3

Interrupt Processing

3.1 Introduction

Different types of processors respond to the occurence of an interrupt in its own unique fashion. In addition, each processor type provides a control mechanism to allow for the proper handling of an interrupt. The processor dependent response to the interrupt modifies the current execution state and results in a change in the execution stream. Most processors require that an interrupt handler utilize some special control mechanisms to return to the normal processing stream. Although **RTEMS** hides many of the processor dependent details of interrupt processing, it is important to understand how the **RTEMS** interrupt manager is mapped onto the processor's unique architecture. Discussed in this chapter are the MC68020's interrupt response and control mechanisms as they pertain to **RTEMS**.

3.2 Vectoring of Interrupt Handler

Upon receipt of an interrupt the MC68020 automatically performs the following actions:

- *saves the current status register (**SR**),*
- *clears the master/interrupt (**M**) bit of the **SR** to indicate the switch from master state to interrupt state,*
- *sets the privilege mode to supervisor,*
- *suppresses tracing,*

- *sets the interrupt mask level equal to the level of the interrupt being serviced,*
- *pushes an interrupt stack frame (**ISF**), which includes the program counter (**PC**), the status register (**SR**), and the format/exception vector offset (**FVO**) word, onto the supervisor and interrupt stacks,*
- *switches the current stack to the interrupt stack and vectors to an interrupt service routine (**ISR**). If the **ISR** was installed with the **interrupt_catch** directive, then thewill begin execution. The **RTEMS** interrupt handler saves all registers which are not preserved according to the calling conventions and invokes the application's **ISR**.*

A nested interrupt is processed similarly by the MC68020 with the exception that only a single **ISF** is placed on the interrupt stack and the current stack need not be switched.

The **FVO** word in the Interrupt Stack Frame is examined by **RTEMS** to determine when an outer most interrupt is being exited. Since the **FVO** is used by **RTEMS** for this purpose, the user application code **MUST NOT** modify this field.

3.3 Interrupt Stack Frame

Status Register		0x0
Program Counter High		0x2
Program Counter Low		0x4
Format	**Vector Offset**	0x6

3.4 Interrupt Levels

Eight levels (0-7) of interrupt priorities are supported by the MC68020 microprocessor with level seven (7) being the highest priority. Level zero (0) indicates that interrupts are fully enabled. Interrupt requests for interrupts with priorities less than or equal to the current interrupt mask level are ignored.

Although **RTEMS** supports 256 interrupt levels, the MC68020 only supports eight. **RTEMS** interrupt levels 0 through 7 directly correspond to MC68020 interrupt levels. All other **RTEMS** interrupt levels are undefined and their behavior is unpredictable.

3.5 Disabling of Interrupts by RTEMS

During the execution of directive calls, critical sections of code may be executed. When these sections are encountered, **RTEMS** disables interrupts to level seven (7) before the execution of this section and restores them to the previous level upon completion of the section. **RTEMS** has been optimized to insure that interrupts are disabled for less than **13.5** microseconds on a 20.0 Mhz MC68020 with zero wait states. These numbers will vary based the number of wait states and processor speed present on the target board.

Non-maskable interrupts (**NMI**) cannot be disabled, and ISRs which execute at this level **MUST NEVER** issue **RTEMS** system calls. If a directive is invoked, unpredictable results may occur due to the inability of **RTEMS** to protect its critical sections. However, ISRs that make no system calls may safely execute as non-maskable interrupts.

3.6 Interrupt Stack

On the MC68020, **RTEMS** allocates the interrupt stack from the **Workspace Area**. The amount of memory allocated for the interrupt stack is determined by the **interrupt_stack_size** field in the **CPU Configuration Table**. During the initialization process, **RTEMS** will install its interrupt stack.