definition at line 126 of file tc-message-construct-errors.c.

The documentation for this struct was generated from the following file:

- testsuites/validation/tc-message-construct-errors.c

## 9.148 RtemsPartReqCreate_Context Struct Reference

Test context for spec:/rtems/part/req/create test case.

### Public Attributes

- rtems_status_code **status**
- rtems_name **name**
- void * **starting_address**
- uintptr_t **length**
- size_t **buffer_size**
- rtems_attribute **attribute_set**
- rtems_id * **id**
- rtems_id **id_value**
- Chain_Control **partitions**
- size_t pcs [6]

  *This member defines the pre-condition states for the next action.*
- bool in_action_loop

  *This member indicates if the test action loop is currently executed.*

### 9.148.1 Detailed Description

Test context for spec:/rtems/part/req/create test case.

Definition at line 119 of file tc-part-create.c.

The documentation for this struct was generated from the following file:

- testsuites/validation/tc-part-create.c

## 9.149 RtemsPartReqDelete_Context Struct Reference

Test context for spec:/rtems/part/req/delete test case.

### Public Attributes

- rtems_status_code **status**
- rtems_id **id**
- rtems_id **id_value**
- void * **buffer**
- size_t pcs [2]

  *This member defines the pre-condition states for the next action.*
- bool in_action_loop

  *This member indicates if the test action loop is currently executed.*

### 9.149.1 Detailed Description

Test context for spec:/rtems/part/req/delete test case.

Definition at line 86 of file tc-part-delete.c.

The documentation for this struct was generated from the following file:

- testsuites/validation/tc-part-delete.c

## 9.150 RtemsPartReqGetBuffer_Context Struct Reference

Test context for spec:/rtems/part/req/get-buffer test case.

### Public Attributes

- rtems_status_code **status**
- rtems_id **id**
- rtems_id **id_value**
- void ** **buffer**
- void * **buffer_pointer**
- void * **stolen_buffer**
- size_t pcs [3]

  *This member defines the pre-condition states for the next action.*
- bool in_action_loop

  *This member indicates if the test action loop is currently executed.*

### 9.150.1 Detailed Description

Test context for spec:/rtems/part/req/get-buffer test case.

Definition at line 93 of file tc-part-get.c.

The documentation for this struct was generated from the following file:

- testsuites/validation/tc-part-get.c

## 9.151 RtemsPartReqReturnBuffer_Context Struct Reference

Test context for spec:/rtems/part/req/return-buffer test case.

### Public Attributes

- rtems_status_code **status**
- rtems_id **id**
- rtems_id **id_value**
- void * **buffer**
- void * **buffer_in_use**
- size_t pcs [2]

  *This member defines the pre-condition states for the next action.*
- bool in_action_loop

  *This member indicates if the test action loop is currently executed.*

### 9.151.1 Detailed Description

Test context for spec:/rtems/part/req/return-buffer test case.

Definition at line 87 of file tc-part-return.c.

The documentation for this struct was generated from the following file:

- testsuites/validation/tc-part-return.c

## 9.152 RtemsReqIdent_Context Struct Reference

Test context for spec:/rtems/req/ident test case.

### Public Attributes

- rtems_status_code **status**
- rtems_name **name**
- uint32_t **node**
- rtems_id * **id**
- rtems_id **id_value**
- rtems_id **id_remote_object**
- rtems_id id_local_object

  *This member contains a copy of the corresponding RtemsReqIdent_Run() parameter.*
- rtems_status_code(* action )(rtems_name, uint32_t, rtems_id *)

  *This member contains a copy of the corresponding RtemsReqIdent_Run() parameter.*
- size_t pcs [3]

  *This member defines the pre-condition states for the next action.*
- bool in_action_loop

  *This member indicates if the test action loop is currently executed.*

#### 9.152.1 Detailed Description

Test context for spec:/rtems/req/ident test case.

Definition at line 67 of file tr-object-ident.c.

The documentation for this struct was generated from the following file:

- testsuites/validation/tr-object-ident.c

## 9.153 RtemsReqIdentLocal_Context Struct Reference

Test context for spec:/rtems/req/ident-local test case.

### Public Attributes

- rtems_status_code **status**
- rtems_name **name**
- rtems_id * **id**
- rtems_id **id_value**
- rtems_id id_local_object

  *This member contains a copy of the corresponding RtemsReqIdentLocal_Run() parameter.*
- rtems_status_code(* action )(rtems_name, rtems_id *)

  *This member contains a copy of the corresponding RtemsReqIdentLocal_Run() parameter.*
- size_t pcs [2]

  *This member defines the pre-condition states for the next action.*
- bool in_action_loop

  *This member indicates if the test action loop is currently executed.*

### 9.153.1 Detailed Description

Test context for spec:/rtems/req/ident-local test case.

Definition at line 67 of file tr-object-ident-local.c.

The documentation for this struct was generated from the following file:

- testsuites/validation/tr-object-ident-local.c

## 9.154 RtemsTaskReqConstructErrors_Context Struct Reference

Test context for spec:/rtems/task/req/construct-errors test case.

### Public Attributes

- rtems_status_code **status**
- rtems_task_config **config**
- rtems_id * **id**
- rtems_id **id_value**
- bool **create_extension_status**
- size_t **stack_size**
- rtems_id **extension_id**
- Chain_Control **tasks**
- size_t pcs [8]

  *This member defines the pre-condition states for the next action.*
- bool in_action_loop

  *This member indicates if the test action loop is currently executed.*

### 9.154.1 Detailed Description

Test context for spec:/rtems/task/req/construct-errors test case.

Definition at line 131 of file tc-task-construct-errors.c.

The documentation for this struct was generated from the following file:

- testsuites/validation/tc-task-construct-errors.c

## 9.155 RtemsTaskReqIdent_Context Struct Reference

Test context for spec:/rtems/task/req/ident test case.

### Public Attributes

- rtems_status_code **status**
- rtems_id * **id**
- rtems_id **id_value**
- rtems_id **id_local_object**
- size_t pcs [1]

  *This member defines the pre-condition states for the next action.*
- bool in_action_loop

  *This member indicates if the test action loop is currently executed.*

### 9.155.1 Detailed Description

Test context for spec:/rtems/task/req/ident test case.

Definition at line 79 of file tc-task-ident.c.

The documentation for this struct was generated from the following file:

- testsuites/validation/tc-task-ident.c

## 9.156 Scheduler_Assignment Struct Reference

Scheduler assignment.

```
#include <scheduler.h>
```

### Public Attributes

- const Scheduler_Control * scheduler

  *The scheduler for this processor.*
- uint32_t attributes

  *The scheduler assignment attributes.*

### 9.156.1 Detailed Description

Scheduler assignment.

Definition at line 344 of file scheduler.h.

### 9.156.2 Member Data Documentation

#### 9.156.2.1 attributes

```
uint32_t Scheduler_Assignment::attributes
```

The scheduler assignment attributes.

Use SCHEDULER_ASSIGN_DEFAULT to select default attributes.

The presence of a processor can be

- SCHEDULER_ASSIGN_PROCESSOR_OPTIONAL, or
- SCHEDULER_ASSIGN_PROCESSOR_MANDATORY.

Definition at line 359 of file scheduler.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/scheduler.h

## 9.157 Scheduler_Context Struct Reference

Scheduler context.

```
#include <scheduler.h>
```

### Public Attributes

- Processor_mask Processors
  *Lock to protect this scheduler instance.*

### 9.157.1 Detailed Description

Scheduler context.

The scheduler context of a particular scheduler implementation must place this structure at the begin of its context structure.

Definition at line 247 of file scheduler.h.

### 9.157.2 Member Data Documentation

#### 9.157.2.1 Processors

```
Processor_mask Scheduler_Context::Processors
```

Lock to protect this scheduler instance.

The set of processors owned by this scheduler instance.

Definition at line 257 of file scheduler.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/scheduler.h

## 9.158 Scheduler_EDF_Context Struct Reference

### Public Attributes

- Scheduler_Context Base
  *Basic scheduler context.*
- RBTree_Control Ready

### 9.158.1 Detailed Description

Definition at line 75 of file scheduleredf.h.

### 9.158.2 Member Data Documentation

#### 9.158.2.1 Ready

```
RBTree_Control Scheduler_EDF_Context::Ready
```

Top of the ready queue.

Definition at line 84 of file scheduleredf.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/scheduleredf.h

## 9.159 Scheduler_EDF_Node Struct Reference

Scheduler node specialization for EDF schedulers.

```
#include <scheduleredf.h>
```

### Public Attributes

- Scheduler_Node Base
  *Basic scheduler node.*
- RBTree_Node Node
- Priority_Control priority
  *The thread priority currently used for this scheduler instance.*

### 9.159.1 Detailed Description

Scheduler node specialization for EDF schedulers.

Definition at line 90 of file scheduleredf.h.

### 9.159.2 Member Data Documentation

#### 9.159.2.1 Node

```
RBTree_Node Scheduler_EDF_Node::Node
```

Rbtree node related to this thread.

Definition at line 99 of file scheduleredf.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/scheduleredf.h

## 9.160 Scheduler_EDF_SMP_Context Struct Reference

### Public Attributes

- Scheduler_SMP_Context **Base**
- int64_t generations [2]
  *Current generation for LIFO (index 0) and FIFO (index 1) ordering.*
- Chain_Control Affine_queues
  *Chain of ready queues with affine threads to determine the highest priority ready thread.*
- Scheduler_EDF_SMP_Ready_queue Ready [RTEMS_ZERO_LENGTH_ARRAY]
  *A table with ready queues.*

### 9.160.1 Detailed Description

Definition at line 85 of file scheduleredfsmp.h.

### 9.160.2 Member Data Documentation

#### 9.160.2.1 Ready

```
Scheduler_EDF_SMP_Ready_queue Scheduler_EDF_SMP_Context::Ready[RTEMS_ZERO_LENGTH_ARRAY]
```

A table with ready queues.

The index zero queue is used for threads with a one-to-all processor affinity. Index one corresponds to processor index zero, and so on.

Definition at line 105 of file scheduleredfsmp.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/scheduleredfsmp.h

## 9.161 Scheduler_EDF_SMP_Node Struct Reference

### Public Attributes

- Scheduler_SMP_Node **Base**
- int64_t generation

  *Generation number to ensure FIFO/LIFO order for threads of the same priority across different ready queues.*
- uint8_t ready_queue_index

  *The ready queue index depending on the processor affinity and pinning of the thread.*
- uint8_t affinity_ready_queue_index

  *Ready queue index according to thread affinity.*
- uint8_t pinning_ready_queue_index

  *Ready queue index according to thread pinning.*

### 9.161.1 Detailed Description

Definition at line 38 of file scheduleredfsmp.h.

### 9.161.2 Member Data Documentation

#### 9.161.2.1 ready_queue_index

```
uint8_t Scheduler_EDF_SMP_Node::ready_queue_index
```

The ready queue index depending on the processor affinity and pinning of the thread.

The ready queue index zero is used for threads with a one-to-all thread processor affinity. Threads with a one-to-one processor affinity use the processor index plus one as the ready queue index.

Definition at line 55 of file scheduleredfsmp.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/scheduleredfsmp.h

## 9.162 Scheduler_EDF_SMP_Ready_queue Struct Reference

### Public Attributes

- Chain_Node Node

  *Chain node for Scheduler_SMP_Context::Affine_queues.*
- RBTree_Control Queue

  *The ready threads of the corresponding affinity.*
- Scheduler_EDF_SMP_Node * scheduled

  *The scheduled thread of the corresponding processor.*

### 9.162.1 Detailed Description

Definition at line 68 of file scheduleredfsmp.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/scheduleredfsmp.h

## 9.163 Scheduler_Node Struct Reference

Scheduler node for per-thread data.

```
#include <schedulernode.h>
```

## Public Attributes

- union {
  Chain_Node **Chain**
  RBTree_Node **RBTree**
  } Node

  *Chain node for usage in various scheduler data structures.*
- int sticky_level

  *The sticky level determines if this scheduler node should use an idle thread in case this node is scheduled and the owner thread is blocked.*
- struct _Thread_Control * user

  *The thread using this node.*
- struct _Thread_Control * idle

  *The idle thread claimed by this node in case the sticky level is greater than zero and the thread is block or is scheduled on another scheduler instance.*
- struct _Thread_Control * owner

  *The thread owning this node.*
- struct {
  Chain_Node Wait_node
  *Node to add this scheduler node to Thread_Control::Scheduler::Wait_nodes.*
  union {
  Chain_Node Chain
  *The node for Thread_Control::Scheduler::Scheduler_nodes.*
  Scheduler_Node * next
  *The next pointer for a temporary remove list.*
  } Scheduler_node
  *Node to add this scheduler node to Thread_Control::Scheduler::Scheduler_nodes or a temporary remove list.*
  Scheduler_Node * next_request
  *Link to the next scheduler node in the Thread_Control::Scheduler::requests list.*
  Scheduler_Node_request request
  *The current scheduler node request.*
  } Thread

  *Block to register and manage this scheduler node in the thread control block of the owner of this scheduler node.*
- struct {
  Priority_Aggregation **Priority**
  } Wait

  *Thread wait support block.*
- struct {
  Priority_Control value
  *The thread priority value of this scheduler node.*
  SMP_sequence_lock_Control Lock
  *Sequence lock to synchronize priority value updates.*
  } Priority

  *The thread priority information used by the scheduler.*

### 9.163.1 Detailed Description

Scheduler node for per-thread data.

Definition at line 79 of file schedulernode.h.

## 9.163.2 Member Data Documentation

### 9.163.2.1 idle

```
struct _Thread_Control* Scheduler_Node::idle
```

The idle thread claimed by this node in case the sticky level is greater than zero and the thread is block or is scheduled on another scheduler instance.

This is necessary to ensure the priority ceiling protocols work across scheduler boundaries.

Definition at line 117 of file schedulernode.h.

### 9.163.2.2 next

```
Scheduler_Node* Scheduler_Node::next
```

The next pointer for a temporary remove list.

**See also**

_Thread_Scheduler_process_requests().

Definition at line 152 of file schedulernode.h.

### 9.163.2.3 Node

```
union { ... } Scheduler_Node::Node
```

Chain node for usage in various scheduler data structures.

Strictly, this is the wrong place for this field since the data structures to manage scheduler nodes belong to the particular scheduler implementation. Currently, all SMP scheduler implementations use chains or red-black trees. The node is here to simplify things, just like the object node in the thread control block.

### 9.163.2.4 Priority

```
struct { ... } Scheduler_Node::Priority
```

The thread priority information used by the scheduler.

The thread priority is manifest in two independent areas. One area is the user visible thread priority along with a potential thread queue. The other is the scheduler. During a thread priority change, the user visible thread priority and the thread queue are first updated and the thread priority value here is changed. Once this is done the scheduler is notified via the update priority operation, so that it can update its internal state and honour a new thread priority value.

#### 9.163.2.5 user

```
struct _Thread_Control* Scheduler_Node::user
```

The thread using this node.

This is either the owner or an idle thread.

Definition at line 107 of file schedulernode.h.

#### 9.163.2.6 value

```
Priority_Control Scheduler_Node::value
```

The thread priority value of this scheduler node.

The producer of this value is _Thread_Change_priority(). The consumer is the scheduler via the unblock and update priority operations.

This priority control consists of two parts. One part is the plain priority value (most-significant 63 bits). The other part is the least-significant bit which indicates if the thread should be appended (bit set) or prepended (bit cleared) to its priority group, see SCHEDULER_PRIORITY_APPEND().

Definition at line 199 of file schedulernode.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/schedulernode.h

## 9.164 Scheduler_Operations Struct Reference

The scheduler operations.

```
#include <scheduler.h>
```

## Public Attributes

- void(∗ initialize )(const Scheduler_Control ∗)
- void(∗ schedule )(const Scheduler_Control ∗, Thread_Control ∗)
- void(∗ yield )(const Scheduler_Control ∗, Thread_Control ∗, Scheduler_Node ∗)
- void(∗ block )(const Scheduler_Control ∗, Thread_Control ∗, Scheduler_Node ∗)
- void(∗ unblock )(const Scheduler_Control ∗, Thread_Control ∗, Scheduler_Node ∗)
- void(∗ update_priority )(const Scheduler_Control ∗, Thread_Control ∗, Scheduler_Node ∗)
- Priority_Control(∗ map_priority )(const Scheduler_Control ∗, Priority_Control)
- Priority_Control(∗ unmap_priority )(const Scheduler_Control ∗, Priority_Control)
- bool(∗ ask_for_help )(const Scheduler_Control ∗scheduler, Thread_Control ∗the_thread, Scheduler_Node ∗node)

  *Ask for help operation.*
- void(∗ reconsider_help_request )(const Scheduler_Control ∗scheduler, Thread_Control ∗the_thread, Scheduler_Node ∗node)

  *Reconsider help operation.*
- void(∗ withdraw_node )(const Scheduler_Control ∗scheduler, Thread_Control ∗the_thread, Scheduler_Node ∗node, Thread_Scheduler_state next_state)

  *Withdraw node operation.*
- void(∗ pin )(const Scheduler_Control ∗scheduler, Thread_Control ∗the_thread, Scheduler_Node ∗node, struct Per_CPU_Control ∗cpu)

  *Pin thread operation.*
- void(∗ unpin )(const Scheduler_Control ∗scheduler, Thread_Control ∗the_thread, Scheduler_Node ∗node, struct Per_CPU_Control ∗cpu)

  *Unpin thread operation.*
- void(∗ add_processor )(const Scheduler_Control ∗scheduler, Thread_Control ∗idle)

  *Add processor operation.*
- Thread_Control ∗(∗ remove_processor )(const Scheduler_Control ∗scheduler, struct Per_CPU_Control ∗cpu)

  *Remove processor operation.*
- void(∗ node_initialize )(const Scheduler_Control ∗, Scheduler_Node ∗, Thread_Control ∗, Priority_Control)
- void(∗ node_destroy )(const Scheduler_Control ∗, Scheduler_Node ∗)
- void(∗ release_job )(const Scheduler_Control ∗, Thread_Control ∗, Priority_Node ∗, uint64_t, Thread_queue_Context ∗)
- void(∗ cancel_job )(const Scheduler_Control ∗, Thread_Control ∗, Priority_Node ∗, Thread_queue_Context ∗)
- void(∗ tick )(const Scheduler_Control ∗, Thread_Control ∗)
- void(∗ start_idle )(const Scheduler_Control ∗, Thread_Control ∗, struct Per_CPU_Control ∗)
- bool(∗ set_affinity )(const Scheduler_Control ∗, Thread_Control ∗, Scheduler_Node ∗, const Processor_mask ∗)

### 9.164.1 Detailed Description

The scheduler operations.

Definition at line 43 of file scheduler.h.

### 9.164.2 Member Data Documentation

#### 9.164.2.1 add_processor

```
void( * Scheduler_Operations::add_processor) (const Scheduler_Control *scheduler, Thread_Control
*idle)
```

Add processor operation.

**Parameters**

| | | |
|---|---|---|
| in | *scheduler* | The scheduler instance to add the processor. |
| in | *idle* | The idle thread of the processor to add. |

Definition at line 173 of file scheduler.h.

### 9.164.2.2 ask_for_help

```
bool( * Scheduler_Operations::ask_for_help) (const Scheduler_Control *scheduler, Thread_Control
*the_thread, Scheduler_Node *node)
```

Ask for help operation.

**Parameters**

| | | |
|---|---|---|
| in | *scheduler* | The scheduler instance to ask for help. |
| in | *the_thread* | The thread needing help. |
| in | *node* | The scheduler node. |

**Return values**

| | |
|---|---|
| *true* | Ask for help was successful. |
| *false* | Otherwise. |

Definition at line 101 of file scheduler.h.

### 9.164.2.3 block

```
void( * Scheduler_Operations::block) (const Scheduler_Control *, Thread_Control *, Scheduler_Node
*)
```

**See also**

_Scheduler_Block()

Definition at line 58 of file scheduler.h.

#### 9.164.2.4 cancel_job

```
void( * Scheduler_Operations::cancel_job) (const Scheduler_Control *, Thread_Control *, Priority_Node
*, Thread_queue_Context *)
```

**See also**

_Scheduler_Cancel_job()

Definition at line 213 of file scheduler.h.

#### 9.164.2.5 initialize

```
void( * Scheduler_Operations::initialize) (const Scheduler_Control *)
```

**See also**

_Scheduler_Handler_initialization()

Definition at line 45 of file scheduler.h.

#### 9.164.2.6 map_priority

```
Priority_Control( * Scheduler_Operations::map_priority) (const Scheduler_Control *, Priority_Control)
```

**See also**

_Scheduler_Map_priority()

Definition at line 79 of file scheduler.h.

#### 9.164.2.7 node_destroy

```
void( * Scheduler_Operations::node_destroy) (const Scheduler_Control *, Scheduler_Node *)
```

**See also**

_Scheduler_Node_destroy()

Definition at line 201 of file scheduler.h.

#### 9.164.2.8 node_initialize

```
void( * Scheduler_Operations::node_initialize) (const Scheduler_Control *, Scheduler_Node *,
Thread_Control *, Priority_Control)
```

**See also**

_Scheduler_Node_initialize()

Definition at line 193 of file scheduler.h.

#### 9.164.2.9 pin

```
void( * Scheduler_Operations::pin) (const Scheduler_Control *scheduler, Thread_Control *the_↩
thread, Scheduler_Node *node, struct Per_CPU_Control *cpu)
```

Pin thread operation.

**Parameters**

| | | |
|---|---|---|
| in | *scheduler* | The scheduler instance of the specified processor. |
| in | *the_thread* | The thread to pin. |
| in | *node* | The scheduler node of the thread. |
| in | *cpu* | The processor to pin the thread. |

Definition at line 145 of file scheduler.h.

#### 9.164.2.10 reconsider_help_request

```
void( * Scheduler_Operations::reconsider_help_request) (const Scheduler_Control *scheduler,
Thread_Control *the_thread, Scheduler_Node *node)
```

Reconsider help operation.

**Parameters**

| | | |
|---|---|---|
| in | *scheduler* | The scheduler instance to reconsider the help request. |
| in | *the_thread* | The thread reconsidering a help request. |
| in | *node* | The scheduler node. |

Definition at line 115 of file scheduler.h.

#### 9.164.2.11 release_job

```
void( * Scheduler_Operations::release_job) (const Scheduler_Control *, Thread_Control *, Priority_Node
*, uint64_t, Thread_queue_Context *)
```

**See also**

_Scheduler_Release_job()

Definition at line 204 of file scheduler.h.

#### 9.164.2.12 remove_processor

```
Thread_Control*( * Scheduler_Operations::remove_processor) (const Scheduler_Control *scheduler,
struct Per_CPU_Control *cpu)
```

Remove processor operation.

**Parameters**

| | | |
|---|---|---|
| in | *scheduler* | The scheduler instance to remove the processor. |
| in | *cpu* | The processor to remove. |

**Returns**

The idle thread of the removed processor.

Definition at line 186 of file scheduler.h.

#### 9.164.2.13 schedule

```
void( * Scheduler_Operations::schedule) (const Scheduler_Control *, Thread_Control *)
```

**See also**

_Scheduler_Schedule()

Definition at line 48 of file scheduler.h.

#### 9.164.2.14 set_affinity

```
bool( * Scheduler_Operations::set_affinity) (const Scheduler_Control *, Thread_Control *, Scheduler_Node
*, const Processor_mask *)
```

**See also**

_Scheduler_Set_affinity()

Definition at line 232 of file scheduler.h.

#### 9.164.2.15 start_idle

```
void( * Scheduler_Operations::start_idle) (const Scheduler_Control *, Thread_Control *, struct
Per_CPU_Control *)
```

**See also**

_Scheduler_Start_idle()

Definition at line 224 of file scheduler.h.

#### 9.164.2.16 tick

```
void( * Scheduler_Operations::tick) (const Scheduler_Control *, Thread_Control *)
```

**See also**

_Scheduler_Tick()

Definition at line 221 of file scheduler.h.

#### 9.164.2.17 unblock

```
void( * Scheduler_Operations::unblock) (const Scheduler_Control *, Thread_Control *, Scheduler_Node
*)
```

**See also**

_Scheduler_Unblock()

Definition at line 65 of file scheduler.h.

#### 9.164.2.18 unmap_priority

```
Priority_Control( * Scheduler_Operations::unmap_priority) (const Scheduler_Control *, Priority_Control)
```

**See also**

_Scheduler_Unmap_priority()

Definition at line 85 of file scheduler.h.

#### 9.164.2.19 unpin

```
void( * Scheduler_Operations::unpin) (const Scheduler_Control *scheduler, Thread_Control *the↩
_thread, Scheduler_Node *node, struct Per_CPU_Control *cpu)
```

Unpin thread operation.

**Parameters**

| | | |
|---|---|---|
| in | *scheduler* | The scheduler instance of the specified processor. |
| in | *the_thread* | The thread to unpin. |
| in | *node* | The scheduler node of the thread. |
| in | *cpu* | The processor to unpin the thread. |

Definition at line 160 of file scheduler.h.

#### 9.164.2.20 update_priority

```
void( * Scheduler_Operations::update_priority) (const Scheduler_Control *, Thread_Control *,
Scheduler_Node *)
```

**See also**

_Scheduler_Update_priority()

Definition at line 72 of file scheduler.h.

#### 9.164.2.21 withdraw_node

```
void( * Scheduler_Operations::withdraw_node) (const Scheduler_Control *scheduler, Thread_Control
*the_thread, Scheduler_Node *node, Thread_Scheduler_state next_state)
```

Withdraw node operation.

**Parameters**

| | | |
|---|---|---|
| in | *scheduler* | The scheduler instance to withdraw the node. |
| in | *the_thread* | The thread using the node. |
| in | *node* | The scheduler node to withdraw. |
| in | *next_state* | The next thread scheduler state in case the node is scheduled. |

Definition at line 130 of file scheduler.h.

#### 9.164.2.22 yield

```
void( * Scheduler_Operations::yield) (const Scheduler_Control *, Thread_Control *, Scheduler_Node
*)
```

**See also**

_Scheduler_Yield()

Definition at line 51 of file scheduler.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/scheduler.h

## 9.165 Scheduler_priority_Context Struct Reference

### Public Attributes

- Scheduler_Context Base
  *Basic scheduler context.*
- Priority_bit_map_Control Bit_map
  *Bit map to indicate non-empty ready queues.*
- Chain_Control Ready [0]
  *One ready queue per priority level.*

### 9.165.1 Detailed Description

Definition at line 65 of file schedulerpriority.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/schedulerpriority.h

## 9.166 Scheduler_priority_Node Struct Reference

Scheduler node specialization for Deterministic Priority schedulers.

`#include <schedulerpriority.h>`

### Public Attributes

- Scheduler_Node Base
  *Basic scheduler node.*
- Scheduler_priority_Ready_queue Ready_queue
  *The associated ready queue of this node.*

### 9.166.1 Detailed Description

Scheduler node specialization for Deterministic Priority schedulers.

Definition at line 101 of file schedulerpriority.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/schedulerpriority.h

## 9.167 Scheduler_priority_Ready_queue Struct Reference

Data for ready queue operations.

`#include <schedulerpriority.h>`

### Public Attributes

- unsigned int current_priority
  *The thread priority currently used by the scheduler.*
- Chain_Control * ready_chain
- Priority_bit_map_Information Priority_map

### 9.167.1 Detailed Description

Data for ready queue operations.

Definition at line 85 of file schedulerpriority.h.

### 9.167.2 Member Data Documentation

#### 9.167.2.1 Priority_map

```
Priority_bit_map_Information Scheduler_priority_Ready_queue::Priority_map
```

This field contains precalculated priority map indices.

Definition at line 95 of file schedulerpriority.h.

#### 9.167.2.2 ready_chain

```
Chain_Control* Scheduler_priority_Ready_queue::ready_chain
```

This field points to the Ready FIFO for this thread's priority.

Definition at line 92 of file schedulerpriority.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/schedulerpriority.h

## 9.168 Scheduler_Processor_removal_context Struct Reference

### Public Attributes

- const Scheduler_Control * **scheduler**
- rtems_status_code **status**

### 9.168.1 Detailed Description

Definition at line 24 of file schedulerremoveprocessor.c.

The documentation for this struct was generated from the following file:

- cpukit/rtems/src/schedulerremoveprocessor.c

## 9.169 Scheduler_simple_Context Struct Reference

Simple scheduler context.

```
#include <schedulersimple.h>
```

### Public Attributes

- Scheduler_Context Base
  *Basic scheduler context.*
- Chain_Control Ready
  *One ready queue for all ready threads.*

### 9.169.1 Detailed Description

Simple scheduler context.

Definition at line 68 of file schedulersimple.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/schedulersimple.h

## 9.170 Scheduler_SMP_Context Struct Reference

Scheduler context specialization for SMP schedulers.

```
#include <schedulersmp.h>
```

### Public Attributes

- Scheduler_Context Base
  *Basic scheduler context.*
- Chain_Control Scheduled
  *The chain of scheduled nodes.*
- Chain_Control Idle_threads
  *Chain of the available idle threads.*

### 9.170.1 Detailed Description

Scheduler context specialization for SMP schedulers.

Definition at line 46 of file schedulersmp.h.

### 9.170.2 Member Data Documentation

#### 9.170.2.1 Idle_threads

```
Chain_Control Scheduler_SMP_Context::Idle_threads
```

Chain of the available idle threads.

Idle threads are used for the scheduler helping protocol. It is crucial that the idle threads preserve their relative order. This is the case for this priority based scheduler.

Definition at line 64 of file schedulersmp.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/schedulersmp.h

## 9.171 Scheduler_SMP_Node Struct Reference

Scheduler node specialization for SMP schedulers.

```
#include <schedulersmp.h>
```

### Public Attributes

- Scheduler_Node Base
  *Basic scheduler node.*
- Scheduler_SMP_Node_state state
  *The state of this node.*
- Priority_Control priority
  *The current priority of thread owning this node.*

### 9.171.1 Detailed Description

Scheduler node specialization for SMP schedulers.

Definition at line 100 of file schedulersmp.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/schedulersmp.h

## 9.172 Sem_Control Struct Reference

### Public Attributes

- Thread_queue_Syslock_queue **Queue**
- unsigned int **count**

### 9.172.1 Detailed Description

Definition at line 41 of file semaphoreimpl.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/semaphoreimpl.h

## 9.173 Semaphore_Control Struct Reference

`#include <semdata.h>`

### Public Attributes

- Objects_Control Object
  *The object management portion of a semaphore instance.*
- union {
  Thread_queue_Control Wait_queue
  *The thread queue present in all other variants.*
  CORE_ceiling_mutex_Control Mutex
  CORE_semaphore_Control Semaphore
  MRSP_Control **MRSP**
  } Core_control

### 9.173.1 Detailed Description

The following defines the control block used to manage each semaphore.

Definition at line 40 of file semdata.h.

### 9.173.2 Member Data Documentation

#### 9.173.2.1 Core_control

`union { ... } Semaphore_Control::Core_control`

This contains the memory associated with the SuperCore Semaphore or Mutex instance that provides the primary functionality of each Classic API Semaphore instance. The structure used is dependent on the attributes specified by the user on the create directive.

**Note**

Only one of these has meaning in a particular Classic API Semaphore instance.

#### 9.173.2.2 Mutex

```
CORE_ceiling_mutex_Control Semaphore_Control::Mutex
```

This is the SuperCore Mutex instance associated with this Classic API Semaphore instance.

Definition at line 71 of file semdata.h.

#### 9.173.2.3 Object

```
Objects_Control Semaphore_Control::Object
```

The object management portion of a semaphore instance.

A pointer of the node of active semaphores contains the semaphore flags, see _Semaphore_Get_flags(). The rational for this optimization is a reduction of the semaphore control size in general and the ability to allow a configuration dependent size of the semaphore control block, e.g. for the MrsP semaphores.

Definition at line 50 of file semdata.h.

#### 9.173.2.4 Semaphore

```
CORE_semaphore_Control Semaphore_Control::Semaphore
```

This is the SuperCore Semaphore instance associated with this Classic API Semaphore instance.

Definition at line 77 of file semdata.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/semdata.h

## 9.174 SHA256Context Struct Reference

### Public Attributes

- uint32_t **state** [8]
- uint64_t **count**
- unsigned char **buf** [64]

### 9.174.1 Detailed Description

Definition at line 34 of file sha256.h.

The documentation for this struct was generated from the following file:

- cpukit/include/sha256.h

## 9.175 SMP_barrier_Control Struct Reference

SMP barrier control.

`#include <smpbarrier.h>`

### Public Attributes

- Atomic_Uint **value**
- Atomic_Uint **sense**

### 9.175.1 Detailed Description

SMP barrier control.

Definition at line 51 of file smpbarrier.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/smpbarrier.h

## 9.176 SMP_barrier_State Struct Reference

SMP barrier per-thread state.

`#include <smpbarrier.h>`

### Public Attributes

- unsigned int **sense**

### 9.176.1 Detailed Description

SMP barrier per-thread state.

Each user of the barrier must provide this per-thread state.

Definition at line 61 of file smpbarrier.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/smpbarrier.h

## 9.177 SMP_lock_Context Struct Reference

Local SMP lock context for acquire and release pairs.

`#include <smplock.h>`

### Public Attributes

- ISR_Level **isr_level**

### 9.177.1 Detailed Description

Local SMP lock context for acquire and release pairs.

Definition at line 90 of file smplock.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/smplock.h

## 9.178 SMP_lock_Control Struct Reference

SMP lock control.

`#include <smplock.h>`

### Public Attributes

- SMP_ticket_lock_Control **Ticket_lock**

### 9.178.1 Detailed Description

SMP lock control.

Definition at line 65 of file smplock.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/smplock.h

## 9.179 SMP_MCS_lock_Context Struct Reference

SMP Mellor-Crummey and Scott (MCS) lock context.

`#include <smplockmcs.h>`

## Public Attributes

- union {
  Atomic_Uintptr atomic
  *The next context as an atomic unsigned integer pointer value.*
  struct SMP_MCS_lock_Context * normal
  *The next context as a normal pointer.*
  } next

  *The next context on the queue if it exists.*
- Atomic_Uint locked

  *Indicates if the lock is owned or free in case a previous context exits on the queue.*

### 9.179.1 Detailed Description

SMP Mellor-Crummey and Scott (MCS) lock context.

Definition at line 40 of file smplockmcs.h.

### 9.179.2 Member Data Documentation

#### 9.179.2.1 locked

```
Atomic_Uint SMP_MCS_lock_Context::locked
```

Indicates if the lock is owned or free in case a previous context exits on the queue.

This field is initialized to a non-zero value. The previous lock owner (which is the owner of the previous context) will set it to zero during its lock release.

Definition at line 66 of file smplockmcs.h.

#### 9.179.2.2 normal

```
struct SMP_MCS_lock_Context* SMP_MCS_lock_Context::normal
```

The next context as a normal pointer.

Only provided for debugging purposes.

Definition at line 55 of file smplockmcs.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/smplockmcs.h

# 9.180 SMP_MCS_lock_Control Struct Reference

SMP Mellor-Crummey and Scott (MCS) lock control.

`#include <smplockmcs.h>`

## Public Attributes

- union {
  Atomic_Uintptr atomic
  *The queue tail context as an atomic unsigned integer pointer value.*
  struct SMP_MCS_lock_Context * normal
  *The queue tail context as a normal pointer.*
  } queue

  *The queue tail context.*

## 9.180.1 Detailed Description

SMP Mellor-Crummey and Scott (MCS) lock control.

Definition at line 78 of file smplockmcs.h.

## 9.180.2 Member Data Documentation

### 9.180.2.1 normal

`struct SMP_MCS_lock_Context* SMP_MCS_lock_Control::normal`

The queue tail context as a normal pointer.

Only provided for debugging purposes.

Definition at line 97 of file smplockmcs.h.

### 9.180.2.2 queue

`union { ... } SMP_MCS_lock_Control::queue`

The queue tail context.

The lock is free, in case this field is zero, otherwise it is locked by the owner of the queue head.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/smplockmcs.h

## 9.181 SMP_Multicast_jobs Struct Reference

### Public Attributes

- Per_CPU_Job_context **Context**
- Per_CPU_Job **Jobs** [CPU_MAXIMUM_PROCESSORS]

### 9.181.1 Detailed Description

Definition at line 135 of file smpmulticastaction.c.

The documentation for this struct was generated from the following file:

- cpukit/score/src/smpmulticastaction.c

## 9.182 SMP_sequence_lock_Control Struct Reference

SMP sequence lock control.

`#include <smplockseq.h>`

### Public Attributes

- Atomic_Uint sequence

  *The sequence number.*

### 9.182.1 Detailed Description

SMP sequence lock control.

The sequence lock offers a consistent data set for readers in the presence of at most one concurrent writer. Due to the read-modify-write operation in _SMP_sequence_lock_Read_retry() the data corresponding to the last written sequence number is observed. To allow multiple writers an additional SMP lock is necessary to serialize writes.

See also Hans-J. Boehm, HP Laboratories, "Can Seqlocks Get Along With Programming Language Memory Models?", http://www.hpl.hp.com/techreports/2012/HPL-2012-68.pdf

Definition at line 50 of file smplockseq.h.

### 9.182.2 Member Data Documentation

#### 9.182.2.1 sequence

```
Atomic_Uint SMP_sequence_lock_Control::sequence
```

The sequence number.

An odd value indicates that a write is in progress.

Definition at line 56 of file smplockseq.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/smplockseq.h

## 9.183 SMP_ticket_lock_Control Struct Reference

SMP ticket lock control.

```
#include <smplockticket.h>
```

### Public Attributes

- Atomic_Uint **next_ticket**
- Atomic_Uint **now_serving**

### 9.183.1 Detailed Description

SMP ticket lock control.

Definition at line 40 of file smplockticket.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/smplockticket.h

## 9.184 SPARC_Counter Struct Reference

### Public Attributes

- SPARC_Counter_read **read_isr_disabled**
- SPARC_Counter_read **read**
- volatile const CPU_Counter_ticks * **counter_register**
- volatile const uint32_t * **pending_register**
- uint32_t **pending_mask**
- CPU_Counter_ticks **accumulated**
- CPU_Counter_ticks **interval**

### 9.184.1 Detailed Description

Definition at line 1091 of file cpu.h.

The documentation for this struct was generated from the following file:

- cpukit/score/cpu/sparc/include/rtems/score/cpu.h

## 9.185 SPARC_Minimum_stack_frame Struct Reference

`#include <cpu.h>`

### Public Attributes

- uint32_t l0
- uint32_t l1
- uint32_t l2
- uint32_t l3
- uint32_t l4
- uint32_t l5
- uint32_t l6
- uint32_t l7
- uint32_t i0
- uint32_t i1
- uint32_t i2
- uint32_t i3
- uint32_t i4
- uint32_t i5
- uint32_t i6_fp
- uint32_t i7
- void * structure_return_address
- uint32_t saved_arg0
- uint32_t saved_arg1
- uint32_t saved_arg2
- uint32_t saved_arg3
- uint32_t saved_arg4
- uint32_t saved_arg5
- uint32_t pad0

### 9.185.1 Detailed Description

This structure represents the organization of the minimum stack frame for the SPARC. More framing information is required in certain situaions such as when there are a large number of out parameters or when the callee must save floating point registers.

Definition at line 170 of file cpu.h.

## 9.185.2 Member Data Documentation

### 9.185.2.1 i0

```
uint32_t SPARC_Minimum_stack_frame::i0
```

This is the offset of the l0 register.

Definition at line 188 of file cpu.h.

### 9.185.2.2 i1

```
uint32_t SPARC_Minimum_stack_frame::i1
```

This is the offset of the i1 register.

Definition at line 190 of file cpu.h.

### 9.185.2.3 i2

```
uint32_t SPARC_Minimum_stack_frame::i2
```

This is the offset of the i2 register.

Definition at line 192 of file cpu.h.

### 9.185.2.4 i3

```
uint32_t SPARC_Minimum_stack_frame::i3
```

This is the offset of the i3 register.

Definition at line 194 of file cpu.h.

### 9.185.2.5 i4

```
uint32_t SPARC_Minimum_stack_frame::i4
```

This is the offset of the i4 register.

Definition at line 196 of file cpu.h.

#### 9.185.2.6 i5

```
uint32_t SPARC_Minimum_stack_frame::i5
```

This is the offset of the i5 register.

Definition at line 198 of file cpu.h.

#### 9.185.2.7 i6_fp

```
uint32_t SPARC_Minimum_stack_frame::i6_fp
```

This is the offset of the i6 register.

Definition at line 200 of file cpu.h.

#### 9.185.2.8 i7

```
uint32_t SPARC_Minimum_stack_frame::i7
```

This is the offset of the i7 register.

Definition at line 202 of file cpu.h.

#### 9.185.2.9 l0

```
uint32_t SPARC_Minimum_stack_frame::l0
```

This is the offset of the l0 register.

Definition at line 172 of file cpu.h.

#### 9.185.2.10 l1

```
uint32_t SPARC_Minimum_stack_frame::l1
```

This is the offset of the l1 register.

Definition at line 174 of file cpu.h.

#### 9.185.2.11 l2

```
uint32_t SPARC_Minimum_stack_frame::l2
```

This is the offset of the l2 register.

Definition at line 176 of file cpu.h.

#### 9.185.2.12 l3

```
uint32_t SPARC_Minimum_stack_frame::l3
```

This is the offset of the l3 register.

Definition at line 178 of file cpu.h.

#### 9.185.2.13 l4

```
uint32_t SPARC_Minimum_stack_frame::l4
```

This is the offset of the l4 register.

Definition at line 180 of file cpu.h.

#### 9.185.2.14 l5

```
uint32_t SPARC_Minimum_stack_frame::l5
```

This is the offset of the l5 register.

Definition at line 182 of file cpu.h.

#### 9.185.2.15 l6

```
uint32_t SPARC_Minimum_stack_frame::l6
```

This is the offset of the l6 register.

Definition at line 184 of file cpu.h.

#### 9.185.2.16 l7

```
uint32_t SPARC_Minimum_stack_frame::l7
```

This is the offset of the l7 register.

Definition at line 186 of file cpu.h.

#### 9.185.2.17 pad0

```
uint32_t SPARC_Minimum_stack_frame::pad0
```

This field pads the structure so ldd and std instructions can be used.

Definition at line 223 of file cpu.h.

#### 9.185.2.18 saved_arg0

```
uint32_t SPARC_Minimum_stack_frame::saved_arg0
```

This is the offset of the register for saved argument 0.

Definition at line 211 of file cpu.h.

#### 9.185.2.19 saved_arg1

```
uint32_t SPARC_Minimum_stack_frame::saved_arg1
```

This is the offset of the register for saved argument 1.

Definition at line 213 of file cpu.h.

#### 9.185.2.20 saved_arg2

```
uint32_t SPARC_Minimum_stack_frame::saved_arg2
```

This is the offset of the register for saved argument 2.

Definition at line 215 of file cpu.h.

#### 9.185.2.21 saved_arg3

```
uint32_t SPARC_Minimum_stack_frame::saved_arg3
```

This is the offset of the register for saved argument 3.

Definition at line 217 of file cpu.h.

#### 9.185.2.22 saved_arg4

```
uint32_t SPARC_Minimum_stack_frame::saved_arg4
```

This is the offset of the register for saved argument 4.

Definition at line 219 of file cpu.h.

#### 9.185.2.23 saved_arg5

```
uint32_t SPARC_Minimum_stack_frame::saved_arg5
```

This is the offset of the register for saved argument 5.

Definition at line 221 of file cpu.h.

#### 9.185.2.24 structure_return_address

```
void* SPARC_Minimum_stack_frame::structure_return_address
```

This is the offset of the register used to return structures.

Definition at line 204 of file cpu.h.

The documentation for this struct was generated from the following file:

- cpukit/score/cpu/sparc/include/rtems/score/cpu.h

## 9.186 Stack_Control Struct Reference

```
#include <stack.h>
```

### Public Attributes

- size_t size
- void * area

### 9.186.1 Detailed Description

The following defines the control block used to manage each stack.

Definition at line 53 of file stack.h.

### 9.186.2 Member Data Documentation

#### 9.186.2.1 area

```
void* Stack_Control::area
```

This is the low memory address of stack.

Definition at line 57 of file stack.h.

#### 9.186.2.2 size

```
size_t Stack_Control::size
```

This is the stack size.

Definition at line 55 of file stack.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/stack.h

## 9.187 statvfs Struct Reference

### Public Attributes

- unsigned long f_bsize
- unsigned long f_frsize
- fsblkcnt_t f_blocks
- fsblkcnt_t f_bfree
- fsblkcnt_t f_bavail
- fsfilcnt_t f_files
- fsfilcnt_t f_ffree
- fsfilcnt_t f_favail
- unsigned long f_fsid
- unsigned long f_flag
- unsigned long f_namemax

## 9.187.1 Detailed Description

Definition at line 35 of file statvfs.h.

## 9.187.2 Member Data Documentation

### 9.187.2.1 f_bavail

```
fsblkcnt_t statvfs::f_bavail
```

Number of free blocks available to non-privileged process.

Definition at line 42 of file statvfs.h.

### 9.187.2.2 f_bfree

```
fsblkcnt_t statvfs::f_bfree
```

Total number of free blocks.

Definition at line 41 of file statvfs.h.

### 9.187.2.3 f_blocks

```
fsblkcnt_t statvfs::f_blocks
```

Total number of blocks on file system in units of f_frsize.

Definition at line 39 of file statvfs.h.

### 9.187.2.4 f_bsize

```
unsigned long statvfs::f_bsize
```

File system block size.

Definition at line 37 of file statvfs.h.

#### 9.187.2.5 f_favail

```
fsfilcnt_t statvfs::f_favail
```

Number of file serial numbers available to non-privileged process.

Definition at line 46 of file statvfs.h.

#### 9.187.2.6 f_ffree

```
fsfilcnt_t statvfs::f_ffree
```

Total number of free file serial numbers.

Definition at line 45 of file statvfs.h.

#### 9.187.2.7 f_files

```
fsfilcnt_t statvfs::f_files
```

Total number of file serial numbers.

Definition at line 44 of file statvfs.h.

#### 9.187.2.8 f_flag

```
unsigned long statvfs::f_flag
```

Bit mask of f_flag values.

Definition at line 49 of file statvfs.h.

#### 9.187.2.9 f_frsize

```
unsigned long statvfs::f_frsize
```

Fundamental file system block size.

Definition at line 38 of file statvfs.h.

#### 9.187.2.10 f_fsid

```
unsigned long statvfs::f_fsid
```

File system ID.

Definition at line 48 of file statvfs.h.

#### 9.187.2.11 f_namemax

```
unsigned long statvfs::f_namemax
```

Maximum filename length.

Definition at line 50 of file statvfs.h.

The documentation for this struct was generated from the following file:

- cpukit/include/sys/statvfs.h

## 9.188 T_case_context Struct Reference

### Public Attributes

- const char * **name**
- void(* **body** )(void)
- const T_fixture * **fixture**
- const struct T_case_context * **next**

### 9.188.1 Detailed Description

Definition at line 115 of file test.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/test.h

## 9.189 T_check_context Struct Reference

### Public Attributes

- const char * **file**
- int **line**
- unsigned int **flags**

### 9.189.1 Detailed Description

Definition at line 139 of file test.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/test.h

## 9.190 T_check_context_msg Struct Reference

### Public Attributes

- T_check_context **base**
- const char * **msg**

### 9.190.1 Detailed Description

Definition at line 145 of file test.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/test.h

## 9.191 T_config Struct Reference

### Public Attributes

- const char * **name**
- char * **buf**
- size_t **buf_size**
- T_putchar **putchar**
- void * **putchar_arg**
- T_verbosity **verbosity**
- T_time(* **now** )(void)
- size_t **action_count**
- const T_action * **actions**

### 9.191.1 Detailed Description

Definition at line 2281 of file test.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/test.h

## 9.192 T_context Struct Reference

### Public Attributes

- pthread_spinlock_t **lock**
- char * **buf**
- unsigned int **buf_mask**
- atomic_uint **buf_head**
- atomic_uint **buf_tail**
- void(* **putchar** )(int, void *)
- void * **putchar_arg**
- T_verbosity **verbosity**
- const T_case_context * **registered_cases**
- const T_case_context * **current_case**
- T_fixture_node * **fixtures**
- T_fixture_node **case_fixture**

### 9.192.1 Detailed Description

Definition at line 52 of file t-test.c.

The documentation for this struct was generated from the following file:

- cpukit/libtest/t-test.c

## 9.193 T_destructor Struct Reference

### Public Attributes

- struct {
  struct T_destructor * **le_next**
  struct T_destructor ** **le_prev**
  } **node**

- void(* **destroy** )(struct T_destructor *)

### 9.193.1 Detailed Description

Definition at line 2509 of file test.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/test.h

## 9.194 T_fixture Struct Reference

### Public Attributes

- void(∗ **setup** )(void ∗)
- void(∗ **stop** )(void ∗)
- void(∗ **teardown** )(void ∗)
- size_t(∗ **scope** )(void ∗, char ∗, size_t)
- void ∗ **initial_context**

### 9.194.1 Detailed Description

Definition at line 61 of file test.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/test.h

## 9.195 T_fixture_node Struct Reference

### Public Attributes

- struct T_fixture_node ∗ **next**
- struct T_fixture_node ∗ **previous**
- const T_fixture ∗ **fixture**
- void ∗ **context**
- unsigned int **next_planned_steps**
- unsigned int **next_steps**
- unsigned int **failures**

### 9.195.1 Detailed Description

Definition at line 69 of file test.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/test.h

## 9.196 T_interrupt_clock_time Struct Reference

### Public Attributes

- int64_t **t**
- int64_t **d**

### 9.196.1 Detailed Description

Definition at line 77 of file t-test-interrupt.c.

The documentation for this struct was generated from the following file:

- cpukit/libtest/t-test-interrupt.c

## 9.197 T_interrupt_context Struct Reference

### Public Attributes

- uint_fast32_t **one_tick_busy**
- int64_t **t0**
- Thread_Control * **self**
- Atomic_Uint **state**
- void(* **prepare** )(void *)
- void(* **action** )(void *)
- T_interrupt_test_state(* **interrupt** )(void *)
- void(* **blocked** )(void *)
- void * **arg**
- Per_CPU_Job **job**
- Per_CPU_Job_context **job_context**
- Watchdog_Control **wdg**
- User_extensions_Control **ext**
- T_fixture_node **node**

### 9.197.1 Detailed Description

Definition at line 58 of file t-test-interrupt.c.

The documentation for this struct was generated from the following file:

- cpukit/libtest/t-test-interrupt.c

## 9.198 T_interrupt_test_config Struct Reference

### Public Attributes

- void(* **prepare** )(void *)
- void(* **action** )(void *)
- T_interrupt_test_state(* **interrupt** )(void *)
- void(* **blocked** )(void *)
- uint32_t **max_iteration_count**

### 9.198.1 Detailed Description

Definition at line 2404 of file test.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/test.h

## 9.199 T_measure_runtime_config Struct Reference

### Public Attributes

- size_t **sample_count**

### 9.199.1 Detailed Description

Definition at line 2535 of file test.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/test.h

## 9.200 T_measure_runtime_context Struct Reference

### Public Attributes

- T_destructor **destructor**
- size_t **sample_count**
- T_ticks * **samples**
- size_t **cache_line_size**
- size_t **chunk_size**
- volatile unsigned int * **chunk**
- rtems_id **runner**
- uint32_t **load_count**
- load_context * **load_contexts**

### 9.200.1 Detailed Description

Definition at line 45 of file t-test-rtems-measure.c.

The documentation for this struct was generated from the following file:

- cpukit/libtest/t-test-rtems-measure.c

## 9.201 T_measure_runtime_request Struct Reference

### Public Attributes

- const char * **name**
- int **flags**
- void(* **setup** )(void *)
- void(* **body** )(void *)
- bool(* **teardown** )(void *, T_ticks *, uint32_t, uint32_t, unsigned int)
- void * **arg**

### 9.201.1 Detailed Description

Definition at line 2539 of file test.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/test.h

## 9.202 T_putchar_string_context Struct Reference

### Public Attributes

- char * **s**
- size_t **n**

### 9.202.1 Detailed Description

Definition at line 128 of file t-test.c.

The documentation for this struct was generated from the following file:

- cpukit/libtest/t-test.c

## 9.203 T_report_hash_sha256_context Struct Reference

### Public Attributes

- SHA256_CTX **sha256**
- T_putchar **putchar**
- void * **putchar_arg**

### 9.203.1 Detailed Description

Definition at line 36 of file t-test-hash-sha256.c.

The documentation for this struct was generated from the following file:

- cpukit/libtest/t-test-hash-sha256.c

## 9.204 T_thread_switch_context Struct Reference

### Public Attributes

- T_thread_switch_log * **active**
- User_extensions_Control **ext**
- T_destructor **dtor**

### 9.204.1 Detailed Description

Definition at line 48 of file t-test-thread-switch.c.

The documentation for this struct was generated from the following file:

- cpukit/libtest/t-test-thread-switch.c

## 9.205 T_thread_switch_event Struct Reference

### Public Attributes

- uint32_t **executing**
- uint32_t **heir**
- uint32_t **cpu**
- T_time **instant**

### 9.205.1 Detailed Description

Definition at line 2422 of file test.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/test.h

## 9.206 T_thread_switch_log Struct Reference

### Public Attributes

- size_t **recorded**
- size_t **capacity**
- uint64_t **switches**
- T_thread_switch_event **events** [T_ZERO_LENGTH_ARRAY]

### 9.206.1 Detailed Description

Definition at line 2429 of file test.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/test.h

## 9.207 T_thread_switch_log_10 Struct Reference

### Public Attributes

- T_thread_switch_log **log**
- T_thread_switch_event **events** [T_ZERO_LENGTH_ARRAY_EXTENSION(10)]

### 9.207.1 Detailed Description

Definition at line 2446 of file test.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/test.h

## 9.208 T_thread_switch_log_2 Struct Reference

### Public Attributes

- T_thread_switch_log **log**
- T_thread_switch_event **events** [T_ZERO_LENGTH_ARRAY_EXTENSION(2)]

### 9.208.1 Detailed Description

Definition at line 2436 of file test.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/test.h

## 9.209 T_thread_switch_log_4 Struct Reference

### Public Attributes

- T_thread_switch_log **log**
- T_thread_switch_event **events** [T_ZERO_LENGTH_ARRAY_EXTENSION(4)]

### 9.209.1 Detailed Description

Definition at line 2441 of file test.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/test.h

## 9.210 Thread_Action Struct Reference

Thread action.

`#include <thread.h>`

### Public Attributes

- Chain_Node **Node**
- Thread_Action_handler **handler**

### 9.210.1 Detailed Description

Thread action.

Thread actions can be chained together to trigger a set of actions on particular events like for example a thread post-switch. Use _Thread_Action_initialize() to initialize this structure.

Thread actions are the building block for efficient implementation of

- Classic signals delivery,
- POSIX signals delivery, and
- thread life-cycle changes.

**See also**

_Thread_Add_post_switch_action() and _Thread_Run_post_switch_actions().

Definition at line 633 of file thread.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/thread.h

## 9.211 Thread_Action_control Struct Reference

Control block to manage thread actions.

`#include <thread.h>`

### Public Attributes

- Chain_Control **Chain**

### 9.211.1 Detailed Description

Control block to manage thread actions.

Use _Thread_Action_control_initialize() to initialize this structure.

Definition at line 658 of file thread.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/thread.h

## 9.212 Thread_Capture_control Struct Reference

### Public Attributes

- uint32_t **flags**
- void * **control**

### 9.212.1 Detailed Description

Definition at line 711 of file thread.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/thread.h

## 9.213 Thread_Close_context Struct Reference

### Public Attributes

- Thread_queue_Context **Base**
- Thread_Control * **cancel**

### 9.213.1 Detailed Description

Definition at line 371 of file threadimpl.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/threadimpl.h

## 9.214 Thread_Configuration Struct Reference

The configuration of a new thread to initialize.

`#include <threadimpl.h>`

### Public Attributes

- const struct _Scheduler_Control * scheduler
  *The scheduler control instance for the thread.*
- void * stack_area
  *The starting address of the stack area.*
- size_t stack_size
  *The size of the stack area in bytes.*
- void(* stack_free )(void *)
  *This member contains the handler to free the stack.*
- Priority_Control priority
  *The new thread's priority.*
- Thread_CPU_budget_algorithms budget_algorithm
  *The thread's budget algorithm.*
- Thread_CPU_budget_algorithm_callout budget_callout
  *The thread's initial budget callout.*
- Objects_Name name
  *Name of the object for the thread.*
- uint32_t isr_level
  *The thread's initial ISR level.*
- bool is_fp
  *Indicates whether the thread needs a floating-point area.*
- bool is_preemptible
  *Indicates whether the new thread is preemptible.*

### 9.214.1 Detailed Description

The configuration of a new thread to initialize.

Definition at line 130 of file threadimpl.h.

### 9.214.2 Member Data Documentation

#### 9.214.2.1 stack_free

```
void( * Thread_Configuration::stack_free) (void *)
```

This member contains the handler to free the stack.

It shall not be NULL. Use _Stack_Free_nothing() if nothing is to free.

Definition at line 151 of file threadimpl.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/threadimpl.h

## 9.215 Thread_Control_add_on Struct Reference

Thread control add-on.

```
#include <thread.h>
```

### Public Attributes

- size_t destination_offset

  *Offset of the pointer field in Thread_Control referencing an application configuration dependent memory area in the thread control block.*
- size_t source_offset

  *Offset relative to the thread control block begin to an application configuration dependent memory area.*

### 9.215.1 Detailed Description

Thread control add-on.

Definition at line 893 of file thread.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/thread.h

## 9.216 Thread_Entry_idle Struct Reference

Data for idle thread entry.

```
#include <thread.h>
```

### Public Attributes

- void ∗(∗ **entry** )(uintptr_t argument)

### 9.216.1 Detailed Description

Data for idle thread entry.

Definition at line 106 of file thread.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/thread.h

## 9.217 Thread_Entry_information Struct Reference

Thread entry information.

`#include <thread.h>`

### Public Attributes

- void(∗ adaptor )(Thread_Control ∗executing)

  *Thread entry adaptor.*

- union {
  Thread_Entry_idle **Idle**
  Thread_Entry_numeric **Numeric**
  Thread_Entry_pointer **Pointer**
  } Kinds

  *Thread entry data used by the adaptor to call the thread entry function with the right parameters.*

### 9.217.1 Detailed Description

Thread entry information.

Definition at line 130 of file thread.h.

### 9.217.2 Member Data Documentation

#### 9.217.2.1 adaptor

```
void( * Thread_Entry_information::adaptor) (Thread_Control *executing)
```

Thread entry adaptor.

Calls the corresponding thread entry with the right parameters.

**Parameters**

| | |
|---|---|
| *executing* | The executing thread. |

Definition at line 138 of file thread.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/thread.h

## 9.218 Thread_Entry_numeric Struct Reference

Data for thread entry with one numeric argument and no return value.

`#include <thread.h>`

### Public Attributes

- void(∗ **entry** )(Thread_Entry_numeric_type argument)
- Thread_Entry_numeric_type **argument**

### 9.218.1 Detailed Description

Data for thread entry with one numeric argument and no return value.

Definition at line 113 of file thread.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/thread.h

## 9.219 Thread_Entry_pointer Struct Reference

Data for thread entry with one pointer argument and a pointer return value.

`#include <thread.h>`

### Public Attributes

- void ∗(∗ **entry** )(void ∗argument)
- void ∗ **argument**

### 9.219.1 Detailed Description

Data for thread entry with one pointer argument and a pointer return value.

Definition at line 122 of file thread.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/thread.h

## 9.220 Thread_Information Struct Reference

The thread object information.

`#include <thread.h>`

### Public Attributes

- Objects_Information Objects
  *The object information.*
- union {
  Thread_queue_Configured_heads * initial
  *Contains the initial set of thread queue heads.*
  Freechain_Control Free
  *The free thread queue heads.*
  } Thread_queue_heads
  *Thread queue heads maintenance.*

### 9.220.1 Detailed Description

The thread object information.

Definition at line 1000 of file thread.h.

### 9.220.2 Member Data Documentation

#### 9.220.2.1 Free

`Freechain_Control Thread_Information::Free`

The free thread queue heads.

This member is initialized by _Thread_Initialize_information().

Definition at line 1022 of file thread.h.

#### 9.220.2.2 initial

```
Thread_queue_Configured_heads* Thread_Information::initial
```

Contains the initial set of thread queue heads.

This is set by <rtems/confdefs.h> via THREAD_INFORMATION_DEFINE().

Definition at line 1015 of file thread.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/thread.h

## 9.221 Thread_Join_context Struct Reference

### Public Attributes

- Thread_queue_Context **Base**
- void * **exit_value**

### 9.221.1 Detailed Description

Definition at line 81 of file threadrestart.c.

The documentation for this struct was generated from the following file:

- cpukit/score/src/threadrestart.c

## 9.222 Thread_Keys_information Struct Reference

Per-thread information for POSIX Keys.

```
#include <thread.h>
```

### Public Attributes

- RBTree_Control Key_value_pairs
  *Key value pairs registered for this thread.*

### 9.222.1 Detailed Description

Per-thread information for POSIX Keys.

Definition at line 641 of file thread.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/thread.h

## 9.223 Thread_Life_control Struct Reference

Thread life control.

`#include <thread.h>`

### Public Attributes

- Thread_Action Action
  *Thread life action used to react upon thread restart and delete requests.*
- Thread_Life_state state
  *The current thread life state.*
- uint32_t pending_life_change_requests
  *The count of pending life change requests.*
- void * exit_value
  *The thread exit value.*

### 9.223.1 Detailed Description

Thread life control.

Definition at line 683 of file thread.h.

### 9.223.2 Member Data Documentation

#### 9.223.2.1 exit_value

`void* Thread_Life_control::exit_value`

The thread exit value.

It is,

- the value passed to pthread_exit(), or
- PTHREAD_CANCELED in case it is cancelled via pthread_cancel(), or
- NULL.

Definition at line 708 of file thread.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/thread.h

# 9.224 Thread_Proxy_control Struct Reference

`#include <thread.h>`

## Public Attributes

- Objects_Control Object
- Thread_queue_Control Join_queue
- States_Control current_state
- Priority_Node Real_priority

  *The base priority of this thread in its home scheduler instance.*
- Thread_Scheduler_control Scheduler

  *Scheduler related control.*
- Thread_Wait_information Wait
- Thread_Timer_information Timer

## 9.224.1 Detailed Description

The following defines the control block used to manage each thread proxy.

**Note**

It is critical that proxies and threads have identical memory images for the shared part.

Definition at line 512 of file thread.h.

## 9.224.2 Member Data Documentation

### 9.224.2.1 current_state

`States_Control Thread_Proxy_control::current_state`

This field is the current execution state of this proxy.

Definition at line 522 of file thread.h.

### 9.224.2.2 Join_queue

`Thread_queue_Control Thread_Proxy_control::Join_queue`

**See also**

Thread_Control::Join_queue

Definition at line 519 of file thread.h.

#### 9.224.2.3 Object

```
Objects_Control Thread_Proxy_control::Object
```

This field is the object management structure for each proxy.

Definition at line 514 of file thread.h.

#### 9.224.2.4 Timer

```
Thread_Timer_information Thread_Proxy_control::Timer
```

This field is the Watchdog used to manage proxy delays and timeouts.

Definition at line 542 of file thread.h.

#### 9.224.2.5 Wait

```
Thread_Wait_information Thread_Proxy_control::Wait
```

This field is the blocking information for this proxy.

Definition at line 540 of file thread.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/thread.h

## 9.225 Thread_queue_Context Struct Reference

Thread queue context for the thread queue methods.

```
#include <threadq.h>
```

## Public Attributes

- Thread_queue_Lock_context Lock_context

  *The lock context for the thread queue acquire and release operations.*
- States_Control thread_state

  *The thread state for _Thread_queue_Enqueue().*
- Thread_queue_Enqueue_callout enqueue_callout

  *The enqueue callout for _Thread_queue_Enqueue().*
- union {
  Watchdog_Interval ticks
  *The timeout in ticks.*
  const void * arg
  *The timeout argument, e.g. pointer to struct timespec.*
  } Timeout

  *Interval to wait.*
- struct {
  Chain_Control Links
  *The chain of thread queue links defining the thread queue path.*
  Thread_queue_Link Start
  *The start of a thread queue path.*
  Thread_queue_Link Deadlock
  *In case of a deadlock, a link for the first thread on the path that tries to enqueue on a thread queue.*
  } Path

  *Representation of a thread queue path from a start thread queue to the terminal thread queue.*
- struct {
  Priority_Actions Actions
  *A priority action list.*
  size_t update_count
  *Count of threads to update the priority via _Thread_Priority_update().*
  Thread_Control * update [2]
  *Threads to update the priority via _Thread_Priority_update().*
  } Priority

  *Block to manage thread priority changes due to a thread queue operation.*
- Thread_queue_Deadlock_callout deadlock_callout

  *Invoked in case of a detected deadlock.*

### 9.225.1 Detailed Description

Thread queue context for the thread queue methods.

**See also**

_Thread_queue_Context_initialize().

Definition at line 198 of file threadq.h.

### 9.225.2 Member Data Documentation

#### 9.225.2.1 deadlock_callout

```
Thread_queue_Deadlock_callout Thread_queue_Context::deadlock_callout
```

Invoked in case of a detected deadlock.

Must be initialized for _Thread_queue_Enqueue() in case the thread queue may have an owner, e.g. for mutex objects.

**See also**

_Thread_queue_Context_set_deadlock_callout().

Definition at line 304 of file threadq.h.

#### 9.225.2.2 enqueue_callout

```
Thread_queue_Enqueue_callout Thread_queue_Context::enqueue_callout
```

The enqueue callout for _Thread_queue_Enqueue().

The callout is invoked after the release of the thread queue lock with thread dispatching disabled. Afterwards the thread is blocked. This callout must be used to install the thread watchdog for timeout handling.

**See also**

_Thread_queue_Enqueue_do_nothing_extra(). _Thread_queue_Add_timeout_ticks(), and _Thread_queue_Add_timeout_realti

Definition at line 221 of file threadq.h.

#### 9.225.2.3 Path

```
struct { ... } Thread_queue_Context::Path
```

Representation of a thread queue path from a start thread queue to the terminal thread queue.

The start thread queue is determined by the object on which a thread intends to block. The terminal thread queue is the thread queue reachable via thread queue links whose owner is not blocked on a thread queue. The thread queue links are determined by the thread queue owner and thread wait queue relationships.

#### 9.225.2.4 Timeout

```
union { ... } Thread_queue_Context::Timeout
```

Interval to wait.

May be used by the enqueue callout to register a timeout handler.

#### 9.225.2.5 update

```
Thread_Control* Thread_queue_Context::update[2]
```

Threads to update the priority via _Thread_Priority_update().

Currently, a maximum of two threads need an update in one rush, for example the thread of the thread queue operation and the owner of the thread queue.

Definition at line 293 of file threadq.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/threadq.h

## 9.226 Thread_queue_Control Struct Reference

```
#include <threadq.h>
```

### Public Attributes

- Thread_queue_Queue Queue
  *The actual thread queue.*

### 9.226.1 Detailed Description

This is the structure used to manage sets of tasks which are blocked waiting to acquire a resource.

Definition at line 552 of file threadq.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/threadq.h

## 9.227 Thread_queue_Gate Struct Reference

The thread queue gate is an SMP synchronization means.

```
#include <threadq.h>
```

### Public Attributes

- Chain_Node **Node**
- Atomic_Uint **go_ahead**

### 9.227.1 Detailed Description

The thread queue gate is an SMP synchronization means.

The gates are added to a list of requests. A busy wait is performed to make sure that preceding requests are carried out. Each predecessor notifies its successor about on request completion.

**See also**

_Thread_queue_Gate_add(), _Thread_queue_Gate_wait(), and _Thread_queue_Gate_open().

Definition at line 118 of file threadq.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/threadq.h

## 9.228 Thread_queue_Link Struct Reference

A thread queue link from one thread to another specified by the thread queue owner and thread wait queue relationships.

`#include <threadq.h>`

### Public Attributes

- RBTree_Node Registry_node

  *Node to register this link in the global thread queue links lookup tree.*
- Thread_queue_Queue * source

  *The source thread queue determined by the thread queue owner.*
- Thread_queue_Queue * target

  *The target thread queue determined by the thread wait queue of the source owner.*
- Chain_Node Path_node

  *Node to add this link to a thread queue path.*
- Thread_Control * owner

  *The owner of this thread queue link.*
- Thread_queue_Lock_context Lock_context

  *The queue lock context used to acquire the thread wait lock of the owner.*

### 9.228.1 Detailed Description

A thread queue link from one thread to another specified by the thread queue owner and thread wait queue relationships.

Definition at line 157 of file threadq.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/threadq.h

# 9.229 Thread_queue_Links Struct Reference

## Public Attributes

- ISR_lock_Control **Lock**
- RBTree_Control **Links**

### 9.229.1 Detailed Description

Definition at line 50 of file threadqenqueue.c.

The documentation for this struct was generated from the following file:

- cpukit/score/src/threadqenqueue.c

# 9.230 Thread_queue_Lock_context Struct Reference

## Public Attributes

- ISR_lock_Context Lock_context

  *The lock context for the thread queue acquire and release operations.*

- struct {
  Thread_queue_Gate Gate
  *Gate to synchronize thread wait lock requests.*
  Thread_queue_Queue * queue
  *The thread queue in case the thread is blocked on a thread queue.*
  } Wait

  *Data to support thread queue enqueue operations.*

### 9.230.1 Detailed Description

Definition at line 125 of file threadq.h.

### 9.230.2 Member Data Documentation

#### 9.230.2.1 Gate

```
Thread_queue_Gate Thread_queue_Lock_context::Gate
```

Gate to synchronize thread wait lock requests.

**See also**

_Thread_Wait_acquire_critical() and _Thread_Wait_tranquilize().

Definition at line 142 of file threadq.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/threadq.h

## 9.231 Thread_queue_Object Struct Reference

Helper structure to ensure that all objects containing a thread queue have the right layout.

```
#include <threadqimpl.h>
```

### Public Attributes

- Objects_Control **Object**
- Thread_queue_Control **Wait_queue**

### 9.231.1 Detailed Description

Helper structure to ensure that all objects containing a thread queue have the right layout.

**See also**

_Thread_Wait_get_id() and THREAD_QUEUE_OBJECT_ASSERT().

Definition at line 1445 of file threadqimpl.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/threadqimpl.h

## 9.232 Thread_queue_Operations Struct Reference

Thread queue operations.

```
#include <threadq.h>
```

## Public Attributes

- Thread_queue_Priority_actions_operation priority_actions
  *Thread queue priority actions operation.*
- Thread_queue_Enqueue_operation enqueue
  *Thread queue enqueue operation.*
- Thread_queue_Extract_operation extract
  *Thread queue extract operation.*
- Thread_queue_Surrender_operation surrender
  *Thread queue surrender operation.*
- Thread_queue_First_operation first
  *Thread queue first operation.*

### 9.232.1 Detailed Description

Thread queue operations.

**See also**

_Thread_wait_Set_operations().

Definition at line 517 of file threadq.h.

### 9.232.2 Member Data Documentation

#### 9.232.2.1 enqueue

```
Thread_queue_Enqueue_operation Thread_queue_Operations::enqueue
```

Thread queue enqueue operation.

Called by object routines to enqueue the thread.

Definition at line 528 of file threadq.h.

#### 9.232.2.2 extract

```
Thread_queue_Extract_operation Thread_queue_Operations::extract
```

Thread queue extract operation.

Called by object routines to extract a thread from a thread queue.

Definition at line 535 of file threadq.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/threadq.h

## 9.233 Thread_queue_Priority_queue Struct Reference

Thread priority queue.

`#include <threadq.h>`

### Public Attributes

- Chain_Node Node

  *Node to enqueue this queue in the FIFO chain of the corresponding heads structure.*
- Priority_Aggregation Queue

  *The actual thread priority queue.*
- struct Scheduler_Node * scheduler_node

  *This priority queue is added to a scheduler node of the owner in case of priority inheritance.*

### 9.233.1 Detailed Description

Thread priority queue.

Definition at line 324 of file threadq.h.

### 9.233.2 Member Data Documentation

#### 9.233.2.1 Node

`Chain_Node Thread_queue_Priority_queue::Node`

Node to enqueue this queue in the FIFO chain of the corresponding heads structure.

**See also**

Thread_queue_Heads::Heads::Fifo.

Definition at line 332 of file threadq.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/threadq.h

# 9.234 Thread_queue_Queue Struct Reference

## Public Attributes

- SMP_ticket_lock_Control Lock
  *Lock to protect this thread queue.*
- Thread_queue_Heads * heads
  *The thread queue heads.*
- Thread_Control * owner
  *The thread queue owner.*
- const char * name
  *The thread queue name.*

## 9.234.1 Detailed Description

Definition at line 402 of file threadq.h.

## 9.234.2 Member Data Documentation

### 9.234.2.1 heads

```
Thread_queue_Heads* Thread_queue_Queue::heads
```

The thread queue heads.

This pointer is NULL, if and only if no threads are enqueued. The first thread to enqueue will give its spare thread queue heads to this thread queue.

Definition at line 426 of file threadq.h.

### 9.234.2.2 Lock

```
SMP_ticket_lock_Control Thread_queue_Queue::Lock
```

Lock to protect this thread queue.

It may be used to protect additional state of the object embedding this thread queue.

Must be the first component of this structure to be able to re-use implementation parts for structures defined by Newlib <sys/lock.h>.

**See also**

_Thread_queue_Acquire(), _Thread_queue_Acquire_critical() and _Thread_queue_Release().

Definition at line 416 of file threadq.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/threadq.h

## 9.235 Thread_queue_Syslock_queue Struct Reference

Thread queue with a layout compatible to struct _Thread_queue_Queue defined in Newlib <sys/lock.h>.

`#include <threadqimpl.h>`

### Public Attributes

- Thread_queue_Queue **Queue**

### 9.235.1 Detailed Description

Thread queue with a layout compatible to struct _Thread_queue_Queue defined in Newlib <sys/lock.h>.

Definition at line 54 of file threadqimpl.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/threadqimpl.h

## 9.236 Thread_Scheduler_control Struct Reference

Thread scheduler control.

`#include <thread.h>`

### Public Attributes

- ISR_lock_Control Lock
  *Lock to protect the scheduler node change requests.*
- Thread_Scheduler_state state
  *The current scheduler state of this thread.*
- const struct _Scheduler_Control * home_scheduler
  *The home scheduler of this thread.*
- const struct _Scheduler_Control * pinned_scheduler
  *The pinned scheduler of this thread.*
- struct Per_CPU_Control * cpu
  *The processor assigned by the current scheduler.*
- Chain_Control Wait_nodes
  *Scheduler nodes immediately available to the thread by its home scheduler and due to thread queue ownerships.*
- Chain_Control Scheduler_nodes
  *Scheduler nodes immediately available to the schedulers for this thread.*
- Chain_Node Help_node
  *Node for the Per_CPU_Control::Threads_in_need_for_help chain.*
- size_t helping_nodes
  *Count of nodes scheduler nodes minus one.*
- Scheduler_Node * requests
  *List of pending scheduler node requests.*
- int pin_level
  *The thread pinning to current processor level.*
- Processor_mask Affinity
  *The thread processor affinity set.*
- Scheduler_Node * nodes
  *The scheduler nodes of this thread.*

## 9.236.1 Detailed Description

Thread scheduler control.

Definition at line 243 of file thread.h.

## 9.236.2 Member Data Documentation

### 9.236.2.1 Help_node

```
Chain_Node Thread_Scheduler_control::Help_node
```

Node for the Per_CPU_Control::Threads_in_need_for_help chain.

This chain is protected by the Per_CPU_Control::Lock lock of the assigned processor.

Definition at line 302 of file thread.h.

### 9.236.2.2 helping_nodes

```
size_t Thread_Scheduler_control::helping_nodes
```

Count of nodes scheduler nodes minus one.

This chain is protected by the thread state lock.

Definition at line 309 of file thread.h.

### 9.236.2.3 nodes

```
Scheduler_Node* Thread_Scheduler_control::nodes
```

The scheduler nodes of this thread.

Each thread has a scheduler node for each scheduler instance.

Definition at line 351 of file thread.h.

#### 9.236.2.4 pin_level

```
int Thread_Scheduler_control::pin_level
```

The thread pinning to current processor level.

Must be touched only by the executing thread with thread dispatching disabled. If non-zero, then the thread is pinned to its current processor. The pin level is incremented and decremented by two. The least-significant bit indicates that the thread was pre-empted and must undo the pinning with respect to the scheduler once the level changes from three to one.

The thread pinning may be used to access per-processor data structures in critical sections with enabled thread dispatching, e.g. a pinned thread is allowed to block.

Thread pinning should be used only for short critical sections and not all the time. Thread pinning is a very low overhead operation in case the thread is not preempted during the pinning.

**See also**

_Thread_Pin() and _Thread_Unpin().

Definition at line 338 of file thread.h.

#### 9.236.2.5 requests

```
Scheduler_Node* Thread_Scheduler_control::requests
```

List of pending scheduler node requests.

This list is protected by the thread scheduler lock.

Definition at line 316 of file thread.h.

#### 9.236.2.6 Scheduler_nodes

```
Chain_Control Thread_Scheduler_control::Scheduler_nodes
```

Scheduler nodes immediately available to the schedulers for this thread.

This chain is protected by the thread state lock.

This chain is never empty for normal threads (the only exception are idle threads associated with an online processor which is not used by a scheduler). In case a pinned scheduler is set for this thread, then the first scheduler node of this chain belongs to the pinned scheduler, otherwise the first scheduler node of this chain belongs to the home scheduler.

Definition at line 294 of file thread.h.

#### 9.236.2.7 Wait_nodes

```
Chain_Control Thread_Scheduler_control::Wait_nodes
```

Scheduler nodes immediately available to the thread by its home scheduler and due to thread queue ownerships.

This chain is protected by the thread wait lock.

This chain is never empty. The first scheduler node on the chain is the scheduler node of the home scheduler.

Definition at line 279 of file thread.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/thread.h

## 9.237 Thread_Start_information Struct Reference

```
#include <thread.h>
```

### Public Attributes

- Thread_Entry_information Entry
- bool is_preemptible
- Thread_CPU_budget_algorithms budget_algorithm
- Thread_CPU_budget_algorithm_callout budget_callout
- uint32_t isr_level
- Priority_Control initial_priority
- void(∗ stack_free )(void ∗)

  *This field points to the handler which should free the stack.*
- Stack_Control Initial_stack
- void ∗ tls_area

### 9.237.1 Detailed Description

The following structure contains the information which defines the starting state of a thread.

Definition at line 178 of file thread.h.

### 9.237.2 Member Data Documentation

#### 9.237.2.1 budget_algorithm

```
Thread_CPU_budget_algorithms Thread_Start_information::budget_algorithm
```

This field indicates the CPU budget algorith.

Definition at line 187 of file thread.h.

#### 9.237.2.2 budget_callout

```
Thread_CPU_budget_algorithm_callout Thread_Start_information::budget_callout
```

This field is the routine to invoke when the CPU allotment is consumed.

Definition at line 191 of file thread.h.

#### 9.237.2.3 Entry

```
Thread_Entry_information Thread_Start_information::Entry
```

This field contains the thread entry information.

Definition at line 180 of file thread.h.

#### 9.237.2.4 initial_priority

```
Priority_Control Thread_Start_information::initial_priority
```

This field is the initial priority.

Definition at line 195 of file thread.h.

#### 9.237.2.5 Initial_stack

```
Stack_Control Thread_Start_information::Initial_stack
```

This field is the stack information.

Definition at line 201 of file thread.h.

#### 9.237.2.6 is_preemptible

```
bool Thread_Start_information::is_preemptible
```

This field indicates whether the thread was preemptible when it started.

Definition at line 185 of file thread.h.

#### 9.237.2.7 isr_level

```
uint32_t Thread_Start_information::isr_level
```

This field is the initial ISR disable level of this thread.

Definition at line 193 of file thread.h.

#### 9.237.2.8 tls_area

```
void* Thread_Start_information::tls_area
```

The thread-local storage (TLS) area

Definition at line 207 of file thread.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/thread.h

## 9.238 Thread_Timer_information Struct Reference

Information required to manage a thread timer.

```
#include <thread.h>
```

### Public Attributes

- Watchdog_Header * **header**
- Watchdog_Control **Watchdog**

### 9.238.1 Detailed Description

Information required to manage a thread timer.

Definition at line 499 of file thread.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/thread.h

## 9.239 Thread_Wait_information Struct Reference

Information required to manage a thread while it is blocked.

`#include <thread.h>`

### Public Attributes

- uint32_t count
- void * return_argument
- Thread_Wait_information_Object_argument_type return_argument_second
- uint32_t option
- uint32_t return_code
- Atomic_Uint flags

  *This field contains several flags used to control the wait class and state of a thread in case fine-grained locking is used.*
- struct {
  ISR_lock_Control Default
  *Thread wait default lock.*
  Chain_Control Pending_requests
  *The pending thread wait lock acquire or tranquilize requests in case the thread is enqueued on a thread queue.*
  Thread_queue_Gate Tranquilizer
  *Tranquilizer gate used by _Thread_Wait_tranquilize().*
  } Lock

  *Thread wait lock control block.*
- Thread_queue_Link Link

  *Thread queue link provided for use by the thread wait lock owner to build a thread queue path.*
- Thread_queue_Queue * queue

  *The current thread queue.*
- const Thread_queue_Operations * operations

  *The current thread queue operations.*
- Thread_queue_Heads * **spare_heads**

### 9.239.1 Detailed Description

Information required to manage a thread while it is blocked.

This contains the information required to manage a thread while it is blocked and to return information to it.

Definition at line 391 of file thread.h.

### 9.239.2 Member Data Documentation

#### 9.239.2.1 count

```
uint32_t Thread_Wait_information::count
```

This field is used to return an integer while when blocked.

Definition at line 400 of file thread.h.

#### 9.239.2.2 Lock

```
struct { ... } Thread_Wait_information::Lock
```

Thread wait lock control block.

Parts of the thread wait information are protected by the thread wait default lock and additionally a thread queue lock in case the thread is enqueued on a thread queue.

The thread wait lock mechanism protects the following thread variables

- POSIX_API_Control::Attributes,
- Scheduler_Node::Wait,
- Thread_Control::Wait::Lock::Pending_requests,
- Thread_Control::Wait::queue, and
- Thread_Control::Wait::operations.

**See also**

_Thread_Wait_acquire(), _Thread_Wait_release(), _Thread_Wait_claim(), _Thread_Wait_restore_default() and _Thread_Wait_tranquilize().

#### 9.239.2.3 operations

```
const Thread_queue_Operations* Thread_Wait_information::operations
```

The current thread queue operations.

This field is protected by the thread lock wait default lock.

**See also**

_Thread_Wait_claim().

Definition at line 491 of file thread.h.

#### 9.239.2.4 option

```
uint32_t Thread_Wait_information::option
```

This field contains any options in effect on this blocking operation.

Definition at line 407 of file thread.h.

#### 9.239.2.5 queue

```
Thread_queue_Queue* Thread_Wait_information::queue
```

The current thread queue.

If this field is NULL the thread is not enqueued on a thread queue. This field is protected by the thread wait default lock.

**See also**

_Thread_Wait_claim().

Definition at line 482 of file thread.h.

#### 9.239.2.6 return_argument

```
void* Thread_Wait_information::return_argument
```

This field is for a pointer to a user return argument.

Definition at line 402 of file thread.h.

#### 9.239.2.7 return_argument_second

```
Thread_Wait_information_Object_argument_type Thread_Wait_information::return_argument_second
```

This field is for a pointer to a second user return argument.

Definition at line 405 of file thread.h.

#### 9.239.2.8 return_code

```
uint32_t Thread_Wait_information::return_code
```

This field will contain the return status from a blocking operation.

**Note**

The following assumes that all API return codes can be treated as an uint32_t.

Definition at line 413 of file thread.h.

#### 9.239.2.9 Tranquilizer

```
Thread_queue_Gate Thread_Wait_information::Tranquilizer
```

Tranquilizer gate used by _Thread_Wait_tranquilize().

This gate is closed by _Thread_Wait_claim(). In case there are no pending requests during a _Thread_Wait_restore_default(), then this gate is opened immediately, otherwise it is placed on the pending request chain and opened by _Thread_Wait_remove_request_locked() as the last gate on the chain to signal overall request completion.

Definition at line 464 of file thread.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/thread.h

## 9.240 Thread_Wait_information_Object_argument_type Union Reference

Union type to hold a pointer to an immutable or a mutable object.

```
#include <thread.h>
```

### Public Attributes

- void * **mutable_object**
- const void * **immutable_object**

### 9.240.1 Detailed Description

Union type to hold a pointer to an immutable or a mutable object.

The main purpose is to enable passing of pointers to read-only send buffers in the message passing subsystem. This approach is somewhat fragile since it prevents the compiler to check if the operations on objects are valid with respect to the constant qualifier. An alternative would be to add a third pointer argument for immutable objects, but this would increase the structure size.

Definition at line 364 of file thread.h.

The documentation for this union was generated from the following file:

- cpukit/include/rtems/score/thread.h

## 9.241 Thread_Zombie_control Struct Reference

### Public Attributes

- Chain_Control **Chain**
- ISR_lock_Control **Lock**

### 9.241.1 Detailed Description

Definition at line 46 of file threadrestart.c.

The documentation for this struct was generated from the following file:

- cpukit/score/src/threadrestart.c

## 9.242 timecounter Struct Reference

### Public Attributes

- timecounter_get_t * **tc_get_timecount**
- uint32_t **tc_counter_mask**
- uint64_t **tc_frequency**
- const char * **tc_name**
- int **tc_quality**

### 9.242.1 Detailed Description

Definition at line 46 of file timetc.h.

The documentation for this struct was generated from the following file:

- cpukit/include/sys/timetc.h

## 9.243 timehands Struct Reference

### Public Attributes

- struct timecounter * **th_counter**
- int64_t **th_adjustment**
- uint64_t **th_scale**
- uint32_t **th_offset_count**
- struct bintime **th_offset**
- struct bintime **th_bintime**
- struct timeval **th_microtime**
- struct timespec **th_nanotime**
- struct bintime **th_boottime**
- Atomic_Uint **th_generation**
- struct timehands * **th_next**

### 9.243.1 Detailed Description

Definition at line 156 of file kern_tc.c.

The documentation for this struct was generated from the following file:

- cpukit/score/src/kern_tc.c

## 9.244 Timer_Control Struct Reference

`#include <timerdata.h>`

### Public Attributes

- Objects_Control Object
- Watchdog_Control Ticker
- Timer_Classes the_class
- rtems_timer_service_routine_entry routine
- void * user_data
- Watchdog_Interval initial
- Watchdog_Interval start_time
- Watchdog_Interval stop_time

### 9.244.1 Detailed Description

The following records define the control block used to manage each timer.

Definition at line 41 of file timerdata.h.

### 9.244.2 Member Data Documentation

#### 9.244.2.1 initial

`Watchdog_Interval Timer_Control::initial`

This field is the timer interval in ticks or seconds.

Definition at line 53 of file timerdata.h.

#### 9.244.2.2 Object

```
Objects_Control Timer_Control::Object
```

This field is the object management portion of a Timer instance.

Definition at line 43 of file timerdata.h.

#### 9.244.2.3 routine

```
rtems_timer_service_routine_entry Timer_Control::routine
```

This field is the timer service routine.

Definition at line 49 of file timerdata.h.

#### 9.244.2.4 start_time

```
Watchdog_Interval Timer_Control::start_time
```

This field is the timer start time point in ticks.

Definition at line 55 of file timerdata.h.

#### 9.244.2.5 stop_time

```
Watchdog_Interval Timer_Control::stop_time
```

This field is the timer stop time point in ticks.

Definition at line 57 of file timerdata.h.

#### 9.244.2.6 the_class

```
Timer_Classes Timer_Control::the_class
```

This field indicates what type of timer this currently is.

Definition at line 47 of file timerdata.h.

#### 9.244.2.7 Ticker

```
Watchdog_Control Timer_Control::Ticker
```

This field is the Watchdog instance which will be the scheduled.

Definition at line 45 of file timerdata.h.

#### 9.244.2.8 user_data

```
void* Timer_Control::user_data
```

This field is the timer service routine user data.

Definition at line 51 of file timerdata.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/timerdata.h

## 9.245 Timer_server_Control Struct Reference

### Public Attributes

- Chain_Control **Pending**
- Objects_Id **server_id**

### 9.245.1 Detailed Description

Definition at line 40 of file timerimpl.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/rtems/timerimpl.h

## 9.246 timex Struct Reference

### Public Attributes

- unsigned int **modes**
- long **offset**
- long **freq**
- long **maxerror**
- long **esterror**
- int **status**
- long **constant**
- long **precision**
- long **tolerance**
- long **ppsfreq**
- long **jitter**
- int **shift**
- long **stabil**
- long **jitcnt**
- long **calcnt**
- long **errcnt**
- long **stbcnt**

### 9.246.1 Detailed Description

Definition at line 129 of file timex.h.

The documentation for this struct was generated from the following file:

- cpukit/include/sys/timex.h

## 9.247 TLS_Dynamic_thread_vector Struct Reference

### Public Attributes

- uint32_t **generation_number**
- void * **tls_blocks** [1]

### 9.247.1 Detailed Description

Definition at line 73 of file tls.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/tls.h

## 9.248 TLS_Index Struct Reference

### Public Attributes

- uintptr_t **module**
- uintptr_t **offset**

### 9.248.1 Detailed Description

Definition at line 98 of file tls.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/tls.h

## 9.249 TLS_Thread_control_block Struct Reference

### Public Attributes

- TLS_Dynamic_thread_vector * **dtv**

### 9.249.1 Detailed Description

Definition at line 83 of file tls.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/tls.h

## 9.250 TOD_Control Struct Reference

TOD control.

`#include <todimpl.h>`

### Public Attributes

- bool is_set
  *Indicates if the time of day is set.*

### 9.250.1 Detailed Description

TOD control.

Definition at line 135 of file todimpl.h.

### 9.250.2 Member Data Documentation

#### 9.250.2.1 is_set

`bool TOD_Control::is_set`

Indicates if the time of day is set.

This is true if the application has set the current time of day, and false otherwise.

Definition at line 142 of file todimpl.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/todimpl.h

## 9.251 TOD_Hook Struct Reference

Structure to manage each TOD action hook.

`#include <todimpl.h>`

### Public Attributes

- Chain_Node Node
- Status_Control(* handler )(TOD_Action, const struct timespec *)

### 9.251.1 Detailed Description

Structure to manage each TOD action hook.

Definition at line 369 of file todimpl.h.

### 9.251.2 Member Data Documentation

#### 9.251.2.1 handler

```
Status_Control( * TOD_Hook::handler) (TOD_Action, const struct timespec *)
```

This is the TOD action hook that is invoked.

Definition at line 374 of file todimpl.h.

#### 9.251.2.2 Node

```
Chain_Node TOD_Hook::Node
```

This is the chain node portion of an object.

Definition at line 371 of file todimpl.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/todimpl.h

## 9.252 ttywakeup Struct Reference

### Public Attributes

- void(∗ **sw_pfn** )(struct termios ∗tty, void ∗arg)
- void ∗ **sw_arg**

### 9.252.1 Detailed Description

Definition at line 43 of file termiostypes.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/termiostypes.h

## 9.253 User_extensions_Control Struct Reference

Manages each user extension set.

`#include <userextdata.h>`

### Public Attributes

- Chain_Node **Node**
- User_extensions_Switch_control **Switch**
- User_extensions_Table **Callouts**

### 9.253.1 Detailed Description

Manages each user extension set.

The switch control is part of the extensions control even if not used due to the extension not having a switch handler.

Definition at line 50 of file userextdata.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/userextdata.h

## 9.254 User_extensions_Fatal_context Struct Reference

### Public Attributes

- Internal_errors_Source **source**
- Internal_errors_t **error**

### 9.254.1 Detailed Description

Definition at line 232 of file userextimpl.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/userextimpl.h

## 9.255 User_extensions_Iterator Struct Reference

Chain iterator for dynamic user extensions.

`#include <userextimpl.h>`

### Public Attributes

- Chain_Iterator **Iterator**
- struct User_extensions_Iterator * **previous**

### 9.255.1 Detailed Description

Chain iterator for dynamic user extensions.

Since user extensions may delete or restart the executing thread, we must clean up registered iterators.

**See also**

_User_extensions_Iterate(), _User_extensions_Destroy_iterators() and Thread_Control::last_user_extensions_iterator.

Definition at line 46 of file userextimpl.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/userextimpl.h

## 9.256 User_extensions_List Struct Reference

### Public Attributes

- Chain_Control Active

  *Active dynamically added user extensions.*
- Chain_Iterator_registry Iterators

  *Chain iterator registration.*

### 9.256.1 Detailed Description

Definition at line 51 of file userextimpl.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/userextimpl.h

## 9.257 User_extensions_Switch_control Struct Reference

Manages the switch callouts.

`#include <userextdata.h>`

### Public Attributes

- Chain_Node **Node**
- User_extensions_thread_switch_extension **thread_switch**

### 9.257.1 Detailed Description

Manages the switch callouts.

They are managed separately from other extensions for performance reasons.

Definition at line 39 of file userextdata.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/userextdata.h

## 9.258 User_extensions_Table Struct Reference

User extension table.

`#include <userext.h>`

### Public Attributes

- User_extensions_thread_create_extension **thread_create**
- User_extensions_thread_start_extension **thread_start**
- User_extensions_thread_restart_extension **thread_restart**
- User_extensions_thread_delete_extension **thread_delete**
- User_extensions_thread_switch_extension **thread_switch**
- User_extensions_thread_begin_extension **thread_begin**
- User_extensions_thread_exitted_extension **thread_exitted**
- User_extensions_fatal_extension **fatal**
- User_extensions_thread_terminate_extension **thread_terminate**

### 9.258.1 Detailed Description

User extension table.

Definition at line 230 of file userext.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/userext.h

## 9.259 User_extensions_Thread_create_context Struct Reference

### Public Attributes

- Thread_Control * **created**
- bool **ok**

### 9.259.1 Detailed Description

Definition at line 149 of file userextimpl.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/userextimpl.h

## 9.260 Watchdog_Control Struct Reference

The control block used to manage each watchdog timer.

`#include <watchdog.h>`

### Public Attributes

- union {
  RBTree_Node RBTree
  *this field is a red-black tree node structure and allows this to be placed on a red-black tree used to manage the scheduled*
  Chain_Node Chain
  *this field is a chain node structure and allows this to be placed on a chain used to manage pending watchdogs by the time*
  } Node

  *Nodes for the watchdog.*
- struct Per_CPU_Control * cpu

  *This field references the processor of this watchdog control.*
- Watchdog_Service_routine_entry routine

  *This field is the function to invoke.*
- uint64_t expire

  *This field is the expiration time point.*

### 9.260.1 Detailed Description

The control block used to manage each watchdog timer.

The following record defines the control block used to manage each watchdog timer.

Definition at line 90 of file watchdog.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/watchdog.h

## 9.261 Watchdog_Header Struct Reference

The watchdog header to manage scheduled watchdogs.

`#include <watchdog.h>`

### Public Attributes

- RBTree_Control Watchdogs

  *Red-black tree of scheduled watchdogs sorted by expiration time.*
- RBTree_Node * first

  *The scheduled watchdog with the earliest expiration time or NULL in case no watchdog is scheduled.*

### 9.261.1 Detailed Description

The watchdog header to manage scheduled watchdogs.

Definition at line 71 of file watchdog.h.

The documentation for this struct was generated from the following file:

- cpukit/include/rtems/score/watchdog.h

# Chapter 10

# File Documentation

## 10.1 bsps/include/bsp/bootcard.h File Reference

```
#include <rtems/config.h>
#include <rtems/bspIo.h>
#include <rtems/malloc.h>
#include <rtems/score/memory.h>
#include <rtems/score/wkspace.h>
#include <bspopts.h>
```

### Functions

- void **bsp_start** (void)
- void **bsp_reset** (void)
- RTEMS_NO_RETURN void boot_card (const char *cmdline)

  *Standard system initialization procedure.*
- void bsp_start_on_secondary_processor (struct Per_CPU_Control *cpu_self)

  *Standard start routine for secondary processors.*

### Variables

- const char * bsp_boot_cmdline

  *Global pointer to the command line of boot_card().*

## 10.2 bsps/include/bsp/default-initial-extension.h File Reference

DEFAULT_INITIAL_EXTENSION Support.

```
#include <rtems.h>
```

### Macros

- #define **BSP_INITIAL_EXTENSION** { NULL, NULL, NULL, NULL, NULL, NULL, NULL, bsp_fatal_extension, NULL }

### Functions

- void **bsp_fatal_extension** (rtems_fatal_source source, bool always_set_to_false, rtems_fatal_code error)

### 10.2.1 Detailed Description

DEFAULT_INITIAL_EXTENSION Support.

## 10.3 cpukit/include/rtems/fatal.h File Reference

This header file defines the Fatal Error Manager API.

```
#include <stdint.h>
#include <rtems/extension.h>
#include <rtems/score/basedefs.h>
#include <rtems/score/cpu.h>
#include <rtems/score/interr.h>
```

### Classes

- struct rtems_assert_context
  %

### Typedefs

- typedef CPU_Exception_frame rtems_exception_frame
  %

### Functions

- RTEMS_NO_RETURN void rtems_fatal_error_occurred (uint32_t the_error)
  %
- static void rtems_exception_frame_print (const rtems_exception_frame *frame)
  %
- static RTEMS_NO_RETURN void rtems_fatal (rtems_fatal_source fatal_source, rtems_fatal_code error_↩ code)
  %
- const char * rtems_internal_error_text (rtems_fatal_code error)
  %
- RTEMS_NO_RETURN RTEMS_PRINTFLIKE (1, 2) void rtems_panic(const char *fmt
  %
- RTEMS_NO_RETURN const char * rtems_fatal_source_text (rtems_fatal_source source)
  %

### 10.3.1 Detailed Description

This header file defines the Fatal Error Manager API.

## 10.4 bsps/include/bsp/irq-generic.h File Reference

Generic BSP interrupt support API.

```
#include <stdbool.h>
#include <rtems/irq-extension.h>
#include <rtems/score/assert.h>
#include <rtems/score/atomic.h>
#include <bsp/irq.h>
```

### Classes

- struct bsp_interrupt_handler_entry

### Macros

- #define **BSP_INTERRUPT_VECTOR_NUMBER** (BSP_INTERRUPT_VECTOR_MAX - BSP_INTERRU↩ PT_VECTOR_MIN + 1)
- #define **BSP_INTERRUPT_HANDLER_TABLE_SIZE** BSP_INTERRUPT_VECTOR_NUMBER
- #define **bsp_interrupt_disable**(level) do { (void) level; } while (0)
- #define **bsp_interrupt_enable**(level) do { } while (0)
- #define **bsp_interrupt_fence**(order) _Atomic_Fence(order)
- #define **bsp_interrupt_assert**(e) _Assert(e)

### Typedefs

- typedef struct bsp_interrupt_handler_entry **bsp_interrupt_handler_entry**

### Functions

- static rtems_vector_number **bsp_interrupt_handler_index** (rtems_vector_number vector)
- static bool bsp_interrupt_is_valid_vector (rtems_vector_number vector)

  *Returns true if the interrupt vector with number vector is valid.*
- void bsp_interrupt_handler_default (rtems_vector_number vector)

  *Default interrupt handler.*
- void bsp_interrupt_initialize (void)

  *Initialize BSP interrupt support.*
- rtems_status_code bsp_interrupt_facility_initialize (void)

  *BSP specific initialization.*
- void bsp_interrupt_vector_enable (rtems_vector_number vector)

  *Enables the interrupt vector with number vector.*
- void bsp_interrupt_vector_disable (rtems_vector_number vector)

  *Disables the interrupt vector with number vector.*
- static void bsp_interrupt_handler_dispatch (rtems_vector_number vector)

  *Sequencially calls all interrupt handlers for the vector number vector.*
- bool bsp_interrupt_handler_is_empty (rtems_vector_number vector)

  *Is interrupt handler empty.*
- void **bsp_interrupt_lock** (void)
- void **bsp_interrupt_unlock** (void)

### Variables

- bsp_interrupt_handler_entry **bsp_interrupt_handler_table** [ ]

### 10.4.1 Detailed Description

Generic BSP interrupt support API.

## 10.5 bsps/include/grlib/ambapp.h File Reference

```
#include "ambapp_ids.h"
```

### Classes

- struct ambapp_common_info
- struct ambapp_apb_info
- struct ambapp_ahb_info
- struct ambapp_dev
- struct ambapp_core
- struct ambapp_ahb_bus
- struct ambapp_mmap
- struct ambapp_bus
- struct ambapp_pnp_ahb
- struct ambapp_pnp_apb

### Macros

- #define **AHB_BUS_MAX** 6
- #define **AMBAPP_FLAG_FFACT_DIR** 0x100 /∗ Frequency factor direction, 0=down, 1=up ∗/
- #define **AMBAPP_FLAG_FFACT** 0x0f0 /∗ Frequency factor against top bus ∗/
- #define **AMBAPP_FLAG_MBUS** 0x00c
- #define **AMBAPP_FLAG_SBUS** 0x003
- #define **DEV_TO_APB**(adev) ((struct ambapp_apb_info ∗)((adev)->devinfo))
- #define **DEV_TO_AHB**(adev) ((struct ambapp_ahb_info ∗)((adev)->devinfo))
- #define **DEV_TO_COMMON**(adev) (((adev)->devinfo))
- #define **DEV_IS_FREE**(dev) (dev->owner == NULL)
- #define **DEV_IS_ALLOCATED**(dev) (dev->owner != NULL)
- #define **OPTIONS_AHB_MSTS** 0x00000001
- #define **OPTIONS_AHB_SLVS** 0x00000002
- #define **OPTIONS_APB_SLVS** 0x00000004
- #define **OPTIONS_ALL_DEVS** (OPTIONS_AHB_MSTS|OPTIONS_AHB_SLVS|OPTIONS_APB_SLVS)
- #define **OPTIONS_FREE** 0x00000010
- #define **OPTIONS_ALLOCATED** 0x00000020
- #define **OPTIONS_ALL** (OPTIONS_FREE|OPTIONS_ALLOCATED)
- #define **OPTIONS_DEPTH_FIRST** 0x00000100
- #define **DEV_AHB_NONE** 0
- #define **DEV_AHB_MST** 1
- #define **DEV_AHB_SLV** 2

- #define **DEV_APB_SLV** 3
- #define **ambapp_pnp_vendor**(id) (((id) >> 24) & 0xff)
- #define **ambapp_pnp_device**(id) (((id) >> 12) & 0xfff)
- #define **ambapp_pnp_ver**(id) (((id)>>5) & 0x1f)
- #define **ambapp_pnp_irq**(id) ((id) & 0x1f)
- #define **ambapp_pnp_start**(mbar) (((mbar) & 0xfff00000) & (((mbar) & 0xfff0) << 16))
- #define **ambapp_pnp_mbar_mask**(mbar) (((mbar)>>4) & 0xfff)
- #define **ambapp_pnp_mbar_type**(mbar) ((mbar) & 0xf)
- #define **ambapp_pnp_apb_start**(iobar, base) ((base) | ((((iobar) & 0xfff00000)>>12) & (((iobar) & 0xfff0)<<4)) )
- #define **ambapp_pnp_apb_mask**(iobar) ((~(ambapp_pnp_mbar_mask(iobar)<<8) & 0x000fffff) + 1)
- #define **AMBA_TYPE_AHBIO_ADDR**(addr, base_ioarea) ((unsigned int)(base_ioarea) | ((addr) >> 12))
- #define **AMBA_TYPE_APBIO** 0x1
- #define **AMBA_TYPE_MEM** 0x2
- #define **AMBA_TYPE_AHBIO** 0x3

## Typedefs

- typedef int(* **ambapp_func_t**) (struct ambapp_dev *dev, int index, void *arg)
- typedef void *(* **ambapp_memcpy_t**) (void *dest, const void *src, int n, struct ambapp_bus *abus)

## Functions

- int **ambapp_scan** (struct ambapp_bus *abus, unsigned int ioarea, ambapp_memcpy_t memfunc, struct ambapp_mmap *mmaps)
- void **ambapp_freq_init** (struct ambapp_bus *abus, struct ambapp_dev *dev, unsigned int freq)
- unsigned int **ambapp_freq_get** (struct ambapp_bus *abus, struct ambapp_dev *dev)
- int **ambapp_for_each** (struct ambapp_bus *abus, unsigned int options, int vendor, int device, ambapp_↩ func_t func, void *arg)
- int **ambapp_find_by_idx** (struct ambapp_dev *dev, int index, void *pcount)
- int **ambapp_dev_count** (struct ambapp_bus *abus, unsigned int options, int vendor, int device)
- void **ambapp_print** (struct ambapp_bus *abus, int show_depth)
- int **ambapp_alloc_dev** (struct ambapp_dev *dev, void *owner)
- void **ambapp_free_dev** (struct ambapp_dev *dev)
- struct ambapp_dev * **ambapp_find_parent** (struct ambapp_dev *dev)
- int **ambapp_depth** (struct ambapp_dev *dev)
- char * **ambapp_device_id2str** (int vendor, int id)
- char * **ambapp_vendor_id2str** (int vendor)
- int **ambapp_vendev_id2str** (int vendor, int id, char *buf)
- int **ambapp_find_apbslv** (struct ambapp_bus *abus, int vendor, int device, struct ambapp_apb_info *dev)
- int **ambapp_find_apbslv_next** (struct ambapp_bus *abus, int vendor, int device, struct ambapp_apb_info *dev, int index)
- int **ambapp_find_apbslvs_next** (struct ambapp_bus *abus, int vendor, int device, struct ambapp_apb_info *dev, int index, int maxno)
- int **ambapp_find_apbslvs** (struct ambapp_bus *abus, int vendor, int device, struct ambapp_apb_info *dev, int maxno)
- int **ambapp_find_ahbslv** (struct ambapp_bus *abus, int vendor, int device, struct ambapp_ahb_info *dev)
- int **ambapp_find_ahbslv_next** (struct ambapp_bus *abus, int vendor, int device, struct ambapp_ahb_info *dev, int index)
- int **ambapp_find_ahbslvs_next** (struct ambapp_bus *abus, int vendor, int device, struct ambapp_ahb_info *dev, int index, int maxno)
- int **ambapp_find_ahbslvs** (struct ambapp_bus *abus, int vendor, int device, struct ambapp_ahb_info *dev, int maxno)
- int **ambapp_get_number_ahbslv_devices** (struct ambapp_bus *abus, int vendor, int device)
- int **ambapp_get_number_apbslv_devices** (struct ambapp_bus *abus, int vendor, int device)

## 10.6 bsps/include/grlib/ambapp_ids.h File Reference

AMBA Plug & Play Bus Vendor and Device IDs.

### Macros

- #define **VENDOR_RESERVED** 0x00
- #define **VENDOR_GAISLER** 0x01
- #define **VENDOR_PENDER** 0x02
- #define **VENDOR_ESA** 0x04
- #define **VENDOR_ASTRIUM** 0x06
- #define **VENDOR_OPENCHIP** 0x07
- #define **VENDOR_OPENCORES** 0x08
- #define **VENDOR_CONTRIB** 0x09
- #define **VENDOR_DLR** 0x0a
- #define **VENDOR_EONIC** 0x0b
- #define **VENDOR_TELECOMPT** 0x0c
- #define **VENDOR_DTU** 0x0d
- #define **VENDOR_BSC** 0x0e
- #define **VENDOR_RADIONOR** 0x0f
- #define **VENDOR_GLEICHMANN** 0x10
- #define **VENDOR_MENTA** 0x11
- #define **VENDOR_SUN** 0x13
- #define **VENDOR_MOVIDIA** 0x14
- #define **VENDOR_ORBITA** 0x17
- #define **VENDOR_SIEMENS** 0x1a
- #define **VENDOR_SYNOPSYS** 0x21
- #define **VENDOR_NASA** 0x22
- #define **VENDOR_NIIET** 0x23
- #define **VENDOR_S3** 0x31
- #define **VENDOR_ACTEL** 0xac
- #define **VENDOR_APPLECORE** 0xae
- #define **VENDOR_C3E** 0xc3
- #define **VENDOR_CBKPAN** 0xc8
- #define **VENDOR_CAL** 0xca
- #define **VENDOR_CETON** 0xcb
- #define **VENDOR_EMBEDDIT** 0xea
- #define **VENDOR_NASA_GSFC** 0xfc
- #define **VENDOR_AZST** 0xfe
- #define **GAISLER_LEON2DSU** 0x002
- #define **GAISLER_LEON3** 0x003
- #define **GAISLER_LEON3DSU** 0x004
- #define **GAISLER_ETHAHB** 0x005
- #define **GAISLER_APBMST** 0x006
- #define **GAISLER_AHBUART** 0x007
- #define **GAISLER_SRCTRL** 0x008
- #define **GAISLER_SDCTRL** 0x009
- #define **GAISLER_SSRCTRL** 0x00a
- #define **GAISLER_I2C2AHB** 0x00b
- #define **GAISLER_APBUART** 0x00c
- #define **GAISLER_IRQMP** 0x00d
- #define **GAISLER_AHBRAM** 0x00e
- #define **GAISLER_AHBDPRAM** 0x00f

- #define **GAISLER_GRIOMMU2** 0x010
- #define **GAISLER_GPTIMER** 0x011
- #define **GAISLER_PCITRG** 0x012
- #define **GAISLER_PCISBRG** 0x013
- #define **GAISLER_PCIFBRG** 0x014
- #define **GAISLER_PCITRACE** 0x015
- #define **GAISLER_DMACTRL** 0x016
- #define **GAISLER_AHBTRACE** 0x017
- #define **GAISLER_DSUCTRL** 0x018
- #define **GAISLER_CANAHB** 0x019
- #define **GAISLER_GPIO** 0x01a
- #define **GAISLER_AHBROM** 0x01b
- #define **GAISLER_AHBJTAG** 0x01c
- #define **GAISLER_ETHMAC** 0x01d
- #define **GAISLER_SWNODE** 0x01e
- #define **GAISLER_SPW** 0x01f
- #define **GAISLER_AHB2AHB** 0x020
- #define **GAISLER_USBDC** 0x021
- #define **GAISLER_USB_DCL** 0x022
- #define **GAISLER_DDRMP** 0x023
- #define **GAISLER_ATACTRL** 0x024
- #define **GAISLER_DDRSP** 0x025
- #define **GAISLER_EHCI** 0x026
- #define **GAISLER_UHCI** 0x027
- #define **GAISLER_I2CMST** 0x028
- #define **GAISLER_SPW2** 0x029
- #define **GAISLER_AHBDMA** 0x02a
- #define **GAISLER_NUHOSP3** 0x02b
- #define **GAISLER_CLKGATE** 0x02c
- #define **GAISLER_SPICTRL** 0x02d
- #define **GAISLER_DDR2SP** 0x02e
- #define **GAISLER_SLINK** 0x02f
- #define **GAISLER_GRTM** 0x030
- #define **GAISLER_GRTC** 0x031
- #define **GAISLER_GRPW** 0x032
- #define **GAISLER_GRCTM** 0x033
- #define **GAISLER_GRHCAN** 0x034
- #define **GAISLER_GRFIFO** 0x035
- #define **GAISLER_GRADCDAC** 0x036
- #define **GAISLER_GRPULSE** 0x037
- #define **GAISLER_GRTIMER** 0x038
- #define **GAISLER_AHB2PP** 0x039
- #define **GAISLER_GRVERSION** 0x03a
- #define **GAISLER_APB2PW** 0x03b
- #define **GAISLER_PW2APB** 0x03c
- #define **GAISLER_GRCAN** 0x03d
- #define **GAISLER_I2CSLV** 0x03e
- #define **GAISLER_U16550** 0x03f
- #define **GAISLER_AHBMST_EM** 0x040
- #define **GAISLER_AHBSLV_EM** 0x041
- #define **GAISLER_GRTESTMOD** 0x042
- #define **GAISLER_ASCS** 0x043
- #define **GAISLER_IPMVBCTRL** 0x044
- #define **GAISLER_SPIMCTRL** 0x045
- #define **GAISLER_L4STAT** 0x047

- #define **GAISLER_LEON4** 0x048
- #define **GAISLER_LEON4DSU** 0x049
- #define **GAISLER_GRPWM** 0x04a
- #define **GAISLER_PWM** 0x04a
- #define **GAISLER_L2CACHE** 0x04b
- #define **GAISLER_SDCTRL64** 0x04c
- #define **GAISLER_GR1553B** 0x04d
- #define **GAISLER_1553TST** 0x04e
- #define **GAISLER_GRIOMMU** 0x04f
- #define **GAISLER_FTAHBRAM** 0x050
- #define **GAISLER_FTSRCTRL** 0x051
- #define **GAISLER_AHBSTAT** 0x052
- #define **GAISLER_LEON3FT** 0x053
- #define **GAISLER_FTMCTRL** 0x054
- #define **GAISLER_FTSDCTRL** 0x055
- #define **GAISLER_FTSRCTRL8** 0x056
- #define **GAISLER_MEMSCRUB** 0x057
- #define **GAISLER_FTSDCTRL64** 0x058
- #define **GAISLER_NANDFCTRL** 0x059
- #define **GAISLER_N2DLLCTRL** 0x05a
- #define **GAISLER_N2PLLCTRL** 0x05b
- #define **GAISLER_SPI2AHB** 0x05c
- #define **GAISLER_DDRSDMUX** 0x05d
- #define **GAISLER_AHBFROM** 0x05e
- #define **GAISLER_PCIEXP** 0x05f
- #define **GAISLER_APBPS2** 0x060
- #define **GAISLER_VGACTRL** 0x061
- #define **GAISLER_LOGAN** 0x062
- #define **GAISLER_SVGACTRL** 0x063
- #define **GAISLER_T1AHB** 0x064
- #define **GAISLER_MP7WRAP** 0x065
- #define **GAISLER_GRSYSMON** 0x066
- #define **GAISLER_GRACECTRL** 0x067
- #define **GAISLER_ATAHBSLV** 0x068
- #define **GAISLER_ATAHBMST** 0x069
- #define **GAISLER_ATAPBSLV** 0x06a
- #define **GAISLER_MIGDDR2** 0x06b
- #define **GAISLER_LCDCTRL** 0x06c
- #define **GAISLER_SWITCHOVER** 0x06d
- #define **GAISLER_FIFOUART** 0x06e
- #define **GAISLER_MUXCTRL** 0x06f
- #define **GAISLER_B1553BC** 0x070
- #define **GAISLER_B1553RT** 0x071
- #define **GAISLER_B1553BRM** 0x072
- #define **GAISLER_GRAES** 0x073
- #define **GAISLER_AES** 0x073
- #define **GAISLER_ECC** 0x074
- #define **GAISLER_PCIF** 0x075
- #define **GAISLER_CLKMOD** 0x076
- #define **GAISLER_HAPSTRAK** 0x077
- #define **GAISLER_TEST_1X2** 0x078
- #define **GAISLER_WILD2AHB** 0x079
- #define **GAISLER_BIO1** 0x07a
- #define **GAISLER_GRAESDMA** 0x07b
- #define **GAISLER_AESDMA** 0x07b

- #define **GAISLER_GRPCI2** 0x07c
- #define **GAISLER_GRPCI2_DMA** 0x07d
- #define **GAISLER_GRPCI2_TB** 0x07e
- #define **GAISLER_MMA** 0x07f
- #define **GAISLER_SATCAN** 0x080
- #define **GAISLER_CANMUX** 0x081
- #define **GAISLER_GRTMRX** 0x082
- #define **GAISLER_GRTCTX** 0x083
- #define **GAISLER_GRTMDESC** 0x084
- #define **GAISLER_GRTMVC** 0x085
- #define **GAISLER_GEFFE** 0x086
- #define **GAISLER_GPREG** 0x087
- #define **GAISLER_GRTMPAHB** 0x088
- #define **GAISLER_SPWCUC** 0x089
- #define **GAISLER_SPW2_DMA** 0x08a
- #define **GAISLER_SPW_ROUTER** 0x08b
- #define **GAISLER_SPWROUTER** 0x08b
- #define **GAISLER_EDCLMST** 0x08c
- #define **GAISLER_GRPWTX** 0x08d
- #define **GAISLER_GRPWRX** 0x08e
- #define **GAISLER_GPREGBANK** 0x08f
- #define **GAISLER_MIG_7SERIES** 0x090
- #define **GAISLER_GRSPW2_SIST** 0x091
- #define **GAISLER_SGMII** 0x092
- #define **GAISLER_RGMII** 0x093
- #define **GAISLER_IRQGEN** 0x094
- #define **GAISLER_GRDMAC** 0x095
- #define **GAISLER_AHB2AVLA** 0x096
- #define **GAISLER_SPWTDP** 0x097
- #define **GAISLER_L3STAT** 0x098
- #define **GAISLER_GR740THS** 0x099
- #define **GAISLER_GRRM** 0x09a
- #define **GAISLER_CMAP** 0x09b
- #define **GAISLER_CPGEN** 0x09c
- #define **GAISLER_AMBAPROT** 0x09d
- #define **GAISLER_IGLOO2_BRIDGE** 0x09e
- #define **GAISLER_AHB2AXI** 0x09f
- #define **GAISLER_AXI2AHB** 0x0a0
- #define **GAISLER_FDIR_RSTCTRL** 0x0a1
- #define **GAISLER_APB3MST** 0x0a2
- #define **GAISLER_LRAM** 0x0a3
- #define **GAISLER_BOOTSEQ** 0x0a4
- #define **GAISLER_TCCOP** 0x0a5
- #define **GAISLER_SPIMASTER** 0x0a6
- #define **GAISLER_SPISLAVE** 0x0a7
- #define **GAISLER_GRSRIO** 0x0a8
- #define **GAISLER_PIPEWRAPPER** 0xffa
- #define **GAISLER_L2TIME** 0xffd /∗ internal device: leon2 timer ∗/
- #define **GAISLER_L2C** 0xffe /∗ internal device: leon2compat ∗/
- #define **GAISLER_PLUGPLAY** 0xfff /∗ internal device: plug & play configarea ∗/
- #define **ESA_LEON2** 0x002
- #define **ESA_LEON2APB** 0x003
- #define **ESA_IRQ** 0x005
- #define **ESA_TIMER** 0x006
- #define **ESA_UART** 0x007

- #define **ESA_CFG** 0x008
- #define **ESA_IO** 0x009
- #define **ESA_MCTRL** 0x00f
- #define **ESA_PCIARB** 0x010
- #define **ESA_HURRICANE** 0x011
- #define **ESA_SPW_RMAP** 0x012
- #define **ESA_SPW2** 0x012
- #define **ESA_AHBUART** 0x013
- #define **ESA_SPWA** 0x014
- #define **ESA_BOSCHCAN** 0x015
- #define **ESA_IRQ2** 0x016
- #define **ESA_AHBSTAT** 0x017
- #define **ESA_WPROT** 0x018
- #define **ESA_WPROT2** 0x019
- #define **ESA_PDEC3AMBA** 0x020
- #define **ESA_PTME3AMBA** 0x021
- #define **OPENCHIP_APBGPIO** 0x001
- #define **OPENCHIP_APBI2C** 0x002
- #define **OPENCHIP_APBSPI** 0x003
- #define **OPENCHIP_APBCHARLCD** 0x004
- #define **OPENCHIP_APBPWM** 0x005
- #define **OPENCHIP_APBPS2** 0x006
- #define **OPENCHIP_APBMMCSD** 0x007
- #define **OPENCHIP_APBNAND** 0x008
- #define **OPENCHIP_APBLPC** 0x009
- #define **OPENCHIP_APBCF** 0x00a
- #define **OPENCHIP_APBSYSACE** 0x00b
- #define **OPENCHIP_APB1WIRE** 0x00c
- #define **OPENCHIP_APBJTAG** 0x00d
- #define **OPENCHIP_APBSUI** 0x00e
- #define **CONTRIB_CORE1** 0x001
- #define **CONTRIB_CORE2** 0x002
- #define **GLEICHMANN_CUSTOM** 0x001
- #define **GLEICHMANN_GEOLCD01** 0x002
- #define **GLEICHMANN_DAC** 0x003
- #define **GLEICHMANN_HPI** 0x004
- #define **GLEICHMANN_SPI** 0x005
- #define **GLEICHMANN_HIFC** 0x006
- #define **GLEICHMANN_ADCDAC** 0x007
- #define **GLEICHMANN_SPIOC** 0x008
- #define **GLEICHMANN_AC97** 0x009
- #define **SUN_T1** 0x001
- #define **SUN_S1** 0x011
- #define **ORBITA_1553B** 0x001
- #define **ORBITA_429** 0x002
- #define **ORBITA_SPI** 0x003
- #define **ORBITA_I2C** 0x004
- #define **ORBITA_SMARTCARD** 0x064
- #define **ORBITA_SDCARD** 0x065
- #define **ORBITA_UART16550** 0x066
- #define **ORBITA_CRYPTO** 0x067
- #define **ORBITA_SYSIF** 0x068
- #define **ORBITA_PIO** 0x069
- #define **ORBITA_RTC** 0x0c8
- #define **ORBITA_COLORLCD** 0x12c

- #define **ORBITA_PCI** 0x190
- #define **ORBITA_DSP** 0x1f4
- #define **ORBITA_USBHOST** 0x258
- #define **ORBITA_USBDEV** 0x2bc
- #define **NASA_EP32** 0x001
- #define **CAL_DDRCTRL** 0x188
- #define **ACTEL_COREMP7** 0x001
- #define **OPENCORES_PCIBR** 0x4
- #define **OPENCORES_ETHMAC** 0x5

### 10.6.1 Detailed Description

AMBA Plug & Play Bus Vendor and Device IDs.

## 10.7 bsps/include/grlib/apbuart.h File Reference

```
#include "ambapp.h"
#include "grlib.h"
```

### Macros

- #define **APBUART_CTRL_RE** 0x1
- #define **APBUART_CTRL_TE** 0x2
- #define **APBUART_CTRL_RI** 0x4
- #define **APBUART_CTRL_TI** 0x8
- #define **APBUART_CTRL_PS** 0x10
- #define **APBUART_CTRL_PE** 0x20
- #define **APBUART_CTRL_FL** 0x40
- #define **APBUART_CTRL_LB** 0x80
- #define **APBUART_CTRL_EC** 0x100
- #define **APBUART_CTRL_TF** 0x200
- #define **APBUART_CTRL_RF** 0x400
- #define **APBUART_CTRL_DB** 0x800
- #define **APBUART_CTRL_BI** 0x1000
- #define **APBUART_CTRL_DI** 0x2000
- #define **APBUART_CTRL_FA** 0x80000000
- #define **APBUART_STATUS_DR** 0x1
- #define **APBUART_STATUS_TS** 0x2
- #define **APBUART_STATUS_TE** 0x4
- #define **APBUART_STATUS_BR** 0x8
- #define **APBUART_STATUS_OV** 0x10
- #define **APBUART_STATUS_PE** 0x20
- #define **APBUART_STATUS_FE** 0x40
- #define **APBUART_STATUS_ERR** 0x78
- #define **APBUART_STATUS_TH** 0x80
- #define **APBUART_STATUS_RH** 0x100
- #define **APBUART_STATUS_TF** 0x200
- #define **APBUART_STATUS_RF** 0x400

### Functions

- void **apbuart_outbyte_polled** (struct apbuart_regs *regs, unsigned char ch, int do_cr_on_newline, int wait↩ _sent)
- int **apbuart_inbyte_nonblocking** (struct apbuart_regs *regs)

## 10.8 bsps/include/grlib/grlib.h File Reference

Common GRLIB AMBA Core Register definitions.

```
#include <stdbool.h>
```

### Classes

- struct mctrl_regs
- struct apbuart_regs
- struct irqmp_timestamp_regs
- struct irqmp_regs
- struct gptimer_timer_regs
- struct gptimer_regs
- struct grgpio_regs
- struct l2c_regs

### Macros

- #define **GPTIMER_TIMER_CTRL_EN** 0x00000001U
- #define **GPTIMER_TIMER_CTRL_RS** 0x00000002U
- #define **GPTIMER_TIMER_CTRL_LD** 0x00000004U
- #define **GPTIMER_TIMER_CTRL_IE** 0x00000008U
- #define **GPTIMER_TIMER_CTRL_IP** 0x00000010U
- #define **GPTIMER_TIMER_CTRL_CH** 0x00000020U
- #define **GPTIMER_TIMER_CTRL_DH** 0x00000040U

### Functions

- static bool **irqmp_has_timestamp** (volatile struct irqmp_timestamp_regs *irqmp_ts)

### 10.8.1 Detailed Description

Common GRLIB AMBA Core Register definitions.

# 10.9 bsps/shared/dev/clock/clockimpl.h File Reference

Clock Tick Device Driver Shell.

```
#include <stdlib.h>
#include <bsp.h>
#include <rtems/clockdrv.h>
#include <rtems/score/percpu.h>
#include <rtems/score/smpimpl.h>
#include <rtems/score/timecounter.h>
#include <rtems/score/thread.h>
#include <rtems/score/watchdogimpl.h>
```

## Macros

- #define Clock_driver_support_install_isr(isr)

  *Do nothing by default.*
- #define Clock_driver_support_find_timer()

  *This method is rarely used so default it.*
- #define Clock_driver_support_at_tick()

  *Do nothing by default.*
- #define Clock_driver_support_set_interrupt_affinity(online_processors)

  *Do nothing by default.*

## Functions

- static void **Clock_driver_timecounter_tick** (void)
- void Clock_isr (void *arg)

  *Clock_isr.*
- void _Clock_Initialize (void)

  *Initialize the clock driver.*

## Variables

- volatile uint32_t Clock_driver_ticks

  *ISRs until next clock tick.*

## 10.9.1 Detailed Description

Clock Tick Device Driver Shell.

## 10.9.2 Function Documentation

### 10.9.2.1 Clock_isr()

```
void Clock_isr (
            void * arg )
```

Clock_isr.

This is the clock tick interrupt handler.

**Parameters**

| | |
|---|---|
| *vector* | Vector number. |

Definition at line 130 of file clockimpl.h.

## 10.10 bsps/shared/irq/irq-generic.c File Reference

Generic BSP interrupt support implementation.

```
#include <bsp/irq-generic.h>
#include <bsp/fatal.h>
#include <stdlib.h>
#include <rtems/score/processormask.h>
#include <rtems/malloc.h>
```

### Functions

- static void **bsp_interrupt_handler_empty** (void *arg)
- static void **bsp_interrupt_handler_do_nothing** (void *arg)
- static bool **bsp_interrupt_is_handler_unique** (rtems_vector_number index)
- static void **bsp_interrupt_set_handler_unique** (rtems_vector_number index, bool unique)
- static bool **bsp_interrupt_is_initialized** (void)
- static void **bsp_interrupt_set_initialized** (void)
- static bool **bsp_interrupt_is_empty_handler_entry** (const bsp_interrupt_handler_entry *e)
- static void **bsp_interrupt_clear_handler_entry** (bsp_interrupt_handler_entry *e, rtems_vector_number vector)
- static bool **bsp_interrupt_allocate_handler_index** (rtems_vector_number vector, rtems_vector_number *index)
- static bsp_interrupt_handler_entry * **bsp_interrupt_allocate_handler_entry** (void)
- static void **bsp_interrupt_free_handler_entry** (bsp_interrupt_handler_entry *e)
- void bsp_interrupt_initialize (void)

  *Initialize BSP interrupt support.*
- static rtems_status_code bsp_interrupt_handler_install (rtems_vector_number vector, const char *info, rtems_option options, rtems_interrupt_handler handler, void *arg)

  *Installs an interrupt handler.*
- static rtems_status_code bsp_interrupt_handler_remove (rtems_vector_number vector, rtems_interrupt_handler handler, void *arg)

  *Removes an interrupt handler.*
- static rtems_status_code bsp_interrupt_handler_iterate (rtems_vector_number vector, rtems_interrupt_per_handler_routine routine, void *arg)

  *Iterates over all installed interrupt handler of a vector.*
- rtems_status_code rtems_interrupt_handler_install (rtems_vector_number vector, const char *info, rtems_option options, rtems_interrupt_handler handler, void *arg)

  *Installs the interrupt handler routine handler for the interrupt vector with number vector.*
- rtems_status_code rtems_interrupt_handler_remove (rtems_vector_number vector, rtems_interrupt_handler handler, void *arg)

  *Removes the interrupt handler routine handler with argument arg for the interrupt vector with number vector.*
- rtems_status_code rtems_interrupt_handler_iterate (rtems_vector_number vector, rtems_interrupt_per_handler_routine routine, void *arg)

*Iterates over all installed interrupt handler of the interrupt vector with number vector.*

- bool bsp_interrupt_handler_is_empty (rtems_vector_number vector)

  *Is interrupt handler empty.*
- rtems_status_code rtems_interrupt_set_affinity (rtems_vector_number vector, size_t affinity_size, const cpu_set_t *affinity)

  *Sets the processor affinity set of an interrupt vector.*
- rtems_status_code rtems_interrupt_get_affinity (rtems_vector_number vector, size_t affinity_size, cpu_set_t *affinity)

  *Gets the processor affinity set of an interrupt vector.*

### Variables

- bsp_interrupt_handler_entry **bsp_interrupt_handler_table** [BSP_INTERRUPT_HANDLER_TABLE_SIZE]
- static uint8_t **bsp_interrupt_handler_unique_table** [(BSP_INTERRUPT_HANDLER_TABLE_SIZE+7+1)/8]

### 10.10.1 Detailed Description

Generic BSP interrupt support implementation.

## 10.11 bsps/shared/irq/irq-lock.c File Reference

BSP interrupt support lock implementation.

```
#include <bsp/irq-generic.h>
#include <rtems/score/apimutex.h>
#include <rtems/score/sysstate.h>
```

### Functions

- void **bsp_interrupt_lock** (void)
- void **bsp_interrupt_unlock** (void)

### 10.11.1 Detailed Description

BSP interrupt support lock implementation.

## 10.12 bsps/shared/start/bootcard.c File Reference

```
#include <bsp/bootcard.h>
#include <rtems.h>
#include <rtems/sysinit.h>
```

### Enumerations

- enum {
  **_Sysinit_bsp_start** = 0x00030080, **_Sysinit_bsp_debug_uart_init** = 0x00030003, **_Sysinit_amba_↩ initialize** = 0x00030001, **_Sysinit_leon3_cpu_index_init** = 0x00030000,
  **_Sysinit_leon3_counter_initialize** = 0x00040000, **_Sysinit_bsp_memory_initialize** = 0x00018080, _↩ **Sysinit__Malloc_Initialize** = 0x00028080, **_Sysinit__Barrier_Manager_initialization** = 0x00140080,
  **_Sysinit__Message_queue_Manager_initialization** = 0x000e0080, **_Sysinit__Partition_Manager_↩ initialization** = 0x00100080, **_Sysinit__Rate_monotonic_Manager_initialization** = 0x00130080, _↩ **Sysinit__Semaphore_Manager_initialization** = 0x000f0080,
  **_Sysinit__RTEMS_tasks_Manager_initialization** = 0x000a0080, **_Sysinit__Timer_Manager_↩ initialization** = 0x000b0080, **_Sysinit_rtems_initialize_data_structures** = 0x00070080, **_Sysinit__↩ Extension_Manager_initialization** = 0x00090080,
  **_Sysinit__Per_CPU_Data_initialize** = 0x00022080, **_Sysinit__Workspace_Handler_initialization** = 0x00026080, **_Sysinit_init_task** = 0x00290080, **_Sysinit_init_runner_task** = 0x00290080 }

### Functions

- void boot_card (const char ∗cmdline)

  *Standard system initialization procedure.*

### Variables

- const char ∗ bsp_boot_cmdline

  *Global pointer to the command line of boot_card().*
- rtems_sysinit_item const **_Linker_set__Sysinit_bsp_start** = { bsp_start }

## 10.13 bsps/shared/start/mallocinitone.c File Reference

_Malloc_Initialize() Implementation

```
#include <rtems/mallocinitone.h>
#include <rtems/score/wkspacedata.h>
```

### Functions

- Heap_Control ∗ _Workspace_Malloc_initialize_separate (void)

  *Initializes the C Program Heap separated from the RTEMS Workspace.*

### Variables

- static Heap_Control **_Malloc_Heap**

### 10.13.1 Detailed Description

_Malloc_Initialize() Implementation

## 10.14 bsps/shared/start/wkspaceinitone.c File Reference

_Workspace_Handler_initialization() Implementation

```
#include <rtems/score/wkspaceinitone.h>
```

### Functions

- void _Workspace_Handler_initialization (void)

  *Initializes the workspace handler.*

### 10.14.1 Detailed Description

_Workspace_Handler_initialization() Implementation

## 10.15 bsps/sparc/leon3/include/amba.h File Reference

```
#include <grlib/ambapp.h>
#include <grlib/grlib.h>
```

### Macros

- #define **LEON3_IO_AREA** 0xfff00000
- #define **LEON3_CONF_AREA** 0xff000
- #define **LEON3_AHB_SLAVE_CONF_AREA** (1 << 11)
- #define **LEON3_AHB_CONF_WORDS** 8
- #define **LEON3_APB_CONF_WORDS** 2
- #define **LEON3_AHB_MASTERS** 64
- #define **LEON3_AHB_SLAVES** 64
- #define **LEON3_APB_SLAVES** 16

### Variables

- struct ambapp_bus **ambapp_plb**

## 10.16 bsps/sparc/leon3/include/bsp.h File Reference

Global BSP definitions.

```
#include <bspopts.h>
#include <bsp/default-initial-extension.h>
#include <rtems.h>
#include <leon.h>
#include <rtems/irq-extension.h>
```

## Macros

- #define **LEON3** 1
- #define **BSP_IDLE_TASK_BODY** bsp_idle_thread
- #define **BSP_NUMBER_OF_TERMIOS_PORTS** 8
- #define **RTEMS_BSP_NETWORK_DRIVER_NAME_OPENETH** "open_eth1"
- #define **RTEMS_BSP_NETWORK_DRIVER_ATTACH_OPENETH** rtems_leon_open_eth_driver_attach
- #define **RTEMS_BSP_NETWORK_DRIVER_NAME_SMC91111** "smc_eth1"
- #define **RTEMS_BSP_NETWORK_DRIVER_ATTACH_SMC91111** rtems_smc91111_driver_attach_leon3
- #define **RTEMS_BSP_NETWORK_DRIVER_NAME_GRETH** "gr_eth1"
- #define **RTEMS_BSP_NETWORK_DRIVER_ATTACH_GRETH** rtems_leon_greth_driver_attach
- #define **RTEMS_BSP_NETWORK_DRIVER_NAME** RTEMS_BSP_NETWORK_DRIVER_NAME_GRETH
- #define **RTEMS_BSP_NETWORK_DRIVER_ATTACH** RTEMS_BSP_NETWORK_DRIVER_ATTACH_G↩ RETH
- #define **HAS_SMC91111**
- #define **GRETH_SUPPORTED**
- #define **GRETH_MEM_LOAD**(addr) leon_r32_no_cache((uintptr_t)addr)
- #define **AMBAPPBUS_INFO_AVAIL** /∗ AMBAPP Bus driver ∗/
- #define **APBUART_INFO_AVAIL** /∗ APBUART Console driver ∗/
- #define **GPTIMER_INFO_AVAIL** /∗ GPTIMER Timer driver ∗/
- #define **GRETH_INFO_AVAIL** /∗ GRETH Ethernet driver ∗/

## Typedefs

- typedef void(∗ **bsp_shared_isr**) (void ∗arg)

## Functions

- void ∗ **bsp_idle_thread** (uintptr_t ignored)
- int **rtems_leon_open_eth_driver_attach** (struct rtems_bsdnet_ifconfig ∗config, int attach)
- int **rtems_smc91111_driver_attach_leon3** (struct rtems_bsdnet_ifconfig ∗config, int attach)
- int **rtems_leon_greth_driver_attach** (struct rtems_bsdnet_ifconfig ∗config, int attach)
- rtems_isr_entry **set_vector** (rtems_isr_entry handler, rtems_vector_number vector, int type)
- void **BSP_fatal_exit** (uint32_t error)
- void **bsp_spurious_initialize** (void)
- void **rtems_bsp_delay** (int usecs)
- void **BSP_shared_interrupt_init** (void)
- void **bsp_isr_handler** (rtems_vector_number vector)
- static __inline__ int **BSP_shared_interrupt_register** (int irq, const char ∗info, bsp_shared_isr isr, void ∗arg)
- static __inline__ int **BSP_shared_interrupt_unregister** (int irq, bsp_shared_isr isr, void ∗arg)
- void **BSP_shared_interrupt_clear** (int irq)
- void **BSP_shared_interrupt_unmask** (int irq)
- void **BSP_shared_interrupt_mask** (int irq)

## Variables

- int **CPU_SPARC_HAS_SNOOPING**
- int **RAM_START**
- int **RAM_END**
- int **RAM_SIZE**
- int **PROM_START**
- int **PROM_END**
- int **PROM_SIZE**
- int **CLOCK_SPEED**
- int **end**
- const unsigned char **LEON3_mp_irq**
- const unsigned char **LEON3_irq_to_cpu** [32]

### 10.16.1 Detailed Description

Global BSP definitions.

## 10.17 cpukit/include/rtems/confdefs/bsp.h File Reference

Evaluate BSP Related Configuration Options.

### 10.17.1 Detailed Description

Evaluate BSP Related Configuration Options.

## 10.18 bsps/sparc/leon3/include/bsp/irq.h File Reference

LEON3 generic shared IRQ setup.

```
#include <leon.h>
#include <rtems/score/processormask.h>
```

### Macros

- #define **BSP_INTERRUPT_VECTOR_MAX_STD** 15 /∗ Standard IRQ controller ∗/
- #define **BSP_INTERRUPT_VECTOR_MAX_EXT** 31 /∗ Extended IRQ controller ∗/
- #define **BSP_INTERRUPT_VECTOR_MIN** 0
- #define **BSP_INTERRUPT_VECTOR_MAX** BSP_INTERRUPT_VECTOR_MAX_EXT
- #define **BSP_INTERRUPT_CUSTOM_VALID_VECTOR**

### Functions

- static bool bsp_interrupt_is_valid_vector (rtems_vector_number vector)

  *Returns true if the interrupt vector with number vector is valid.*
- void **bsp_interrupt_set_affinity** (rtems_vector_number vector, const Processor_mask ∗affinity)
- void **bsp_interrupt_get_affinity** (rtems_vector_number vector, Processor_mask ∗affinity)

### 10.18.1 Detailed Description

LEON3 generic shared IRQ setup.

Based on libbsp/shared/include/irq.h.

## 10.19 bsps/sparc/leon3/include/leon.h File Reference

LEON3 BSP data types and macros.

```
#include <rtems.h>
#include <amba.h>
```

### Macros

- #define **LEON_INTERRUPT_EXTERNAL_1** 5
- #define **LEON_TRAP_TYPE**(_source) SPARC_INTERRUPT_SOURCE_TO_TRAP( _source )
- #define **LEON_TRAP_SOURCE**(_trap) SPARC_INTERRUPT_TRAP_TO_SOURCE( _trap )
- #define **LEON_INT_TRAP**(_trap) SPARC_IS_INTERRUPT_TRAP( _trap )
- #define **LEON_MEMORY_CONFIGURATION_PROM_SIZE_MASK** 0x0003C000
- #define **LEON_MEMORY_CONFIGURATION_RAM_SIZE_MASK** 0x00001E00
- #define **LEON_REG_TIMER_CONTROL_EN** 0x00000001 /* 1 = enable counting */
- #define **LEON_REG_TIMER_CONTROL_RL** 0x00000002 /* 1 = reload at 0 */
- #define **LEON_REG_TIMER_CONTROL_LD** 0x00000004 /* 1 = load counter */
- #define **LEON_REG_UART_CONTROL_RTD** 0x000000FF /* RX/TX data */
- #define **LEON_REG_UART_STATUS_DR** 0x00000001 /* Data Ready */
- #define **LEON_REG_UART_STATUS_TSE** 0x00000002 /* TX Send Register Empty */
- #define **LEON_REG_UART_STATUS_THE** 0x00000004 /* TX Hold Register Empty */
- #define **LEON_REG_UART_STATUS_BR** 0x00000008 /* Break Error */
- #define **LEON_REG_UART_STATUS_OE** 0x00000010 /* RX Overrun Error */
- #define **LEON_REG_UART_STATUS_PE** 0x00000020 /* RX Parity Error */
- #define **LEON_REG_UART_STATUS_FE** 0x00000040 /* RX Framing Error */
- #define **LEON_REG_UART_STATUS_TF** 0x00000200 /* FIFO Full */
- #define **LEON_REG_UART_STATUS_ERR** 0x00000078 /* Error Mask */
- #define **LEON_REG_UART_CTRL_RE** 0x00000001 /* Receiver enable */
- #define **LEON_REG_UART_CTRL_TE** 0x00000002 /* Transmitter enable */
- #define **LEON_REG_UART_CTRL_RI** 0x00000004 /* Receiver interrupt enable */
- #define **LEON_REG_UART_CTRL_TI** 0x00000008 /* Transmitter interrupt enable */
- #define **LEON_REG_UART_CTRL_PS** 0x00000010 /* Parity select */
- #define **LEON_REG_UART_CTRL_PE** 0x00000020 /* Parity enable */
- #define **LEON_REG_UART_CTRL_FL** 0x00000040 /* Flow control enable */
- #define **LEON_REG_UART_CTRL_LB** 0x00000080 /* Loop Back enable */
- #define **LEON_REG_UART_CTRL_DB** 0x00000800 /* Debug FIFO enable */
- #define **LEON_REG_UART_CTRL_SI** 0x00004000 /* TX shift register empty IRQ enable */
- #define **LEON_REG_UART_CTRL_FA** 0x80000000 /* FIFO Available */
- #define **LEON_REG_UART_CTRL_FA_BIT** 31
- #define **LEON3_REG_CACHE_CTRL_FI** 0x00200000 /* Flush instruction cache */
- #define **LEON3_REG_CACHE_CTRL_DS** 0x00800000 /* Data cache snooping */
- #define **LEON3_IRQMPSTATUS_CPUNR** 28
- #define **LEON3_IRQMPSTATUS_BROADCAST** 27
- #define **LEON3_IRQCTRL_ACQUIRE**(_lock_context) rtems_interrupt_lock_acquire( &LEON3_IrqCtrl_↩ Lock, _lock_context )
- #define **LEON3_IRQCTRL_RELEASE**(_lock_context) rtems_interrupt_lock_release( &LEON3_IrqCtrl_↩ Lock, _lock_context )
- #define **LEON_Clear_interrupt**(_source)
- #define **LEON_Force_interrupt**(_source)
- #define **LEON_Enable_interrupt_broadcast**(_source)
- #define **LEON_Disable_interrupt_broadcast**(_source)

- #define **LEON_Is_interrupt_pending**(_source) (LEON3_IrqCtrl_Regs->ipend & (1U << (_source)))
- #define **LEON_Cpu_Is_interrupt_masked**(_source, _cpu) (!(LEON3_IrqCtrl_Regs->mask[_cpu] & (1U << (_source))))
- #define **LEON_Cpu_Mask_interrupt**(_source, _cpu)
- #define **LEON_Cpu_Unmask_interrupt**(_source, _cpu)
- #define **LEON_Cpu_Disable_interrupt**(_source, _previous, _cpu)
- #define **LEON_Cpu_Restore_interrupt**(_source, _previous, _cpu)
- #define **LEON_Is_interrupt_masked**(_source) LEON_Cpu_Is_interrupt_masked(_source, _LEON3_Get↩ _current_processor())
- #define **LEON_Mask_interrupt**(_source) LEON_Cpu_Mask_interrupt(_source, _LEON3_Get_current_↩ processor())
- #define **LEON_Unmask_interrupt**(_source) LEON_Cpu_Unmask_interrupt(_source, _LEON3_Get_↩ current_processor())
- #define **LEON_Disable_interrupt**(_source, _previous) LEON_Cpu_Disable_interrupt(_source, _previous, _LEON3_Get_current_processor())
- #define **LEON_Restore_interrupt**(_source, _previous) LEON_Cpu_Restore_interrupt(_source, _previous, _LEON3_Get_current_processor())
- #define **BSP_Clear_interrupt**(_source) LEON_Clear_interrupt(_source)
- #define **BSP_Force_interrupt**(_source) LEON_Force_interrupt(_source)
- #define **BSP_Is_interrupt_pending**(_source) LEON_Is_interrupt_pending(_source)
- #define **BSP_Is_interrupt_masked**(_source) LEON_Is_interrupt_masked(_source)
- #define **BSP_Unmask_interrupt**(_source) LEON_Unmask_interrupt(_source)
- #define **BSP_Mask_interrupt**(_source) LEON_Mask_interrupt(_source)
- #define **BSP_Disable_interrupt**(_source, _previous) LEON_Disable_interrupt(_source, _prev)
- #define **BSP_Restore_interrupt**(_source, _previous) LEON_Restore_interrupt(_source, _previous)
- #define **BSP_Cpu_Is_interrupt_masked**(_source, _cpu) LEON_Cpu_Is_interrupt_masked(_source, _cpu)
- #define **BSP_Cpu_Unmask_interrupt**(_source, _cpu) LEON_Cpu_Unmask_interrupt(_source, _cpu)
- #define **BSP_Cpu_Mask_interrupt**(_source, _cpu) LEON_Cpu_Mask_interrupt(_source, _cpu)
- #define **BSP_Cpu_Disable_interrupt**(_source, _previous, _cpu) LEON_Cpu_Disable_interrupt(_source, _prev, _cpu)
- #define **BSP_Cpu_Restore_interrupt**(_source, _previous, _cpu) LEON_Cpu_Restore_interrupt(_source, _previous, _cpu)
- #define **LEON_REG_TIMER_COUNTER_RELOAD_AT_ZERO** 0x00000002
- #define **LEON_REG_TIMER_COUNTER_STOP_AT_ZERO** 0x00000000
- #define **LEON_REG_TIMER_COUNTER_LOAD_COUNTER** 0x00000004
- #define **LEON_REG_TIMER_COUNTER_ENABLE_COUNTING** 0x00000001
- #define **LEON_REG_TIMER_COUNTER_DISABLE_COUNTING** 0x00000000
- #define **LEON_REG_TIMER_COUNTER_RELOAD_MASK** 0x00000002
- #define **LEON_REG_TIMER_COUNTER_ENABLE_MASK** 0x00000001
- #define **LEON_REG_TIMER_COUNTER_DEFINED_MASK** 0x00000003
- #define **LEON_REG_TIMER_COUNTER_CURRENT_MODE_MASK** 0x00000003
- #define **LEON3_CLOCK_INDEX** 0
- #define **LEON3_COUNTER_GPTIMER_INDEX** (LEON3_CLOCK_INDEX + 1)
- #define **LEON3_GPTIMER_0_FREQUENCY_SET_BY_BOOT_LOADER** 1000000

## Functions

- static __inline__ int **bsp_irq_fixup** (int irq)
- static unsigned int **leon_r32_no_cache** (uintptr_t addr)
- void **leon3_ext_irq_init** (void)
- RTEMS_NO_RETURN void **leon3_power_down_loop** (void)
- static uint32_t **leon3_get_cpu_count** (volatile struct irqmp_regs *irqmp)
- static void **leon3_set_system_register** (uint32_t addr, uint32_t val)
- static uint32_t **leon3_get_system_register** (uint32_t addr)

- static void **leon3_set_cache_control_register** (uint32_t val)
- static uint32_t **leon3_get_cache_control_register** (void)
- static bool **leon3_data_cache_snooping_enabled** (void)
- static uint32_t **leon3_get_inst_cache_config_register** (void)
- static uint32_t **leon3_get_data_cache_config_register** (void)
- static uint32_t **leon3_up_counter_low** (void)
- static uint32_t **leon3_up_counter_high** (void)
- static void **leon3_up_counter_enable** (void)
- static bool **leon3_up_counter_is_available** (void)
- static uint32_t **leon3_up_counter_frequency** (void)

## Variables

- volatile struct irqmp_regs * **LEON3_IrqCtrl_Regs**
- struct ambapp_dev * **LEON3_IrqCtrl_Adev**
- volatile struct gptimer_regs * **LEON3_Timer_Regs**
- struct ambapp_dev * **LEON3_Timer_Adev**
- uint32_t **LEON3_Cpu_Index**
- int **LEON3_IrqCtrl_Eirq**
- rtems_interrupt_lock **LEON3_IrqCtrl_Lock**
- int **syscon_uart_index**
- int **leon3_debug_uart_index**
- int **leon3_timer_core_index**
- unsigned int **leon3_timer_prescaler**

### 10.19.1 Detailed Description

LEON3 BSP data types and macros.

### 10.19.2 Macro Definition Documentation

#### 10.19.2.1 LEON_Clear_interrupt

```
#define LEON_Clear_interrupt(
              _source )
```

**Value:**

```
do { \
  LEON3_IrqCtrl_Regs->iclear = (1U << (_source)); \
} while (0)
```

Definition at line 185 of file leon.h.

#### 10.19.2.2 LEON_Cpu_Disable_interrupt

```
#define LEON_Cpu_Disable_interrupt(
              _source,
              _previous,
              _cpu )
```

**Value:**

```
  do { \
    rtems_interrupt_lock_context _lock_context; \
    uint32_t _mask = 1U « (_source); \
    LEON3_IRQCTRL_ACQUIRE( &_lock_context ); \
     (_previous) = LEON3_IrqCtrl_Regs->mask[_cpu]; \
     LEON3_IrqCtrl_Regs->mask[_cpu] = _previous & ~_mask; \
    LEON3_IRQCTRL_RELEASE( &_lock_context ); \
    (_previous) &= _mask; \
  } while (0)
```

Definition at line 235 of file leon.h.

#### 10.19.2.3 LEON_Cpu_Mask_interrupt

```
#define LEON_Cpu_Mask_interrupt(
              _source,
              _cpu )
```

**Value:**

```
  do { \
    rtems_interrupt_lock_context _lock_context; \
    LEON3_IRQCTRL_ACQUIRE( &_lock_context ); \
     LEON3_IrqCtrl_Regs->mask[_cpu]  &= ~(1U « (_source)); \
    LEON3_IRQCTRL_RELEASE( &_lock_context ); \
  } while (0)
```

Definition at line 219 of file leon.h.

#### 10.19.2.4 LEON_Cpu_Restore_interrupt

```
#define LEON_Cpu_Restore_interrupt(
              _source,
              _previous,
              _cpu )
```

**Value:**

```
  do { \
    rtems_interrupt_lock_context _lock_context; \
    uint32_t _mask = 1U « (_source); \
    LEON3_IRQCTRL_ACQUIRE( &_lock_context ); \
      LEON3_IrqCtrl_Regs->mask[_cpu] = \
        (LEON3_IrqCtrl_Regs->mask[_cpu] & ~_mask) | (_previous); \
    LEON3_IRQCTRL_RELEASE( &_lock_context ); \
  } while (0)
```

Definition at line 246 of file leon.h.

#### 10.19.2.5 LEON_Cpu_Unmask_interrupt

```
#define LEON_Cpu_Unmask_interrupt(
              _source,
              _cpu )
```

**Value:**

```
  do { \
    rtems_interrupt_lock_context _lock_context; \
    LEON3_IRQCTRL_ACQUIRE( &_lock_context ); \
    LEON3_IrqCtrl_Regs->mask[_cpu]  |= (1U « (_source)); \
    LEON3_IRQCTRL_RELEASE( &_lock_context ); \
  } while (0)
```

Definition at line 227 of file leon.h.

#### 10.19.2.6 LEON_Disable_interrupt_broadcast

```
#define LEON_Disable_interrupt_broadcast(
              _source )
```

**Value:**

```
  do { \
    rtems_interrupt_lock_context _lock_context; \
    uint32_t _mask = 1U « ( _source ); \
    LEON3_IRQCTRL_ACQUIRE( &_lock_context ); \
    LEON3_IrqCtrl_Regs->bcast &= ~_mask; \
    LEON3_IRQCTRL_RELEASE( &_lock_context ); \
  } while (0)
```

Definition at line 204 of file leon.h.

#### 10.19.2.7 LEON_Enable_interrupt_broadcast

```
#define LEON_Enable_interrupt_broadcast(
              _source )
```

**Value:**

```
  do { \
    rtems_interrupt_lock_context _lock_context; \
    uint32_t _mask = 1U « ( _source ); \
    LEON3_IRQCTRL_ACQUIRE( &_lock_context ); \
    LEON3_IrqCtrl_Regs->bcast |= _mask; \
    LEON3_IRQCTRL_RELEASE( &_lock_context ); \
  } while (0)
```

Definition at line 195 of file leon.h.

#### 10.19.2.8 LEON_Force_interrupt

```
#define LEON_Force_interrupt(
              _source )
```

**Value:**

```
  do { \
    LEON3_IrqCtrl_Regs->iforce = (1U « (_source)); \
  } while (0)
```

Definition at line 190 of file leon.h.

# 10.20 bsps/sparc/leon3/include/tm27.h File Reference

Implementations for interrupt mechanisms for Time Test 27.

## Macros

- #define **SIS_USE_SYNCHRONOUS_TRAP** 0
- #define **TEST_INTERRUPT_SOURCE** LEON_INTERRUPT_EXTERNAL_1
- #define **TEST_VECTOR** LEON_TRAP_TYPE( TEST_INTERRUPT_SOURCE )
- #define **TEST_INTERRUPT_SOURCE2** LEON_INTERRUPT_EXTERNAL_1+1
- #define **TEST_VECTOR2** LEON_TRAP_TYPE( TEST_INTERRUPT_SOURCE2 )
- #define **MUST_WAIT_FOR_INTERRUPT** 1
- #define **Install_tm27_vector**(handler)
- #define **Cause_tm27_intr**()
- #define **Clear_tm27_intr**() LEON_Clear_interrupt( TEST_INTERRUPT_SOURCE )
- #define **Lower_tm27_intr**() /∗ empty ∗/

### 10.20.1 Detailed Description

Implementations for interrupt mechanisms for Time Test 27.

### 10.20.2 Macro Definition Documentation

#### 10.20.2.1 Cause_tm27_intr

```
#define Cause_tm27_intr( )
```

**Value:**

```
  do { \
    LEON_Force_interrupt( TEST_INTERRUPT_SOURCE+(Interrupt_nest»1)); \
    nop(); \
    nop(); \
    nop(); \
  } while (0)
```

Definition at line 69 of file tm27.h.

#### 10.20.2.2 Install_tm27_vector

```
#define Install_tm27_vector(
              handler )
```

**Value:**

```
  set_vector( (handler), TEST_VECTOR, 1 ); \
  set_vector( (handler), TEST_VECTOR2, 1 );
```

Definition at line 65 of file tm27.h.

## 10.21 bsps/sparc/leon3/start/bsp_fatal_halt.c File Reference

LEON3 BSP Fatal_halt handler.

```
#include <bsp.h>
#include <leon.h>
```

### Functions

- void _CPU_Fatal_halt (uint32_t source, uint32_t error)

### 10.21.1 Detailed Description

LEON3 BSP Fatal_halt handler.



### 10.21.2 Function Documentation

#### 10.21.2.1 _CPU_Fatal_halt()

```
void _CPU_Fatal_halt (
            uint32_t source,
            uint32_t error )
```

This routine copies _error into a known place – typically a stack location or a register, optionally disables interrupts, and halts/stops the CPU.

Definition at line 31 of file bsp_fatal_halt.c.

## 10.22 bsps/sparc/leon3/start/bspclean.c File Reference

LEON3 BSP fatal extension.

```
#include <bsp.h>
#include <bsp/bootcard.h>
#include <rtems/bspIo.h>
#include <rtems/score/smpimpl.h>
```

### Functions

- void **bsp_fatal_extension** (rtems_fatal_source source, bool always_set_to_false, rtems_fatal_code code)

### 10.22.1 Detailed Description

LEON3 BSP fatal extension.



embedded brains GmbH Dornierstr. 4 82178 Puchheim Germany rtems@embedded-brains.de



## 10.23 bsps/sparc/leon3/start/bspdelay.c File Reference

```
#include <bsp.h>
```

### Functions

- void **rtems_bsp_delay** (int usecs)

### 10.23.1 Detailed Description

LEON3 BSP Delay Method

## 10.24 bsps/sparc/leon3/start/bspsmp.c File Reference

LEON3 SMP BSP Support.

```
#include <bsp.h>
#include <bsp/bootcard.h>
#include <bsp/fatal.h>
#include <leon.h>
#include <rtems/bspIo.h>
#include <rtems/score/smpimpl.h>
#include <stdlib.h>
```

### Functions

- uint32_t **_CPU_SMP_Get_current_processor** (void)
- static rtems_isr **bsp_inter_processor_interrupt** (rtems_vector_number vector)
- void bsp_start_on_secondary_processor (Per_CPU_Control *cpu_self)

  *Standard start routine for secondary processors.*
- uint32_t **_CPU_SMP_Initialize** (void)
- bool **_CPU_SMP_Start_processor** (uint32_t cpu_index)
- void **_CPU_SMP_Finalize_initialization** (uint32_t cpu_count)
- void **_CPU_SMP_Prepare_start_multitasking** (void)
- void **_CPU_SMP_Send_interrupt** (uint32_t target_processor_index)

#### 10.24.1 Detailed Description

LEON3 SMP BSP Support.

## 10.25 bsps/sparc/leon3/start/setvec.c File Reference

Install an interrupt vector on SPARC.

```
#include <bsp.h>
```

### Functions

- rtems_isr_entry **set_vector** (rtems_isr_entry handler, rtems_vector_number vector, int type)

#### 10.25.1 Detailed Description

Install an interrupt vector on SPARC.

## 10.26 bsps/sparc/shared/start/bsp_fatal_exit.c File Reference

ERC32/LEON2/LEON3 BSP specific exit handler.

```
#include <bsp.h>
#include <rtems/score/cpu.h>
```

### Functions

- void **BSP_fatal_exit** (uint32_t error)

### 10.26.1 Detailed Description

ERC32/LEON2/LEON3 BSP specific exit handler.

## 10.27 cpukit/include/rtems.h File Reference

This header file defines the RTEMS Classic API.

```
#include <rtems/config.h>
#include <rtems/extension.h>
#include <rtems/fatal.h>
#include <rtems/init.h>
#include <rtems/io.h>
#include <rtems/rtems/barrier.h>
#include <rtems/rtems/cache.h>
#include <rtems/rtems/clock.h>
#include <rtems/rtems/dpmem.h>
#include <rtems/rtems/event.h>
#include <rtems/rtems/intr.h>
#include <rtems/rtems/message.h>
#include <rtems/rtems/object.h>
#include <rtems/rtems/options.h>
#include <rtems/rtems/part.h>
#include <rtems/rtems/ratemon.h>
#include <rtems/rtems/region.h>
#include <rtems/rtems/sem.h>
#include <rtems/rtems/signal.h>
#include <rtems/rtems/status.h>
#include <rtems/rtems/support.h>
#include <rtems/rtems/tasks.h>
#include <rtems/rtems/timer.h>
#include <rtems/rtems/types.h>
```

### 10.27.1 Detailed Description

This header file defines the RTEMS Classic API.

## 10.28 cpukit/include/rtems/assoc.h File Reference

RTEMS Associativity Routines.

```
#include <stddef.h>
#include <stdint.h>
```

### Classes

- struct rtems_assoc_t
- struct rtems_assoc_32_pair

### Macros

- #define **RTEMS_ASSOC_DEFAULT_NAME** "(default)"

### Functions

- const rtems_assoc_t * rtems_assoc_ptr_by_name (const rtems_assoc_t *, const char *)

  *RTEMS Associate Pointer by Name.*
- const rtems_assoc_t * rtems_assoc_ptr_by_remote (const rtems_assoc_t *, uint32_t)

  *RTEMS Associate Pointer by Remote.*
- uint32_t **rtems_assoc_remote_by_local** (const rtems_assoc_t *, uint32_t)
- uint32_t rtems_assoc_local_by_remote (const rtems_assoc_t *, uint32_t)

  *RTEMS Associate Local by Remote.*
- uint32_t rtems_assoc_remote_by_name (const rtems_assoc_t *, const char *)

  *RTEMS Associate Remote by Name.*
- uint32_t rtems_assoc_local_by_name (const rtems_assoc_t *, const char *)

  *RTEMS Associate Local by Name.*
- const char * rtems_assoc_name_by_local (const rtems_assoc_t *, uint32_t)

  *RTEMS Associate Name by Local.*
- const char * rtems_assoc_name_by_remote (const rtems_assoc_t *, uint32_t)

  *RTEMS Associate Name by Remote.*
- uint32_t rtems_assoc_remote_by_local_bitfield (const rtems_assoc_t *, uint32_t)

  *RTEMS Assoc Routines.*
- char * rtems_assoc_name_by_local_bitfield (const rtems_assoc_t *, uint32_t, char *)

  *RTEMS Associate Name by Local Bitfield.*
- char * rtems_assoc_name_by_remote_bitfield (const rtems_assoc_t *, uint32_t, char *)

  *RTEMS Associate Name by Remote Bitfield.*
- uint32_t **rtems_assoc_local_by_remote_bitfield** (const rtems_assoc_t *, uint32_t)
- const rtems_assoc_t * rtems_assoc_ptr_by_local (const rtems_assoc_t *ap, uint32_t local_value)

  *RTEMS Associate Pointer by Local.*
- size_t rtems_assoc_32_to_string (uint32_t value, char *buffer, size_t buffer_size, const rtems_assoc_32_pair *pairs, size_t pair_count, const char *separator, const char *fallback)

  *Converts the specified value into a text representation.*
- size_t rtems_assoc_thread_states_to_string (uint32_t states, char *buffer, size_t buffer_size)

  *Converts the specified thread states into a text representation.*

#### 10.28.1 Detailed Description

RTEMS Associativity Routines.

## 10.29 cpukit/include/rtems/bsplo.h File Reference

Interface to Kernel Print Methods.

```
#include <rtems/score/basedefs.h>
#include <stdarg.h>
```

## Typedefs

- typedef void(∗ BSP_output_char_function_type) (char c)
- typedef int(∗ BSP_polling_getchar_function_type) (void)

## Functions

- int getchark (void)

  *Get Character (kernel I/O)*
- int vprintk (const char ∗fmt, va_list ap)

  *Variable Argument printk()*
- int **rtems_printk_printer** (void ∗ignored, const char ∗format, va_list ap)
- int printk (const char ∗fmt,...) RTEMS_PRINTFLIKE(1

  *Kernel Print.*
- int int putk (const char ∗s)

  *Kernel Put String.*
- void rtems_putc (char c)

  *Kernel Put Character.*
- void rtems_put_char (int c, void ∗arg)

  *Puts the character via rtems_putc().*

## Variables

- BSP_output_char_function_type BSP_output_char
- BSP_polling_getchar_function_type BSP_poll_char

### 10.29.1 Detailed Description

Interface to Kernel Print Methods.

This include file defines the interface to kernel print methods.

### 10.29.2 Typedef Documentation

#### 10.29.2.1 BSP_output_char_function_type

```
typedef void(* BSP_output_char_function_type) (char c)
```

This type defines the prototype for the BSP provided method to print a single character. It is assumed to be polled.

Definition at line 49 of file bsplo.h.

#### 10.29.2.2 BSP_polling_getchar_function_type

```
typedef int(* BSP_polling_getchar_function_type) (void)
```

This type defines the prototype for the BSP provided method to input a single character. It is assumed to be polled.

Definition at line 55 of file bsplo.h.

### 10.29.3 Function Documentation

#### 10.29.3.1 getchark()

```
int getchark (
             void  )
```

Get Character (kernel I/O)

This method polls for a key in the simplest possible fashion from whatever the debug console device is.

**Returns**

If a character is available, it is returned. Otherwise this method returns -1.

**Note**

This method uses the BSP_poll_char pointer to a BSP provided method.

#### 10.29.3.2 printk()

```
int printk (
             const char * fmt,
              ...  )
```

Kernel Print.

This method allows the user to perform a debug printk(). It performs a character translation from "\n" to "\r\n".

**Parameters**

| | | |
|---|---|---|
| in | *fmt* | is a printf()-style format string |

**Returns**

The number of characters output.

#### 10.29.3.3 putk()

```
int int putk (
            const char * s )
```

Kernel Put String.

This method allows the user to perform a debug puts().

**Parameters**

| | | |
|---|---|---|
| in | *s* | is the string to print |

**Returns**

The number of characters output.

#### 10.29.3.4 rtems_put_char()

```
void rtems_put_char (
            int c,
            void * arg )
```

Puts the character via rtems_putc().

This is a compatibility function to support an internal API.

**Parameters**

| | |
|---|---|
| *c* | The character to put. |
| *arg* | Ignored. |

Definition at line 34 of file rtems_put_char.c.

#### 10.29.3.5 rtems_putc()

```
void rtems_putc (
            char c )
```

Kernel Put Character.

This method allows the user to perform a debug putc(). It performs a character translation from "\n" to "\r\n".

**Parameters**

| | | |
|---|---|---|
| in | *c* | is the character to print |

Definition at line 28 of file rtems_putc.c.

#### 10.29.3.6 vprintk()

```
int vprintk (
            const char * fmt,
            va_list ap )
```

Variable Argument printk()

This method allows the user to access printk() functionality with a va_list style argument.

**Parameters**

| | | |
|---|---|---|
| in | *fmt* | is a printf()-style format string |
| in | *ap* | is a va_list pointer to arguments |

**Returns**

The number of characters output.

Definition at line 29 of file vprintk.c.

### 10.29.4 Variable Documentation

#### 10.29.4.1 BSP_output_char

```
BSP_output_char_function_type BSP_output_char  [extern]
```

This variable points to the BSP provided method to output a character for the purposes of debug output.

It must output only the specific character. It must not perform character translations, e.g. "\n" to "\r\n".

Definition at line 99 of file printk_support.c.

#### 10.29.4.2 BSP_poll_char

```
BSP_polling_getchar_function_type BSP_poll_char  [extern]
```

This variable points to the BSP provided method to input a character for the purposes of debug input.

Definition at line 113 of file printk_support.c.

## 10.30 cpukit/include/rtems/chain.h File Reference

Chain API.

```
#include <rtems/score/chainimpl.h>
#include <rtems/rtems/event.h>
```

### Macros

- #define RTEMS_CHAIN_INITIALIZER_EMPTY(name) CHAIN_INITIALIZER_EMPTY( name )

  *Chain initializer for an empty chain with designator name.*
- #define RTEMS_CHAIN_INITIALIZER_ONE_NODE(node) CHAIN_INITIALIZER_ONE_NODE( node )

  *Chain initializer for a chain with one node.*
- #define RTEMS_CHAIN_NODE_INITIALIZER_ONE_NODE_CHAIN(chain) CHAIN_NODE_INITIALIZER_ONE_NODE_CHAI chain )

  *Chain node initializer for a chain containing exactly this node.*
- #define RTEMS_CHAIN_DEFINE_EMPTY(name) rtems_chain_control name = RTEMS_CHAIN_INITIALIZER_EMPTY( name )

  *Chain definition for an empty chain with designator name.*

### Typedefs

- typedef Chain_Node **rtems_chain_node**
- typedef Chain_Control **rtems_chain_control**

### Functions

- rtems_status_code rtems_chain_append_with_notification (rtems_chain_control *chain, rtems_chain_node *node, rtems_id task, rtems_event_set events)

  *Appends the node to the chain and sends the events to the task if the chain was empty before the append.*
- rtems_status_code rtems_chain_prepend_with_notification (rtems_chain_control *chain, rtems_chain_node *node, rtems_id task, rtems_event_set events)

  *Prepends the node to the chain and sends the events to the task if the chain was empty before the prepend.*
- rtems_status_code rtems_chain_get_with_notification (rtems_chain_control *chain, rtems_id task, rtems_event_set events, rtems_chain_node **node)

  *Gets the first node of the chain and sends the events to the task if the chain is empty after the get.*
- rtems_status_code rtems_chain_get_with_wait (rtems_chain_control *chain, rtems_event_set events, rtems_interval timeout, rtems_chain_node **node)

  *Gets the first node of the chain and sends the events to the task if the chain is empty afterwards.*

- static __inline__ void rtems_chain_initialize (rtems_chain_control *the_chain, void *starting_address, size_t number_nodes, size_t node_size)

  *Initialize a chain Header.*
- static __inline__ void rtems_chain_initialize_empty (rtems_chain_control *the_chain)

  *Initialize this chain as empty.*
- static __inline__ void rtems_chain_set_off_chain (rtems_chain_node *node)

  *Set off chain.*
- static __inline__ void rtems_chain_initialize_node (rtems_chain_node *node)

  *Initializes a chain node.*
- static __inline__ bool rtems_chain_is_node_off_chain (const rtems_chain_node *node)

  *Is the node off chain.*
- static __inline__ bool rtems_chain_is_null_node (const rtems_chain_node *the_node)

  *Is the chain node pointer NULL.*
- static __inline__ rtems_chain_node * rtems_chain_head (rtems_chain_control *the_chain)

  *Return pointer to Chain Head.*
- static __inline__ const rtems_chain_node * rtems_chain_immutable_head (const rtems_chain_control *the_chain)

  *Return pointer to immutable Chain Head.*
- static __inline__ rtems_chain_node * rtems_chain_tail (rtems_chain_control *the_chain)

  *Return pointer to Chain Tail.*
- static __inline__ const rtems_chain_node * rtems_chain_immutable_tail (const rtems_chain_control *the_chain)

  *Return pointer to immutable Chain Tail.*
- static __inline__ rtems_chain_node * rtems_chain_first (const rtems_chain_control *the_chain)

  *Return pointer to Chain's First node after the permanent head.*
- static __inline__ const rtems_chain_node * rtems_chain_immutable_first (const rtems_chain_control *the_chain)

  *Return pointer to immutable Chain's First node.*
- static __inline__ rtems_chain_node * rtems_chain_last (const rtems_chain_control *the_chain)

  *Return pointer to Chain's Last node before the permanent tail.*
- static __inline__ const rtems_chain_node * rtems_chain_immutable_last (const rtems_chain_control *the_chain)

  *Return pointer to immutable Chain's Last node.*
- static __inline__ rtems_chain_node * rtems_chain_next (const rtems_chain_node *the_node)

  *Return pointer the next node from this node.*
- static __inline__ const rtems_chain_node * rtems_chain_immutable_next (const rtems_chain_node *the_node)

  *Return pointer the immutable next node from this node.*
- static __inline__ rtems_chain_node * rtems_chain_previous (const rtems_chain_node *the_node)

  *Return pointer the previous node from this node.*
- static __inline__ const rtems_chain_node * rtems_chain_immutable_previous (const rtems_chain_node *the_node)

  *Return pointer the immutable previous node from this node.*
- static __inline__ bool rtems_chain_are_nodes_equal (const rtems_chain_node *left, const rtems_chain_node *right)

  *Are Two nodes equal.*
- static __inline__ bool rtems_chain_is_empty (const rtems_chain_control *the_chain)

  *Is the chain empty.*
- static __inline__ bool rtems_chain_is_first (const rtems_chain_node *the_node)

  *Is this the first node on the chain.*
- static __inline__ bool rtems_chain_is_last (const rtems_chain_node *the_node)

  *Is this the last node on the chain.*

- static __inline__ bool rtems_chain_has_only_one_node (const rtems_chain_control *the_chain)

  *Does this chain have only one node.*
- static __inline__ bool rtems_chain_is_head (const rtems_chain_control *the_chain, const rtems_chain_node *the_node)

  *Is this node the chain head.*
- static __inline__ bool rtems_chain_is_tail (const rtems_chain_control *the_chain, const rtems_chain_node *the_node)

  *Is this node the chain tail.*
- void rtems_chain_extract (rtems_chain_node *the_node)

  *Extract the specified node from a chain.*
- static __inline__ void rtems_chain_extract_unprotected (rtems_chain_node *the_node)

  *Extract the specified node from a chain (unprotected).*
- rtems_chain_node * rtems_chain_get (rtems_chain_control *the_chain)

  *Obtain the first node on a chain.*
- static __inline__ rtems_chain_node * rtems_chain_get_unprotected (rtems_chain_control *the_chain)

  *See _Chain_Get_unprotected().*
- static __inline__ rtems_chain_node * rtems_chain_get_first_unprotected (rtems_chain_control *the_chain)

  *See _Chain_Get_first_unprotected().*
- void rtems_chain_insert (rtems_chain_node *after_node, rtems_chain_node *the_node)

  *Insert a node on a chain.*
- static __inline__ void rtems_chain_insert_unprotected (rtems_chain_node *after_node, rtems_chain_node *the_node)

  *See _Chain_Insert_unprotected().*
- void rtems_chain_append (rtems_chain_control *the_chain, rtems_chain_node *the_node)

  *Append a node on the end of a chain.*
- static __inline__ void rtems_chain_append_unprotected (rtems_chain_control *the_chain, rtems_chain_node *the_node)

  *Append a node on the end of a chain (unprotected).*
- void rtems_chain_prepend (rtems_chain_control *the_chain, rtems_chain_node *the_node)

  *Prepend a node.*
- static __inline__ void rtems_chain_prepend_unprotected (rtems_chain_control *the_chain, rtems_chain_node *the_node)

  *Prepend a node (unprotected).*
- bool rtems_chain_append_with_empty_check (rtems_chain_control *chain, rtems_chain_node *node)

  *Checks if the chain is empty and appends the node.*
- bool rtems_chain_prepend_with_empty_check (rtems_chain_control *chain, rtems_chain_node *node)

  *Checks if the chain is empty and prepends the node.*
- bool rtems_chain_get_with_empty_check (rtems_chain_control *chain, rtems_chain_node **node)

  *Tries to get the first node and check if the chain is empty afterwards.*
- static __inline__ size_t rtems_chain_node_count_unprotected (const rtems_chain_control *chain)

  *Returns the node count of the chain.*

### 10.30.1 Detailed Description

Chain API.

## 10.31 cpukit/include/rtems/score/chain.h File Reference

Chain Handler API.

### Classes

- struct Chain_Node_struct
- struct Chain_Control

### Typedefs

- typedef struct Chain_Node_struct Chain_Node

### 10.31.1 Detailed Description

Chain Handler API.

## 10.32 cpukit/include/rtems/clockdrv.h File Reference

Clock Driver API.

```
#include <stdint.h>
```

### Functions

- void _Clock_Initialize (void)

  *Initialize the clock driver.*

### Variables

- volatile uint32_t Clock_driver_ticks

  *Count of clock driver ticks since system boot or last overflow.*

### 10.32.1 Detailed Description

Clock Driver API.

This file defines the Clock Driver API.

## 10.33 cpukit/include/rtems/confdefs.h File Reference

Evaluate Configuration Options.

```
#include <rtems/confdefs/obsolete.h>
#include <rtems/confdefs/bdbuf.h>
#include <rtems/confdefs/clock.h>
#include <rtems/confdefs/console.h>
#include <rtems/confdefs/extensions.h>
#include <rtems/confdefs/inittask.h>
#include <rtems/confdefs/initthread.h>
#include <rtems/confdefs/iodrivers.h>
#include <rtems/confdefs/libio.h>
#include <rtems/confdefs/libpci.h>
#include <rtems/confdefs/malloc.h>
#include <rtems/confdefs/mpci.h>
#include <rtems/confdefs/newlib.h>
#include <rtems/confdefs/objectsclassic.h>
#include <rtems/confdefs/objectsposix.h>
#include <rtems/confdefs/percpu.h>
#include <rtems/confdefs/scheduler.h>
#include <rtems/confdefs/threads.h>
#include <rtems/confdefs/wkspace.h>
```

### 10.33.1 Detailed Description

Evaluate Configuration Options.

This header file includes a couple of header files which evaluate the configuration options specified by the application. The macros and defines used to configure the system are documented in the Configuring a System chapter of the Classic API User's Guide.

## 10.34 cpukit/include/rtems/confdefs/bdbuf.h File Reference

Evaluate Block Device Cache Configuration Options.

### 10.34.1 Detailed Description

Evaluate Block Device Cache Configuration Options.

It defines

- _CONFIGURE_LIBBLOCK_TASKS and
- _CONFIGURE_LIBBLOCK_TASKS_STACK_EXTRA

for use by other configuration header files.

## 10.35 cpukit/include/rtems/confdefs/clock.h File Reference

Evaluate Clock Configuration Options.

### 10.35.1 Detailed Description

Evaluate Clock Configuration Options.

## 10.36 cpukit/include/rtems/rtems/clock.h File Reference

This header file defines the Clock Manager API.

```
#include <stdbool.h>
#include <stdint.h>
#include <time.h>
#include <rtems/config.h>
#include <sys/_timespec.h>
#include <sys/_timeval.h>
#include <rtems/rtems/status.h>
#include <rtems/rtems/types.h>
#include <rtems/score/watchdogticks.h>
```

### Macros

- #define rtems_clock_get_ticks_per_second() _Watchdog_Ticks_per_second
  %
- #define rtems_clock_get_ticks_since_boot() _Watchdog_Ticks_since_boot
  %

### Functions

- rtems_status_code rtems_clock_get_seconds_since_epoch (rtems_interval *the_interval)
  %
- rtems_status_code rtems_clock_get_tod (rtems_time_of_day *time_buffer)
  %
- rtems_status_code rtems_clock_get_tod_timeval (struct timeval *time)
  %
- rtems_status_code rtems_clock_get_uptime (struct timespec *uptime)
  %
- uint64_t rtems_clock_get_uptime_nanoseconds (void)
  %
- time_t rtems_clock_get_uptime_seconds (void)
  %
- void rtems_clock_get_uptime_timeval (struct timeval *uptime)
  %
- rtems_status_code rtems_clock_set (const rtems_time_of_day *time_buffer)

%

- rtems_status_code rtems_clock_tick (void)

  %

- static bool rtems_clock_tick_before (rtems_interval tick)

  *Returns true if the current ticks counter value indicates a time before the time specified by the tick value and false otherwise.*

- static rtems_interval rtems_clock_tick_later (rtems_interval delta)

  *Returns the ticks counter value delta ticks in the future.*

- static rtems_interval rtems_clock_tick_later_usec (rtems_interval delta_in_usec)

  *Returns the ticks counter value at least delta microseconds in the future.*

- Watchdog_Interval _TOD_To_seconds (const rtems_time_of_day *the_tod)

  %

- bool _TOD_Validate (const rtems_time_of_day *the_tod)

  %

## 10.36.1 Detailed Description

This header file defines the Clock Manager API.

## 10.36.2 Function Documentation

### 10.36.2.1 _TOD_To_seconds()

```
Watchdog_Interval _TOD_To_seconds (
            const rtems_time_of_day * the_tod )
```

%

**Parameters**

| | |
|---|---|
| *the_tod* | % |

Definition at line 45 of file clocktodtoseconds.c.

### 10.36.2.2 _TOD_Validate()

```
bool _TOD_Validate (
            const rtems_time_of_day * the_tod )
```

%

**Parameters**

| | |
|---|---|
| *the_tod* | % |

Definition at line 36 of file clocktodvalidate.c.

## 10.37 cpukit/include/rtems/confdefs/console.h File Reference

Evaluate Console Driver Configuration Options.

### 10.37.1 Detailed Description

Evaluate Console Driver Configuration Options.

## 10.38 cpukit/include/rtems/confdefs/extensions.h File Reference

Evaluate User Extensions Configuration Options.

### 10.38.1 Detailed Description

Evaluate User Extensions Configuration Options.

## 10.39 cpukit/include/rtems/confdefs/inittask.h File Reference

Evaluate User Initialization Task Configuration Options.

### 10.39.1 Detailed Description

Evaluate User Initialization Task Configuration Options.

This header file defines _CONFIGURE_INIT_TASK_STACK_EXTRA for use by other configuration header files.

## 10.40 cpukit/include/rtems/confdefs/initthread.h File Reference

Evaluate POSIX Initialization Thread Configuration Options.

### 10.40.1 Detailed Description

Evaluate POSIX Initialization Thread Configuration Options.

This header file defines _CONFIGURE_POSIX_INIT_THREAD_STACK_EXTRA for use by other configuration header files.

## 10.41 cpukit/include/rtems/confdefs/iodrivers.h File Reference

Evaluate IO Driver Configuration Options.

### 10.41.1 Detailed Description

Evaluate IO Driver Configuration Options.

## 10.42 cpukit/include/rtems/confdefs/libio.h File Reference

Evaluate IO Library Configuration Options.

### 10.42.1 Detailed Description

Evaluate IO Library Configuration Options.

## 10.43 cpukit/include/rtems/libio.h File Reference

Basic IO API.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioccom.h>
#include <sys/statvfs.h>
#include <sys/uio.h>
#include <unistd.h>
#include <termios.h>
#include <rtems.h>
#include <rtems/fs.h>
#include <rtems/chain.h>
#include <rtems/score/atomic.h>
```

### Classes

- struct rtems_filesystem_eval_path_context_t

  *Path evaluation context.*
- struct _rtems_filesystem_operations_table

  *File system operations table.*
- struct _rtems_filesystem_file_handlers_r

  *File system node operations table.*
- struct rtems_filesystem_limits_and_options_t

  *Contain file system specific information which is required to support fpathconf().*
- struct rtems_libio_tt

  *An open file data structure.*

- struct rtems_libio_rw_args_t

  *Paramameter block for read/write.*
- struct rtems_libio_open_close_args_t

  *Parameter block for open/close.*
- struct rtems_libio_ioctl_args_t

  *Parameter block for ioctl.*
- union __rtems_dev_t
- struct rtems_filesystem_mount_table_entry_tt

  *Mount table entry.*
- struct rtems_filesystem_table_t

  *File system table entry.*
- struct rtems_filesystem_mount_configuration
- struct rtems_termios_callbacks

## Macros

- #define **rtems_filesystem_split_dev_t**(_dev, _major, _minor)
- #define rtems_filesystem_pathconf(_mte) ((_mte)->pathconf_limits_and_options)

  *The pathconf setting for a file system.*
- #define rtems_filesystem_type(_mte) ((_mte)->type)

  *The type of file system. Its name.*
- #define rtems_filesystem_mount_point(_mte) ((_mte)->target)

  *The mount point of a file system.*
- #define rtems_filesystem_mount_device(_mte) ((_mte)->dev)

  *The device entry of a file system.*

### Flag Values

- #define **LIBIO_FLAGS_NO_DELAY** 0x0001U /* return immediately if no data */
- #define **LIBIO_FLAGS_READ** 0x0002U /* reading */
- #define **LIBIO_FLAGS_WRITE** 0x0004U /* writing */
- #define **LIBIO_FLAGS_OPEN** 0x0100U /* device is open */
- #define **LIBIO_FLAGS_APPEND** 0x0200U /* all writes append */
- #define **LIBIO_FLAGS_CLOSE_ON_EXEC** 0x0800U /* close on process exec() */
- #define **LIBIO_FLAGS_READ_WRITE** (LIBIO_FLAGS_READ | LIBIO_FLAGS_WRITE)
- #define **LIBIO_FLAGS_REFERENCE_INC** 0x1000U

### Permission Macros

- #define **RTEMS_FS_PERMS_READ** 0x4
- #define **RTEMS_FS_PERMS_WRITE** 0x2
- #define **RTEMS_FS_PERMS_EXEC** 0x1
- #define **RTEMS_FS_PERMS_RWX** (RTEMS_FS_PERMS_READ | RTEMS_FS_PERMS_WRITE | RTEMS_FS_PERMS_EXEC)
- #define **RTEMS_FS_FOLLOW_HARD_LINK** 0x8
- #define **RTEMS_FS_FOLLOW_SYM_LINK** 0x10
- #define **RTEMS_FS_FOLLOW_LINK** (RTEMS_FS_FOLLOW_HARD_LINK | RTEMS_FS_FOLLOW_SYM_LINK)
- #define **RTEMS_FS_MAKE** 0x20
- #define **RTEMS_FS_EXCLUSIVE** 0x40
- #define **RTEMS_FS_ACCEPT_RESIDUAL_DELIMITERS** 0x80
- #define **RTEMS_FS_REJECT_TERMINAL_DOT** 0x100

### File System Types

- #define **RTEMS_FILESYSTEM_TYPE_IMFS** "imfs"
- #define **RTEMS_FILESYSTEM_TYPE_FTPFS** "ftpfs"
- #define **RTEMS_FILESYSTEM_TYPE_TFTPFS** "tftpfs"
- #define **RTEMS_FILESYSTEM_TYPE_NFS** "nfs"
- #define **RTEMS_FILESYSTEM_TYPE_DOSFS** "dosfs"
- #define **RTEMS_FILESYSTEM_TYPE_RFS** "rfs"
- #define **RTEMS_FILESYSTEM_TYPE_JFFS2** "jffs2"

## Typedefs

- typedef void(∗ rtems_filesystem_mt_entry_lock_t) (const rtems_filesystem_mount_table_entry_t ∗mt_entry)

  *Locks a file system instance.*
- typedef void(∗ rtems_filesystem_mt_entry_unlock_t) (const rtems_filesystem_mount_table_entry_t ∗mt_↩ entry)

  *Unlocks a file system instance.*
- typedef void(∗ rtems_filesystem_eval_path_t) (rtems_filesystem_eval_path_context_t ∗ctx)

  *Path evaluation.*
- typedef int(∗ rtems_filesystem_link_t) (const rtems_filesystem_location_info_t ∗parentloc, const rtems_filesystem_location_info ∗targetloc, const char ∗name, size_t namelen)

  *Creates a new link for the existing file.*
- typedef int(∗ rtems_filesystem_fchmod_t) (const rtems_filesystem_location_info_t ∗loc, mode_t mode)

  *Changes the mode of a node.*
- typedef int(∗ rtems_filesystem_chown_t) (const rtems_filesystem_location_info_t ∗loc, uid_t owner, gid_↩ t group)

  *Changes owner and group of a node.*
- typedef int(∗ rtems_filesystem_clonenode_t) (rtems_filesystem_location_info_t ∗loc)

  *Clones a location.*
- typedef void(∗ rtems_filesystem_freenode_t) (const rtems_filesystem_location_info_t ∗loc)

  *Frees the location of a node.*
- typedef int(∗ rtems_filesystem_mount_t) (rtems_filesystem_mount_table_entry_t ∗mt_entry)

  *Mounts a file system instance in a mount point (directory).*
- typedef int(∗ rtems_filesystem_fsmount_me_t) (rtems_filesystem_mount_table_entry_t ∗mt_entry, const void ∗data)

  *Initializes a file system instance.*
- typedef int(∗ rtems_filesystem_unmount_t) (rtems_filesystem_mount_table_entry_t ∗mt_entry)

  *Unmounts a file system instance in a mount point (directory).*
- typedef void(∗ rtems_filesystem_fsunmount_me_t) (rtems_filesystem_mount_table_entry_t ∗mt_entry)

  *Destroys a file system instance.*
- typedef bool(∗ rtems_filesystem_are_nodes_equal_t) (const rtems_filesystem_location_info_t ∗a, const rtems_filesystem_location_info_t ∗b)

  *Tests if the node of one location is equal to the node of the other location.*
- typedef int(∗ rtems_filesystem_mknod_t) (const rtems_filesystem_location_info_t ∗parentloc, const char ∗name, size_t namelen, mode_t mode, dev_t dev)

  *Creates a new node.*
- typedef int(∗ rtems_filesystem_rmnod_t) (const rtems_filesystem_location_info_t ∗parentloc, const rtems_filesystem_location_info_t ∗loc)

  *Removes a node.*
- typedef int(∗ rtems_filesystem_utime_t) (const rtems_filesystem_location_info_t ∗loc, time_t actime, time_t modtime)

  *Set node access and modification times.*
- typedef int(∗ rtems_filesystem_symlink_t) (const rtems_filesystem_location_info_t ∗parentloc, const char ∗name, size_t namelen, const char ∗target)

  *Makes a symbolic link to a node.*
- typedef ssize_t(∗ rtems_filesystem_readlink_t) (const rtems_filesystem_location_info_t ∗loc, char ∗buf, size_t bufsize)

  *Reads the contents of a symbolic link.*
- typedef int(∗ rtems_filesystem_rename_t) (const rtems_filesystem_location_info_t ∗oldparentloc, const rtems_filesystem_location_info_t ∗oldloc, const rtems_filesystem_location_info_t ∗newparentloc, const char ∗name, size_t namelen)

  *Renames a node.*

- typedef int(∗ rtems_filesystem_statvfs_t) (const rtems_filesystem_location_info_t ∗loc, struct statvfs ∗buf)

  *Gets file system information.*
- typedef int(∗ rtems_filesystem_open_t) (rtems_libio_t ∗iop, const char ∗path, int oflag, mode_t mode)

  *Opens a node.*
- typedef int(∗ rtems_filesystem_close_t) (rtems_libio_t ∗iop)

  *Closes a node.*
- typedef ssize_t(∗ rtems_filesystem_read_t) (rtems_libio_t ∗iop, void ∗buffer, size_t count)

  *Reads from a node.*
- typedef ssize_t(∗ rtems_filesystem_readv_t) (rtems_libio_t ∗iop, const struct iovec ∗iov, int iovcnt, ssize_t total)

  *Reads an IO vector from a node.*
- typedef ssize_t(∗ rtems_filesystem_write_t) (rtems_libio_t ∗iop, const void ∗buffer, size_t count)

  *Writes to a node.*
- typedef ssize_t(∗ rtems_filesystem_writev_t) (rtems_libio_t ∗iop, const struct iovec ∗iov, int iovcnt, ssize_t total)

  *Writes an IO vector to a node.*
- typedef int(∗ rtems_filesystem_ioctl_t) (rtems_libio_t ∗iop, ioctl_command_t request, void ∗buffer)

  *IO control of a node.*
- typedef off_t(∗ rtems_filesystem_lseek_t) (rtems_libio_t ∗iop, off_t offset, int whence)

  *Moves the read/write file offset.*
- typedef int(∗ rtems_filesystem_fstat_t) (const rtems_filesystem_location_info_t ∗loc, struct stat ∗buf)

  *Gets a node status.*
- typedef int(∗ rtems_filesystem_ftruncate_t) (rtems_libio_t ∗iop, off_t length)

  *Truncates a file to a specified length.*
- typedef int(∗ rtems_filesystem_fsync_t) (rtems_libio_t ∗iop)

  *Synchronizes changes to a file.*
- typedef int(∗ rtems_filesystem_fdatasync_t) (rtems_libio_t ∗iop)

  *Synchronizes the data of a file.*
- typedef int(∗ rtems_filesystem_fcntl_t) (rtems_libio_t ∗iop, int cmd)

  *File control.*
- typedef int(∗ rtems_filesystem_poll_t) (rtems_libio_t ∗iop, int events)

  *Poll and select support.*
- typedef int(∗ rtems_filesystem_kqfilter_t) (rtems_libio_t ∗iop, struct knote ∗kn)

  *Kernel event filter support.*
- typedef int(∗ rtems_filesystem_mmap_t) (rtems_libio_t ∗iop, void ∗∗addr, size_t len, int prot, off_t off)

  *MMAP support.*
- typedef off_t **rtems_off64_t**
- typedef struct rtems_filesystem_table_t rtems_filesystem_table_t

  *File system table entry.*
- typedef bool(∗ rtems_per_filesystem_routine) (const rtems_filesystem_table_t ∗fs_entry, void ∗arg)

  *Per file system type routine.*
- typedef bool(∗ rtems_filesystem_mt_entry_visitor) (const rtems_filesystem_mount_table_entry_t ∗mt_entry, void ∗arg)

  *Mount table entry visitor.*
- typedef struct rtems_termios_callbacks **rtems_termios_callbacks**
- typedef rtems_termios_iproc_status_code(∗ rtems_termios_isig_handler) (unsigned char c, struct rtems_termios_tty ∗tty)

  *Type for ISIG (VINTR/VKILL) handler.*

### External I/O Handlers

- typedef int(∗ **rtems_libio_open_t**) (const char ∗pathname, uint32_t flag, uint32_t mode)
- typedef int(∗ **rtems_libio_close_t**) (int fd)
- typedef ssize_t(∗ **rtems_libio_read_t**) (int fd, void ∗buffer, size_t count)
- typedef ssize_t(∗ **rtems_libio_write_t**) (int fd, const void ∗buffer, size_t count)
- typedef int(∗ **rtems_libio_ioctl_t**) (int fd, uint32_t command, void ∗buffer)
- typedef off_t(∗ **rtems_libio_lseek_t**) (int fd, off_t offset, int whence)

## Enumerations

- enum rtems_filesystem_options_t { **RTEMS_FILESYSTEM_READ_ONLY**, **RTEMS_FILESYSTEM_REA↩ D_WRITE**, **RTEMS_FILESYSTEM_BAD_OPTIONS** }

  *File system options.*
- enum rtems_termios_iproc_status_code { RTEMS_TERMIOS_IPROC_CONTINUE, RTEMS_TERMIOS_IPROC_INTERRUPT RTEMS_TERMIOS_IPROC_DONE }

  *The status code returned by all Termios input processing (iproc) functions and input signal (isig) handlers.*

## Functions

- void rtems_filesystem_default_lock (const rtems_filesystem_mount_table_entry_t ∗mt_entry)

  *Obtains the IO library mutex.*
- void rtems_filesystem_default_unlock (const rtems_filesystem_mount_table_entry_t ∗mt_entry)

  *Releases the IO library mutex.*
- void rtems_filesystem_default_eval_path (rtems_filesystem_eval_path_context_t ∗ctx)

  *Terminates the path evaluation and replaces the current location with the null location.*
- int rtems_filesystem_default_link (const rtems_filesystem_location_info_t ∗parentloc, const rtems_filesystem_location_info_t ∗targetloc, const char ∗name, size_t namelen)
- bool rtems_filesystem_default_are_nodes_equal (const rtems_filesystem_location_info_t ∗a, const rtems_filesystem_location_info_t ∗b)

  *Tests if the node access pointer of one location is equal to the node access pointer of the other location.*
- int rtems_filesystem_default_mknod (const rtems_filesystem_location_info_t ∗parentloc, const char ∗name, size_t namelen, mode_t mode, dev_t dev)
- int rtems_filesystem_default_rmnod (const rtems_filesystem_location_info_t ∗parentloc, const rtems_filesystem_location_info_t ∗loc)
- int rtems_filesystem_default_fchmod (const rtems_filesystem_location_info_t ∗loc, mode_t mode)
- int rtems_filesystem_default_chown (const rtems_filesystem_location_info_t ∗loc, uid_t owner, gid_t group)
- int rtems_filesystem_default_clonenode (rtems_filesystem_location_info_t ∗loc)
- void rtems_filesystem_default_freenode (const rtems_filesystem_location_info_t ∗loc)
- int rtems_filesystem_default_mount (rtems_filesystem_mount_table_entry_t ∗mt_entry)
- int rtems_filesystem_default_unmount (rtems_filesystem_mount_table_entry_t ∗mt_entry)
- void rtems_filesystem_default_fsunmount (rtems_filesystem_mount_table_entry_t ∗mt_entry)
- int rtems_filesystem_default_utime (const rtems_filesystem_location_info_t ∗loc, time_t actime, time_t mod-time)
- int rtems_filesystem_default_symlink (const rtems_filesystem_location_info_t ∗parentloc, const char ∗name, size_t namelen, const char ∗target)
- ssize_t rtems_filesystem_default_readlink (const rtems_filesystem_location_info_t ∗loc, char ∗buf, size_↩ t bufsize)
- int rtems_filesystem_default_rename (const rtems_filesystem_location_info_t ∗oldparentloc, const rtems_filesystem_location_info_t ∗oldloc, const rtems_filesystem_location_info_t ∗newparentloc, const char ∗name, size_t namelen)
- int rtems_filesystem_default_statvfs (const rtems_filesystem_location_info_t ∗loc, struct statvfs ∗buf)
- int rtems_filesystem_default_open (rtems_libio_t ∗iop, const char ∗path, int oflag, mode_t mode)
- int rtems_filesystem_default_close (rtems_libio_t ∗iop)
- ssize_t rtems_filesystem_default_read (rtems_libio_t ∗iop, void ∗buffer, size_t count)

- ssize_t rtems_filesystem_default_readv (rtems_libio_t *iop, const struct iovec *iov, int iovcnt, ssize_t total)

  *Calls the read handler for each IO vector entry.*
- ssize_t rtems_filesystem_default_write (rtems_libio_t *iop, const void *buffer, size_t count)
- ssize_t rtems_filesystem_default_writev (rtems_libio_t *iop, const struct iovec *iov, int iovcnt, ssize_t total)

  *Calls the write handler for each IO vector entry.*
- int rtems_filesystem_default_ioctl (rtems_libio_t *iop, ioctl_command_t request, void *buffer)
- off_t rtems_filesystem_default_lseek (rtems_libio_t *iop, off_t offset, int whence)
- off_t rtems_filesystem_default_lseek_directory (rtems_libio_t *iop, off_t offset, int whence)

  *An offset 0 with a whence of SEEK_SET will perform a directory rewind operation.*
- off_t rtems_filesystem_default_lseek_file (rtems_libio_t *iop, off_t offset, int whence)

  *Default lseek() handler for files.*
- int rtems_filesystem_default_fstat (const rtems_filesystem_location_info_t *loc, struct stat *buf)

  *Sets the mode to S_IRWXU | S_IRWXG | S_IRWXO.*
- int rtems_filesystem_default_ftruncate (rtems_libio_t *iop, off_t length)
- int rtems_filesystem_default_ftruncate_directory (rtems_libio_t *iop, off_t length)
- int rtems_filesystem_default_fsync_or_fdatasync (rtems_libio_t *iop)
- int rtems_filesystem_default_fsync_or_fdatasync_success (rtems_libio_t *iop)
- int rtems_filesystem_default_fcntl (rtems_libio_t *iop, int cmd)
- int rtems_filesystem_default_poll (rtems_libio_t *iop, int events)

  *Default poll handler.*
- int rtems_filesystem_default_kqfilter (rtems_libio_t *iop, struct knote *kn)

  *Default kernel event filter handler.*
- int rtems_filesystem_default_mmap (rtems_libio_t *iop, void **addr, size_t len, int prot, off_t off)

  *Default MMAP handler.*
- rtems_filesystem_fsmount_me_t rtems_filesystem_get_mount_handler (const char *type)

  *Gets the mount handler for the file system type.*
- static unsigned int **rtems_libio_iop_flags** (const rtems_libio_t *iop)
- static bool rtems_libio_iop_is_no_delay (const rtems_libio_t *iop)

  *Returns true if this is a no delay iop, otherwise returns false.*
- static bool rtems_libio_iop_is_readable (const rtems_libio_t *iop)

  *Returns true if this is a readable iop, otherwise returns false.*
- static bool rtems_libio_iop_is_writeable (const rtems_libio_t *iop)

  *Returns true if this is a writeable iop, otherwise returns false.*
- static bool rtems_libio_iop_is_append (const rtems_libio_t *iop)

  *Returns true if this is an append iop, otherwise returns false.*
- static dev_t **rtems_filesystem_make_dev_t** (rtems_device_major_number _major, rtems_device_minor_number _minor)
- static dev_t **rtems_filesystem_make_dev_t_from_pointer** (const void *pointer)
- static rtems_device_major_number **rtems_filesystem_dev_major_t** (dev_t device)
- static rtems_device_minor_number **rtems_filesystem_dev_minor_t** (dev_t device)
- void rtems_filesystem_initialize (void)

  *Base File System Initialization.*
- void **rtems_libio_post_driver** (void)
- void **rtems_libio_exit** (void)
- int rtems_mkdir (const char *path, mode_t mode)

  *Creates a directory and all its parent directories according to path.*
- int rtems_filesystem_register (const char *type, rtems_filesystem_fsmount_me_t mount_h)

  *Registers a file system type.*
- int rtems_filesystem_unregister (const char *type)

  *Unregisters a file system type.*
- int unmount (const char *mount_path)

  *Unmounts the file system instance at the specified mount path.*

- int mount (const char *source, const char *target, const char *filesystemtype, rtems_filesystem_options_t options, const void *data)

  *Mounts a file system instance at the specified target path.*
- int mount_and_make_target_path (const char *source, const char *target, const char *filesystemtype, rtems_filesystem_options_t options, const void *data)

  *Mounts a file system and makes the target path.*
- bool rtems_filesystem_iterate (rtems_per_filesystem_routine routine, void *routine_arg)

  *Iterates over all file system types.*
- bool rtems_filesystem_mount_iterate (rtems_filesystem_mt_entry_visitor visitor, void *visitor_arg)

  *Iterates over all file system mount entries.*
- static __inline__ void **rtems_termios_initialize** (void)
- rtems_status_code **rtems_termios_bufsize** (size_t cbufsize, size_t raw_input, size_t raw_output)
- rtems_termios_iproc_status_code rtems_termios_default_isig_handler (unsigned char c, struct rtems_termios_tty *tty)

  *Default handler for ISIG (VINTR/VKILL)*
- rtems_termios_iproc_status_code rtems_termios_posix_isig_handler (unsigned char c, struct rtems_termios_tty *tty)

  *POSIX handler for ISIG (VINTR/VKILL)*
- rtems_status_code rtems_termios_register_isig_handler (rtems_termios_isig_handler handler)

  *Register handler for ISIG (VINTR/VKILL)*
- int **rtems_termios_enqueue_raw_characters** (void *ttyp, const char *buf, int len)
- rtems_status_code **rtems_termios_open** (rtems_device_major_number major, rtems_device_minor_number minor, void *arg, const rtems_termios_callbacks *callbacks)
- rtems_status_code **rtems_termios_close** (void *arg)
- rtems_status_code **rtems_termios_read** (void *arg)
- rtems_status_code **rtems_termios_write** (void *arg)
- rtems_status_code **rtems_termios_ioctl** (void *arg)
- int **rtems_termios_dequeue_characters** (void *ttyp, int len)

## Variables

- const rtems_filesystem_operations_table rtems_filesystem_operations_default

  *File system operations table with default operations.*
- const rtems_filesystem_file_handlers_r rtems_filesystem_handlers_default

  *File system node handler table with default node handlers.*
- const rtems_filesystem_limits_and_options_t rtems_filesystem_default_pathconf

  *Default pathconf settings.*
- const rtems_filesystem_table_t rtems_filesystem_table [ ]

  *Static table of file systems.*
- rtems_chain_control **rtems_filesystem_mount_table**
- const rtems_filesystem_mount_configuration **rtems_filesystem_root_configuration**

### 10.43.1 Detailed Description

Basic IO API.

This file contains the support infrastructure used to manage the table of integer style file descriptors used by the low level POSIX system calls like open(), read, fstat(), etc.

## 10.44 cpukit/include/rtems/confdefs/libpci.h File Reference

This header file evaluates PCI library configuration options.

### 10.44.1 Detailed Description

This header file evaluates PCI library configuration options.

## 10.45 cpukit/include/rtems/confdefs/malloc.h File Reference

Evaluate C Program Heap Configuration Options.

### 10.45.1 Detailed Description

Evaluate C Program Heap Configuration Options.

## 10.46 cpukit/include/rtems/malloc.h File Reference

```
#include <rtems.h>
#include <rtems/bspIo.h>
#include <rtems/libcsupport.h>
#include <stdint.h>
```

### Typedefs

- typedef void *(* **rtems_heap_extend_handler**) (Heap_Control *heap, size_t alloc_size)
- typedef void(* **rtems_malloc_dirtier_t**) (void *, size_t)

### Functions

- void **_Malloc_Initialize** (void)
- static void **rtems_heap_set_sbrk_amount** (ptrdiff_t sbrk_amount)
- void * rtems_heap_extend_via_sbrk (Heap_Control *heap, size_t alloc_size)

  *RTEMS Extend Heap via Sbrk.*
- void * **rtems_heap_null_extend** (Heap_Control *heap, size_t alloc_size)
- void rtems_malloc_dirty_memory (void *start, size_t size)

  *Dirty Memory Function.*
- int rtems_memalign (void **pointer, size_t alignment, size_t size)

  *RTEMS Variation on Aligned Memory Allocation.*
- void * rtems_heap_allocate_aligned_with_boundary (size_t size, uintptr_t alignment, uintptr_t boundary) RTEMS_MALLOCLIKE RTEMS_ALLOC_SIZE(1) RTEMS_ALLOC_ALIGN(2) RTEMS_WARN_UNUSED_RESULT

  *Allocates a memory area of size size bytes from the heap.*
- void * rtems_malloc (size_t size) RTEMS_MALLOCLIKE RTEMS_ALLOC_SIZE(1) RTEMS_WARN_UNUSED_RESULT

*Allocates a memory area of the specified size from the heap.*

- void * rtems_calloc (size_t nelem, size_t elsize) RTEMS_MALLOCLIKE RTEMS_ALLOC_SIZE_2(1

  *Allocates a memory area for the specified count of elements from the heap.*
- rtems_status_code rtems_heap_extend (void *area_begin, uintptr_t area_size)

  *Extends the memory available for the heap using the memory area starting at area_begin of size area_size bytes.*
- void * rtems_heap_greedy_allocate (const uintptr_t *block_sizes, size_t block_count)

  *Greedy allocate that empties the heap.*
- void * rtems_heap_greedy_allocate_all_except_largest (uintptr_t *allocatable_size)

  *Greedy allocate all blocks except the largest free block.*
- void rtems_heap_greedy_free (void *opaque)

  *Frees space of a greedy allocation.*

## Variables

- Heap_Control * RTEMS_Malloc_Heap

  *C program heap control.*
- ptrdiff_t **RTEMS_Malloc_Sbrk_amount**
- const rtems_heap_extend_handler **rtems_malloc_extend_handler**
- rtems_malloc_dirtier_t **rtems_malloc_dirty_helper**
- void **RTEMS_WARN_UNUSED_RESULT**

## 10.46.1 Detailed Description

This file defines the interface to RTEMS extensions to the Malloc Family.

## 10.46.2 Function Documentation

### 10.46.2.1 rtems_calloc()

```
void* rtems_calloc (
            size_t nelem,
            size_t elsize )
```

Allocates a memory area for the specified count of elements from the heap.

The allocated memory area is fully filled with zero bits.

This function is almost identical to calloc(). The only exception is that errno is not set in case of a memory allocation failure.

**Parameters**

| | | |
|---|---|---|
| in | *nelem* | The count of elements. |
| in | *elsize* | The size of each elements. |

**Return values**

| | |
|---|---|
| *NULL* | The memory allocation failed or *nelem* is zero or *elsize* is zero. |
| *otherwise* | The begin address of the allocated memory area. |

#### 10.46.2.2 rtems_heap_allocate_aligned_with_boundary()

```
void* rtems_heap_allocate_aligned_with_boundary (
            size_t size,
            uintptr_t alignment,
            uintptr_t boundary )
```

Allocates a memory area of size *size* bytes from the heap.

If the alignment parameter *alignment* is not equal to zero, the allocated memory area will begin at an address aligned by this value.

If the boundary parameter *boundary* is not equal to zero, the allocated memory area will comply with a boundary constraint. The boundary value specifies the set of addresses which are aligned by the boundary value. The interior of the allocated memory area will not contain an element of this set. The begin or end address of the area may be a member of the set.

A size value of zero will return a unique address which may be freed with free().

The memory allocated by this function can be released with a call to free().

**Returns**

A pointer to the begin of the allocated memory area, or `NULL` if no memory is available or the parameters are inconsistent.

#### 10.46.2.3 rtems_heap_extend()

```
rtems_status_code rtems_heap_extend (
            void * area_begin,
            uintptr_t area_size )
```

Extends the memory available for the heap using the memory area starting at *area_begin* of size *area_size* bytes.

There are no alignment requirements. The memory area must be big enough to contain some maintenance blocks. It must not overlap parts of the current heap areas. Disconnected subordinate heap areas will lead to used blocks which cover the gaps. Extending with an inappropriate memory area will corrupt the heap.

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation. |
| *RTEMS_INVALID_ADDRESS* | Invalid memory area. |

#### 10.46.2.4 rtems_heap_greedy_allocate()

```
void* rtems_heap_greedy_allocate (
            const uintptr_t * block_sizes,
            size_t block_count )
```

Greedy allocate that empties the heap.

Afterwards the heap has at most *block_count* allocatable blocks of sizes specified by *block_sizes*. The *block_sizes* must point to an array with *block_count* members. All other blocks are used.

**See also**

rtems_heap_greedy_free().

#### 10.46.2.5 rtems_heap_greedy_allocate_all_except_largest()

```
void* rtems_heap_greedy_allocate_all_except_largest (
            uintptr_t * allocatable_size )
```

Greedy allocate all blocks except the largest free block.

Afterwards the heap has at most one allocatable block. This block is the largest free block if it exists. The allocatable size of this block is stored in *allocatable_size*. All other blocks are used.

**See also**

rtems_heap_greedy_free().

#### 10.46.2.6 rtems_heap_greedy_free()

```
void rtems_heap_greedy_free (
            void * opaque )
```

Frees space of a greedy allocation.

The *opaque* argument must be the return value of rtems_heap_greedy_allocate() or rtems_heap_greedy_allocate_all_except_largest(

#### 10.46.2.7 rtems_malloc()

```
void* rtems_malloc (
            size_t size )
```

Allocates a memory area of the specified size from the heap.

This function is almost identical to malloc(). The only exception is that errno is not set in case of a memory allocation failure.

**Parameters**

| | | |
|---|---|---|
| in | *size* | The memory area size in bytes. |

**Return values**

| | |
|---|---|
| *NULL* | The memory allocation failed or *size* is zero. |
| *otherwise* | The begin address of the allocated memory area. |

#### 10.46.2.8 rtems_malloc_dirty_memory()

```
void rtems_malloc_dirty_memory (
             void * start,
             size_t size )
```

Dirty Memory Function.

This method fills the specified area with a non-zero pattern to aid in debugging programs which do not initialize their memory allocated from the heap.

#### 10.46.2.9 rtems_memalign()

```
int rtems_memalign (
             void ** pointer,
             size_t alignment,
             size_t size )
```

RTEMS Variation on Aligned Memory Allocation.

This method is a help memalign implementation which does all error checking done by posix_memalign() EXCEPT it does NOT place numeric restrictions on the alignment value.

**Parameters**

| | | |
|---|---|---|
| in | *pointer* | points to the user pointer |
| in | *alignment* | is the desired alignment |
| in | *size* | is the allocation request size in bytes |

**Returns**

This methods returns zero on success and a POSIX errno value to indicate the failure condition. On success ∗pointer will contain the address of the allocated memory.

### 10.46.3 Variable Documentation

#### 10.46.3.1 RTEMS_Malloc_Heap

```
Heap_Control* RTEMS_Malloc_Heap  [extern]
```

C program heap control.

This is the pointer to the heap control structure used to manage the C program heap.

Definition at line 45 of file mallocheap.c.

## 10.47 cpukit/include/rtems/confdefs/mpci.h File Reference

Evaluate MPCI Configuration Options.

### 10.47.1 Detailed Description

Evaluate MPCI Configuration Options.

## 10.48 cpukit/include/rtems/confdefs/newlib.h File Reference

Evaluate Newlib Configuration Options.

### 10.48.1 Detailed Description

Evaluate Newlib Configuration Options.

This header file defines _CONFIGURE_ENABLE_NEWLIB_REENTRANCY for use by other configuration header files.

## 10.49 cpukit/include/rtems/confdefs/objectsclassic.h File Reference

Evaluate Classic API Objects Configuration Options.

### 10.49.1 Detailed Description

Evaluate Classic API Objects Configuration Options.

For the task objects configuration see <rtems/confdefs/threads.h>.

## 10.50 cpukit/include/rtems/confdefs/objectsposix.h File Reference

Evaluate POSIX API Objects Configuration Options.

### 10.50.1 Detailed Description

Evaluate POSIX API Objects Configuration Options.

For the POSIX thread objects configuration see <rtems/confdefs/threads.h>.

## 10.51 cpukit/include/rtems/confdefs/obsolete.h File Reference

Evaluate Obsolete Configuration Options.

### 10.51.1 Detailed Description

Evaluate Obsolete Configuration Options.

## 10.52 cpukit/include/rtems/confdefs/percpu.h File Reference

Evaluate Per-CPU Configuration Options.

### 10.52.1 Detailed Description

Evaluate Per-CPU Configuration Options.

Since the idle thread stack size (CONFIGURE_IDLE_TASK_STACK_SIZE) depends on CONFIGURE_MINIM↩UM_TASK_STACK_SIZE, the POSIX-specific CONFIGURE_POSIX_INIT_THREAD_STACK_SIZE configuration option is also evaluated in this header file for simplicity.

This header file defines _CONFIGURE_MAXIMUM_PROCESSORS for use by other configuration header files.

## 10.53 cpukit/include/rtems/score/percpu.h File Reference

```
#include <rtems/score/cpuimpl.h>
#include <rtems/score/assert.h>
#include <rtems/score/chain.h>
#include <rtems/score/isrlock.h>
#include <rtems/score/smp.h>
#include <rtems/score/timestamp.h>
#include <rtems/score/watchdog.h>
```

## Classes

- struct Per_CPU_Job_context

  *Context for per-processor jobs.*
- struct Per_CPU_Job

  *A per-processor job.*
- struct Per_CPU_Stats

  *Per-CPU statistics.*
- struct Per_CPU_Control

  *Per CPU Core Structure.*
- struct Per_CPU_Control_envelope

## Macros

- #define **PER_CPU_CONTROL_SIZE_APPROX** ( 180 + CPU_PER_CPU_CONTROL_SIZE + CPU_INTERRUPT_FRAME_SIZE )
- #define **PER_CPU_CONTROL_SIZE_LOG2** 8
- #define **PER_CPU_CONTROL_SIZE** ( 1 << PER_CPU_CONTROL_SIZE_LOG2 )
- #define **PER_CPU_JOB_DONE** 1
- #define **_Per_CPU_Acquire**(cpu, lock_context) _ISR_lock_Acquire( &( cpu )->Lock, lock_context )
- #define **_Per_CPU_Release**(cpu, lock_context) _ISR_lock_Release( &( cpu )->Lock, lock_context )
- #define **_Per_CPU_Get_snapshot**() ( &_Per_CPU_Information[ _SMP_Get_current_processor() ].per_cpu )
- #define **_Thread_Dispatch_disable_level** _Per_CPU_Get()->thread_dispatch_disable_level
- #define **_Thread_Heir** _Per_CPU_Get()->heir
- #define **_Thread_Executing** _Per_CPU_Get_executing( _Per_CPU_Get() )
- #define **_ISR_Nest_level** _Per_CPU_Get()->isr_nest_level
- #define **_CPU_Interrupt_stack_low** _Per_CPU_Get()->interrupt_stack_low
- #define **_CPU_Interrupt_stack_high** _Per_CPU_Get()->interrupt_stack_high
- #define **_Thread_Dispatch_necessary** _Per_CPU_Get()->dispatch_necessary

## Typedefs

- typedef void(* **Per_CPU_Job_handler**) (void *arg)
- typedef struct Per_CPU_Job Per_CPU_Job

  *A per-processor job.*
- typedef struct Per_CPU_Control Per_CPU_Control

  *Per CPU Core Structure.*

## Enumerations

- enum Per_CPU_State {
  PER_CPU_STATE_INITIAL, PER_CPU_STATE_READY_TO_START_MULTITASKING, PER_CPU_STATE_REQUEST_STA
  PER_CPU_STATE_UP,
  PER_CPU_STATE_SHUTDOWN }

  *State of a processor.*
- enum Per_CPU_Watchdog_index { PER_CPU_WATCHDOG_TICKS, PER_CPU_WATCHDOG_REALTIME, PER_CPU_WATCHDOG_MONOTONIC, PER_CPU_WATCHDOG_COUNT }

  *Per-CPU watchdog header index.*

### Functions

- static Per_CPU_Control * **_Per_CPU_Get** (void)
- static Per_CPU_Control * **_Per_CPU_Get_by_index** (uint32_t index)
- static uint32_t **_Per_CPU_Get_index** (const Per_CPU_Control *cpu)
- static struct _Thread_Control * **_Per_CPU_Get_executing** (const Per_CPU_Control *cpu)
- static bool **_Per_CPU_Is_processor_online** (const Per_CPU_Control *cpu)
- static bool **_Per_CPU_Is_boot_processor** (const Per_CPU_Control *cpu)
- static __inline__ void **_Per_CPU_Acquire_all** (ISR_lock_Context *lock_context)
- static __inline__ void **_Per_CPU_Release_all** (ISR_lock_Context *lock_context)
- void _Per_CPU_Initialize (void)

  *Allocate and Initialize Per CPU Structures.*
- void **_Per_CPU_State_change** (Per_CPU_Control *cpu, Per_CPU_State new_state)
- bool _Per_CPU_State_wait_for_non_initial_state (uint32_t cpu_index, uint32_t timeout_in_ns)

  *Waits for a processor to change into a non-initial state.*
- void _Per_CPU_Perform_jobs (Per_CPU_Control *cpu)

  *Performs the jobs of the specified processor in FIFO order.*
- void _Per_CPU_Add_job (Per_CPU_Control *cpu, Per_CPU_Job *job)

  *Adds the job to the tail of the processing list of the specified processor.*
- void _Per_CPU_Wait_for_job (const Per_CPU_Control *cpu, const Per_CPU_Job *job)

  *Waits for the job carried out by the specified processor.*
- static __inline__ struct _Thread_Control * _Thread_Get_executing (void)

  *Returns the thread control block of the executing thread.*

### Variables

- Per_CPU_Control_envelope _Per_CPU_Information[ ] CPU_STRUCTURE_ALIGNMENT

  *Set of Per CPU Core Information.*

#### 10.53.1 Detailed Description

This include file defines the per CPU information required by RTEMS.

## 10.54 cpukit/include/rtems/confdefs/scheduler.h File Reference

Evaluate Scheduler Configuration Options.

#### 10.54.1 Detailed Description

Evaluate Scheduler Configuration Options.

## 10.55 cpukit/include/rtems/scheduler.h File Reference

Scheduler Configuration API.

```
#include <rtems/score/scheduler.h>
```

### Macros

- #define **SCHEDULER_CONTEXT_NAME**(name) _Configuration_Scheduler_ ## name
- #define **SCHEDULER_CONTROL_IS_NON_PREEMPT_MODE_SUPPORTED**(value) , value
- #define **RTEMS_SCHEDULER_ASSIGN_DEFAULT** SCHEDULER_ASSIGN_DEFAULT
- #define **RTEMS_SCHEDULER_ASSIGN_PROCESSOR_OPTIONAL** SCHEDULER_ASSIGN_PROCESSOR_OPTIONAL
- #define **RTEMS_SCHEDULER_ASSIGN_PROCESSOR_MANDATORY** SCHEDULER_ASSIGN_PROCESSOR_MANDATO
- #define **RTEMS_SCHEDULER_ASSIGN**(index, attr)
- #define **RTEMS_SCHEDULER_ASSIGN_NO_SCHEDULER** { NULL, 0 }

### Variables

- const Scheduler_Control **RTEMS_SCHEDULER_INVALID_INDEX**

## 10.55.1 Detailed Description

Scheduler Configuration API.

## 10.55.2 Macro Definition Documentation

### 10.55.2.1 RTEMS_SCHEDULER_ASSIGN

```
#define RTEMS_SCHEDULER_ASSIGN(
              index,
              attr )
```

**Value:**

```
{ \
  ( index ) < RTEMS_ARRAY_SIZE( _Scheduler_Table ) ? \
    &_Scheduler_Table[ ( index ) ] : &RTEMS_SCHEDULER_INVALID_INDEX, \
  ( attr ) \
}
```

Definition at line 49 of file scheduler.h.

# 10.56 cpukit/include/rtems/score/scheduler.h File Reference

Constants and Structures Associated with the Scheduler.

```
#include <rtems/score/thread.h>
```

## Classes

- struct Scheduler_Operations

  *The scheduler operations.*
- struct Scheduler_Context

  *Scheduler context.*
- struct _Scheduler_Control

  *Scheduler control.*
- struct Scheduler_Assignment

  *Scheduler assignment.*

## Macros

- #define SCHEDULER_ASSIGN_DEFAULT UINT32_C(0x0)

  *The scheduler assignment default attributes.*
- #define SCHEDULER_ASSIGN_PROCESSOR_OPTIONAL SCHEDULER_ASSIGN_DEFAULT

  *The presence of this processor is optional.*
- #define SCHEDULER_ASSIGN_PROCESSOR_MANDATORY UINT32_C(0x1)

  *The presence of this processor is mandatory.*
- #define **SCHEDULER_OPERATION_DEFAULT_ASK_FOR_HELP**
- #define **SCHEDULER_OPERATION_DEFAULT_GET_SET_AFFINITY** , _Scheduler_default_Set_affinity
- #define PRIORITY_MAXIMUM ( _Scheduler_Table[ 0 ].maximum_priority )

  *This defines the lowest (least important) thread priority of the first scheduler instance.*

## Typedefs

- typedef struct _Scheduler_Control **Scheduler_Control**
- typedef struct Scheduler_Context Scheduler_Context

  *Scheduler context.*

## Functions

- Priority_Control _Scheduler_default_Map_priority (const Scheduler_Control *scheduler, Priority_Control priority)

  *Returns the scheduler internal thread priority mapped by SCHEDULER_PRIORITY_MAP().*
- Priority_Control _Scheduler_default_Unmap_priority (const Scheduler_Control *scheduler, Priority_Control priority)

  *Returns the user visible thread priority unmapped by SCHEDULER_PRIORITY_UNMAP().*
- bool _Scheduler_default_Ask_for_help (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Does nothing.*
- void _Scheduler_default_Reconsider_help_request (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Does nothing.*
- void _Scheduler_default_Withdraw_node (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node, Thread_Scheduler_state next_state)

  *Does nothing.*
- void _Scheduler_default_Pin_or_unpin (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node, struct Per_CPU_Control *cpu)

  *Does nothing in a single processor system, otherwise a fatal error is issued.*

- void _Scheduler_default_Schedule (const Scheduler_Control *scheduler, Thread_Control *the_thread)

  *Does nothing.*
- void _Scheduler_default_Node_initialize (const Scheduler_Control *scheduler, Scheduler_Node *node, Thread_Control *the_thread, Priority_Control priority)

  *Performs the scheduler base node initialization.*
- void _Scheduler_default_Node_destroy (const Scheduler_Control *scheduler, Scheduler_Node *node)

  *Does nothing.*
- void _Scheduler_default_Release_job (const Scheduler_Control *scheduler, Thread_Control *the_thread, Priority_Node *priority_node, uint64_t deadline, Thread_queue_Context *queue_context)

  *Does nothing.*
- void _Scheduler_default_Cancel_job (const Scheduler_Control *scheduler, Thread_Control *the_thread, Priority_Node *priority_node, Thread_queue_Context *queue_context)

  *Does nothing.*
- void _Scheduler_default_Tick (const Scheduler_Control *scheduler, Thread_Control *executing)

  *Performs tick operations depending on the CPU budget algorithm for each executing thread.*
- void _Scheduler_default_Start_idle (const Scheduler_Control *scheduler, Thread_Control *the_thread, struct Per_CPU_Control *cpu)

  *Starts an idle thread.*
- bool _Scheduler_default_Set_affinity (const Scheduler_Control *scheduler, Thread_Control *thread, Scheduler_Node *node, const Processor_mask *affinity)

  *Checks if the processor set of the scheduler is the subset of the affinity set.*

### Variables

- const Scheduler_Control _Scheduler_Table [ ]

  *This table contains the configured schedulers.*
- const size_t _Scheduler_Count

  *This constant contains the count of configured schedulers.*
- const Scheduler_Assignment _Scheduler_Initial_assignments [ ]

  *The scheduler assignments.*

### 10.56.1 Detailed Description

Constants and Structures Associated with the Scheduler.

This include file contains all the constants and structures associated with the scheduler.

## 10.57 cpukit/include/rtems/confdefs/threads.h File Reference

Evaluate Thread Configuration Options.

### 10.57.1 Detailed Description

Evaluate Thread Configuration Options.

## 10.58 cpukit/include/rtems/confdefs/unlimited.h File Reference

Evaluate Unlimited Objects Configuration Options.

### 10.58.1 Detailed Description

Evaluate Unlimited Objects Configuration Options.

## 10.59 cpukit/include/rtems/confdefs/wkspace.h File Reference

Evaluate Workspace Configuration Options.

### 10.59.1 Detailed Description

Evaluate Workspace Configuration Options.

## 10.60 cpukit/include/rtems/score/wkspace.h File Reference

Information Related to the RAM Workspace.

```
#include <rtems/score/heap.h>
```

### Functions

- void _Workspace_Handler_initialization (void)

  *Initializes the workspace handler.*
- void * _Workspace_Allocate (size_t size)

  *Allocates a memory block of the specified size from the workspace.*
- void _Workspace_Free (void *block)

  *Frees memory to the workspace.*
- char * _Workspace_String_duplicate (const char *string, size_t len)

  *Duplicates string with memory from the workspace.*

### Variables

- Heap_Control _Workspace_Area

  *Executive workspace control.*

### 10.60.1 Detailed Description

Information Related to the RAM Workspace.

This include file contains information related to the RAM Workspace. This Handler provides mechanisms which can be used to define, initialize and manipulate the workspace.

# 10.61 cpukit/include/rtems/confdefs/wkspacesupport.h File Reference

Configuration Options Workspace Support Macros.

## 10.61.1 Detailed Description

Configuration Options Workspace Support Macros.

# 10.62 cpukit/include/rtems/config.h File Reference

This header file provides parts of the application configuration information API.

```
#include <stddef.h>
#include <stdint.h>
#include <rtems/rtems/config.h>
#include <rtems/score/cpu.h>
#include <rtems/score/isr.h>
#include <rtems/score/memory.h>
#include <rtems/score/object.h>
#include <rtems/score/smp.h>
#include <rtems/score/stack.h>
#include <rtems/score/threadidledata.h>
#include <rtems/score/userextdata.h>
#include <rtems/score/watchdogticks.h>
#include <rtems/score/wkspacedata.h>
```

## Macros

- #define rtems_configuration_get_do_zero_of_workspace() _Memory_Zero_before_use

  *Indicates if the RTEMS Workspace is configured to be zeroed during system initialization for this application.*
- #define rtems_configuration_get_idle_task() _Thread_Idle_body

  *Gets the IDLE task entry of this application.*
- #define rtems_configuration_get_idle_task_stack_size() _Thread_Idle_stack_size

  *Gets the IDLE task stack size in bytes of this application.*
- #define rtems_configuration_get_interrupt_stack_size() ((size_t) _ISR_Stack_size)

  *Gets the interrupt stack size in bytes of this application.*
- #define rtems_configuration_get_maximum_processors() _SMP_Processor_configured_maximum

  *Gets the maximum number of processors configured for this application.*
- #define rtems_configuration_get_microseconds_per_tick() _Watchdog_Microseconds_per_tick

  *Gets the number of microseconds per clock tick configured for this application.*
- #define rtems_configuration_get_milliseconds_per_tick() ( _Watchdog_Microseconds_per_tick / 1000 )

  *Gets the number of milliseconds per clock tick configured for this application.*
- #define rtems_configuration_get_nanoseconds_per_tick() _Watchdog_Nanoseconds_per_tick

  *Gets the number of microseconds per clock tick configured for this application.*
- #define rtems_configuration_get_number_of_initial_extensions() ((uint32_t) _User_extensions_Initial_count)

  *Gets the number of initial extensions configured for this application.*
- #define rtems_configuration_get_stack_allocate_hook() _Stack_Allocator_allocate

  *Gets the thread stack allocator allocate hook configured for this application.*

- #define rtems_configuration_get_stack_allocate_init_hook() _Stack_Allocator_initialize

  *Gets the thread stack allocator initialization hook configured for this application.*
- #define rtems_configuration_get_stack_allocator_avoids_work_space() _Stack_Allocator_avoids_workspace

  *Indicates if the thread stack allocator is configured to avoid the RTEMS Workspace for this application.*
- #define rtems_configuration_get_stack_free_hook() _Stack_Allocator_free

  *Gets the thread stack allocator free hook configured for this application.*
- #define rtems_configuration_get_ticks_per_timeslice() _Watchdog_Ticks_per_timeslice

  *Gets the clock ticks per timeslice configured for this application.*
- #define rtems_configuration_get_unified_work_area() _Workspace_Is_unified

  *Indicates if the RTEMS Workspace and C Program Heap are configured to be unified for this application.*
- #define rtems_configuration_get_user_extension_table() _User_extensions_Initial_extensions

  *Gets the initial extensions table configured for this application.*
- #define rtems_configuration_get_user_multiprocessing_table() NULL

  *Gets the MPCI configuration table configured for this application.*
- #define rtems_configuration_get_work_space_size()

  *Gets the RTEMS Workspace size in bytes configured for this application.*
- #define RTEMS_HAS_HARDWARE_FP CPU_HARDWARE_FP

  *This constant evaluates to TRUE, if this processor variant has hardware floating point support, otherwise to FALSE.*
- #define rtems_resource_is_unlimited(_resource) _Objects_Is_unlimited(_resource)

  *Indicates if the resource is unlimited.*
- #define rtems_resource_maximum_per_allocation(_resource) _Objects_Maximum_per_allocation(_↩ resource)

  *Gets the maximum number per allocation of a resource number.*
- #define RTEMS_UNLIMITED_OBJECTS OBJECTS_UNLIMITED_OBJECTS

  *This flag is used in augment a resource number so that it indicates an unlimited resource.*
- #define rtems_resource_unlimited(_resource) ( (_resource) | RTEMS_UNLIMITED_OBJECTS )

  *Augments the resource number so that it indicates an unlimited resource.*

## Typedefs

- typedef Stack_Allocator_allocate rtems_stack_allocate_hook

  *A thread stack allocator allocate handler shall have this type.*
- typedef Stack_Allocator_initialize rtems_stack_allocate_init_hook

  *A thread stack allocator initialization handler shall have this type.*
- typedef Stack_Allocator_free rtems_stack_free_hook

  *A thread stack allocator free handler shall have this type.*

## Functions

- const char * rtems_get_copyright_notice (void)

  *Gets the RTEMS copyright notice.*
- uint32_t rtems_configuration_get_maximum_extensions (void)

  *Gets the maximum number of Classic API User Extensions configured for this application.*
- uintptr_t rtems_configuration_get_stack_space_size (void)

  *Gets the thread stack space size in bytes of configured for this application.*
- const char * rtems_get_version_string (void)

  *Gets the RTEMS version string.*

### 10.62.1 Detailed Description

This header file provides parts of the application configuration information API.

## 10.63 cpukit/include/rtems/rtems/config.h File Reference

This header file provides parts of the application configuration information API.

```
#include <stdbool.h>
#include <stdint.h>
#include <rtems/rtems/tasks.h>
```

### Classes

- struct rtems_api_configuration_table

  *This structure contains a summary of the Classic API configuration.*

### Functions

- const rtems_api_configuration_table * rtems_configuration_get_rtems_api_configuration (void)

  *Gets the Classic API Configuration Table of this application.*
- uint32_t rtems_configuration_get_maximum_barriers (void)

  *Gets the maximum number of Classic API Barriers configured for this application.*
- uint32_t rtems_configuration_get_maximum_message_queues (void)

  *Gets the maximum number of Classic API Message Queues configured for this application.*
- uint32_t rtems_configuration_get_maximum_partitions (void)

  *Gets the maximum number of Classic API Partitions configured for this application.*
- uint32_t rtems_configuration_get_maximum_periods (void)

  *Gets the maximum number of Classic API Rate Monotonic Periods configured for this application.*
- uint32_t rtems_configuration_get_maximum_ports (void)

  *Gets the maximum number of Classic API Dual-Ported Memories configured for this application.*
- uint32_t rtems_configuration_get_maximum_regions (void)

  *Gets the maximum number of Classic API Regions configured for this application.*
- uint32_t rtems_configuration_get_maximum_semaphores (void)

  *Gets the maximum number of Classic API Semaphores configured for this application.*
- uint32_t rtems_configuration_get_maximum_tasks (void)

  *Gets the maximum number of Classic API Tasks configured for this application.*
- uint32_t rtems_configuration_get_maximum_timers (void)

  *Gets the maximum number of Classic API Timers configured for this application.*

### 10.63.1 Detailed Description

This header file provides parts of the application configuration information API.

## 10.64 cpukit/include/rtems/counter.h File Reference

Free-Running Counter and Busy Wait Delay API.

```
#include <rtems/score/cpu.h>
```

### Typedefs

- typedef CPU_Counter_ticks rtems_counter_ticks

  *Unsigned integer type for counter values.*

### Functions

- static uint32_t rtems_counter_frequency (void)

  *Returns the current counter frequency in Hz.*
- static rtems_counter_ticks rtems_counter_read (void)

  *Reads the current counter value.*
- static rtems_counter_ticks rtems_counter_difference (rtems_counter_ticks second, rtems_counter_ticks first)

  *Returns the difference between the second and first CPU counter value.*
- uint64_t rtems_counter_ticks_to_nanoseconds (rtems_counter_ticks ticks)

  *Converts counter ticks into nanoseconds.*
- rtems_counter_ticks rtems_counter_nanoseconds_to_ticks (uint32_t nanoseconds)

  *Converts nanoseconds into counter ticks.*
- int64_t rtems_counter_ticks_to_sbintime (rtems_counter_ticks ticks)

  *Converts counter ticks into signed binary time (sbintime_t).*
- rtems_counter_ticks rtems_counter_sbintime_to_ticks (int64_t sbt)

  *Converts signed binary time (sbintime_t) into counter ticks.*
- void rtems_counter_initialize_converter (uint32_t frequency)

  *Initializes the counter ticks to/from nanoseconds converter functions.*
- void rtems_counter_delay_ticks (rtems_counter_ticks ticks)

  *Busy wait for some counter ticks.*
- void rtems_counter_delay_nanoseconds (uint32_t nanoseconds)

  *Busy wait for some nanoseconds.*

### 10.64.1 Detailed Description

Free-Running Counter and Busy Wait Delay API.

## 10.65 cpukit/include/rtems/extension.h File Reference

This header file defines the User Extensions Manager API.

```
#include <rtems/rtems/status.h>
#include <rtems/rtems/types.h>
#include <rtems/score/interr.h>
#include <rtems/score/userextdata.h>
```

### Typedefs

- typedef User_extensions_fatal_extension rtems_fatal_extension
  %
- typedef Internal_errors_t rtems_fatal_code
  %
- typedef Internal_errors_Source rtems_fatal_source
  %
- typedef User_extensions_Table rtems_extensions_table
  %
- typedef User_extensions_thread_begin_extension rtems_task_begin_extension
  %
- typedef User_extensions_thread_create_extension rtems_task_create_extension
  %
- typedef User_extensions_thread_delete_extension rtems_task_delete_extension
  %
- typedef User_extensions_thread_exitted_extension rtems_task_exitted_extension
  %
- typedef User_extensions_thread_restart_extension rtems_task_restart_extension
  %
- typedef User_extensions_thread_start_extension rtems_task_start_extension
  %
- typedef User_extensions_thread_switch_extension rtems_task_switch_extension
  %
- typedef User_extensions_thread_terminate_extension rtems_task_terminate_extension
  %

### Functions

- rtems_status_code rtems_extension_delete (rtems_id id)
  %
- rtems_status_code rtems_extension_ident (rtems_name name, rtems_id *id)
  *Identifies an extension set object by the specified object name.*
- rtems_status_code rtems_extension_create (rtems_name name, const rtems_extensions_table *extension↩
  _table, rtems_id *id)
  %

#### 10.65.1 Detailed Description

This header file defines the User Extensions Manager API.

## 10.66 cpukit/include/rtems/extensiondata.h File Reference

Classic User Extensions Data Structures.

```
#include <rtems/extension.h>
#include <rtems/score/objectdata.h>
#include <rtems/score/userextdata.h>
```

### Classes

- struct Extension_Control

### Macros

- #define EXTENSION_INFORMATION_DEFINE(max)

  *Macro to define the objects information for the Classic Extensions objects.*

### Variables

- Objects_Information _Extension_Information

  *The Classic Extensions objects information.*

#### 10.66.1 Detailed Description

Classic User Extensions Data Structures.

## 10.67 cpukit/include/rtems/extensionimpl.h File Reference

Classic User Extensions Implementation.

```
#include <rtems/extensiondata.h>
#include <rtems/score/objectimpl.h>
```

### Functions

- static __inline__ Extension_Control * **_Extension_Allocate** (void)
- static __inline__ void **_Extension_Free** (Extension_Control *the_extension)
- static __inline__ Extension_Control * **_Extension_Get** (Objects_Id id)

#### 10.67.1 Detailed Description

Classic User Extensions Implementation.

## 10.68 cpukit/include/rtems/fs.h File Reference

Basic Filesystem Types.

```
#include <rtems/chain.h>
```

### Classes

- struct rtems_filesystem_location_info_tt

  *File system location.*
- struct rtems_filesystem_global_location_t

  *Global file system location.*

### Macros

- #define **rtems_filesystem_location_mount**(_pl) ((_pl)->mt_entry)

### Typedefs

- typedef struct rtems_libio_tt **rtems_libio_t**
- typedef struct rtems_filesystem_mount_table_entry_tt **rtems_filesystem_mount_table_entry_t**
- typedef struct _rtems_filesystem_file_handlers_r **rtems_filesystem_file_handlers_r**
- typedef struct _rtems_filesystem_operations_table **rtems_filesystem_operations_table**
- typedef struct rtems_filesystem_location_info_tt rtems_filesystem_location_info_t

  *File system location.*
- typedef struct rtems_filesystem_global_location_t rtems_filesystem_global_location_t

  *Global file system location.*

### 10.68.1 Detailed Description

Basic Filesystem Types.

This file defines basic filesystem types

## 10.69 cpukit/include/rtems/init.h File Reference

This header file defines the Initialization Manager API.

```
#include <stdint.h>
#include <rtems/score/basedefs.h>
```

### Functions

- RTEMS_NO_RETURN void rtems_initialize_executive (void)

  *Initializes the system and starts multitasking.*
- RTEMS_NO_RETURN void rtems_shutdown_executive (uint32_t result)

  *Shuts down the RTEMS environment.*

### 10.69.1 Detailed Description

This header file defines the Initialization Manager API.

## 10.70 cpukit/include/rtems/io.h File Reference

This header file defines the IO Manager API.

```
#include <stdint.h>
#include <rtems/rtems/status.h>
```

### Classes

- struct rtems_driver_address_table

  *This structure contains the device driver entries.*

### Typedefs

- typedef rtems_status_code rtems_device_driver

  *This type shall be used in device driver entry declarations and definitions.*
- typedef uint32_t rtems_device_major_number

  *This integer type represents the major number of devices.*
- typedef uint32_t rtems_device_minor_number

  *This integer type represents the minor number of devices.*
- typedef rtems_device_driver(∗ rtems_device_driver_entry) (rtems_device_major_number, rtems_device_minor_number, void ∗)

  *Device driver entries shall have this type.*

### Functions

- rtems_status_code rtems_io_register_driver (rtems_device_major_number major, const rtems_driver_address_table ∗driver_table, rtems_device_major_number ∗registered_major)

  *Registers and initializes the device with the specified device driver address table and device major number in the Device Driver Table.*
- rtems_status_code rtems_io_unregister_driver (rtems_device_major_number major)

  *Removes a device driver specified by the device major number from the Device Driver Table.*
- rtems_status_code rtems_io_initialize (rtems_device_major_number major, rtems_device_minor_number minor, void ∗argument)

  *Initializes the device specified by the device major and minor numbers.*
- rtems_status_code rtems_io_register_name (const char ∗device_name, rtems_device_major_number major, rtems_device_minor_number minor)

  *Registers the device specified by the device major and minor numbers in the file system under the specified name.*
- rtems_status_code rtems_io_open (rtems_device_major_number major, rtems_device_minor_number minor, void ∗argument)

  *Opens the device specified by the device major and minor numbers.*
- rtems_status_code rtems_io_close (rtems_device_major_number major, rtems_device_minor_number minor, void ∗argument)

  *Closes the device specified by the device major and minor numbers.*
- rtems_status_code rtems_io_read (rtems_device_major_number major, rtems_device_minor_number minor, void ∗argument)

  *Reads from the device specified by the device major and minor numbers.*
- rtems_status_code rtems_io_write (rtems_device_major_number major, rtems_device_minor_number minor, void ∗argument)

  *Writes to the device specified by the device major and minor numbers.*
- rtems_status_code rtems_io_control (rtems_device_major_number major, rtems_device_minor_number minor, void ∗argument)

  *Controls the device specified by the device major and minor numbers.*

### 10.70.1 Detailed Description

This header file defines the IO Manager API.

## 10.71 cpukit/include/rtems/irq-extension.h File Reference

Header file for the Interrupt Manager Extension.

```
#include <rtems.h>
#include <rtems/chain.h>
```

### Classes

- struct rtems_interrupt_server_action

  *An interrupt server action.*
- struct rtems_interrupt_server_control

  *An interrupt server control.*
- struct rtems_interrupt_server_config

  *An interrupt server configuration.*
- struct rtems_interrupt_server_entry

  *An interrupt server entry.*
- struct rtems_interrupt_server_request

  *An interrupt server request.*

### Macros

- #define RTEMS_INTERRUPT_UNIQUE ((rtems_option) 0x00000001)

  *Makes the interrupt handler unique. Prevents other handler from using the same interrupt vector.*
- #define RTEMS_INTERRUPT_SHARED ((rtems_option) 0x00000000)

  *Allows that this interrupt handler may share a common interrupt vector with other handler.*
- #define RTEMS_INTERRUPT_REPLACE ((rtems_option) 0x00000002)

  *Forces that this interrupt handler replaces the first handler with the same argument.*
- #define RTEMS_INTERRUPT_IS_UNIQUE(options) ((options) & RTEMS_INTERRUPT_UNIQUE)

  *Returns true if the interrupt handler unique option is set.*
- #define RTEMS_INTERRUPT_IS_SHARED(options) (!RTEMS_INTERRUPT_IS_UNIQUE( options))

  *Returns true if the interrupt handler shared option is set.*
- #define RTEMS_INTERRUPT_IS_REPLACE(options) ((options) & RTEMS_INTERRUPT_REPLACE)

  *Returns true if the interrupt handler replace option is set.*
- #define RTEMS_INTERRUPT_SERVER_DEFAULT 0

  *The interrupt server index of the default interrupt server.*

## Typedefs

- typedef void(∗ rtems_interrupt_handler) (void ∗)

  *Interrupt handler routine type.*
- typedef void(∗ rtems_interrupt_per_handler_routine) (void ∗, const char ∗, rtems_option, rtems_interrupt_handler, void ∗)

  *Interrupt handler iteration routine type.*
- typedef struct rtems_interrupt_server_action rtems_interrupt_server_action

  *An interrupt server action.*
- typedef struct rtems_interrupt_server_control rtems_interrupt_server_control

  *An interrupt server control.*

## Functions

- rtems_status_code rtems_interrupt_handler_install (rtems_vector_number vector, const char ∗info, rtems_option options, rtems_interrupt_handler handler, void ∗arg)

  *Installs the interrupt handler routine handler for the interrupt vector with number vector.*
- rtems_status_code rtems_interrupt_handler_remove (rtems_vector_number vector, rtems_interrupt_handler handler, void ∗arg)

  *Removes the interrupt handler routine handler with argument arg for the interrupt vector with number vector.*
- rtems_status_code rtems_interrupt_handler_iterate (rtems_vector_number vector, rtems_interrupt_per_handler_routine routine, void ∗arg)

  *Iterates over all installed interrupt handler of the interrupt vector with number vector.*
- rtems_status_code rtems_interrupt_set_affinity (rtems_vector_number vector, size_t affinity_size, const cpu_set_t ∗affinity)

  *Sets the processor affinity set of an interrupt vector.*
- rtems_status_code rtems_interrupt_get_affinity (rtems_vector_number vector, size_t affinity_size, cpu_set↩ _t ∗affinity)

  *Gets the processor affinity set of an interrupt vector.*
- rtems_status_code rtems_interrupt_server_initialize (rtems_task_priority priority, size_t stack_size, rtems_mode modes, rtems_attribute attributes, uint32_t ∗server_count)

  *Initializes the interrupt server tasks.*
- rtems_status_code rtems_interrupt_server_create (rtems_interrupt_server_control ∗control, const rtems_interrupt_server_config ∗config, uint32_t ∗server_index)

  *Creates an interrupt server.*
- rtems_status_code rtems_interrupt_server_delete (uint32_t server_index)

  *Destroys the interrupt server.*
- rtems_status_code rtems_interrupt_server_handler_install (uint32_t server_index, rtems_vector_number vector, const char ∗info, rtems_option options, rtems_interrupt_handler handler, void ∗arg)

  *Installs the interrupt handler routine handler for the interrupt vector with number vector on the server server.*
- rtems_status_code rtems_interrupt_server_handler_remove (uint32_t server_index, rtems_vector_number vector, rtems_interrupt_handler handler, void ∗arg)

  *Removes the interrupt handler routine handler with argument arg for the interrupt vector with number vector from the server server.*
- rtems_status_code rtems_interrupt_server_handler_iterate (uint32_t server_index, rtems_vector_number vector, rtems_interrupt_per_handler_routine routine, void ∗arg)

  *Iterates over all interrupt handler of the interrupt vector with number vector which are installed on the interrupt server specified by server.*
- rtems_status_code rtems_interrupt_server_move (uint32_t source_server_index, rtems_vector_number vector, uint32_t destination_server_index)

  *Moves the interrupt handlers installed on the specified source interrupt server to the destination interrupt server.*
- rtems_status_code rtems_interrupt_server_suspend (uint32_t server_index)

*Suspends the specified interrupt server.*

- rtems_status_code rtems_interrupt_server_resume (uint32_t server_index)

  *Resumes the specified interrupt server.*

- rtems_status_code rtems_interrupt_server_set_affinity (uint32_t server_index, size_t affinity_size, const cpu_set_t *affinity, rtems_task_priority priority)

  *Sets the processor affinity of the specified interrupt server.*

- rtems_status_code rtems_interrupt_server_entry_initialize (uint32_t server_index, rtems_interrupt_server_entry *entry)

  *Initializes the specified interrupt server entry.*

- void rtems_interrupt_server_action_prepend (rtems_interrupt_server_entry *entry, rtems_interrupt_server_action *action, rtems_interrupt_handler handler, void *arg)

  *Prepends the specified interrupt server action to the list of actions of the specified interrupt server entry.*

- void rtems_interrupt_server_entry_submit (rtems_interrupt_server_entry *entry)

  *Submits the specified interrupt server entry so that its interrupt server actions can be invoked by the specified interrupt server.*

- rtems_status_code rtems_interrupt_server_entry_move (rtems_interrupt_server_entry *entry, uint32_t destination_server_index)

  *Moves the interrupt server entry to the specified destination interrupt server.*

- void rtems_interrupt_server_entry_destroy (rtems_interrupt_server_entry *entry)

  *Destroys the specified interrupt server entry.*

- rtems_status_code rtems_interrupt_server_request_initialize (uint32_t server_index, rtems_interrupt_server_request *request, rtems_interrupt_handler handler, void *arg)

  *Initializes the specified interrupt server request.*

- static __inline__ void rtems_interrupt_server_request_set_vector (rtems_interrupt_server_request *request, rtems_vector_number vector)

  *Sets the interrupt vector in the specified interrupt server request.*

- static __inline__ void rtems_interrupt_server_request_submit (rtems_interrupt_server_request *request)

  *Submits the specified interrupt server request so that its interrupt server action can be invoked by the specified interrupt server.*

- static __inline__ void rtems_interrupt_server_request_destroy (rtems_interrupt_server_request *request)

  *Destroys the specified interrupt server request.*

### 10.71.1 Detailed Description

Header file for the Interrupt Manager Extension.

## 10.72 cpukit/include/rtems/libcsupport.h File Reference

Standard C Library Support.

```
#include <sys/types.h>
#include <stdint.h>
#include <rtems/score/heap.h>
#include <rtems/rtems/tasks.h>
```

### Classes

- struct rtems_resource_rtems_api
- struct rtems_resource_posix_api
- struct rtems_resource_snapshot

### Macros

- #define **RTEMS_NEWLIB_EXTENSION**

### Functions

- void **malloc_dump** (void)
- bool malloc_walk (int source, bool printf_enabled)

  *Malloc walk.*
- void malloc_set_heap_pointer (Heap_Control *new_heap)

  *Set malloc heap pointer.*
- Heap_Control * malloc_get_heap_pointer (void)

  *Get malloc heap pointer.*
- size_t malloc_free_space (void)

  *Get free malloc information.*
- int malloc_info (Heap_Information_block *the_info)

  *Get malloc status information.*
- bool **newlib_create_hook** (rtems_tcb *current_task, rtems_tcb *creating_task)
- void **newlib_terminate_hook** (rtems_tcb *current_task)
- void rtems_resource_snapshot_take (rtems_resource_snapshot *snapshot)

  *Tasks a snapshot of the resource usage of the system.*
- bool rtems_resource_snapshot_equal (const rtems_resource_snapshot *a, const rtems_resource_snapshot *b)

  *Compares two resource snapshots for equality.*
- bool rtems_resource_snapshot_check (const rtems_resource_snapshot *snapshot)

  *Takes a new resource snapshot and checks that it is equal to the given resource snapshot.*

#### 10.72.1 Detailed Description

Standard C Library Support.

This include file contains the information regarding the RTEMS specific support for the standard C library.

## 10.73 cpukit/include/rtems/mallocinitone.h File Reference

Malloc Support Initialization API.

```
#include <rtems/malloc.h>
#include <rtems/score/assert.h>
#include <rtems/score/heapimpl.h>
```

### Functions

- static __inline__ Heap_Control * **_Malloc_Initialize_with_one_area** (Heap_Control *heap)

### 10.73.1 Detailed Description

Malloc Support Initialization API.

## 10.74 cpukit/include/rtems/posix/spinlockimpl.h File Reference

Inlined Routines from the POSIX Spinlock Manager.

```
#include <pthread.h>
#include <rtems/score/isrlevel.h>
#include <rtems/score/percpu.h>
#include <rtems/score/smplockticket.h>
```

### Classes

- struct POSIX_Spinlock_Control

### Functions

- static __inline__ POSIX_Spinlock_Control * **_POSIX_Spinlock_Get** (pthread_spinlock_t *lock)

### 10.74.1 Detailed Description

Inlined Routines from the POSIX Spinlock Manager.

This file contains the static inlin implementation of the inlined routines from the POSIX Spinlock Manager.

## 10.75 cpukit/include/rtems/print.h File Reference

User print interface to the bspIO print plug in.

```
#include <rtems/score/basedefs.h>
#include <stdarg.h>
```

### Typedefs

- typedef struct rtems_printer **rtems_printer**

### Functions

- int rtems_printf (const rtems_printer *printer, const char *format,...) RTEMS_PRINTFLIKE(2

  *Print to the kernel plugin handler. This has to be a macro because there is no vprint version of the plug in handlers.*
- int int rtems_vprintf (const rtems_printer *printer, const char *format, va_list ap)

  *Print to the kernel plugin handler. This has to be a macro because there is no vprint version of the plug in handlers.*

### 10.75.1 Detailed Description

User print interface to the bspIO print plug in.

This include file defines the user interface to kernel print methods.

### 10.75.2 Function Documentation

#### 10.75.2.1 rtems_printf()

```
int rtems_printf (
            const rtems_printer * printer,
            const char * format,
             ...  )
```

Print to the kernel plugin handler. This has to be a macro because there is no vprint version of the plug in handlers.

**Parameters**

| | | |
|---|---|---|
| in | *printer* | Pointer to the printer structure. |
| in | *fmt* | Print format string. |
| in | ... | Print variable argument list. |

**Returns**

int Number of characters printed.

#### 10.75.2.2 rtems_vprintf()

```
int int rtems_vprintf (
            const rtems_printer * printer,
            const char * format,
            va_list ap )
```

Print to the kernel plugin handler. This has to be a macro because there is no vprint version of the plug in handlers.

**Parameters**

| | | |
|---|---|---|
| in | *printer* | Pointer to the printer structure. |
| in | *fmt* | Print format string. |
| in | *ap* | Print variable argument list pointer. |

**Returns**

int Number of characters printed.

Definition at line 23 of file print_printf.c.

## 10.76 cpukit/include/rtems/printer.h File Reference

User print interface to the bspIO print plug in.

```
#include <rtems/print.h>
#include <rtems/chain.h>
#include <rtems/rtems/intr.h>
#include <rtems/rtems/tasks.h>
#include <stdio.h>
#include <string.h>
```

### Classes

- struct rtems_printer
- struct rtems_printer_task_context

### Typedefs

- typedef int(∗ rtems_print_printer) (void ∗, const char ∗format, va_list ap)

### Functions

- static bool rtems_print_printer_valid (const rtems_printer ∗printer)

  *check if the printer is valid.*
- static void rtems_print_printer_empty (rtems_printer ∗printer)

  *Initializes the printer to print nothing.*
- void rtems_print_printer_printk (rtems_printer ∗printer)

  *Initializes the printer to print via printk().*
- void rtems_print_printer_printf (rtems_printer ∗printer)

  *Initializes the printer to print via printf().*
- void rtems_print_printer_fprintf (rtems_printer ∗printer, FILE ∗file)

  *Initializes the printer to print via fprintf() using the specified file stream.*
- void rtems_print_printer_fprintf_putc (rtems_printer ∗printer)

  *Initializes the printer to print via fprintf() using an unbuffered FILE stream with output through rtems_putc().*
- static void **rtems_printer_task_initialize** (rtems_printer_task_context ∗context)
- static void **rtems_printer_task_set_stack_size** (rtems_printer_task_context ∗context, size_t stack_size)
- static void **rtems_printer_task_set_priority** (rtems_printer_task_context ∗context, rtems_task_priority priority)
- static void **rtems_printer_task_set_file_descriptor** (rtems_printer_task_context ∗context, int fd)
- static void **rtems_printer_task_set_buffer_table** (rtems_printer_task_context ∗context, void ∗buffer_table)
- static void **rtems_printer_task_set_buffer_count** (rtems_printer_task_context ∗context, size_t buffer_↩ count)
- static void **rtems_printer_task_set_buffer_size** (rtems_printer_task_context ∗context, size_t buffer_size)
- int rtems_print_printer_task (rtems_printer ∗printer, rtems_printer_task_context ∗context)

  *Creates a printer task.*
- void rtems_printer_task_drain (rtems_printer_task_context ∗context)

  *Drains the work queue of the printer task.*

### 10.76.1 Detailed Description

User print interface to the bspIO print plug in.

This include file defines the user interface to kernel print methods.

## 10.77 cpukit/include/rtems/rtems/asr.h File Reference

This header file defines the parts of the Signal Manager API.

```
#include <stdint.h>
```

### Macros

- #define RTEMS_SIGNAL_0 0x00000001

  *This constant defines the bit in the signal set associated with signal 0.*
- #define RTEMS_SIGNAL_1 0x00000002

  *This constant defines the bit in the signal set associated with signal 1.*
- #define RTEMS_SIGNAL_10 0x00000400

  *This constant defines the bit in the signal set associated with signal 10.*
- #define RTEMS_SIGNAL_11 0x00000800

  *This constant defines the bit in the signal set associated with signal 11.*
- #define RTEMS_SIGNAL_12 0x00001000

  *This constant defines the bit in the signal set associated with signal 12.*
- #define RTEMS_SIGNAL_13 0x00002000

  *This constant defines the bit in the signal set associated with signal 13.*
- #define RTEMS_SIGNAL_14 0x00004000

  *This constant defines the bit in the signal set associated with signal 14.*
- #define RTEMS_SIGNAL_15 0x00008000

  *This constant defines the bit in the signal set associated with signal 15.*
- #define RTEMS_SIGNAL_16 0x00010000

  *This constant defines the bit in the signal set associated with signal 16.*
- #define RTEMS_SIGNAL_17 0x00020000

  *This constant defines the bit in the signal set associated with signal 17.*
- #define RTEMS_SIGNAL_18 0x00040000

  *This constant defines the bit in the signal set associated with signal 18.*
- #define RTEMS_SIGNAL_19 0x00080000

  *This constant defines the bit in the signal set associated with signal 19.*
- #define RTEMS_SIGNAL_2 0x00000004

  *This constant defines the bit in the signal set associated with signal 2.*
- #define RTEMS_SIGNAL_20 0x00100000

  *This constant defines the bit in the signal set associated with signal 20.*
- #define RTEMS_SIGNAL_21 0x00200000

  *This constant defines the bit in the signal set associated with signal 21.*
- #define RTEMS_SIGNAL_22 0x00400000

  *This constant defines the bit in the signal set associated with signal 22.*
- #define RTEMS_SIGNAL_23 0x00800000

  *This constant defines the bit in the signal set associated with signal 23.*

- #define RTEMS_SIGNAL_24 0x01000000

  *This constant defines the bit in the signal set associated with signal 24.*
- #define RTEMS_SIGNAL_25 0x02000000

  *This constant defines the bit in the signal set associated with signal 25.*
- #define RTEMS_SIGNAL_26 0x04000000

  *This constant defines the bit in the signal set associated with signal 26.*
- #define RTEMS_SIGNAL_27 0x08000000

  *This constant defines the bit in the signal set associated with signal 27.*
- #define RTEMS_SIGNAL_28 0x10000000

  *This constant defines the bit in the signal set associated with signal 28.*
- #define RTEMS_SIGNAL_29 0x20000000

  *This constant defines the bit in the signal set associated with signal 29.*
- #define RTEMS_SIGNAL_3 0x00000008

  *This constant defines the bit in the signal set associated with signal 3.*
- #define RTEMS_SIGNAL_30 0x40000000

  *This constant defines the bit in the signal set associated with signal 30.*
- #define RTEMS_SIGNAL_31 0x80000000

  *This constant defines the bit in the signal set associated with signal 31.*
- #define RTEMS_SIGNAL_4 0x00000010

  *This constant defines the bit in the signal set associated with signal 4.*
- #define RTEMS_SIGNAL_5 0x00000020

  *This constant defines the bit in the signal set associated with signal 5.*
- #define RTEMS_SIGNAL_6 0x00000040

  *This constant defines the bit in the signal set associated with signal 6.*
- #define RTEMS_SIGNAL_7 0x00000080

  *This constant defines the bit in the signal set associated with signal 7.*
- #define RTEMS_SIGNAL_8 0x00000100

  *This constant defines the bit in the signal set associated with signal 8.*
- #define RTEMS_SIGNAL_9 0x00000200

  *This constant defines the bit in the signal set associated with signal 9.*

### Typedefs

- typedef void rtems_asr

  *%*
- typedef uint32_t rtems_signal_set

  *%*
- typedef rtems_asr(∗ rtems_asr_entry) (rtems_signal_set)

  *%*

### 10.77.1 Detailed Description

This header file defines the parts of the Signal Manager API.

## 10.78 cpukit/include/rtems/rtems/asrdata.h File Reference

Classic ASR Data Structures.

```
#include <rtems/rtems/asr.h>
```

### Classes

- struct ASR_Information

### 10.78.1 Detailed Description

Classic ASR Data Structures.

## 10.79 cpukit/include/rtems/rtems/asrimpl.h File Reference

Classic ASR Implementation.

```
#include <rtems/rtems/asrdata.h>
#include <string.h>
```

### Functions

- RTEMS_INLINE_ROUTINE void _ASR_Initialize (ASR_Information *asr)

  *ASR_Initialize.*
- RTEMS_INLINE_ROUTINE rtems_signal_set **_ASR_Swap_signals** (ASR_Information *asr)
- RTEMS_INLINE_ROUTINE void **_ASR_Post_signals** (rtems_signal_set signals, rtems_signal_set *signal_set)
- RTEMS_INLINE_ROUTINE rtems_signal_set **_ASR_Get_posted_signals** (ASR_Information *asr)

### 10.79.1 Detailed Description

Classic ASR Implementation.

## 10.80 cpukit/include/rtems/rtems/attr.h File Reference

This header file provides Classic API directive attributes.

```
#include <stdint.h>
```

## Macros

- #define RTEMS_APPLICATION_TASK 0x00000000

  *This attribute constant indicates that the Classic API task created by rtems_task_create() or rtems_task_construct() shall be an application task.*
- #define RTEMS_BARRIER_AUTOMATIC_RELEASE 0x00000200

  *This attribute constant indicates that the Classic API barrier created by rtems_barrier_create() shall use the automatic release protocol.*
- #define RTEMS_BARRIER_MANUAL_RELEASE 0x00000000

  *This attribute constant indicates that the Classic API barrier created by rtems_barrier_create() shall use the manual release protocol.*
- #define RTEMS_BINARY_SEMAPHORE 0x00000010

  *This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() shall be a proper binary semaphore or mutex.*
- #define RTEMS_COUNTING_SEMAPHORE 0x00000000

  *This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() shall be a counting semaphore.*
- #define RTEMS_DEFAULT_ATTRIBUTES 0x00000000

  *This attribute constant represents the default attribute set.*
- #define RTEMS_FIFO 0x00000000

  *This attribute constant indicates that the Classic API object shall manage blocking tasks using the FIFO discipline.*
- #define RTEMS_FLOATING_POINT 0x00000001

  *This attribute constant indicates that the Classic API task created by rtems_task_create() or rtems_task_construct() shall be able to use the floating point hardware.*
- #define RTEMS_GLOBAL 0x00000002

  *This attribute constant indicates that the Classic API object shall be a global resource in a multiprocessing network.*
- #define RTEMS_INHERIT_PRIORITY 0x00000040

  *This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() shall use the Priority Inheritance Protocol.*
- #define RTEMS_LOCAL 0x00000000

  *This attribute constant indicates that the Classic API object shall be a local resource in a multiprocessing network.*
- #define RTEMS_MULTIPROCESSOR_RESOURCE_SHARING 0x00000100

  *This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() shall use the Multiprocessor Resource Sharing Protocol.*
- #define RTEMS_NO_FLOATING_POINT 0x00000000

  *This attribute constant indicates that the Classic API task created by rtems_task_create() or rtems_task_construct() will not use the floating point hardware.*
- #define RTEMS_NO_INHERIT_PRIORITY 0x00000000

  *This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() will not use the Priority Inheritance Protocol.*
- #define RTEMS_NO_MULTIPROCESSOR_RESOURCE_SHARING 0x00000000

  *This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() will not use the Multiprocessor Resource Sharing Protocol.*
- #define RTEMS_NO_PRIORITY_CEILING 0x00000000

  *This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() will not use the Priority Ceiling Protocol.*
- #define RTEMS_PRIORITY 0x00000004

  *This attribute constant indicates that the Classic API object shall manage blocking tasks using the task priority discipline.*
- #define RTEMS_PRIORITY_CEILING 0x00000080

  *This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() shall use the Priority Ceiling Protocol.*
- #define RTEMS_SEMAPHORE_CLASS 0x00000030

*This attribute constant represents the mask of bits associated with the Classic API semaphore classes RTEMS_BINARY_SEMAPHORE, RTEMS_COUNTING_SEMAPHORE, and RTEMS_SIMPLE_BINARY_SEMAPHORE.*

- #define RTEMS_SIMPLE_BINARY_SEMAPHORE 0x00000020

  *This attribute constant indicates that the Classic API semaphore created by rtems_semaphore_create() shall be a simple binary semaphore.*

- #define RTEMS_SYSTEM_TASK 0x00008000

  *This attribute constant indicates that the Classic API task created by rtems_task_create() or rtems_task_construct() shall be a system task.*

### Typedefs

- typedef uint32_t rtems_attribute

  *This type represents Classic API attributes.*

### 10.80.1 Detailed Description

This header file provides Classic API directive attributes.

## 10.81 cpukit/include/rtems/rtems/attrimpl.h File Reference

Classic Attributes Implementation.

```
#include <rtems/rtems/attr.h>
#include <rtems/score/cpu.h>
```

### Macros

- #define ATTRIBUTES_NOT_SUPPORTED 0
- #define ATTRIBUTES_REQUIRED RTEMS_FLOATING_POINT

### Functions

- RTEMS_INLINE_ROUTINE rtems_attribute _Attributes_Set (rtems_attribute new_attributes, rtems_attribute attribute_set)

  *Sets the requested new_attributes in the attribute_set passed in.*

- RTEMS_INLINE_ROUTINE rtems_attribute _Attributes_Clear (rtems_attribute attribute_set, rtems_attribute mask)

  *Clears the requested new_attributes in the attribute_set passed in.*

- RTEMS_INLINE_ROUTINE bool _Attributes_Is_floating_point (rtems_attribute attribute_set)

  *Checks if the floating point attribute is enabled in the attribute_set.*

- RTEMS_INLINE_ROUTINE bool _Attributes_Is_priority (rtems_attribute attribute_set)

  *Checks if the priority attribute is enabled in the attribute_set.*

- RTEMS_INLINE_ROUTINE bool _Attributes_Is_binary_semaphore (rtems_attribute attribute_set)

  *Checks if the binary semaphore attribute is enabled in the attribute_set.*

- RTEMS_INLINE_ROUTINE bool _Attributes_Is_simple_binary_semaphore (rtems_attribute attribute_set)

  *Checks if the simple binary semaphore attribute is enabled in the attribute_set.*

- RTEMS_INLINE_ROUTINE bool _Attributes_Is_counting_semaphore (rtems_attribute attribute_set)

  *Checks if the counting semaphore attribute is enabled in the attribute_set.*
- RTEMS_INLINE_ROUTINE bool _Attributes_Is_inherit_priority (rtems_attribute attribute_set)

  *Checks if the priority inheritance attribute is enabled in the attribute_set.*
- RTEMS_INLINE_ROUTINE bool _Attributes_Has_at_most_one_protocol (rtems_attribute attribute_set)

  *Returns true if the attribute set has at most one protocol, and false otherwise.*
- RTEMS_INLINE_ROUTINE bool _Attributes_Is_priority_ceiling (rtems_attribute attribute_set)

  *Checks if the priority ceiling attribute is enabled in the attribute_set.*
- RTEMS_INLINE_ROUTINE bool _Attributes_Is_multiprocessor_resource_sharing (rtems_attribute attribute↩ _set)

  *Checks if the Multiprocessor Resource Sharing Protocol attribute is enabled in the attribute_set.*
- RTEMS_INLINE_ROUTINE bool _Attributes_Is_barrier_automatic (rtems_attribute attribute_set)

  *Checks if the barrier automatic release attribute is enabled in the attribute_set.*
- RTEMS_INLINE_ROUTINE bool _Attributes_Is_system_task (rtems_attribute attribute_set)

  *Checks if the system task attribute is enabled in the attribute_set.*

### 10.81.1 Detailed Description

Classic Attributes Implementation.

## 10.82 cpukit/include/rtems/rtems/barrier.h File Reference

This header file defines the Barrier Manager API.

```
#include <stdint.h>
#include <rtems/rtems/attr.h>
#include <rtems/rtems/status.h>
#include <rtems/rtems/types.h>
```

### Functions

- rtems_status_code rtems_barrier_create (rtems_name name, rtems_attribute attribute_set, uint32_↩ t maximum_waiters, rtems_id *id)

  %
- rtems_status_code rtems_barrier_delete (rtems_id id)

  %
- rtems_status_code rtems_barrier_ident (rtems_name name, rtems_id *id)

  *Identifies a barrier object by the specified object name.*
- rtems_status_code rtems_barrier_release (rtems_id id, uint32_t *released)

  %
- rtems_status_code rtems_barrier_wait (rtems_id id, rtems_interval timeout)

  %

### 10.82.1 Detailed Description

This header file defines the Barrier Manager API.

## 10.83 cpukit/include/rtems/rtems/barrierdata.h File Reference

Classic Barrier Manager Data Structures.

```
#include <rtems/rtems/barrier.h>
#include <rtems/score/objectdata.h>
#include <rtems/score/corebarrier.h>
```

### Classes

- struct Barrier_Control

### Macros

- #define BARRIER_INFORMATION_DEFINE(max)

  *Macro to define the objects information for the Classic Barrier objects.*

### Variables

- Objects_Information _Barrier_Information

  *The Classic Barrier objects information.*

### 10.83.1 Detailed Description

Classic Barrier Manager Data Structures.

## 10.84 cpukit/include/rtems/rtems/barrierimpl.h File Reference

Classic Barrier Manager Implementation.

```
#include <rtems/rtems/barrierdata.h>
#include <rtems/score/corebarrierimpl.h>
#include <rtems/score/objectimpl.h>
```

### Functions

- static __inline__ Barrier_Control * _Barrier_Allocate (void)

  *_Barrier_Allocate*
- static __inline__ void _Barrier_Free (Barrier_Control *the_barrier)

  *_Barrier_Free*
- static __inline__ Barrier_Control * **_Barrier_Get** (Objects_Id id, Thread_queue_Context *queue_context)

### 10.84.1 Detailed Description

Classic Barrier Manager Implementation.

## 10.85 cpukit/include/rtems/rtems/cache.h File Reference

This header file defines the Cache Manager API.

```
#include <stddef.h>
#include <stdint.h>
```

### Functions

- void * rtems_cache_aligned_malloc (size_t nbytes)
  %
- void rtems_cache_coherent_add_area (void *area_begin, uintptr_t area_size)
  %
- void * rtems_cache_coherent_allocate (size_t size, uintptr_t alignment, uintptr_t boundary)
  %
- void rtems_cache_coherent_free (void *ptr)
  %
- void rtems_cache_disable_data (void)
  %
- void rtems_cache_disable_instruction (void)
  %
- void rtems_cache_enable_data (void)
  %
- void rtems_cache_enable_instruction (void)
  %
- void rtems_cache_flush_entire_data (void)
  %
- void rtems_cache_flush_multiple_data_lines (const void *addr, size_t size)
  %
- void rtems_cache_freeze_data (void)
  %
- void rtems_cache_freeze_instruction (void)
  %
- size_t rtems_cache_get_data_line_size (void)
  %
- size_t rtems_cache_get_data_cache_size (uint32_t level)
  %
- size_t rtems_cache_get_instruction_line_size (void)
  %
- size_t rtems_cache_get_instruction_cache_size (uint32_t level)
  %
- size_t rtems_cache_get_maximal_line_size (void)
  %
- void rtems_cache_instruction_sync_after_code_change (const void *code_addr, size_t n_bytes)

%

- void rtems_cache_invalidate_entire_data (void)

  %
- void rtems_cache_invalidate_entire_instruction (void)

  %
- void rtems_cache_invalidate_multiple_data_lines (const void *addr, size_t size)

  %
- void rtems_cache_invalidate_multiple_instruction_lines (const void *addr, size_t size)

  %
- void rtems_cache_unfreeze_data (void)

  %
- void rtems_cache_unfreeze_instruction (void)

  %

### 10.85.1 Detailed Description

This header file defines the Cache Manager API.

## 10.86 cpukit/include/rtems/rtems/dpmem.h File Reference

This header file defines the Dual-Ported Memory Manager API.

```
#include <stdint.h>
#include <rtems/rtems/status.h>
#include <rtems/rtems/types.h>
```

### Functions

- rtems_status_code rtems_port_create (rtems_name name, void *internal_start, void *external_start, uint32_t length, rtems_id *id)

  %
- rtems_status_code rtems_port_delete (rtems_id id)

  %
- rtems_status_code rtems_port_external_to_internal (rtems_id id, void *external, void **internal)

  %
- rtems_status_code rtems_port_ident (rtems_name name, rtems_id *id)

  *Identifies a port object by the specified object name.*
- rtems_status_code rtems_port_internal_to_external (rtems_id id, void *internal, void **external)

  %

### 10.86.1 Detailed Description

This header file defines the Dual-Ported Memory Manager API.

# 10.87 cpukit/include/rtems/rtems/dpmemdata.h File Reference

Classic Dual Ported Memory Manager Data Structures.

```
#include <rtems/rtems/dpmem.h>
#include <rtems/score/objectdata.h>
```

## Classes

- struct Dual_ported_memory_Control

## Macros

- #define DUAL_PORTED_MEMORY_INFORMATION_DEFINE(max)

  *Macro to define the objects information for the Classic Dual Ported Memory objects.*

## Variables

- Objects_Information _Dual_ported_memory_Information

  *The Classic Dual Ported Memory objects information.*

### 10.87.1 Detailed Description

Classic Dual Ported Memory Manager Data Structures.

# 10.88 cpukit/include/rtems/rtems/dpmemimpl.h File Reference

Dual Ported Memory Manager Implementation.

```
#include <rtems/rtems/dpmemdata.h>
#include <rtems/score/objectimpl.h>
```

## Functions

- static __inline__ Dual_ported_memory_Control * _Dual_ported_memory_Allocate (void)

  *Allocates a port control block from the inactive chain of free port control blocks.*
- static __inline__ void _Dual_ported_memory_Free (Dual_ported_memory_Control *the_port)

  *Frees a port control block to the inactive chain of free port control blocks.*
- static __inline__ Dual_ported_memory_Control * **_Dual_ported_memory_Get** (Objects_Id id, ISR_lock_Context *lock_context)

### 10.88.1 Detailed Description

Dual Ported Memory Manager Implementation.

## 10.89 cpukit/include/rtems/rtems/event.h File Reference

This header file provides the Event Manager API.

```
#include <stdint.h>
#include <rtems/rtems/options.h>
#include <rtems/rtems/status.h>
#include <rtems/rtems/types.h>
```

### Macros

- #define RTEMS_ALL_EVENTS 0xffffffff

  *This event set constant represents all events of an event set.*
- #define RTEMS_EVENT_0 0x00000001

  *This event set constant represents the bit in the event set associated with event 0.*
- #define RTEMS_EVENT_1 0x00000002

  *This event set constant represents the bit in the event set associated with event 1.*
- #define RTEMS_EVENT_2 0x00000004

  *This event set constant represents the bit in the event set associated with event 2.*
- #define RTEMS_EVENT_3 0x00000008

  *This event set constant represents the bit in the event set associated with event 3.*
- #define RTEMS_EVENT_4 0x00000010

  *This event set constant represents the bit in the event set associated with event 4.*
- #define RTEMS_EVENT_5 0x00000020

  *This event set constant represents the bit in the event set associated with event 5.*
- #define RTEMS_EVENT_6 0x00000040

  *This event set constant represents the bit in the event set associated with event 6.*
- #define RTEMS_EVENT_7 0x00000080

  *This event set constant represents the bit in the event set associated with event 7.*
- #define RTEMS_EVENT_8 0x00000100

  *This event set constant represents the bit in the event set associated with event 8.*
- #define RTEMS_EVENT_9 0x00000200

  *This event set constant represents the bit in the event set associated with event 9.*
- #define RTEMS_EVENT_10 0x00000400

  *This event set constant represents the bit in the event set associated with event 10.*
- #define RTEMS_EVENT_11 0x00000800

  *This event set constant represents the bit in the event set associated with event 11.*
- #define RTEMS_EVENT_12 0x00001000

  *This event set constant represents the bit in the event set associated with event 12.*
- #define RTEMS_EVENT_13 0x00002000

  *This event set constant represents the bit in the event set associated with event 13.*
- #define RTEMS_EVENT_14 0x00004000

  *This event set constant represents the bit in the event set associated with event 14.*
- #define RTEMS_EVENT_15 0x00008000

  *This event set constant represents the bit in the event set associated with event 15.*
- #define RTEMS_EVENT_16 0x00010000

  *This event set constant represents the bit in the event set associated with event 16.*
- #define RTEMS_EVENT_17 0x00020000

  *This event set constant represents the bit in the event set associated with event 17.*

- #define RTEMS_EVENT_18 0x00040000

  *This event set constant represents the bit in the event set associated with event 18.*
- #define RTEMS_EVENT_19 0x00080000

  *This event set constant represents the bit in the event set associated with event 19.*
- #define RTEMS_EVENT_20 0x00100000

  *This event set constant represents the bit in the event set associated with event 20.*
- #define RTEMS_EVENT_21 0x00200000

  *This event set constant represents the bit in the event set associated with event 21.*
- #define RTEMS_EVENT_22 0x00400000

  *This event set constant represents the bit in the event set associated with event 22.*
- #define RTEMS_EVENT_23 0x00800000

  *This event set constant represents the bit in the event set associated with event 23.*
- #define RTEMS_EVENT_24 0x01000000

  *This event set constant represents the bit in the event set associated with event 24.*
- #define RTEMS_EVENT_25 0x02000000

  *This event set constant represents the bit in the event set associated with event 25.*
- #define RTEMS_EVENT_26 0x04000000

  *This event set constant represents the bit in the event set associated with event 26.*
- #define RTEMS_EVENT_27 0x08000000

  *This event set constant represents the bit in the event set associated with event 27.*
- #define RTEMS_EVENT_28 0x10000000

  *This event set constant represents the bit in the event set associated with event 28.*
- #define RTEMS_EVENT_29 0x20000000

  *This event set constant represents the bit in the event set associated with event 29.*
- #define RTEMS_EVENT_30 0x40000000

  *This event set constant represents the bit in the event set associated with event 30.*
- #define RTEMS_EVENT_31 0x80000000

  *This event set constant represents the bit in the event set associated with event 31.*
- #define RTEMS_PENDING_EVENTS 0

  *This event set constant indicates that rtems_event_receive() shall return the set of pending events.*
- #define RTEMS_EVENT_SYSTEM_NETWORK_CLOSE RTEMS_EVENT_26

  *This event set constant represents the reserved system event for a network socket close.*
- #define RTEMS_EVENT_SYSTEM_NETWORK_SBWAIT RTEMS_EVENT_24

  *This event set constant represents the reserved system event for a network socket buffer wait usage.*
- #define RTEMS_EVENT_SYSTEM_NETWORK_SOSLEEP RTEMS_EVENT_25

  *This event set constant represents the reserved system event for a network socket sleep.*
- #define RTEMS_EVENT_SYSTEM_SERVER RTEMS_EVENT_30

  *This event set constant represents the reserved system event for server thread usage, for example the timer or interrupt server.*
- #define RTEMS_EVENT_SYSTEM_SERVER_RESUME RTEMS_EVENT_29

  *This event set constant represents the reserved system event to resume a server thread, for example the timer or interrupt server.*
- #define RTEMS_EVENT_SYSTEM_TRANSIENT RTEMS_EVENT_31

  *This event set constant represents the reserved system event for transient usage.*

## Typedefs

- typedef uint32_t rtems_event_set

  *This integer type represents a bit field which can hold exactly 32 individual events.*

## Functions

- rtems_status_code rtems_event_system_receive (rtems_event_set event_in, rtems_option option_set, rtems_interval ticks, rtems_event_set *event_out)

  *Receives or gets a system event set from the executing task.*
- rtems_status_code rtems_event_system_send (rtems_id id, rtems_event_set event_in)

  *Sends the system event set to the task.*
- static void rtems_event_transient_clear (void)

  *Clears the transient event.*
- static rtems_status_code rtems_event_transient_receive (rtems_option option_set, rtems_interval ticks)

  *Receives the transient event.*
- static rtems_status_code rtems_event_transient_send (rtems_id id)

  *Sends the transient event to the task.*
- rtems_status_code rtems_event_send (rtems_id id, rtems_event_set event_in)

  *Sends the event set to the task.*
- rtems_status_code rtems_event_receive (rtems_event_set event_in, rtems_option option_set, rtems_interval ticks, rtems_event_set *event_out)

  *Receives or gets an event set from the calling task.*

### 10.89.1 Detailed Description

This header file provides the Event Manager API.

### 10.89.2 Function Documentation

#### 10.89.2.1 rtems_event_system_receive()

```
rtems_status_code rtems_event_system_receive (
            rtems_event_set event_in,
            rtems_option option_set,
            rtems_interval ticks,
            rtems_event_set * event_out )
```

Receives or gets a system event set from the executing task.

This directive performs the same actions as the rtems_event_receive() directive except that it operates with a different set of events for each task.

**Parameters**

| | |
|---|---|
| *event_in* | is the event set of interest. Use RTEMS_PENDING_EVENTS to get the pending events. |
| *option_set* | is the option set. |
| *ticks* | is the timeout in clock ticks if the RTEMS_WAIT option was set. Use RTEMS_NO_TIMEOUT to wait potentially forever. |
| *event_out* | is the pointer to an event set. The received or pending events are stored in the referenced event set if the operation was successful. |

Definition at line 33 of file systemeventreceive.c.

### 10.89.2.2 rtems_event_system_send()

```
rtems_status_code rtems_event_system_send (
            rtems_id id,
            rtems_event_set event_in )
```

Sends the system event set to the task.

**Parameters**

| | |
|---|---|
| *id* | is the identifier of the target task to receive the event set. |
| *event_in* | is the event set to send. |

Definition at line 32 of file systemeventsend.c.

### 10.89.2.3 rtems_event_transient_receive()

```
static rtems_status_code rtems_event_transient_receive (
            rtems_option option_set,
            rtems_interval ticks )  [inline], [static]
```

Receives the transient event.

**Parameters**

| | |
|---|---|
| *option_set* | is the option set. |
| *ticks* | is the optional timeout in clock ticks. |

Definition at line 551 of file event.h.

### 10.89.2.4 rtems_event_transient_send()

```
static rtems_status_code rtems_event_transient_send (
            rtems_id id )  [inline], [static]
```

Sends the transient event to the task.

**Parameters**

| | |
|---|---|
| *id* | is the identifier of the task to receive the transient event. |

Definition at line 573 of file event.h.

## 10.90 cpukit/include/rtems/rtems/eventdata.h File Reference

This header file defines the API used by the Application Configuration to define the configured Thread Control Block (TCB).

```
#include <rtems/rtems/event.h>
```

### Classes

- struct Event_Control

  *This structure is used to manage a set of events.*

### 10.90.1 Detailed Description

This header file defines the API used by the Application Configuration to define the configured Thread Control Block (TCB).

## 10.91 cpukit/include/rtems/rtems/eventimpl.h File Reference

This header file defines interfaces used by the event implementation.

```
#include <rtems/rtems/eventdata.h>
#include <rtems/score/thread.h>
```

### Functions

- rtems_status_code _Event_Seize (rtems_event_set event_in, rtems_option option_set, rtems_interval ticks, rtems_event_set *event_out, Thread_Control *executing, Event_Control *event, Thread_Wait_flags wait_↩ class, States_Control block_state, ISR_lock_Context *lock_context)

  *Seizes a set of events.*
- rtems_status_code _Event_Surrender (Thread_Control *the_thread, rtems_event_set event_in, Event_Control *event, Thread_Wait_flags wait_class, ISR_lock_Context *lock_context)

  *Surrenders a set of events.*
- static __inline__ void _Event_Initialize (Event_Control *event)

  *Initializes an event control block to have no pending events.*
- static __inline__ bool _Event_sets_Is_empty (rtems_event_set the_event_set)

  *Checks if the event set is empty.*
- static __inline__ void _Event_sets_Post (rtems_event_set the_new_events, rtems_event_set *the_event_↩ set)

  *Posts the events in the specified event set.*
- static __inline__ rtems_event_set _Event_sets_Get (rtems_event_set the_event_set, rtems_event_set the↩ _event_condition)

  *Gets the events of the specified event condition.*
- static __inline__ rtems_event_set _Event_sets_Clear (rtems_event_set the_event_set, rtems_event_set the_mask)

  *Clears a set of events from an event set.*

### 10.91.1 Detailed Description

This header file defines interfaces used by the event implementation.

## 10.92 cpukit/include/rtems/rtems/intr.h File Reference

This header file defines the Interrupt Manager API.

```
#include <rtems/rtems/status.h>
#include <rtems/score/basedefs.h>
#include <rtems/score/cpu.h>
#include <rtems/score/isr.h>
#include <rtems/score/isrlevel.h>
#include <rtems/score/isrlock.h>
#include <rtems/score/smplock.h>
```

### Macros

- #define rtems_interrupt_cause(_interrupt_to_cause) %
  %
- #define rtems_interrupt_clear(_interrupt_to_clear) %
  %
- #define rtems_interrupt_is_in_progress() _ISR_Is_in_progress()
  %
- #define rtems_interrupt_local_disable(_isr_cookie) _ISR_Local_disable( _isr_cookie )
  %
- #define rtems_interrupt_local_enable(_isr_cookie) _ISR_Local_enable( _isr_cookie )
  %
- #define rtems_interrupt_lock_acquire(_lock, _lock_context) _ISR_lock_ISR_disable_and_acquire( _lock, ↩ _lock_context )
  %
- #define rtems_interrupt_lock_acquire_isr(_lock, _lock_context)
  %
- #define RTEMS_INTERRUPT_LOCK_DECLARE(_qualifier, _designator) ISR_LOCK_DECLARE( _qualifier, _designator )
  %
- #define RTEMS_INTERRUPT_LOCK_DEFINE(_qualifier, _designator, _name) ISR_LOCK_DEFINE( _↩ qualifier, _designator, _name )
  %
- #define rtems_interrupt_lock_destroy(_lock) _ISR_lock_Destroy( _lock )
  %
- #define rtems_interrupt_lock_initialize(_lock, _name) _ISR_lock_Initialize( _lock, _name )
  %
- #define RTEMS_INTERRUPT_LOCK_INITIALIZER(_name) ISR_LOCK_INITIALIZER( _name )
  %
- #define rtems_interrupt_lock_interrupt_disable(_lock_context) _ISR_lock_ISR_disable( _lock_context )
  %
- #define RTEMS_INTERRUPT_LOCK_MEMBER(_designator) ISR_LOCK_MEMBER( _designator )
  %

- #define RTEMS_INTERRUPT_LOCK_REFERENCE(_designator, _target) ISR_LOCK_REFERENCE( _designator, _target )

  %
- #define rtems_interrupt_lock_release(_lock, _lock_context) _ISR_lock_Release_and_ISR_enable( _lock, _lock_context )

  %
- #define rtems_interrupt_lock_release_isr(_lock, _lock_context)

  %

### Typedefs

- typedef ISR_Handler rtems_isr

  %
- typedef void(* rtems_isr_entry) (void *)

  *Interrupt service routines installed by rtems_interrupt_catch() shall have this function pointer type.*
- typedef ISR_Level rtems_interrupt_level

  %
- typedef ISR_lock_Control rtems_interrupt_lock

  %
- typedef ISR_lock_Context rtems_interrupt_lock_context

  %
- typedef ISR_Vector_number rtems_vector_number

  %

### Functions

- rtems_status_code rtems_interrupt_catch (rtems_isr_entry new_isr_handler, rtems_vector_number vector, rtems_isr_entry *old_isr_handler)

  %

#### 10.92.1 Detailed Description

This header file defines the Interrupt Manager API.

## 10.93 cpukit/include/rtems/rtems/mainpage.h File Reference

#### 10.93.1 Detailed Description

This file exists to provide a top level description of RTEMS for Doxygen.

# 10.94 cpukit/include/rtems/rtems/message.h File Reference

This header file defines the Message Manager API.

```
#include <stddef.h>
#include <stdint.h>
#include <rtems/rtems/attr.h>
#include <rtems/rtems/options.h>
#include <rtems/rtems/status.h>
#include <rtems/rtems/types.h>
#include <rtems/score/coremsgbuffer.h>
```

## Classes

- struct rtems_message_queue_config

  *This structure defines the configuration of a message queue constructed by rtems_message_queue_construct().*

## Macros

- #define RTEMS_MESSAGE_QUEUE_BUFFER(_maximum_message_size)

  *Defines a structure which can be used as a message queue buffer for messages of the specified maximum size.*

## Functions

- rtems_status_code rtems_message_queue_broadcast (rtems_id id, const void *buffer, size_t size, uint32_t *count)

  %
- rtems_status_code rtems_message_queue_construct (const rtems_message_queue_config *config, rtems_id *id)

  *Constructs a message queue from the specified the message queue configuration.*
- rtems_status_code rtems_message_queue_create (rtems_name name, uint32_t count, size_t max_↩ message_size, rtems_attribute attribute_set, rtems_id *id)

  %
- rtems_status_code rtems_message_queue_delete (rtems_id id)

  %
- rtems_status_code rtems_message_queue_flush (rtems_id id, uint32_t *count)

  %
- rtems_status_code rtems_message_queue_get_number_pending (rtems_id id, uint32_t *count)

  %
- rtems_status_code rtems_message_queue_ident (rtems_name name, uint32_t node, rtems_id *id)

  *Identifies a message queue object by the specified object name.*
- rtems_status_code rtems_message_queue_receive (rtems_id id, void *buffer, size_t *size, rtems_option option_set, rtems_interval timeout)

  %
- rtems_status_code rtems_message_queue_send (rtems_id id, const void *buffer, size_t size)

  %
- rtems_status_code rtems_message_queue_urgent (rtems_id id, const void *buffer, size_t size)

  %

### 10.94.1 Detailed Description

This header file defines the Message Manager API.

## 10.95 cpukit/include/rtems/rtems/messagedata.h File Reference

Classic Message Queue Manager API.

```
#include <rtems/rtems/message.h>
#include <rtems/score/coremsg.h>
#include <rtems/score/objectdata.h>
```

### Classes

- struct Message_queue_Control

### Macros

- #define MESSAGE_QUEUE_INFORMATION_DEFINE(max)

  *Macro to define the objects information for the Classic Message Queue objects.*

### Variables

- Objects_Information _Message_queue_Information

  *The Classic Message Queue objects information.*

### 10.95.1 Detailed Description

Classic Message Queue Manager API.

## 10.96 cpukit/include/rtems/rtems/messageimpl.h File Reference

Classic Message Queue Manager Implementation.

```
#include <rtems/rtems/messagedata.h>
#include <rtems/score/objectimpl.h>
#include <rtems/score/coremsgimpl.h>
```

### Enumerations

- enum Message_queue_Submit_types { MESSAGE_QUEUE_SEND_REQUEST = 0, MESSAGE_QUEUE_URGENT_REQUES
  = 1 }

### Functions

- rtems_status_code _Message_queue_Submit (rtems_id id, const void *buffer, size_t size, Message_queue_Submit_types submit_type)

  *Message_queue_Submit.*
- static __inline__ void _Message_queue_Free (Message_queue_Control *the_message_queue)

  *Deallocates a message queue control block into the inactive chain of free message queue control blocks.*
- static __inline__ Message_queue_Control * **_Message_queue_Get** (Objects_Id id, Thread_queue_Context *queue_context)
- static __inline__ Message_queue_Control * **_Message_queue_Allocate** (void)
- rtems_status_code _Message_queue_Create (const rtems_message_queue_config *config, rtems_id *id, CORE_message_queue_Allocate_buffers allocate_buffers)

  *Creates a message queue.*

### 10.96.1 Detailed Description

Classic Message Queue Manager Implementation.

## 10.97 cpukit/include/rtems/rtems/modes.h File Reference

This header file provides the task modes API of the Task Manager.

```
#include <stdint.h>
#include <rtems/score/cpu.h>
```

### Macros

- #define RTEMS_ALL_MODE_MASKS 0x0000ffff

  *This task mode constant is a bit mask with all mode bits set.*
- #define RTEMS_ASR 0x00000000

  *This task mode constant indicates that signal processing is disabled.*
- #define RTEMS_ASR_MASK 0x00000400

  *This mode constant corresponds to the signal enable/disable bit.*
- #define RTEMS_CURRENT_MODE 0

  *This task mode constant indicates that the current task mode of the executing task shall be returned by rtems_task_mode().*
- #define RTEMS_DEFAULT_MODES 0x00000000

  *This task mode constant represents the default mode set.*
- #define RTEMS_INTERRUPT_MASK CPU_MODES_INTERRUPT_MASK

  *This task mode constant corresponds to the interrupt enable/disable bits.*
- #define RTEMS_INTERRUPT_LEVEL(_interrupt_level) ( ( _interrupt_level ) & RTEMS_INTERRUPT_MASK )

  *Maps the interrupt level to the associated processor-dependent task mode interrupt level.*
- #define RTEMS_NO_ASR 0x00000400

  *This task mode constant indicates that signal processing is disabled.*
- #define RTEMS_NO_PREEMPT 0x00000100

  *This task mode constant indicates that preemption is disabled.*
- #define RTEMS_NO_TIMESLICE 0x00000000

*This task mode constant indicates that timeslicing is disabled.*

- #define RTEMS_PREEMPT 0x00000000

  *This task mode constant indicates that preemption is enabled.*

- #define RTEMS_PREEMPT_MASK 0x00000100

  *This task mode constant corresponds to the preemption enable/disable bit.*

- #define RTEMS_TIMESLICE 0x00000200

  *This task mode constant indicates that timeslicing is enabled.*

- #define RTEMS_TIMESLICE_MASK 0x00000200

  *This task mode constant corresponds to the timeslice enable/disable bit.*

### Typedefs

- typedef uint32_t rtems_mode

  *This type represents a Classic API task mode set.*

### Functions

- rtems_mode rtems_interrupt_level_body (uint32_t level)

  *Maps the interrupt level to the associated processor-dependent task mode interrupt level.*

### Variables

- const uint32_t rtems_interrupt_mask

  *This task mode constant has the same value as RTEMS_INTERRUPT_MASK.*

#### 10.97.1 Detailed Description

This header file provides the task modes API of the Task Manager.

## 10.98 cpukit/include/rtems/rtems/modesimpl.h File Reference

Classic Modes Implementation.

```
#include <rtems/rtems/modes.h>
#include <rtems/score/isrlevel.h>
```

## Functions

- static __inline__ bool _Modes_Mask_changed (rtems_mode mode_set, rtems_mode masks)

  *Checks if any of the mode flags in mask are set in mode_set.*
- static __inline__ bool _Modes_Is_asr_disabled (rtems_mode mode_set)

  *Checks if mode_set says that Asynchronous Signal Processing is disabled.*
- static __inline__ bool _Modes_Is_preempt (rtems_mode mode_set)

  *Checks if mode_set indicates that preemption is enabled.*
- static __inline__ bool _Modes_Is_timeslice (rtems_mode mode_set)

  *Checks if mode_set indicates that timeslicing is enabled.*
- static __inline__ ISR_Level _Modes_Get_interrupt_level (rtems_mode mode_set)

  *Gets the interrupt level portion of the mode_set.*
- static __inline__ void _Modes_Set_interrupt_level (rtems_mode mode_set)

  *Sets the current interrupt level to that specified in the mode_set.*
- static __inline__ void _Modes_Change (rtems_mode old_mode_set, rtems_mode new_mode_set, rtems_mode mask, rtems_mode *out_mode_set, rtems_mode *changed)

  *Changes the modes in old_mode_set indicated by mask to the requested values in new_mode_set.*

### 10.98.1 Detailed Description

Classic Modes Implementation.

# 10.99 cpukit/include/rtems/rtems/object.h File Reference

This header file defines the Object Manager API.

```
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <rtems/rtems/status.h>
#include <rtems/rtems/types.h>
#include <rtems/score/object.h>
```

## Classes

- struct rtems_object_api_class_information

  %

## Macros

- #define rtems_build_id(_api, _class, _node, _index) _Objects_Build_id( _api, _class, _node, _index )
  %
- #define rtems_build_name(_C1, _C2, _C3, _C4) _Objects_Build_name( _C1, _C2, _C3, _C4 )
  %
- #define rtems_object_id_api_maximum() OBJECTS_APIS_LAST
  %
- #define rtems_object_id_api_minimum() OBJECTS_INTERNAL_API
  %
- #define RTEMS_OBJECT_ID_FINAL OBJECTS_ID_FINAL
  %
- #define RTEMS_OBJECT_ID_FINAL_INDEX OBJECTS_ID_FINAL_INDEX
  %
- #define rtems_object_id_get_api(_id) _Objects_Get_API( _id )
  %
- #define rtems_object_id_get_class(_id) _Objects_Get_class( _id )
  %
- #define rtems_object_id_get_index(_id) _Objects_Get_index( _id )
  %
- #define rtems_object_id_get_node(_id) _Objects_Get_node( _id )
  %
- #define RTEMS_OBJECT_ID_INITIAL(_api, _class, _node) OBJECTS_ID_INITIAL( _api, _class, _node )
  %
- #define RTEMS_OBJECT_ID_INITIAL_INDEX OBJECTS_ID_INITIAL_INDEX
  %
- #define RTEMS_SEARCH_ALL_NODES OBJECTS_SEARCH_ALL_NODES
  %
- #define RTEMS_SEARCH_LOCAL_NODE OBJECTS_SEARCH_LOCAL_NODE
  %
- #define RTEMS_SEARCH_OTHER_NODES OBJECTS_SEARCH_OTHER_NODES
  %
- #define RTEMS_WHO_AM_I OBJECTS_WHO_AM_I
  %

## Functions

- int rtems_object_api_maximum_class (int api)
  %
- int rtems_object_api_minimum_class (int api)
  %
- const char * rtems_object_get_api_class_name (int the_api, int the_class)
  %
- const char * rtems_object_get_api_name (int api)
  %
- rtems_status_code rtems_object_get_class_information (int the_api, int the_class, rtems_object_api_class_information *info)
  %
- rtems_status_code rtems_object_get_classic_name (rtems_id id, rtems_name *name)
  %

- static uint16_t rtems_object_get_local_node (void)
  %
- char * rtems_object_get_name (rtems_id id, size_t length, char *name)
  %
- int rtems_object_id_api_maximum_class (int api)
  %
- rtems_status_code rtems_object_set_name (rtems_id id, const char *name)
  %

### 10.99.1 Detailed Description

This header file defines the Object Manager API.

## 10.100 cpukit/include/rtems/score/object.h File Reference

Constants and Structures Associated with the Object Handler.

```
#include <rtems/score/basedefs.h>
#include <rtems/score/cpu.h>
```

### Classes

- union Objects_Name

### Macros

- #define OBJECTS_INDEX_START_BIT 0U
- #define OBJECTS_NODE_START_BIT 16U
- #define OBJECTS_API_START_BIT 24U
- #define OBJECTS_CLASS_START_BIT 27U
- #define OBJECTS_INDEX_MASK (Objects_Id)0x0000ffffU
- #define OBJECTS_NODE_MASK (Objects_Id)0x00ff0000U
- #define OBJECTS_API_MASK (Objects_Id)0x07000000U
- #define OBJECTS_CLASS_MASK (Objects_Id)0xf8000000U
- #define OBJECTS_INDEX_VALID_BITS (Objects_Id)0x0000ffffU
- #define OBJECTS_NODE_VALID_BITS (Objects_Id)0x000000ffU
- #define OBJECTS_API_VALID_BITS (Objects_Id)0x00000007U
- #define OBJECTS_CLASS_VALID_BITS (Objects_Id)0x0000001fU
- #define OBJECTS_UNLIMITED_OBJECTS 0x80000000U
- #define OBJECTS_ID_INITIAL_INDEX (0)
- #define OBJECTS_ID_FINAL_INDEX (0xffffU)
- #define OBJECTS_APIS_LAST OBJECTS_POSIX_API
- #define OBJECTS_ID_NONE 0
- #define OBJECTS_ID_OF_SELF ((Objects_Id) 0)
- #define OBJECTS_SEARCH_ALL_NODES 0
- #define OBJECTS_SEARCH_OTHER_NODES 0x7FFFFFFE
- #define OBJECTS_SEARCH_LOCAL_NODE 0x7FFFFFFF

- #define OBJECTS_WHO_AM_I 0
- #define OBJECTS_ID_INITIAL(_api, _class, _node) _Objects_Build_id( (_api), (_class), (_node), OBJECTS_ID_INITIAL_INDEX )
- #define OBJECTS_ID_FINAL ((Objects_Id)~0)
- #define _Objects_Build_name(_C1, _C2, _C3, _C4)
- #define _Objects_Build_id(the_api, the_class, node, index)

  *Builds an object ID from its components.*
- #define _Objects_Is_unlimited(maximum) ( ( ( maximum ) & OBJECTS_UNLIMITED_OBJECTS ) != 0 )
- #define **_Objects_Maximum_per_allocation**(maximum) ((Objects_Maximum) ((maximum) & ~OBJECTS_UNLIMITED_OBJE
- #define _Objects_Local_node ((uint16_t) 1)

  *The local MPCI node number.*

### Typedefs

- typedef uint32_t Objects_Id
- typedef uint16_t Objects_Maximum

### Enumerations

- enum Objects_APIs {
  **OBJECTS_NO_API** = 0, **OBJECTS_INTERNAL_API** = 1, **OBJECTS_CLASSIC_API** = 2, **OBJECTS_PO↩ SIX_API** = 3,
  **OBJECTS_FAKE_OBJECTS_API** = 7 }

### Functions

- static __inline__ Objects_APIs _Objects_Get_API (Objects_Id id)

  *Returns the API portion of the ID.*
- static __inline__ uint32_t _Objects_Get_class (Objects_Id id)

  *Returns the class portion of the ID.*
- static __inline__ uint32_t _Objects_Get_node (Objects_Id id)

  *Returns the node portion of the ID.*
- static __inline__ Objects_Maximum _Objects_Get_index (Objects_Id id)

  *Returns the index portion of the ID.*

#### 10.100.1 Detailed Description

Constants and Structures Associated with the Object Handler.

This include file contains all the constants and structures associated with the Object Handler. This Handler provides mechanisms which can be used to initialize and manipulate all objects which have ids.

## 10.101 cpukit/include/rtems/rtems/objectimpl.h File Reference

Implementation Interfaces for Classic Objects.

```
#include <rtems/score/objectimpl.h>
#include <rtems/rtems/status.h>
```

### Functions

- rtems_status_code _RTEMS_Name_to_id (uint32_t name, uint32_t node, Objects_Id *id, const Objects_Information *information)

  *Calls _Objects_Name_to_id_u32() and converts the status.*

### 10.101.1 Detailed Description

Implementation Interfaces for Classic Objects.

## 10.102 cpukit/include/rtems/score/objectimpl.h File Reference

Inlined Routines in the Object Handler.

```
#include <rtems/score/objectdata.h>
#include <rtems/score/apimutex.h>
#include <rtems/score/assert.h>
#include <rtems/score/isrlock.h>
#include <rtems/score/sysstate.h>
#include <rtems/score/threaddispatch.h>
```

### Macros

- #define OBJECTS_INTERNAL_CLASSES_LAST OBJECTS_INTERNAL_THREADS
- #define OBJECTS_RTEMS_CLASSES_LAST OBJECTS_RTEMS_BARRIERS
- #define OBJECTS_POSIX_CLASSES_LAST OBJECTS_POSIX_SHMS
- #define _Objects_Maximum_nodes 1
- #define OBJECTS_INDEX_MINIMUM 1U
- #define OBJECTS_NAME_ERRORS_FIRST OBJECTS_NAME_OR_ID_LOOKUP_SUCCESSFUL
- #define OBJECTS_NAME_ERRORS_LAST OBJECTS_INVALID_NODE

### Typedefs

- typedef bool(* Objects_Name_comparators) (void *, void *, uint16_t)

### Enumerations

- enum **Objects_Fake_objects_API** { **OBJECTS_FAKE_OBJECTS_NO_CLASS** = 0, **OBJECTS_FAKE_↩ OBJECTS_SCHEDULERS** = 1 }
- enum Objects_Name_or_id_lookup_errors { **OBJECTS_NAME_OR_ID_LOOKUP_SUCCESSFUL**, **OBJECTS_INVALID_NAME**, **OBJECTS_INVALI↩ D_ADDRESS**, **OBJECTS_INVALID_ID**, **OBJECTS_INVALID_NODE** }
- enum **Objects_Get_by_name_error** { **OBJECTS_GET_BY_NAME_INVALID_NAME**, **OBJECTS_GET_↩ BY_NAME_NAME_TOO_LONG**, **OBJECTS_GET_BY_NAME_NO_OBJECT** }

## Functions

- Objects_Maximum _Objects_Extend_information (Objects_Information *information)

  *Extends an object class information record.*
- void _Objects_Free_objects_block (Objects_Information *information, Objects_Maximum block)

  *Free the objects block with the specified index.*
- void _Objects_Shrink_information (Objects_Information *information)

  *Shrinks an object class information record.*
- void _Objects_Initialize_information (Objects_Information *information)

  *Initializes the specified objects information.*
- unsigned int _Objects_API_maximum_class (uint32_t api)

  *Returns highest numeric value of a valid API for the specified API.*
- Objects_Control * _Objects_Allocate (Objects_Information *information)

  *Allocates an object.*
- Objects_Name_or_id_lookup_errors _Objects_Name_to_id_u32 (uint32_t name, uint32_t node, Objects_Id *id, const Objects_Information *information)

  *Searches an object of the specified class with the specified name on the specified set of nodes.*
- Objects_Control * _Objects_Get_by_name (const Objects_Information *information, const char *name, size_t *name_length_p, Objects_Get_by_name_error *error)

  *Gets an object control block identified by its name.*
- Objects_Name_or_id_lookup_errors _Objects_Id_to_name (Objects_Id id, Objects_Name *name)

  *Returns the name associated with object id.*
- Objects_Control * _Objects_Get (Objects_Id id, ISR_lock_Context *lock_context, const Objects_Information *information)

  *Maps the specified object identifier to the associated local object control block.*
- Objects_Control * _Objects_Get_no_protection (Objects_Id id, const Objects_Information *information)

  *Maps object ids to object control blocks.*
- Objects_Control * _Objects_Get_next (Objects_Id id, const Objects_Information *information, Objects_Id *next_id_p)

  *Gets the next open object after the specified object identifier.*
- Objects_Information * _Objects_Get_information (Objects_APIs the_api, uint16_t the_class)

  *Gets object information.*
- Objects_Information * _Objects_Get_information_id (Objects_Id id)

  *Gets information of an object from an ID.*
- static __inline__ bool _Objects_Has_string_name (const Objects_Information *information)

  *Returns if the object has a string name.*
- char * _Objects_Get_name_as_string (Objects_Id id, size_t length, char *name)

  *Gets object name in the form of a C string.*
- size_t _Objects_Name_to_string (Objects_Name name, bool is_string, char *buffer, size_t buffer_size)

  *Converts the specified object name to a text representation.*
- bool _Objects_Set_name (const Objects_Information *information, Objects_Control *the_object, const char *name)

  *Sets objects name.*
- static __inline__ void _Objects_Namespace_remove_u32 (const Objects_Information *information, Objects_Control *the_object)

  *Removes object with a 32-bit integer name from its namespace.*
- void _Objects_Namespace_remove_string (const Objects_Information *information, Objects_Control *the_object)

  *Removes object with a string name from its namespace.*
- void _Objects_Close (const Objects_Information *information, Objects_Control *the_object)

  *Closes object.*
- Objects_Maximum _Objects_Active_count (const Objects_Information *information)

*Returns the count of active objects.*

- static __inline__ Objects_Maximum _Objects_Extend_size (const Objects_Information *information)

  *Returns the object's objects per block.*
- static __inline__ bool _Objects_Is_api_valid (uint32_t the_api)

  *Checks if the api is valid.*
- static __inline__ bool _Objects_Is_local_node (uint32_t node)

  *Checks if the node is of the local object.*
- static __inline__ bool _Objects_Is_local_id (Objects_Id id RTEMS_UNUSED)

  *Checks if the id is of a local object.*
- static __inline__ bool _Objects_Are_ids_equal (Objects_Id left, Objects_Id right)

  *Checks if two object IDs are equal.*
- static __inline__ Objects_Id _Objects_Get_minimum_id (Objects_Id id)

  *Returns the identifier with the minimum index for the specified identifier.*
- static __inline__ Objects_Maximum _Objects_Get_maximum_index (const Objects_Information *information)

  *Returns the maximum index of the specified object class.*
- static __inline__ Objects_Control * _Objects_Get_inactive (Objects_Information *information)

  *Get an inactive object or NULL.*
- static __inline__ Objects_Maximum _Objects_Is_auto_extend (const Objects_Information *information)

  *Checks if the automatic object extension (unlimited objects) is enabled.*
- static __inline__ void _Objects_Set_local_object (const Objects_Information *information, uint32_t index, Objects_Control *the_object)

  *Sets the pointer to the local_table object referenced by the index.*
- static __inline__ void _Objects_Invalidate_Id (const Objects_Information *information, Objects_Control *the_object)

  *Invalidates an object Id.*
- static __inline__ void _Objects_Open (Objects_Information *information, Objects_Control *the_object, Objects_Name name)

  *Places the_object control pointer and object name in the Local Pointer and Local Name Tables, respectively.*
- static __inline__ void _Objects_Open_u32 (const Objects_Information *information, Objects_Control *the_object, uint32_t name)

  *Places the_object control pointer and object name in the Local Pointer and Local Name Tables, respectively.*
- static __inline__ void _Objects_Open_string (const Objects_Information *information, Objects_Control *the_object, const char *name)

  *Places the_object control pointer and object name in the Local Pointer and Local Name Tables, respectively.*
- static __inline__ void _Objects_Allocator_lock (void)

  *Locks the object allocator mutex.*
- static __inline__ void _Objects_Allocator_unlock (void)

  *Unlocks the object allocator mutex.*
- static __inline__ bool _Objects_Allocator_is_owner (void)

  *Checks if the allocator is the owner of the object allocator mutex.*
- static __inline__ Objects_Control * _Objects_Allocate_unprotected (Objects_Information *information)

  *Allocates an object without locking the allocator mutex.*
- static __inline__ void _Objects_Free (Objects_Information *information, Objects_Control *the_object)

  *Frees an object.*
- static __inline__ void _Objects_Activate_unlimited (Objects_Information *information, Objects_Control *the_object)

  *Activate the object.*
- static __inline__ Objects_Control * _Objects_Allocate_with_extend (Objects_Information *information, void(*extend)(Objects_Information *))

  *Allocate an object and extend the objects information on demand.*

### Variables

- Objects_Information **const _Objects_Information_table [OBJECTS_APIS_LAST+1]

### 10.102.1 Detailed Description

Inlined Routines in the Object Handler.

This include file contains the static inline implementation of all of the inlined routines in the Object Handler.

## 10.103 cpukit/include/rtems/rtems/options.h File Reference

This header file provides the Classic API directive options.

```
#include <stdint.h>
```

### Macros

- #define RTEMS_DEFAULT_OPTIONS 0x00000000

  *This option constant represents the default option set.*
- #define RTEMS_EVENT_ALL 0x00000000

  *This option constant indicates that the task wishes to wait until all events of interest are available in rtems_event_receive() and rtems_event_system_receive().*
- #define RTEMS_EVENT_ANY 0x00000002

  *This option constant indicates that the task wishes to wait until at least one of the events of interest is available in rtems_event_receive() and rtems_event_system_receive().*
- #define RTEMS_NO_WAIT 0x00000001

  *This option constant indicates that the task does not want to wait on the resource.*
- #define RTEMS_WAIT 0x00000000

  *This option constant indicates that the task wants to wait on the resource.*

### Typedefs

- typedef uint32_t rtems_option

  *This type represents a Classic API directive option set.*

### 10.103.1 Detailed Description

This header file provides the Classic API directive options.

## 10.104 cpukit/include/rtems/rtems/optionsimpl.h File Reference

Classic Options Implementation.

```
#include <rtems/rtems/options.h>
```

### Functions

- RTEMS_INLINE_ROUTINE bool _Options_Is_no_wait (rtems_option option_set)

  *Checks if the RTEMS_NO_WAIT option is enabled in option_set.*
- RTEMS_INLINE_ROUTINE bool _Options_Is_any (rtems_option option_set)

  *Checks if the RTEMS_EVENT_ANY option is enabled in OPTION_SET.*

### 10.104.1 Detailed Description

Classic Options Implementation.

## 10.105 cpukit/include/rtems/rtems/part.h File Reference

This header file provides the Partition Manager API.

```
#include <stddef.h>
#include <stdint.h>
#include <rtems/rtems/attr.h>
#include <rtems/rtems/status.h>
#include <rtems/rtems/types.h>
#include <rtems/score/cpu.h>
```

### Macros

- #define RTEMS_PARTITION_ALIGNMENT CPU_SIZEOF_POINTER

  *This constant defines the minimum alignment of a partition buffer in bytes.*

### Functions

- rtems_status_code rtems_partition_create (rtems_name name, void *starting_address, uintptr_t length, size_t buffer_size, rtems_attribute attribute_set, rtems_id *id)

  *Creates a partition.*
- rtems_status_code rtems_partition_ident (rtems_name name, uint32_t node, rtems_id *id)

  *Identifies a partition by the object name.*
- rtems_status_code rtems_partition_delete (rtems_id id)

  *Deletes the partition.*
- rtems_status_code rtems_partition_get_buffer (rtems_id id, void **buffer)

  *Tries to get a buffer from the partition.*
- rtems_status_code rtems_partition_return_buffer (rtems_id id, void *buffer)

  *Returns the buffer to the partition.*

### 10.105.1 Detailed Description

This header file provides the Partition Manager API.

## 10.106 cpukit/include/rtems/rtems/partdata.h File Reference

This header file defines the API used by the Application Configuration to define the Partition Manager information.

```
#include <rtems/rtems/part.h>
#include <rtems/score/isrlock.h>
#include <rtems/score/objectdata.h>
```

### Classes

- struct Partition_Control

  *This structure is the Partition Control Block (PTCB).*

### Macros

- #define PARTITION_INFORMATION_DEFINE(_max)

  *Defines the Partition Manager objects information.*

### Variables

- Objects_Information _Partition_Information

  *The Partition Manager objects information is used to manage the objects of this class.*

### 10.106.1 Detailed Description

This header file defines the API used by the Application Configuration to define the Partition Manager information.

## 10.107 cpukit/include/rtems/rtems/partimpl.h File Reference

This header file defines interfaces used by the Partition Manager implementation.

```
#include <rtems/rtems/partdata.h>
#include <rtems/score/objectimpl.h>
```

### Functions

- static __inline__ Partition_Control * _Partition_Get (Objects_Id id, ISR_lock_Context *lock_context)

  *Calls _Objects_Get() using the Partition Manager information.*
- static __inline__ void _Partition_Acquire_critical (Partition_Control *the_partition, ISR_lock_Context *lock↩_context)

  *Acquires the partition lock in an ISR disabled section.*
- static __inline__ void _Partition_Release (Partition_Control *the_partition, ISR_lock_Context *lock_context)

  *Releases the partition lock and restores the ISR level.*

### 10.107.1 Detailed Description

This header file defines interfaces used by the Partition Manager implementation.

## 10.108 cpukit/include/rtems/rtems/ratemon.h File Reference

This header file defines the Rate-Monotonic Manager API.

```
#include <stdint.h>
#include <sys/_timespec.h>
#include <rtems/rtems/status.h>
#include <rtems/rtems/types.h>
#include <rtems/score/watchdogticks.h>
```

### Classes

- struct rtems_rate_monotonic_period_statistics
  %
- struct rtems_rate_monotonic_period_status
  %

### Macros

- #define RTEMS_PERIOD_STATUS WATCHDOG_NO_TIMEOUT
  *This constant is the interval passed to the rtems_rate_monotonic_period() directive to obtain status information.*

### Enumerations

- enum rtems_rate_monotonic_period_states { RATE_MONOTONIC_INACTIVE, RATE_MONOTONIC_ACTIVE, RATE_MONOTONIC_EXPIRED }
  %

### Functions

- rtems_status_code rtems_rate_monotonic_cancel (rtems_id id)
  %
- rtems_status_code rtems_rate_monotonic_create (rtems_name name, rtems_id *id)
  %
- rtems_status_code rtems_rate_monotonic_delete (rtems_id id)
  %
- rtems_status_code rtems_rate_monotonic_ident (rtems_name name, rtems_id *id)
  *Identifies a period object by the specified object name.*
- rtems_status_code rtems_rate_monotonic_period (rtems_id id, rtems_interval length)
  %
- rtems_status_code rtems_rate_monotonic_get_statistics (rtems_id id, rtems_rate_monotonic_period_statistics *statistics)

%

- rtems_status_code rtems_rate_monotonic_get_status (rtems_id id, rtems_rate_monotonic_period_status *status)

  %

- void rtems_rate_monotonic_report_statistics (void)

  %

- void rtems_rate_monotonic_report_statistics_with_plugin (const struct rtems_printer *printer)

  %

- void rtems_rate_monotonic_reset_all_statistics (void)

  %

- rtems_status_code rtems_rate_monotonic_reset_statistics (rtems_id id)

  %

### 10.108.1 Detailed Description

This header file defines the Rate-Monotonic Manager API.

## 10.109 cpukit/include/rtems/rtems/ratemondata.h File Reference

Classic Rate Monotonic Scheduler Data Structures.

```
#include <rtems/rtems/ratemon.h>
#include <rtems/score/timestamp.h>
#include <rtems/score/thread.h>
#include <rtems/score/watchdog.h>
```

### Classes

- struct Rate_monotonic_Statistics
- struct Rate_monotonic_Control

  *The following structure defines the control block used to manage each period.*

### Macros

- #define RATE_MONOTONIC_INFORMATION_DEFINE(max)

  *Macro to define the objects information for the Classic Rate Monotonic objects.*

### Variables

- Objects_Information _Rate_monotonic_Information

  *The Classic Rate Monotonic objects information.*

### 10.109.1 Detailed Description

Classic Rate Monotonic Scheduler Data Structures.

# 10.110 cpukit/include/rtems/rtems/ratemonimpl.h File Reference

Classic Rate Monotonic Scheduler Implementation.

```
#include <rtems/rtems/ratemondata.h>
#include <rtems/score/objectimpl.h>
#include <rtems/score/schedulerimpl.h>
#include <rtems/score/threadimpl.h>
#include <rtems/score/watchdogimpl.h>
#include <string.h>
```

## Macros

- #define **RATE_MONOTONIC_INTEND_TO_BLOCK** ( THREAD_WAIT_CLASS_PERIOD | THREAD_WAIT_STATE_INTEND )
- #define **RATE_MONOTONIC_BLOCKED** ( THREAD_WAIT_CLASS_PERIOD | THREAD_WAIT_STATE_BLOCKED )
- #define **RATE_MONOTONIC_READY_AGAIN** ( THREAD_WAIT_CLASS_PERIOD | THREAD_WAIT_STATE_READY_AGA )

## Functions

- static __inline__ Rate_monotonic_Control * _Rate_monotonic_Allocate (void)

  *Allocates a period control block from the inactive chain of free period control blocks.*
- static __inline__ void **_Rate_monotonic_Acquire_critical** (Rate_monotonic_Control *the_period, ISR_lock_Context *lock_context)
- static __inline__ void **_Rate_monotonic_Release** (Rate_monotonic_Control *the_period, ISR_lock_Context *lock_context)
- static __inline__ Rate_monotonic_Control * **_Rate_monotonic_Get** (Objects_Id id, ISR_lock_Context *lock_context)
- void **_Rate_monotonic_Timeout** (Watchdog_Control *watchdog)
- bool _Rate_monotonic_Get_status (const Rate_monotonic_Control *the_period, Timestamp_Control *wall↩ _since_last_period, Timestamp_Control *cpu_since_last_period)

  *_Rate_monotonic_Get_status(*
- void **_Rate_monotonic_Restart** (Rate_monotonic_Control *the_period, Thread_Control *owner, ISR_lock_Context *lock_context)
- void **_Rate_monotonic_Cancel** (Rate_monotonic_Control *the_period, Thread_Control *owner, ISR_lock_Context *lock_context)
- static __inline__ void **_Rate_monotonic_Reset_min_time** (Timestamp_Control *min_time)
- static __inline__ void **_Rate_monotonic_Reset_statistics** (Rate_monotonic_Control *the_period)

### 10.110.1 Detailed Description

Classic Rate Monotonic Scheduler Implementation.

## 10.111 cpukit/include/rtems/rtems/region.h File Reference

This header file defines the Region Manager API.

```
#include <stdint.h>
#include <rtems/rtems/attr.h>
#include <rtems/rtems/options.h>
#include <rtems/rtems/status.h>
#include <rtems/rtems/types.h>
#include <rtems/score/heapinfo.h>
```

### Functions

- rtems_status_code rtems_region_create (rtems_name name, void *starting_address, uintptr_t length, uintptr_t page_size, rtems_attribute attribute_set, rtems_id *id)
  %
- rtems_status_code rtems_region_delete (rtems_id id)
  %
- rtems_status_code rtems_region_extend (rtems_id id, void *starting_address, uintptr_t length)
  %
- rtems_status_code rtems_region_get_free_information (rtems_id id, Heap_Information_block *the_info)
  %
- rtems_status_code rtems_region_get_information (rtems_id id, Heap_Information_block *the_info)
  %
- rtems_status_code rtems_region_get_segment (rtems_id id, uintptr_t size, rtems_option option_set, rtems_interval timeout, void **segment)
  %
- rtems_status_code rtems_region_get_segment_size (rtems_id id, void *segment, uintptr_t *size)
  %
- rtems_status_code rtems_region_ident (rtems_name name, rtems_id *id)
  *Identifies a region object by the specified object name.*
- rtems_status_code rtems_region_resize_segment (rtems_id id, void *segment, uintptr_t size, uintptr_t *old_size)
  %
- rtems_status_code rtems_region_return_segment (rtems_id id, void *segment)
  %

### 10.111.1 Detailed Description

This header file defines the Region Manager API.

## 10.112 cpukit/include/rtems/rtems/regiondata.h File Reference

Classic Region Manager Data Structures.

```
#include <rtems/rtems/region.h>
#include <rtems/score/heap.h>
#include <rtems/score/objectdata.h>
#include <rtems/score/threadq.h>
```

### Classes

- struct Region_Control

### Macros

- #define REGION_INFORMATION_DEFINE(max)

  *Macro to define the objects information for the Classic Region objects.*

### Variables

- Objects_Information _Region_Information

  *The Classic Region objects information.*

### 10.112.1 Detailed Description

Classic Region Manager Data Structures.

## 10.113 cpukit/include/rtems/rtems/regionimpl.h File Reference

Classic Region Manager Implementation.

```
#include <rtems/rtems/regiondata.h>
#include <rtems/score/apimutex.h>
#include <rtems/score/heapimpl.h>
#include <rtems/score/objectimpl.h>
#include <rtems/score/threadqimpl.h>
```

### Macros

- #define **REGION_OF_THREAD_QUEUE_QUEUE**(queue) RTEMS_CONTAINER_OF( queue, Region_Control, Wait_queue.Queue )

### Functions

- static __inline__ Region_Control * _Region_Allocate (void)

  *Region_Allocate.*
- static __inline__ void _Region_Free (Region_Control *the_region)

  *Region_Free.*
- static __inline__ Region_Control * **_Region_Get_and_lock** (Objects_Id id)
- static __inline__ void **_Region_Unlock** (Region_Control *the_region)
- static __inline__ void * _Region_Allocate_segment (Region_Control *the_region, uintptr_t size)

  *Region_Allocate_segment.*
- static __inline__ bool _Region_Free_segment (Region_Control *the_region, void *the_segment)

  *Region_Free_segment.*
- void _Region_Process_queue (Region_Control *the_region)

  *Process Region Queue.*

### 10.113.1 Detailed Description

Classic Region Manager Implementation.

## 10.114 cpukit/include/rtems/rtems/sem.h File Reference

This header file defines the Semaphore Manager API.

```
#include <stdint.h>
#include <rtems/rtems/attr.h>
#include <rtems/rtems/options.h>
#include <rtems/rtems/status.h>
#include <rtems/rtems/tasks.h>
#include <rtems/rtems/types.h>
```

### Functions

- rtems_status_code rtems_semaphore_create (rtems_name name, uint32_t count, rtems_attribute attribute_set, rtems_task_priority priority_ceiling, rtems_id *id)

  *Creates a semaphore with the specified properties and returns its identifier.*
- rtems_status_code rtems_semaphore_delete (rtems_id id)

  %
- rtems_status_code rtems_semaphore_flush (rtems_id id)

  %
- rtems_status_code rtems_semaphore_ident (rtems_name name, uint32_t node, rtems_id *id)

  *Identifies a semaphore object by the specified object name.*
- rtems_status_code rtems_semaphore_obtain (rtems_id id, rtems_option option_set, rtems_interval timeout)

  %
- rtems_status_code rtems_semaphore_release (rtems_id id)

  %
- rtems_status_code rtems_semaphore_set_priority (rtems_id semaphore_id, rtems_id scheduler_id, rtems_task_priority new_priority, rtems_task_priority *old_priority)

  %

### 10.114.1 Detailed Description

This header file defines the Semaphore Manager API.

## 10.115 cpukit/include/rtems/rtems/semdata.h File Reference

Classic Semaphore Manager Data Structures.

```
#include <rtems/rtems/sem.h>
#include <rtems/score/coremutex.h>
#include <rtems/score/coresem.h>
#include <rtems/score/mrsp.h>
#include <rtems/score/object.h>
```

### Classes

- struct Semaphore_Control

### Macros

- #define SEMAPHORE_INFORMATION_DEFINE(max, scheduler_count)

  *Macro to define the objects information for the Classic Semaphore objects.*

### Variables

- Objects_Information _Semaphore_Information

  *The Classic Semaphore objects information.*

### 10.115.1 Detailed Description

Classic Semaphore Manager Data Structures.

## 10.116 cpukit/include/rtems/rtems/semimpl.h File Reference

Classic Semaphore Manager Implementation.

```
#include <rtems/rtems/semdata.h>
#include <rtems/score/coremuteximpl.h>
#include <rtems/score/coresemimpl.h>
#include <rtems/score/mrspimpl.h>
```

### Enumerations

- enum Semaphore_Variant {
  **SEMAPHORE_VARIANT_MUTEX_INHERIT_PRIORITY**, **SEMAPHORE_VARIANT_MUTEX_PRIORITY_CEILING**, **SEMAPHORE_VARIANT_MUTEX_NO_PROTOCOL**, **SEMAPHORE_VARIANT_SIMPLE_BINARY**,
  **SEMAPHORE_VARIANT_COUNTING**, **SEMAPHORE_VARIANT_MRSP** }

  *Classic semaphore variants.*
- enum **Semaphore_Discipline** { **SEMAPHORE_DISCIPLINE_PRIORITY**, **SEMAPHORE_DISCIPLINE_FIFO** }

### Functions

- static __inline__ uintptr_t **_Semaphore_Get_flags** (const Semaphore_Control *the_semaphore)
- static __inline__ void **_Semaphore_Set_flags** (Semaphore_Control *the_semaphore, uintptr_t flags)
- static __inline__ Semaphore_Variant **_Semaphore_Get_variant** (uintptr_t flags)
- static __inline__ uintptr_t **_Semaphore_Set_variant** (uintptr_t flags, Semaphore_Variant variant)
- static __inline__ Semaphore_Discipline **_Semaphore_Get_discipline** (uintptr_t flags)
- static __inline__ uintptr_t **_Semaphore_Set_discipline** (uintptr_t flags, Semaphore_Discipline discipline)
- static __inline__ const Thread_queue_Operations * **_Semaphore_Get_operations** (uintptr_t flags)
- static __inline__ Semaphore_Control * _Semaphore_Allocate (void)

  *Allocates a semaphore control block from the inactive chain of free semaphore control blocks.*
- static __inline__ void _Semaphore_Free (Semaphore_Control *the_semaphore)

  *Frees a semaphore control block to the inactive chain of free semaphore control blocks.*
- static __inline__ Semaphore_Control * **_Semaphore_Get** (Objects_Id id, Thread_queue_Context *queue_context)

### 10.116.1 Detailed Description

Classic Semaphore Manager Implementation.

## 10.117 cpukit/include/rtems/rtems/signal.h File Reference

This header file defines the parts of the Signal Manager API.

```
#include <rtems/rtems/asr.h>
#include <rtems/rtems/modes.h>
#include <rtems/rtems/status.h>
#include <rtems/rtems/types.h>
```

### Functions

- rtems_status_code rtems_signal_catch (rtems_asr_entry asr_handler, rtems_mode mode_set)
  %
- rtems_status_code rtems_signal_send (rtems_id id, rtems_signal_set signal_set)
  %

### 10.117.1 Detailed Description

This header file defines the parts of the Signal Manager API.

## 10.118 cpukit/include/rtems/rtems/signalimpl.h File Reference

Signals Implementation.

```
#include <rtems/rtems/signal.h>
#include <rtems/score/thread.h>
```

### Functions

- void **_Signal_Action_handler** (Thread_Control *executing, Thread_Action *action, ISR_lock_Context *lock_context)

### 10.118.1 Detailed Description

Signals Implementation.

# 10.119 cpukit/include/rtems/rtems/status.h File Reference

This header file provides the status codes of Classic API directives and support functions.

```
#include <stdbool.h>
```

## Macros

- #define RTEMS_STATUS_CODES_FIRST RTEMS_SUCCESSFUL

  *This constant represents the lowest valid value for a Classic API directive status code.*
- #define RTEMS_STATUS_CODES_LAST RTEMS_PROXY_BLOCKING

  *This constant represents the highest valid value for a Classic API directive status code.*

## Enumerations

- enum rtems_status_code {
  RTEMS_SUCCESSFUL = 0, RTEMS_TASK_EXITTED = 1, RTEMS_MP_NOT_CONFIGURED = 2,
  RTEMS_INVALID_NAME = 3,
  RTEMS_INVALID_ID = 4, RTEMS_TOO_MANY = 5, RTEMS_TIMEOUT = 6, RTEMS_OBJECT_WAS_DELETED
  = 7,
  RTEMS_INVALID_SIZE = 8, RTEMS_INVALID_ADDRESS = 9, RTEMS_INVALID_NUMBER = 10,
  RTEMS_NOT_DEFINED = 11,
  RTEMS_RESOURCE_IN_USE = 12, RTEMS_UNSATISFIED = 13, RTEMS_INCORRECT_STATE = 14,
  RTEMS_ALREADY_SUSPENDED = 15,
  RTEMS_ILLEGAL_ON_SELF = 16, RTEMS_ILLEGAL_ON_REMOTE_OBJECT = 17, RTEMS_CALLED_FROM_ISR
  = 18, RTEMS_INVALID_PRIORITY = 19,
  RTEMS_INVALID_CLOCK = 20, RTEMS_INVALID_NODE = 21, RTEMS_NOT_CONFIGURED = 22,
  RTEMS_NOT_OWNER_OF_RESOURCE = 23,
  RTEMS_NOT_IMPLEMENTED = 24, RTEMS_INTERNAL_ERROR = 25, RTEMS_NO_MEMORY = 26,
  RTEMS_IO_ERROR = 27,
  RTEMS_INTERRUPTED = 28, RTEMS_PROXY_BLOCKING = 29 }

  *This enumeration provides status codes for directives of the Classic API.*

## Functions

- int rtems_status_code_to_errno (rtems_status_code status_code)

  *Maps the RTEMS status code to a POSIX error number.*
- static bool rtems_are_statuses_equal (rtems_status_code left_status_code, rtems_status_code right_↩ status_code)

  *Checks if the status codes are equal.*
- static bool rtems_is_status_successful (rtems_status_code status_code)

  *Checks if the status code is RTEMS_SUCCESSFUL.*
- const char * rtems_status_text (rtems_status_code status_code)

  *Maps the status code to a descriptive text.*

### 10.119.1 Detailed Description

This header file provides the status codes of Classic API directives and support functions.

## 10.120 cpukit/include/rtems/rtems/statusimpl.h File Reference

Classic Status Implementation.

```
#include <rtems/rtems/status.h>
#include <rtems/score/threadimpl.h>
```

### Functions

- static __inline__ rtems_status_code **_Status_Get** (Status_Control status)
- static __inline__ rtems_status_code **_Status_Get_after_wait** (const Thread_Control *executing)

### Variables

- const rtems_status_code _Status_Object_name_errors_to_status [ ]

  *Status Object Name Errors to Status Array.*

### 10.120.1 Detailed Description

Classic Status Implementation.

## 10.121 cpukit/include/rtems/rtems/support.h File Reference

This header file defines support services of the API.

```
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <rtems/config.h>
#include <rtems/rtems/status.h>
#include <rtems/rtems/types.h>
#include <rtems/score/heapinfo.h>
```

### Macros

- #define RTEMS_MICROSECONDS_TO_TICKS(_us) ( ( _us ) / rtems_configuration_get_microseconds_per_tick() )

  *Returns the number of clock ticks for the specified microseconds value.*
- #define RTEMS_MILLISECONDS_TO_MICROSECONDS(_ms) ( ( _ms ) * 1000UL )

  *Returns the number of microseconds for the specified milliseconds value.*
- #define RTEMS_MILLISECONDS_TO_TICKS(_ms) RTEMS_MICROSECONDS_TO_TICKS( RTEMS_MILLISECONDS_TO_ _ms ) )

  *Returns the number of clock ticks for the specified milliseconds value.*

### Functions

- static bool rtems_is_name_valid (rtems_status_code name)

  *Returns true, if the specified object name is valid, otherwise returns false.*
- static void rtems_name_to_characters (rtems_name name, char *c1, char *c2, char *c3, char *c4)

  *Breaks the object name into the four component characters.*
- bool rtems_workspace_allocate (size_t bytes, void **pointer)

  %
- bool rtems_workspace_free (void *pointer)

  %
- bool rtems_workspace_get_information (Heap_Information_block *the_info)

  %
- void * rtems_workspace_greedy_allocate (const uintptr_t *block_sizes, size_t block_count)

  %
- void * rtems_workspace_greedy_allocate_all_except_largest (uintptr_t *allocatable_size)

  %
- void rtems_workspace_greedy_free (void *opaque)

  %

### 10.121.1 Detailed Description

This header file defines support services of the API.

## 10.122 cpukit/include/rtems/rtems/tasks.h File Reference

This header file defines the main parts of the Tasks Manager API.

```
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <sys/cpuset.h>
#include <rtems/rtems/attr.h>
#include <rtems/rtems/modes.h>
#include <rtems/rtems/status.h>
#include <rtems/rtems/types.h>
#include <rtems/score/basedefs.h>
#include <rtems/score/context.h>
#include <rtems/score/cpu.h>
#include <rtems/score/object.h>
#include <rtems/score/smp.h>
#include <rtems/score/stack.h>
#include <rtems/score/watchdogticks.h>
```

### Classes

- struct rtems_task_config

  *This structure defines the configuration of a task constructed by rtems_task_construct().*
- struct rtems_initialization_tasks_table

  %

## Macros

- #define rtems_scheduler_get_processor() _SMP_Get_current_processor()

  *Returns the index of the current processor.*
- #define rtems_scheduler_get_processor_maximum() _SMP_Get_processor_maximum()

  *Returns the processor maximum supported by the system.*
- #define RTEMS_CURRENT_PRIORITY 0

  *This constant is passed to {set-priority:/name}() when the caller wants to obtain the current priority.*
- #define RTEMS_CONFIGURED_MINIMUM_STACK_SIZE 0

  %
- #define RTEMS_MINIMUM_PRIORITY 1

  %
- #define RTEMS_MINIMUM_STACK_SIZE STACK_MINIMUM_SIZE

  %
- #define RTEMS_NO_PRIORITY RTEMS_CURRENT_PRIORITY

  %
- #define RTEMS_SELF OBJECTS_ID_OF_SELF

  %
- #define RTEMS_TASK_STORAGE_ALIGNMENT CPU_HEAP_ALIGNMENT

  *This constant defines the recommended alignment of a task storage area in bytes.*
- #define RTEMS_TASK_STORAGE_SIZE(_size, _attributes)

  *Returns the recommended task storage area size for the specified size and task attributes.*
- #define RTEMS_YIELD_PROCESSOR WATCHDOG_NO_TIMEOUT

  %
- #define RTEMS_MAXIMUM_PRIORITY _RTEMS_Maximum_priority()

  %

## Typedefs

- typedef CPU_Uint32ptr rtems_task_argument

  *This type is used to represent task argument values.*
- typedef uint32_t rtems_task_priority

  %
- typedef void rtems_task

  %
- typedef rtems_task(∗ rtems_task_entry) (rtems_task_argument)

  *This type defines the entry point of an RTEMS task.*
- typedef struct _Thread_Control rtems_tcb

  %
- typedef bool(∗ rtems_task_visitor) (rtems_tcb ∗, void ∗)

  %

## Functions

- rtems_status_code rtems_scheduler_add_processor (rtems_id scheduler_id, uint32_t cpu_index)

  *Adds the processor to the set of processors owned by the scheduler instance.*
- rtems_status_code rtems_scheduler_get_processor_set (rtems_id scheduler_id, size_t cpusetsize, cpu_↩set_t *cpuset)

  *Gets the set of processors owned by the scheduler instance.*
- rtems_status_code rtems_scheduler_ident (rtems_name name, rtems_id *id)

  *Identifies a scheduler instance by its name.*
- rtems_status_code rtems_scheduler_ident_by_processor (uint32_t cpu_index, rtems_id *id)

  *Identifies a scheduler instance by a processor index.*
- rtems_status_code rtems_scheduler_ident_by_processor_set (size_t cpusetsize, const cpu_set_t *cpuset, rtems_id *id)

  *Identifies a scheduler instance by a processor set.*
- rtems_status_code rtems_scheduler_remove_processor (rtems_id scheduler_id, uint32_t cpu_index)

  *Removes a processor from set of processors owned by the scheduler instance.*
- rtems_status_code rtems_scheduler_get_maximum_priority (rtems_id scheduler_id, rtems_task_priority *priority)

  *Gets the maximum task priority of the scheduler instance.*
- rtems_status_code rtems_scheduler_map_priority_from_posix (rtems_id scheduler_id, int posix_priority, rtems_task_priority *priority)

  *Maps a POSIX thread priority to the corresponding Classic API task priority.*
- rtems_status_code rtems_scheduler_map_priority_to_posix (rtems_id scheduler_id, rtems_task_priority priority, int *posix_priority)

  *Maps a Classic API task priority to the corresponding POSIX thread priority.*
- rtems_status_code rtems_task_construct (const rtems_task_config *config, rtems_id *id)

  *Constructs a task from the specified the task configuration.*
- rtems_status_code rtems_task_create (rtems_name name, rtems_task_priority initial_priority, size_t stack↩_size, rtems_mode initial_modes, rtems_attribute attribute_set, rtems_id *id)

  *Creates a task object.*
- rtems_status_code rtems_task_delete (rtems_id id)

  %
- RTEMS_NO_RETURN void rtems_task_exit (void)

  %
- rtems_status_code rtems_task_get_affinity (rtems_id id, size_t cpusetsize, cpu_set_t *cpuset)

  %
- rtems_status_code rtems_task_get_priority (rtems_id task_id, rtems_id scheduler_id, rtems_task_priority *priority)

  %
- rtems_status_code rtems_task_get_scheduler (rtems_id task_id, rtems_id *scheduler_id)

  %
- rtems_status_code rtems_task_ident (rtems_name name, uint32_t node, rtems_id *id)

  *Identifies a task object by the specified object name.*
- rtems_status_code rtems_task_is_suspended (rtems_id id)

  %
- rtems_status_code rtems_task_mode (rtems_mode mode_set, rtems_mode mask, rtems_mode *previous↩_mode_set)

  %
- rtems_status_code rtems_task_restart (rtems_id id, rtems_task_argument argument)

  %
- rtems_status_code rtems_task_resume (rtems_id id)

  %

- rtems_id rtems_task_self (void)

  %
- rtems_status_code rtems_task_set_affinity (rtems_id id, size_t cpusetsize, const cpu_set_t *cpuset)

  %
- rtems_status_code rtems_task_set_priority (rtems_id id, rtems_task_priority new_priority, rtems_task_priority *old_priority)

  %
- rtems_status_code rtems_task_set_scheduler (rtems_id task_id, rtems_id scheduler_id, rtems_task_priority priority)

  %
- rtems_status_code rtems_task_start (rtems_id id, rtems_task_entry entry_point, rtems_task_argument argument)

  %
- rtems_status_code rtems_task_suspend (rtems_id id)

  %
- void rtems_task_iterate (rtems_task_visitor visitor, void *arg)

  %
- rtems_status_code rtems_task_wake_after (rtems_interval ticks)

  %
- rtems_status_code rtems_task_wake_when (rtems_time_of_day *time_buffer)

  %
- rtems_task_priority _RTEMS_Maximum_priority (void)

  *Returns the maximum priority of the scheduler with index zero.*

### 10.122.1 Detailed Description

This header file defines the main parts of the Tasks Manager API.

## 10.123 cpukit/include/rtems/rtems/tasksdata.h File Reference

Classic Tasks Manager Data Structures.

```
#include <rtems/rtems/tasks.h>
#include <rtems/rtems/asrdata.h>
#include <rtems/rtems/eventdata.h>
#include <rtems/score/thread.h>
```

### Classes

- struct RTEMS_API_Control

### Functions

- void _RTEMS_tasks_Initialize_user_task (void)

  *System initialization handler to create and start the first user task.*

### Variables

- const rtems_initialization_tasks_table _RTEMS_tasks_User_task_table

  *Initialization table for the first user task.*
- Thread_Information _RTEMS_tasks_Information

### 10.123.1 Detailed Description

Classic Tasks Manager Data Structures.

## 10.124 cpukit/include/rtems/rtems/tasksimpl.h File Reference

Classic Tasks Manager Implementation.

```
#include <rtems/rtems/tasksdata.h>
#include <rtems/score/objectimpl.h>
#include <rtems/score/schedulerimpl.h>
#include <rtems/score/threadimpl.h>
```

### Typedefs

- typedef rtems_status_code(∗ **RTEMS_tasks_Prepare_stack**) (Thread_Configuration ∗, const rtems_task_config ∗)

### Functions

- void _RTEMS_tasks_Initialize_user_tasks (void)

  *RTEMS User Task Initialization.*
- rtems_status_code **_RTEMS_tasks_Create** (const rtems_task_config ∗config, rtems_id ∗id, RTEMS_↩ tasks_Prepare_stack prepare_stack)
- static __inline__ Thread_Control ∗ **_RTEMS_tasks_Allocate** (void)
- static __inline__ void _RTEMS_tasks_Free (Thread_Control ∗the_task)

  *Frees a task control block.*
- static __inline__ Priority_Control _RTEMS_Priority_To_core (const Scheduler_Control ∗scheduler, rtems_task_priority priority, bool ∗valid)

  *Converts the RTEMS API priority to the corresponding SuperCore priority and validates it.*
- static __inline__ rtems_task_priority _RTEMS_Priority_From_core (const Scheduler_Control ∗scheduler, Priority_Control priority)

  *Converts the SuperCore priority to the corresponding RTEMS API priority.*

### 10.124.1 Detailed Description

Classic Tasks Manager Implementation.

## 10.125 cpukit/include/rtems/rtems/timer.h File Reference

This header file defines the Timer Manager API.

```
#include <stddef.h>
#include <rtems/rtems/attr.h>
#include <rtems/rtems/status.h>
#include <rtems/rtems/tasks.h>
#include <rtems/rtems/types.h>
#include <rtems/score/watchdogticks.h>
```

### Classes

- struct rtems_timer_information

  %

### Macros

- #define TIMER_CLASS_BIT_NOT_DORMANT 0x4

  %
- #define TIMER_CLASS_BIT_ON_TASK 0x2

  %
- #define TIMER_CLASS_BIT_TIME_OF_DAY 0x1

  %
- #define RTEMS_TIMER_SERVER_DEFAULT_PRIORITY ( (rtems_task_priority) -1 )

  %

### Typedefs

- typedef void rtems_timer_service_routine

  %
- typedef rtems_timer_service_routine(* rtems_timer_service_routine_entry) (rtems_id, void *)

  %

### Enumerations

- enum Timer_Classes {
  TIMER_DORMANT, TIMER_INTERVAL = TIMER_CLASS_BIT_NOT_DORMANT, TIMER_INTERVAL_ON_TASK,
  TIMER_TIME_OF_DAY,
  TIMER_TIME_OF_DAY_ON_TASK }

  %

### Functions

- rtems_status_code rtems_timer_cancel (rtems_id id)

  %
- rtems_status_code rtems_timer_create (rtems_name name, rtems_id *id)

  %
- rtems_status_code rtems_timer_delete (rtems_id id)

  %
- rtems_status_code rtems_timer_ident (rtems_name name, rtems_id *id)

  *Identifies a timer object by the specified object name.*
- rtems_status_code rtems_timer_get_information (rtems_id id, rtems_timer_information *the_info)

  %
- rtems_status_code rtems_timer_initiate_server (rtems_task_priority priority, size_t stack_size, rtems_attribute attribute_set)

  %
- rtems_status_code rtems_timer_reset (rtems_id id)

  %
- rtems_status_code rtems_timer_fire_after (rtems_id id, rtems_interval ticks, rtems_timer_service_routine_entry routine, void *user_data)

  %
- rtems_status_code rtems_timer_fire_when (rtems_id id, rtems_time_of_day *wall_time, rtems_timer_service_routine_entry routine, void *user_data)

  %
- rtems_status_code rtems_timer_server_fire_after (rtems_id id, rtems_interval ticks, rtems_timer_service_routine_entry routine, void *user_data)

  %
- rtems_status_code rtems_timer_server_fire_when (rtems_id id, rtems_time_of_day *wall_time, rtems_timer_service_routine_e routine, void *user_data)

  %

### 10.125.1 Detailed Description

This header file defines the Timer Manager API.

## 10.126 cpukit/include/rtems/rtems/timerdata.h File Reference

Classic Partition Manager Data Structures.

```
#include <rtems/rtems/timer.h>
#include <rtems/score/objectdata.h>
#include <rtems/score/watchdog.h>
```

### Classes

- struct Timer_Control

### Macros

- #define TIMER_INFORMATION_DEFINE(max)

  *Macro to define the objects information for the Classic Timer objects.*

### Variables

- Objects_Information _Timer_Information

  *The Classic Timer objects information.*

#### 10.126.1 Detailed Description

Classic Partition Manager Data Structures.

## 10.127 cpukit/include/rtems/rtems/timerimpl.h File Reference

Classic Timer Implementation.

```
#include <rtems/rtems/timerdata.h>
#include <rtems/score/objectimpl.h>
#include <rtems/score/thread.h>
#include <rtems/score/watchdogimpl.h>
```

### Classes

- struct Timer_server_Control

### Typedefs

- typedef struct Timer_server_Control **Timer_server_Control**

### Functions

- static __inline__ Timer_Control * _Timer_Allocate (void)

  *Timer_Allocate.*
- static __inline__ void _Timer_Free (Timer_Control *the_timer)

  *Timer_Free.*
- static __inline__ Timer_Control * **_Timer_Get** (Objects_Id id, ISR_lock_Context *lock_context)
- static __inline__ Per_CPU_Control * **_Timer_Acquire_critical** (Timer_Control *the_timer, ISR_lock_Context *lock_context)
- static __inline__ void **_Timer_Release** (Per_CPU_Control *cpu, ISR_lock_Context *lock_context)
- static __inline__ bool **_Timer_Is_interval_class** (Timer_Classes the_class)
- static __inline__ bool **_Timer_Is_on_task_class** (Timer_Classes the_class)
- static __inline__ Per_CPU_Watchdog_index **_Timer_Watchdog_header_index** (Timer_Classes the_class)
- static __inline__ Watchdog_Interval **_Timer_Get_CPU_ticks** (const Per_CPU_Control *cpu)

- rtems_status_code **_Timer_Fire** (rtems_id id, rtems_interval interval, rtems_timer_service_routine_entry routine, void *user_data, Timer_Classes the_class, Watchdog_Service_routine_entry adaptor)
- rtems_status_code **_Timer_Fire_after** (rtems_id id, rtems_interval ticks, rtems_timer_service_routine_entry routine, void *user_data, Timer_Classes the_class, Watchdog_Service_routine_entry adaptor)
- rtems_status_code **_Timer_Fire_when** (rtems_id id, const rtems_time_of_day *wall_time, rtems_timer_service_routine_entry routine, void *user_data, Timer_Classes the_class, Watchdog_Service_routine_entry adaptor)
- void **_Timer_Cancel** (Per_CPU_Control *cpu, Timer_Control *the_timer)
- void **_Timer_Routine_adaptor** (Watchdog_Control *the_watchdog)
- void **_Timer_server_Routine_adaptor** (Watchdog_Control *the_watchdog)
- static __inline__ void **_Timer_server_Acquire_critical** (Timer_server_Control *timer_server, ISR_lock_Context *lock_context)
- static __inline__ void **_Timer_server_Release_critical** (Timer_server_Control *timer_server, ISR_lock_Context *lock_context)

### Variables

- Timer_server_Control *volatile _Timer_server

  *Pointer to default timer server control block.*

### 10.127.1 Detailed Description

Classic Timer Implementation.

## 10.128 cpukit/include/rtems/rtems/types.h File Reference

This header file provides types used by the Classic API.

```
#include <stdint.h>
#include <sys/_timespec.h>
#include <sys/_timeval.h>
#include <sys/cpuset.h>
#include <rtems/rtems/modes.h>
#include <rtems/score/mppkt.h>
#include <rtems/score/object.h>
#include <rtems/score/watchdogticks.h>
```

### Classes

- struct rtems_time_of_day

  *This type represents Classic API calendar times.*

### Macros

- #define RTEMS_ID_NONE OBJECTS_ID_NONE

  *This constant represents an invalid RTEMS object identifier.*
- #define RTEMS_NO_TIMEOUT ( (rtems_interval) WATCHDOG_NO_TIMEOUT )

  *This clock tick interval constant indicates that the calling task is willing to wait potentially forever on a resource.*

### Typedefs

- typedef Objects_Id rtems_id

  *This type represents RTEMS object identifiers.*
- typedef Watchdog_Interval rtems_interval

  *This type represents clock tick intervals.*
- typedef uint32_t rtems_name

  *This type represents Classic API object names.*

#### 10.128.1 Detailed Description

This header file provides types used by the Classic API.

## 10.129 cpukit/include/rtems/score/address.h File Reference

Information Required to Manipulate Physical Addresses.

```
#include <rtems/score/cpu.h>
```

### Functions

- RTEMS_INLINE_ROUTINE void * _Addresses_Add_offset (const void *base, uintptr_t offset)

  *Adds offset to an address.*
- RTEMS_INLINE_ROUTINE void * _Addresses_Subtract_offset (const void *base, uintptr_t offset)

  *Subtracts offset from an address.*
- RTEMS_INLINE_ROUTINE intptr_t _Addresses_Subtract (const void *left, const void *right)

  *Subtracts two addresses.*
- RTEMS_INLINE_ROUTINE bool _Addresses_Is_aligned (const void *address)

  *Checks if address is aligned.*
- RTEMS_INLINE_ROUTINE bool _Addresses_Is_in_range (const void *address, const void *base, const void *limit)

  *Checks if address is in range.*
- RTEMS_INLINE_ROUTINE void * _Addresses_Align_up (void *address, size_t alignment)

  *Aligns address to nearest multiple of alignment, rounding up.*
- RTEMS_INLINE_ROUTINE void * _Addresses_Align_down (void *address, size_t alignment)

  *Aligns address to nearest multiple of alignment, truncating.*

#### 10.129.1 Detailed Description

Information Required to Manipulate Physical Addresses.

This include file contains the information required to manipulate physical addresses.

# 10.130 cpukit/include/rtems/score/apimutex.h File Reference

API Mutex Handler API.

```
#include <rtems/score/thread.h>
#include <sys/lock.h>
```

## Classes

- struct API_Mutex_Control

  *Control block used to manage each API mutex.*

## Macros

- #define API_MUTEX_INITIALIZER(name) { _MUTEX_RECURSIVE_NAMED_INITIALIZER( name ), 0 }

  *Statically initialize an API mutex.*

## Functions

- void _API_Mutex_Lock (API_Mutex_Control *mutex)

  *Acquires the specified API mutex.*
- void _API_Mutex_Unlock (API_Mutex_Control *mutex)

  *Releases the specified API mutex.*
- bool _API_Mutex_Is_owner (const API_Mutex_Control *mutex)

  *Checks if the specified API mutex is owned by the executing thread.*
- void **_RTEMS_Lock_allocator** (void)
- void **_RTEMS_Unlock_allocator** (void)
- bool **_RTEMS_Allocator_is_owner** (void)

### 10.130.1 Detailed Description

API Mutex Handler API.

# 10.131 cpukit/include/rtems/score/assert.h File Reference

Information for the Assert Handler.

```
#include <rtems/score/basedefs.h>
```

## Macros

- #define _Assert(_e) ( ( void ) 0 )

  *Assertion similar to assert() controlled via RTEMS_DEBUG instead of NDEBUG.*
- #define _SMP_Assert(_e) _Assert( _e )

  *Like _Assert(), but only armed if RTEMS_SMP is defined.*

### 10.131.1 Detailed Description

Information for the Assert Handler.

## 10.132 cpukit/include/rtems/score/atomic.h File Reference

Atomic Operations API.

```
#include <rtems/score/cpuatomic.h>
```

### Macros

- #define **ATOMIC_ORDER_RELAXED** CPU_ATOMIC_ORDER_RELAXED
- #define **ATOMIC_ORDER_ACQUIRE** CPU_ATOMIC_ORDER_ACQUIRE
- #define **ATOMIC_ORDER_RELEASE** CPU_ATOMIC_ORDER_RELEASE
- #define **ATOMIC_ORDER_ACQ_REL** CPU_ATOMIC_ORDER_ACQ_REL
- #define **ATOMIC_ORDER_SEQ_CST** CPU_ATOMIC_ORDER_SEQ_CST
- #define **ATOMIC_INITIALIZER_UINT**(value) CPU_ATOMIC_INITIALIZER_UINT( value )
- #define **ATOMIC_INITIALIZER_ULONG**(value) CPU_ATOMIC_INITIALIZER_ULONG( value )
- #define **ATOMIC_INITIALIZER_UINTPTR**(value) CPU_ATOMIC_INITIALIZER_UINTPTR( value )
- #define **ATOMIC_INITIALIZER_FLAG** CPU_ATOMIC_INITIALIZER_FLAG
- #define **_Atomic_Fence**(order) _CPU_atomic_Fence( order )
- #define **_Atomic_Init_uint**(obj, desired) _CPU_atomic_Init_uint( obj, desired )
- #define **_Atomic_Init_ulong**(obj, desired) _CPU_atomic_Init_ulong( obj, desired )
- #define **_Atomic_Init_uintptr**(obj, desired) _CPU_atomic_Init_uintptr( obj, desired )
- #define **_Atomic_Load_uint**(obj, order) _CPU_atomic_Load_uint( obj, order )
- #define **_Atomic_Load_ulong**(obj, order) _CPU_atomic_Load_ulong( obj, order )
- #define **_Atomic_Load_uintptr**(obj, order) _CPU_atomic_Load_uintptr( obj, order )
- #define **_Atomic_Store_uint**(obj, desr, order) _CPU_atomic_Store_uint( obj, desr, order )
- #define **_Atomic_Store_ulong**(obj, desr, order) _CPU_atomic_Store_ulong( obj, desr, order )
- #define **_Atomic_Store_uintptr**(obj, desr, order) _CPU_atomic_Store_uintptr( obj, desr, order )
- #define **_Atomic_Fetch_add_uint**(obj, arg, order) _CPU_atomic_Fetch_add_uint( obj, arg, order )
- #define **_Atomic_Fetch_add_ulong**(obj, arg, order) _CPU_atomic_Fetch_add_ulong( obj, arg, order )
- #define **_Atomic_Fetch_add_uintptr**(obj, arg, order) _CPU_atomic_Fetch_add_uintptr( obj, arg, order )
- #define **_Atomic_Fetch_sub_uint**(obj, arg, order) _CPU_atomic_Fetch_sub_uint( obj, arg, order )
- #define **_Atomic_Fetch_sub_ulong**(obj, arg, order) _CPU_atomic_Fetch_sub_ulong( obj, arg, order )
- #define **_Atomic_Fetch_sub_uintptr**(obj, arg, order) _CPU_atomic_Fetch_sub_uintptr( obj, arg, order )
- #define **_Atomic_Fetch_or_uint**(obj, arg, order) _CPU_atomic_Fetch_or_uint( obj, arg, order )
- #define **_Atomic_Fetch_or_ulong**(obj, arg, order) _CPU_atomic_Fetch_or_ulong( obj, arg, order )
- #define **_Atomic_Fetch_or_uintptr**(obj, arg, order) _CPU_atomic_Fetch_or_uintptr( obj, arg, order )
- #define **_Atomic_Fetch_and_uint**(obj, arg, order) _CPU_atomic_Fetch_and_uint( obj, arg, order )
- #define **_Atomic_Fetch_and_ulong**(obj, arg, order) _CPU_atomic_Fetch_and_ulong( obj, arg, order )
- #define **_Atomic_Fetch_and_uintptr**(obj, arg, order) _CPU_atomic_Fetch_and_uintptr( obj, arg, order )
- #define **_Atomic_Exchange_uint**(obj, desr, order) _CPU_atomic_Exchange_uint( obj, desr, order )
- #define **_Atomic_Exchange_ulong**(obj, desr, order) _CPU_atomic_Exchange_ulong( obj, desr, order )
- #define **_Atomic_Exchange_uintptr**(obj, desr, order) _CPU_atomic_Exchange_uintptr( obj, desr, order )
- #define **_Atomic_Compare_exchange_uint**(obj, expected, desired, succ, fail) _CPU_atomic_Compare_exchange_uint( obj, expected, desired, succ, fail )
- #define **_Atomic_Compare_exchange_ulong**(obj, expected, desired, succ, fail) _CPU_atomic_Compare_exchange_ulong( obj, expected, desired, succ, fail )
- #define **_Atomic_Compare_exchange_uintptr**(obj, expected, desired, succ, fail) _CPU_atomic_Compare_exchange_uintptr( obj, expected, desired, succ, fail )
- #define **_Atomic_Flag_clear**(obj, order) _CPU_atomic_Flag_clear( obj, order )
- #define **_Atomic_Flag_test_and_set**(obj, order) _CPU_atomic_Flag_test_and_set( obj, order )

### Typedefs

- typedef CPU_atomic_Uint **Atomic_Uint**
- typedef CPU_atomic_Ulong **Atomic_Ulong**
- typedef CPU_atomic_Uintptr **Atomic_Uintptr**
- typedef CPU_atomic_Flag **Atomic_Flag**
- typedef CPU_atomic_Order **Atomic_Order**

### 10.132.1 Detailed Description

Atomic Operations API.

## 10.133 cpukit/include/rtems/score/basedefs.h File Reference

This header file provides basic definitions used by the API and the implementation.

```
#include <rtems/score/cpuopts.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
```

### Macros

- #define RTEMS_ALIAS(_target) __attribute__((__alias__(#_target)))

  *Instructs the compiler to generate an alias to the target function.*
- #define RTEMS_ALIGN_DOWN(_value, _alignment) ( ( _value ) & ~( ( _alignment ) - 1 ) )

  *Aligns down the value to the alignment.*
- #define RTEMS_ALIGN_UP(_value, _alignment) ( ( ( _value ) + ( _alignment ) - 1 ) & ~( ( _alignment ) - 1 ) )

  *Aligns up the value to the alignment.*
- #define RTEMS_ALIGNED(_alignment) __attribute__((__aligned__(_alignment)))

  *Instructs the compiler in a declaration or definition to enforce the alignment.*
- #define RTEMS_ALLOC_ALIGN(_index) __attribute__((__alloc_align__(_index)))

  *Tells the compiler in a declaration that the memory allocation alignment parameter of this function is similar to aligned_alloc().*
- #define RTEMS_ALLOC_SIZE(_index) __attribute__((__alloc_size__(_index)))

  *Tells the compiler in a declaration that the memory allocation size parameter of this function is similar to malloc().*
- #define RTEMS_ALLOC_SIZE_2(_count_index, _size_index) __attribute__((__alloc_size__(_count_index, _size_index)))

  *Tells the compiler in a declaration that the memory allocation item count and item size parameter of this function is similar to calloc().*
- #define RTEMS_ARRAY_SIZE(_array) ( sizeof( _array ) / sizeof( ( _array )[ 0 ] ) )

  *Gets the element count of the array.*
- #define RTEMS_COMPILER_MEMORY_BARRIER() __asm__ volatile( "" ::: "memory" )

  *This macro forbids the compiler to reorder read and write commands around it.*
- #define RTEMS_CONCAT(_x, _y) _x##_y

  *Concatenates _x and _y without expanding.*
- #define RTEMS_CONST __attribute__((__const__))

*Tells the compiler in a function declaration that this function has no effect except the return value and that the return value depends only on the value of parameters.*

- #define RTEMS_CONTAINER_OF(_m, _type, _member_name) ( (_type *) ( (uintptr_t) ( _m ) - offsetof( _type, _member_name ) ) )

  *Gets the container of a member.*

- #define RTEMS_DECLARE_GLOBAL_SYMBOL(_name) extern char _name[]

  *Declares a global symbol with the name.*

- #define RTEMS_DEPRECATED __attribute__((__deprecated__))

  *Instructs the compiler in a declaration to issue a warning whenever a variable, function, or type using this declaration will be used.*

- #define RTEMS_COMPILER_DEPRECATED_ATTRIBUTE RTEMS_DEPRECATED

  *Provided for backward compatibility.*

- #define RTEMS_EXPAND(_token) _token

  *Helper macro to perform a macro expansion on the token.*

- #define RTEMS_STRING(_x) #_x

  *Stringifies _x without expanding.*

- #define RTEMS_TYPEOF_REFX(_level, _target) __typeof__( _level( union { int z; __typeof__( _target ) x; } ){ 0 }.x )

  *Gets the pointer reference type.*

- #define RTEMS_XCONCAT(_x, _y) RTEMS_CONCAT( _x, _y )

  *Concatenates expansion of _x and expansion of _y.*

- #define _Assert_Unreachable() do { } while ( 0 )

  *Asserts that this program point is unreachable.*

- #define RTEMS_DEQUALIFY_DEPTHX(_ptr_level, _type, _var)

  *Performs a type cast which removes qualifiers without warnings to the type for the variable.*

- #define RTEMS_DECONST(_type, _var) RTEMS_DEQUALIFY_DEPTHX(*, _type, _var)

  *Performs a type cast which removes const qualifiers without warnings to the type for the pointer variable.*

- #define RTEMS_DEQUALIFY(_type, _var) RTEMS_DEQUALIFY_DEPTHX(*, _type, _var)

  *Performs a type cast which removes all qualifiers without warnings to the type for the pointer variable.*

- #define RTEMS_DEVOLATILE(_type, _var) RTEMS_DEQUALIFY_DEPTHX(*, _type, _var)

  *Performs a type cast which removes volatile qualifiers without warnings to the type for the pointer variable.*

- #define FALSE 0

  *If FALSE is undefined, then FALSE is defined to 0.*

- #define RTEMS_HAVE_MEMBER_SAME_TYPE(_t_lhs, _m_lhs, _t_rhs, _m_rhs)

  *Checks if members of two types have compatible types.*

- #define RTEMS_INLINE_ROUTINE static __inline__

  *Gives a hint to the compiler in a function declaration to inline this function.*

- #define RTEMS_MALLOCLIKE __attribute__((__malloc__))

  *Tells the compiler in a declaration that this function is a memory allocation function similar to malloc().*

- #define RTEMS_NO_INLINE __attribute__((__noinline__))

  *Instructs the compiler in a function declaration to not inline this function.*

- #define RTEMS_NO_RETURN _Noreturn

  *Tells the compiler in a function declaration that this function does not return.*

- #define RTEMS_COMPILER_NO_RETURN_ATTRIBUTE RTEMS_NO_RETURN

  *Provided for backward compatibility.*

- #define RTEMS_OBFUSCATE_VARIABLE(_var) __asm__( "" : "+r" ( _var ) )

  *Obfuscates the variable so that the compiler cannot perform optimizations based on the variable value.*

- #define RTEMS_PACKED __attribute__((__packed__))

  *Instructs the compiler in a type definition to place members of a structure or union so that the memory required is minimized.*

- #define RTEMS_COMPILER_PACKED_ATTRIBUTE RTEMS_PACKED

*Provided for backward compatibility.*

- #define RTEMS_PREDICT_FALSE(_exp) __builtin_expect( ( _exp ), 0 )

  *Evaluates the integral expression and tells the compiler that the predicted value is false (0).*
- #define RTEMS_PREDICT_TRUE(_exp) __builtin_expect( ( _exp ), 0 )

  *Evaluates the integral expression and tells the compiler that the predicted value is true (1).*
- #define RTEMS_PRINTFLIKE(_format_pos, _ap_pos) __attribute__((__format__(__printf__, _format_pos, _ap_pos)))

  *Tells the compiler in a declaration that this function expects printf()-like arguments.*
- #define RTEMS_PURE __attribute__((__pure__))

  *Tells the compiler in a function declaration that this function has no effect except the return value and that the return value depends only on the value of parameters and/or global variables.*
- #define RTEMS_COMPILER_PURE_ATTRIBUTE RTEMS_PURE

  *Provided for backward compatibility.*
- #define RTEMS_RETURN_ADDRESS() __builtin_return_address(0)

  *Gets the return address of the current function.*
- #define RTEMS_SECTION(_section) __attribute__((__section__(_section)))

  *Instructs the compiler to place the variable or function in the section.*
- #define RTEMS_STATIC_ASSERT(_cond, _msg) _Static_assert(_cond, # _msg)

  *Asserts at compile time that the condition is satisfied.*
- #define RTEMS_SYMBOL_NAME(_name) RTEMS_EXPAND(_name)

  *Maps the name to the associated symbol name.*
- #define TRUE 1

  *If TRUE is undefined, then TRUE is defined to 1.*
- #define RTEMS_UNREACHABLE()

  *Tells the compiler that this program point is unreachable.*
- #define RTEMS_UNUSED __attribute__((__unused__))

  *Tells the compiler that the variable or function is deliberately unused.*
- #define RTEMS_USED __attribute__((__used__))

  *Tells the compiler that the variable or function is used.*
- #define RTEMS_COMPILER_USED_ATTRIBUTE RTEMS_USED

  *Provided for backward compatibility.*
- #define RTEMS_WARN_UNUSED_RESULT __attribute__((__warn_unused_result__))

  *Tells the compiler in a declaration that the result of this function should be used.*
- #define RTEMS_WEAK __attribute__((__weak__))

  *Tells the compiler in a function definition that this function should be weak.*
- #define RTEMS_WEAK_ALIAS(_target) __attribute__((__weak__, __alias__(#_target)))

  *Instructs the compiler to generate a weak alias to the target function.*
- #define RTEMS_XSTRING(_x) RTEMS_STRING( _x )

  *Stringifies the expansion of _x.*
- #define RTEMS_DEFINE_GLOBAL_SYMBOL(_name, _value)

  *Defines a global symbol with the name and value.*
- #define RTEMS_ZERO_LENGTH_ARRAY

  *This constant represents the element count of a zero-length array.*

## Functions

- void * RTEMS_DEQUALIFY_types_not_compatible (void)

  *A not implemented function to trigger compile time errors with an error message.*

### 10.133.1 Detailed Description

This header file provides basic definitions used by the API and the implementation.

### 10.133.2 Macro Definition Documentation

#### 10.133.2.1 RTEMS_UNREACHABLE

```
#define RTEMS_UNREACHABLE( )
```

**Value:**

```
do { \
  __builtin_unreachable(); \
  _Assert_Unreachable(); \
} while ( 0 )
```

Tells the compiler that this program point is unreachable.

Definition at line 804 of file basedefs.h.

## 10.134 cpukit/include/rtems/score/chainimpl.h File Reference

Chain Handler API.

```
#include <rtems/score/chain.h>
#include <rtems/score/assert.h>
```

### Classes

- struct Chain_Iterator

  *A chain iterator which is updated during node extraction if it is properly registered.*
- struct Chain_Iterator_registry

  *A registry for chain iterators.*

### Macros

- #define CHAIN_INITIALIZER_EMPTY(name) { { { &(name).Tail.Node, NULL }, &(name).Head.Node } }

  *Chain initializer for an empty chain with designator name.*
- #define CHAIN_INITIALIZER_ONE_NODE(node) { { { (node), NULL }, (node) } }

  *Chain initializer for a chain with one node.*
- #define CHAIN_NODE_INITIALIZER_ONE_NODE_CHAIN(chain) { &(chain)->Tail.Node, &(chain)->Head.Node }

  *Chain node initializer for a chain containing exactly this node.*
- #define CHAIN_DEFINE_EMPTY(name) Chain_Control name = CHAIN_INITIALIZER_EMPTY(name)

  *Chain definition for an empty chain with designator name.*
- #define CHAIN_ITERATOR_REGISTRY_INITIALIZER(name) { CHAIN_INITIALIZER_EMPTY( name.↩ Iterators ) }

  *Chain iterator registry initializer for static initialization.*

## Typedefs

- typedef bool(* Chain_Node_order) (const void *left, const Chain_Node *right)

  *Chain node order.*

## Enumerations

- enum Chain_Iterator_direction { CHAIN_ITERATOR_FORWARD, CHAIN_ITERATOR_BACKWARD }

  *The chain iterator direction.*

## Functions

- void _Chain_Initialize (Chain_Control *the_chain, void *starting_address, size_t number_nodes, size_t node_size)

  *Initializes a chain header.*
- size_t _Chain_Node_count_unprotected (const Chain_Control *chain)

  *Returns the node count of the chain.*
- static __inline__ void _Chain_Set_off_chain (Chain_Node *node)

  *Sets off chain.*
- static __inline__ void _Chain_Initialize_node (Chain_Node *the_node)

  *Initializes a chain node.*
- static __inline__ bool _Chain_Is_node_off_chain (const Chain_Node *node)

  *Checks if the node is off chain.*
- static __inline__ bool _Chain_Are_nodes_equal (const Chain_Node *left, const Chain_Node *right)

  *Checks if two nodes are equal.*
- static __inline__ Chain_Node * _Chain_Head (Chain_Control *the_chain)

  *Returns pointer to chain head.*
- static __inline__ const Chain_Node * _Chain_Immutable_head (const Chain_Control *the_chain)

  *Returns pointer to immutable chain head.*
- static __inline__ Chain_Node * _Chain_Tail (Chain_Control *the_chain)

  *Returns pointer to chain tail.*
- static __inline__ const Chain_Node * _Chain_Immutable_tail (const Chain_Control *the_chain)

  *Returns pointer to immutable chain tail.*
- static __inline__ Chain_Node * _Chain_First (const Chain_Control *the_chain)

  *Returns pointer to chain's first node.*
- static __inline__ const Chain_Node * _Chain_Immutable_first (const Chain_Control *the_chain)

  *Returns pointer to immutable chain's first node.*
- static __inline__ Chain_Node * _Chain_Last (const Chain_Control *the_chain)

  *Returns pointer to chain's last node.*
- static __inline__ const Chain_Node * _Chain_Immutable_last (const Chain_Control *the_chain)

  *Returns pointer to immutable chain's last node.*
- static __inline__ Chain_Node * _Chain_Next (const Chain_Node *the_node)

  *Returns pointer to the next node from this node.*
- static __inline__ const Chain_Node * _Chain_Immutable_next (const Chain_Node *the_node)

  *Returns pointer to the immutable next node from this node.*
- static __inline__ Chain_Node * _Chain_Previous (const Chain_Node *the_node)

  *Returns pointer to the previous node from this node.*
- static __inline__ const Chain_Node * _Chain_Immutable_previous (const Chain_Node *the_node)

  *Returns pointer to the immutable previous node from this node.*
- static __inline__ bool _Chain_Is_empty (const Chain_Control *the_chain)

*Checks if the chain is empty.*

- static __inline__ bool _Chain_Is_first (const Chain_Node *the_node)

  *Checks if this is the first node on the chain.*
- static __inline__ bool _Chain_Is_last (const Chain_Node *the_node)

  *Checks if this is the last node on the chain.*
- static __inline__ bool _Chain_Has_only_one_node (const Chain_Control *the_chain)

  *Checks if this chain has only one node.*
- static __inline__ bool _Chain_Is_head (const Chain_Control *the_chain, const Chain_Node *the_node)

  *Checks if this node is the chain head.*
- static __inline__ bool _Chain_Is_tail (const Chain_Control *the_chain, const Chain_Node *the_node)

  *Checks if this node is the chain tail.*
- static __inline__ void _Chain_Initialize_empty (Chain_Control *the_chain)

  *Initializes this chain as empty.*
- static __inline__ void _Chain_Initialize_one (Chain_Control *the_chain, Chain_Node *the_node)

  *Initializes this chain to contain exactly the specified node.*
- static __inline__ void _Chain_Extract_unprotected (Chain_Node *the_node)

  *Extracts this node (unprotected).*
- static __inline__ Chain_Node * _Chain_Get_first_unprotected (Chain_Control *the_chain)

  *Gets the first node (unprotected).*
- static __inline__ Chain_Node * _Chain_Get_unprotected (Chain_Control *the_chain)

  *Gets the first node (unprotected).*
- static __inline__ void _Chain_Insert_unprotected (Chain_Node *after_node, Chain_Node *the_node)

  *Inserts a node (unprotected).*
- static __inline__ void _Chain_Append_unprotected (Chain_Control *the_chain, Chain_Node *the_node)

  *Appends a node (unprotected).*
- static __inline__ void _Chain_Append_if_is_off_chain_unprotected (Chain_Control *the_chain, Chain_Node *the_node)

  *Appends a node on the end of a chain if the node is in the off chain state (unprotected).*
- static __inline__ void _Chain_Prepend_unprotected (Chain_Control *the_chain, Chain_Node *the_node)

  *Prepends a node (unprotected).*
- static __inline__ bool _Chain_Append_with_empty_check_unprotected (Chain_Control *the_chain, Chain_Node *the_node)

  *Appends a node and checks if the chain was empty before (unprotected).*
- static __inline__ bool _Chain_Prepend_with_empty_check_unprotected (Chain_Control *the_chain, Chain_Node *the_node)

  *Prepends a node and checks if the chain was empty before (unprotected).*
- static __inline__ bool _Chain_Get_with_empty_check_unprotected (Chain_Control *the_chain, Chain_Node **the_node)

  *Gets the first node and checks if the chain is empty afterwards (unprotected).*
- static __inline__ void _Chain_Insert_ordered_unprotected (Chain_Control *the_chain, Chain_Node *to_insert, const void *left, Chain_Node_order order)

  *Inserts a node into the chain according to the order relation.*
- static __inline__ void _Chain_Iterator_registry_initialize (Chain_Iterator_registry *the_registry)

  *Initializes a chain iterator registry.*
- static __inline__ void _Chain_Iterator_registry_update (Chain_Iterator_registry *the_registry, Chain_Node *the_node_to_extract)

  *Updates all iterators present in the chain iterator registry in case of a node extraction.*
- static __inline__ void _Chain_Iterator_initialize (Chain_Control *the_chain, Chain_Iterator_registry *the_registry, Chain_Iterator *the_iterator, Chain_Iterator_direction direction)

  *Initializes the chain iterator.*
- static __inline__ Chain_Node * _Chain_Iterator_next (const Chain_Iterator *the_iterator)

  *Returns the next node in the iterator direction.*

- static __inline__ void _Chain_Iterator_set_position (Chain_Iterator *the_iterator, Chain_Node *the_node)
  *Sets the iterator position.*
- static __inline__ void _Chain_Iterator_destroy (Chain_Iterator *the_iterator)
  *Destroys the iterator.*

### 10.134.1 Detailed Description

Chain Handler API.

## 10.135 cpukit/include/rtems/score/context.h File Reference

Information About Each Thread's Context.

```
#include <rtems/score/cpu.h>
```

### Macros

- #define CONTEXT_FP_SIZE
  *Size of floating point context area.*
- #define _Context_Initialize(_the_context, _stack, _size, _isr, _entry, _is_fp, _tls_area)
  *Initialize context area.*
- #define _Context_Initialization_at_thread_begin()
- #define _Context_Switch(_executing, _heir) _CPU_Context_switch( _executing, _heir )
  *Perform context switch.*
- #define _Context_Restart_self(_the_context) _CPU_Context_Restart_self( _the_context )
  *Restart currently executing thread.*
- #define _Context_Initialize_fp(_fp_area) _CPU_Context_Initialize_fp( _fp_area )
  *Initialize floating point context area.*
- #define _Context_Restore_fp(_fp) _CPU_Context_restore_fp( _fp )
  *Restore floating point context area.*
- #define _Context_Save_fp(_fp) _CPU_Context_save_fp( _fp )
  *Save floating point context area.*
- #define **_Context_Destroy**(_the_thread, _the_context)

### 10.135.1 Detailed Description

Information About Each Thread's Context.

This include file contains all information about each thread's context.

## 10.136 cpukit/include/rtems/score/copyrt.h File Reference

Copyright Notice for RTEMS.

### Variables

- const char _Copyright_Notice [ ]
- const char _RTEMS_version [ ]

  *This constant provides the RTEMS version string.*

#### 10.136.1 Detailed Description

Copyright Notice for RTEMS.

This include file contains the copyright notice for RTEMS which is included in every binary copy of the executive.

## 10.137 cpukit/include/rtems/score/corebarrier.h File Reference

Constants and Structures Associated with the Barrier Handler.

```
#include <rtems/score/threadq.h>
```

### Classes

- struct CORE_barrier_Attributes
- struct CORE_barrier_Control

### Enumerations

- enum CORE_barrier_Disciplines { CORE_BARRIER_AUTOMATIC_RELEASE, CORE_BARRIER_MANUAL_RELEASE }

#### 10.137.1 Detailed Description

Constants and Structures Associated with the Barrier Handler.

This include file contains all the constants and structures associated with the Barrier Handler.

## 10.138 cpukit/include/rtems/score/corebarrierimpl.h File Reference

Inlined Routines Associated with the SuperCore Barrier.

```
#include <rtems/score/corebarrier.h>
#include <rtems/score/status.h>
#include <rtems/score/threadqimpl.h>
```

### Macros

- #define **CORE_BARRIER_TQ_OPERATIONS** &_Thread_queue_Operations_FIFO

### Functions

- void _CORE_barrier_Initialize (CORE_barrier_Control *the_barrier, CORE_barrier_Attributes *the_barrier↩ _attributes)

  *Initializes the core barrier.*
- static __inline__ void _CORE_barrier_Destroy (CORE_barrier_Control *the_barrier)

  *Destroys the core barrier.*
- static __inline__ void _CORE_barrier_Acquire_critical (CORE_barrier_Control *the_barrier, Thread_queue_Context *queue_context)

  *Acquires critical core barrier.*
- static __inline__ void _CORE_barrier_Release (CORE_barrier_Control *the_barrier, Thread_queue_Context *queue_context)

  *Releases core barrier.*
- Status_Control _CORE_barrier_Seize (CORE_barrier_Control *the_barrier, Thread_Control *executing, bool wait, Thread_queue_Context *queue_context)

  *Waits for the barrier.*
- uint32_t _CORE_barrier_Do_flush (CORE_barrier_Control *the_barrier, Thread_queue_Flush_filter filter, Thread_queue_Context *queue_context)

  *Flushes the barrier.*
- static __inline__ uint32_t _CORE_barrier_Surrender (CORE_barrier_Control *the_barrier, Thread_queue_Context *queue_context)

  *Manually releases the barrier.*
- static __inline__ void _CORE_barrier_Flush (CORE_barrier_Control *the_barrier, Thread_queue_Context *queue_context)

  *Flushes the barrier using _CORE_barrier_Do_flush().*
- static __inline__ bool _CORE_barrier_Is_automatic (CORE_barrier_Attributes *the_attribute)

  *Checks if the barrier is automatic.*
- static __inline__ uint32_t _CORE_barrier_Get_number_of_waiting_threads (CORE_barrier_Control *the_↩ barrier)

  *Returns the number of currently waiting threads.*

### 10.138.1 Detailed Description

Inlined Routines Associated with the SuperCore Barrier.

This include file contains all of the inlined routines associated with the SuperCore barrier.

## 10.139 cpukit/include/rtems/score/coremsg.h File Reference

Constants and Structures Associated with the Message Queue Handler.

```
#include <rtems/score/coremsgbuffer.h>
#include <rtems/score/isrlock.h>
#include <rtems/score/threadq.h>
#include <rtems/score/watchdog.h>
```

### Classes

- struct CORE_message_queue_Control

  *Control block used to manage each message queue.*

### Macros

- #define RTEMS_SCORE_COREMSG_ENABLE_BLOCKING_SEND

### Typedefs

- typedef struct CORE_message_queue_Control **CORE_message_queue_Control**

### Enumerations

- enum CORE_message_queue_Disciplines { CORE_MESSAGE_QUEUE_DISCIPLINES_FIFO, CORE_MESSAGE_QUEUE_D
  }

  *The possible blocking disciplines for a message queue.*

### 10.139.1 Detailed Description

Constants and Structures Associated with the Message Queue Handler.

This include file contains all the constants and structures associated with the Message queue Handler.

## 10.140 cpukit/include/rtems/score/coremsgbuffer.h File Reference

This header file defines the buffer data structure used by the Message Queue Handler.

```
#include <rtems/score/basedefs.h>
#include <rtems/score/chain.h>
```

### Classes

- struct CORE_message_queue_Buffer

  *The structure is used to organize message buffers of a message queue.*

### Macros

- #define RTEMS_SCORE_COREMSG_ENABLE_MESSAGE_PRIORITY

### 10.140.1 Detailed Description

This header file defines the buffer data structure used by the Message Queue Handler.

# 10.141 cpukit/include/rtems/score/coremsgimpl.h File Reference

Inlined Routines in the Core Message Handler.

```
#include <rtems/score/coremsg.h>
#include <rtems/score/status.h>
#include <rtems/score/chainimpl.h>
#include <rtems/score/threaddispatch.h>
#include <rtems/score/threadqimpl.h>
#include <limits.h>
#include <string.h>
```

## Macros

- #define CORE_MESSAGE_QUEUE_SEND_REQUEST INT_MAX

  *Used when appending messages onto a message queue.*
- #define CORE_MESSAGE_QUEUE_URGENT_REQUEST INT_MIN

  *Used when prepending messages onto a message queue.*
- #define _CORE_message_queue_Set_notify(the_message_queue, the_handler) do { } while ( 0 )

  *Initializes notification information.*

## Typedefs

- typedef int CORE_message_queue_Submit_types

  *The modes in which a message may be submitted to a message queue.*
- typedef void *(* CORE_message_queue_Allocate_buffers) (CORE_message_queue_Control *the_↩ message_queue, size_t size, const void *arg)

  *This handler shall allocate the message buffer storage area for a message queue.*

## Functions

- void * _CORE_message_queue_Workspace_allocate (CORE_message_queue_Control *the_message_↩ queue, size_t size, const void *arg)

  *This handler allocates the message buffer storage area for a message queue from the RTEMS Workspace.*
- Status_Control _CORE_message_queue_Initialize (CORE_message_queue_Control *the_message_↩ queue, CORE_message_queue_Disciplines discipline, uint32_t maximum_pending_messages, size_↩ t maximum_message_size, CORE_message_queue_Allocate_buffers allocate_buffers, const void *arg)

  *Initializes a message queue.*
- void _CORE_message_queue_Close (CORE_message_queue_Control *the_message_queue, Thread_queue_Context *queue_context)

  *Closes a message queue.*
- uint32_t _CORE_message_queue_Flush (CORE_message_queue_Control *the_message_queue, Thread_queue_Context *queue_context)

  *Flushes pending messages.*
- Status_Control _CORE_message_queue_Broadcast (CORE_message_queue_Control *the_message_↩ queue, const void *buffer, size_t size, uint32_t *count, Thread_queue_Context *queue_context)

  *Broadcasts a message to the message queue.*
- Status_Control _CORE_message_queue_Submit (CORE_message_queue_Control *the_message_queue, Thread_Control *executing, const void *buffer, size_t size, CORE_message_queue_Submit_types submit↩ _type, bool wait, Thread_queue_Context *queue_context)

*Submits a message to the message queue.*

- Status_Control _CORE_message_queue_Seize (CORE_message_queue_Control *the_message_queue, Thread_Control *executing, void *buffer, size_t *size_p, bool wait, Thread_queue_Context *queue_context)

  *Seizes a message from the message queue.*

- void _CORE_message_queue_Insert_message (CORE_message_queue_Control *the_message_queue, CORE_message_queue_Buffer *the_message, const void *content_source, size_t content_size, CORE_message_queue_Submit_types submit_type)

  *Inserts a message into the message queue.*

- static __inline__ Status_Control _CORE_message_queue_Send (CORE_message_queue_Control *the_message_queue, const void *buffer, size_t size, bool wait, Thread_queue_Context *queue_context)

  *Sends a message to the message queue.*

- static __inline__ Status_Control _CORE_message_queue_Urgent (CORE_message_queue_Control *the_message_queue, const void *buffer, size_t size, bool wait, Thread_queue_Context *queue_context)

  *Sends an urgent message to the message queue.*

- static __inline__ void _CORE_message_queue_Acquire (CORE_message_queue_Control *the_message_queue, Thread_queue_Context *queue_context)

  *Acquires the message queue.*

- static __inline__ void _CORE_message_queue_Acquire_critical (CORE_message_queue_Control *the_message_queue, Thread_queue_Context *queue_context)

  *Acquires the message queue critical.*

- static __inline__ void _CORE_message_queue_Release (CORE_message_queue_Control *the_message_queue, Thread_queue_Context *queue_context)

  *Releases the message queue.*

- static __inline__ void _CORE_message_queue_Copy_buffer (const void *source, void *destination, size_t size)

  *Copies the source message buffer to the destination message buffer.*

- static __inline__ CORE_message_queue_Buffer * _CORE_message_queue_Allocate_message_buffer (CORE_message_queue_Control *the_message_queue)

  *Allocates a message buffer from the inactive message buffer chain.*

- static __inline__ void _CORE_message_queue_Free_message_buffer (CORE_message_queue_Control *the_message_queue, CORE_message_queue_Buffer *the_message)

  *Frees a message buffer to inactive message buffer chain.*

- static __inline__ int _CORE_message_queue_Get_message_priority (const CORE_message_queue_Buffer *the_message)

  *Gets message priority.*

- static __inline__ CORE_message_queue_Buffer * _CORE_message_queue_Get_pending_message (CORE_message_queue_Control *the_message_queue)

  *Gets first message of message queue and removes it.*

- static __inline__ Thread_Control * _CORE_message_queue_Dequeue_receiver (CORE_message_queue_Control *the_message_queue, const void *buffer, size_t size, CORE_message_queue_Submit_types submit_type, Thread_queue_Context *queue_context)

  *Gets the first locked thread waiting to receive and dequeues it.*

### 10.141.1 Detailed Description

Inlined Routines in the Core Message Handler.

This include file contains the static inline implementation of all inlined routines in the Core Message Handler.

## 10.142 cpukit/include/rtems/score/coremutex.h File Reference

CORE Mutex API.

```
#include <rtems/score/thread.h>
#include <rtems/score/threadq.h>
#include <rtems/score/priority.h>
#include <rtems/score/watchdog.h>
#include <rtems/score/interr.h>
```

### Classes

- struct CORE_mutex_Control

  *Control block used to manage each mutex.*
- struct CORE_recursive_mutex_Control

  *The recursive mutex control.*
- struct CORE_ceiling_mutex_Control

  *The recursive mutex control with priority ceiling protocol support.*

### 10.142.1 Detailed Description

CORE Mutex API.

This include file contains all the constants and structures associated with the Mutex Handler. A mutex is an enhanced version of the standard Dijkstra binary semaphore used to provide synchronization and mutual exclusion capabilities.

## 10.143 cpukit/include/rtems/score/coremuteximpl.h File Reference

CORE Mutex Implementation.

```
#include <rtems/score/coremutex.h>
#include <rtems/score/chainimpl.h>
#include <rtems/score/schedulerimpl.h>
#include <rtems/score/status.h>
#include <rtems/score/threadimpl.h>
#include <rtems/score/threadqimpl.h>
```

### Macros

- #define **CORE_MUTEX_TQ_OPERATIONS** &_Thread_queue_Operations_priority
- #define **CORE_MUTEX_TQ_PRIORITY_INHERIT_OPERATIONS** &_Thread_queue_Operations_priority↩ _inherit

## Functions

- static __inline__ void _CORE_mutex_Initialize (CORE_mutex_Control *the_mutex)

  *Initializes the mutex.*
- static __inline__ void _CORE_mutex_Destroy (CORE_mutex_Control *the_mutex)

  *Destroys the mutex.*
- static __inline__ void _CORE_mutex_Acquire_critical (CORE_mutex_Control *the_mutex, Thread_queue_Context *queue_context)

  *Acquires the mutex critical.*
- static __inline__ void _CORE_mutex_Release (CORE_mutex_Control *the_mutex, Thread_queue_Context *queue_context)

  *Releases the mutex.*
- static __inline__ Thread_Control * _CORE_mutex_Get_owner (const CORE_mutex_Control *the_mutex)

  *Gets the owner of the mutex.*
- static __inline__ bool _CORE_mutex_Is_locked (const CORE_mutex_Control *the_mutex)

  *Checks if the mutex is locked.*
- Status_Control _CORE_mutex_Seize_slow (CORE_mutex_Control *the_mutex, const Thread_queue_Operations *operations, Thread_Control *executing, bool wait, Thread_queue_Context *queue_context)

  *Seize the mutex slowly.*
- static __inline__ void _CORE_mutex_Set_owner (CORE_mutex_Control *the_mutex, Thread_Control *owner)

  *Sets the owner of the mutex.*
- static __inline__ bool _CORE_mutex_Is_owner (const CORE_mutex_Control *the_mutex, const Thread_Control *the_thread)

  *Checks if the the thread is the owner of the mutex.*
- static __inline__ void _CORE_recursive_mutex_Initialize (CORE_recursive_mutex_Control *the_mutex)

  *Initializes a recursive mutex.*
- static __inline__ Status_Control _CORE_recursive_mutex_Seize_nested (CORE_recursive_mutex_Control *the_mutex)

  *Seizes the recursive mutex nested.*
- static __inline__ Status_Control _CORE_recursive_mutex_Seize (CORE_recursive_mutex_Control *the_mutex, const Thread_queue_Operations *operations, Thread_Control *executing, bool wait, Status_Control(*nested)(CORE_recursive_mutex_Control *), Thread_queue_Context *queue_context)

  *Seizes the recursive mutex.*
- static __inline__ Status_Control _CORE_recursive_mutex_Surrender (CORE_recursive_mutex_Control *the_mutex, const Thread_queue_Operations *operations, Thread_Control *executing, Thread_queue_Context *queue_context)

  *Surrenders the recursive mutex.*
- static __inline__ void _CORE_ceiling_mutex_Initialize (CORE_ceiling_mutex_Control *the_mutex, const Scheduler_Control *scheduler, Priority_Control priority_ceiling)

  *initializes a ceiling mutex.*
- static __inline__ const Scheduler_Control * _CORE_ceiling_mutex_Get_scheduler (const CORE_ceiling_mutex_Control *the_mutex)

  *Gets the scheduler of the ceiling mutex.*
- static __inline__ void _CORE_ceiling_mutex_Set_priority (CORE_ceiling_mutex_Control *the_mutex, Priority_Control priority_ceiling, Thread_queue_Context *queue_context)

  *Sets the priority of the ceiling mutex.*
- static __inline__ Priority_Control _CORE_ceiling_mutex_Get_priority (const CORE_ceiling_mutex_Control *the_mutex)

  *Gets the priority of the ceiling mutex.*
- static __inline__ Status_Control _CORE_ceiling_mutex_Set_owner (CORE_ceiling_mutex_Control *the_mutex, Thread_Control *owner, Thread_queue_Context *queue_context)

  *Sets the owner of the ceiling mutex.*

- static __inline__ Status_Control _CORE_ceiling_mutex_Seize (CORE_ceiling_mutex_Control *the_↩ mutex, Thread_Control *executing, bool wait, Status_Control(*nested)(CORE_recursive_mutex_Control *), Thread_queue_Context *queue_context)

  *Seizes the ceiling mutex.*
- static __inline__ Status_Control _CORE_ceiling_mutex_Surrender (CORE_ceiling_mutex_Control *the_↩ mutex, Thread_Control *executing, Thread_queue_Context *queue_context)

  *Surrenders the ceiling mutex.*

### 10.143.1 Detailed Description

CORE Mutex Implementation.

## 10.144 cpukit/include/rtems/score/coresem.h File Reference

Data Associated with the Counting Semaphore Handler.

```
#include <rtems/score/threadq.h>
```

### Classes

- struct CORE_semaphore_Control

### 10.144.1 Detailed Description

Data Associated with the Counting Semaphore Handler.

This include file contains all the constants and structures associated with the Counting Semaphore Handler. A counting semaphore is the standard Dijkstra binary semaphore used to provide synchronization and mutual exclusion capabilities.

## 10.145 cpukit/include/rtems/score/coresemimpl.h File Reference

Inlined Routines Associated with the SuperCore Semaphore.

```
#include <rtems/score/coresem.h>
#include <rtems/score/objectimpl.h>
#include <rtems/score/threaddispatch.h>
#include <rtems/score/threadimpl.h>
#include <rtems/score/threadqimpl.h>
#include <rtems/score/statesimpl.h>
#include <rtems/score/status.h>
```

### Functions

- void _CORE_semaphore_Initialize (CORE_semaphore_Control *the_semaphore, uint32_t initial_value)

  *Initializes the semaphore based on the parameters passed.*
- static __inline__ void _CORE_semaphore_Acquire_critical (CORE_semaphore_Control *the_semaphore, Thread_queue_Context *queue_context)

  *Acquires the semaphore critical.*
- static __inline__ void _CORE_semaphore_Release (CORE_semaphore_Control *the_semaphore, Thread_queue_Context *queue_context)

  *Releases the semaphore.*
- static __inline__ void _CORE_semaphore_Destroy (CORE_semaphore_Control *the_semaphore, const Thread_queue_Operations *operations, Thread_queue_Context *queue_context)

  *Destroys the semaphore.*
- static __inline__ Status_Control _CORE_semaphore_Surrender (CORE_semaphore_Control *the_↩ semaphore, const Thread_queue_Operations *operations, uint32_t maximum_count, Thread_queue_Context *queue_context)

  *Surrenders a unit to the semaphore.*
- static __inline__ uint32_t _CORE_semaphore_Get_count (const CORE_semaphore_Control *the_↩ semaphore)

  *Returns the current count associated with the semaphore.*
- static __inline__ Status_Control _CORE_semaphore_Seize (CORE_semaphore_Control *the_semaphore, const Thread_queue_Operations *operations, Thread_Control *executing, bool wait, Thread_queue_Context *queue_context)

  *Seizes the semaphore.*

#### 10.145.1 Detailed Description

Inlined Routines Associated with the SuperCore Semaphore.

This include file contains all of the inlined routines associated with the SuperCore semaphore.

## 10.146 cpukit/include/rtems/score/cpustdatomic.h File Reference

Atomic Operations CPU API.

```
#include <rtems/score/basedefs.h>
#include <stdatomic.h>
```

### Macros

- #define **_RTEMS_SCORE_CPUSTDATOMIC_USE_STDATOMIC**
- #define **CPU_ATOMIC_ORDER_RELAXED** memory_order_relaxed
- #define **CPU_ATOMIC_ORDER_ACQUIRE** memory_order_acquire
- #define **CPU_ATOMIC_ORDER_RELEASE** memory_order_release
- #define **CPU_ATOMIC_ORDER_ACQ_REL** memory_order_acq_rel
- #define **CPU_ATOMIC_ORDER_SEQ_CST** memory_order_seq_cst
- #define **CPU_ATOMIC_INITIALIZER_UINT**(value) ATOMIC_VAR_INIT( value )
- #define **CPU_ATOMIC_INITIALIZER_ULONG**(value) ATOMIC_VAR_INIT( value )
- #define **CPU_ATOMIC_INITIALIZER_UINTPTR**(value) ATOMIC_VAR_INIT( value )
- #define **CPU_ATOMIC_INITIALIZER_FLAG** ATOMIC_FLAG_INIT

## Typedefs

- typedef atomic_uint **CPU_atomic_Uint**
- typedef atomic_ulong **CPU_atomic_Ulong**
- typedef atomic_uintptr_t **CPU_atomic_Uintptr**
- typedef atomic_flag **CPU_atomic_Flag**
- typedef memory_order **CPU_atomic_Order**

## Functions

- static void _CPU_atomic_Fence (CPU_atomic_Order order)

  *Sets up a cpu fence.*
- static void _CPU_atomic_Init_uint (CPU_atomic_Uint *obj, unsigned int desired)

  *Initializes Uint.*
- static void _CPU_atomic_Init_ulong (CPU_atomic_Ulong *obj, unsigned long desired)

  *Initializes Ulong.*
- static void _CPU_atomic_Init_uintptr (CPU_atomic_Uintptr *obj, uintptr_t desired)

  *Initializes Uintptr.*
- static unsigned int _CPU_atomic_Load_uint (const CPU_atomic_Uint *obj, CPU_atomic_Order order)

  *Loads value of Uint considering the order.*
- static unsigned long _CPU_atomic_Load_ulong (const CPU_atomic_Ulong *obj, CPU_atomic_Order order)

  *Loads value of Ulong considering the order.*
- static uintptr_t _CPU_atomic_Load_uintptr (const CPU_atomic_Uintptr *obj, CPU_atomic_Order order)

  *Loads value of Uintptr considering the order.*
- static void _CPU_atomic_Store_uint (CPU_atomic_Uint *obj, unsigned int desired, CPU_atomic_Order order)

  *Stores a value to Uint considering the order.*
- static void _CPU_atomic_Store_ulong (CPU_atomic_Ulong *obj, unsigned long desired, CPU_atomic_Order order)

  *Stores a value to Ulong considering the order.*
- static void _CPU_atomic_Store_uintptr (CPU_atomic_Uintptr *obj, uintptr_t desired, CPU_atomic_Order order)

  *Stores a value to Uintptr considering the order.*
- static unsigned int _CPU_atomic_Fetch_add_uint (CPU_atomic_Uint *obj, unsigned int arg, CPU_atomic_Order order)

  *Fetches current value of Uint and adds a value to the stored value.*
- static unsigned long _CPU_atomic_Fetch_add_ulong (CPU_atomic_Ulong *obj, unsigned long arg, CPU_atomic_Order order)

  *Fetches current value of Ulong and adds a value to the stored value.*
- static uintptr_t _CPU_atomic_Fetch_add_uintptr (CPU_atomic_Uintptr *obj, uintptr_t arg, CPU_atomic_Order order)

  *Fetches current value of Uintptr and adds a value to the stored value.*
- static unsigned int _CPU_atomic_Fetch_sub_uint (CPU_atomic_Uint *obj, unsigned int arg, CPU_atomic_Order order)

  *Fetches current value of Uint and subtracts a value from the stored value.*
- static unsigned long _CPU_atomic_Fetch_sub_ulong (CPU_atomic_Ulong *obj, unsigned long arg, CPU_atomic_Order order)

  *Fetches current value of Ulong and subtracts a value from the stored value.*
- static uintptr_t _CPU_atomic_Fetch_sub_uintptr (CPU_atomic_Uintptr *obj, uintptr_t arg, CPU_atomic_Order order)

  *Fetches current value of Uintptr and subtracts a value from the stored value.*

- static unsigned int _CPU_atomic_Fetch_or_uint (CPU_atomic_Uint *obj, unsigned int arg, CPU_atomic_Order order)

  *Fetches current value of Uint and ORs a value with the stored value.*
- static unsigned long _CPU_atomic_Fetch_or_ulong (CPU_atomic_Ulong *obj, unsigned long arg, CPU_atomic_Order order)

  *Fetches current value of Ulong and ORs a value with the stored value.*
- static uintptr_t _CPU_atomic_Fetch_or_uintptr (CPU_atomic_Uintptr *obj, uintptr_t arg, CPU_atomic_Order order)

  *Fetches current value of Uintptr and ORs a value with the stored value.*
- static unsigned int _CPU_atomic_Fetch_and_uint (CPU_atomic_Uint *obj, unsigned int arg, CPU_atomic_Order order)

  *Fetches current value of Uint and ANDs a value with the stored value.*
- static unsigned long _CPU_atomic_Fetch_and_ulong (CPU_atomic_Ulong *obj, unsigned long arg, CPU_atomic_Order order)

  *Fetches current value of Ulong and ANDs a value with the stored value.*
- static uintptr_t _CPU_atomic_Fetch_and_uintptr (CPU_atomic_Uintptr *obj, uintptr_t arg, CPU_atomic_Order order)

  *Fetches current value of Uintptr and ANDs a value with the stored value.*
- static unsigned int _CPU_atomic_Exchange_uint (CPU_atomic_Uint *obj, unsigned int desired, CPU_atomic_Order order)

  *Fetches current value of Uint and sets its value.*
- static unsigned long _CPU_atomic_Exchange_ulong (CPU_atomic_Ulong *obj, unsigned long desired, CPU_atomic_Order order)

  *Fetches current value of Ulong and sets its value.*
- static uintptr_t _CPU_atomic_Exchange_uintptr (CPU_atomic_Uintptr *obj, uintptr_t desired, CPU_atomic_Order order)

  *Fetches current value of Uintptr and sets its value.*
- static bool _CPU_atomic_Compare_exchange_uint (CPU_atomic_Uint *obj, unsigned int *expected, unsigned int desired, CPU_atomic_Order succ, CPU_atomic_Order fail)

  *Checks if value of Uint is as expected.*
- static bool _CPU_atomic_Compare_exchange_ulong (CPU_atomic_Ulong *obj, unsigned long *expected, unsigned long desired, CPU_atomic_Order succ, CPU_atomic_Order fail)

  *Checks if value of Ulong is as expected.*
- static bool _CPU_atomic_Compare_exchange_uintptr (CPU_atomic_Uintptr *obj, uintptr_t *expected, uintptr_t desired, CPU_atomic_Order succ, CPU_atomic_Order fail)

  *Checks if value of Uintptr is as expected.*
- static void _CPU_atomic_Flag_clear (CPU_atomic_Flag *obj, CPU_atomic_Order order)

  *Clears the atomic flag.*
- static bool _CPU_atomic_Flag_test_and_set (CPU_atomic_Flag *obj, CPU_atomic_Order order)

  *Returns current flag state and sets it.*

### 10.146.1 Detailed Description

Atomic Operations CPU API.

## 10.147 cpukit/include/rtems/score/freechain.h File Reference

Freechain Handler API.

```
#include <rtems/score/chain.h>
```

### Classes

- struct Freechain_Control

  *The freechain control.*

### 10.147.1 Detailed Description

Freechain Handler API.

## 10.148 cpukit/include/rtems/score/freechainimpl.h File Reference

Freechain Handler API.

```
#include <rtems/score/freechain.h>
#include <rtems/score/basedefs.h>
#include <rtems/score/chainimpl.h>
```

### Typedefs

- typedef void *(* Freechain_Allocator) (size_t size)

  *Allocator function.*

### Functions

- static __inline__ void _Freechain_Initialize (Freechain_Control *freechain, void *initial_nodes, size_t number_nodes, size_t node_size)

  *Initializes a freechain.*
- static __inline__ bool _Freechain_Is_empty (const Freechain_Control *freechain)

  *Return true if the freechain is empty, otherwise false.*
- static __inline__ void * _Freechain_Pop (Freechain_Control *freechain)

  *Pops a node from the freechain.*
- static void __inline__ _Freechain_Push (Freechain_Control *freechain, void *node)

  *Pushes a node back to the freechain.*
- void * _Freechain_Extend (Freechain_Control *freechain, Freechain_Allocator allocator, size_t number_nodes_to_extend, size_t node_size)

  *Extend the freechain with new nodes.*
- void * _Freechain_Get (Freechain_Control *freechain, Freechain_Allocator allocator, size_t number_nodes_to_extend, size_t node_size)

  *Gets a node from the freechain.*
- void _Freechain_Put (Freechain_Control *freechain, void *node)

  *Puts a node back onto the freechain.*

### 10.148.1 Detailed Description

Freechain Handler API.

## 10.149 cpukit/include/rtems/score/heap.h File Reference

Heap Handler API.

```
#include <rtems/score/cpu.h>
#include <rtems/score/heapinfo.h>
```

### Classes

- struct Heap_Error_context

  *Context of a heap error.*
- struct Heap_Block

  *Description for free or used blocks.*
- struct Heap_Control

  *Control block used to manage a heap.*
- struct Heap_Area

  *Heap area structure for table based heap initialization and extension.*

### Macros

- #define **HEAP_PROTECTION_HEADER_SIZE** 0
- #define HEAP_BLOCK_HEADER_SIZE (2 * sizeof(uintptr_t) + HEAP_PROTECTION_HEADER_SIZE)

  *The block header consists of the two size fields (Heap_Block::prev_size and Heap_Block::size_and_flag).*

### Typedefs

- typedef struct Heap_Control **Heap_Control**
- typedef struct Heap_Block **Heap_Block**
- typedef uintptr_t(* Heap_Initialization_or_extend_handler) (Heap_Control *heap, void *area_begin, uintptr_t area_size, uintptr_t page_size_or_unused)

  *Heap initialization and extend handler type.*

### Enumerations

- enum Heap_Error_reason {
  HEAP_ERROR_BROKEN_PROTECTOR, HEAP_ERROR_FREE_PATTERN, HEAP_ERROR_DOUBLE_FREE, HEAP_ERROR_BAD_USED_BLOCK,
  HEAP_ERROR_BAD_FREE_BLOCK }

  *The heap error reason.*

### Functions

- uintptr_t _Heap_Extend (Heap_Control *heap, void *area_begin, uintptr_t area_size, uintptr_t unused)

  *Extends the memory available for the heap.*
- uintptr_t _Heap_No_extend (Heap_Control *unused_0, void *unused_1, uintptr_t unused_2, uintptr_t unused_3)

  *This function returns always zero.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Align_up (uintptr_t value, uintptr_t alignment)

  *Aligns the value to a given alignment, rounding up.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Min_block_size (uintptr_t page_size)

  *Returns the minimal Heap Block size for the given page_size.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Area_overhead (uintptr_t page_size)

  *Returns the worst case overhead to manage a memory area.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Size_with_overhead (uintptr_t page_size, uintptr_t size, uintptr_t alignment)

  *Returns the size with administration and alignment overhead for one allocation.*

### 10.149.1 Detailed Description

Heap Handler API.

## 10.150 cpukit/include/rtems/score/heapimpl.h File Reference

Heap Handler Implementation.

```
#include <rtems/score/heap.h>
```

### Macros

- #define HEAP_PREV_BLOCK_USED ((uintptr_t) 1)

  *See also Heap_Block::size_and_flag.*
- #define HEAP_ALLOC_BONUS sizeof(uintptr_t)

  *Size of the part at the block begin which may be used for allocation in charge of the previous block.*
- #define **_Heap_Protection_block_initialize**(heap, block) ((void) 0)
- #define **_Heap_Protection_block_check**(heap, block) ((void) 0)
- #define **_Heap_Protection_block_error**(heap, block, reason) ((void) 0)
- #define **_Heap_Protection_free_all_delayed_blocks**(heap) ((void) 0)
- #define **_HAssert**(cond) ((void) 0)

### Typedefs

- typedef bool(* Heap_Block_visitor) (const Heap_Block *block, uintptr_t block_size, bool block_is_used, void *visitor_arg)

  *Heap block visitor.*

## Enumerations

- enum Heap_Resize_status { **HEAP_RESIZE_SUCCESSFUL**, **HEAP_RESIZE_UNSATISFIED**, **HEAP_RESIZE_FATAL_ERROR** }

  *See _Heap_Resize_block().*

## Functions

- bool _Heap_Get_first_and_last_block (uintptr_t heap_area_begin, uintptr_t heap_area_size, uintptr_t page_size, uintptr_t min_block_size, Heap_Block **first_block_ptr, Heap_Block **last_block_ptr)

  *Gets the first and last block for the heap area.*
- uintptr_t _Heap_Initialize (Heap_Control *heap, void *area_begin, uintptr_t area_size, uintptr_t page_size)

  *Initializes the heap control block.*
- void * _Heap_Allocate_aligned_with_boundary (Heap_Control *heap, uintptr_t size, uintptr_t alignment, uintptr_t boundary)

  *Allocates an aligned memory area with boundary constraint.*
- RTEMS_INLINE_ROUTINE void * _Heap_Allocate_aligned (Heap_Control *heap, uintptr_t size, uintptr_t alignment)

  *Allocates an aligned memory area.*
- RTEMS_INLINE_ROUTINE void * _Heap_Allocate (Heap_Control *heap, uintptr_t size)

  *Allocates a memory area.*
- bool _Heap_Free (Heap_Control *heap, void *addr)

  *Frees the allocated memory area.*
- bool _Heap_Walk (Heap_Control *heap, int source, bool dump)

  *Verifies the integrity of the heap.*
- void _Heap_Iterate (Heap_Control *heap, Heap_Block_visitor visitor, void *visitor_arg)

  *Iterates over all blocks of the heap.*
- Heap_Block * _Heap_Greedy_allocate (Heap_Control *heap, const uintptr_t *block_sizes, size_t block_count)

  *Greedily allocates and empties the heap.*
- Heap_Block * _Heap_Greedy_allocate_all_except_largest (Heap_Control *heap, uintptr_t *allocatable_size)

  *Greedily allocates all blocks except the largest free block.*
- void _Heap_Greedy_free (Heap_Control *heap, Heap_Block *blocks)

  *Frees blocks of a greedy allocation.*
- void _Heap_Get_information (Heap_Control *heap, Heap_Information_block *info)

  *Returns information about used and free blocks for the heap.*
- void _Heap_Get_free_information (Heap_Control *heap, Heap_Information *info)

  *Returns information about free blocks for the heap.*
- bool _Heap_Size_of_alloc_area (Heap_Control *heap, void *addr, uintptr_t *size)

  *Returns the size of the allocatable memory area.*
- Heap_Resize_status _Heap_Resize_block (Heap_Control *heap, void *addr, uintptr_t size, uintptr_t *old_size, uintptr_t *new_size)

  *Resizes the block of the allocated memory area.*
- Heap_Block * _Heap_Block_allocate (Heap_Control *heap, Heap_Block *block, uintptr_t alloc_begin, uintptr_t alloc_size)

  *Allocates the memory area. starting at alloc_begin of size alloc_size bytes in the block block.*
- RTEMS_INLINE_ROUTINE void _Heap_Protection_set_delayed_free_fraction (Heap_Control *heap, uintptr_t fraction)

  *Sets the fraction of delayed free blocks that is actually freed during memory shortage.*
- RTEMS_INLINE_ROUTINE Heap_Block * _Heap_Free_list_head (Heap_Control *heap)

  *Returns the head of the free list of the heap.*

- RTEMS_INLINE_ROUTINE Heap_Block * _Heap_Free_list_tail (Heap_Control *heap)

  *Returns the tail of the free list of the heap.*
- RTEMS_INLINE_ROUTINE Heap_Block * _Heap_Free_list_first (Heap_Control *heap)

  *Returns the first block of the free list of the heap.*
- RTEMS_INLINE_ROUTINE Heap_Block * _Heap_Free_list_last (Heap_Control *heap)

  *Returns the last block of the free list of the heap.*
- RTEMS_INLINE_ROUTINE void _Heap_Free_list_remove (Heap_Block *block)

  *Removes the block from the free list.*
- RTEMS_INLINE_ROUTINE void _Heap_Free_list_replace (Heap_Block *old_block, Heap_Block *new_block)

  *Replaces one block in the free list by another.*
- RTEMS_INLINE_ROUTINE void _Heap_Free_list_insert_after (Heap_Block *block_before, Heap_Block *new_block)

  *Inserts a block after an existing block in the free list.*
- RTEMS_INLINE_ROUTINE void _Heap_Free_list_insert_before (Heap_Block *block_next, Heap_Block *new_block)

  *Inserts a block before an existing block in the free list.*
- RTEMS_INLINE_ROUTINE bool _Heap_Is_aligned (uintptr_t value, uintptr_t alignment)

  *Checks if the value is aligned to the given alignment.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Align_down (uintptr_t value, uintptr_t alignment)

  *Returns the aligned value, truncating.*
- RTEMS_INLINE_ROUTINE Heap_Block * _Heap_Block_at (const Heap_Block *block, uintptr_t offset)

  *Returns the block which is offset away from block.*
- RTEMS_INLINE_ROUTINE Heap_Block * _Heap_Prev_block (const Heap_Block *block)

  *Returns the address of the previous block.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Alloc_area_of_block (const Heap_Block *block)

  *Returns the first address in the block without the heap header.*
- RTEMS_INLINE_ROUTINE Heap_Block * _Heap_Block_of_alloc_area (uintptr_t alloc_begin, uintptr_t page_size)

  *Returns the starting address of the block corresponding to the allocatable area.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Block_size (const Heap_Block *block)

  *Returns the block size.*
- RTEMS_INLINE_ROUTINE void _Heap_Block_set_size (Heap_Block *block, uintptr_t size)

  *Sets the block size.*
- RTEMS_INLINE_ROUTINE bool _Heap_Is_prev_used (const Heap_Block *block)

  *Returns if the previous heap block is used.*
- RTEMS_INLINE_ROUTINE bool _Heap_Is_used (const Heap_Block *block)

  *Returns if the heap block is used.*
- RTEMS_INLINE_ROUTINE bool _Heap_Is_free (const Heap_Block *block)

  *Returns if the heap block is free.*
- RTEMS_INLINE_ROUTINE bool _Heap_Is_block_in_heap (const Heap_Control *heap, const Heap_Block *block)

  *Returns if the block is part of the heap.*
- RTEMS_INLINE_ROUTINE void _Heap_Set_last_block_size (Heap_Control *heap)

  *Sets the size of the last block for the heap.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Get_size (const Heap_Control *heap)

  *Returns the size of the allocatable area in bytes.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Max (uintptr_t a, uintptr_t b)

  *Returns the bigger one of the two arguments.*
- RTEMS_INLINE_ROUTINE uintptr_t _Heap_Min (uintptr_t a, uintptr_t b)

  *Returns the smaller one of the two arguments.*

### 10.150.1 Detailed Description

Heap Handler Implementation.

## 10.151 cpukit/include/rtems/score/heapinfo.h File Reference

Heap Handler Information API.

```
#include <stdint.h>
```

### Classes

- struct Heap_Statistics

  *Run-time heap statistics.*
- struct Heap_Information

  *Information about blocks.*
- struct Heap_Information_block

  *Information block returned by _Heap_Get_information().*

### 10.151.1 Detailed Description

Heap Handler Information API.

## 10.152 cpukit/include/rtems/score/interr.h File Reference

Constants and Prototypes Related to the Internal Error Handler.

```
#include <rtems/score/cpu.h>
```

### Classes

- struct Internal_errors_Information

### Typedefs

- typedef CPU_Uint32ptr **Internal_errors_t**

## Enumerations

- enum Internal_errors_Source {
  INTERNAL_ERROR_CORE = 0, INTERNAL_ERROR_RTEMS_API = 1, INTERNAL_ERROR_POSIX_API = 2, RTEMS_FATAL_SOURCE_BDBUF = 3,
  RTEMS_FATAL_SOURCE_APPLICATION = 4, RTEMS_FATAL_SOURCE_EXIT = 5, RTEMS_FATAL_SOURCE_BSP = 6, RTEMS_FATAL_SOURCE_ASSERT = 7,
  RTEMS_FATAL_SOURCE_STACK_CHECKER = 8, RTEMS_FATAL_SOURCE_EXCEPTION = 9, RTEMS_FATAL_SOURCE_SMP = 10, RTEMS_FATAL_SOURCE_PANIC = 11,
  RTEMS_FATAL_SOURCE_INVALID_HEAP_FREE = 12, RTEMS_FATAL_SOURCE_HEAP = 13, RTEMS_FATAL_SOURCE_LAST = 0xffffffff }

  *This type lists the possible sources from which an error can be reported.*
- enum Internal_errors_Core_list {
  **INTERNAL_ERROR_TOO_LITTLE_WORKSPACE** = 2, **INTERNAL_ERROR_THREAD_EXITTED** = 5, **INTERNAL_ERROR_INCONSISTENT_MP_INFORMATION** = 6, **INTERNAL_ERROR_INVALID_NODE** = 7, **INTERNAL_ERROR_NO_MPCI** = 8, **INTERNAL_ERROR_BAD_PACKET** = 9, **INTERNAL_ERROR_OUT_OF_PACKETS** = 10, **INTERNAL_ERROR_OUT_OF_GLOBAL_OBJECTS** = 11,
  **INTERNAL_ERROR_OUT_OF_PROXIES** = 12, **INTERNAL_ERROR_INVALID_GLOBAL_ID** = 13, **INTERNAL_ERROR_BAD_STACK_HOOK** = 14, **INTERNAL_ERROR_GXX_KEY_ADD_FAILED** = 21,
  **INTERNAL_ERROR_GXX_MUTEX_INIT_FAILED** = 22, **INTERNAL_ERROR_NO_MEMORY_FOR_HEAP** = 23, **INTERNAL_ERROR_CPU_ISR_INSTALL_VECTOR** = 24, **INTERNAL_ERROR_RESOURCE_IN_USE** = 25,
  **INTERNAL_ERROR_RTEMS_INIT_TASK_ENTRY_IS_NULL** = 26, **INTERNAL_ERROR_THREAD_QUEUE_DEADLOCK** = 28, **INTERNAL_ERROR_THREAD_QUEUE_ENQUEUE_STICKY_FROM_BAD_STATE** = 29, **INTERNAL_ERROR_BAD_THREAD_DISPATCH_DISABLE_LEVEL** = 30,
  **INTERNAL_ERROR_BAD_THREAD_DISPATCH_ENVIRONMENT** = 31, **INTERNAL_ERROR_RTEMS_INIT_TASK_CREATE_FAILED** = 32, **INTERNAL_ERROR_POSIX_INIT_THREAD_CREATE_FAILED** = 33, **INTERNAL_ERROR_LIBIO_STDOUT_FD_OPEN_FAILED** = 36,
  **INTERNAL_ERROR_LIBIO_STDERR_FD_OPEN_FAILED** = 37, **INTERNAL_ERROR_ILLEGAL_USE_OF_FLOATING_POINT_UNIT** = 38, **INTERNAL_ERROR_ARC4RANDOM_GETENTROPY_FAIL** = 39, **INTERNAL_ERROR_NO_MEMORY_FOR_PER_CPU_DATA** = 40,
  **INTERNAL_ERROR_TOO_LARGE_TLS_SIZE** = 41 }

  *A list of errors which are generated internally by the executive core.*

## Functions

- RTEMS_NO_RETURN void _Terminate (Internal_errors_Source the_source, Internal_errors_t the_error)

  *Initiates system termination.*
- RTEMS_NO_RETURN void _Internal_error (Internal_errors_Core_list core_error)

  *Terminates the system with an INTERNAL_ERROR_CORE fatal source and the specified core error code.*

## Variables

- Internal_errors_Information _Internal_errors_What_happened

### 10.152.1 Detailed Description

Constants and Prototypes Related to the Internal Error Handler.

This include file contains constants and prototypes related to the Internal Error Handler.

## 10.153 cpukit/include/rtems/score/isr.h File Reference

Data Related to the Management of Processor Interrupt Levels.

```
#include <rtems/score/isrlevel.h>
```

### Macros

- #define _ISR_Install_vector(_vector, _new_handler, _old_handler) _CPU_ISR_install_vector( _vector, _↩ new_handler, _old_handler )

  *Install interrupt handler vector.*

### Typedefs

- typedef uint32_t ISR_Vector_number
- typedef void ISR_Handler
- typedef void ∗ **ISR_Handler_entry**

### Functions

- RTEMS_DECLARE_GLOBAL_SYMBOL (_ISR_Stack_size)

  *Global symbol with a value equal to the configure interrupt stack size.*
- void _ISR_Handler_initialization (void)

  *Initializes the ISR handler.*
- void _ISR_Handler (void)

  *ISR interrupt dispatcher.*
- bool _ISR_Is_in_progress (void)

  *Checks if an ISR in progress.*

### Variables

- char _ISR_Stack_area_begin [ ]

  *The interrupt stack area begin.*
- const char _ISR_Stack_area_end [ ]

  *The interrupt stack area end.*

### 10.153.1 Detailed Description

Data Related to the Management of Processor Interrupt Levels.

This include file contains all the constants and structures associated with the management of processor interrupt levels. This handler supports interrupt critical sections, vectoring of user interrupt handlers, nesting of interrupts, and manipulating interrupt levels.

## 10.154 cpukit/include/rtems/score/isrlevel.h File Reference

ISR Level Type.

```
#include <rtems/score/cpu.h>
#include <rtems/score/assert.h>
```

### Macros

- #define _ISR_Local_disable(_level)
  *Disables interrupts on this processor.*
- #define _ISR_Local_enable(_level)
  *Enables interrupts on this processor.*
- #define _ISR_Local_flash(_level)
  *Temporarily enables interrupts on this processor.*
- #define _ISR_Is_enabled(_level) _CPU_ISR_Is_enabled( _level )
  *Returns true if interrupts are enabled in the specified interrupt level, otherwise returns false.*
- #define _ISR_Get_level() _CPU_ISR_Get_level()
  *Return current interrupt level.*
- #define _ISR_Set_level(_new_level)
  *Set current interrupt level.*

### Typedefs

- typedef uint32_t ISR_Level

### 10.154.1 Detailed Description

ISR Level Type.

This include file defines the ISR Level type. It exists to simplify include dependencies. It is part of the ISR Handler.

## 10.155 cpukit/include/rtems/score/isrlock.h File Reference

ISR Locks.

```
#include <rtems/score/isrlevel.h>
#include <rtems/score/smplock.h>
```

### Classes

- struct ISR_lock_Control
  *ISR lock control.*
- struct ISR_lock_Context
  *Local ISR lock context for acquire and release pairs.*

## Macros

- #define ISR_LOCK_MEMBER(_designator) ISR_lock_Control _designator;

  *Defines an ISR lock member.*
- #define ISR_LOCK_DECLARE(_qualifier, _designator) _qualifier ISR_lock_Control _designator;

  *Declares an ISR lock variable.*
- #define ISR_LOCK_DEFINE(_qualifier, _designator, _name) _qualifier ISR_lock_Control _designator = { SMP_LOCK_INITIALIZER( _name ) };

  *Defines an ISR lock variable.*
- #define ISR_LOCK_REFERENCE(_designator, _target) ISR_lock_Control *_designator = _target;

  *Defines an ISR lock variable reference.*
- #define ISR_LOCK_INITIALIZER(_name) { SMP_LOCK_INITIALIZER( _name ) }

  *Initializer for static initialization of ISR locks.*
- #define _ISR_lock_Initialize(_lock, _name) _SMP_lock_Initialize( &( _lock )->Lock, _name )

  *Initializes an ISR lock.*
- #define _ISR_lock_Destroy(_lock) _SMP_lock_Destroy( &( _lock )->Lock )

  *Destroys an ISR lock.*
- #define _ISR_lock_Set_name(_lock, _name) _SMP_lock_Set_name( &( _lock )->Lock, _name )

  *Sets the name of an ISR lock.*
- #define _ISR_lock_ISR_disable_and_acquire(_lock, _context)

  *Acquires an ISR lock.*
- #define _ISR_lock_Release_and_ISR_enable(_lock, _context)

  *Releases an ISR lock.*
- #define _ISR_lock_Acquire(_lock, _context)

  *Acquires an ISR lock inside an ISR disabled section.*
- #define _ISR_lock_Release(_lock, _context)

  *Releases an ISR lock inside an ISR disabled section.*
- #define _ISR_lock_Acquire_inline(_lock, _context)

  *Acquires an ISR lock inside an ISR disabled section (inline).*
- #define _ISR_lock_Release_inline(_lock, _context)

  *Releases an ISR lock inside an ISR disabled section (inline).*
- #define **_ISR_lock_ISR_disable_profile**(_context)
- #define _ISR_lock_ISR_disable(_context)

  *Disables interrupts and saves the previous interrupt state in the ISR lock context.*
- #define _ISR_lock_ISR_enable(_context) _ISR_Local_enable( ( _context )->Lock_context.isr_level )

  *Restores the saved interrupt state of the ISR lock context.*

## Functions

- static __inline__ void _ISR_lock_Context_set_level (ISR_lock_Context *context, ISR_Level level)

  *Sets the ISR level in the ISR lock context.*

### 10.155.1 Detailed Description

ISR Locks.

# 10.156 cpukit/include/rtems/score/memory.h File Reference

Memory Handler API.

```
#include <rtems/score/basedefs.h>
#include <rtems/score/assert.h>
```

## Classes

- struct Memory_Area
  *The memory area description.*
- struct Memory_Information
  *The memory information.*

## Macros

- #define MEMORY_INFORMATION_INITIALIZER(areas) { RTEMS_ARRAY_SIZE( areas ), ( areas ) }
  *Statically initialize a memory information.*
- #define MEMORY_INITIALIZER(begin, end) { ( begin ), ( begin ), ( end ) }
  *Statically initialize a memory area.*

## Functions

- static __inline__ size_t _Memory_Get_count (const Memory_Information *information)
  *Get the memory area count.*
- static __inline__ Memory_Area * _Memory_Get_area (const Memory_Information *information, size_t index)
  *Get a memory area by index.*
- static __inline__ void _Memory_Initialize (Memory_Area *area, void *begin, void *end)
  *Initialize the memory area.*
- static __inline__ void _Memory_Initialize_by_size (Memory_Area *area, void *begin, uintptr_t size)
  *Initialize the memory area by size.*
- static __inline__ const void * _Memory_Get_begin (const Memory_Area *area)
  *Get the memory area begin.*
- static __inline__ void _Memory_Set_begin (Memory_Area *area, const void *begin)
  *Set the memory area begin.*
- static __inline__ const void * _Memory_Get_end (const Memory_Area *area)
  *Get the memory area end.*
- static __inline__ void _Memory_Set_end (Memory_Area *area, const void *end)
  *Set the memory area end.*
- static __inline__ uintptr_t _Memory_Get_size (const Memory_Area *area)
  *Get the memory area size.*
- static __inline__ void * _Memory_Get_free_begin (const Memory_Area *area)
  *Get the begin of the free area of the memory area.*
- static __inline__ void _Memory_Set_free_begin (Memory_Area *area, void *begin)
  *Set the begin of the free area of the memory area.*
- static __inline__ uintptr_t _Memory_Get_free_size (const Memory_Area *area)
  *Get the size of the free memory area of the memory area.*
- static __inline__ void _Memory_Consume (Memory_Area *area, uintptr_t consume)

*Consume the specified size from the free memory area of the memory area.*

- const Memory_Information * _Memory_Get (void)

  *Return the memory information of this platform.*

- void * _Memory_Allocate (const Memory_Information *information, uintptr_t size, uintptr_t alignment)

  *Allocate a memory area from the memory information.*

- void _Memory_Fill (const Memory_Information *information, int c)

  *Fill all free memory areas of the memory information with a constant byte.*

- void _Memory_Zero_free_areas (void)

  *Zeros all free memory areas of the system.*

- void _Memory_Dirty_free_areas (void)

  *Dirty all free memory areas of the system.*

### Variables

- const bool _Memory_Zero_before_use

  *Indicates if the memory is zeroed during system initialization.*

### 10.156.1 Detailed Description

Memory Handler API.

## 10.157 cpukit/include/rtems/score/mppkt.h File Reference

Specification for the Packet Handler.

```
#include <rtems/score/object.h>
#include <rtems/score/priority.h>
#include <rtems/score/watchdogticks.h>
```

### Classes

- struct MP_packet_Prefix

### Macros

- #define MP_PACKET_CLASSES_FIRST MP_PACKET_MPCI_INTERNAL
- #define MP_PACKET_CLASSES_LAST MP_PACKET_SIGNAL
- #define MP_PACKET_MINIMUM_PACKET_SIZE 64
- #define MP_PACKET_MINIMUN_HETERO_CONVERSION ( sizeof( MP_packet_Prefix ) / sizeof( uint32_t ) )

### Enumerations

- enum MP_packet_Classes {
  **MP_PACKET_MPCI_INTERNAL** = 0, **MP_PACKET_TASKS** = 1, **MP_PACKET_MESSAGE_QUEUE** = 2, **MP_PACKET_SEMAPHORE** = 3,
  **MP_PACKET_PARTITION** = 4, **MP_PACKET_REGION** = 5, **MP_PACKET_EVENT** = 6, **MP_PACKET_SIGNAL** = 7 }

### 10.157.1 Detailed Description

Specification for the Packet Handler.

This package is the specification for the Packet Handler. This handler defines the basic packet and provides mechanisms to utilize packets based on this prefix. Packets are the fundamental basis for messages passed between nodes in an MP system.

## 10.158 cpukit/include/rtems/score/mrsp.h File Reference

Definitions for Multiprocessor Resource Sharing Protocol (MrsP).

```
#include <rtems/score/cpuopts.h>
#include <rtems/score/threadq.h>
```

### Classes

- struct MRSP_Control

  *MrsP control block.*

### 10.158.1 Detailed Description

Definitions for Multiprocessor Resource Sharing Protocol (MrsP).

## 10.159 cpukit/include/rtems/score/mrspimpl.h File Reference

Definitions for Multiprocessor Resource Sharing Protocol (MrsP) Implementation.

```
#include <rtems/score/mrsp.h>
#include <rtems/score/assert.h>
#include <rtems/score/status.h>
#include <rtems/score/threadqimpl.h>
#include <rtems/score/watchdogimpl.h>
```

### Macros

- #define **MRSP_TQ_OPERATIONS** &_Thread_queue_Operations_priority_inherit

## Functions

- static __inline__ void _MRSP_Acquire_critical (MRSP_Control *mrsp, Thread_queue_Context *queue_↩ context)

  *Acquires critical accordingt to MrsP.*
- static __inline__ void _MRSP_Release (MRSP_Control *mrsp, Thread_queue_Context *queue_context)

  *Releases according to MrsP.*
- static __inline__ Thread_Control * _MRSP_Get_owner (const MRSP_Control *mrsp)

  *Gets owner of the MrsP control.*
- static __inline__ void _MRSP_Set_owner (MRSP_Control *mrsp, Thread_Control *owner)

  *Sets owner of the MrsP control.*
- static __inline__ Priority_Control _MRSP_Get_priority (const MRSP_Control *mrsp, const Scheduler_Control *scheduler)

  *Gets priority of the MrsP control.*
- static __inline__ void _MRSP_Set_priority (MRSP_Control *mrsp, const Scheduler_Control *scheduler, Priority_Control new_priority)

  *Sets priority of the MrsP control.*
- static __inline__ Status_Control _MRSP_Raise_priority (MRSP_Control *mrsp, Thread_Control *thread, Priority_Node *priority_node, Thread_queue_Context *queue_context)

  *Adds the priority to the given thread.*
- static __inline__ void _MRSP_Remove_priority (Thread_Control *thread, Priority_Node *priority_node, Thread_queue_Context *queue_context)

  *Removes the priority from the given thread.*
- static __inline__ void _MRSP_Replace_priority (MRSP_Control *mrsp, Thread_Control *thread, Priority_Node *ceiling_priority)

  *Replaces the given priority node with the ceiling priority of the MrsP control.*
- static __inline__ Status_Control _MRSP_Claim_ownership (MRSP_Control *mrsp, Thread_Control *executing, Thread_queue_Context *queue_context)

  *Claims ownership of the MrsP control.*
- static __inline__ Status_Control _MRSP_Initialize (MRSP_Control *mrsp, const Scheduler_Control *scheduler, Priority_Control ceiling_priority, Thread_Control *executing, bool initially_locked)

  *Initializes a MrsP control.*
- static __inline__ Status_Control _MRSP_Wait_for_ownership (MRSP_Control *mrsp, Thread_Control *executing, Thread_queue_Context *queue_context)

  *Waits for the ownership of the MrsP control.*
- static __inline__ Status_Control _MRSP_Seize (MRSP_Control *mrsp, Thread_Control *executing, bool wait, Thread_queue_Context *queue_context)

  *Seizes the MrsP control.*
- static __inline__ Status_Control _MRSP_Surrender (MRSP_Control *mrsp, Thread_Control *executing, Thread_queue_Context *queue_context)

  *Surrenders the MrsP control.*
- static __inline__ Status_Control _MRSP_Can_destroy (MRSP_Control *mrsp)

  *Checks if the MrsP control can be destroyed.*
- static __inline__ void _MRSP_Destroy (MRSP_Control *mrsp, Thread_queue_Context *queue_context)

  *Destroys the MrsP control.*

### 10.159.1 Detailed Description

Definitions for Multiprocessor Resource Sharing Protocol (MrsP) Implementation.

# 10.160 cpukit/include/rtems/score/muteximpl.h File Reference

Structures for the implementation of mutexes.

```
#include <rtems/score/threadqimpl.h>
```

## Classes

- struct Mutex_Control
- struct Mutex_recursive_Control

### 10.160.1 Detailed Description

Structures for the implementation of mutexes.

# 10.161 cpukit/include/rtems/score/objectdata.h File Reference

Object Handler Data Structures.

```
#include <rtems/score/object.h>
#include <rtems/score/chainimpl.h>
#include <rtems/score/rbtree.h>
```

## Classes

- struct Objects_Control
- struct Objects_Information

  *The information structure used to manage each API class of objects.*

## Macros

- #define OBJECTS_NO_STRING_NAME 0

  *Constant for the object information string name length to indicate that this object class has no string names.*
- #define **OBJECTS_INFORMATION_MP**(name, extract)
- #define OBJECTS_INFORMATION_DEFINE_ZERO(name, api, cls, nl)

  *Statically initializes an objects information.*
- #define OBJECTS_INFORMATION_DEFINE(name, api, cls, type, max, nl, ex)

  *Statically initializes an objects information.*

## Typedefs

- typedef struct Objects_Information **Objects_Information**

### Enumerations

- enum Objects_Internal_API { **OBJECTS_INTERNAL_NO_CLASS** = 0, **OBJECTS_INTERNAL_THREADS** = 1 }
- enum Objects_Classic_API { **OBJECTS_CLASSIC_NO_CLASS** = 0, **OBJECTS_RTEMS_TASKS** = 1, **OBJECTS_RTEMS_TIMERS**, **OBJECTS_RTEMS_SEMAPHORES**, **OBJECTS_RTEMS_MESSAGE_QUEUES**, **OBJECTS_RTEMS_PARTITIONS**, **OBJECTS_RTEMS_REGIONS**, **OBJECTS_RTEMS_PORTS**, **OBJECTS_RTEMS_PERIODS**, **OBJECTS_RTEMS_EXTENSIONS**, **OBJECTS_RTEMS_BARRIERS** }
- enum Objects_POSIX_API { **OBJECTS_POSIX_NO_CLASS** = 0, **OBJECTS_POSIX_THREADS** = 1, **OBJECTS_POSIX_KEYS**, **OBJECTS_POSIX_MESSAGE_QUEUES**, **OBJECTS_POSIX_SEMAPHORES**, **OBJECTS_POSIX_TIMERS**, **OBJECTS_POSIX_SHMS** }

### Functions

- Objects_Control * _Objects_Allocate_none (Objects_Information *information)

  *Always return NULL.*
- Objects_Control * _Objects_Allocate_static (Objects_Information *information)

  *Return an inactive object or NULL.*
- Objects_Control * _Objects_Allocate_unlimited (Objects_Information *information)

  *Return an inactive object or NULL.*
- void _Objects_Free_static (Objects_Information *information, Objects_Control *the_object)

  *Free the object.*
- void _Objects_Free_unlimited (Objects_Information *information, Objects_Control *the_object)

  *Free the object.*

#### 10.161.1 Detailed Description

Object Handler Data Structures.

## 10.162 cpukit/include/rtems/score/percpudata.h File Reference

Definition of custom per-CPU items.

```
#include <rtems/score/percpu.h>
#include <rtems/linkersets.h>
```

### Macros

- #define PER_CPU_DATA_ITEM_DECLARE(type, item) RTEMS_LINKER_RWSET_ITEM_DECLARE( _Per_CPU_Data, type, item )

  *Declares a per-CPU item of the specified type.*
- #define PER_CPU_DATA_ITEM(type, item) RTEMS_LINKER_RWSET_ITEM( _Per_CPU_Data, type, item )

  *Defines a per-CPU item of the specified type.*
- #define PER_CPU_DATA_OFFSET(item)

  *Returns the offset of the per-CPU item to the begin of the per-CPU data area.*
- #define PER_CPU_DATA_GET_BY_OFFSET(cpu, type, offset) (type *) ( cpu->data + offset )

  *Returns a pointer of the specified type to the per-CPU item at the specified offset for the specified processor.*
- #define PER_CPU_DATA_GET(cpu, type, item) PER_CPU_DATA_GET_BY_OFFSET( cpu, type, PER_CPU_DATA_OFFSET( item ) )

  *Returns a pointer of the specified type to the specified per-CPU item for the specified processor.*

### Functions

- **RTEMS_LINKER_RWSET_DECLARE** (_Per_CPU_Data, char)

### 10.162.1 Detailed Description

Definition of custom per-CPU items.

## 10.163 cpukit/include/rtems/score/priority.h File Reference

Priority Handler API.

```
#include <rtems/score/chain.h>
#include <rtems/score/cpu.h>
#include <rtems/score/rbtree.h>
```

### Classes

- struct Priority_Node

  *The priority node to build up a priority aggregation.*
- struct Priority_Aggregation

  *The priority aggregation.*
- struct Priority_Actions

  *A list of priority actions.*

### Macros

- #define PRIORITY_MINIMUM 0

  *The highest (most important) thread priority value.*
- #define PRIORITY_PSEUDO_ISR PRIORITY_MINIMUM

  *The priority value of pseudo-ISR threads.*
- #define PRIORITY_DEFAULT_MAXIMUM 255

  *The default lowest (least important) thread priority value.*

### Typedefs

- typedef uint64_t Priority_Control

  *The thread priority control.*
- typedef struct Priority_Aggregation **Priority_Aggregation**

### Enumerations

- enum Priority_Action_type { **PRIORITY_ACTION_ADD**, **PRIORITY_ACTION_CHANGE**, **PRIORITY_ACTION_REMOVE**, **PRIORITY_ACTION_INVALID** }

  *The priority action type.*

### 10.163.1 Detailed Description

Priority Handler API.

## 10.164 cpukit/include/rtems/score/prioritybitmap.h File Reference

Manipulation Routines for the Bitmap Priority Queue Implementation.

```
#include <rtems/score/cpu.h>
```

### Classes

- struct Priority_bit_map_Control
- struct Priority_bit_map_Information

### Typedefs

- typedef uint16_t **Priority_bit_map_Word**

### 10.164.1 Detailed Description

Manipulation Routines for the Bitmap Priority Queue Implementation.

This include file contains all thread priority manipulation routines for the bit map priority queue implementation.

## 10.165 cpukit/include/rtems/score/prioritybitmapimpl.h File Reference

Inlined Routines in the Priority Handler Bit Map Implementation.

```
#include <rtems/score/prioritybitmap.h>
#include <string.h>
```

### Functions

- RTEMS_INLINE_ROUTINE unsigned int _Bitfield_Find_first_bit (unsigned int value)

  *Returns the bit number of the first bit set in the specified value.*
- RTEMS_INLINE_ROUTINE Priority_bit_map_Word _Priority_Mask (unsigned int bit_number)

  *Returns the priority bit mask for the specified major or minor bit number.*
- RTEMS_INLINE_ROUTINE unsigned int _Priority_Bits_index (unsigned int bit_number)

  *Returns the bit index position for the specified major or minor bit number.*
- RTEMS_INLINE_ROUTINE unsigned int _Priority_Major (unsigned int the_priority)

  *Returns the major portion of the_priority.*
- RTEMS_INLINE_ROUTINE unsigned int _Priority_Minor (unsigned int the_priority)

  *Returns the minor portion of the_priority.*
- RTEMS_INLINE_ROUTINE void _Priority_bit_map_Initialize (Priority_bit_map_Control *bit_map)

  *Initializes a bit map.*
- RTEMS_INLINE_ROUTINE void _Priority_bit_map_Add (Priority_bit_map_Control *bit_map, Priority_bit_map_Information *bit_map_info)

  *Adds Priority queue bit map information.*
- RTEMS_INLINE_ROUTINE void _Priority_bit_map_Remove (Priority_bit_map_Control *bit_map, Priority_bit_map_Information *bit_map_info)

  *Removes Priority queue bit map information.*
- RTEMS_INLINE_ROUTINE unsigned int _Priority_bit_map_Get_highest (const Priority_bit_map_Control *bit_map)

  *Gets highest portion of Priority queue bit map.*
- RTEMS_INLINE_ROUTINE bool _Priority_bit_map_Is_empty (const Priority_bit_map_Control *bit_map)

  *Checks if the Priority queue bit map is empty.*
- RTEMS_INLINE_ROUTINE void _Priority_bit_map_Initialize_information (Priority_bit_map_Control *bit_map, Priority_bit_map_Information *bit_map_info, unsigned int new_priority)

  *Initializes the bit map information.*

### Variables

- const unsigned char _Bitfield_Leading_zeros [256]

### 10.165.1 Detailed Description

Inlined Routines in the Priority Handler Bit Map Implementation.

This file contains the static inline implementation of all inlined routines in the Priority Handler bit map implementation

## 10.166 cpukit/include/rtems/score/priorityimpl.h File Reference

Priority Handler API Implementation.

```
#include <rtems/score/priority.h>
#include <rtems/score/scheduler.h>
```

## Typedefs

- typedef void(∗ **Priority_Add_handler**) (Priority_Aggregation ∗aggregation, Priority_Actions ∗actions, void ∗arg)
- typedef void(∗ **Priority_Change_handler**) (Priority_Aggregation ∗aggregation, bool prepend_↩ it, Priority_Actions ∗actions, void ∗arg)
- typedef void(∗ **Priority_Remove_handler**) (Priority_Aggregation ∗aggregation, Priority_Actions ∗actions, void ∗arg)

## Functions

- static __inline__ void _Priority_Actions_initialize_empty (Priority_Actions ∗actions)

  *Initializes the priority actions empty.*
- static __inline__ void _Priority_Actions_initialize_one (Priority_Actions ∗actions, Priority_Aggregation ∗aggregation, Priority_Node ∗node, Priority_Action_type type)

  *Initializes the priority actions with the given information.*
- static __inline__ bool _Priority_Actions_is_empty (const Priority_Actions ∗actions)

  *Checks if the priority actions is empty.*
- static __inline__ bool _Priority_Actions_is_valid (const Priority_Aggregation ∗aggregation)

  *Checks if the priority actions is valid.*
- static __inline__ Priority_Aggregation ∗ _Priority_Actions_move (Priority_Actions ∗actions)

  *Moves the priority actions' actions.*
- static __inline__ void _Priority_Actions_add (Priority_Actions ∗actions, Priority_Aggregation ∗aggregation)

  *Adds actions to the priority actions' actions.*
- static __inline__ void _Priority_Node_initialize (Priority_Node ∗node, Priority_Control priority)

  *Initializes the priority node to the given priority.*
- static __inline__ void _Priority_Node_set_priority (Priority_Node ∗node, Priority_Control priority)

  *Sets the priority of the priority node to the given priority.*
- static __inline__ void _Priority_Node_set_inactive (Priority_Node ∗node)

  *Sets the priority node inactive.*
- static __inline__ bool _Priority_Node_is_active (const Priority_Node ∗node)

  *Checks if the priority node is active.*
- static __inline__ void _Priority_Initialize_empty (Priority_Aggregation ∗aggregation)

  *Initializes the priority aggregation empty.*
- static __inline__ void _Priority_Initialize_one (Priority_Aggregation ∗aggregation, Priority_Node ∗node)

  *Initializes the priority aggregation with the given information.*
- static __inline__ bool _Priority_Is_empty (const Priority_Aggregation ∗aggregation)

  *Checks if the priority aggregation is empty.*
- static __inline__ Priority_Control _Priority_Get_priority (const Priority_Aggregation ∗aggregation)

  *Gets the priority aggregation's priority.*
- static __inline__ const Scheduler_Control ∗ _Priority_Get_scheduler (const Priority_Aggregation ∗aggregation)

  *Gets the priority aggregation's scheduler.*
- static __inline__ Priority_Node ∗ _Priority_Get_minimum_node (const Priority_Aggregation ∗aggregation)

  *Gets the minimum node of the priority aggregation.*
- static __inline__ void _Priority_Set_action_node (Priority_Aggregation ∗aggregation, Priority_Node ∗node)

  *Sets the action node of the priority aggregation.*
- static __inline__ void _Priority_Set_action_type (Priority_Aggregation ∗aggregation, Priority_Action_type type)

  *Sets the action type of the priority aggregation.*

- static __inline__ void _Priority_Set_action (Priority_Aggregation *aggregation, Priority_Node *node, Priority_Action_type type)

  *Sets the action type and action node of the priority aggregation.*
- static __inline__ Priority_Aggregation * _Priority_Get_next_action (const Priority_Aggregation *aggregation)

  *Gets the next action of the priority aggregation.*
- static __inline__ bool _Priority_Less (const void *left, const RBTree_Node *right)

  *Compares two priorities.*
- static __inline__ bool _Priority_Plain_insert (Priority_Aggregation *aggregation, Priority_Node *node, Priority_Control priority)

  *Inserts the node with the given priority into the priority aggregation's contributors.*
- static __inline__ void _Priority_Plain_extract (Priority_Aggregation *aggregation, Priority_Node *node)

  *Extracts the priority node from the aggregation.*
- static __inline__ void _Priority_Plain_changed (Priority_Aggregation *aggregation, Priority_Node *node)

  *Updates the priority of the node in the aggregation.*
- static __inline__ void _Priority_Change_nothing (Priority_Aggregation *aggregation, bool prepend_it, Priority_Actions *actions, void *arg)

  *Does nothing.*
- static __inline__ void _Priority_Remove_nothing (Priority_Aggregation *aggregation, Priority_Actions *actions, void *arg)

  *Does nothing.*
- static __inline__ void _Priority_Non_empty_insert (Priority_Aggregation *aggregation, Priority_Node *node, Priority_Actions *actions, Priority_Change_handler change, void *arg)

  *Inserts the node in a nonempty aggregation and handles change if the node is the new minimum.*
- static __inline__ void _Priority_Insert (Priority_Aggregation *aggregation, Priority_Node *node, Priority_Actions *actions, Priority_Add_handler add, Priority_Change_handler change, void *arg)
- static __inline__ void _Priority_Extract (Priority_Aggregation *aggregation, Priority_Node *node, Priority_Actions *actions, Priority_Remove_handler remove, Priority_Change_handler change, void *arg)

  *Extracts the node from the aggregation.*
- static __inline__ void _Priority_Extract_non_empty (Priority_Aggregation *aggregation, Priority_Node *node, Priority_Actions *actions, Priority_Change_handler change, void *arg)

  *Extracts the node from the aggregation.*
- static __inline__ void _Priority_Changed (Priority_Aggregation *aggregation, Priority_Node *node, bool prepend_it, Priority_Actions *actions, Priority_Change_handler change, void *arg)

  *Updates the priority of the node in the aggregation.*
- static __inline__ void _Priority_Replace (Priority_Aggregation *aggregation, Priority_Node *victim, Priority_Node *replacement)

  *Replaces one node by another.*

### 10.166.1 Detailed Description

Priority Handler API Implementation.

## 10.167 cpukit/include/rtems/score/processormask.h File Reference

Processor Mask API.

```
#include <rtems/score/cpu.h>
#include <sys/cpuset.h>
#include <strings.h>
```

## Enumerations

- enum **Processor_mask_Copy_status** { **PROCESSOR_MASK_COPY_LOSSLESS**, **PROCESSOR_MA↩ SK_COPY_PARTIAL_LOSS**, **PROCESSOR_MASK_COPY_COMPLETE_LOSS**, **PROCESSOR_MASK↩ _COPY_INVALID_SIZE** }

## Functions

- typedef BITSET_DEFINE (Processor_mask, CPU_MAXIMUM_PROCESSORS) Processor_mask

  *A bit map which is large enough to provide one bit for each processor in the system.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_Zero (Processor_mask *mask)

  *Sets the bits of the mask to zero, also considers CPU_MAXIMUM_PROCESSORS.*
- RTEMS_INLINE_ROUTINE bool _Processor_mask_Is_zero (const Processor_mask *mask)

  *Checks if the mask is zero, also considers CPU_MAXIMUM_PROCESSORS.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_Fill (Processor_mask *mask)

  *Fills the mask, also considers CPU_MAXIMUM_PROCESSORS.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_Assign (Processor_mask *dst, const Processor_mask *src)

  *Copies the mask to another mask, also considers CPU_MAXIMUM_PROCESSORS.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_Set (Processor_mask *mask, uint32_t index)

  *Sets the specified index bit of the mask.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_Clear (Processor_mask *mask, uint32_t index)

  *Clears the specified index bit of the mask.*
- RTEMS_INLINE_ROUTINE bool _Processor_mask_Is_set (const Processor_mask *mask, uint32_t index)

  *Checks if the specified index bit of the mask is set.*
- RTEMS_INLINE_ROUTINE bool _Processor_mask_Is_equal (const Processor_mask *a, const Processor↩ _mask *b)

  *Checks if the processor sets a and b are equal.*
- RTEMS_INLINE_ROUTINE bool _Processor_mask_Has_overlap (const Processor_mask *a, const Processor_mask *b)

  *Checks if the intersection of the processor sets a and b is non-empty.*
- RTEMS_INLINE_ROUTINE bool _Processor_mask_Is_subset (const Processor_mask *big, const Processor_mask *small)

  *Checks if the processor set small is a subset of processor set big.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_And (Processor_mask *a, const Processor_mask *b, const Processor_mask *c)

  *Performs a bitwise a = b & c.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_Nand (Processor_mask *a, const Processor_mask *b, const Processor_mask *c)

  *Performs a bitwise a = b & ~c.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_Or (Processor_mask *a, const Processor_mask *b, const Processor_mask *c)

  *Performs a bitwise a = b | c.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_Xor (Processor_mask *a, const Processor_mask *b, const Processor_mask *c)

  *Performs a bitwise a = b ^ c.*
- RTEMS_INLINE_ROUTINE uint32_t _Processor_mask_Count (const Processor_mask *a)

  *Gets the number of set bits in the processor mask.*
- RTEMS_INLINE_ROUTINE uint32_t _Processor_mask_Find_last_set (const Processor_mask *a)

  *Finds the last set of the processor mask.*
- RTEMS_INLINE_ROUTINE uint32_t _Processor_mask_To_uint32_t (const Processor_mask *mask, uint32_t index)

*Returns the subset of 32 processors containing the specified index as an unsigned 32-bit integer.*

- RTEMS_INLINE_ROUTINE void _Processor_mask_From_uint32_t (Processor_mask *mask, uint32_t bits, uint32_t index)

  *Creates a processor set from an unsigned 32-bit integer relative to the specified index.*
- RTEMS_INLINE_ROUTINE void _Processor_mask_From_index (Processor_mask *mask, uint32_t index)

  *Creates a processor set from the specified index.*
- RTEMS_INLINE_ROUTINE bool _Processor_mask_Is_at_most_partial_loss (Processor_mask_Copy_status status)

  *Checks if the copy status guarantees at most partial loss.*
- Processor_mask_Copy_status _Processor_mask_Copy (long *dst, size_t dst_size, const long *src, size_t src_size)

  *Copies one mask to another.*
- RTEMS_INLINE_ROUTINE Processor_mask_Copy_status _Processor_mask_To_cpu_set_t (const Processor_mask *src, size_t dst_size, cpu_set_t *dst)

  *Copies one mask to another.*
- RTEMS_INLINE_ROUTINE Processor_mask_Copy_status _Processor_mask_From_cpu_set_t (Processor_mask *dst, size_t src_size, const cpu_set_t *src)

  *Copies one mask to another.*

## Variables

- const Processor_mask **_Processor_mask_The_one_and_only**

### 10.167.1 Detailed Description

Processor Mask API.

# 10.168 cpukit/include/rtems/score/profiling.h File Reference

Profiling Support API.

```
#include <rtems/score/percpu.h>
#include <rtems/score/isrlock.h>
```

## Functions

- static void _Profiling_Thread_dispatch_disable (Per_CPU_Control *cpu, uint32_t previous_thread_dispatch_disable_level)

  *Disables the thread dispatch if the previous thread dispatch disable level is zero.*
- static void _Profiling_Thread_dispatch_disable_critical (Per_CPU_Control *cpu, uint32_t previous_thread_dispatch_disable_level, const ISR_lock_Context *lock_context)

  *Disables the thread dispatch.*
- static void _Profiling_Thread_dispatch_enable (Per_CPU_Control *cpu, uint32_t new_thread_dispatch_disable_level)

  *Enables the thread dispatch.*
- static void _Profiling_Update_max_interrupt_delay (Per_CPU_Control *cpu, CPU_Counter_ticks interrupt_delay)

  *Updates the maximum interrupt delay.*
- void _Profiling_Outer_most_interrupt_entry_and_exit (Per_CPU_Control *cpu, CPU_Counter_ticks interrupt_entry_instant, CPU_Counter_ticks interrupt_exit_instant)

  *Updates the interrupt profiling statistics.*

### 10.168.1 Detailed Description

Profiling Support API.

## 10.169 cpukit/include/rtems/score/protectedheap.h File Reference

Protected Heap Handler API.

```
#include <rtems/score/heapimpl.h>
#include <rtems/score/apimutex.h>
```

### Functions

- static __inline__ uintptr_t _Protected_heap_Initialize (Heap_Control *heap, void *area_begin, uintptr_t area_size, uintptr_t page_size)

  *Initializes the protected heap.*
- bool _Protected_heap_Extend (Heap_Control *heap, void *area_begin, uintptr_t area_size)

  *Extends the protected heap.*
- void * _Protected_heap_Allocate_aligned_with_boundary (Heap_Control *heap, uintptr_t size, uintptr_t alignment, uintptr_t boundary)

  *Allocates an aligned memory area with boundary constraint for the protected heap.*
- static __inline__ void * _Protected_heap_Allocate_aligned (Heap_Control *heap, uintptr_t size, uintptr_t alignment)

  *Allocates an aligned memory area.*
- static __inline__ void * _Protected_heap_Allocate (Heap_Control *heap, uintptr_t size)

  *Allocates a memory area.*
- bool _Protected_heap_Get_block_size (Heap_Control *heap, void *addr, uintptr_t *size)

  *Returns the size of the allocatable memory area.*
- bool _Protected_heap_Resize_block (Heap_Control *heap, void *addr, uintptr_t size)

  *Resizes the block of the allocated memory area.*
- bool _Protected_heap_Free (Heap_Control *heap, void *addr)

  *Frees the allocated memory area.*
- bool _Protected_heap_Walk (Heap_Control *heap, int source, bool dump)

  *Verifies the integrity of the heap.*
- void _Protected_heap_Iterate (Heap_Control *heap, Heap_Block_visitor visitor, void *visitor_arg)

  *Iterates over all blocks of the heap.*
- bool _Protected_heap_Get_information (Heap_Control *heap, Heap_Information_block *info)

  *Returns information about used and free blocks for the heap.*
- bool _Protected_heap_Get_free_information (Heap_Control *heap, Heap_Information *info)

  *Returns information about free blocks for the heap.*
- uintptr_t _Protected_heap_Get_size (Heap_Control *heap)

  *Returns the size of the allocatable area in bytes.*

### 10.169.1 Detailed Description

Protected Heap Handler API.

# 10.170 cpukit/include/rtems/score/rbtree.h File Reference

Constants and Structures Associated with the Red-Black Tree Handler.

```
#include <sys/tree.h>
#include <rtems/score/basedefs.h>
#include <rtems/score/assert.h>
```

## Classes

- struct RBTree_Node

  *Red-black tree node.*

## Macros

- #define RBTREE_INITIALIZER_EMPTY(name) RB_INITIALIZER( name )

  *Initializer for an empty red-black tree with designator name.*
- #define RBTREE_DEFINE_EMPTY(name) RBTree_Control name = RBTREE_INITIALIZER_EMPTY( name )

  *Definition for an empty red-black tree with designator name.*

## Typedefs

- typedef struct RBTree_Node RBTree_Node

  *Red-black tree node.*

## Functions

- typedef RB_HEAD (RBTree_Control, RBTree_Node) RBTree_Control

  *Red-black tree control.*
- static __inline__ void _RBTree_Set_off_tree (RBTree_Node *the_node)

  *Sets a red-black tree node as off-tree.*
- static __inline__ bool _RBTree_Is_node_off_tree (const RBTree_Node *the_node)

  *Checks if this red-black tree node is off-tree.*
- void _RBTree_Insert_color (RBTree_Control *the_rbtree, RBTree_Node *the_node)

  *Rebalances the red-black tree after insertion of the node.*
- static __inline__ void _RBTree_Initialize_node (RBTree_Node *the_node)

  *Initializes a red-black tree node.*
- static __inline__ void _RBTree_Add_child (RBTree_Node *child, RBTree_Node *parent, RBTree_Node **link)

  *Adds a child node to a parent node.*
- static __inline__ void _RBTree_Insert_with_parent (RBTree_Control *the_rbtree, RBTree_Node *the_node, RBTree_Node *parent, RBTree_Node **link)

  *Inserts the node into the red-black tree using the specified parent node and link.*
- void _RBTree_Extract (RBTree_Control *the_rbtree, RBTree_Node *the_node)

  *Extracts (removes) the node from the red-black tree.*
- static __inline__ RBTree_Node * _RBTree_Root (const RBTree_Control *the_rbtree)

*Returns a pointer to root node of the red-black tree.*

- static __inline__ RBTree_Node ** _RBTree_Root_reference (RBTree_Control *the_rbtree)

  *Returns a reference to the root pointer of the red-black tree.*
- static __inline__ RBTree_Node *const * _RBTree_Root_const_reference (const RBTree_Control *the_rbtree)

  *Returns a constant reference to the root pointer of the red-black tree.*
- static __inline__ RBTree_Node * _RBTree_Parent (const RBTree_Node *the_node)

  *Returns a pointer to the parent of this node.*
- static __inline__ RBTree_Node * _RBTree_Left (const RBTree_Node *the_node)

  *Returns pointer to the left of this node.*
- static __inline__ RBTree_Node ** _RBTree_Left_reference (RBTree_Node *the_node)

  *Returns a reference to the left child pointer of the red-black tree node.*
- static __inline__ RBTree_Node * _RBTree_Right (const RBTree_Node *the_node)

  *Returns pointer to the right of this node.*
- static __inline__ RBTree_Node ** _RBTree_Right_reference (RBTree_Node *the_node)

  *Returns a reference to the right child pointer of the red-black tree node.*
- static __inline__ bool _RBTree_Is_empty (const RBTree_Control *the_rbtree)

  *Checks if the RBTree is empty.*
- static __inline__ bool _RBTree_Is_root (const RBTree_Node *the_node)

  *Checks if this node is the root node of a red-black tree.*
- static __inline__ void _RBTree_Initialize_empty (RBTree_Control *the_rbtree)

  *Initializes this RBTree as empty.*
- static __inline__ void _RBTree_Initialize_one (RBTree_Control *the_rbtree, RBTree_Node *the_node)

  *Initializes this red-black tree to contain exactly the specified node.*
- RBTree_Node * _RBTree_Minimum (const RBTree_Control *the_rbtree)

  *Returns the minimum node of the red-black tree.*
- RBTree_Node * _RBTree_Maximum (const RBTree_Control *the_rbtree)

  *Returns the maximum node of the red-black tree.*
- RBTree_Node * _RBTree_Predecessor (const RBTree_Node *node)

  *Returns the predecessor of a node.*
- RBTree_Node * _RBTree_Successor (const RBTree_Node *node)

  *Returns the successor of a node.*
- void _RBTree_Replace_node (RBTree_Control *the_rbtree, RBTree_Node *victim, RBTree_Node *replacement)

  *Replaces a node in the red-black tree without a rebalance.*
- static __inline__ bool _RBTree_Insert_inline (RBTree_Control *the_rbtree, RBTree_Node *the_node, const void *key, bool(*less)(const void *, const RBTree_Node *))

  *Inserts the node into the red-black tree.*
- static __inline__ void * _RBTree_Find_inline (const RBTree_Control *the_rbtree, const void *key, bool(*equal)(const void *, const RBTree_Node *), bool(*less)(const void *, const RBTree_Node *), void *(*map)(RBTree_Node *))

  *Finds an object in the red-black tree with the specified key.*
- void * _RBTree_Postorder_first (const RBTree_Control *the_rbtree, size_t offset)

  *Returns the container of the first node of the specified red-black tree in postorder.*
- void * _RBTree_Postorder_next (const RBTree_Node *the_node, size_t offset)

  *Returns the container of the next node in postorder.*

### 10.170.1 Detailed Description

Constants and Structures Associated with the Red-Black Tree Handler.

This include file contains all the constants and structures associated with the Red-Black Tree Handler.

# 10.171 cpukit/include/rtems/score/rbtreeimpl.h File Reference

Inlined Routines Associated with Red-Black Trees.

```
#include <rtems/score/rbtree.h>
```

## Typedefs

- typedef bool(∗ RBTree_Visitor) (const RBTree_Node ∗node, void ∗visitor_arg)

  *Red-black tree visitor.*

## Functions

- void _RBTree_Iterate (const RBTree_Control ∗rbtree, RBTree_Visitor visitor, void ∗visitor_arg)

  *Red-black tree iteration.*

### 10.171.1 Detailed Description

Inlined Routines Associated with Red-Black Trees.

This include file contains the bodies of the routines which are associated with Red-Black Trees and inlined.

**Note**

The routines in this file are ordered from simple to complex. No other RBTree Handler routine is referenced unless it has already been defined.

# 10.172 cpukit/include/rtems/score/scheduleredf.h File Reference

Data Related to the Manipulation of Threads for the EDF Scheduler.

```
#include <rtems/score/priority.h>
#include <rtems/score/scheduler.h>
#include <rtems/score/schedulerpriority.h>
#include <rtems/score/rbtree.h>
#include <limits.h>
```

## Classes

- struct Scheduler_EDF_Context
- struct Scheduler_EDF_Node

  *Scheduler node specialization for EDF schedulers.*

## Macros

- #define **SCHEDULER_EDF_MAXIMUM_PRIORITY** INT_MAX
- #define SCHEDULER_EDF_ENTRY_POINTS

## Functions

- void _Scheduler_EDF_Initialize (const Scheduler_Control *scheduler)

  *Initializes EDF scheduler.*
- void _Scheduler_EDF_Block (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Removes the blocking thread from the ready queue and schedules is only again if the thread is executing or the heir thread.*
- void _Scheduler_EDF_Schedule (const Scheduler_Control *scheduler, Thread_Control *the_thread)

  *Sets the heir thread to be the next ready thread in the rbtree ready queue.*
- void _Scheduler_EDF_Node_initialize (const Scheduler_Control *scheduler, Scheduler_Node *node, Thread_Control *the_thread, Priority_Control priority)

  *Initializes an EDF specific scheduler node of the_thread.*
- void _Scheduler_EDF_Unblock (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Performs an unblocking of the thread.*
- void _Scheduler_EDF_Update_priority (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Updates the priority of the scheduler node.*
- Priority_Control _Scheduler_EDF_Map_priority (const Scheduler_Control *scheduler, Priority_Control priority)

  *Gets the mapped priority map of the priority control.*
- Priority_Control _Scheduler_EDF_Unmap_priority (const Scheduler_Control *scheduler, Priority_Control priority)

  *Gets the unmapped priority map of the priority control.*
- void _Scheduler_EDF_Yield (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Executes a thread yield for the thread.*
- void _Scheduler_EDF_Release_job (const Scheduler_Control *scheduler, Thread_Control *the_thread, Priority_Node *priority_node, uint64_t deadline, Thread_queue_Context *queue_context)

  *Releases a EDF job.*
- void _Scheduler_EDF_Cancel_job (const Scheduler_Control *scheduler, Thread_Control *the_thread, Priority_Node *priority_node, Thread_queue_Context *queue_context)

  *Cancels a job and removes the thread from the queue context.*

### 10.172.1 Detailed Description

Data Related to the Manipulation of Threads for the EDF Scheduler.

This include file contains all the constants and structures associated with the manipulation of threads for the EDF scheduler.

# 10.173 cpukit/include/rtems/score/scheduleredfimpl.h File Reference

EDF Scheduler Implementation.

```
#include <rtems/score/scheduleredf.h>
#include <rtems/score/schedulerimpl.h>
```

## Macros

- #define SCHEDULER_EDF_PRIO_MSB 0x8000000000000000

## Functions

- static __inline__ Scheduler_EDF_Context * _Scheduler_EDF_Get_context (const Scheduler_Control *scheduler)

  *Gets the context of the scheduler.*
- static __inline__ Scheduler_EDF_Node * _Scheduler_EDF_Thread_get_node (Thread_Control *the_thread)

  *Gets the scheduler EDF node of the thread.*
- static __inline__ Scheduler_EDF_Node * _Scheduler_EDF_Node_downcast (Scheduler_Node *node)

  *Returns the scheduler EDF node for the scheduler node.*
- static __inline__ bool _Scheduler_EDF_Less (const void *left, const RBTree_Node *right)

  *Checks if left is less than the priority of the node right.*
- static __inline__ bool _Scheduler_EDF_Priority_less_equal (const void *left, const RBTree_Node *right)

  *Checks if left is less or equal than the priority of the node right.*
- static __inline__ void _Scheduler_EDF_Enqueue (Scheduler_EDF_Context *context, Scheduler_EDF_Node *node, Priority_Control insert_priority)

  *Inserts the scheduler node with the given priority in the ready queue of the context.*
- static __inline__ void _Scheduler_EDF_Extract (Scheduler_EDF_Context *context, Scheduler_EDF_Node *node)

  *Extracts the scheduler node from the ready queue of the context.*
- static __inline__ void _Scheduler_EDF_Extract_body (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Extracts the node from the context of the given scheduler.*
- static __inline__ void _Scheduler_EDF_Schedule_body (const Scheduler_Control *scheduler, Thread_Control *the_thread, bool force_dispatch)

  *Schedules the next ready thread as the heir.*

### 10.173.1 Detailed Description

EDF Scheduler Implementation.

# 10.174 cpukit/include/rtems/score/scheduleredfsmp.h File Reference

EDF SMP Scheduler API.

```
#include <rtems/score/scheduler.h>
#include <rtems/score/scheduleredf.h>
#include <rtems/score/schedulersmp.h>
```

## Classes

- struct Scheduler_EDF_SMP_Node
- struct Scheduler_EDF_SMP_Ready_queue
- struct Scheduler_EDF_SMP_Context

## Macros

- #define **SCHEDULER_EDF_SMP_ENTRY_POINTS**

## Functions

- void _Scheduler_EDF_SMP_Initialize (const Scheduler_Control *scheduler)

  *Initializes the context of the scheduler control.*
- void _Scheduler_EDF_SMP_Node_initialize (const Scheduler_Control *scheduler, Scheduler_Node *node, Thread_Control *the_thread, Priority_Control priority)

  *Initializes the node with the given priority.*
- void _Scheduler_EDF_SMP_Block (const Scheduler_Control *scheduler, Thread_Control *thread, Scheduler_Node *node)

  *Blocks the thread.*
- void _Scheduler_EDF_SMP_Unblock (const Scheduler_Control *scheduler, Thread_Control *thread, Scheduler_Node *node)

  *Unblocks the thread.*
- void _Scheduler_EDF_SMP_Update_priority (const Scheduler_Control *scheduler, Thread_Control *the_↩ thread, Scheduler_Node *node)

  *Updates the priority of the node.*
- bool _Scheduler_EDF_SMP_Ask_for_help (const Scheduler_Control *scheduler, Thread_Control *the_↩ thread, Scheduler_Node *node)

  *Asks for help operation.*
- void _Scheduler_EDF_SMP_Reconsider_help_request (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Reconsiders help operation.*
- void _Scheduler_EDF_SMP_Withdraw_node (const Scheduler_Control *scheduler, Thread_Control *the_↩ thread, Scheduler_Node *node, Thread_Scheduler_state next_state)

  *Withdraws node operation.*
- void _Scheduler_EDF_SMP_Pin (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node, struct Per_CPU_Control *cpu)

  *Pin thread operation.*
- void _Scheduler_EDF_SMP_Unpin (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node, struct Per_CPU_Control *cpu)

  *Unpin thread operation.*
- void _Scheduler_EDF_SMP_Add_processor (const Scheduler_Control *scheduler, Thread_Control *idle)

  *Adds processor.*
- Thread_Control * _Scheduler_EDF_SMP_Remove_processor (const Scheduler_Control *scheduler, struct Per_CPU_Control *cpu)

  *Removes an idle thread from the given cpu.*
- void _Scheduler_EDF_SMP_Yield (const Scheduler_Control *scheduler, Thread_Control *thread, Scheduler_Node *node)

  *Performs the yield of a thread.*
- void _Scheduler_EDF_SMP_Start_idle (const Scheduler_Control *scheduler, Thread_Control *idle, struct Per_CPU_Control *cpu)

  *Starts an idle thread.*
- bool _Scheduler_EDF_SMP_Set_affinity (const Scheduler_Control *scheduler, Thread_Control *thread, Scheduler_Node *node, const Processor_mask *affinity)

  *Checks if the processor set of the scheduler is the subset of the affinity set.*

### 10.174.1 Detailed Description

EDF SMP Scheduler API.

## 10.175 cpukit/include/rtems/score/schedulerimpl.h File Reference

Inlined Routines Associated with the Manipulation of the Scheduler.

```
#include <rtems/score/scheduler.h>
#include <rtems/score/assert.h>
#include <rtems/score/priorityimpl.h>
#include <rtems/score/smpimpl.h>
#include <rtems/score/status.h>
#include <rtems/score/threadimpl.h>
```

### Typedefs

- typedef Thread_Control *(* Scheduler_Get_idle_thread) (Scheduler_Context *context)

  *Gets an idle thread from the scheduler instance.*
- typedef void(* Scheduler_Release_idle_thread) (Scheduler_Context *context, Thread_Control *idle)

  *Releases an idle thread to the scheduler instance for reuse.*

### Enumerations

- enum Scheduler_Try_to_schedule_action { **SCHEDULER_TRY_TO_SCHEDULE_DO_SCHEDULE**, **SCHEDULER_TRY_TO_SCHEDULE_DO_IDLE_EXCHANGE**, **SCHEDULER_TRY_TO_SCHEDULE_DO_BLOCK** }

  *This enumeration defines what a scheduler should do with a node which could be scheduled.*

### Functions

- void _Scheduler_Handler_initialization (void)

  *Initializes the scheduler to the policy chosen by the user.*
- static __inline__ Scheduler_Context * _Scheduler_Get_context (const Scheduler_Control *scheduler)

  *Gets the context of the scheduler.*
- static __inline__ const Scheduler_Control * _Scheduler_Get_by_CPU (const Per_CPU_Control *cpu)

  *Gets the scheduler for the cpu.*
- static __inline__ void _Scheduler_Acquire_critical (const Scheduler_Control *scheduler, ISR_lock_Context *lock_context)

  *Acquires the scheduler instance inside a critical section (interrupts disabled).*
- static __inline__ void _Scheduler_Release_critical (const Scheduler_Control *scheduler, ISR_lock_Context *lock_context)

  *Releases the scheduler instance inside a critical section (interrupts disabled).*
- static __inline__ bool _Scheduler_Is_non_preempt_mode_supported (const Scheduler_Control *scheduler)

  *Indicate if the thread non-preempt mode is supported by the scheduler.*
- void **_Scheduler_Request_ask_for_help** (Thread_Control *the_thread)
- static __inline__ void _Scheduler_Ask_for_help (Thread_Control *the_thread)

*Registers an ask for help request if necessary.*

- static __inline__ void _Scheduler_Schedule (Thread_Control *the_thread)

  *General scheduling decision.*

- static __inline__ void _Scheduler_Yield (Thread_Control *the_thread)

  *Scheduler yield with a particular thread.*

- static __inline__ void _Scheduler_Block (Thread_Control *the_thread)

  *Blocks a thread with respect to the scheduler.*

- static __inline__ void _Scheduler_Unblock (Thread_Control *the_thread)

  *Unblocks a thread with respect to the scheduler.*

- static __inline__ void _Scheduler_Update_priority (Thread_Control *the_thread)

  *Propagates a priority change of a thread to the scheduler.*

- static __inline__ void _Scheduler_Priority_and_sticky_update (Thread_Control *the_thread, int sticky_level_change)

  *Changes the sticky level of the home scheduler node and propagates a priority change of a thread to the scheduler.*

- static __inline__ Priority_Control _Scheduler_Map_priority (const Scheduler_Control *scheduler, Priority_Control priority)

  *Maps a thread priority from the user domain to the scheduler domain.*

- static __inline__ Priority_Control _Scheduler_Unmap_priority (const Scheduler_Control *scheduler, Priority_Control priority)

  *Unmaps a thread priority from the scheduler domain to the user domain.*

- static __inline__ void _Scheduler_Node_initialize (const Scheduler_Control *scheduler, Scheduler_Node *node, Thread_Control *the_thread, Priority_Control priority)

  *Initializes a scheduler node.*

- static __inline__ void _Scheduler_Node_destroy (const Scheduler_Control *scheduler, Scheduler_Node *node)

  *Destroys a scheduler node.*

- static __inline__ void _Scheduler_Release_job (Thread_Control *the_thread, Priority_Node *priority_node, uint64_t deadline, Thread_queue_Context *queue_context)

  *Releases a job of a thread with respect to the scheduler.*

- static __inline__ void _Scheduler_Cancel_job (Thread_Control *the_thread, Priority_Node *priority_node, Thread_queue_Context *queue_context)

  *Cancels a job of a thread with respect to the scheduler.*

- static __inline__ void _Scheduler_Tick (const Per_CPU_Control *cpu)

  *Scheduler method invoked at each clock tick.*

- static __inline__ void _Scheduler_Start_idle (const Scheduler_Control *scheduler, Thread_Control *the_thread, Per_CPU_Control *cpu)

  *Starts the idle thread for a particular processor.*

- static __inline__ bool _Scheduler_Has_processor_ownership (const Scheduler_Control *scheduler, uint32_t cpu_index)

  *Checks if the scheduler of the cpu with the given index is equal to the given scheduler.*

- static __inline__ const Processor_mask * _Scheduler_Get_processors (const Scheduler_Control *scheduler)

  *Gets the processors of the scheduler.*

- bool _Scheduler_Get_affinity (Thread_Control *the_thread, size_t cpusetsize, cpu_set_t *cpuset)

  *Copies the thread's scheduler's affinity to the given cpuset.*

- static __inline__ bool _Scheduler_default_Set_affinity_body (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node, const Processor_mask *affinity)

  *Checks if the affinity is a subset of the online processors.*

- bool _Scheduler_Set_affinity (Thread_Control *the_thread, size_t cpusetsize, const cpu_set_t *cpuset)

  *Sets the thread's scheduler's affinity.*

- static __inline__ void _Scheduler_Generic_block (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node, void(*extract)(const Scheduler_Control *, Thread_Control *, Scheduler_Node *), void(*schedule)(const Scheduler_Control *, Thread_Control *, bool))

*Blocks the thread.*

- static __inline__ uint32_t _Scheduler_Get_processor_count (const Scheduler_Control *scheduler)

  *Gets the number of processors of the scheduler.*

- static __inline__ Objects_Id _Scheduler_Build_id (uint32_t scheduler_index)

  *Builds an object build id.*

- static __inline__ uint32_t _Scheduler_Get_index_by_id (Objects_Id id)

  *Gets the scheduler index from the given object build id.*

- static __inline__ const Scheduler_Control * _Scheduler_Get_by_id (Objects_Id id)

  *Gets the scheduler from the given object build id.*

- static __inline__ uint32_t _Scheduler_Get_index (const Scheduler_Control *scheduler)

  *Gets the index of the scheduler.*

- static __inline__ void _Scheduler_Thread_change_state (Thread_Control *the_thread, Thread_Scheduler_state new_state)

  *Changes the threads state to the given new state.*

- static __inline__ void _Scheduler_Set_idle_thread (Scheduler_Node *node, Thread_Control *idle)

  *Sets the scheduler node's idle thread.*

- static __inline__ Thread_Control * _Scheduler_Use_idle_thread (Scheduler_Context *context, Scheduler_Node *node, Per_CPU_Control *cpu, Scheduler_Get_idle_thread get_idle_thread)

  *Uses an idle thread for this scheduler node.*

- static __inline__ Scheduler_Try_to_schedule_action _Scheduler_Try_to_schedule_node (Scheduler_Context *context, Scheduler_Node *node, const Thread_Control *idle, Scheduler_Get_idle_thread get_idle_thread)

  *Tries to schedule the scheduler node.*

- static __inline__ Thread_Control * _Scheduler_Release_idle_thread (Scheduler_Context *context, Scheduler_Node *node, Scheduler_Release_idle_thread release_idle_thread)

  *Releases an idle thread using this scheduler node.*

- static __inline__ void _Scheduler_Exchange_idle_thread (Scheduler_Node *needs_idle, Scheduler_Node *uses_idle, Thread_Control *idle)

  *Exchanges an idle thread from the scheduler node that uses it right now to another scheduler node.*

- static __inline__ Per_CPU_Control * _Scheduler_Block_node (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node, bool is_scheduled, Scheduler_Get_idle_thread get_idle_thread)

  *Blocks this scheduler node.*

- static __inline__ void _Scheduler_Discard_idle_thread (Scheduler_Context *context, Thread_Control *the↩ _thread, Scheduler_Node *node, Scheduler_Release_idle_thread release_idle_thread)

  *Discard the idle thread from the scheduler node.*

- static __inline__ bool _Scheduler_Unblock_node (Scheduler_Context *context, Thread_Control *the_thread, Scheduler_Node *node, bool is_scheduled, Scheduler_Release_idle_thread release_idle_thread)

  *Unblocks this scheduler node.*

- static __inline__ void _Scheduler_Update_heir (Thread_Control *new_heir, bool force_dispatch)

  *Updates the heir.*

- static __inline__ Status_Control _Scheduler_Set (const Scheduler_Control *new_scheduler, Thread_Control *the_thread, Priority_Control priority)

  *Sets a new scheduler.*

## 10.175.1 Detailed Description

Inlined Routines Associated with the Manipulation of the Scheduler.

This inline file contains all of the inlined routines associated with the manipulation of the scheduler.

## 10.176 cpukit/include/rtems/score/schedulernode.h File Reference

Handles Scheduler Nodes.

```
#include <rtems/score/basedefs.h>
#include <rtems/score/chain.h>
#include <rtems/score/priority.h>
#include <rtems/score/smplockseq.h>
```

### Classes

- struct Scheduler_Node

  *Scheduler node for per-thread data.*

### Macros

- #define **SCHEDULER_NODE_OF_THREAD_WAIT_NODE**(node) RTEMS_CONTAINER_OF( node, Scheduler_Node, Thread.Wait_node )
- #define **SCHEDULER_NODE_OF_THREAD_SCHEDULER_NODE**(node) RTEMS_CONTAINER_OF( node, Scheduler_Node, Thread.Scheduler_node.Chain )

### Typedefs

- typedef struct Scheduler_Node **Scheduler_Node**

### Enumerations

- enum Scheduler_Node_request { SCHEDULER_NODE_REQUEST_NOT_PENDING, SCHEDULER_NODE_REQUEST_ADD SCHEDULER_NODE_REQUEST_REMOVE, SCHEDULER_NODE_REQUEST_NOTHING }

  *The scheduler node requests.*

### Variables

- const size_t _Scheduler_Node_size

  *The size of a scheduler node.*

### 10.176.1 Detailed Description

Handles Scheduler Nodes.

## 10.177 cpukit/include/rtems/score/schedulernodeimpl.h File Reference

Scheduler Node Implementation.

```
#include <rtems/score/schedulernode.h>
#include <rtems/score/priorityimpl.h>
```

## Macros

- #define **SCHEDULER_NODE_OF_WAIT_PRIORITY_NODE**(node) RTEMS_CONTAINER_OF( node, Scheduler_Node, Wait.Priority.Node.Node.Chain )
- #define **SCHEDULER_NODE_OF_WAIT_PRIORITY**(node) RTEMS_CONTAINER_OF( node, Scheduler_Node, Wait.Priority )
- #define SCHEDULER_PRIORITY_APPEND_FLAG 1

  *Priority append indicator for the priority control used for the scheduler node priority.*
- #define SCHEDULER_PRIORITY_MAP(priority) ( ( priority ) << 1 )

  *Maps a priority value to support the append indicator.*
- #define SCHEDULER_PRIORITY_UNMAP(priority) ( ( priority ) >> 1 )

  *Returns the plain priority value.*
- #define SCHEDULER_PRIORITY_PURIFY(priority) ( ( priority ) & ~( (Priority_Control) SCHEDULER_PRIORITY_APPEND_FL ) )

  *Clears the priority append indicator bit.*
- #define SCHEDULER_PRIORITY_APPEND(priority) ( ( priority ) | SCHEDULER_PRIORITY_APPEND_FLAG )

  *Returns the priority control with the append indicator bit set.*
- #define SCHEDULER_PRIORITY_IS_APPEND(priority) ( ( ( priority ) & SCHEDULER_PRIORITY_APPEND_FLAG ) != 0 )

  *Returns true, if the item should be appended to its priority group, otherwise returns false and the item should be prepended to its priority group.*

## Functions

- static __inline__ void _Scheduler_Node_do_initialize (const struct _Scheduler_Control *scheduler, Scheduler_Node *node, Thread_Control *the_thread, Priority_Control priority)

  *Initializes a node.*
- static __inline__ const Scheduler_Control * _Scheduler_Node_get_scheduler (const Scheduler_Node *node)

  *Gets the scheduler of the node.*
- static __inline__ Thread_Control * _Scheduler_Node_get_owner (const Scheduler_Node *node)

  *Gets the owner of the node.*
- static __inline__ Priority_Control _Scheduler_Node_get_priority (Scheduler_Node *node)

  *Gets the priority of the node.*
- static __inline__ void _Scheduler_Node_set_priority (Scheduler_Node *node, Priority_Control new_priority, bool prepend_it)

  *Sets the priority of the node.*
- static __inline__ Thread_Control * _Scheduler_Node_get_user (const Scheduler_Node *node)

  *Gets the user of the node.*
- static __inline__ void _Scheduler_Node_set_user (Scheduler_Node *node, Thread_Control *user)

  *Sets the user of the node.*
- static __inline__ Thread_Control * _Scheduler_Node_get_idle (const Scheduler_Node *node)

  *Gets the idle thread of the node.*

### 10.177.1 Detailed Description

Scheduler Node Implementation.

## 10.178 cpukit/include/rtems/score/schedulerpriority.h File Reference

Thread Manipulation with the Priority-Based Scheduler.

```
#include <rtems/score/chain.h>
#include <rtems/score/prioritybitmap.h>
#include <rtems/score/scheduler.h>
```

### Classes

- struct Scheduler_priority_Context
- struct Scheduler_priority_Ready_queue

  *Data for ready queue operations.*
- struct Scheduler_priority_Node

  *Scheduler node specialization for Deterministic Priority schedulers.*

### Macros

- #define SCHEDULER_PRIORITY_ENTRY_POINTS

### Functions

- void _Scheduler_priority_Initialize (const Scheduler_Control *scheduler)

  *Initializes the priority scheduler.*
- void _Scheduler_priority_Block (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Blocks the thread.*
- void _Scheduler_priority_Schedule (const Scheduler_Control *scheduler, Thread_Control *the_thread)

  *Sets the heir thread to be the next ready thread.*
- void _Scheduler_priority_Unblock (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Unblocks the thread.*
- void _Scheduler_priority_Update_priority (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *base_node)

  *Updates the priority of the node.*
- void _Scheduler_priority_Node_initialize (const Scheduler_Control *scheduler, Scheduler_Node *node, Thread_Control *the_thread, Priority_Control priority)

  *Initializes the node with the given priority.*
- void _Scheduler_priority_Yield (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Performs the yield of a thread.*

### 10.178.1 Detailed Description

Thread Manipulation with the Priority-Based Scheduler.

This include file contains all the constants and structures associated with the manipulation of threads for the priority-based scheduler.

# 10.179 cpukit/include/rtems/score/schedulerpriorityimpl.h File Reference

Inlined Routines Associated with the Manipulation of the Priority-Based Scheduling Structures.

```
#include <rtems/score/schedulerpriority.h>
#include <rtems/score/chainimpl.h>
#include <rtems/score/prioritybitmapimpl.h>
#include <rtems/score/schedulerimpl.h>
#include <rtems/score/thread.h>
```

## Functions

- static __inline__ Scheduler_priority_Context * _Scheduler_priority_Get_context (const Scheduler_Control *scheduler)

  *Gets the context of the scheduler.*
- static __inline__ Scheduler_priority_Node * _Scheduler_priority_Thread_get_node (Thread_Control *the↩ _thread)

  *Gets the scheduler node of the thread.*
- static __inline__ Scheduler_priority_Node * _Scheduler_priority_Node_downcast (Scheduler_Node *node)

  *Gets the priority node of the scheduler node.*
- static __inline__ void _Scheduler_priority_Ready_queue_initialize (Chain_Control *ready_queues, Priority_Control maximum_priority)

  *Ready queue initialization.*
- static __inline__ void _Scheduler_priority_Ready_queue_enqueue (Chain_Node *node, Scheduler_priority_Ready_queue *ready_queue, Priority_bit_map_Control *bit_map)

  *Enqueues a node on the specified ready queue.*
- static __inline__ void _Scheduler_priority_Ready_queue_enqueue_first (Chain_Node *node, Scheduler_priority_Ready_queue *ready_queue, Priority_bit_map_Control *bit_map)

  *Enqueues a node on the specified ready queue as first.*
- static __inline__ void _Scheduler_priority_Ready_queue_extract (Chain_Node *node, Scheduler_priority_Ready_queue *ready_queue, Priority_bit_map_Control *bit_map)

  *Extracts a node from the specified ready queue.*
- static __inline__ void _Scheduler_priority_Extract_body (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Extracts a node from the context of the scheduler.*
- static __inline__ Chain_Node * _Scheduler_priority_Ready_queue_first (Priority_bit_map_Control *bit_map, Chain_Control *ready_queues)

  *Returns a pointer to the first node.*
- static __inline__ void _Scheduler_priority_Schedule_body (const Scheduler_Control *scheduler, Thread_Control *the_thread, bool force_dispatch)

  *Scheduling decision logic.*
- static __inline__ void _Scheduler_priority_Ready_queue_update (Scheduler_priority_Ready_queue *ready_queue, unsigned int new_priority, Priority_bit_map_Control *bit_map, Chain_Control *ready_↩ queues)

  *Updates the specified ready queue data according to the new priority value.*

### 10.179.1 Detailed Description

Inlined Routines Associated with the Manipulation of the Priority-Based Scheduling Structures.

This inline file contains all of the inlined routines associated with the manipulation of the priority-based scheduling structures.

# 10.180 cpukit/include/rtems/score/schedulersimple.h File Reference

Manipulation of Threads Simple-Priority-Based Ready Queue.

```
#include <rtems/score/scheduler.h>
#include <rtems/score/schedulerpriority.h>
```

## Classes

- struct Scheduler_simple_Context

  *Simple scheduler context.*

## Macros

- #define **SCHEDULER_SIMPLE_MAXIMUM_PRIORITY** 255
- #define SCHEDULER_SIMPLE_ENTRY_POINTS

## Functions

- void _Scheduler_simple_Initialize (const Scheduler_Control *scheduler)

  *Initializes simple scheduler.*
- void _Scheduler_simple_Schedule (const Scheduler_Control *scheduler, Thread_Control *the_thread)

  *Schedules threads.*
- void _Scheduler_simple_Yield (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Performs the yield of a thread.*
- void _Scheduler_simple_Block (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Blocks the thread.*
- void _Scheduler_simple_Unblock (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Unblocks the thread.*
- void _Scheduler_simple_Update_priority (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Updates the priority of the node.*

### 10.180.1 Detailed Description

Manipulation of Threads Simple-Priority-Based Ready Queue.

This include file contains all the constants and structures associated with the manipulation of threads on a simple-priority-based ready queue.

# 10.181 cpukit/include/rtems/score/schedulersimpleimpl.h File Reference

Inlined Routines Associated with the Manipulation of the Priority-Based Scheduling Structures.

```
#include <rtems/score/schedulersimple.h>
#include <rtems/score/chainimpl.h>
#include <rtems/score/schedulerimpl.h>
```

## Functions

- static __inline__ Scheduler_simple_Context * _Scheduler_simple_Get_context (const Scheduler_Control *scheduler)

  *Gets context of the scheduler.*
- static __inline__ bool _Scheduler_simple_Priority_less_equal (const void *to_insert, const Chain_Node *next)

  *Checks if the priority is less or equal than the priority of the node.*
- static __inline__ void _Scheduler_simple_Insert (Chain_Control *chain, Thread_Control *to_insert, unsigned int insert_priority)

  *Inserts the thread control with the given priority into the chain.*
- static __inline__ void _Scheduler_simple_Extract (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Extracts the threads node.*
- static __inline__ void _Scheduler_simple_Schedule_body (const Scheduler_Control *scheduler, Thread_Control *the_thread, bool force_dispatch)

  *Scheduling decision logic.*

### 10.181.1 Detailed Description

Inlined Routines Associated with the Manipulation of the Priority-Based Scheduling Structures.

This inline file contains all of the inlined routines associated with the manipulation of the priority-based scheduling structures.

# 10.182 cpukit/include/rtems/score/schedulersmp.h File Reference

SMP Scheduler API.

```
#include <rtems/score/chain.h>
#include <rtems/score/scheduler.h>
```

## Classes

- struct Scheduler_SMP_Context

  *Scheduler context specialization for SMP schedulers.*
- struct Scheduler_SMP_Node

  *Scheduler node specialization for SMP schedulers.*

### Enumerations

- enum Scheduler_SMP_Node_state { SCHEDULER_SMP_NODE_BLOCKED, SCHEDULER_SMP_NODE_SCHEDULED, SCHEDULER_SMP_NODE_READY }

  *SMP scheduler node states.*

### Functions

- void _Scheduler_SMP_Start_idle (const Scheduler_Control *scheduler, Thread_Control *idle, struct Per_CPU_Control *cpu)

  *Starts an idle thread on the specified cpu.*

#### 10.182.1 Detailed Description

SMP Scheduler API.

## 10.183 cpukit/include/rtems/score/schedulersmpimpl.h File Reference

SMP Scheduler Implementation.

```
#include <rtems/score/schedulersmp.h>
#include <rtems/score/assert.h>
#include <rtems/score/chainimpl.h>
#include <rtems/score/schedulersimpleimpl.h>
#include <rtems/bspIo.h>
```

### Typedefs

- typedef bool(* **Scheduler_SMP_Has_ready**) (Scheduler_Context *context)
- typedef Scheduler_Node *(* **Scheduler_SMP_Get_highest_ready**) (Scheduler_Context *context, Scheduler_Node *node)
- typedef Scheduler_Node *(* **Scheduler_SMP_Get_lowest_scheduled**) (Scheduler_Context *context, Scheduler_Node *filter)
- typedef void(* **Scheduler_SMP_Extract**) (Scheduler_Context *context, Scheduler_Node *node_to_extract)
- typedef void(* **Scheduler_SMP_Insert**) (Scheduler_Context *context, Scheduler_Node *node_to_insert, Priority_Control insert_priority)
- typedef void(* **Scheduler_SMP_Move**) (Scheduler_Context *context, Scheduler_Node *node_to_move)
- typedef bool(* **Scheduler_SMP_Ask_for_help**) (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node)
- typedef void(* **Scheduler_SMP_Update**) (Scheduler_Context *context, Scheduler_Node *node_to_update, Priority_Control new_priority)
- typedef void(* **Scheduler_SMP_Set_affinity**) (Scheduler_Context *context, Scheduler_Node *node, void *arg)
- typedef bool(* **Scheduler_SMP_Enqueue**) (Scheduler_Context *context, Scheduler_Node *node_to_↩ enqueue, Priority_Control priority)
- typedef void(* **Scheduler_SMP_Allocate_processor**) (Scheduler_Context *context, Scheduler_Node *scheduled, Scheduler_Node *victim, Per_CPU_Control *victim_cpu)
- typedef void(* **Scheduler_SMP_Register_idle**) (Scheduler_Context *context, Scheduler_Node *idle, Per_CPU_Control *cpu)

## Functions

- static void _Scheduler_SMP_Do_nothing_register_idle (Scheduler_Context *context, Scheduler_Node *idle, Per_CPU_Control *cpu)

  *Does nothing.*
- static bool _Scheduler_SMP_Priority_less_equal (const void *to_insert, const Chain_Node *next)

  *Checks if to_insert is less or equal than the priority of the chain node.*
- static Scheduler_SMP_Context * _Scheduler_SMP_Get_self (Scheduler_Context *context)

  *Gets the scheduler smp context.*
- static void _Scheduler_SMP_Initialize (Scheduler_SMP_Context *self)

  *Initializes the scheduler smp context.*
- static Scheduler_SMP_Node * _Scheduler_SMP_Thread_get_node (Thread_Control *thread)

  *Gets the scheduler smp node of the thread.*
- static Scheduler_SMP_Node * _Scheduler_SMP_Thread_get_own_node (Thread_Control *thread)

  *Gets the scheduler smp node of the thread.*
- static Scheduler_SMP_Node * _Scheduler_SMP_Node_downcast (Scheduler_Node *node)

  *Gets the scheduler smp node.*
- static Scheduler_SMP_Node_state _Scheduler_SMP_Node_state (const Scheduler_Node *node)

  *Gets the state of the node.*
- static Priority_Control _Scheduler_SMP_Node_priority (const Scheduler_Node *node)

  *Gets the priority of the node.*
- static void _Scheduler_SMP_Node_initialize (const Scheduler_Control *scheduler, Scheduler_SMP_Node *node, Thread_Control *thread, Priority_Control priority)

  *Initializes the scheduler smp node.*
- static void _Scheduler_SMP_Node_update_priority (Scheduler_SMP_Node *node, Priority_Control new_priority)

  *Updates the priority of the node to the new priority.*
- static void _Scheduler_SMP_Node_change_state (Scheduler_Node *node, Scheduler_SMP_Node_state new_state)

  *Changes the state of the node to the given state.*
- static bool _Scheduler_SMP_Is_processor_owned_by_us (const Scheduler_Context *context, const Per_CPU_Control *cpu)

  *Checks if the processor is owned by the given context.*
- static Thread_Control * _Scheduler_SMP_Get_idle_thread (Scheduler_Context *context)

  *Gets The first idle thread of the given context.*
- static void _Scheduler_SMP_Release_idle_thread (Scheduler_Context *context, Thread_Control *idle)

  *Releases the thread and adds it to the idle threads.*
- static void _Scheduler_SMP_Exctract_idle_thread (Thread_Control *idle)

  *Extracts the node of the idle thread.*
- static void _Scheduler_SMP_Allocate_processor_lazy (Scheduler_Context *context, Scheduler_Node *scheduled, Scheduler_Node *victim, Per_CPU_Control *victim_cpu)

  *Allocates the cpu for the scheduled thread.*
- static void _Scheduler_SMP_Allocate_processor_exact (Scheduler_Context *context, Scheduler_Node *scheduled, Scheduler_Node *victim, Per_CPU_Control *victim_cpu)

  *Allocates the cpu for the scheduled thread.*
- static void _Scheduler_SMP_Allocate_processor (Scheduler_Context *context, Scheduler_Node *scheduled, Scheduler_Node *victim, Per_CPU_Control *victim_cpu, Scheduler_SMP_Allocate_processor allocate_processor)

  *Allocates the cpu for the scheduled thread using the given allocation function.*
- static Thread_Control * _Scheduler_SMP_Preempt (Scheduler_Context *context, Scheduler_Node *scheduled, Scheduler_Node *victim, Scheduler_SMP_Allocate_processor allocate_processor)

  *Preempts the victim's thread and allocates a cpu for the scheduled thread.*

- static Scheduler_Node * _Scheduler_SMP_Get_lowest_scheduled (Scheduler_Context *context, Scheduler_Node *filter)

  *Returns the lowest member of the scheduled nodes.*
- static void _Scheduler_SMP_Enqueue_to_scheduled (Scheduler_Context *context, Scheduler_Node *node, Priority_Control priority, Scheduler_Node *lowest_scheduled, Scheduler_SMP_Insert insert_scheduled, Scheduler_SMP_Move move_from_scheduled_to_ready, Scheduler_SMP_Allocate_processor allocate_processor)

  *Tries to schedule the given node.*
- static bool _Scheduler_SMP_Enqueue (Scheduler_Context *context, Scheduler_Node *node, Priority_Control insert_priority, Chain_Node_order order, Scheduler_SMP_Insert insert_ready, Scheduler_SMP_Insert insert_scheduled, Scheduler_SMP_Move move_from_scheduled_to_ready, Scheduler_SMP_Get_lowest_scheduled get_lowest_scheduled, Scheduler_SMP_Allocate_processor allocate_processor)

  *Enqueues a node according to the specified order function.*
- static bool _Scheduler_SMP_Enqueue_scheduled (Scheduler_Context *context, Scheduler_Node *const node, Priority_Control insert_priority, Chain_Node_order order, Scheduler_SMP_Extract extract_from_ready, Scheduler_SMP_Get_highest_ready get_highest_ready, Scheduler_SMP_Insert insert_ready, Scheduler_SMP_Insert insert_scheduled, Scheduler_SMP_Move move_from_ready_to_scheduled, Scheduler_SMP_Allocate_processor allocate_processor)

  *Enqueues a scheduled node according to the specified order function.*
- static void _Scheduler_SMP_Extract_from_scheduled (Scheduler_Context *context, Scheduler_Node *node)

  *Extracts a scheduled node from the scheduled nodes.*
- static void _Scheduler_SMP_Schedule_highest_ready (Scheduler_Context *context, Scheduler_Node *victim, Per_CPU_Control *victim_cpu, Scheduler_SMP_Extract extract_from_ready, Scheduler_SMP_Get_highest_ready get_highest_ready, Scheduler_SMP_Move move_from_ready_to_scheduled, Scheduler_SMP_Allocate_processor allocate_processor)

  *Schedules the highest ready node.*
- static void _Scheduler_SMP_Preempt_and_schedule_highest_ready (Scheduler_Context *context, Scheduler_Node *victim, Per_CPU_Control *victim_cpu, Scheduler_SMP_Extract extract_from_ready, Scheduler_SMP_Get_highest_ready get_highest_ready, Scheduler_SMP_Move move_from_ready_to_scheduled, Scheduler_SMP_Allocate_processor allocate_processor)

  *Schedules the highest ready node and preempts a currently executing one.*
- static void _Scheduler_SMP_Block (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node, Scheduler_SMP_Extract extract_from_scheduled, Scheduler_SMP_Extract extract_from_ready, Scheduler_SMP_Get_highest_ready get_highest_ready, Scheduler_SMP_Move move_from_ready_to_scheduled, Scheduler_SMP_Allocate_processor allocate_processor)

  *Blocks the thread.*
- static void _Scheduler_SMP_Unblock (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node, Scheduler_SMP_Update update, Scheduler_SMP_Enqueue enqueue)

  *Unblocks the thread.*
- static void _Scheduler_SMP_Update_priority (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node, Scheduler_SMP_Extract extract_from_ready, Scheduler_SMP_Update update, Scheduler_SMP_Enqueue enqueue, Scheduler_SMP_Enqueue enqueue_scheduled, Scheduler_SMP_Ask_for_help ask_for_help)

  *Updates the priority of the node and the position in the queues it is in.*
- static void _Scheduler_SMP_Yield (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node, Scheduler_SMP_Extract extract_from_ready, Scheduler_SMP_Enqueue enqueue, Scheduler_SMP_Enqueue enqueue_scheduled)

  *Performs a yield and asks for help if necessary.*
- static void _Scheduler_SMP_Insert_scheduled (Scheduler_Context *context, Scheduler_Node *node_to_insert, Priority_Control priority_to_insert)

  *Inserts the node with the given priority into the scheduled nodes.*
- static bool _Scheduler_SMP_Ask_for_help (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node, Chain_Node_order order, Scheduler_SMP_Insert insert_ready, Scheduler_SMP_Insert insert_scheduled, Scheduler_SMP_Move move_from_scheduled_to_ready, Scheduler_SMP_Get_lowest_scheduled get_lowest_scheduled, Scheduler_SMP_Allocate_processor allocate_processor)

*Asks for help.*

- static void _Scheduler_SMP_Reconsider_help_request (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node, Scheduler_SMP_Extract extract_from_ready)

  *Reconsiders help request.*

- static void _Scheduler_SMP_Withdraw_node (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node, Thread_Scheduler_state next_state, Scheduler_SMP_Extract extract_from_ready, Scheduler_SMP_Get_highest_ready get_highest_ready, Scheduler_SMP_Move move_from_ready_to_scheduled, Scheduler_SMP_Allocate_processor allocate_processor)

  *Withdraws the node.*

- static void _Scheduler_SMP_Do_start_idle (Scheduler_Context *context, Thread_Control *idle, Per_CPU_Control *cpu, Scheduler_SMP_Register_idle register_idle)

  *Starts the idle thread on the given processor.*

- static void _Scheduler_SMP_Add_processor (Scheduler_Context *context, Thread_Control *idle, Scheduler_SMP_Has_ready has_ready, Scheduler_SMP_Enqueue enqueue_scheduled, Scheduler_SMP_Register_idle register_idle)

  *Adds the idle thread to the processor.*

- static Thread_Control * _Scheduler_SMP_Remove_processor (Scheduler_Context *context, Per_CPU_Control *cpu, Scheduler_SMP_Extract extract_from_ready, Scheduler_SMP_Enqueue enqueue)

  *Removes an idle thread from the processor.*

- static void _Scheduler_SMP_Set_affinity (Scheduler_Context *context, Thread_Control *thread, Scheduler_Node *node, void *arg, Scheduler_SMP_Set_affinity set_affinity, Scheduler_SMP_Extract extract_from_ready, Scheduler_SMP_Get_highest_ready get_highest_ready, Scheduler_SMP_Move move_from_ready_to_scheduled, Scheduler_SMP_Enqueue enqueue, Scheduler_SMP_Allocate_processor allocate_processor)

  *Sets the affinity of the node.*

### 10.183.1 Detailed Description

SMP Scheduler Implementation.

## 10.184 cpukit/include/rtems/score/semaphoreimpl.h File Reference

Semaphore Implementation.

```
#include <sys/lock.h>
#include <rtems/score/percpu.h>
#include <rtems/score/threadqimpl.h>
```

### Classes

- struct Sem_Control

### Macros

- #define **SEMAPHORE_TQ_OPERATIONS** &_Thread_queue_Operations_priority

### Functions

- static Sem_Control * _Sem_Get (struct _Semaphore_Control *_sem)

  *Gets the Sem_Control * of the semaphore.*
- static Thread_Control * _Sem_Queue_acquire_critical (Sem_Control *sem, Thread_queue_Context *queue_context)

  *Acquires the semaphore queue critical.*
- static void _Sem_Queue_release (Sem_Control *sem, ISR_Level level, Thread_queue_Context *queue_↩ context)

  *Releases the semaphore queue.*

#### 10.184.1 Detailed Description

Semaphore Implementation.

## 10.185 cpukit/include/rtems/score/smp.h File Reference

SuperCore SMP Support API.

```
#include <rtems/score/cpu.h>
```

### Functions

- static uint32_t **_SMP_Get_processor_maximum** (void)
- static uint32_t **_SMP_Get_current_processor** (void)

### Variables

- const uint32_t _SMP_Processor_configured_maximum

  *The configured processor maximum.*
- uint32_t **_SMP_Processor_maximum**

#### 10.185.1 Detailed Description

SuperCore SMP Support API.

## 10.186 cpukit/include/rtems/score/smpbarrier.h File Reference

SMP Barrier API.

```
#include <rtems/score/cpuopts.h>
#include <rtems/score/atomic.h>
```

### Classes

- struct SMP_barrier_Control

  *SMP barrier control.*
- struct SMP_barrier_State

  *SMP barrier per-thread state.*

### Macros

- #define SMP_BARRIER_CONTROL_INITIALIZER { ATOMIC_INITIALIZER_UINT( 0U ), ATOMIC_INITIA↩ LIZER_UINT( 0U ) }

  *SMP barrier control initializer for static initialization.*
- #define SMP_BARRIER_STATE_INITIALIZER { 0U }

  *SMP barrier per-thread state initializer for static initialization.*

### Functions

- static void _SMP_barrier_Control_initialize (SMP_barrier_Control *control)

  *Initializes a SMP barrier control.*
- static void _SMP_barrier_State_initialize (SMP_barrier_State *state)

  *Initializes a SMP barrier per-thread state.*
- bool _SMP_barrier_Wait (SMP_barrier_Control *control, SMP_barrier_State *state, unsigned int count)

  *Waits on the SMP barrier until count threads rendezvoused.*

#### 10.186.1 Detailed Description

SMP Barrier API.

## 10.187 cpukit/include/rtems/score/smpimpl.h File Reference

SuperCore SMP Implementation.

```
#include <rtems/score/smp.h>
#include <rtems/score/percpu.h>
#include <rtems/score/processormask.h>
#include <rtems/fatal.h>
```

### Macros

- #define SMP_MESSAGE_SHUTDOWN 0x1UL

  *SMP message to request a processor shutdown.*
- #define SMP_MESSAGE_PERFORM_JOBS 0x2UL

  *SMP message to perform per-processor jobs.*

## Typedefs

- typedef void(∗ **SMP_Action_handler**) (void ∗arg)

## Enumerations

- enum SMP_Fatal_code {
  **SMP_FATAL_BOOT_PROCESSOR_NOT_ASSIGNED_TO_SCHEDULER**, **SMP_FATAL_MANDATOR↩ Y_PROCESSOR_NOT_PRESENT**, **SMP_FATAL_MULTITASKING_START_ON_INVALID_PROCESS↩ OR**, **SMP_FATAL_MULTITASKING_START_ON_UNASSIGNED_PROCESSOR**, **SMP_FATAL_SHUTDOWN**, **SMP_FATAL_SHUTDOWN_RESPONSE**, **SMP_FATAL_START_OF_MAN↩ DATORY_PROCESSOR_FAILED**, **SMP_FATAL_SCHEDULER_PIN_OR_UNPIN_NOT_SUPPORTED**, **SMP_FATAL_WRONG_CPU_STATE_TO_PERFORM_JOBS** }

  *SMP fatal codes.*

## Functions

- static void _SMP_Fatal (SMP_Fatal_code code)

  *Terminates with the given code.*
- void _SMP_Handler_initialize (void)

  *Initializes SMP Handler.*
- RTEMS_NO_RETURN void _SMP_Start_multitasking_on_secondary_processor (Per_CPU_Control ∗cpu↩ _self)

  *Performs high-level initialization of a secondary processor and runs the application threads.*
- static long unsigned _SMP_Inter_processor_interrupt_handler (Per_CPU_Control ∗cpu_self)

  *Interrupts handler for inter-processor interrupts.*
- bool _SMP_Should_start_processor (uint32_t cpu_index)

  *Checks if the processor with the specified index should be started.*
- void _SMP_Send_message (uint32_t cpu_index, unsigned long message)

  *Sends an SMP message to a processor.*
- void _SMP_Send_message_broadcast (unsigned long message)

  *Sends an SMP message to all other online processors.*
- void _SMP_Send_message_multicast (const Processor_mask ∗targets, unsigned long message)

  *Sends an SMP message to a set of processors.*
- void _SMP_Multicast_action (const Processor_mask ∗targets, SMP_Action_handler handler, void ∗arg)

  *Initiates an SMP multicast action to the set of target processors.*
- void _SMP_Broadcast_action (SMP_Action_handler handler, void ∗arg)

  *Initiates an SMP multicast action to the set of all online processors.*
- void _SMP_Othercast_action (SMP_Action_handler handler, void ∗arg)

  *Initiates an SMP multicast action to the set of all online processors excluding the current processor.*
- void _SMP_Unicast_action (uint32_t cpu_index, SMP_Action_handler handler, void ∗arg)

  *Initiates an SMP action on the specified target processor.*
- void _SMP_Synchronize (void)

  *Ensures that all store operations issued by the current processor before the call this function are visible to all other online processors.*
- void _SMP_Request_start_multitasking (void)

  *Requests a multitasking start on all configured and available processors.*
- void _SMP_Request_shutdown (void)

  *Requests a shutdown of all processors.*
- static __inline__ const Processor_mask ∗ _SMP_Get_online_processors (void)

  *Gets all online processors.*
- static __inline__ const bool _SMP_Need_inter_processor_interrupts (void)

  *Indicate if inter-processor interrupts are needed.*

### Variables

- Processor_mask _SMP_Online_processors

  *Set of online processors.*

### 10.187.1 Detailed Description

SuperCore SMP Implementation.

## 10.188 cpukit/include/rtems/score/smplock.h File Reference

SMP Lock API.

```
#include <rtems/score/cpuopts.h>
#include <rtems/score/smplockstats.h>
#include <rtems/score/smplockticket.h>
#include <rtems/score/isrlevel.h>
```

### Classes

- struct SMP_lock_Control

  *SMP lock control.*
- struct SMP_lock_Context

  *Local SMP lock context for acquire and release pairs.*

### Macros

- #define SMP_LOCK_INITIALIZER(name) { SMP_TICKET_LOCK_INITIALIZER }

  *SMP lock control initializer for static initialization.*
- #define _SMP_lock_Initialize(lock, name) _SMP_lock_Initialize_inline( lock, name )

  *Initializes an SMP lock.*
- #define _SMP_lock_Destroy(lock) _SMP_lock_Destroy_inline( lock )

  *Destroys an SMP lock.*
- #define _SMP_lock_Release(lock, context) _SMP_lock_Release_inline( lock, context )

  *Releases an SMP lock.*
- #define _SMP_lock_Release_and_ISR_enable(lock, context) _SMP_lock_Release_and_ISR_enable_inline( lock, context )

  *Releases the SMP lock and enables interrupts.*

### Functions

- static void _SMP_lock_Initialize_inline (SMP_lock_Control *lock, const char *name)

  *Initializes the SMP lock with the given name.*
- static void _SMP_lock_Destroy_inline (SMP_lock_Control *lock)

  *Destroys the SMP lock.*
- static void _SMP_lock_Set_name (SMP_lock_Control *lock, const char *name)

  *Sets the name of an SMP lock.*
- static void _SMP_lock_Acquire_inline (SMP_lock_Control *lock, SMP_lock_Context *context)

  *Gets my index.*
- void _SMP_lock_Acquire (SMP_lock_Control *lock, SMP_lock_Context *context)

  *Acquires an SMP lock.*
- static void _SMP_lock_Release_inline (SMP_lock_Control *lock, SMP_lock_Context *context)

  *Releases an SMP lock.*
- static void _SMP_lock_ISR_disable_and_acquire_inline (SMP_lock_Control *lock, SMP_lock_Context *context)

  *Disables interrupts and acquires the SMP lock.*
- void _SMP_lock_ISR_disable_and_acquire (SMP_lock_Control *lock, SMP_lock_Context *context)

  *Disables interrupts and acquires the SMP lock.*
- static void _SMP_lock_Release_and_ISR_enable_inline (SMP_lock_Control *lock, SMP_lock_Context *context)

  *Releases the SMP lock and enables interrupts.*

#### 10.188.1 Detailed Description

SMP Lock API.

## 10.189 cpukit/include/rtems/score/smplockmcs.h File Reference

SMP Lock API.

```
#include <rtems/score/cpuopts.h>
#include <rtems/score/atomic.h>
#include <rtems/score/smplockstats.h>
```

### Classes

- struct SMP_MCS_lock_Context

  *SMP Mellor-Crummey and Scott (MCS) lock context.*
- struct SMP_MCS_lock_Control

  *SMP Mellor-Crummey and Scott (MCS) lock control.*

### Macros

- #define SMP_MCS_LOCK_INITIALIZER { { ATOMIC_INITIALIZER_UINTPTR( 0 ) } }

  *SMP MCS lock control initializer for static initialization.*
- #define _SMP_MCS_lock_Acquire(lock, context, stats) _SMP_MCS_lock_Do_acquire( lock, context )

  *Acquires an SMP MCS lock.*

### Typedefs

- typedef struct SMP_MCS_lock_Context SMP_MCS_lock_Context

  *SMP Mellor-Crummey and Scott (MCS) lock context.*

### Functions

- static void _SMP_MCS_lock_Initialize (SMP_MCS_lock_Control *lock)

  *Initializes the SMP MCS lock.*
- static void _SMP_MCS_lock_Destroy (SMP_MCS_lock_Control *lock)

  *Destroys the SMP MCS lock.*
- static void _SMP_MCS_lock_Do_acquire (SMP_MCS_lock_Control *lock, SMP_MCS_lock_Context *context)

  *Acquires the SMP MCS lock.*
- static void _SMP_MCS_lock_Release (SMP_MCS_lock_Control *lock, SMP_MCS_lock_Context *context)

  *Releases an SMP MCS lock.*

#### 10.189.1 Detailed Description

SMP Lock API.

## 10.190 cpukit/include/rtems/score/smplockseq.h File Reference

SMP Lock API.

```
#include <rtems/score/cpuopts.h>
#include <rtems/score/assert.h>
#include <rtems/score/atomic.h>
```

### Classes

- struct SMP_sequence_lock_Control

  *SMP sequence lock control.*

### Macros

- #define SMP_SEQUENCE_LOCK_INITIALIZER { ATOMIC_INITIALIZER_UINT( 0 ) }

  *SMP sequence lock control initializer for static initialization.*

### Functions

- static void _SMP_sequence_lock_Initialize (SMP_sequence_lock_Control *lock)

  *Initializes an SMP sequence lock.*
- static void _SMP_sequence_lock_Destroy (SMP_sequence_lock_Control *lock)

  *Destroys an SMP sequence lock.*
- static unsigned int _SMP_sequence_lock_Write_begin (SMP_sequence_lock_Control *lock)

  *Begins an SMP sequence lock write operation.*
- static void _SMP_sequence_lock_Write_end (SMP_sequence_lock_Control *lock, unsigned int seq)

  *Ends an SMP sequence lock write operation.*
- static unsigned int _SMP_sequence_lock_Read_begin (const SMP_sequence_lock_Control *lock)

  *Begins an SMP sequence lock read operation.*
- static bool _SMP_sequence_lock_Read_retry (SMP_sequence_lock_Control *lock, unsigned int seq)

  *Ends an SMP sequence lock read operation and indicates if a retry is necessary.*

#### 10.190.1 Detailed Description

SMP Lock API.

## 10.191 cpukit/include/rtems/score/smplockstats.h File Reference

SMP Lock API.

```
#include <rtems/score/cpu.h>
#include <rtems/score/chain.h>
```

### Macros

- #define **_SMP_lock_Stats_initialize**(stats, name) do { } while ( 0 )
- #define **_SMP_lock_Stats_destroy**(stats) do { } while ( 0 )

#### 10.191.1 Detailed Description

SMP Lock API.

## 10.192 cpukit/include/rtems/score/smplockticket.h File Reference

SMP Lock API.

```
#include <rtems/score/cpuopts.h>
#include <rtems/score/atomic.h>
#include <rtems/score/smplockstats.h>
```

### Classes

- struct SMP_ticket_lock_Control

  *SMP ticket lock control.*

### Macros

- #define SMP_TICKET_LOCK_INITIALIZER

  *SMP ticket lock control initializer for static initialization.*
- #define _SMP_ticket_lock_Acquire(lock, stats, stats_context) _SMP_ticket_lock_Do_acquire( lock )

  *Acquires an SMP ticket lock.*
- #define _SMP_ticket_lock_Release(lock, stats_context) _SMP_ticket_lock_Do_release( lock )

  *Releases an SMP ticket lock.*

### Functions

- static void _SMP_ticket_lock_Initialize (SMP_ticket_lock_Control *lock)

  *Initializes the SMP ticket lock.*
- static void _SMP_ticket_lock_Destroy (SMP_ticket_lock_Control *lock)

  *Destroys the SMP ticket lock.*
- static void _SMP_ticket_lock_Do_acquire (SMP_ticket_lock_Control *lock)

  *Acquires the SMP ticket lock.*
- static void _SMP_ticket_lock_Do_release (SMP_ticket_lock_Control *lock)

  *Releases the SMP ticket lock.*

#### 10.192.1 Detailed Description

SMP Lock API.

## 10.193 cpukit/include/rtems/score/stack.h File Reference

Information About the Thread Stack Handler.

```
#include <rtems/score/basedefs.h>
```

### Classes

- struct Stack_Control

### Macros

- #define STACK_MINIMUM_SIZE CPU_STACK_MINIMUM_SIZE

### Typedefs

- typedef void(∗ Stack_Allocator_initialize) (size_t stack_space_size)

  *The stack allocator initialization handler.*
- typedef void ∗(∗ Stack_Allocator_allocate) (size_t stack_size)

  *Stack allocator allocate handler.*
- typedef void(∗ Stack_Allocator_free) (void ∗addr)

  *Stack allocator free handler.*

### Functions

- void _Stack_Allocator_do_initialize (void)

  *Do the stack allocator initialization during system initialize.*

### Variables

- uint32_t rtems_minimum_stack_size

  *The minimum stack size.*
- const uintptr_t _Stack_Space_size

  *The configured stack space size.*
- const bool _Stack_Allocator_avoids_workspace

  *Indicates if the stack allocator avoids the workspace.*
- const Stack_Allocator_initialize _Stack_Allocator_initialize

  *The stack allocator initialization handler.*
- const Stack_Allocator_allocate _Stack_Allocator_allocate

  *The stack allocator allocate handler.*
- const Stack_Allocator_free _Stack_Allocator_free

  *The stack allocator free handler.*

#### 10.193.1 Detailed Description

Information About the Thread Stack Handler.

This include file contains all information about the thread Stack Handler. This Handler provides mechanisms which can be used to initialize and utilize stacks.

## 10.194 cpukit/include/rtems/score/stackimpl.h File Reference

Inlined Routines from the Stack Handler.

```
#include <rtems/score/stack.h>
#include <rtems/score/context.h>
#include <rtems/score/tls.h>
```

### Functions

- static __inline__ void _Stack_Initialize (Stack_Control *the_stack, void *starting_address, size_t size)

  *Initializes stack with the given starting address and size.*
- static __inline__ uint32_t _Stack_Minimum (void)

  *Returns the minimum stack size.*
- static __inline__ bool _Stack_Is_enough (size_t size, bool is_fp)

  *Checks if the size is enough for a valid stack area on this processor.*
- static __inline__ size_t _Stack_Ensure_minimum (size_t size)

  *Returns the appropriate stack size for the requested size.*
- static __inline__ size_t _Stack_Extend_size (size_t stack_size, bool is_fp)

  *Extend the stack size to account for additional data structures allocated in the stack area of a thread.*
- void * _Stack_Allocate (size_t stack_size)

  *Allocate the requested stack space.*
- void _Stack_Free (void *stack_area)

  *Free the stack area allocated by _Stack_Allocate().*
- void _Stack_Free_nothing (void *stack_area)

  *This function does nothing.*

#### 10.194.1 Detailed Description

Inlined Routines from the Stack Handler.

This file contains the static inline implementation of the inlined routines from the Stack Handler.

## 10.195 cpukit/include/rtems/score/states.h File Reference

Thread Execution State Information.

```
#include <stdint.h>
```

### Typedefs

- typedef uint32_t States_Control

#### 10.195.1 Detailed Description

Thread Execution State Information.

This include file defines thread execution state information.

## 10.196 cpukit/include/rtems/score/statesimpl.h File Reference

Inlined Routines Associated with Thread State Information.

```
#include <rtems/score/states.h>
#include <rtems/score/basedefs.h>
```

## Macros

- #define STATES_READY 0x00000000
- #define STATES_WAITING_FOR_MUTEX 0x00000001
- #define STATES_WAITING_FOR_SEMAPHORE 0x00000002
- #define STATES_WAITING_FOR_EVENT 0x00000004
- #define STATES_WAITING_FOR_SYSTEM_EVENT 0x00000008
- #define STATES_WAITING_FOR_MESSAGE 0x00000010
- #define STATES_WAITING_FOR_CONDITION_VARIABLE 0x00000020
- #define STATES_WAITING_FOR_FUTEX 0x00000040
- #define STATES_WAITING_FOR_BSD_WAKEUP 0x00000080
- #define STATES_WAITING_FOR_TIME 0x00000100

  *This macro corresponds to a task which is waiting for a relative or absolute timeout.*
- #define STATES_WAITING_FOR_PERIOD 0x00000200
- #define STATES_WAITING_FOR_SIGNAL 0x00000400
- #define STATES_WAITING_FOR_BARRIER 0x00000800
- #define STATES_WAITING_FOR_RWLOCK 0x00001000
- #define STATES_WAITING_FOR_JOIN_AT_EXIT 0x00002000
- #define STATES_WAITING_FOR_JOIN 0x00004000
- #define STATES_SUSPENDED 0x00008000
- #define STATES_WAITING_FOR_SEGMENT 0x00010000
- #define STATES_LIFE_IS_CHANGING 0x00020000
- #define STATES_DEBUGGER 0x08000000
- #define STATES_INTERRUPTIBLE_BY_SIGNAL 0x10000000
- #define STATES_WAITING_FOR_RPC_REPLY 0x20000000
- #define STATES_ZOMBIE 0x40000000
- #define STATES_DORMANT 0x80000000
- #define STATES_LOCALLY_BLOCKED
- #define STATES_BLOCKED
- #define STATES_ALL_SET 0xffffffff

## Functions

- static __inline__ States_Control _States_Set (States_Control states_to_set, States_Control current_state)

  *Returns the result of setting the states to set to the given state.*
- static __inline__ States_Control _States_Clear (States_Control states_to_clear, States_Control current_↩ state)

  *Clears the states into the passed current state and returns the result.*
- static __inline__ bool _States_Is_ready (States_Control the_states)

  *Checks if the state is ready.*
- static __inline__ bool _States_Is_dormant (States_Control the_states)

  *Checks if DORMANT state is set.*
- static __inline__ bool _States_Is_suspended (States_Control the_states)

  *Checks if SUSPENDED state is set.*
- static __inline__ bool _States_Is_waiting_for_rpc_reply (States_Control the_states)

  *Checks if WAITING_FOR_TIME state is set.*
- static __inline__ bool _States_Is_waiting_for_join_at_exit (States_Control the_states)

  *Checks if WAITING_FOR_JOIN_AT_EXIT state is set.*
- static __inline__ bool _States_Is_interruptible_by_signal (States_Control the_states)

  *Checks if the state is set to be interruptible.*
- static __inline__ bool _States_Is_locally_blocked (States_Control the_states)

  *Checks if the state is blocked waiting on a local resource.*

### 10.196.1 Detailed Description

Inlined Routines Associated with Thread State Information.

This file contains the static inline implementation of the inlined routines associated with thread state information.

## 10.197 cpukit/include/rtems/score/sysstate.h File Reference

System State Handler API.

```
#include <rtems/score/basedefs.h>
```

### Macros

- #define **SYSTEM_STATE_CODES_FIRST** SYSTEM_STATE_BEFORE_INITIALIZATION
- #define **SYSTEM_STATE_CODES_LAST** SYSTEM_STATE_TERMINATED

### Enumerations

- enum System_state_Codes { SYSTEM_STATE_BEFORE_INITIALIZATION, SYSTEM_STATE_BEFORE_MULTITASKING, SYSTEM_STATE_UP, SYSTEM_STATE_TERMINATED }

  *System states.*

### Functions

- static __inline__ void _System_state_Set (System_state_Codes state)

  *Sets the current system state to the given state.*
- static __inline__ System_state_Codes _System_state_Get (void)

  *Gets the current system state.*
- static __inline__ bool _System_state_Is_before_initialization (System_state_Codes state)

  *Checks if the state is before initialization.*
- static __inline__ bool _System_state_Is_before_multitasking (System_state_Codes state)

  *Checks if the state is before multitasking.*
- static __inline__ bool _System_state_Is_up (System_state_Codes state)

  *Checks if the state is up.*
- static __inline__ bool _System_state_Is_terminated (System_state_Codes state)

  *Checks if the state is terminated.*

### Variables

- System_state_Codes **_System_state_Current**

### 10.197.1 Detailed Description

System State Handler API.

## 10.198 cpukit/include/rtems/score/thread.h File Reference

Constants and Structures Related with the Thread Control Block.

```
#include <rtems/score/atomic.h>
#include <rtems/score/context.h>
#include <rtems/score/freechain.h>
#include <rtems/score/isrlock.h>
#include <rtems/score/objectdata.h>
#include <rtems/score/priority.h>
#include <rtems/score/schedulernode.h>
#include <rtems/score/stack.h>
#include <rtems/score/states.h>
#include <rtems/score/threadq.h>
#include <rtems/score/timestamp.h>
#include <rtems/score/watchdog.h>
#include <rtems/score/processormask.h>
```

### Classes

- struct Thread_Entry_idle

  *Data for idle thread entry.*
- struct Thread_Entry_numeric

  *Data for thread entry with one numeric argument and no return value.*
- struct Thread_Entry_pointer

  *Data for thread entry with one pointer argument and a pointer return value.*
- struct Thread_Entry_information

  *Thread entry information.*
- struct Thread_Start_information
- struct Thread_Scheduler_control

  *Thread scheduler control.*
- union Thread_Wait_information_Object_argument_type

  *Union type to hold a pointer to an immutable or a mutable object.*
- struct Thread_Wait_information

  *Information required to manage a thread while it is blocked.*
- struct Thread_Timer_information

  *Information required to manage a thread timer.*
- struct Thread_Proxy_control
- struct Thread_Action

  *Thread action.*
- struct Thread_Keys_information

  *Per-thread information for POSIX Keys.*
- struct Thread_Action_control

  *Control block to manage thread actions.*
- struct Thread_Life_control

  *Thread life control.*
- struct Thread_Capture_control
- struct _Thread_Control
- struct Thread_Control_add_on

  *Thread control add-on.*
- struct Thread_Information

  *The thread object information.*

## Macros

- #define **RTEMS_SCORE_THREAD_ENABLE_EXHAUST_TIMESLICE**
- #define **RTEMS_SCORE_THREAD_ENABLE_SCHEDULER_CALLOUT**
- #define THREAD_API_FIRST THREAD_API_RTEMS
- #define THREAD_API_LAST THREAD_API_POSIX
- #define THREAD_DEFAULT_MAXIMUM_NAME_SIZE 16

  *The default maximum size of a thread name in characters (including the terminating '\0' character).*
- #define **THREAD_INFORMATION_DEFINE_ZERO**(name, api, cls)
- #define **THREAD_INFORMATION_DEFINE**(name, api, cls, max)

## Typedefs

- typedef CPU_Uint32ptr Thread_Entry_numeric_type

  *Type of the numeric argument of a thread entry function with at least one numeric argument.*
- typedef void(∗ Thread_CPU_budget_algorithm_callout) (Thread_Control ∗)
- typedef unsigned int Thread_Wait_flags

  *This type is able to contain several flags used to control the wait class and state of a thread.*
- typedef struct Thread_Action **Thread_Action**
- typedef void(∗ Thread_Action_handler) (Thread_Control ∗the_thread, Thread_Action ∗action, ISR_lock_Context ∗lock_context)

  *Thread action handler.*
- typedef void(∗ **rtems_per_thread_routine**) (Thread_Control ∗)
- typedef struct Thread_Configured_control Thread_Configured_control

  *The configured thread control block.*
- typedef struct Thread_queue_Configured_heads Thread_queue_Configured_heads

  *The configured thread queue heads.*

## Enumerations

- enum Thread_CPU_budget_algorithms { **THREAD_CPU_BUDGET_ALGORITHM_NONE**, **THREAD_C↩ PU_BUDGET_ALGORITHM_RESET_TIMESLICE**, **THREAD_CPU_BUDGET_ALGORITHM_EXHAUST↩ _TIMESLICE**, **THREAD_CPU_BUDGET_ALGORITHM_CALLOUT** }
- enum Thread_Scheduler_state { THREAD_SCHEDULER_BLOCKED, THREAD_SCHEDULER_SCHEDULED, THREAD_SCHEDULER_READY }

  *The thread state with respect to the scheduler.*
- enum Thread_APIs { THREAD_API_RTEMS, THREAD_API_POSIX }
- enum Thread_Life_state {
  **THREAD_LIFE_PROTECTED** = 0x1, **THREAD_LIFE_RESTARTING** = 0x2, **THREAD_LIFE_TERMINAT↩ ING** = 0x4, **THREAD_LIFE_CHANGE_DEFERRED** = 0x8,
  **THREAD_LIFE_DETACHED** = 0x10 }

  *Thread life states.*

## Functions

- void rtems_iterate_over_all_threads (rtems_per_thread_routine routine) RTEMS_DEPRECATED

  *Deprecated, use rtems_task_iterate() instead.*
- Objects_Control ∗ _Thread_Allocate_unlimited (Objects_Information ∗information)

  *Return an inactive thread object or NULL.*

### Variables

- const Thread_Control_add_on _Thread_Control_add_ons [ ]

  *Thread control add-ons.*
- const size_t _Thread_Control_add_on_count

  *Thread control add-on count.*
- const size_t _Thread_Initial_thread_count

  *Count of configured threads.*
- const size_t _Thread_Maximum_name_size

  *Maximum size of a thread name in characters (including the terminating '\0' character).*
- const size_t _Thread_Maximum_TLS_size

  *If this constant is greater than zero, then it defines the maximum thread-local storage size, otherwise the thread-local storage size is defined by the linker depending on the thread-local storage objects used by the application in the statically-linked executable.*
- const size_t _Thread_queue_Heads_size

  *Size of the thread queue heads of a particular application.*
- Thread_Information _Thread_Information

  *The internal thread objects information.*
- char _Thread_Idle_stacks [ ]

  *The idle thread stacks.*

#### 10.198.1 Detailed Description

Constants and Structures Related with the Thread Control Block.

This include file contains all constants and structures associated with the thread control block.

## 10.199 cpukit/include/rtems/score/threaddispatch.h File Reference

Constants and Structures Related with Thread Dispatch.

```
#include <rtems/score/percpu.h>
#include <rtems/score/isrlock.h>
#include <rtems/score/profiling.h>
```

### Macros

- #define RTEMS_SCORE_ROBUST_THREAD_DISPATCH

  *Enables a robust thread dispatch.*

### Functions

- static __inline__ bool _Thread_Dispatch_is_enabled (void)

  *Indicates if the executing thread is inside a thread dispatch critical section.*
- static __inline__ uint32_t _Thread_Dispatch_get_disable_level (void)

  *Gets thread dispatch disable level.*
- static __inline__ void _Thread_Dispatch_initialization (void)

  *Thread dispatch initialization.*
- void _Thread_Dispatch (void)

  *Performs a thread dispatch if necessary.*
- void _Thread_Dispatch_direct (Per_CPU_Control *cpu_self)

  *Directly do a thread dispatch.*
- void _Thread_Do_dispatch (Per_CPU_Control *cpu_self, ISR_Level level)

  *Performs a thread dispatch on the current processor.*
- static __inline__ Per_CPU_Control * _Thread_Dispatch_disable_with_CPU (Per_CPU_Control *cpu_self, const ISR_lock_Context *lock_context)

  *Disables thread dispatching inside a critical section (interrupts disabled) with the current processor.*
- static __inline__ Per_CPU_Control * _Thread_Dispatch_disable_critical (const ISR_lock_Context *lock_↩ context)

  *Disables thread dispatching inside a critical section (interrupts disabled).*
- static __inline__ Per_CPU_Control * _Thread_Dispatch_disable (void)

  *Disables thread dispatching.*
- void _Thread_Dispatch_enable (Per_CPU_Control *cpu_self)

  *Enables thread dispatching.*
- static __inline__ void _Thread_Dispatch_unnest (Per_CPU_Control *cpu_self)

  *Unnests thread dispatching.*
- static __inline__ void _Thread_Dispatch_request (Per_CPU_Control *cpu_self, Per_CPU_Control *cpu_↩ target)

  *Requests a thread dispatch on the target processor.*

### 10.199.1 Detailed Description

Constants and Structures Related with Thread Dispatch.

## 10.200 cpukit/include/rtems/score/threadidledata.h File Reference

Constants for the idle threads.

```
#include <rtems/score/basedefs.h>
```

### Typedefs

- typedef void *(* Thread_Idle_body) (uintptr_t)

  *The idle thread body type.*

### Variables

- const size_t _Thread_Idle_stack_size

  *The idle thread stack size in bytes.*
- const Thread_Idle_body _Thread_Idle_body

  *The idle thread body.*

#### 10.200.1 Detailed Description

Constants for the idle threads.

## 10.201 cpukit/include/rtems/score/threadimpl.h File Reference

Inlined Routines from the Thread Handler.

```
#include <rtems/score/thread.h>
#include <rtems/score/assert.h>
#include <rtems/score/chainimpl.h>
#include <rtems/score/interr.h>
#include <rtems/score/isr.h>
#include <rtems/score/objectimpl.h>
#include <rtems/score/schedulernodeimpl.h>
#include <rtems/score/statesimpl.h>
#include <rtems/score/status.h>
#include <rtems/score/sysstate.h>
#include <rtems/score/timestampimpl.h>
#include <rtems/score/threadqimpl.h>
#include <rtems/score/todimpl.h>
#include <rtems/score/watchdogimpl.h>
#include <rtems/config.h>
```

### Classes

- struct Thread_Configuration

  *The configuration of a new thread to initialize.*
- struct Thread_Close_context

### Macros

- #define THREAD_OF_SCHEDULER_HELP_NODE(node) RTEMS_CONTAINER_OF( node, Thread_Control, Scheduler.Help_node )
- #define **THREAD_QUEUE_CONTEXT_OF_REQUEST**(node) RTEMS_CONTAINER_OF( node, Thread_queue_Context, Lock_context.Wait.Gate.Node )
- #define THREAD_WAIT_FLAGS_INITIAL 0x0U

  *The initial thread wait flags value set by _Thread_Initialize().*
- #define THREAD_WAIT_STATE_MASK 0xffU

  *Mask to get the thread wait state flags.*
- #define THREAD_WAIT_STATE_INTEND_TO_BLOCK 0x1U

*Indicates that the thread begins with the blocking operation.*

- #define THREAD_WAIT_STATE_BLOCKED 0x2U

  *Indicates that the thread completed the blocking operation.*
- #define THREAD_WAIT_STATE_READY_AGAIN 0x4U

  *Indicates that a condition to end the thread wait occurred.*
- #define THREAD_WAIT_CLASS_MASK 0xff00U

  *Mask to get the thread wait class flags.*
- #define THREAD_WAIT_CLASS_EVENT 0x100U

  *Indicates that the thread waits for an event.*
- #define THREAD_WAIT_CLASS_SYSTEM_EVENT 0x200U

  *Indicates that the thread waits for a system event.*
- #define THREAD_WAIT_CLASS_OBJECT 0x400U

  *Indicates that the thread waits for an object.*
- #define THREAD_WAIT_CLASS_PERIOD 0x800U

  *Indicates that the thread waits for a period.*
- #define **THREAD_PIN_STEP** 2
- #define **THREAD_PIN_PREEMPTION** 1

## Typedefs

- typedef bool(∗ **Thread_Visitor**) (Thread_Control ∗the_thread, void ∗arg)

## Functions

- void _Thread_Iterate (Thread_Visitor visitor, void ∗arg)

  *Calls the visitor with all threads and the given argument until it is done.*
- void _Thread_Initialize_information (Thread_Information ∗information)

  *Initializes the thread information.*
- void _Thread_Handler_initialization (void)

  *Initializes thread handler.*
- void _Thread_Create_idle (void)

  *Creates idle thread.*
- RTEMS_NO_RETURN void _Thread_Start_multitasking (void)

  *Starts thread multitasking.*
- bool _Thread_Initialize (Thread_Information ∗information, Thread_Control ∗the_thread, const Thread_Configuration ∗config)

  *Initializes thread.*
- bool _Thread_Start (Thread_Control ∗the_thread, const Thread_Entry_information ∗entry, ISR_lock_Context ∗lock_context)

  *Starts the specified thread.*
- RTEMS_NO_RETURN void _Thread_Restart_self (Thread_Control ∗executing, const Thread_Entry_information ∗entry, ISR_lock_Context ∗lock_context)

  *Restarts the currently executing thread.*
- bool _Thread_Restart_other (Thread_Control ∗the_thread, const Thread_Entry_information ∗entry, ISR_lock_Context ∗lock_context)

  *Restarts the thread.*
- void _Thread_Yield (Thread_Control ∗executing)

  *Yields the currently executing thread.*
- Thread_Life_state _Thread_Change_life (Thread_Life_state clear, Thread_Life_state set, Thread_Life_state ignore)

*Changes the currently executing thread to a new state with the sets.*

- Thread_Life_state _Thread_Set_life_protection (Thread_Life_state state)

  *Set the thread to life protected.*
- void _Thread_Kill_zombies (void)

  *Kills all zombie threads in the system.*
- void _Thread_Exit (Thread_Control *executing, Thread_Life_state set, void *exit_value)

  *Exits the currently executing thread.*
- void _Thread_Join (Thread_Control *the_thread, States_Control waiting_for_join, Thread_Control *executing, Thread_queue_Context *queue_context)

  *Joins the currently executing thread with the given thread to wait for.*
- void _Thread_Cancel (Thread_Control *the_thread, Thread_Control *executing, void *exit_value)

  *Cancels the thread.*
- void _Thread_Close (Thread_Control *the_thread, Thread_Control *executing, Thread_Close_context *context)

  *Closes the thread.*
- static __inline__ bool _Thread_Is_ready (const Thread_Control *the_thread)

  *Checks if the thread is ready.*
- States_Control _Thread_Clear_state_locked (Thread_Control *the_thread, States_Control state)

  *Clears the specified thread state without locking the lock context.*
- States_Control _Thread_Clear_state (Thread_Control *the_thread, States_Control state)

  *Clears the specified thread state.*
- States_Control _Thread_Set_state_locked (Thread_Control *the_thread, States_Control state)

  *Sets the specified thread state without locking the lock context.*
- States_Control _Thread_Set_state (Thread_Control *the_thread, States_Control state)

  *Sets the specified thread state.*
- void _Thread_Load_environment (Thread_Control *the_thread)

  *Initializes enviroment for a thread.*
- void _Thread_Entry_adaptor_idle (Thread_Control *executing)

  *Calls the start kinds idle entry of the thread.*
- void _Thread_Entry_adaptor_numeric (Thread_Control *executing)

  *Calls the start kinds numeric entry of the thread.*
- void _Thread_Entry_adaptor_pointer (Thread_Control *executing)

  *Calls the start kinds pointer entry of the thread.*
- void _Thread_Handler (void)

  *Wrapper function for all threads.*
- static __inline__ void _Thread_State_acquire_critical (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Acquires the lock context in a critical section.*
- static __inline__ void _Thread_State_acquire (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Disables interrupts and acquires the lock_context.*
- static __inline__ Thread_Control * _Thread_State_acquire_for_executing (ISR_lock_Context *lock_context)

  *Disables interrupts and acquires the lock context for the currently executing thread.*
- static __inline__ void _Thread_State_release_critical (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Release the lock context in a critical section.*
- static __inline__ void _Thread_State_release (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Releases the lock context and enables interrupts.*
- void _Thread_Priority_perform_actions (Thread_Control *start_of_path, Thread_queue_Context *queue_context)

  *Checks if the thread is owner of the lock of the join queue.*

- void _Thread_Priority_add (Thread_Control *the_thread, Priority_Node *priority_node, Thread_queue_Context *queue_context)

  *Adds the specified thread priority node to the corresponding thread priority aggregation.*
- void _Thread_Priority_remove (Thread_Control *the_thread, Priority_Node *priority_node, Thread_queue_Context *queue_context)

  *Removes the specified thread priority node from the corresponding thread priority aggregation.*
- void _Thread_Priority_changed (Thread_Control *the_thread, Priority_Node *priority_node, bool prepend_it, Thread_queue_Context *queue_context)

  *Propagates a thread priority value change in the specified thread priority node to the corresponding thread priority aggregation.*
- static __inline__ void _Thread_Priority_change (Thread_Control *the_thread, Priority_Node *priority_node, Priority_Control new_priority, bool prepend_it, Thread_queue_Context *queue_context)

  *Changes the thread priority value of the specified thread priority node in the corresponding thread priority aggregation.*
- void _Thread_Priority_replace (Thread_Control *the_thread, Priority_Node *victim_node, Priority_Node *replacement_node)

  *Replaces the victim priority node with the replacement priority node in the corresponding thread priority aggregation.*
- void _Thread_Priority_update (Thread_queue_Context *queue_context)

  *Updates the priority of all threads in the set.*
- void _Thread_Priority_and_sticky_update (Thread_Control *the_thread, int sticky_level_change)

  *Updates the priority of the thread and changes it sticky level.*
- static __inline__ bool _Thread_Priority_less_than (Priority_Control left, Priority_Control right)

  *Checks if the left thread priority is less than the right thread priority in the intuitive sense of priority.*
- static __inline__ Priority_Control _Thread_Priority_highest (Priority_Control left, Priority_Control right)

  *Returns the highest priority of the left and right thread priorities in the intuitive sense of priority.*
- static __inline__ Objects_Information * _Thread_Get_objects_information (Objects_Id id)

  *Gets object information for the object id.*
- Thread_Control * _Thread_Get (Objects_Id id, ISR_lock_Context *lock_context)

  *Gets a thread by its identifier.*
- static __inline__ Per_CPU_Control * _Thread_Get_CPU (const Thread_Control *thread)

  *Gets the cpu of the thread's scheduler.*
- static __inline__ void _Thread_Set_CPU (Thread_Control *thread, Per_CPU_Control *cpu)

  *Sets the cpu of the thread's scheduler.*
- static __inline__ bool _Thread_Is_executing (const Thread_Control *the_thread)

  *Checks if the thread is the currently executing thread.*
- static __inline__ bool _Thread_Is_executing_on_a_processor (const Thread_Control *the_thread)

  *Checks if the thread executes currently on some processor in the system.*
- static __inline__ bool _Thread_Is_heir (const Thread_Control *the_thread)

  *Checks if the thread is the heir.*
- static __inline__ void _Thread_Unblock (Thread_Control *the_thread)

  *Unblocks the thread.*
- static __inline__ void _Thread_Save_fp (Thread_Control *executing)

  *Checks if the floating point context of the thread is currently loaded in the floating point unit.*
- static __inline__ void _Thread_Restore_fp (Thread_Control *executing)

  *Restores the executing thread's floating point area.*
- static __inline__ bool _Thread_Is_context_switch_necessary (void)

  *Deallocates the currently loaded floating point context.*
- static __inline__ uint32_t _Thread_Get_maximum_internal_threads (void)

  *Gets the maximum number of internal threads.*
- static __inline__ Thread_Control * _Thread_Internal_allocate (void)

  *Allocates an internal thread and returns it.*
- static __inline__ Thread_Control * _Thread_Get_heir_and_make_it_executing (Per_CPU_Control *cpu_self)

*Gets the heir of the processor and makes it executing.*

- static __inline__ void _Thread_Update_CPU_time_used (Thread_Control *the_thread, Per_CPU_Control *cpu)

  *Updates the cpu time used of the thread.*

- static __inline__ void _Thread_Dispatch_update_heir (Per_CPU_Control *cpu_self, Per_CPU_Control *cpu_for_heir, Thread_Control *heir)

  *Updates the used cpu time for the heir and dispatches a new heir.*

- void _Thread_Get_CPU_time_used (Thread_Control *the_thread, Timestamp_Control *cpu_time_used)

  *Gets the used cpu time of the thread and stores it in the given Timestamp_Control.*

- static __inline__ void _Thread_Action_control_initialize (Thread_Action_control *action_control)

  *Initializes the control chain of the action control.*

- static __inline__ void _Thread_Action_initialize (Thread_Action *action)

  *Initializes the Thread action.*

- static __inline__ void _Thread_Add_post_switch_action (Thread_Control *the_thread, Thread_Action *action, Thread_Action_handler handler)

  *Adds a post switch action to the thread with the given handler.*

- static __inline__ bool _Thread_Is_life_restarting (Thread_Life_state life_state)

  *Checks if the thread life state is restarting.*

- static __inline__ bool _Thread_Is_life_terminating (Thread_Life_state life_state)

  *Checks if the thread life state is terminating.*

- static __inline__ bool _Thread_Is_life_change_allowed (Thread_Life_state life_state)

  *Checks if the thread life state allos life change.*

- static __inline__ bool _Thread_Is_life_changing (Thread_Life_state life_state)

  *Checks if the thread life state is life changing.*

- static __inline__ bool _Thread_Is_joinable (const Thread_Control *the_thread)

  *Checks if the thread is joinable.*

- static __inline__ void _Thread_Resource_count_increment (Thread_Control *the_thread)

  *Increments the thread's resource count.*

- static __inline__ void _Thread_Resource_count_decrement (Thread_Control *the_thread)

  *Decrements the thread's resource count.*

- static __inline__ void _Thread_Scheduler_cancel_need_for_help (Thread_Control *the_thread, Per_CPU_Control *cpu)

  *Cancels the thread's need for help.*

- static __inline__ const Scheduler_Control * _Thread_Scheduler_get_home (const Thread_Control *the_thread)

  *Gets the home scheduler of the thread.*

- static __inline__ Scheduler_Node * _Thread_Scheduler_get_home_node (const Thread_Control *the_thread)

  *Gets the scheduler's home node.*

- static __inline__ Scheduler_Node * _Thread_Scheduler_get_node_by_index (const Thread_Control *the_thread, size_t scheduler_index)

  *Gets the thread's scheduler node by index.*

- static __inline__ void _Thread_Scheduler_acquire_critical (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Acquires the lock context in a critical section.*

- static __inline__ void _Thread_Scheduler_release_critical (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Releases the lock context in a critical section.*

- void _Thread_Scheduler_process_requests (Thread_Control *the_thread)

  *Process the thread's scheduler requests.*

- static __inline__ void _Thread_Scheduler_add_request (Thread_Control *the_thread, Scheduler_Node *scheduler_node, Scheduler_Node_request request)

*Add a scheduler request to the thread.*

- static __inline__ void _Thread_Scheduler_add_wait_node (Thread_Control *the_thread, Scheduler_Node *scheduler_node)

  *Adds a wait node to the thread and adds a corresponding request to the thread.*
- static __inline__ void _Thread_Scheduler_remove_wait_node (Thread_Control *the_thread, Scheduler_Node *scheduler_node)

  *Remove a wait node from the thread and add a corresponding request to it.*
- static __inline__ Priority_Control _Thread_Get_priority (const Thread_Control *the_thread)

  *Returns the priority of the thread.*
- static __inline__ Priority_Control _Thread_Get_unmapped_priority (const Thread_Control *the_thread)

  *Returns the unmapped priority of the thread.*
- static __inline__ Priority_Control _Thread_Get_unmapped_real_priority (const Thread_Control *the_thread)

  *Returns the unmapped real priority of the thread.*
- static __inline__ void _Thread_Wait_acquire_default_critical (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Acquires the thread wait default lock inside a critical section (interrupts disabled).*
- static __inline__ Thread_Control * _Thread_Wait_acquire_default_for_executing (ISR_lock_Context *lock↩_context)

  *Acquires the thread wait default lock and returns the executing thread.*
- static __inline__ void _Thread_Wait_acquire_default (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Acquires the thread wait default lock and disables interrupts.*
- static __inline__ void _Thread_Wait_release_default_critical (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Releases the thread wait default lock inside a critical section (interrupts disabled).*
- static __inline__ void _Thread_Wait_release_default (Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Releases the thread wait default lock and restores the previous interrupt status.*
- static __inline__ void _Thread_Wait_remove_request_locked (Thread_Control *the_thread, Thread_queue_Lock_context *queue_lock_context)

  *Removes the first pending wait lock request.*
- static __inline__ void _Thread_Wait_acquire_queue_critical (Thread_queue_Queue *queue, Thread_queue_Lock_context *queue_lock_context)

  *Acquires the wait queue inside a critical section.*
- static __inline__ void _Thread_Wait_release_queue_critical (Thread_queue_Queue *queue, Thread_queue_Lock_context *queue_lock_context)

  *Releases the wait queue inside a critical section.*
- static __inline__ void _Thread_Wait_acquire_critical (Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Acquires the thread wait lock inside a critical section (interrupts disabled).*
- static __inline__ void _Thread_Wait_acquire (Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Acquires the thread wait default lock and disables interrupts.*
- static __inline__ void _Thread_Wait_release_critical (Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Releases the thread wait lock inside a critical section (interrupts disabled).*
- static __inline__ void _Thread_Wait_release (Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Releases the thread wait lock and restores the previous interrupt status.*
- static __inline__ void _Thread_Wait_claim (Thread_Control *the_thread, Thread_queue_Queue *queue)

  *Claims the thread wait queue.*
- static __inline__ void _Thread_Wait_claim_finalize (Thread_Control *the_thread, const Thread_queue_Operations *operations)

*Finalizes the thread wait queue claim via registration of the corresponding thread queue operations.*

- static __inline__ void _Thread_Wait_remove_request (Thread_Control *the_thread, Thread_queue_Lock_context *queue_lock_context)

  *Removes a thread wait lock request.*
- static __inline__ void _Thread_Wait_restore_default (Thread_Control *the_thread)

  *Restores the default thread wait queue and operations.*
- static __inline__ void _Thread_Wait_tranquilize (Thread_Control *the_thread)

  *Tranquilizes the thread after a wait on a thread queue.*
- static __inline__ void _Thread_Wait_cancel (Thread_Control *the_thread, Thread_queue_Context *queue↩ _context)

  *Cancels a thread wait on a thread queue.*
- static __inline__ void _Thread_Wait_flags_set (Thread_Control *the_thread, Thread_Wait_flags flags)

  *Sets the thread's wait flags.*
- static __inline__ Thread_Wait_flags _Thread_Wait_flags_get (const Thread_Control *the_thread)

  *Gets the thread's wait flags according to the ATOMIC_ORDER_RELAXED.*
- static __inline__ Thread_Wait_flags _Thread_Wait_flags_get_acquire (const Thread_Control *the_thread)

  *Gets the thread's wait flags according to the ATOMIC_ORDER_ACQUIRE.*
- static __inline__ bool _Thread_Wait_flags_try_change_release (Thread_Control *the_thread, Thread_Wait_flags expected_flags, Thread_Wait_flags desired_flags)

  *Tries to change the thread wait flags with release semantics in case of success.*
- static __inline__ bool _Thread_Wait_flags_try_change_acquire (Thread_Control *the_thread, Thread_Wait_flags expected_flags, Thread_Wait_flags desired_flags)

  *Tries to change the thread wait flags with acquire semantics.*
- Objects_Id _Thread_Wait_get_id (const Thread_Control *the_thread)

  *Returns the object identifier of the object containing the current thread wait queue.*
- static __inline__ Status_Control _Thread_Wait_get_status (const Thread_Control *the_thread)

  *Get the status of the wait return code of the thread.*
- void _Thread_Continue (Thread_Control *the_thread, Status_Control status)

  *Cancels a blocking operation so that the thread can continue its execution.*
- void _Thread_Timeout (Watchdog_Control *the_watchdog)

  *General purpose thread wait timeout.*
- static __inline__ void _Thread_Timer_initialize (Thread_Timer_information *timer, Per_CPU_Control *cpu)

  *Initializes the thread timer.*
- static __inline__ void _Thread_Add_timeout_ticks (Thread_Control *the_thread, Per_CPU_Control *cpu, Watchdog_Interval ticks)

  *Adds timeout ticks to the thread.*
- static __inline__ void _Thread_Timer_insert_realtime (Thread_Control *the_thread, Per_CPU_Control *cpu, Watchdog_Service_routine_entry routine, uint64_t expire)

  *Inserts the cpu's watchdog realtime into the thread's timer.*
- static __inline__ void _Thread_Timer_remove (Thread_Control *the_thread)

  *Remove the watchdog timer from the thread.*
- static __inline__ void _Thread_Remove_timer_and_unblock (Thread_Control *the_thread, Thread_queue_Queue *queue)

  *Remove the watchdog timer from the thread and unblock if necessary.*
- Status_Control _Thread_Set_name (Thread_Control *the_thread, const char *name)

  *Sets the name of the thread.*
- size_t _Thread_Get_name (const Thread_Control *the_thread, char *buffer, size_t buffer_size)

  *Gets the name of the thread.*
- void _Thread_Do_unpin (Thread_Control *executing, Per_CPU_Control *cpu_self)

  *Unpins the thread.*
- static __inline__ void _Thread_Pin (Thread_Control *executing)

  *Pin the executing thread.*
- static __inline__ void _Thread_Unpin (Thread_Control *executing, Per_CPU_Control *cpu_self)

  *Unpins the thread.*

### Variables

- void * rtems_ada_self
- Objects_Id _Thread_Global_constructor

  *Object identifier of the global constructor thread.*

### 10.201.1 Detailed Description

Inlined Routines from the Thread Handler.

This file contains the macro implementation of the inlined routines from the Thread handler.

## 10.202 cpukit/include/rtems/score/threadq.h File Reference

Constants and Structures Needed to Declare a Thread Queue.

```
#include <rtems/score/chain.h>
#include <rtems/score/isrlock.h>
#include <rtems/score/object.h>
#include <rtems/score/priority.h>
#include <rtems/score/rbtree.h>
#include <rtems/score/states.h>
#include <rtems/score/watchdogticks.h>
```

### Classes

- struct Thread_queue_Gate

  *The thread queue gate is an SMP synchronization means.*
- struct Thread_queue_Lock_context
- struct Thread_queue_Link

  *A thread queue link from one thread to another specified by the thread queue owner and thread wait queue relationships.*
- struct Thread_queue_Context

  *Thread queue context for the thread queue methods.*
- struct Thread_queue_Priority_queue

  *Thread priority queue.*
- struct _Thread_queue_Heads

  *Thread queue heads.*
- struct Thread_queue_Queue
- struct Thread_queue_Operations

  *Thread queue operations.*
- struct Thread_queue_Control

### Typedefs

- typedef struct _Thread_Control **Thread_Control**
- typedef struct Thread_queue_Context **Thread_queue_Context**
- typedef struct Thread_queue_Queue **Thread_queue_Queue**
- typedef struct Thread_queue_Operations **Thread_queue_Operations**
- typedef void(* Thread_queue_Enqueue_callout) (Thread_queue_Queue *queue, Thread_Control *the_thread, struct Per_CPU_Control *cpu_self, Thread_queue_Context *queue_context)

  *Thread queue enqueue callout.*
- typedef void(* Thread_queue_Deadlock_callout) (Thread_Control *the_thread)

  *Thread queue deadlock callout.*
- typedef struct _Thread_queue_Heads Thread_queue_Heads

  *Thread queue heads.*
- typedef void(* Thread_queue_Priority_actions_operation) (Thread_queue_Queue *queue, Priority_Actions *priority_actions)

  *Thread queue action operation.*
- typedef void(* Thread_queue_Enqueue_operation) (Thread_queue_Queue *queue, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Thread queue enqueue operation.*
- typedef void(* Thread_queue_Extract_operation) (Thread_queue_Queue *queue, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Thread queue extract operation.*
- typedef Thread_Control *(* Thread_queue_Surrender_operation) (Thread_queue_Queue *queue, Thread_queue_Heads *heads, Thread_Control *previous_owner, Thread_queue_Context *queue_context)

  *Thread queue surrender operation.*
- typedef Thread_Control *(* Thread_queue_First_operation) (Thread_queue_Heads *heads)

  *Thread queue first operation.*

### 10.202.1 Detailed Description

Constants and Structures Needed to Declare a Thread Queue.

This include file contains all the constants and structures needed to declare a thread queue.

## 10.203 cpukit/include/rtems/score/threadqimpl.h File Reference

Constants and Structures Associated with the Manipulation of Objects.

```
#include <rtems/score/threadq.h>
#include <rtems/score/chainimpl.h>
#include <rtems/score/priorityimpl.h>
#include <rtems/score/scheduler.h>
#include <rtems/score/smp.h>
#include <rtems/score/status.h>
#include <rtems/score/thread.h>
#include <rtems/score/threaddispatch.h>
```

## Classes

- struct Thread_queue_Syslock_queue

  *Thread queue with a layout compatible to struct _Thread_queue_Queue defined in Newlib <sys/lock.h>.*
- struct Thread_queue_Object

  *Helper structure to ensure that all objects containing a thread queue have the right layout.*

## Macros

- #define **THREAD_QUEUE_LINK_OF_PATH_NODE**(node) RTEMS_CONTAINER_OF( node, Thread_queue_Link, Path_node );
- #define **_Thread_queue_Context_ISR_disable**(queue_context, level)
- #define _Thread_queue_Context_set_MP_callout(queue_context, mp_callout)

  *Sets the MP callout in the thread queue context.*
- #define **_Thread_queue_Queue_acquire_critical**(queue, lock_stats, lock_context) _Thread_queue_Queue_do_acquire_critic queue, lock_context )
- #define _Thread_queue_Dequeue(the_thread_queue, operations, mp_callout)

  *Gets a pointer to a thread waiting on the_thread_queue.*
- #define **THREAD_QUEUE_INITIALIZER**(_name)
- #define **THREAD_QUEUE_OBJECT_ASSERT**(object_type, wait_queue_member, msg)
- #define **THREAD_QUEUE_QUEUE_TO_OBJECT**(queue)

## Typedefs

- typedef Thread_Control *(* Thread_queue_Flush_filter) (Thread_Control *the_thread, Thread_queue_Queue *queue, Thread_queue_Context *queue_context)

  *Thread queue flush filter function.*

## Functions

- void _Thread_queue_Enqueue_do_nothing_extra (Thread_queue_Queue *queue, Thread_Control *the_↩ thread, Per_CPU_Control *cpu_self, Thread_queue_Context *queue_context)

  *Does nothing.*
- void _Thread_queue_Add_timeout_ticks (Thread_queue_Queue *queue, Thread_Control *the_thread, Per_CPU_Control *cpu_self, Thread_queue_Context *queue_context)

  *Adds timeout ticks of the queue to the thread.*
- void _Thread_queue_Add_timeout_monotonic_timespec (Thread_queue_Queue *queue, Thread_Control *the_thread, Per_CPU_Control *cpu_self, Thread_queue_Context *queue_context)

  *Adds a monotonic timespec to the thread and sets the watchdog header to monotonic.*
- void _Thread_queue_Add_timeout_realtime_timespec (Thread_queue_Queue *queue, Thread_Control *the_thread, Per_CPU_Control *cpu_self, Thread_queue_Context *queue_context)

  *Adds a monotonic timespec to the thread and sets the watchdog header to realtime.*
- void _Thread_queue_Deadlock_status (Thread_Control *the_thread)

  *Sets the thread wait return code to STATUS_DEADLOCK.*
- void _Thread_queue_Deadlock_fatal (Thread_Control *the_thread)

  *Results in an INTERNAL_ERROR_THREAD_QUEUE_DEADLOCK fatal error.*
- static __inline__ void _Thread_queue_Context_initialize (Thread_queue_Context *queue_context)

  *Initializes a thread queue context.*
- static __inline__ void _Thread_queue_Context_set_thread_state (Thread_queue_Context *queue_context, States_Control thread_state)

*Sets the thread state for the thread to enqueue in the thread queue context.*

- static __inline__ void _Thread_queue_Context_set_timeout_ticks (Thread_queue_Context *queue_context, Watchdog_Interval ticks)

  *Sets the timeout ticks in the thread queue context.*

- static __inline__ void _Thread_queue_Context_set_timeout_argument (Thread_queue_Context *queue_context, const void *arg)

  *Sets the timeout argument in the thread queue context.*

- static __inline__ void _Thread_queue_Context_set_enqueue_callout (Thread_queue_Context *queue_context, Thread_queue_Enqueue_callout enqueue_callout)

  *Sets the enqueue callout in the thread queue context.*

- static __inline__ void _Thread_queue_Context_set_enqueue_do_nothing_extra (Thread_queue_Context *queue_context)

  *Sets the do nothing enqueue callout in the thread queue context.*

- static __inline__ void _Thread_queue_Context_set_enqueue_timeout_ticks (Thread_queue_Context *queue_context, Watchdog_Interval ticks)

  *Sets the enqueue callout to add a relative monotonic timeout in ticks.*

- static __inline__ void _Thread_queue_Context_set_enqueue_timeout_monotonic_timespec (Thread_queue_Context *queue_context, const struct timespec *abstime)

  *Sets the enqueue callout to add an absolute monotonic timeout in timespec format.*

- static __inline__ void _Thread_queue_Context_set_enqueue_timeout_realtime_timespec (Thread_queue_Context *queue_context, const struct timespec *abstime)

  *Sets the enqueue callout to add an absolute realtime timeout in timespec format.*

- static __inline__ void _Thread_queue_Context_set_deadlock_callout (Thread_queue_Context *queue_context, Thread_queue_Deadlock_callout deadlock_callout)

  *Sets the deadlock callout in the thread queue context.*

- static __inline__ void _Thread_queue_Context_clear_priority_updates (Thread_queue_Context *queue_context)

  *Clears the priority update count of the thread queue context.*

- static __inline__ size_t _Thread_queue_Context_save_priority_updates (Thread_queue_Context *queue_context)

  *Returns the priority update count of the thread queue context.*

- static __inline__ void _Thread_queue_Context_restore_priority_updates (Thread_queue_Context *queue_context, size_t update_count)

  *Sets the priority update count of the thread queue context.*

- static __inline__ void _Thread_queue_Context_add_priority_update (Thread_queue_Context *queue_context, Thread_Control *the_thread)

  *Adds a priority update of the thread to the thread queue context.*

- static __inline__ void _Thread_queue_Context_set_ISR_level (Thread_queue_Context *queue_context, ISR_Level level)

  *Sets the thread queue context ISR level.*

- static __inline__ Per_CPU_Control * _Thread_queue_Dispatch_disable (Thread_queue_Context *queue_context)

  *Disables dispatching in a critical section.*

- static __inline__ void _Thread_queue_Gate_close (Thread_queue_Gate *gate)

  *Closes the gate.*

- static __inline__ void _Thread_queue_Gate_add (Chain_Control *chain, Thread_queue_Gate *gate)

  *Adds the gate to the chain.*

- static __inline__ void _Thread_queue_Gate_open (Thread_queue_Gate *gate)

  *Opens the gate.*

- static __inline__ void _Thread_queue_Gate_wait (Thread_queue_Gate *gate)

  *Waits on a gate to open.*

- static __inline__ void _Thread_queue_Heads_initialize (Thread_queue_Heads *heads)

*Initializes the thread queue heads.*

- static __inline__ void _Thread_queue_Queue_initialize (Thread_queue_Queue *queue, const char *name)

  *Initializes the thread queue queue with the given name.*

- static __inline__ void _Thread_queue_Queue_do_acquire_critical (Thread_queue_Queue *queue, ISR_lock_Context *lock_context)

  *Acquires the thread queue queue in a critical section.*

- static __inline__ void _Thread_queue_Queue_release_critical (Thread_queue_Queue *queue, ISR_lock_Context *lock_context)

  *Releases the thread queue queue in a critical section.*

- static __inline__ void _Thread_queue_Queue_release (Thread_queue_Queue *queue, ISR_lock_Context *lock_context)

  *Releases the thread queue queue and enables interrupts.*

- size_t _Thread_queue_Queue_get_name_and_id (const Thread_queue_Queue *queue, char *buffer, size_t buffer_size, Objects_Id *id)

  *Copies the thread queue name to the specified buffer.*

- void _Thread_queue_Do_acquire_critical (Thread_queue_Control *the_thread_queue, ISR_lock_Context *lock_context)

  *Acquires the thread queue control in a critical section.*

- static __inline__ void _Thread_queue_Acquire_critical (Thread_queue_Control *the_thread_queue, Thread_queue_Context *queue_context)

  *Acquires the thread queue control in a critical section.*

- void _Thread_queue_Acquire (Thread_queue_Control *the_thread_queue, Thread_queue_Context *queue_context)

  *Acquires the thread queue control in a critical section.*

- void _Thread_queue_Do_release_critical (Thread_queue_Control *the_thread_queue, ISR_lock_Context *lock_context)

  *Checks if the thread queue control is the owner of the lock.*

- static __inline__ void _Thread_queue_Release_critical (Thread_queue_Control *the_thread_queue, Thread_queue_Context *queue_context)

  *Releases the thread queue control in a critical section.*

- void _Thread_queue_Release (Thread_queue_Control *the_thread_queue, Thread_queue_Context *queue_context)

  *Releases the thread queue control and enables interrupts.*

- Thread_Control * _Thread_queue_Do_dequeue (Thread_queue_Control *the_thread_queue, const Thread_queue_Operations *operations)

  *Dequeues the first thread waiting on the thread queue and returns it.*

- void _Thread_queue_Enqueue (Thread_queue_Queue *queue, const Thread_queue_Operations *operations, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Blocks the thread and places it on the thread queue.*

- Status_Control _Thread_queue_Enqueue_sticky (Thread_queue_Queue *queue, const Thread_queue_Operations *operations, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Enqueues the thread on the thread queue and busy waits for dequeue.*

- bool _Thread_queue_Extract_locked (Thread_queue_Queue *queue, const Thread_queue_Operations *operations, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Extracts the thread from the thread queue, restores the default wait operations and restores the default thread lock.*

- void _Thread_queue_Unblock_critical (bool unblock, Thread_queue_Queue *queue, Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Unblocks the thread which was on the thread queue before.*

- void _Thread_queue_Extract_critical (Thread_queue_Queue *queue, const Thread_queue_Operations *operations, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Extracts the thread from the thread queue and unblocks it.*

- void _Thread_queue_Extract (Thread_Control *the_thread)

  *Extracts thread from thread queue.*

- void _Thread_queue_Extract_with_proxy (Thread_Control *the_thread)

  *Extracts the_thread from the_thread_queue.*
- void _Thread_queue_Surrender (Thread_queue_Queue *queue, Thread_queue_Heads *heads, Thread_Control *previous_owner, Thread_queue_Context *queue_context, const Thread_queue_Operations *operations)

  *Surrenders the thread queue previously owned by the thread to the first enqueued thread.*
- void _Thread_queue_Surrender_sticky (Thread_queue_Queue *queue, Thread_queue_Heads *heads, Thread_Control *previous_owner, Thread_queue_Context *queue_context, const Thread_queue_Operations *operations)

  *Surrenders the thread queue previously owned by the thread to the first enqueued thread.*
- static __inline__ bool _Thread_queue_Is_empty (const Thread_queue_Queue *queue)

  *Checks if the thread queue queue is empty.*
- static __inline__ Thread_Control * _Thread_queue_First_locked (Thread_queue_Control *the_thread_queue, const Thread_queue_Operations *operations)

  *Returns the first thread on the thread queue if it exists, otherwise NULL.*
- Thread_Control * _Thread_queue_First (Thread_queue_Control *the_thread_queue, const Thread_queue_Operations *operations)

  *Returns the first thread on the thread queue if it exists, otherwise NULL.*
- Thread_Control * _Thread_queue_Flush_default_filter (Thread_Control *the_thread, Thread_queue_Queue *queue, Thread_queue_Context *queue_context)

  *Default thread queue flush filter function.*
- Thread_Control * _Thread_queue_Flush_status_unavailable (Thread_Control *the_thread, Thread_queue_Queue *queue, Thread_queue_Context *queue_context)

  *Status unavailable thread queue flush filter function.*
- Thread_Control * _Thread_queue_Flush_status_object_was_deleted (Thread_Control *the_thread, Thread_queue_Queue *queue, Thread_queue_Context *queue_context)

  *Status object was deleted thread queue flush filter function.*
- size_t _Thread_queue_Flush_critical (Thread_queue_Queue *queue, const Thread_queue_Operations *operations, Thread_queue_Flush_filter filter, Thread_queue_Context *queue_context)

  *Unblocks all threads enqueued on the thread queue.*
- void _Thread_queue_Initialize (Thread_queue_Control *the_thread_queue, const char *name)

  *Initializes the thread queue control to the given name.*
- static __inline__ void _Thread_queue_Destroy (Thread_queue_Control *the_thread_queue)

  *Destroys the thread queue.*
- bool _Thread_queue_Path_acquire_critical (Thread_queue_Queue *queue, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Does nothing.*
- void _Thread_queue_Path_release_critical (Thread_queue_Context *queue_context)

  *Releases the thread queue path in a critical section.*
- void _Thread_queue_Object_initialize (Thread_queue_Control *the_thread_queue)

  *Initializes a thread queue embedded in an object with identifier.*

## Variables

- const Thread_queue_Operations **_Thread_queue_Operations_default**
- const Thread_queue_Operations **_Thread_queue_Operations_FIFO**
- const Thread_queue_Operations **_Thread_queue_Operations_priority**
- const Thread_queue_Operations **_Thread_queue_Operations_priority_inherit**
- const char _Thread_queue_Object_name [ ]

  *The special thread queue name to indicated that the thread queue is embedded in an object with identifier.*

### 10.203.1 Detailed Description

Constants and Structures Associated with the Manipulation of Objects.

This include file contains all the constants and structures associated with the manipulation of objects.

## 10.204 cpukit/include/rtems/score/timecounter.h File Reference

Timecounter API.

```
#include <sys/time.h>
#include <sys/timetc.h>
#include <machine/_timecounter.h>
#include <rtems/score/isrlock.h>
```

### Macros

- #define _Timecounter_Acquire(lock_context) _ISR_lock_ISR_disable_and_acquire( &_Timecounter_Lock, lock_context )

  *Lock to protect the timecounter mechanic.*
- #define _Timecounter_Release(lock_context) _ISR_lock_Release_and_ISR_enable(&_Timecounter_Lock, lock_context)

  *Releases the timecounter lock.*

### Functions

- void _Timecounter_Bintime (struct bintime *bt)

  *Returns the wall clock time in the bintime format.*
- void _Timecounter_Nanotime (struct timespec *ts)

  *Returns the wall clock time in the timespec format.*
- void _Timecounter_Microtime (struct timeval *tv)

  *Returns the wall clock time in the timeval format.*
- void _Timecounter_Binuptime (struct bintime *bt)

  *Returns the uptime in the bintime format.*
- int64_t _Timecounter_Sbinuptime (void)

  *Returns the uptime in the sbintime_t format.*
- void _Timecounter_Nanouptime (struct timespec *ts)

  *Returns the uptime in the timespec format.*
- void _Timecounter_Microuptime (struct timeval *tv)

  *Returns the uptime in the timeval format.*
- void _Timecounter_Getbintime (struct bintime *bt)

  *Returns the wall clock time in the bintime format.*
- void _Timecounter_Getnanotime (struct timespec *ts)

  *Returns the wall clock time in the timespec format.*
- void _Timecounter_Getmicrotime (struct timeval *tv)

  *Returns the wall clock time in the timeval format.*
- void _Timecounter_Getbinuptime (struct bintime *bt)

  *Returns the uptime in the bintime format.*

- void _Timecounter_Getnanouptime (struct timespec *ts)

  *Returns the uptime in the timespec format.*
- void _Timecounter_Getmicrouptime (struct timeval *tv)

  *Returns the uptime in the timeval format.*
- void _Timecounter_Getboottime (struct timeval *tv)

  *Returns the boot time in the timeval format.*
- void _Timecounter_Getboottimebin (struct bintime *bt)

  *Returns the boot time in the bintime format.*
- void _Timecounter_Install (struct timecounter *tc)

  *Installs the timecounter.*
- void _Timecounter_Tick (void)

  *Performs a timecounter tick.*
- void _Timecounter_Tick_simple (uint32_t delta, uint32_t offset, ISR_lock_Context *lock_context)

  *Performs a simple timecounter tick.*

### Variables

- volatile time_t _Timecounter_Time_second

  *The wall clock time in seconds.*
- volatile int32_t _Timecounter_Time_uptime

  *The uptime in seconds.*
- struct timecounter * _Timecounter

  *The current timecounter.*

#### 10.204.1 Detailed Description

Timecounter API.

## 10.205 cpukit/include/rtems/timecounter.h File Reference

Timecounter API.

```
#include <rtems/score/timecounter.h>
#include <rtems/score/basedefs.h>
```

### Classes

- struct rtems_timecounter_simple

  *Simple timecounter to support legacy clock drivers.*

### Macros

- #define RTEMS_TIMECOUNTER_QUALITY_CLOCK_DRIVER 100

  *Timecounter quality for the clock drivers.*

### Typedefs

- typedef void rtems_timecounter_simple_at_tick(rtems_timecounter_simple *tc)

  *At tick handling done under protection of the timecounter lock.*
- typedef uint32_t rtems_timecounter_simple_get(rtems_timecounter_simple *tc)

  *Returns the current value of a simple timecounter.*
- typedef bool rtems_timecounter_simple_is_pending(rtems_timecounter_simple *tc)

  *Returns true if the interrupt of a simple timecounter is pending, and false otherwise.*

### Functions

- static __inline__ void rtems_timecounter_install (struct timecounter *tc)

  *Installs the timecounter.*
- static __inline__ void rtems_timecounter_tick (void)

  *Performs a timecounter tick.*
- void rtems_timecounter_simple_install (rtems_timecounter_simple *tc, uint32_t counter_frequency_in_hz, uint32_t counter_ticks_per_clock_tick, timecounter_get_t *get_timecount)

  *Initializes and installs a simple timecounter.*
- static __inline__ uint32_t rtems_timecounter_simple_scale (const rtems_timecounter_simple *tc, uint32_t value)

  *Maps a simple timecounter value into its binary frequency domain.*
- static __inline__ void rtems_timecounter_simple_downcounter_tick (rtems_timecounter_simple *tc, rtems_timecounter_simple_get get, rtems_timecounter_simple_at_tick at_tick)

  *Performs a simple timecounter tick for downcounters.*
- static __inline__ void rtems_timecounter_simple_upcounter_tick (rtems_timecounter_simple *tc, rtems_timecounter_simple_ge get, rtems_timecounter_simple_at_tick at_tick)

  *Performs a simple timecounter tick for upcounters.*
- static __inline__ uint32_t rtems_timecounter_simple_downcounter_get (struct timecounter *tc_base, rtems_timecounter_simple_get get, rtems_timecounter_simple_is_pending is_pending)

  *Gets the simple timecounter value mapped to its binary frequency domain for downcounters.*
- static __inline__ uint32_t rtems_timecounter_simple_upcounter_get (struct timecounter *tc_base, rtems_timecounter_simple_get get, rtems_timecounter_simple_is_pending is_pending)

  *Gets the simple timecounter value mapped to its binary frequency domain for upcounters.*

#### 10.205.1 Detailed Description

Timecounter API.

## 10.206 cpukit/include/rtems/score/timecounterimpl.h File Reference

Timecounter Implementation.

```
#include <rtems/score/timecounter.h>
#include <sys/timetc.h>
```

### Functions

- void _Timecounter_Set_clock (const struct bintime *bt, ISR_lock_Context *lock_context)

  *Sets the timecounter clock to the given value.*

### 10.206.1 Detailed Description

Timecounter Implementation.

## 10.207 cpukit/include/rtems/score/timespec.h File Reference

Contains Helpers for Manipulating Timespecs.

```
#include <stdbool.h>
#include <stdint.h>
#include <time.h>
```

### Macros

- #define _Timespec_Set(_time, _seconds, _nanoseconds)

  *Set timespec to seconds nanosecond.*
- #define _Timespec_Set_to_zero(_time)

  *Sets the Timespec to Zero.*
- #define _Timespec_Get_seconds(_time) ((_time)->tv_sec)

  *Get seconds portion of timespec.*
- #define _Timespec_Get_nanoseconds(_time) ((_time)->tv_nsec)

  *Get nanoseconds portion of timespec.*
- #define _Timespec_Greater_than(_lhs, _rhs) _Timespec_Less_than( _rhs, _lhs )

  *The Timespec "greater than" operator.*
- #define _Timespec_Equal_to(lhs, rhs)

  *The Timespec "equal to" operator.*

### Functions

- uint64_t _Timespec_Get_as_nanoseconds (const struct timespec *time)

  *Gets the timestamp as nanoseconds.*
- bool _Timespec_Is_valid (const struct timespec *time)

  *Checks if timespec is valid.*
- bool _Timespec_Less_than (const struct timespec *lhs, const struct timespec *rhs)

  *Checks if the left hand side timespec is less than the right one.*
- uint32_t _Timespec_Add_to (struct timespec *time, const struct timespec *add)

  *Adds two timespecs.*
- uint32_t _Timespec_To_ticks (const struct timespec *time)

  *Converts timespec to number of ticks.*
- void _Timespec_From_ticks (uint32_t ticks, struct timespec *time)

  *Converts ticks to timespec.*
- void _Timespec_Subtract (const struct timespec *start, const struct timespec *end, struct timespec *result)

  *Subtracts two timespec.*
- void _Timespec_Divide_by_integer (const struct timespec *time, uint32_t iterations, struct timespec *result)

  *Divides timespec by an integer.*
- void _Timespec_Divide (const struct timespec *lhs, const struct timespec *rhs, uint32_t *ival_percentage, uint32_t *fval_percentage)

  *Divides a timespec by another timespec.*

### 10.207.1 Detailed Description

Contains Helpers for Manipulating Timespecs.

## 10.208 cpukit/include/rtems/score/timestamp.h File Reference

Helpers for Manipulating Timestamps.

```
#include <rtems/score/timespec.h>
```

### Typedefs

- typedef int64_t Timestamp_Control

### 10.208.1 Detailed Description

Helpers for Manipulating Timestamps.

This include file contains helpers for manipulating timestamps.

## 10.209 cpukit/include/rtems/score/timestampimpl.h File Reference

Helpers for Manipulating Timestamps.

```
#include <rtems/score/timestamp.h>
#include <rtems/score/basedefs.h>
#include <sys/time.h>
```

### Functions

- static __inline__ void _Timestamp_Set (Timestamp_Control *_time, time_t _seconds, long _nanoseconds)

  *Sets timestamp to specified seconds and nanoseconds.*
- static __inline__ void _Timestamp_Set_to_zero (Timestamp_Control *_time)

  *Sets the timestamp to zero.*
- static __inline__ bool _Timestamp_Less_than (const Timestamp_Control *_lhs, const Timestamp_Control *_rhs)

  *Checks if the left hand side timestamp is less than the right one.*
- static __inline__ bool _Timestamp_Greater_than (const Timestamp_Control *_lhs, const Timestamp_Control *_rhs)

  *Checks if the left hand side timestamp is greater than the right one.*
- static __inline__ bool _Timestamp_Equal_to (const Timestamp_Control *_lhs, const Timestamp_Control *↩ _rhs)

  *Checks if the timestamps are equal.*
- static __inline__ void _Timestamp_Add_to (Timestamp_Control *_time, const Timestamp_Control *_add)

*Adds two timestamps.*

- static __inline__ void _Timestamp_Subtract (const Timestamp_Control *_start, const Timestamp_Control *_end, Timestamp_Control *_result)

  *Subtracts two timestamps.*

- static __inline__ void _Timestamp_Divide (const Timestamp_Control *_lhs, const Timestamp_Control *_rhs, uint32_t *_ival_percentage, uint32_t *_fval_percentage)

  *Divides a timestamp by another timestamp.*

- static __inline__ time_t _Timestamp_Get_seconds (const Timestamp_Control *_time)

  *Gets seconds portion of timestamp.*

- static __inline__ uint32_t _Timestamp_Get_nanoseconds (const Timestamp_Control *_time)

  *Gets nanoseconds portion of timestamp.*

- static __inline__ uint64_t _Timestamp_Get_as_nanoseconds (const Timestamp_Control *_time)

  *Gets the timestamp as nanoseconds.*

- static __inline__ void _Timestamp_To_timespec (const Timestamp_Control *_timestamp, struct timespec *_timespec)

  *Converts timestamp to struct timespec.*

- static __inline__ void _Timestamp_To_timeval (const Timestamp_Control *_timestamp, struct timeval *_timeval)

  *Converts timestamp to struct timeval.*

### 10.209.1 Detailed Description

Helpers for Manipulating Timestamps.

This include file contains helpers for manipulating timestamps.

## 10.210 cpukit/include/rtems/score/tls.h File Reference

Thread-Local Storage (TLS)

```
#include <rtems/score/cpu.h>
#include <string.h>
```

### Classes

- struct TLS_Dynamic_thread_vector
- struct TLS_Thread_control_block
- struct TLS_Index

### Typedefs

- typedef struct TLS_Thread_control_block **TLS_Thread_control_block**

### Functions

- static uintptr_t _TLS_Get_size (void)

  *Gets the TLS size.*
- static uintptr_t _TLS_Heap_align_up (uintptr_t val)

  *Returns the value aligned up to the heap alignment.*
- static uintptr_t _TLS_Get_thread_control_block_area_size (uintptr_t alignment)

  *Returns the size of the thread control block area size for this alignment, or the minimum size if alignment is too small.*
- uintptr_t _TLS_Get_allocation_size (void)

  *Return the TLS area allocation size.*
- static void * _TLS_Copy_and_clear (void *tls_area)

  *Copies TLS size bytes from the address tls_area and returns a pointer to the start of the area after clearing it.*
- static void * _TLS_Initialize (void *tls_block, TLS_Thread_control_block *tcb, TLS_Dynamic_thread_vector *dtv)

  *Initializes the dynamic thread vector.*
- static void * _TLS_TCB_at_area_begin_initialize (void *tls_area)

  *Initializes a dynamic thread vector beginning at the given starting address.*
- static void * _TLS_TCB_before_TLS_block_initialize (void *tls_area)

  *Initializes a dynamic thread vector with the area before a given starting address as thread control block.*
- static void * _TLS_TCB_after_TLS_block_initialize (void *tls_area)

  *Initializes a dynamic thread vector with the area after a given starting address as thread control block.*

### Variables

- char **_TLS_Data_begin** [ ]
- char **_TLS_Data_end** [ ]
- char **_TLS_Data_size** [ ]
- char **_TLS_BSS_begin** [ ]
- char **_TLS_BSS_end** [ ]
- char **_TLS_BSS_size** [ ]
- char **_TLS_Size** [ ]
- char _TLS_Alignment [ ]

  *The TLS section alignment.*

#### 10.210.1 Detailed Description

Thread-Local Storage (TLS)

## 10.211 cpukit/include/rtems/score/todimpl.h File Reference

Time of Day Handler API.

```
#include <rtems/score/status.h>
#include <rtems/score/timestamp.h>
#include <rtems/score/timecounterimpl.h>
#include <rtems/score/watchdog.h>
#include <sys/time.h>
#include <time.h>
```

## Classes

- struct TOD_Control

  *TOD control.*
- struct TOD_Hook

  *Structure to manage each TOD action hook.*

## Macros

- #define TOD_SECONDS_PER_MINUTE (uint32_t)60
- #define TOD_MINUTES_PER_HOUR (uint32_t)60
- #define TOD_MONTHS_PER_YEAR (uint32_t)12
- #define TOD_DAYS_PER_YEAR (uint32_t)365
- #define TOD_HOURS_PER_DAY (uint32_t)24
- #define TOD_SECONDS_PER_DAY
- #define TOD_SECONDS_PER_NON_LEAP_YEAR (365 * TOD_SECONDS_PER_DAY)
- #define TOD_MILLISECONDS_PER_SECOND (uint32_t)1000
- #define TOD_MICROSECONDS_PER_SECOND (uint32_t)1000000
- #define TOD_NANOSECONDS_PER_SECOND (uint32_t)1000000000
- #define TOD_NANOSECONDS_PER_MICROSECOND (uint32_t)1000
- #define TOD_SECONDS_1970_THROUGH_1988
- #define TOD_BASE_YEAR 1988

  *Earliest year to which an time of day can be initialized.*
- #define TOD_TICKS_PER_SECOND TOD_TICKS_PER_SECOND_method()

  *Gets number of ticks in a second.*

## Typedefs

- typedef struct TOD_Hook TOD_Hook

  *Structure to manage each TOD action hook.*

## Enumerations

- enum TOD_Action { TOD_ACTION_SET_CLOCK }

  *Possible actions where a registered hook could be invoked.*

## Functions

- void _TOD_Lock (void)

  *Locks the time of day mutex.*
- void _TOD_Unlock (void)

  *Unlocks the time of day mutex.*
- static void _TOD_Acquire (ISR_lock_Context *lock_context)

  *Checks if api mutex is owner of the time of day mutex.*
- static void _TOD_Release (ISR_lock_Context *lock_context)

  *Releases the lock context for the timecounter.*
- Status_Control _TOD_Set (const struct timespec *tod, ISR_lock_Context *lock_context)

  *Sets the time of day.*
- static void _TOD_Get (struct timespec *tod)

*Gets the current time in the timespec format.*

- static void _TOD_Get_uptime (Timestamp_Control *time)

  *Gets the system uptime with potential accuracy to the nanosecond.*
- static void _TOD_Get_zero_based_uptime (Timestamp_Control *time)

  *Gets the system uptime with potential accuracy to the nanosecond.*
- static void _TOD_Get_zero_based_uptime_as_timespec (struct timespec *time)

  *Gets the system uptime with potential accuracy to the nanosecond.*
- static uint32_t _TOD_Seconds_since_epoch (void)

  *Returns Number of seconds Since RTEMS epoch.*
- uint32_t TOD_TICKS_PER_SECOND_method (void)

  *Gets number of ticks in a second.*
- static __inline__ void _TOD_Get_timeval (struct timeval *time)

  *This routine returns a timeval based upon the internal timespec format TOD.*
- void _TOD_Adjust (const struct timespec *delta)

  *Adjusts the Time of Time.*
- static __inline__ bool _TOD_Is_set (void)

  *Check if the TOD is Set.*
- void _TOD_Hook_Register (TOD_Hook *hook)

  *Add a TOD Action Hook.*
- void _TOD_Hook_Unregister (TOD_Hook *hook)

  *Remove a TOD Action Hook.*
- Status_Control _TOD_Hook_Run (TOD_Action action, const struct timespec *tod)

  *Run the TOD Action Hooks.*

## Variables

- TOD_Control _TOD

  *TOD Management information.*
- Chain_Control _TOD_Hooks

  *Set of registered methods for TOD Actions.*

### 10.211.1 Detailed Description

Time of Day Handler API.

### 10.211.2 Macro Definition Documentation

#### 10.211.2.1 TOD_BASE_YEAR

```
#define TOD_BASE_YEAR 1988
```

Earliest year to which an time of day can be initialized.

The following constant define the earliest year to which an time of day can be initialized. This is considered the epoch.

Definition at line 122 of file todimpl.h.

#### 10.211.2.2 TOD_SECONDS_1970_THROUGH_1988

```
#define TOD_SECONDS_1970_THROUGH_1988
```

**Value:**

```
(((1987 - 1970 + 1)  * TOD_SECONDS_PER_NON_LEAP_YEAR) + \
(4 * TOD_SECONDS_PER_DAY))
```

Seconds from January 1, 1970 to January 1, 1988. Used to account for differences between POSIX API and RTEMS core. The timespec format time is kept in POSIX compliant form.

Definition at line 111 of file todimpl.h.

## 10.212 cpukit/include/rtems/score/userext.h File Reference

User Extension Handler API.

```
#include <rtems/score/interr.h>
```

### Classes

- struct User_extensions_Table

  *User extension table.*

### Typedefs

- typedef bool(* User_extensions_thread_create_extension) (struct _Thread_Control *executing, struct _Thread_Control *created)

  *Task create extension.*
- typedef void(* User_extensions_thread_delete_extension) (struct _Thread_Control *executing, struct _Thread_Control *deleted)

  *Task delete extension.*
- typedef void(* User_extensions_thread_start_extension) (struct _Thread_Control *executing, struct _Thread_Control *started)

  *Task start extension.*
- typedef void(* User_extensions_thread_restart_extension) (struct _Thread_Control *executing, struct _Thread_Control *restarted)

  *Task restart extension.*
- typedef void(* User_extensions_thread_switch_extension) (struct _Thread_Control *executing, struct _Thread_Control *heir)

  *Task switch extension.*
- typedef void(* User_extensions_thread_begin_extension) (struct _Thread_Control *executing)

  *Task begin extension.*
- typedef void(* User_extensions_thread_exitted_extension) (struct _Thread_Control *executing)

  *Task exitted extension.*
- typedef void(* User_extensions_fatal_extension) (Internal_errors_Source source, bool always_set_to_false, Internal_errors_t code)

  *Fatal error extension.*
- typedef void(* User_extensions_thread_terminate_extension) (struct _Thread_Control *terminated)

  *Task termination extension.*

### 10.212.1 Detailed Description

User Extension Handler API.

## 10.213 cpukit/include/rtems/score/userextdata.h File Reference

User Extension Handler Data Structures.

```
#include <rtems/score/userext.h>
#include <rtems/score/chain.h>
```

### Classes

- struct User_extensions_Switch_control

  *Manages the switch callouts.*
- struct User_extensions_Control

  *Manages each user extension set.*

### Variables

- const size_t _User_extensions_Initial_count

  *The count of initial user extensions.*
- const User_extensions_Table _User_extensions_Initial_extensions [ ]

  *The table of initial user extensions.*
- User_extensions_Switch_control _User_extensions_Initial_switch_controls [ ]

  *A spare switch control for each initial user extension.*

### 10.213.1 Detailed Description

User Extension Handler Data Structures.

## 10.214 cpukit/include/rtems/score/userextimpl.h File Reference

User Extension Handler API.

```
#include <rtems/score/userextdata.h>
#include <rtems/score/chainimpl.h>
#include <rtems/score/isrlock.h>
#include <rtems/score/thread.h>
#include <rtems/score/percpu.h>
```

## Classes

- struct User_extensions_Iterator

  *Chain iterator for dynamic user extensions.*
- struct User_extensions_List
- struct User_extensions_Thread_create_context
- struct User_extensions_Fatal_context

## Typedefs

- typedef struct User_extensions_Iterator User_extensions_Iterator

  *Chain iterator for dynamic user extensions.*

## Functions

### Extension Callout Dispatcher

- static bool _User_extensions_Thread_create (Thread_Control *created)

  *Creates a thread.*
- static void _User_extensions_Thread_delete (Thread_Control *deleted)

  *Deletes a thread.*
- static void _User_extensions_Thread_start (Thread_Control *started)

  *Starts a thread.*
- static void _User_extensions_Thread_restart (Thread_Control *restarted)

  *Restarts a thread.*
- static void _User_extensions_Thread_begin (Thread_Control *executing)

  *Begins a thread.*
- static void _User_extensions_Thread_switch (Thread_Control *executing, Thread_Control *heir)

  *Switches the thread from the executing to the heir.*
- static void _User_extensions_Thread_exitted (Thread_Control *executing)

  *A user extension thread exitted.*
- static void _User_extensions_Fatal (Internal_errors_Source source, Internal_errors_t error)

  *Forwards all visitors that there was a fatal failure.*
- static void _User_extensions_Thread_terminate (Thread_Control *executing)

  *Terminates the executing thread.*
- static void _User_extensions_Acquire (ISR_lock_Context *lock_context)

  *Disables interrupts and acquires the lock context.*
- static void _User_extensions_Release (ISR_lock_Context *lock_context)

  *Releases the lock context and enables interrupts.*
- static void _User_extensions_Destroy_iterators (Thread_Control *the_thread)

  *Destroys all user extension iterators of a thread.*

## Variables

- User_extensions_List _User_extensions_List

  *List of active extensions.*
- Chain_Control _User_extensions_Switches_list

  *List of active task switch extensions.*

### Extension Maintainance

- typedef void(∗ User_extensions_Visitor) (Thread_Control ∗executing, void ∗arg, const User_extensions_Table ∗callouts)

  *User extension visitor.*
- void _User_extensions_Handler_initialization (void)

  *Initializes the user extensions handler.*
- void _User_extensions_Add_set (User_extensions_Control ∗extension)

  *Adds a user extension.*
- static __inline__ void _User_extensions_Add_API_set (User_extensions_Control ∗extension)

  *Adds a user extension.*
- static __inline__ void _User_extensions_Add_set_with_table (User_extensions_Control ∗extension, const User_extensions_Table ∗extension_table)

  *Adds a user extension with the given extension table as callouts.*
- void _User_extensions_Remove_set (User_extensions_Control ∗extension)

  *Removes a user extension.*
- void _User_extensions_Thread_create_visitor (Thread_Control ∗executing, void ∗arg, const User_extensions_Table ∗callouts)

  *Creates a visitor.*
- void _User_extensions_Thread_delete_visitor (Thread_Control ∗executing, void ∗arg, const User_extensions_Table ∗callouts)

  *Deletes a visitor.*
- void _User_extensions_Thread_start_visitor (Thread_Control ∗executing, void ∗arg, const User_extensions_Table ∗callouts)

  *Starts a visitor.*
- void _User_extensions_Thread_restart_visitor (Thread_Control ∗executing, void ∗arg, const User_extensions_Table ∗callouts)

  *Restarts a visitor.*
- void _User_extensions_Thread_begin_visitor (Thread_Control ∗executing, void ∗arg, const User_extensions_Table ∗callouts)

  *Calls the begin function of the thread callout for the visitor.*
- void _User_extensions_Thread_exitted_visitor (Thread_Control ∗executing, void ∗arg, const User_extensions_Table ∗callouts)

  *Calls the exitted function of the thread callout for the visitor.*
- void _User_extensions_Fatal_visitor (Thread_Control ∗executing, void ∗arg, const User_extensions_Table ∗callouts)

  *Calls the fatal function of the thread callout for the visitor.*
- void _User_extensions_Thread_terminate_visitor (Thread_Control ∗executing, void ∗arg, const User_extensions_Table ∗callouts)

  *Terminates a visitor.*
- void _User_extensions_Iterate (void ∗arg, User_extensions_Visitor visitor, Chain_Iterator_direction direction)

  *Iterates through all user extensions and calls the visitor for each.*

## 10.214.1 Detailed Description

User Extension Handler API.

## 10.215 cpukit/include/rtems/score/watchdog.h File Reference

Constants and Structures Associated with Watchdog Timers.

```
#include <rtems/score/basedefs.h>
#include <rtems/score/chain.h>
#include <rtems/score/rbtree.h>
```

### Classes

- struct Watchdog_Header

  *The watchdog header to manage scheduled watchdogs.*
- struct Watchdog_Control

  *The control block used to manage each watchdog timer.*

### Typedefs

- typedef struct Watchdog_Control **Watchdog_Control**
- typedef void Watchdog_Service_routine

  *Return type from a Watchdog Service Routine.*
- typedef Watchdog_Service_routine(* Watchdog_Service_routine_entry) (Watchdog_Control *)

  *Pointer to a watchdog service routine.*

### 10.215.1 Detailed Description

Constants and Structures Associated with Watchdog Timers.

This include file contains all the constants and structures associated with watchdog timers. This Handler provides mechanisms which can be used to initialize and manipulate watchdog timers.

## 10.216 cpukit/include/rtems/score/watchdogimpl.h File Reference

Inlined Routines in the Watchdog Handler.

```
#include <rtems/score/watchdog.h>
#include <rtems/score/watchdogticks.h>
#include <rtems/score/assert.h>
#include <rtems/score/isrlock.h>
#include <rtems/score/percpu.h>
#include <rtems/score/rbtreeimpl.h>
#include <sys/types.h>
#include <sys/timespec.h>
```

## Macros

- #define WATCHDOG_INITIALIZER(routine)

  *Watchdog initializer for static initialization.*
- #define **_Watchdog_Tickle**(header, first, now, lock, lock_context) _Watchdog_Do_tickle( header, first, now, lock, lock_context )
- #define WATCHDOG_MAXIMUM_TICKS UINT64_MAX

  *The maximum watchdog ticks value for the far future.*
- #define **WATCHDOG_NANOSECONDS_PER_SECOND** 1000000000
- #define WATCHDOG_BITS_FOR_1E9_NANOSECONDS 30

  *The bits necessary to store 1000000000 (= WATCHDOG_NANOSECONDS_PER_SECOND) nanoseconds.*
- #define WATCHDOG_MAX_SECONDS 0x3fffffff

  *The maximum number of seconds representable in the nanoseconds watchdog format.*

## Enumerations

- enum Watchdog_State { WATCHDOG_SCHEDULED_BLACK, WATCHDOG_SCHEDULED_RED, WATCHDOG_INACTIVE, WATCHDOG_PENDING }

  *Watchdog states.*

## Functions

- static __inline__ void _Watchdog_Header_initialize (Watchdog_Header *header)

  *Initializes the watchdog header.*
- static __inline__ Watchdog_Control * _Watchdog_Header_first (const Watchdog_Header *header)

  *Returns the first of the watchdog header.*
- static __inline__ void _Watchdog_Header_destroy (Watchdog_Header *header)

  *Destroys the watchdog header.*
- void _Watchdog_Tick (struct Per_CPU_Control *cpu)

  *Performs a watchdog tick.*
- static __inline__ Watchdog_State _Watchdog_Get_state (const Watchdog_Control *the_watchdog)

  *Gets the state of the watchdog.*
- static __inline__ void _Watchdog_Set_state (Watchdog_Control *the_watchdog, Watchdog_State state)

  *Sets the state of the watchdog.*
- static __inline__ Per_CPU_Control * _Watchdog_Get_CPU (const Watchdog_Control *the_watchdog)

  *Gets the watchdog's cpu.*
- static __inline__ void _Watchdog_Set_CPU (Watchdog_Control *the_watchdog, Per_CPU_Control *cpu)

  *Sets the cpu for the watchdog.*
- static __inline__ void _Watchdog_Preinitialize (Watchdog_Control *the_watchdog, Per_CPU_Control *cpu)

  *Pre-initializes a watchdog.*
- static __inline__ void _Watchdog_Initialize (Watchdog_Control *the_watchdog, Watchdog_Service_routine_entry routine)

  *Initializes a watchdog with a new service routine.*
- void _Watchdog_Do_tickle (Watchdog_Header *header, Watchdog_Control *first, uint64_t now, ISR_lock_Control *lock, ISR_lock_Context *lock_context)

  *Calls the routine of each not expired watchdog control node.*
- void _Watchdog_Insert (Watchdog_Header *header, Watchdog_Control *the_watchdog, uint64_t expire)

  *Inserts a watchdog into the set of scheduled watchdogs according to the specified expiration time.*
- void _Watchdog_Remove (Watchdog_Header *header, Watchdog_Control *the_watchdog)

  *In the case the watchdog is scheduled, then it is removed from the set of scheduled watchdogs.*

- static __inline__ uint64_t _Watchdog_Cancel (Watchdog_Header *header, Watchdog_Control *the_watchdog, uint64_t now)

  *In the case the watchdog is scheduled, then it is removed from the set of scheduled watchdogs.*
- static __inline__ bool _Watchdog_Is_scheduled (const Watchdog_Control *the_watchdog)

  *Checks if the watchdog is scheduled.*
- static __inline__ void _Watchdog_Next_first (Watchdog_Header *header, Watchdog_Control *the_watchdog)

  *Sets the first node of the header.*
- static __inline__ bool _Watchdog_Is_valid_timespec (const struct timespec *ts)

  *Checks if the timespec is a valid timespec for a watchdog.*
- static __inline__ bool _Watchdog_Is_valid_interval_timespec (const struct timespec *ts)

  *Checks if the timespec is a valid interval timespec for a watchdog.*
- static __inline__ const struct timespec * _Watchdog_Future_timespec (struct timespec *now, const struct timespec *delta)

  *Adds the delta timespec to the current time if the delta is a valid interval timespec.*
- static __inline__ bool _Watchdog_Is_far_future_timespec (const struct timespec *ts)

  *Checks if the timespec is too far in the future.*
- static __inline__ uint64_t _Watchdog_Ticks_from_seconds (uint32_t seconds)

  *Converts the seconds to ticks.*
- static __inline__ uint64_t _Watchdog_Ticks_from_timespec (const struct timespec *ts)

  *Converts the timespec in ticks.*
- static __inline__ uint64_t _Watchdog_Ticks_from_sbintime (int64_t sbt)

  *Converts the sbintime in ticks.*
- static __inline__ void _Watchdog_Per_CPU_acquire_critical (Per_CPU_Control *cpu, ISR_lock_Context *lock_context)

  *Acquires the per cpu watchdog lock in a critical section.*
- static __inline__ void _Watchdog_Per_CPU_release_critical (Per_CPU_Control *cpu, ISR_lock_Context *lock_context)

  *Releases the per cpu watchdog lock in a critical section.*
- static __inline__ uint64_t _Watchdog_Per_CPU_insert_ticks (Watchdog_Control *the_watchdog, Per_CPU_Control *cpu, Watchdog_Interval ticks)

  *Sets the watchdog's cpu to the given instance and sets its expiration time to the watchdog expiration time of the cpu plus the ticks.*
- static __inline__ uint64_t _Watchdog_Per_CPU_insert (Watchdog_Control *the_watchdog, Per_CPU_Control *cpu, Watchdog_Header *header, uint64_t expire)

  *Sets the watchdog's cpu and inserts it with the given expiration time in the scheduled watchdogs.*
- static __inline__ void _Watchdog_Per_CPU_remove (Watchdog_Control *the_watchdog, Per_CPU_Control *cpu, Watchdog_Header *header)

  *Removes the watchdog from the cpu and the scheduled watchdogs.*
- static __inline__ void _Watchdog_Per_CPU_remove_ticks (Watchdog_Control *the_watchdog)

  *Removes the watchdog from the cpu and the scheduled watchdogs.*

### 10.216.1 Detailed Description

Inlined Routines in the Watchdog Handler.

This file contains the static inline implementation of all inlined routines in the Watchdog Handler.

# 10.217 cpukit/include/rtems/score/watchdogticks.h File Reference

Constants for the watchdog ticks.

```
#include <rtems/score/basedefs.h>
```

## Macros

- #define WATCHDOG_NO_TIMEOUT 0

  *Special watchdog ticks value to indicate an infinite wait.*
- #define WATCHDOG_TICKS_PER_TIMESLICE_DEFAULT 50

  *Default value for the watchdog ticks per timeslice.*

## Typedefs

- typedef uint32_t Watchdog_Interval

  *Type is used to specify the length of intervals.*

## Variables

- volatile Watchdog_Interval _Watchdog_Ticks_since_boot

  *The watchdog ticks counter.*
- const uint32_t _Watchdog_Microseconds_per_tick

  *The watchdog microseconds per tick.*
- const uint32_t _Watchdog_Nanoseconds_per_tick

  *The watchdog nanoseconds per tick.*
- const uint32_t _Watchdog_Ticks_per_second

  *The watchdog ticks per second.*
- const uint32_t _Watchdog_Ticks_per_timeslice

  *The watchdog ticks per timeslice.*

### 10.217.1 Detailed Description

Constants for the watchdog ticks.

# 10.218 cpukit/include/rtems/score/wkspacedata.h File Reference

Constants defined by the application configuration for the idle threads.

```
#include <rtems/score/basedefs.h>
```

### Functions

- struct Heap_Control * _Workspace_Malloc_initialize_separate (void)

  *Initializes the C Program Heap separated from the RTEMS Workspace.*
- struct Heap_Control * _Workspace_Malloc_initialize_unified (void)

  *Initializes the C Program Heap so that it is unified with the RTEMS Workspace.*

### Variables

- const uintptr_t _Workspace_Size

  *The workspace size in bytes.*
- const bool _Workspace_Is_unified

  *Indicates if the workspace and C program heap are unified.*
- struct Heap_Control *(*const _Workspace_Malloc_initializer )(void)

  *This constant provides the C Program Heap initialization handler.*

#### 10.218.1 Detailed Description

Constants defined by the application configuration for the idle threads.

## 10.219 cpukit/include/rtems/score/wkspaceinitone.h File Reference

Workspace Handler Initialization API.

```
#include <rtems/score/wkspace.h>
#include <rtems/score/assert.h>
#include <rtems/score/heapimpl.h>
#include <rtems/score/interr.h>
#include <rtems/score/memory.h>
#include <rtems/config.h>
```

### Functions

- static __inline__ void **_Workspace_Initialize_with_one_area** (void)

#### 10.219.1 Detailed Description

Workspace Handler Initialization API.

# 10.220 cpukit/include/rtems/termiostypes.h File Reference

```
#include <rtems.h>
#include <rtems/libio.h>
#include <rtems/assoc.h>
#include <rtems/chain.h>
#include <rtems/thread.h>
#include <sys/ioccom.h>
#include <stdint.h>
#include <termios.h>
```

## Classes

- struct ttywakeup
- struct rtems_termios_rawbuf
- struct rtems_termios_device_context

  *Termios device context.*
- struct rtems_termios_device_handler

  *Termios device handler.*
- struct rtems_termios_device_flow

  *Termios device flow control handler.*
- struct rtems_termios_device_node

  *Termios device node for installed devices.*
- struct rtems_termios_tty
- struct rtems_termios_linesw

## Macros

- #define RTEMS_TERMIOS_DEVICE_CONTEXT_INITIALIZER(name)

  *Initializer for static initialization of Termios device contexts.*
- #define **TTYDISC** 0 /* termios tty line discipline */
- #define **TABLDISC** 3 /* tablet discipline */
- #define **SLIPDISC** 4 /* serial IP discipline */
- #define **PPPDISC** 5 /* PPP discipline */
- #define **MAXLDISC** 8
- #define **RTEMS_IO_SNDWAKEUP** _IOW('t', 11, struct ttywakeup ) /* send tty wakeup */
- #define **RTEMS_IO_RCVWAKEUP** _IOW('t', 12, struct ttywakeup ) /* recv tty wakeup */
- #define **OLCUC** 0x00000100 /* map lower case to upper case on output */
- #define **IUCLC** 0x00004000 /* map upper case to lower case on input */
- #define **RTEMS_TERMIOS_NUMBER_BAUD_RATES** 25

## Typedefs

- typedef struct rtems_termios_device_context rtems_termios_device_context

  *Termios device context.*
- typedef struct rtems_termios_device_node rtems_termios_device_node

  *Termios device node for installed devices.*
- typedef struct rtems_termios_tty **rtems_termios_tty**
- typedef uint32_t **rtems_termios_baud_t**

## Enumerations

- enum **rtems_termios_device_mode** { **TERMIOS_POLLED**, **TERMIOS_IRQ_DRIVEN**, **TERMIOS_TASK_DRIVEN**, **TERMIOS_IRQ_SERVER_DRIVEN** }

## Functions

- void **rtems_termios_device_lock_acquire_default** (rtems_termios_device_context *ctx, rtems_interrupt_lock_context *lock_context)
- void **rtems_termios_device_lock_release_default** (rtems_termios_device_context *ctx, rtems_interrupt_lock_context *lock_context)
- static __inline__ void rtems_termios_device_context_initialize (rtems_termios_device_context *context, const char *name)

  *Initializes a device context.*
- rtems_status_code rtems_termios_device_install (const char *device_file, const rtems_termios_device_handler *handler, const rtems_termios_device_flow *flow, rtems_termios_device_context *context)

  *Installs a Termios device.*
- static __inline__ void * rtems_termios_get_device_context (const rtems_termios_tty *tty)

  *Returns the device context of an installed Termios device.*
- static __inline__ void rtems_termios_device_lock_acquire (rtems_termios_device_context *context, rtems_interrupt_lock_context *lock_context)

  *Acquires the device lock.*
- static __inline__ void rtems_termios_device_lock_release (rtems_termios_device_context *context, rtems_interrupt_lock_context *lock_context)

  *Releases the device lock.*
- void rtems_termios_set_best_baud (struct termios *term, uint32_t baud)

  *Sets the best baud value in the Termios control.*
- void **rtems_termios_rxirq_occured** (struct rtems_termios_tty *tty)
- void **rtems_termios_puts** (const void *buf, size_t len, struct rtems_termios_tty *tty)
- speed_t rtems_termios_number_to_baud (rtems_termios_baud_t baud)

  *Converts the Integral Baud value baud to the Termios Control Flag Representation.*
- rtems_termios_baud_t rtems_termios_baud_to_number (speed_t baud)

  *Converts the baud flags to an integral baud value.*
- int rtems_termios_baud_to_index (rtems_termios_baud_t termios_baud)

  *Convert Bxxx Constant to Index.*
- int rtems_termios_set_initial_baud (struct rtems_termios_tty *tty, rtems_termios_baud_t baud)

  *Sets the initial baud in the Termios context tty.*
- int rtems_termios_kqfilter (rtems_libio_t *iop, struct knote *kn)

  *Termios kqueue() filter filesystem node handler.*
- int rtems_termios_mmap (rtems_libio_t *iop, void **addr, size_t len, int prot, off_t off)

  *Termios mmap() filter filesystem node handler.*
- int rtems_termios_poll (rtems_libio_t *iop, int events)

  *Termios poll() filesystem node handler.*

## Variables

- struct rtems_termios_linesw **rtems_termios_linesw** [ ]
- int **rtems_termios_nlinesw**
- const rtems_assoc_t rtems_termios_baud_table [ ]

  *RTEMS Termios Baud Table.*

## 10.220.1 Detailed Description

RTEMS termios device support internal data structures

## 10.220.2 Macro Definition Documentation

### 10.220.2.1 RTEMS_TERMIOS_DEVICE_CONTEXT_INITIALIZER

```
#define RTEMS_TERMIOS_DEVICE_CONTEXT_INITIALIZER(
              name )
```

**Value:**

```
  { \
    { RTEMS_INTERRUPT_LOCK_INITIALIZER( name ) }, \
    rtems_termios_device_lock_acquire_default, \
    rtems_termios_device_lock_release_default \
  }
```

Initializer for static initialization of Termios device contexts.

**Parameters**

| | |
|---|---|
| *name* | The name for the interrupt lock. It must be a string. The name is only used if profiling is enabled. |

Definition at line 129 of file termiostypes.h.

## 10.220.3 Typedef Documentation

### 10.220.3.1 rtems_termios_device_context

```
typedef struct rtems_termios_device_context rtems_termios_device_context
```

Termios device context.

**See also**

RTEMS_TERMIOS_DEVICE_CONTEXT_INITIALIZER(), rtems_termios_device_context_initialize() and rtems_termios_device_install().

#### 10.220.3.2 rtems_termios_device_node

```
typedef struct rtems_termios_device_node rtems_termios_device_node
```

Termios device node for installed devices.

**See also**

rtems_termios_device_install().

### 10.220.4 Function Documentation

#### 10.220.4.1 rtems_termios_baud_to_number()

```
rtems_termios_baud_t rtems_termios_baud_to_number (
            speed_t baud )
```

Converts the baud flags to an integral baud value.

**Return values**

| | |
|---|---|
| *0* | Invalid baud value or a baud value of B0. |
| *other* | Integral baud value. |

#### 10.220.4.2 rtems_termios_device_context_initialize()

```
static __inline__ void rtems_termios_device_context_initialize (
            rtems_termios_device_context * context,
            const char * name )  [static]
```

Initializes a device context.

**Parameters**

| | | |
|---|---|---|
| in | *context* | The Termios device context. |
| in | *name* | The name for the interrupt lock. This name must be a string persistent throughout the life time of this lock. The name is only used if profiling is enabled. |

Definition at line 113 of file termiostypes.h.

#### 10.220.4.3 rtems_termios_device_install()

```
rtems_status_code rtems_termios_device_install (
```

```
            const char * device_file,
            const rtems_termios_device_handler * handler,
            const rtems_termios_device_flow * flow,
            rtems_termios_device_context * context )
```

Installs a Termios device.

The installed Termios device may be removed via unlink().

**Parameters**

| | | |
|---|---|---|
| in | *device_file* | The device file path. |
| in | *handler* | The device handler. It must be persistent throughout the installed time of the device. |
| in | *flow* | The device flow control handler. The device flow control handler are optional and may be NULL. If present must be persistent throughout the installed time of the device. |
| in | *context* | The device context. It must be persistent throughout the installed time of the device. |

**Return values**

| | |
|---|---|
| *RTEMS_SUCCESSFUL* | Successful operation. |
| *RTEMS_NO_MEMORY* | Not enough memory to create a device node. |
| *RTEMS_UNSATISFIED* | Creation of the device file failed. |
| *RTEMS_INCORRECT_STATE* | Termios is not initialized. |

**See also**

rtems_termios_get_device_context().

#### 10.220.4.4 rtems_termios_device_lock_acquire()

```
static __inline__ void rtems_termios_device_lock_acquire (
            rtems_termios_device_context * context,
            rtems_interrupt_lock_context * lock_context )  [static]
```

Acquires the device lock.

**Parameters**

| | | |
|---|---|---|
| in | *context* | The device context. |
| in | *lock_context* | The local interrupt lock context for an acquire and release pair. |

Definition at line 442 of file termiostypes.h.

#### 10.220.4.5 rtems_termios_device_lock_release()

```
static __inline__ void rtems_termios_device_lock_release (
```

```
            rtems_termios_device_context * context,
            rtems_interrupt_lock_context * lock_context )  [static]
```

Releases the device lock.

**Parameters**

| | | |
|---|---|---|
| in | *context* | The device context. |
| in | *lock_context* | The local interrupt lock context for an acquire and release pair. |

Definition at line 457 of file termiostypes.h.

#### 10.220.4.6 rtems_termios_get_device_context()

```
static __inline__ void* rtems_termios_get_device_context (
            const rtems_termios_tty * tty )  [static]
```

Returns the device context of an installed Termios device.

**Parameters**

| | | |
|---|---|---|
| in | *tty* | The Termios control. |

Definition at line 428 of file termiostypes.h.

#### 10.220.4.7 rtems_termios_kqfilter()

```
int rtems_termios_kqfilter (
            rtems_libio_t * iop,
            struct knote * kn )
```

Termios kqueue() filter filesystem node handler.

Real implementation is provided by libbsd.

#### 10.220.4.8 rtems_termios_mmap()

```
int rtems_termios_mmap (
            rtems_libio_t * iop,
            void ** addr,
            size_t len,
            int prot,
            off_t off )
```

Termios mmap() filter filesystem node handler.

Real implementation is provided by libbsd.

#### 10.220.4.9 rtems_termios_number_to_baud()

```
speed_t rtems_termios_number_to_baud (
            rtems_termios_baud_t baud )
```

Converts the Integral Baud value *baud* to the Termios Control Flag Representation.

**Return values**

| | |
|---|---|
| *B0* | Invalid baud value or a baud value of 0. |
| *other* | Baud constant according to *baud*. |

#### 10.220.4.10 rtems_termios_poll()

```
int rtems_termios_poll (
            rtems_libio_t * iop,
            int events )
```

Termios poll() filesystem node handler.

Real implementation is provided by libbsd.

#### 10.220.4.11 rtems_termios_set_best_baud()

```
void rtems_termios_set_best_baud (
            struct termios * term,
            uint32_t baud )
```

Sets the best baud value in the Termios control.

The valid Termios baud values are between 0 and 460800. The Termios baud value is chosen which minimizes the difference to the value specified.

**Parameters**

| | | |
|---|---|---|
| in | *term* | The Termios attributes. |
| in | *baud* | The current baud setting of the device. |

#### 10.220.4.12 rtems_termios_set_initial_baud()

```
int rtems_termios_set_initial_baud (
            struct rtems_termios_tty * tty,
            rtems_termios_baud_t baud )
```

Sets the initial *baud* in the Termios context *tty*.

**Return values**

| | |
|---|---|
| *0* | Successful operation. |
| *-1* | Invalid baud value. |

## 10.221 cpukit/include/rtems/version.h File Reference

Version API.

```
#include <stdbool.h>
```

### Functions

- const char * rtems_version (void)
  *Returns the version string.*
- int rtems_version_major (void)
  *Returns the version's major number.*
- int rtems_version_minor (void)
  *Returns the version's minor number.*
- int rtems_version_revision (void)
  *Returns the version's revision number.*
- const char * rtems_version_control_key (void)
  *Returns the version control key for the current version of code that has been built.*
- static bool rtems_version_control_key_is_valid (const char *key)
  *Returns true, if the version control key is valid, otherwise false.*
- const char * rtems_board_support_package (void)
  *Returns the board support package name.*

#### 10.221.1 Detailed Description

Version API.

## 10.222 cpukit/include/sys/statvfs.h File Reference

Interface to the statvfs() Set of API Methods.

```
#include <stdint.h>
```

### Classes

- struct statvfs

### Typedefs

- typedef uint64_t **fsblkcnt_t**
- typedef uint32_t **fsfilcnt_t**

### Functions

- int **statvfs** (const char *__restrict, struct statvfs *__restrict)
- int **fstatvfs** (int, struct statvfs *)

#### 10.222.1 Detailed Description

Interface to the statvfs() Set of API Methods.

This include file defines the interface to the statvfs() set of API methods. The statvfs as defined by the SUS:

- http://www.opengroup.org/onlinepubs/009695399/basedefs/sys/statvfs.h.html

## 10.223 cpukit/libcsupport/src/malloc_deferred.c File Reference

Malloc Deferred Support.

#### 10.223.1 Detailed Description

Malloc Deferred Support.

## 10.224 cpukit/libcsupport/src/mallocheap.c File Reference

This source file provides the C Program Heap control along with the system initialization handler.

```
#include <rtems/malloc.h>
#include <rtems/sysinit.h>
#include <rtems/score/wkspacedata.h>
```

### Enumerations

- enum {
  **_Sysinit_bsp_start** = 0x00030080, **_Sysinit_bsp_debug_uart_init** = 0x00030003, **_Sysinit_amba_initialize** = 0x00030001, **_Sysinit_leon3_cpu_index_init** = 0x00030000,
  **_Sysinit_leon3_counter_initialize** = 0x00040000, **_Sysinit_bsp_memory_initialize** = 0x00018080, **_Sysinit__Malloc_Initialize** = 0x00028080, **_Sysinit__Barrier_Manager_initialization** = 0x00140080,
  **_Sysinit__Message_queue_Manager_initialization** = 0x000e0080, **_Sysinit__Partition_Manager_initialization** = 0x00100080, **_Sysinit__Rate_monotonic_Manager_initialization** = 0x00130080, **_Sysinit__Semaphore_Manager_initialization** = 0x000f0080,
  **_Sysinit__RTEMS_tasks_Manager_initialization** = 0x000a0080, **_Sysinit__Timer_Manager_initialization** = 0x000b0080, **_Sysinit_rtems_initialize_data_structures** = 0x00070080, **_Sysinit__Extension_Manager_initialization** = 0x00090080,
  **_Sysinit__Per_CPU_Data_initialize** = 0x00022080, **_Sysinit__Workspace_Handler_initialization** = 0x00026080, **_Sysinit_init_task** = 0x00290080, **_Sysinit_init_runner_task** = 0x00290080 }

### Functions

- void **_Malloc_Initialize** (void)

### Variables

- Heap_Control * RTEMS_Malloc_Heap

  *C program heap control.*
- rtems_sysinit_item const **_Linker_set__Sysinit__Malloc_Initialize** = { _Malloc_Initialize }

## 10.224.1 Detailed Description

This source file provides the C Program Heap control along with the system initialization handler.

## 10.224.2 Variable Documentation

### 10.224.2.1 RTEMS_Malloc_Heap

```
Heap_Control* RTEMS_Malloc_Heap
```

C program heap control.

This is the pointer to the heap control structure used to manage the C program heap.

Definition at line 45 of file mallocheap.c.

# 10.225 cpukit/libcsupport/src/print_printf.c File Reference

RTEMS Print Support.

```
#include <rtems/printer.h>
```

### Functions

- int rtems_vprintf (const rtems_printer *printer, const char *format, va_list ap)

  *Print to the kernel plugin handler. This has to be a macro because there is no vprint version of the plug in handlers.*
- int **rtems_printf** (const rtems_printer *printer, const char *format,...)

## 10.225.1 Detailed Description

RTEMS Print Support.

### 10.225.2 Function Documentation

#### 10.225.2.1 rtems_vprintf()

```
int rtems_vprintf (
            const rtems_printer * printer,
            const char * format,
            va_list ap )
```

Print to the kernel plugin handler. This has to be a macro because there is no vprint version of the plug in handlers.

**Parameters**

| | | |
|---|---|---|
| in | *printer* | Pointer to the printer structure. |
| in | *fmt* | Print format string. |
| in | *ap* | Print variable argument list pointer. |

**Returns**

int Number of characters printed.

Definition at line 23 of file print_printf.c.

## 10.226 cpukit/libcsupport/src/printk.c File Reference

Kernel Printf Function.

```
#include <stdarg.h>
#include <stdio.h>
#include <rtems/bspIo.h>
```

### Functions

- int printk (const char *fmt,...)

### 10.226.1 Detailed Description

Kernel Printf Function.

### 10.226.2 Function Documentation

#### 10.226.2.1 printk()

```
int printk (
             const char * fmt,
              ...  )
```

Kernel printf function requiring minimal infrastructure.

Definition at line 35 of file printk.c.

## 10.227 cpukit/libcsupport/src/printk_plugin.c File Reference

Plugin Printk.

```
#include <rtems/printer.h>
#include <rtems/bspIo.h>
```

### Functions

- int **rtems_printk_printer** (void *ignored, const char *format, va_list ap)
- void rtems_print_printer_printk (rtems_printer *printer)

  *Initializes the printer to print via printk().*

### 10.227.1 Detailed Description

Plugin Printk.

## 10.228 cpukit/libcsupport/src/rtems_putc.c File Reference

RTEMS Output a Character.

```
#include <rtems/bspIo.h>
```

### Functions

- void rtems_putc (char c)

  *Kernel Put Character.*

### 10.228.1 Detailed Description

RTEMS Output a Character.

## 10.228.2 Function Documentation

### 10.228.2.1 rtems_putc()

```
void rtems_putc (
             char c )
```

Kernel Put Character.

This method allows the user to perform a debug putc(). It performs a character translation from "\n" to "\r\n".

**Parameters**

| | | |
|---|---|---|
| in | *c* | is the character to print |

Definition at line 28 of file rtems_putc.c.

## 10.229 cpukit/libcsupport/src/termiosinitialize.c File Reference

Termios Initialization.

```
#include <rtems/termiostypes.h>
```

### Functions

- void **rtems_termios_device_lock_acquire_default** (rtems_termios_device_context *ctx, rtems_interrupt_lock_context *lock_context)
- void **rtems_termios_device_lock_release_default** (rtems_termios_device_context *ctx, rtems_interrupt_lock_context *lock_context)

### 10.229.1 Detailed Description

Termios Initialization.

## 10.230 cpukit/libcsupport/src/vprintk.c File Reference

Print Formatted Output.

```
#include <rtems/bspIo.h>
#include <rtems/score/io.h>
```

### Functions

- int vprintk (const char *fmt, va_list ap)
  *Variable Argument printk()*

### 10.230.1 Detailed Description

Print Formatted Output.

### 10.230.2 Function Documentation

#### 10.230.2.1 vprintk()

```
int vprintk (
            const char * fmt,
            va_list ap )
```

Variable Argument printk()

This method allows the user to access printk() functionality with a va_list style argument.

**Parameters**

| | | |
|---|---|---|
| in | *fmt* | is a printf()-style format string |
| in | *ap* | is a va_list pointer to arguments |

**Returns**

The number of characters output.

Definition at line 29 of file vprintk.c.

## 10.231 cpukit/libtest/t-test-busy-tick.c File Reference

Implementation of T_get_one_clock_tick_busy().

```
#include <rtems/test.h>
#include <rtems.h>
```

### Functions

- static uint_fast32_t **T_estimate_busy_loop_maximum** (void)
- static uint_fast32_t **T_wait_for_tick_change** (void)
- uint_fast32_t **T_get_one_clock_tick_busy** (void)

### 10.231.1 Detailed Description

Implementation of T_get_one_clock_tick_busy().

## 10.232 cpukit/libtest/t-test-busy.c File Reference

Implementation of T_busy().

```
#include <rtems/test.h>
```

### Functions

- void **T_busy** (uint_fast32_t count)

### 10.232.1 Detailed Description

Implementation of T_busy().

## 10.233 cpukit/libtest/t-test-interrupt.c File Reference

Implementation of T_interrupt_test().

```
#include <rtems/test.h>
#include <rtems/score/atomic.h>
#include <rtems/score/percpu.h>
#include <rtems/score/thread.h>
#include <rtems/score/timecounter.h>
#include <rtems/score/timestampimpl.h>
#include <rtems/score/userextimpl.h>
#include <rtems/score/watchdogimpl.h>
#include <rtems/score/smpimpl.h>
```

### Classes

- struct T_interrupt_context
- struct T_interrupt_clock_time

### Macros

- #define **T_INTERRUPT_SAMPLE_COUNT** 8

### Typedefs

- typedef T_interrupt_test_state(∗ **T_interrupt_test_handler**) (void ∗)

### Functions

- static void **T_interrupt_sort** (T_interrupt_clock_time ∗ct, size_t n)
- static int64_t **T_interrupt_time_close_to_tick** (void)
- static void **T_interrupt_watchdog** (Watchdog_Control ∗wdg)
- static void **T_interrupt_watchdog_insert** (T_interrupt_context ∗ctx)
- static void **T_interrupt_watchdog_remove** (T_interrupt_context ∗ctx)
- static void **T_interrupt_init_once** (T_interrupt_context ∗ctx)
- static T_interrupt_test_state **T_interrupt_continue** (void ∗arg)
- static void **T_interrupt_do_nothing** (void ∗arg)
- static void **T_interrupt_blocked** (void ∗arg)
- static void **T_interrupt_thread_switch** (Thread_Control ∗, Thread_Control ∗)
- T_interrupt_test_state **T_interrupt_test_change_state** (T_interrupt_test_state expected_state, T_↩ interrupt_test_state desired_state)
- T_interrupt_test_state **T_interrupt_test_get_state** (void)
- void **T_interrupt_test_busy_wait_for_interrupt** (void)
- static T_interrupt_context ∗ **T_interrupt_setup** (const T_interrupt_test_config ∗config, void ∗arg)
- static void **T_interrupt_teardown** (void ∗arg)
- T_interrupt_test_state **T_interrupt_test** (const T_interrupt_test_config ∗config, void ∗arg)

## Variables

- static T_interrupt_context **T_interrupt_instance**
- static const T_fixture **T_interrupt_fixture**

### 10.233.1 Detailed Description

Implementation of T_interrupt_test().

### 10.233.2 Variable Documentation

#### 10.233.2.1 T_interrupt_fixture

```
const T_fixture T_interrupt_fixture  [static]
```

**Initial value:**

```
= {
    .teardown = T_interrupt_teardown,
    .initial_context = &T_interrupt_instance
}
```

Definition at line 355 of file t-test-interrupt.c.

#### 10.233.2.2 T_interrupt_instance

```
T_interrupt_context T_interrupt_instance  [static]
```

**Initial value:**

```
= {
    .interrupt = T_interrupt_continue,
    .blocked = T_interrupt_do_nothing,
    .job = {
        .context = &T_interrupt_instance.job_context
    },
    .job_context = {
        .handler = T_interrupt_blocked,
        .arg = &T_interrupt_instance
    },
    .wdg = WATCHDOG_INITIALIZER(T_interrupt_watchdog),
    .ext = {
        .Callouts = {
            .thread_switch = T_interrupt_thread_switch
        }
    }
}
```

Definition at line 223 of file t-test-interrupt.c.

## 10.234 cpukit/libtest/t-test-rtems-objs.c File Reference

RTEMS Objects Support for Test Framework.

```
#include "t-test-rtems.h"
#include <rtems/test.h>
#include <inttypes.h>
#include <rtems/score/threadimpl.h>
```

### Functions

- Objects_Maximum **T_objects_count** (Objects_APIs api, uint16_t cls)
- void **T_objects_check** (Objects_APIs api, uint16_t cls, Objects_Maximum *expected, const char *name)
- static void **T_rtems_barriers_run_initialize** (void)
- static void **T_rtems_barriers_case_end** (void)
- void **T_check_rtems_barriers** (T_event event, const char *name)
- static void **T_rtems_extensions_run_initialize** (void)
- static void **T_rtems_extensions_case_end** (void)
- void **T_check_rtems_extensions** (T_event event, const char *name)
- static void **T_rtems_message_queues_run_initialize** (void)
- static void **T_rtems_message_queues_case_end** (void)
- void **T_check_rtems_message_queues** (T_event event, const char *name)
- static void **T_rtems_partitions_run_initialize** (void)
- static void **T_rtems_partitions_case_end** (void)
- void **T_check_rtems_partitions** (T_event event, const char *name)
- static void **T_rtems_periods_run_initialize** (void)
- static void **T_rtems_periods_case_end** (void)
- void **T_check_rtems_periods** (T_event event, const char *name)
- static void **T_rtems_regions_run_initialize** (void)
- static void **T_rtems_regions_case_end** (void)
- void **T_check_rtems_regions** (T_event event, const char *name)
- static void **T_rtems_semaphores_run_initialize** (void)
- static void **T_rtems_semaphores_case_end** (void)
- void **T_check_rtems_semaphores** (T_event event, const char *name)
- static void **T_rtems_tasks_run_initialize** (void)
- static void **T_rtems_tasks_case_end** (void)
- void **T_check_rtems_tasks** (T_event event, const char *name)
- static void **T_rtems_timers_run_initialize** (void)
- static void **T_rtems_timers_case_end** (void)
- void **T_check_rtems_timers** (T_event event, const char *name)

### Variables

- static Objects_Maximum **T_barrier_count**
- static Objects_Maximum **T_extension_count**
- static Objects_Maximum **T_mq_count**
- static Objects_Maximum **T_part_count**
- static Objects_Maximum **T_period_count**
- static Objects_Maximum **T_region_count**
- static Objects_Maximum **T_sema_count**
- static Objects_Maximum **T_task_count**
- static Objects_Maximum **T_timer_count**

### 10.234.1 Detailed Description

RTEMS Objects Support for Test Framework.

## 10.235 cpukit/libtest/t-test-rtems.h File Reference

RTEMS Support for Test Framework.

```
#include <rtems/score/objectimpl.h>
```

### Functions

- Objects_Maximum **T_objects_count** (Objects_APIs api, uint16_t cls)
- void **T_objects_check** (Objects_APIs api, uint16_t cls, Objects_Maximum *expected, const char *name)

### 10.235.1 Detailed Description

RTEMS Support for Test Framework.

## 10.236 cpukit/libtest/t-test-thread-switch.c File Reference

Implementation of T_thread_switch_record().

```
#include <rtems/test.h>
#include <rtems.h>
#include <rtems/score/percpu.h>
#include <rtems/score/smp.h>
#include <rtems/score/thread.h>
#include <rtems/score/userextimpl.h>
```

### Classes

- struct T_thread_switch_context

### Functions

- static void **T_thread_switch_recorder** (Thread_Control *, Thread_Control *)
- static void **T_thread_switch_destroy** (T_destructor *dtor)
- T_thread_switch_log * **T_thread_switch_record** (T_thread_switch_log *log)
- T_thread_switch_log * **T_thread_switch_record_2** (T_thread_switch_log_2 *log)
- T_thread_switch_log * **T_thread_switch_record_4** (T_thread_switch_log_4 *log)
- T_thread_switch_log * **T_thread_switch_record_10** (T_thread_switch_log_10 *log)

### Variables

- static T_thread_switch_context **T_thread_switch_instance**

### 10.236.1 Detailed Description

Implementation of T_thread_switch_record().

### 10.236.2 Variable Documentation

#### 10.236.2.1 T_thread_switch_instance

```
T_thread_switch_context T_thread_switch_instance  [static]
```

**Initial value:**

```
= {
    .lock = RTEMS_INTERRUPT_LOCK_INITIALIZER("Test Thread Switches"),
    .ext = {
        .Callouts = {
            .thread_switch = T_thread_switch_recorder
        }
    }
}
```

Definition at line 57 of file t-test-thread-switch.c.

## 10.237 cpukit/libtest/testrun.c File Reference

Implementation of rtems_test_run_default().

```
#include <rtems/test-info.h>
#include <rtems/test.h>
```

### Functions

- void rtems_test_run (rtems_task_argument arg, const RTEMS_TEST_STATE state)

  *Runs the test cases of the RTEMS Test Framework using a default configuration in the context of a task.*

### Variables

- static char **buffer** [512]
- static const T_action **actions** [ ]
- static const T_config **config**

### 10.237.1 Detailed Description

Implementation of rtems_test_run_default().

### 10.237.2 Variable Documentation

#### 10.237.2.1 actions

```
const T_action actions[]  [static]
```

**Initial value:**

```
= {
  T_report_hash_sha256,
  T_check_task_context,
  T_check_file_descriptors,
  T_check_rtems_barriers,
  T_check_rtems_extensions,
  T_check_rtems_message_queues,
  T_check_rtems_partitions,
  T_check_rtems_periods,
  T_check_rtems_regions,
  T_check_rtems_semaphores,
  T_check_rtems_tasks,
  T_check_rtems_timers,
  T_check_posix_keys
}
```

Definition at line 45 of file testrun.c.

#### 10.237.2.2 config

```
const T_config config  [static]
```

**Initial value:**

```
= {
  .name = rtems_test_name,
  .buf = buffer,
  .buf_size = sizeof( buffer ),
  .putchar = T_putchar_default,
  .verbosity = T_VERBOSE,
  .now = T_now_clock,
  .action_count = T_ARRAY_SIZE( actions ),
  .actions = actions
}
```

Definition at line 61 of file testrun.c.

## 10.238 cpukit/posix/src/pspindestroy.c File Reference

Destroy a Spinlock.

```
#include <rtems/posix/spinlockimpl.h>
```

### Functions

- int **pthread_spin_destroy** (pthread_spinlock_t *spinlock)

#### 10.238.1 Detailed Description

Destroy a Spinlock.

## 10.239 cpukit/posix/src/pspininit.c File Reference

POSIX Function Initializes a Spinlock Instance.

```
#include <rtems/posix/spinlockimpl.h>
```

### Functions

- int **pthread_spin_init** (pthread_spinlock_t *spinlock, int pshared)

#### 10.239.1 Detailed Description

POSIX Function Initializes a Spinlock Instance.

## 10.240 cpukit/posix/src/pspinlock.c File Reference

Wait at a Spinlock.

```
#include <rtems/posix/spinlockimpl.h>
```

### Functions

- **RTEMS_STATIC_ASSERT** (offsetof(POSIX_Spinlock_Control, Lock.next_ticket)==offsetof(pthread_↩ spinlock_t, _Lock._next_ticket), POSIX_SPINLOCK_T_LOCK_NEXT_TICKET)
- **RTEMS_STATIC_ASSERT** (offsetof(POSIX_Spinlock_Control, Lock.now_serving)==offsetof(pthread_↩ spinlock_t, _Lock._now_serving), POSIX_SPINLOCK_T_LOCK_NOW_SERVING)
- **RTEMS_STATIC_ASSERT** (offsetof(POSIX_Spinlock_Control, interrupt_state)==offsetof(pthread_↩ spinlock_t, _interrupt_state), POSIX_SPINLOCK_T_INTERRUPT_STATE)
- **RTEMS_STATIC_ASSERT** (sizeof(POSIX_Spinlock_Control)==sizeof(pthread_spinlock_t), POSIX_SPIN↩ LOCK_T_SIZE)
- int **pthread_spin_lock** (pthread_spinlock_t *spinlock)
- int **pthread_spin_trylock** (pthread_spinlock_t *spinlock) RTEMS_ALIAS(pthread_spin_lock)

### 10.240.1 Detailed Description

Wait at a Spinlock.

## 10.241 cpukit/posix/src/pspinunlock.c File Reference

Function Unlocks a Spin Lock Object.

```
#include <rtems/posix/spinlockimpl.h>
```

### Functions

- int **pthread_spin_unlock** (pthread_spinlock_t *lock)

### 10.241.1 Detailed Description

Function Unlocks a Spin Lock Object.

## 10.242 cpukit/rtems/src/barrier.c File Reference

Classic Barrier Information with Zero Objects.

```
#include <rtems/rtems/barrierdata.h>
```

### Variables

- Objects_Information _Barrier_Information = { ( (Objects_Id) ( (Objects_Id) OBJECTS_CLASSIC_API << 24U ) | ( (Objects_Id) OBJECTS_RTEMS_BARRIERS << 27U ) | ( (Objects_Id) 1 << 16U ) | ( (Objects_Id) 0 << 0U ) ), NULL, _Objects_Allocate_none, NULL, 0, 0, 0, 0 , { { { &( _Barrier_Information.Inactive ).Tail.↩ Node, NULL }, &( _Barrier_Information.Inactive ).Head.Node } }, NULL, NULL, NULL }

  *The Classic Barrier objects information.*

### 10.242.1 Detailed Description

Classic Barrier Information with Zero Objects.

## 10.243 cpukit/rtems/src/barriercreate.c File Reference

RTEMS Create Barrier.

```
#include <rtems/rtems/barrierimpl.h>
#include <rtems/rtems/status.h>
#include <rtems/rtems/support.h>
#include <rtems/rtems/attrimpl.h>
#include <rtems/score/isr.h>
#include <rtems/sysinit.h>
```

### Enumerations

- enum {
  **_Sysinit_bsp_start** = 0x00030080, **_Sysinit_bsp_debug_uart_init** = 0x00030003, **_Sysinit_amba_↩ initialize** = 0x00030001, **_Sysinit_leon3_cpu_index_init** = 0x00030000,
  **_Sysinit_leon3_counter_initialize** = 0x00040000, **_Sysinit_bsp_memory_initialize** = 0x00018080, **_↩ Sysinit__Malloc_Initialize** = 0x00028080, **_Sysinit__Barrier_Manager_initialization** = 0x00140080,
  **_Sysinit__Message_queue_Manager_initialization** = 0x000e0080, **_Sysinit__Partition_Manager_↩ initialization** = 0x00100080, **_Sysinit__Rate_monotonic_Manager_initialization** = 0x00130080, **_↩ Sysinit__Semaphore_Manager_initialization** = 0x000f0080,
  **_Sysinit__RTEMS_tasks_Manager_initialization** = 0x000a0080, **_Sysinit__Timer_Manager_↩ initialization** = 0x000b0080, **_Sysinit_rtems_initialize_data_structures** = 0x00070080, **_Sysinit__↩ Extension_Manager_initialization** = 0x00090080,
  **_Sysinit__Per_CPU_Data_initialize** = 0x00022080, **_Sysinit__Workspace_Handler_initialization** = 0x00026080, **_Sysinit_init_task** = 0x00290080, **_Sysinit_init_runner_task** = 0x00290080 }

### Functions

- rtems_status_code rtems_barrier_create (rtems_name name, rtems_attribute attribute_set, uint32_↩ t maximum_waiters, rtems_id *id)
  %
- static void **_Barrier_Manager_initialization** (void)

### Variables

- rtems_sysinit_item const **_Linker_set__Sysinit__Barrier_Manager_initialization** = { _Barrier_Manager↩ _initialization }

### 10.243.1 Detailed Description

RTEMS Create Barrier.

## 10.244 cpukit/rtems/src/barrierdelete.c File Reference

RTEMS Delete Barrier.

```
#include <rtems/rtems/barrierimpl.h>
```

### Functions

- rtems_status_code rtems_barrier_delete (rtems_id id)

  %

### 10.244.1 Detailed Description

RTEMS Delete Barrier.

## 10.245 cpukit/rtems/src/barrierident.c File Reference

rtems_barrier_ident() Implementation

```
#include <rtems/rtems/barrierimpl.h>
#include <rtems/rtems/objectimpl.h>
```

### Functions

- rtems_status_code rtems_barrier_ident (rtems_name name, rtems_id *id)

  *Identifies a barrier object by the specified object name.*

### 10.245.1 Detailed Description

rtems_barrier_ident() Implementation

## 10.246 cpukit/rtems/src/barrierrelease.c File Reference

RTEMS Barrier Release.

```
#include <rtems/rtems/barrierimpl.h>
```

### Functions

- rtems_status_code rtems_barrier_release (rtems_id id, uint32_t *released)

  %

### 10.246.1 Detailed Description

RTEMS Barrier Release.

## 10.247 cpukit/rtems/src/barrierwait.c File Reference

RTEMS Barrier Wait.

```
#include <rtems/rtems/barrierimpl.h>
#include <rtems/rtems/statusimpl.h>
```

### Functions

- **THREAD_QUEUE_OBJECT_ASSERT** (Barrier_Control, Barrier.Wait_queue, BARRIER_CONTROL)
- rtems_status_code rtems_barrier_wait (rtems_id id, rtems_interval timeout)

  %

### 10.247.1 Detailed Description

RTEMS Barrier Wait.

## 10.248 cpukit/rtems/src/clockgetuptime.c File Reference

Obtain the System Uptime.

```
#include <rtems/rtems/clock.h>
#include <rtems/score/todimpl.h>
```

### Functions

- rtems_status_code rtems_clock_get_uptime (struct timespec *uptime)

  %

### 10.248.1 Detailed Description

Obtain the System Uptime.

## 10.249 cpukit/rtems/src/clocktodtoseconds.c File Reference

TOD to Seconds.

```
#include <rtems/rtems/clock.h>
#include <rtems/score/todimpl.h>
```

## Macros

- #define **TOD_SECONDS_AT_2100_03_01_00_00** 4107538800UL

## Functions

- Watchdog_Interval _TOD_To_seconds (const rtems_time_of_day *the_tod)

  %

## Variables

- const uint16_t **_TOD_Days_to_date** [2][13]
- const uint16_t **_TOD_Days_since_last_leap_year** [4] = { 0, 366, 731, 1096 }

### 10.249.1 Detailed Description

TOD to Seconds.

### 10.249.2 Function Documentation

#### 10.249.2.1 _TOD_To_seconds()

```
Watchdog_Interval _TOD_To_seconds (
            const rtems_time_of_day * the_tod )
```

%

**Parameters**

| | |
|---|---|
| *the_tod* | % |

Definition at line 45 of file clocktodtoseconds.c.

### 10.249.3 Variable Documentation

#### 10.249.3.1 _TOD_Days_to_date

```
const uint16_t _TOD_Days_to_date[2][13]
```

**Initial value:**

```
= {
  { 0, 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334 },
  { 0, 0, 31, 60, 91, 121, 152, 182, 213, 244, 274, 305, 335 }
}
```

Definition at line 31 of file clocktodtoseconds.c.

## 10.250 cpukit/rtems/src/clocktodvalidate.c File Reference

TOD Validate.

```
#include <rtems/rtems/clock.h>
#include <rtems/score/todimpl.h>
#include <rtems/config.h>
```

### Functions

- bool _TOD_Validate (const rtems_time_of_day *the_tod)
  %

### Variables

- const uint32_t **_TOD_Days_per_month** [2][13]

### 10.250.1 Detailed Description

TOD Validate.

### 10.250.2 Function Documentation

#### 10.250.2.1 _TOD_Validate()

```
bool _TOD_Validate (
            const rtems_time_of_day * the_tod )
```

%

**Parameters**

| | |
|---|---|
| *the_tod* | % |

Definition at line 36 of file clocktodvalidate.c.

### 10.250.3 Variable Documentation

#### 10.250.3.1 _TOD_Days_per_month

```
const uint32_t _TOD_Days_per_month[2][13]
```

**Initial value:**

```
= {
  { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 },
  { 0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 }
}
```

Definition at line 31 of file clocktodvalidate.c.

## 10.251 cpukit/rtems/src/eventreceive.c File Reference

This source file contains the implementation of rtems_event_receive().

```
#include <rtems/rtems/eventimpl.h>
#include <rtems/rtems/tasksdata.h>
#include <rtems/score/statesimpl.h>
#include <rtems/score/threadimpl.h>
```

### Functions

- rtems_status_code rtems_event_receive (rtems_event_set event_in, rtems_option option_set, rtems_interval ticks, rtems_event_set *event_out)

  *Receives or gets an event set from the calling task.*

### 10.251.1 Detailed Description

This source file contains the implementation of rtems_event_receive().

## 10.252 cpukit/rtems/src/eventseize.c File Reference

This source file contains the implementation of _Event_Seize() and the task event MPCI support system initialization.

```
#include <rtems/sysinit.h>
#include <rtems/rtems/eventimpl.h>
#include <rtems/rtems/optionsimpl.h>
#include <rtems/rtems/statusimpl.h>
#include <rtems/score/threadimpl.h>
#include <rtems/score/watchdogimpl.h>
```

### Functions

- rtems_status_code _Event_Seize (rtems_event_set event_in, rtems_option option_set, rtems_interval ticks, rtems_event_set *event_out, Thread_Control *executing, Event_Control *event, Thread_Wait_flags wait_class, States_Control block_state, ISR_lock_Context *lock_context)

  *Seizes a set of events.*

#### 10.252.1 Detailed Description

This source file contains the implementation of _Event_Seize() and the task event MPCI support system initialization.

## 10.253 cpukit/rtems/src/eventsend.c File Reference

This source file contains the implementation of rtems_event_send().

```
#include <rtems/rtems/eventimpl.h>
#include <rtems/rtems/tasksdata.h>
#include <rtems/score/threadimpl.h>
```

### Functions

- rtems_status_code rtems_event_send (rtems_id id, rtems_event_set event_in)

  *Sends the event set to the task.*

#### 10.253.1 Detailed Description

This source file contains the implementation of rtems_event_send().

## 10.254 cpukit/rtems/src/eventsurrender.c File Reference

This source file contains the implementation of _Event_Surrender().

```
#include <rtems/rtems/eventimpl.h>
#include <rtems/rtems/optionsimpl.h>
#include <rtems/score/threadimpl.h>
#include <rtems/score/watchdogimpl.h>
```

### Functions

- static void **_Event_Satisfy** (Thread_Control *the_thread, Event_Control *event, rtems_event_set pending_events, rtems_event_set seized_events)
- static bool **_Event_Is_blocking_on_event** (const Thread_Control *the_thread, Thread_Wait_flags wait_class)
- static bool **_Event_Is_satisfied** (const Thread_Control *the_thread, rtems_event_set pending_events, rtems_event_set *seized_events)
- rtems_status_code _Event_Surrender (Thread_Control *the_thread, rtems_event_set event_in, Event_Control *event, Thread_Wait_flags wait_class, ISR_lock_Context *lock_context)

  *Surrenders a set of events.*

### 10.254.1 Detailed Description

This source file contains the implementation of _Event_Surrender().

## 10.255 cpukit/rtems/src/intrcatch.c File Reference

RTEMS Interrupt Catch.

```
#include <rtems/rtems/intr.h>
#include <rtems/score/isr.h>
```

### 10.255.1 Detailed Description

RTEMS Interrupt Catch.

## 10.256 cpukit/rtems/src/msg.c File Reference

Classic Message Queue Information with Zero Objects.

```
#include <rtems/rtems/messagedata.h>
```

### Variables

- Objects_Information _Message_queue_Information = { ( (Objects_Id) ( (Objects_Id) OBJECTS_CLASSIC↩ _API << 24U ) | ( (Objects_Id) OBJECTS_RTEMS_MESSAGE_QUEUES << 27U ) | ( (Objects_Id) 1 << 16U ) | ( (Objects_Id) 0 << 0U ) ), NULL, _Objects_Allocate_none, NULL, 0, 0, 0, 0 , { { { &( _Message↩ _queue_Information.Inactive ).Tail.Node, NULL }, &( _Message_queue_Information.Inactive ).Head.Node } }, NULL, NULL, NULL }

  *The Classic Message Queue objects information.*

### 10.256.1 Detailed Description

Classic Message Queue Information with Zero Objects.

## 10.257 cpukit/rtems/src/msgqbroadcast.c File Reference

RTEMS Broadcast Message Queue.

```
#include <rtems/rtems/messageimpl.h>
#include <rtems/rtems/statusimpl.h>
```

### Functions

- rtems_status_code rtems_message_queue_broadcast (rtems_id id, const void *buffer, size_t size, uint32_t *count)

  %

#### 10.257.1 Detailed Description

RTEMS Broadcast Message Queue.

## 10.258 cpukit/rtems/src/msgqconstruct.c File Reference

RTEMS Create Message Queue.

```
#include <rtems/rtems/messageimpl.h>
#include <rtems/rtems/attrimpl.h>
#include <rtems/rtems/support.h>
#include <rtems/score/coremsgimpl.h>
#include <rtems/sysinit.h>
```

### Enumerations

- enum {
  **_Sysinit_bsp_start** = 0x00030080, **_Sysinit_bsp_debug_uart_init** = 0x00030003, **_Sysinit_amba_initialize** = 0x00030001, **_Sysinit_leon3_cpu_index_init** = 0x00030000,
  **_Sysinit_leon3_counter_initialize** = 0x00040000, **_Sysinit_bsp_memory_initialize** = 0x00018080, **_Sysinit__Malloc_Initialize** = 0x00028080, **_Sysinit__Barrier_Manager_initialization** = 0x00140080,
  **_Sysinit__Message_queue_Manager_initialization** = 0x000e0080, **_Sysinit__Partition_Manager_initialization** = 0x00100080, **_Sysinit__Rate_monotonic_Manager_initialization** = 0x00130080, **_Sysinit__Semaphore_Manager_initialization** = 0x000f0080,
  **_Sysinit__RTEMS_tasks_Manager_initialization** = 0x000a0080, **_Sysinit__Timer_Manager_initialization** = 0x000b0080, **_Sysinit_rtems_initialize_data_structures** = 0x00070080, **_Sysinit__Extension_Manager_initialization** = 0x00090080,
  **_Sysinit__Per_CPU_Data_initialize** = 0x00022080, **_Sysinit__Workspace_Handler_initialization** = 0x00026080, **_Sysinit_init_task** = 0x00290080, **_Sysinit_init_runner_task** = 0x00290080 }

### Functions

- static void * **_Message_queue_Get_buffers** (CORE_message_queue_Control *the_message_queue, size_t size, const void *arg)
- rtems_status_code rtems_message_queue_construct (const rtems_message_queue_config *config, rtems_id *id)

  *Constructs a message queue from the specified the message queue configuration.*
- rtems_status_code _Message_queue_Create (const rtems_message_queue_config *config, rtems_id *id, CORE_message_queue_Allocate_buffers allocate_buffers)

  *Creates a message queue.*
- static void **_Message_queue_Manager_initialization** (void)

### Variables

- rtems_sysinit_item const **_Linker_set__Sysinit__Message_queue_Manager_initialization** = { _↩ Message_queue_Manager_initialization }

### 10.258.1 Detailed Description

RTEMS Create Message Queue.

## 10.259 cpukit/rtems/src/msgqcreate.c File Reference

This source file contains the implementation of rtems_message_queue_create().

```
#include <rtems/rtems/messageimpl.h>
#include <rtems/score/coremsgimpl.h>
#include <string.h>
```

### Functions

- rtems_status_code rtems_message_queue_create (rtems_name name, uint32_t count, size_t max_↩ message_size, rtems_attribute attribute_set, rtems_id *id)

  %

### 10.259.1 Detailed Description

This source file contains the implementation of rtems_message_queue_create().

## 10.260 cpukit/rtems/src/msgqdelete.c File Reference

RTEMS Delete Message Queue.

```
#include <rtems/rtems/messageimpl.h>
#include <rtems/rtems/attrimpl.h>
```

### Functions

- rtems_status_code rtems_message_queue_delete (rtems_id id)

  %

### 10.260.1 Detailed Description

RTEMS Delete Message Queue.

## 10.261 cpukit/rtems/src/msgqflush.c File Reference

rtems_message_queue_flush

```
#include <rtems/rtems/messageimpl.h>
```

### Functions

- rtems_status_code rtems_message_queue_flush (rtems_id id, uint32_t *count)
  %

### 10.261.1 Detailed Description

rtems_message_queue_flush

## 10.262 cpukit/rtems/src/msgqgetnumberpending.c File Reference

RTEMS Message Queue Get Number Pending.

```
#include <rtems/rtems/messageimpl.h>
```

### Functions

- rtems_status_code rtems_message_queue_get_number_pending (rtems_id id, uint32_t *count)
  %

### 10.262.1 Detailed Description

RTEMS Message Queue Get Number Pending.

## 10.263 cpukit/rtems/src/msgqident.c File Reference

rtems_message_queue_ident() Implementation

```
#include <rtems/rtems/messageimpl.h>
#include <rtems/rtems/objectimpl.h>
```

### Functions

- rtems_status_code rtems_message_queue_ident (rtems_name name, uint32_t node, rtems_id *id)
  *Identifies a message queue object by the specified object name.*

### 10.263.1 Detailed Description

rtems_message_queue_ident() Implementation

## 10.264 cpukit/rtems/src/msgqreceive.c File Reference

RTEMS Message Queue Receive.

```
#include <rtems/rtems/messageimpl.h>
#include <rtems/rtems/optionsimpl.h>
#include <rtems/rtems/statusimpl.h>
```

### Functions

- **THREAD_QUEUE_OBJECT_ASSERT** (Message_queue_Control, message_queue.Wait_queue, MESS↩ AGE_QUEUE_CONTROL)
- rtems_status_code rtems_message_queue_receive (rtems_id id, void *buffer, size_t *size, rtems_option option_set, rtems_interval timeout)

  %

### 10.264.1 Detailed Description

RTEMS Message Queue Receive.

## 10.265 cpukit/rtems/src/msgqsend.c File Reference

rtems_message_queue_send

```
#include <rtems/rtems/messageimpl.h>
#include <rtems/rtems/statusimpl.h>
```

### Functions

- rtems_status_code rtems_message_queue_send (rtems_id id, const void *buffer, size_t size)

  %

### 10.265.1 Detailed Description

rtems_message_queue_send

## 10.266 cpukit/rtems/src/msgqurgent.c File Reference

RTEMS Urgent Message Queue.

```
#include <rtems/rtems/messageimpl.h>
#include <rtems/rtems/statusimpl.h>
```

### Functions

- rtems_status_code rtems_message_queue_urgent (rtems_id id, const void *buffer, size_t size)

  %

### 10.266.1 Detailed Description

RTEMS Urgent Message Queue.

## 10.267 cpukit/rtems/src/part.c File Reference

This source file contains the default _Partition_Information with zero objects.

```
#include <rtems/rtems/partdata.h>
```

### Variables

- Objects_Information _Partition_Information = { ( (Objects_Id) ( (Objects_Id) OBJECTS_CLASSIC_API << 24U ) | ( (Objects_Id) OBJECTS_RTEMS_PARTITIONS << 27U ) | ( (Objects_Id) 1 << 16U ) | ( (Objects_Id) 0 << 0U ) ), NULL, _Objects_Allocate_none, NULL, 0, 0, 0, 0 , { { { &( _Partition_Information.↩ Inactive ).Tail.Node, NULL }, &( _Partition_Information.Inactive ).Head.Node } }, NULL, NULL, NULL }

  *The Partition Manager objects information is used to manage the objects of this class.*

### 10.267.1 Detailed Description

This source file contains the default _Partition_Information with zero objects.

## 10.268 cpukit/rtems/src/partcreate.c File Reference

This source file contains the implementation of rtems_partition_create().

```
#include <rtems/rtems/partimpl.h>
#include <rtems/rtems/attrimpl.h>
#include <rtems/rtems/support.h>
#include <rtems/score/chainimpl.h>
#include <rtems/score/sysstate.h>
#include <rtems/sysinit.h>
```

### Enumerations

- enum {
  **_Sysinit_bsp_start** = 0x00030080, **_Sysinit_bsp_debug_uart_init** = 0x00030003, **_Sysinit_amba_↩ initialize** = 0x00030001, **_Sysinit_leon3_cpu_index_init** = 0x00030000,
  **_Sysinit_leon3_counter_initialize** = 0x00040000, **_Sysinit_bsp_memory_initialize** = 0x00018080, _↩ **Sysinit__Malloc_Initialize** = 0x00028080, **_Sysinit__Barrier_Manager_initialization** = 0x00140080,
  **_Sysinit__Message_queue_Manager_initialization** = 0x000e0080, **_Sysinit__Partition_Manager_↩ initialization** = 0x00100080, **_Sysinit__Rate_monotonic_Manager_initialization** = 0x00130080, _↩ **Sysinit__Semaphore_Manager_initialization** = 0x000f0080,
  **_Sysinit__RTEMS_tasks_Manager_initialization** = 0x000a0080, **_Sysinit__Timer_Manager_↩ initialization** = 0x000b0080, **_Sysinit_rtems_initialize_data_structures** = 0x00070080, **_Sysinit__↩ Extension_Manager_initialization** = 0x00090080,
  **_Sysinit__Per_CPU_Data_initialize** = 0x00022080, **_Sysinit__Workspace_Handler_initialization** = 0x00026080, **_Sysinit_init_task** = 0x00290080, **_Sysinit_init_runner_task** = 0x00290080 }

### Functions

- static bool **_Partition_Is_buffer_size_aligned** (size_t buffer_size)
- static bool **_Partition_Is_buffer_area_aligned** (const void *starting_address)
- static Partition_Control * **_Partition_Allocate** (void)
- static void **_Partition_Initialize** (Partition_Control *the_partition, void *starting_address, uintptr_t length, size_t buffer_size, rtems_attribute attribute_set)
- rtems_status_code rtems_partition_create (rtems_name name, void *starting_address, uintptr_t length, size_t buffer_size, rtems_attribute attribute_set, rtems_id *id)

  *Creates a partition.*
- static void **_Partition_Manager_initialization** (void)

### Variables

- rtems_sysinit_item const **_Linker_set__Sysinit__Partition_Manager_initialization** = { _Partition_↩ Manager_initialization }

### 10.268.1 Detailed Description

This source file contains the implementation of rtems_partition_create().

## 10.269 cpukit/rtems/src/partdelete.c File Reference

This source file contains the implementation of rtems_partition_delete().

```
#include <rtems/rtems/partimpl.h>
#include <rtems/rtems/attrimpl.h>
```

### Functions

- rtems_status_code rtems_partition_delete (rtems_id id)

  *Deletes the partition.*

### 10.269.1 Detailed Description

This source file contains the implementation of rtems_partition_delete().

## 10.270 cpukit/rtems/src/partgetbuffer.c File Reference

This source file contains the implementation of rtems_partition_get_buffer().

```
#include <rtems/rtems/partimpl.h>
#include <rtems/score/chainimpl.h>
```

### Functions

- static void * **_Partition_Allocate_buffer** (Partition_Control *the_partition)
- rtems_status_code rtems_partition_get_buffer (rtems_id id, void **buffer)

  *Tries to get a buffer from the partition.*

### 10.270.1 Detailed Description

This source file contains the implementation of rtems_partition_get_buffer().

## 10.271 cpukit/rtems/src/partident.c File Reference

This source file contains the implementation of rtems_partition_ident().

```
#include <rtems/rtems/partimpl.h>
#include <rtems/rtems/objectimpl.h>
```

### Functions

- rtems_status_code rtems_partition_ident (rtems_name name, uint32_t node, rtems_id *id)

  *Identifies a partition by the object name.*

### 10.271.1 Detailed Description

This source file contains the implementation of rtems_partition_ident().

# 10.272 cpukit/rtems/src/partreturnbuffer.c File Reference

This source file contains the implementation of rtems_partition_return_buffer().

```
#include <rtems/rtems/partimpl.h>
#include <rtems/score/address.h>
#include <rtems/score/chainimpl.h>
```

## Functions

- static bool **_Partition_Is_buffer_on_boundary** (const Partition_Control *the_partition, const void *the_buffer)
- static bool **_Partition_Is_buffer_valid** (const Partition_Control *the_partition, const void *the_buffer)
- static void **_Partition_Free_buffer** (Partition_Control *the_partition, void *the_buffer)
- rtems_status_code rtems_partition_return_buffer (rtems_id id, void *buffer)

  *Returns the buffer to the partition.*

### 10.272.1 Detailed Description

This source file contains the implementation of rtems_partition_return_buffer().

# 10.273 cpukit/rtems/src/ratemon.c File Reference

Classic Rate Monotonic Information with Zero Objects.

```
#include <rtems/rtems/ratemondata.h>
```

## Variables

- Objects_Information _Rate_monotonic_Information = { ( (Objects_Id) ( (Objects_Id) OBJECTS_CLASSIC_API << 24U ) | ( (Objects_Id) OBJECTS_RTEMS_PERIODS << 27U ) | ( (Objects_Id) 1 << 16U ) | ( (Objects_Id) 0 << 0U ) ), NULL, _Objects_Allocate_none, NULL, 0, 0, 0, 0 , { { { &( _Rate_monotonic_Information.Inactive ).Tail.Node, NULL }, &( _Rate_monotonic_Information.Inactive ).Head.Node } }, NULL, NULL, NULL }

  *The Classic Rate Monotonic objects information.*

### 10.273.1 Detailed Description

Classic Rate Monotonic Information with Zero Objects.

# 10.274 cpukit/rtems/src/ratemoncancel.c File Reference

RTEMS Rate Monotonic Cancel.

```
#include <rtems/rtems/ratemonimpl.h>
```

### Functions

- void **_Rate_monotonic_Cancel** (Rate_monotonic_Control *the_period, Thread_Control *owner, ISR_lock_Context *lock_context)
- rtems_status_code rtems_rate_monotonic_cancel (rtems_id id)

  %

#### 10.274.1 Detailed Description

RTEMS Rate Monotonic Cancel.

## 10.275 cpukit/rtems/src/ratemoncreate.c File Reference

Create a Period.

```
#include <rtems/rtems/ratemonimpl.h>
#include <rtems/rtems/status.h>
#include <rtems/rtems/support.h>
#include <rtems/score/isr.h>
#include <rtems/score/thread.h>
#include <rtems/score/watchdogimpl.h>
#include <rtems/sysinit.h>
```

### Enumerations

- enum {
  **_Sysinit_bsp_start** = 0x00030080, **_Sysinit_bsp_debug_uart_init** = 0x00030003, **_Sysinit_amba_↩ initialize** = 0x00030001, **_Sysinit_leon3_cpu_index_init** = 0x00030000,
  **_Sysinit_leon3_counter_initialize** = 0x00040000, **_Sysinit_bsp_memory_initialize** = 0x00018080, _↩ **Sysinit__Malloc_Initialize** = 0x00028080, **_Sysinit__Barrier_Manager_initialization** = 0x00140080,
  **_Sysinit__Message_queue_Manager_initialization** = 0x000e0080, **_Sysinit__Partition_Manager_↩ initialization** = 0x00100080, **_Sysinit__Rate_monotonic_Manager_initialization** = 0x00130080, _↩ **Sysinit__Semaphore_Manager_initialization** = 0x000f0080,
  **_Sysinit__RTEMS_tasks_Manager_initialization** = 0x000a0080, **_Sysinit__Timer_Manager_↩ initialization** = 0x000b0080, **_Sysinit_rtems_initialize_data_structures** = 0x00070080, **_Sysinit__↩ Extension_Manager_initialization** = 0x00090080,
  **_Sysinit__Per_CPU_Data_initialize** = 0x00022080, **_Sysinit__Workspace_Handler_initialization** = 0x00026080, **_Sysinit_init_task** = 0x00290080, **_Sysinit_init_runner_task** = 0x00290080 }

### Functions

- rtems_status_code rtems_rate_monotonic_create (rtems_name name, rtems_id *id)

  %
- static void **_Rate_monotonic_Manager_initialization** (void)

### Variables

- rtems_sysinit_item const **_Linker_set__Sysinit__Rate_monotonic_Manager_initialization** = { _Rate_↩ monotonic_Manager_initialization }

### 10.275.1 Detailed Description

Create a Period.

## 10.276 cpukit/rtems/src/ratemondelete.c File Reference

RTEMS Delete Rate Monotonic.

```
#include <rtems/rtems/ratemonimpl.h>
```

### Functions

- rtems_status_code rtems_rate_monotonic_delete (rtems_id id)
  %

### 10.276.1 Detailed Description

RTEMS Delete Rate Monotonic.

## 10.277 cpukit/rtems/src/ratemongetstatistics.c File Reference

RTEMS Rate Monotonic Get Statistics.

```
#include <rtems/rtems/ratemonimpl.h>
```

### Functions

- rtems_status_code rtems_rate_monotonic_get_statistics (rtems_id id, rtems_rate_monotonic_period_statistics *dst)
  %

### 10.277.1 Detailed Description

RTEMS Rate Monotonic Get Statistics.

## 10.278 cpukit/rtems/src/ratemongetstatus.c File Reference

RTEMS Rate Monotonic Get Status.

```
#include <rtems/rtems/ratemonimpl.h>
```

### Functions

- rtems_status_code rtems_rate_monotonic_get_status (rtems_id id, rtems_rate_monotonic_period_status *period_status)

  %

### 10.278.1 Detailed Description

RTEMS Rate Monotonic Get Status.

## 10.279 cpukit/rtems/src/ratemonident.c File Reference

rtems_rate_monotonic_ident() Implementation

```
#include <rtems/rtems/ratemonimpl.h>
#include <rtems/rtems/objectimpl.h>
```

### Functions

- rtems_status_code rtems_rate_monotonic_ident (rtems_name name, rtems_id *id)

  *Identifies a period object by the specified object name.*

### 10.279.1 Detailed Description

rtems_rate_monotonic_ident() Implementation

## 10.280 cpukit/rtems/src/ratemonperiod.c File Reference

Rate Monotonic Support.

```
#include <rtems/rtems/ratemonimpl.h>
#include <rtems/score/schedulerimpl.h>
#include <rtems/score/todimpl.h>
```

### Functions

- bool _Rate_monotonic_Get_status (const Rate_monotonic_Control *the_period, Timestamp_Control *wall↩ _since_last_period, Timestamp_Control *cpu_since_last_period)

  *_Rate_monotonic_Get_status(*
- static void **_Rate_monotonic_Release_postponed_job** (Rate_monotonic_Control *the_period, Thread_Control *owner, rtems_interval next_length, ISR_lock_Context *lock_context)
- static void **_Rate_monotonic_Release_job** (Rate_monotonic_Control *the_period, Thread_Control *owner, rtems_interval next_length, ISR_lock_Context *lock_context)
- void **_Rate_monotonic_Restart** (Rate_monotonic_Control *the_period, Thread_Control *owner, ISR_lock_Context *lock_context)
- static void **_Rate_monotonic_Update_statistics** (Rate_monotonic_Control *the_period)
- static rtems_status_code **_Rate_monotonic_Get_status_for_state** (rtems_rate_monotonic_period_states state)
- static rtems_status_code **_Rate_monotonic_Activate** (Rate_monotonic_Control *the_period, rtems_interval length, Thread_Control *executing, ISR_lock_Context *lock_context)
- static rtems_status_code **_Rate_monotonic_Block_while_active** (Rate_monotonic_Control *the_period, rtems_interval length, Thread_Control *executing, ISR_lock_Context *lock_context)
- static rtems_status_code **_Rate_monotonic_Block_while_expired** (Rate_monotonic_Control *the_period, rtems_interval length, Thread_Control *executing, ISR_lock_Context *lock_context)
- rtems_status_code rtems_rate_monotonic_period (rtems_id id, rtems_interval length)

  *%*

#### 10.280.1 Detailed Description

Rate Monotonic Support.

## 10.281 cpukit/rtems/src/ratemonresetstatistics.c File Reference

RTEMS Rate Monotonic Reset Statistics.

```
#include <rtems/rtems/ratemonimpl.h>
```

### Functions

- rtems_status_code rtems_rate_monotonic_reset_statistics (rtems_id id)

  *%*

#### 10.281.1 Detailed Description

RTEMS Rate Monotonic Reset Statistics.

## 10.282 cpukit/rtems/src/ratemontimeout.c File Reference

Rate Monotonic Timeout.

```
#include <rtems/rtems/ratemonimpl.h>
```

### Functions

- static void **_Rate_monotonic_Renew_deadline** (Rate_monotonic_Control *the_period, ISR_lock_Context *lock_context)
- void **_Rate_monotonic_Timeout** (Watchdog_Control *the_watchdog)

### 10.282.1 Detailed Description

Rate Monotonic Timeout.

## 10.283 cpukit/rtems/src/rtemsnametoid.c File Reference

_RTEMS_Name_to_id() Implementation

```
#include <rtems/rtems/objectimpl.h>
#include <rtems/rtems/statusimpl.h>
```

### Functions

- rtems_status_code _RTEMS_Name_to_id (uint32_t name, uint32_t node, Objects_Id *id, const Objects_Information *information)

  *Calls _Objects_Name_to_id_u32() and converts the status.*

### 10.283.1 Detailed Description

_RTEMS_Name_to_id() Implementation

## 10.284 cpukit/rtems/src/sem.c File Reference

Classic Semaphore Information with Zero Objects.

```
#include <rtems/rtems/semdata.h>
```

### Variables

- Objects_Information _Semaphore_Information = { ( (Objects_Id) ( (Objects_Id) OBJECTS_CLASSIC_API << 24U ) | ( (Objects_Id) OBJECTS_RTEMS_SEMAPHORES << 27U ) | ( (Objects_Id) 1 << 16U ) | ( (Objects_Id) 0 << 0U ) ), NULL, _Objects_Allocate_none, NULL, 0, 0, 0, 0 , { { { &( _Semaphore_Information.Inactive ).Tail.Node, NULL }, &( _Semaphore_Information.Inactive ).Head.Node } }, NULL, NULL, NULL }

  *The Classic Semaphore objects information.*

### 10.284.1 Detailed Description

Classic Semaphore Information with Zero Objects.

## 10.285 cpukit/rtems/src/semcreate.c File Reference

rtems_semaphore_create

```
#include <rtems/rtems/semimpl.h>
#include <rtems/rtems/attrimpl.h>
#include <rtems/rtems/statusimpl.h>
#include <rtems/rtems/support.h>
#include <rtems/rtems/tasksimpl.h>
#include <rtems/score/schedulerimpl.h>
#include <rtems/score/sysstate.h>
#include <rtems/sysinit.h>
```

### Macros

- #define **SEMAPHORE_KIND_MASK**

### Enumerations

- enum {
  **_Sysinit_bsp_start** = 0x00030080, **_Sysinit_bsp_debug_uart_init** = 0x00030003, **_Sysinit_amba_↩ initialize** = 0x00030001, **_Sysinit_leon3_cpu_index_init** = 0x00030000,
  **_Sysinit_leon3_counter_initialize** = 0x00040000, **_Sysinit_bsp_memory_initialize** = 0x00018080, **_↩ Sysinit__Malloc_Initialize** = 0x00028080, **_Sysinit__Barrier_Manager_initialization** = 0x00140080,
  **_Sysinit__Message_queue_Manager_initialization** = 0x000e0080, **_Sysinit__Partition_Manager_↩ initialization** = 0x00100080, **_Sysinit__Rate_monotonic_Manager_initialization** = 0x00130080, **_↩ Sysinit__Semaphore_Manager_initialization** = 0x000f0080,
  **_Sysinit__RTEMS_tasks_Manager_initialization** = 0x000a0080, **_Sysinit__Timer_Manager_↩ initialization** = 0x000b0080, **_Sysinit_rtems_initialize_data_structures** = 0x00070080, **_Sysinit__↩ Extension_Manager_initialization** = 0x00090080,
  **_Sysinit__Per_CPU_Data_initialize** = 0x00022080, **_Sysinit__Workspace_Handler_initialization** = 0x00026080, **_Sysinit_init_task** = 0x00290080, **_Sysinit_init_runner_task** = 0x00290080 }

### Functions

- rtems_status_code rtems_semaphore_create (rtems_name name, uint32_t count, rtems_attribute attribute↩ _set, rtems_task_priority priority_ceiling, rtems_id *id)

  *Creates a semaphore with the specified properties and returns its identifier.*
- static void **_Semaphore_Manager_initialization** (void)

### Variables

- rtems_sysinit_item const **_Linker_set__Sysinit__Semaphore_Manager_initialization** = { _Semaphore↩ _Manager_initialization }

### 10.285.1 Detailed Description

rtems_semaphore_create

### 10.285.2 Macro Definition Documentation

#### 10.285.2.1 SEMAPHORE_KIND_MASK

```
#define SEMAPHORE_KIND_MASK
```

**Value:**

```
  ( RTEMS_SEMAPHORE_CLASS | RTEMS_INHERIT_PRIORITY \
  | RTEMS_PRIORITY_CEILING | RTEMS_MULTIPROCESSOR_RESOURCE_SHARING )
```

Definition at line 31 of file semcreate.c.

## 10.286 cpukit/rtems/src/semdelete.c File Reference

RTEMS Delete Semaphore.

```
#include <rtems/rtems/semimpl.h>
#include <rtems/rtems/statusimpl.h>
```

### Functions

- rtems_status_code rtems_semaphore_delete (rtems_id id)

  %

### 10.286.1 Detailed Description

RTEMS Delete Semaphore.

## 10.287 cpukit/rtems/src/semident.c File Reference

rtems_semaphore_ident() Implementation

```
#include <rtems/rtems/semimpl.h>
#include <rtems/rtems/objectimpl.h>
```

### Functions

- rtems_status_code rtems_semaphore_ident (rtems_name name, uint32_t node, rtems_id *id)

  *Identifies a semaphore object by the specified object name.*

### 10.287.1 Detailed Description

rtems_semaphore_ident() Implementation

## 10.288 cpukit/rtems/src/semobtain.c File Reference

RTEMS Obtain Semaphore.

```
#include <rtems/rtems/semimpl.h>
#include <rtems/rtems/optionsimpl.h>
#include <rtems/rtems/statusimpl.h>
```

### Functions

- **THREAD_QUEUE_OBJECT_ASSERT** (Semaphore_Control, Core_control.Wait_queue, SEMAPHORE_↩ CONTROL_GENERIC)
- **THREAD_QUEUE_OBJECT_ASSERT** (Semaphore_Control, Core_control.Mutex.Recursive.Mutex.Wait↩ _queue, SEMAPHORE_CONTROL_MUTEX)
- **THREAD_QUEUE_OBJECT_ASSERT** (Semaphore_Control, Core_control.Semaphore.Wait_queue, SE↩ MAPHORE_CONTROL_SEMAPHORE)
- **THREAD_QUEUE_OBJECT_ASSERT** (Semaphore_Control, Core_control.MRSP.Wait_queue, SEMAPH↩ ORE_CONTROL_MRSP)
- rtems_status_code rtems_semaphore_obtain (rtems_id id, rtems_option option_set, rtems_interval timeout)

  *%*

### 10.288.1 Detailed Description

RTEMS Obtain Semaphore.

## 10.289 cpukit/rtems/src/semrelease.c File Reference

RTEMS Semaphore Release.

```
#include <rtems/rtems/semimpl.h>
#include <rtems/rtems/statusimpl.h>
```

### Functions

- rtems_status_code rtems_semaphore_release (rtems_id id)

  *%*

### 10.289.1 Detailed Description

RTEMS Semaphore Release.

This file contains the implementation of the Classic API directive rtems_semaphore_release().

## 10.290 cpukit/rtems/src/status.c File Reference

Status Mapping Arrays.

```
#include <rtems/rtems/statusimpl.h>
```

### Variables

- const rtems_status_code _Status_Object_name_errors_to_status [ ]
  *Status Object Name Errors to Status Array.*

### 10.290.1 Detailed Description

Status Mapping Arrays.

## 10.291 cpukit/rtems/src/statustext.c File Reference

```
#include <rtems.h>
```

### Functions

- const char * rtems_status_text (rtems_status_code code)
  *Maps the status code to a descriptive text.*

### Variables

- static const char *const **status_code_text** [ ]

## 10.292 cpukit/rtems/src/systemeventreceive.c File Reference

This source file contains the implementation of rtems_event_system_receive().

```
#include <rtems/rtems/eventimpl.h>
#include <rtems/rtems/tasksdata.h>
#include <rtems/score/statesimpl.h>
#include <rtems/score/threadimpl.h>
```

## Functions

- rtems_status_code rtems_event_system_receive (rtems_event_set event_in, rtems_option option_set, rtems_interval ticks, rtems_event_set *event_out)

  *Receives or gets a system event set from the executing task.*

### 10.292.1 Detailed Description

This source file contains the implementation of rtems_event_system_receive().

### 10.292.2 Function Documentation

#### 10.292.2.1 rtems_event_system_receive()

```
rtems_status_code rtems_event_system_receive (
            rtems_event_set event_in,
            rtems_option option_set,
            rtems_interval ticks,
            rtems_event_set * event_out )
```

Receives or gets a system event set from the executing task.

This directive performs the same actions as the rtems_event_receive() directive except that it operates with a different set of events for each task.

**Parameters**

| | |
|---|---|
| *event_in* | is the event set of interest. Use RTEMS_PENDING_EVENTS to get the pending events. |
| *option_set* | is the option set. |
| *ticks* | is the timeout in clock ticks if the RTEMS_WAIT option was set. Use RTEMS_NO_TIMEOUT to wait potentially forever. |
| *event_out* | is the pointer to an event set. The received or pending events are stored in the referenced event set if the operation was successful. |

Definition at line 33 of file systemeventreceive.c.

## 10.293 cpukit/rtems/src/systemeventsend.c File Reference

This source file contains the implementation of rtems_event_system_send().

```
#include <rtems/rtems/eventimpl.h>
#include <rtems/rtems/tasksdata.h>
#include <rtems/score/threadimpl.h>
```

## Functions

- rtems_status_code rtems_event_system_send (rtems_id id, rtems_event_set event_in)

  *Sends the system event set to the task.*

### 10.293.1 Detailed Description

This source file contains the implementation of rtems_event_system_send().

### 10.293.2 Function Documentation

#### 10.293.2.1 rtems_event_system_send()

```
rtems_status_code rtems_event_system_send (
            rtems_id id,
            rtems_event_set event_in )
```

Sends the system event set to the task.

**Parameters**

| | |
|---|---|
| *id* | is the identifier of the target task to receive the event set. |
| *event_in* | is the event set to send. |

Definition at line 32 of file systemeventsend.c.

## 10.294 cpukit/rtems/src/taskconstruct.c File Reference

RTEMS Task Create from Config.

```
#include <rtems/rtems/tasksimpl.h>
#include <rtems/rtems/attrimpl.h>
#include <rtems/rtems/eventimpl.h>
#include <rtems/rtems/modesimpl.h>
#include <rtems/rtems/support.h>
#include <rtems/score/apimutex.h>
#include <rtems/score/schedulerimpl.h>
#include <rtems/score/stackimpl.h>
#include <rtems/score/threadimpl.h>
#include <rtems/score/userextimpl.h>
#include <rtems/sysinit.h>
#include <string.h>
```

## Enumerations

- enum {
  **_Sysinit_bsp_start** = 0x00030080, **_Sysinit_bsp_debug_uart_init** = 0x00030003, **_Sysinit_amba_↩ initialize** = 0x00030001, **_Sysinit_leon3_cpu_index_init** = 0x00030000,
  **_Sysinit_leon3_counter_initialize** = 0x00040000, **_Sysinit_bsp_memory_initialize** = 0x00018080, **_↩ Sysinit__Malloc_Initialize** = 0x00028080, **_Sysinit__Barrier_Manager_initialization** = 0x00140080,
  **_Sysinit__Message_queue_Manager_initialization** = 0x000e0080, **_Sysinit__Partition_Manager_↩ initialization** = 0x00100080, **_Sysinit__Rate_monotonic_Manager_initialization** = 0x00130080, **_↩ Sysinit__Semaphore_Manager_initialization** = 0x000f0080,
  **_Sysinit__RTEMS_tasks_Manager_initialization** = 0x000a0080, **_Sysinit__Timer_Manager_↩ initialization** = 0x000b0080, **_Sysinit_rtems_initialize_data_structures** = 0x00070080, **_Sysinit__↩ Extension_Manager_initialization** = 0x00090080,
  **_Sysinit__Per_CPU_Data_initialize** = 0x00022080, **_Sysinit__Workspace_Handler_initialization** = 0x00026080, **_Sysinit_init_task** = 0x00290080, **_Sysinit_init_runner_task** = 0x00290080 }

## Functions

- static rtems_status_code **_RTEMS_tasks_Prepare_user_stack** (Thread_Configuration *thread_config, const rtems_task_config *config)
- rtems_status_code rtems_task_construct (const rtems_task_config *config, rtems_id *id)

  *Constructs a task from the specified the task configuration.*
- rtems_status_code **_RTEMS_tasks_Create** (const rtems_task_config *config, rtems_id *id, RTEMS_↩ tasks_Prepare_stack prepare_stack)
- static void **_RTEMS_tasks_Start_extension** (Thread_Control *executing, Thread_Control *started)
- static void **_RTEMS_tasks_Manager_initialization** (void)

## Variables

- static User_extensions_Control **_RTEMS_tasks_User_extensions**
- rtems_sysinit_item const **_Linker_set__Sysinit__RTEMS_tasks_Manager_initialization** = { _RTEMS_↩ tasks_Manager_initialization }

### 10.294.1 Detailed Description

RTEMS Task Create from Config.

### 10.294.2 Variable Documentation

#### 10.294.2.1 _RTEMS_tasks_User_extensions

```
User_extensions_Control _RTEMS_tasks_User_extensions  [static]
```

**Initial value:**

```
= {
  .Callouts = {
    .thread_start     = _RTEMS_tasks_Start_extension,
    .thread_restart   = _RTEMS_tasks_Start_extension
  }
}
```

Definition at line 282 of file taskconstruct.c.

## 10.295 cpukit/rtems/src/taskdelete.c File Reference

RTEMS Delete Task.

```
#include <rtems/rtems/tasksimpl.h>
#include <rtems/score/threadimpl.h>
```

### Functions

- rtems_status_code rtems_task_delete (rtems_id id)
  %

### 10.295.1 Detailed Description

RTEMS Delete Task.

## 10.296 cpukit/rtems/src/taskgetaffinity.c File Reference

RTEMS Task Get Affinity.

```
#include <rtems/rtems/tasks.h>
#include <rtems/score/threadimpl.h>
#include <rtems/score/schedulerimpl.h>
```

### Functions

- rtems_status_code rtems_task_get_affinity (rtems_id id, size_t cpusetsize, cpu_set_t *cpuset)
  %

### 10.296.1 Detailed Description

RTEMS Task Get Affinity.

## 10.297 cpukit/rtems/src/taskident.c File Reference

rtems_task_ident() Implementation

```
#include <rtems/rtems/tasksimpl.h>
#include <rtems/rtems/objectimpl.h>
#include <rtems/score/percpu.h>
```

### Functions

- rtems_status_code rtems_task_ident (rtems_name name, uint32_t node, rtems_id *id)

  *Identifies a task object by the specified object name.*

### 10.297.1 Detailed Description

rtems_task_ident() Implementation

## 10.298 cpukit/rtems/src/taskinitusers.c File Reference

_RTEMS_tasks_Initialize_user_tasks_body

```
#include <rtems/rtems/tasksimpl.h>
#include <rtems/score/assert.h>
#include <rtems/score/threadimpl.h>
#include <rtems/score/interr.h>
```

### Functions

- void _RTEMS_tasks_Initialize_user_task (void)

  *System initialization handler to create and start the first user task.*

### 10.298.1 Detailed Description

_RTEMS_tasks_Initialize_user_tasks_body

## 10.299 cpukit/rtems/src/taskissuspended.c File Reference

rtems_task_is_suspended

```
#include <rtems/rtems/tasks.h>
#include <rtems/score/threadimpl.h>
```

### Functions

- rtems_status_code rtems_task_is_suspended (rtems_id id)

  %

### 10.299.1 Detailed Description

rtems_task_is_suspended

## 10.300 cpukit/rtems/src/taskrestart.c File Reference

RTEMS Task Restart.

```
#include <rtems/rtems/tasks.h>
#include <rtems/score/threadimpl.h>
```

### Functions

- rtems_status_code rtems_task_restart (rtems_id id, rtems_task_argument argument)
  %

### 10.300.1 Detailed Description

RTEMS Task Restart.

## 10.301 cpukit/rtems/src/taskresume.c File Reference

RTEMS Resume Task.

```
#include <rtems/rtems/tasksimpl.h>
#include <rtems/score/threadimpl.h>
```

### Functions

- rtems_status_code rtems_task_resume (rtems_id id)
  %

### 10.301.1 Detailed Description

RTEMS Resume Task.

## 10.302 cpukit/rtems/src/tasks.c File Reference

RTEMS Task API Extensions.

```
#include <rtems/rtems/tasksdata.h>
```

### Functions

- **THREAD_INFORMATION_DEFINE_ZERO** (_RTEMS_tasks, OBJECTS_CLASSIC_API, OBJECTS_RT↩ EMS_TASKS)

### 10.302.1 Detailed Description

RTEMS Task API Extensions.

## 10.303 cpukit/rtems/src/taskself.c File Reference

RTEMS Get Self Task Id.

```
#include <rtems/rtems/tasksimpl.h>
```

### Functions

- rtems_id rtems_task_self (void)
  %

### 10.303.1 Detailed Description

RTEMS Get Self Task Id.

## 10.304 cpukit/rtems/src/tasksetaffinity.c File Reference

RTEMS Task Set Affinity.

```
#include <rtems/rtems/tasks.h>
#include <rtems/score/threadimpl.h>
#include <rtems/score/schedulerimpl.h>
```

### Functions

- rtems_status_code rtems_task_set_affinity (rtems_id id, size_t cpusetsize, const cpu_set_t *cpuset)
  %

### 10.304.1 Detailed Description

RTEMS Task Set Affinity.

## 10.305 cpukit/rtems/src/tasksetpriority.c File Reference

RTEMS Set Task Priority.

```
#include <rtems/rtems/tasksimpl.h>
#include <rtems/score/schedulerimpl.h>
#include <rtems/score/threadimpl.h>
```

### Functions

- static rtems_status_code **_RTEMS_tasks_Set_priority** (Thread_Control *the_thread, const Scheduler_Control *scheduler, Priority_Control new_priority, Thread_queue_Context *queue_context)
- rtems_status_code rtems_task_set_priority (rtems_id id, rtems_task_priority new_priority, rtems_task_priority *old_priority_p)

  %

#### 10.305.1 Detailed Description

RTEMS Set Task Priority.

## 10.306 cpukit/rtems/src/taskstart.c File Reference

RTEMS Start Task.

```
#include <rtems/rtems/tasks.h>
#include <rtems/score/threadimpl.h>
```

### Functions

- rtems_status_code rtems_task_start (rtems_id id, rtems_task_entry entry_point, rtems_task_argument argument)

  %

#### 10.306.1 Detailed Description

RTEMS Start Task.

## 10.307 cpukit/rtems/src/tasksuspend.c File Reference

RTEMS Suspend Task.

```
#include <rtems/rtems/tasksimpl.h>
#include <rtems/score/threadimpl.h>
```

### Functions

- rtems_status_code rtems_task_suspend (rtems_id id)

  %

### 10.307.1 Detailed Description

RTEMS Suspend Task.

## 10.308 cpukit/rtems/src/taskwakeafter.c File Reference

RTEMS Task Wake After.

```
#include <rtems/rtems/tasks.h>
#include <rtems/score/threadimpl.h>
#include <rtems/score/watchdogimpl.h>
```

### Functions

- rtems_status_code rtems_task_wake_after (rtems_interval ticks)
  %

### 10.308.1 Detailed Description

RTEMS Task Wake After.

## 10.309 cpukit/rtems/src/timercreate.c File Reference

RTEMS Create Timer.

```
#include <rtems/rtems/timerimpl.h>
#include <rtems/rtems/clock.h>
#include <rtems/rtems/status.h>
#include <rtems/rtems/support.h>
#include <rtems/score/assert.h>
#include <rtems/score/chainimpl.h>
#include <rtems/score/thread.h>
#include <rtems/score/todimpl.h>
#include <rtems/score/watchdogimpl.h>
#include <rtems/sysinit.h>
```

### Enumerations

- enum {
  **_Sysinit_bsp_start** = 0x00030080, **_Sysinit_bsp_debug_uart_init** = 0x00030003, **_Sysinit_amba_↩ initialize** = 0x00030001, **_Sysinit_leon3_cpu_index_init** = 0x00030000,
  **_Sysinit_leon3_counter_initialize** = 0x00040000, **_Sysinit_bsp_memory_initialize** = 0x00018080, _↩ **Sysinit__Malloc_Initialize** = 0x00028080, **_Sysinit__Barrier_Manager_initialization** = 0x00140080,
  **_Sysinit__Message_queue_Manager_initialization** = 0x000e0080, **_Sysinit__Partition_Manager_↩ initialization** = 0x00100080, **_Sysinit__Rate_monotonic_Manager_initialization** = 0x00130080, _↩ **Sysinit__Semaphore_Manager_initialization** = 0x000f0080,
  **_Sysinit__RTEMS_tasks_Manager_initialization** = 0x000a0080, **_Sysinit__Timer_Manager_↩ initialization** = 0x000b0080, **_Sysinit_rtems_initialize_data_structures** = 0x00070080, **_Sysinit__↩ Extension_Manager_initialization** = 0x00090080,
  **_Sysinit__Per_CPU_Data_initialize** = 0x00022080, **_Sysinit__Workspace_Handler_initialization** = 0x00026080, **_Sysinit_init_task** = 0x00290080, **_Sysinit_init_runner_task** = 0x00290080 }

### Functions

- **RTEMS_STATIC_ASSERT** (PER_CPU_WATCHDOG_REALTIME==TIMER_CLASS_BIT_TIME_OF_DAY, TIMER_CLASS_BIT_TIME_OF_DAY)
- void **_Timer_Routine_adaptor** (Watchdog_Control *the_watchdog)
- rtems_status_code **_Timer_Fire** (rtems_id id, rtems_interval interval, rtems_timer_service_routine_entry routine, void *user_data, Timer_Classes the_class, Watchdog_Service_routine_entry adaptor)
- rtems_status_code **_Timer_Fire_after** (rtems_id id, rtems_interval ticks, rtems_timer_service_routine_entry routine, void *user_data, Timer_Classes the_class, Watchdog_Service_routine_entry adaptor)
- rtems_status_code **_Timer_Fire_when** (rtems_id id, const rtems_time_of_day *wall_time, rtems_timer_service_routine_entry routine, void *user_data, Timer_Classes the_class, Watchdog_Service_routine_entry adaptor)
- void **_Timer_Cancel** (Per_CPU_Control *cpu, Timer_Control *the_timer)
- rtems_status_code rtems_timer_create (rtems_name name, rtems_id *id)

  %
- static void **_Timer_Manager_initialization** (void)

### Variables

- Timer_server_Control *volatile _Timer_server

  *Pointer to default timer server control block.*
- rtems_sysinit_item const **_Linker_set__Sysinit__Timer_Manager_initialization** = { _Timer_Manager_↩ initialization }

#### 10.309.1 Detailed Description

RTEMS Create Timer.

## 10.310 cpukit/rtems/src/timerdelete.c File Reference

RTEMS Delete Timer.

```
#include <rtems/rtems/timerimpl.h>
```

### Functions

- rtems_status_code rtems_timer_delete (rtems_id id)

  %

#### 10.310.1 Detailed Description

RTEMS Delete Timer.

## 10.311 cpukit/rtems/src/timerfireafter.c File Reference

RTEMS Timer Fire After.

```
#include <rtems/rtems/timerimpl.h>
```

### Functions

- rtems_status_code rtems_timer_fire_after (rtems_id id, rtems_interval ticks, rtems_timer_service_routine_entry routine, void *user_data)

  %

### 10.311.1 Detailed Description

RTEMS Timer Fire After.

## 10.312 cpukit/rtems/src/timerident.c File Reference

rtems_timer_ident() Implementation

```
#include <rtems/rtems/timerimpl.h>
#include <rtems/rtems/objectimpl.h>
```

### Functions

- rtems_status_code rtems_timer_ident (rtems_name name, rtems_id *id)

  *Identifies a timer object by the specified object name.*

### 10.312.1 Detailed Description

rtems_timer_ident() Implementation

## 10.313 cpukit/rtems/src/timerreset.c File Reference

RTEMS Timer Reset.

```
#include <rtems/rtems/timerimpl.h>
```

### Functions

- rtems_status_code rtems_timer_reset (rtems_id id)

  %

### 10.313.1 Detailed Description

RTEMS Timer Reset.

## 10.314 cpukit/sapi/src/exinit.c File Reference

Initialization Manager.

```
#include <rtems/config.h>
#include <rtems/extensionimpl.h>
#include <rtems/init.h>
#include <rtems/sysinit.h>
#include <rtems/score/sysstate.h>
#include <rtems/score/copyrt.h>
#include <rtems/score/heap.h>
#include <rtems/score/interr.h>
#include <rtems/score/isr.h>
#include <rtems/score/percpudata.h>
#include <rtems/score/priority.h>
#include <rtems/score/schedulerimpl.h>
#include <rtems/score/smpimpl.h>
#include <rtems/score/timecounter.h>
#include <rtems/score/threadimpl.h>
#include <rtems/score/todimpl.h>
#include <rtems/score/wkspace.h>
```

### Enumerations

- enum {
  **_Sysinit_bsp_start** = 0x00030080, **_Sysinit_bsp_debug_uart_init** = 0x00030003, **_Sysinit_amba_initialize** = 0x00030001, **_Sysinit_leon3_cpu_index_init** = 0x00030000,
  **_Sysinit_leon3_counter_initialize** = 0x00040000, **_Sysinit_bsp_memory_initialize** = 0x00018080, **_Sysinit__Malloc_Initialize** = 0x00028080, **_Sysinit__Barrier_Manager_initialization** = 0x00140080,
  **_Sysinit__Message_queue_Manager_initialization** = 0x000e0080, **_Sysinit__Partition_Manager_initialization** = 0x00100080, **_Sysinit__Rate_monotonic_Manager_initialization** = 0x00130080, **_Sysinit__Semaphore_Manager_initialization** = 0x000f0080,
  **_Sysinit__RTEMS_tasks_Manager_initialization** = 0x000a0080, **_Sysinit__Timer_Manager_initialization** = 0x000b0080, **_Sysinit_rtems_initialize_data_structures** = 0x00070080, **_Sysinit__Extension_Manager_initialization** = 0x00090080,
  **_Sysinit__Per_CPU_Data_initialize** = 0x00022080, **_Sysinit__Workspace_Handler_initialization** = 0x00026080, **_Sysinit_init_task** = 0x00290080, **_Sysinit_init_runner_task** = 0x00290080 }
- enum { **_Sysinit_leon3_pre_driver_hook** = 0x00220080, **_Sysinit__Thread_Create_idle** = 0x001d0080 }

### Functions

- **RTEMS_SECTION** (".rtemsroset.copyright") const
- **RTEMS_LINKER_RWSET** (_Per_CPU_Data, char)
- static void **rtems_initialize_data_structures** (void)
- **RTEMS_LINKER_ROSET** (_Sysinit, rtems_sysinit_item)
- void rtems_initialize_executive (void)

  *Initializes the system and starts multitasking.*

### Variables

- rtems_sysinit_item const **_Linker_set__Sysinit_rtems_initialize_data_structures** = { rtems_initialize_↩ data_structures }
- rtems_sysinit_item const **_Linker_set__Sysinit__Thread_Create_idle** = { _Thread_Create_idle }

### 10.314.1 Detailed Description

Initialization Manager.

## 10.315 cpukit/sapi/src/extension.c File Reference

Extension Manager Information with Zero Objects.

```
#include <rtems/extensiondata.h>
```

### Variables

- Objects_Information _Extension_Information = { ( (Objects_Id) ( (Objects_Id) OBJECTS_CLASSIC_API << 24U ) | ( (Objects_Id) OBJECTS_RTEMS_EXTENSIONS << 27U ) | ( (Objects_Id) 1 << 16U ) | ( (Objects_Id) 0 << 0U ) ), NULL, _Objects_Allocate_none, NULL, 0, 0, 0, 0 , { { { &( _Extension_Information.↩ Inactive ).Tail.Node, NULL }, &( _Extension_Information.Inactive ).Head.Node } }, NULL, NULL, NULL }

  *The Classic Extensions objects information.*

### 10.315.1 Detailed Description

Extension Manager Information with Zero Objects.

## 10.316 cpukit/sapi/src/extensioncreate.c File Reference

User Extensions Implementation.

```
#include <rtems/extensionimpl.h>
#include <rtems/rtems/support.h>
#include <rtems/score/userextimpl.h>
#include <rtems/sysinit.h>
```

### Enumerations

- enum {
  **_Sysinit_bsp_start** = 0x00030080, **_Sysinit_bsp_debug_uart_init** = 0x00030003, **_Sysinit_amba_↩ initialize** = 0x00030001, **_Sysinit_leon3_cpu_index_init** = 0x00030000,
  **_Sysinit_leon3_counter_initialize** = 0x00040000, **_Sysinit_bsp_memory_initialize** = 0x00018080, _↩ **Sysinit__Malloc_Initialize** = 0x00028080, **_Sysinit__Barrier_Manager_initialization** = 0x00140080,
  **_Sysinit__Message_queue_Manager_initialization** = 0x000e0080, **_Sysinit__Partition_Manager_↩ initialization** = 0x00100080, **_Sysinit__Rate_monotonic_Manager_initialization** = 0x00130080, _↩ **Sysinit__Semaphore_Manager_initialization** = 0x000f0080,
  **_Sysinit__RTEMS_tasks_Manager_initialization** = 0x000a0080, **_Sysinit__Timer_Manager_↩ initialization** = 0x000b0080, **_Sysinit_rtems_initialize_data_structures** = 0x00070080, **_Sysinit__↩ Extension_Manager_initialization** = 0x00090080,
  **_Sysinit__Per_CPU_Data_initialize** = 0x00022080, **_Sysinit__Workspace_Handler_initialization** = 0x00026080, **_Sysinit_init_task** = 0x00290080, **_Sysinit_init_runner_task** = 0x00290080 }

### Functions

- rtems_status_code rtems_extension_create (rtems_name name, const rtems_extensions_table *extension↩ _table, rtems_id *id)

  %
- static void **_Extension_Manager_initialization** (void)

### Variables

- rtems_sysinit_item const **_Linker_set__Sysinit__Extension_Manager_initialization** = { _Extension_↩ Manager_initialization }

#### 10.316.1 Detailed Description

User Extensions Implementation.

## 10.317 cpukit/sapi/src/extensiondelete.c File Reference

User Extensions Implementation.

```
#include <rtems/extensionimpl.h>
#include <rtems/score/userextimpl.h>
```

### Functions

- rtems_status_code rtems_extension_delete (rtems_id id)

  %

#### 10.317.1 Detailed Description

User Extensions Implementation.

# 10.318 cpukit/sapi/src/extensionident.c File Reference

rtems_extension_ident() Implementation

```
#include <rtems/extensionimpl.h>
#include <rtems/rtems/objectimpl.h>
```

## Functions

- rtems_status_code rtems_extension_ident (rtems_name name, rtems_id *id)

  *Identifies an extension set object by the specified object name.*

### 10.318.1 Detailed Description

rtems_extension_ident() Implementation

# 10.319 cpukit/sapi/src/getversionstring.c File Reference

Get the RTEMS Version as a String.

```
#include <rtems/config.h>
#include <rtems/score/copyrt.h>
```

## Functions

- const char * rtems_get_version_string (void)

  *Gets the RTEMS version string.*

### 10.319.1 Detailed Description

Get the RTEMS Version as a String.

# 10.320 cpukit/sapi/src/version.c File Reference

Creates the version strings from the various pieces of version information. The main version number is part of the build system and is stamped into rtems/score/cpuopts.h. The version control key string is extracted from the version control tool when the code is being built and is updated if it has changed. The key may indicate there are local modification.

```
#include <rtems.h>
#include <rtems/version.h>
#include "version-vc-key.h"
```

### Functions

- const char * rtems_version (void)

  *Returns the version string.*
- int rtems_version_major (void)

  *Returns the version's major number.*
- int rtems_version_minor (void)

  *Returns the version's minor number.*
- int rtems_version_revision (void)

  *Returns the version's revision number.*
- const char * rtems_version_control_key (void)

  *Returns the version control key for the current version of code that has been built.*

#### 10.320.1 Detailed Description

Creates the version strings from the various pieces of version information. The main version number is part of the build system and is stamped into rtems/score/cpuopts.h. The version control key string is extracted from the version control tool when the code is being built and is updated if it has changed. The key may indicate there are local modification.

## 10.321 cpukit/score/cpu/sparc/cpu.c File Reference

SPARC CPU Dependent Source.

```
#include <rtems/score/isr.h>
#include <rtems/score/percpu.h>
#include <rtems/score/tls.h>
#include <rtems/score/thread.h>
#include <rtems/rtems/cache.h>
```

### Macros

- #define **SPARC_ASSERT_OFFSET**(field, off)
- #define **SPARC_ASSERT_ISF_OFFSET**(field, off)
- #define **SPARC_ASSERT_FP_OFFSET**(field, off)
- #define **HIGH_BITS_MASK** 0xFFFFFC00
- #define **HIGH_BITS_SHIFT** 10
- #define **LOW_BITS_MASK** 0x000003FF

## Functions

- **SPARC_ASSERT_OFFSET** (g5, G5)
- **SPARC_ASSERT_OFFSET** (g7, G7)
- **RTEMS_STATIC_ASSERT** (offsetof(Context_Control, l0_and_l1)==L0_OFFSET, Context_Control_offset↩ _L0)
- **RTEMS_STATIC_ASSERT** (offsetof(Context_Control, l0_and_l1)+4==L1_OFFSET, Context_Control_↩ offset_L1)
- **SPARC_ASSERT_OFFSET** (l2, L2)
- **SPARC_ASSERT_OFFSET** (l3, L3)
- **SPARC_ASSERT_OFFSET** (l4, L4)
- **SPARC_ASSERT_OFFSET** (l5, L5)
- **SPARC_ASSERT_OFFSET** (l6, L6)
- **SPARC_ASSERT_OFFSET** (l7, L7)
- **SPARC_ASSERT_OFFSET** (i0, I0)
- **SPARC_ASSERT_OFFSET** (i1, I1)
- **SPARC_ASSERT_OFFSET** (i2, I2)
- **SPARC_ASSERT_OFFSET** (i3, I3)
- **SPARC_ASSERT_OFFSET** (i4, I4)
- **SPARC_ASSERT_OFFSET** (i5, I5)
- **SPARC_ASSERT_OFFSET** (i6_fp, I6_FP)
- **SPARC_ASSERT_OFFSET** (i7, I7)
- **SPARC_ASSERT_OFFSET** (o6_sp, O6_SP)
- **SPARC_ASSERT_OFFSET** (o7, O7)
- **SPARC_ASSERT_OFFSET** (psr, PSR)
- **SPARC_ASSERT_OFFSET** (isr_dispatch_disable, ISR_DISPATCH_DISABLE_STACK)
- **SPARC_ASSERT_OFFSET** (is_executing, SPARC_CONTEXT_CONTROL_IS_EXECUTING)
- **SPARC_ASSERT_ISF_OFFSET** (psr, PSR)
- **SPARC_ASSERT_ISF_OFFSET** (pc, PC)
- **SPARC_ASSERT_ISF_OFFSET** (npc, NPC)
- **SPARC_ASSERT_ISF_OFFSET** (g1, G1)
- **SPARC_ASSERT_ISF_OFFSET** (g2, G2)
- **SPARC_ASSERT_ISF_OFFSET** (g3, G3)
- **SPARC_ASSERT_ISF_OFFSET** (g4, G4)
- **SPARC_ASSERT_ISF_OFFSET** (g5, G5)
- **SPARC_ASSERT_ISF_OFFSET** (g7, G7)
- **SPARC_ASSERT_ISF_OFFSET** (i0, I0)
- **SPARC_ASSERT_ISF_OFFSET** (i1, I1)
- **SPARC_ASSERT_ISF_OFFSET** (i2, I2)
- **SPARC_ASSERT_ISF_OFFSET** (i3, I3)
- **SPARC_ASSERT_ISF_OFFSET** (i4, I4)
- **SPARC_ASSERT_ISF_OFFSET** (i5, I5)
- **SPARC_ASSERT_ISF_OFFSET** (i6_fp, I6_FP)
- **SPARC_ASSERT_ISF_OFFSET** (i7, I7)
- **SPARC_ASSERT_ISF_OFFSET** (y, Y)
- **SPARC_ASSERT_ISF_OFFSET** (tpc, TPC)
- **SPARC_ASSERT_FP_OFFSET** (f0_f1, F0_F1)
- **SPARC_ASSERT_FP_OFFSET** (f2_f3, F2_F3)
- **SPARC_ASSERT_FP_OFFSET** (f4_f5, F4_F5)
- **SPARC_ASSERT_FP_OFFSET** (f6_f7, F6_F7)
- **SPARC_ASSERT_FP_OFFSET** (f8_f9, F8_F9)
- **SPARC_ASSERT_FP_OFFSET** (f10_f11, F10_F11)
- **SPARC_ASSERT_FP_OFFSET** (f12_f13, F12_F13)
- **SPARC_ASSERT_FP_OFFSET** (f14_f15, F14_F15)
- **SPARC_ASSERT_FP_OFFSET** (f16_f17, F16_F17)

- **SPARC_ASSERT_FP_OFFSET** (f18_f19, F18_F19)
- **SPARC_ASSERT_FP_OFFSET** (f20_f21, F20_F21)
- **SPARC_ASSERT_FP_OFFSET** (f22_f23, F22_F23)
- **SPARC_ASSERT_FP_OFFSET** (f24_f25, F24_F25)
- **SPARC_ASSERT_FP_OFFSET** (f26_f27, F26_F27)
- **SPARC_ASSERT_FP_OFFSET** (f28_f29, F28_F29)
- **SPARC_ASSERT_FP_OFFSET** (f30_f31, F30_F31)
- **SPARC_ASSERT_FP_OFFSET** (fsr, FSR)
- **RTEMS_STATIC_ASSERT** (sizeof(SPARC_Minimum_stack_frame)==SPARC_MINIMUM_STACK_FRAME_SIZE, SPARC_MINIMUM_STACK_FRAME_SIZE)
- **RTEMS_STATIC_ASSERT** (sizeof(CPU_Interrupt_frame) % CPU_ALIGNMENT==0, CPU_Interrupt_↩ frame_alignment)
- void _CPU_Initialize (void)

  *SPARC specific initialization.*
- uint32_t _CPU_ISR_Get_level (void)

  *Obtain the current interrupt disable level.*
- void _CPU_ISR_install_raw_handler (uint32_t vector, CPU_ISR_raw_handler new_handler, CPU_ISR_raw↩ _handler *old_handler)

  *SPARC specific raw ISR installer.*
- void _CPU_ISR_install_vector (uint32_t vector, CPU_ISR_handler new_handler, CPU_ISR_handler *old_↩ handler)

  *SPARC specific RTEMS ISR installer.*
- void _CPU_Context_Initialize (Context_Control *the_context, uint32_t *stack_base, uint32_t size, uint32_t new_level, void *entry_point, bool is_fp, void *tls_area)

## Variables

- const CPU_Trap_table_entry _CPU_Trap_slot_template

### 10.321.1 Detailed Description

SPARC CPU Dependent Source.

### 10.321.2 Macro Definition Documentation

#### 10.321.2.1 SPARC_ASSERT_FP_OFFSET

```
#define SPARC_ASSERT_FP_OFFSET(
              field,
              off )
```

**Value:**

```
RTEMS_STATIC_ASSERT( \
  offsetof(Context_Control_fp, field) == SPARC_FP_CONTEXT_OFFSET_ ## off, \
  Context_Control_fp_offset_ ## field \
)
```

Definition at line 112 of file cpu.c.

#### 10.321.2.2 SPARC_ASSERT_ISF_OFFSET

```
#define SPARC_ASSERT_ISF_OFFSET(
              field,
              off )
```

**Value:**

```
RTEMS_STATIC_ASSERT( \
  offsetof(CPU_Interrupt_frame, field) == ISF_ ## off ## _OFFSET, \
  CPU_Interrupt_frame_offset_ ## field \
)
```

Definition at line 86 of file cpu.c.

#### 10.321.2.3 SPARC_ASSERT_OFFSET

```
#define SPARC_ASSERT_OFFSET(
              field,
              off )
```

**Value:**

```
RTEMS_STATIC_ASSERT( \
  offsetof(Context_Control, field) == off ## _OFFSET, \
  Context_Control_offset_ ## field \
)
```

Definition at line 44 of file cpu.c.

### 10.321.3 Function Documentation

#### 10.321.3.1 _CPU_Context_Initialize()

```
void _CPU_Context_Initialize (
             Context_Control * the_context,
             uint32_t * stack_base,
             uint32_t size,
             uint32_t new_level,
             void * entry_point,
             bool is_fp,
             void * tls_area )
```

Initialize the context to a state suitable for starting a task after a context restore operation. Generally, this involves:

- setting a starting address
- preparing the stack
- preparing the stack and frame pointers
- setting the proper interrupt level in the context
- initializing the floating point context

**Parameters**

| | | |
|---|---|---|
| in | *the_context* | points to the context area |
| in | *stack_base* | is the low address of the allocated stack area |
| in | *size* | is the size of the stack area in bytes |
| in | *new_level* | is the interrupt level for the task |
| in | *entry_point* | is the task's entry point |
| in | *is_fp* | is set to TRUE if the task is a floating point task |
| in | *tls_area* | is the thread-local storage (TLS) area |

NOTE: Implemented as a subroutine for the SPARC port.

Definition at line 354 of file cpu.c.

#### 10.321.3.2 _CPU_Initialize()

```
void _CPU_Initialize (
             void  )
```

SPARC specific initialization.

This routine performs CPU dependent initialization.

Definition at line 176 of file cpu.c.

#### 10.321.3.3 _CPU_ISR_Get_level()

```
uint32_t _CPU_ISR_Get_level (
             void  )
```

Obtain the current interrupt disable level.

This method is invoked to return the current interrupt disable level.

**Returns**

This method returns the current interrupt disable level.

Definition at line 191 of file cpu.c.

#### 10.321.3.4 _CPU_ISR_install_raw_handler()

```
void _CPU_ISR_install_raw_handler (
             uint32_t vector,
             CPU_ISR_raw_handler new_handler,
             CPU_ISR_raw_handler * old_handler )
```

SPARC specific raw ISR installer.

This routine installs *new_handler* to be directly called from the trap table.

**Parameters**

| | | |
|---|---|---|
| in | *vector* | is the vector number |
| in | *new_handler* | is the new ISR handler |
| in | *old_handler* | will contain the old ISR handler |

Definition at line 237 of file cpu.c.

#### 10.321.3.5 _CPU_ISR_install_vector()

```
void _CPU_ISR_install_vector (
             uint32_t vector,
             CPU_ISR_handler new_handler,
             CPU_ISR_handler * old_handler )
```

SPARC specific RTEMS ISR installer.

This routine installs an interrupt vector.

**Parameters**

| | | |
|---|---|---|
| in | *vector* | is the vector number |
| in | *new_handler* | is the new ISR handler |
| in | *old_handler* | will contain the old ISR handler |

Definition at line 318 of file cpu.c.

### 10.321.4 Variable Documentation

#### 10.321.4.1 _CPU_Trap_slot_template

```
const CPU_Trap_table_entry _CPU_Trap_slot_template
```

**Initial value:**

```
= {
  0xa1480000,
  0x29000000,
  0x81c52000,
  0xa6102000
}
```

This is the set of opcodes for the instructions loaded into a trap table entry. The routine which installs a handler is responsible for filling in the fields for the _handler address and the _vector trap type.

The constants following this structure are masks for the fields which must be filled in when the handler is installed.

Definition at line 156 of file cpu.c.

## 10.322 cpukit/score/cpu/sparc/include/rtems/asm.h File Reference

Address the Problems Caused by Incompatible Flavor of Assemblers and Toolsets.

```
#include <rtems/score/cpuopts.h>
#include <rtems/score/cpu.h>
```

### Macros

- #define **ASM**
- #define **__USER_LABEL_PREFIX__** _
- #define **__REGISTER_PREFIX__**
- #define **SYM**(x) RTEMS_XCONCAT(__USER_LABEL_PREFIX__, x)
- #define **REG**(x) RTEMS_XCONCAT(__REGISTER_PREFIX__, x)
- #define **BEGIN_CODE_DCL** .text
- #define **END_CODE_DCL**
- #define **BEGIN_DATA_DCL** .data
- #define **END_DATA_DCL**
- #define **BEGIN_CODE** .text
- #define **END_CODE**
- #define **BEGIN_DATA**
- #define **END_DATA**
- #define **BEGIN_BSS**
- #define **END_BSS**
- #define **END**
- #define **PUBLIC**(sym) .globl SYM (sym)
- #define **EXTERN**(sym) .globl SYM (sym)
- #define **TRAP**(_vector, _handler)
- #define **RTRAP**(_vector, _handler)

### 10.322.1 Detailed Description

Address the Problems Caused by Incompatible Flavor of Assemblers and Toolsets.

This include file attempts to address the problems caused by incompatible flavors of assemblers and toolsets. It primarily addresses variations in the use of leading underscores on symbols and the requirement that register names be preceded by a %.

NOTE: The spacing in the use of these macros is critical to them working as advertised.

## 10.323 cpukit/score/cpu/sparc/include/rtems/score/cpu.h File Reference

SPARC CPU Department Source.

```
#include <rtems/score/basedefs.h>
#include <rtems/score/sparc.h>
```

## Classes

- struct SPARC_Minimum_stack_frame
- struct Context_Control

  *SPARC basic context.*
- struct Context_Control_fp

  *SPARC basic context.*
- struct CPU_Interrupt_frame

  *Interrupt stack frame (ISF).*
- struct CPU_Trap_table_entry
- struct CPU_Exception_frame
- struct SPARC_Counter

## Macros

- #define CPU_SIMPLE_VECTORED_INTERRUPTS TRUE
- #define CPU_ISR_PASSES_FRAME_POINTER FALSE
- #define CPU_HARDWARE_FP FALSE
- #define CPU_SOFTWARE_FP FALSE
- #define CPU_ALL_TASKS_ARE_FP FALSE
- #define CPU_IDLE_TASK_IS_FP FALSE
- #define **CPU_USE_DEFERRED_FP_SWITCH** FALSE
- #define **CPU_ENABLE_ROBUST_THREAD_DISPATCH** FALSE
- #define CPU_STACK_GROWS_UP FALSE
- #define **CPU_CACHE_LINE_BYTES** 64
- #define **CPU_STRUCTURE_ALIGNMENT** RTEMS_ALIGNED( CPU_CACHE_LINE_BYTES )
- #define CPU_MODES_INTERRUPT_MASK 0x0000000F
- #define CPU_STACK_FRAME_L0_OFFSET 0x00
- #define CPU_STACK_FRAME_L1_OFFSET 0x04
- #define CPU_STACK_FRAME_L2_OFFSET 0x08
- #define CPU_STACK_FRAME_L3_OFFSET 0x0c
- #define CPU_STACK_FRAME_L4_OFFSET 0x10
- #define CPU_STACK_FRAME_L5_OFFSET 0x14
- #define CPU_STACK_FRAME_L6_OFFSET 0x18
- #define CPU_STACK_FRAME_L7_OFFSET 0x1c
- #define CPU_STACK_FRAME_I0_OFFSET 0x20
- #define CPU_STACK_FRAME_I1_OFFSET 0x24
- #define CPU_STACK_FRAME_I2_OFFSET 0x28
- #define CPU_STACK_FRAME_I3_OFFSET 0x2c
- #define CPU_STACK_FRAME_I4_OFFSET 0x30
- #define CPU_STACK_FRAME_I5_OFFSET 0x34
- #define CPU_STACK_FRAME_I6_FP_OFFSET 0x38
- #define CPU_STACK_FRAME_I7_OFFSET 0x3c
- #define CPU_STRUCTURE_RETURN_ADDRESS_OFFSET 0x40
- #define CPU_STACK_FRAME_SAVED_ARG0_OFFSET 0x44
- #define CPU_STACK_FRAME_SAVED_ARG1_OFFSET 0x48
- #define CPU_STACK_FRAME_SAVED_ARG2_OFFSET 0x4c
- #define CPU_STACK_FRAME_SAVED_ARG3_OFFSET 0x50
- #define CPU_STACK_FRAME_SAVED_ARG4_OFFSET 0x54
- #define CPU_STACK_FRAME_SAVED_ARG5_OFFSET 0x58
- #define CPU_STACK_FRAME_PAD0_OFFSET 0x5c
- #define **CPU_MAXIMUM_PROCESSORS** 32
- #define _CPU_Context_Get_SP(_context) (_context)->o6_sp

- #define G5_OFFSET 0x00
- #define G7_OFFSET 0x04
- #define L0_OFFSET 0x08
- #define L1_OFFSET 0x0C
- #define L2_OFFSET 0x10
- #define L3_OFFSET 0x14
- #define L4_OFFSET 0x18
- #define L5_OFFSET 0x1C
- #define L6_OFFSET 0x20
- #define L7_OFFSET 0x24
- #define I0_OFFSET 0x28
- #define I1_OFFSET 0x2C
- #define I2_OFFSET 0x30
- #define I3_OFFSET 0x34
- #define I4_OFFSET 0x38
- #define I5_OFFSET 0x3C
- #define I6_FP_OFFSET 0x40
- #define I7_OFFSET 0x44
- #define O6_SP_OFFSET 0x48
- #define O7_OFFSET 0x4C
- #define PSR_OFFSET 0x50
- #define ISR_DISPATCH_DISABLE_STACK_OFFSET 0x54
- #define **SPARC_CONTEXT_CONTROL_IS_EXECUTING_OFFSET** 0x58
- #define FO_F1_OFFSET 0x00
- #define F2_F3_OFFSET 0x08
- #define F4_F5_OFFSET 0x10
- #define F6_F7_OFFSET 0x18
- #define F8_F9_OFFSET 0x20
- #define F1O_F11_OFFSET 0x28
- #define F12_F13_OFFSET 0x30
- #define F14_F15_OFFSET 0x38
- #define F16_F17_OFFSET 0x40
- #define F18_F19_OFFSET 0x48
- #define F2O_F21_OFFSET 0x50
- #define F22_F23_OFFSET 0x58
- #define F24_F25_OFFSET 0x60
- #define F26_F27_OFFSET 0x68
- #define F28_F29_OFFSET 0x70
- #define F3O_F31_OFFSET 0x78
- #define FSR_OFFSET 0x80
- #define CONTEXT_CONTROL_FP_SIZE 0x84
- #define CPU_CONTEXT_FP_SIZE sizeof( Context_Control_fp )
- #define CPU_MPCI_RECEIVE_SERVER_EXTRA_STACK 1024
- #define CPU_INTERRUPT_NUMBER_OF_VECTORS 256
- #define CPU_INTERRUPT_MAXIMUM_VECTOR_NUMBER 511
- #define CPU_PROVIDES_ISR_IS_IN_PROGRESS FALSE
- #define CPU_STACK_MINIMUM_SIZE (1024∗4)
- #define CPU_SIZEOF_POINTER 4
- #define CPU_ALIGNMENT 8
- #define CPU_HEAP_ALIGNMENT CPU_ALIGNMENT
- #define CPU_STACK_ALIGNMENT CPU_ALIGNMENT
- #define **CPU_INTERRUPT_STACK_ALIGNMENT** CPU_CACHE_LINE_BYTES
- #define _CPU_Initialize_vectors()
- #define _CPU_ISR_Disable(_level) (_level) = sparc_disable_interrupts()
- #define _CPU_ISR_Enable(_level) sparc_enable_interrupts( _level )

- #define _CPU_ISR_Flash(_level) sparc_flash_interrupts( _level )
- #define **_CPU_ISR_Is_enabled**(_isr_cookie) sparc_interrupt_is_enabled( _isr_cookie )
- #define _CPU_ISR_Set_level(_newlevel) sparc_enable_interrupts( _newlevel << 8)
- #define _CPU_Context_Initialization_at_thread_begin()
- #define _CPU_Context_Restart_self(_the_context) _CPU_Context_restore( (_the_context) );
- #define _CPU_Context_Initialize_fp(_destination) do { } while ( 0 )

  *Nothing to do due to the synchronous or lazy floating point switch.*
- #define _CPU_Context_save_fp(_fp_context_ptr) do { } while ( 0 )

  *Nothing to do due to the synchronous or lazy floating point switch.*
- #define _CPU_Context_restore_fp(_fp_context_ptr) do { } while ( 0 )

  *Nothing to do due to the synchronous or lazy floating point switch.*
- #define **CPU_USE_LIBC_INIT_FINI_ARRAY** FALSE
- #define CPU_USE_GENERIC_BITFIELD_CODE TRUE
- #define CPU_swap_u16(value) (((value&0xff) << 8) | ((value >> 8)&0xff))

  *SPARC specific method to endian swap an uint16_t.*

## Typedefs

- typedef struct Context_Control_fp **Context_Control_fp**
- typedef void(* **CPU_ISR_raw_handler**) (void)
- typedef void(* **CPU_ISR_handler**) (uint32_t)
- typedef uint32_t **CPU_Counter_ticks**
- typedef CPU_Counter_ticks(* **SPARC_Counter_read**) (void)
- typedef uintptr_t CPU_Uint32ptr

## Functions

- static bool **_CPU_Context_Get_is_executing** (const Context_Control *context)
- static void **_CPU_Context_Set_is_executing** (Context_Control *context, bool is_executing)
- static __inline__ bool **_CPU_ISR_Is_enabled** (uint32_t level)
- uint32_t _CPU_ISR_Get_level (void)

  *Obtain the current interrupt disable level.*
- void _CPU_Context_Initialize (Context_Control *the_context, uint32_t *stack_base, uint32_t size, uint32_t new_level, void *entry_point, bool is_fp, void *tls_area)
- RTEMS_NO_RETURN void _CPU_Fatal_halt (uint32_t source, uint32_t error)
- void _CPU_Initialize (void)

  *SPARC specific initialization.*
- void _CPU_ISR_install_raw_handler (uint32_t vector, CPU_ISR_raw_handler new_handler, CPU_ISR_raw↩ _handler *old_handler)

  *SPARC specific raw ISR installer.*
- void _CPU_ISR_install_vector (uint32_t vector, CPU_ISR_handler new_handler, CPU_ISR_handler *old_↩ handler)

  *SPARC specific RTEMS ISR installer.*
- void * **_CPU_Thread_Idle_body** (uintptr_t ignored)
- void _CPU_Context_switch (Context_Control *run, Context_Control *heir)

  *SPARC specific context switch.*
- RTEMS_NO_RETURN void _CPU_Context_restore (Context_Control *new_context)

  *SPARC specific context restore.*
- uint32_t **_CPU_SMP_Initialize** (void)
- bool **_CPU_SMP_Start_processor** (uint32_t cpu_index)
- void **_CPU_SMP_Finalize_initialization** (uint32_t cpu_count)

- void **_CPU_SMP_Prepare_start_multitasking** (void)
- uint32_t **_CPU_SMP_Get_current_processor** (void)
- void **_CPU_SMP_Send_interrupt** (uint32_t target_processor_index)
- static void **_CPU_SMP_Processor_event_broadcast** (void)
- static void **_CPU_SMP_Processor_event_receive** (void)
- void **_CPU_Exception_frame_print** (const CPU_Exception_frame *frame)
- static uint32_t CPU_swap_u32 (uint32_t value)

  *SPARC specific method to endian swap an uint32_t.*
- uint32_t **_CPU_Counter_frequency** (void)
- static CPU_Counter_ticks **_CPU_Counter_read** (void)
- static CPU_Counter_ticks **_CPU_Counter_difference** (CPU_Counter_ticks second, CPU_Counter_ticks first)

## Variables

- const CPU_Trap_table_entry _CPU_Trap_slot_template
- const SPARC_Counter **_SPARC_Counter**

### 10.323.1 Detailed Description

SPARC CPU Department Source.

This include file contains information pertaining to the port of the executive to the SPARC processor.

### 10.323.2 Macro Definition Documentation

#### 10.323.2.1 _CPU_Context_Initialization_at_thread_begin

```
#define _CPU_Context_Initialization_at_thread_begin( )
```

**Value:**

```
do { \
  __asm__ volatile ("set _Thread_Handler,%%i7\n"::); \
} while (0)
```

This macro is invoked from _Thread_Handler to do whatever CPU specific magic is required that must be done in the context of the thread when it starts.

On the SPARC, this is setting the frame pointer so GDB is happy. Make GDB stop unwinding at _Thread_Handler, previous register window Frame pointer is 0 and calling address must be a function with starting with a SAVE instruction. If return address is leaf-function (no SAVE) GDB will not look at prev reg window fp.

_Thread_Handler is known to start with SAVE.

Definition at line 852 of file cpu.h.

#### 10.323.2.2 _CPU_Context_Restart_self

```
#define _CPU_Context_Restart_self(
              _the_context )  _CPU_Context_restore( (_the_context) );
```

This routine is responsible for somehow restarting the currently executing task.

On the SPARC, this is is relatively painless but requires a small amount of wrapper code before using the regular restore code in of the context switch.

Definition at line 865 of file cpu.h.

#### 10.323.2.3 _CPU_Initialize_vectors

```
#define _CPU_Initialize_vectors( )
```

Support routine to initialize the RTEMS vector table after it is allocated.

Definition at line 753 of file cpu.h.

#### 10.323.2.4 _CPU_ISR_Disable

```
#define _CPU_ISR_Disable(
              _level )  (_level) = sparc_disable_interrupts()
```

Disable all interrupts for a critical section. The previous level is returned in _level.

Definition at line 759 of file cpu.h.

#### 10.323.2.5 _CPU_ISR_Enable

```
#define _CPU_ISR_Enable(
              _level )  sparc_enable_interrupts( _level )
```

Enable interrupts to the previous level (returned by _CPU_ISR_Disable). This indicates the end of a critical section. The parameter _level is not modified.

Definition at line 767 of file cpu.h.

#### 10.323.2.6 _CPU_ISR_Flash

```
#define _CPU_ISR_Flash(
              _level )  sparc_flash_interrupts( _level )
```

This temporarily restores the interrupt to _level before immediately disabling them again. This is used to divide long critical sections into two or more parts. The parameter _level is not modified.

Definition at line 776 of file cpu.h.

#### 10.323.2.7 _CPU_ISR_Set_level

```
#define _CPU_ISR_Set_level(
              _newlevel )  sparc_enable_interrupts( _newlevel << 8)
```

Map interrupt level in task mode onto the hardware that the CPU actually provides. Currently, interrupt levels which do not map onto the CPU in a straight fashion are undefined.

Definition at line 792 of file cpu.h.

#### 10.323.2.8 CPU_ALIGNMENT

```
#define CPU_ALIGNMENT 8
```

CPU's worst alignment requirement for data types on a byte boundary. This alignment does not take into account the requirements for the stack.

On the SPARC, this is required for double word loads and stores.

Definition at line 721 of file cpu.h.

#### 10.323.2.9 CPU_ALL_TASKS_ARE_FP

```
#define CPU_ALL_TASKS_ARE_FP FALSE
```

Are all tasks FLOATING_POINT tasks implicitly?

- If TRUE, then the FLOATING_POINT task attribute is assumed.
- If FALSE, then the FLOATING_POINT task attribute is followed.

The SPARC GCC port does not implicitly use floating point registers.

Definition at line 121 of file cpu.h.

#### 10.323.2.10 CPU_CONTEXT_FP_SIZE

```
#define CPU_CONTEXT_FP_SIZE sizeof( Context_Control_fp )
```

The size of the floating point context area.

Definition at line 652 of file cpu.h.

#### 10.323.2.11 CPU_HARDWARE_FP

```
#define CPU_HARDWARE_FP FALSE
```

Does the CPU have hardware floating point?

- If TRUE, then the FLOATING_POINT task attribute is supported.
- If FALSE, then the FLOATING_POINT task attribute is ignored.

This is set based upon the multilib settings.

Definition at line 104 of file cpu.h.

#### 10.323.2.12 CPU_HEAP_ALIGNMENT

```
#define CPU_HEAP_ALIGNMENT CPU_ALIGNMENT
```

This number corresponds to the byte alignment requirement for the heap handler. This alignment requirement may be stricter than that for the data types alignment specified by CPU_ALIGNMENT. It is common for the heap to follow the same alignment requirement as CPU_ALIGNMENT. If the CPU_ALIGNMENT is strict enough for the heap, then this should be set to CPU_ALIGNMENT.

NOTE: This does not have to be a power of 2. It does have to be greater or equal to than CPU_ALIGNMENT.

Definition at line 734 of file cpu.h.

#### 10.323.2.13 CPU_IDLE_TASK_IS_FP

```
#define CPU_IDLE_TASK_IS_FP FALSE
```

Should the IDLE task have a floating point context?

- If TRUE, then the IDLE task is created as a FLOATING_POINT task and it has a floating point context which is switched in and out.
- If FALSE, then the IDLE task does not have a floating point context.

The IDLE task does not have to be floating point on the SPARC.

Definition at line 132 of file cpu.h.

#### 10.323.2.14 CPU_INTERRUPT_MAXIMUM_VECTOR_NUMBER

```
#define CPU_INTERRUPT_MAXIMUM_VECTOR_NUMBER 511
```

The SPARC has 256 vectors but the port treats 256-512 as synchronous traps.

Definition at line 692 of file cpu.h.

#### 10.323.2.15 CPU_INTERRUPT_NUMBER_OF_VECTORS

```
#define CPU_INTERRUPT_NUMBER_OF_VECTORS 256
```

This defines the number of entries in the ISR_Vector_table managed by the executive.

On the SPARC, there are really only 256 vectors. However, the executive has no easy, fast, reliable way to determine which traps are synchronous and which are asynchronous. By default, synchronous traps return to the instruction which caused the interrupt. So if you install a software trap handler as an executive interrupt handler (which is desirable since RTEMS takes care of window and register issues), then the executive needs to know that the return address is to the trap rather than the instruction following the trap.

So vectors 0 through 255 are treated as regular asynchronous traps which provide the "correct" return address. Vectors 256 through 512 are assumed by the executive to be synchronous and to require that the return address be fudged.

If you use this mechanism to install a trap handler which must reexecute the instruction which caused the trap, then it should be installed as an asynchronous trap. This will avoid the executive changing the return address.

Definition at line 686 of file cpu.h.

#### 10.323.2.16 CPU_ISR_PASSES_FRAME_POINTER

```
#define CPU_ISR_PASSES_FRAME_POINTER FALSE
```

Does the RTEMS invoke the user's ISR with the vector number and a pointer to the saved interrupt frame (1) or just the vector number (0)?

The SPARC port does not pass an Interrupt Stack Frame pointer to interrupt handlers.

Definition at line 91 of file cpu.h.

#### 10.323.2.17 CPU_MODES_INTERRUPT_MASK

```
#define CPU_MODES_INTERRUPT_MASK 0x0000000F
```

The following defines the number of bits actually used in the interrupt field of the task mode. How those bits map to the CPU interrupt levels is defined by the routine _CPU_ISR_Set_level().

The SPARC has 16 interrupt levels in the PIL field of the PSR.

Definition at line 161 of file cpu.h.

#### 10.323.2.18 CPU_MPCI_RECEIVE_SERVER_EXTRA_STACK

```
#define CPU_MPCI_RECEIVE_SERVER_EXTRA_STACK 1024
```

Amount of extra stack (above minimum stack size) required by MPCI receive server thread. Remember that in a multiprocessor system this thread must exist and be able to process all directives.

Definition at line 661 of file cpu.h.

#### 10.323.2.19 CPU_PROVIDES_ISR_IS_IN_PROGRESS

```
#define CPU_PROVIDES_ISR_IS_IN_PROGRESS FALSE
```

This is defined if the port has a special way to report the ISR nesting level. Most ports maintain the variable _ISR_Nest_level.

Definition at line 698 of file cpu.h.

#### 10.323.2.20 CPU_SIMPLE_VECTORED_INTERRUPTS

```
#define CPU_SIMPLE_VECTORED_INTERRUPTS TRUE
```

Does the CPU follow the simple vectored interrupt model?

- If TRUE, then RTEMS allocates the vector table it internally manages.
- If FALSE, then the BSP is assumed to allocate and manage the vector table

THe SPARC is a simple vectored architecture. Usually there is no PIC and the CPU directly vectors the interrupts.

Definition at line 81 of file cpu.h.

#### 10.323.2.21 CPU_SIZEOF_POINTER

```
#define CPU_SIZEOF_POINTER 4
```

What is the size of a pointer on this architecture?

Definition at line 713 of file cpu.h.

#### 10.323.2.22 CPU_SOFTWARE_FP

```
#define CPU_SOFTWARE_FP FALSE
```

The SPARC GCC port does not have a software floating point library that requires RTEMS assistance.

Definition at line 111 of file cpu.h.

#### 10.323.2.23 CPU_STACK_ALIGNMENT

```
#define CPU_STACK_ALIGNMENT CPU_ALIGNMENT
```

Stack frames must be doubleword aligned according to the System V ABI for SPARC.

Definition at line 740 of file cpu.h.

#### 10.323.2.24 CPU_STACK_FRAME_I0_OFFSET

```
#define CPU_STACK_FRAME_I0_OFFSET 0x20
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 245 of file cpu.h.

#### 10.323.2.25 CPU_STACK_FRAME_I1_OFFSET

```
#define CPU_STACK_FRAME_I1_OFFSET 0x24
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 247 of file cpu.h.

#### 10.323.2.26 CPU_STACK_FRAME_I2_OFFSET

```
#define CPU_STACK_FRAME_I2_OFFSET 0x28
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 249 of file cpu.h.

#### 10.323.2.27 CPU_STACK_FRAME_I3_OFFSET

```
#define CPU_STACK_FRAME_I3_OFFSET 0x2c
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 251 of file cpu.h.

#### 10.323.2.28 CPU_STACK_FRAME_I4_OFFSET

```
#define CPU_STACK_FRAME_I4_OFFSET 0x30
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 253 of file cpu.h.

#### 10.323.2.29 CPU_STACK_FRAME_I5_OFFSET

```
#define CPU_STACK_FRAME_I5_OFFSET 0x34
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 255 of file cpu.h.

#### 10.323.2.30 CPU_STACK_FRAME_I6_FP_OFFSET

```
#define CPU_STACK_FRAME_I6_FP_OFFSET 0x38
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 257 of file cpu.h.

#### 10.323.2.31 CPU_STACK_FRAME_I7_OFFSET

```
#define CPU_STACK_FRAME_I7_OFFSET 0x3c
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 259 of file cpu.h.

#### 10.323.2.32 CPU_STACK_FRAME_L0_OFFSET

```
#define CPU_STACK_FRAME_L0_OFFSET 0x00
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 229 of file cpu.h.

#### 10.323.2.33 CPU_STACK_FRAME_L1_OFFSET

```
#define CPU_STACK_FRAME_L1_OFFSET 0x04
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 231 of file cpu.h.

#### 10.323.2.34 CPU_STACK_FRAME_L2_OFFSET

```
#define CPU_STACK_FRAME_L2_OFFSET 0x08
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 233 of file cpu.h.

#### 10.323.2.35 CPU_STACK_FRAME_L3_OFFSET

```
#define CPU_STACK_FRAME_L3_OFFSET 0x0c
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 235 of file cpu.h.

#### 10.323.2.36 CPU_STACK_FRAME_L4_OFFSET

```
#define CPU_STACK_FRAME_L4_OFFSET 0x10
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 237 of file cpu.h.

#### 10.323.2.37 CPU_STACK_FRAME_L5_OFFSET

```
#define CPU_STACK_FRAME_L5_OFFSET 0x14
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 239 of file cpu.h.

#### 10.323.2.38 CPU_STACK_FRAME_L6_OFFSET

```
#define CPU_STACK_FRAME_L6_OFFSET 0x18
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 241 of file cpu.h.

#### 10.323.2.39 CPU_STACK_FRAME_L7_OFFSET

```
#define CPU_STACK_FRAME_L7_OFFSET 0x1c
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 243 of file cpu.h.

#### 10.323.2.40 CPU_STACK_FRAME_PAD0_OFFSET

```
#define CPU_STACK_FRAME_PAD0_OFFSET 0x5c
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 275 of file cpu.h.

#### 10.323.2.41 CPU_STACK_FRAME_SAVED_ARG0_OFFSET

```
#define CPU_STACK_FRAME_SAVED_ARG0_OFFSET 0x44
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 263 of file cpu.h.

#### 10.323.2.42 CPU_STACK_FRAME_SAVED_ARG1_OFFSET

```
#define CPU_STACK_FRAME_SAVED_ARG1_OFFSET 0x48
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 265 of file cpu.h.

#### 10.323.2.43 CPU_STACK_FRAME_SAVED_ARG2_OFFSET

```
#define CPU_STACK_FRAME_SAVED_ARG2_OFFSET 0x4c
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 267 of file cpu.h.

#### 10.323.2.44 CPU_STACK_FRAME_SAVED_ARG3_OFFSET

```
#define CPU_STACK_FRAME_SAVED_ARG3_OFFSET 0x50
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 269 of file cpu.h.

#### 10.323.2.45 CPU_STACK_FRAME_SAVED_ARG4_OFFSET

```
#define CPU_STACK_FRAME_SAVED_ARG4_OFFSET 0x54
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 271 of file cpu.h.

#### 10.323.2.46 CPU_STACK_FRAME_SAVED_ARG5_OFFSET

```
#define CPU_STACK_FRAME_SAVED_ARG5_OFFSET 0x58
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 273 of file cpu.h.

#### 10.323.2.47 CPU_STACK_GROWS_UP

```
#define CPU_STACK_GROWS_UP FALSE
```

Does the stack grow up (toward higher addresses) or down (toward lower addresses)?

- If TRUE, then the grows upward.
- If FALSE, then the grows toward smaller addresses.

The stack grows to lower addresses on the SPARC.

Definition at line 147 of file cpu.h.

#### 10.323.2.48 CPU_STACK_MINIMUM_SIZE

```
#define CPU_STACK_MINIMUM_SIZE (1024*4)
```

Should be large enough to run all tests. This ensures that a "reasonable" small application should not have any problems.

This appears to be a fairly generous number for the SPARC since represents a call depth of about 20 routines based on the minimum stack frame.

Definition at line 708 of file cpu.h.

#### 10.323.2.49 CPU_STRUCTURE_RETURN_ADDRESS_OFFSET

```
#define CPU_STRUCTURE_RETURN_ADDRESS_OFFSET 0x40
```

This macro defines an offset into the stack frame for use in assembly.

Definition at line 261 of file cpu.h.

#### 10.323.2.50 CPU_swap_u16

```
#define CPU_swap_u16(
              value )  (((value&0xff) << 8) | ((value >> 8)&0xff))
```

SPARC specific method to endian swap an uint16_t.

The following routine swaps the endian format of a uint16_t.

**Parameters**

| | | |
|---|---|---|
| in | *value* | is the value to endian swap |

Definition at line 1075 of file cpu.h.

#### 10.323.2.51 CPU_USE_GENERIC_BITFIELD_CODE

```
#define CPU_USE_GENERIC_BITFIELD_CODE TRUE
```

The SPARC port uses the generic C algorithm for bitfield scan if the CPU model does not have a scan instruction.

Definition at line 907 of file cpu.h.

### 10.323.3 Typedef Documentation

#### 10.323.3.1 CPU_Uint32ptr

```
typedef uintptr_t CPU_Uint32ptr
```

Type that can store a 32-bit integer or a pointer.

Definition at line 1117 of file cpu.h.

### 10.323.4 Function Documentation

#### 10.323.4.1 _CPU_Context_Initialize()

```
void _CPU_Context_Initialize (
             Context_Control * the_context,
             uint32_t * stack_base,
             uint32_t size,
             uint32_t new_level,
             void * entry_point,
             bool is_fp,
             void * tls_area )
```

Initialize the context to a state suitable for starting a task after a context restore operation. Generally, this involves:

- setting a starting address
- preparing the stack
- preparing the stack and frame pointers
- setting the proper interrupt level in the context
- initializing the floating point context

**Parameters**

| | | |
|---|---|---|
| in | *the_context* | points to the context area |
| in | *stack_base* | is the low address of the allocated stack area |
| in | *size* | is the size of the stack area in bytes |
| in | *new_level* | is the interrupt level for the task |
| in | *entry_point* | is the task's entry point |
| in | *is_fp* | is set to TRUE if the task is a floating point task |
| in | *tls_area* | is the thread-local storage (TLS) area |

NOTE: Implemented as a subroutine for the SPARC port.

Definition at line 354 of file cpu.c.

### 10.323.4.2 _CPU_Context_restore()

```
RTEMS_NO_RETURN void _CPU_Context_restore (
            Context_Control * new_context )
```

SPARC specific context restore.

This routine is generally used only to restart self in an efficient manner.

**Parameters**

| | | |
|---|---|---|
| in | *new_context* | is the context to restore |

### 10.323.4.3 _CPU_Context_switch()

```
void _CPU_Context_switch (
            Context_Control * run,
            Context_Control * heir )
```

SPARC specific context switch.

This routine switches from the run context to the heir context.

**Parameters**

| | | |
|---|---|---|
| in | *run* | is the currently executing thread |
| in | *heir* | will become the currently executing thread |

#### 10.323.4.4 _CPU_Fatal_halt()

```
RTEMS_NO_RETURN void _CPU_Fatal_halt (
             uint32_t source,
             uint32_t error )
```

This routine copies _error into a known place – typically a stack location or a register, optionally disables interrupts, and halts/stops the CPU.

Definition at line 31 of file bsp_fatal_halt.c.

#### 10.323.4.5 _CPU_Initialize()

```
void _CPU_Initialize (
             void  )
```

SPARC specific initialization.

This routine performs CPU dependent initialization.

Definition at line 176 of file cpu.c.

#### 10.323.4.6 _CPU_ISR_Get_level()

```
uint32_t _CPU_ISR_Get_level (
             void  )
```

Obtain the current interrupt disable level.

This method is invoked to return the current interrupt disable level.

**Returns**

This method returns the current interrupt disable level.

Definition at line 191 of file cpu.c.

#### 10.323.4.7 _CPU_ISR_install_raw_handler()

```
void _CPU_ISR_install_raw_handler (
             uint32_t vector,
             CPU_ISR_raw_handler new_handler,
             CPU_ISR_raw_handler * old_handler )
```

SPARC specific raw ISR installer.

This routine installs *new_handler* to be directly called from the trap table.

**Parameters**

| | | |
|---|---|---|
| in | *vector* | is the vector number |
| in | *new_handler* | is the new ISR handler |
| in | *old_handler* | will contain the old ISR handler |

Definition at line 237 of file cpu.c.

#### 10.323.4.8 _CPU_ISR_install_vector()

```
void _CPU_ISR_install_vector (
             uint32_t vector,
             CPU_ISR_handler new_handler,
             CPU_ISR_handler * old_handler )
```

SPARC specific RTEMS ISR installer.

This routine installs an interrupt vector.

**Parameters**

| | | |
|---|---|---|
| in | *vector* | is the vector number |
| in | *new_handler* | is the new ISR handler |
| in | *old_handler* | will contain the old ISR handler |

Definition at line 318 of file cpu.c.

#### 10.323.4.9 CPU_swap_u32()

```
static uint32_t CPU_swap_u32 (
             uint32_t value )  [inline], [static]
```

SPARC specific method to endian swap an uint32_t.

The following routine swaps the endian format of an unsigned int. It must be static because it is referenced indirectly.

**Parameters**

| | | |
|---|---|---|
| in | *value* | is the value to endian swap |

This version will work on any processor, but if you come across a better way for the SPARC PLEASE use it. The most common way to swap a 32-bit entity as shown below is not any more efficient on the SPARC.

- swap least significant two bytes with 16-bit rotate

- swap upper and lower 16-bits
- swap most significant two bytes with 16-bit rotate

It is not obvious how the SPARC can do significantly better than the generic code. gcc 2.7.0 only generates about 12 instructions for the following code at optimization level four (i.e. -O4).

Definition at line 1053 of file cpu.h.

### 10.323.5 Variable Documentation

#### 10.323.5.1 _CPU_Trap_slot_template

```
const CPU_Trap_table_entry _CPU_Trap_slot_template  [extern]
```

This is the set of opcodes for the instructions loaded into a trap table entry. The routine which installs a handler is responsible for filling in the fields for the _handler address and the _vector trap type.

The constants following this structure are masks for the fields which must be filled in when the handler is installed.

Definition at line 156 of file cpu.c.

## 10.324 cpukit/score/cpu/sparc/include/rtems/score/cpuimpl.h File Reference

CPU Port Implementation API.

```
#include <rtems/score/cpu.h>
```

### Classes

- struct CPU_Per_CPU_control

## Macros

- #define SPARC_MINIMUM_STACK_FRAME_SIZE 0x60
- #define ISF_PSR_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x00
- #define ISF_PC_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x04
- #define ISF_NPC_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x08
- #define ISF_G1_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x0c
- #define ISF_G2_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x10
- #define ISF_G3_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x14
- #define ISF_G4_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x18
- #define ISF_G5_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x1c
- #define ISF_G7_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x24
- #define ISF_I0_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x28
- #define ISF_I1_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x2c
- #define ISF_I2_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x30
- #define ISF_I3_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x34
- #define ISF_I4_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x38
- #define ISF_I5_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x3c
- #define ISF_I6_FP_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x40
- #define ISF_I7_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x44
- #define ISF_Y_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x48
- #define ISF_TPC_OFFSET SPARC_MINIMUM_STACK_FRAME_SIZE + 0x4c
- #define CPU_INTERRUPT_FRAME_SIZE SPARC_MINIMUM_STACK_FRAME_SIZE + 0x50
- #define **SPARC_FP_CONTEXT_OFFSET_F0_F1** 0
- #define **SPARC_FP_CONTEXT_OFFSET_F2_F3** 8
- #define **SPARC_FP_CONTEXT_OFFSET_F4_F5** 16
- #define **SPARC_FP_CONTEXT_OFFSET_F6_F7** 24
- #define **SPARC_FP_CONTEXT_OFFSET_F8_F9** 32
- #define **SPARC_FP_CONTEXT_OFFSET_F10_F11** 40
- #define **SPARC_FP_CONTEXT_OFFSET_F12_F13** 48
- #define **SPARC_FP_CONTEXT_OFFSET_F14_F15** 56
- #define **SPARC_FP_CONTEXT_OFFSET_F16_F17** 64
- #define **SPARC_FP_CONTEXT_OFFSET_F18_F19** 72
- #define **SPARC_FP_CONTEXT_OFFSET_F20_F21** 80
- #define **SPARC_FP_CONTEXT_OFFSET_F22_F23** 88
- #define **SPARC_FP_CONTEXT_OFFSET_F24_F25** 96
- #define **SPARC_FP_CONTEXT_OFFSET_F26_F27** 104
- #define **SPARC_FP_CONTEXT_OFFSET_F28_F29** 112
- #define **SPARC_FP_CONTEXT_OFFSET_F30_F31** 120
- #define **SPARC_FP_CONTEXT_OFFSET_FSR** 128
- #define **CPU_PER_CPU_CONTROL_SIZE** 0
- #define **_CPU_Get_current_per_CPU_control**() _SPARC_Per_CPU_current
- #define **_CPU_Get_thread_executing**() ( _SPARC_Per_CPU_current->executing )

## Functions

- void **_CPU_Context_volatile_clobber** (uintptr_t pattern)
- void **_CPU_Context_validate** (uintptr_t pattern)
- RTEMS_INLINE_ROUTINE void **_CPU_Instruction_illegal** (void)
- RTEMS_INLINE_ROUTINE void **_CPU_Instruction_no_operation** (void)

### Variables

- register struct Per_CPU_Control * _SPARC_Per_CPU_current

  *The pointer to the current per-CPU control is available via register g6.*

### 10.324.1 Detailed Description

CPU Port Implementation API.

## 10.325 cpukit/score/cpu/sparc/include/rtems/score/sparc.h File Reference

Information Required to Build RTEMS for a Particular Member of the SPARC Family.

```
#include <rtems/score/basedefs.h>
```

### Macros

- #define SPARC_HAS_BITSCAN 0
- #define SPARC_NUMBER_OF_REGISTER_WINDOWS 8
- #define SPARC_LEON3FT_B2BST_NOP
- #define SPARC_HAS_FPU 1
- #define CPU_MODEL_NAME "w/FPU"
- #define CPU_NAME "SPARC"
- #define SPARC_PSR_CWP_MASK 0x07 /* bits 0 - 4 */
- #define SPARC_PSR_ET_MASK 0x00000020 /* bit 5 */
- #define SPARC_PSR_PS_MASK 0x00000040 /* bit 6 */
- #define SPARC_PSR_S_MASK 0x00000080 /* bit 7 */
- #define SPARC_PSR_PIL_MASK 0x00000F00 /* bits 8 - 11 */
- #define SPARC_PSR_EF_MASK 0x00001000 /* bit 12 */
- #define SPARC_PSR_EC_MASK 0x00002000 /* bit 13 */
- #define SPARC_PSR_ICC_MASK 0x00F00000 /* bits 20 - 23 */
- #define SPARC_PSR_VER_MASK 0x0F000000 /* bits 24 - 27 */
- #define SPARC_PSR_IMPL_MASK 0xF0000000 /* bits 28 - 31 */
- #define SPARC_PSR_CWP_BIT_POSITION 0 /* bits 0 - 4 */
- #define SPARC_PSR_ET_BIT_POSITION 5 /* bit 5 */
- #define SPARC_PSR_PS_BIT_POSITION 6 /* bit 6 */
- #define SPARC_PSR_S_BIT_POSITION 7 /* bit 7 */
- #define SPARC_PSR_PIL_BIT_POSITION 8 /* bits 8 - 11 */
- #define SPARC_PSR_EF_BIT_POSITION 12 /* bit 12 */
- #define SPARC_PSR_EC_BIT_POSITION 13 /* bit 13 */
- #define SPARC_PSR_ICC_BIT_POSITION 20 /* bits 20 - 23 */
- #define SPARC_PSR_VER_BIT_POSITION 24 /* bits 24 - 27 */
- #define SPARC_PSR_IMPL_BIT_POSITION 28 /* bits 28 - 31 */
- #define **LEON3_ASR17_PROCESSOR_INDEX_SHIFT** 28
- #define **SPARC_SWTRAP_SYSCALL** 0
- #define **SPARC_SWTRAP_IRQDIS** 9
- #define **SPARC_SWTRAP_IRQEN** 10
- #define **SPARC_SWTRAP_IRQDIS_FP** 11

- #define SPARC_SYNCHRONOUS_TRAP_BIT_MASK 0x100

  *This is the bit step in a vector number to indicate it is being installed as a synchronous trap.*
- #define SPARC_ASYNCHRONOUS_TRAP(_vector) ( _vector )

  *Maps the real hardware vector number to the associated asynchronous trap number.*
- #define SPARC_SYNCHRONOUS_TRAP(_vector) ( ( _vector ) + 256 )

  *Maps the real hardware vector number to the associated synchronous trap number.*
- #define SPARC_REAL_TRAP_NUMBER(_trap) ( ( _trap ) % 256 )

  *Maps the synchronous or asynchronous trap number to the associated real hardware vector number.*
- #define SPARC_IS_INTERRUPT_TRAP(_trap)

  *Checks if the real hardware vector number, synchronous trap number, or asynchronous trap number is an interrupt trap.*
- #define SPARC_INTERRUPT_TRAP_TO_SOURCE(_trap) ( SPARC_REAL_TRAP_NUMBER( _trap ) - 0x10 )

  *Maps the interrupt trap number to the associated interrupt source number.*
- #define SPARC_INTERRUPT_SOURCE_TO_TRAP(_source) ( SPARC_ASYNCHRONOUS_TRAP( _source ) + 0x10 )

  *Maps the interrupt source number to the associated asynchronous trap number.*
- #define nop()
- #define sparc_get_psr(_psr)

  *Macro to obtain the PSR.*
- #define sparc_set_psr(_psr)

  *Macro to set the PSR.*
- #define sparc_get_tbr(_tbr)

  *Macro to obtain the TBR.*
- #define sparc_set_tbr(_tbr)

  *Macro to set the TBR.*
- #define sparc_get_wim(_wim)

  *Macro to obtain the WIM.*
- #define sparc_set_wim(_wim)

  *Macro to set the WIM.*
- #define sparc_get_y(_y)

  *Macro to obtain the Y register.*
- #define sparc_set_y(_y)

  *Macro to set the Y register.*
- #define sparc_flash_interrupts(_psr)

  *SPARC flash processor interrupts.*
- #define sparc_get_interrupt_level(_level)

  *SPARC obtain interrupt level.*

## Functions

- static uint32_t sparc_disable_interrupts (void)

  *SPARC disable processor interrupts.*
- static void sparc_enable_interrupts (uint32_t psr)

  *SPARC enable processor interrupts.*
- RTEMS_NO_RETURN void sparc_syscall_exit (uint32_t exitcode1, uint32_t exitcode2)

  *SPARC exit through system call 1.*
- static uint32_t **_LEON3_Get_current_processor** (void)

### 10.325.1 Detailed Description

Information Required to Build RTEMS for a Particular Member of the SPARC Family.

This file contains the information required to build RTEMS for a particular member of the SPARC family. It does this by setting variables to indicate which implementation dependent features are present in a particular member of the family.

### 10.325.2 Macro Definition Documentation

#### 10.325.2.1 CPU_MODEL_NAME

```
#define CPU_MODEL_NAME "w/FPU"
```

This macro contains a string describing the multilib variant being build.

Definition at line 91 of file sparc.h.

#### 10.325.2.2 CPU_NAME

```
#define CPU_NAME "SPARC"
```

Define the name of the CPU family.

Definition at line 99 of file sparc.h.

#### 10.325.2.3 nop

```
#define nop( )
```

**Value:**

```
do { \
  __asm__ volatile ( "nop" ); \
} while ( 0 )
```

This macro is a standard nop instruction.

Definition at line 260 of file sparc.h.

#### 10.325.2.4 SPARC_ASYNCHRONOUS_TRAP

```
#define SPARC_ASYNCHRONOUS_TRAP(
              _vector ) ( _vector )
```

Maps the real hardware vector number to the associated asynchronous trap number.

**Parameters**

| | |
|---|---|
| *_vector* | is the real hardware vector number to map. |

**Returns**

Returns the asynchronous trap number associated with the real hardware vector number.

Definition at line 185 of file sparc.h.

### 10.325.2.5 sparc_flash_interrupts

```
#define sparc_flash_interrupts(
              _psr )
```

**Value:**

```
do { \
  sparc_enable_interrupts( (_psr) ); \
  _psr = sparc_disable_interrupts(); \
} while ( 0 )
```

SPARC flash processor interrupts.

This method is invoked to temporarily enable all maskable interrupts.

**Parameters**

| | | |
|---|---|---|
| in | *_psr* | is the PSR returned by sparc_disable_interrupts. |

Definition at line 465 of file sparc.h.

### 10.325.2.6 sparc_get_interrupt_level

```
#define sparc_get_interrupt_level(
              _level )
```

**Value:**

```
do { \
  uint32_t _psr_level = 0; \
  \
  sparc_get_psr( _psr_level ); \
  (_level) = \
    (_psr_level & SPARC_PSR_PIL_MASK) >> SPARC_PSR_PIL_BIT_POSITION; \
} while ( 0 )
```

SPARC obtain interrupt level.

This method is invoked to obtain the current interrupt disable level.

**Parameters**

| | | |
|---|---|---|
| in | *_level* | is the PSR returned by sparc_disable_interrupts. |

Definition at line 478 of file sparc.h.

#### 10.325.2.7 sparc_get_psr

```
#define sparc_get_psr(
              _psr )
```

**Value:**

```
  do { \
     (_psr) = 0; \
     __asm__ volatile( "rd %%psr, %0" :  "=r" (_psr) : "0" (_psr) ); \
  } while ( 0 )
```

Macro to obtain the PSR.

This macro returns the current contents of the PSR register in *_psr*.

Definition at line 279 of file sparc.h.

#### 10.325.2.8 sparc_get_tbr

```
#define sparc_get_tbr(
              _tbr )
```

**Value:**

```
  do { \
     (_tbr) = 0; /* to avoid unitialized warnings */ \
     __asm__ volatile( "rd %%tbr, %0" :  "=r" (_tbr) : "0" (_tbr) ); \
  } while ( 0 )
```

Macro to obtain the TBR.

This macro returns the current contents of the TBR register in *_tbr*.

Definition at line 325 of file sparc.h.

#### 10.325.2.9 sparc_get_wim

```
#define sparc_get_wim(
              _wim )
```

**Value:**

```
  do { \
    __asm__ volatile( "rd %%wim, %0" :  "=r" (_wim) : "0" (_wim) ); \
  } while ( 0 )
```

Macro to obtain the WIM.

This macro returns the current contents of the WIM field in *_wim*.

Definition at line 359 of file sparc.h.

#### 10.325.2.10 sparc_get_y

```
#define sparc_get_y(
              _y )
```

**Value:**

```
  do { \
    __asm__ volatile( "rd %%y, %0" :  "=r" (_y) : "0" (_y) ); \
  } while ( 0 )
```

Macro to obtain the Y register.

This macro returns the current contents of the Y register in *_y*.

Definition at line 382 of file sparc.h.

#### 10.325.2.11 SPARC_HAS_BITSCAN

```
#define SPARC_HAS_BITSCAN 0
```

Some higher end SPARCs have a bitscan instructions. It would be nice to take advantage of them. Right now, there is no port to a CPU model with this feature and no (untested) code that is based on this feature flag.

Definition at line 55 of file sparc.h.

#### 10.325.2.12 SPARC_HAS_FPU

```
#define SPARC_HAS_FPU 1
```

This macro indicates whether this multilib variation has hardware floating point or not. We use the gcc cpp predefine _SOFT_FLOAT to determine that.

Definition at line 83 of file sparc.h.

#### 10.325.2.13 SPARC_INTERRUPT_SOURCE_TO_TRAP

```
#define SPARC_INTERRUPT_SOURCE_TO_TRAP(
              _source )   ( SPARC_ASYNCHRONOUS_TRAP( _source ) + 0x10 )
```

Maps the interrupt source number to the associated asynchronous trap number.

**Parameters**

| | |
|---|---|
| *_source* | is the interrupt source number to map. |

**Returns**

Returns the asynchronous trap number associated with the interrupt source number.

Definition at line 252 of file sparc.h.

### 10.325.2.14 SPARC_INTERRUPT_TRAP_TO_SOURCE

```
#define SPARC_INTERRUPT_TRAP_TO_SOURCE(
              _trap )   ( SPARC_REAL_TRAP_NUMBER( _trap ) - 0x10 )
```

Maps the interrupt trap number to the associated interrupt source number.

Interrupt trap numbers are real hardware vector numbers, synchronous trap numbers, and asynchronous trap numbers for which SPARC_IS_INTERRUPT_TRAP() returns true.

**Parameters**

| | |
|---|---|
| *_trap* | is the real hardware vector number, synchronous trap number, or asynchronous trap number to map. |

**Returns**

Returns the interrupt source number associated with the interrupt trap number.

Definition at line 240 of file sparc.h.

### 10.325.2.15 SPARC_IS_INTERRUPT_TRAP

```
#define SPARC_IS_INTERRUPT_TRAP(
              _trap )
```

**Value:**

```
( SPARC_REAL_TRAP_NUMBER( _trap ) >= 0x11 && \
  SPARC_REAL_TRAP_NUMBER( _trap ) <= 0x1f )
```

Checks if the real hardware vector number, synchronous trap number, or asynchronous trap number is an interrupt trap.

Interrupt traps are defined by Table 7-1 "Exception and Interrupt Request Priority and tt Values" in "The SPARC Architecture Manual: Version 8".

**Parameters**

| | |
|---|---|
| *_trap* | is the real hardware vector number, synchronous trap number, or asynchronous trap number to check. |

**Returns**

Returns true, if the real hardware vector number, synchronous trap number, or asynchronous trap number is an interrupt trap, otherwise false.

Definition at line 222 of file sparc.h.

#### 10.325.2.16 SPARC_LEON3FT_B2BST_NOP

```
#define SPARC_LEON3FT_B2BST_NOP
```

See GRLIB-TN-0009: "LEON3FT Stale Cache Entry After Store with Data Tag Parity Error"

Definition at line 72 of file sparc.h.

#### 10.325.2.17 SPARC_NUMBER_OF_REGISTER_WINDOWS

```
#define SPARC_NUMBER_OF_REGISTER_WINDOWS 8
```

This should be OK until a port to a higher end SPARC processor is made that has more than 8 register windows. If this cannot be determined based on multilib settings (v7/v8/v9), then the cpu_asm.S code that depends on this will have to move to libcpu.

Definition at line 63 of file sparc.h.

#### 10.325.2.18 SPARC_PSR_CWP_BIT_POSITION

```
#define SPARC_PSR_CWP_BIT_POSITION 0 /* bits 0 - 4 */
```

This constant is the starting bit position of the CWP in the PSR.

Definition at line 140 of file sparc.h.

#### 10.325.2.19 SPARC_PSR_CWP_MASK

```
#define SPARC_PSR_CWP_MASK 0x07 /* bits 0 - 4 */
```

PSR masks and starting bit positions

NOTE: Reserved bits are ignored.

Definition at line 111 of file sparc.h.

#### 10.325.2.20 SPARC_PSR_EC_BIT_POSITION

```
#define SPARC_PSR_EC_BIT_POSITION 13 /* bit 13 */
```

This constant is the starting bit position of the EC in the PSR.

Definition at line 152 of file sparc.h.

#### 10.325.2.21 SPARC_PSR_EC_MASK

```
#define SPARC_PSR_EC_MASK 0x00002000 /* bit 13 */
```

This constant is a mask for the EC bits in the PSR.

Definition at line 131 of file sparc.h.

#### 10.325.2.22 SPARC_PSR_EF_BIT_POSITION

```
#define SPARC_PSR_EF_BIT_POSITION 12 /* bit 12 */
```

This constant is the starting bit position of the EF in the PSR.

Definition at line 150 of file sparc.h.

#### 10.325.2.23 SPARC_PSR_EF_MASK

```
#define SPARC_PSR_EF_MASK 0x00001000 /* bit 12 */
```

This constant is a mask for the EF bits in the PSR.

Definition at line 129 of file sparc.h.

#### 10.325.2.24 SPARC_PSR_ET_BIT_POSITION

```
#define SPARC_PSR_ET_BIT_POSITION 5 /* bit 5 */
```

This constant is the starting bit position of the ET in the PSR.

Definition at line 142 of file sparc.h.

#### 10.325.2.25 SPARC_PSR_ET_MASK

```
#define SPARC_PSR_ET_MASK 0x00000020 /* bit 5 */
```

This constant is a mask for the ET bits in the PSR.

Definition at line 121 of file sparc.h.

#### 10.325.2.26 SPARC_PSR_ICC_BIT_POSITION

```
#define SPARC_PSR_ICC_BIT_POSITION 20 /* bits 20 - 23 */
```

This constant is the starting bit position of the ICC in the PSR.

Definition at line 154 of file sparc.h.

#### 10.325.2.27 SPARC_PSR_ICC_MASK

```
#define SPARC_PSR_ICC_MASK 0x00F00000 /* bits 20 - 23 */
```

This constant is a mask for the ICC bits in the PSR.

Definition at line 133 of file sparc.h.

#### 10.325.2.28 SPARC_PSR_IMPL_BIT_POSITION

```
#define SPARC_PSR_IMPL_BIT_POSITION 28 /* bits 28 - 31 */
```

This constant is the starting bit position of the IMPL in the PSR.

Definition at line 158 of file sparc.h.

#### 10.325.2.29 SPARC_PSR_IMPL_MASK

```
#define SPARC_PSR_IMPL_MASK 0xF0000000 /* bits 28 - 31 */
```

This constant is a mask for the IMPL bits in the PSR.

Definition at line 137 of file sparc.h.

#### 10.325.2.30 SPARC_PSR_PIL_BIT_POSITION

```
#define SPARC_PSR_PIL_BIT_POSITION 8 /* bits 8 - 11 */
```

This constant is the starting bit position of the PIL in the PSR.

Definition at line 148 of file sparc.h.

#### 10.325.2.31 SPARC_PSR_PIL_MASK

```
#define SPARC_PSR_PIL_MASK 0x00000F00 /* bits 8 - 11 */
```

This constant is a mask for the PIL bits in the PSR.

Definition at line 127 of file sparc.h.

#### 10.325.2.32 SPARC_PSR_PS_BIT_POSITION

```
#define SPARC_PSR_PS_BIT_POSITION 6 /* bit 6 */
```

This constant is the starting bit position of the PS in the PSR.

Definition at line 144 of file sparc.h.

#### 10.325.2.33 SPARC_PSR_PS_MASK

```
#define SPARC_PSR_PS_MASK 0x00000040 /* bit 6 */
```

This constant is a mask for the PS bits in the PSR.

Definition at line 123 of file sparc.h.

#### 10.325.2.34 SPARC_PSR_S_BIT_POSITION

```
#define SPARC_PSR_S_BIT_POSITION 7 /* bit 7 */
```

This constant is the starting bit position of the S in the PSR.

Definition at line 146 of file sparc.h.

#### 10.325.2.35 SPARC_PSR_S_MASK

```
#define SPARC_PSR_S_MASK 0x00000080 /* bit 7 */
```

This constant is a mask for the S bits in the PSR.

Definition at line 125 of file sparc.h.

#### 10.325.2.36 SPARC_PSR_VER_BIT_POSITION

```
#define SPARC_PSR_VER_BIT_POSITION 24 /* bits 24 - 27 */
```

This constant is the starting bit position of the VER in the PSR.

Definition at line 156 of file sparc.h.

#### 10.325.2.37 SPARC_PSR_VER_MASK

```
#define SPARC_PSR_VER_MASK 0x0F000000 /* bits 24 - 27 */
```

This constant is a mask for the VER bits in the PSR.

Definition at line 135 of file sparc.h.

#### 10.325.2.38 SPARC_REAL_TRAP_NUMBER

```
#define SPARC_REAL_TRAP_NUMBER(
              _trap ) ( ( _trap ) % 256 )
```

Maps the synchronous or asynchronous trap number to the associated real hardware vector number.

**Parameters**

| | |
|---|---|
| *_trap* | is the synchronous or asynchronous trap number to map. |

**Returns**

Returns the real hardware vector number associated with the synchronous or asynchronous trap number.

Definition at line 207 of file sparc.h.

#### 10.325.2.39 sparc_set_psr

```
#define sparc_set_psr(
              _psr )
```

**Value:**

```
  do { \
    __asm__ volatile ( "mov  %0, %%psr " : "=r" ((_psr)) : "0" ((_psr)) ); \
    nop(); \
    nop(); \
    nop(); \
  } while ( 0 )
```

Macro to set the PSR.

This macro sets the PSR register to the value in *_psr*.

Definition at line 301 of file sparc.h.

#### 10.325.2.40 sparc_set_tbr

```
#define sparc_set_tbr(
              _tbr )
```

**Value:**

```
  do { \
     __asm__ volatile( "wr %0, 0, %%tbr" :  "=r" (_tbr) : "0" (_tbr) ); \
  } while ( 0 )
```

Macro to set the TBR.

This macro sets the TBR register to the value in *_tbr*.

Definition at line 347 of file sparc.h.

#### 10.325.2.41 sparc_set_wim

```
#define sparc_set_wim(
              _wim )
```

**Value:**

```
  do { \
    __asm__ volatile( "wr %0, %%wim" :  "=r" (_wim) : "0" (_wim) ); \
    nop(); \
    nop(); \
    nop(); \
  } while ( 0 )
```

Macro to set the WIM.

This macro sets the WIM field to the value in *_wim*.

Definition at line 369 of file sparc.h.

#### 10.325.2.42 sparc_set_y

```
#define sparc_set_y(
              _y )
```

**Value:**

```
do { \
  __asm__ volatile( "wr %0, %%y" :  "=r" (_y) : "0" (_y) ); \
} while ( 0 )
```

Macro to set the Y register.

This macro sets the Y register to the value in *_y*.

Definition at line 392 of file sparc.h.

#### 10.325.2.43 SPARC_SYNCHRONOUS_TRAP

```
#define SPARC_SYNCHRONOUS_TRAP(
             _vector ) ( ( _vector ) + 256 )
```

Maps the real hardware vector number to the associated synchronous trap number.

**Parameters**

| | |
|---|---|
| *_vector* | is the real hardware vector number to map. |

**Returns**

Returns the synchronous trap number associated with the real hardware vector number.

Definition at line 196 of file sparc.h.

### 10.325.3 Function Documentation

#### 10.325.3.1 sparc_disable_interrupts()

```
static uint32_t sparc_disable_interrupts (
             void  )  [inline], [static]
```

SPARC disable processor interrupts.

This method is invoked to disable all maskable interrupts.

**Returns**

This method returns the entire PSR contents.

Definition at line 404 of file sparc.h.

#### 10.325.3.2 sparc_enable_interrupts()

```
static void sparc_enable_interrupts (
            uint32_t psr )  [inline], [static]
```

SPARC enable processor interrupts.

This method is invoked to enable all maskable interrupts.

**Parameters**

| | | |
|---|---|---|
| in | *psr* | is the PSR returned by sparc_disable_interrupts. |

Definition at line 418 of file sparc.h.

#### 10.325.3.3 sparc_syscall_exit()

```
RTEMS_NO_RETURN void sparc_syscall_exit (
            uint32_t exitcode1,
            uint32_t exitcode2 )
```

SPARC exit through system call 1.

This method is invoked to go into system error halt. The optional arguments can be given to hypervisor, hardware debugger, simulator or similar.

System error mode is entered when taking a trap when traps have been disabled. What happens when error mode is entered depends on the motherboard. In a typical development systems the CPU relingish control to the debugger, simulator, hypervisor or similar. The following steps are taken:

1. Going into system error mode by Software Trap 0
2. g1=1 (syscall 1 - Exit)
3. g2=Primary exit code
4. g3=Secondary exit code. Dependends on g2 exit type.

This function never returns.

**Parameters**

| | | |
|---|---|---|
| in | *exitcode1* | Primary exit code stored in CPU g2 register after exit |
| in | *exitcode2* | Primary exit code stored in CPU g3 register after exit |

## 10.326 cpukit/score/src/apimutexisowner.c File Reference

```
#include <rtems/score/apimutex.h>
```

```
#include <rtems/score/percpu.h>
```

### Functions

- bool _API_Mutex_Is_owner (const API_Mutex_Control *the_mutex)

  *Checks if the specified API mutex is owned by the executing thread.*

## 10.327 cpukit/score/src/apimutexlock.c File Reference

Acquires the specified API mutex.

```
#include <rtems/score/apimutex.h>
#include <rtems/score/threadimpl.h>
```

### Functions

- void _API_Mutex_Lock (API_Mutex_Control *the_mutex)

  *Acquires the specified API mutex.*

### 10.327.1 Detailed Description

Acquires the specified API mutex.

## 10.328 cpukit/score/src/apimutexunlock.c File Reference

Releases the Specified API Mutex.

```
#include <rtems/score/apimutex.h>
#include <rtems/score/threadimpl.h>
```

### Functions

- void _API_Mutex_Unlock (API_Mutex_Control *the_mutex)

  *Releases the specified API mutex.*

### 10.328.1 Detailed Description

Releases the Specified API Mutex.

## 10.329 cpukit/score/src/chain.c File Reference

Initialize a Chain Header.

```
#include <rtems/score/chainimpl.h>
#include <rtems/score/address.h>
#include <rtems/score/assert.h>
```

### Functions

- void _Chain_Initialize (Chain_Control *the_chain, void *starting_address, size_t number_nodes, size_t node_size)

  *Initializes a chain header.*

### 10.329.1 Detailed Description

Initialize a Chain Header.

## 10.330 cpukit/score/src/corebarrier.c File Reference

Initialize CORE Barrier.

```
#include <rtems/score/corebarrierimpl.h>
```

### Functions

- void _CORE_barrier_Initialize (CORE_barrier_Control *the_barrier, CORE_barrier_Attributes *the_barrier_attributes)

  *Initializes the core barrier.*

### 10.330.1 Detailed Description

Initialize CORE Barrier.

## 10.331 cpukit/score/src/corebarrierrelease.c File Reference

Manually releases the Barrier.

```
#include <rtems/score/corebarrierimpl.h>
```

### Functions

- uint32_t _CORE_barrier_Do_flush (CORE_barrier_Control *the_barrier, Thread_queue_Flush_filter filter, Thread_queue_Context *queue_context)

  *Flushes the barrier.*

### 10.331.1 Detailed Description

Manually releases the Barrier.

## 10.332 cpukit/score/src/corebarrierwait.c File Reference

Wait For The Barrier.

```
#include <rtems/score/corebarrierimpl.h>
#include <rtems/score/statesimpl.h>
#include <rtems/score/threadimpl.h>
```

### Functions

- Status_Control _CORE_barrier_Seize (CORE_barrier_Control *the_barrier, Thread_Control *executing, bool wait, Thread_queue_Context *queue_context)

  *Waits for the barrier.*

### 10.332.1 Detailed Description

Wait For The Barrier.

## 10.333 cpukit/score/src/coremsg.c File Reference

Initialize a Message Queue.

```
#include <rtems/score/coremsgimpl.h>
#include <rtems/score/assert.h>
```

### Macros

- #define **MESSAGE_SIZE_LIMIT** ( SIZE_MAX - sizeof( uintptr_t ) + 1 - sizeof( CORE_message_queue_Buffer ) )

### Functions

- **RTEMS_STATIC_ASSERT** (offsetof(CORE_message_queue_Buffer, buffer)==sizeof(CORE_message_queue_Buffer), CORE_MESSAGE_QUEUE_BUFFER_OFFSET)
- Status_Control _CORE_message_queue_Initialize (CORE_message_queue_Control *the_message_↩ queue, CORE_message_queue_Disciplines discipline, uint32_t maximum_pending_messages, size_↩ t maximum_message_size, CORE_message_queue_Allocate_buffers allocate_buffers, const void *arg)

  *Initializes a message queue.*

#### 10.333.1 Detailed Description

Initialize a Message Queue.

## 10.334 cpukit/score/src/coremsgbroadcast.c File Reference

Broadcast a Message to the Message Queue.

```
#include <rtems/score/coremsgimpl.h>
#include <rtems/score/objectimpl.h>
```

### Functions

- Status_Control _CORE_message_queue_Broadcast (CORE_message_queue_Control *the_message_↩ queue, const void *buffer, size_t size, uint32_t *count, Thread_queue_Context *queue_context)

  *Broadcasts a message to the message queue.*

#### 10.334.1 Detailed Description

Broadcast a Message to the Message Queue.

## 10.335 cpukit/score/src/coremsgclose.c File Reference

Close a Message Queue.

```
#include <rtems/score/coremsgimpl.h>
```

### Functions

- static Thread_Control * **_CORE_message_queue_Was_deleted** (Thread_Control *the_thread, Thread_queue_Queue *queue, Thread_queue_Context *queue_context)
- void _CORE_message_queue_Close (CORE_message_queue_Control *the_message_queue, Thread_queue_Context *queue_context)

  *Closes a message queue.*

### 10.335.1 Detailed Description

Close a Message Queue.

## 10.336 cpukit/score/src/coremsgflush.c File Reference

Flush Messages Routine.

```
#include <rtems/score/coremsgimpl.h>
```

### Functions

- uint32_t _CORE_message_queue_Flush (CORE_message_queue_Control *the_message_queue, Thread_queue_Context *queue_context)

  *Flushes pending messages.*

### 10.336.1 Detailed Description

Flush Messages Routine.

## 10.337 cpukit/score/src/coremsginsert.c File Reference

Insert a Message into the Message Queue.

```
#include <rtems/score/coremsgimpl.h>
```

### Functions

- static bool **_CORE_message_queue_Order** (const void *left, const Chain_Node *right)
- void _CORE_message_queue_Insert_message (CORE_message_queue_Control *the_message_queue, CORE_message_queue_Buffer *the_message, const void *content_source, size_t content_size, CORE_message_queue_Submit_types submit_type)

  *Inserts a message into the message queue.*

### 10.337.1 Detailed Description

Insert a Message into the Message Queue.

## 10.338 cpukit/score/src/coremsgseize.c File Reference

Seize a Message from the Message Queue.

```
#include <rtems/score/chain.h>
#include <rtems/score/isr.h>
#include <rtems/score/coremsgimpl.h>
#include <rtems/score/threadimpl.h>
#include <rtems/score/statesimpl.h>
```

### Functions

- Status_Control _CORE_message_queue_Seize (CORE_message_queue_Control *the_message_queue, Thread_Control *executing, void *buffer, size_t *size_p, bool wait, Thread_queue_Context *queue_context)

  *Seizes a message from the message queue.*

### 10.338.1 Detailed Description

Seize a Message from the Message Queue.

## 10.339 cpukit/score/src/coremsgsubmit.c File Reference

CORE Message Queue Submit.

```
#include <rtems/score/coremsgimpl.h>
#include <rtems/score/objectimpl.h>
#include <rtems/score/isr.h>
#include <rtems/score/threadimpl.h>
#include <rtems/score/statesimpl.h>
#include <rtems/score/wkspace.h>
```

### Functions

- Status_Control _CORE_message_queue_Submit (CORE_message_queue_Control *the_message_queue, Thread_Control *executing, const void *buffer, size_t size, CORE_message_queue_Submit_types submit↩ _type, bool wait, Thread_queue_Context *queue_context)

  *Submits a message to the message queue.*

### 10.339.1 Detailed Description

CORE Message Queue Submit.

# 10.340 cpukit/score/src/coremsgwkspace.c File Reference

This source file contains the implementation of _CORE_message_queue_Workspace_allocate().

```
#include <rtems/score/coremsgimpl.h>
#include <rtems/score/wkspace.h>
```

## Functions

- void * _CORE_message_queue_Workspace_allocate (CORE_message_queue_Control *the_message_↩ queue, size_t size, const void *arg)

  *This handler allocates the message buffer storage area for a message queue from the RTEMS Workspace.*

### 10.340.1 Detailed Description

This source file contains the implementation of _CORE_message_queue_Workspace_allocate().

# 10.341 cpukit/score/src/coremutexseize.c File Reference

Seize Mutex with Blocking.

```
#include <rtems/score/coremuteximpl.h>
#include <rtems/score/statesimpl.h>
#include <rtems/score/thread.h>
#include <rtems/score/watchdog.h>
```

## Functions

- Status_Control _CORE_mutex_Seize_slow (CORE_mutex_Control *the_mutex, const Thread_queue_Operations *operations, Thread_Control *executing, bool wait, Thread_queue_Context *queue_context)

  *Seize the mutex slowly.*

### 10.341.1 Detailed Description

Seize Mutex with Blocking.

# 10.342 cpukit/score/src/coresem.c File Reference

Core Semaphore Initialize.

```
#include <rtems/score/coresemimpl.h>
```

### Functions

- void _CORE_semaphore_Initialize (CORE_semaphore_Control *the_semaphore, uint32_t initial_value)

  *Initializes the semaphore based on the parameters passed.*

#### 10.342.1 Detailed Description

Core Semaphore Initialize.

## 10.343 cpukit/score/src/coretod.c File Reference

Initializes the Time of Day Handler.

```
#include <rtems/score/todimpl.h>
#include <rtems/score/apimutex.h>
```

### Functions

- void _TOD_Lock (void)

  *Locks the time of day mutex.*
- void _TOD_Unlock (void)

  *Unlocks the time of day mutex.*

### Variables

- TOD_Control _TOD

  *TOD Management information.*
- static API_Mutex_Control **_TOD_Mutex** = API_MUTEX_INITIALIZER( "_TOD" )

#### 10.343.1 Detailed Description

Initializes the Time of Day Handler.

## 10.344 cpukit/score/src/heap.c File Reference

Heap Handler implementation.

```
#include <rtems/score/heapimpl.h>
#include <rtems/score/threadimpl.h>
#include <rtems/score/interr.h>
#include <string.h>
```

### Functions

- bool _Heap_Get_first_and_last_block (uintptr_t heap_area_begin, uintptr_t heap_area_size, uintptr_t page↩ _size, uintptr_t min_block_size, Heap_Block **first_block_ptr, Heap_Block **last_block_ptr)

  *Gets the first and last block for the heap area.*
- uintptr_t _Heap_Initialize (Heap_Control *heap, void *heap_area_begin_ptr, uintptr_t heap_area_size, uintptr_t page_size)

  *Initializes the heap control block.*
- static void **_Heap_Block_split** (Heap_Control *heap, Heap_Block *block, Heap_Block *next_block, Heap_Block *free_list_anchor, uintptr_t alloc_size)
- static Heap_Block * **_Heap_Block_allocate_from_begin** (Heap_Control *heap, Heap_Block *block, Heap_Block *next_block, Heap_Block *free_list_anchor, uintptr_t alloc_size)
- static Heap_Block * **_Heap_Block_allocate_from_end** (Heap_Control *heap, Heap_Block *block, Heap_Block *next_block, Heap_Block *free_list_anchor, uintptr_t alloc_begin, uintptr_t alloc_size)
- Heap_Block * _Heap_Block_allocate (Heap_Control *heap, Heap_Block *block, uintptr_t alloc_begin, uintptr_t alloc_size)

  *Allocates the memory area. starting at alloc_begin of size alloc_size bytes in the block block.*

### 10.344.1 Detailed Description

Heap Handler implementation.

## 10.345 cpukit/score/src/heapallocate.c File Reference

Heap Handler implementation.

```
#include <rtems/score/heapimpl.h>
```

### Macros

- #define **_Heap_Protection_free_delayed_blocks**(heap, alloc_begin) false
- #define **_Heap_Check_allocation**(h, b, ab, as, ag, bd) ((void) 0)

### Functions

- static uintptr_t **_Heap_Check_block** (const Heap_Control *heap, const Heap_Block *block, uintptr_t alloc↩ _size, uintptr_t alignment, uintptr_t boundary)
- void * _Heap_Allocate_aligned_with_boundary (Heap_Control *heap, uintptr_t alloc_size, uintptr_t alignment, uintptr_t boundary)

  *Allocates an aligned memory area with boundary constraint.*

### 10.345.1 Detailed Description

Heap Handler implementation.

## 10.346 cpukit/score/src/interr.c File Reference

Initiates system termination.

```
#include <rtems/score/interr.h>
#include <rtems/score/smpimpl.h>
#include <rtems/score/sysstate.h>
#include <rtems/score/userextimpl.h>
```

### Functions

- void _Terminate (Internal_errors_Source the_source, Internal_errors_t the_error)

  *Initiates system termination.*
- void _Internal_error (Internal_errors_Core_list core_error)

  *Terminates the system with an INTERNAL_ERROR_CORE fatal source and the specified core error code.*

### Variables

- System_state_Codes **_System_state_Current**
- Internal_errors_Information _Internal_errors_What_happened

### 10.346.1 Detailed Description

Initiates system termination.

## 10.347 cpukit/score/src/isr.c File Reference

Initialize the ISR handler.

```
#include <rtems/score/isr.h>
#include <rtems/score/address.h>
#include <rtems/score/interr.h>
#include <rtems/score/percpu.h>
#include <rtems/score/stackimpl.h>
#include <rtems/config.h>
```

### Functions

- void _ISR_Handler_initialization (void)

  *Initializes the ISR handler.*

### 10.347.1 Detailed Description

Initialize the ISR handler.

# 10.348 cpukit/score/src/isrisinprogress.c File Reference

ISR Is In Progress Default Implementation.

```
#include <rtems/score/isr.h>
#include <rtems/score/percpu.h>
```

## Functions

- bool _ISR_Is_in_progress (void)
  *Checks if an ISR in progress.*

### 10.348.1 Detailed Description

ISR Is In Progress Default Implementation.

# 10.349 cpukit/score/src/objectallocate.c File Reference

Allocate Object.

```
#include <rtems/score/objectimpl.h>
```

## Functions

- Objects_Control * _Objects_Allocate (Objects_Information *information)
  *Allocates an object.*

### 10.349.1 Detailed Description

Allocate Object.

# 10.350 cpukit/score/src/objectallocatenone.c File Reference

```
#include <rtems/score/objectdata.h>
```

## Functions

- Objects_Control * _Objects_Allocate_none (Objects_Information *information)
  *Always return NULL.*

## 10.351 cpukit/score/src/objectallocatestatic.c File Reference

```
#include <rtems/score/objectdata.h>
#include <rtems/score/objectimpl.h>
```

### Functions

- Objects_Control * _Objects_Allocate_static (Objects_Information *information)

  *Return an inactive object or NULL.*

## 10.352 cpukit/score/src/objectapimaximumclass.c File Reference

Object API Maximum Class.

```
#include <rtems/score/objectimpl.h>
```

### Functions

- unsigned int _Objects_API_maximum_class (uint32_t api)

  *Returns highest numeric value of a valid API for the specified API.*

### 10.352.1 Detailed Description

Object API Maximum Class.

## 10.353 cpukit/score/src/objectclose.c File Reference

Close Object.

```
#include <rtems/score/objectimpl.h>
```

### Functions

- void _Objects_Close (const Objects_Information *information, Objects_Control *the_object)

  *Closes object.*

### 10.353.1 Detailed Description

Close Object.

## 10.354 cpukit/score/src/objectfree.c File Reference

Free Object.

```
#include <rtems/score/objectimpl.h>
#include <rtems/score/chainimpl.h>
```

### Functions

- void _Objects_Free_unlimited (Objects_Information *information, Objects_Control *the_object)
  *Free the object.*

### 10.354.1 Detailed Description

Free Object.

## 10.355 cpukit/score/src/objectfreestatic.c File Reference

```
#include <rtems/score/objectdata.h>
#include <rtems/score/chainimpl.h>
```

### Functions

- void _Objects_Free_static (Objects_Information *information, Objects_Control *the_object)
  *Free the object.*

## 10.356 cpukit/score/src/objectgetinfo.c File Reference

Get Object Information.

```
#include <rtems/score/objectimpl.h>
```

### Functions

- Objects_Information * _Objects_Get_information (Objects_APIs the_api, uint16_t the_class)
  *Gets object information.*

### 10.356.1 Detailed Description

Get Object Information.

## 10.357 cpukit/score/src/objectgetinfoid.c File Reference

Get Information of an Object from an ID.

```
#include <rtems/score/objectimpl.h>
```

### Functions

- Objects_Information * _Objects_Get_information_id (Objects_Id id)

  *Gets information of an object from an ID.*

### 10.357.1 Detailed Description

Get Information of an Object from an ID.

## 10.358 cpukit/score/src/objectgetlocal.c File Reference

Object Get Local.

```
#include <rtems/score/objectimpl.h>
```

### Functions

- Objects_Control * _Objects_Get (Objects_Id id, ISR_lock_Context *lock_context, const Objects_Information *information)

  *Maps the specified object identifier to the associated local object control block.*

### 10.358.1 Detailed Description

Object Get Local.

## 10.359 cpukit/score/src/objectgetnameasstring.c File Reference

Extracts a Node from a Chain.

```
#include <rtems/score/threadimpl.h>
```

### Functions

- static bool **_Objects_Name_char_is_printable** (char c)
- size_t _Objects_Name_to_string (Objects_Name name, bool is_string, char *buffer, size_t buffer_size)

  *Converts the specified object name to a text representation.*
- char * _Objects_Get_name_as_string (Objects_Id id, size_t length, char *name)

  *Gets object name in the form of a C string.*

### 10.359.1 Detailed Description

Extracts a Node from a Chain.

## 10.360 cpukit/score/src/objectgetnoprotection.c File Reference

Get Object without Dispatching Protection.

```
#include <rtems/score/objectimpl.h>
```

### Functions

- Objects_Control * _Objects_Get_no_protection (Objects_Id id, const Objects_Information *information)

  *Maps object ids to object control blocks.*

### 10.360.1 Detailed Description

Get Object without Dispatching Protection.

## 10.361 cpukit/score/src/objectinitializeinformation.c File Reference

Initialize Object Information.

```
#include <rtems/score/objectimpl.h>
#include <rtems/score/address.h>
#include <rtems/score/chainimpl.h>
#include <rtems/score/interr.h>
#include <rtems/score/sysstate.h>
#include <rtems/score/wkspace.h>
```

### Functions

- void _Objects_Initialize_information (Objects_Information *information)

  *Initializes the specified objects information.*

### 10.361.1 Detailed Description

Initialize Object Information.

## 10.362 cpukit/score/src/objectnamespaceremove.c File Reference

Removes Object from Namespace.

```
#include <rtems/score/objectimpl.h>
#include <rtems/score/wkspace.h>
```

### Functions

- void _Objects_Namespace_remove_string (const Objects_Information *information, Objects_Control *the↩ _object)

  *Removes object with a string name from its namespace.*

### 10.362.1 Detailed Description

Removes Object from Namespace.

## 10.363 cpukit/score/src/objectnametoid.c File Reference

Object Name To Id.

```
#include <rtems/score/objectimpl.h>
```

### Functions

- Objects_Name_or_id_lookup_errors _Objects_Name_to_id_u32 (uint32_t name, uint32_t node, Objects_Id *id, const Objects_Information *information)

  *Searches an object of the specified class with the specified name on the specified set of nodes.*

### 10.363.1 Detailed Description

Object Name To Id.

# 10.364 cpukit/score/src/percpu.c File Reference

Allocate and Initialize Per CPU Structures.

```
#include <rtems/score/percpu.h>
#include <rtems/score/assert.h>
#include <rtems/score/isrlock.h>
#include <rtems/score/smpimpl.h>
#include <rtems/config.h>
```

## Functions

- **RTEMS_STATIC_ASSERT** (sizeof(CPU_Uint32ptr) >=sizeof(uintptr_t), CPU_Uint32ptr_greater_equal_↩ uintptr_t)
- **RTEMS_STATIC_ASSERT** (sizeof(CPU_Uint32ptr) >=sizeof(uint32_t), CPU_Uint32ptr_greater_equal_↩ uint32_t)
- static void **_Per_CPU_State_acquire** (ISR_lock_Context *lock_context)
- static void **_Per_CPU_State_release** (ISR_lock_Context *lock_context)
- static void **_Per_CPU_State_busy_wait** (Per_CPU_Control *cpu, Per_CPU_State new_state)
- static Per_CPU_State **_Per_CPU_State_get_next** (Per_CPU_State current_state, Per_CPU_State new_↩ state)
- void **_Per_CPU_State_change** (Per_CPU_Control *cpu, Per_CPU_State new_state)

### 10.364.1 Detailed Description

Allocate and Initialize Per CPU Structures.

# 10.365 cpukit/score/src/processormaskcopy.c File Reference

Processor Mask Implementation.

```
#include <rtems/score/processormask.h>
```

## Functions

- Processor_mask_Copy_status _Processor_mask_Copy (long *dst, size_t dst_size, const long *src, size_t src_size)

  *Copies one mask to another.*

## Variables

- const Processor_mask **_Processor_mask_The_one_and_only** = { .__bits[ 0 ] = 1 }

### 10.365.1 Detailed Description

Processor Mask Implementation.

## 10.366 cpukit/score/src/rbtreenext.c File Reference

_RBTree_Next() and _RBTree_Next() implementation.

```
#include <rtems/score/rbtreeimpl.h>
#include <rtems/score/basedefs.h>
```

### Functions

- RBTree_Node * _RBTree_Minimum (const RBTree_Control *tree)

  *Returns the minimum node of the red-black tree.*
- RBTree_Node * _RBTree_Maximum (const RBTree_Control *tree)

  *Returns the maximum node of the red-black tree.*
- RBTree_Node * _RBTree_Successor (const RBTree_Node *node)

  *Returns the successor of a node.*
- RBTree_Node * _RBTree_Predecessor (const RBTree_Node *node)

  *Returns the predecessor of a node.*

### 10.366.1 Detailed Description

_RBTree_Next() and _RBTree_Next() implementation.

## 10.367 cpukit/score/src/rbtreereplace.c File Reference

_RBTree_Replace_node() implementation.

```
#include <rtems/score/rbtree.h>
```

### Functions

- void _RBTree_Replace_node (RBTree_Control *the_rbtree, RBTree_Node *victim, RBTree_Node *replacement)

  *Replaces a node in the red-black tree without a rebalance.*

### 10.367.1 Detailed Description

_RBTree_Replace_node() implementation.

# 10.368 cpukit/score/src/scheduler.c File Reference

Scheduler Initialize.

```
#include <rtems/score/schedulerimpl.h>
```

## Functions

- void _Scheduler_Handler_initialization (void)

  *Initializes the scheduler to the policy chosen by the user.*

### 10.368.1 Detailed Description

Scheduler Initialize.

# 10.369 cpukit/score/src/schedulerdefaultnodedestroy.c File Reference

Scheduler Default Node Destruction Operation.

```
#include <rtems/score/scheduler.h>
```

## Functions

- void _Scheduler_default_Node_destroy (const Scheduler_Control *scheduler, Scheduler_Node *node)

  *Does nothing.*

### 10.369.1 Detailed Description

Scheduler Default Node Destruction Operation.

# 10.370 cpukit/score/src/schedulerdefaultnodeinit.c File Reference

Scheduler Default Node Initialization Operation.

```
#include <rtems/score/schedulerimpl.h>
```

## Functions

- void _Scheduler_default_Node_initialize (const Scheduler_Control *scheduler, Scheduler_Node *node, Thread_Control *the_thread, Priority_Control priority)

  *Performs the scheduler base node initialization.*

### 10.370.1 Detailed Description

Scheduler Default Node Initialization Operation.

## 10.371 cpukit/score/src/schedulerdefaultreleasejob.c File Reference

Default Scheduler Release Job Operation.

```
#include <rtems/score/scheduler.h>
```

### Functions

- void _Scheduler_default_Release_job (const Scheduler_Control *scheduler, Thread_Control *the_thread, Priority_Node *priority_node, uint64_t deadline, Thread_queue_Context *queue_context)

  *Does nothing.*
- void _Scheduler_default_Cancel_job (const Scheduler_Control *scheduler, Thread_Control *the_thread, Priority_Node *priority_node, Thread_queue_Context *queue_context)

  *Does nothing.*

### 10.371.1 Detailed Description

Default Scheduler Release Job Operation.

## 10.372 cpukit/score/src/schedulerdefaultsetaffinity.c File Reference

Scheduler Default Set Affinity Operation.

```
#include <rtems/score/schedulerimpl.h>
```

### Functions

- bool _Scheduler_default_Set_affinity (const Scheduler_Control *scheduler, Thread_Control *thread, Scheduler_Node *node, const Processor_mask *affinity)

  *Checks if the processor set of the scheduler is the subset of the affinity set.*

### 10.372.1 Detailed Description

Scheduler Default Set Affinity Operation.

# 10.373 cpukit/score/src/schedulerdefaulttick.c File Reference

Default Scheduler At Tick Handler.

```
#include <rtems/score/schedulerimpl.h>
#include <rtems/score/threadimpl.h>
#include <rtems/score/smp.h>
#include <rtems/config.h>
```

## Functions

- void _Scheduler_default_Tick (const Scheduler_Control *scheduler, Thread_Control *executing)

  *Performs tick operations depending on the CPU budget algorithm for each executing thread.*

### 10.373.1 Detailed Description

Default Scheduler At Tick Handler.

# 10.374 cpukit/score/src/scheduleredfreleasejob.c File Reference

Scheduler EDF Release Job.

```
#include <rtems/score/scheduleredfimpl.h>
```

## Functions

- Priority_Control _Scheduler_EDF_Map_priority (const Scheduler_Control *scheduler, Priority_Control priority)

  *Gets the mapped priority map of the priority control.*
- Priority_Control _Scheduler_EDF_Unmap_priority (const Scheduler_Control *scheduler, Priority_Control priority)

  *Gets the unmapped priority map of the priority control.*
- void _Scheduler_EDF_Release_job (const Scheduler_Control *scheduler, Thread_Control *the_thread, Priority_Node *priority_node, uint64_t deadline, Thread_queue_Context *queue_context)

  *Releases a EDF job.*
- void _Scheduler_EDF_Cancel_job (const Scheduler_Control *scheduler, Thread_Control *the_thread, Priority_Node *priority_node, Thread_queue_Context *queue_context)

  *Cancels a job and removes the thread from the queue context.*

### 10.374.1 Detailed Description

Scheduler EDF Release Job.

## 10.375 cpukit/score/src/scheduleredfsmp.c File Reference

EDF SMP Scheduler Implementation.

```
#include <rtems/score/scheduleredfsmp.h>
#include <rtems/score/schedulersmpimpl.h>
```

### Functions

- static Scheduler_EDF_SMP_Context * **_Scheduler_EDF_SMP_Get_context** (const Scheduler_Control *scheduler)
- static Scheduler_EDF_SMP_Context * **_Scheduler_EDF_SMP_Get_self** (Scheduler_Context *context)
- static Scheduler_EDF_SMP_Node * **_Scheduler_EDF_SMP_Node_downcast** (Scheduler_Node *node)
- static bool **_Scheduler_EDF_SMP_Priority_less_equal** (const void *left, const RBTree_Node *right)
- void _Scheduler_EDF_SMP_Initialize (const Scheduler_Control *scheduler)

  *Initializes the context of the scheduler control.*
- void _Scheduler_EDF_SMP_Node_initialize (const Scheduler_Control *scheduler, Scheduler_Node *node, Thread_Control *the_thread, Priority_Control priority)

  *Initializes the node with the given priority.*
- static void **_Scheduler_EDF_SMP_Do_update** (Scheduler_Context *context, Scheduler_Node *node, Priority_Control new_priority)
- static bool **_Scheduler_EDF_SMP_Has_ready** (Scheduler_Context *context)
- static bool **_Scheduler_EDF_SMP_Overall_less** (const Scheduler_EDF_SMP_Node *left, const Scheduler_EDF_SMP_Node *right)
- static Scheduler_EDF_SMP_Node * **_Scheduler_EDF_SMP_Challenge_highest_ready** (Scheduler_EDF_SMP_Context *self, Scheduler_EDF_SMP_Node *highest_ready, RBTree_Control *ready_queue)
- static Scheduler_Node * **_Scheduler_EDF_SMP_Get_highest_ready** (Scheduler_Context *context, Scheduler_Node *filter)
- static void **_Scheduler_EDF_SMP_Set_scheduled** (Scheduler_EDF_SMP_Context *self, Scheduler_EDF_SMP_Node *scheduled, const Per_CPU_Control *cpu)
- static Scheduler_EDF_SMP_Node * **_Scheduler_EDF_SMP_Get_scheduled** (const Scheduler_EDF_SMP_Context *self, uint8_t rqi)
- static Scheduler_Node * **_Scheduler_EDF_SMP_Get_lowest_scheduled** (Scheduler_Context *context, Scheduler_Node *filter_base)
- static void **_Scheduler_EDF_SMP_Insert_ready** (Scheduler_Context *context, Scheduler_Node *node_↩ base, Priority_Control insert_priority)
- static void **_Scheduler_EDF_SMP_Extract_from_scheduled** (Scheduler_Context *context, Scheduler_Node *node_to_extract)
- static void **_Scheduler_EDF_SMP_Extract_from_ready** (Scheduler_Context *context, Scheduler_Node *node_to_extract)
- static void **_Scheduler_EDF_SMP_Move_from_scheduled_to_ready** (Scheduler_Context *context, Scheduler_Node *scheduled_to_ready)
- static void **_Scheduler_EDF_SMP_Move_from_ready_to_scheduled** (Scheduler_Context *context, Scheduler_Node *ready_to_scheduled)
- static void **_Scheduler_EDF_SMP_Allocate_processor** (Scheduler_Context *context, Scheduler_Node *scheduled_base, Scheduler_Node *victim_base, Per_CPU_Control *victim_cpu)
- void _Scheduler_EDF_SMP_Block (const Scheduler_Control *scheduler, Thread_Control *thread, Scheduler_Node *node)

  *Blocks the thread.*
- static bool **_Scheduler_EDF_SMP_Enqueue** (Scheduler_Context *context, Scheduler_Node *node, Priority_Control insert_priority)
- static bool **_Scheduler_EDF_SMP_Enqueue_scheduled** (Scheduler_Context *context, Scheduler_Node *node, Priority_Control insert_priority)

- void _Scheduler_EDF_SMP_Unblock (const Scheduler_Control *scheduler, Thread_Control *thread, Scheduler_Node *node)

  *Unblocks the thread.*
- static bool **_Scheduler_EDF_SMP_Do_ask_for_help** (Scheduler_Context *context, Thread_Control *the↩ _thread, Scheduler_Node *node)
- void _Scheduler_EDF_SMP_Update_priority (const Scheduler_Control *scheduler, Thread_Control *thread, Scheduler_Node *node)

  *Updates the priority of the node.*
- bool _Scheduler_EDF_SMP_Ask_for_help (const Scheduler_Control *scheduler, Thread_Control *the_↩ thread, Scheduler_Node *node)

  *Asks for help operation.*
- void _Scheduler_EDF_SMP_Reconsider_help_request (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Reconsiders help operation.*
- void _Scheduler_EDF_SMP_Withdraw_node (const Scheduler_Control *scheduler, Thread_Control *the_↩ thread, Scheduler_Node *node, Thread_Scheduler_state next_state)

  *Withdraws node operation.*
- static void **_Scheduler_EDF_SMP_Register_idle** (Scheduler_Context *context, Scheduler_Node *idle_↩ base, Per_CPU_Control *cpu)
- void _Scheduler_EDF_SMP_Add_processor (const Scheduler_Control *scheduler, Thread_Control *idle)

  *Adds processor.*
- Thread_Control * _Scheduler_EDF_SMP_Remove_processor (const Scheduler_Control *scheduler, Per_CPU_Control *cpu)

  *Removes an idle thread from the given cpu.*
- void _Scheduler_EDF_SMP_Yield (const Scheduler_Control *scheduler, Thread_Control *thread, Scheduler_Node *node)

  *Performs the yield of a thread.*
- static void **_Scheduler_EDF_SMP_Do_set_affinity** (Scheduler_Context *context, Scheduler_Node *node_base, void *arg)
- void _Scheduler_EDF_SMP_Start_idle (const Scheduler_Control *scheduler, Thread_Control *idle, Per_CPU_Control *cpu)

  *Starts an idle thread.*
- void _Scheduler_EDF_SMP_Pin (const Scheduler_Control *scheduler, Thread_Control *thread, Scheduler_Node *node_base, struct Per_CPU_Control *cpu)

  *Pin thread operation.*
- void _Scheduler_EDF_SMP_Unpin (const Scheduler_Control *scheduler, Thread_Control *thread, Scheduler_Node *node_base, struct Per_CPU_Control *cpu)

  *Unpin thread operation.*
- bool _Scheduler_EDF_SMP_Set_affinity (const Scheduler_Control *scheduler, Thread_Control *thread, Scheduler_Node *node_base, const Processor_mask *affinity)

  *Checks if the processor set of the scheduler is the subset of the affinity set.*

### 10.375.1 Detailed Description

EDF SMP Scheduler Implementation.

## 10.376 cpukit/score/src/schedulerpriorityblock.c File Reference

Scheduler Priority Block.

```
#include <rtems/score/schedulerprorityimpl.h>
```

### Functions

- void _Scheduler_priority_Block (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Blocks the thread.*

#### 10.376.1 Detailed Description

Scheduler Priority Block.

## 10.377 cpukit/score/src/schedulerprioritychangepriority.c File Reference

Removes Thread from Thread Queue.

```
#include <rtems/score/schedulerpriorityimpl.h>
```

### Functions

- void _Scheduler_priority_Update_priority (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Updates the priority of the node.*

#### 10.377.1 Detailed Description

Removes Thread from Thread Queue.

## 10.378 cpukit/score/src/schedulerpriorityschedule.c File Reference

Priority Scheduler Schedule Method.

```
#include <rtems/score/schedulerpriorityimpl.h>
```

### Functions

- void _Scheduler_priority_Schedule (const Scheduler_Control *scheduler, Thread_Control *the_thread)

  *Sets the heir thread to be the next ready thread.*

#### 10.378.1 Detailed Description

Priority Scheduler Schedule Method.

# 10.379 cpukit/score/src/schedulerpriorityunblock.c File Reference

Scheduler Priority Unblock.

```
#include <rtems/score/schedulerprorityimpl.h>
```

## Functions

- void _Scheduler_priority_Unblock (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Unblocks the thread.*

### 10.379.1 Detailed Description

Scheduler Priority Unblock.

# 10.380 cpukit/score/src/schedulerpriorityyield.c File Reference

Scheduler Priority Yield.

```
#include <rtems/score/schedulerprorityimpl.h>
#include <rtems/score/threadimpl.h>
```

## Functions

- void _Scheduler_priority_Yield (const Scheduler_Control *scheduler, Thread_Control *the_thread, Scheduler_Node *node)

  *Performs the yield of a thread.*

### 10.380.1 Detailed Description

Scheduler Priority Yield.

# 10.381 cpukit/score/src/smp.c File Reference

SMP Support.

```
#include <rtems/score/smpimpl.h>
#include <rtems/score/assert.h>
#include <rtems/score/memory.h>
#include <rtems/score/percpudata.h>
#include <rtems/score/schedulerimpl.h>
#include <rtems/score/threadimpl.h>
#include <rtems/config.h>
#include <rtems/sysinit.h>
#include <string.h>
```

## Enumerations

- enum {
  **_Sysinit_bsp_start** = 0x00030080, **_Sysinit_bsp_debug_uart_init** = 0x00030003, **_Sysinit_amba_↩ initialize** = 0x00030001, **_Sysinit_leon3_cpu_index_init** = 0x00030000,
  **_Sysinit_leon3_counter_initialize** = 0x00040000, **_Sysinit_bsp_memory_initialize** = 0x00018080, _↩ **Sysinit__Malloc_Initialize** = 0x00028080, **_Sysinit__Barrier_Manager_initialization** = 0x00140080,
  **_Sysinit__Message_queue_Manager_initialization** = 0x000e0080, **_Sysinit__Partition_Manager_↩ initialization** = 0x00100080, **_Sysinit__Rate_monotonic_Manager_initialization** = 0x00130080, _↩ **Sysinit__Semaphore_Manager_initialization** = 0x000f0080,
  **_Sysinit__RTEMS_tasks_Manager_initialization** = 0x000a0080, **_Sysinit__Timer_Manager_↩ initialization** = 0x000b0080, **_Sysinit_rtems_initialize_data_structures** = 0x00070080, **_Sysinit__↩ Extension_Manager_initialization** = 0x00090080,
  **_Sysinit__Per_CPU_Data_initialize** = 0x00022080, **_Sysinit__Workspace_Handler_initialization** = 0x00026080, **_Sysinit_init_task** = 0x00290080, **_Sysinit_init_runner_task** = 0x00290080 }

## Functions

- static const Scheduler_Assignment * **_Scheduler_Get_initial_assignment** (uint32_t cpu_index)
- static bool **_Scheduler_Is_mandatory_processor** (const Scheduler_Assignment *assignment)
- static bool **_Scheduler_Should_start_processor** (const Scheduler_Assignment *assignment)
- static void **_SMP_Start_processors** (uint32_t cpu_max)
- void _SMP_Handler_initialize (void)

  *Initializes SMP Handler.*
- void _SMP_Request_start_multitasking (void)

  *Requests a multitasking start on all configured and available processors.*
- bool _SMP_Should_start_processor (uint32_t cpu_index)

  *Checks if the processor with the specified index should be started.*
- void _SMP_Start_multitasking_on_secondary_processor (Per_CPU_Control *cpu_self)

  *Performs high-level initialization of a secondary processor and runs the application threads.*
- void _SMP_Request_shutdown (void)

  *Requests a shutdown of all processors.*
- void _SMP_Send_message (uint32_t cpu_index, unsigned long message)

  *Sends an SMP message to a processor.*
- void _SMP_Send_message_broadcast (unsigned long message)

  *Sends an SMP message to all other online processors.*
- void _SMP_Send_message_multicast (const Processor_mask *targets, unsigned long message)

  *Sends an SMP message to a set of processors.*
- static void **_Per_CPU_Data_initialize** (void)

## Variables

- Processor_mask _SMP_Online_processors

  *Set of online processors.*
- uint32_t **_SMP_Processor_maximum**
- rtems_sysinit_item const **_Linker_set__Sysinit__Per_CPU_Data_initialize** = { _Per_CPU_Data_initialize }

### 10.381.1 Detailed Description

SMP Support.

## 10.382 cpukit/score/src/stackallocatorfreenothing.c File Reference

_Stack_Free_nothing() Implementation

```
#include <rtems/score/stackimpl.h>
```

### Functions

- void _Stack_Free_nothing (void *stack_area)

  *This function does nothing.*

### 10.382.1 Detailed Description

_Stack_Free_nothing() Implementation

## 10.383 cpukit/score/src/thread.c File Reference

Initialize Thread Handler.

```
#include <rtems/score/threadimpl.h>
#include <rtems/score/freechainimpl.h>
#include <rtems/score/objectimpl.h>
#include <rtems/score/scheduler.h>
```

### Macros

- #define **THREAD_OFFSET_ASSERT**(field)

### Functions

- **THREAD_OFFSET_ASSERT** (Object)
- **THREAD_OFFSET_ASSERT** (Join_queue)
- **THREAD_OFFSET_ASSERT** (current_state)
- **THREAD_OFFSET_ASSERT** (Real_priority)
- **THREAD_OFFSET_ASSERT** (Scheduler)
- **THREAD_OFFSET_ASSERT** (Wait)
- **THREAD_OFFSET_ASSERT** (Timer)
- void _Thread_Initialize_information (Thread_Information *information)

  *Initializes the thread information.*
- void _Thread_Handler_initialization (void)

  *Initializes thread handler.*

### 10.383.1 Detailed Description

Initialize Thread Handler.

### 10.383.2 Macro Definition Documentation

#### 10.383.2.1 THREAD_OFFSET_ASSERT

```
#define THREAD_OFFSET_ASSERT(
              field )
```

**Value:**

```
RTEMS_STATIC_ASSERT( \
  offsetof( Thread_Control, field ) == offsetof( Thread_Proxy_control, field ), \
  field \
)
```

Definition at line 26 of file thread.c.

## 10.384 cpukit/score/src/threadchangepriority.c File Reference

Changes the Priority of a Thread.

```
#include <rtems/score/threadimpl.h>
#include <rtems/score/assert.h>
#include <rtems/score/schedulerimpl.h>
```

### Functions

- static void **_Thread_Set_scheduler_node_priority** (Priority_Aggregation *priority_aggregation, bool prepend_it)
- static void **_Thread_Priority_action_add** (Priority_Aggregation *priority_aggregation, Priority_Actions *priority_actions, void *arg)
- static void **_Thread_Priority_action_remove** (Priority_Aggregation *priority_aggregation, Priority_Actions *priority_actions, void *arg)
- static void **_Thread_Priority_action_change** (Priority_Aggregation *priority_aggregation, bool prepend_it, Priority_Actions *priority_actions, void *arg)
- static void **_Thread_Priority_do_perform_actions** (Thread_Control *the_thread, Thread_queue_Queue *queue, const Thread_queue_Operations *operations, bool prepend_it, Thread_queue_Context *queue_↩ context)
- void _Thread_Priority_perform_actions (Thread_Control *start_of_path, Thread_queue_Context *queue_↩ context)

  *Checks if the thread is owner of the lock of the join queue.*
- static void **_Thread_Priority_apply** (Thread_Control *the_thread, Priority_Node *priority_action_node, Thread_queue_Context *queue_context, bool prepend_it, Priority_Action_type priority_action_type)
- void _Thread_Priority_add (Thread_Control *the_thread, Priority_Node *priority_node, Thread_queue_Context *queue_context)

  *Adds the specified thread priority node to the corresponding thread priority aggregation.*
- void _Thread_Priority_remove (Thread_Control *the_thread, Priority_Node *priority_node, Thread_queue_Context *queue_context)

  *Removes the specified thread priority node from the corresponding thread priority aggregation.*
- void _Thread_Priority_changed (Thread_Control *the_thread, Priority_Node *priority_node, bool prepend↩ _it, Thread_queue_Context *queue_context)

*Propagates a thread priority value change in the specified thread priority node to the corresponding thread priority aggregation.*

- void _Thread_Priority_replace (Thread_Control *the_thread, Priority_Node *victim_node, Priority_Node *replacement_node)

  *Replaces the victim priority node with the replacement priority node in the corresponding thread priority aggregation.*
- void _Thread_Priority_update (Thread_queue_Context *queue_context)

  *Updates the priority of all threads in the set.*
- void _Thread_Priority_and_sticky_update (Thread_Control *the_thread, int sticky_level_change)

  *Updates the priority of the thread and changes it sticky level.*

### 10.384.1 Detailed Description

Changes the Priority of a Thread.

## 10.385 cpukit/score/src/threadclearstate.c File Reference

Clear Thread state.

```
#include <rtems/score/threadimpl.h>
#include <rtems/score/assert.h>
#include <rtems/score/schedulerimpl.h>
```

### Functions

- States_Control _Thread_Clear_state_locked (Thread_Control *the_thread, States_Control state)

  *Clears the specified thread state without locking the lock context.*
- States_Control _Thread_Clear_state (Thread_Control *the_thread, States_Control state)

  *Clears the specified thread state.*

### 10.385.1 Detailed Description

Clear Thread state.

## 10.386 cpukit/score/src/threadcreateidle.c File Reference

Create Idle Thread.

```
#include <rtems/score/threadidledata.h>
#include <rtems/score/threadimpl.h>
#include <rtems/score/assert.h>
#include <rtems/score/schedulerimpl.h>
#include <rtems/score/stackimpl.h>
#include <rtems/score/sysstate.h>
#include <rtems/score/userextimpl.h>
#include <string.h>
```

### Functions

- static void **_Thread_Create_idle_for_CPU** (Per_CPU_Control *cpu)
- void _Thread_Create_idle (void)

  *Creates idle thread.*

### 10.386.1 Detailed Description

Create Idle Thread.

## 10.387 cpukit/score/src/threaddispatch.c File Reference

Dispatch Thread.

```
#include <rtems/score/threaddispatch.h>
#include <rtems/score/assert.h>
#include <rtems/score/isr.h>
#include <rtems/score/schedulerimpl.h>
#include <rtems/score/threadimpl.h>
#include <rtems/score/todimpl.h>
#include <rtems/score/userextimpl.h>
#include <rtems/score/wkspace.h>
#include <rtems/config.h>
```

### Functions

- **CHAIN_DEFINE_EMPTY** (_User_extensions_Switches_list)
- static ISR_Level **_Thread_Check_pinning** (Thread_Control *executing, Per_CPU_Control *cpu_self, ISR_Level level)
- static void **_Thread_Ask_for_help** (Thread_Control *the_thread)
- static bool **_Thread_Can_ask_for_help** (const Thread_Control *executing)
- static ISR_Level **_Thread_Preemption_intervention** (Thread_Control *executing, Per_CPU_Control *cpu_self, ISR_Level level)
- static void **_Thread_Post_switch_cleanup** (Thread_Control *executing)
- static Thread_Action * **_Thread_Get_post_switch_action** (Thread_Control *executing)
- static void **_Thread_Run_post_switch_actions** (Thread_Control *executing)
- void _Thread_Do_dispatch (Per_CPU_Control *cpu_self, ISR_Level level)

  *Performs a thread dispatch on the current processor.*
- void _Thread_Dispatch (void)

  *Performs a thread dispatch if necessary.*
- void _Thread_Dispatch_direct (Per_CPU_Control *cpu_self)

  *Directly do a thread dispatch.*
- void _Thread_Dispatch_enable (Per_CPU_Control *cpu_self)

  *Enables thread dispatching.*

### 10.387.1 Detailed Description

Dispatch Thread.

# 10.388 cpukit/score/src/threadget.c File Reference

Maps Thread IDs to TCB Pointer.

```
#include <rtems/score/threadimpl.h>
```

## Functions

- Thread_Control * _Thread_Get (Objects_Id id, ISR_lock_Context *lock_context)

  *Gets a thread by its identifier.*

### 10.388.1 Detailed Description

Maps Thread IDs to TCB Pointer.

# 10.389 cpukit/score/src/threadhandler.c File Reference

Thread Handler.

```
#include <rtems/score/threadimpl.h>
#include <rtems/score/assert.h>
#include <rtems/score/interr.h>
#include <rtems/score/isrlevel.h>
#include <rtems/score/userextimpl.h>
```

## Functions

- **RTEMS_STATIC_ASSERT** (CPU_USE_LIBC_INIT_FINI_ARRAY==TRUE||CPU_USE_LIBC_INIT_FINI_↩ ARRAY==FALSE, CPU_USE_LIBC_INIT_FINI_ARRAY)
- static void **_Thread_Global_construction** (Thread_Control *executing)
- void _Thread_Handler (void)

  *Wrapper function for all threads.*

## Variables

- Objects_Id _Thread_Global_constructor

  *Object identifier of the global constructor thread.*

### 10.389.1 Detailed Description

Thread Handler.

## 10.390 cpukit/score/src/threadinitialize.c File Reference

Initialize Thread.

```
#include <rtems/score/threadimpl.h>
#include <rtems/score/freechainimpl.h>
#include <rtems/score/schedulerimpl.h>
#include <rtems/score/stackimpl.h>
#include <rtems/score/tls.h>
#include <rtems/score/userextimpl.h>
#include <rtems/score/watchdogimpl.h>
#include <rtems/config.h>
```

### Functions

- bool _Thread_Initialize (Thread_Information *information, Thread_Control *the_thread, const Thread_Configuration *config)

  *Initializes thread.*

### 10.390.1 Detailed Description

Initialize Thread.

## 10.391 cpukit/score/src/threadloadenv.c File Reference

Initializes Enviroment for A Thread.

```
#include <rtems/score/threadimpl.h>
```

### Functions

- void _Thread_Load_environment (Thread_Control *the_thread)

  *Initializes enviroment for a thread.*

### 10.391.1 Detailed Description

Initializes Enviroment for A Thread.

## 10.392 cpukit/score/src/threadq.c File Reference

Thread Queue Initialize.

```
#include <string.h>
#include <rtems/score/threadqimpl.h>
#include <rtems/score/rbtreeimpl.h>
#include <rtems/score/threadimpl.h>
```

### Functions

- **RTEMS_STATIC_ASSERT** (offsetof(Thread_queue_Syslock_queue, Queue.Lock.next_ticket)==offsetof(struct _Thread_queue_Queue, _Lock._next_ticket), THREAD_QUEUE_SYSLOCK_QUEUE_NEXT_TICKET)
- **RTEMS_STATIC_ASSERT** (offsetof(Thread_queue_Syslock_queue, Queue.Lock.now_serving)==offsetof(struct _Thread_queue_Queue, _Lock._now_serving), THREAD_QUEUE_SYSLOCK_QUEUE_NOW_SERVING)
- **RTEMS_STATIC_ASSERT** (offsetof(Thread_queue_Syslock_queue, Queue.heads)==offsetof(struct _Thread_queue_Queue, _heads), THREAD_QUEUE_SYSLOCK_QUEUE_HEADS)
- **RTEMS_STATIC_ASSERT** (offsetof(Thread_queue_Syslock_queue, Queue.owner)==offsetof(struct _Thread_queue_Queue, _owner), THREAD_QUEUE_SYSLOCK_QUEUE_OWNER)
- **RTEMS_STATIC_ASSERT** (offsetof(Thread_queue_Syslock_queue, Queue.name)==offsetof(struct _Thread_queue_Queue, _name), THREAD_QUEUE_SYSLOCK_QUEUE_NAME)
- **RTEMS_STATIC_ASSERT** (sizeof(Thread_queue_Syslock_queue)==sizeof(struct _Thread_queue_Queue), THREAD_QUEUE_SYSLOCK_QUEUE_SIZE)
- void _Thread_queue_Do_acquire_critical (Thread_queue_Control *the_thread_queue, ISR_lock_Context *lock_context)

  *Acquires the thread queue control in a critical section.*
- void _Thread_queue_Acquire (Thread_queue_Control *the_thread_queue, Thread_queue_Context *queue_context)

  *Acquires the thread queue control in a critical section.*
- void _Thread_queue_Do_release_critical (Thread_queue_Control *the_thread_queue, ISR_lock_Context *lock_context)

  *Checks if the thread queue control is the owner of the lock.*
- void _Thread_queue_Release (Thread_queue_Control *the_thread_queue, Thread_queue_Context *queue_context)

  *Releases the thread queue control and enables interrupts.*
- void _Thread_queue_Initialize (Thread_queue_Control *the_thread_queue, const char *name)

  *Initializes the thread queue control to the given name.*
- void _Thread_queue_Object_initialize (Thread_queue_Control *the_thread_queue)

  *Initializes a thread queue embedded in an object with identifier.*
- size_t _Thread_queue_Queue_get_name_and_id (const Thread_queue_Queue *queue, char *buffer, size_t buffer_size, Objects_Id *id)

  *Copies the thread queue name to the specified buffer.*

### Variables

- const char _Thread_queue_Object_name [ ] = { '\0' }

  *The special thread queue name to indicated that the thread queue is embedded in an object with identifier.*

#### 10.392.1 Detailed Description

Thread Queue Initialize.

## 10.393 cpukit/score/src/threadqenqueue.c File Reference

Thread Queue Operations.

```
#include <rtems/score/threadqimpl.h>
#include <rtems/score/assert.h>
#include <rtems/score/threaddispatch.h>
#include <rtems/score/threadimpl.h>
#include <rtems/score/status.h>
#include <rtems/score/watchdogimpl.h>
```

## Classes

- struct Thread_queue_Links

## Macros

- #define **THREAD_QUEUE_INTEND_TO_BLOCK** (THREAD_WAIT_CLASS_OBJECT | THREAD_WAIT_STATE_INTEND_T
- #define **THREAD_QUEUE_BLOCKED** (THREAD_WAIT_CLASS_OBJECT | THREAD_WAIT_STATE_BLOCKED)
- #define **THREAD_QUEUE_READY_AGAIN** (THREAD_WAIT_CLASS_OBJECT | THREAD_WAIT_STATE_READY_AGAIN)

## Functions

- static bool **_Thread_queue_Link_equal** (const void *left, const RBTree_Node *right)
- static bool **_Thread_queue_Link_less** (const void *left, const RBTree_Node *right)
- static void * **_Thread_queue_Link_map** (RBTree_Node *node)
- static Thread_queue_Link * **_Thread_queue_Link_find** (Thread_queue_Links *links, Thread_queue_Queue *source)
- static bool **_Thread_queue_Link_add** (Thread_queue_Link *link, Thread_queue_Queue *source, Thread_queue_Queue *target)
- static void **_Thread_queue_Link_remove** (Thread_queue_Link *link)
- void _Thread_queue_Path_release_critical (Thread_queue_Context *queue_context)

  *Releases the thread queue path in a critical section.*
- static void **_Thread_queue_Path_append_deadlock_thread** (Thread_Control *the_thread, Thread_queue_Context *queue_context)
- bool _Thread_queue_Path_acquire_critical (Thread_queue_Queue *queue, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Does nothing.*
- void _Thread_queue_Enqueue_do_nothing_extra (Thread_queue_Queue *queue, Thread_Control *the_thread, Per_CPU_Control *cpu_self, Thread_queue_Context *queue_context)

  *Does nothing.*
- void _Thread_queue_Deadlock_status (Thread_Control *the_thread)

  *Sets the thread wait return code to STATUS_DEADLOCK.*
- void _Thread_queue_Deadlock_fatal (Thread_Control *the_thread)

  *Results in an INTERNAL_ERROR_THREAD_QUEUE_DEADLOCK fatal error.*
- void _Thread_queue_Enqueue (Thread_queue_Queue *queue, const Thread_queue_Operations *operations, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Blocks the thread and places it on the thread queue.*
- Status_Control _Thread_queue_Enqueue_sticky (Thread_queue_Queue *queue, const Thread_queue_Operations *operations, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Enqueues the thread on the thread queue and busy waits for dequeue.*
- static bool **_Thread_queue_Make_ready_again** (Thread_Control *the_thread)
- bool _Thread_queue_Extract_locked (Thread_queue_Queue *queue, const Thread_queue_Operations *operations, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Extracts the thread from the thread queue, restores the default wait operations and restores the default thread lock.*
- void _Thread_queue_Unblock_critical (bool unblock, Thread_queue_Queue *queue, Thread_Control *the_thread, ISR_lock_Context *lock_context)

  *Unblocks the thread which was on the thread queue before.*
- void _Thread_queue_Extract_critical (Thread_queue_Queue *queue, const Thread_queue_Operations *operations, Thread_Control *the_thread, Thread_queue_Context *queue_context)

  *Extracts the thread from the thread queue and unblocks it.*
- void _Thread_queue_Extract (Thread_Control *the_thread)

  *Extracts thread from thread queue.*

- void _Thread_queue_Surrender (Thread_queue_Queue *queue, Thread_queue_Heads *heads, Thread_Control *previous_owner, Thread_queue_Context *queue_context, const Thread_queue_Operations *operations)

  *Surrenders the thread queue previously owned by the thread to the first enqueued thread.*
- void _Thread_queue_Surrender_sticky (Thread_queue_Queue *queue, Thread_queue_Heads *heads, Thread_Control *previous_owner, Thread_queue_Context *queue_context, const Thread_queue_Operations *operations)

  *Surrenders the thread queue previously owned by the thread to the first enqueued thread.*
- Thread_Control * _Thread_queue_Do_dequeue (Thread_queue_Control *the_thread_queue, const Thread_queue_Operations *operations)

  *Dequeues the first thread waiting on the thread queue and returns it.*

## Variables

- static Thread_queue_Links **_Thread_queue_Links**

### 10.393.1 Detailed Description

Thread Queue Operations.

### 10.393.2 Variable Documentation

#### 10.393.2.1 _Thread_queue_Links

```
Thread_queue_Links _Thread_queue_Links  [static]
```

**Initial value:**

```
= {
  ISR_LOCK_INITIALIZER( "Thread Queue Links" ),
}
```

Definition at line 56 of file threadqenqueue.c.

## 10.394 cpukit/score/src/threadqflush.c File Reference

Thread Queue Flush.

```
#include <rtems/score/threadimpl.h>
#include <rtems/score/schedulerimpl.h>
#include <rtems/score/status.h>
```

### Functions

- Thread_Control * _Thread_queue_Flush_default_filter (Thread_Control *the_thread, Thread_queue_Queue *queue, Thread_queue_Context *queue_context)

  *Default thread queue flush filter function.*
- Thread_Control * _Thread_queue_Flush_status_object_was_deleted (Thread_Control *the_thread, Thread_queue_Queue *queue, Thread_queue_Context *queue_context)

  *Status object was deleted thread queue flush filter function.*
- Thread_Control * _Thread_queue_Flush_status_unavailable (Thread_Control *the_thread, Thread_queue_Queue *queue, Thread_queue_Context *queue_context)

  *Status unavailable thread queue flush filter function.*
- size_t _Thread_queue_Flush_critical (Thread_queue_Queue *queue, const Thread_queue_Operations *operations, Thread_queue_Flush_filter filter, Thread_queue_Context *queue_context)

  *Unblocks all threads enqueued on the thread queue.*

#### 10.394.1 Detailed Description

Thread Queue Flush.

## 10.395 cpukit/score/src/threadrestart.c File Reference

Restart Thread.

```
#include <rtems/score/threadimpl.h>
#include <rtems/score/apimutex.h>
#include <rtems/score/assert.h>
#include <rtems/score/chainimpl.h>
#include <rtems/score/freechainimpl.h>
#include <rtems/score/isrlock.h>
#include <rtems/score/schedulerimpl.h>
#include <rtems/score/stackimpl.h>
#include <rtems/score/sysstate.h>
#include <rtems/score/threadqimpl.h>
#include <rtems/score/userextimpl.h>
#include <rtems/score/watchdogimpl.h>
#include <rtems/score/wkspace.h>
```

### Classes

- struct Thread_Zombie_control
- struct Thread_Join_context

### Macros

- #define **THREAD_JOIN_TQ_OPERATIONS** &_Thread_queue_Operations_priority

## Functions

- static void **_Thread_Life_action_handler** (Thread_Control *executing, Thread_Action *action, ISR_lock_Context *lock_context)
- static void **_Thread_Raise_real_priority** (Thread_Control *the_thread, Priority_Control priority)
- static Thread_Control * **_Thread_Join_flush_filter** (Thread_Control *the_thread, Thread_queue_Queue *queue, Thread_queue_Context *queue_context)
- static void **_Thread_Wake_up_joining_threads** (Thread_Control *the_thread)
- static void **_Thread_Add_to_zombie_chain** (Thread_Control *the_thread)
- static void **_Thread_Make_zombie** (Thread_Control *the_thread)
- static void **_Thread_Free** (Thread_Control *the_thread)
- static void **_Thread_Wait_for_execution_stop** (Thread_Control *the_thread)
- void _Thread_Kill_zombies (void)

  *Kills all zombie threads in the system.*
- static Thread_Life_state **_Thread_Change_life_locked** (Thread_Control *the_thread, Thread_Life_state clear, Thread_Life_state set, Thread_Life_state ignore)
- static Per_CPU_Control * **_Thread_Wait_for_join** (Thread_Control *executing, Per_CPU_Control *cpu_self)
- static void **_Thread_Add_life_change_request** (Thread_Control *the_thread)
- static void **_Thread_Remove_life_change_request** (Thread_Control *the_thread)
- static void **_Thread_Finalize_life_change** (Thread_Control *the_thread, Priority_Control priority)
- void _Thread_Join (Thread_Control *the_thread, States_Control waiting_for_join, Thread_Control *executing, Thread_queue_Context *queue_context)

  *Joins the currently executing thread with the given thread to wait for.*
- static void **_Thread_Set_exit_value** (Thread_Control *the_thread, void *exit_value)
- void _Thread_Cancel (Thread_Control *the_thread, Thread_Control *executing, void *exit_value)

  *Cancels the thread.*
- static void **_Thread_Close_enqueue_callout** (Thread_queue_Queue *queue, Thread_Control *the_thread, Per_CPU_Control *cpu_self, Thread_queue_Context *queue_context)
- void _Thread_Close (Thread_Control *the_thread, Thread_Control *executing, Thread_Close_context *context)

  *Closes the thread.*
- void _Thread_Exit (Thread_Control *executing, Thread_Life_state set, void *exit_value)

  *Exits the currently executing thread.*
- bool _Thread_Restart_other (Thread_Control *the_thread, const Thread_Entry_information *entry, ISR_lock_Context *lock_context)

  *Restarts the thread.*
- void _Thread_Restart_self (Thread_Control *executing, const Thread_Entry_information *entry, ISR_lock_Context *lock_context)

  *Restarts the currently executing thread.*
- Thread_Life_state _Thread_Change_life (Thread_Life_state clear, Thread_Life_state set, Thread_Life_state ignore)

  *Changes the currently executing thread to a new state with the sets.*
- Thread_Life_state _Thread_Set_life_protection (Thread_Life_state state)

  *Set the thread to life protected.*

## Variables

- static Thread_Zombie_control **_Thread_Zombies**

### 10.395.1 Detailed Description

Restart Thread.

### 10.395.2 Variable Documentation

#### 10.395.2.1 _Thread_Zombies

```
Thread_Zombie_control _Thread_Zombies  [static]
```

**Initial value:**

```
= {
  .Chain = CHAIN_INITIALIZER_EMPTY( _Thread_Zombies.Chain ),
  .Lock = ISR_LOCK_INITIALIZER( "thread zombies" )
}
```

Definition at line 51 of file threadrestart.c.

## 10.396 cpukit/score/src/threadsetstate.c File Reference

Sets States for a Thread.

```
#include <rtems/score/threadimpl.h>
#include <rtems/score/assert.h>
#include <rtems/score/schedulerimpl.h>
```

### Functions

- States_Control _Thread_Set_state_locked (Thread_Control *the_thread, States_Control state)

  *Sets the specified thread state without locking the lock context.*
- States_Control _Thread_Set_state (Thread_Control *the_thread, States_Control state)

  *Sets the specified thread state.*

### 10.396.1 Detailed Description

Sets States for a Thread.

## 10.397 cpukit/score/src/threadstackallocate.c File Reference

Stack Allocate Helper.

```
#include <rtems/score/stackimpl.h>
#include <rtems/config.h>
```

### Functions

- void * _Stack_Allocate (size_t stack_size)

  *Allocate the requested stack space.*

### 10.397.1 Detailed Description

Stack Allocate Helper.

## 10.398 cpukit/score/src/threadstart.c File Reference

Initializes Thread and Executes it.

```
#include <rtems/score/threadimpl.h>
#include <rtems/score/isrlevel.h>
#include <rtems/score/userextimpl.h>
```

### Functions

- bool _Thread_Start (Thread_Control *the_thread, const Thread_Entry_information *entry, ISR_lock_Context *lock_context)

  *Starts the specified thread.*

### 10.398.1 Detailed Description

Initializes Thread and Executes it.

## 10.399 cpukit/score/src/threadstartmultitasking.c File Reference

Start Thread Multitasking.

```
#include <rtems/score/threadimpl.h>
#include <rtems/score/assert.h>
```

### Functions

- void _Thread_Start_multitasking (void)

  *Starts thread multitasking.*

### 10.399.1 Detailed Description

Start Thread Multitasking.

## 10.400 cpukit/score/src/threadtimeout.c File Reference

Thread Wait Timeout.

```
#include <rtems/score/threadimpl.h>
#include <rtems/score/status.h>
```

### Functions

- void _Thread_Continue (Thread_Control *the_thread, Status_Control status)

  *Cancels a blocking operation so that the thread can continue its execution.*
- void _Thread_Timeout (Watchdog_Control *the_watchdog)

  *General purpose thread wait timeout.*

### 10.400.1 Detailed Description

Thread Wait Timeout.

## 10.401 cpukit/score/src/threadyield.c File Reference

Thread Yield.

```
#include <rtems/score/threadimpl.h>
#include <rtems/score/schedulerimpl.h>
```

### Functions

- void _Thread_Yield (Thread_Control *executing)

  *Yields the currently executing thread.*

### 10.401.1 Detailed Description

Thread Yield.

## 10.402 cpukit/score/src/userext.c File Reference

User Extension Handler implementation.

```
#include <rtems/score/userextimpl.h>
```

### Functions

- void _User_extensions_Handler_initialization (void)

  *Initializes the user extensions handler.*

### 10.402.1 Detailed Description

User Extension Handler implementation.

## 10.403 cpukit/score/src/userextaddset.c File Reference

User Extension Handler implementation.

```
#include <rtems/score/userextimpl.h>
#include <rtems/score/smp.h>
#include <rtems/score/percpu.h>
```

### Functions

- static void **_User_extensions_Set_ancestors** (void)
- void _User_extensions_Add_set (User_extensions_Control *the_extension)

  *Adds a user extension.*

### 10.403.1 Detailed Description

User Extension Handler implementation.

## 10.404 cpukit/score/src/userextiterate.c File Reference

User Extension Iteration Helpers.

```
#include <rtems/score/userextimpl.h>
#include <pthread.h>
```

### Functions

- void _User_extensions_Thread_create_visitor (Thread_Control *executing, void *arg, const User_extensions_Table *callouts)

  *Creates a visitor.*
- void _User_extensions_Thread_delete_visitor (Thread_Control *executing, void *arg, const User_extensions_Table *callouts)

  *Deletes a visitor.*
- void _User_extensions_Thread_start_visitor (Thread_Control *executing, void *arg, const User_extensions_Table *callouts)

  *Starts a visitor.*
- void _User_extensions_Thread_restart_visitor (Thread_Control *executing, void *arg, const User_extensions_Table *callouts)

  *Restarts a visitor.*
- void _User_extensions_Thread_begin_visitor (Thread_Control *executing, void *arg, const User_extensions_Table *callouts)

  *Calls the begin function of the thread callout for the visitor.*
- void _User_extensions_Thread_exitted_visitor (Thread_Control *executing, void *arg, const User_extensions_Table *callouts)

  *Calls the exitted function of the thread callout for the visitor.*
- void _User_extensions_Fatal_visitor (Thread_Control *executing, void *arg, const User_extensions_Table *callouts)

  *Calls the fatal function of the thread callout for the visitor.*
- void _User_extensions_Thread_terminate_visitor (Thread_Control *executing, void *arg, const User_extensions_Table *callouts)

  *Terminates a visitor.*
- void _User_extensions_Iterate (void *arg, User_extensions_Visitor visitor, Chain_Iterator_direction direction)

  *Iterates through all user extensions and calls the visitor for each.*

### Variables

- User_extensions_List _User_extensions_List

  *List of active extensions.*

#### 10.404.1 Detailed Description

User Extension Iteration Helpers.

## 10.405 cpukit/score/src/userextremoveset.c File Reference

User Extension Handler implementation.

```
#include <rtems/score/userextimpl.h>
#include <rtems/score/percpu.h>
```

### Functions

- void _User_extensions_Remove_set (User_extensions_Control *the_extension)

  *Removes a user extension.*

### 10.405.1 Detailed Description

User Extension Handler implementation.

## 10.406 cpukit/score/src/watchdoginsert.c File Reference

Watchdog Insert.

```
#include <rtems/score/watchdogimpl.h>
```

### Functions

- void _Watchdog_Insert (Watchdog_Header *header, Watchdog_Control *the_watchdog, uint64_t expire)
  *Inserts a watchdog into the set of scheduled watchdogs according to the specified expiration time.*

### 10.406.1 Detailed Description

Watchdog Insert.

## 10.407 cpukit/score/src/watchdogremove.c File Reference

Remove Watchdog.

```
#include <rtems/score/watchdogimpl.h>
```

### Functions

- void _Watchdog_Remove (Watchdog_Header *header, Watchdog_Control *the_watchdog)
  *In the case the watchdog is scheduled, then it is removed from the set of scheduled watchdogs.*

### 10.407.1 Detailed Description

Remove Watchdog.

## 10.408 cpukit/score/src/watchdogtickssinceboot.c File Reference

Watchdog Ticks Since Boot.

```
#include <rtems/score/watchdogticks.h>
```

### Variables

- volatile Watchdog_Interval _Watchdog_Ticks_since_boot

  *The watchdog ticks counter.*

### 10.408.1 Detailed Description

Watchdog Ticks Since Boot.

## 10.409 cpukit/score/src/wkspace.c File Reference

Workspace Handler System Initialization.

```
#include <rtems/score/wkspace.h>
#include <rtems/sysinit.h>
```

### Enumerations

- enum {
  **_Sysinit_bsp_start** = 0x00030080, **_Sysinit_bsp_debug_uart_init** = 0x00030003, **_Sysinit_amba_↩ initialize** = 0x00030001, **_Sysinit_leon3_cpu_index_init** = 0x00030000,
  **_Sysinit_leon3_counter_initialize** = 0x00040000, **_Sysinit_bsp_memory_initialize** = 0x00018080, **_↩ Sysinit__Malloc_Initialize** = 0x00028080, **_Sysinit__Barrier_Manager_initialization** = 0x00140080,
  **_Sysinit__Message_queue_Manager_initialization** = 0x000e0080, **_Sysinit__Partition_Manager_↩ initialization** = 0x00100080, **_Sysinit__Rate_monotonic_Manager_initialization** = 0x00130080, **_↩ Sysinit__Semaphore_Manager_initialization** = 0x000f0080,
  **_Sysinit__RTEMS_tasks_Manager_initialization** = 0x000a0080, **_Sysinit__Timer_Manager_↩ initialization** = 0x000b0080, **_Sysinit_rtems_initialize_data_structures** = 0x00070080, **_Sysinit__↩ Extension_Manager_initialization** = 0x00090080,
  **_Sysinit__Per_CPU_Data_initialize** = 0x00022080, **_Sysinit__Workspace_Handler_initialization** = 0x00026080, **_Sysinit_init_task** = 0x00290080, **_Sysinit_init_runner_task** = 0x00290080 }

### Variables

- Heap_Control _Workspace_Area

  *Executive workspace control.*
- rtems_sysinit_item const **_Linker_set__Sysinit__Workspace_Handler_initialization** = { _Workspace_Handler_initialization }

### 10.409.1 Detailed Description

Workspace Handler System Initialization.

# 10.410 cpukit/score/src/wkspaceallocate.c File Reference

_Workspace_Allocate() Implementation

```
#include <rtems/score/wkspace.h>
#include <rtems/score/heapimpl.h>
```

## Functions

- void * _Workspace_Allocate (size_t size)

  *Allocates a memory block of the specified size from the workspace.*

### 10.410.1 Detailed Description

_Workspace_Allocate() Implementation

# 10.411 cpukit/score/src/wkspacemallocinitdefault.c File Reference

This source file provides the default definition of _Workspace_Malloc_initializer.

```
#include <rtems/score/wkspacedata.h>
```

## Variables

- struct Heap_Control *(*const _Workspace_Malloc_initializer )(void)

  *This constant provides the C Program Heap initialization handler.*

### 10.411.1 Detailed Description

This source file provides the default definition of _Workspace_Malloc_initializer.

# 10.412 cpukit/score/src/wkspacemallocinitunified.c File Reference

This source file provides the implementation of _Workspace_Malloc_initialize_unified().

```
#include <rtems/score/wkspacedata.h>
#include <rtems/score/wkspace.h>
```

## Functions

- Heap_Control * _Workspace_Malloc_initialize_unified (void)

  *Initializes the C Program Heap so that it is unified with the RTEMS Workspace.*

### 10.412.1 Detailed Description

This source file provides the implementation of _Workspace_Malloc_initialize_unified().

## 10.413 testsuites/validation/tc-attr.c File Reference

```
#include <rtems.h>
#include <rtems/test.h>
```

### Functions

- static bool **IsPowerOfTwo** (rtems_attribute attribute)
- static int **PopCount** (rtems_attribute attributes)
- void **T_case_body_RtemsAttrValAttr** (void)

## 10.414 testsuites/validation/tc-barrier-ident.c File Reference

```
#include "tr-object-ident-local.h"
#include <rtems/test.h>
```

### Functions

- static rtems_status_code **ClassicBarrierIdentAction** (rtems_name name, rtems_id *id)
- void **T_case_body_RtemsBarrierValIdent** (void)

## 10.415 testsuites/validation/tc-event-send-receive.c File Reference

```
#include <rtems/rtems/eventimpl.h>
#include <rtems/rtems/tasksdata.h>
#include <rtems/score/statesimpl.h>
#include <rtems/score/threadimpl.h>
#include "tr-event-send-receive.h"
#include <rtems/test.h>
```

### Functions

- static rtems_status_code **EventSend** (rtems_id id, rtems_event_set event_in)
- static rtems_status_code **EventReceive** (rtems_id event_in, rtems_option option_set, rtems_interval ticks, rtems_event_set *event_out)
- static rtems_event_set **GetPendingEvents** (Thread_Control *thread)
- void **T_case_body_RtemsEventValSendReceive** (void)
- static rtems_status_code **EventSystemSend** (rtems_id id, rtems_event_set event_in)
- static rtems_status_code **EventSystemReceive** (rtems_id event_in, rtems_option option_set, rtems_interval ticks, rtems_event_set *event_out)
- static rtems_event_set **GetPendingSystemEvents** (Thread_Control *thread)
- void **T_case_body_RtemsEventValSystemSendReceive** (void)

## 10.416 testsuites/validation/tc-events.c File Reference

```
#include <rtems.h>
#include "tr-event-constant.h"
#include <rtems/test.h>
```

### Functions

- void **T_case_body_RtemsEventValEvents** (void)

### Variables

- static const rtems_event_set **events** [ ]

## 10.417 testsuites/validation/tc-message-construct-errors.c File Reference

```
#include <rtems.h>
#include <string.h>
#include <rtems/score/chainimpl.h>
#include <rtems/score/objectimpl.h>
#include <rtems/test.h>
```

### Classes

- struct RtemsMessageReqConstructErrors_Context

  *Test context for spec:/rtems/message/req/construct-errors test case.*

### Macros

- #define **MAX_MESSAGE_QUEUES** 4
- #define **MAX_PENDING_MESSAGES** 1
- #define **MAX_MESSAGE_SIZE** 1

## Enumerations

- enum **RtemsMessageReqConstructErrors_Pre_Id** { **RtemsMessageReqConstructErrors_Pre_Id_Id**, **RtemsMessageReqConstructErrors_Pre_Id_Null**, **RtemsMessageReqConstructErrors_Pre_Id_NA** }
- enum **RtemsMessageReqConstructErrors_Pre_Name** { **RtemsMessageReqConstructErrors_Pre_Name_Valid**, **RtemsMessageReqConstructErrors_Pre_Name_Invalid**, **RtemsMessageReqConstructErrors_Pre_Name_NA** }
- enum **RtemsMessageReqConstructErrors_Pre_MaxPending** { **RtemsMessageReqConstructErrors_Pre_MaxPending_Valid**, **RtemsMessageReqConstructErrors_Pre_MaxPending_Zero**, **RtemsMessageReqConstructErrors_Pre_MaxPending_Big**, **RtemsMessageReqConstructErrors_Pre_MaxPending_NA** }
- enum **RtemsMessageReqConstructErrors_Pre_MaxSize** { **RtemsMessageReqConstructErrors_Pre_MaxSize_Valid**, **RtemsMessageReqConstructErrors_Pre_MaxSize_Zero**, **RtemsMessageReqConstructErrors_Pre_MaxSize_Big**, **RtemsMessageReqConstructErrors_Pre_MaxSize_NA** }
- enum **RtemsMessageReqConstructErrors_Pre_Queues** { **RtemsMessageReqConstructErrors_Pre_Queues_Avail**, **RtemsMessageReqConstructErrors_Pre_Queues_None**, **RtemsMessageReqConstructErrors_Pre_Queues_NA** }
- enum **RtemsMessageReqConstructErrors_Pre_Area** { **RtemsMessageReqConstructErrors_Pre_Area_Valid**, **RtemsMessageReqConstructErrors_Pre_Area_Null**, **RtemsMessageReqConstructErrors_Pre_Area_NA** }
- enum **RtemsMessageReqConstructErrors_Pre_AreaSize** { **RtemsMessageReqConstructErrors_Pre_AreaSize_Valid**, **RtemsMessageReqConstructErrors_Pre_AreaSize_Invalid**, **RtemsMessageReqConstructErrors_Pre_AreaSize_NA** }
- enum **RtemsMessageReqConstructErrors_Post_Status** { **RtemsMessageReqConstructErrors_Post_Status_Ok**, **RtemsMessageReqConstructErrors_Post_Status_InvAddress**, **RtemsMessageReqConstructErrors_Post_Status_InvName**, **RtemsMessageReqConstructErrors_Post_Status_InvNumber**, **RtemsMessageReqConstructErrors_Post_Status_InvSize**, **RtemsMessageReqConstructErrors_Post_Status_TooMany**, **RtemsMessageReqConstructErrors_Post_Status_Unsatisfied**, **RtemsMessageReqConstructErrors_Post_Status_NA** }

## Functions

- static **RTEMS_MESSAGE_QUEUE_BUFFER** (static RTEMS_MESSAGE_QUEUE_BUFFER MAX_MESSAGE_SIZE)
- static void **RtemsMessageReqConstructErrors_Pre_Name_Prepare** (RtemsMessageReqConstructErrors_Context *ctx, RtemsMessageReqConstructErrors_Pre_Name state)
- static void **RtemsMessageReqConstructErrors_Pre_MaxPending_Prepare** (RtemsMessageReqConstructErrors_Context *ctx, RtemsMessageReqConstructErrors_Pre_MaxPending state)
- static void **RtemsMessageReqConstructErrors_Pre_MaxSize_Prepare** (RtemsMessageReqConstructErrors_Context *ctx, RtemsMessageReqConstructErrors_Pre_MaxSize state)
- static void **RtemsMessageReqConstructErrors_Pre_Queues_Prepare** (RtemsMessageReqConstructErrors_Context *ctx, RtemsMessageReqConstructErrors_Pre_Queues state)
- static void **RtemsMessageReqConstructErrors_Pre_Area_Prepare** (RtemsMessageReqConstructErrors_Context *ctx, RtemsMessageReqConstructErrors_Pre_Area state)
- static void **RtemsMessageReqConstructErrors_Pre_AreaSize_Prepare** (RtemsMessageReqConstructErrors_Context *ctx, RtemsMessageReqConstructErrors_Pre_AreaSize state)
- static void **RtemsMessageReqConstructErrors_Post_Status_Check** (RtemsMessageReqConstructErrors_Context *ctx, RtemsMessageReqConstructErrors_Post_Status state)
- static void **RtemsMessageReqConstructErrors_Setup** (RtemsMessageReqConstructErrors_Context *ctx)
- static void **RtemsMessageReqConstructErrors_Setup_Wrap** (void *arg)
- static size_t **RtemsMessageReqConstructErrors_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsMessageReqConstructErrors_Prepare** (RtemsMessageReqConstructErrors_Context *ctx)
- static void **RtemsMessageReqConstructErrors_Action** (RtemsMessageReqConstructErrors_Context *ctx)
- static void **RtemsMessageReqConstructErrors_Cleanup** (RtemsMessageReqConstructErrors_Context *ctx)
- **T_TEST_CASE_FIXTURE** (RtemsMessageReqConstructErrors, &RtemsMessageReqConstructErrors_Fixture)

### Variables

- static RtemsMessageReqConstructErrors_Context **RtemsMessageReqConstructErrors_Instance**
- static const char *const **RtemsMessageReqConstructErrors_PreDesc_Id** [ ]
- static const char *const **RtemsMessageReqConstructErrors_PreDesc_Name** [ ]
- static const char *const **RtemsMessageReqConstructErrors_PreDesc_MaxPending** [ ]
- static const char *const **RtemsMessageReqConstructErrors_PreDesc_MaxSize** [ ]
- static const char *const **RtemsMessageReqConstructErrors_PreDesc_Queues** [ ]
- static const char *const **RtemsMessageReqConstructErrors_PreDesc_Area** [ ]
- static const char *const **RtemsMessageReqConstructErrors_PreDesc_AreaSize** [ ]
- static const char *const *const **RtemsMessageReqConstructErrors_PreDesc** [ ]
- static T_fixture **RtemsMessageReqConstructErrors_Fixture**
- static const uint8_t **RtemsMessageReqConstructErrors_TransitionMap** [ ][1]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1
  uint16_t **Pre_Id_NA**: 1
  uint16_t **Pre_Name_NA**: 1
  uint16_t **Pre_Prio_NA**: 1
  uint16_t **Pre_Tasks_NA**: 1
  uint16_t **Pre_TLS_NA**: 1
  uint16_t **Pre_Stack_NA**: 1
  uint16_t **Pre_Ext_NA**: 1
  uint16_t **Pre_Preempt_NA**: 1
  uint8_t **Pre_Pre_NA**: 1
  uint8_t **Pre_Send_NA**: 1
  uint8_t **Pre_ReceiverState_NA**: 1
  uint8_t **Pre_Satisfy_NA**: 1
  uint8_t **Pre_Node_NA**: 1
  } **RtemsMessageReqConstructErrors_TransitionInfo** [ ]

## 10.418 testsuites/validation/tc-message-ident.c File Reference

```
#include "tr-object-ident.h"
#include <rtems/test.h>
```

### Functions

- static **RTEMS_MESSAGE_QUEUE_BUFFER** (1)
- static rtems_status_code **ClassicMessageIdentAction** (rtems_name name, uint32_t node, rtems_id *id)
- void **T_case_body_RtemsMessageValIdent** (void)

## 10.419 testsuites/validation/tc-modes.c File Reference

```
#include <rtems.h>
#include <rtems/test.h>
```

### Functions

- static bool **IsPowerOfTwo** (rtems_mode mode)
- static int **PopCount** (rtems_mode modes)
- void **T_case_body_RtemsModeValModes** (void)

## 10.420 testsuites/validation/tc-options.c File Reference

```
#include <rtems.h>
#include <rtems/test.h>
```

### Functions

- static bool **IsPowerOfTwo** (rtems_option option)
- static int **PopCount** (rtems_option options)
- void **T_case_body_RtemsOptionValOptions** (void)

## 10.421 testsuites/validation/tc-part-create.c File Reference

```
#include <rtems.h>
#include <string.h>
#include <rtems/score/chainimpl.h>
#include <rtems/score/objectimpl.h>
#include <rtems/test.h>
```

### Classes

- struct RtemsPartReqCreate_Context

  *Test context for spec:/rtems/part/req/create test case.*

### Macros

- #define **PART_NAME** rtems_build_name( 'N', 'A', 'M', 'E' )
- #define **MAX_PARTITIONS** 4
- #define **BUFFER_COUNT** 2
- #define **BUFFER_SIZE** ( 2 * sizeof( void * ) )

## Enumerations

- enum **RtemsPartReqCreate_Pre_Id** { **RtemsPartReqCreate_Pre_Id_Id**, **RtemsPartReqCreate_Pre_Id_Null**, **RtemsPartReqCreate_Pre_Id_NA** }
- enum **RtemsPartReqCreate_Pre_Name** { **RtemsPartReqCreate_Pre_Name_Valid**, **RtemsPartReqCreate_Pre_Name_Invalid**, **RtemsPartReqCreate_Pre_Name_NA** }
- enum **RtemsPartReqCreate_Pre_Start** { **RtemsPartReqCreate_Pre_Start_Valid**, **RtemsPartReqCreate_Pre_Start_Null**, **RtemsPartReqCreate_Pre_Start_BadAlign**, **RtemsPartReqCreate_Pre_Start_NA** }
- enum **RtemsPartReqCreate_Pre_Length** { **RtemsPartReqCreate_Pre_Length_Valid**, **RtemsPartReqCreate_Pre_Length_Zero**, **RtemsPartReqCreate_Pre_Length_Invalid**, **RtemsPartReqCreate_Pre_Length_NA** }
- enum **RtemsPartReqCreate_Pre_Size** { **RtemsPartReqCreate_Pre_Size_Valid**, **RtemsPartReqCreate_Pre_Size_Zero**, **RtemsPartReqCreate_Pre_Size_Small**, **RtemsPartReqCreate_Pre_Size_NA** }
- enum **RtemsPartReqCreate_Pre_Parts** { **RtemsPartReqCreate_Pre_Parts_Avail**, **RtemsPartReqCreate_Pre_Parts_None**, **RtemsPartReqCreate_Pre_Parts_NA** }
- enum **RtemsPartReqCreate_Post_Status** { **RtemsPartReqCreate_Post_Status_Ok**, **RtemsPartReqCreate_Post_Status_InvAddress**, **RtemsPartReqCreate_Post_Status_InvName**, **RtemsPartReqCreate_Post_Status_InvNumber**, **RtemsPartReqCreate_Post_Status_InvSize**, **RtemsPartReqCreate_Post_Status_TooMany**, **RtemsPartReqCreate_Post_Status_NA** }

## Functions

- static **RTEMS_ALIGNED** (static RTEMS_ALIGNED RTEMS_PARTITION_ALIGNMENT)
- static void **RtemsPartReqCreate_Pre_Name_Prepare** (RtemsPartReqCreate_Context *ctx, RtemsPartReqCreate_Pre_Name state)
- static void **RtemsPartReqCreate_Pre_Start_Prepare** (RtemsPartReqCreate_Context *ctx, RtemsPartReqCreate_Pre_Start state)
- static void **RtemsPartReqCreate_Pre_Length_Prepare** (RtemsPartReqCreate_Context *ctx, RtemsPartReqCreate_Pre_Length state)
- static void **RtemsPartReqCreate_Pre_Size_Prepare** (RtemsPartReqCreate_Context *ctx, RtemsPartReqCreate_Pre_Size state)
- static void **RtemsPartReqCreate_Pre_Parts_Prepare** (RtemsPartReqCreate_Context *ctx, RtemsPartReqCreate_Pre_Parts state)
- static void **RtemsPartReqCreate_Post_Status_Check** (RtemsPartReqCreate_Context *ctx, RtemsPartReqCreate_Post_Status state)
- static void **RtemsPartReqCreate_Setup** (RtemsPartReqCreate_Context *ctx)
- static void **RtemsPartReqCreate_Setup_Wrap** (void *arg)
- static size_t **RtemsPartReqCreate_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsPartReqCreate_Prepare** (RtemsPartReqCreate_Context *ctx)
- static void **RtemsPartReqCreate_Action** (RtemsPartReqCreate_Context *ctx)
- static void **RtemsPartReqCreate_Cleanup** (RtemsPartReqCreate_Context *ctx)
- **T_TEST_CASE_FIXTURE** (RtemsPartReqCreate, &RtemsPartReqCreate_Fixture)

## Variables

- static RtemsPartReqCreate_Context **RtemsPartReqCreate_Instance**
- static const char *const **RtemsPartReqCreate_PreDesc_Id** [ ]
- static const char *const **RtemsPartReqCreate_PreDesc_Name** [ ]
- static const char *const **RtemsPartReqCreate_PreDesc_Start** [ ]
- static const char *const **RtemsPartReqCreate_PreDesc_Length** [ ]
- static const char *const **RtemsPartReqCreate_PreDesc_Size** [ ]
- static const char *const **RtemsPartReqCreate_PreDesc_Parts** [ ]

- static const char *const *const **RtemsPartReqCreate_PreDesc** [ ]
- static T_fixture **RtemsPartReqCreate_Fixture**
- static const uint8_t **RtemsPartReqCreate_TransitionMap** [ ][1]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1
  uint16_t **Pre_Id_NA**: 1
  uint16_t **Pre_Name_NA**: 1
  uint16_t **Pre_Prio_NA**: 1
  uint16_t **Pre_Tasks_NA**: 1
  uint16_t **Pre_TLS_NA**: 1
  uint16_t **Pre_Stack_NA**: 1
  uint16_t **Pre_Ext_NA**: 1
  uint16_t **Pre_Preempt_NA**: 1
  uint8_t **Pre_Pre_NA**: 1
  uint8_t **Pre_Send_NA**: 1
  uint8_t **Pre_ReceiverState_NA**: 1
  uint8_t **Pre_Satisfy_NA**: 1
  uint8_t **Pre_Node_NA**: 1
  } **RtemsPartReqCreate_TransitionInfo** [ ]

## 10.422 testsuites/validation/tc-part-delete.c File Reference

```
#include <rtems.h>
#include <rtems/test.h>
```

### Classes

- struct RtemsPartReqDelete_Context

  *Test context for spec:/rtems/part/req/delete test case.*

### Macros

- #define **PART_NAME** rtems_build_name( 'N', 'A', 'M', 'E' )
- #define **BUFFER_COUNT** 1
- #define **BUFFER_SIZE** ( 2 * sizeof( void * ) )

## Enumerations

- enum **RtemsPartReqDelete_Pre_Id** { **RtemsPartReqDelete_Pre_Id_Id**, **RtemsPartReqDelete_Pre_Id_Invalid**, **RtemsPartReqDelete_Pre_Id_NA** }
- enum **RtemsPartReqDelete_Pre_InUse** { **RtemsPartReqDelete_Pre_InUse_Yes**, **RtemsPartReqDelete_Pre_InUse_No**, **RtemsPartReqDelete_Pre_InUse_NA** }
- enum **RtemsPartReqDelete_Post_Status** { **RtemsPartReqDelete_Post_Status_Ok**, **RtemsPartReqDelete_Post_Status_InvId**, **RtemsPartReqDelete_Post_Status_InUse**, **RtemsPartReqDelete_Post_Status_NA** }

## Functions

- static **RTEMS_ALIGNED** (RTEMS_PARTITION_ALIGNMENT)
- static void **RtemsPartReqDelete_Pre_InUse_Prepare** (RtemsPartReqDelete_Context *ctx, RtemsPartReqDelete_Pre_InUse state)
- static void **RtemsPartReqDelete_Post_Status_Check** (RtemsPartReqDelete_Context *ctx, RtemsPartReqDelete_Post_Status state)
- static size_t **RtemsPartReqDelete_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsPartReqDelete_Prepare** (RtemsPartReqDelete_Context *ctx)
- static void **RtemsPartReqDelete_Action** (RtemsPartReqDelete_Context *ctx)
- static void **RtemsPartReqDelete_Cleanup** (RtemsPartReqDelete_Context *ctx)
- **T_TEST_CASE_FIXTURE** (RtemsPartReqDelete, &RtemsPartReqDelete_Fixture)

## Variables

- static RtemsPartReqDelete_Context **RtemsPartReqDelete_Instance**
- static const char *const **RtemsPartReqDelete_PreDesc_Id** [ ]
- static const char *const **RtemsPartReqDelete_PreDesc_InUse** [ ]
- static const char *const *const **RtemsPartReqDelete_PreDesc** [ ]
- static T_fixture **RtemsPartReqDelete_Fixture**
- static const uint8_t **RtemsPartReqDelete_TransitionMap** [ ][1]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1
  uint16_t **Pre_Id_NA**: 1
  uint16_t **Pre_Name_NA**: 1
  uint16_t **Pre_Prio_NA**: 1
  uint16_t **Pre_Tasks_NA**: 1
  uint16_t **Pre_TLS_NA**: 1
  uint16_t **Pre_Stack_NA**: 1
  uint16_t **Pre_Ext_NA**: 1

uint16_t **Pre_Preempt_NA**: 1
uint8_t **Pre_Pre_NA**: 1
uint8_t **Pre_Send_NA**: 1
uint8_t **Pre_ReceiverState_NA**: 1
uint8_t **Pre_Satisfy_NA**: 1
uint8_t **Pre_Node_NA**: 1
} **RtemsPartReqDelete_TransitionInfo** [ ]

## 10.423 testsuites/validation/tc-part-get.c File Reference

```
#include <rtems.h>
#include <rtems/test.h>
```

### Classes

- struct RtemsPartReqGetBuffer_Context

  *Test context for spec:/rtems/part/req/get-buffer test case.*

### Macros

- #define **BUFFER_COUNT** 1
- #define **BUFFER_SIZE** ( 2 * sizeof( void * ) )

### Enumerations

- enum **RtemsPartReqGetBuffer_Pre_Id** { **RtemsPartReqGetBuffer_Pre_Id_Id**, **RtemsPartReqGetBuffer_Pre_Id_Invalid**, **RtemsPartReqGetBuffer_Pre_Id_NA** }
- enum **RtemsPartReqGetBuffer_Pre_Buf** { **RtemsPartReqGetBuffer_Pre_Buf_Valid**, **RtemsPartReqGetBuffer_Pre_Buf_Null**, **RtemsPartReqGetBuffer_Pre_Buf_NA** }
- enum **RtemsPartReqGetBuffer_Pre_Avail** { **RtemsPartReqGetBuffer_Pre_Avail_Yes**, **RtemsPartReqGetBuffer_Pre_Avail_No**, **RtemsPartReqGetBuffer_Pre_Avail_NA** }
- enum **RtemsPartReqGetBuffer_Post_Status** { **RtemsPartReqGetBuffer_Post_Status_Ok**, **RtemsPartReqGetBuffer_Post_Status_InvId**, **RtemsPartReqGetBuffer_Post_Status_InvAddr**, **RtemsPartReqGetBuffer_Post_Status_Unsatisfied**, **RtemsPartReqGetBuffer_Post_Status_NA** }

### Functions

- static **RTEMS_ALIGNED** (RTEMS_PARTITION_ALIGNMENT)
- static void **RtemsPartReqGetBuffer_Pre_Buf_Prepare** (RtemsPartReqGetBuffer_Context *ctx, RtemsPartReqGetBuffer_Pre_Buf state)
- static void **RtemsPartReqGetBuffer_Pre_Avail_Prepare** (RtemsPartReqGetBuffer_Context *ctx, RtemsPartReqGetBuffer_Pre_Avail state)
- static void **RtemsPartReqGetBuffer_Post_Status_Check** (RtemsPartReqGetBuffer_Context *ctx, RtemsPartReqGetBuffer_Post_Status state)
- static void **RtemsPartReqGetBuffer_Setup** (RtemsPartReqGetBuffer_Context *ctx)
- static void **RtemsPartReqGetBuffer_Setup_Wrap** (void *arg)
- static void **RtemsPartReqGetBuffer_Teardown** (RtemsPartReqGetBuffer_Context *ctx)
- static void **RtemsPartReqGetBuffer_Teardown_Wrap** (void *arg)
- static size_t **RtemsPartReqGetBuffer_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsPartReqGetBuffer_Prepare** (RtemsPartReqGetBuffer_Context *ctx)
- static void **RtemsPartReqGetBuffer_Action** (RtemsPartReqGetBuffer_Context *ctx)
- static void **RtemsPartReqGetBuffer_Cleanup** (RtemsPartReqGetBuffer_Context *ctx)
- **T_TEST_CASE_FIXTURE** (RtemsPartReqGetBuffer, &RtemsPartReqGetBuffer_Fixture)

### Variables

- static RtemsPartReqGetBuffer_Context **RtemsPartReqGetBuffer_Instance**
- static const char *const **RtemsPartReqGetBuffer_PreDesc_Id** [ ]
- static const char *const **RtemsPartReqGetBuffer_PreDesc_Buf** [ ]
- static const char *const **RtemsPartReqGetBuffer_PreDesc_Avail** [ ]
- static const char *const *const **RtemsPartReqGetBuffer_PreDesc** [ ]
- static T_fixture **RtemsPartReqGetBuffer_Fixture**
- static const uint8_t **RtemsPartReqGetBuffer_TransitionMap** [ ][1]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1
  uint16_t **Pre_Id_NA**: 1
  uint16_t **Pre_Name_NA**: 1
  uint16_t **Pre_Prio_NA**: 1
  uint16_t **Pre_Tasks_NA**: 1
  uint16_t **Pre_TLS_NA**: 1
  uint16_t **Pre_Stack_NA**: 1
  uint16_t **Pre_Ext_NA**: 1
  uint16_t **Pre_Preempt_NA**: 1
  uint8_t **Pre_Pre_NA**: 1
  uint8_t **Pre_Send_NA**: 1
  uint8_t **Pre_ReceiverState_NA**: 1
  uint8_t **Pre_Satisfy_NA**: 1
  uint8_t **Pre_Node_NA**: 1
  } **RtemsPartReqGetBuffer_TransitionInfo** [ ]

## 10.424 testsuites/validation/tc-part-ident.c File Reference

```
#include "tr-object-ident.h"
#include <rtems/test.h>
```

### Functions

- static rtems_status_code **ClassicPartIdentAction** (rtems_name name, uint32_t node, rtems_id *id)
- void **T_case_body_RtemsPartValIdent** (void)

## 10.425 testsuites/validation/tc-part-return.c File Reference

```
#include <rtems.h>
#include <rtems/test.h>
```

### Classes

- struct RtemsPartReqReturnBuffer_Context

  *Test context for spec:/rtems/part/req/return-buffer test case.*

### Macros

- #define **BUFFER_COUNT** 1
- #define **BUFFER_SIZE** ( 2 * sizeof( void * ) )

### Enumerations

- enum **RtemsPartReqReturnBuffer_Pre_Id** { **RtemsPartReqReturnBuffer_Pre_Id_Id**, **RtemsPartReqReturnBuffer_Pre_Id_Invalid**, **RtemsPartReqReturnBuffer_Pre_Id_NA** }
- enum **RtemsPartReqReturnBuffer_Pre_Buf** { **RtemsPartReqReturnBuffer_Pre_Buf_Valid**, **RtemsPartReqReturnBuffer_Pre_Buf_Invalid**, **RtemsPartReqReturnBuffer_Pre_Buf_NA** }
- enum **RtemsPartReqReturnBuffer_Post_Status** { **RtemsPartReqReturnBuffer_Post_Status_Ok**, **RtemsPartReqReturnBuffer_Post_Status_InvId**, **RtemsPartReqReturnBuffer_Post_Status_InvAddr**, **RtemsPartReqReturnBuffer_Post_Status_NA** }

### Functions

- static **RTEMS_ALIGNED** (RTEMS_PARTITION_ALIGNMENT)
- static void **RtemsPartReqReturnBuffer_Pre_Buf_Prepare** (RtemsPartReqReturnBuffer_Context *ctx, RtemsPartReqReturnBuffer_Pre_Buf state)
- static void **RtemsPartReqReturnBuffer_Post_Status_Check** (RtemsPartReqReturnBuffer_Context *ctx, RtemsPartReqReturnBuffer_Post_Status state)
- static void **RtemsPartReqReturnBuffer_Setup** (RtemsPartReqReturnBuffer_Context *ctx)
- static void **RtemsPartReqReturnBuffer_Setup_Wrap** (void *arg)
- static void **RtemsPartReqReturnBuffer_Teardown** (RtemsPartReqReturnBuffer_Context *ctx)
- static void **RtemsPartReqReturnBuffer_Teardown_Wrap** (void *arg)
- static size_t **RtemsPartReqReturnBuffer_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsPartReqReturnBuffer_Action** (RtemsPartReqReturnBuffer_Context *ctx)
- **T_TEST_CASE_FIXTURE** (RtemsPartReqReturnBuffer, &RtemsPartReqReturnBuffer_Fixture)

### Variables

- static RtemsPartReqReturnBuffer_Context **RtemsPartReqReturnBuffer_Instance**
- static const char *const **RtemsPartReqReturnBuffer_PreDesc_Id** [ ]
- static const char *const **RtemsPartReqReturnBuffer_PreDesc_Buf** [ ]
- static const char *const *const **RtemsPartReqReturnBuffer_PreDesc** [ ]
- static T_fixture **RtemsPartReqReturnBuffer_Fixture**
- static const uint8_t **RtemsPartReqReturnBuffer_TransitionMap** [ ][1]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1
  uint16_t **Pre_Id_NA**: 1
  uint16_t **Pre_Name_NA**: 1
  uint16_t **Pre_Prio_NA**: 1
  uint16_t **Pre_Tasks_NA**: 1
  uint16_t **Pre_TLS_NA**: 1
  uint16_t **Pre_Stack_NA**: 1
  uint16_t **Pre_Ext_NA**: 1
  uint16_t **Pre_Preempt_NA**: 1
  uint8_t **Pre_Pre_NA**: 1
  uint8_t **Pre_Send_NA**: 1
  uint8_t **Pre_ReceiverState_NA**: 1
  uint8_t **Pre_Satisfy_NA**: 1
  uint8_t **Pre_Node_NA**: 1
  } **RtemsPartReqReturnBuffer_TransitionInfo** [ ]

## 10.426 testsuites/validation/tc-part.c File Reference

```
#include <rtems.h>
#include <rtems/test.h>
```

### Functions

- void **T_case_body_RtemsPartValPart** (void)

## 10.427 testsuites/validation/tc-ratemon-ident.c File Reference

```
#include "tr-object-ident-local.h"
#include <rtems/test.h>
```

### Functions

- static rtems_status_code **ClassicRatemonIdentAction** (rtems_name name, rtems_id *id)
- void **T_case_body_RtemsRatemonValIdent** (void)

## 10.428 testsuites/validation/tc-sem-ident.c File Reference

```
#include "tr-object-ident.h"
#include <rtems/test.h>
```

### Functions

- static rtems_status_code **ClassicSemIdentAction** (rtems_name name, uint32_t node, rtems_id *id)
- void **T_case_body_RtemsSemValIdent** (void)

## 10.429 testsuites/validation/tc-space-profile.c File Reference

```
#include <rtems.h>
#include <stdlib.h>
#include <rtems/malloc.h>
#include <rtems/test.h>
```

### Functions

- void **T_case_body_TestsuitesValidationCLibrary** (void)
- void **T_case_body_TestsuitesValidationClassicBarrier** (void)

## 10.430 testsuites/validation/tc-task-construct-errors.c File Reference

```
#include <rtems.h>
#include <string.h>
#include <rtems/score/chainimpl.h>
#include <rtems/score/objectimpl.h>
#include <rtems/test.h>
```

## Classes

- struct RtemsTaskReqConstructErrors_Context

  *Test context for spec:/rtems/task/req/construct-errors test case.*

## Macros

- #define **MAX_TLS_SIZE** RTEMS_ALIGN_UP( 128, RTEMS_TASK_STORAGE_ALIGNMENT )

## Enumerations

- enum **RtemsTaskReqConstructErrors_Pre_Id** { **RtemsTaskReqConstructErrors_Pre_Id_Id**, **RtemsTaskReqConstructErrors_Pre_Id_Null**, **RtemsTaskReqConstructErrors_Pre_Id_NA** }
- enum **RtemsTaskReqConstructErrors_Pre_Name** { **RtemsTaskReqConstructErrors_Pre_Name_Valid**, **RtemsTaskReqConstructErrors_Pre_Name_Inv**, **RtemsTaskReqConstructErrors_Pre_Name_NA** }
- enum **RtemsTaskReqConstructErrors_Pre_Prio** { **RtemsTaskReqConstructErrors_Pre_Prio_Valid**, **RtemsTaskReqConstructErrors_Pre_Prio_Zero**, **RtemsTaskReqConstructErrors_Pre_Prio_Inv**, **RtemsTaskReqConstructErrors_Pre_Prio_NA** }
- enum **RtemsTaskReqConstructErrors_Pre_Tasks** { **RtemsTaskReqConstructErrors_Pre_Tasks_Avail**, **RtemsTaskReqConstructErrors_Pre_Tasks_None**, **RtemsTaskReqConstructErrors_Pre_Tasks_NA** }
- enum **RtemsTaskReqConstructErrors_Pre_TLS** { **RtemsTaskReqConstructErrors_Pre_TLS_Enough**, **RtemsTaskReqConstructErrors_Pre_TLS_Small**, **RtemsTaskReqConstructErrors_Pre_TLS_NA** }
- enum **RtemsTaskReqConstructErrors_Pre_Stack** { **RtemsTaskReqConstructErrors_Pre_Stack_Enough**, **RtemsTaskReqConstructErrors_Pre_Stack_Small**, **RtemsTaskReqConstructErrors_Pre_Stack_NA** }
- enum **RtemsTaskReqConstructErrors_Pre_Ext** { **RtemsTaskReqConstructErrors_Pre_Ext_Ok**, **RtemsTaskReqConstructErrors_Pre_Ext_Err**, **RtemsTaskReqConstructErrors_Pre_Ext_NA** }
- enum **RtemsTaskReqConstructErrors_Pre_Preempt** { **RtemsTaskReqConstructErrors_Pre_Preempt_Yes**, **RtemsTaskReqConstructErrors_Pre_Preempt_No**, **RtemsTaskReqConstructErrors_Pre_Preempt_NA** }
- enum **RtemsTaskReqConstructErrors_Post_Status** { **RtemsTaskReqConstructErrors_Post_Status_Ok**, **RtemsTaskReqConstructErrors_Post_Status_InvAddress**, **RtemsTaskReqConstructErrors_Post_Status_InvName**, **RtemsTaskReqConstructErrors_Post_Status_InvPrio**, **RtemsTaskReqConstructErrors_Post_Status_InvSize**, **RtemsTaskReqConstructErrors_Post_Status_TooMany**, **RtemsTaskReqConstructErrors_Post_Status_Unsatisfied**, **RtemsTaskReqConstructErrors_Post_Status_NA** }

## Functions

- **RTEMS_ALIGNED** (RTEMS_TASK_STORAGE_ALIGNMENT)
- static bool **ThreadCreate** (rtems_tcb *executing, rtems_tcb *created)
- static void **RtemsTaskReqConstructErrors_Pre_Id_Prepare** (RtemsTaskReqConstructErrors_Context *ctx, RtemsTaskReqConstructErrors_Pre_Id state)
- static void **RtemsTaskReqConstructErrors_Pre_Name_Prepare** (RtemsTaskReqConstructErrors_Context *ctx, RtemsTaskReqConstructErrors_Pre_Name state)
- static void **RtemsTaskReqConstructErrors_Pre_Prio_Prepare** (RtemsTaskReqConstructErrors_Context *ctx, RtemsTaskReqConstructErrors_Pre_Prio state)
- static void **RtemsTaskReqConstructErrors_Pre_Tasks_Prepare** (RtemsTaskReqConstructErrors_Context *ctx, RtemsTaskReqConstructErrors_Pre_Tasks state)
- static void **RtemsTaskReqConstructErrors_Pre_TLS_Prepare** (RtemsTaskReqConstructErrors_Context *ctx, RtemsTaskReqConstructErrors_Pre_TLS state)

- static void **RtemsTaskReqConstructErrors_Pre_Stack_Prepare** (RtemsTaskReqConstructErrors_Context *ctx, RtemsTaskReqConstructErrors_Pre_Stack state)
- static void **RtemsTaskReqConstructErrors_Pre_Ext_Prepare** (RtemsTaskReqConstructErrors_Context *ctx, RtemsTaskReqConstructErrors_Pre_Ext state)
- static void **RtemsTaskReqConstructErrors_Pre_Preempt_Prepare** (RtemsTaskReqConstructErrors_Context *ctx, RtemsTaskReqConstructErrors_Pre_Preempt state)
- static void **RtemsTaskReqConstructErrors_Post_Status_Check** (RtemsTaskReqConstructErrors_Context *ctx, RtemsTaskReqConstructErrors_Post_Status state)
- static void **RtemsTaskReqConstructErrors_Setup** (RtemsTaskReqConstructErrors_Context *ctx)
- static void **RtemsTaskReqConstructErrors_Setup_Wrap** (void *arg)
- static void **RtemsTaskReqConstructErrors_Teardown** (RtemsTaskReqConstructErrors_Context *ctx)
- static void **RtemsTaskReqConstructErrors_Teardown_Wrap** (void *arg)
- static size_t **RtemsTaskReqConstructErrors_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsTaskReqConstructErrors_Prepare** (RtemsTaskReqConstructErrors_Context *ctx)
- static void **RtemsTaskReqConstructErrors_Action** (RtemsTaskReqConstructErrors_Context *ctx)
- static void **RtemsTaskReqConstructErrors_Cleanup** (RtemsTaskReqConstructErrors_Context *ctx)
- **T_TEST_CASE_FIXTURE** (RtemsTaskReqConstructErrors, &RtemsTaskReqConstructErrors_Fixture)

## Variables

- static RtemsTaskReqConstructErrors_Context **RtemsTaskReqConstructErrors_Instance**
- static const char *const **RtemsTaskReqConstructErrors_PreDesc_Id** [ ]
- static const char *const **RtemsTaskReqConstructErrors_PreDesc_Name** [ ]
- static const char *const **RtemsTaskReqConstructErrors_PreDesc_Prio** [ ]
- static const char *const **RtemsTaskReqConstructErrors_PreDesc_Tasks** [ ]
- static const char *const **RtemsTaskReqConstructErrors_PreDesc_TLS** [ ]
- static const char *const **RtemsTaskReqConstructErrors_PreDesc_Stack** [ ]
- static const char *const **RtemsTaskReqConstructErrors_PreDesc_Ext** [ ]
- static const char *const **RtemsTaskReqConstructErrors_PreDesc_Preempt** [ ]
- static const char *const *const **RtemsTaskReqConstructErrors_PreDesc** [ ]
- static _Thread_local int **tls_variable**
- static const rtems_extensions_table **extensions**
- static T_fixture **RtemsTaskReqConstructErrors_Fixture**
- static const uint8_t **RtemsTaskReqConstructErrors_TransitionMap** [ ][1]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1
  uint16_t **Pre_Id_NA**: 1
  uint16_t **Pre_Name_NA**: 1
  uint16_t **Pre_Prio_NA**: 1
  uint16_t **Pre_Tasks_NA**: 1

uint16_t **Pre_TLS_NA**: 1
uint16_t **Pre_Stack_NA**: 1
uint16_t **Pre_Ext_NA**: 1
uint16_t **Pre_Preempt_NA**: 1
uint8_t **Pre_Pre_NA**: 1
uint8_t **Pre_Send_NA**: 1
uint8_t **Pre_ReceiverState_NA**: 1
uint8_t **Pre_Satisfy_NA**: 1
uint8_t **Pre_Node_NA**: 1
} **RtemsTaskReqConstructErrors_TransitionInfo** [ ]

## 10.431 testsuites/validation/tc-task-ident.c File Reference

```
#include "tr-object-ident.h"
#include <rtems/test.h>
```

### Classes

- struct RtemsTaskReqIdent_Context

  *Test context for spec:/rtems/task/req/ident test case.*

### Macros

- #define **TASK_ATTRIBUTES** RTEMS_DEFAULT_ATTRIBUTES
- #define **MAX_TLS_SIZE** RTEMS_ALIGN_UP( 64, RTEMS_TASK_STORAGE_ALIGNMENT )

### Enumerations

- enum **RtemsTaskReqIdent_Pre_Pre** { **RtemsTaskReqIdent_Pre_Pre_Self**, **RtemsTaskReqIdent_Pre_↩ Pre_Generic**, **RtemsTaskReqIdent_Pre_Pre_NA** }
- enum **RtemsTaskReqIdent_Post_Post** { **RtemsTaskReqIdent_Post_Post_OkAndSelfId**, **RtemsTask↩ ReqIdent_Post_Post_Generic**, **RtemsTaskReqIdent_Post_Post_NA** }

### Functions

- static rtems_status_code **ClassicTaskIdentAction** (rtems_name name, uint32_t node, rtems_id *id)
- static void **RtemsTaskReqIdent_Pre_Pre_Prepare** (RtemsTaskReqIdent_Context *ctx, RtemsTaskReq↩ Ident_Pre_Pre state)
- static void **RtemsTaskReqIdent_Post_Post_Check** (RtemsTaskReqIdent_Context *ctx, RtemsTaskReq↩ Ident_Post_Post state)
- static void **RtemsTaskReqIdent_Setup** (RtemsTaskReqIdent_Context *ctx)
- static void **RtemsTaskReqIdent_Setup_Wrap** (void *arg)
- static void **RtemsTaskReqIdent_Teardown** (RtemsTaskReqIdent_Context *ctx)
- static void **RtemsTaskReqIdent_Teardown_Wrap** (void *arg)
- static size_t **RtemsTaskReqIdent_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsTaskReqIdent_Action** (RtemsTaskReqIdent_Context *ctx)
- **T_TEST_CASE_FIXTURE** (RtemsTaskReqIdent, &RtemsTaskReqIdent_Fixture)

### Variables

- static RtemsTaskReqIdent_Context **RtemsTaskReqIdent_Instance**
- static const char *const **RtemsTaskReqIdent_PreDesc_Pre** [ ]
- static const char *const *const **RtemsTaskReqIdent_PreDesc** [ ]
- static char **ClassicTaskIdentStorage** [RTEMS_TASK_STORAGE_SIZE(MAX_TLS_SIZE+RTEMS_MINIMUM_STACK_SIZE, TASK_ATTRIBUTES)]
- static const rtems_task_config **ClassicTaskIdentConfig**
- static T_fixture **RtemsTaskReqIdent_Fixture**
- static const uint8_t **RtemsTaskReqIdent_TransitionMap** [ ][1]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1
  uint16_t **Pre_Id_NA**: 1
  uint16_t **Pre_Name_NA**: 1
  uint16_t **Pre_Prio_NA**: 1
  uint16_t **Pre_Tasks_NA**: 1
  uint16_t **Pre_TLS_NA**: 1
  uint16_t **Pre_Stack_NA**: 1
  uint16_t **Pre_Ext_NA**: 1
  uint16_t **Pre_Preempt_NA**: 1
  uint8_t **Pre_Pre_NA**: 1
  uint8_t **Pre_Send_NA**: 1
  uint8_t **Pre_ReceiverState_NA**: 1
  uint8_t **Pre_Satisfy_NA**: 1
  uint8_t **Pre_Node_NA**: 1
  } **RtemsTaskReqIdent_TransitionInfo** [ ]

## 10.432 testsuites/validation/tc-timer-ident.c File Reference

```
#include "tr-object-ident-local.h"
#include <rtems/test.h>
```

### Functions

- static rtems_status_code **ClassicTimerIdentAction** (rtems_name name, rtems_id *id)
- void **T_case_body_RtemsTimerValIdent** (void)

## 10.433 testsuites/validation/tc-userext-ident.c File Reference

```
#include "tr-object-ident-local.h"
#include <rtems/test.h>
```

### Functions

- static rtems_status_code **ClassicUserExtIdentAction** (rtems_name name, rtems_id *id)
- void **T_case_body_RtemsUserextValIdent** (void)

## 10.434 testsuites/validation/tr-event-constant.c File Reference

```
#include <rtems.h>
#include "tr-event-constant.h"
#include <rtems/test.h>
```

### Functions

- static void **RtemsEventValEventConstant_Wrap** (rtems_event_set event, int number)
- void RtemsEventValEventConstant_Run (rtems_event_set event, int number)

  *Runs the parameterized test case.*

### Variables

- static T_fixture_node **RtemsEventValEventConstant_Node**

## 10.435 testsuites/validation/tr-event-constant.h File Reference

```
#include <rtems.h>
```

### Functions

- void RtemsEventValEventConstant_Run (rtems_event_set event, int number)

  *Runs the parameterized test case.*

## 10.436 testsuites/validation/tr-event-send-receive.c File Reference

```
#include <rtems/score/threadimpl.h>
#include "tr-event-send-receive.h"
#include <rtems/test.h>
```

## Classes

- struct RtemsEventReqSendReceive_Context

  *Test context for spec:/rtems/event/req/send-receive test case.*

## Macros

- #define **INPUT_EVENTS** ( RTEMS_EVENT_5 | RTEMS_EVENT_23 )
- #define **WORKER_ATTRIBUTES** RTEMS_DEFAULT_ATTRIBUTES
- #define **MAX_TLS_SIZE** RTEMS_ALIGN_UP( 64, RTEMS_TASK_STORAGE_ALIGNMENT )

## Typedefs

- typedef RtemsEventReqSendReceive_Context **Context**

## Enumerations

- enum **Priorities** { **PRIO_HIGH** = 1, **PRIO_NORMAL**, **PRIO_LOW**, **PRIO_OTHER** }
- enum **SenderTypes** {
  **SENDER_NONE**, **SENDER_SELF**, **SENDER_SELF_2**, **SENDER_WORKER**,
  **SENDER_INTERRUPT** }
- enum **ReceiveTypes** { **RECEIVE_SKIP**, **RECEIVE_NORMAL**, **RECEIVE_INTERRUPT** }
- enum **ReceiveConditionStates** { **RECEIVE_COND_UNKNOWN**, **RECEIVE_COND_SATSIFIED**, **RECE↩
  IVE_COND_UNSATISFIED** }

## Functions

- **RTEMS_ALIGNED** (RTEMS_TASK_STORAGE_ALIGNMENT)
- static rtems_id **CreateWakeupSema** (void)
- static void **DeleteWakeupSema** (rtems_id id)
- static void **Wait** (rtems_id id)
- static void **Wakeup** (rtems_id id)
- static bool **BlockedForEvent** (Context *ctx, Thread_Wait_flags flags)
- static bool **IntendsToBlockForEvent** (Context *ctx, Thread_Wait_flags flags)
- static bool **EventReadyAgain** (Context *ctx, Thread_Wait_flags flags)
- static bool **IsSatisfiedFlags** (Context *ctx)
- static bool **IsSatisfiedState** (Context *ctx)
- static void **SendAction** (Context *ctx)
- static void **Send** (Context *ctx, bool(*is_satsified)(Context *))
- static void **Worker** (rtems_task_argument arg)
- static rtems_event_set **GetPendingEvents** (Context *ctx)
- static void **RtemsEventReqSendReceive_Cleanup** (Context *ctx)
- static void **InterruptPrepare** (void *arg)
- static void **InterruptAction** (void *arg)
- static void **InterruptContinue** (Context *ctx)
- static T_interrupt_test_state **Interrupt** (void *arg)
- static void **RtemsEventReqSendReceive_Pre_Id_Prepare** (RtemsEventReqSendReceive_Context *ctx, RtemsEventReqSendReceive_Pre_Id state)
- static void **RtemsEventReqSendReceive_Pre_Send_Prepare** (RtemsEventReqSendReceive_Context *ctx, RtemsEventReqSendReceive_Pre_Send state)

- static void **RtemsEventReqSendReceive_Pre_ReceiverState_Prepare** (RtemsEventReqSendReceive_Context *ctx, RtemsEventReqSendReceive_Pre_ReceiverState state)
- static void **RtemsEventReqSendReceive_Pre_Satisfy_Prepare** (RtemsEventReqSendReceive_Context *ctx, RtemsEventReqSendReceive_Pre_Satisfy state)
- static void **RtemsEventReqSendReceive_Post_SendStatus_Check** (RtemsEventReqSendReceive_Context *ctx, RtemsEventReqSendReceive_Post_SendStatus state)
- static void **RtemsEventReqSendReceive_Post_ReceiveStatus_Check** (RtemsEventReqSendReceive_Context *ctx, RtemsEventReqSendReceive_Post_ReceiveStatus state)
- static void **RtemsEventReqSendReceive_Post_SenderPreemption_Check** (RtemsEventReqSendReceive_Context *ctx, RtemsEventReqSendReceive_Post_SenderPreemption state)
- static void **RtemsEventReqSendReceive_Setup** (RtemsEventReqSendReceive_Context *ctx)
- static void **RtemsEventReqSendReceive_Setup_Wrap** (void *arg)
- static void **RtemsEventReqSendReceive_Teardown** (RtemsEventReqSendReceive_Context *ctx)
- static void **RtemsEventReqSendReceive_Teardown_Wrap** (void *arg)
- static size_t **RtemsEventReqSendReceive_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsEventReqSendReceive_Prepare** (RtemsEventReqSendReceive_Context *ctx)
- static void **RtemsEventReqSendReceive_Action** (RtemsEventReqSendReceive_Context *ctx)
- void RtemsEventReqSendReceive_Run (rtems_status_code(*send)(rtems_id, rtems_event_set), rtems_status_code(*receive)(rtems_option, rtems_interval, rtems_event_set *), rtems_event_set( *get_pending_events)(Thread_Control *), unsigned int wait_class, int waiting_for_event)

  *Runs the parameterized test case.*

## Variables

- static RtemsEventReqSendReceive_Context **RtemsEventReqSendReceive_Instance**
- static const char *const **RtemsEventReqSendReceive_PreDesc_Id** [ ]
- static const char *const **RtemsEventReqSendReceive_PreDesc_Send** [ ]
- static const char *const **RtemsEventReqSendReceive_PreDesc_ReceiverState** [ ]
- static const char *const **RtemsEventReqSendReceive_PreDesc_Satisfy** [ ]
- static const char *const *const **RtemsEventReqSendReceive_PreDesc** [ ]
- static const T_interrupt_test_config **InterruptConfig**
- static T_fixture **RtemsEventReqSendReceive_Fixture**
- static const uint8_t **RtemsEventReqSendReceive_TransitionMap** [ ][3]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1
  uint16_t **Pre_Id_NA**: 1
  uint16_t **Pre_Name_NA**: 1
  uint16_t **Pre_Prio_NA**: 1
  uint16_t **Pre_Tasks_NA**: 1
  uint16_t **Pre_TLS_NA**: 1

uint16_t **Pre_Stack_NA**: 1
uint16_t **Pre_Ext_NA**: 1
uint16_t **Pre_Preempt_NA**: 1
uint8_t **Pre_Pre_NA**: 1
uint8_t **Pre_Send_NA**: 1
uint8_t **Pre_ReceiverState_NA**: 1
uint8_t **Pre_Satisfy_NA**: 1
uint8_t **Pre_Node_NA**: 1
} **RtemsEventReqSendReceive_TransitionInfo** [ ]

- static T_fixture_node **RtemsEventReqSendReceive_Node**

## 10.437 testsuites/validation/tr-event-send-receive.h File Reference

```
#include <rtems.h>
#include <rtems/score/thread.h>
```

### Enumerations

- enum **RtemsEventReqSendReceive_Pre_Id** { **RtemsEventReqSendReceive_Pre_Id_InvId**, **RtemsEventReqSendReceive_Pre_Id_Task**, **RtemsEventReqSendReceive_Pre_Id_NA** }
- enum **RtemsEventReqSendReceive_Pre_Send** { **RtemsEventReqSendReceive_Pre_Send_Zero**, **RtemsEventReqSendReceive_Pre_Send_Unrelated**, **RtemsEventReqSendReceive_Pre_Send_Any**, **RtemsEventReqSendReceive_Pre_Send_All**, **RtemsEventReqSendReceive_Pre_Send_MixedAny**, **RtemsEventReqSendReceive_Pre_Send_MixedAll**, **RtemsEventReqSendReceive_Pre_Send_NA** }
- enum **RtemsEventReqSendReceive_Pre_ReceiverState** { **RtemsEventReqSendReceive_Pre_ReceiverState_NotWaiting**, **RtemsEventReqSendReceive_Pre_ReceiverState_Poll**, **RtemsEventReqSendReceive_Pre_ReceiverState_Timeout**, **RtemsEventReqSendReceive_Pre_ReceiverState_Lower**, **RtemsEventReqSendReceive_Pre_ReceiverState_Equal**, **RtemsEventReqSendReceive_Pre_ReceiverState_Higher**, **RtemsEventReqSendReceive_Pre_ReceiverState_Other**, **RtemsEventReqSendReceive_Pre_ReceiverState_Intend**, **RtemsEventReqSendReceive_Pre_ReceiverState_NA** }
- enum **RtemsEventReqSendReceive_Pre_Satisfy** { **RtemsEventReqSendReceive_Pre_Satisfy_All**, **RtemsEventReqSendReceive_Pre_Satisfy_Any**, **RtemsEventReqSendReceive_Pre_Satisfy_NA** }
- enum **RtemsEventReqSendReceive_Post_SendStatus** { **RtemsEventReqSendReceive_Post_SendStatus_Ok**, **RtemsEventReqSendReceive_Post_SendStatus_InvId**, **RtemsEventReqSendReceive_Post_SendStatus_NA** }
- enum **RtemsEventReqSendReceive_Post_ReceiveStatus** { **RtemsEventReqSendReceive_Post_ReceiveStatus_None**, **RtemsEventReqSendReceive_Post_ReceiveStatus_Pending**, **RtemsEventReqSendReceive_Post_ReceiveStatus_Timeout**, **RtemsEventReqSendReceive_Post_ReceiveStatus_Satisfied**, **RtemsEventReqSendReceive_Post_ReceiveStatus_Unsatisfied**, **RtemsEventReqSendReceive_Post_ReceiveStatus_Blocked**, **RtemsEventReqSendReceive_Post_ReceiveStatus_NA** }
- enum **RtemsEventReqSendReceive_Post_SenderPreemption** { **RtemsEventReqSendReceive_Post_SenderPreemption_No**, **RtemsEventReqSendReceive_Post_SenderPreemption_Yes**, **RtemsEventReqSendReceive_Post_SenderPreemption_NA** }

### Functions

- void RtemsEventReqSendReceive_Run (rtems_status_code(∗send)(rtems_id, rtems_event_set), rtems_status_code(∗receive)(rtems_option, rtems_interval, rtems_event_set ∗), rtems_event_set( ∗get_pending_events)(Thread_Control ∗), unsigned int wait_class, int waiting_for_event)

  *Runs the parameterized test case.*

# 10.438 testsuites/validation/tr-object-ident-local.c File Reference

```
#include "tr-object-ident-local.h"
#include <rtems/test.h>
```

## Classes

- struct RtemsReqIdentLocal_Context

  *Test context for spec:/rtems/req/ident-local test case.*

## Functions

- static void **RtemsReqIdentLocal_Pre_Name_Prepare** (RtemsReqIdentLocal_Context *ctx, RtemsReq↩ IdentLocal_Pre_Name state)
- static void **RtemsReqIdentLocal_Pre_Id_Prepare** (RtemsReqIdentLocal_Context *ctx, RtemsReqIdent↩ Local_Pre_Id state)
- static void **RtemsReqIdentLocal_Post_Status_Check** (RtemsReqIdentLocal_Context *ctx, RtemsReq↩ IdentLocal_Post_Status state)
- static void **RtemsReqIdentLocal_Post_Id_Check** (RtemsReqIdentLocal_Context *ctx, RtemsReqIdent↩ Local_Post_Id state)
- static size_t **RtemsReqIdentLocal_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsReqIdentLocal_Action** (RtemsReqIdentLocal_Context *ctx)
- void RtemsReqIdentLocal_Run (rtems_id id_local_object, rtems_status_code(*action)(rtems_name, rtems_id *))

  *Runs the parameterized test case.*

## Variables

- static RtemsReqIdentLocal_Context **RtemsReqIdentLocal_Instance**
- static const char *const **RtemsReqIdentLocal_PreDesc_Name** [ ]
- static const char *const **RtemsReqIdentLocal_PreDesc_Id** [ ]
- static const char *const *const **RtemsReqIdentLocal_PreDesc** [ ]
- static T_fixture **RtemsReqIdentLocal_Fixture**
- static const uint8_t **RtemsReqIdentLocal_TransitionMap** [ ][2]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1

uint16_t **Pre_Id_NA**: 1
uint16_t **Pre_Name_NA**: 1
uint16_t **Pre_Prio_NA**: 1
uint16_t **Pre_Tasks_NA**: 1
uint16_t **Pre_TLS_NA**: 1
uint16_t **Pre_Stack_NA**: 1
uint16_t **Pre_Ext_NA**: 1
uint16_t **Pre_Preempt_NA**: 1
uint8_t **Pre_Pre_NA**: 1
uint8_t **Pre_Send_NA**: 1
uint8_t **Pre_ReceiverState_NA**: 1
uint8_t **Pre_Satisfy_NA**: 1
uint8_t **Pre_Node_NA**: 1
} **RtemsReqIdentLocal_TransitionInfo** [ ]

- static T_fixture_node **RtemsReqIdentLocal_Node**

## 10.439 testsuites/validation/tr-object-ident-local.h File Reference

```
#include <rtems.h>
```

### Macros

- #define **ClassicObjectLocalIdentName** rtems_build_name( 'I', 'D', 'N', 'T' )

### Enumerations

- enum **RtemsReqIdentLocal_Pre_Name** { **RtemsReqIdentLocal_Pre_Name_Invalid**, **RtemsReqIdentLocal_Pre_Name_Valid**, **RtemsReqIdentLocal_Pre_Name_NA** }
- enum **RtemsReqIdentLocal_Pre_Id** { **RtemsReqIdentLocal_Pre_Id_NullPtr**, **RtemsReqIdentLocal_Pre_Id_Valid**, **RtemsReqIdentLocal_Pre_Id_NA** }
- enum **RtemsReqIdentLocal_Post_Status** { **RtemsReqIdentLocal_Post_Status_Ok**, **RtemsReqIdentLocal_Post_Status_InvAddr**, **RtemsReqIdentLocal_Post_Status_InvName**, **RtemsReqIdentLocal_Post_Status_NA** }
- enum **RtemsReqIdentLocal_Post_Id** { **RtemsReqIdentLocal_Post_Id_Nop**, **RtemsReqIdentLocal_Post_Id_NullPtr**, **RtemsReqIdentLocal_Post_Id_Id**, **RtemsReqIdentLocal_Post_Id_NA** }

### Functions

- void RtemsReqIdentLocal_Run (rtems_id id_local_object, rtems_status_code(*action)(rtems_name, rtems_id *))

  *Runs the parameterized test case.*

## 10.440 testsuites/validation/tr-object-ident.c File Reference

```
#include "tr-object-ident.h"
#include <rtems/test.h>
```

## Classes

- struct RtemsReqIdent_Context

  *Test context for spec:/rtems/req/ident test case.*

## Functions

- static void **RtemsReqIdent_Pre_Name_Prepare** (RtemsReqIdent_Context *ctx, RtemsReqIdent_Pre_↩ Name state)
- static void **RtemsReqIdent_Pre_Node_Prepare** (RtemsReqIdent_Context *ctx, RtemsReqIdent_Pre_Node state)
- static void **RtemsReqIdent_Pre_Id_Prepare** (RtemsReqIdent_Context *ctx, RtemsReqIdent_Pre_Id state)
- static void **RtemsReqIdent_Post_Status_Check** (RtemsReqIdent_Context *ctx, RtemsReqIdent_Post_↩ Status state)
- static void **RtemsReqIdent_Post_Id_Check** (RtemsReqIdent_Context *ctx, RtemsReqIdent_Post_Id state)
- static size_t **RtemsReqIdent_Scope** (void *arg, char *buf, size_t n)
- static void **RtemsReqIdent_Action** (RtemsReqIdent_Context *ctx)
- void RtemsReqIdent_Run (rtems_id id_local_object, rtems_status_code(*action)(rtems_name, uint32_↩ t, rtems_id *))

  *Runs the parameterized test case.*

## Variables

- static RtemsReqIdent_Context **RtemsReqIdent_Instance**
- static const char *const **RtemsReqIdent_PreDesc_Name** [ ]
- static const char *const **RtemsReqIdent_PreDesc_Node** [ ]
- static const char *const **RtemsReqIdent_PreDesc_Id** [ ]
- static const char *const *const **RtemsReqIdent_PreDesc** [ ]
- static T_fixture **RtemsReqIdent_Fixture**
- static const uint8_t **RtemsReqIdent_TransitionMap** [ ][2]
- struct {
  uint8_t **Skip**: 1
  uint8_t **Pre_Id_NA**: 1
  uint8_t **Pre_Name_NA**: 1
  uint8_t **Pre_MaxPending_NA**: 1
  uint8_t **Pre_MaxSize_NA**: 1
  uint8_t **Pre_Queues_NA**: 1
  uint8_t **Pre_Area_NA**: 1
  uint8_t **Pre_AreaSize_NA**: 1
  uint8_t **Pre_Start_NA**: 1
  uint8_t **Pre_Length_NA**: 1
  uint8_t **Pre_Size_NA**: 1
  uint8_t **Pre_Parts_NA**: 1
  uint8_t **Pre_InUse_NA**: 1
  uint8_t **Pre_Buf_NA**: 1
  uint8_t **Pre_Avail_NA**: 1
  uint16_t **Skip**: 1
  uint16_t **Pre_Id_NA**: 1
  uint16_t **Pre_Name_NA**: 1
  uint16_t **Pre_Prio_NA**: 1
  uint16_t **Pre_Tasks_NA**: 1
  uint16_t **Pre_TLS_NA**: 1
  uint16_t **Pre_Stack_NA**: 1
  uint16_t **Pre_Ext_NA**: 1

uint16_t **Pre_Preempt_NA**: 1
uint8_t **Pre_Pre_NA**: 1
uint8_t **Pre_Send_NA**: 1
uint8_t **Pre_ReceiverState_NA**: 1
uint8_t **Pre_Satisfy_NA**: 1
uint8_t **Pre_Node_NA**: 1
} **RtemsReqIdent_TransitionInfo** [ ]

- static T_fixture_node **RtemsReqIdent_Node**

## 10.441 testsuites/validation/tr-object-ident.h File Reference

```
#include <rtems.h>
```

### Macros

- #define **ClassicObjectIdentName** rtems_build_name( 'I', 'D', 'N', 'T' )

### Enumerations

- enum **RtemsReqIdent_Pre_Name** { **RtemsReqIdent_Pre_Name_Invalid**, **RtemsReqIdent_Pre_Name_Valid**, **RtemsReqIdent_Pre_Name_NA** }
- enum **RtemsReqIdent_Pre_Node** { **RtemsReqIdent_Pre_Node_Local**, **RtemsReqIdent_Pre_Node_Remote**, **RtemsReqIdent_Pre_Node_Invalid**, **RtemsReqIdent_Pre_Node_SearchAll**, **RtemsReqIdent_Pre_Node_SearchOther**, **RtemsReqIdent_Pre_Node_SearchLocal**, **RtemsReqIdent_Pre_Node_NA** }
- enum **RtemsReqIdent_Pre_Id** { **RtemsReqIdent_Pre_Id_NullPtr**, **RtemsReqIdent_Pre_Id_Valid**, **RtemsReqIdent_Pre_Id_NA** }
- enum **RtemsReqIdent_Post_Status** { **RtemsReqIdent_Post_Status_Ok**, **RtemsReqIdent_Post_Status_InvAddr**, **RtemsReqIdent_Post_Status_InvName**, **RtemsReqIdent_Post_Status_InvNode**, **RtemsReqIdent_Post_Status_NA** }
- enum **RtemsReqIdent_Post_Id** { **RtemsReqIdent_Post_Id_Nop**, **RtemsReqIdent_Post_Id_NullPtr**, **RtemsReqIdent_Post_Id_LocalObj**, **RtemsReqIdent_Post_Id_RemoteObj**, **RtemsReqIdent_Post_Id_NA** }

### Functions

- void RtemsReqIdent_Run (rtems_id id_local_object, rtems_status_code(*action)(rtems_name, uint32_t, rtems_id *))

  *Runs the parameterized test case.*

# 10.442 testsuites/validation/ts-space-profile.c File Reference

```
#include <rtems.h>
#include <rtems/bspIo.h>
#include <rtems/sysinit.h>
#include <rtems/score/sysstate.h>
#include <rtems/test.h>
#include <rtems/confdefs.h>
```

## Macros

- #define **NAME** rtems_build_name('N', 'A', 'M', 'E')
- #define **INIT_TASK_ATTRIBUTES** RTEMS_DEFAULT_ATTRIBUTES
- #define **MAX_TLS_SIZE** RTEMS_ALIGN_UP(64, RTEMS_TASK_STORAGE_ALIGNMENT)
- #define **CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER**
- #define **CONFIGURE_MAXIMUM_PROCESSORS** 4
- #define **CONFIGURE_MAXIMUM_BARRIERS** 1
- #define **CONFIGURE_MAXIMUM_MESSAGE_QUEUES** 1
- #define **CONFIGURE_MAXIMUM_PARTITIONS** 1
- #define **CONFIGURE_MAXIMUM_PERIODS** 1
- #define **CONFIGURE_MAXIMUM_SEMAPHORES** 1
- #define **CONFIGURE_MAXIMUM_TASKS** 1
- #define **CONFIGURE_MAXIMUM_TIMERS** 1
- #define **CONFIGURE_MAXIMUM_USER_EXTENSIONS** 1
- #define **CONFIGURE_MESSAGE_BUFFER_MEMORY** 1
- #define **CONFIGURE_MICROSECONDS_PER_TICK** 10000
- #define **CONFIGURE_SCHEDULER_NAME** NAME
- #define **CONFIGURE_INITIAL_EXTENSIONS** { .fatal = fatal_extension }
- #define **CONFIGURE_MAXIMUM_FILE_DESCRIPTORS** 0
- #define **CONFIGURE_DISABLE_NEWLIB_REENTRANCY**
- #define **CONFIGURE_APPLICATION_DISABLE_FILESYSTEM**
- #define **CONFIGURE_IDLE_TASK_INITIALIZES_APPLICATION**
- #define **CONFIGURE_IDLE_TASK_BODY** _CPU_Thread_Idle_body
- #define **CONFIGURE_INIT**

## Enumerations

- enum {
  **_Sysinit_bsp_start** = 0x00030080, **_Sysinit_bsp_debug_uart_init** = 0x00030003, **_Sysinit_amba_↩ initialize** = 0x00030001, **_Sysinit_leon3_cpu_index_init** = 0x00030000,
  **_Sysinit_leon3_counter_initialize** = 0x00040000, **_Sysinit_bsp_memory_initialize** = 0x00018080, _↩ **Sysinit__Malloc_Initialize** = 0x00028080, **_Sysinit__Barrier_Manager_initialization** = 0x00140080,
  **_Sysinit__Message_queue_Manager_initialization** = 0x000e0080, **_Sysinit__Partition_Manager_↩ initialization** = 0x00100080, **_Sysinit__Rate_monotonic_Manager_initialization** = 0x00130080, _↩ **Sysinit__Semaphore_Manager_initialization** = 0x000f0080,
  **_Sysinit__RTEMS_tasks_Manager_initialization** = 0x000a0080, **_Sysinit__Timer_Manager_↩ initialization** = 0x000b0080, **_Sysinit_rtems_initialize_data_structures** = 0x00070080, **_Sysinit__↩ Extension_Manager_initialization** = 0x00090080,
  **_Sysinit__Per_CPU_Data_initialize** = 0x00022080, **_Sysinit__Workspace_Handler_initialization** = 0x00026080, **_Sysinit_init_task** = 0x00290080, **_Sysinit_init_runner_task** = 0x00290080 }

### Functions

- static void **fatal_extension** (rtems_fatal_source source, bool always_set_to_false, rtems_fatal_code error)
- static void **Init** (rtems_task_argument arg)
- **RTEMS_ALIGNED** (RTEMS_TASK_STORAGE_ALIGNMENT)
- static void **init_task** (void)

### Variables

- static const T_action **actions** [ ]
- static const T_config **test_config**
- static const rtems_task_config **task_config**
- rtems_sysinit_item const **_Linker_set__Sysinit_init_task** = { init_task }

## 10.443 testsuites/validation/ts-validation-0.c File Reference

```
#include <rtems.h>
#include <rtems/bspIo.h>
#include <rtems/sysinit.h>
#include <rtems/test-info.h>
#include <rtems/testopts.h>
#include <rtems/test.h>
#include <rtems/scheduler.h>
#include <rtems/confdefs.h>
```

### Macros

- #define **MAX_TLS_SIZE** RTEMS_ALIGN_UP( 64, RTEMS_TASK_STORAGE_ALIGNMENT )
- #define **ATTRIBUTES** RTEMS_FLOATING_POINT
- #define **CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER**
- #define **CONFIGURE_MAXIMUM_PROCESSORS** 4
- #define **CONFIGURE_MAXIMUM_BARRIERS** 3
- #define **CONFIGURE_MAXIMUM_MESSAGE_QUEUES** 3
- #define **CONFIGURE_MAXIMUM_PARTITIONS** 3
- #define **CONFIGURE_MAXIMUM_PERIODS** 3
- #define **CONFIGURE_MAXIMUM_SEMAPHORES** 3
- #define **CONFIGURE_MAXIMUM_TASKS** 3
- #define **CONFIGURE_MINIMUM_TASKS_WITH_USER_PROVIDED_STORAGE** CONFIGURE_MAXIMUM_TASKS
- #define **CONFIGURE_MAXIMUM_TIMERS** 3
- #define **CONFIGURE_MAXIMUM_USER_EXTENSIONS** 3
- #define **CONFIGURE_MICROSECONDS_PER_TICK** 1000
- #define **CONFIGURE_MAXIMUM_FILE_DESCRIPTORS** 0
- #define **CONFIGURE_DISABLE_NEWLIB_REENTRANCY**
- #define **CONFIGURE_APPLICATION_DISABLE_FILESYSTEM**
- #define **CONFIGURE_IDLE_TASK_INITIALIZES_APPLICATION**
- #define **CONFIGURE_IDLE_TASK_BODY** _CPU_Thread_Idle_body
- #define **CONFIGURE_SCHEDULER_EDF_SMP**
- #define **CONFIGURE_SCHEDULER_TABLE_ENTRIES**
- #define **CONFIGURE_SCHEDULER_ASSIGNMENTS**
- #define **CONFIGURE_INIT**

## Enumerations

- enum {
  **_Sysinit_bsp_start** = 0x00030080, **_Sysinit_bsp_debug_uart_init** = 0x00030003, **_Sysinit_amba_↩ initialize** = 0x00030001, **_Sysinit_leon3_cpu_index_init** = 0x00030000,
  **_Sysinit_leon3_counter_initialize** = 0x00040000, **_Sysinit_bsp_memory_initialize** = 0x00018080, _↩ **Sysinit__Malloc_Initialize** = 0x00028080, **_Sysinit__Barrier_Manager_initialization** = 0x00140080,
  **_Sysinit__Message_queue_Manager_initialization** = 0x000e0080, **_Sysinit__Partition_Manager_↩ initialization** = 0x00100080, **_Sysinit__Rate_monotonic_Manager_initialization** = 0x00130080, _↩ **Sysinit__Semaphore_Manager_initialization** = 0x000f0080,
  **_Sysinit__RTEMS_tasks_Manager_initialization** = 0x000a0080, **_Sysinit__Timer_Manager_↩ initialization** = 0x000b0080, **_Sysinit_rtems_initialize_data_structures** = 0x00070080, **_Sysinit__↩ Extension_Manager_initialization** = 0x00090080,
  **_Sysinit__Per_CPU_Data_initialize** = 0x00022080, **_Sysinit__Workspace_Handler_initialization** = 0x00026080, **_Sysinit_init_task** = 0x00290080, **_Sysinit_init_runner_task** = 0x00290080 }

## Functions

- static void **runner_task** (rtems_task_argument arg)
- static void **init_runner_task** (void)
- **RTEMS_SCHEDULER_EDF_SMP** (a)
- **RTEMS_SCHEDULER_EDF_SMP** (b)
- **RTEMS_SCHEDULER_EDF_SMP** (c)

## Variables

- const char rtems_test_name [ ] = "Validation0"

  *Each test must define a test name string.*
- static char **buffer** [512]
- static const T_action **actions** [ ]
- static const T_config **test_config**
- static char **runner_task_storage** [RTEMS_TASK_STORAGE_SIZE(MAX_TLS_SIZE+RTEMS_MINIMUM_STACK_SIZE, ATTRIBUTES)]
- static const rtems_task_config **runner_task_config**
- rtems_sysinit_item const **_Linker_set__Sysinit_init_runner_task** = { init_runner_task }

# Index


_API_Mutex_Is_owner
    API Mutex Handler, 50
_API_Mutex_Lock
    API Mutex Handler, 50
_API_Mutex_Unlock
    API Mutex Handler, 50
_ASR_Initialize
    Classic ASR Implementation, 215
_Addresses_Add_offset
    Address Handler, 51
_Addresses_Align_down
    Address Handler, 52
_Addresses_Align_up
    Address Handler, 52
_Addresses_Is_aligned
    Address Handler, 53
_Addresses_Is_in_range
    Address Handler, 53
_Addresses_Subtract
    Address Handler, 54
_Addresses_Subtract_offset
    Address Handler, 54
_Attributes_Clear
    Classic Attributes Implementation, 217
_Attributes_Has_at_most_one_protocol
    Classic Attributes Implementation, 217
_Attributes_Is_barrier_automatic
    Classic Attributes Implementation, 217
_Attributes_Is_binary_semaphore
    Classic Attributes Implementation, 218
_Attributes_Is_counting_semaphore
    Classic Attributes Implementation, 218
_Attributes_Is_floating_point
    Classic Attributes Implementation, 218
_Attributes_Is_inherit_priority
    Classic Attributes Implementation, 218
_Attributes_Is_multiprocessor_resource_sharing
    Classic Attributes Implementation, 219
_Attributes_Is_priority
    Classic Attributes Implementation, 219
_Attributes_Is_priority_ceiling
    Classic Attributes Implementation, 219
_Attributes_Is_simple_binary_semaphore
    Classic Attributes Implementation, 219
_Attributes_Is_system_task
    Classic Attributes Implementation, 220
_Attributes_Set
    Classic Attributes Implementation, 220
_Barrier_Allocate
    Classic Barrier Implementation, 222
_Barrier_Free
    Classic Barrier Implementation, 222
_Bitfield_Find_first_bit
    Priority Handler, 665
_Bitfield_Leading_zeros
    Priority Handler, 684
_CORE_barrier_Acquire_critical
    Barrier Handler, 106
_CORE_barrier_Destroy
    Barrier Handler, 107
_CORE_barrier_Do_flush
    Barrier Handler, 107
_CORE_barrier_Flush
    Barrier Handler, 108
_CORE_barrier_Get_number_of_waiting_threads
    Barrier Handler, 108
_CORE_barrier_Initialize
    Barrier Handler, 108
_CORE_barrier_Is_automatic
    Barrier Handler, 109
_CORE_barrier_Release
    Barrier Handler, 109
_CORE_barrier_Seize
    Barrier Handler, 109
_CORE_barrier_Surrender
    Barrier Handler, 110
_CORE_ceiling_mutex_Get_priority
    Mutex Handler, 582
_CORE_ceiling_mutex_Get_scheduler
    Mutex Handler, 583
_CORE_ceiling_mutex_Initialize
    Mutex Handler, 583
_CORE_ceiling_mutex_Seize
    Mutex Handler, 584
_CORE_ceiling_mutex_Set_owner
    Mutex Handler, 584
_CORE_ceiling_mutex_Set_priority
    Mutex Handler, 585
_CORE_ceiling_mutex_Surrender
    Mutex Handler, 585
_CORE_message_queue_Acquire
    Message Queue Handler, 557
_CORE_message_queue_Acquire_critical
    Message Queue Handler, 557
_CORE_message_queue_Allocate_message_buffer
    Message Queue Handler, 557
_CORE_message_queue_Broadcast
    Message Queue Handler, 558

_CORE_message_queue_Close
Message Queue Handler, 559
_CORE_message_queue_Copy_buffer
Message Queue Handler, 559
_CORE_message_queue_Dequeue_receiver
Message Queue Handler, 560
_CORE_message_queue_Flush
Message Queue Handler, 560
_CORE_message_queue_Free_message_buffer
Message Queue Handler, 561
_CORE_message_queue_Get_message_priority
Message Queue Handler, 561
_CORE_message_queue_Get_pending_message
Message Queue Handler, 562
_CORE_message_queue_Initialize
Message Queue Handler, 562
_CORE_message_queue_Insert_message
Message Queue Handler, 563
_CORE_message_queue_Release
Message Queue Handler, 563
_CORE_message_queue_Seize
Message Queue Handler, 564
_CORE_message_queue_Send
Message Queue Handler, 564
_CORE_message_queue_Set_notify
Message Queue Handler, 554
_CORE_message_queue_Submit
Message Queue Handler, 565
_CORE_message_queue_Urgent
Message Queue Handler, 566
_CORE_message_queue_Workspace_allocate
Message Queue Handler, 567
_CORE_mutex_Acquire_critical
Mutex Handler, 586
_CORE_mutex_Destroy
Mutex Handler, 586
_CORE_mutex_Get_owner
Mutex Handler, 586
_CORE_mutex_Initialize
Mutex Handler, 587
_CORE_mutex_Is_locked
Mutex Handler, 587
_CORE_mutex_Is_owner
Mutex Handler, 587
_CORE_mutex_Release
Mutex Handler, 588
_CORE_mutex_Seize_slow
Mutex Handler, 588
_CORE_mutex_Set_owner
Mutex Handler, 589
_CORE_recursive_mutex_Initialize
Mutex Handler, 589
_CORE_recursive_mutex_Seize
Mutex Handler, 589
_CORE_recursive_mutex_Seize_nested
Mutex Handler, 590
_CORE_recursive_mutex_Surrender
Mutex Handler, 590
_CORE_semaphore_Acquire_critical
Semaphore Handler, 893
_CORE_semaphore_Destroy
Semaphore Handler, 893
_CORE_semaphore_Get_count
Semaphore Handler, 894
_CORE_semaphore_Initialize
Semaphore Handler, 894
_CORE_semaphore_Release
Semaphore Handler, 894
_CORE_semaphore_Seize
Semaphore Handler, 895
_CORE_semaphore_Surrender
Semaphore Handler, 895
_CPU_Context_Get_SP
SPARC Context Structures, 838
_CPU_Context_Initialization_at_thread_begin
cpu.h, 1854
_CPU_Context_Initialize
cpu.c, 1847
cpu.h, 1866
_CPU_Context_Restart_self
cpu.h, 1854
_CPU_Context_restore
cpu.h, 1867
_CPU_Context_switch
cpu.h, 1867
_CPU_Fatal_halt
bsp_fatal_halt.c, 1568
cpu.h, 1867
_CPU_ISR_Disable
cpu.h, 1855
_CPU_ISR_Enable
cpu.h, 1855
_CPU_ISR_Flash
cpu.h, 1855
_CPU_ISR_Get_level
cpu.c, 1848
cpu.h, 1868
_CPU_ISR_Set_level
cpu.h, 1856
_CPU_ISR_install_raw_handler
cpu.c, 1848
cpu.h, 1868
_CPU_ISR_install_vector
cpu.c, 1849
cpu.h, 1869
_CPU_Initialize
cpu.c, 1848
cpu.h, 1868
_CPU_Initialize_vectors
cpu.h, 1855
_CPU_Trap_slot_template
cpu.c, 1849
cpu.h, 1870
_CPU_atomic_Compare_exchange_uint
Atomic Operations CPU, 81
_CPU_atomic_Compare_exchange_uintptr

Atomic Operations CPU, 81
_CPU_atomic_Compare_exchange_ulong
Atomic Operations CPU, 82
_CPU_atomic_Exchange_uint
Atomic Operations CPU, 83
_CPU_atomic_Exchange_uintptr
Atomic Operations CPU, 83
_CPU_atomic_Exchange_ulong
Atomic Operations CPU, 84
_CPU_atomic_Fence
Atomic Operations CPU, 84
_CPU_atomic_Fetch_add_uint
Atomic Operations CPU, 84
_CPU_atomic_Fetch_add_uintptr
Atomic Operations CPU, 85
_CPU_atomic_Fetch_add_ulong
Atomic Operations CPU, 85
_CPU_atomic_Fetch_and_uint
Atomic Operations CPU, 86
_CPU_atomic_Fetch_and_uintptr
Atomic Operations CPU, 86
_CPU_atomic_Fetch_and_ulong
Atomic Operations CPU, 87
_CPU_atomic_Fetch_or_uint
Atomic Operations CPU, 87
_CPU_atomic_Fetch_or_uintptr
Atomic Operations CPU, 88
_CPU_atomic_Fetch_or_ulong
Atomic Operations CPU, 88
_CPU_atomic_Fetch_sub_uint
Atomic Operations CPU, 88
_CPU_atomic_Fetch_sub_uintptr
Atomic Operations CPU, 89
_CPU_atomic_Fetch_sub_ulong
Atomic Operations CPU, 89
_CPU_atomic_Flag_clear
Atomic Operations CPU, 90
_CPU_atomic_Flag_test_and_set
Atomic Operations CPU, 90
_CPU_atomic_Init_uint
Atomic Operations CPU, 91
_CPU_atomic_Init_uintptr
Atomic Operations CPU, 91
_CPU_atomic_Init_ulong
Atomic Operations CPU, 91
_CPU_atomic_Load_uint
Atomic Operations CPU, 92
_CPU_atomic_Load_uintptr
Atomic Operations CPU, 92
_CPU_atomic_Load_ulong
Atomic Operations CPU, 92
_CPU_atomic_Store_uint
Atomic Operations CPU, 93
_CPU_atomic_Store_uintptr
Atomic Operations CPU, 93
_CPU_atomic_Store_ulong
Atomic Operations CPU, 94
_Chain_Append_if_is_off_chain_unprotected
Chain Handler, 156
_Chain_Append_unprotected
Chain Handler, 156
_Chain_Append_with_empty_check_unprotected
Chain Handler, 157
_Chain_Are_nodes_equal
Chain Handler, 157
_Chain_Extract_unprotected
Chain Handler, 158
_Chain_First
Chain Handler, 158
_Chain_Get_first_unprotected
Chain Handler, 158
_Chain_Get_unprotected
Chain Handler, 159
_Chain_Get_with_empty_check_unprotected
Chain Handler, 160
_Chain_Has_only_one_node
Chain Handler, 160
_Chain_Head
Chain Handler, 161
_Chain_Immutable_first
Chain Handler, 161
_Chain_Immutable_head
Chain Handler, 161
_Chain_Immutable_last
Chain Handler, 162
_Chain_Immutable_next
Chain Handler, 162
_Chain_Immutable_previous
Chain Handler, 163
_Chain_Immutable_tail
Chain Handler, 163
_Chain_Initialize
Chain Handler, 164
_Chain_Initialize_empty
Chain Handler, 164
_Chain_Initialize_node
Chain Handler, 164
_Chain_Initialize_one
Chain Handler, 165
_Chain_Insert_ordered_unprotected
Chain Handler, 165
_Chain_Insert_unprotected
Chain Handler, 166
_Chain_Is_empty
Chain Handler, 166
_Chain_Is_first
Chain Handler, 166
_Chain_Is_head
Chain Handler, 167
_Chain_Is_last
Chain Handler, 167
_Chain_Is_node_off_chain
Chain Handler, 168
_Chain_Is_tail
Chain Handler, 168
_Chain_Iterator_destroy

Chain Handler, 169
_Chain_Iterator_initialize
Chain Handler, 169
_Chain_Iterator_next
Chain Handler, 170
_Chain_Iterator_registry_initialize
Chain Handler, 171
_Chain_Iterator_registry_update
Chain Handler, 171
_Chain_Iterator_set_position
Chain Handler, 171
_Chain_Last
Chain Handler, 172
_Chain_Next
Chain Handler, 172
_Chain_Node_count_unprotected
Chain Handler, 173
_Chain_Prepend_unprotected
Chain Handler, 173
_Chain_Prepend_with_empty_check_unprotected
Chain Handler, 173
_Chain_Previous
Chain Handler, 174
_Chain_Set_off_chain
Chain Handler, 174
_Chain_Tail
Chain Handler, 175
_Context_Initialization_at_thread_begin
Context Handler, 251
_Context_Initialize
Context Handler, 251
_Context_Initialize_fp
Context Handler, 252
_Context_Restart_self
Context Handler, 252
_Context_Restore_fp
Context Handler, 253
_Context_Save_fp
Context Handler, 253
_Context_Switch
Context Handler, 254
_Copyright_Notice
RTEMS Copyright Notice, 709
_Dual_ported_memory_Allocate
Dual Ported Memory Manager Implementation, 291
_Dual_ported_memory_Free
Dual Ported Memory Manager Implementation, 291
_Event_Initialize
Event Implementation, 316
_Event_Seize
Event Implementation, 316
_Event_Surrender
Event Implementation, 318
_Event_sets_Clear
Event Implementation, 317
_Event_sets_Get
Event Implementation, 317
_Event_sets_Is_empty
Event Implementation, 317
_Event_sets_Post
Event Implementation, 318
_Freechain_Extend
Freechain Handler, 398
_Freechain_Get
Freechain Handler, 398
_Freechain_Initialize
Freechain Handler, 399
_Freechain_Is_empty
Freechain Handler, 399
_Freechain_Pop
Freechain Handler, 399
_Freechain_Push
Freechain Handler, 400
_Freechain_Put
Freechain Handler, 400
_Heap_Align_down
Heap Handler, 430
_Heap_Align_up
Heap Handler, 430
_Heap_Alloc_area_of_block
Heap Handler, 430
_Heap_Allocate
Heap Handler, 431
_Heap_Allocate_aligned
Heap Handler, 431
_Heap_Allocate_aligned_with_boundary
Heap Handler, 432
_Heap_Area_overhead
Heap Handler, 432
_Heap_Block_allocate
Heap Handler, 433
_Heap_Block_at
Heap Handler, 433
_Heap_Block_of_alloc_area
Heap Handler, 434
_Heap_Block_set_size
Heap Handler, 434
_Heap_Block_size
Heap Handler, 435
_Heap_Extend
Heap Handler, 435
_Heap_Free
Heap Handler, 436
_Heap_Free_list_first
Heap Handler, 436
_Heap_Free_list_head
Heap Handler, 437
_Heap_Free_list_insert_after
Heap Handler, 437
_Heap_Free_list_insert_before
Heap Handler, 437
_Heap_Free_list_last
Heap Handler, 438
_Heap_Free_list_remove
Heap Handler, 438
_Heap_Free_list_replace

Heap Handler, 438
_Heap_Free_list_tail
Heap Handler, 439
_Heap_Get_first_and_last_block
Heap Handler, 439
_Heap_Get_free_information
Heap Handler, 440
_Heap_Get_information
Heap Handler, 440
_Heap_Get_size
Heap Handler, 440
_Heap_Greedy_allocate
Heap Handler, 441
_Heap_Greedy_allocate_all_except_largest
Heap Handler, 441
_Heap_Greedy_free
Heap Handler, 442
_Heap_Initialize
Heap Handler, 442
_Heap_Is_aligned
Heap Handler, 443
_Heap_Is_block_in_heap
Heap Handler, 443
_Heap_Is_free
Heap Handler, 444
_Heap_Is_prev_used
Heap Handler, 444
_Heap_Is_used
Heap Handler, 444
_Heap_Iterate
Heap Handler, 445
_Heap_Max
Heap Handler, 445
_Heap_Min
Heap Handler, 446
_Heap_Min_block_size
Heap Handler, 446
_Heap_No_extend
Heap Handler, 446
_Heap_Prev_block
Heap Handler, 447
_Heap_Protection_set_delayed_free_fraction
Heap Handler, 447
_Heap_Resize_block
Heap Handler, 448
_Heap_Set_last_block_size
Heap Handler, 448
_Heap_Size_of_alloc_area
Heap Handler, 449
_Heap_Size_with_overhead
Heap Handler, 449
_Heap_Walk
Heap Handler, 450
_ISR_Get_level
ISR Handler, 475
_ISR_Handler
ISR Handler, 479
_ISR_Handler_initialization
ISR Handler, 479
_ISR_Install_vector
ISR Handler, 475
_ISR_Is_enabled
ISR Handler, 476
_ISR_Is_in_progress
ISR Handler, 480
_ISR_Local_disable
ISR Handler, 476
_ISR_Local_enable
ISR Handler, 477
_ISR_Local_flash
ISR Handler, 477
_ISR_Set_level
ISR Handler, 478
_ISR_Stack_area_begin
ISR Handler, 480
_ISR_Stack_area_end
ISR Handler, 481
_ISR_lock_Acquire
ISR Locks, 483
_ISR_lock_Acquire_inline
ISR Locks, 483
_ISR_lock_Context_set_level
ISR Locks, 490
_ISR_lock_Destroy
ISR Locks, 484
_ISR_lock_ISR_disable
ISR Locks, 485
_ISR_lock_ISR_disable_and_acquire
ISR Locks, 485
_ISR_lock_ISR_enable
ISR Locks, 486
_ISR_lock_Initialize
ISR Locks, 484
_ISR_lock_Release
ISR Locks, 486
_ISR_lock_Release_and_ISR_enable
ISR Locks, 487
_ISR_lock_Release_inline
ISR Locks, 487
_ISR_lock_Set_name
ISR Locks, 488
_Internal_error
Internal Error Handler, 499
_Internal_errors_What_happened
Internal Error Handler, 500
_MRSP_Acquire_critical
Multiprocessor Resource Sharing Protocol Handler, 573
_MRSP_Can_destroy
Multiprocessor Resource Sharing Protocol Handler, 574
_MRSP_Claim_ownership
Multiprocessor Resource Sharing Protocol Handler, 574
_MRSP_Destroy

Multiprocessor Resource Sharing Protocol Handler, 574
_MRSP_Get_owner
Multiprocessor Resource Sharing Protocol Handler, 575
_MRSP_Get_priority
Multiprocessor Resource Sharing Protocol Handler, 575
_MRSP_Initialize
Multiprocessor Resource Sharing Protocol Handler, 576
_MRSP_Raise_priority
Multiprocessor Resource Sharing Protocol Handler, 576
_MRSP_Release
Multiprocessor Resource Sharing Protocol Handler, 577
_MRSP_Remove_priority
Multiprocessor Resource Sharing Protocol Handler, 577
_MRSP_Replace_priority
Multiprocessor Resource Sharing Protocol Handler, 577
_MRSP_Seize
Multiprocessor Resource Sharing Protocol Handler, 578
_MRSP_Set_owner
Multiprocessor Resource Sharing Protocol Handler, 578
_MRSP_Set_priority
Multiprocessor Resource Sharing Protocol Handler, 579
_MRSP_Surrender
Multiprocessor Resource Sharing Protocol Handler, 579
_MRSP_Wait_for_ownership
Multiprocessor Resource Sharing Protocol Handler, 580
_Memory_Allocate
Memory Handler, 538
_Memory_Consume
Memory Handler, 538
_Memory_Fill
Memory Handler, 539
_Memory_Get
Memory Handler, 539
_Memory_Get_area
Memory Handler, 539
_Memory_Get_begin
Memory Handler, 540
_Memory_Get_count
Memory Handler, 540
_Memory_Get_end
Memory Handler, 540
_Memory_Get_free_begin
Memory Handler, 541
_Memory_Get_free_size
Memory Handler, 541
_Memory_Get_size
Memory Handler, 542
_Memory_Initialize
Memory Handler, 542
_Memory_Initialize_by_size
Memory Handler, 542
_Memory_Set_begin
Memory Handler, 543
_Memory_Set_end
Memory Handler, 543
_Memory_Set_free_begin
Memory Handler, 543
_Memory_Zero_before_use
Memory Handler, 544
_Message_queue_Create
Classic Message Queue Implementation, 225
_Message_queue_Free
Classic Message Queue Implementation, 225
_Message_queue_Submit
Classic Message Queue Implementation, 225
_Modes_Change
Classic Modes Implementation, 226
_Modes_Get_interrupt_level
Classic Modes Implementation, 226
_Modes_Is_asr_disabled
Classic Modes Implementation, 227
_Modes_Is_preempt
Classic Modes Implementation, 227
_Modes_Is_timeslice
Classic Modes Implementation, 227
_Modes_Mask_changed
Classic Modes Implementation, 227
_Modes_Set_interrupt_level
Classic Modes Implementation, 228
_Objects_API_maximum_class
Object Handler, 612
_Objects_Activate_unlimited
Object Handler, 607
_Objects_Active_count
Object Handler, 608
_Objects_Allocate
Object Handler, 608
_Objects_Allocate_none
Object Handler, 609
_Objects_Allocate_static
Object Handler, 609
_Objects_Allocate_unlimited
Object Handler, 610
_Objects_Allocate_unprotected
Object Handler, 610
_Objects_Allocate_with_extend
Object Handler, 611
_Objects_Allocator_is_owner
Object Handler, 611
_Objects_Allocator_lock
Object Handler, 611
_Objects_Allocator_unlock
Object Handler, 612

_Objects_Are_ids_equal
Object Handler, 613
_Objects_Build_id
Object Handler, 596
_Objects_Build_name
Object Handler, 597
_Objects_Close
Object Handler, 613
_Objects_Extend_information
Object Handler, 613
_Objects_Extend_size
Object Handler, 614
_Objects_Free
Object Handler, 614
_Objects_Free_objects_block
Object Handler, 615
_Objects_Free_static
Object Handler, 615
_Objects_Free_unlimited
Object Handler, 616
_Objects_Get
Object Handler, 616
_Objects_Get_API
Object Handler, 617
_Objects_Get_by_name
Object Handler, 617
_Objects_Get_class
Object Handler, 618
_Objects_Get_inactive
Object Handler, 618
_Objects_Get_index
Object Handler, 618
_Objects_Get_information
Object Handler, 619
_Objects_Get_information_id
Object Handler, 619
_Objects_Get_maximum_index
Object Handler, 620
_Objects_Get_minimum_id
Object Handler, 620
_Objects_Get_name_as_string
Object Handler, 620
_Objects_Get_next
Object Handler, 621
_Objects_Get_no_protection
Object Handler, 621
_Objects_Get_node
Object Handler, 622
_Objects_Has_string_name
Object Handler, 622
_Objects_Id_to_name
Object Handler, 624
_Objects_Information_table
Object Handler, 631
_Objects_Initialize_information
Object Handler, 624
_Objects_Invalidate_Id
Object Handler, 625
_Objects_Is_api_valid
Object Handler, 625
_Objects_Is_auto_extend
Object Handler, 626
_Objects_Is_local_id
Object Handler, 626
_Objects_Is_local_node
Object Handler, 626
_Objects_Is_unlimited
Object Handler, 598
_Objects_Maximum_nodes
Object Handler, 598
_Objects_Name_to_id_u32
Object Handler, 627
_Objects_Name_to_string
Object Handler, 628
_Objects_Namespace_remove_string
Object Handler, 628
_Objects_Namespace_remove_u32
Object Handler, 628
_Objects_Open
Object Handler, 629
_Objects_Open_string
Object Handler, 629
_Objects_Open_u32
Object Handler, 630
_Objects_Set_local_object
Object Handler, 630
_Objects_Set_name
Object Handler, 630
_Objects_Shrink_information
Object Handler, 631
_Options_Is_any
Classic Options Implementation, 230
_Options_Is_no_wait
Classic Options Implementation, 230
_Partition_Acquire_critical
Partition Manager Implementation, 657
_Partition_Get
Partition Manager Implementation, 657
_Partition_Information
Partition Manager Implementation, 658
_Partition_Release
Partition Manager Implementation, 658
_Per_CPU_Add_job
RTEMS Per CPU Information, 714
_Per_CPU_Initialize
RTEMS Per CPU Information, 715
_Per_CPU_Perform_jobs
RTEMS Per CPU Information, 715
_Per_CPU_State_wait_for_non_initial_state
RTEMS Per CPU Information, 715
_Per_CPU_Wait_for_job
RTEMS Per CPU Information, 716
_Priority_Actions_add
Priority Handler, 665
_Priority_Actions_initialize_empty
Priority Handler, 666

_Priority_Actions_initialize_one
Priority Handler, 666
_Priority_Actions_is_empty
Priority Handler, 667
_Priority_Actions_is_valid
Priority Handler, 667
_Priority_Actions_move
Priority Handler, 667
_Priority_Bits_index
Priority Handler, 671
_Priority_Change_nothing
Priority Handler, 671
_Priority_Changed
Priority Handler, 672
_Priority_Extract
Priority Handler, 672
_Priority_Extract_non_empty
Priority Handler, 674
_Priority_Get_minimum_node
Priority Handler, 674
_Priority_Get_next_action
Priority Handler, 675
_Priority_Get_priority
Priority Handler, 675
_Priority_Get_scheduler
Priority Handler, 675
_Priority_Initialize_empty
Priority Handler, 676
_Priority_Initialize_one
Priority Handler, 676
_Priority_Insert
Priority Handler, 677
_Priority_Is_empty
Priority Handler, 677
_Priority_Less
Priority Handler, 677
_Priority_Major
Priority Handler, 678
_Priority_Mask
Priority Handler, 678
_Priority_Minor
Priority Handler, 679
_Priority_Node_initialize
Priority Handler, 679
_Priority_Node_is_active
Priority Handler, 679
_Priority_Node_set_inactive
Priority Handler, 680
_Priority_Node_set_priority
Priority Handler, 680
_Priority_Non_empty_insert
Priority Handler, 680
_Priority_Plain_changed
Priority Handler, 681
_Priority_Plain_extract
Priority Handler, 681
_Priority_Plain_insert
Priority Handler, 682
_Priority_Remove_nothing
Priority Handler, 682
_Priority_Replace
Priority Handler, 683
_Priority_Set_action
Priority Handler, 683
_Priority_Set_action_node
Priority Handler, 683
_Priority_Set_action_type
Priority Handler, 684
_Priority_bit_map_Add
Priority Handler, 669
_Priority_bit_map_Get_highest
Priority Handler, 669
_Priority_bit_map_Initialize
Priority Handler, 669
_Priority_bit_map_Initialize_information
Priority Handler, 670
_Priority_bit_map_Is_empty
Priority Handler, 670
_Priority_bit_map_Remove
Priority Handler, 671
_Processor_mask_And
Processor Mask, 686
_Processor_mask_Assign
Processor Mask, 687
_Processor_mask_Clear
Processor Mask, 687
_Processor_mask_Copy
Processor Mask, 688
_Processor_mask_Count
Processor Mask, 688
_Processor_mask_Fill
Processor Mask, 689
_Processor_mask_Find_last_set
Processor Mask, 689
_Processor_mask_From_cpu_set_t
Processor Mask, 689
_Processor_mask_From_index
Processor Mask, 690
_Processor_mask_From_uint32_t
Processor Mask, 690
_Processor_mask_Has_overlap
Processor Mask, 690
_Processor_mask_Is_at_most_partial_loss
Processor Mask, 691
_Processor_mask_Is_equal
Processor Mask, 691
_Processor_mask_Is_set
Processor Mask, 692
_Processor_mask_Is_subset
Processor Mask, 692
_Processor_mask_Is_zero
Processor Mask, 693
_Processor_mask_Nand
Processor Mask, 693
_Processor_mask_Or
Processor Mask, 693

_Processor_mask_Set
Processor Mask, 694
_Processor_mask_To_cpu_set_t
Processor Mask, 694
_Processor_mask_To_uint32_t
Processor Mask, 695
_Processor_mask_Xor
Processor Mask, 695
_Processor_mask_Zero
Processor Mask, 696
_Profiling_Outer_most_interrupt_entry_and_exit
Profiling Support, 697
_Profiling_Thread_dispatch_disable
Profiling Support, 698
_Profiling_Thread_dispatch_disable_critical
Profiling Support, 698
_Profiling_Thread_dispatch_enable
Profiling Support, 698
_Profiling_Update_max_interrupt_delay
Profiling Support, 699
_Protected_heap_Allocate
Protected Heap Handler, 701
_Protected_heap_Allocate_aligned
Protected Heap Handler, 701
_Protected_heap_Allocate_aligned_with_boundary
Protected Heap Handler, 702
_Protected_heap_Extend
Protected Heap Handler, 702
_Protected_heap_Free
Protected Heap Handler, 703
_Protected_heap_Get_block_size
Protected Heap Handler, 703
_Protected_heap_Get_free_information
Protected Heap Handler, 704
_Protected_heap_Get_information
Protected Heap Handler, 704
_Protected_heap_Get_size
Protected Heap Handler, 704
_Protected_heap_Initialize
Protected Heap Handler, 705
_Protected_heap_Iterate
Protected Heap Handler, 705
_Protected_heap_Resize_block
Protected Heap Handler, 706
_Protected_heap_Walk
Protected Heap Handler, 706
_RBTree_Add_child
Red-Black Tree Handler, 756
_RBTree_Extract
Red-Black Tree Handler, 756
_RBTree_Find_inline
Red-Black Tree Handler, 756
_RBTree_Initialize_empty
Red-Black Tree Handler, 757
_RBTree_Initialize_node
Red-Black Tree Handler, 757
_RBTree_Initialize_one
Red-Black Tree Handler, 758
_RBTree_Insert_color
Red-Black Tree Handler, 758
_RBTree_Insert_inline
Red-Black Tree Handler, 758
_RBTree_Insert_with_parent
Red-Black Tree Handler, 759
_RBTree_Is_empty
Red-Black Tree Handler, 760
_RBTree_Is_node_off_tree
Red-Black Tree Handler, 760
_RBTree_Is_root
Red-Black Tree Handler, 761
_RBTree_Iterate
Red-Black Tree Handler, 761
_RBTree_Left
Red-Black Tree Handler, 762
_RBTree_Left_reference
Red-Black Tree Handler, 762
_RBTree_Maximum
Red-Black Tree Handler, 762
_RBTree_Minimum
Red-Black Tree Handler, 763
_RBTree_Parent
Red-Black Tree Handler, 763
_RBTree_Postorder_first
Red-Black Tree Handler, 764
_RBTree_Postorder_next
Red-Black Tree Handler, 764
_RBTree_Predecessor
Red-Black Tree Handler, 765
_RBTree_Replace_node
Red-Black Tree Handler, 765
_RBTree_Right
Red-Black Tree Handler, 766
_RBTree_Right_reference
Red-Black Tree Handler, 766
_RBTree_Root
Red-Black Tree Handler, 767
_RBTree_Root_const_reference
Red-Black Tree Handler, 767
_RBTree_Root_reference
Red-Black Tree Handler, 768
_RBTree_Set_off_tree
Red-Black Tree Handler, 768
_RBTree_Successor
Red-Black Tree Handler, 768
_RTEMS_Name_to_id
Classic Object Implementation, 229
_RTEMS_Priority_From_core
Classic Tasks Manager Implementation, 238
_RTEMS_Priority_To_core
Classic Tasks Manager Implementation, 239
_RTEMS_tasks_Free
Classic Tasks Manager Implementation, 239
_RTEMS_tasks_Information
Classic Tasks Manager Implementation, 240
_RTEMS_tasks_Initialize_user_tasks
Classic Tasks Manager Implementation, 240

_RTEMS_tasks_User_extensions
taskconstruct.c, 1831
_RTEMS_tasks_User_task_table
Classic Tasks Manager Implementation, 240
_RTEMS_version
RTEMS Copyright Notice, 709
_Rate_monotonic_Allocate
Classic Rate Monotonic Scheduler Implementation, 232
_Rate_monotonic_Get_status
Classic Rate Monotonic Scheduler Implementation, 232
_Region_Allocate
Classic Region Manager Implementation, 235
_Region_Allocate_segment
Classic Region Manager Implementation, 235
_Region_Free
Classic Region Manager Implementation, 235
_Region_Free_segment
Classic Region Manager Implementation, 236
_Region_Process_queue
Classic Region Manager Implementation, 236
_SMP_Broadcast_action
SMP Support, 821
_SMP_Fatal
SMP Support, 821
_SMP_Get_online_processors
SMP Support, 822
_SMP_Handler_initialize
SMP Support, 822
_SMP_Inter_processor_interrupt_handler
SMP Support, 822
_SMP_MCS_lock_Acquire
SMP Locks, 783
_SMP_MCS_lock_Destroy
SMP Locks, 787
_SMP_MCS_lock_Do_acquire
SMP Locks, 788
_SMP_MCS_lock_Initialize
SMP Locks, 788
_SMP_MCS_lock_Release
SMP Locks, 788
_SMP_Multicast_action
SMP Support, 823
_SMP_Need_inter_processor_interrupts
SMP Support, 823
_SMP_Online_processors
SMP Support, 827
_SMP_Othercast_action
SMP Support, 823
_SMP_Processor_configured_maximum
SMP Support, 827
_SMP_Request_shutdown
SMP Support, 824
_SMP_Send_message
SMP Support, 824
_SMP_Send_message_broadcast
SMP Support, 825
_SMP_Send_message_multicast
SMP Support, 825
_SMP_Should_start_processor
SMP Support, 825
_SMP_Start_multitasking_on_secondary_processor
SMP Support, 826
_SMP_Synchronize
SMP Support, 826
_SMP_Unicast_action
SMP Support, 827
_SMP_barrier_Control_initialize
SMP Barriers, 777
_SMP_barrier_State_initialize
SMP Barriers, 778
_SMP_barrier_Wait
SMP Barriers, 778
_SMP_lock_Acquire
SMP Locks, 784
_SMP_lock_Acquire_inline
SMP Locks, 784
_SMP_lock_Destroy
SMP Locks, 781
_SMP_lock_Destroy_inline
SMP Locks, 785
_SMP_lock_ISR_disable_and_acquire
SMP Locks, 786
_SMP_lock_ISR_disable_and_acquire_inline
SMP Locks, 786
_SMP_lock_Initialize
SMP Locks, 781
_SMP_lock_Initialize_inline
SMP Locks, 785
_SMP_lock_Release
SMP Locks, 782
_SMP_lock_Release_and_ISR_enable
SMP Locks, 782
_SMP_lock_Release_and_ISR_enable_inline
SMP Locks, 786
_SMP_lock_Release_inline
SMP Locks, 787
_SMP_lock_Set_name
SMP Locks, 787
_SMP_sequence_lock_Destroy
SMP Locks, 789
_SMP_sequence_lock_Initialize
SMP Locks, 789
_SMP_sequence_lock_Read_begin
SMP Locks, 789
_SMP_sequence_lock_Read_retry
SMP Locks, 790
_SMP_sequence_lock_Write_begin
SMP Locks, 790
_SMP_sequence_lock_Write_end
SMP Locks, 791
_SMP_ticket_lock_Acquire
SMP Locks, 783
_SMP_ticket_lock_Destroy
SMP Locks, 791

_SMP_ticket_lock_Do_acquire
SMP Locks, 791
_SMP_ticket_lock_Do_release
SMP Locks, 792
_SMP_ticket_lock_Initialize
SMP Locks, 792
_SMP_ticket_lock_Release
SMP Locks, 783
_Scheduler_Acquire_critical
Scheduler Handler, 855
_Scheduler_Ask_for_help
Scheduler Handler, 855
_Scheduler_Block
Scheduler Handler, 855
_Scheduler_Block_node
Scheduler Handler, 856
_Scheduler_Build_id
Scheduler Handler, 856
_Scheduler_Cancel_job
Scheduler Handler, 857
_Scheduler_Control, 1299
maximum_priority, 1299
_Scheduler_Count
Scheduler Handler, 883
_Scheduler_Discard_idle_thread
Scheduler Handler, 864
_Scheduler_EDF_Block
EDF Scheduler, 306
_Scheduler_EDF_Cancel_job
EDF Scheduler, 307
_Scheduler_EDF_Enqueue
EDF Scheduler, 307
_Scheduler_EDF_Extract
EDF Scheduler, 307
_Scheduler_EDF_Extract_body
EDF Scheduler, 308
_Scheduler_EDF_Get_context
EDF Scheduler, 308
_Scheduler_EDF_Initialize
EDF Scheduler, 308
_Scheduler_EDF_Less
EDF Scheduler, 309
_Scheduler_EDF_Map_priority
EDF Scheduler, 309
_Scheduler_EDF_Node_downcast
EDF Scheduler, 310
_Scheduler_EDF_Node_initialize
EDF Scheduler, 310
_Scheduler_EDF_Priority_less_equal
EDF Scheduler, 310
_Scheduler_EDF_Release_job
EDF Scheduler, 311
_Scheduler_EDF_SMP_Add_processor
EDF Priority SMP Scheduler, 297
_Scheduler_EDF_SMP_Ask_for_help
EDF Priority SMP Scheduler, 297
_Scheduler_EDF_SMP_Block
EDF Priority SMP Scheduler, 297
_Scheduler_EDF_SMP_Initialize
EDF Priority SMP Scheduler, 298
_Scheduler_EDF_SMP_Node_initialize
EDF Priority SMP Scheduler, 298
_Scheduler_EDF_SMP_Pin
EDF Priority SMP Scheduler, 298
_Scheduler_EDF_SMP_Reconsider_help_request
EDF Priority SMP Scheduler, 299
_Scheduler_EDF_SMP_Remove_processor
EDF Priority SMP Scheduler, 299
_Scheduler_EDF_SMP_Set_affinity
EDF Priority SMP Scheduler, 300
_Scheduler_EDF_SMP_Start_idle
EDF Priority SMP Scheduler, 300
_Scheduler_EDF_SMP_Unblock
EDF Priority SMP Scheduler, 301
_Scheduler_EDF_SMP_Unpin
EDF Priority SMP Scheduler, 301
_Scheduler_EDF_SMP_Update_priority
EDF Priority SMP Scheduler, 301
_Scheduler_EDF_SMP_Withdraw_node
EDF Priority SMP Scheduler, 303
_Scheduler_EDF_SMP_Yield
EDF Priority SMP Scheduler, 303
_Scheduler_EDF_Schedule
EDF Scheduler, 311
_Scheduler_EDF_Schedule_body
EDF Scheduler, 312
_Scheduler_EDF_Thread_get_node
EDF Scheduler, 312
_Scheduler_EDF_Unblock
EDF Scheduler, 312
_Scheduler_EDF_Unmap_priority
EDF Scheduler, 313
_Scheduler_EDF_Update_priority
EDF Scheduler, 313
_Scheduler_EDF_Yield
EDF Scheduler, 314
_Scheduler_Exchange_idle_thread
Scheduler Handler, 864
_Scheduler_Generic_block
Scheduler Handler, 864
_Scheduler_Get_affinity
Scheduler Handler, 865
_Scheduler_Get_by_CPU
Scheduler Handler, 865
_Scheduler_Get_by_id
Scheduler Handler, 866
_Scheduler_Get_context
Scheduler Handler, 866
_Scheduler_Get_index
Scheduler Handler, 866
_Scheduler_Get_index_by_id
Scheduler Handler, 867
_Scheduler_Get_processor_count
Scheduler Handler, 867
_Scheduler_Get_processors
Scheduler Handler, 867

_Scheduler_Handler_initialization
Scheduler Handler, 868
_Scheduler_Has_processor_ownership
Scheduler Handler, 868
_Scheduler_Initial_assignments
Scheduler Handler, 883
_Scheduler_Is_non_preempt_mode_supported
Scheduler Handler, 869
_Scheduler_Map_priority
Scheduler Handler, 869
_Scheduler_Node_destroy
Scheduler Handler, 869
_Scheduler_Node_do_initialize
Scheduler Handler, 870
_Scheduler_Node_get_idle
Scheduler Handler, 870
_Scheduler_Node_get_owner
Scheduler Handler, 871
_Scheduler_Node_get_priority
Scheduler Handler, 871
_Scheduler_Node_get_scheduler
Scheduler Handler, 871
_Scheduler_Node_get_user
Scheduler Handler, 872
_Scheduler_Node_initialize
Scheduler Handler, 872
_Scheduler_Node_set_priority
Scheduler Handler, 873
_Scheduler_Node_set_user
Scheduler Handler, 873
_Scheduler_Node_size
Scheduler Handler, 883
_Scheduler_Priority_and_sticky_update
Scheduler Handler, 873
_Scheduler_Release_critical
Scheduler Handler, 874
_Scheduler_Release_idle_thread
Scheduler Handler, 874
_Scheduler_Release_job
Scheduler Handler, 875
_Scheduler_SMP_Add_processor
SMP Scheduler, 800
_Scheduler_SMP_Allocate_processor
SMP Scheduler, 800
_Scheduler_SMP_Allocate_processor_exact
SMP Scheduler, 801
_Scheduler_SMP_Allocate_processor_lazy
SMP Scheduler, 801
_Scheduler_SMP_Ask_for_help
SMP Scheduler, 802
_Scheduler_SMP_Block
SMP Scheduler, 802
_Scheduler_SMP_Do_nothing_register_idle
SMP Scheduler, 803
_Scheduler_SMP_Do_start_idle
SMP Scheduler, 803
_Scheduler_SMP_Enqueue
SMP Scheduler, 804
_Scheduler_SMP_Enqueue_scheduled
SMP Scheduler, 805
_Scheduler_SMP_Enqueue_to_scheduled
SMP Scheduler, 805
_Scheduler_SMP_Exctract_idle_thread
SMP Scheduler, 806
_Scheduler_SMP_Extract_from_scheduled
SMP Scheduler, 806
_Scheduler_SMP_Get_idle_thread
SMP Scheduler, 806
_Scheduler_SMP_Get_lowest_scheduled
SMP Scheduler, 807
_Scheduler_SMP_Get_self
SMP Scheduler, 807
_Scheduler_SMP_Initialize
SMP Scheduler, 808
_Scheduler_SMP_Insert_scheduled
SMP Scheduler, 808
_Scheduler_SMP_Is_processor_owned_by_us
SMP Scheduler, 808
_Scheduler_SMP_Node_change_state
SMP Scheduler, 809
_Scheduler_SMP_Node_downcast
SMP Scheduler, 809
_Scheduler_SMP_Node_initialize
SMP Scheduler, 809
_Scheduler_SMP_Node_priority
SMP Scheduler, 810
_Scheduler_SMP_Node_state
SMP Scheduler, 810
_Scheduler_SMP_Node_update_priority
SMP Scheduler, 811
_Scheduler_SMP_Preempt
SMP Scheduler, 811
_Scheduler_SMP_Preempt_and_schedule_highest_ready
SMP Scheduler, 811
_Scheduler_SMP_Priority_less_equal
SMP Scheduler, 812
_Scheduler_SMP_Reconsider_help_request
SMP Scheduler, 812
_Scheduler_SMP_Release_idle_thread
SMP Scheduler, 813
_Scheduler_SMP_Remove_processor
SMP Scheduler, 813
_Scheduler_SMP_Schedule_highest_ready
SMP Scheduler, 814
_Scheduler_SMP_Set_affinity
SMP Scheduler, 814
_Scheduler_SMP_Start_idle
SMP Scheduler, 815
_Scheduler_SMP_Thread_get_node
SMP Scheduler, 815
_Scheduler_SMP_Thread_get_own_node
SMP Scheduler, 816
_Scheduler_SMP_Unblock
SMP Scheduler, 816
_Scheduler_SMP_Update_priority
SMP Scheduler, 816

_Scheduler_SMP_Withdraw_node
SMP Scheduler, 817
_Scheduler_SMP_Yield
SMP Scheduler, 818
_Scheduler_Schedule
Scheduler Handler, 875
_Scheduler_Set
Scheduler Handler, 875
_Scheduler_Set_affinity
Scheduler Handler, 876
_Scheduler_Set_idle_thread
Scheduler Handler, 876
_Scheduler_Start_idle
Scheduler Handler, 878
_Scheduler_Table
Scheduler Handler, 884
_Scheduler_Thread_change_state
Scheduler Handler, 878
_Scheduler_Tick
Scheduler Handler, 878
_Scheduler_Try_to_schedule_node
Scheduler Handler, 879
_Scheduler_Unblock
Scheduler Handler, 880
_Scheduler_Unblock_node
Scheduler Handler, 880
_Scheduler_Unmap_priority
Scheduler Handler, 881
_Scheduler_Update_heir
Scheduler Handler, 881
_Scheduler_Update_priority
Scheduler Handler, 881
_Scheduler_Use_idle_thread
Scheduler Handler, 882
_Scheduler_Yield
Scheduler Handler, 882
_Scheduler_default_Ask_for_help
Scheduler Handler, 857
_Scheduler_default_Cancel_job
Scheduler Handler, 858
_Scheduler_default_Map_priority
Scheduler Handler, 858
_Scheduler_default_Node_destroy
Scheduler Handler, 859
_Scheduler_default_Node_initialize
Scheduler Handler, 859
_Scheduler_default_Pin_or_unpin
Scheduler Handler, 859
_Scheduler_default_Reconsider_help_request
Scheduler Handler, 860
_Scheduler_default_Release_job
Scheduler Handler, 860
_Scheduler_default_Schedule
Scheduler Handler, 861
_Scheduler_default_Set_affinity
Scheduler Handler, 861
_Scheduler_default_Set_affinity_body
Scheduler Handler, 862
_Scheduler_default_Start_idle
Scheduler Handler, 862
_Scheduler_default_Tick
Scheduler Handler, 862
_Scheduler_default_Unmap_priority
Scheduler Handler, 863
_Scheduler_default_Withdraw_node
Scheduler Handler, 863
_Scheduler_priority_Block
Deterministic Priority Scheduler, 259
_Scheduler_priority_Extract_body
Deterministic Priority Scheduler, 259
_Scheduler_priority_Get_context
Deterministic Priority Scheduler, 260
_Scheduler_priority_Initialize
Deterministic Priority Scheduler, 260
_Scheduler_priority_Node_downcast
Deterministic Priority Scheduler, 260
_Scheduler_priority_Node_initialize
Deterministic Priority Scheduler, 261
_Scheduler_priority_Ready_queue_enqueue
Deterministic Priority Scheduler, 261
_Scheduler_priority_Ready_queue_enqueue_first
Deterministic Priority Scheduler, 262
_Scheduler_priority_Ready_queue_extract
Deterministic Priority Scheduler, 262
_Scheduler_priority_Ready_queue_first
Deterministic Priority Scheduler, 262
_Scheduler_priority_Ready_queue_initialize
Deterministic Priority Scheduler, 264
_Scheduler_priority_Ready_queue_update
Deterministic Priority Scheduler, 264
_Scheduler_priority_Schedule
Deterministic Priority Scheduler, 265
_Scheduler_priority_Schedule_body
Deterministic Priority Scheduler, 265
_Scheduler_priority_Thread_get_node
Deterministic Priority Scheduler, 265
_Scheduler_priority_Unblock
Deterministic Priority Scheduler, 266
_Scheduler_priority_Update_priority
Deterministic Priority Scheduler, 266
_Scheduler_priority_Yield
Deterministic Priority Scheduler, 266
_Scheduler_simple_Block
Simple Priority Scheduler, 926
_Scheduler_simple_Extract
Simple Priority Scheduler, 927
_Scheduler_simple_Get_context
Simple Priority Scheduler, 927
_Scheduler_simple_Initialize
Simple Priority Scheduler, 927
_Scheduler_simple_Insert
Simple Priority Scheduler, 928
_Scheduler_simple_Priority_less_equal
Simple Priority Scheduler, 928
_Scheduler_simple_Schedule
Simple Priority Scheduler, 928

_Scheduler_simple_Schedule_body
Simple Priority Scheduler, 929
_Scheduler_simple_Unblock
Simple Priority Scheduler, 929
_Scheduler_simple_Update_priority
Simple Priority Scheduler, 930
_Scheduler_simple_Yield
Simple Priority Scheduler, 930
_Sem_Get
Semaphore Handler, 896
_Sem_Queue_acquire_critical
Semaphore Handler, 896
_Sem_Queue_release
Semaphore Handler, 897
_Semaphore_Allocate
Semaphore Manager Implementation, 906
_Semaphore_Free
Semaphore Manager Implementation, 906
_Stack_Allocate
Stack Handler, 934
_Stack_Allocator_allocate
Stack Handler, 938
_Stack_Allocator_avoids_workspace
Stack Handler, 938
_Stack_Allocator_do_initialize
Stack Handler, 935
_Stack_Allocator_free
Stack Handler, 938
_Stack_Allocator_initialize
Stack Handler, 938
_Stack_Ensure_minimum
Stack Handler, 935
_Stack_Extend_size
Stack Handler, 935
_Stack_Free
Stack Handler, 936
_Stack_Free_nothing
Stack Handler, 936
_Stack_Initialize
Stack Handler, 936
_Stack_Is_enough
Stack Handler, 937
_Stack_Minimum
Stack Handler, 937
_Stack_Space_size
Stack Handler, 938
_States_Clear
Thread States, 1114
_States_Is_dormant
Thread States, 1115
_States_Is_interruptible_by_signal
Thread States, 1115
_States_Is_locally_blocked
Thread States, 1116
_States_Is_ready
Thread States, 1116
_States_Is_suspended
Thread States, 1116
_States_Is_waiting_for_join_at_exit
Thread States, 1117
_States_Is_waiting_for_rpc_reply
Thread States, 1117
_States_Set
Thread States, 1118
_Status_Object_name_errors_to_status
Classic Status Implementation, 237
_System_state_Get
System State Handler, 954
_System_state_Is_before_initialization
System State Handler, 954
_System_state_Is_before_multitasking
System State Handler, 955
_System_state_Is_terminated
System State Handler, 955
_System_state_Is_up
System State Handler, 955
_System_state_Set
System State Handler, 956
_TLS_Alignment
Thread-Local Storage (TLS), 1124
_TLS_Copy_and_clear
Thread-Local Storage (TLS), 1120
_TLS_Get_allocation_size
Thread-Local Storage (TLS), 1120
_TLS_Get_size
Thread-Local Storage (TLS), 1120
_TLS_Get_thread_control_block_area_size
Thread-Local Storage (TLS), 1121
_TLS_Heap_align_up
Thread-Local Storage (TLS), 1121
_TLS_Initialize
Thread-Local Storage (TLS), 1122
_TLS_TCB_after_TLS_block_initialize
Thread-Local Storage (TLS), 1122
_TLS_TCB_at_area_begin_initialize
Thread-Local Storage (TLS), 1123
_TLS_TCB_before_TLS_block_initialize
Thread-Local Storage (TLS), 1123
_TOD_Acquire
Time of Day Handler, 1131
_TOD_Adjust
Time of Day Handler, 1131
_TOD_Days_per_month
clocktodvalidate.c, 1809
_TOD_Days_to_date
clocktodtoseconds.c, 1807
_TOD_Get
Time of Day Handler, 1132
_TOD_Get_timeval
Time of Day Handler, 1132
_TOD_Get_uptime
Time of Day Handler, 1132
_TOD_Get_zero_based_uptime
Time of Day Handler, 1133
_TOD_Get_zero_based_uptime_as_timespec
Time of Day Handler, 1133

_TOD_Hook_Register
Time of Day Handler Action Hooks, 1138
_TOD_Hook_Run
Time of Day Handler Action Hooks, 1138
_TOD_Hook_Unregister
Time of Day Handler Action Hooks, 1138
_TOD_Is_set
Time of Day Handler, 1133
_TOD_Release
Time of Day Handler, 1134
_TOD_Seconds_since_epoch
Time of Day Handler, 1134
_TOD_Set
Time of Day Handler, 1134
_TOD_To_seconds
clock.h, 1583
clocktodtoseconds.c, 1807
_TOD_Validate
clock.h, 1583
clocktodvalidate.c, 1808
_Terminate
Internal Error Handler, 499
_Thread_Action_control_initialize
Thread Handler, 1014
_Thread_Action_initialize
Thread Handler, 1014
_Thread_Add_post_switch_action
Thread Handler, 1014
_Thread_Add_timeout_ticks
Thread Handler, 1015
_Thread_Allocate_unlimited
Thread Handler, 1015
_Thread_Cancel
Thread Handler, 1015
_Thread_Change_life
Thread Handler, 1016
_Thread_Clear_state
Thread Handler, 1016
_Thread_Clear_state_locked
Thread Handler, 1016
_Thread_Close
Thread Handler, 1017
_Thread_Continue
Thread Handler, 1017
_Thread_Control, 1300
API_Extensions, 1301
budget_algorithm, 1301
budget_callout, 1301
cpu_time_budget, 1301
cpu_time_used, 1301
current_state, 1302
extensions, 1302
is_fp, 1302
is_idle, 1302
is_preemptible, 1302
Join_queue, 1303
libc_reent, 1303
Life, 1303
Object, 1304
Registers, 1304
Start, 1304
Timer, 1304
Wait, 1304
_Thread_Control_add_on_count
Thread Handler, 1066
_Thread_Control_add_ons
Thread Handler, 1066
_Thread_Create_idle
Thread Handler, 1018
_Thread_Dispatch
Thread Handler, 1018
_Thread_Dispatch_direct
Thread Handler, 1018
_Thread_Dispatch_disable
Thread Handler, 1019
_Thread_Dispatch_disable_critical
Thread Handler, 1019
_Thread_Dispatch_disable_with_CPU
Thread Handler, 1020
_Thread_Dispatch_enable
Thread Handler, 1020
_Thread_Dispatch_get_disable_level
Thread Handler, 1020
_Thread_Dispatch_initialization
Thread Handler, 1021
_Thread_Dispatch_is_enabled
Thread Handler, 1021
_Thread_Dispatch_request
Thread Handler, 1021
_Thread_Dispatch_unnest
Thread Handler, 1022
_Thread_Dispatch_update_heir
Thread Handler, 1022
_Thread_Do_dispatch
Thread Handler, 1022
_Thread_Do_unpin
Thread Handler, 1023
_Thread_Entry_adaptor_idle
Thread Handler, 1023
_Thread_Entry_adaptor_numeric
Thread Handler, 1023
_Thread_Entry_adaptor_pointer
Thread Handler, 1024
_Thread_Exit
Thread Handler, 1024
_Thread_Get
Thread Handler, 1024
_Thread_Get_CPU
Thread Handler, 1025
_Thread_Get_CPU_time_used
Thread Handler, 1025
_Thread_Get_executing
RTEMS Per CPU Information, 716
_Thread_Get_heir_and_make_it_executing
Thread Handler, 1025
_Thread_Get_maximum_internal_threads

Thread Handler, 1026
_Thread_Get_name
Thread Handler, 1026
_Thread_Get_objects_information
Thread Handler, 1027
_Thread_Get_priority
Thread Handler, 1027
_Thread_Get_unmapped_priority
Thread Handler, 1028
_Thread_Get_unmapped_real_priority
Thread Handler, 1028
_Thread_Global_constructor
Thread Handler, 1067
_Thread_Handler
Thread Handler, 1028
_Thread_Handler_initialization
Thread Handler, 1029
_Thread_Idle_body
Thread Handler, 1067
_Thread_Idle_stack_size
Thread Handler, 1067
_Thread_Idle_stacks
Thread Handler, 1067
_Thread_Initial_thread_count
Thread Handler, 1068
_Thread_Initialize
Thread Handler, 1029
_Thread_Initialize_information
Thread Handler, 1030
_Thread_Internal_allocate
Thread Handler, 1030
_Thread_Is_context_switch_necessary
Thread Handler, 1030
_Thread_Is_executing
Thread Handler, 1031
_Thread_Is_executing_on_a_processor
Thread Handler, 1031
_Thread_Is_heir
Thread Handler, 1032
_Thread_Is_joinable
Thread Handler, 1032
_Thread_Is_life_change_allowed
Thread Handler, 1032
_Thread_Is_life_changing
Thread Handler, 1033
_Thread_Is_life_restarting
Thread Handler, 1033
_Thread_Is_life_terminating
Thread Handler, 1034
_Thread_Is_ready
Thread Handler, 1034
_Thread_Iterate
Thread Handler, 1034
_Thread_Join
Thread Handler, 1035
_Thread_Kill_zombies
Thread Handler, 1035
_Thread_Load_environment
Thread Handler, 1035
_Thread_Maximum_TLS_size
Thread Handler, 1068
_Thread_Maximum_name_size
Thread Handler, 1068
_Thread_Pin
Thread Handler, 1036
_Thread_Priority_add
Thread Handler, 1036
_Thread_Priority_and_sticky_update
Thread Handler, 1037
_Thread_Priority_change
Thread Handler, 1037
_Thread_Priority_changed
Thread Handler, 1038
_Thread_Priority_highest
Thread Handler, 1038
_Thread_Priority_less_than
Thread Handler, 1039
_Thread_Priority_perform_actions
Thread Handler, 1039
_Thread_Priority_remove
Thread Handler, 1041
_Thread_Priority_replace
Thread Handler, 1041
_Thread_Priority_update
Thread Handler, 1042
_Thread_Remove_timer_and_unblock
Thread Handler, 1042
_Thread_Resource_count_decrement
Thread Handler, 1043
_Thread_Resource_count_increment
Thread Handler, 1043
_Thread_Restart_other
Thread Handler, 1043
_Thread_Restart_self
Thread Handler, 1044
_Thread_Restore_fp
Thread Handler, 1044
_Thread_Save_fp
Thread Handler, 1044
_Thread_Scheduler_acquire_critical
Thread Handler, 1045
_Thread_Scheduler_add_request
Thread Handler, 1045
_Thread_Scheduler_add_wait_node
Thread Handler, 1046
_Thread_Scheduler_cancel_need_for_help
Thread Handler, 1046
_Thread_Scheduler_get_home
Thread Handler, 1046
_Thread_Scheduler_get_home_node
Thread Handler, 1047
_Thread_Scheduler_get_node_by_index
Thread Handler, 1047
_Thread_Scheduler_process_requests
Thread Handler, 1048
_Thread_Scheduler_release_critical

Thread Handler, 1048
_Thread_Scheduler_remove_wait_node
Thread Handler, 1048
_Thread_Set_CPU
Thread Handler, 1049
_Thread_Set_life_protection
Thread Handler, 1049
_Thread_Set_name
Thread Handler, 1049
_Thread_Set_state
Thread Handler, 1050
_Thread_Set_state_locked
Thread Handler, 1050
_Thread_Start
Thread Handler, 1051
_Thread_Start_multitasking
Thread Handler, 1051
_Thread_State_acquire
Thread Handler, 1052
_Thread_State_acquire_critical
Thread Handler, 1052
_Thread_State_acquire_for_executing
Thread Handler, 1052
_Thread_State_release
Thread Handler, 1053
_Thread_State_release_critical
Thread Handler, 1053
_Thread_Timeout
Thread Handler, 1053
_Thread_Timer_initialize
Thread Handler, 1054
_Thread_Timer_insert_realtime
Thread Handler, 1054
_Thread_Timer_remove
Thread Handler, 1054
_Thread_Unblock
Thread Handler, 1055
_Thread_Unpin
Thread Handler, 1055
_Thread_Update_CPU_time_used
Thread Handler, 1055
_Thread_Wait_acquire
Thread Handler, 1056
_Thread_Wait_acquire_critical
Thread Handler, 1056
_Thread_Wait_acquire_default
Thread Handler, 1056
_Thread_Wait_acquire_default_critical
Thread Handler, 1057
_Thread_Wait_acquire_default_for_executing
Thread Handler, 1057
_Thread_Wait_acquire_queue_critical
Thread Handler, 1058
_Thread_Wait_cancel
Thread Handler, 1058
_Thread_Wait_claim
Thread Handler, 1058
_Thread_Wait_claim_finalize
Thread Handler, 1059
_Thread_Wait_flags_get
Thread Handler, 1059
_Thread_Wait_flags_get_acquire
Thread Handler, 1060
_Thread_Wait_flags_set
Thread Handler, 1060
_Thread_Wait_flags_try_change_acquire
Thread Handler, 1060
_Thread_Wait_flags_try_change_release
Thread Handler, 1061
_Thread_Wait_get_id
Thread Handler, 1062
_Thread_Wait_get_status
Thread Handler, 1062
_Thread_Wait_release
Thread Handler, 1062
_Thread_Wait_release_critical
Thread Handler, 1063
_Thread_Wait_release_default
Thread Handler, 1063
_Thread_Wait_release_default_critical
Thread Handler, 1063
_Thread_Wait_release_queue_critical
Thread Handler, 1064
_Thread_Wait_remove_request
Thread Handler, 1064
_Thread_Wait_remove_request_locked
Thread Handler, 1064
_Thread_Wait_restore_default
Thread Handler, 1065
_Thread_Wait_tranquilize
Thread Handler, 1065
_Thread_Yield
Thread Handler, 1066
_Thread_Zombies
threadrestart.c, 1924
_Thread_queue_Acquire
Thread Queue Handler, 1079
_Thread_queue_Acquire_critical
Thread Queue Handler, 1080
_Thread_queue_Add_timeout_monotonic_timespec
Thread Queue Handler, 1080
_Thread_queue_Add_timeout_realtime_timespec
Thread Queue Handler, 1081
_Thread_queue_Add_timeout_ticks
Thread Queue Handler, 1081
_Thread_queue_Context_ISR_disable
Thread Queue Handler, 1074
_Thread_queue_Context_add_priority_update
Thread Queue Handler, 1081
_Thread_queue_Context_clear_priority_updates
Thread Queue Handler, 1082
_Thread_queue_Context_initialize
Thread Queue Handler, 1082
_Thread_queue_Context_restore_priority_updates
Thread Queue Handler, 1082
_Thread_queue_Context_save_priority_updates

Thread Queue Handler, 1083
_Thread_queue_Context_set_ISR_level
Thread Queue Handler, 1086
_Thread_queue_Context_set_MP_callout
Thread Queue Handler, 1074
_Thread_queue_Context_set_deadlock_callout
Thread Queue Handler, 1083
_Thread_queue_Context_set_enqueue_callout
Thread Queue Handler, 1084
_Thread_queue_Context_set_enqueue_do_nothing_extra
Thread Queue Handler, 1084
_Thread_queue_Context_set_enqueue_timeout_monotonic_timespec
Thread Queue Handler, 1084
_Thread_queue_Context_set_enqueue_timeout_realtime_timespec
Thread Queue Handler, 1085
_Thread_queue_Context_set_enqueue_timeout_ticks
Thread Queue Handler, 1085
_Thread_queue_Context_set_thread_state
Thread Queue Handler, 1086
_Thread_queue_Context_set_timeout_argument
Thread Queue Handler, 1086
_Thread_queue_Context_set_timeout_ticks
Thread Queue Handler, 1088
_Thread_queue_Deadlock_fatal
Thread Queue Handler, 1088
_Thread_queue_Deadlock_status
Thread Queue Handler, 1088
_Thread_queue_Dequeue
Thread Queue Handler, 1074
_Thread_queue_Destroy
Thread Queue Handler, 1089
_Thread_queue_Dispatch_disable
Thread Queue Handler, 1089
_Thread_queue_Do_acquire_critical
Thread Queue Handler, 1089
_Thread_queue_Do_dequeue
Thread Queue Handler, 1090
_Thread_queue_Do_release_critical
Thread Queue Handler, 1090
_Thread_queue_Enqueue
Thread Queue Handler, 1091
_Thread_queue_Enqueue_do_nothing_extra
Thread Queue Handler, 1092
_Thread_queue_Enqueue_sticky
Thread Queue Handler, 1092
_Thread_queue_Extract
Thread Queue Handler, 1093
_Thread_queue_Extract_critical
Thread Queue Handler, 1093
_Thread_queue_Extract_locked
Thread Queue Handler, 1094
_Thread_queue_Extract_with_proxy
Thread Queue Handler, 1095
_Thread_queue_First
Thread Queue Handler, 1095
_Thread_queue_First_locked
Thread Queue Handler, 1096
_Thread_queue_Flush_critical
Thread Queue Handler, 1096
_Thread_queue_Flush_default_filter
Thread Queue Handler, 1097
_Thread_queue_Flush_status_object_was_deleted
Thread Queue Handler, 1097
_Thread_queue_Flush_status_unavailable
Thread Queue Handler, 1098
_Thread_queue_Gate_add
Thread Queue Handler, 1098
_Thread_queue_Gate_close
Thread Queue Handler, 1099
_Thread_queue_Gate_open
Thread Queue Handler, 1099
_Thread_queue_Gate_wait
Thread Queue Handler, 1099
_Thread_queue_Heads, 1305
Fifo, 1305
Heads, 1306
_Thread_queue_Heads_initialize
Thread Queue Handler, 1100
_Thread_queue_Heads_size
Thread Handler, 1068
_Thread_queue_Initialize
Thread Queue Handler, 1100
_Thread_queue_Is_empty
Thread Queue Handler, 1100
_Thread_queue_Links
threadqenqueue.c, 1921
_Thread_queue_Object_initialize
Thread Queue Handler, 1101
_Thread_queue_Object_name
Thread Queue Handler, 1106
_Thread_queue_Path_acquire_critical
Thread Queue Handler, 1101
_Thread_queue_Path_release_critical
Thread Queue Handler, 1102
_Thread_queue_Queue_do_acquire_critical
Thread Queue Handler, 1102
_Thread_queue_Queue_get_name_and_id
Thread Queue Handler, 1102
_Thread_queue_Queue_initialize
Thread Queue Handler, 1103
_Thread_queue_Queue_release
Thread Queue Handler, 1103
_Thread_queue_Queue_release_critical
Thread Queue Handler, 1104
_Thread_queue_Release
Thread Queue Handler, 1104
_Thread_queue_Release_critical
Thread Queue Handler, 1104
_Thread_queue_Surrender
Thread Queue Handler, 1105
_Thread_queue_Surrender_sticky
Thread Queue Handler, 1105
_Thread_queue_Unblock_critical
Thread Queue Handler, 1106
_Timecounter_Acquire
Timecounter Handler, 1141

_Timecounter_Bintime
Timecounter Handler, 1142
_Timecounter_Binuptime
Timecounter Handler, 1142
_Timecounter_Getbintime
Timecounter Handler, 1142
_Timecounter_Getbinuptime
Timecounter Handler, 1143
_Timecounter_Getboottime
Timecounter Handler, 1143
_Timecounter_Getboottimebin
Timecounter Handler, 1143
_Timecounter_Getmicrotime
Timecounter Handler, 1145
_Timecounter_Getmicrouptime
Timecounter Handler, 1145
_Timecounter_Getnanotime
Timecounter Handler, 1145
_Timecounter_Getnanouptime
Timecounter Handler, 1147
_Timecounter_Install
Timecounter Handler, 1147
_Timecounter_Microtime
Timecounter Handler, 1147
_Timecounter_Microuptime
Timecounter Handler, 1148
_Timecounter_Nanotime
Timecounter Handler, 1148
_Timecounter_Nanouptime
Timecounter Handler, 1148
_Timecounter_Release
Timecounter Handler, 1142
_Timecounter_Sbinuptime
Timecounter Handler, 1148
_Timecounter_Set_clock
Timecounter Handler, 1149
_Timecounter_Tick_simple
Timecounter Handler, 1149
_Timecounter_Time_uptime
Timecounter Handler, 1150
_Timer_Allocate
Classic Timer Implementation, 242
_Timer_Free
Classic Timer Implementation, 242
_Timer_server
Classic Timer Implementation, 243
_Timespec_Add_to
Helpers, 455
_Timespec_Divide
Helpers, 455
_Timespec_Divide_by_integer
Helpers, 455
_Timespec_Equal_to
Helpers, 452
_Timespec_From_ticks
Helpers, 456
_Timespec_Get_as_nanoseconds
Helpers, 456
_Timespec_Get_nanoseconds
Helpers, 452
_Timespec_Get_seconds
Helpers, 453
_Timespec_Greater_than
Helpers, 453
_Timespec_Is_valid
Helpers, 457
_Timespec_Less_than
Helpers, 457
_Timespec_Set
Helpers, 454
_Timespec_Set_to_zero
Helpers, 454
_Timespec_Subtract
Helpers, 457
_Timespec_To_ticks
Helpers, 458
_Timestamp_Add_to
Score Timestamp, 886
_Timestamp_Divide
Score Timestamp, 886
_Timestamp_Equal_to
Score Timestamp, 887
_Timestamp_Get_as_nanoseconds
Score Timestamp, 887
_Timestamp_Get_nanoseconds
Score Timestamp, 888
_Timestamp_Get_seconds
Score Timestamp, 888
_Timestamp_Greater_than
Score Timestamp, 889
_Timestamp_Less_than
Score Timestamp, 889
_Timestamp_Set
Score Timestamp, 890
_Timestamp_Set_to_zero
Score Timestamp, 890
_Timestamp_Subtract
Score Timestamp, 890
_Timestamp_To_timespec
Score Timestamp, 891
_Timestamp_To_timeval
Score Timestamp, 891
_User_extensions_Acquire
User Extension Handler, 1183
_User_extensions_Add_API_set
User Extension Handler, 1184
_User_extensions_Add_set
User Extension Handler, 1184
_User_extensions_Add_set_with_table
User Extension Handler, 1184
_User_extensions_Destroy_iterators
User Extension Handler, 1184
_User_extensions_Fatal
User Extension Handler, 1185
_User_extensions_Fatal_visitor
User Extension Handler, 1185

_User_extensions_Initial_count
User Extension Handler, 1192
_User_extensions_Initial_extensions
User Extension Handler, 1192
_User_extensions_Initial_switch_controls
User Extension Handler, 1192
_User_extensions_Iterate
User Extension Handler, 1185
_User_extensions_Release
User Extension Handler, 1186
_User_extensions_Remove_set
User Extension Handler, 1186
_User_extensions_Thread_begin
User Extension Handler, 1186
_User_extensions_Thread_begin_visitor
User Extension Handler, 1187
_User_extensions_Thread_create
User Extension Handler, 1187
_User_extensions_Thread_create_visitor
User Extension Handler, 1188
_User_extensions_Thread_delete
User Extension Handler, 1188
_User_extensions_Thread_delete_visitor
User Extension Handler, 1188
_User_extensions_Thread_exitted
User Extension Handler, 1189
_User_extensions_Thread_exitted_visitor
User Extension Handler, 1189
_User_extensions_Thread_restart
User Extension Handler, 1189
_User_extensions_Thread_restart_visitor
User Extension Handler, 1190
_User_extensions_Thread_start
User Extension Handler, 1190
_User_extensions_Thread_start_visitor
User Extension Handler, 1190
_User_extensions_Thread_switch
User Extension Handler, 1191
_User_extensions_Thread_terminate
User Extension Handler, 1191
_User_extensions_Thread_terminate_visitor
User Extension Handler, 1191
_Watchdog_Cancel
Watchdog Handler, 1206
_Watchdog_Do_tickle
Watchdog Handler, 1206
_Watchdog_Future_timespec
Watchdog Handler, 1207
_Watchdog_Get_CPU
Watchdog Handler, 1207
_Watchdog_Get_state
Watchdog Handler, 1208
_Watchdog_Header_destroy
Watchdog Handler, 1208
_Watchdog_Header_first
Watchdog Handler, 1208
_Watchdog_Header_initialize
Watchdog Handler, 1209
_Watchdog_Initialize
Watchdog Handler, 1209
_Watchdog_Insert
Watchdog Handler, 1209
_Watchdog_Is_far_future_timespec
Watchdog Handler, 1210
_Watchdog_Is_scheduled
Watchdog Handler, 1210
_Watchdog_Is_valid_interval_timespec
Watchdog Handler, 1211
_Watchdog_Is_valid_timespec
Watchdog Handler, 1211
_Watchdog_Microseconds_per_tick
Watchdog Handler, 1217
_Watchdog_Nanoseconds_per_tick
Watchdog Handler, 1217
_Watchdog_Next_first
Watchdog Handler, 1211
_Watchdog_Per_CPU_acquire_critical
Watchdog Handler, 1212
_Watchdog_Per_CPU_insert
Watchdog Handler, 1212
_Watchdog_Per_CPU_insert_ticks
Watchdog Handler, 1213
_Watchdog_Per_CPU_release_critical
Watchdog Handler, 1213
_Watchdog_Per_CPU_remove
Watchdog Handler, 1213
_Watchdog_Per_CPU_remove_ticks
Watchdog Handler, 1214
_Watchdog_Preinitialize
Watchdog Handler, 1214
_Watchdog_Remove
Watchdog Handler, 1215
_Watchdog_Set_CPU
Watchdog Handler, 1215
_Watchdog_Set_state
Watchdog Handler, 1215
_Watchdog_Tick
Watchdog Handler, 1216
_Watchdog_Ticks_from_sbintime
Watchdog Handler, 1216
_Watchdog_Ticks_from_seconds
Watchdog Handler, 1216
_Watchdog_Ticks_from_timespec
Watchdog Handler, 1217
_Watchdog_Ticks_per_second
Watchdog Handler, 1217
_Watchdog_Ticks_per_timeslice
Watchdog Handler, 1218
_Watchdog_Ticks_since_boot
Watchdog Handler, 1218
_Workspace_Allocate
Workspace Handler, 1220
_Workspace_Area
Workspace Handler, 1222
_Workspace_Free
Workspace Handler, 1220

_Workspace_Handler_initialization
Workspace Handler, 1220
_Workspace_Is_unified
Workspace Handler, 1222
_Workspace_Malloc_initialize_separate
Workspace Handler, 1221
_Workspace_Malloc_initialize_unified
Workspace Handler, 1221
_Workspace_Malloc_initializer
Workspace Handler, 1222
_Workspace_Size
Workspace Handler, 1222
_Workspace_String_duplicate
Workspace Handler, 1221
__rtems_dev_t, 1297
_rtems_filesystem_file_handlers_r, 1297
_rtems_filesystem_operations_table, 1298

actions
spec:/testsuites/validation-0, 1290
spec:/testsuites/validation/profile, 1295
testrun.c, 1801
adaptor
Thread_Entry_information, 1504
add_processor
Scheduler_Operations, 1462
Address Handler, 51
_Addresses_Add_offset, 51
_Addresses_Align_down, 52
_Addresses_Align_up, 52
_Addresses_Is_aligned, 53
_Addresses_Is_in_range, 53
_Addresses_Subtract, 54
_Addresses_Subtract_offset, 54
allocate
Objects_Information, 1367
AMBA, 45
ambapp_ahb_bus, 1306
ambapp_ahb_info, 1306
ambapp_apb_info, 1307
ambapp_bus, 1307
ambapp_common_info, 1307
ambapp_core, 1308
ambapp_dev, 1308
ambapp_mmap, 1309
ambapp_pnp_ahb, 1309
ambapp_pnp_apb, 1309
ancestor
Per_CPU_Control, 1373
apbuart_regs, 1310
API, 48
API Mutex Handler, 49
_API_Mutex_Is_owner, 50
_API_Mutex_Lock, 50
_API_Mutex_Unlock, 50
API_Extensions
_Thread_Control, 1301
API_Mutex_Control, 1310
Mutex, 1310

Application Configuration, 56
Application Configuration Information, 58
rtems_configuration_get_do_zero_of_workspace, 60
rtems_configuration_get_idle_task, 60
rtems_configuration_get_idle_task_stack_size, 60
rtems_configuration_get_interrupt_stack_size, 61
rtems_configuration_get_maximum_barriers, 67
rtems_configuration_get_maximum_extensions, 67
rtems_configuration_get_maximum_message_queues, 68
rtems_configuration_get_maximum_partitions, 68
rtems_configuration_get_maximum_periods, 68
rtems_configuration_get_maximum_ports, 69
rtems_configuration_get_maximum_processors, 61
rtems_configuration_get_maximum_regions, 69
rtems_configuration_get_maximum_semaphores, 69
rtems_configuration_get_maximum_tasks, 70
rtems_configuration_get_maximum_timers, 70
rtems_configuration_get_microseconds_per_tick, 61
rtems_configuration_get_milliseconds_per_tick, 62
rtems_configuration_get_nanoseconds_per_tick, 62
rtems_configuration_get_number_of_initial_extensions, 62
rtems_configuration_get_rtems_api_configuration, 70
rtems_configuration_get_stack_allocate_hook, 63
rtems_configuration_get_stack_allocate_init_hook, 63
rtems_configuration_get_stack_allocator_avoids_work_space, 63
rtems_configuration_get_stack_free_hook, 64
rtems_configuration_get_stack_space_size, 71
rtems_configuration_get_ticks_per_timeslice, 64
rtems_configuration_get_unified_work_area, 64
rtems_configuration_get_user_extension_table, 65
rtems_configuration_get_user_multiprocessing_table, 65
rtems_configuration_get_work_space_size, 65
rtems_get_copyright_notice, 71
rtems_get_version_string, 71
rtems_resource_is_unlimited, 66
rtems_resource_maximum_per_allocation, 66
rtems_resource_unlimited, 67
Application Configuration Options, 72
area
Stack_Control, 1488
argument
rtems_initialization_tasks_table, 1412
ask_for_help
Scheduler_Operations, 1463
ASR_Information, 1311
handler, 1311

is_enabled, 1311
mode_set, 1311
nest_level, 1312
signals_pending, 1312
signals_posted, 1312
Assert Handler, 73
Associativity Routines, 74
rtems_assoc_32_to_string, 75
rtems_assoc_thread_states_to_string, 75
Atomic Operations, 77
Atomic Operations CPU, 79
_CPU_atomic_Compare_exchange_uint, 81
_CPU_atomic_Compare_exchange_uintptr, 81
_CPU_atomic_Compare_exchange_ulong, 82
_CPU_atomic_Exchange_uint, 83
_CPU_atomic_Exchange_uintptr, 83
_CPU_atomic_Exchange_ulong, 84
_CPU_atomic_Fence, 84
_CPU_atomic_Fetch_add_uint, 84
_CPU_atomic_Fetch_add_uintptr, 85
_CPU_atomic_Fetch_add_ulong, 85
_CPU_atomic_Fetch_and_uint, 86
_CPU_atomic_Fetch_and_uintptr, 86
_CPU_atomic_Fetch_and_ulong, 87
_CPU_atomic_Fetch_or_uint, 87
_CPU_atomic_Fetch_or_uintptr, 88
_CPU_atomic_Fetch_or_ulong, 88
_CPU_atomic_Fetch_sub_uint, 88
_CPU_atomic_Fetch_sub_uintptr, 89
_CPU_atomic_Fetch_sub_ulong, 89
_CPU_atomic_Flag_clear, 90
_CPU_atomic_Flag_test_and_set, 90
_CPU_atomic_Init_uint, 91
_CPU_atomic_Init_uintptr, 91
_CPU_atomic_Init_ulong, 91
_CPU_atomic_Load_uint, 92
_CPU_atomic_Load_uintptr, 92
_CPU_atomic_Load_ulong, 92
_CPU_atomic_Store_uint, 93
_CPU_atomic_Store_uintptr, 93
_CPU_atomic_Store_ulong, 94
attribute_set
Barrier_Control, 1313
rtems_initialization_tasks_table, 1412
Attributes
CORE_barrier_Control, 1328
attributes
Scheduler_Assignment, 1453
ATTRIBUTES_NOT_SUPPORTED
Classic Attributes Implementation, 216
ATTRIBUTES_REQUIRED
Classic Attributes Implementation, 217
auto_extend
rtems_object_api_class_information, 1422

Barrier
Barrier_Control, 1313
Barrier Handler, 105
_CORE_barrier_Acquire_critical, 106
_CORE_barrier_Destroy, 107
_CORE_barrier_Do_flush, 107
_CORE_barrier_Flush, 108
_CORE_barrier_Get_number_of_waiting_threads, 108
_CORE_barrier_Initialize, 108
_CORE_barrier_Is_automatic, 109
_CORE_barrier_Release, 109
_CORE_barrier_Seize, 109
_CORE_barrier_Surrender, 110
CORE_BARRIER_AUTOMATIC_RELEASE, 106
CORE_barrier_Disciplines, 106
CORE_BARRIER_MANUAL_RELEASE, 106
Barrier Manager, 111
rtems_barrier_create, 111
rtems_barrier_delete, 111
rtems_barrier_ident, 112
rtems_barrier_release, 112
rtems_barrier_wait, 113
Barrier_Control, 1312
attribute_set, 1313
Barrier, 1313
Object, 1313
BARRIER_INFORMATION_DEFINE
Classic Barrier Implementation, 221
Base Definitions, 114
RTEMS_ALIAS, 116
RTEMS_ALIGN_DOWN, 116
RTEMS_ALIGN_UP, 118
RTEMS_ALIGNED, 118
RTEMS_ALLOC_ALIGN, 119
RTEMS_ALLOC_SIZE, 119
RTEMS_ALLOC_SIZE_2, 119
RTEMS_ARRAY_SIZE, 120
RTEMS_CONCAT, 120
RTEMS_CONTAINER_OF, 120
RTEMS_DECLARE_GLOBAL_SYMBOL, 121
RTEMS_DECONST, 121
RTEMS_DEFINE_GLOBAL_SYMBOL, 122
RTEMS_DEQUALIFY, 122
RTEMS_DEQUALIFY_DEPTHX, 122
RTEMS_DEVOLATILE, 123
RTEMS_EXPAND, 123
RTEMS_HAVE_MEMBER_SAME_TYPE, 123
RTEMS_OBFUSCATE_VARIABLE, 124
RTEMS_PREDICT_FALSE, 124
RTEMS_PREDICT_TRUE, 126
RTEMS_PRINTFLIKE, 126
RTEMS_RETURN_ADDRESS, 126
RTEMS_SECTION, 127
RTEMS_STATIC_ASSERT, 127
RTEMS_STRING, 127
RTEMS_SYMBOL_NAME, 128
RTEMS_TYPEOF_REFX, 128
RTEMS_WEAK, 129
RTEMS_WEAK_ALIAS, 129
RTEMS_XCONCAT, 129
RTEMS_XSTRING, 130

RTEMS_ZERO_LENGTH_ARRAY, 130
basedefs.h
RTEMS_UNREACHABLE, 1676
Basic Types, 131
RTEMS_ID_NONE, 131
rtems_name, 131
bit_map
Priority_bit_map_Control, 1386
Bitmap Priority Thread Routines, 132
block
Scheduler_Operations, 1463
Block Device Cache Configuration, 133
CONFIGURE_APPLICATION_NEEDS_LIBBLOCK, 133
CONFIGURE_BDBUF_BUFFER_MAX_SIZE, 134
CONFIGURE_BDBUF_BUFFER_MIN_SIZE, 134
CONFIGURE_BDBUF_CACHE_MEMORY_SIZE, 135
CONFIGURE_BDBUF_MAX_READ_AHEAD_BLOCKS, 135
CONFIGURE_BDBUF_MAX_WRITE_BLOCKS, 136
CONFIGURE_BDBUF_READ_AHEAD_TASK_PRIORITY, 136
CONFIGURE_BDBUF_TASK_STACK_SIZE, 137
CONFIGURE_SWAPOUT_BLOCK_HOLD, 137
CONFIGURE_SWAPOUT_SWAP_PERIOD, 138
CONFIGURE_SWAPOUT_TASK_PRIORITY, 138
CONFIGURE_SWAPOUT_WORKER_TASK_PRIORITY, 138
CONFIGURE_SWAPOUT_WORKER_TASKS, 139
block_major
Priority_bit_map_Information, 1387
block_minor
Priority_bit_map_Information, 1387
Board Support Packages, 140
body
rtems_test_parallel_job, 1442
Boolean Checks, 141
T_assert_false, 141
T_false, 141
T_quiet_false, 141
T_step_assert_false, 142
T_step_false, 142
boot_card
Bootcard, 143
Bootcard, 143
boot_card, 143
bsp_start_on_secondary_processor, 143
BSP Interrupt Support, 95
bsp_interrupt_facility_initialize, 96
bsp_interrupt_handler_default, 97
bsp_interrupt_handler_dispatch, 97
bsp_interrupt_handler_install, 97
bsp_interrupt_handler_is_empty, 98
bsp_interrupt_handler_iterate, 98
bsp_interrupt_handler_remove, 99
bsp_interrupt_initialize, 99
bsp_interrupt_vector_disable, 99
bsp_interrupt_vector_enable, 100
BSP Related Configuration Options, 101
BSP_IDLE_TASK_BODY, 101
BSP_IDLE_TASK_STACK_SIZE, 101
BSP_INITIAL_EXTENSION, 102
BSP_INTERRUPT_STACK_SIZE, 102
CONFIGURE_BSP_PREREQUISITE_DRIVERS, 103
CONFIGURE_DISABLE_BSP_SETTINGS, 103
CONFIGURE_MALLOC_BSP_SUPPORTS_SBRK, 104
bsp_fatal_halt.c
_CPU_Fatal_halt, 1568
BSP_IDLE_TASK_BODY
BSP Related Configuration Options, 101
BSP_IDLE_TASK_STACK_SIZE
BSP Related Configuration Options, 101
BSP_INITIAL_EXTENSION
BSP Related Configuration Options, 102
bsp_interrupt_facility_initialize
BSP Interrupt Support, 96
bsp_interrupt_handler_default
BSP Interrupt Support, 97
bsp_interrupt_handler_dispatch
BSP Interrupt Support, 97
bsp_interrupt_handler_entry, 1314
bsp_interrupt_handler_install
BSP Interrupt Support, 97
bsp_interrupt_handler_is_empty
BSP Interrupt Support, 98
bsp_interrupt_handler_iterate
BSP Interrupt Support, 98
bsp_interrupt_handler_remove
BSP Interrupt Support, 99
bsp_interrupt_initialize
BSP Interrupt Support, 99
BSP_INTERRUPT_STACK_SIZE
BSP Related Configuration Options, 102
bsp_interrupt_vector_disable
BSP Interrupt Support, 99
bsp_interrupt_vector_enable
BSP Interrupt Support, 100
BSP_output_char
bsplo.h, 1576
BSP_output_char_function_type
bsplo.h, 1573
BSP_poll_char
bsplo.h, 1576
BSP_polling_getchar_function_type
bsplo.h, 1573
bsp_start_on_secondary_processor
Bootcard, 143
bsplo.h
BSP_output_char, 1576
BSP_output_char_function_type, 1573
BSP_poll_char, 1576
BSP_polling_getchar_function_type, 1573

getchark, 1574
printk, 1574
putk, 1575
rtems_put_char, 1575
rtems_putc, 1575
vprintk, 1576
bsps/include/bsp/bootcard.h, 1543
bsps/include/bsp/default-initial-extension.h, 1543
bsps/include/bsp/irq-generic.h, 1545
bsps/include/grlib/ambapp.h, 1546
bsps/include/grlib/ambapp_ids.h, 1548
bsps/include/grlib/apbuart.h, 1553
bsps/include/grlib/grlib.h, 1554
bsps/shared/dev/clock/clockimpl.h, 1555
bsps/shared/irq/irq-generic.c, 1556
bsps/shared/irq/irq-lock.c, 1557
bsps/shared/start/bootcard.c, 1557
bsps/shared/start/mallocinitone.c, 1558
bsps/shared/start/wkspaceinitone.c, 1559
bsps/sparc/leon3/include/amba.h, 1559
bsps/sparc/leon3/include/bsp.h, 1559
bsps/sparc/leon3/include/bsp/irq.h, 1561
bsps/sparc/leon3/include/leon.h, 1562
bsps/sparc/leon3/include/tm27.h, 1567
bsps/sparc/leon3/start/bsp_fatal_halt.c, 1568
bsps/sparc/leon3/start/bspclean.c, 1568
bsps/sparc/leon3/start/bspdelay.c, 1569
bsps/sparc/leon3/start/bspsmp.c, 1569
bsps/sparc/leon3/start/setvec.c, 1570
bsps/sparc/shared/start/bsp_fatal_exit.c, 1570
budget_algorithm
_Thread_Control, 1301
Thread_Start_information, 1523
budget_callout
_Thread_Control, 1301
Thread_Start_information, 1524
buffer
CORE_message_queue_Buffer, 1330

Cache Manager, 146
rtems_cache_aligned_malloc, 147
rtems_cache_coherent_add_area, 147
rtems_cache_coherent_allocate, 147
rtems_cache_coherent_free, 148
rtems_cache_flush_multiple_data_lines, 148
rtems_cache_get_data_cache_size, 148
rtems_cache_get_instruction_cache_size, 148
rtems_cache_instruction_sync_after_code_change, 149
rtems_cache_invalidate_multiple_data_lines, 149
rtems_cache_invalidate_multiple_instruction_lines, 149
cancel_job
Scheduler_Operations, 1463
cascade
rtems_test_parallel_job, 1442
Cause_tm27_intr
tm27.h, 1567
Chain Handler, 151
_Chain_Append_if_is_off_chain_unprotected, 156
_Chain_Append_unprotected, 156
_Chain_Append_with_empty_check_unprotected, 157
_Chain_Are_nodes_equal, 157
_Chain_Extract_unprotected, 158
_Chain_First, 158
_Chain_Get_first_unprotected, 158
_Chain_Get_unprotected, 159
_Chain_Get_with_empty_check_unprotected, 160
_Chain_Has_only_one_node, 160
_Chain_Head, 161
_Chain_Immutable_first, 161
_Chain_Immutable_head, 161
_Chain_Immutable_last, 162
_Chain_Immutable_next, 162
_Chain_Immutable_previous, 163
_Chain_Immutable_tail, 163
_Chain_Initialize, 164
_Chain_Initialize_empty, 164
_Chain_Initialize_node, 164
_Chain_Initialize_one, 165
_Chain_Insert_ordered_unprotected, 165
_Chain_Insert_unprotected, 166
_Chain_Is_empty, 166
_Chain_Is_first, 166
_Chain_Is_head, 167
_Chain_Is_last, 167
_Chain_Is_node_off_chain, 168
_Chain_Is_tail, 168
_Chain_Iterator_destroy, 169
_Chain_Iterator_initialize, 169
_Chain_Iterator_next, 170
_Chain_Iterator_registry_initialize, 171
_Chain_Iterator_registry_update, 171
_Chain_Iterator_set_position, 171
_Chain_Last, 172
_Chain_Next, 172
_Chain_Node_count_unprotected, 173
_Chain_Prepend_unprotected, 173
_Chain_Prepend_with_empty_check_unprotected, 173
_Chain_Previous, 174
_Chain_Set_off_chain, 174
_Chain_Tail, 175
CHAIN_INITIALIZER_ONE_NODE, 154
CHAIN_ITERATOR_BACKWARD, 155
Chain_Iterator_direction, 155
CHAIN_ITERATOR_FORWARD, 155
CHAIN_ITERATOR_REGISTRY_INITIALIZER, 154
Chain_Node, 155
CHAIN_NODE_INITIALIZER_ONE_NODE_CHAIN, 154
Chain_Node_order, 155
Chain_Control, 1314
CHAIN_INITIALIZER_ONE_NODE
Chain Handler, 154

Chain_Iterator, 1315
direction, 1315
position, 1315
Registry_node, 1316
CHAIN_ITERATOR_BACKWARD
Chain Handler, 155
Chain_Iterator_direction
Chain Handler, 155
CHAIN_ITERATOR_FORWARD
Chain Handler, 155
Chain_Iterator_registry, 1316
CHAIN_ITERATOR_REGISTRY_INITIALIZER
Chain Handler, 154
Chain_Node
Chain Handler, 155
CHAIN_NODE_INITIALIZER_ONE_NODE_CHAIN
Chain Handler, 154
Chain_Node_order
Chain Handler, 155
Chain_Node_struct, 1317
next, 1317
previous, 1317
Chains, 176
rtems_chain_append, 179
rtems_chain_append_unprotected, 179
rtems_chain_append_with_empty_check, 179
rtems_chain_append_with_notification, 180
rtems_chain_are_nodes_equal, 180
rtems_chain_extract, 181
rtems_chain_extract_unprotected, 181
rtems_chain_first, 181
rtems_chain_get, 182
rtems_chain_get_with_empty_check, 182
rtems_chain_get_with_notification, 182
rtems_chain_get_with_wait, 183
rtems_chain_has_only_one_node, 183
rtems_chain_head, 184
rtems_chain_immutable_first, 184
rtems_chain_immutable_head, 184
rtems_chain_immutable_last, 185
rtems_chain_immutable_next, 185
rtems_chain_immutable_previous, 186
rtems_chain_immutable_tail, 186
rtems_chain_initialize, 186
rtems_chain_initialize_empty, 187
rtems_chain_initialize_node, 187
RTEMS_CHAIN_INITIALIZER_ONE_NODE, 178
rtems_chain_insert, 188
rtems_chain_is_empty, 188
rtems_chain_is_first, 188
rtems_chain_is_head, 189
rtems_chain_is_last, 189
rtems_chain_is_node_off_chain, 190
rtems_chain_is_null_node, 190
rtems_chain_is_tail, 191
rtems_chain_last, 191
rtems_chain_next, 191
rtems_chain_node_count_unprotected, 192
RTEMS_CHAIN_NODE_INITIALIZER_ONE_NODE_CHAIN, 178
rtems_chain_prepend, 192
rtems_chain_prepend_unprotected, 193
rtems_chain_prepend_with_empty_check, 193
rtems_chain_prepend_with_notification, 193
rtems_chain_previous, 194
rtems_chain_set_off_chain, 194
rtems_chain_tail, 195
Character Checks, 196
Classic, 197, 198
Classic API Configuration, 203
CONFIGURE_MAXIMUM_BARRIERS, 203
CONFIGURE_MAXIMUM_MESSAGE_QUEUES, 204
CONFIGURE_MAXIMUM_PARTITIONS, 204
CONFIGURE_MAXIMUM_PERIODS, 205
CONFIGURE_MAXIMUM_PORTS, 205
CONFIGURE_MAXIMUM_REGIONS, 206
CONFIGURE_MAXIMUM_SEMAPHORES, 206
CONFIGURE_MAXIMUM_TASKS, 207
CONFIGURE_MAXIMUM_THREAD_LOCAL_STORAGE_SIZE, 208
CONFIGURE_MAXIMUM_TIMERS, 209
CONFIGURE_MAXIMUM_USER_EXTENSIONS, 209
CONFIGURE_MINIMUM_TASKS_WITH_USER_PROVIDED_STORA 210
Classic API Initialization Task Configuration, 211
CONFIGURE_INIT_TASK_ARGUMENTS, 211
CONFIGURE_INIT_TASK_ATTRIBUTES, 211
CONFIGURE_INIT_TASK_ENTRY_POINT, 212
CONFIGURE_INIT_TASK_INITIAL_MODES, 212
CONFIGURE_INIT_TASK_NAME, 213
CONFIGURE_INIT_TASK_PRIORITY, 213
CONFIGURE_INIT_TASK_STACK_SIZE, 214
CONFIGURE_RTEMS_INIT_TASKS_TABLE, 214
Classic ASR Implementation, 215
_ASR_Initialize, 215
Classic Attributes Implementation, 216
_Attributes_Clear, 217
_Attributes_Has_at_most_one_protocol, 217
_Attributes_Is_barrier_automatic, 217
_Attributes_Is_binary_semaphore, 218
_Attributes_Is_counting_semaphore, 218
_Attributes_Is_floating_point, 218
_Attributes_Is_inherit_priority, 218
_Attributes_Is_multiprocessor_resource_sharing, 219
_Attributes_Is_priority, 219
_Attributes_Is_priority_ceiling, 219
_Attributes_Is_simple_binary_semaphore, 219
_Attributes_Is_system_task, 220
_Attributes_Set, 220
ATTRIBUTES_NOT_SUPPORTED, 216
ATTRIBUTES_REQUIRED, 217
Classic Barrier Implementation, 221
_Barrier_Allocate, 222

_Barrier_Free, 222
BARRIER_INFORMATION_DEFINE, 221
Classic Message Queue Implementation, 223
_Message_queue_Create, 225
_Message_queue_Free, 225
_Message_queue_Submit, 225
MESSAGE_QUEUE_INFORMATION_DEFINE, 224
MESSAGE_QUEUE_SEND_REQUEST, 224
Message_queue_Submit_types, 224
MESSAGE_QUEUE_URGENT_REQUEST, 224
Classic Modes Implementation, 226
_Modes_Change, 226
_Modes_Get_interrupt_level, 226
_Modes_Is_asr_disabled, 227
_Modes_Is_preempt, 227
_Modes_Is_timeslice, 227
_Modes_Mask_changed, 227
_Modes_Set_interrupt_level, 228
Classic Object Implementation, 229
_RTEMS_Name_to_id, 229
Classic Options Implementation, 230
_Options_Is_any, 230
_Options_Is_no_wait, 230
Classic Rate Monotonic Scheduler Implementation, 231
_Rate_monotonic_Allocate, 232
_Rate_monotonic_Get_status, 232
RATE_MONOTONIC_INFORMATION_DEFINE, 232
Classic Region Manager Implementation, 234
_Region_Allocate, 235
_Region_Allocate_segment, 235
_Region_Free, 235
_Region_Free_segment, 236
_Region_Process_queue, 236
REGION_INFORMATION_DEFINE, 234
Classic Status Implementation, 237
_Status_Object_name_errors_to_status, 237
Classic Tasks Manager Implementation, 238
_RTEMS_Priority_From_core, 238
_RTEMS_Priority_To_core, 239
_RTEMS_tasks_Free, 239
_RTEMS_tasks_Information, 240
_RTEMS_tasks_Initialize_user_tasks, 240
_RTEMS_tasks_User_task_table, 240
Classic Timer Implementation, 241
_Timer_Allocate, 242
_Timer_Free, 242
_Timer_server, 243
TIMER_INFORMATION_DEFINE, 242
ClassicTaskIdentConfig
spec:/rtems/task/req/ident, 1283
Clock Driver, 244
Clock_driver_ticks, 244
Clock Manager, 245
rtems_clock_get_seconds_since_epoch, 245
rtems_clock_get_tod, 246
rtems_clock_get_tod_timeval, 246
rtems_clock_get_uptime, 246
rtems_clock_get_uptime_timeval, 246
rtems_clock_set, 247
rtems_clock_tick_before, 247
rtems_clock_tick_later, 247
rtems_clock_tick_later_usec, 248
Clock Support, 249
clock.h
_TOD_To_seconds, 1583
_TOD_Validate, 1583
Clock_driver_ticks
Clock Driver, 244
Clock_isr
clockimpl.h, 1555
clockimpl.h
Clock_isr, 1555
clocktodtoseconds.c
_TOD_Days_to_date, 1807
_TOD_To_seconds, 1807
clocktodvalidate.c
_TOD_Days_per_month, 1809
_TOD_Validate, 1808
close_entry
rtems_driver_address_table, 1403
config
testrun.c, 1801
CONFIGURE_APPLICATION_DISABLE_FILESYSTEM
Filesystem Configuration, 376
CONFIGURE_APPLICATION_DOES_NOT_NEED_CLOCK_DRIVER
Device Driver Configuration, 268
CONFIGURE_APPLICATION_EXTRA_DRIVERS
Device Driver Configuration, 269
CONFIGURE_APPLICATION_NEEDS_ATA_DRIVER
Device Driver Configuration, 269
CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER
Device Driver Configuration, 270
CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER
Device Driver Configuration, 270
CONFIGURE_APPLICATION_NEEDS_FRAME_BUFFER_DRIVER
Device Driver Configuration, 271
CONFIGURE_APPLICATION_NEEDS_IDE_DRIVER
Device Driver Configuration, 271
CONFIGURE_APPLICATION_NEEDS_LIBBLOCK
Block Device Cache Configuration, 133
CONFIGURE_APPLICATION_NEEDS_NULL_DRIVER
Device Driver Configuration, 272
CONFIGURE_APPLICATION_NEEDS_RTC_DRIVER
Device Driver Configuration, 272
CONFIGURE_APPLICATION_NEEDS_SIMPLE_CONSOLE_DRIVER
Device Driver Configuration, 273
CONFIGURE_APPLICATION_NEEDS_SIMPLE_TASK_CONSOLE_DRIV
Device Driver Configuration, 273
CONFIGURE_APPLICATION_NEEDS_STUB_DRIVER
Device Driver Configuration, 274
CONFIGURE_APPLICATION_NEEDS_TIMER_DRIVER
Device Driver Configuration, 274
CONFIGURE_APPLICATION_NEEDS_WATCHDOG_DRIVER
Device Driver Configuration, 275

CONFIGURE_APPLICATION_NEEDS_ZERO_DRIVER
Device Driver Configuration, 275
CONFIGURE_APPLICATION_PREREQUISITE_DRIVERS
Device Driver Configuration, 276
CONFIGURE_ATA_DRIVER_TASK_PRIORITY
Device Driver Configuration, 276
CONFIGURE_BDBUF_BUFFER_MAX_SIZE
Block Device Cache Configuration, 134
CONFIGURE_BDBUF_BUFFER_MIN_SIZE
Block Device Cache Configuration, 134
CONFIGURE_BDBUF_CACHE_MEMORY_SIZE
Block Device Cache Configuration, 135
CONFIGURE_BDBUF_MAX_READ_AHEAD_BLOCKS
Block Device Cache Configuration, 135
CONFIGURE_BDBUF_MAX_WRITE_BLOCKS
Block Device Cache Configuration, 136
CONFIGURE_BDBUF_READ_AHEAD_TASK_PRIORITY
Block Device Cache Configuration, 136
CONFIGURE_BDBUF_TASK_STACK_SIZE
Block Device Cache Configuration, 137
CONFIGURE_BSP_PREREQUISITE_DRIVERS
BSP Related Configuration Options, 103
CONFIGURE_CBS_MAXIMUM_SERVERS
General Scheduler Configuration, 403
CONFIGURE_DIRTY_MEMORY
General System Configuration, 412
CONFIGURE_DISABLE_BSP_SETTINGS
BSP Related Configuration Options, 103
CONFIGURE_DISABLE_NEWLIB_REENTRANCY
General System Configuration, 412
CONFIGURE_EXECUTIVE_RAM_SIZE
General System Configuration, 412
CONFIGURE_EXTRA_MPCI_RECEIVE_SERVER_STACK
Multiprocessing Configuration, 568
CONFIGURE_EXTRA_TASK_STACKS
General System Configuration, 413
CONFIGURE_FILESYSTEM_ALL
Filesystem Configuration, 377
CONFIGURE_FILESYSTEM_DOSFS
Filesystem Configuration, 377
CONFIGURE_FILESYSTEM_FTPFS
Filesystem Configuration, 378
CONFIGURE_FILESYSTEM_IMFS
Filesystem Configuration, 378
CONFIGURE_FILESYSTEM_JFFS2
Filesystem Configuration, 378
CONFIGURE_FILESYSTEM_NFS
Filesystem Configuration, 379
CONFIGURE_FILESYSTEM_RFS
Filesystem Configuration, 379
CONFIGURE_FILESYSTEM_TFTPFS
Filesystem Configuration, 379
CONFIGURE_IDLE_TASK_BODY
Idle Task Configuration, 491
CONFIGURE_IDLE_TASK_INITIALIZES_APPLICATION
Idle Task Configuration, 491
CONFIGURE_IDLE_TASK_STACK_SIZE
Idle Task Configuration, 492
CONFIGURE_IMFS_DISABLE_CHMOD
Filesystem Configuration, 380
CONFIGURE_IMFS_DISABLE_CHOWN
Filesystem Configuration, 380
CONFIGURE_IMFS_DISABLE_LINK
Filesystem Configuration, 380
CONFIGURE_IMFS_DISABLE_MKNOD
Filesystem Configuration, 381
CONFIGURE_IMFS_DISABLE_MKNOD_DEVICE
Filesystem Configuration, 381
CONFIGURE_IMFS_DISABLE_MKNOD_FILE
Filesystem Configuration, 381
CONFIGURE_IMFS_DISABLE_MOUNT
Filesystem Configuration, 382
CONFIGURE_IMFS_DISABLE_READDIR
Filesystem Configuration, 382
CONFIGURE_IMFS_DISABLE_READLINK
Filesystem Configuration, 382
CONFIGURE_IMFS_DISABLE_RENAME
Filesystem Configuration, 383
CONFIGURE_IMFS_DISABLE_RMNOD
Filesystem Configuration, 383
CONFIGURE_IMFS_DISABLE_SYMLINK
Filesystem Configuration, 383
CONFIGURE_IMFS_DISABLE_UNMOUNT
Filesystem Configuration, 384
CONFIGURE_IMFS_DISABLE_UTIME
Filesystem Configuration, 384
CONFIGURE_IMFS_ENABLE_MKFIFO
Filesystem Configuration, 384
CONFIGURE_IMFS_MEMFILE_BYTES_PER_BLOCK
Filesystem Configuration, 385
CONFIGURE_INIT_TASK_ARGUMENTS
Classic API Initialization Task Configuration, 211
CONFIGURE_INIT_TASK_ATTRIBUTES
Classic API Initialization Task Configuration, 211
CONFIGURE_INIT_TASK_ENTRY_POINT
Classic API Initialization Task Configuration, 212
CONFIGURE_INIT_TASK_INITIAL_MODES
Classic API Initialization Task Configuration, 212
CONFIGURE_INIT_TASK_NAME
Classic API Initialization Task Configuration, 213
CONFIGURE_INIT_TASK_PRIORITY
Classic API Initialization Task Configuration, 213
CONFIGURE_INIT_TASK_STACK_SIZE
Classic API Initialization Task Configuration, 214
CONFIGURE_INITIAL_EXTENSIONS
General System Configuration, 413
CONFIGURE_INTERRUPT_STACK_SIZE
General System Configuration, 414
CONFIGURE_MALLOC_BSP_SUPPORTS_SBRK
BSP Related Configuration Options, 104
CONFIGURE_MALLOC_DIRTY
General System Configuration, 414
CONFIGURE_MAXIMUM_BARRIERS
Classic API Configuration, 203
CONFIGURE_MAXIMUM_DRIVERS
Device Driver Configuration, 277

CONFIGURE_MAXIMUM_FILE_DESCRIPTORS
General System Configuration, 415
CONFIGURE_MAXIMUM_MESSAGE_QUEUES
Classic API Configuration, 204
CONFIGURE_MAXIMUM_PARTITIONS
Classic API Configuration, 204
CONFIGURE_MAXIMUM_PERIODS
Classic API Configuration, 205
CONFIGURE_MAXIMUM_PORTS
Classic API Configuration, 205
CONFIGURE_MAXIMUM_POSIX_KEY_VALUE_PAIRS
POSIX API Configuration, 640
CONFIGURE_MAXIMUM_POSIX_KEYS
POSIX API Configuration, 641
CONFIGURE_MAXIMUM_POSIX_MESSAGE_QUEUES
POSIX API Configuration, 641
CONFIGURE_MAXIMUM_POSIX_QUEUED_SIGNALS
POSIX API Configuration, 642
CONFIGURE_MAXIMUM_POSIX_SEMAPHORES
POSIX API Configuration, 643
CONFIGURE_MAXIMUM_POSIX_SHMS
POSIX API Configuration, 643
CONFIGURE_MAXIMUM_POSIX_THREADS
POSIX API Configuration, 644
CONFIGURE_MAXIMUM_POSIX_TIMERS
POSIX API Configuration, 645
CONFIGURE_MAXIMUM_PRIORITY
General Scheduler Configuration, 403
CONFIGURE_MAXIMUM_PROCESSORS
General System Configuration, 415
CONFIGURE_MAXIMUM_REGIONS
Classic API Configuration, 206
CONFIGURE_MAXIMUM_SEMAPHORES
Classic API Configuration, 206
CONFIGURE_MAXIMUM_TASKS
Classic API Configuration, 207
CONFIGURE_MAXIMUM_THREAD_LOCAL_STORAGE_SIZE
Classic API Configuration, 208
CONFIGURE_MAXIMUM_THREAD_NAME_SIZE
General System Configuration, 416
CONFIGURE_MAXIMUM_TIMERS
Classic API Configuration, 209
CONFIGURE_MAXIMUM_USER_EXTENSIONS
Classic API Configuration, 209
CONFIGURE_MEMORY_OVERHEAD
General System Configuration, 416
CONFIGURE_MESSAGE_BUFFER_MEMORY
General System Configuration, 417
CONFIGURE_MICROSECONDS_PER_TICK
General System Configuration, 418
CONFIGURE_MINIMUM_POSIX_THREAD_STACK_SIZE
POSIX API Configuration, 645
CONFIGURE_MINIMUM_TASK_STACK_SIZE
General System Configuration, 419
CONFIGURE_MINIMUM_TASKS_WITH_USER_PROVIDED_STORAGE
Classic API Configuration, 210
CONFIGURE_MP_APPLICATION
Multiprocessing Configuration, 568
CONFIGURE_MP_MAXIMUM_GLOBAL_OBJECTS
Multiprocessing Configuration, 569
CONFIGURE_MP_MAXIMUM_NODES
Multiprocessing Configuration, 569
CONFIGURE_MP_MAXIMUM_PROXIES
Multiprocessing Configuration, 570
CONFIGURE_MP_MPCI_TABLE_POINTER
Multiprocessing Configuration, 570
CONFIGURE_MP_NODE_NUMBER
Multiprocessing Configuration, 571
CONFIGURE_POSIX_INIT_THREAD_ENTRY_POINT
POSIX Initialization Thread Configuration, 647
CONFIGURE_POSIX_INIT_THREAD_STACK_SIZE
POSIX Initialization Thread Configuration, 647
CONFIGURE_POSIX_INIT_THREAD_TABLE
POSIX Initialization Thread Configuration, 648
CONFIGURE_RECORD_EXTENSIONS_ENABLED
Event Recording Configuration, 325
CONFIGURE_RECORD_FATAL_DUMP_BASE64
Event Recording Configuration, 325
CONFIGURE_RECORD_FATAL_DUMP_BASE64_ZLIB
Event Recording Configuration, 326
CONFIGURE_RECORD_PER_PROCESSOR_ITEMS
Event Recording Configuration, 326
CONFIGURE_RTEMS_INIT_TASKS_TABLE
Classic API Initialization Task Configuration, 214
CONFIGURE_SCHEDULER_ASSIGNMENTS
General Scheduler Configuration, 404
spec:/testsuites/validation-0, 1290
CONFIGURE_SCHEDULER_CBS
General Scheduler Configuration, 404
CONFIGURE_SCHEDULER_EDF
General Scheduler Configuration, 405
CONFIGURE_SCHEDULER_EDF_SMP
General Scheduler Configuration, 405
CONFIGURE_SCHEDULER_NAME
General Scheduler Configuration, 406
CONFIGURE_SCHEDULER_PRIORITY
General Scheduler Configuration, 406
CONFIGURE_SCHEDULER_PRIORITY_AFFINITY_SMP
General Scheduler Configuration, 407
CONFIGURE_SCHEDULER_PRIORITY_SMP
General Scheduler Configuration, 407
CONFIGURE_SCHEDULER_SIMPLE
General Scheduler Configuration, 408
CONFIGURE_SCHEDULER_SIMPLE_SMP
General Scheduler Configuration, 408
CONFIGURE_SCHEDULER_STRONG_APA
General Scheduler Configuration, 409
CONFIGURE_SCHEDULER_TABLE_ENTRIES
spec:/testsuites/validation-0, 1290
CONFIGURE_SCHEDULER_USER
General Scheduler Configuration, 409
CONFIGURE_STACK_CHECKER_ENABLED
General System Configuration, 419
CONFIGURE_SWAPOUT_BLOCK_HOLD
Block Device Cache Configuration, 137
CONFIGURE_SWAPOUT_SWAP_PERIOD

Block Device Cache Configuration, 138
CONFIGURE_SWAPOUT_TASK_PRIORITY
Block Device Cache Configuration, 138
CONFIGURE_SWAPOUT_WORKER_TASK_PRIORITY
Block Device Cache Configuration, 138
CONFIGURE_SWAPOUT_WORKER_TASKS
Block Device Cache Configuration, 139
CONFIGURE_TASK_STACK_ALLOCATOR
Task Stack Allocator Configuration, 983
CONFIGURE_TASK_STACK_ALLOCATOR_AVOIDS_WORK_SPACE
Task Stack Allocator Configuration, 983
CONFIGURE_TASK_STACK_ALLOCATOR_INIT
Task Stack Allocator Configuration, 984
CONFIGURE_TASK_STACK_DEALLOCATOR
Task Stack Allocator Configuration, 984
CONFIGURE_TASK_STACK_FROM_ALLOCATOR
Task Stack Allocator Configuration, 985
CONFIGURE_TICKS_PER_TIMESLICE
General System Configuration, 420
CONFIGURE_UNIFIED_WORK_AREAS
General System Configuration, 420
CONFIGURE_UNLIMITED_ALLOCATION_SIZE
General System Configuration, 421
CONFIGURE_UNLIMITED_OBJECTS
General System Configuration, 421
CONFIGURE_USE_DEVFS_AS_BASE_FILESYSTEM
Filesystem Configuration, 385
CONFIGURE_USE_MINIIMFS_AS_BASE_FILESYSTEM
Filesystem Configuration, 386
CONFIGURE_VERBOSE_SYSTEM_INITIALIZATION
General System Configuration, 422
CONFIGURE_ZERO_WORKSPACE_AUTOMATICALLY
General System Configuration, 422
Console Driver Support, 250
context
Per_CPU_Control, 1374
Context Handler, 251
_Context_Initialization_at_thread_begin, 251
_Context_Initialize, 251
_Context_Initialize_fp, 252
_Context_Restart_self, 252
_Context_Restore_fp, 253
_Context_Save_fp, 253
_Context_Switch, 254
CONTEXT_FP_SIZE, 254
Context_Control, 1318
g5, 1318
g7, 1319
i0, 1319
i1, 1319
i2, 1319
i3, 1319
i4, 1320
i5, 1320
i6_fp, 1320
i7, 1320
isr_dispatch_disable, 1320
l0_and_l1, 1321
l2, 1321
l3, 1321
l4, 1321
l5, 1321
l6, 1322
l7, 1322
o6_sp, 1322
o7, 1322
psr, 1322
SPARC_context_control_fp, 1323
f0_f1, 1323
f10_f11, 1324
f12_f13, 1324
f14_f15, 1324
f16_f17, 1324
f18_f19, 1324
f20_f21, 1325
f22_f23, 1325
f24_f25, 1325
f26_f27, 1325
f28_f29, 1325
f2_f3, 1326
f30_f31, 1326
f4_f5, 1326
f6_f7, 1326
f8_f9, 1326
fsr, 1327
CONTEXT_CONTROL_FP_SIZE
SPARC Context Structures, 838
CONTEXT_FP_SIZE
Context Handler, 254
control
Per_CPU_Control, 1374
control_entry
rtems_driver_address_table, 1403
CORE_barrier_Attributes, 1327
discipline, 1327
maximum_count, 1327
CORE_BARRIER_AUTOMATIC_RELEASE
Barrier Handler, 106
CORE_barrier_Control, 1328
Attributes, 1328
number_of_waiting_threads, 1328
Wait_queue, 1329
CORE_barrier_Disciplines
Barrier Handler, 106
CORE_BARRIER_MANUAL_RELEASE
Barrier Handler, 106
CORE_ceiling_mutex_Control, 1329
Core_control
Semaphore_Control, 1474
CORE_message_queue_Allocate_buffers
Message Queue Handler, 555
CORE_message_queue_Buffer, 1330
buffer, 1330
CORE_message_queue_Control, 1330
free_message_buffers, 1331
Inactive_messages, 1331

maximum_message_size, 1331
maximum_pending_messages, 1332
message_buffers, 1332
number_of_pending_messages, 1332
Pending_messages, 1332
Wait_queue, 1332
CORE_message_queue_Disciplines
Message Queue Handler, 556
CORE_MESSAGE_QUEUE_DISCIPLINES_FIFO
Message Queue Handler, 557
CORE_MESSAGE_QUEUE_DISCIPLINES_PRIORITY
Message Queue Handler, 557
CORE_MESSAGE_QUEUE_SEND_REQUEST
Message Queue Handler, 555
CORE_message_queue_Submit_types
Message Queue Handler, 556
CORE_MESSAGE_QUEUE_URGENT_REQUEST
Message Queue Handler, 555
CORE_mutex_Control, 1333
Wait_queue, 1333
CORE_recursive_mutex_Control, 1334
CORE_semaphore_Control, 1334
count, 1334
Wait_queue, 1335
count
CORE_semaphore_Control, 1334
Rate_monotonic_Statistics, 1392
rtems_rate_monotonic_period_statistics, 1425
Thread_Wait_information, 1526
CPU Architecture Support, 145
cpu.c
_CPU_Context_Initialize, 1847
_CPU_ISR_Get_level, 1848
_CPU_ISR_install_raw_handler, 1848
_CPU_ISR_install_vector, 1849
_CPU_Initialize, 1848
_CPU_Trap_slot_template, 1849
SPARC_ASSERT_FP_OFFSET, 1846
SPARC_ASSERT_ISF_OFFSET, 1846
SPARC_ASSERT_OFFSET, 1847
cpu.h
_CPU_Context_Initialization_at_thread_begin, 1854
_CPU_Context_Initialize, 1866
_CPU_Context_Restart_self, 1854
_CPU_Context_restore, 1867
_CPU_Context_switch, 1867
_CPU_Fatal_halt, 1867
_CPU_ISR_Disable, 1855
_CPU_ISR_Enable, 1855
_CPU_ISR_Flash, 1855
_CPU_ISR_Get_level, 1868
_CPU_ISR_Set_level, 1856
_CPU_ISR_install_raw_handler, 1868
_CPU_ISR_install_vector, 1869
_CPU_Initialize, 1868
_CPU_Initialize_vectors, 1855
_CPU_Trap_slot_template, 1870
CPU_ALIGNMENT, 1856
CPU_ALL_TASKS_ARE_FP, 1856
CPU_CONTEXT_FP_SIZE, 1856
CPU_HARDWARE_FP, 1857
CPU_HEAP_ALIGNMENT, 1857
CPU_IDLE_TASK_IS_FP, 1857
CPU_INTERRUPT_MAXIMUM_VECTOR_NUMBER, 1857
CPU_INTERRUPT_NUMBER_OF_VECTORS, 1858
CPU_ISR_PASSES_FRAME_POINTER, 1858
CPU_MODES_INTERRUPT_MASK, 1858
CPU_MPCI_RECEIVE_SERVER_EXTRA_STACK, 1858
CPU_PROVIDES_ISR_IS_IN_PROGRESS, 1859
CPU_SIMPLE_VECTORED_INTERRUPTS, 1859
CPU_SIZEOF_POINTER, 1859
CPU_SOFTWARE_FP, 1859
CPU_STACK_ALIGNMENT, 1860
CPU_STACK_FRAME_I0_OFFSET, 1860
CPU_STACK_FRAME_I1_OFFSET, 1860
CPU_STACK_FRAME_I2_OFFSET, 1860
CPU_STACK_FRAME_I3_OFFSET, 1860
CPU_STACK_FRAME_I4_OFFSET, 1861
CPU_STACK_FRAME_I5_OFFSET, 1861
CPU_STACK_FRAME_I6_FP_OFFSET, 1861
CPU_STACK_FRAME_I7_OFFSET, 1861
CPU_STACK_FRAME_L0_OFFSET, 1861
CPU_STACK_FRAME_L1_OFFSET, 1862
CPU_STACK_FRAME_L2_OFFSET, 1862
CPU_STACK_FRAME_L3_OFFSET, 1862
CPU_STACK_FRAME_L4_OFFSET, 1862
CPU_STACK_FRAME_L5_OFFSET, 1862
CPU_STACK_FRAME_L6_OFFSET, 1863
CPU_STACK_FRAME_L7_OFFSET, 1863
CPU_STACK_FRAME_PAD0_OFFSET, 1863
CPU_STACK_FRAME_SAVED_ARG0_OFFSET, 1863
CPU_STACK_FRAME_SAVED_ARG1_OFFSET, 1863
CPU_STACK_FRAME_SAVED_ARG2_OFFSET, 1864
CPU_STACK_FRAME_SAVED_ARG3_OFFSET, 1864
CPU_STACK_FRAME_SAVED_ARG4_OFFSET, 1864
CPU_STACK_FRAME_SAVED_ARG5_OFFSET, 1864
CPU_STACK_GROWS_UP, 1864
CPU_STACK_MINIMUM_SIZE, 1865
CPU_STRUCTURE_RETURN_ADDRESS_OFFSET, 1865
CPU_swap_u16, 1865
CPU_swap_u32, 1869
CPU_Uint32ptr, 1866
CPU_USE_GENERIC_BITFIELD_CODE, 1866
CPU_ALIGNMENT
cpu.h, 1856

CPU_ALL_TASKS_ARE_FP
cpu.h, 1856
CPU_CONTEXT_FP_SIZE
cpu.h, 1856
CPU_Exception_frame, 1335
CPU_HARDWARE_FP
cpu.h, 1857
CPU_HEAP_ALIGNMENT
cpu.h, 1857
CPU_IDLE_TASK_IS_FP
cpu.h, 1857
CPU_Interrupt_frame, 1335
g1, 1336
g2, 1336
g3, 1337
g4, 1337
g5, 1337
g7, 1337
i0, 1337
i1, 1338
i2, 1338
i3, 1338
i4, 1338
i5, 1338
i6_fp, 1339
i7, 1339
npc, 1339
pc, 1339
psr, 1339
reserved_for_alignment, 1340
Stack_frame, 1340
tpc, 1340
y, 1340
CPU_INTERRUPT_FRAME_SIZE
SPARC, 829
CPU_INTERRUPT_MAXIMUM_VECTOR_NUMBER
cpu.h, 1857
CPU_INTERRUPT_NUMBER_OF_VECTORS
cpu.h, 1858
CPU_ISR_PASSES_FRAME_POINTER
cpu.h, 1858
CPU_MODEL_NAME
sparc.h, 1874
CPU_MODES_INTERRUPT_MASK
cpu.h, 1858
CPU_MPCI_RECEIVE_SERVER_EXTRA_STACK
cpu.h, 1858
CPU_NAME
sparc.h, 1874
CPU_Per_CPU_control, 1341
CPU_PROVIDES_ISR_IS_IN_PROGRESS
cpu.h, 1859
CPU_SIMPLE_VECTORED_INTERRUPTS
cpu.h, 1859
CPU_SIZEOF_POINTER
cpu.h, 1859
CPU_SOFTWARE_FP
cpu.h, 1859
CPU_STACK_ALIGNMENT
cpu.h, 1860
CPU_STACK_FRAME_I0_OFFSET
cpu.h, 1860
CPU_STACK_FRAME_I1_OFFSET
cpu.h, 1860
CPU_STACK_FRAME_I2_OFFSET
cpu.h, 1860
CPU_STACK_FRAME_I3_OFFSET
cpu.h, 1860
CPU_STACK_FRAME_I4_OFFSET
cpu.h, 1861
CPU_STACK_FRAME_I5_OFFSET
cpu.h, 1861
CPU_STACK_FRAME_I6_FP_OFFSET
cpu.h, 1861
CPU_STACK_FRAME_I7_OFFSET
cpu.h, 1861
CPU_STACK_FRAME_L0_OFFSET
cpu.h, 1861
CPU_STACK_FRAME_L1_OFFSET
cpu.h, 1862
CPU_STACK_FRAME_L2_OFFSET
cpu.h, 1862
CPU_STACK_FRAME_L3_OFFSET
cpu.h, 1862
CPU_STACK_FRAME_L4_OFFSET
cpu.h, 1862
CPU_STACK_FRAME_L5_OFFSET
cpu.h, 1862
CPU_STACK_FRAME_L6_OFFSET
cpu.h, 1863
CPU_STACK_FRAME_L7_OFFSET
cpu.h, 1863
CPU_STACK_FRAME_PAD0_OFFSET
cpu.h, 1863
CPU_STACK_FRAME_SAVED_ARG0_OFFSET
cpu.h, 1863
CPU_STACK_FRAME_SAVED_ARG1_OFFSET
cpu.h, 1863
CPU_STACK_FRAME_SAVED_ARG2_OFFSET
cpu.h, 1864
CPU_STACK_FRAME_SAVED_ARG3_OFFSET
cpu.h, 1864
CPU_STACK_FRAME_SAVED_ARG4_OFFSET
cpu.h, 1864
CPU_STACK_FRAME_SAVED_ARG5_OFFSET
cpu.h, 1864
CPU_STACK_GROWS_UP
cpu.h, 1864
CPU_STACK_MINIMUM_SIZE
cpu.h, 1865
CPU_STRUCTURE_ALIGNMENT
RTEMS Per CPU Information, 716
CPU_STRUCTURE_RETURN_ADDRESS_OFFSET
cpu.h, 1865
CPU_swap_u16
cpu.h, 1865

CPU_swap_u32
    cpu.h, 1869
cpu_time_budget
    _Thread_Control, 1301
cpu_time_used
    _Thread_Control, 1301
CPU_Trap_table_entry, 1341
    jmp_to_low_of_handler_plus_l4, 1341
    mov_psr_l0, 1341
    mov_vector_l3, 1342
    sethi_of_handler_to_l4, 1342
CPU_Uint32ptr
    cpu.h, 1866
cpu_usage_period_initiated
    Rate_monotonic_Control, 1390
cpu_usage_timestamp
    Per_CPU_Control, 1374
CPU_USE_GENERIC_BITFIELD_CODE
    cpu.h, 1866
cpukit/include/rtems.h, 1571
cpukit/include/rtems/assoc.h, 1571
cpukit/include/rtems/bsplo.h, 1572
cpukit/include/rtems/chain.h, 1577
cpukit/include/rtems/clockdrv.h, 1580
cpukit/include/rtems/confdefs.h, 1581
cpukit/include/rtems/confdefs/bdbuf.h, 1581
cpukit/include/rtems/confdefs/bsp.h, 1561
cpukit/include/rtems/confdefs/clock.h, 1582
cpukit/include/rtems/confdefs/console.h, 1584
cpukit/include/rtems/confdefs/extensions.h, 1584
cpukit/include/rtems/confdefs/inittask.h, 1584
cpukit/include/rtems/confdefs/initthread.h, 1584
cpukit/include/rtems/confdefs/iodrivers.h, 1585
cpukit/include/rtems/confdefs/libio.h, 1585
cpukit/include/rtems/confdefs/libpci.h, 1592
cpukit/include/rtems/confdefs/malloc.h, 1592
cpukit/include/rtems/confdefs/mpci.h, 1597
cpukit/include/rtems/confdefs/newlib.h, 1597
cpukit/include/rtems/confdefs/objectsclassic.h, 1597
cpukit/include/rtems/confdefs/objectsposix.h, 1597
cpukit/include/rtems/confdefs/obsolete.h, 1598
cpukit/include/rtems/confdefs/percpu.h, 1598
cpukit/include/rtems/confdefs/scheduler.h, 1600
cpukit/include/rtems/confdefs/threads.h, 1603
cpukit/include/rtems/confdefs/unlimited.h, 1604
cpukit/include/rtems/confdefs/wkspace.h, 1604
cpukit/include/rtems/confdefs/wkspacesupport.h, 1605
cpukit/include/rtems/config.h, 1605
cpukit/include/rtems/counter.h, 1608
cpukit/include/rtems/extension.h, 1608
cpukit/include/rtems/extensiondata.h, 1609
cpukit/include/rtems/extensionimpl.h, 1610
cpukit/include/rtems/fatal.h, 1544
cpukit/include/rtems/fs.h, 1610
cpukit/include/rtems/init.h, 1611
cpukit/include/rtems/io.h, 1612
cpukit/include/rtems/irq-extension.h, 1613
cpukit/include/rtems/libcsupport.h, 1615
cpukit/include/rtems/libio.h, 1585
cpukit/include/rtems/malloc.h, 1592
cpukit/include/rtems/mallocinitone.h, 1616
cpukit/include/rtems/posix/spinlockimpl.h, 1617
cpukit/include/rtems/print.h, 1617
cpukit/include/rtems/printer.h, 1619
cpukit/include/rtems/rtems/asr.h, 1620
cpukit/include/rtems/rtems/asrdata.h, 1621
cpukit/include/rtems/rtems/asrimpl.h, 1622
cpukit/include/rtems/rtems/attr.h, 1622
cpukit/include/rtems/rtems/attrimpl.h, 1624
cpukit/include/rtems/rtems/barrier.h, 1625
cpukit/include/rtems/rtems/barrierdata.h, 1626
cpukit/include/rtems/rtems/barrierimpl.h, 1626
cpukit/include/rtems/rtems/cache.h, 1627
cpukit/include/rtems/rtems/clock.h, 1582
cpukit/include/rtems/rtems/config.h, 1607
cpukit/include/rtems/rtems/dpmem.h, 1628
cpukit/include/rtems/rtems/dpmemdata.h, 1629
cpukit/include/rtems/rtems/dpmemimpl.h, 1629
cpukit/include/rtems/rtems/event.h, 1630
cpukit/include/rtems/rtems/eventdata.h, 1634
cpukit/include/rtems/rtems/eventimpl.h, 1634
cpukit/include/rtems/rtems/intr.h, 1635
cpukit/include/rtems/rtems/mainpage.h, 1636
cpukit/include/rtems/rtems/message.h, 1637
cpukit/include/rtems/rtems/messagedata.h, 1638
cpukit/include/rtems/rtems/messageimpl.h, 1638
cpukit/include/rtems/rtems/modes.h, 1639
cpukit/include/rtems/rtems/modesimpl.h, 1640
cpukit/include/rtems/rtems/object.h, 1641
cpukit/include/rtems/rtems/objectimpl.h, 1644
cpukit/include/rtems/rtems/options.h, 1648
cpukit/include/rtems/rtems/optionsimpl.h, 1648
cpukit/include/rtems/rtems/part.h, 1649
cpukit/include/rtems/rtems/partdata.h, 1650
cpukit/include/rtems/rtems/partimpl.h, 1650
cpukit/include/rtems/rtems/ratemon.h, 1651
cpukit/include/rtems/rtems/ratemondata.h, 1652
cpukit/include/rtems/rtems/ratemonimpl.h, 1653
cpukit/include/rtems/rtems/region.h, 1654
cpukit/include/rtems/rtems/regiondata.h, 1654
cpukit/include/rtems/rtems/regionimpl.h, 1655
cpukit/include/rtems/rtems/sem.h, 1656
cpukit/include/rtems/rtems/semdata.h, 1656
cpukit/include/rtems/rtems/semimpl.h, 1657
cpukit/include/rtems/rtems/signal.h, 1658
cpukit/include/rtems/rtems/signalimpl.h, 1658
cpukit/include/rtems/rtems/status.h, 1659
cpukit/include/rtems/rtems/statusimpl.h, 1660
cpukit/include/rtems/rtems/support.h, 1660
cpukit/include/rtems/rtems/tasks.h, 1661
cpukit/include/rtems/rtems/tasksdata.h, 1664
cpukit/include/rtems/rtems/tasksimpl.h, 1665
cpukit/include/rtems/rtems/timer.h, 1666
cpukit/include/rtems/rtems/timerdata.h, 1667
cpukit/include/rtems/rtems/timerimpl.h, 1668
cpukit/include/rtems/rtems/types.h, 1669

cpukit/include/rtems/scheduler.h, 1600
cpukit/include/rtems/score/address.h, 1670
cpukit/include/rtems/score/apimutex.h, 1671
cpukit/include/rtems/score/assert.h, 1671
cpukit/include/rtems/score/atomic.h, 1672
cpukit/include/rtems/score/basedefs.h, 1673
cpukit/include/rtems/score/chain.h, 1579
cpukit/include/rtems/score/chainimpl.h, 1676
cpukit/include/rtems/score/context.h, 1679
cpukit/include/rtems/score/copyrt.h, 1679
cpukit/include/rtems/score/corebarrier.h, 1680
cpukit/include/rtems/score/corebarrierimpl.h, 1680
cpukit/include/rtems/score/coremsg.h, 1681
cpukit/include/rtems/score/coremsgbuffer.h, 1682
cpukit/include/rtems/score/coremsgimpl.h, 1683
cpukit/include/rtems/score/coremutex.h, 1685
cpukit/include/rtems/score/coremuteximpl.h, 1685
cpukit/include/rtems/score/coresem.h, 1687
cpukit/include/rtems/score/coresemimpl.h, 1687
cpukit/include/rtems/score/cpustdatomic.h, 1688
cpukit/include/rtems/score/freechain.h, 1690
cpukit/include/rtems/score/freechainimpl.h, 1691
cpukit/include/rtems/score/heap.h, 1692
cpukit/include/rtems/score/heapimpl.h, 1693
cpukit/include/rtems/score/heapinfo.h, 1696
cpukit/include/rtems/score/interr.h, 1696
cpukit/include/rtems/score/isr.h, 1698
cpukit/include/rtems/score/isrlevel.h, 1699
cpukit/include/rtems/score/isrlock.h, 1699
cpukit/include/rtems/score/memory.h, 1701
cpukit/include/rtems/score/mppkt.h, 1702
cpukit/include/rtems/score/mrsp.h, 1703
cpukit/include/rtems/score/mrspimpl.h, 1703
cpukit/include/rtems/score/muteximpl.h, 1705
cpukit/include/rtems/score/object.h, 1643
cpukit/include/rtems/score/objectdata.h, 1705
cpukit/include/rtems/score/objectimpl.h, 1645
cpukit/include/rtems/score/percpu.h, 1598
cpukit/include/rtems/score/percpudata.h, 1706
cpukit/include/rtems/score/priority.h, 1707
cpukit/include/rtems/score/prioritybitmap.h, 1708
cpukit/include/rtems/score/prioritybitmapimpl.h, 1708
cpukit/include/rtems/score/priorityimpl.h, 1709
cpukit/include/rtems/score/processormask.h, 1711
cpukit/include/rtems/score/profiling.h, 1713
cpukit/include/rtems/score/protectedheap.h, 1714
cpukit/include/rtems/score/rbtree.h, 1715
cpukit/include/rtems/score/rbtreeimpl.h, 1717
cpukit/include/rtems/score/scheduler.h, 1601
cpukit/include/rtems/score/scheduleredf.h, 1717
cpukit/include/rtems/score/scheduleredfimpl.h, 1719
cpukit/include/rtems/score/scheduleredfsmp.h, 1719
cpukit/include/rtems/score/schedulerimpl.h, 1721
cpukit/include/rtems/score/schedulernode.h, 1724
cpukit/include/rtems/score/schedulernodeimpl.h, 1724
cpukit/include/rtems/score/schedulerpriority.h, 1726
cpukit/include/rtems/score/schedulerpriorityimpl.h, 1727
cpukit/include/rtems/score/schedulersimple.h, 1728
cpukit/include/rtems/score/schedulersimpleimpl.h, 1729
cpukit/include/rtems/score/schedulersmp.h, 1729
cpukit/include/rtems/score/schedulersmpimpl.h, 1730
cpukit/include/rtems/score/semaphoreimpl.h, 1733
cpukit/include/rtems/score/smp.h, 1734
cpukit/include/rtems/score/smpbarrier.h, 1734
cpukit/include/rtems/score/smpimpl.h, 1735
cpukit/include/rtems/score/smplock.h, 1737
cpukit/include/rtems/score/smplockmcs.h, 1738
cpukit/include/rtems/score/smplockseq.h, 1739
cpukit/include/rtems/score/smplockstats.h, 1740
cpukit/include/rtems/score/smplockticket.h, 1740
cpukit/include/rtems/score/stack.h, 1741
cpukit/include/rtems/score/stackimpl.h, 1742
cpukit/include/rtems/score/states.h, 1743
cpukit/include/rtems/score/statesimpl.h, 1743
cpukit/include/rtems/score/sysstate.h, 1745
cpukit/include/rtems/score/thread.h, 1746
cpukit/include/rtems/score/threaddispatch.h, 1748
cpukit/include/rtems/score/threadidledata.h, 1749
cpukit/include/rtems/score/threadimpl.h, 1750
cpukit/include/rtems/score/threadq.h, 1757
cpukit/include/rtems/score/threadqimpl.h, 1758
cpukit/include/rtems/score/timecounter.h, 1763
cpukit/include/rtems/score/timecounterimpl.h, 1765
cpukit/include/rtems/score/timespec.h, 1766
cpukit/include/rtems/score/timestamp.h, 1767
cpukit/include/rtems/score/timestampimpl.h, 1767
cpukit/include/rtems/score/tls.h, 1768
cpukit/include/rtems/score/todimpl.h, 1769
cpukit/include/rtems/score/userext.h, 1772
cpukit/include/rtems/score/userextdata.h, 1773
cpukit/include/rtems/score/userextimpl.h, 1773
cpukit/include/rtems/score/watchdog.h, 1776
cpukit/include/rtems/score/watchdogimpl.h, 1776
cpukit/include/rtems/score/watchdogticks.h, 1779
cpukit/include/rtems/score/wkspace.h, 1604
cpukit/include/rtems/score/wkspacedata.h, 1779
cpukit/include/rtems/score/wkspaceinitone.h, 1780
cpukit/include/rtems/termiostypes.h, 1781
cpukit/include/rtems/timecounter.h, 1764
cpukit/include/rtems/version.h, 1788
cpukit/include/sys/statvfs.h, 1788
cpukit/libcsupport/src/malloc_deferred.c, 1789
cpukit/libcsupport/src/mallocheap.c, 1789
cpukit/libcsupport/src/print_printf.c, 1790
cpukit/libcsupport/src/printk.c, 1791
cpukit/libcsupport/src/printk_plugin.c, 1792
cpukit/libcsupport/src/rtems_putc.c, 1792
cpukit/libcsupport/src/termiosinitialize.c, 1794
cpukit/libcsupport/src/vprintk.c, 1794
cpukit/libtest/t-test-busy-tick.c, 1795
cpukit/libtest/t-test-busy.c, 1795
cpukit/libtest/t-test-interrupt.c, 1796
cpukit/libtest/t-test-rtems-objs.c, 1798
cpukit/libtest/t-test-rtems.h, 1799
cpukit/libtest/t-test-thread-switch.c, 1799
cpukit/libtest/testrun.c, 1800

cpukit/posix/src/pspindestroy.c, 1801
cpukit/posix/src/pspininit.c, 1802
cpukit/posix/src/pspinlock.c, 1802
cpukit/posix/src/pspinunlock.c, 1803
cpukit/rtems/src/barrier.c, 1803
cpukit/rtems/src/barriercreate.c, 1804
cpukit/rtems/src/barrierdelete.c, 1804
cpukit/rtems/src/barrierident.c, 1805
cpukit/rtems/src/barrierrelease.c, 1805
cpukit/rtems/src/barrierwait.c, 1806
cpukit/rtems/src/clockgetuptime.c, 1806
cpukit/rtems/src/clocktodtoseconds.c, 1806
cpukit/rtems/src/clocktodvalidate.c, 1808
cpukit/rtems/src/eventreceive.c, 1809
cpukit/rtems/src/eventseize.c, 1809
cpukit/rtems/src/eventsend.c, 1810
cpukit/rtems/src/eventsurrender.c, 1810
cpukit/rtems/src/intrcatch.c, 1811
cpukit/rtems/src/msg.c, 1811
cpukit/rtems/src/msgqbroadcast.c, 1811
cpukit/rtems/src/msgqconstruct.c, 1812
cpukit/rtems/src/msgqcreate.c, 1813
cpukit/rtems/src/msgqdelete.c, 1813
cpukit/rtems/src/msgqflush.c, 1814
cpukit/rtems/src/msgqgetnumberpending.c, 1814
cpukit/rtems/src/msgqident.c, 1814
cpukit/rtems/src/msgqreceive.c, 1815
cpukit/rtems/src/msgqsend.c, 1815
cpukit/rtems/src/msgqurgent.c, 1816
cpukit/rtems/src/part.c, 1816
cpukit/rtems/src/partcreate.c, 1816
cpukit/rtems/src/partdelete.c, 1817
cpukit/rtems/src/partgetbuffer.c, 1818
cpukit/rtems/src/partident.c, 1818
cpukit/rtems/src/partreturnbuffer.c, 1819
cpukit/rtems/src/ratemon.c, 1819
cpukit/rtems/src/ratemoncancel.c, 1819
cpukit/rtems/src/ratemoncreate.c, 1820
cpukit/rtems/src/ratemondelete.c, 1821
cpukit/rtems/src/ratemongetstatistics.c, 1821
cpukit/rtems/src/ratemongetstatus.c, 1821
cpukit/rtems/src/ratemonident.c, 1822
cpukit/rtems/src/ratemonperiod.c, 1822
cpukit/rtems/src/ratemonresetstatistics.c, 1823
cpukit/rtems/src/ratemontimeout.c, 1823
cpukit/rtems/src/rtemsnametoid.c, 1824
cpukit/rtems/src/sem.c, 1824
cpukit/rtems/src/semcreate.c, 1825
cpukit/rtems/src/semdelete.c, 1826
cpukit/rtems/src/semident.c, 1826
cpukit/rtems/src/semobtain.c, 1827
cpukit/rtems/src/semrelease.c, 1827
cpukit/rtems/src/status.c, 1828
cpukit/rtems/src/statustext.c, 1828
cpukit/rtems/src/systemeventreceive.c, 1828
cpukit/rtems/src/systemeventsend.c, 1829
cpukit/rtems/src/taskconstruct.c, 1830
cpukit/rtems/src/taskdelete.c, 1832
cpukit/rtems/src/taskgetaffinity.c, 1832
cpukit/rtems/src/taskident.c, 1832
cpukit/rtems/src/taskinitusers.c, 1833
cpukit/rtems/src/taskissuspended.c, 1833
cpukit/rtems/src/taskrestart.c, 1834
cpukit/rtems/src/taskresume.c, 1834
cpukit/rtems/src/tasks.c, 1834
cpukit/rtems/src/taskself.c, 1835
cpukit/rtems/src/tasksetaffinity.c, 1835
cpukit/rtems/src/tasksetpriority.c, 1835
cpukit/rtems/src/taskstart.c, 1836
cpukit/rtems/src/tasksuspend.c, 1836
cpukit/rtems/src/taskwakeafter.c, 1837
cpukit/rtems/src/timercreate.c, 1837
cpukit/rtems/src/timerdelete.c, 1838
cpukit/rtems/src/timerfireafter.c, 1839
cpukit/rtems/src/timerident.c, 1839
cpukit/rtems/src/timerreset.c, 1839
cpukit/sapi/src/exinit.c, 1840
cpukit/sapi/src/extension.c, 1841
cpukit/sapi/src/extensioncreate.c, 1841
cpukit/sapi/src/extensiondelete.c, 1842
cpukit/sapi/src/extensionident.c, 1843
cpukit/sapi/src/getversionstring.c, 1843
cpukit/sapi/src/version.c, 1843
cpukit/score/cpu/sparc/cpu.c, 1844
cpukit/score/cpu/sparc/include/rtems/asm.h, 1850
cpukit/score/cpu/sparc/include/rtems/score/cpu.h, 1850
cpukit/score/cpu/sparc/include/rtems/score/cpuimpl.h, 1870
cpukit/score/cpu/sparc/include/rtems/score/sparc.h, 1872
cpukit/score/src/apimutexisowner.c, 1886
cpukit/score/src/apimutexlock.c, 1887
cpukit/score/src/apimutexunlock.c, 1887
cpukit/score/src/chain.c, 1888
cpukit/score/src/corebarrier.c, 1888
cpukit/score/src/corebarrierrelease.c, 1888
cpukit/score/src/corebarrierwait.c, 1889
cpukit/score/src/coremsg.c, 1889
cpukit/score/src/coremsgbroadcast.c, 1890
cpukit/score/src/coremsgclose.c, 1890
cpukit/score/src/coremsgflush.c, 1891
cpukit/score/src/coremsginsert.c, 1891
cpukit/score/src/coremsgseize.c, 1892
cpukit/score/src/coremsgsubmit.c, 1892
cpukit/score/src/coremsgwkspace.c, 1893
cpukit/score/src/coremutexseize.c, 1893
cpukit/score/src/coresem.c, 1893
cpukit/score/src/coretod.c, 1894
cpukit/score/src/heap.c, 1894
cpukit/score/src/heapallocate.c, 1895
cpukit/score/src/interr.c, 1896
cpukit/score/src/isr.c, 1896
cpukit/score/src/isrisinprogress.c, 1897
cpukit/score/src/objectallocate.c, 1897
cpukit/score/src/objectallocatenone.c, 1897
cpukit/score/src/objectallocatestatic.c, 1898

cpukit/score/src/objectapimaximumclass.c, 1898
cpukit/score/src/objectclose.c, 1898
cpukit/score/src/objectfree.c, 1899
cpukit/score/src/objectfreestatic.c, 1899
cpukit/score/src/objectgetinfo.c, 1899
cpukit/score/src/objectgetinfoid.c, 1900
cpukit/score/src/objectgetlocal.c, 1900
cpukit/score/src/objectgetnameasstring.c, 1900
cpukit/score/src/objectgetnoprotection.c, 1901
cpukit/score/src/objectinitializeinformation.c, 1901
cpukit/score/src/objectnamespaceremove.c, 1902
cpukit/score/src/objectnametoid.c, 1902
cpukit/score/src/percpu.c, 1903
cpukit/score/src/processormaskcopy.c, 1903
cpukit/score/src/rbtreenext.c, 1904
cpukit/score/src/rbtreereplace.c, 1904
cpukit/score/src/scheduler.c, 1905
cpukit/score/src/schedulerdefaultnodedestroy.c, 1905
cpukit/score/src/schedulerdefaultnodeinit.c, 1905
cpukit/score/src/schedulerdefaultreleasejob.c, 1906
cpukit/score/src/schedulerdefaultsetaffinity.c, 1906
cpukit/score/src/schedulerdefaulttick.c, 1907
cpukit/score/src/scheduleredfreleasejob.c, 1907
cpukit/score/src/scheduleredfsmp.c, 1908
cpukit/score/src/schedulerpriorityblock.c, 1909
cpukit/score/src/schedulerprioritychangepriority.c, 1910
cpukit/score/src/schedulerpriorityschedule.c, 1910
cpukit/score/src/schedulerpriorityunblock.c, 1911
cpukit/score/src/schedulerpriorityyield.c, 1911
cpukit/score/src/smp.c, 1911
cpukit/score/src/stackallocatorfreenothing.c, 1913
cpukit/score/src/thread.c, 1913
cpukit/score/src/threadchangepriority.c, 1914
cpukit/score/src/threadclearstate.c, 1915
cpukit/score/src/threadcreateidle.c, 1915
cpukit/score/src/threaddispatch.c, 1916
cpukit/score/src/threadget.c, 1917
cpukit/score/src/threadhandler.c, 1917
cpukit/score/src/threadinitialize.c, 1918
cpukit/score/src/threadloadenv.c, 1918
cpukit/score/src/threadq.c, 1918
cpukit/score/src/threadqenqueue.c, 1919
cpukit/score/src/threadqflush.c, 1921
cpukit/score/src/threadrestart.c, 1922
cpukit/score/src/threadsetstate.c, 1924
cpukit/score/src/threadstackallocate.c, 1924
cpukit/score/src/threadstart.c, 1925
cpukit/score/src/threadstartmultitasking.c, 1925
cpukit/score/src/threadtimeout.c, 1926
cpukit/score/src/threadyield.c, 1926
cpukit/score/src/userext.c, 1926
cpukit/score/src/userextaddset.c, 1927
cpukit/score/src/userextiterate.c, 1927
cpukit/score/src/userextremoveset.c, 1928
cpukit/score/src/watchdoginsert.c, 1929
cpukit/score/src/watchdogremove.c, 1929
cpukit/score/src/watchdogtickssinceboot.c, 1929
cpukit/score/src/wkspace.c, 1930
cpukit/score/src/wkspaceallocate.c, 1931
cpukit/score/src/wkspacemallocinitdefault.c, 1931
cpukit/score/src/wkspacemallocinitunified.c, 1931
current_state
    _Thread_Control, 1302
    Thread_Proxy_control, 1509
currentloc
    rtems_filesystem_eval_path_context_t, 1405

data
    Per_CPU_Control, 1374
deadlock_callout
    Thread_queue_Context, 1511
deallocate
    Objects_Information, 1367
DEFAULT_INITIAL_EXTENSION Support, 255
deferred_released_count
    rtems_filesystem_global_location_t, 1408
deferred_released_next
    rtems_filesystem_global_location_t, 1408
destroy
    rtems_interrupt_server_config, 1415
Destructors, 256
Deterministic Priority Scheduler, 257
    _Scheduler_priority_Block, 259
    _Scheduler_priority_Extract_body, 259
    _Scheduler_priority_Get_context, 260
    _Scheduler_priority_Initialize, 260
    _Scheduler_priority_Node_downcast, 260
    _Scheduler_priority_Node_initialize, 261
    _Scheduler_priority_Ready_queue_enqueue, 261
    _Scheduler_priority_Ready_queue_enqueue_first, 262
    _Scheduler_priority_Ready_queue_extract, 262
    _Scheduler_priority_Ready_queue_first, 262
    _Scheduler_priority_Ready_queue_initialize, 264
    _Scheduler_priority_Ready_queue_update, 264
    _Scheduler_priority_Schedule, 265
    _Scheduler_priority_Schedule_body, 265
    _Scheduler_priority_Thread_get_node, 265
    _Scheduler_priority_Unblock, 266
    _Scheduler_priority_Update_priority, 266
    _Scheduler_priority_Yield, 266
    SCHEDULER_PRIORITY_ENTRY_POINTS, 258
Device Driver Configuration, 268
    CONFIGURE_APPLICATION_DOES_NOT_NEED_CLOCK_DRIVER 268
    CONFIGURE_APPLICATION_EXTRA_DRIVERS, 269
    CONFIGURE_APPLICATION_NEEDS_ATA_DRIVER, 269
    CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER, 270
    CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER, 270
    CONFIGURE_APPLICATION_NEEDS_FRAME_BUFFER_DRIVER, 271
    CONFIGURE_APPLICATION_NEEDS_IDE_DRIVER, 271

CONFIGURE_APPLICATION_NEEDS_NULL_DRIVER, 272
CONFIGURE_APPLICATION_NEEDS_RTC_DRIVER, 272
CONFIGURE_APPLICATION_NEEDS_SIMPLE_CONSOLE_DRIVER, 273
CONFIGURE_APPLICATION_NEEDS_SIMPLE_TASK_CONSOLE_DRIVER, 273
CONFIGURE_APPLICATION_NEEDS_STUB_DRIVER, 274
CONFIGURE_APPLICATION_NEEDS_TIMER_DRIVER, 274
CONFIGURE_APPLICATION_NEEDS_WATCHDOG_DRIVER, 275
CONFIGURE_APPLICATION_NEEDS_ZERO_DRIVER, 275
CONFIGURE_APPLICATION_PREREQUISITE_DRIVERS, 276
CONFIGURE_ATA_DRIVER_TASK_PRIORITY, 276
CONFIGURE_MAXIMUM_DRIVERS, 277
Device Drivers, 279
direction
Chain_Iterator, 1315
Directive Attributes, 280
rtems_attribute, 283
RTEMS_FLOATING_POINT, 281
RTEMS_GLOBAL, 281
RTEMS_INHERIT_PRIORITY, 281
RTEMS_LOCAL, 282
RTEMS_MULTIPROCESSOR_RESOURCE_SHARING, 282
RTEMS_NO_FLOATING_POINT, 282
RTEMS_PRIORITY_CEILING, 282
Directive Options, 284
RTEMS_NO_WAIT, 284
RTEMS_WAIT, 284
Directive Status Codes, 285
RTEMS_ALREADY_SUSPENDED, 286
rtems_are_statuses_equal, 287
RTEMS_CALLED_FROM_ISR, 286
RTEMS_ILLEGAL_ON_REMOTE_OBJECT, 286
RTEMS_ILLEGAL_ON_SELF, 286
RTEMS_INCORRECT_STATE, 286
RTEMS_INTERNAL_ERROR, 287
RTEMS_INTERRUPTED, 287
RTEMS_INVALID_ADDRESS, 286
RTEMS_INVALID_CLOCK, 286
RTEMS_INVALID_ID, 286
RTEMS_INVALID_NAME, 286
RTEMS_INVALID_NODE, 286
RTEMS_INVALID_NUMBER, 286
RTEMS_INVALID_PRIORITY, 286
RTEMS_INVALID_SIZE, 286
RTEMS_IO_ERROR, 287
rtems_is_status_successful, 287
RTEMS_MP_NOT_CONFIGURED, 286
RTEMS_NO_MEMORY, 287
RTEMS_NOT_CONFIGURED, 286
RTEMS_NOT_DEFINED, 286
RTEMS_NOT_IMPLEMENTED, 287
RTEMS_NOT_OWNER_OF_RESOURCE, 286
RTEMS_OBJECT_WAS_DELETED, 286
RTEMS_PROXY_BLOCKING, 287
RTEMS_RESOURCE_IN_USE, 286
rtems_status_code, 286
rtems_status_code_to_errno, 288
rtems_status_text, 289
RTEMS_SUCCESSFUL, 286
RTEMS_TASK_EXITTED, 286
RTEMS_TIMEOUT, 286
RTEMS_TOO_MANY, 286
RTEMS_UNSATISFIED, 286
discipline
CORE_barrier_Attributes, 1327
dispatch_necessary
Per_CPU_Control, 1375
done
Per_CPU_Job, 1380
Dual Ported Memory Manager Implementation, 290
_Dual_ported_memory_Allocate, 291
_Dual_ported_memory_Free, 291
DUAL_PORTED_MEMORY_INFORMATION_DEFINE, 290
Dual-Ported Memory Manager, 292
rtems_port_create, 292
rtems_port_delete, 293
rtems_port_external_to_internal, 293
rtems_port_ident, 293
rtems_port_internal_to_external, 294
Dual_ported_memory_Control, 1342
external_base, 1343
internal_base, 1343
length, 1343
Object, 1343
DUAL_PORTED_MEMORY_INFORMATION_DEFINE
Dual Ported Memory Manager Implementation, 290

EDF Priority SMP Scheduler, 295
_Scheduler_EDF_SMP_Add_processor, 297
_Scheduler_EDF_SMP_Ask_for_help, 297
_Scheduler_EDF_SMP_Block, 297
_Scheduler_EDF_SMP_Initialize, 298
_Scheduler_EDF_SMP_Node_initialize, 298
_Scheduler_EDF_SMP_Pin, 298
_Scheduler_EDF_SMP_Reconsider_help_request, 299
_Scheduler_EDF_SMP_Remove_processor, 299
_Scheduler_EDF_SMP_Set_affinity, 300
_Scheduler_EDF_SMP_Start_idle, 300
_Scheduler_EDF_SMP_Unblock, 301
_Scheduler_EDF_SMP_Unpin, 301
_Scheduler_EDF_SMP_Update_priority, 301
_Scheduler_EDF_SMP_Withdraw_node, 303
_Scheduler_EDF_SMP_Yield, 303
SCHEDULER_EDF_SMP_ENTRY_POINTS, 296
EDF Scheduler, 304

_Scheduler_EDF_Block, 306
_Scheduler_EDF_Cancel_job, 307
_Scheduler_EDF_Enqueue, 307
_Scheduler_EDF_Extract, 307
_Scheduler_EDF_Extract_body, 308
_Scheduler_EDF_Get_context, 308
_Scheduler_EDF_Initialize, 308
_Scheduler_EDF_Less, 309
_Scheduler_EDF_Map_priority, 309
_Scheduler_EDF_Node_downcast, 310
_Scheduler_EDF_Node_initialize, 310
_Scheduler_EDF_Priority_less_equal, 310
_Scheduler_EDF_Release_job, 311
_Scheduler_EDF_Schedule, 311
_Scheduler_EDF_Schedule_body, 312
_Scheduler_EDF_Thread_get_node, 312
_Scheduler_EDF_Unblock, 312
_Scheduler_EDF_Unmap_priority, 313
_Scheduler_EDF_Update_priority, 313
_Scheduler_EDF_Yield, 314
SCHEDULER_EDF_ENTRY_POINTS, 305
SCHEDULER_EDF_PRIO_MSB, 306
enqueue
Thread_queue_Operations, 1517
enqueue_callout
Thread_queue_Context, 1512
Entry
Thread_Start_information, 1524
entry_point
rtems_initialization_tasks_table, 1412
Event
RTEMS_API_Control, 1399
Event Implementation, 315
_Event_Initialize, 316
_Event_Seize, 316
_Event_Surrender, 318
_Event_sets_Clear, 317
_Event_sets_Get, 317
_Event_sets_Is_empty, 317
_Event_sets_Post, 318
Event Manager, 320
RTEMS_ALL_EVENTS, 321
rtems_event_receive, 322
rtems_event_send, 323
Event Recording Configuration, 325
CONFIGURE_RECORD_EXTENSIONS_ENABLED, 325
CONFIGURE_RECORD_FATAL_DUMP_BASE64, 325
CONFIGURE_RECORD_FATAL_DUMP_BASE64_ZLIB, 326
CONFIGURE_RECORD_PER_PROCESSOR_ITEMS, 326
event.h
rtems_event_system_receive, 1632
rtems_event_system_send, 1633
rtems_event_transient_receive, 1633
rtems_event_transient_send, 1633
Event_Control, 1344
executed_since_last_period
rtems_rate_monotonic_period_status, 1428
executing
Per_CPU_Control, 1375
exit_value
Thread_Life_control, 1508
Extension_Control, 1344
EXTENSION_INFORMATION_DEFINE
User Extensions Implementation, 1193
extensions
_Thread_Control, 1302
spec:/rtems/task/req/construct-errors, 1278
external_base
Dual_ported_memory_Control, 1343
extract
Thread_queue_Operations, 1517

f0_f1
Context_Control_fp, 1323
f10_f11
Context_Control_fp, 1324
f12_f13
Context_Control_fp, 1324
F12_F13_OFFSET
SPARC Context Structures, 839
f14_f15
Context_Control_fp, 1324
F14_F15_OFFSET
SPARC Context Structures, 839
f16_f17
Context_Control_fp, 1324
F16_F17_OFFSET
SPARC Context Structures, 839
f18_f19
Context_Control_fp, 1324
F18_F19_OFFSET
SPARC Context Structures, 839
F1O_F11_OFFSET
SPARC Context Structures, 839
f20_f21
Context_Control_fp, 1325
f22_f23
Context_Control_fp, 1325
F22_F23_OFFSET
SPARC Context Structures, 840
f24_f25
Context_Control_fp, 1325
F24_F25_OFFSET
SPARC Context Structures, 840
f26_f27
Context_Control_fp, 1325
F26_F27_OFFSET
SPARC Context Structures, 840
f28_f29
Context_Control_fp, 1325
F28_F29_OFFSET
SPARC Context Structures, 840
f2_f3

Context_Control_fp, 1326
F2_F3_OFFSET
SPARC Context Structures, 840
F2O_F21_OFFSET
SPARC Context Structures, 841
f30_f31
Context_Control_fp, 1326
F3O_F31_OFFSET
SPARC Context Structures, 841
f4_f5
Context_Control_fp, 1326
F4_F5_OFFSET
SPARC Context Structures, 841
f6_f7
Context_Control_fp, 1326
F6_F7_OFFSET
SPARC Context Structures, 841
f8_f9
Context_Control_fp, 1326
F8_F9_OFFSET
SPARC Context Structures, 841
f_bavail
statvfs, 1489
f_bfree
statvfs, 1489
f_blocks
statvfs, 1489
f_bsize
statvfs, 1489
f_favail
statvfs, 1489
f_ffree
statvfs, 1490
f_files
statvfs, 1490
f_flag
statvfs, 1490
f_frsize
statvfs, 1490
f_fsid
statvfs, 1490
f_namemax
statvfs, 1491
failed_expression
rtems_assert_context, 1400
Fatal Error Manager, 328
rtems_exception_frame_print, 328
rtems_fatal, 329
rtems_fatal_error_occurred, 329
rtems_fatal_source_text, 329
rtems_internal_error_text, 330
RTEMS_PRINTFLIKE, 330
ffclock_estimate, 1344
Fifo
_Thread_queue_Heads, 1305
file
rtems_assert_context, 1400
File System Node Handler, 331
rtems_filesystem_close_t, 332
rtems_filesystem_default_close, 341
rtems_filesystem_default_fcntl, 341
rtems_filesystem_default_fstat, 341
rtems_filesystem_default_fsync_or_fdatasync, 342
rtems_filesystem_default_fsync_or_fdatasync_success, 342
rtems_filesystem_default_ftruncate, 342
rtems_filesystem_default_ftruncate_directory, 343
rtems_filesystem_default_ioctl, 343
rtems_filesystem_default_kqfilter, 343
rtems_filesystem_default_lseek, 344
rtems_filesystem_default_lseek_directory, 344
rtems_filesystem_default_lseek_file, 345
rtems_filesystem_default_mmap, 345
rtems_filesystem_default_open, 346
rtems_filesystem_default_poll, 346
rtems_filesystem_default_read, 346
rtems_filesystem_default_readv, 347
rtems_filesystem_default_write, 347
rtems_filesystem_default_writev, 347
rtems_filesystem_fcntl_t, 333
rtems_filesystem_fdatasync_t, 333
rtems_filesystem_fstat_t, 334
rtems_filesystem_fsync_t, 334
rtems_filesystem_ftruncate_t, 335
rtems_filesystem_ioctl_t, 335
rtems_filesystem_kqfilter_t, 336
rtems_filesystem_lseek_t, 336
rtems_filesystem_mmap_t, 337
rtems_filesystem_open_t, 337
rtems_filesystem_poll_t, 338
rtems_filesystem_read_t, 338
rtems_filesystem_readv_t, 339
rtems_filesystem_write_t, 340
rtems_filesystem_writev_t, 340
File System Operations, 349
rtems_filesystem_are_nodes_equal_t, 351
rtems_filesystem_chown_t, 351
rtems_filesystem_clonenode_t, 352
rtems_filesystem_default_are_nodes_equal, 361
rtems_filesystem_default_chown, 361
rtems_filesystem_default_clonenode, 362
rtems_filesystem_default_eval_path, 362
rtems_filesystem_default_fchmod, 362
rtems_filesystem_default_freenode, 363
rtems_filesystem_default_fsunmount, 363
rtems_filesystem_default_link, 363
rtems_filesystem_default_lock, 363
rtems_filesystem_default_mknod, 364
rtems_filesystem_default_mount, 364
rtems_filesystem_default_readlink, 364
rtems_filesystem_default_rename, 365
rtems_filesystem_default_rmnod, 365
rtems_filesystem_default_statvfs, 366
rtems_filesystem_default_symlink, 366
rtems_filesystem_default_unlock, 366
rtems_filesystem_default_unmount, 366

rtems_filesystem_default_utime, 367
rtems_filesystem_eval_path_t, 352
rtems_filesystem_fchmod_t, 353
rtems_filesystem_freenode_t, 353
rtems_filesystem_fsmount_me_t, 354
rtems_filesystem_fsunmount_me_t, 354
rtems_filesystem_link_t, 354
rtems_filesystem_mknod_t, 355
rtems_filesystem_mount_t, 356
rtems_filesystem_mt_entry_lock_t, 356
rtems_filesystem_mt_entry_unlock_t, 357
rtems_filesystem_readlink_t, 357
rtems_filesystem_rename_t, 358
rtems_filesystem_rmnod_t, 358
rtems_filesystem_statvfs_t, 359
rtems_filesystem_symlink_t, 359
rtems_filesystem_unmount_t, 360
rtems_filesystem_utime_t, 360
File System Types and Mount, 368
mount, 370
mount_and_make_target_path, 371
rtems_filesystem_iterate, 372
rtems_filesystem_mount_iterate, 372
rtems_filesystem_mt_entry_visitor, 369
rtems_filesystem_register, 373
rtems_filesystem_table, 374
rtems_filesystem_unregister, 373
rtems_per_filesystem_routine, 369
unmount, 373
Filesystem Configuration, 375
CONFIGURE_APPLICATION_DISABLE_FILESYSTEM, 376
CONFIGURE_FILESYSTEM_ALL, 377
CONFIGURE_FILESYSTEM_DOSFS, 377
CONFIGURE_FILESYSTEM_FTPFS, 378
CONFIGURE_FILESYSTEM_IMFS, 378
CONFIGURE_FILESYSTEM_JFFS2, 378
CONFIGURE_FILESYSTEM_NFS, 379
CONFIGURE_FILESYSTEM_RFS, 379
CONFIGURE_FILESYSTEM_TFTPFS, 379
CONFIGURE_IMFS_DISABLE_CHMOD, 380
CONFIGURE_IMFS_DISABLE_CHOWN, 380
CONFIGURE_IMFS_DISABLE_LINK, 380
CONFIGURE_IMFS_DISABLE_MKNOD, 381
CONFIGURE_IMFS_DISABLE_MKNOD_DEVICE, 381
CONFIGURE_IMFS_DISABLE_MKNOD_FILE, 381
CONFIGURE_IMFS_DISABLE_MOUNT, 382
CONFIGURE_IMFS_DISABLE_READDIR, 382
CONFIGURE_IMFS_DISABLE_READLINK, 382
CONFIGURE_IMFS_DISABLE_RENAME, 383
CONFIGURE_IMFS_DISABLE_RMNOD, 383
CONFIGURE_IMFS_DISABLE_SYMLINK, 383
CONFIGURE_IMFS_DISABLE_UNMOUNT, 384
CONFIGURE_IMFS_DISABLE_UTIME, 384
CONFIGURE_IMFS_ENABLE_MKFIFO, 384
CONFIGURE_IMFS_MEMFILE_BYTES_PER_BLOCK, 385
CONFIGURE_USE_DEVFS_AS_BASE_FILESYSTEM, 385
CONFIGURE_USE_MINIIMFS_AS_BASE_FILESYSTEM, 386
fini
rtems_test_parallel_job, 1442
first_open
rtems_termios_device_handler, 1435
flags
rtems_filesystem_eval_path_context_t, 1405
Flexible Per-CPU Data, 388
PER_CPU_DATA_GET, 388
PER_CPU_DATA_GET_BY_OFFSET, 389
PER_CPU_DATA_ITEM, 389
PER_CPU_DATA_ITEM_DECLARE, 389
PER_CPU_DATA_OFFSET, 390
FO_F1_OFFSET
SPARC Context Structures, 842
Free
Thread_Information, 1506
Free-Running Counter and Busy Wait Delay, 391
rtems_counter_delay_nanoseconds, 392
rtems_counter_delay_ticks, 392
rtems_counter_difference, 392
rtems_counter_frequency, 393
rtems_counter_initialize_converter, 393
rtems_counter_nanoseconds_to_ticks, 393
rtems_counter_read, 395
rtems_counter_sbintime_to_ticks, 395
rtems_counter_ticks_to_nanoseconds, 395
rtems_counter_ticks_to_sbintime, 396
free_message_buffers
CORE_message_queue_Control, 1331
free_size
Heap_Statistics, 1352
Freechain Handler, 397
_Freechain_Extend, 398
_Freechain_Get, 398
_Freechain_Initialize, 399
_Freechain_Is_empty, 399
_Freechain_Pop, 399
_Freechain_Push, 400
_Freechain_Put, 400
Freechain_Control, 1345
fsr
Context_Control_fp, 1327
FSR_OFFSET
SPARC Context Structures, 842
function
rtems_assert_context, 1401

g1
CPU_Interrupt_frame, 1336
g2
CPU_Interrupt_frame, 1336
g3
CPU_Interrupt_frame, 1337

g4
CPU_Interrupt_frame, 1337
g5
Context_Control, 1318
CPU_Interrupt_frame, 1337
G5_OFFSET
SPARC Context Structures, 842
g7
Context_Control, 1319
CPU_Interrupt_frame, 1337
G7_OFFSET
SPARC Context Structures, 842
Gate
Thread_queue_Lock_context, 1515
General Scheduler Configuration, 402
CONFIGURE_CBS_MAXIMUM_SERVERS, 403
CONFIGURE_MAXIMUM_PRIORITY, 403
CONFIGURE_SCHEDULER_ASSIGNMENTS, 404
CONFIGURE_SCHEDULER_CBS, 404
CONFIGURE_SCHEDULER_EDF, 405
CONFIGURE_SCHEDULER_EDF_SMP, 405
CONFIGURE_SCHEDULER_NAME, 406
CONFIGURE_SCHEDULER_PRIORITY, 406
CONFIGURE_SCHEDULER_PRIORITY_AFFINITY_SMP, 407
CONFIGURE_SCHEDULER_PRIORITY_SMP, 407
CONFIGURE_SCHEDULER_SIMPLE, 408
CONFIGURE_SCHEDULER_SIMPLE_SMP, 408
CONFIGURE_SCHEDULER_STRONG_APA, 409
CONFIGURE_SCHEDULER_USER, 409
General System Configuration, 411
CONFIGURE_DIRTY_MEMORY, 412
CONFIGURE_DISABLE_NEWLIB_REENTRANCY, 412
CONFIGURE_EXECUTIVE_RAM_SIZE, 412
CONFIGURE_EXTRA_TASK_STACKS, 413
CONFIGURE_INITIAL_EXTENSIONS, 413
CONFIGURE_INTERRUPT_STACK_SIZE, 414
CONFIGURE_MALLOC_DIRTY, 414
CONFIGURE_MAXIMUM_FILE_DESCRIPTORS, 415
CONFIGURE_MAXIMUM_PROCESSORS, 415
CONFIGURE_MAXIMUM_THREAD_NAME_SIZE, 416
CONFIGURE_MEMORY_OVERHEAD, 416
CONFIGURE_MESSAGE_BUFFER_MEMORY, 417
CONFIGURE_MICROSECONDS_PER_TICK, 418
CONFIGURE_MINIMUM_TASK_STACK_SIZE, 419
CONFIGURE_STACK_CHECKER_ENABLED, 419
CONFIGURE_TICKS_PER_TIMESLICE, 420
CONFIGURE_UNIFIED_WORK_AREAS, 420
CONFIGURE_UNLIMITED_ALLOCATION_SIZE, 421
CONFIGURE_UNLIMITED_OBJECTS, 421
CONFIGURE_VERBOSE_SYSTEM_INITIALIZATION, 422
CONFIGURE_ZERO_WORKSPACE_AUTOMATICALLY, 422
Generic Checks, 423
getchark
bsplo.h, 1574
gptimer_regs, 1345
gptimer_timer_regs, 1346
grgpio_regs, 1346
GRLIB, 401

handler
ASR_Information, 1311
TOD_Hook, 1536
head
Per_CPU_Control, 1375
Header
Per_CPU_Control, 1376
Heads
_Thread_queue_Heads, 1306
heads
Thread_queue_Queue, 1519
Heap Handler, 424
_Heap_Align_down, 430
_Heap_Align_up, 430
_Heap_Alloc_area_of_block, 430
_Heap_Allocate, 431
_Heap_Allocate_aligned, 431
_Heap_Allocate_aligned_with_boundary, 432
_Heap_Area_overhead, 432
_Heap_Block_allocate, 433
_Heap_Block_at, 433
_Heap_Block_of_alloc_area, 434
_Heap_Block_set_size, 434
_Heap_Block_size, 435
_Heap_Extend, 435
_Heap_Free, 436
_Heap_Free_list_first, 436
_Heap_Free_list_head, 437
_Heap_Free_list_insert_after, 437
_Heap_Free_list_insert_before, 437
_Heap_Free_list_last, 438
_Heap_Free_list_remove, 438
_Heap_Free_list_replace, 438
_Heap_Free_list_tail, 439
_Heap_Get_first_and_last_block, 439
_Heap_Get_free_information, 440
_Heap_Get_information, 440
_Heap_Get_size, 440
_Heap_Greedy_allocate, 441
_Heap_Greedy_allocate_all_except_largest, 441
_Heap_Greedy_free, 442
_Heap_Initialize, 442
_Heap_Is_aligned, 443
_Heap_Is_block_in_heap, 443
_Heap_Is_free, 444
_Heap_Is_prev_used, 444

_Heap_Is_used, 444
_Heap_Iterate, 445
_Heap_Max, 445
_Heap_Min, 446
_Heap_Min_block_size, 446
_Heap_No_extend, 446
_Heap_Prev_block, 447
_Heap_Protection_set_delayed_free_fraction, 447
_Heap_Resize_block, 448
_Heap_Set_last_block_size, 448
_Heap_Size_of_alloc_area, 449
_Heap_Size_with_overhead, 449
_Heap_Walk, 450
Heap_Block_visitor, 428
HEAP_ERROR_BAD_FREE_BLOCK, 429
HEAP_ERROR_BAD_USED_BLOCK, 429
HEAP_ERROR_BROKEN_PROTECTOR, 429
HEAP_ERROR_DOUBLE_FREE, 429
HEAP_ERROR_FREE_PATTERN, 429
Heap_Error_reason, 429
Heap_Initialization_or_extend_handler, 429
Heap_Area, 1347
Heap_Block, 1347
next, 1348
prev, 1348
prev_size, 1348
size_and_flag, 1348
Heap_Block_visitor
Heap Handler, 428
Heap_Control, 1349
HEAP_ERROR_BAD_FREE_BLOCK
Heap Handler, 429
HEAP_ERROR_BAD_USED_BLOCK
Heap Handler, 429
HEAP_ERROR_BROKEN_PROTECTOR
Heap Handler, 429
Heap_Error_context, 1350
HEAP_ERROR_DOUBLE_FREE
Heap Handler, 429
HEAP_ERROR_FREE_PATTERN
Heap Handler, 429
Heap_Error_reason
Heap Handler, 429
Heap_Information, 1350
Heap_Information_block, 1351
Heap_Initialization_or_extend_handler
Heap Handler, 429
Heap_Statistics, 1351
free_size, 1352
lifetime_allocated, 1353
lifetime_freed, 1353
min_free_size, 1353
size, 1353
heir
Per_CPU_Control, 1376
Help_node
Thread_Scheduler_control, 1521
Helpers, 451
_Timespec_Add_to, 455
_Timespec_Divide, 455
_Timespec_Divide_by_integer, 455
_Timespec_Equal_to, 452
_Timespec_From_ticks, 456
_Timespec_Get_as_nanoseconds, 456
_Timespec_Get_nanoseconds, 452
_Timespec_Get_seconds, 453
_Timespec_Greater_than, 453
_Timespec_Is_valid, 457
_Timespec_Less_than, 457
_Timespec_Set, 454
_Timespec_Set_to_zero, 454
_Timespec_Subtract, 457
_Timespec_To_ticks, 458
helping_nodes
Thread_Scheduler_control, 1521

I/O Manager, 459
rtems_device_driver, 460
rtems_device_major_number, 460
rtems_device_minor_number, 460
rtems_io_close, 460
rtems_io_control, 461
rtems_io_initialize, 461
rtems_io_open, 462
rtems_io_read, 463
rtems_io_register_driver, 463
rtems_io_register_name, 464
rtems_io_unregister_driver, 465
rtems_io_write, 465
i0
Context_Control, 1319
CPU_Interrupt_frame, 1337
SPARC_Minimum_stack_frame, 1483
I0_OFFSET
SPARC Context Structures, 842
i1
Context_Control, 1319
CPU_Interrupt_frame, 1338
SPARC_Minimum_stack_frame, 1483
I1_OFFSET
SPARC Context Structures, 843
i2
Context_Control, 1319
CPU_Interrupt_frame, 1338
SPARC_Minimum_stack_frame, 1483
I2_OFFSET
SPARC Context Structures, 843
i3
Context_Control, 1319
CPU_Interrupt_frame, 1338
SPARC_Minimum_stack_frame, 1483
I3_OFFSET
SPARC Context Structures, 843
i4
Context_Control, 1320
CPU_Interrupt_frame, 1338
SPARC_Minimum_stack_frame, 1483

I4_OFFSET
SPARC Context Structures, 843
i5
Context_Control, 1320
CPU_Interrupt_frame, 1338
SPARC_Minimum_stack_frame, 1483
I5_OFFSET
SPARC Context Structures, 843
i6_fp
Context_Control, 1320
CPU_Interrupt_frame, 1339
SPARC_Minimum_stack_frame, 1484
I6_FP_OFFSET
SPARC Context Structures, 844
i7
Context_Control, 1320
CPU_Interrupt_frame, 1339
SPARC_Minimum_stack_frame, 1484
I7_OFFSET
SPARC Context Structures, 844
id
MP_packet_Prefix, 1361
Objects_Control, 1365
idle
Scheduler_Node, 1460
Idle Task Configuration, 491
CONFIGURE_IDLE_TASK_BODY, 491
CONFIGURE_IDLE_TASK_INITIALIZES_APPLICATION, 491
CONFIGURE_IDLE_TASK_STACK_SIZE, 492
Idle_threads
Scheduler_SMP_Context, 1472
Implementation, 493
Inactive
Objects_Information, 1368
inactive
Objects_Information, 1367
Inactive_messages
CORE_message_queue_Control, 1331
inactive_per_block
Objects_Information, 1368
init
rtems_test_parallel_job, 1443
initial
rtems_timer_information, 1445
Thread_Information, 1506
Timer_Control, 1531
initial_objects
Objects_Information, 1368
initial_priority
rtems_initialization_tasks_table, 1413
Thread_Start_information, 1524
Initial_stack
Thread_Start_information, 1524
Initialization and Shutdown, 494
rtems_initialize_executive, 494
rtems_shutdown_executive, 494
initialization_entry
rtems_driver_address_table, 1403
initialize
Scheduler_Operations, 1464
Install_tm27_vector
tm27.h, 1567
Internal Error Handler, 496
_Internal_error, 499
_Internal_errors_What_happened, 500
_Terminate, 499
INTERNAL_ERROR_CORE, 498
INTERNAL_ERROR_POSIX_API, 498
INTERNAL_ERROR_RTEMS_API, 498
Internal_errors_Core_list, 497
Internal_errors_Source, 497
RTEMS_FATAL_SOURCE_APPLICATION, 498
RTEMS_FATAL_SOURCE_ASSERT, 498
RTEMS_FATAL_SOURCE_BDBUF, 498
RTEMS_FATAL_SOURCE_BSP, 498
RTEMS_FATAL_SOURCE_EXCEPTION, 498
RTEMS_FATAL_SOURCE_EXIT, 498
RTEMS_FATAL_SOURCE_HEAP, 499
RTEMS_FATAL_SOURCE_INVALID_HEAP_FREE, 499
RTEMS_FATAL_SOURCE_LAST, 499
RTEMS_FATAL_SOURCE_PANIC, 498
RTEMS_FATAL_SOURCE_SMP, 498
RTEMS_FATAL_SOURCE_STACK_CHECKER, 498
internal_base
Dual_ported_memory_Control, 1343
INTERNAL_ERROR_CORE
Internal Error Handler, 498
INTERNAL_ERROR_POSIX_API
Internal Error Handler, 498
INTERNAL_ERROR_RTEMS_API
Internal Error Handler, 498
Internal_errors_Core_list
Internal Error Handler, 497
Internal_errors_Information, 1354
the_error, 1354
the_source, 1354
Internal_errors_Source
Internal Error Handler, 497
Interrupt Manager, 501
rtems_interrupt_catch, 508
rtems_interrupt_cause, 502
rtems_interrupt_clear, 502
rtems_interrupt_local_disable, 503
rtems_interrupt_local_enable, 503
rtems_interrupt_lock_acquire, 503
rtems_interrupt_lock_acquire_isr, 504
RTEMS_INTERRUPT_LOCK_DECLARE, 504
RTEMS_INTERRUPT_LOCK_DEFINE, 504
rtems_interrupt_lock_destroy, 505
rtems_interrupt_lock_initialize, 505
RTEMS_INTERRUPT_LOCK_INITIALIZER, 505
rtems_interrupt_lock_interrupt_disable, 506
RTEMS_INTERRUPT_LOCK_MEMBER, 506

RTEMS_INTERRUPT_LOCK_REFERENCE, 506
rtems_interrupt_lock_release, 507
rtems_interrupt_lock_release_isr, 507
Interrupt Manager Extension, 509
rtems_interrupt_get_affinity, 512
rtems_interrupt_handler_install, 513
rtems_interrupt_handler_iterate, 514
rtems_interrupt_handler_remove, 514
rtems_interrupt_per_handler_routine, 511
rtems_interrupt_server_action, 511
rtems_interrupt_server_action_prepend, 515
rtems_interrupt_server_control, 512
rtems_interrupt_server_create, 515
rtems_interrupt_server_delete, 516
rtems_interrupt_server_entry_destroy, 516
rtems_interrupt_server_entry_initialize, 517
rtems_interrupt_server_entry_move, 517
rtems_interrupt_server_entry_submit, 518
rtems_interrupt_server_handler_install, 518
rtems_interrupt_server_handler_iterate, 519
rtems_interrupt_server_handler_remove, 519
rtems_interrupt_server_initialize, 520
rtems_interrupt_server_move, 520
rtems_interrupt_server_request_destroy, 521
rtems_interrupt_server_request_initialize, 521
rtems_interrupt_server_request_set_vector, 522
rtems_interrupt_server_request_submit, 523
rtems_interrupt_server_resume, 523
rtems_interrupt_server_set_affinity, 524
rtems_interrupt_server_suspend, 524
rtems_interrupt_set_affinity, 525
InterruptConfig
spec:/rtems/event/req/send-receive, 1230
IO, 467
IO Library, 468
rtems_filesystem_default_pathconf, 473
rtems_filesystem_get_mount_handler, 471
rtems_filesystem_global_location_t, 470
rtems_filesystem_initialize, 471
rtems_filesystem_split_dev_t, 470
rtems_libio_iop_is_append, 471
rtems_libio_iop_is_no_delay, 472
rtems_libio_iop_is_readable, 472
rtems_libio_iop_is_writeable, 472
rtems_mkdir, 473
ioctl
rtems_termios_device_handler, 1436
irqmp_regs, 1355
irqmp_timestamp_regs, 1356
is_enabled
ASR_Information, 1311
is_fp
_Thread_Control, 1302
is_idle
_Thread_Control, 1302
is_preemptible
_Thread_Control, 1302
Thread_Start_information, 1524
is_set
TOD_Control, 1535
ISF_G1_OFFSET
SPARC, 829
ISF_G2_OFFSET
SPARC, 829
ISF_G3_OFFSET
SPARC, 829
ISF_G4_OFFSET
SPARC, 830
ISF_G5_OFFSET
SPARC, 830
ISF_G7_OFFSET
SPARC, 830
ISF_I0_OFFSET
SPARC, 830
ISF_I1_OFFSET
SPARC, 830
ISF_I2_OFFSET
SPARC, 831
ISF_I3_OFFSET
SPARC, 831
ISF_I4_OFFSET
SPARC, 831
ISF_I5_OFFSET
SPARC, 831
ISF_I6_FP_OFFSET
SPARC, 831
ISF_I7_OFFSET
SPARC, 832
ISF_NPC_OFFSET
SPARC, 832
ISF_PC_OFFSET
SPARC, 832
ISF_PSR_OFFSET
SPARC, 832
ISF_TPC_OFFSET
SPARC, 832
ISF_Y_OFFSET
SPARC, 833
ISR Handler, 474
_ISR_Get_level, 475
_ISR_Handler, 479
_ISR_Handler_initialization, 479
_ISR_Install_vector, 475
_ISR_Is_enabled, 476
_ISR_Is_in_progress, 480
_ISR_Local_disable, 476
_ISR_Local_enable, 477
_ISR_Local_flash, 477
_ISR_Set_level, 478
_ISR_Stack_area_begin, 480
_ISR_Stack_area_end, 481
ISR_Handler, 478
ISR_Level, 479
ISR_Vector_number, 479
RTEMS_DECLARE_GLOBAL_SYMBOL, 480
ISR Locks, 482

_ISR_lock_Acquire, 483
_ISR_lock_Acquire_inline, 483
_ISR_lock_Context_set_level, 490
_ISR_lock_Destroy, 484
_ISR_lock_ISR_disable, 485
_ISR_lock_ISR_disable_and_acquire, 485
_ISR_lock_ISR_enable, 486
_ISR_lock_Initialize, 484
_ISR_lock_Release, 486
_ISR_lock_Release_and_ISR_enable, 487
_ISR_lock_Release_inline, 487
_ISR_lock_Set_name, 488
ISR_LOCK_DECLARE, 488
ISR_LOCK_DEFINE, 488
ISR_LOCK_INITIALIZER, 489
ISR_LOCK_MEMBER, 489
ISR_LOCK_REFERENCE, 489
isr_dispatch_disable
Context_Control, 1320
Per_CPU_Control, 1376
ISR_DISPATCH_DISABLE_STACK_OFFSET
SPARC Context Structures, 844
ISR_Handler
ISR Handler, 478
ISR_Level
ISR Handler, 479
isr_level
Thread_Start_information, 1525
ISR_lock_Context, 1356
ISR_lock_Control, 1356
ISR_LOCK_DECLARE
ISR Locks, 488
ISR_LOCK_DEFINE
ISR Locks, 488
ISR_LOCK_INITIALIZER
ISR Locks, 489
ISR_LOCK_MEMBER
ISR Locks, 489
ISR_LOCK_REFERENCE
ISR Locks, 489
isr_nest_level
Per_CPU_Control, 1377
ISR_Vector_number
ISR Handler, 479

jmp_to_low_of_handler_plus_l4
CPU_Trap_table_entry, 1341
Jobs
Per_CPU_Control, 1377
Join_queue
_Thread_Control, 1303
Thread_Proxy_control, 1509

Kernel Print Support, 526

l0
SPARC_Minimum_stack_frame, 1484
l0_and_l1
Context_Control, 1321
L0_OFFSET
SPARC Context Structures, 844
l1
SPARC_Minimum_stack_frame, 1484
L1_OFFSET
SPARC Context Structures, 844
l2
Context_Control, 1321
SPARC_Minimum_stack_frame, 1484
L2_OFFSET
SPARC Context Structures, 845
l2c_regs, 1357
l3
Context_Control, 1321
SPARC_Minimum_stack_frame, 1485
L3_OFFSET
SPARC Context Structures, 845
l4
Context_Control, 1321
SPARC_Minimum_stack_frame, 1485
L4_OFFSET
SPARC Context Structures, 845
l5
Context_Control, 1321
SPARC_Minimum_stack_frame, 1485
L5_OFFSET
SPARC Context Structures, 845
l6
Context_Control, 1322
SPARC_Minimum_stack_frame, 1485
L6_OFFSET
SPARC Context Structures, 845
l7
Context_Control, 1322
SPARC_Minimum_stack_frame, 1485
L7_OFFSET
SPARC Context Structures, 846
last_close
rtems_termios_device_handler, 1436
latest_deadline
Rate_monotonic_Control, 1390
Legacy Benchmark Drivers, 530
length
Dual_ported_memory_Control, 1343
MP_packet_Prefix, 1361
leon.h
LEON_Clear_interrupt, 1564
LEON_Cpu_Disable_interrupt, 1564
LEON_Cpu_Mask_interrupt, 1565
LEON_Cpu_Restore_interrupt, 1565
LEON_Cpu_Unmask_interrupt, 1565
LEON_Disable_interrupt_broadcast, 1566
LEON_Enable_interrupt_broadcast, 1566
LEON_Force_interrupt, 1566
LEON3 AMBA Driver Handler, 527
LEON3 and LEON4, 528
LEON_Clear_interrupt
leon.h, 1564

LEON_Cpu_Disable_interrupt
leon.h, 1564
LEON_Cpu_Mask_interrupt
leon.h, 1565
LEON_Cpu_Restore_interrupt
leon.h, 1565
LEON_Cpu_Unmask_interrupt
leon.h, 1565
LEON_Disable_interrupt_broadcast
leon.h, 1566
LEON_Enable_interrupt_broadcast
leon.h, 1566
LEON_Force_interrupt
leon.h, 1566
libc_reent
_Thread_Control, 1303
libfs, 1223
Life
_Thread_Control, 1303
lifetime_allocated
Heap_Statistics, 1353
lifetime_freed
Heap_Statistics, 1353
line
rtems_assert_context, 1401
load_context, 1358
Local Packages, 531
local_table
Objects_Information, 1368
Lock
Per_CPU_Control, 1377
Thread_queue_Queue, 1519
Thread_Wait_information, 1527
Lock_context
Per_CPU_Control, 1377
locked
SMP_MCS_lock_Context, 1478

Malloc Support, 534
malloc.h
rtems_calloc, 1593
rtems_heap_allocate_aligned_with_boundary, 1594
rtems_heap_extend, 1594
rtems_heap_greedy_allocate, 1595
rtems_heap_greedy_allocate_all_except_largest, 1595
rtems_heap_greedy_free, 1595
rtems_malloc, 1595
rtems_malloc_dirty_memory, 1596
RTEMS_Malloc_Heap, 1596
rtems_memalign, 1596
malloc_free_space
Standard C Library Support, 940
malloc_get_heap_pointer
Standard C Library Support, 941
malloc_info
Standard C Library Support, 941
malloc_set_heap_pointer
Standard C Library Support, 941
mallocheap.c
RTEMS_Malloc_Heap, 1790
map_priority
Scheduler_Operations, 1464
max_cpu_time
Rate_monotonic_Statistics, 1392
rtems_rate_monotonic_period_statistics, 1425
max_wall_time
Rate_monotonic_Statistics, 1393
rtems_rate_monotonic_period_statistics, 1425
maximum
rtems_object_api_class_information, 1422
maximum_barriers
rtems_api_configuration_table, 1396
maximum_count
CORE_barrier_Attributes, 1327
maximum_id
Objects_Information, 1369
rtems_object_api_class_information, 1422
maximum_message_queues
rtems_api_configuration_table, 1396
maximum_message_size
CORE_message_queue_Control, 1331
maximum_partitions
rtems_api_configuration_table, 1396
maximum_pending_messages
CORE_message_queue_Control, 1332
maximum_periods
rtems_api_configuration_table, 1396
maximum_ports
rtems_api_configuration_table, 1397
maximum_priority
_Scheduler_Control, 1299
maximum_regions
rtems_api_configuration_table, 1397
maximum_semaphores
rtems_api_configuration_table, 1397
maximum_tasks
rtems_api_configuration_table, 1397
maximum_thread_local_storage_size
rtems_task_config, 1431
maximum_timers
rtems_api_configuration_table, 1398
mctrl_regs, 1358
Memory Area Checks, 535
Memory Handler, 536
_Memory_Allocate, 538
_Memory_Consume, 538
_Memory_Fill, 539
_Memory_Get, 539
_Memory_Get_area, 539
_Memory_Get_begin, 540
_Memory_Get_count, 540
_Memory_Get_end, 540
_Memory_Get_free_begin, 541
_Memory_Get_free_size, 541
_Memory_Get_size, 542

_Memory_Initialize, 542
_Memory_Initialize_by_size, 542
_Memory_Set_begin, 543
_Memory_Set_end, 543
_Memory_Set_free_begin, 543
_Memory_Zero_before_use, 544
MEMORY_INFORMATION_INITIALIZER, 537
MEMORY_INITIALIZER, 537
Memory_Area, 1358
Memory_Information, 1359
MEMORY_INFORMATION_INITIALIZER
Memory Handler, 537
MEMORY_INITIALIZER
Memory Handler, 537
message
Per_CPU_Control, 1378
Message Manager, 545
rtems_message_queue_broadcast, 546
RTEMS_MESSAGE_QUEUE_BUFFER, 546
rtems_message_queue_construct, 546
rtems_message_queue_create, 547
rtems_message_queue_delete, 548
rtems_message_queue_flush, 548
rtems_message_queue_get_number_pending, 549
rtems_message_queue_ident, 549
rtems_message_queue_receive, 550
rtems_message_queue_send, 550
rtems_message_queue_urgent, 551
Message Queue Handler, 552
_CORE_message_queue_Acquire, 557
_CORE_message_queue_Acquire_critical, 557
_CORE_message_queue_Allocate_message_buffer, 557
_CORE_message_queue_Broadcast, 558
_CORE_message_queue_Close, 559
_CORE_message_queue_Copy_buffer, 559
_CORE_message_queue_Dequeue_receiver, 560
_CORE_message_queue_Flush, 560
_CORE_message_queue_Free_message_buffer, 561
_CORE_message_queue_Get_message_priority, 561
_CORE_message_queue_Get_pending_message, 562
_CORE_message_queue_Initialize, 562
_CORE_message_queue_Insert_message, 563
_CORE_message_queue_Release, 563
_CORE_message_queue_Seize, 564
_CORE_message_queue_Send, 564
_CORE_message_queue_Set_notify, 554
_CORE_message_queue_Submit, 565
_CORE_message_queue_Urgent, 566
_CORE_message_queue_Workspace_allocate, 567
CORE_message_queue_Allocate_buffers, 555
CORE_message_queue_Disciplines, 556
CORE_MESSAGE_QUEUE_DISCIPLINES_FIFO, 557
CORE_MESSAGE_QUEUE_DISCIPLINES_PRIORITY, 557
CORE_MESSAGE_QUEUE_SEND_REQUEST, 555
CORE_message_queue_Submit_types, 556
CORE_MESSAGE_QUEUE_URGENT_REQUEST, 555
RTEMS_SCORE_COREMSG_ENABLE_BLOCKING_SEND, 555
RTEMS_SCORE_COREMSG_ENABLE_MESSAGE_PRIORITY, 555
message_buffers
CORE_message_queue_Control, 1332
message_queue
Message_queue_Control, 1360
Message_queue_Control, 1359
message_queue, 1360
Object, 1360
MESSAGE_QUEUE_INFORMATION_DEFINE
Classic Message Queue Implementation, 224
MESSAGE_QUEUE_SEND_REQUEST
Classic Message Queue Implementation, 224
Message_queue_Submit_types
Classic Message Queue Implementation, 224
MESSAGE_QUEUE_URGENT_REQUEST
Classic Message Queue Implementation, 224
min_cpu_time
Rate_monotonic_Statistics, 1393
rtems_rate_monotonic_period_statistics, 1426
min_free_size
Heap_Statistics, 1353
min_wall_time
Rate_monotonic_Statistics, 1393
rtems_rate_monotonic_period_statistics, 1426
minimum_id
rtems_object_api_class_information, 1423
minor
Priority_bit_map_Information, 1388
missed_count
Rate_monotonic_Statistics, 1393
rtems_rate_monotonic_period_statistics, 1426
mode_set
ASR_Information, 1311
rtems_initialization_tasks_table, 1413
mount
File System Types and Mount, 370
mount_and_make_target_path
File System Types and Mount, 371
mov_psr_l0
CPU_Trap_table_entry, 1341
mov_vector_l3
CPU_Trap_table_entry, 1342
MP Packet Handler, 532
MP_packet_Classes, 533
MP_PACKET_CLASSES_FIRST, 532
MP_PACKET_CLASSES_LAST, 532

MP_PACKET_MINIMUM_PACKET_SIZE, 533
MP_PACKET_MINIMUN_HETERO_CONVERSION, 533
MP_packet_Classes
MP Packet Handler, 533
MP_PACKET_CLASSES_FIRST
MP Packet Handler, 532
MP_PACKET_CLASSES_LAST
MP Packet Handler, 532
MP_PACKET_MINIMUM_PACKET_SIZE
MP Packet Handler, 533
MP_PACKET_MINIMUN_HETERO_CONVERSION
MP Packet Handler, 533
MP_packet_Prefix, 1360
id, 1361
length, 1361
return_code, 1361
source_priority, 1362
source_tid, 1362
the_class, 1362
timeout, 1362
to_convert, 1362
MRSP_Control, 1363
Multiprocessing Configuration, 568
CONFIGURE_EXTRA_MPCI_RECEIVE_SERVER_STACK, 568
CONFIGURE_MP_APPLICATION, 568
CONFIGURE_MP_MAXIMUM_GLOBAL_OBJECTS, 569
CONFIGURE_MP_MAXIMUM_NODES, 569
CONFIGURE_MP_MAXIMUM_PROXIES, 570
CONFIGURE_MP_MPCI_TABLE_POINTER, 570
CONFIGURE_MP_NODE_NUMBER, 571
Multiprocessor Resource Sharing Protocol Handler, 572
_MRSP_Acquire_critical, 573
_MRSP_Can_destroy, 574
_MRSP_Claim_ownership, 574
_MRSP_Destroy, 574
_MRSP_Get_owner, 575
_MRSP_Get_priority, 575
_MRSP_Initialize, 576
_MRSP_Raise_priority, 576
_MRSP_Release, 577
_MRSP_Remove_priority, 577
_MRSP_Replace_priority, 577
_MRSP_Seize, 578
_MRSP_Set_owner, 578
_MRSP_Set_priority, 579
_MRSP_Surrender, 579
_MRSP_Wait_for_ownership, 580
Mutex
API_Mutex_Control, 1310
Semaphore_Control, 1474
Mutex Handler, 581
_CORE_ceiling_mutex_Get_priority, 582
_CORE_ceiling_mutex_Get_scheduler, 583
_CORE_ceiling_mutex_Initialize, 583
_CORE_ceiling_mutex_Seize, 584
_CORE_ceiling_mutex_Set_owner, 584
_CORE_ceiling_mutex_Set_priority, 585
_CORE_ceiling_mutex_Surrender, 585
_CORE_mutex_Acquire_critical, 586
_CORE_mutex_Destroy, 586
_CORE_mutex_Get_owner, 586
_CORE_mutex_Initialize, 587
_CORE_mutex_Is_locked, 587
_CORE_mutex_Is_owner, 587
_CORE_mutex_Release, 588
_CORE_mutex_Seize_slow, 588
_CORE_mutex_Set_owner, 589
_CORE_recursive_mutex_Initialize, 589
_CORE_recursive_mutex_Seize, 589
_CORE_recursive_mutex_Seize_nested, 590
_CORE_recursive_mutex_Surrender, 590
Mutex_Control, 1363
Mutex_recursive_Control, 1364

name
Objects_Control, 1365
rtems_initialization_tasks_table, 1413
name_length
Objects_Information, 1369
name_p
Objects_Name, 1370
name_u32
Objects_Name, 1371
nest_level
ASR_Information, 1312
next
Chain_Node_struct, 1317
Heap_Block, 1348
Scheduler_Node, 1460
next_length
Rate_monotonic_Control, 1390
nfs Client, 1224
Node
Objects_Control, 1365
Priority_Aggregation, 1386
Scheduler_EDF_Node, 1456
Scheduler_Node, 1460
Thread_queue_Priority_queue, 1518
TOD_Hook, 1536
node_destroy
Scheduler_Operations, 1464
node_initialize
Scheduler_Operations, 1464
nodes
Thread_Scheduler_control, 1521
nop
sparc.h, 1874
normal
SMP_MCS_lock_Context, 1478
SMP_MCS_lock_Control, 1479
npc
CPU_Interrupt_frame, 1339
ntp_fp, 1364
ntptimeval, 1364

number_of_initialization_tasks
rtems_api_configuration_table, 1398
number_of_pending_messages
CORE_message_queue_Control, 1332
number_of_waiting_threads
CORE_barrier_Control, 1328

o6_sp
Context_Control, 1322
O6_SP_OFFSET
SPARC Context Structures, 846
o7
Context_Control, 1322
O7_OFFSET
SPARC Context Structures, 846
Object
_Thread_Control, 1304
Barrier_Control, 1313
Dual_ported_memory_Control, 1343
Message_queue_Control, 1360
Rate_monotonic_Control, 1390
Semaphore_Control, 1475
Thread_Proxy_control, 1509
Timer_Control, 1531
Object Handler, 592
_Objects_API_maximum_class, 612
_Objects_Activate_unlimited, 607
_Objects_Active_count, 608
_Objects_Allocate, 608
_Objects_Allocate_none, 609
_Objects_Allocate_static, 609
_Objects_Allocate_unlimited, 610
_Objects_Allocate_unprotected, 610
_Objects_Allocate_with_extend, 611
_Objects_Allocator_is_owner, 611
_Objects_Allocator_lock, 611
_Objects_Allocator_unlock, 612
_Objects_Are_ids_equal, 613
_Objects_Build_id, 596
_Objects_Build_name, 597
_Objects_Close, 613
_Objects_Extend_information, 613
_Objects_Extend_size, 614
_Objects_Free, 614
_Objects_Free_objects_block, 615
_Objects_Free_static, 615
_Objects_Free_unlimited, 616
_Objects_Get, 616
_Objects_Get_API, 617
_Objects_Get_by_name, 617
_Objects_Get_class, 618
_Objects_Get_inactive, 618
_Objects_Get_index, 618
_Objects_Get_information, 619
_Objects_Get_information_id, 619
_Objects_Get_maximum_index, 620
_Objects_Get_minimum_id, 620
_Objects_Get_name_as_string, 620
_Objects_Get_next, 621
_Objects_Get_no_protection, 621
_Objects_Get_node, 622
_Objects_Has_string_name, 622
_Objects_Id_to_name, 624
_Objects_Information_table, 631
_Objects_Initialize_information, 624
_Objects_Invalidate_Id, 625
_Objects_Is_api_valid, 625
_Objects_Is_auto_extend, 626
_Objects_Is_local_id, 626
_Objects_Is_local_node, 626
_Objects_Is_unlimited, 598
_Objects_Maximum_nodes, 598
_Objects_Name_to_id_u32, 627
_Objects_Name_to_string, 628
_Objects_Namespace_remove_string, 628
_Objects_Namespace_remove_u32, 628
_Objects_Open, 629
_Objects_Open_string, 629
_Objects_Open_u32, 630
_Objects_Set_local_object, 630
_Objects_Set_name, 630
_Objects_Shrink_information, 631
OBJECTS_API_MASK, 598
OBJECTS_API_START_BIT, 598
OBJECTS_API_VALID_BITS, 598
Objects_APIs, 606
OBJECTS_APIS_LAST, 599
OBJECTS_CLASS_MASK, 599
OBJECTS_CLASS_START_BIT, 599
OBJECTS_CLASS_VALID_BITS, 599
Objects_Classic_API, 606
Objects_Id, 606
OBJECTS_ID_FINAL, 599
OBJECTS_ID_FINAL_INDEX, 600
OBJECTS_ID_INITIAL, 600
OBJECTS_ID_INITIAL_INDEX, 600
OBJECTS_ID_NONE, 600
OBJECTS_ID_OF_SELF, 600
OBJECTS_INDEX_MASK, 601
OBJECTS_INDEX_MINIMUM, 601
OBJECTS_INDEX_START_BIT, 601
OBJECTS_INDEX_VALID_BITS, 601
OBJECTS_INFORMATION_DEFINE, 601
OBJECTS_INFORMATION_DEFINE_ZERO, 602
Objects_Internal_API, 607
OBJECTS_INTERNAL_CLASSES_LAST, 603
Objects_Maximum, 606
Objects_Name_comparators, 606
OBJECTS_NAME_ERRORS_FIRST, 603
OBJECTS_NAME_ERRORS_LAST, 603
Objects_Name_or_id_lookup_errors, 607
OBJECTS_NODE_MASK, 603
OBJECTS_NODE_START_BIT, 604
OBJECTS_NODE_VALID_BITS, 604
Objects_POSIX_API, 607
OBJECTS_POSIX_CLASSES_LAST, 604
OBJECTS_RTEMS_CLASSES_LAST, 604

OBJECTS_SEARCH_ALL_NODES, 604
OBJECTS_SEARCH_LOCAL_NODE, 605
OBJECTS_SEARCH_OTHER_NODES, 605
OBJECTS_UNLIMITED_OBJECTS, 605
OBJECTS_WHO_AM_I, 605
Object Services, 632
rtems_build_id, 633
rtems_build_name, 634
rtems_object_api_maximum_class, 637
rtems_object_api_minimum_class, 637
rtems_object_get_api_class_name, 637
rtems_object_get_api_name, 638
rtems_object_get_class_information, 638
rtems_object_get_classic_name, 638
rtems_object_get_name, 639
rtems_object_id_api_maximum_class, 639
rtems_object_id_get_api, 634
rtems_object_id_get_class, 634
rtems_object_id_get_index, 635
rtems_object_id_get_node, 635
RTEMS_OBJECT_ID_INITIAL, 635
rtems_object_set_name, 639
object_blocks
Objects_Information, 1369
object_size
Objects_Information, 1369
OBJECTS_API_MASK
Object Handler, 598
OBJECTS_API_START_BIT
Object Handler, 598
OBJECTS_API_VALID_BITS
Object Handler, 598
Objects_APIs
Object Handler, 606
OBJECTS_APIS_LAST
Object Handler, 599
OBJECTS_CLASS_MASK
Object Handler, 599
OBJECTS_CLASS_START_BIT
Object Handler, 599
OBJECTS_CLASS_VALID_BITS
Object Handler, 599
Objects_Classic_API
Object Handler, 606
Objects_Control, 1365
id, 1365
name, 1365
Node, 1365
Objects_Id
Object Handler, 606
OBJECTS_ID_FINAL
Object Handler, 599
OBJECTS_ID_FINAL_INDEX
Object Handler, 600
OBJECTS_ID_INITIAL
Object Handler, 600
OBJECTS_ID_INITIAL_INDEX
Object Handler, 600
OBJECTS_ID_NONE
Object Handler, 600
OBJECTS_ID_OF_SELF
Object Handler, 600
OBJECTS_INDEX_MASK
Object Handler, 601
OBJECTS_INDEX_MINIMUM
Object Handler, 601
OBJECTS_INDEX_START_BIT
Object Handler, 601
OBJECTS_INDEX_VALID_BITS
Object Handler, 601
Objects_Information, 1366
allocate, 1367
deallocate, 1367
Inactive, 1368
inactive, 1367
inactive_per_block, 1368
initial_objects, 1368
local_table, 1368
maximum_id, 1369
name_length, 1369
object_blocks, 1369
object_size, 1369
objects_per_block, 1370
OBJECTS_INFORMATION_DEFINE
Object Handler, 601
OBJECTS_INFORMATION_DEFINE_ZERO
Object Handler, 602
Objects_Internal_API
Object Handler, 607
OBJECTS_INTERNAL_CLASSES_LAST
Object Handler, 603
Objects_Maximum
Object Handler, 606
Objects_Name, 1370
name_p, 1370
name_u32, 1371
Objects_Name_comparators
Object Handler, 606
OBJECTS_NAME_ERRORS_FIRST
Object Handler, 603
OBJECTS_NAME_ERRORS_LAST
Object Handler, 603
Objects_Name_or_id_lookup_errors
Object Handler, 607
OBJECTS_NODE_MASK
Object Handler, 603
OBJECTS_NODE_START_BIT
Object Handler, 604
OBJECTS_NODE_VALID_BITS
Object Handler, 604
objects_per_block
Objects_Information, 1370
Objects_POSIX_API
Object Handler, 607
OBJECTS_POSIX_CLASSES_LAST
Object Handler, 604

OBJECTS_RTEMS_CLASSES_LAST
Object Handler, 604
OBJECTS_SEARCH_ALL_NODES
Object Handler, 604
OBJECTS_SEARCH_LOCAL_NODE
Object Handler, 605
OBJECTS_SEARCH_OTHER_NODES
Object Handler, 605
OBJECTS_UNLIMITED_OBJECTS
Object Handler, 605
OBJECTS_WHO_AM_I
Object Handler, 605
open_entry
rtems_driver_address_table, 1403
operations
Thread_Wait_information, 1527
option
Thread_Wait_information, 1527
owner
Rate_monotonic_Control, 1390
rtems_rate_monotonic_period_status, 1428

pad0
SPARC_Minimum_stack_frame, 1486
Partition Manager, 650
RTEMS_PARTITION_ALIGNMENT, 650
rtems_partition_create, 650
rtems_partition_delete, 652
rtems_partition_get_buffer, 653
rtems_partition_ident, 653
rtems_partition_return_buffer, 654
Partition Manager Implementation, 656
_Partition_Acquire_critical, 657
_Partition_Get, 657
_Partition_Information, 658
_Partition_Release, 658
PARTITION_INFORMATION_DEFINE, 656
Partition_Control, 1371
PARTITION_INFORMATION_DEFINE
Partition Manager Implementation, 656
Path
Thread_queue_Context, 1512
path
rtems_filesystem_eval_path_context_t, 1405
pathlen
rtems_filesystem_eval_path_context_t, 1406
pc
CPU_Interrupt_frame, 1339
Pending_messages
CORE_message_queue_Control, 1332
Per_CPU_Control, 1372
ancestor, 1373
context, 1374
control, 1374
cpu_usage_timestamp, 1374
data, 1374
dispatch_necessary, 1375
executing, 1375
head, 1375
Header, 1376
heir, 1376
isr_dispatch_disable, 1376
isr_nest_level, 1377
Jobs, 1377
Lock, 1377
Lock_context, 1377
message, 1378
RTEMS Per CPU Information, 711
state, 1378
tail, 1378
Threads_in_need_for_help, 1378
ticks, 1379
Per_CPU_Control_envelope, 1379
PER_CPU_DATA_GET
Flexible Per-CPU Data, 388
PER_CPU_DATA_GET_BY_OFFSET
Flexible Per-CPU Data, 389
PER_CPU_DATA_ITEM
Flexible Per-CPU Data, 389
PER_CPU_DATA_ITEM_DECLARE
Flexible Per-CPU Data, 389
PER_CPU_DATA_OFFSET
Flexible Per-CPU Data, 390
Per_CPU_Job, 1379
done, 1380
RTEMS Per CPU Information, 712
Per_CPU_Job_context, 1380
Per_CPU_State
RTEMS Per CPU Information, 712
PER_CPU_STATE_INITIAL
RTEMS Per CPU Information, 713
PER_CPU_STATE_READY_TO_START_MULTITASKING
RTEMS Per CPU Information, 713
PER_CPU_STATE_REQUEST_START_MULTITASKING
RTEMS Per CPU Information, 714
PER_CPU_STATE_SHUTDOWN
RTEMS Per CPU Information, 714
PER_CPU_STATE_UP
RTEMS Per CPU Information, 714
Per_CPU_Stats, 1381
PER_CPU_WATCHDOG_COUNT
RTEMS Per CPU Information, 714
Per_CPU_Watchdog_index
RTEMS Per CPU Information, 714
PER_CPU_WATCHDOG_MONOTONIC
RTEMS Per CPU Information, 714
PER_CPU_WATCHDOG_REALTIME
RTEMS Per CPU Information, 714
PER_CPU_WATCHDOG_TICKS
RTEMS Per CPU Information, 714
pin
Scheduler_Operations, 1465
pin_level
Thread_Scheduler_control, 1521
Pointer Checks, 659
poll_read
rtems_termios_device_handler, 1437

position
Chain_Iterator, 1315
POSIX API Configuration, 640
CONFIGURE_MAXIMUM_POSIX_KEY_VALUE_PAIRS, 640
CONFIGURE_MAXIMUM_POSIX_KEYS, 641
CONFIGURE_MAXIMUM_POSIX_MESSAGE_QUEUES, 641
CONFIGURE_MAXIMUM_POSIX_QUEUED_SIGNALS, 642
CONFIGURE_MAXIMUM_POSIX_SEMAPHORES, 643
CONFIGURE_MAXIMUM_POSIX_SHMS, 643
CONFIGURE_MAXIMUM_POSIX_THREADS, 644
CONFIGURE_MAXIMUM_POSIX_TIMERS, 645
CONFIGURE_MINIMUM_POSIX_THREAD_STACK_SIZE, 645
POSIX Initialization Thread Configuration, 647
CONFIGURE_POSIX_INIT_THREAD_ENTRY_POINT, 647
CONFIGURE_POSIX_INIT_THREAD_STACK_SIZE, 647
CONFIGURE_POSIX_INIT_THREAD_TABLE, 648
POSIX Status and Error Number Checks, 649
POSIX_Spinlock_Control, 1381
postponed_jobs
Rate_monotonic_Control, 1391
postponed_jobs_count
rtems_rate_monotonic_period_status, 1428
pps_fetch_args, 1382
pps_fetch_ffc_args, 1382
pps_info_ffc_t, 1383
pps_info_t, 1383
pps_kcbind_args, 1383
pps_params_t, 1384
pps_timeu, 1384
prev
Heap_Block, 1348
prev_size
Heap_Block, 1348
previous
Chain_Node_struct, 1317
Print Support, 660
print.h
rtems_printf, 1618
rtems_vprintf, 1618
print_printf.c
rtems_vprintf, 1791
printk
bspIo.h, 1574
printk.c, 1791
printk.c
printk, 1791
Priority
Scheduler_Node, 1460
Priority Handler, 661
_Bitfield_Find_first_bit, 665
_Bitfield_Leading_zeros, 684
_Priority_Actions_add, 665
_Priority_Actions_initialize_empty, 666
_Priority_Actions_initialize_one, 666
_Priority_Actions_is_empty, 667
_Priority_Actions_is_valid, 667
_Priority_Actions_move, 667
_Priority_Bits_index, 671
_Priority_Change_nothing, 671
_Priority_Changed, 672
_Priority_Extract, 672
_Priority_Extract_non_empty, 674
_Priority_Get_minimum_node, 674
_Priority_Get_next_action, 675
_Priority_Get_priority, 675
_Priority_Get_scheduler, 675
_Priority_Initialize_empty, 676
_Priority_Initialize_one, 676
_Priority_Insert, 677
_Priority_Is_empty, 677
_Priority_Less, 677
_Priority_Major, 678
_Priority_Mask, 678
_Priority_Minor, 679
_Priority_Node_initialize, 679
_Priority_Node_is_active, 679
_Priority_Node_set_inactive, 680
_Priority_Node_set_priority, 680
_Priority_Non_empty_insert, 680
_Priority_Plain_changed, 681
_Priority_Plain_extract, 681
_Priority_Plain_insert, 682
_Priority_Remove_nothing, 682
_Priority_Replace, 683
_Priority_Set_action, 683
_Priority_Set_action_node, 683
_Priority_Set_action_type, 684
_Priority_bit_map_Add, 669
_Priority_bit_map_Get_highest, 669
_Priority_bit_map_Initialize, 669
_Priority_bit_map_Initialize_information, 670
_Priority_bit_map_Is_empty, 670
_Priority_bit_map_Remove, 671
Priority_Control, 664
PRIORITY_DEFAULT_MAXIMUM, 664
PRIORITY_PSEUDO_ISR, 664
Priority_Actions, 1384
Priority_Aggregation, 1385
Node, 1386
Priority_bit_map_Control, 1386
bit_map, 1386
Priority_bit_map_Information, 1387
block_major, 1387
block_minor, 1387
minor, 1388
ready_major, 1388
ready_minor, 1388
Priority_Control
Priority Handler, 664

PRIORITY_DEFAULT_MAXIMUM
Priority Handler, 664
Priority_map
Scheduler_priority_Ready_queue, 1470
Priority_Node, 1388
PRIORITY_PSEUDO_ISR
Priority Handler, 664
Processor Mask, 685
_Processor_mask_And, 686
_Processor_mask_Assign, 687
_Processor_mask_Clear, 687
_Processor_mask_Copy, 688
_Processor_mask_Count, 688
_Processor_mask_Fill, 689
_Processor_mask_Find_last_set, 689
_Processor_mask_From_cpu_set_t, 689
_Processor_mask_From_index, 690
_Processor_mask_From_uint32_t, 690
_Processor_mask_Has_overlap, 690
_Processor_mask_Is_at_most_partial_loss, 691
_Processor_mask_Is_equal, 691
_Processor_mask_Is_set, 692
_Processor_mask_Is_subset, 692
_Processor_mask_Is_zero, 693
_Processor_mask_Nand, 693
_Processor_mask_Or, 693
_Processor_mask_Set, 694
_Processor_mask_To_cpu_set_t, 694
_Processor_mask_To_uint32_t, 695
_Processor_mask_Xor, 695
_Processor_mask_Zero, 696
Processors
Scheduler_Context, 1454
Profiling Support, 697
_Profiling_Outer_most_interrupt_entry_and_exit, 697
_Profiling_Thread_dispatch_disable, 698
_Profiling_Thread_dispatch_disable_critical, 698
_Profiling_Thread_dispatch_enable, 698
_Profiling_Update_max_interrupt_delay, 699
Protected Heap Handler, 700
_Protected_heap_Allocate, 701
_Protected_heap_Allocate_aligned, 701
_Protected_heap_Allocate_aligned_with_boundary, 702
_Protected_heap_Extend, 702
_Protected_heap_Free, 703
_Protected_heap_Get_block_size, 703
_Protected_heap_Get_free_information, 704
_Protected_heap_Get_information, 704
_Protected_heap_Get_size, 704
_Protected_heap_Initialize, 705
_Protected_heap_Iterate, 705
_Protected_heap_Resize_block, 706
_Protected_heap_Walk, 706
psr
Context_Control, 1322
CPU_Interrupt_frame, 1339
PSR_OFFSET
SPARC Context Structures, 846
putk
bsplo.h, 1575

queue
SMP_MCS_lock_Control, 1479
Thread_Wait_information, 1528

Rate-Monotonic Manager, 748
RATE_MONOTONIC_ACTIVE, 749
RATE_MONOTONIC_EXPIRED, 749
RATE_MONOTONIC_INACTIVE, 749
rtems_rate_monotonic_cancel, 749
rtems_rate_monotonic_create, 749
rtems_rate_monotonic_delete, 750
rtems_rate_monotonic_get_statistics, 750
rtems_rate_monotonic_get_status, 750
rtems_rate_monotonic_ident, 751
rtems_rate_monotonic_period, 751
rtems_rate_monotonic_period_states, 749
rtems_rate_monotonic_report_statistics_with_plugin, 752
rtems_rate_monotonic_reset_statistics, 752
RATE_MONOTONIC_ACTIVE
Rate-Monotonic Manager, 749
Rate_monotonic_Control, 1389
cpu_usage_period_initiated, 1390
latest_deadline, 1390
next_length, 1390
Object, 1390
owner, 1390
postponed_jobs, 1391
state, 1391
Statistics, 1391
time_period_initiated, 1391
Timer, 1391
RATE_MONOTONIC_EXPIRED
Rate-Monotonic Manager, 749
RATE_MONOTONIC_INACTIVE
Rate-Monotonic Manager, 749
RATE_MONOTONIC_INFORMATION_DEFINE
Classic Rate Monotonic Scheduler Implementation, 232
Rate_monotonic_Statistics, 1392
count, 1392
max_cpu_time, 1392
max_wall_time, 1393
min_cpu_time, 1393
min_wall_time, 1393
missed_count, 1393
total_cpu_time, 1393
total_wall_time, 1394
RB_HEAD
Red-Black Tree Handler, 769
RBTree_Node, 1394
Red-Black Tree Handler, 755
RBTree_Visitor
Red-Black Tree Handler, 755

read_entry
rtems_driver_address_table, 1404
Ready
Scheduler_EDF_Context, 1455
Scheduler_EDF_SMP_Context, 1457
ready_chain
Scheduler_priority_Ready_queue, 1471
ready_major
Priority_bit_map_Information, 1388
ready_minor
Priority_bit_map_Information, 1388
ready_queue_index
Scheduler_EDF_SMP_Node, 1457
reconsider_help_request
Scheduler_Operations, 1465
recursionlevel
rtems_filesystem_eval_path_context_t, 1406
Red-Black Tree Handler, 753
_RBTree_Add_child, 756
_RBTree_Extract, 756
_RBTree_Find_inline, 756
_RBTree_Initialize_empty, 757
_RBTree_Initialize_node, 757
_RBTree_Initialize_one, 758
_RBTree_Insert_color, 758
_RBTree_Insert_inline, 758
_RBTree_Insert_with_parent, 759
_RBTree_Is_empty, 760
_RBTree_Is_node_off_tree, 760
_RBTree_Is_root, 761
_RBTree_Iterate, 761
_RBTree_Left, 762
_RBTree_Left_reference, 762
_RBTree_Maximum, 762
_RBTree_Minimum, 763
_RBTree_Parent, 763
_RBTree_Postorder_first, 764
_RBTree_Postorder_next, 764
_RBTree_Predecessor, 765
_RBTree_Replace_node, 765
_RBTree_Right, 766
_RBTree_Right_reference, 766
_RBTree_Root, 767
_RBTree_Root_const_reference, 767
_RBTree_Root_reference, 768
_RBTree_Set_off_tree, 768
_RBTree_Successor, 768
RB_HEAD, 769
RBTree_Node, 755
RBTree_Visitor, 755
Region Manager, 770
rtems_region_create, 770
rtems_region_delete, 771
rtems_region_extend, 771
rtems_region_get_free_information, 771
rtems_region_get_information, 772
rtems_region_get_segment, 772
rtems_region_get_segment_size, 772
rtems_region_ident, 774
rtems_region_resize_segment, 774
rtems_region_return_segment, 775
Region_Control, 1394
REGION_INFORMATION_DEFINE
Classic Region Manager Implementation, 234
Registers
_Thread_Control, 1304
Registry_node
Chain_Iterator, 1316
release_job
Scheduler_Operations, 1465
remove_processor
Scheduler_Operations, 1466
requests
Thread_Scheduler_control, 1522
reserved_for_alignment
CPU_Interrupt_frame, 1340
return_argument
Thread_Wait_information, 1528
return_argument_second
Thread_Wait_information, 1528
return_code
MP_packet_Prefix, 1361
Thread_Wait_information, 1528
rootloc
rtems_filesystem_eval_path_context_t, 1406
routine
Timer_Control, 1532
RTEMS Allocator Mutex, 708
RTEMS Copyright Notice, 709
_Copyright_Notice, 709
_RTEMS_version, 709
RTEMS Per CPU Information, 710
_Per_CPU_Add_job, 714
_Per_CPU_Initialize, 715
_Per_CPU_Perform_jobs, 715
_Per_CPU_State_wait_for_non_initial_state, 715
_Per_CPU_Wait_for_job, 716
_Thread_Get_executing, 716
CPU_STRUCTURE_ALIGNMENT, 716
Per_CPU_Control, 711
Per_CPU_Job, 712
Per_CPU_State, 712
PER_CPU_STATE_INITIAL, 713
PER_CPU_STATE_READY_TO_START_MULTITASKING, 713
PER_CPU_STATE_REQUEST_START_MULTITASKING, 714
PER_CPU_STATE_SHUTDOWN, 714
PER_CPU_STATE_UP, 714
PER_CPU_WATCHDOG_COUNT, 714
Per_CPU_Watchdog_index, 714
PER_CPU_WATCHDOG_MONOTONIC, 714
PER_CPU_WATCHDOG_REALTIME, 714
PER_CPU_WATCHDOG_TICKS, 714
RTEMS Print Support, 717
rtems_print_printer, 717

rtems_print_printer_empty, 718
rtems_print_printer_fprintf, 718
rtems_print_printer_fprintf_putc, 718
rtems_print_printer_printf, 719
rtems_print_printer_printk, 719
rtems_print_printer_task, 719
rtems_print_printer_valid, 720
rtems_printer_task_drain, 720
RTEMS Status Code Checks, 722
RTEMS Termios Device Support, 723
RTEMS Test Framework, 724
RTEMS Test Framework Implementation, 726
T_flags_eno, 729
T_flags_eno_success, 729
T_flags_eq, 729
T_flags_eq_char, 730
T_flags_eq_int, 730
T_flags_eq_ll, 730
T_flags_eq_long, 731
T_flags_eq_mem, 731
T_flags_eq_nstr, 731
T_flags_eq_ptr, 732
T_flags_eq_str, 732
T_flags_eq_uint, 732
T_flags_eq_ull, 733
T_flags_eq_ulong, 733
T_flags_ge_int, 733
T_flags_ge_ll, 734
T_flags_ge_long, 734
T_flags_ge_uint, 734
T_flags_ge_ull, 735
T_flags_ge_ulong, 735
T_flags_gt_int, 735
T_flags_gt_ll, 736
T_flags_gt_long, 736
T_flags_gt_uint, 736
T_flags_gt_ull, 737
T_flags_gt_ulong, 737
T_flags_le_int, 737
T_flags_le_ll, 738
T_flags_le_long, 738
T_flags_le_uint, 738
T_flags_le_ull, 739
T_flags_le_ulong, 739
T_flags_lt_int, 739
T_flags_lt_ll, 740
T_flags_lt_long, 740
T_flags_lt_uint, 740
T_flags_lt_ull, 741
T_flags_lt_ulong, 741
T_flags_ne, 741
T_flags_ne_char, 742
T_flags_ne_int, 742
T_flags_ne_ll, 742
T_flags_ne_long, 743
T_flags_ne_mem, 743
T_flags_ne_nstr, 743
T_flags_ne_ptr, 744
T_flags_ne_str, 744
T_flags_ne_uint, 744
T_flags_ne_ull, 745
T_flags_ne_ulong, 745
T_flags_not_null, 745
T_flags_null, 746
T_flags_psx_error, 746
T_flags_psx_success, 746
T_flags_true, 747
T_VA_ARGS_KIND, 747
rtems_ada_self
Thread Handler, 1068
RTEMS_ALIAS
Base Definitions, 116
RTEMS_ALIGN_DOWN
Base Definitions, 116
RTEMS_ALIGN_UP
Base Definitions, 118
RTEMS_ALIGNED
Base Definitions, 118
RTEMS_ALL_EVENTS
Event Manager, 321
RTEMS_ALLOC_ALIGN
Base Definitions, 119
RTEMS_ALLOC_SIZE
Base Definitions, 119
RTEMS_ALLOC_SIZE_2
Base Definitions, 119
RTEMS_ALREADY_SUSPENDED
Directive Status Codes, 286
rtems_api_configuration_table, 1395
maximum_barriers, 1396
maximum_message_queues, 1396
maximum_partitions, 1396
maximum_periods, 1396
maximum_ports, 1397
maximum_regions, 1397
maximum_semaphores, 1397
maximum_tasks, 1397
maximum_timers, 1398
number_of_initialization_tasks, 1398
User_initialization_tasks_table, 1398
RTEMS_API_Control, 1399
Event, 1399
Signal, 1399
System_event, 1399
rtems_are_statuses_equal
Directive Status Codes, 287
RTEMS_ARRAY_SIZE
Base Definitions, 120
rtems_assert_context, 1400
failed_expression, 1400
file, 1400
function, 1401
line, 1401
rtems_assoc_32_pair, 1401
rtems_assoc_32_to_string
Associativity Routines, 75

rtems_assoc_t, 1402
rtems_assoc_thread_states_to_string
Associativity Routines, 75
rtems_attribute
Directive Attributes, 283
rtems_barrier_create
Barrier Manager, 111
rtems_barrier_delete
Barrier Manager, 111
rtems_barrier_ident
Barrier Manager, 112
rtems_barrier_release
Barrier Manager, 112
rtems_barrier_wait
Barrier Manager, 113
rtems_binary_semaphore, 1402
rtems_board_support_package
Version, 1198
rtems_build_id
Object Services, 633
rtems_build_name
Object Services, 634
rtems_cache_aligned_malloc
Cache Manager, 147
rtems_cache_coherent_add_area
Cache Manager, 147
rtems_cache_coherent_allocate
Cache Manager, 147
rtems_cache_coherent_free
Cache Manager, 148
rtems_cache_flush_multiple_data_lines
Cache Manager, 148
rtems_cache_get_data_cache_size
Cache Manager, 148
rtems_cache_get_instruction_cache_size
Cache Manager, 148
rtems_cache_instruction_sync_after_code_change
Cache Manager, 149
rtems_cache_invalidate_multiple_data_lines
Cache Manager, 149
rtems_cache_invalidate_multiple_instruction_lines
Cache Manager, 149
RTEMS_CALLED_FROM_ISR
Directive Status Codes, 286
rtems_calloc
malloc.h, 1593
rtems_chain_append
Chains, 179
rtems_chain_append_unprotected
Chains, 179
rtems_chain_append_with_empty_check
Chains, 179
rtems_chain_append_with_notification
Chains, 180
rtems_chain_are_nodes_equal
Chains, 180
rtems_chain_extract
Chains, 181
rtems_chain_extract_unprotected
Chains, 181
rtems_chain_first
Chains, 181
rtems_chain_get
Chains, 182
rtems_chain_get_with_empty_check
Chains, 182
rtems_chain_get_with_notification
Chains, 182
rtems_chain_get_with_wait
Chains, 183
rtems_chain_has_only_one_node
Chains, 183
rtems_chain_head
Chains, 184
rtems_chain_immutable_first
Chains, 184
rtems_chain_immutable_head
Chains, 184
rtems_chain_immutable_last
Chains, 185
rtems_chain_immutable_next
Chains, 185
rtems_chain_immutable_previous
Chains, 186
rtems_chain_immutable_tail
Chains, 186
rtems_chain_initialize
Chains, 186
rtems_chain_initialize_empty
Chains, 187
rtems_chain_initialize_node
Chains, 187
RTEMS_CHAIN_INITIALIZER_ONE_NODE
Chains, 178
rtems_chain_insert
Chains, 188
rtems_chain_is_empty
Chains, 188
rtems_chain_is_first
Chains, 188
rtems_chain_is_head
Chains, 189
rtems_chain_is_last
Chains, 189
rtems_chain_is_node_off_chain
Chains, 190
rtems_chain_is_null_node
Chains, 190
rtems_chain_is_tail
Chains, 191
rtems_chain_last
Chains, 191
rtems_chain_next
Chains, 191
rtems_chain_node_count_unprotected
Chains, 192

RTEMS_CHAIN_NODE_INITIALIZER_ONE_NODE_CHAIN
Chains, 178
rtems_chain_prepend
Chains, 192
rtems_chain_prepend_unprotected
Chains, 193
rtems_chain_prepend_with_empty_check
Chains, 193
rtems_chain_prepend_with_notification
Chains, 193
rtems_chain_previous
Chains, 194
rtems_chain_set_off_chain
Chains, 194
rtems_chain_tail
Chains, 195
rtems_clock_get_seconds_since_epoch
Clock Manager, 245
rtems_clock_get_tod
Clock Manager, 246
rtems_clock_get_tod_timeval
Clock Manager, 246
rtems_clock_get_uptime
Clock Manager, 246
rtems_clock_get_uptime_timeval
Clock Manager, 246
rtems_clock_set
Clock Manager, 247
rtems_clock_tick_before
Clock Manager, 247
rtems_clock_tick_later
Clock Manager, 247
rtems_clock_tick_later_usec
Clock Manager, 248
RTEMS_CONCAT
Base Definitions, 120
rtems_configuration_get_do_zero_of_workspace
Application Configuration Information, 60
rtems_configuration_get_idle_task
Application Configuration Information, 60
rtems_configuration_get_idle_task_stack_size
Application Configuration Information, 60
rtems_configuration_get_interrupt_stack_size
Application Configuration Information, 61
rtems_configuration_get_maximum_barriers
Application Configuration Information, 67
rtems_configuration_get_maximum_extensions
Application Configuration Information, 67
rtems_configuration_get_maximum_message_queues
Application Configuration Information, 68
rtems_configuration_get_maximum_partitions
Application Configuration Information, 68
rtems_configuration_get_maximum_periods
Application Configuration Information, 68
rtems_configuration_get_maximum_ports
Application Configuration Information, 69
rtems_configuration_get_maximum_processors
Application Configuration Information, 61
rtems_configuration_get_maximum_regions
Application Configuration Information, 69
rtems_configuration_get_maximum_semaphores
Application Configuration Information, 69
rtems_configuration_get_maximum_tasks
Application Configuration Information, 70
rtems_configuration_get_maximum_timers
Application Configuration Information, 70
rtems_configuration_get_microseconds_per_tick
Application Configuration Information, 61
rtems_configuration_get_milliseconds_per_tick
Application Configuration Information, 62
rtems_configuration_get_nanoseconds_per_tick
Application Configuration Information, 62
rtems_configuration_get_number_of_initial_extensions
Application Configuration Information, 62
rtems_configuration_get_rtems_api_configuration
Application Configuration Information, 70
rtems_configuration_get_stack_allocate_hook
Application Configuration Information, 63
rtems_configuration_get_stack_allocate_init_hook
Application Configuration Information, 63
rtems_configuration_get_stack_allocator_avoids_work_space
Application Configuration Information, 63
rtems_configuration_get_stack_free_hook
Application Configuration Information, 64
rtems_configuration_get_stack_space_size
Application Configuration Information, 71
rtems_configuration_get_ticks_per_timeslice
Application Configuration Information, 64
rtems_configuration_get_unified_work_area
Application Configuration Information, 64
rtems_configuration_get_user_extension_table
Application Configuration Information, 65
rtems_configuration_get_user_multiprocessing_table
Application Configuration Information, 65
rtems_configuration_get_work_space_size
Application Configuration Information, 65
RTEMS_CONTAINER_OF
Base Definitions, 120
rtems_counter_delay_nanoseconds
Free-Running Counter and Busy Wait Delay, 392
rtems_counter_delay_ticks
Free-Running Counter and Busy Wait Delay, 392
rtems_counter_difference
Free-Running Counter and Busy Wait Delay, 392
rtems_counter_frequency
Free-Running Counter and Busy Wait Delay, 393
rtems_counter_initialize_converter
Free-Running Counter and Busy Wait Delay, 393
rtems_counter_nanoseconds_to_ticks
Free-Running Counter and Busy Wait Delay, 393
rtems_counter_read
Free-Running Counter and Busy Wait Delay, 395
rtems_counter_sbintime_to_ticks
Free-Running Counter and Busy Wait Delay, 395
rtems_counter_ticks_to_nanoseconds
Free-Running Counter and Busy Wait Delay, 395

rtems_counter_ticks_to_sbintime
Free-Running Counter and Busy Wait Delay, 396
RTEMS_DECLARE_GLOBAL_SYMBOL
Base Definitions, 121
ISR Handler, 480
RTEMS_DECONST
Base Definitions, 121
RTEMS_DEFINE_GLOBAL_SYMBOL
Base Definitions, 122
RTEMS_DEQUALIFY
Base Definitions, 122
RTEMS_DEQUALIFY_DEPTHX
Base Definitions, 122
rtems_device_driver
I/O Manager, 460
rtems_device_major_number
I/O Manager, 460
rtems_device_minor_number
I/O Manager, 460
RTEMS_DEVOLATILE
Base Definitions, 123
rtems_driver_address_table, 1402
close_entry, 1403
control_entry, 1403
initialization_entry, 1403
open_entry, 1403
read_entry, 1404
write_entry, 1404
rtems_event_receive
Event Manager, 322
rtems_event_send
Event Manager, 323
rtems_event_system_receive
event.h, 1632
systemeventreceive.c, 1829
rtems_event_system_send
event.h, 1633
systemeventsend.c, 1830
rtems_event_transient_receive
event.h, 1633
rtems_event_transient_send
event.h, 1633
rtems_exception_frame_print
Fatal Error Manager, 328
RTEMS_EXPAND
Base Definitions, 123
rtems_extension_create
User Extensions Manager, 1196
rtems_extension_delete
User Extensions Manager, 1196
rtems_extension_ident
User Extensions Manager, 1196
rtems_fatal
Fatal Error Manager, 329
rtems_fatal_error_occurred
Fatal Error Manager, 329
RTEMS_FATAL_SOURCE_APPLICATION
Internal Error Handler, 498
RTEMS_FATAL_SOURCE_ASSERT
Internal Error Handler, 498
RTEMS_FATAL_SOURCE_BDBUF
Internal Error Handler, 498
RTEMS_FATAL_SOURCE_BSP
Internal Error Handler, 498
RTEMS_FATAL_SOURCE_EXCEPTION
Internal Error Handler, 498
RTEMS_FATAL_SOURCE_EXIT
Internal Error Handler, 498
RTEMS_FATAL_SOURCE_HEAP
Internal Error Handler, 499
RTEMS_FATAL_SOURCE_INVALID_HEAP_FREE
Internal Error Handler, 499
RTEMS_FATAL_SOURCE_LAST
Internal Error Handler, 499
RTEMS_FATAL_SOURCE_PANIC
Internal Error Handler, 498
RTEMS_FATAL_SOURCE_SMP
Internal Error Handler, 498
RTEMS_FATAL_SOURCE_STACK_CHECKER
Internal Error Handler, 498
rtems_fatal_source_text
Fatal Error Manager, 329
rtems_filesystem_are_nodes_equal_t
File System Operations, 351
rtems_filesystem_chown_t
File System Operations, 351
rtems_filesystem_clonenode_t
File System Operations, 352
rtems_filesystem_close_t
File System Node Handler, 332
rtems_filesystem_default_are_nodes_equal
File System Operations, 361
rtems_filesystem_default_chown
File System Operations, 361
rtems_filesystem_default_clonenode
File System Operations, 362
rtems_filesystem_default_close
File System Node Handler, 341
rtems_filesystem_default_eval_path
File System Operations, 362
rtems_filesystem_default_fchmod
File System Operations, 362
rtems_filesystem_default_fcntl
File System Node Handler, 341
rtems_filesystem_default_freenode
File System Operations, 363
rtems_filesystem_default_fstat
File System Node Handler, 341
rtems_filesystem_default_fsunmount
File System Operations, 363
rtems_filesystem_default_fsync_or_fdatasync
File System Node Handler, 342
rtems_filesystem_default_fsync_or_fdatasync_success
File System Node Handler, 342
rtems_filesystem_default_ftruncate
File System Node Handler, 342

rtems_filesystem_default_ftruncate_directory
File System Node Handler, 343
rtems_filesystem_default_ioctl
File System Node Handler, 343
rtems_filesystem_default_kqfilter
File System Node Handler, 343
rtems_filesystem_default_link
File System Operations, 363
rtems_filesystem_default_lock
File System Operations, 363
rtems_filesystem_default_lseek
File System Node Handler, 344
rtems_filesystem_default_lseek_directory
File System Node Handler, 344
rtems_filesystem_default_lseek_file
File System Node Handler, 345
rtems_filesystem_default_mknod
File System Operations, 364
rtems_filesystem_default_mmap
File System Node Handler, 345
rtems_filesystem_default_mount
File System Operations, 364
rtems_filesystem_default_open
File System Node Handler, 346
rtems_filesystem_default_pathconf
IO Library, 473
rtems_filesystem_default_poll
File System Node Handler, 346
rtems_filesystem_default_read
File System Node Handler, 346
rtems_filesystem_default_readlink
File System Operations, 364
rtems_filesystem_default_readv
File System Node Handler, 347
rtems_filesystem_default_rename
File System Operations, 365
rtems_filesystem_default_rmnod
File System Operations, 365
rtems_filesystem_default_statvfs
File System Operations, 366
rtems_filesystem_default_symlink
File System Operations, 366
rtems_filesystem_default_unlock
File System Operations, 366
rtems_filesystem_default_unmount
File System Operations, 366
rtems_filesystem_default_utime
File System Operations, 367
rtems_filesystem_default_write
File System Node Handler, 347
rtems_filesystem_default_writev
File System Node Handler, 347
rtems_filesystem_eval_path_context_t, 1404
currentloc, 1405
flags, 1405
path, 1405
pathlen, 1406
recursionlevel, 1406
rootloc, 1406
startloc, 1406
token, 1406
tokenlen, 1407
rtems_filesystem_eval_path_t
File System Operations, 352
rtems_filesystem_fchmod_t
File System Operations, 353
rtems_filesystem_fcntl_t
File System Node Handler, 333
rtems_filesystem_fdatasync_t
File System Node Handler, 333
rtems_filesystem_freenode_t
File System Operations, 353
rtems_filesystem_fsmount_me_t
File System Operations, 354
rtems_filesystem_fstat_t
File System Node Handler, 334
rtems_filesystem_fsunmount_me_t
File System Operations, 354
rtems_filesystem_fsync_t
File System Node Handler, 334
rtems_filesystem_ftruncate_t
File System Node Handler, 335
rtems_filesystem_get_mount_handler
IO Library, 471
rtems_filesystem_global_location_t, 1407
deferred_released_count, 1408
deferred_released_next, 1408
IO Library, 470
rtems_filesystem_initialize
IO Library, 471
rtems_filesystem_ioctl_t
File System Node Handler, 335
rtems_filesystem_iterate
File System Types and Mount, 372
rtems_filesystem_kqfilter_t
File System Node Handler, 336
rtems_filesystem_limits_and_options_t, 1408
rtems_filesystem_link_t
File System Operations, 354
rtems_filesystem_location_info_tt, 1409
rtems_filesystem_lseek_t
File System Node Handler, 336
rtems_filesystem_mknod_t
File System Operations, 355
rtems_filesystem_mmap_t
File System Node Handler, 337
rtems_filesystem_mount_configuration, 1409
rtems_filesystem_mount_iterate
File System Types and Mount, 372
rtems_filesystem_mount_t
File System Operations, 356
rtems_filesystem_mount_table_entry_tt, 1410
unmount_task, 1410
rtems_filesystem_mt_entry_lock_t
File System Operations, 356
rtems_filesystem_mt_entry_unlock_t

File System Operations, 357
rtems_filesystem_mt_entry_visitor
File System Types and Mount, 369
rtems_filesystem_open_t
File System Node Handler, 337
rtems_filesystem_poll_t
File System Node Handler, 338
rtems_filesystem_read_t
File System Node Handler, 338
rtems_filesystem_readlink_t
File System Operations, 357
rtems_filesystem_readv_t
File System Node Handler, 339
rtems_filesystem_register
File System Types and Mount, 373
rtems_filesystem_rename_t
File System Operations, 358
rtems_filesystem_rmnod_t
File System Operations, 358
rtems_filesystem_split_dev_t
IO Library, 470
rtems_filesystem_statvfs_t
File System Operations, 359
rtems_filesystem_symlink_t
File System Operations, 359
rtems_filesystem_table
File System Types and Mount, 374
rtems_filesystem_table_t, 1411
rtems_filesystem_unmount_t
File System Operations, 360
rtems_filesystem_unregister
File System Types and Mount, 373
rtems_filesystem_utime_t
File System Operations, 360
rtems_filesystem_write_t
File System Node Handler, 340
rtems_filesystem_writev_t
File System Node Handler, 340
RTEMS_FLOATING_POINT
Directive Attributes, 281
rtems_get_copyright_notice
Application Configuration Information, 71
rtems_get_version_string
Application Configuration Information, 71
RTEMS_GLOBAL
Directive Attributes, 281
RTEMS_HAVE_MEMBER_SAME_TYPE
Base Definitions, 123
rtems_heap_allocate_aligned_with_boundary
malloc.h, 1594
rtems_heap_extend
malloc.h, 1594
rtems_heap_greedy_allocate
malloc.h, 1595
rtems_heap_greedy_allocate_all_except_largest
malloc.h, 1595
rtems_heap_greedy_free
malloc.h, 1595
RTEMS_ID_NONE
Basic Types, 131
RTEMS_ILLEGAL_ON_REMOTE_OBJECT
Directive Status Codes, 286
RTEMS_ILLEGAL_ON_SELF
Directive Status Codes, 286
RTEMS_INCORRECT_STATE
Directive Status Codes, 286
RTEMS_INHERIT_PRIORITY
Directive Attributes, 281
rtems_initialization_tasks_table, 1411
argument, 1412
attribute_set, 1412
entry_point, 1412
initial_priority, 1413
mode_set, 1413
name, 1413
stack_size, 1413
rtems_initialize_executive
Initialization and Shutdown, 494
RTEMS_INTERNAL_ERROR
Directive Status Codes, 287
rtems_internal_error_text
Fatal Error Manager, 330
rtems_interrupt_catch
Interrupt Manager, 508
rtems_interrupt_cause
Interrupt Manager, 502
rtems_interrupt_clear
Interrupt Manager, 502
rtems_interrupt_get_affinity
Interrupt Manager Extension, 512
rtems_interrupt_handler_install
Interrupt Manager Extension, 513
rtems_interrupt_handler_iterate
Interrupt Manager Extension, 514
rtems_interrupt_handler_remove
Interrupt Manager Extension, 514
RTEMS_INTERRUPT_LEVEL
Task Modes, 981
rtems_interrupt_level_body
Task Modes, 981
rtems_interrupt_local_disable
Interrupt Manager, 503
rtems_interrupt_local_enable
Interrupt Manager, 503
rtems_interrupt_lock_acquire
Interrupt Manager, 503
rtems_interrupt_lock_acquire_isr
Interrupt Manager, 504
RTEMS_INTERRUPT_LOCK_DECLARE
Interrupt Manager, 504
RTEMS_INTERRUPT_LOCK_DEFINE
Interrupt Manager, 504
rtems_interrupt_lock_destroy
Interrupt Manager, 505
rtems_interrupt_lock_initialize
Interrupt Manager, 505

RTEMS_INTERRUPT_LOCK_INITIALIZER
Interrupt Manager, 505
rtems_interrupt_lock_interrupt_disable
Interrupt Manager, 506
RTEMS_INTERRUPT_LOCK_MEMBER
Interrupt Manager, 506
RTEMS_INTERRUPT_LOCK_REFERENCE
Interrupt Manager, 506
rtems_interrupt_lock_release
Interrupt Manager, 507
rtems_interrupt_lock_release_isr
Interrupt Manager, 507
rtems_interrupt_mask
Task Modes, 982
rtems_interrupt_per_handler_routine
Interrupt Manager Extension, 511
rtems_interrupt_server_action, 1414
Interrupt Manager Extension, 511
rtems_interrupt_server_action_prepend
Interrupt Manager Extension, 515
rtems_interrupt_server_config, 1414
destroy, 1415
storage_area, 1415
storage_size, 1415
rtems_interrupt_server_control, 1416
Interrupt Manager Extension, 512
rtems_interrupt_server_create
Interrupt Manager Extension, 515
rtems_interrupt_server_delete
Interrupt Manager Extension, 516
rtems_interrupt_server_entry, 1417
rtems_interrupt_server_entry_destroy
Interrupt Manager Extension, 516
rtems_interrupt_server_entry_initialize
Interrupt Manager Extension, 517
rtems_interrupt_server_entry_move
Interrupt Manager Extension, 517
rtems_interrupt_server_entry_submit
Interrupt Manager Extension, 518
rtems_interrupt_server_handler_install
Interrupt Manager Extension, 518
rtems_interrupt_server_handler_iterate
Interrupt Manager Extension, 519
rtems_interrupt_server_handler_remove
Interrupt Manager Extension, 519
rtems_interrupt_server_initialize
Interrupt Manager Extension, 520
rtems_interrupt_server_move
Interrupt Manager Extension, 520
rtems_interrupt_server_request, 1417
rtems_interrupt_server_request_destroy
Interrupt Manager Extension, 521
rtems_interrupt_server_request_initialize
Interrupt Manager Extension, 521
rtems_interrupt_server_request_set_vector
Interrupt Manager Extension, 522
rtems_interrupt_server_request_submit
Interrupt Manager Extension, 523
rtems_interrupt_server_resume
Interrupt Manager Extension, 523
rtems_interrupt_server_set_affinity
Interrupt Manager Extension, 524
rtems_interrupt_server_suspend
Interrupt Manager Extension, 524
rtems_interrupt_set_affinity
Interrupt Manager Extension, 525
RTEMS_INTERRUPTED
Directive Status Codes, 287
RTEMS_INVALID_ADDRESS
Directive Status Codes, 286
RTEMS_INVALID_CLOCK
Directive Status Codes, 286
RTEMS_INVALID_ID
Directive Status Codes, 286
RTEMS_INVALID_NAME
Directive Status Codes, 286
RTEMS_INVALID_NODE
Directive Status Codes, 286
RTEMS_INVALID_NUMBER
Directive Status Codes, 286
RTEMS_INVALID_PRIORITY
Directive Status Codes, 286
RTEMS_INVALID_SIZE
Directive Status Codes, 286
rtems_io_close
I/O Manager, 460
rtems_io_control
I/O Manager, 461
RTEMS_IO_ERROR
Directive Status Codes, 287
rtems_io_initialize
I/O Manager, 461
rtems_io_open
I/O Manager, 462
rtems_io_read
I/O Manager, 463
rtems_io_register_driver
I/O Manager, 463
rtems_io_register_name
I/O Manager, 464
rtems_io_unregister_driver
I/O Manager, 465
rtems_io_write
I/O Manager, 465
rtems_is_name_valid
Support Services, 950
rtems_is_status_successful
Directive Status Codes, 287
rtems_iterate_over_all_threads
Thread Handler, 1066
rtems_libio_ioctl_args_t, 1418
rtems_libio_iop_is_append
IO Library, 471
rtems_libio_iop_is_no_delay
IO Library, 472
rtems_libio_iop_is_readable

IO Library, 472
rtems_libio_iop_is_writeable
IO Library, 472
rtems_libio_open_close_args_t, 1418
rtems_libio_rw_args_t, 1419
rtems_libio_tt, 1419
RTEMS_LOCAL
Directive Attributes, 282
rtems_malloc
malloc.h, 1595
rtems_malloc_dirty_memory
malloc.h, 1596
RTEMS_Malloc_Heap
malloc.h, 1596
mallocheap.c, 1790
rtems_memalign
malloc.h, 1596
rtems_message_queue_broadcast
Message Manager, 546
RTEMS_MESSAGE_QUEUE_BUFFER
Message Manager, 546
rtems_message_queue_config, 1420
storage_area, 1421
storage_free, 1421
rtems_message_queue_construct
Message Manager, 546
rtems_message_queue_create
Message Manager, 547
rtems_message_queue_delete
Message Manager, 548
rtems_message_queue_flush
Message Manager, 548
rtems_message_queue_get_number_pending
Message Manager, 549
rtems_message_queue_ident
Message Manager, 549
rtems_message_queue_receive
Message Manager, 550
rtems_message_queue_send
Message Manager, 550
rtems_message_queue_urgent
Message Manager, 551
RTEMS_MICROSECONDS_TO_TICKS
Support Services, 948
RTEMS_MILLISECONDS_TO_MICROSECONDS
Support Services, 949
RTEMS_MILLISECONDS_TO_TICKS
Support Services, 949
rtems_minimum_stack_size
Stack Handler, 938
rtems_mkdir
IO Library, 473
RTEMS_MP_NOT_CONFIGURED
Directive Status Codes, 286
RTEMS_MULTIPROCESSOR_RESOURCE_SHARING
Directive Attributes, 282
rtems_name
Basic Types, 131
rtems_name_to_characters
Support Services, 950
RTEMS_NEWLIB_EXTENSION
Standard C Library Support, 940
RTEMS_NO_FLOATING_POINT
Directive Attributes, 282
RTEMS_NO_MEMORY
Directive Status Codes, 287
RTEMS_NO_WAIT
Directive Options, 284
RTEMS_NOT_CONFIGURED
Directive Status Codes, 286
RTEMS_NOT_DEFINED
Directive Status Codes, 286
RTEMS_NOT_IMPLEMENTED
Directive Status Codes, 287
RTEMS_NOT_OWNER_OF_RESOURCE
Directive Status Codes, 286
RTEMS_OBFUSCATE_VARIABLE
Base Definitions, 124
rtems_object_api_class_information, 1421
auto_extend, 1422
maximum, 1422
maximum_id, 1422
minimum_id, 1423
unallocated, 1423
rtems_object_api_maximum_class
Object Services, 637
rtems_object_api_minimum_class
Object Services, 637
rtems_object_get_api_class_name
Object Services, 637
rtems_object_get_api_name
Object Services, 638
rtems_object_get_class_information
Object Services, 638
rtems_object_get_classic_name
Object Services, 638
rtems_object_get_name
Object Services, 639
rtems_object_id_api_maximum_class
Object Services, 639
rtems_object_id_get_api
Object Services, 634
rtems_object_id_get_class
Object Services, 634
rtems_object_id_get_index
Object Services, 635
rtems_object_id_get_node
Object Services, 635
RTEMS_OBJECT_ID_INITIAL
Object Services, 635
rtems_object_set_name
Object Services, 639
RTEMS_OBJECT_WAS_DELETED
Directive Status Codes, 286
RTEMS_PARTITION_ALIGNMENT
Partition Manager, 650

rtems_partition_create
Partition Manager, 650
rtems_partition_delete
Partition Manager, 652
rtems_partition_get_buffer
Partition Manager, 653
rtems_partition_ident
Partition Manager, 653
rtems_partition_return_buffer
Partition Manager, 654
rtems_per_filesystem_routine
File System Types and Mount, 369
rtems_port_create
Dual-Ported Memory Manager, 292
rtems_port_delete
Dual-Ported Memory Manager, 293
rtems_port_external_to_internal
Dual-Ported Memory Manager, 293
rtems_port_ident
Dual-Ported Memory Manager, 293
rtems_port_internal_to_external
Dual-Ported Memory Manager, 294
RTEMS_PREDICT_FALSE
Base Definitions, 124
RTEMS_PREDICT_TRUE
Base Definitions, 126
rtems_print_printer
RTEMS Print Support, 717
rtems_print_printer_empty
RTEMS Print Support, 718
rtems_print_printer_fprintf
RTEMS Print Support, 718
rtems_print_printer_fprintf_putc
RTEMS Print Support, 718
rtems_print_printer_printf
RTEMS Print Support, 719
rtems_print_printer_printk
RTEMS Print Support, 719
rtems_print_printer_task
RTEMS Print Support, 719
rtems_print_printer_valid
RTEMS Print Support, 720
rtems_printer, 1423
rtems_printer_task_context, 1424
rtems_printer_task_drain
RTEMS Print Support, 720
rtems_printf
print.h, 1618
RTEMS_PRINTFLIKE
Base Definitions, 126
Fatal Error Manager, 330
RTEMS_PRIORITY_CEILING
Directive Attributes, 282
RTEMS_PROXY_BLOCKING
Directive Status Codes, 287
rtems_put_char
bsplo.h, 1575
rtems_putc
bsplo.h, 1575
rtems_putc.c, 1793
rtems_putc.c
rtems_putc, 1793
rtems_rate_monotonic_cancel
Rate-Monotonic Manager, 749
rtems_rate_monotonic_create
Rate-Monotonic Manager, 749
rtems_rate_monotonic_delete
Rate-Monotonic Manager, 750
rtems_rate_monotonic_get_statistics
Rate-Monotonic Manager, 750
rtems_rate_monotonic_get_status
Rate-Monotonic Manager, 750
rtems_rate_monotonic_ident
Rate-Monotonic Manager, 751
rtems_rate_monotonic_period
Rate-Monotonic Manager, 751
rtems_rate_monotonic_period_states
Rate-Monotonic Manager, 749
rtems_rate_monotonic_period_statistics, 1424
count, 1425
max_cpu_time, 1425
max_wall_time, 1425
min_cpu_time, 1426
min_wall_time, 1426
missed_count, 1426
total_cpu_time, 1426
total_wall_time, 1427
rtems_rate_monotonic_period_status, 1427
executed_since_last_period, 1428
owner, 1428
postponed_jobs_count, 1428
since_last_period, 1428
state, 1428
rtems_rate_monotonic_report_statistics_with_plugin
Rate-Monotonic Manager, 752
rtems_rate_monotonic_reset_statistics
Rate-Monotonic Manager, 752
rtems_region_create
Region Manager, 770
rtems_region_delete
Region Manager, 771
rtems_region_extend
Region Manager, 771
rtems_region_get_free_information
Region Manager, 771
rtems_region_get_information
Region Manager, 772
rtems_region_get_segment
Region Manager, 772
rtems_region_get_segment_size
Region Manager, 772
rtems_region_ident
Region Manager, 774
rtems_region_resize_segment
Region Manager, 774
rtems_region_return_segment

Region Manager, 775
RTEMS_RESOURCE_IN_USE
Directive Status Codes, 286
rtems_resource_is_unlimited
Application Configuration Information, 66
rtems_resource_maximum_per_allocation
Application Configuration Information, 66
rtems_resource_posix_api, 1429
rtems_resource_rtems_api, 1429
rtems_resource_snapshot, 1430
rtems_resource_snapshot_check
Standard C Library Support, 941
rtems_resource_snapshot_equal
Standard C Library Support, 942
rtems_resource_snapshot_take
Standard C Library Support, 942
rtems_resource_unlimited
Application Configuration Information, 67
RTEMS_RETURN_ADDRESS
Base Definitions, 126
rtems_scheduler_add_processor
Task Manager, 965
RTEMS_SCHEDULER_ASSIGN
scheduler.h, 1601
rtems_scheduler_get_maximum_priority
Task Manager, 966
rtems_scheduler_get_processor
Task Manager, 963
rtems_scheduler_get_processor_maximum
Task Manager, 964
rtems_scheduler_get_processor_set
Task Manager, 966
rtems_scheduler_ident
Task Manager, 967
rtems_scheduler_ident_by_processor
Task Manager, 967
rtems_scheduler_ident_by_processor_set
Task Manager, 968
rtems_scheduler_map_priority_from_posix
Task Manager, 969
rtems_scheduler_map_priority_to_posix
Task Manager, 969
rtems_scheduler_remove_processor
Task Manager, 970
RTEMS_SCORE_COREMSG_ENABLE_BLOCKING_SEND
Message Queue Handler, 555
RTEMS_SCORE_COREMSG_ENABLE_MESSAGE_PRIORITY
Message Queue Handler, 555
RTEMS_SCORE_ROBUST_THREAD_DISPATCH
Thread Handler, 1007
RTEMS_SECTION
Base Definitions, 127
rtems_semaphore_create
Semaphore Manager, 898
rtems_semaphore_delete
Semaphore Manager, 900
rtems_semaphore_flush
Semaphore Manager, 901
rtems_semaphore_ident
Semaphore Manager, 901
rtems_semaphore_obtain
Semaphore Manager, 902
rtems_semaphore_release
Semaphore Manager, 902
rtems_semaphore_set_priority
Semaphore Manager, 903
rtems_shutdown_executive
Initialization and Shutdown, 494
rtems_signal_catch
Signal Manager, 911
rtems_signal_send
Signal Manager, 912
RTEMS_STATIC_ASSERT
Base Definitions, 127
rtems_status_code
Directive Status Codes, 286
rtems_status_code_to_errno
Directive Status Codes, 288
rtems_status_text
Directive Status Codes, 289
RTEMS_STRING
Base Definitions, 127
RTEMS_SUCCESSFUL
Directive Status Codes, 286
RTEMS_SYMBOL_NAME
Base Definitions, 128
rtems_sysinit_item, 1430
rtems_task_argument
Task Manager, 965
rtems_task_config, 1431
maximum_thread_local_storage_size, 1431
storage_area, 1431
storage_free, 1432
storage_size, 1432
rtems_task_construct
Task Manager, 970
rtems_task_create
Task Manager, 971
rtems_task_delete
Task Manager, 972
RTEMS_TASK_EXITTED
Directive Status Codes, 286
rtems_task_get_affinity
Task Manager, 973
rtems_task_get_priority
Task Manager, 973
rtems_task_get_scheduler
Task Manager, 973
rtems_task_ident
Task Manager, 974
rtems_task_is_suspended
Task Manager, 975
rtems_task_iterate
Task Manager, 975
rtems_task_mode
Task Manager, 975

rtems_task_restart
Task Manager, 976
rtems_task_resume
Task Manager, 976
rtems_task_set_affinity
Task Manager, 976
rtems_task_set_priority
Task Manager, 977
rtems_task_set_scheduler
Task Manager, 977
rtems_task_start
Task Manager, 977
RTEMS_TASK_STORAGE_ALIGNMENT
Task Manager, 964
RTEMS_TASK_STORAGE_SIZE
Task Manager, 964
rtems_task_suspend
Task Manager, 978
rtems_task_wake_after
Task Manager, 978
rtems_task_wake_when
Task Manager, 978
rtems_termios_baud_to_number
termiostypes.h, 1784
rtems_termios_callbacks, 1432
rtems_termios_default_isig_handler
Termios, 988
rtems_termios_device_context, 1433
termiostypes.h, 1783
rtems_termios_device_context_initialize
termiostypes.h, 1784
RTEMS_TERMIOS_DEVICE_CONTEXT_INITIALIZER
termiostypes.h, 1783
rtems_termios_device_flow, 1433
start_remote_tx, 1434
stop_remote_tx, 1434
rtems_termios_device_handler, 1435
first_open, 1435
ioctl, 1436
last_close, 1436
poll_read, 1437
set_attributes, 1437
write, 1438
rtems_termios_device_install
termiostypes.h, 1784
rtems_termios_device_lock_acquire
termiostypes.h, 1785
rtems_termios_device_lock_release
termiostypes.h, 1785
rtems_termios_device_node, 1438
termiostypes.h, 1783
rtems_termios_get_device_context
termiostypes.h, 1786
RTEMS_TERMIOS_IPROC_CONTINUE
Termios, 987
RTEMS_TERMIOS_IPROC_DONE
Termios, 988
RTEMS_TERMIOS_IPROC_INTERRUPT
Termios, 988
rtems_termios_iproc_status_code
Termios, 987
rtems_termios_isig_handler
Termios, 987
rtems_termios_kqfilter
termiostypes.h, 1786
rtems_termios_linesw, 1439
rtems_termios_mmap
termiostypes.h, 1786
rtems_termios_number_to_baud
termiostypes.h, 1786
rtems_termios_poll
termiostypes.h, 1787
rtems_termios_posix_isig_handler
Termios, 988
rtems_termios_rawbuf, 1439
rtems_termios_register_isig_handler
Termios, 989
rtems_termios_set_best_baud
termiostypes.h, 1787
rtems_termios_set_initial_baud
termiostypes.h, 1787
rtems_termios_tty, 1439
rtems_test_begin
Test Support, 993
rtems_test_busy_cpu_usage
Test Support, 993
rtems_test_end
Test Support, 993
rtems_test_parallel
Test Support, 993
rtems_test_parallel_context, 1441
rtems_test_parallel_get_task_id
Test Support, 994
rtems_test_parallel_is_master_worker
Test Support, 994
rtems_test_parallel_job, 1441
body, 1442
cascade, 1442
fini, 1442
init, 1443
rtems_test_parallel_stop_job
Test Support, 995
rtems_test_parallel_worker_setup
Test Support, 992
rtems_test_printf
Test Support, 995
rtems_test_run
Test Support, 995
rtems_time_of_day, 1443
rtems_timecounter_install
Timecounter Support, 1152
RTEMS_TIMECOUNTER_QUALITY_CLOCK_DRIVER
Timecounter Support, 1152
rtems_timecounter_simple, 1444
rtems_timecounter_simple_downcounter_get
Timecounter Support, 1152

rtems_timecounter_simple_downcounter_tick
Timecounter Support, 1153
rtems_timecounter_simple_install
Timecounter Support, 1153
rtems_timecounter_simple_scale
Timecounter Support, 1154
rtems_timecounter_simple_upcounter_get
Timecounter Support, 1155
rtems_timecounter_simple_upcounter_tick
Timecounter Support, 1155
rtems_timecounter_tick
Timecounter Support, 1155
RTEMS_TIMEOUT
Directive Status Codes, 286
rtems_timer_cancel
Timer Manager, 1158
rtems_timer_create
Timer Manager, 1159
rtems_timer_delete
Timer Manager, 1159
rtems_timer_fire_after
Timer Manager, 1159
rtems_timer_fire_when
Timer Manager, 1160
rtems_timer_get_information
Timer Manager, 1160
rtems_timer_ident
Timer Manager, 1160
rtems_timer_information, 1445
initial, 1445
start_time, 1445
stop_time, 1445
the_class, 1446
rtems_timer_initiate_server
Timer Manager, 1161
rtems_timer_reset
Timer Manager, 1161
rtems_timer_server_fire_after
Timer Manager, 1162
rtems_timer_server_fire_when
Timer Manager, 1162
RTEMS_TOO_MANY
Directive Status Codes, 286
RTEMS_TYPEOF_REFX
Base Definitions, 128
RTEMS_UNREACHABLE
basedefs.h, 1676
RTEMS_UNSATISFIED
Directive Status Codes, 286
rtems_version
Version, 1198
rtems_version_control_key
Version, 1199
rtems_version_control_key_is_valid
Version, 1199
rtems_version_major
Version, 1199
rtems_version_minor
Version, 1200
rtems_version_revision
Version, 1200
rtems_vprintf
print.h, 1618
print_printf.c, 1791
RTEMS_WAIT
Directive Options, 284
RTEMS_WEAK
Base Definitions, 129
RTEMS_WEAK_ALIAS
Base Definitions, 129
rtems_workspace_allocate
Support Services, 950
rtems_workspace_free
Support Services, 951
rtems_workspace_get_information
Support Services, 951
rtems_workspace_greedy_allocate
Support Services, 951
rtems_workspace_greedy_allocate_all_except_largest
Support Services, 952
rtems_workspace_greedy_free
Support Services, 952
RTEMS_XCONCAT
Base Definitions, 129
RTEMS_XSTRING
Base Definitions, 130
RTEMS_ZERO_LENGTH_ARRAY
Base Definitions, 130
RtemsEventReqSendReceive_Context, 1446
RtemsEventReqSendReceive_Fixture
spec:/rtems/event/req/send-receive, 1230
RtemsEventReqSendReceive_PreDesc
spec:/rtems/event/req/send-receive, 1230
RtemsEventReqSendReceive_PreDesc_Id
spec:/rtems/event/req/send-receive, 1231
RtemsEventReqSendReceive_PreDesc_ReceiverState
spec:/rtems/event/req/send-receive, 1231
RtemsEventReqSendReceive_PreDesc_Satisfy
spec:/rtems/event/req/send-receive, 1231
RtemsEventReqSendReceive_PreDesc_Send
spec:/rtems/event/req/send-receive, 1231
RtemsEventReqSendReceive_Run
spec:/rtems/event/req/send-receive, 1229
RtemsEventValEventConstant_Run
spec:/rtems/event/val/event-constant, 1234
RtemsMessageReqConstructErrors_Context, 1448
RtemsMessageReqConstructErrors_Fixture
spec:/rtems/message/req/construct-errors, 1242
RtemsMessageReqConstructErrors_PreDesc
spec:/rtems/message/req/construct-errors, 1242
RtemsMessageReqConstructErrors_PreDesc_Area
spec:/rtems/message/req/construct-errors, 1242
RtemsMessageReqConstructErrors_PreDesc_AreaSize
spec:/rtems/message/req/construct-errors, 1243
RtemsMessageReqConstructErrors_PreDesc_Id
spec:/rtems/message/req/construct-errors, 1243

RtemsMessageReqConstructErrors_PreDesc_MaxPending
spec:/rtems/message/req/construct-errors, 1243
RtemsMessageReqConstructErrors_PreDesc_MaxSize
spec:/rtems/message/req/construct-errors, 1243
RtemsMessageReqConstructErrors_PreDesc_Name
spec:/rtems/message/req/construct-errors, 1244
RtemsMessageReqConstructErrors_PreDesc_Queues
spec:/rtems/message/req/construct-errors, 1244
RtemsPartReqCreate_Context, 1448
RtemsPartReqCreate_Fixture
spec:/rtems/part/req/create, 1250
RtemsPartReqCreate_PreDesc
spec:/rtems/part/req/create, 1250
RtemsPartReqCreate_PreDesc_Id
spec:/rtems/part/req/create, 1250
RtemsPartReqCreate_PreDesc_Length
spec:/rtems/part/req/create, 1250
RtemsPartReqCreate_PreDesc_Name
spec:/rtems/part/req/create, 1251
RtemsPartReqCreate_PreDesc_Parts
spec:/rtems/part/req/create, 1251
RtemsPartReqCreate_PreDesc_Size
spec:/rtems/part/req/create, 1251
RtemsPartReqCreate_PreDesc_Start
spec:/rtems/part/req/create, 1251
RtemsPartReqDelete_Context, 1449
RtemsPartReqDelete_Fixture
spec:/rtems/part/req/delete, 1253
RtemsPartReqDelete_PreDesc
spec:/rtems/part/req/delete, 1254
RtemsPartReqDelete_PreDesc_Id
spec:/rtems/part/req/delete, 1254
RtemsPartReqDelete_PreDesc_InUse
spec:/rtems/part/req/delete, 1254
RtemsPartReqDelete_TransitionInfo
spec:/rtems/part/req/delete, 1254
RtemsPartReqDelete_TransitionMap
spec:/rtems/part/req/delete, 1255
RtemsPartReqGetBuffer_Context, 1449
RtemsPartReqGetBuffer_Fixture
spec:/rtems/part/req/get-buffer, 1257
RtemsPartReqGetBuffer_PreDesc
spec:/rtems/part/req/get-buffer, 1258
RtemsPartReqGetBuffer_PreDesc_Avail
spec:/rtems/part/req/get-buffer, 1258
RtemsPartReqGetBuffer_PreDesc_Buf
spec:/rtems/part/req/get-buffer, 1258
RtemsPartReqGetBuffer_PreDesc_Id
spec:/rtems/part/req/get-buffer, 1258
RtemsPartReqGetBuffer_TransitionInfo
spec:/rtems/part/req/get-buffer, 1259
RtemsPartReqGetBuffer_TransitionMap
spec:/rtems/part/req/get-buffer, 1259
RtemsPartReqReturnBuffer_Context, 1450
RtemsPartReqReturnBuffer_Fixture
spec:/rtems/part/req/return-buffer, 1261
RtemsPartReqReturnBuffer_PreDesc
spec:/rtems/part/req/return-buffer, 1262
RtemsPartReqReturnBuffer_PreDesc_Buf
spec:/rtems/part/req/return-buffer, 1262
RtemsPartReqReturnBuffer_PreDesc_Id
spec:/rtems/part/req/return-buffer, 1262
RtemsPartReqReturnBuffer_TransitionInfo
spec:/rtems/part/req/return-buffer, 1262
RtemsPartReqReturnBuffer_TransitionMap
spec:/rtems/part/req/return-buffer, 1263
RtemsReqIdent_Context, 1450
RtemsReqIdent_Fixture
spec:/rtems/req/ident, 1269
RtemsReqIdent_PreDesc
spec:/rtems/req/ident, 1269
RtemsReqIdent_PreDesc_Id
spec:/rtems/req/ident, 1269
RtemsReqIdent_PreDesc_Name
spec:/rtems/req/ident, 1269
RtemsReqIdent_PreDesc_Node
spec:/rtems/req/ident, 1270
RtemsReqIdent_Run
spec:/rtems/req/ident, 1268
RtemsReqIdentLocal_Context, 1451
RtemsReqIdentLocal_Fixture
spec:/rtems/req/ident-local, 1273
RtemsReqIdentLocal_PreDesc
spec:/rtems/req/ident-local, 1273
RtemsReqIdentLocal_PreDesc_Id
spec:/rtems/req/ident-local, 1273
RtemsReqIdentLocal_PreDesc_Name
spec:/rtems/req/ident-local, 1273
RtemsReqIdentLocal_Run
spec:/rtems/req/ident-local, 1272
RtemsReqIdentLocal_TransitionInfo
spec:/rtems/req/ident-local, 1274
RtemsReqIdentLocal_TransitionMap
spec:/rtems/req/ident-local, 1274
RtemsTaskReqConstructErrors_Context, 1452
RtemsTaskReqConstructErrors_Fixture
spec:/rtems/task/req/construct-errors, 1278
RtemsTaskReqConstructErrors_PreDesc
spec:/rtems/task/req/construct-errors, 1279
RtemsTaskReqConstructErrors_PreDesc_Ext
spec:/rtems/task/req/construct-errors, 1279
RtemsTaskReqConstructErrors_PreDesc_Id
spec:/rtems/task/req/construct-errors, 1279
RtemsTaskReqConstructErrors_PreDesc_Name
spec:/rtems/task/req/construct-errors, 1279
RtemsTaskReqConstructErrors_PreDesc_Preempt
spec:/rtems/task/req/construct-errors, 1280
RtemsTaskReqConstructErrors_PreDesc_Prio
spec:/rtems/task/req/construct-errors, 1280
RtemsTaskReqConstructErrors_PreDesc_Stack
spec:/rtems/task/req/construct-errors, 1280
RtemsTaskReqConstructErrors_PreDesc_Tasks
spec:/rtems/task/req/construct-errors, 1280
RtemsTaskReqConstructErrors_PreDesc_TLS
spec:/rtems/task/req/construct-errors, 1281
RtemsTaskReqIdent_Context, 1452

RtemsTaskReqIdent_Fixture
spec:/rtems/task/req/ident, 1284
RtemsTaskReqIdent_PreDesc
spec:/rtems/task/req/ident, 1284
RtemsTaskReqIdent_PreDesc_Pre
spec:/rtems/task/req/ident, 1284
RtemsTaskReqIdent_TransitionInfo
spec:/rtems/task/req/ident, 1284
RtemsTaskReqIdent_TransitionMap
spec:/rtems/task/req/ident, 1285
RTRAP
SPARC Assembler Support, 835
runner_task_config
spec:/testsuites/validation-0, 1291
Runtime Measurements, 776

saved_arg0
SPARC_Minimum_stack_frame, 1486
saved_arg1
SPARC_Minimum_stack_frame, 1486
saved_arg2
SPARC_Minimum_stack_frame, 1486
saved_arg3
SPARC_Minimum_stack_frame, 1486
saved_arg4
SPARC_Minimum_stack_frame, 1487
saved_arg5
SPARC_Minimum_stack_frame, 1487
schedule
Scheduler_Operations, 1466
Scheduler Handler, 847
_Scheduler_Acquire_critical, 855
_Scheduler_Ask_for_help, 855
_Scheduler_Block, 855
_Scheduler_Block_node, 856
_Scheduler_Build_id, 856
_Scheduler_Cancel_job, 857
_Scheduler_Count, 883
_Scheduler_Discard_idle_thread, 864
_Scheduler_Exchange_idle_thread, 864
_Scheduler_Generic_block, 864
_Scheduler_Get_affinity, 865
_Scheduler_Get_by_CPU, 865
_Scheduler_Get_by_id, 866
_Scheduler_Get_context, 866
_Scheduler_Get_index, 866
_Scheduler_Get_index_by_id, 867
_Scheduler_Get_processor_count, 867
_Scheduler_Get_processors, 867
_Scheduler_Handler_initialization, 868
_Scheduler_Has_processor_ownership, 868
_Scheduler_Initial_assignments, 883
_Scheduler_Is_non_preempt_mode_supported, 869
_Scheduler_Map_priority, 869
_Scheduler_Node_destroy, 869
_Scheduler_Node_do_initialize, 870
_Scheduler_Node_get_idle, 870
_Scheduler_Node_get_owner, 871
_Scheduler_Node_get_priority, 871
_Scheduler_Node_get_scheduler, 871
_Scheduler_Node_get_user, 872
_Scheduler_Node_initialize, 872
_Scheduler_Node_set_priority, 873
_Scheduler_Node_set_user, 873
_Scheduler_Node_size, 883
_Scheduler_Priority_and_sticky_update, 873
_Scheduler_Release_critical, 874
_Scheduler_Release_idle_thread, 874
_Scheduler_Release_job, 875
_Scheduler_Schedule, 875
_Scheduler_Set, 875
_Scheduler_Set_affinity, 876
_Scheduler_Set_idle_thread, 876
_Scheduler_Start_idle, 878
_Scheduler_Table, 884
_Scheduler_Thread_change_state, 878
_Scheduler_Tick, 878
_Scheduler_Try_to_schedule_node, 879
_Scheduler_Unblock, 880
_Scheduler_Unblock_node, 880
_Scheduler_Unmap_priority, 881
_Scheduler_Update_heir, 881
_Scheduler_Update_priority, 881
_Scheduler_Use_idle_thread, 882
_Scheduler_Yield, 882
_Scheduler_default_Ask_for_help, 857
_Scheduler_default_Cancel_job, 858
_Scheduler_default_Map_priority, 858
_Scheduler_default_Node_destroy, 859
_Scheduler_default_Node_initialize, 859
_Scheduler_default_Pin_or_unpin, 859
_Scheduler_default_Reconsider_help_request, 860
_Scheduler_default_Release_job, 860
_Scheduler_default_Schedule, 861
_Scheduler_default_Set_affinity, 861
_Scheduler_default_Set_affinity_body, 862
_Scheduler_default_Start_idle, 862
_Scheduler_default_Tick, 862
_Scheduler_default_Unmap_priority, 863
_Scheduler_default_Withdraw_node, 863
Scheduler_Context, 853
Scheduler_Get_idle_thread, 853
Scheduler_Node_request, 854
SCHEDULER_NODE_REQUEST_ADD, 854
SCHEDULER_NODE_REQUEST_NOT_PENDING, 854
SCHEDULER_NODE_REQUEST_NOTHING, 854
SCHEDULER_NODE_REQUEST_REMOVE, 854
SCHEDULER_OPERATION_DEFAULT_ASK_FOR_HELP, 853
Scheduler_Release_idle_thread, 854
scheduler.h
RTEMS_SCHEDULER_ASSIGN, 1601
Scheduler_Assignment, 1453
attributes, 1453

Scheduler_Context, 1454
Processors, 1454
Scheduler Handler, 853
Scheduler_EDF_Context, 1455
Ready, 1455
SCHEDULER_EDF_ENTRY_POINTS
EDF Scheduler, 305
Scheduler_EDF_Node, 1455
Node, 1456
SCHEDULER_EDF_PRIO_MSB
EDF Scheduler, 306
Scheduler_EDF_SMP_Context, 1456
Ready, 1457
SCHEDULER_EDF_SMP_ENTRY_POINTS
EDF Priority SMP Scheduler, 296
Scheduler_EDF_SMP_Node, 1457
ready_queue_index, 1457
Scheduler_EDF_SMP_Ready_queue, 1458
Scheduler_Get_idle_thread
Scheduler Handler, 853
Scheduler_Node, 1458
idle, 1460
next, 1460
Node, 1460
Priority, 1460
user, 1460
value, 1461
Scheduler_Node_request
Scheduler Handler, 854
SCHEDULER_NODE_REQUEST_ADD
Scheduler Handler, 854
SCHEDULER_NODE_REQUEST_NOT_PENDING
Scheduler Handler, 854
SCHEDULER_NODE_REQUEST_NOTHING
Scheduler Handler, 854
SCHEDULER_NODE_REQUEST_REMOVE
Scheduler Handler, 854
Scheduler_nodes
Thread_Scheduler_control, 1522
SCHEDULER_OPERATION_DEFAULT_ASK_FOR_HELP
Scheduler Handler, 853
Scheduler_Operations, 1461
add_processor, 1462
ask_for_help, 1463
block, 1463
cancel_job, 1463
initialize, 1464
map_priority, 1464
node_destroy, 1464
node_initialize, 1464
pin, 1465
reconsider_help_request, 1465
release_job, 1465
remove_processor, 1466
schedule, 1466
set_affinity, 1466
start_idle, 1467
tick, 1467
unblock, 1467
unmap_priority, 1467
unpin, 1468
update_priority, 1468
withdraw_node, 1468
yield, 1469
Scheduler_priority_Context, 1469
SCHEDULER_PRIORITY_ENTRY_POINTS
Deterministic Priority Scheduler, 258
Scheduler_priority_Node, 1470
Scheduler_priority_Ready_queue, 1470
Priority_map, 1470
ready_chain, 1471
Scheduler_Processor_removal_context, 1471
Scheduler_Release_idle_thread
Scheduler Handler, 854
Scheduler_simple_Context, 1471
SCHEDULER_SIMPLE_ENTRY_POINTS
Simple Priority Scheduler, 926
Scheduler_SMP_Context, 1472
Idle_threads, 1472
Scheduler_SMP_Node, 1473
SCHEDULER_SMP_NODE_BLOCKED
SMP Scheduler, 799
SCHEDULER_SMP_NODE_READY
SMP Scheduler, 800
SCHEDULER_SMP_NODE_SCHEDULED
SMP Scheduler, 800
Scheduler_SMP_Node_state
SMP Scheduler, 799
Score Timestamp, 885
_Timestamp_Add_to, 886
_Timestamp_Divide, 886
_Timestamp_Equal_to, 887
_Timestamp_Get_as_nanoseconds, 887
_Timestamp_Get_nanoseconds, 888
_Timestamp_Get_seconds, 888
_Timestamp_Greater_than, 889
_Timestamp_Less_than, 889
_Timestamp_Set, 890
_Timestamp_Set_to_zero, 890
_Timestamp_Subtract, 890
_Timestamp_To_timespec, 891
_Timestamp_To_timeval, 891
Timestamp_Control, 886
Sem_Control, 1473
Semaphore
Semaphore_Control, 1475
Semaphore Handler, 892
_CORE_semaphore_Acquire_critical, 893
_CORE_semaphore_Destroy, 893
_CORE_semaphore_Get_count, 894
_CORE_semaphore_Initialize, 894
_CORE_semaphore_Release, 894
_CORE_semaphore_Seize, 895
_CORE_semaphore_Surrender, 895
_Sem_Get, 896
_Sem_Queue_acquire_critical, 896

_Sem_Queue_release, 897
Semaphore Manager, 898
rtems_semaphore_create, 898
rtems_semaphore_delete, 900
rtems_semaphore_flush, 901
rtems_semaphore_ident, 901
rtems_semaphore_obtain, 902
rtems_semaphore_release, 902
rtems_semaphore_set_priority, 903
Semaphore Manager Implementation, 904
_Semaphore_Allocate, 906
_Semaphore_Free, 906
SEMAPHORE_INFORMATION_DEFINE, 905
Semaphore_Variant, 905
Semaphore_Control, 1474
Core_control, 1474
Mutex, 1474
Object, 1475
Semaphore, 1475
SEMAPHORE_INFORMATION_DEFINE
Semaphore Manager Implementation, 905
SEMAPHORE_KIND_MASK
semcreate.c, 1826
Semaphore_Variant
Semaphore Manager Implementation, 905
semcreate.c
SEMAPHORE_KIND_MASK, 1826
sequence
SMP_sequence_lock_Control, 1480
Serial Mouse, 907
set_affinity
Scheduler_Operations, 1466
set_attributes
rtems_termios_device_handler, 1437
sethi_of_handler_to_l4
CPU_Trap_table_entry, 1342
SHA256Context, 1475
Shared, 908, 909
Signal
RTEMS_API_Control, 1399
Signal Manager, 910
rtems_signal_catch, 911
rtems_signal_send, 912
Signals Implementation, 913
signals_pending
ASR_Information, 1312
signals_posted
ASR_Information, 1312
Signed 16-Bit Integer Checks, 914
Signed 32-Bit Integer Checks, 915
Signed 64-Bit Integer Checks, 916
Signed 8-Bit Integer Checks, 917
Signed Character Checks, 918
Signed Integer Checks, 919
Signed Long Integer Checks, 920
Signed Pointer Value Checks, 922
Signed Short Integer Checks, 923
Signed Size Checks, 924
Simple Priority Scheduler, 925
_Scheduler_simple_Block, 926
_Scheduler_simple_Extract, 927
_Scheduler_simple_Get_context, 927
_Scheduler_simple_Initialize, 927
_Scheduler_simple_Insert, 928
_Scheduler_simple_Priority_less_equal, 928
_Scheduler_simple_Schedule, 928
_Scheduler_simple_Schedule_body, 929
_Scheduler_simple_Unblock, 929
_Scheduler_simple_Update_priority, 930
_Scheduler_simple_Yield, 930
SCHEDULER_SIMPLE_ENTRY_POINTS, 926
since_last_period
rtems_rate_monotonic_period_status, 1428
size
Heap_Statistics, 1353
Stack_Control, 1488
Size Checks, 931
size_and_flag
Heap_Block, 1348
SMP Barriers, 777
_SMP_barrier_Control_initialize, 777
_SMP_barrier_State_initialize, 778
_SMP_barrier_Wait, 778
SMP Locks, 779
_SMP_MCS_lock_Acquire, 783
_SMP_MCS_lock_Destroy, 787
_SMP_MCS_lock_Do_acquire, 788
_SMP_MCS_lock_Initialize, 788
_SMP_MCS_lock_Release, 788
_SMP_lock_Acquire, 784
_SMP_lock_Acquire_inline, 784
_SMP_lock_Destroy, 781
_SMP_lock_Destroy_inline, 785
_SMP_lock_ISR_disable_and_acquire, 786
_SMP_lock_ISR_disable_and_acquire_inline, 786
_SMP_lock_Initialize, 781
_SMP_lock_Initialize_inline, 785
_SMP_lock_Release, 782
_SMP_lock_Release_and_ISR_enable, 782
_SMP_lock_Release_and_ISR_enable_inline, 786
_SMP_lock_Release_inline, 787
_SMP_lock_Set_name, 787
_SMP_sequence_lock_Destroy, 789
_SMP_sequence_lock_Initialize, 789
_SMP_sequence_lock_Read_begin, 789
_SMP_sequence_lock_Read_retry, 790
_SMP_sequence_lock_Write_begin, 790
_SMP_sequence_lock_Write_end, 791
_SMP_ticket_lock_Acquire, 783
_SMP_ticket_lock_Destroy, 791
_SMP_ticket_lock_Do_acquire, 791
_SMP_ticket_lock_Do_release, 792
_SMP_ticket_lock_Initialize, 792
_SMP_ticket_lock_Release, 783
SMP_TICKET_LOCK_INITIALIZER, 784
SMP Scheduler, 793

_Scheduler_SMP_Add_processor, 800
_Scheduler_SMP_Allocate_processor, 800
_Scheduler_SMP_Allocate_processor_exact, 801
_Scheduler_SMP_Allocate_processor_lazy, 801
_Scheduler_SMP_Ask_for_help, 802
_Scheduler_SMP_Block, 802
_Scheduler_SMP_Do_nothing_register_idle, 803
_Scheduler_SMP_Do_start_idle, 803
_Scheduler_SMP_Enqueue, 804
_Scheduler_SMP_Enqueue_scheduled, 805
_Scheduler_SMP_Enqueue_to_scheduled, 805
_Scheduler_SMP_Exctract_idle_thread, 806
_Scheduler_SMP_Extract_from_scheduled, 806
_Scheduler_SMP_Get_idle_thread, 806
_Scheduler_SMP_Get_lowest_scheduled, 807
_Scheduler_SMP_Get_self, 807
_Scheduler_SMP_Initialize, 808
_Scheduler_SMP_Insert_scheduled, 808
_Scheduler_SMP_Is_processor_owned_by_us, 808
_Scheduler_SMP_Node_change_state, 809
_Scheduler_SMP_Node_downcast, 809
_Scheduler_SMP_Node_initialize, 809
_Scheduler_SMP_Node_priority, 810
_Scheduler_SMP_Node_state, 810
_Scheduler_SMP_Node_update_priority, 811
_Scheduler_SMP_Preempt, 811
_Scheduler_SMP_Preempt_and_schedule_highest_ready, 811
_Scheduler_SMP_Priority_less_equal, 812
_Scheduler_SMP_Reconsider_help_request, 812
_Scheduler_SMP_Release_idle_thread, 813
_Scheduler_SMP_Remove_processor, 813
_Scheduler_SMP_Schedule_highest_ready, 814
_Scheduler_SMP_Set_affinity, 814
_Scheduler_SMP_Start_idle, 815
_Scheduler_SMP_Thread_get_node, 815
_Scheduler_SMP_Thread_get_own_node, 816
_Scheduler_SMP_Unblock, 816
_Scheduler_SMP_Update_priority, 816
_Scheduler_SMP_Withdraw_node, 817
_Scheduler_SMP_Yield, 818
SCHEDULER_SMP_NODE_BLOCKED, 799
SCHEDULER_SMP_NODE_READY, 800
SCHEDULER_SMP_NODE_SCHEDULED, 800
Scheduler_SMP_Node_state, 799
SMP Support, 819
_SMP_Broadcast_action, 821
_SMP_Fatal, 821
_SMP_Get_online_processors, 822
_SMP_Handler_initialize, 822
_SMP_Inter_processor_interrupt_handler, 822
_SMP_Multicast_action, 823
_SMP_Need_inter_processor_interrupts, 823
_SMP_Online_processors, 827
_SMP_Othercast_action, 823
_SMP_Processor_configured_maximum, 827
_SMP_Request_shutdown, 824
_SMP_Send_message, 824
_SMP_Send_message_broadcast, 825
_SMP_Send_message_multicast, 825
_SMP_Should_start_processor, 825
_SMP_Start_multitasking_on_secondary_processor, 826
_SMP_Synchronize, 826
_SMP_Unicast_action, 827
SMP_MESSAGE_PERFORM_JOBS, 820
SMP_MESSAGE_SHUTDOWN, 821
SMP_barrier_Control, 1476
SMP_barrier_State, 1476
SMP_lock_Context, 1477
SMP_lock_Control, 1477
SMP_MCS_lock_Context, 1477
locked, 1478
normal, 1478
SMP_MCS_lock_Control, 1479
normal, 1479
queue, 1479
SMP_MESSAGE_PERFORM_JOBS
SMP Support, 820
SMP_MESSAGE_SHUTDOWN
SMP Support, 821
SMP_Multicast_jobs, 1480
SMP_sequence_lock_Control, 1480
sequence, 1480
SMP_ticket_lock_Control, 1481
SMP_TICKET_LOCK_INITIALIZER
SMP Locks, 784
source_priority
MP_packet_Prefix, 1362
source_tid
MP_packet_Prefix, 1362
SPARC, 828, 834
CPU_INTERRUPT_FRAME_SIZE, 829
ISF_G1_OFFSET, 829
ISF_G2_OFFSET, 829
ISF_G3_OFFSET, 829
ISF_G4_OFFSET, 830
ISF_G5_OFFSET, 830
ISF_G7_OFFSET, 830
ISF_I0_OFFSET, 830
ISF_I1_OFFSET, 830
ISF_I2_OFFSET, 831
ISF_I3_OFFSET, 831
ISF_I4_OFFSET, 831
ISF_I5_OFFSET, 831
ISF_I6_FP_OFFSET, 831
ISF_I7_OFFSET, 832
ISF_NPC_OFFSET, 832
ISF_PC_OFFSET, 832
ISF_PSR_OFFSET, 832
ISF_TPC_OFFSET, 832
ISF_Y_OFFSET, 833
SPARC_MINIMUM_STACK_FRAME_SIZE, 833
SPARC Assembler Support, 835
RTRAP, 835

TRAP, 835
SPARC Context Structures, 837
_CPU_Context_Get_SP, 838
CONTEXT_CONTROL_FP_SIZE, 838
F12_F13_OFFSET, 839
F14_F15_OFFSET, 839
F16_F17_OFFSET, 839
F18_F19_OFFSET, 839
F1O_F11_OFFSET, 839
F22_F23_OFFSET, 840
F24_F25_OFFSET, 840
F26_F27_OFFSET, 840
F28_F29_OFFSET, 840
F2_F3_OFFSET, 840
F2O_F21_OFFSET, 841
F3O_F31_OFFSET, 841
F4_F5_OFFSET, 841
F6_F7_OFFSET, 841
F8_F9_OFFSET, 841
FO_F1_OFFSET, 842
FSR_OFFSET, 842
G5_OFFSET, 842
G7_OFFSET, 842
I0_OFFSET, 842
I1_OFFSET, 843
I2_OFFSET, 843
I3_OFFSET, 843
I4_OFFSET, 843
I5_OFFSET, 843
I6_FP_OFFSET, 844
I7_OFFSET, 844
ISR_DISPATCH_DISABLE_STACK_OFFSET, 844
L0_OFFSET, 844
L1_OFFSET, 844
L2_OFFSET, 845
L3_OFFSET, 845
L4_OFFSET, 845
L5_OFFSET, 845
L6_OFFSET, 845
L7_OFFSET, 846
O6_SP_OFFSET, 846
O7_OFFSET, 846
PSR_OFFSET, 846
sparc.h
CPU_MODEL_NAME, 1874
CPU_NAME, 1874
nop, 1874
SPARC_ASYNCHRONOUS_TRAP, 1874
sparc_disable_interrupts, 1885
sparc_enable_interrupts, 1885
sparc_flash_interrupts, 1875
sparc_get_interrupt_level, 1875
sparc_get_psr, 1876
sparc_get_tbr, 1876
sparc_get_wim, 1876
sparc_get_y, 1876
SPARC_HAS_BITSCAN, 1877
SPARC_HAS_FPU, 1877
SPARC_INTERRUPT_SOURCE_TO_TRAP, 1877
SPARC_INTERRUPT_TRAP_TO_SOURCE, 1878
SPARC_IS_INTERRUPT_TRAP, 1878
SPARC_LEON3FT_B2BST_NOP, 1879
SPARC_NUMBER_OF_REGISTER_WINDOWS, 1879
SPARC_PSR_CWP_BIT_POSITION, 1879
SPARC_PSR_CWP_MASK, 1879
SPARC_PSR_EC_BIT_POSITION, 1879
SPARC_PSR_EC_MASK, 1880
SPARC_PSR_EF_BIT_POSITION, 1880
SPARC_PSR_EF_MASK, 1880
SPARC_PSR_ET_BIT_POSITION, 1880
SPARC_PSR_ET_MASK, 1880
SPARC_PSR_ICC_BIT_POSITION, 1881
SPARC_PSR_ICC_MASK, 1881
SPARC_PSR_IMPL_BIT_POSITION, 1881
SPARC_PSR_IMPL_MASK, 1881
SPARC_PSR_PIL_BIT_POSITION, 1881
SPARC_PSR_PIL_MASK, 1882
SPARC_PSR_PS_BIT_POSITION, 1882
SPARC_PSR_PS_MASK, 1882
SPARC_PSR_S_BIT_POSITION, 1882
SPARC_PSR_S_MASK, 1882
SPARC_PSR_VER_BIT_POSITION, 1883
SPARC_PSR_VER_MASK, 1883
SPARC_REAL_TRAP_NUMBER, 1883
sparc_set_psr, 1883
sparc_set_tbr, 1884
sparc_set_wim, 1884
sparc_set_y, 1884
SPARC_SYNCHRONOUS_TRAP, 1885
sparc_syscall_exit, 1886
SPARC_ASSERT_FP_OFFSET
cpu.c, 1846
SPARC_ASSERT_ISF_OFFSET
cpu.c, 1846
SPARC_ASSERT_OFFSET
cpu.c, 1847
SPARC_ASYNCHRONOUS_TRAP
sparc.h, 1874
SPARC_Counter, 1481
sparc_disable_interrupts
sparc.h, 1885
sparc_enable_interrupts
sparc.h, 1885
sparc_flash_interrupts
sparc.h, 1875
sparc_get_interrupt_level
sparc.h, 1875
sparc_get_psr
sparc.h, 1876
sparc_get_tbr
sparc.h, 1876
sparc_get_wim
sparc.h, 1876
sparc_get_y
sparc.h, 1876

SPARC_HAS_BITSCAN
sparc.h, 1877
SPARC_HAS_FPU
sparc.h, 1877
SPARC_INTERRUPT_SOURCE_TO_TRAP
sparc.h, 1877
SPARC_INTERRUPT_TRAP_TO_SOURCE
sparc.h, 1878
SPARC_IS_INTERRUPT_TRAP
sparc.h, 1878
SPARC_LEON3FT_B2BST_NOP
sparc.h, 1879
SPARC_Minimum_stack_frame, 1482
i0, 1483
i1, 1483
i2, 1483
i3, 1483
i4, 1483
i5, 1483
i6_fp, 1484
i7, 1484
l0, 1484
l1, 1484
l2, 1484
l3, 1485
l4, 1485
l5, 1485
l6, 1485
l7, 1485
pad0, 1486
saved_arg0, 1486
saved_arg1, 1486
saved_arg2, 1486
saved_arg3, 1486
saved_arg4, 1487
saved_arg5, 1487
structure_return_address, 1487
SPARC_MINIMUM_STACK_FRAME_SIZE
SPARC, 833
SPARC_NUMBER_OF_REGISTER_WINDOWS
sparc.h, 1879
SPARC_PSR_CWP_BIT_POSITION
sparc.h, 1879
SPARC_PSR_CWP_MASK
sparc.h, 1879
SPARC_PSR_EC_BIT_POSITION
sparc.h, 1879
SPARC_PSR_EC_MASK
sparc.h, 1880
SPARC_PSR_EF_BIT_POSITION
sparc.h, 1880
SPARC_PSR_EF_MASK
sparc.h, 1880
SPARC_PSR_ET_BIT_POSITION
sparc.h, 1880
SPARC_PSR_ET_MASK
sparc.h, 1880
SPARC_PSR_ICC_BIT_POSITION
sparc.h, 1881
SPARC_PSR_ICC_MASK
sparc.h, 1881
SPARC_PSR_IMPL_BIT_POSITION
sparc.h, 1881
SPARC_PSR_IMPL_MASK
sparc.h, 1881
SPARC_PSR_PIL_BIT_POSITION
sparc.h, 1881
SPARC_PSR_PIL_MASK
sparc.h, 1882
SPARC_PSR_PS_BIT_POSITION
sparc.h, 1882
SPARC_PSR_PS_MASK
sparc.h, 1882
SPARC_PSR_S_BIT_POSITION
sparc.h, 1882
SPARC_PSR_S_MASK
sparc.h, 1882
SPARC_PSR_VER_BIT_POSITION
sparc.h, 1883
SPARC_PSR_VER_MASK
sparc.h, 1883
SPARC_REAL_TRAP_NUMBER
sparc.h, 1883
sparc_set_psr
sparc.h, 1883
sparc_set_tbr
sparc.h, 1884
sparc_set_wim
sparc.h, 1884
sparc_set_y
sparc.h, 1884
SPARC_SYNCHRONOUS_TRAP
sparc.h, 1885
sparc_syscall_exit
sparc.h, 1886
spec:/rtems/attr/val/attr, 1225
spec:/rtems/barrier/val/ident, 1226
spec:/rtems/event/req/send-receive, 1227
InterruptConfig, 1230
RtemsEventReqSendReceive_Fixture, 1230
RtemsEventReqSendReceive_PreDesc, 1230
RtemsEventReqSendReceive_PreDesc_Id, 1231
RtemsEventReqSendReceive_PreDesc_ReceiverState, 1231
RtemsEventReqSendReceive_PreDesc_Satisfy, 1231
RtemsEventReqSendReceive_PreDesc_Send, 1231
RtemsEventReqSendReceive_Run, 1229
spec:/rtems/event/val/event-constant, 1233
RtemsEventValEventConstant_Run, 1234
spec:/rtems/event/val/events, 1237
spec:/rtems/event/val/send-receive, 1238
spec:/rtems/event/val/system-send-receive, 1239
spec:/rtems/message/req/construct-errors, 1240
RtemsMessageReqConstructErrors_Fixture, 1242

RtemsMessageReqConstructErrors_PreDesc, 1242
RtemsMessageReqConstructErrors_PreDesc_Area, 1242
RtemsMessageReqConstructErrors_PreDesc_AreaSize, 1243
RtemsMessageReqConstructErrors_PreDesc_Id, 1243
RtemsMessageReqConstructErrors_PreDesc_MaxPending, 1243
RtemsMessageReqConstructErrors_PreDesc_MaxSize, 1243
RtemsMessageReqConstructErrors_PreDesc_Name, 1244
RtemsMessageReqConstructErrors_PreDesc_Queues, 1244
spec:/rtems/message/val/ident, 1245
spec:/rtems/mode/val/modes, 1246
spec:/rtems/option/val/options, 1247
spec:/rtems/part/req/create, 1248
RtemsPartReqCreate_Fixture, 1250
RtemsPartReqCreate_PreDesc, 1250
RtemsPartReqCreate_PreDesc_Id, 1250
RtemsPartReqCreate_PreDesc_Length, 1250
RtemsPartReqCreate_PreDesc_Name, 1251
RtemsPartReqCreate_PreDesc_Parts, 1251
RtemsPartReqCreate_PreDesc_Size, 1251
RtemsPartReqCreate_PreDesc_Start, 1251
spec:/rtems/part/req/delete, 1252
RtemsPartReqDelete_Fixture, 1253
RtemsPartReqDelete_PreDesc, 1254
RtemsPartReqDelete_PreDesc_Id, 1254
RtemsPartReqDelete_PreDesc_InUse, 1254
RtemsPartReqDelete_TransitionInfo, 1254
RtemsPartReqDelete_TransitionMap, 1255
spec:/rtems/part/req/get-buffer, 1256
RtemsPartReqGetBuffer_Fixture, 1257
RtemsPartReqGetBuffer_PreDesc, 1258
RtemsPartReqGetBuffer_PreDesc_Avail, 1258
RtemsPartReqGetBuffer_PreDesc_Buf, 1258
RtemsPartReqGetBuffer_PreDesc_Id, 1258
RtemsPartReqGetBuffer_TransitionInfo, 1259
RtemsPartReqGetBuffer_TransitionMap, 1259
spec:/rtems/part/req/return-buffer, 1260
RtemsPartReqReturnBuffer_Fixture, 1261
RtemsPartReqReturnBuffer_PreDesc, 1262
RtemsPartReqReturnBuffer_PreDesc_Buf, 1262
RtemsPartReqReturnBuffer_PreDesc_Id, 1262
RtemsPartReqReturnBuffer_TransitionInfo, 1262
RtemsPartReqReturnBuffer_TransitionMap, 1263
spec:/rtems/part/val/ident, 1264
spec:/rtems/part/val/part, 1265
spec:/rtems/ratemon/val/ident, 1266
spec:/rtems/req/ident, 1267
RtemsReqIdent_Fixture, 1269
RtemsReqIdent_PreDesc, 1269
RtemsReqIdent_PreDesc_Id, 1269
RtemsReqIdent_PreDesc_Name, 1269
RtemsReqIdent_PreDesc_Node, 1270
RtemsReqIdent_Run, 1268
spec:/rtems/req/ident-local, 1271
RtemsReqIdentLocal_Fixture, 1273
RtemsReqIdentLocal_PreDesc, 1273
RtemsReqIdentLocal_PreDesc_Id, 1273
RtemsReqIdentLocal_PreDesc_Name, 1273
RtemsReqIdentLocal_Run, 1272
RtemsReqIdentLocal_TransitionInfo, 1274
RtemsReqIdentLocal_TransitionMap, 1274
spec:/rtems/sem/val/ident, 1275
spec:/rtems/task/req/construct-errors, 1276
extensions, 1278
RtemsTaskReqConstructErrors_Fixture, 1278
RtemsTaskReqConstructErrors_PreDesc, 1279
RtemsTaskReqConstructErrors_PreDesc_Ext, 1279
RtemsTaskReqConstructErrors_PreDesc_Id, 1279
RtemsTaskReqConstructErrors_PreDesc_Name, 1279
RtemsTaskReqConstructErrors_PreDesc_Preempt, 1280
RtemsTaskReqConstructErrors_PreDesc_Prio, 1280
RtemsTaskReqConstructErrors_PreDesc_Stack, 1280
RtemsTaskReqConstructErrors_PreDesc_Tasks, 1280
RtemsTaskReqConstructErrors_PreDesc_TLS, 1281
spec:/rtems/task/req/ident, 1282
ClassicTaskIdentConfig, 1283
RtemsTaskReqIdent_Fixture, 1284
RtemsTaskReqIdent_PreDesc, 1284
RtemsTaskReqIdent_PreDesc_Pre, 1284
RtemsTaskReqIdent_TransitionInfo, 1284
RtemsTaskReqIdent_TransitionMap, 1285
spec:/rtems/timer/val/ident, 1286
spec:/rtems/userext/val/ident, 1287
spec:/testsuites/validation-0, 1288
actions, 1290
CONFIGURE_SCHEDULER_ASSIGNMENTS, 1290
CONFIGURE_SCHEDULER_TABLE_ENTRIES, 1290
runner_task_config, 1291
test_config, 1291
spec:/testsuites/validation/c-library, 1292
spec:/testsuites/validation/classic-barrier, 1293
spec:/testsuites/validation/profile, 1294
actions, 1295
task_config, 1295
test_config, 1295
Stack Handler, 932
_Stack_Allocate, 934
_Stack_Allocator_allocate, 938
_Stack_Allocator_avoids_workspace, 938
_Stack_Allocator_do_initialize, 935

_Stack_Allocator_free, 938
_Stack_Allocator_initialize, 938
_Stack_Ensure_minimum, 935
_Stack_Extend_size, 935
_Stack_Free, 936
_Stack_Free_nothing, 936
_Stack_Initialize, 936
_Stack_Is_enough, 937
_Stack_Minimum, 937
_Stack_Space_size, 938
rtems_minimum_stack_size, 938
Stack_Allocator_allocate, 933
Stack_Allocator_free, 934
Stack_Allocator_initialize, 934
STACK_MINIMUM_SIZE, 933
Stack_Allocator_allocate
Stack Handler, 933
Stack_Allocator_free
Stack Handler, 934
Stack_Allocator_initialize
Stack Handler, 934
Stack_Control, 1487
area, 1488
size, 1488
Stack_frame
CPU_Interrupt_frame, 1340
stack_free
Thread_Configuration, 1503
STACK_MINIMUM_SIZE
Stack Handler, 933
stack_size
rtems_initialization_tasks_table, 1413
Standard C Library Support, 939
malloc_free_space, 940
malloc_get_heap_pointer, 941
malloc_info, 941
malloc_set_heap_pointer, 941
RTEMS_NEWLIB_EXTENSION, 940
rtems_resource_snapshot_check, 941
rtems_resource_snapshot_equal, 942
rtems_resource_snapshot_take, 942
Start
_Thread_Control, 1304
start_idle
Scheduler_Operations, 1467
start_remote_tx
rtems_termios_device_flow, 1434
start_time
rtems_timer_information, 1445
Timer_Control, 1532
startloc
rtems_filesystem_eval_path_context_t, 1406
state
Per_CPU_Control, 1378
Rate_monotonic_Control, 1391
rtems_rate_monotonic_period_status, 1428
STATES_ALL_SET
Thread States, 1108
STATES_BLOCKED
Thread States, 1108
States_Control
Thread States, 1114
STATES_DEBUGGER
Thread States, 1109
STATES_DORMANT
Thread States, 1109
STATES_INTERRUPTIBLE_BY_SIGNAL
Thread States, 1109
STATES_LIFE_IS_CHANGING
Thread States, 1109
STATES_LOCALLY_BLOCKED
Thread States, 1110
STATES_READY
Thread States, 1110
STATES_SUSPENDED
Thread States, 1110
STATES_WAITING_FOR_BARRIER
Thread States, 1110
STATES_WAITING_FOR_BSD_WAKEUP
Thread States, 1111
STATES_WAITING_FOR_CONDITION_VARIABLE
Thread States, 1111
STATES_WAITING_FOR_EVENT
Thread States, 1111
STATES_WAITING_FOR_FUTEX
Thread States, 1111
STATES_WAITING_FOR_JOIN
Thread States, 1111
STATES_WAITING_FOR_JOIN_AT_EXIT
Thread States, 1112
STATES_WAITING_FOR_MESSAGE
Thread States, 1112
STATES_WAITING_FOR_MUTEX
Thread States, 1112
STATES_WAITING_FOR_PERIOD
Thread States, 1112
STATES_WAITING_FOR_RPC_REPLY
Thread States, 1112
STATES_WAITING_FOR_RWLOCK
Thread States, 1113
STATES_WAITING_FOR_SEGMENT
Thread States, 1113
STATES_WAITING_FOR_SEMAPHORE
Thread States, 1113
STATES_WAITING_FOR_SIGNAL
Thread States, 1113
STATES_WAITING_FOR_SYSTEM_EVENT
Thread States, 1113
STATES_ZOMBIE
Thread States, 1114
Statistics
Rate_monotonic_Control, 1391
statvfs, 1488
f_bavail, 1489
f_bfree, 1489
f_blocks, 1489

f_bsize, 1489
f_favail, 1489
f_ffree, 1490
f_files, 1490
f_flag, 1490
f_frsize, 1490
f_fsid, 1490
f_namemax, 1491
stop_remote_tx
rtems_termios_device_flow, 1434
stop_time
rtems_timer_information, 1445
Timer_Control, 1532
storage_area
rtems_interrupt_server_config, 1415
rtems_message_queue_config, 1421
rtems_task_config, 1431
storage_free
rtems_message_queue_config, 1421
rtems_task_config, 1432
storage_size
rtems_interrupt_server_config, 1415
rtems_task_config, 1432
String Checks, 944
structure_return_address
SPARC_Minimum_stack_frame, 1487
SuperCore, 945
Support Services, 948
rtems_is_name_valid, 950
RTEMS_MICROSECONDS_TO_TICKS, 948
RTEMS_MILLISECONDS_TO_MICROSECONDS, 949
RTEMS_MILLISECONDS_TO_TICKS, 949
rtems_name_to_characters, 950
rtems_workspace_allocate, 950
rtems_workspace_free, 951
rtems_workspace_get_information, 951
rtems_workspace_greedy_allocate, 951
rtems_workspace_greedy_allocate_all_except_largest, 952
rtems_workspace_greedy_free, 952
System State Handler, 953
_System_state_Get, 954
_System_state_Is_before_initialization, 954
_System_state_Is_before_multitasking, 955
_System_state_Is_terminated, 955
_System_state_Is_up, 955
_System_state_Set, 956
SYSTEM_STATE_BEFORE_INITIALIZATION, 954
SYSTEM_STATE_BEFORE_MULTITASKING, 954
System_state_Codes, 953
SYSTEM_STATE_TERMINATED, 954
SYSTEM_STATE_UP, 954
System_event
RTEMS_API_Control, 1399
SYSTEM_STATE_BEFORE_INITIALIZATION
System State Handler, 954
SYSTEM_STATE_BEFORE_MULTITASKING
System State Handler, 954
System_state_Codes
System State Handler, 953
SYSTEM_STATE_TERMINATED
System State Handler, 954
SYSTEM_STATE_UP
System State Handler, 954
systemeventreceive.c
rtems_event_system_receive, 1829
systemeventsend.c
rtems_event_system_send, 1830

t-test-interrupt.c
T_interrupt_fixture, 1797
T_interrupt_instance, 1797
t-test-thread-switch.c
T_thread_switch_instance, 1800
T_assert_false
Boolean Checks, 141
T_case_context, 1491
T_check_context, 1491
T_check_context_msg, 1492
T_config, 1492
T_context, 1493
T_destructor, 1493
T_false
Boolean Checks, 141
T_fixture, 1494
T_fixture_node, 1494
T_flags_eno
RTEMS Test Framework Implementation, 729
T_flags_eno_success
RTEMS Test Framework Implementation, 729
T_flags_eq
RTEMS Test Framework Implementation, 729
T_flags_eq_char
RTEMS Test Framework Implementation, 730
T_flags_eq_int
RTEMS Test Framework Implementation, 730
T_flags_eq_ll
RTEMS Test Framework Implementation, 730
T_flags_eq_long
RTEMS Test Framework Implementation, 731
T_flags_eq_mem
RTEMS Test Framework Implementation, 731
T_flags_eq_nstr
RTEMS Test Framework Implementation, 731
T_flags_eq_ptr
RTEMS Test Framework Implementation, 732
T_flags_eq_str
RTEMS Test Framework Implementation, 732
T_flags_eq_uint
RTEMS Test Framework Implementation, 732
T_flags_eq_ull
RTEMS Test Framework Implementation, 733
T_flags_eq_ulong
RTEMS Test Framework Implementation, 733
T_flags_ge_int
RTEMS Test Framework Implementation, 733

T_flags_ge_ll
RTEMS Test Framework Implementation, 734
T_flags_ge_long
RTEMS Test Framework Implementation, 734
T_flags_ge_uint
RTEMS Test Framework Implementation, 734
T_flags_ge_ull
RTEMS Test Framework Implementation, 735
T_flags_ge_ulong
RTEMS Test Framework Implementation, 735
T_flags_gt_int
RTEMS Test Framework Implementation, 735
T_flags_gt_ll
RTEMS Test Framework Implementation, 736
T_flags_gt_long
RTEMS Test Framework Implementation, 736
T_flags_gt_uint
RTEMS Test Framework Implementation, 736
T_flags_gt_ull
RTEMS Test Framework Implementation, 737
T_flags_gt_ulong
RTEMS Test Framework Implementation, 737
T_flags_le_int
RTEMS Test Framework Implementation, 737
T_flags_le_ll
RTEMS Test Framework Implementation, 738
T_flags_le_long
RTEMS Test Framework Implementation, 738
T_flags_le_uint
RTEMS Test Framework Implementation, 738
T_flags_le_ull
RTEMS Test Framework Implementation, 739
T_flags_le_ulong
RTEMS Test Framework Implementation, 739
T_flags_lt_int
RTEMS Test Framework Implementation, 739
T_flags_lt_ll
RTEMS Test Framework Implementation, 740
T_flags_lt_long
RTEMS Test Framework Implementation, 740
T_flags_lt_uint
RTEMS Test Framework Implementation, 740
T_flags_lt_ull
RTEMS Test Framework Implementation, 741
T_flags_lt_ulong
RTEMS Test Framework Implementation, 741
T_flags_ne
RTEMS Test Framework Implementation, 741
T_flags_ne_char
RTEMS Test Framework Implementation, 742
T_flags_ne_int
RTEMS Test Framework Implementation, 742
T_flags_ne_ll
RTEMS Test Framework Implementation, 742
T_flags_ne_long
RTEMS Test Framework Implementation, 743
T_flags_ne_mem
RTEMS Test Framework Implementation, 743
T_flags_ne_nstr
RTEMS Test Framework Implementation, 743
T_flags_ne_ptr
RTEMS Test Framework Implementation, 744
T_flags_ne_str
RTEMS Test Framework Implementation, 744
T_flags_ne_uint
RTEMS Test Framework Implementation, 744
T_flags_ne_ull
RTEMS Test Framework Implementation, 745
T_flags_ne_ulong
RTEMS Test Framework Implementation, 745
T_flags_not_null
RTEMS Test Framework Implementation, 745
T_flags_null
RTEMS Test Framework Implementation, 746
T_flags_psx_error
RTEMS Test Framework Implementation, 746
T_flags_psx_success
RTEMS Test Framework Implementation, 746
T_flags_true
RTEMS Test Framework Implementation, 747
T_interrupt_clock_time, 1494
T_interrupt_context, 1495
T_interrupt_fixture
t-test-interrupt.c, 1797
T_interrupt_instance
t-test-interrupt.c, 1797
T_interrupt_test_config, 1495
T_measure_runtime_config, 1496
T_measure_runtime_context, 1496
T_measure_runtime_request, 1497
T_putchar_string_context, 1497
T_quiet_false
Boolean Checks, 141
T_report_hash_sha256_context, 1497
T_step_assert_false
Boolean Checks, 142
T_step_false
Boolean Checks, 142
T_thread_switch_context, 1498
T_thread_switch_event, 1498
T_thread_switch_instance
t-test-thread-switch.c, 1800
T_thread_switch_log, 1499
T_thread_switch_log_10, 1499
T_thread_switch_log_2, 1499
T_thread_switch_log_4, 1500
T_VA_ARGS_KIND
RTEMS Test Framework Implementation, 747
tail
Per_CPU_Control, 1378
Task Manager, 957
rtems_scheduler_add_processor, 965
rtems_scheduler_get_maximum_priority, 966
rtems_scheduler_get_processor, 963
rtems_scheduler_get_processor_maximum, 964
rtems_scheduler_get_processor_set, 966

rtems_scheduler_ident, 967
rtems_scheduler_ident_by_processor, 967
rtems_scheduler_ident_by_processor_set, 968
rtems_scheduler_map_priority_from_posix, 969
rtems_scheduler_map_priority_to_posix, 969
rtems_scheduler_remove_processor, 970
rtems_task_argument, 965
rtems_task_construct, 970
rtems_task_create, 971
rtems_task_delete, 972
rtems_task_get_affinity, 973
rtems_task_get_priority, 973
rtems_task_get_scheduler, 973
rtems_task_ident, 974
rtems_task_is_suspended, 975
rtems_task_iterate, 975
rtems_task_mode, 975
rtems_task_restart, 976
rtems_task_resume, 976
rtems_task_set_affinity, 976
rtems_task_set_priority, 977
rtems_task_set_scheduler, 977
rtems_task_start, 977
RTEMS_TASK_STORAGE_ALIGNMENT, 964
RTEMS_TASK_STORAGE_SIZE, 964
rtems_task_suspend, 978
rtems_task_wake_after, 978
rtems_task_wake_when, 978
Task Modes, 980
RTEMS_INTERRUPT_LEVEL, 981
rtems_interrupt_level_body, 981
rtems_interrupt_mask, 982
Task Stack Allocator Configuration, 983
CONFIGURE_TASK_STACK_ALLOCATOR, 983
CONFIGURE_TASK_STACK_ALLOCATOR_AVOIDS_WORK_SPACE, 983
CONFIGURE_TASK_STACK_ALLOCATOR_INIT, 984
CONFIGURE_TASK_STACK_DEALLOCATOR, 984
CONFIGURE_TASK_STACK_FROM_ALLOCATOR, 985
task_config
spec:/testsuites/validation/profile, 1295
taskconstruct.c
_RTEMS_tasks_User_extensions, 1831
Termios, 986
rtems_termios_default_isig_handler, 988
RTEMS_TERMIOS_IPROC_CONTINUE, 987
RTEMS_TERMIOS_IPROC_DONE, 988
RTEMS_TERMIOS_IPROC_INTERRUPT, 988
rtems_termios_iproc_status_code, 987
rtems_termios_isig_handler, 987
rtems_termios_posix_isig_handler, 988
rtems_termios_register_isig_handler, 989
termiostypes.h
rtems_termios_baud_to_number, 1784
rtems_termios_device_context, 1783
rtems_termios_device_context_initialize, 1784
RTEMS_TERMIOS_DEVICE_CONTEXT_INITIALIZER, 1783
rtems_termios_device_install, 1784
rtems_termios_device_lock_acquire, 1785
rtems_termios_device_lock_release, 1785
rtems_termios_device_node, 1783
rtems_termios_get_device_context, 1786
rtems_termios_kqfilter, 1786
rtems_termios_mmap, 1786
rtems_termios_number_to_baud, 1786
rtems_termios_poll, 1787
rtems_termios_set_best_baud, 1787
rtems_termios_set_initial_baud, 1787
Test Suites, 990
Test Support, 991
rtems_test_begin, 993
rtems_test_busy_cpu_usage, 993
rtems_test_end, 993
rtems_test_parallel, 993
rtems_test_parallel_get_task_id, 994
rtems_test_parallel_is_master_worker, 994
rtems_test_parallel_stop_job, 995
rtems_test_parallel_worker_setup, 992
rtems_test_printf, 995
rtems_test_run, 995
test_config
spec:/testsuites/validation-0, 1291
spec:/testsuites/validation/profile, 1295
testrun.c
actions, 1801
config, 1801
testsuites/validation/tc-attr.c, 1932
testsuites/validation/tc-barrier-ident.c, 1932
testsuites/validation/tc-event-send-receive.c, 1932
testsuites/validation/tc-events.c, 1933
testsuites/validation/tc-message-construct-errors.c, 1933
testsuites/validation/tc-message-ident.c, 1935
testsuites/validation/tc-modes.c, 1936
testsuites/validation/tc-options.c, 1936
testsuites/validation/tc-part-create.c, 1936
testsuites/validation/tc-part-delete.c, 1938
testsuites/validation/tc-part-get.c, 1940
testsuites/validation/tc-part-ident.c, 1941
testsuites/validation/tc-part-return.c, 1942
testsuites/validation/tc-part.c, 1943
testsuites/validation/tc-ratemon-ident.c, 1944
testsuites/validation/tc-sem-ident.c, 1944
testsuites/validation/tc-space-profile.c, 1944
testsuites/validation/tc-task-construct-errors.c, 1944
testsuites/validation/tc-task-ident.c, 1947
testsuites/validation/tc-timer-ident.c, 1948
testsuites/validation/tc-userext-ident.c, 1949
testsuites/validation/tr-event-constant.c, 1949
testsuites/validation/tr-event-constant.h, 1949
testsuites/validation/tr-event-send-receive.c, 1949
testsuites/validation/tr-event-send-receive.h, 1952

testsuites/validation/tr-object-ident-local.c, 1953
testsuites/validation/tr-object-ident-local.h, 1954
testsuites/validation/tr-object-ident.c, 1954
testsuites/validation/tr-object-ident.h, 1956
testsuites/validation/ts-space-profile.c, 1957
testsuites/validation/ts-validation-0.c, 1958
the_class
    MP_packet_Prefix, 1362
    rtems_timer_information, 1446
    Timer_Control, 1532
the_error
    Internal_errors_Information, 1354
the_source
    Internal_errors_Information, 1354
Thread Handler, 997
    _Thread_Action_control_initialize, 1014
    _Thread_Action_initialize, 1014
    _Thread_Add_post_switch_action, 1014
    _Thread_Add_timeout_ticks, 1015
    _Thread_Allocate_unlimited, 1015
    _Thread_Cancel, 1015
    _Thread_Change_life, 1016
    _Thread_Clear_state, 1016
    _Thread_Clear_state_locked, 1016
    _Thread_Close, 1017
    _Thread_Continue, 1017
    _Thread_Control_add_on_count, 1066
    _Thread_Control_add_ons, 1066
    _Thread_Create_idle, 1018
    _Thread_Dispatch, 1018
    _Thread_Dispatch_direct, 1018
    _Thread_Dispatch_disable, 1019
    _Thread_Dispatch_disable_critical, 1019
    _Thread_Dispatch_disable_with_CPU, 1020
    _Thread_Dispatch_enable, 1020
    _Thread_Dispatch_get_disable_level, 1020
    _Thread_Dispatch_initialization, 1021
    _Thread_Dispatch_is_enabled, 1021
    _Thread_Dispatch_request, 1021
    _Thread_Dispatch_unnest, 1022
    _Thread_Dispatch_update_heir, 1022
    _Thread_Do_dispatch, 1022
    _Thread_Do_unpin, 1023
    _Thread_Entry_adaptor_idle, 1023
    _Thread_Entry_adaptor_numeric, 1023
    _Thread_Entry_adaptor_pointer, 1024
    _Thread_Exit, 1024
    _Thread_Get, 1024
    _Thread_Get_CPU, 1025
    _Thread_Get_CPU_time_used, 1025
    _Thread_Get_heir_and_make_it_executing, 1025
    _Thread_Get_maximum_internal_threads, 1026
    _Thread_Get_name, 1026
    _Thread_Get_objects_information, 1027
    _Thread_Get_priority, 1027
    _Thread_Get_unmapped_priority, 1028
    _Thread_Get_unmapped_real_priority, 1028
    _Thread_Global_constructor, 1067
    _Thread_Handler, 1028
    _Thread_Handler_initialization, 1029
    _Thread_Idle_body, 1067
    _Thread_Idle_stack_size, 1067
    _Thread_Idle_stacks, 1067
    _Thread_Initial_thread_count, 1068
    _Thread_Initialize, 1029
    _Thread_Initialize_information, 1030
    _Thread_Internal_allocate, 1030
    _Thread_Is_context_switch_necessary, 1030
    _Thread_Is_executing, 1031
    _Thread_Is_executing_on_a_processor, 1031
    _Thread_Is_heir, 1032
    _Thread_Is_joinable, 1032
    _Thread_Is_life_change_allowed, 1032
    _Thread_Is_life_changing, 1033
    _Thread_Is_life_restarting, 1033
    _Thread_Is_life_terminating, 1034
    _Thread_Is_ready, 1034
    _Thread_Iterate, 1034
    _Thread_Join, 1035
    _Thread_Kill_zombies, 1035
    _Thread_Load_environment, 1035
    _Thread_Maximum_TLS_size, 1068
    _Thread_Maximum_name_size, 1068
    _Thread_Pin, 1036
    _Thread_Priority_add, 1036
    _Thread_Priority_and_sticky_update, 1037
    _Thread_Priority_change, 1037
    _Thread_Priority_changed, 1038
    _Thread_Priority_highest, 1038
    _Thread_Priority_less_than, 1039
    _Thread_Priority_perform_actions, 1039
    _Thread_Priority_remove, 1041
    _Thread_Priority_replace, 1041
    _Thread_Priority_update, 1042
    _Thread_Remove_timer_and_unblock, 1042
    _Thread_Resource_count_decrement, 1043
    _Thread_Resource_count_increment, 1043
    _Thread_Restart_other, 1043
    _Thread_Restart_self, 1044
    _Thread_Restore_fp, 1044
    _Thread_Save_fp, 1044
    _Thread_Scheduler_acquire_critical, 1045
    _Thread_Scheduler_add_request, 1045
    _Thread_Scheduler_add_wait_node, 1046
    _Thread_Scheduler_cancel_need_for_help, 1046
    _Thread_Scheduler_get_home, 1046
    _Thread_Scheduler_get_home_node, 1047
    _Thread_Scheduler_get_node_by_index, 1047
    _Thread_Scheduler_process_requests, 1048
    _Thread_Scheduler_release_critical, 1048
    _Thread_Scheduler_remove_wait_node, 1048
    _Thread_Set_CPU, 1049
    _Thread_Set_life_protection, 1049
    _Thread_Set_name, 1049
    _Thread_Set_state, 1050
    _Thread_Set_state_locked, 1050

_Thread_Start, 1051
_Thread_Start_multitasking, 1051
_Thread_State_acquire, 1052
_Thread_State_acquire_critical, 1052
_Thread_State_acquire_for_executing, 1052
_Thread_State_release, 1053
_Thread_State_release_critical, 1053
_Thread_Timeout, 1053
_Thread_Timer_initialize, 1054
_Thread_Timer_insert_realtime, 1054
_Thread_Timer_remove, 1054
_Thread_Unblock, 1055
_Thread_Unpin, 1055
_Thread_Update_CPU_time_used, 1055
_Thread_Wait_acquire, 1056
_Thread_Wait_acquire_critical, 1056
_Thread_Wait_acquire_default, 1056
_Thread_Wait_acquire_default_critical, 1057
_Thread_Wait_acquire_default_for_executing, 1057
_Thread_Wait_acquire_queue_critical, 1058
_Thread_Wait_cancel, 1058
_Thread_Wait_claim, 1058
_Thread_Wait_claim_finalize, 1059
_Thread_Wait_flags_get, 1059
_Thread_Wait_flags_get_acquire, 1060
_Thread_Wait_flags_set, 1060
_Thread_Wait_flags_try_change_acquire, 1060
_Thread_Wait_flags_try_change_release, 1061
_Thread_Wait_get_id, 1062
_Thread_Wait_get_status, 1062
_Thread_Wait_release, 1062
_Thread_Wait_release_critical, 1063
_Thread_Wait_release_default, 1063
_Thread_Wait_release_default_critical, 1063
_Thread_Wait_release_queue_critical, 1064
_Thread_Wait_remove_request, 1064
_Thread_Wait_remove_request_locked, 1064
_Thread_Wait_restore_default, 1065
_Thread_Wait_tranquilize, 1065
_Thread_Yield, 1066
_Thread_queue_Heads_size, 1068
rtems_ada_self, 1068
rtems_iterate_over_all_threads, 1066
RTEMS_SCORE_ROBUST_THREAD_DISPATCH, 1007
Thread_Action_handler, 1009
THREAD_API_FIRST, 1007
THREAD_API_LAST, 1007
THREAD_API_POSIX, 1013
THREAD_API_RTEMS, 1013
Thread_APIs, 1012
Thread_Configured_control, 1011
Thread_CPU_budget_algorithm_callout, 1011
Thread_CPU_budget_algorithms, 1013
THREAD_DEFAULT_MAXIMUM_NAME_SIZE, 1007
Thread_Entry_numeric_type, 1011
THREAD_INFORMATION_DEFINE, 1008
THREAD_INFORMATION_DEFINE_ZERO, 1008
Thread_Life_state, 1013
THREAD_OF_SCHEDULER_HELP_NODE, 1009
Thread_queue_Configured_heads, 1011
THREAD_SCHEDULER_BLOCKED, 1013
THREAD_SCHEDULER_READY, 1013
THREAD_SCHEDULER_SCHEDULED, 1013
Thread_Scheduler_state, 1013
Thread_Wait_flags, 1012
THREAD_WAIT_STATE_INTEND_TO_BLOCK, 1009
THREAD_WAIT_STATE_READY_AGAIN, 1009
Thread Queue Handler, 1069
_Thread_queue_Acquire, 1079
_Thread_queue_Acquire_critical, 1080
_Thread_queue_Add_timeout_monotonic_timespec, 1080
_Thread_queue_Add_timeout_realtime_timespec, 1081
_Thread_queue_Add_timeout_ticks, 1081
_Thread_queue_Context_ISR_disable, 1074
_Thread_queue_Context_add_priority_update, 1081
_Thread_queue_Context_clear_priority_updates, 1082
_Thread_queue_Context_initialize, 1082
_Thread_queue_Context_restore_priority_updates, 1082
_Thread_queue_Context_save_priority_updates, 1083
_Thread_queue_Context_set_ISR_level, 1086
_Thread_queue_Context_set_MP_callout, 1074
_Thread_queue_Context_set_deadlock_callout, 1083
_Thread_queue_Context_set_enqueue_callout, 1084
_Thread_queue_Context_set_enqueue_do_nothing_extra, 1084
_Thread_queue_Context_set_enqueue_timeout_monotonic_timespe 1084
_Thread_queue_Context_set_enqueue_timeout_realtime_timespec, 1085
_Thread_queue_Context_set_enqueue_timeout_ticks, 1085
_Thread_queue_Context_set_thread_state, 1086
_Thread_queue_Context_set_timeout_argument, 1086
_Thread_queue_Context_set_timeout_ticks, 1088
_Thread_queue_Deadlock_fatal, 1088
_Thread_queue_Deadlock_status, 1088
_Thread_queue_Dequeue, 1074
_Thread_queue_Destroy, 1089
_Thread_queue_Dispatch_disable, 1089
_Thread_queue_Do_acquire_critical, 1089
_Thread_queue_Do_dequeue, 1090
_Thread_queue_Do_release_critical, 1090
_Thread_queue_Enqueue, 1091

_Thread_queue_Enqueue_do_nothing_extra, 1092
_Thread_queue_Enqueue_sticky, 1092
_Thread_queue_Extract, 1093
_Thread_queue_Extract_critical, 1093
_Thread_queue_Extract_locked, 1094
_Thread_queue_Extract_with_proxy, 1095
_Thread_queue_First, 1095
_Thread_queue_First_locked, 1096
_Thread_queue_Flush_critical, 1096
_Thread_queue_Flush_default_filter, 1097
_Thread_queue_Flush_status_object_was_deleted, 1097
_Thread_queue_Flush_status_unavailable, 1098
_Thread_queue_Gate_add, 1098
_Thread_queue_Gate_close, 1099
_Thread_queue_Gate_open, 1099
_Thread_queue_Gate_wait, 1099
_Thread_queue_Heads_initialize, 1100
_Thread_queue_Initialize, 1100
_Thread_queue_Is_empty, 1100
_Thread_queue_Object_initialize, 1101
_Thread_queue_Object_name, 1106
_Thread_queue_Path_acquire_critical, 1101
_Thread_queue_Path_release_critical, 1102
_Thread_queue_Queue_do_acquire_critical, 1102
_Thread_queue_Queue_get_name_and_id, 1102
_Thread_queue_Queue_initialize, 1103
_Thread_queue_Queue_release, 1103
_Thread_queue_Queue_release_critical, 1104
_Thread_queue_Release, 1104
_Thread_queue_Release_critical, 1104
_Thread_queue_Surrender, 1105
_Thread_queue_Surrender_sticky, 1105
_Thread_queue_Unblock_critical, 1106
Thread_queue_Deadlock_callout, 1076
Thread_queue_Enqueue_callout, 1076
Thread_queue_Enqueue_operation, 1077
Thread_queue_Extract_operation, 1077
Thread_queue_First_operation, 1077
Thread_queue_Flush_filter, 1078
Thread_queue_Heads, 1078
THREAD_QUEUE_INITIALIZER, 1075
THREAD_QUEUE_OBJECT_ASSERT, 1075
Thread_queue_Priority_actions_operation, 1079
THREAD_QUEUE_QUEUE_TO_OBJECT, 1075
Thread_queue_Surrender_operation, 1079
Thread States, 1107
_States_Clear, 1114
_States_Is_dormant, 1115
_States_Is_interruptible_by_signal, 1115
_States_Is_locally_blocked, 1116
_States_Is_ready, 1116
_States_Is_suspended, 1116
_States_Is_waiting_for_join_at_exit, 1117
_States_Is_waiting_for_rpc_reply, 1117
_States_Set, 1118
STATES_ALL_SET, 1108
STATES_BLOCKED, 1108
States_Control, 1114
STATES_DEBUGGER, 1109
STATES_DORMANT, 1109
STATES_INTERRUPTIBLE_BY_SIGNAL, 1109
STATES_LIFE_IS_CHANGING, 1109
STATES_LOCALLY_BLOCKED, 1110
STATES_READY, 1110
STATES_SUSPENDED, 1110
STATES_WAITING_FOR_BARRIER, 1110
STATES_WAITING_FOR_BSD_WAKEUP, 1111
STATES_WAITING_FOR_CONDITION_VARIABLE, 1111
STATES_WAITING_FOR_EVENT, 1111
STATES_WAITING_FOR_FUTEX, 1111
STATES_WAITING_FOR_JOIN, 1111
STATES_WAITING_FOR_JOIN_AT_EXIT, 1112
STATES_WAITING_FOR_MESSAGE, 1112
STATES_WAITING_FOR_MUTEX, 1112
STATES_WAITING_FOR_PERIOD, 1112
STATES_WAITING_FOR_RPC_REPLY, 1112
STATES_WAITING_FOR_RWLOCK, 1113
STATES_WAITING_FOR_SEGMENT, 1113
STATES_WAITING_FOR_SEMAPHORE, 1113
STATES_WAITING_FOR_SIGNAL, 1113
STATES_WAITING_FOR_SYSTEM_EVENT, 1113
STATES_ZOMBIE, 1114
Thread-Local Storage (TLS), 1119
_TLS_Alignment, 1124
_TLS_Copy_and_clear, 1120
_TLS_Get_allocation_size, 1120
_TLS_Get_size, 1120
_TLS_Get_thread_control_block_area_size, 1121
_TLS_Heap_align_up, 1121
_TLS_Initialize, 1122
_TLS_TCB_after_TLS_block_initialize, 1122
_TLS_TCB_at_area_begin_initialize, 1123
_TLS_TCB_before_TLS_block_initialize, 1123
thread.c
THREAD_OFFSET_ASSERT, 1914
Thread_Action, 1500
Thread_Action_control, 1501
Thread_Action_handler
Thread Handler, 1009
THREAD_API_FIRST
Thread Handler, 1007
THREAD_API_LAST
Thread Handler, 1007
THREAD_API_POSIX
Thread Handler, 1013
THREAD_API_RTEMS
Thread Handler, 1013
Thread_APIs
Thread Handler, 1012
Thread_Capture_control, 1501
Thread_Close_context, 1501
Thread_Configuration, 1502
stack_free, 1503

Thread_Configured_control
Thread Handler, 1011
Thread_Control_add_on, 1503
Thread_CPU_budget_algorithm_callout
Thread Handler, 1011
Thread_CPU_budget_algorithms
Thread Handler, 1013
THREAD_DEFAULT_MAXIMUM_NAME_SIZE
Thread Handler, 1007
Thread_Entry_idle, 1503
Thread_Entry_information, 1504
adaptor, 1504
Thread_Entry_numeric, 1505
Thread_Entry_numeric_type
Thread Handler, 1011
Thread_Entry_pointer, 1505
Thread_Information, 1506
Free, 1506
initial, 1506
THREAD_INFORMATION_DEFINE
Thread Handler, 1008
THREAD_INFORMATION_DEFINE_ZERO
Thread Handler, 1008
Thread_Join_context, 1507
Thread_Keys_information, 1507
Thread_Life_control, 1508
exit_value, 1508
Thread_Life_state
Thread Handler, 1013
THREAD_OF_SCHEDULER_HELP_NODE
Thread Handler, 1009
THREAD_OFFSET_ASSERT
thread.c, 1914
Thread_Proxy_control, 1509
current_state, 1509
Join_queue, 1509
Object, 1509
Timer, 1510
Wait, 1510
Thread_queue_Configured_heads
Thread Handler, 1011
Thread_queue_Context, 1510
deadlock_callout, 1511
enqueue_callout, 1512
Path, 1512
Timeout, 1512
update, 1512
Thread_queue_Control, 1513
Thread_queue_Deadlock_callout
Thread Queue Handler, 1076
Thread_queue_Enqueue_callout
Thread Queue Handler, 1076
Thread_queue_Enqueue_operation
Thread Queue Handler, 1077
Thread_queue_Extract_operation
Thread Queue Handler, 1077
Thread_queue_First_operation
Thread Queue Handler, 1077
Thread_queue_Flush_filter
Thread Queue Handler, 1078
Thread_queue_Gate, 1513
Thread_queue_Heads
Thread Queue Handler, 1078
THREAD_QUEUE_INITIALIZER
Thread Queue Handler, 1075
Thread_queue_Link, 1514
Thread_queue_Links, 1515
Thread_queue_Lock_context, 1515
Gate, 1515
Thread_queue_Object, 1516
THREAD_QUEUE_OBJECT_ASSERT
Thread Queue Handler, 1075
Thread_queue_Operations, 1516
enqueue, 1517
extract, 1517
Thread_queue_Priority_actions_operation
Thread Queue Handler, 1079
Thread_queue_Priority_queue, 1518
Node, 1518
Thread_queue_Queue, 1519
heads, 1519
Lock, 1519
THREAD_QUEUE_QUEUE_TO_OBJECT
Thread Queue Handler, 1075
Thread_queue_Surrender_operation
Thread Queue Handler, 1079
Thread_queue_Syslock_queue, 1520
THREAD_SCHEDULER_BLOCKED
Thread Handler, 1013
Thread_Scheduler_control, 1520
Help_node, 1521
helping_nodes, 1521
nodes, 1521
pin_level, 1521
requests, 1522
Scheduler_nodes, 1522
Wait_nodes, 1522
THREAD_SCHEDULER_READY
Thread Handler, 1013
THREAD_SCHEDULER_SCHEDULED
Thread Handler, 1013
Thread_Scheduler_state
Thread Handler, 1013
Thread_Start_information, 1523
budget_algorithm, 1523
budget_callout, 1524
Entry, 1524
initial_priority, 1524
Initial_stack, 1524
is_preemptible, 1524
isr_level, 1525
tls_area, 1525
Thread_Timer_information, 1525
Thread_Wait_flags
Thread Handler, 1012
Thread_Wait_information, 1526

count, 1526
Lock, 1527
operations, 1527
option, 1527
queue, 1528
return_argument, 1528
return_argument_second, 1528
return_code, 1528
Tranquilizer, 1529
Thread_Wait_information_Object_argument_type, 1529
THREAD_WAIT_STATE_INTEND_TO_BLOCK
Thread Handler, 1009
THREAD_WAIT_STATE_READY_AGAIN
Thread Handler, 1009
Thread_Zombie_control, 1530
threadqenqueue.c
_Thread_queue_Links, 1921
threadrestart.c
_Thread_Zombies, 1924
Threads_in_need_for_help
Per_CPU_Control, 1378
tick
Scheduler_Operations, 1467
Ticker
Timer_Control, 1532
ticks
Per_CPU_Control, 1379
Time of Day Handler, 1127
_TOD_Acquire, 1131
_TOD_Adjust, 1131
_TOD_Get, 1132
_TOD_Get_timeval, 1132
_TOD_Get_uptime, 1132
_TOD_Get_zero_based_uptime, 1133
_TOD_Get_zero_based_uptime_as_timespec, 1133
_TOD_Is_set, 1133
_TOD_Release, 1134
_TOD_Seconds_since_epoch, 1134
_TOD_Set, 1134
TOD_DAYS_PER_YEAR, 1128
TOD_HOURS_PER_DAY, 1129
TOD_MICROSECONDS_PER_SECOND, 1129
TOD_MILLISECONDS_PER_SECOND, 1129
TOD_MINUTES_PER_HOUR, 1129
TOD_MONTHS_PER_YEAR, 1129
TOD_NANOSECONDS_PER_MICROSECOND, 1130
TOD_NANOSECONDS_PER_SECOND, 1130
TOD_SECONDS_PER_DAY, 1130
TOD_SECONDS_PER_MINUTE, 1130
TOD_SECONDS_PER_NON_LEAP_YEAR, 1130
TOD_TICKS_PER_SECOND, 1131
TOD_TICKS_PER_SECOND_method, 1136
Time of Day Handler Action Hooks, 1137
_TOD_Hook_Register, 1138
_TOD_Hook_Run, 1138
_TOD_Hook_Unregister, 1138
TOD_Action, 1137
TOD_ACTION_SET_CLOCK, 1138
Time Services, 1125
Time Test 27 Support, 1126
time_period_initiated
Rate_monotonic_Control, 1391
timecounter, 1530
Timecounter Handler, 1140
_Timecounter_Acquire, 1141
_Timecounter_Bintime, 1142
_Timecounter_Binuptime, 1142
_Timecounter_Getbintime, 1142
_Timecounter_Getbinuptime, 1143
_Timecounter_Getboottime, 1143
_Timecounter_Getboottimebin, 1143
_Timecounter_Getmicrotime, 1145
_Timecounter_Getmicrouptime, 1145
_Timecounter_Getnanotime, 1145
_Timecounter_Getnanouptime, 1147
_Timecounter_Install, 1147
_Timecounter_Microtime, 1147
_Timecounter_Microuptime, 1148
_Timecounter_Nanotime, 1148
_Timecounter_Nanouptime, 1148
_Timecounter_Release, 1142
_Timecounter_Sbinuptime, 1148
_Timecounter_Set_clock, 1149
_Timecounter_Tick_simple, 1149
_Timecounter_Time_uptime, 1150
Timecounter Support, 1151
rtems_timecounter_install, 1152
RTEMS_TIMECOUNTER_QUALITY_CLOCK_DRIVER, 1152
rtems_timecounter_simple_downcounter_get, 1152
rtems_timecounter_simple_downcounter_tick, 1153
rtems_timecounter_simple_install, 1153
rtems_timecounter_simple_scale, 1154
rtems_timecounter_simple_upcounter_get, 1155
rtems_timecounter_simple_upcounter_tick, 1155
rtems_timecounter_tick, 1155
timehands, 1530
Timeout
Thread_queue_Context, 1512
timeout
MP_packet_Prefix, 1362
Timer
_Thread_Control, 1304
Rate_monotonic_Control, 1391
Thread_Proxy_control, 1510
Timer Manager, 1157
rtems_timer_cancel, 1158
rtems_timer_create, 1159
rtems_timer_delete, 1159
rtems_timer_fire_after, 1159
rtems_timer_fire_when, 1160
rtems_timer_get_information, 1160

rtems_timer_ident, 1160
rtems_timer_initiate_server, 1161
rtems_timer_reset, 1161
rtems_timer_server_fire_after, 1162
rtems_timer_server_fire_when, 1162
Timer_Classes, 1158
TIMER_DORMANT, 1158
TIMER_INTERVAL, 1158
TIMER_INTERVAL_ON_TASK, 1158
TIMER_TIME_OF_DAY, 1158
TIMER_TIME_OF_DAY_ON_TASK, 1158
Timer_Classes
Timer Manager, 1158
Timer_Control, 1531
initial, 1531
Object, 1531
routine, 1532
start_time, 1532
stop_time, 1532
the_class, 1532
Ticker, 1532
user_data, 1533
TIMER_DORMANT
Timer Manager, 1158
TIMER_INFORMATION_DEFINE
Classic Timer Implementation, 242
TIMER_INTERVAL
Timer Manager, 1158
TIMER_INTERVAL_ON_TASK
Timer Manager, 1158
Timer_server_Control, 1533
TIMER_TIME_OF_DAY
Timer Manager, 1158
TIMER_TIME_OF_DAY_ON_TASK
Timer Manager, 1158
Timestamp_Control
Score Timestamp, 886
timex, 1533
tls_area
Thread_Start_information, 1525
TLS_Dynamic_thread_vector, 1534
TLS_Index, 1534
TLS_Thread_control_block, 1534
tm27.h
Cause_tm27_intr, 1567
Install_tm27_vector, 1567
to_convert
MP_packet_Prefix, 1362
TOD_Action
Time of Day Handler Action Hooks, 1137
TOD_ACTION_SET_CLOCK
Time of Day Handler Action Hooks, 1138
TOD_BASE_YEAR
todimpl.h, 1771
TOD_Control, 1535
is_set, 1535
TOD_DAYS_PER_YEAR
Time of Day Handler, 1128
TOD_Hook, 1536
handler, 1536
Node, 1536
TOD_HOURS_PER_DAY
Time of Day Handler, 1129
TOD_MICROSECONDS_PER_SECOND
Time of Day Handler, 1129
TOD_MILLISECONDS_PER_SECOND
Time of Day Handler, 1129
TOD_MINUTES_PER_HOUR
Time of Day Handler, 1129
TOD_MONTHS_PER_YEAR
Time of Day Handler, 1129
TOD_NANOSECONDS_PER_MICROSECOND
Time of Day Handler, 1130
TOD_NANOSECONDS_PER_SECOND
Time of Day Handler, 1130
TOD_SECONDS_1970_THROUGH_1988
todimpl.h, 1771
TOD_SECONDS_PER_DAY
Time of Day Handler, 1130
TOD_SECONDS_PER_MINUTE
Time of Day Handler, 1130
TOD_SECONDS_PER_NON_LEAP_YEAR
Time of Day Handler, 1130
TOD_TICKS_PER_SECOND
Time of Day Handler, 1131
TOD_TICKS_PER_SECOND_method
Time of Day Handler, 1136
todimpl.h
TOD_BASE_YEAR, 1771
TOD_SECONDS_1970_THROUGH_1988, 1771
token
rtems_filesystem_eval_path_context_t, 1406
tokenlen
rtems_filesystem_eval_path_context_t, 1407
total_cpu_time
Rate_monotonic_Statistics, 1393
rtems_rate_monotonic_period_statistics, 1426
total_wall_time
Rate_monotonic_Statistics, 1394
rtems_rate_monotonic_period_statistics, 1427
tpc
CPU_Interrupt_frame, 1340
Tracing, 1163
Tranquilizer
Thread_Wait_information, 1529
TRAP
SPARC Assembler Support, 835
ttywakeup, 1537

UART, 1164
unallocated
rtems_object_api_class_information, 1423
unblock
Scheduler_Operations, 1467
unmap_priority
Scheduler_Operations, 1467
unmount

File System Types and Mount, 373
unmount_task
rtems_filesystem_mount_table_entry_tt, 1410
unpin
Scheduler_Operations, 1468
Unsigned 16-Bit Integer Checks, 1165
Unsigned 32-Bit Integer Checks, 1166
Unsigned 64-Bit Integer Checks, 1167
Unsigned 8-Bit Integer Checks, 1168
Unsigned Character Checks, 1169
Unsigned Integer Checks, 1170
Unsigned Long Integer Checks, 1171
Unsigned Long Long Integer Checks, 1172
Unsigned Pointer Value Checks, 1173
Unsigned Short Integer Checks, 1174
update
Thread_queue_Context, 1512
update_priority
Scheduler_Operations, 1468
user
Scheduler_Node, 1460
User Extension Handler, 1175
_User_extensions_Acquire, 1183
_User_extensions_Add_API_set, 1184
_User_extensions_Add_set, 1184
_User_extensions_Add_set_with_table, 1184
_User_extensions_Destroy_iterators, 1184
_User_extensions_Fatal, 1185
_User_extensions_Fatal_visitor, 1185
_User_extensions_Initial_count, 1192
_User_extensions_Initial_extensions, 1192
_User_extensions_Initial_switch_controls, 1192
_User_extensions_Iterate, 1185
_User_extensions_Release, 1186
_User_extensions_Remove_set, 1186
_User_extensions_Thread_begin, 1186
_User_extensions_Thread_begin_visitor, 1187
_User_extensions_Thread_create, 1187
_User_extensions_Thread_create_visitor, 1188
_User_extensions_Thread_delete, 1188
_User_extensions_Thread_delete_visitor, 1188
_User_extensions_Thread_exitted, 1189
_User_extensions_Thread_exitted_visitor, 1189
_User_extensions_Thread_restart, 1189
_User_extensions_Thread_restart_visitor, 1190
_User_extensions_Thread_start, 1190
_User_extensions_Thread_start_visitor, 1190
_User_extensions_Thread_switch, 1191
_User_extensions_Thread_terminate, 1191
_User_extensions_Thread_terminate_visitor, 1191
User_extensions_fatal_extension, 1178
User_extensions_Iterator, 1178
User_extensions_thread_begin_extension, 1178
User_extensions_thread_create_extension, 1179
User_extensions_thread_delete_extension, 1179
User_extensions_thread_exitted_extension, 1181
User_extensions_thread_restart_extension, 1181
User_extensions_thread_start_extension, 1181
User_extensions_thread_switch_extension, 1182
User_extensions_thread_terminate_extension, 1182
User_extensions_Visitor, 1183
User Extensions Implementation, 1193
EXTENSION_INFORMATION_DEFINE, 1193
User Extensions Manager, 1195
rtems_extension_create, 1196
rtems_extension_delete, 1196
rtems_extension_ident, 1196
user_data
Timer_Control, 1533
User_extensions_Control, 1537
User_extensions_Fatal_context, 1537
User_extensions_fatal_extension
User Extension Handler, 1178
User_extensions_Iterator, 1538
User Extension Handler, 1178
User_extensions_List, 1538
User_extensions_Switch_control, 1539
User_extensions_Table, 1539
User_extensions_thread_begin_extension
User Extension Handler, 1178
User_extensions_Thread_create_context, 1540
User_extensions_thread_create_extension
User Extension Handler, 1179
User_extensions_thread_delete_extension
User Extension Handler, 1179
User_extensions_thread_exitted_extension
User Extension Handler, 1181
User_extensions_thread_restart_extension
User Extension Handler, 1181
User_extensions_thread_start_extension
User Extension Handler, 1181
User_extensions_thread_switch_extension
User Extension Handler, 1182
User_extensions_thread_terminate_extension
User Extension Handler, 1182
User_extensions_Visitor
User Extension Handler, 1183
User_initialization_tasks_table
rtems_api_configuration_table, 1398

value
Scheduler_Node, 1461
Version, 1198
rtems_board_support_package, 1198
rtems_version, 1198
rtems_version_control_key, 1199
rtems_version_control_key_is_valid, 1199
rtems_version_major, 1199
rtems_version_minor, 1200
rtems_version_revision, 1200
vprintk
bsplo.h, 1576
vprintk.c, 1794
vprintk.c
vprintk, 1794

Wait
_Thread_Control, 1304
Thread_Proxy_control, 1510
Wait_nodes
Thread_Scheduler_control, 1522
Wait_queue
CORE_barrier_Control, 1329
CORE_message_queue_Control, 1332
CORE_mutex_Control, 1333
CORE_semaphore_Control, 1335
Watchdog Handler, 1201
_Watchdog_Cancel, 1206
_Watchdog_Do_tickle, 1206
_Watchdog_Future_timespec, 1207
_Watchdog_Get_CPU, 1207
_Watchdog_Get_state, 1208
_Watchdog_Header_destroy, 1208
_Watchdog_Header_first, 1208
_Watchdog_Header_initialize, 1209
_Watchdog_Initialize, 1209
_Watchdog_Insert, 1209
_Watchdog_Is_far_future_timespec, 1210
_Watchdog_Is_scheduled, 1210
_Watchdog_Is_valid_interval_timespec, 1211
_Watchdog_Is_valid_timespec, 1211
_Watchdog_Microseconds_per_tick, 1217
_Watchdog_Nanoseconds_per_tick, 1217
_Watchdog_Next_first, 1211
_Watchdog_Per_CPU_acquire_critical, 1212
_Watchdog_Per_CPU_insert, 1212
_Watchdog_Per_CPU_insert_ticks, 1213
_Watchdog_Per_CPU_release_critical, 1213
_Watchdog_Per_CPU_remove, 1213
_Watchdog_Per_CPU_remove_ticks, 1214
_Watchdog_Preinitialize, 1214
_Watchdog_Remove, 1215
_Watchdog_Set_CPU, 1215
_Watchdog_Set_state, 1215
_Watchdog_Tick, 1216
_Watchdog_Ticks_from_sbintime, 1216
_Watchdog_Ticks_from_seconds, 1216
_Watchdog_Ticks_from_timespec, 1217
_Watchdog_Ticks_per_second, 1217
_Watchdog_Ticks_per_timeslice, 1218
_Watchdog_Ticks_since_boot, 1218
WATCHDOG_BITS_FOR_1E9_NANOSECONDS, 1204
WATCHDOG_INACTIVE, 1206
WATCHDOG_INITIALIZER, 1204
Watchdog_Interval, 1205
WATCHDOG_MAX_SECONDS, 1204
WATCHDOG_PENDING, 1206
WATCHDOG_SCHEDULED_BLACK, 1206
WATCHDOG_SCHEDULED_RED, 1206
Watchdog_Service_routine, 1205
Watchdog_Service_routine_entry, 1205
Watchdog_State, 1205
WATCHDOG_BITS_FOR_1E9_NANOSECONDS
Watchdog Handler, 1204
Watchdog_Control, 1540
Watchdog_Header, 1541
WATCHDOG_INACTIVE
Watchdog Handler, 1206
WATCHDOG_INITIALIZER
Watchdog Handler, 1204
Watchdog_Interval
Watchdog Handler, 1205
WATCHDOG_MAX_SECONDS
Watchdog Handler, 1204
WATCHDOG_PENDING
Watchdog Handler, 1206
WATCHDOG_SCHEDULED_BLACK
Watchdog Handler, 1206
WATCHDOG_SCHEDULED_RED
Watchdog Handler, 1206
Watchdog_Service_routine
Watchdog Handler, 1205
Watchdog_Service_routine_entry
Watchdog Handler, 1205
Watchdog_State
Watchdog Handler, 1205
withdraw_node
Scheduler_Operations, 1468
Workspace Handler, 1219
_Workspace_Allocate, 1220
_Workspace_Area, 1222
_Workspace_Free, 1220
_Workspace_Handler_initialization, 1220
_Workspace_Is_unified, 1222
_Workspace_Malloc_initialize_separate, 1221
_Workspace_Malloc_initialize_unified, 1221
_Workspace_Malloc_initializer, 1222
_Workspace_Size, 1222
_Workspace_String_duplicate, 1221
write
rtems_termios_device_handler, 1438
write_entry
rtems_driver_address_table, 1404

y
CPU_Interrupt_frame, 1340
yield
Scheduler_Operations, 1469