6

Processor Dependent Information Table

6.1 Introduction

Any highly processor dependent information required to describe a processor to **RTEMS** is provided in the **CPU Dependent Information Table**. This table is not required for all processors supported by **RTEMS**. This chapter describes the contents, if any, for a particular processor type.

6.2 CPU Dependent Information Table

The PA-RISC version of the **RTEMS CPU Dependent Information Table** contains the information required to interface a Board Support Package and **RTEMS** on the PA-RISC. This information is provided to allow **RTEMS** to interoperate effectively with the BSP. The **C** structure definition is given here:

```
struct cpu_configuration_table {
    void       (*pretasking_hook)( void );
    void       (*predriver_hook)( void );
    void       (*postdriver_hook)( void );
    void       (*idle_task)( void );
    boolean      do_zero_of_workspace;
    unsigned32   interrupt_stack_size;
    unsigned32   extra_system_initialization_stack;

    /*
     *  Number of entries in the list of external interrupts used.
     */

    unsigned32   external_interrupts;
    unsigned32   external_interrupt[HPPA_EXTERNAL_INTERRUPTS];

    /*
     *  Optional user handler for spurious interrupts.  The first argument
     *  (mask) is the bits left in the EIRR after all interrupts which
     *  have a handler have been processed.  The second argument
     *  (interrupt_frame) points to the interrupt frame on the stack.
     *  It is assumed that this handler will dismiss all interrupts
     *  reported as spurious in the mask.
     */

    void       (*spurious_handler)( rtems_unsigned32, void* );
};
```

pretasking_hook is the address of the user provided routine which is invoked once **RTEMS** initialization is complete but before interrupts and tasking are enabled. This field may be **NULL** to indicate that the hook is not utilized.

predriver_hook is the address of the user provided routine which is invoked with tasking enabled immediately before the MPCI and device drivers are initialized. **RTEMS** initialization is complete, interrupts and tasking are enabled, but no device drivers are intialized. This field may be **NULL** to indicate that the hook is not utilized.

postdriver_hook is the address of the user provided routine which is invoked with tasking enabled immediately after the MPCI and device drivers are initialized. **RTEMS** initialization is complete, interrupts and tasking are enabled, and the device drivers are intialized. This field may be **NULL** to indicate that the hook is not utilized.

idle_task	is the address of the optional user provided routine which is used as the system's IDLE task. If this field is not NULL, then the **RTEMS** default IDLE task is not used. This field may be **NULL** to indicate that the default IDLE is to be used.
do_zero_of_workspace	indicates whether **RTEMS** should zero the Workspace as part of its initialization. If set to **TRUE**, the Workspace is zeroed. Otherwise, it is not.
interrupt_stack_size	is the size of the **RTEMS** allocated interrupt stack in bytes. This value must be at least as large as **MINIMUM_STACK_SIZE**.
extra_system_ initialization_stack	is the extra stack space allocated for the **RTEMS** system initialization task in bytes. The system initialization task invokes the MPCI and device driver initialization routines and may require extra stack space on some targets.
external_interrupts	is the number of external interrupts (vectored through interrupt four) which are utilized by this application.
external_interrupt	is an array whose entries are integers which indicate the external interrupts utilized by this application and the order in which the **RTEMS** external interrupt handler should service them. Any external interrupts not listed specifically will be handled by the **spurious_handler**.
spurious_handler	is the address of a user provided handler which is invoked when a spurious interrupt is encountered. If this field is **NULL**, then the default **RTEMS** spurious handler is used. This handler simply clears the interrupt in an attempt to dismiss it.