SECTION 3

Interrupts

3.1 INTRODUCTION

Hardware event signals (interrupts) are the lowest level of mechanisms in the Explorer virtual architecture. An interrupt, therefore, cannot rely on any higher level to perform its work. Hence, interrupts are serviced by the lowest level of the microcode implementation without reference to virtual memory, garbage collected memory, or macro instructions. (Virtual memory service for memory map handling is considered very low level and is permitted in some interrupt contexts.)

An interrupt is caused by an I/O device requesting service, another processor signalling an event, or by system bus and internal processor errors. In order to simplify interaction between interrupts and higher levels, interrupts are not automatically processed, but are polled at convenient times by higher levels of the implementation. When an interrupt is noticed by polling, control transfers to an appropriate handler for the interrupt.

Interrupts communicate with higher levels of the system via shared memory in the form of flags in internal processor memories and shared memory. The shared memory must not be garbage collected and must be wired (so no page fault can occur when accessing it) and fixed (so it cannot be moved by the garbage collector). In other words, neither the garbage collector nor the page fault handler can be invoked by the interrupt handler.

A check for pending interrupts is made at most virtual memory operations, especially instruction fetch. Interrupts are also checked at other times during internal processing (e.g. during disk paging waits). As a result, interrupt response time, while usually within a few microseconds, has no guaranteed maximum. Interrupt handlers must, therefore, handle the case that the response was too slow if it could cause problems.

Interrupts are the primary means for external events to signal the Lisp system. In most cases, interrupts set flags for higher level processing to notice or move data between the I/O device and an I/O buffer in wired memory. However, certain time-critical processing may be best performed as part of the interrupt handler.

3.2 INTERRUPTS ON THE EXPLORER PROCESSOR

The Explorer processor has hardware to ease the detection and processing of interrupts. Since the Explorer processor is NuBus based, most interrupt are events signaled over the system bus by writing a word or byte with the low order bit set into special locations in the control space of the Explorer processor. A map of the interrupt locations and the priority of each is shown in Table 3-1.

Table 3-1 Explorer Control Space

NuBus Address (Hex)	Interrupt Priority Level (Decimal)	
FsE0003C	15	(Lowest)
FsE00038	14	
FsE00034	13	
FsE00030	12	
FsE0002C	11	
FsE00028	10	
FsE00024	9	
FsE00020	8	
FsE0001C	7	
FsE00018	6	
FsE00014	5	
FsE00010	4	
FsE0000C	3	
FsE00008	2	(Highest)
FsE00004	1	(Preemptive) Boot request
FsE00000	0	(Preemptive) Powerfail

Interrupt pending is a condition testable individually and in combination with the page fault and sequence break conditions for jump and abbreviated jump microinstructions. Microcode tests whether there is an interrupt pending by performing a conditional call to the interrupt service routine if the interrupt pending condition is true.

The interrupt service routine will process all interrupts before returning to the caller. Interrupts are, of course, processed from highest priority to lowest. Interrupt priority is linked to the location in the control space of the processor as shown above in Table 3-1. The highest priority level with an interrupt pending is indicated by a special field in the machine control register (MCR) of the Explorer, or the Pending Event register at

IO space address #x3A000000 on the Explorer II. On any level with several devices, all devices that could interrupt to that level must be polled. For each interrupt level, there is a list of device descriptor blocks. The device descriptor block is shown in Figure 3-1.

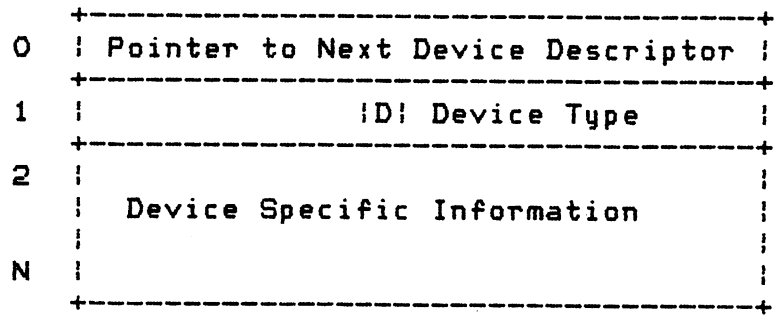

Figure 3-1 Device Descriptor Block

The lists are anchored by the interrupt vector table. The interrupt vector table is indexed by interrupt priority and contains a pointer to the interrupt descriptor block list for all the devices on that interrupt vector or priority. The value zero indicates the null link or empty list. The interrupt vector table is shown in Table 3-2.

Table 3-2 Interrupt Vectors

Interrupt Priority Level	
0	Powerfail (empty vector)
1	Boot request (empty vector)
2	Device descriptor block address
.	
.	
.	
15	Device descriptor block address

Once the highest priority interrupt level has been determined, the event request is cleared by writing a word with the low order bit set to zero into the word of the processor control space that corresponds to the interrupt level of the device. Interrupt levels 0 and 1 are dedicated to Powerfail and Boot request

respectively. These events are handled as aborts, i.e. the processor traps directly to processing of the event. If either of these events are detected as polled interrupt events then the processor failed to trap. The interrupt processor will cause the machine to halt.

For levels 2 to 15 the interrupt service routine uses the highest priority level with an interrupt pending to index into the interrupt vector table. This yields the list of device descriptor blocks for this interrupt level. The interrupt processor then traverses this list. For each block in this list the interrupt type is extracted from word 1 and the specific handler for this interrupt type is called. This interrupt type specific handler is responsible for determining if the device pertaining to this descriptor block has requested interrupt processing, and if so, performing that processing. When this interrupt handler returns, the next block in the list is examined, etc., until the end of the list is reached.

When the end of the list has been reached the interrupt pending condition is tested. If no interrupts are pending, the interrupt processing is complete and the interrupt service routine returns to its caller. If an interrupt is pending, the entire process is repeated.