9

Timing Data

9.1 Introduction

The timing data for the i960CA version of **RTEMS** is provided along with the target dependent aspects concerning the gathering of the timing data. The hardware platform used to gather the times is described to give the reader a better understanding of each directive time provided. Also, provided is a description of the interrupt latency and the context switch times as they pertain to the i960CA version of **RTEMS**.

9.2 Hardware Platform

All times reported except for the maximum period interrupts are disabled by **RTEMS** were measured using a Cyclone Microsystems CVME961 board. The CVME961 is a 33 Mhz board with dynamic RAM which has two wait state dynamic memory (four CPU cycles) for read accesses and one wait state (two CPU cycles) for write accesses. The Z8536 on a SQUALL SQSIO4 mezzanine board was used to measure elapsed time with one-half microsecond resolution. All sources of hardware interrupts are disabled, although the interrupt level of the i960CA allows all interrupts.

The maximum interrupt disable period was measured by summing the number of CPU cycles required by each assembly language instruction executed while interrupts were disabled. Zero wait state memory was assumed. The total CPU cycles executed with interrupts disabled, including the instructions to disable and enable interrupts, was divided by 33 to simulate a i960CA executing at 33 Mhz with zero wait states.

9.3 Interrupt Latency

The maximum period with interrupts disabled within **RTEMS** is less than TBD microseconds including the instructions which disable and re-enable interrupts. The time required for the i960CA to generate an interrupt using the **sysctl** instruction, vectoring to an interrupt handler, and for the **RTEMS** entry overhead before invoking the user's interrupt handler are a total of 13 microseconds. These combine to yield a worst case interrupt latency of less TBD + 13.0 microseconds.

It should be noted again that the maximum period with interrupts disabled within **RTEMS** is hand-timed. The interrupt vector and entry overhead time was generated on the Cyclone CVME961 benchmark platform using the **sysctl** instruction as the interrupt source.

9.4 Context Switch

The **RTEMS** processor context switch time is 19 microseconds on the Cyclone CVME961 benchmark platform. This time represents the raw context switch time with no user extensions configured. Additional execution time is required when a **TSWITCH** user extension is configured. The use of the **TSWITCH** extension is application dependent. Thus, its execution time is not considered part of the base context switch time.

The i960CA has no hardware floating point capability and floating point tasks are not supported.

The following table summarizes the context switch times for the CVME961 benchmark platform:

Description	Microseconds
No Floating Point Contexts	19
Floating Point Contexts	
restore first FP task	na
save initialized, restore initialized	na
save idle, restore initialized	na
save idle, restore idle	na

9.5 Directive Times

This sections is divided into a number of subsections, each of which contains a table listing the execution times of that manager's directives.

9.6 Task Manager

Directive	Microseconds
TASK_CREATE	32
TASK_IDENT	21
TASK_START	41
TASK_RESTART	
calling task	59
suspended task -- returns to caller	52
blocked task -- returns to caller	75
ready task -- returns to caller	57
suspended task -- preempts caller	71
blocked task -- preempts caller	93
ready task -- preempts caller	74
TASK_DELETE	
calling task	70
suspended task	41
blocked task	46
ready task	47
TASK_SUSPEND	
calling task	41
returns to caller	18
TASK_RESUME	
task readied -- returns to caller	20
task readied -- preempts caller	52
TASK_SET_PRIORITY	
obtain current priority	9
returns to caller	29
preempts caller	62
TASK_MODE	
obtain current mode	3
no reschedule	3
reschedule -- returns to caller	13
reschedule -- preempts caller	37
TASK_GET_NOTE	10
TASK_SET_NOTE	11
TASK_WAKE_AFTER	
yield -- returns to caller	5
yield -- preempts caller	30
TASK_WAKE_WHEN	79

9.7 Interrupt Manager

It should be noted that the interrupt entry times include vectoring the interrupt handler.

Directive	Microseconds
Interrupt Entry Overhead	
returns to nested interrupt	16
returns to interrupted task	10
returns to preempting task	9
Interrupt Exit Overhead	
returns to nested interrupt	<1
returns to interrupted task	12
returns to preempting task	40

9.8 Clock Manager

Directive	Microseconds
CLOCK_SET	48
CLOCK_GET	<1
CLOCK_TICK	12

9.9 Timer Manager

Directive	Microseconds
TIMER_CREATE	9
TIMER_IDENT	20
TIMER_DELETE	
inactive	16
active	18
TIMER_FIRE_AFTER	
inactive	38
active	39
TIMER_FIRE_WHEN	
inactive	52
active	53
TIMER_RESET	
inactive	29
active	28
TIMER_CANCEL	
inactive	16
active	14

9.10 Semaphore Manager

Directive	Microseconds
SEMAPHORE_CREATE	30
SEMAPHORE_IDENT	20
SEMAPHORE_DELETE	38
SEMAPHORE_OBTAIN	
available	18
not available -- NO_WAIT	15
not available -- caller blocks	76
SEMAPHORE_RELEASE	
no waiting tasks	17
task readied -- returns to caller	37
task readied -- preempts caller	72

9.11 Message Manager

Directive	Microseconds
MESSAGE_QUEUE_CREATE	29
MESSAGE_QUEUE_IDENT	20
MESSAGE_QUEUE_DELETE	37
MESSAGE_QUEUE_SEND	
no waiting tasks	37
task readied -- returns to caller	37
task readied -- preempts caller	74
MESSAGE_QUEUE_URGENT	
no waiting tasks	39
task readied -- returns to caller	37
task readied -- preempts caller	74
MESSAGE_QUEUE_BROADCAST	
no waiting tasks	22
task readied -- returns to caller	53
task readied -- preempts caller	87
MESSAGE_QUEUE_RECEIVE	
available	19
not available -- NO_WAIT	13
not available -- caller blocks	66
MESSAGE_QUEUE_FLUSH	
no messages flushed	18
messages flushed	23

9.12 Event Manager

Directive	Microseconds
EVENT_SEND	
no task readied	9
task readied -- returns to caller	28
task readied -- preempts caller	64
EVENT_RECEIVE	
obtain current events	<1
available	17
not available -- NO_WAIT	6
not available -- caller blocks	50

9.13 Signal Manager

Directive	Microseconds
SIGNAL_CATCH	12
SIGNAL_SEND	
returns to caller	25
signal to self	30
EXIT ASR OVERHEAD	
returns to calling task	24
returns to preempting task	46

9.14 Partition Manager

Directive	Microseconds
PARTITION_CREATE	35
PARTITION_IDENT	20
PARTITION_DELETE	27
PARTITION_GET_BUFFER	
available	24
not available	19
PARTITION_RETURN_BUFFER	27

9.15 Region Manager

Directive	Microseconds
REGION_CREATE	33
REGION_IDENT	20
REGION_DELETE	24
REGION_GET_SEGMENT	
available	30
not available -- NO_WAIT	30
not available -- caller blocks	76
REGION_RETURN_SEGMENT	
no waiting tasks	33
task readied -- returns to caller	72
task readied -- preempts caller	105

9.16 Dual-Ported Memory Manager

Directive	Microseconds
PORT_CREATE	19
PORT_IDENT	20
PORT_DELETE	24
PORT_INTERNAL_TO_EXTERNAL	8
PORT_EXTERNAL_TO_INTERNAL	11

9.17 I/O Manager

Directive	Microseconds
IO_INITIALIZE	2
IO_OPEN	4
IO_CLOSE	2
IO_READ	2
IO_WRITE	2
IO_CONTROL	2

9.18 Rate Monotonic Manager

Directive	Microseconds
RATE_MONOTONIC_CREATE	20
RATE_MONOTONIC_IDENT	20
RATE_MONOTONIC_CANCEL	28
RATE_MONOTONIC_DELETE	
active	32
inactive	29
RATE_MONOTONIC_PERIOD	
initiate period -- returns to caller	33
conclude period -- caller blocks	54
obtain status	20