CHAPTER 4
INTERRUPTS

INTRODUCTION

The Z8^{PLUS} core allows 15 different interrupts from a variety of sources:

- external inputs
- on-chip peripherals
- software

Interrupts can be masked by using the Interrupt Mask Register. All interrupts can be globally disabled by setting the master Interrupt Enable, bit 7 in the Interrupt Mask Register, to 0, with a Disable Interrupt (DI) instruction. Interrupts are globally enabled by setting bit 7 to 1 with an Enable Interrupt (EI) instruction.

There are four interrupt control registers: the Interrupt Request Registers (IREQ and IREQ2) and the Interrupt Mask registers (IMASK and IMASK2). Figure 4-1 shows addresses and identifiers for the interrupt control registers. Figure 4-2 is a block diagram showing the Interrupt Mask and Interrupt Priority logic.

Register	HEX	Identifier
Interrupt Mask	0FBH	IMASK
Interrupt Request	0FAH	IREQ
Interrupt Mask 2	0F9H	IMASK2
Interrupt Request 2	0F8H	IREQ2

Figure 4-1. Interrupt Control Register Addresses and Identifiers

The Z8PLUS MCU family supports both vectored and polled interrupt handling. Details on vectored and polled interrupts can be found later in this chapter.

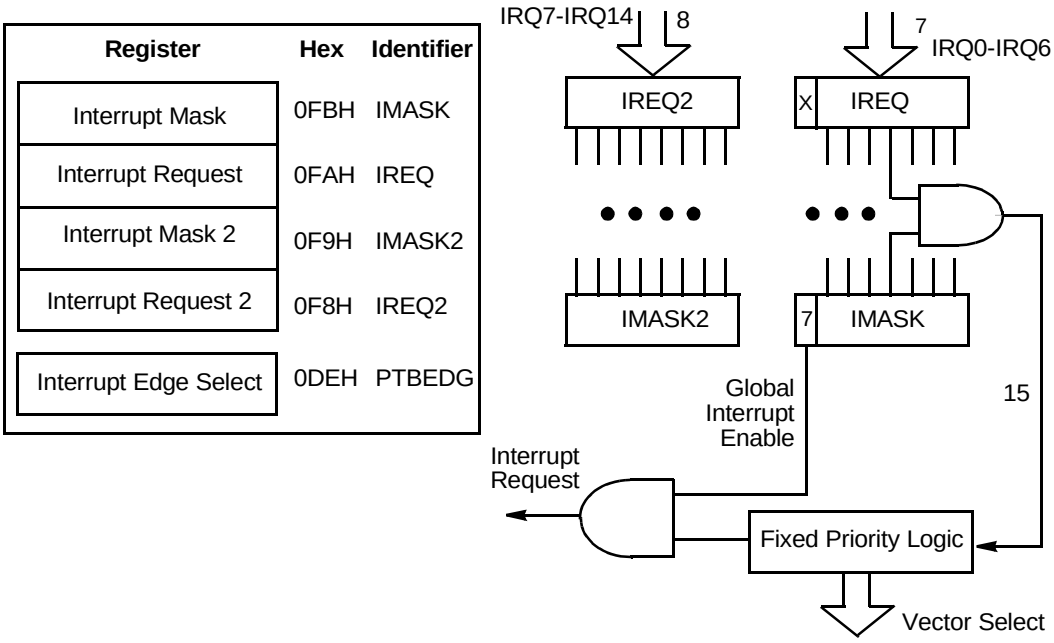

Figure 4-2. Interrupt Block Diagram

NOTE: See the selected Z8PLUS MCU's product specification for the exact interrupt sources supported.

INTERRUPT SOURCES

Table 4-1 presents the interrupt types, sources, and vectors available in the Z8E001. Other processors from the Z8PLUS family may define the interrupts differently.

Table 4-1. Z8E001 Interrupt Types, Sources, and Vectors

Name	Sources	Vector Location	Comments	Fixed Priority
$IREQ_0$	Timer0 Time-out	2,3	Internal	1 (Highest)
$IREQ_1$	PB4 High-to-Low Transition	4,5	External (PB4), Edge Triggered	2
$IREQ_2$	Timer1 Time-out	6,7	Internal	3
$IREQ_3$	PB2 High-to-Low Transition	8,9	External (PB2), Edge Triggered	4
$IREQ_4$	PB4 Low-to-High Transition	A,B	External (PB4), Edge Triggered	5
$IREQ_5$	Timer2 Time-out	C,D	Internal	6 (Lowest)
$IREQ_6$ - $IREQ_{15}$	Reserved		Reserved for future expansion	

External Interrupt Sources

External sources can be generated by a transition on the corresponding Port pin. The interrupt may detect a rising edge, a falling edge, or both.

NOTES:

1. The interrupt sources and trigger conditions are device dependent. See the device product specification to determine available sources (internal and external), triggering edge options, and exact programming details.

2. Although interrupts are edge triggered, minimum interrupt request Low and High times must be observed for proper operation. See the device product specification for exact timing requirements on external interrupt requests (T_WIL, T_WIH).

Internal Interrupt Sources

Internal interrupt sources and trigger conditions are device dependent. On-chip peripherals may set interrupt under various conditions. Some peripherals always set their corresponding IREQ bit while others must be specifically configured to do so.

See the device product specification to determine available sources, triggering edge options, and exact programming details. For more details on the interrupt sources, refer to the chapters describing the timers, comparators, I/O ports, and other peripherals.

INTERRUPT REQUEST (IREQ) REGISTER LOGIC AND TIMING

The Z8PLUS core responds to interrupts as it retires each instruction. If an unmasked interrupt is detected as an instruction is being retired, the Z8PLUS core does not execute an instruction during the next instruction cycle. The Z8PLUS MCU instead selects the highest priority outstanding interrupt to be serviced. The program counter and flags register are pushed to the stack during the next instruction cycle. The appropriate IREQ bit is cleared, the master enable is cleared and the MCU fetches the interrupt vector from program memory. It then jumps to the user's interrupt routine during the following cycle (See Figure 4-3).

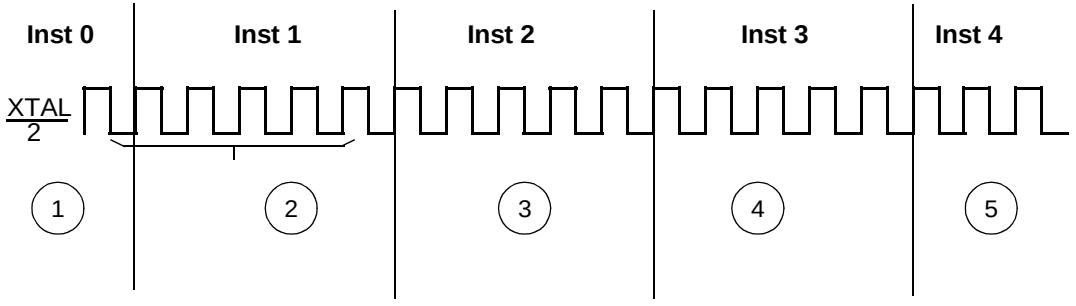

Figure 4-3. Interrupt Service Sequence

NOTES:

1. There are no outstanding, unmasked interrupts.
2. Interrupt source sets an IREQ bit during this interval. This bit is highest priority, has an unmasked IREQ, and is bit-sampled.
3. PC and flags are pushed, IREQ bit cleared, IMASK (7) cleared, and vector fetched.
4. JUMP to interrupt vector.
5. This portion is the first instruction of user's interrupt service routine.

Interrupt Mask Register (IMASK) Initialization

The IMASK register individually or globally enables or disables the interrupts (see Figure 4-4). When bits 0 through bit 6 are set to 1, the corresponding interrupt requests are enabled. The IMASK2 register, bits 0 through 7, enable and disable IRQ7 through IRQ14, respectively. Bit 7 is the master enable bit and must be set before any of the individual interrupt requests can be recognized. Resetting bit 7 disables all the interrupt requests. Bit 7 is set and reset by the EI and DI instructions. It is automatically set to 0 during an interrupt service routine and set to 1 following the execution of an Interrupt Return (IRET) instruction. The IMASK registers are reset to 00H, disabling all interrupts.

NOTE:

1. It is not good programming practice to directly aqssign a value to the master enable bit. A value change should always be accomplished by issuing the EI and DI instructions.

2. Care should be taken not to set or clear IMASK bits while the master enable is set.

Figure 4-4. Interrupt Mask Register

Interrupt Mask Register–IMASK (FBH)

Bit	7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
R = Read W = Write X = Indeterminate U = Undefined/Undetermined								

Bit Position	R/W	Value	Description
7		0 1	Disables Interrupts Enables Interrupts
6		0 1	Disables IRQ5 Enables IRQ5
5		0 1	Disables IRQ5 Enables IRQ5
4		0 1	Disables IRQ4 Enables IRQ4
3		0 1	Disables IRQ3 Enables IRQ3
2		0 1	Disables IRQ2 Enables IRQ2
1		0 1	Disables IRQ1 Enables IRQ1
0		0 1	Disables IRQ0 Enables IRQ0

Figure 4-5. Interrupt Mask 2 Register

Interrupt Mask 2 Register–IMASK2 (F9H)

Bit	7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
R = Read W = Write X = Indeterminate U = Undefined/Undetermined								

Bit Position	R/W	Value	Description
7	R/W	0 1	Disables IRQ14 Enables IRQ14
6	R/W	0 1	Disables IRQ13 Enables IRQ13
5	R/W	0 1	Disables IRQ12 Enables IRQ12
4	R/W	0 1	Disables IRQ11 Enables IRQ11
3	R/W	0 1	Disables IRQ10 Enables IRQ10
2	R/W	0 1	Disables IRQ9 Enables IRQ9
1	R/W	0 1	Disables IRQ8 Enables IRQ8
0	R/W	0 1	Disables IRQ7 Enables IRQ7

Interrupt Request (IREQ) Register Initialization

IREQ (see Figure 4-6) is a register that stores the interrupt requests for both vectored and polled interrupts. When an interrupt is issued, the corresponding bit position in the register is set to 1. Bit 0 to bit 5 are assigned to interrupt requests IREQ0 to IREQ5, respectively.

Whenever RESET is executed, the IREQ resister is set to 00H.

Figure 4-6. Interrupt Request Register.

Interrupt Request Register–IREQ (FAH)

Bit	7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
R = Read W = Write X = Indeterminate U = Undefined/Undetermined								

Bit Position	R/W	Value	Description
7	R/W	0	Reserved,must be 0
6	R/W	0 1	IRQ6 reset IRQ6 set
5	R/W	0 1	IRQ5 reset IRQ5 set
4	R/W	0 1	IRQ4 reset IRQ4 set
3	R/W	0 1	IRQ3 reset IRQ3 set
2	R/W	0 1	IRQ2 reset IRQ2 set
1	R/W	0 1	IRQ1 reset IRQ1 set
0	R/W	0 1	IRQ0 reset IRQ0 set

Figure 4-7. Interrupt Request Register 2

Interrupt Request Register 2–IREQ2 (F8H)

Bit	7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0
R = Read W = Write X = Indeterminate U = Undefined/Undetermined								

Bit Position	R/W	Value	Description
7	R/W	0 1	IRQ14 reset IRQ14 set
6	R/W	0 1	IRQ13 reset IRQ13 set
5	R/W	0 1	IRQ12 reset IRQ12 set
4	R/W	0 1	IRQ11 reset IRQ11 set
3	R/W	0 1	IRQ10 reset IRQ10 set
2	R/W	0 1	IRQ9 reset IRQ9 set
1	R/W	0 1	IRQ8 reset IRQ8 set
0	R/W	0 1	IRQ7 reset IRQ7 set

IREQ SOFTWARE INTERRUPT GENERATION

IREQ can be used to generate software interrupts by specifying IREQ as the destination of any instruction referencing the Z8PLUS Standard Register File. These software interrupts (SWI) are controlled in the same manner as hardware generated requests. In other words, the IMASK controls the enabling of each SWI.

To generate a SWI, the request bit in IREQ is set by the following statement:

```
OR IREQ,#NUMBER
```

The immediate data variable, NUMBER, has a 1 in the bit position corresponding to the required level of SWI. For example, an SWI must be issued when an IREQ5 occurs. Bit 5 of NUMBER must have a value of 1.

```
OR IREQ, #00100000B
```

If the interrupt system is globally enabled, IREQ5 is enabled, and there are no higher priority requests pending , control is transferred to the service routine pointed to by the IREQ5 vector.

NOTE: Note that software may modify the IREQ register at any time. Care should be taken when using any instruction that modifies the IREQ register while interrupt sources are active. The software writeback always takes precedence over the hardware. If a software writeback takes place on the same cycle as an interrupt source tries to set an IREQ bit, the new interrupt is lost.

VECTORED PROCESSING

Each Z8PLUS interrupt level has its own vector. When an interrupt occurs, control passes to the service routine pointed to by the interrupt's vector location in program memory. The sequence of events for vectored interrupts is as follows:

- PUSH the PC Low Byte on the Stack
- PUSH the PC High Byte on the Stack
- PUSH the FLAGS on the Stack
- Disable Global Interrupts (bit 7 of IMASK)
- Fetch the High Byte of the Vector
- Fetch the Low Byte of the Vector
- Branch to the Service Routine specified by Vector

Figure 4-8 and Figure 4-9 show vectored interrupt operation.

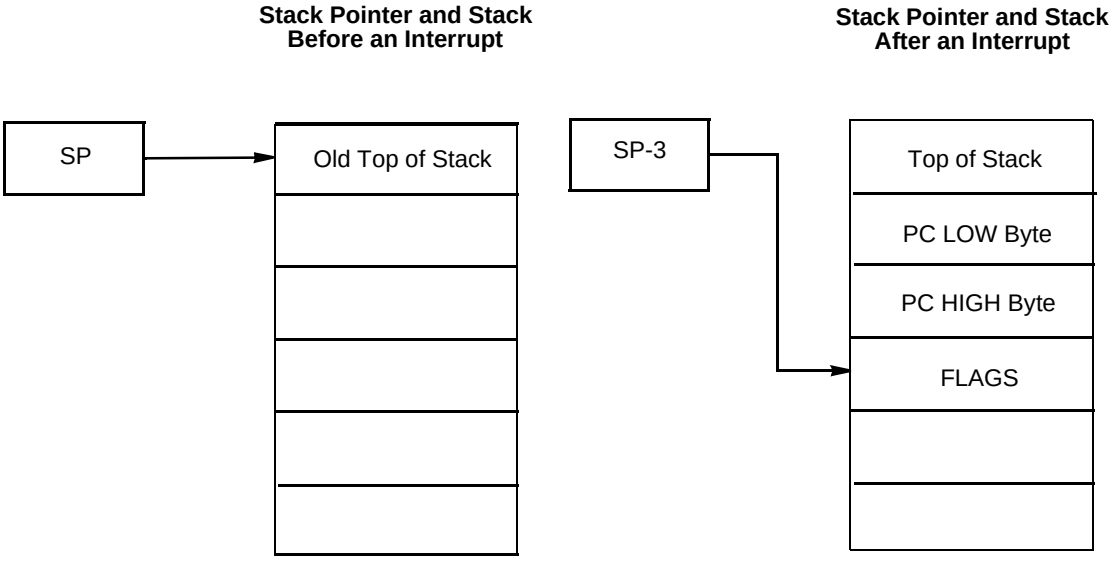

Figure 4-8. Stacks Before and After Interrupt

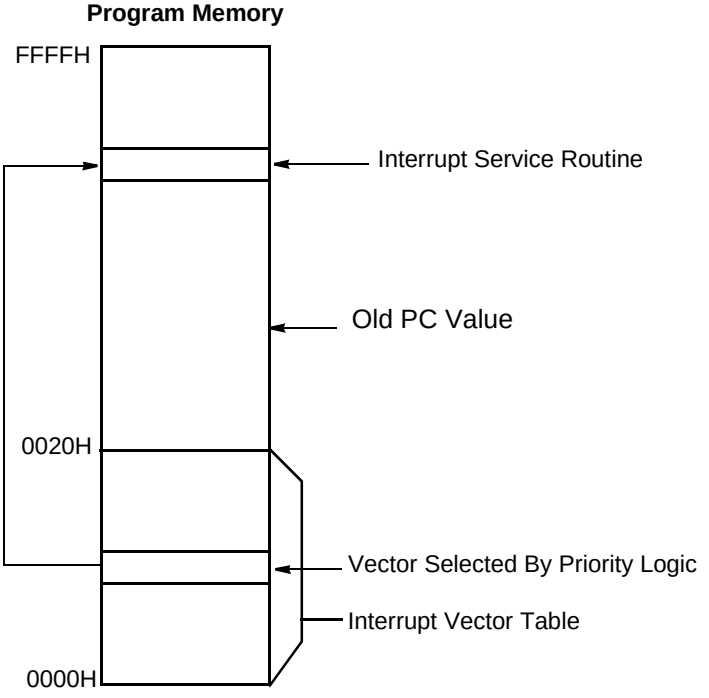

Figure 4-9. Interrupt Vector Table Location

Nesting of Vectored Interrupts

Nesting vectored interrupts allows higher priority requests to interrupt a lower priority request. To initiate vectored interrupt nesting, perform the following steps during the interrupt service routine:

- PUSH the old IMASK on the stack.
- Load IMASK with a new mask to disable lower priority interrupts.
- Execute an EI instruction.
- Proceed with interrupt processing.
- Execute a DI instruction after processing is complete.
- Restore the IMASK to its original value by POPing the previous mask from the stack.
- Execute IRET.

Depending on the application, some simplification of the above procedure may be possible.

POLLED PROCESSING

Polled interrupt processing is supported by masking off the IREQ to be polled. This process is accomplished by setting the corresponding bits in the IMASK to 0.

To initiate polled processing, check the appropriate bits in the IREQ using the Test Under Mask (TM) instruction. If the bit is set to 1, call or branch to the service routine. The service routine services the request, resets its Request Bit in the IREQ, and branches or returns back to the main program. An example of a polling routine is as follows:

```
TM IREQ,#MASKA;Test for request

JR Z, NEXT;If no request go to NEXT

CALL SERVICE;If request is there,then
;service it

NEXT:
.
.
.

SERVICE:;Process Request
.
.
.

AND IREQ, #MASKB ;Clear Request Bit

RET;Return to next
```

In this example, if IREQ2 is being polled, MASKA is 00000100B and MASKB is 11111011B.

RESET CONDITIONS

The IMASK and IREQ registers initialize to 00H on RESET.