9

Timing Data

9.1 Introduction

The timing data for the PA-RISC 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 MC68020 version of **RTEMS**.

9.2 Hardware Platform

No directive execution times are reported for the PA-RISC because the target platform was proprietary and executions times could not be released.

9.3 Interrupt Latency

The maximum period with interrupts disabled within **RTEMS** is less than **XXX** instructions including the instructions which disable and re-enable interrupts. The time required for the PA-RISC is not included in this analysis. The **RTEMS** entry overhead before invoking the user's interrupt handler requires **XXX** instructions. These combine to yield a worst case interrupt latency of less than **XXX** instructions.

It should be noted again that the maximum period with interrupts disabled within **RTEMS** for the PA-RISC is only an instruction count.

9.4 Context Switch

The **RTEMS** processor context switch time is **XXX** instructions for the PA-RISC when no floating point context switch is saved or restored. Saving and restoring the floating point context adds approximately **XXX** instructions to the context switch procedure. Additional execution time is required when a **TASK_SWITCH** user extension is configured. The use of the **TASK_SWITCH** extension is application dependent. Thus, its execution time is not considered part of the raw context switch time.

Since **RTEMS** was designed specifically for embedded missile applications which are floating point intensive, the executive is optimized to avoid unnecessarily saving and restoring the state of the numeric coprocessor. On many processors, the state of the numeric coprocessor is only saved when an **FLOATING_POINT** task is dispatched and that task was not the last task to utilize the coprocessor. In a system with only one **FLOATING_POINT** task, the state of the numeric coprocessor will never be saved or restored. When the first **FLOATING_POINT** task is dispatched, **RTEMS** does not need to save the current state of the numeric coprocessor. As discussed in the **Register Usage** section, on the PA-RISC the every task is considered to be floating point registers and , as a rsule, every context switch involves saving and restoring the state of the floating point unit.

The following table summarizes the context switch times for the PA-RISC processor:

Description	Instructions
No Floating Point Contexts	XXX
Floating Point Contexts	
restore first FP task	XXX
save initialized, restore initialized	XXX
save idle, restore initialized	XXX
save idle, restore idle	XXX

9.5 Directive Times

No execution times are available for the PA-RISC because the target platform was proprietary and no timing information could be released.