Section 8. Interrupts

HIGHLIGHTS

This section of the manual contains the following topics:

8.1 INTRODUCTION

The PIC24F interrupt controller module reduces the numerous peripheral interrupt request signals to a single interrupt request signal to the PIC24F CPU and has the following features:

- Up to 8 processor exceptions and software traps
- 7 user-selectable priority levels
- Interrupt Vector Table (IVT) with up to 118 vectors
- A unique vector for each interrupt or exception source
- Fixed priority within a specified user priority level
- Alternate Interrupt Vector Table (AIVT) for debug support
- Fixed interrupt entry and return latencies

8.1.1 Interrupt Vector Table

The Interrupt Vector Table (IVT) resides in program memory, starting at location 0x000004. The IVT contains 126 vectors, consisting of 8 non-maskable trap vectors, plus up to 118 sources of interrupt. Trap vector details are summarized in Table 8-1. In general, each interrupt source has its own vector. Each interrupt vector contains a 24-bit wide address. The value programmed into each interrupt vector location is the starting address of the associated Interrupt Service Routine (ISR).

8.1.2 Alternate Interrupt Vector Table

The Alternate Interrupt Vector Table (AIVT) is located after the IVT, as shown in Figure 8-1. Access to the AIVT is provided by the ALTIVT control bit (INTCON2<15>). If the ALTIVT bit is set, all interrupt and exception processes will use the alternate vectors instead of the default vectors. The alternate vectors are organized in the same manner as the default vectors.

The AIVT supports emulation and debugging efforts by providing a means to switch between an application and a support environment without requiring the interrupt vectors to be reprogrammed. Sometimes a system may have two applications — a bootloader application and a main application. In this scenario, the bootloader can use one set of vectors and the main application can use the other set.

This feature also enables switching between applications for evaluation of different software algorithms at run time. If the AIVT is not needed, the AIVT should be programmed with the same addresses used in the IVT.

8.1.3 Reset Sequence

A device Reset is not a true exception because the interrupt controller is not involved in the Reset process. The PIC24F device clears its registers in response to a Reset which forces the PC to zero. The processor then begins program execution at location 0x000000. The user programs a `GOTO` instruction at the Reset address, which redirects program execution to the appropriate start-up routine. Refer to **Section 7. "Reset"** for more information on Resets.

Note:	Any unimplemented or unused vector locations in the IVT and AIVT should be programmed with the address of a default interrupt handler routine that contains a `RESET` instruction.

Figure 8-1: Interrupt Vector Table

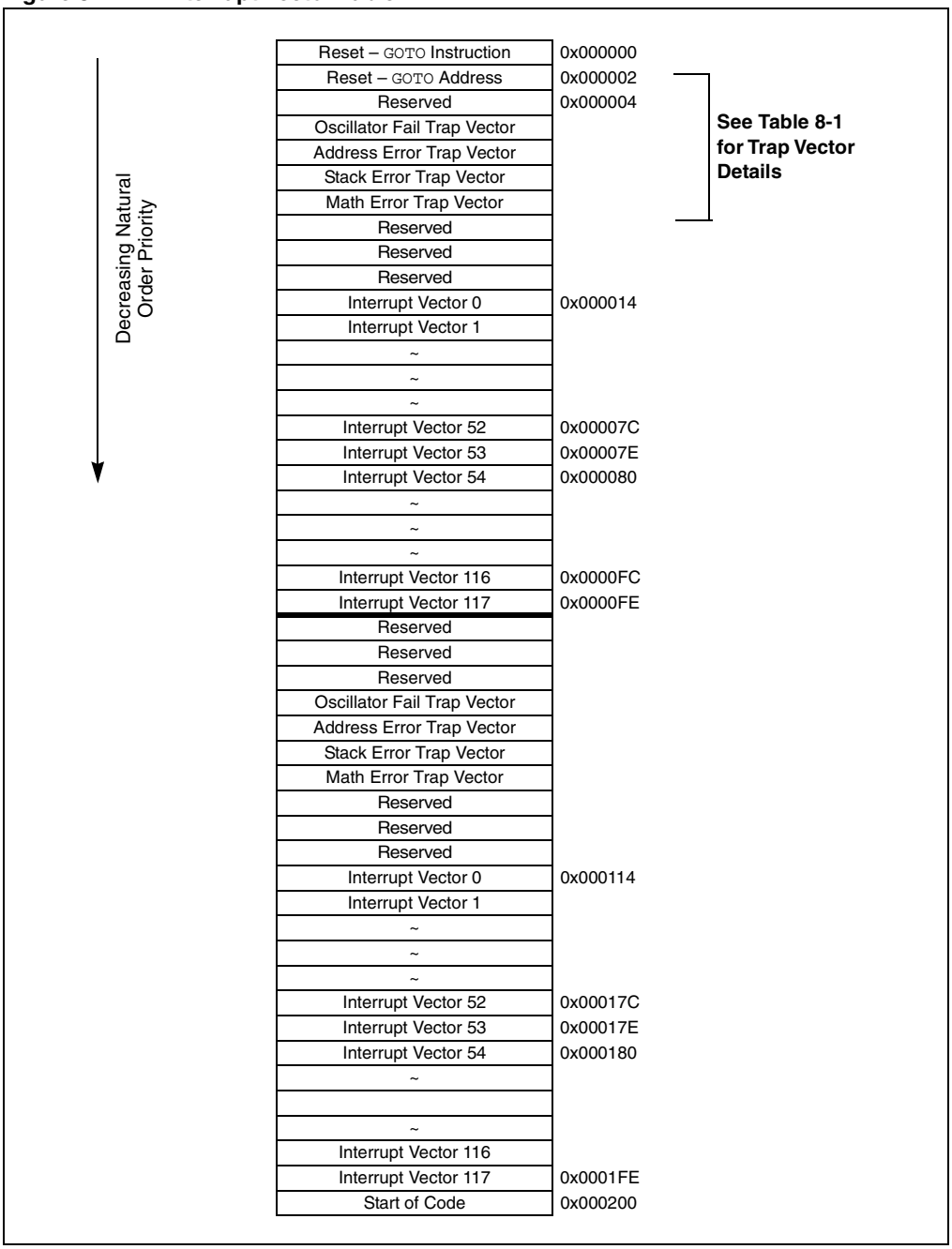


Table 8-1: Trap Vector Details

Vector Number	IVT Address	AIVT Address	Trap Source
0	0x000004	0x000104	Reserved
1	0x000006	0x000106	Oscillator Failure
2	0x000008	0x000108	Address Error
3	0x00000A	0x00010A	Stack Error
4	0x00000C	0x00010C	Math Error
5	0x00000E	0x00010E	Reserved
6	0x000010	0x000110	Reserved
7	0x000012	0x000112	Reserved

8.1.4 CPU Priority Status

The CPU can operate at one of sixteen priority levels, 0-15. An interrupt or trap source must have a priority level greater than the current CPU priority in order to initiate an exception process. Peripheral and external interrupt sources can be programmed for levels 0-7, while CPU priority levels 8-15 are reserved for trap sources. A trap is a non-maskable interrupt source intended to detect hardware and software problems (see **Section 8.2 "Non-Maskable Traps"**). The priority level for each trap source is fixed and only one trap is assigned to a priority level. Note that an interrupt source programmed to priority level 0 is effectively disabled, since it can never be greater than the CPU priority.

The current CPU priority level is indicated by the following four status bits:

- IPL<2:0> status bits located in SR<7:5>
- IPL3 status bit located in CORCON<3>

The IPL<2:0> status bits are readable and writable, so the user may modify these bits to disable all sources of interrupts below a given priority level. If IPL<2:0> = `111`, for example, the CPU would not be interrupted by any source with a programmed priority level of 0, 1, 2 or 3.

Trap events have higher priority (8-15) than any user interrupt source. When the IPL3 bit is set, a trap event is in progress. The IPL3 bit can be cleared, but not set by the user. In some applications, it may be desirable to clear the IPL3 bit when a trap has occurred and branch to an instruction other than the instruction after the one that originally caused the trap to occur.

All user interrupt sources can be disabled by setting IPL<2:0> = `111`.

Note:	The IPL<2:0> bits become read-only bits when interrupt nesting is disabled. See **Section 8.2.4.2 "Interrupt Nesting"** for more information.

8.1.5 Interrupt Priority

Each peripheral interrupt source can be assigned to one of seven priority levels. The user-assignable interrupt priority control bits for each individual interrupt are located in the Least Significant 3 bits of each nibble within the IPCn register(s). Bit 3 of each nibble is not used and is read as '`0`'. These bits define the priority level assigned to a particular interrupt. The usable priority levels start at level 1 as the lowest priority and level 7 as the highest priority. If the IPCn bits associated with an interrupt source are all cleared, then the interrupt source is effectively disabled.

Note:	At a device Reset, the IPCn registers are initialized such that all user interrupt sources are assigned to priority level 4.

Since more than one interrupt request source may be assigned to a specific priority level, a means is provided to resolve priority conflicts within a given user-assigned level. Each source of interrupt has a natural order priority based on its location in the IVT. The lower numbered interrupt vectors have higher natural priority, while the higher numbered vectors have lower natural priority. For example, Interrupt Vector 0 is of the highest natural priority and Interrupt Vector 117 is of the lowest natural priority. The overall priority level for any pending source of interrupt is determined first by the user-assigned priority of that source in the IPCn register, then by the natural order priority within the IVT.

Natural order priority is used only to resolve conflicts between simultaneous pending interrupts with the same user-assigned priority level. Once the priority conflict is resolved and the exception process begins, the CPU can only be interrupted by a source with higher user-assigned priority. Interrupts with the same user-assigned priority, but a higher natural order priority, that become pending after the exception process begins will remain pending until the current exception process completes.

The ability for the user to assign each interrupt source to one of seven priority levels means that the user can give an interrupt with a low natural order priority a very high overall priority level. For example, the Interrupt Vector 0 may be assigned to priority level 1, thus giving it a very low effective priority.

Note:	This document explains the generic interrupt structure. Refer to the specific device data sheet for the peripherals and sources of each interrupt.

8.2 NON-MASKABLE TRAPS

Traps can be considered as non-maskable, nestable interrupts which adhere to a fixed priority structure. Traps are intended to provide the user a means to correct erroneous operation during debug and when operating within the application. If the user does not intend to take corrective action in the event of a trap error condition, these vectors must be loaded with the address of a software routine that will reset the device. Otherwise, the trap vector is programmed with the address of a service routine that will correct the trap condition.

The PIC24F has four implemented sources of non-maskable traps:

- Oscillator Failure Trap
- Stack Error Trap
- Address Error Trap
- Arithmetic Error Trap

The instruction that caused the trap is allowed to complete before exception processing begins. Therefore, the user may have to correct the action of the instruction that caused the trap.

Each trap source has a fixed priority as defined by its position in the IVT. An oscillator failure trap has the highest priority, while an arithmetic error trap has the lowest priority (see Figure 8-1). In addition, trap sources are classified into two distinct categories: ‘Hard’ traps and ‘Soft’ traps.

8.2.1 Soft Traps

The arithmetic error trap (priority level 11) and stack error trap (priority level 12) are categorized as ‘soft’ trap sources. Soft traps can be treated like non-maskable sources of interrupt that adhere to the priority assigned by their position in the IVT. Soft traps are processed like interrupts and require 2 cycles to be sampled and Acknowledged prior to exception processing. Therefore, additional instructions may be executed before a soft trap is Acknowledged.

8.2.1.1 STACK ERROR TRAP (SOFT TRAP, LEVEL 12)

The stack is initialized to 0x0800 during Reset. A stack error trap will be generated should the Stack Pointer address ever be less than 0x0800.

There is a Stack Limit register (SPLIM) associated with the Stack Pointer that is uninitialized at Reset. The stack overflow check is not enabled until a word write to SPLIM occurs.

All Effective Addresses (EA) generated using W15 as a source or destination pointer are compared against the value in SPLIM. Should the EA be greater than the contents of the SPLIM register, then a stack error trap is generated. In addition, a stack error trap will be generated should the EA calculation wrap over the end of data space (0xFFFF).

A stack error can be detected in software by polling the STKERR status bit (INTCON1<2>). To avoid re-entering the Trap Service Routine, the STKERR status flag must be cleared in software prior to returning from the trap with a `RETFIE` instruction.

8.2.1.2 MATH ERROR TRAP (LEVEL 11)

The Math Error trap will execute should an attempt be made to divide by zero. The math error trap can be detected in software by polling the MATHERR status bit (INTCON1<4>). To avoid re-entering the Trap Service Routine, the MATHERR status flag must be cleared in software prior to returning from the trap with a `RETFIE` instruction.

8.2.2 Hard Traps

Hard traps include exceptions of priority level 13 through level 15, inclusive. The address error (level 13) and oscillator error (level 14) traps fall into this category.

Like soft traps, hard traps can also be viewed as non-maskable sources of interrupt. The difference between hard traps and soft traps is that hard traps force the CPU to stop code execution after the instruction causing the trap has completed. Normal program execution flow will not resume until after the trap has been Acknowledged and processed.

8.2.2.1 TRAP PRIORITY AND HARD TRAP CONFLICTS

If a higher priority trap occurs while any lower priority trap is in progress, processing of the lower priority trap will be suspended and the higher priority trap will be Acknowledged and processed. The lower priority trap will remain pending until processing of the higher priority trap completes.

Each hard trap that occurs must be Acknowledged before code execution of any type may continue. If a lower priority hard trap occurs while a higher priority trap is pending, Acknowledged, or is being processed, a hard trap conflict will occur. The conflict occurs because the lower priority trap cannot be Acknowledged until processing for the higher priority trap completes.

The device is automatically reset in a hard trap conflict condition. The TRAPR status bit (RCON<15>) is set when the Reset occurs, so that the condition may be detected in software.

8.2.2.2 OSCILLATOR FAILURE TRAP (HARD TRAP, LEVEL 14)

An oscillator failure trap event will be generated if the Fail-Safe Clock Monitor (FSCM) is enabled and has detected a loss of the system clock source.

An oscillator failure trap event can be detected in software by polling the OSCFAIL status bit (INTCON1<1>) or the CF status bit (OSCCON<3>). To avoid re-entering the Trap Service Routine, the OSCFAIL status flag must be cleared in software prior to returning from the trap with a `RETFIE` instruction.

Refer to **Section 6. "Oscillator"** and **Section 32. "Device Configuration"** for more information about the FSCM.

8.2.2.3 ADDRESS ERROR TRAP (HARD TRAP, LEVEL 13)

The following paragraphs describe operating scenarios that would cause an address error trap to be generated:

1. A misaligned data word fetch is attempted. This condition occurs when an instruction performs a word access with the LSb of the effective address set to '`1`'. The PIC24F CPU requires all word accesses to be aligned to an even address boundary.
2. A bit manipulation instruction using the Indirect Addressing mode with the LSb of the effective address set to '`1`'.
3. A data fetch from unimplemented data address space is attempted.
4. Execution of a "`BRA #literal`" instruction or a "`GOTO #literal`" instruction, where `literal` is an unimplemented program memory address.
5. Executing instructions after modifying the PC to point to unimplemented program memory addresses. The PC may be modified by loading a value into the stack and executing a `RETURN` instruction.

Data space writes will be inhibited whenever an address error trap occurs, so that data is not destroyed.

An address error can be detected in software by polling the ADDRERR status bit (INTCON1<3>). To avoid re-entering the Trap Service Routine, the ADDRERR status flag must be cleared in software prior to returning from the trap with a `RETFIE` instruction.

8.2.3 Disable Interrupts Instruction

The `DISI` (disable interrupts) instruction has the ability to disable interrupts for up to 16384 instruction cycles. This instruction is useful when time critical code segments must be executed.

The `DISI` instruction only disables interrupts with priority levels 1-6. Priority level 7 interrupts and all trap events still have the ability to interrupt the CPU when the `DISI` instruction is active.

The `DISI` instruction works in conjunction with the DISICNT register. When the DISICNT register is non-zero, priority level 1-6 interrupts are disabled. The DISICNT register is decremented on each subsequent instruction cycle. When the DISICNT register counts down to '`0`', priority level 1-6 interrupts will be re-enabled. The value specified in the `DISI` instruction includes all cycles due to PSV accesses, instruction stalls, etc.

The DISICNT register is readable and writable. The user can terminate the effect of a previous `DISI` instruction early by clearing the DISICNT register. The amount of time that interrupts are disabled can also be increased by writing to or adding to DISICNT.

Note that if the DISICNT register is zero, interrupts cannot be disabled by simply writing a non-zero value to the register. Interrupts must first be disabled by using the DISI instruction. Once the DISI instruction has executed and DISICNT holds a non-zero value, the interrupt disable time can be extended by modifying the contents of DISICNT.

Note:	Software modification of the DISICNT register is not recommended.

The DISI status bit (INTCON2<14>) is set whenever interrupts are disabled as a result of the DISI instruction.

Note:	The DISI instruction can be used to quickly disable all user interrupt sources if no source is assigned to CPU priority level 7.

8.2.4 Interrupt Operation

All interrupt event flags are sampled during each instruction cycle. A pending Interrupt Request (IRQ) is indicated by the flag bit being equal to a '1' in an IFSn register. The IRQ will cause an interrupt to occur if the corresponding bit in the Interrupt Enable (IECn) registers is set. For the rest of the instruction cycle in which the IRQ is sampled, the priorities of all pending interrupt requests are evaluated.

No instruction will be aborted when the CPU responds to the IRQ. The instruction that was in progress when the IRQ is sampled will be completed before the ISR is executed.

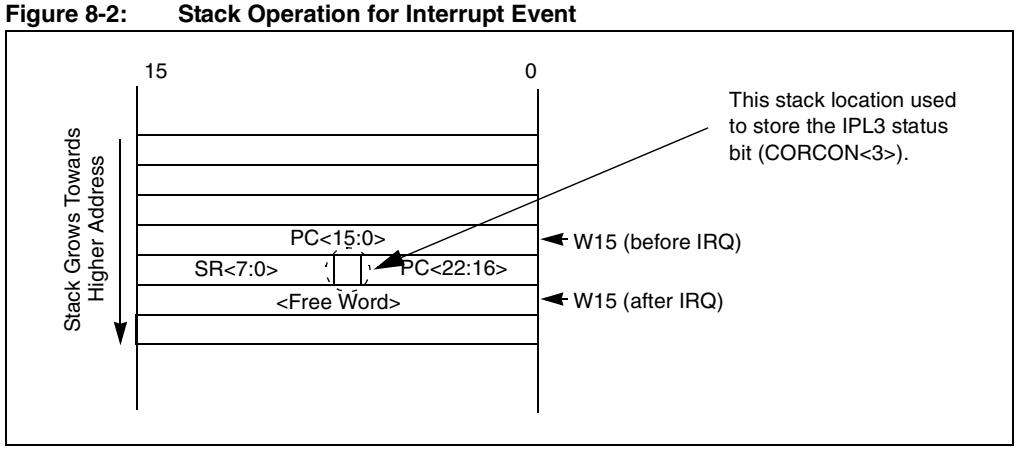
If there is a pending IRQ with a user-assigned priority level greater than the current processor priority level, indicated by the IPL<2:0> status bits (SR<7:5>), an interrupt will be presented to the processor. The processor then saves the following information on the software stack:

- the current PC value
- the low byte of the processor STATUS register (SRL)
- the IPL3 status bit (CORCON<3>)

These three values that are saved on the stack allow the return PC address value, MCU status bits and the current processor priority level to be automatically saved.

After the above information is saved on the stack, the CPU writes the priority level of the pending interrupt into the IPL<2:0> bit locations. This action will disable all interrupts of less than, or equal priority, until the Interrupt Service Routine (ISR) is terminated using the RETFIE instruction.

Figure 8-2: Stack Operation for Interrupt Event



8.2.4.1 RETURN FROM INTERRUPT

The RETFIE (Return from Interrupt) instruction will unstack the PC return address, IPL3 status bit and SRL register, to return the processor to the state and priority level prior to the interrupt sequence.

8.2.4.2 INTERRUPT NESTING

Interrupts, by default, are nestable. Any ISR that is in progress may be interrupted by another source of interrupt with a higher user-assigned priority level. Interrupt nesting may be optionally disabled by setting the NSTDIS control bit (INTCON1<15>). When the NSTDIS control bit is set, all interrupts in progress will force the CPU priority to level 7 by setting IPL<2:0> = 111. This action will effectively mask all other sources of interrupt until a RETFIE instruction is executed. When interrupt nesting is disabled, the user-assigned interrupt priority levels will have no effect, except to resolve conflicts between simultaneous pending interrupts.

The IPL<2:0> bits become read-only when interrupt nesting is disabled. This prevents the user software from setting IPL<2:0> to a lower value which would effectively re-enable interrupt nesting.

8.2.5 Wake-up from Sleep and Idle

Any source of interrupt that is individually enabled, using its corresponding control bit in the IECn registers, can wake-up the processor from Sleep or Idle mode. When the interrupt status flag for a source is set and the interrupt source is enabled via the corresponding bit in the IECn Control registers, a wake-up signal is sent to the PIC24F CPU. When the device wakes from Sleep or Idle mode, one of two actions may occur:

1. If the interrupt priority level for that source is greater than the current CPU priority level, then the processor will process the interrupt and branch to the ISR for the interrupt source.
2. If the user-assigned interrupt priority level for the source is less than or equal to the current CPU priority level, then the processor will simply continue execution, starting with the instruction immediately following the PWRSAV instruction that previously put the CPU in Sleep or Idle mode.

Note:	User interrupt sources that are assigned to CPU priority level 0 cannot wake the CPU from Sleep or Idle mode, because the interrupt source is effectively disabled. To use an interrupt as a wake-up source, the CPU priority level for the interrupt must be assigned to CPU priority level 1 or greater.

8.2.6 A/D Converter External Conversion Request

The external interrupt request pin is shared with the A/D converter as an external conversion request signal. The Interrupt Vector 0 interrupt source has programmable edge polarity which is also available to the A/D converter external conversion request feature. Refer to **Section 17. "10-Bit A/D Converter"** for more information on the A/D converter.

8.2.7 External Interrupt Support

The PIC24F supports up to 5 external interrupt pin sources (Interrupt Vector 0 to Interrupt Vector 4). Each external interrupt pin has edge detection circuitry to detect the interrupt event. The INTCON2 register has five control bits (INT0EP-INT4EP) that select the polarity of the edge detection circuitry. Each external interrupt pin may be programmed to interrupt the CPU on a rising edge or falling edge event. See Register 8-4 for further details.

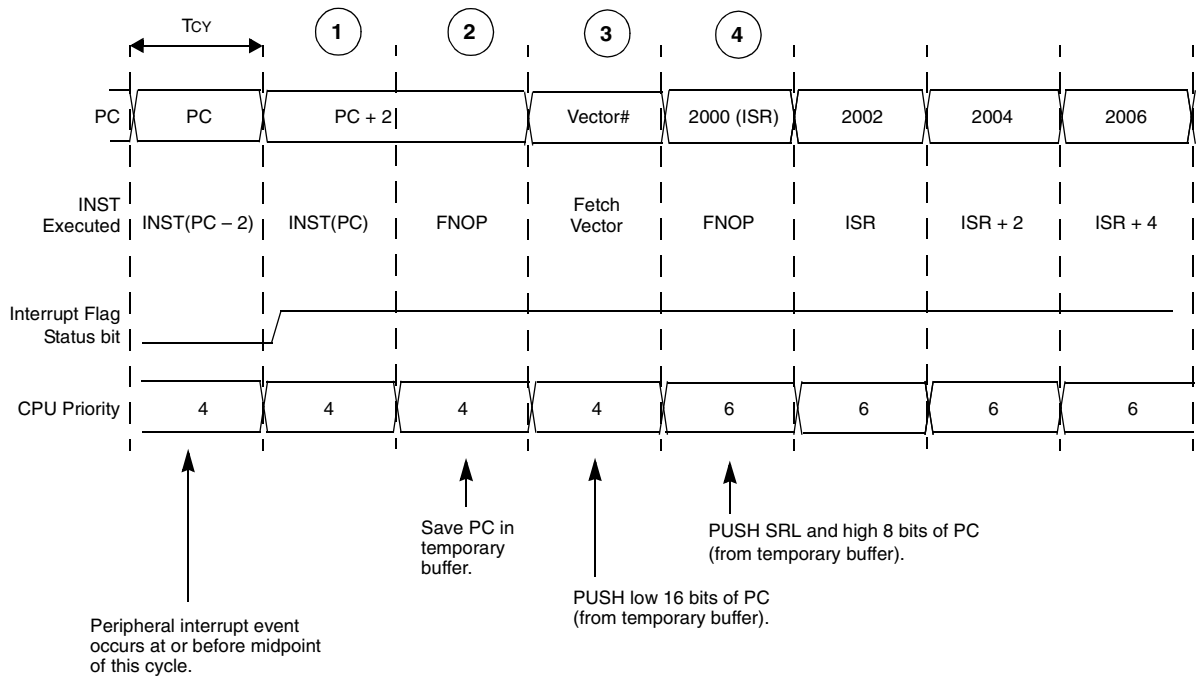
8.3 INTERRUPT PROCESSING TIMING

8.3.1 Interrupt Latency for One-Cycle Instructions

Figure 8-3 shows the sequence of events when a peripheral interrupt is asserted during a one-cycle instruction. The interrupt process takes four instruction cycles. Each cycle is numbered in Figure 8-3 for reference.

The interrupt flag status bit is set during the instruction cycle after the peripheral interrupt occurs. The current instruction completes during this instruction cycle. In the second instruction cycle after the interrupt event, the contents of the PC and SRL registers are saved into a temporary buffer register. The second cycle of the interrupt process is executed as a NOP to maintain consistency with the sequence taken during a two-cycle instruction (see **Section 8.3.2 "Interrupt Latency for Two-Cycle Instructions"**). In the third cycle, the PC is loaded with the vector table address for the interrupt source and the starting address of the ISR is fetched. In the fourth cycle, the PC is loaded with the ISR address. The fourth cycle is executed as a NOP while the first instruction in the ISR is fetched.

Figure 8-3: Interrupt Timing During a One-Cycle Instruction



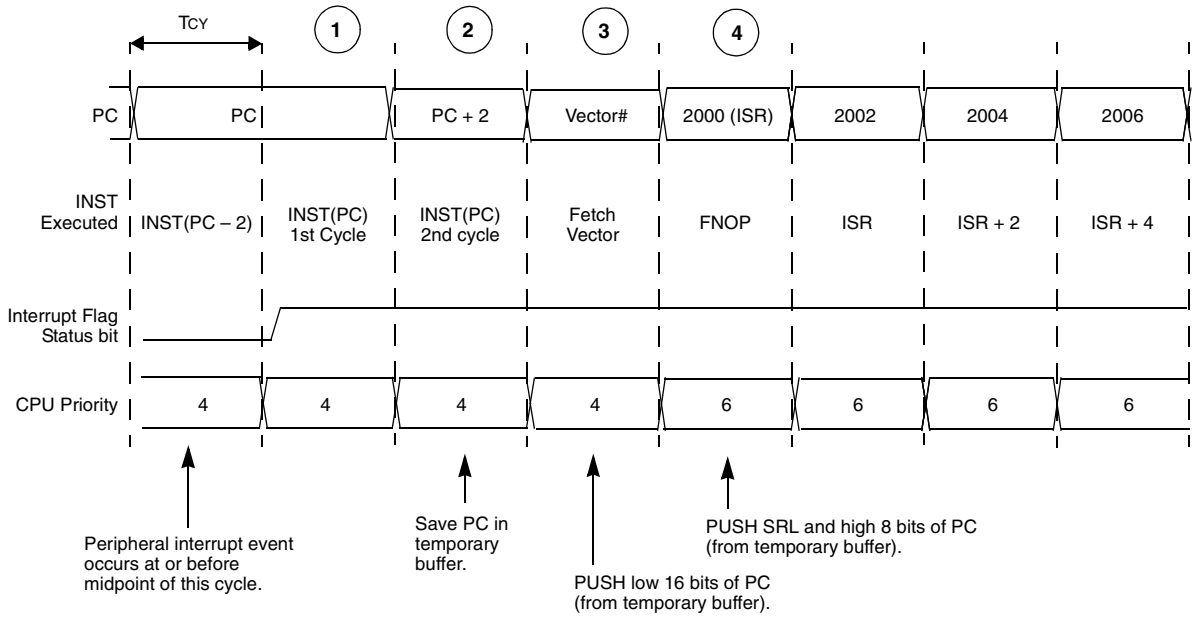
Note: Where FNOP is a forced NOP instruction automatically inserted by the CPU.

8.3.2 Interrupt Latency for Two-Cycle Instructions

The interrupt latency during a two-cycle instruction is the same as during a one-cycle instruction. The first and second cycle of the interrupt process allow the two-cycle instruction to complete execution. The timing diagram in Figure 8-5 shows the case when the peripheral interrupt event occurs in the instruction cycle prior to execution of the two-cycle instruction.

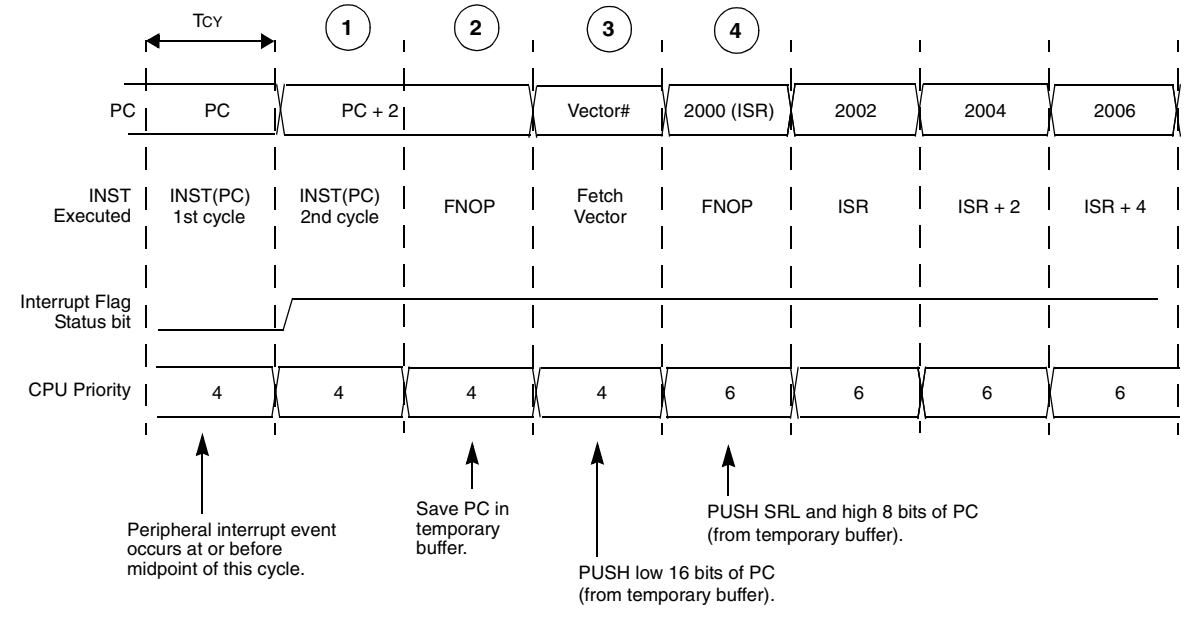
Figure 8-6 shows the timing when a peripheral interrupt is coincident with the first cycle of a two-cycle instruction. In this case, the interrupt process completes as for a one-cycle instruction (see **Section 8.3.1 "Interrupt Latency for One-Cycle Instructions"**).

Figure 8-4: Interrupt Timing During a Two-Cycle Instruction



Note: Where FNOP is a forced NOP instruction automatically inserted by the CPU.

Figure 8-5: Interrupt Timing, Interrupt Occurs During 1st Cycle of a 2-Cycle Instruction



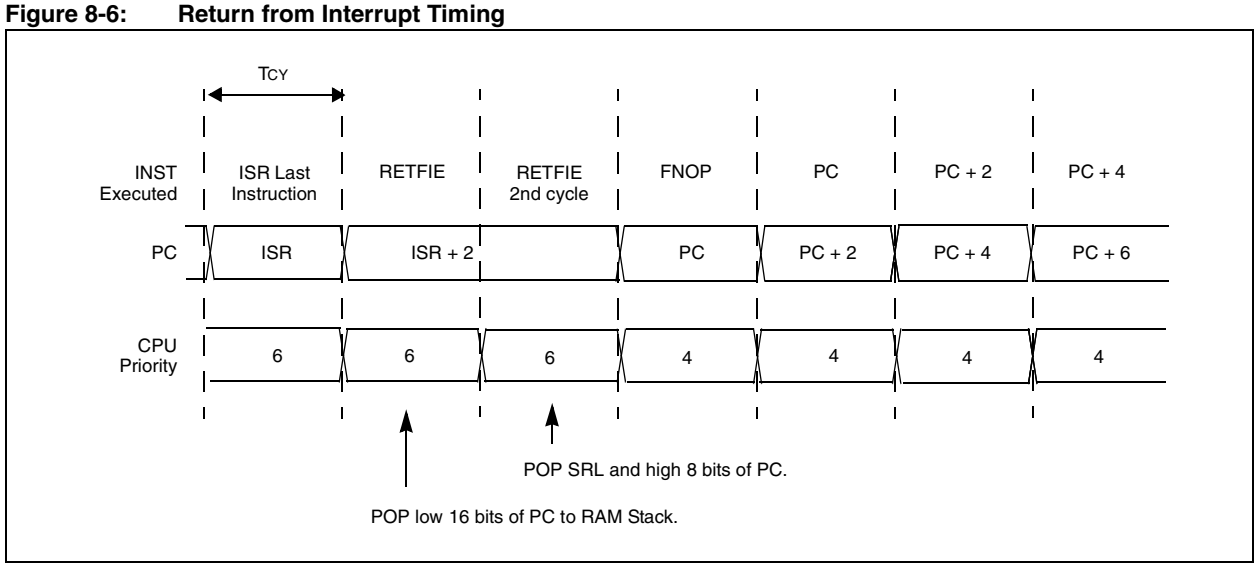
Note: Where FNOP is a forced NOP instruction automatically inserted by the CPU.

8.3.3 Returning from Interrupt

The “Return from Interrupt” instruction, RETFIE, exits an interrupt or trap routine.

During the first cycle of a RETFIE instruction, the upper bits of the PC and the SRL register are popped from the stack. The lower 16 bits of the stacked PC value are popped from the stack during the second cycle. The third instruction cycle is used to fetch the instruction addressed by the updated program counter. This cycle executes as a NOP.

Figure 8-6: Return from Interrupt Timing



8.4 INTERRUPT CONTROL AND STATUS REGISTERS

The following registers are associated with the interrupt controller:

- **INTCON1, INTCON2 Registers**
  Global interrupt control functions are derived from these two registers. INTCON1 contains the Interrupt Nesting Disable (NSTDIS) bit, as well as the control and status flags for the processor trap sources. The INTCON2 register controls the external interrupt request signal behavior and the use of the alternate vector table.
- **IFSn: Interrupt Flag Status Registers**
  All interrupt request flags are maintained in the IFSn registers, where 'n' denotes the register number. Each source of interrupt has a status bit, which is set by the respective peripherals or external signal, and is cleared via software.
- **IECn: Interrupt Enable Control Registers**
  All interrupt enable control bits are maintained in the IECn registers, where 'n' denotes the register number. These control bits are used to individually enable interrupts from the peripherals or external signals.
- **IPCn: Interrupt Priority Control Registers**
  Each user interrupt source can be assigned to one of eight priority levels. The IPCn registers are used to set the interrupt priority level for each source of interrupt.
- **SR: CPU STATUS Register**
  The SR is not specifically part of the interrupt controller hardware, but it contains the IPL<2:0> status bits (SR<7:5>) that indicate the current CPU priority level. The user may change the current CPU priority level by writing to the IPL bits.
- **CORCON: Core Control Register**
  The CORCON is not specifically part of the interrupt controller hardware, but it contains the IPL3 status bit which indicates the current CPU priority level. IPL3 is a read-only bit, so that trap events cannot be masked by the user software.

SR, CORCON, INTCON1 and INTCON2 registers are described in details on the following pages. The generic interrupt registers map is also given on the following pages. Each interrupt is associated with an Interrupt Flag (IF), an Interrupt Enable bit (IE) and three Interrupt Priority Bits (IP2:IP0). Actual number of IFSn, IECn and IPCn registers depends upon the number of interrupts implemented on a particular device. Refer to the specific data sheet for further details.

8.4.1 Assignment of Interrupts to Control Registers

The interrupt sources are assigned to the IFSn, IECn and IPCn registers in a particular sequence. For example, Interrupt Vector 0 has a natural order priority of 0. Thus, the Interrupt Vector 0 status bit is found in IFS0<0>. Interrupt Vector 0 uses IEC0<0> as its enable bit and the IPC0<2:0> bits assign the interrupt priority level for Interrupt Vector 0. Refer to Table 8-2 for a generic summary of all the interrupt related registers.

Register 8-1: SR: CPU STATUS Register

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	DC
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
IPL2(1,2)	IPL1(1,2)	IPL0(1,2)	RA	N	OV	Z	C
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at any Reset	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7-5 **IPL2:IPL0:** CPU Interrupt Priority Level Status bits(1,2)

111 = CPU interrupt priority level is 7 (15). User interrupts disabled.
110 = CPU interrupt priority level is 6 (14)
101 = CPU interrupt priority level is 5 (13)
100 = CPU interrupt priority level is 4 (12)
011 = CPU interrupt priority level is 3 (11)
010 = CPU interrupt priority level is 2 (10)
001 = CPU interrupt priority level is 1 (9)
000 = CPU interrupt priority level is 0 (8)

Note 1: The IPL<2:0> bits are concatenated with the IPL<3> bit (CORCON<3>) to form the CPU interrupt priority level. The value in parentheses indicates the IPL if IPL<3> = 1.

2: The IPL<2:0> status bits are read-only when NSTDIS = 1 (INTCON1<15>).

Register 8-2: CORCON: Core Control Register

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	R/C-0	R/W-0	U-0	U-0
—	—	—	—	IPL3(1)	PSV	—	—
bit 7							bit 0

Legend:	C = Clearable bit		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at any Reset	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 3 **IPL3:** CPU Interrupt Priority Level Status bit 3(1)

1 = CPU interrupt priority level is greater than 7
0 = CPU interrupt priority level is 7 or less

Note 1: The IPL3 bit is concatenated with the IPL<2:0> bits (SR<7:5>) to form the CPU interrupt priority level.

Register 8-3: INTCON1: Interrupt Control Register 1

R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
NSTDIS	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
—	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at any Reset	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **NSTDIS:** Interrupt Nesting Disable bit
1 = Interrupt nesting is disabled
0 = Interrupt nesting is enabled

bit 14-5 **Unimplemented:** Read as '0'

bit 4 **MATHERR:** Arithmetic Error Trap Status bit
1 = Overflow trap has occurred
0 = Overflow trap has not occurred

bit 3 **ADDRERR:** Address Error Trap Status bit
1 = Address error trap has occurred
0 = Address error trap has not occurred

bit 2 **STKERR:** Stack Error Trap Status bit
1 = Stack error trap has occurred
0 = Stack error trap has not occurred

bit 1 **OSCFAIL:** Oscillator Failure Trap Status bit
1 = Oscillator failure trap has occurred
0 = Oscillator failure trap has not occurred

bit 0 **Unimplemented:** Read as '0'

Register 8-4: INTCON2: Interrupt Control Register 2

R/W-0	R-0	U-0	U-0	U-0	U-0	U-0	U-0
ALTIVT	DISI	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at any Reset	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **ALTIVT:** Enable Alternate Interrupt Vector Table bit
1 = Use alternate vector table
0 = Use standard (default) vector table

bit 14 **DISI:** DISI Instruction Status bit
1 = DISI instruction is active
0 = DISI is not active

bit 13-5 **Unimplemented:** Read as '0'

bit 4 **INT4EP:** External Interrupt #4 Edge Detect Polarity Select bit
1 = Interrupt on negative edge
0 = Interrupt on positive edge

bit 3 **INT3EP:** External Interrupt #3 Edge Detect Polarity Select bit
1 = Interrupt on negative edge
0 = Interrupt on positive edge

bit 2 **INT2EP:** External Interrupt #2 Edge Detect Polarity Select bit
1 = Interrupt on negative edge
0 = Interrupt on positive edge

bit 1 **INT1EP:** External Interrupt #1 Edge Detect Polarity Select bit
1 = Interrupt on negative edge
0 = Interrupt on positive edge

bit 0 **INT0EP:** External Interrupt #0 Edge Detect Polarity Select bit
1 = Interrupt on negative edge
0 = Interrupt on positive edge

Register 8-5: IFSn: Interrupt Flag Status Registers 0 Through 6 (Interrupt Vectors 0 Through 111)(1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
V(16n + 15)IF	V(16n + 14)IF	V(16n + 13)IF	V(16n + 12)IF	V(16n + 11)IF	V(16n +1 0)IF	V(16n + 9)IF	V(16n + 8)IF
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
V(16n + 7)IF	V(16n + 6)IF	V(16n + 5)IF	V(16n + 4)IF	V(16n + 3)IF	V(16n + 2)IF	V(16n + 1)IF	V(16n)IF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at any Reset	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-0 **V(16n + x)IF:** Interrupt Status Flag bits for Interrupt Vector 16n + x (where x = bit position number)
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

Note 1: Not all interrupt vectors are implemented on all devices. Refer to the Interrupt Vector Table for the specific device or family data sheet to verify where interrupt vectors are implemented for a specific device.

Register 8-6: IFSn: Interrupt Flag Status Register 7 (Interrupt Vectors 112 Through 117)(1)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
—	—	V117IF	V116IF	V115IF	V114IF	V113IF	V112IF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at any Reset	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-6 **Unimplemented:** Read as '0'

bit 5-0 **V117IF:V112IF** Interrupt Status Flag bits for Interrupt Vectors 117 through 112
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

Note 1: Not all interrupt vectors are implemented on all devices. Refer to the Interrupt Vector Table for the specific device or family data sheet to verify where interrupt vectors are implemented for a specific device.

Register 8-7: IECn: Interrupt Enable Registers 0 Through 6 (Interrupt Vectors 0 Through 111)(1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
V(16n + 15)IE	V(16n + 14)IE	V(16n + 13)IE	V(16n + 12)IE	V(16n + 11)IE	V(16n + 10)IE	V(16n + 9)IE	V(16n + 8)IE
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
V(16n + 7)IE	V(16n + 6)IE	V(16n + 5)IE	V(16n + 4)IE	V(16n + 3)IE	V(16n + 2)IE	V(16n + 1)IE	V(16n + 0)IE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at any Reset	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-0 **V(16n + x)IF:** Interrupt Enable bits for Interrupt Vector 16n + x (where x = bit position number)
1 = Interrupt is enabled
0 = Interrupt is disabled

Note 1: Not all interrupt vectors are implemented on all devices. Refer to the Interrupt Vector Table for the specific device or family data sheet to verify where interrupt vectors are implemented for a specific device.

Register 8-8: IECn: Interrupt Enable Register 7 (Interrupt Vectors 112 Through 117)(1)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	V117IF	V116IF	V115IF	V114IF	V113IF	V112IF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at any Reset	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-6 **Unimplemented:** Read as '0'

bit 5-0 **V117IE:V112IE** Interrupt Enable bits for Interrupt Vectors 117 through 112
1 = Interrupt is enabled
0 = Interrupt is disabled

Note 1: Not all interrupt vectors are implemented on all devices. Refer to the Interrupt Vector Table for the specific device or family data sheet to verify where interrupt vectors are implemented for a specific device.

Register 8-9: IPCn: Interrupt Priority Registers 0 Through 28 (Interrupt Vectors 0 Through 115)(1)

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	V(4n + 3)IP2	V(4n + 3)IP1	V(xn + 3)IP0	—	V(4n + 2)IP2	V(4n + 2)IP1	V(4n + 2)IP0
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	V(4n + 1)IP2	V(4n + 1)IP1	V(xn + 1)IP0	—	V(4n)IP2	V(4n)IP1	V(4n)IP0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at any Reset	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15 **Unimplemented:** Read as '0'

bit 14-12 **V(4n + 3)IP2:V(4n + 3)IP0:** Interrupt Priority bits for Interrupt Vector 4n + 3
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 11 **Unimplemented:** Read as '0'

bit 10-8 **V(4n + 2)IP2:V(4n + 2)IP0:** Interrupt Priority bits for Interrupt Vector 4n + 2
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 7 **Unimplemented:** Read as '0'

bit 6-4 **V(4n + 1)IP2:V(4n + 1)IP0:** Interrupt Priority bits for Interrupt Vector 4n + 1
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 3 **Unimplemented:** Read as '0'

bit 2-0 **V(4n)IP2:V(4n)IP0:** Interrupt Priority bits for Interrupt Vector 4n
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

Note 1: Not all interrupt vectors are implemented on all devices. Refer to the Interrupt Vector Table for the specific device or family data sheet to verify where interrupt vectors are implemented for a specific device.

Register 8-10: IPCn: Interrupt Priority Register 29 (Interrupt Vectors116 and 117)(1)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	V117IP2	V117IP1	V117IP0	—	V116IP2	V116IP1	V116IP0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at any Reset	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-7 **Unimplemented:** Read as '0'

bit 6-4 **V117IP2:V117IP0:** Interrupt Priority bits for Interrupt Vector 117
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 3 **Unimplemented:** Read as '0'

bit 2-0 **V116IP2:V116IP0:** Interrupt Priority bits for Interrupt Vector 116
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

Note 1: Not all interrupt vectors are implemented on all devices. Refer to the Interrupt Vector Table for the specific device or family data sheet to verify where interrupt vectors are implemented for a specific device.

8.5 INTERRUPT SETUP PROCEDURES

8.5.1 Initialization

The following steps describe how to configure a source of interrupt:

1. Set the NSTDIS Control bit (INTCON1<15>) if nested interrupts are not desired.
2. Select each user-assigned priority level for the interrupt source by writing the control bits in the appropriate IPCn Control register. The priority level will depend on the specific application and type of interrupt source. If multiple priority levels are not desired, the IPCn register control bits for all enabled interrupt sources may be programmed to the same non-zero value.

Note:	At a device Reset, the IPCn registers are initialized, such that all user interrupt sources are assigned to priority level 4.

3. Clear the interrupt flag status bit associated with the peripheral in the associated IFSn Status register.
4. Enable the interrupt source by setting the interrupt enable control bit associated with the source in the appropriate IECn Control register.

8.5.2 Interrupt Service Routine

The method that is used to declare an ISR and initialize the IVT and AIVT with the correct vector address will depend on the programming language (i.e., C or assembler) and the language development toolsuite that is used to develop the application. In general, the user must clear the interrupt flag in the appropriate IFSn register for the source of interrupt that the ISR handles. Otherwise, the ISR will be re-entered immediately after exiting the routine. If the ISR is coded in assembly language, it must be terminated using a `RETFIE` instruction to unstack the saved PC value, SRL value and old CPU priority level.

8.5.3 Trap Service Routine

A Trap Service Routine (TSR) is coded like an ISR, except that the appropriate trap status flag in the INTCON1 register must be cleared to avoid re-entry into the TSR.

8.5.4 Interrupt Disable

All user interrupts can be disabled using the following procedure:

1. Push the current SR value onto the software stack using the `PUSH` instruction.
2. Force the CPU to priority level 7 by inclusive ORing the value 0xE0 with SRL.

To enable user interrupts, the `POP` instruction may be used to restore the previous SR value.

Note that only user interrupts with a priority level of 7 or less can be disabled. Trap sources (level 8-level 15) cannot be disabled.

The `DISI` instruction provides a convenient way to disable interrupts of priority levels 1-6 for a fixed period of time. Level 7 interrupt sources are not disabled by the `DISI` instruction.

8.6 REGISTER MAPS

A summary of the Special Function Registers associated with the interrupt controller is provided in Table 8-2.

Table 8-2: Special Function Registers Associated with Interrupt Controller

SFR Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SR	—	—	—	—	—	—	—	DC	IPL2	IPL1	IPL0	RA	N	OV	Z	C
CORCON	—	—	—	—	—	—	—	—	—	—	—	—	IPL3	PSV	—	—
INTCON1	NSTDIS	—	—	—	—	—	—	—	—	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—
INTCON2	ALTIVT	DISI	—	—	—	—	—	—	—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP
IFS0	V15IF	V14IF	V13IF	V12IF	V11IF	V10IF	V09IF	V08IF	V07IF	V06IF	V05IF	V04IF	V03IF	V02IF	V01IF	V00IF
IFS1	V31IF	V30IF	V29IF	V28IF	V27IF	V26IF	V25IF	V24IF	V23IF	V22IF	V21IF	V20IF	V19IF	V18IF	V17IF	V16IF
IFS2	V47IF	V46IF	V45IF	V44IF	V43IF	V41IF	V41IF	V40IF	V39IF	V38IF	V37IF	V36IF	V35IF	V34IF	V33IF	V32IF
IFS3	V63IF	V62IF	V61IF	V60IF	V59IF	V58IF	V57IF	V56IF	V55IF	V54IF	V53IF	V52IF	V51IF	V50IF	V49IF	V48IF
IFS4	V79IF	V78IF	V77IF	V76IF	V75IF	V74IF	V73IF	V72IF	V71IF	V70IF	V69IF	V68IF	V67IF	V66IF	V65IF	V64IF
IFS5	V95IF	V94IF	V93IF	V92IF	V91IF	V90IF	V89IF	V88IF	V87IF	V86IF	V85IF	V84IF	V83IF	V82IF	V81IF	V80IF
IFS6	V111IF	V110IF	V109IF	V108IF	V107IF	V106IF	V105IF	V104IF	V103IF	V102IF	V101IF	V100IF	V99IF	V98IF	V97IF	V96IF
IFS7	—	—	—	—	—	—	—	—	—	—	V117IF	V116IF	V115IF	V114IF	V113IF	V112IF
IEC0	V15IE	V14IE	V13IE	V12IE	V11IE	V10IE	V09IE	V08IE	V07IE	V06IE	V05IE	V04IE	V03IE	V02IE	V01IE	V00IE
IEC1	V31IE	V30IE	V29IE	V28IE	V27IE	V26IE	V25IE	V24IE	V23IE	V22IE	V21IE	V20IE	V19IE	V18IE	V17IE	V16IE
IEC2	V47IE	V46IE	V45IE	V44IE	V43IE	V41IE	V41IE	V40IE	V39IE	V38IE	V37IE	V36IE	V35IE	V34IE	V33IE	V32IE
IEC3	V63IE	V62IE	V61IE	V60IE	V59IE	V58IE	V57IE	V56IE	V55IE	V54IE	V53IE	V52IE	V51IE	V50IE	V49IE	V48IE
IEC4	V79IE	V78IE	V77IE	V76IE	V75IE	V74IE	V73IE	V72IE	V71IE	V70IE	V69IE	V68IE	V67IE	V66IE	V65IE	V64IE
IEC5	V95IE	V94IE	V93IE	V92IE	V91IE	V90IE	V89IE	V88IE	V87IE	V86IE	V85IE	V84IE	V83IE	V82IE	V81IE	V80IE
IEC6	V111IE	V110IE	V109IE	V108IE	V107IE	V106IE	V105IE	V104IE	V103IE	V102IE	V101IE	V100IE	V99IE	V98IE	V97IE	V96IE
IEC7	—	—	—	—	—	—	—	—	—	—	V117IE	V116IE	V115IE	V114IE	V113IE	V112IE
IPC0	—	V03IP2	V03IP1	V03IP0	—	V02IP2	V02IP1	V02IP0	—	V01IP2	V01IP1	V01IP0	—	V00IP2	V00IP1	V00IP0
IPC1	—	V07IP2	V07IP1	V07IP0	—	V06IP2	V06IP1	V06IP0	—	V05IP2	V05IP1	V05IP0	—	V04IP2	V04IP1	V04IP0
IPC2	—	V11IP2	V11IP1	V11IP0	—	V10IP2	V10IP1	V10IP0	—	V09IP2	V09IP1	V09IP0	—	V08IP2	V08IP1	V08IP0
IPC3	—	V15IP2	V15IP1	V15IP0	—	V14IP2	V14IP1	V14IP0	—	V13IP2	V13IP1	V13IP0	—	V12IP2	V12IP1	V12IP0
IPC4	—	V19IP2	V19IP1	V19IP0	—	V18IP2	V18IP1	V18IP0	—	V17IP2	V17IP1	V17IP0	—	V16IP2	V16IP1	V16IP0
IPC5	—	V23IP2	V23IP1	V23IP0	—	V22IP2	V22IP1	V22IP0	—	V21IP2	V21IP1	V21IP0	—	V20IP2	V20IP1	V20IP0
IPC6	—	V27IP2	V27IP1	V27IP0	—	V26IP2	V26IP1	V26IP0	—	V25IP2	V25IP1	V25IP0	—	V24IP2	V24IP1	V24IP0
IPC7	—	V31IP2	V31IP1	V31IP0	—	V30IP2	V30IP1	V30IP0	—	V29IP2	V29IP1	V29IP0	—	V28IP2	V28IP1	V28IP0
IPC8	—	V35IP2	V35IP1	V35IP0	—	V34IP2	V34IP1	V34IP0	—	V33IP2	V33IP1	V33IP0	—	V32IP2	V32IP1	V32IP0
IPC9	—	V39IP2	V39IP1	V39IP0	—	V38IP2	V38IP1	V38IP0	—	V37IP2	V37IP1	V37IP0	—	V36IP2	V36IP1	V36IP0
IPC10	—	V43IP2	V43IP1	V43IP0	—	V42IP2	V42IP1	V42IP0	—	V41IP2	V41IP1	V41IP0	—	V40IP2	V40IP1	V40IP0
IPC11	—	V47IP2	V47IP1	V47IP0	—	V46IP2	V46IP1	V46IP0	—	V45IP2	V45IP1	V45IP0	—	V44IP2	V44IP1	V44IP0

Note: All interrupt sources and their associated control bits may not be available on a particular device. Refer to the device data sheet for details.

Table 8-2: Special Function Registers Associated with Interrupt Controller (Continued)

SFR Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IPC12	—	V51IP2	V51IP1	V51IP0	—	V50IP2	V50IP1	V50IP0	—	V49IP2	V49IP1	V49IP0	—	V48IP2	V48IP1	V48IP0
IPC13	—	V55IP2	V55IP1	V55IP0	—	V54IP2	V54IP1	V54IP0	—	V53IP2	V53IP1	V53IP0	—	V52IP2	V52IP1	V52IP0
IPC14	—	V59IP2	V59IP1	V59IP0	—	V58IP2	V58IP1	V58IP0	—	V57IP2	V57IP1	V57IP0	—	V56IP2	V56IP1	V56IP0
IPC15	—	V63IP2	V63IP1	V63IP0	—	V62IP2	V62IP1	V62IP0	—	V61IP2	V61IP1	V61IP0	—	V60IP2	V60IP1	V60IP0
IPC16	—	V67IP2	V67IP1	V67IP0	—	V66IP2	V66IP1	V66IP0	—	V65IP2	V65IP1	V65IP0	—	V64IP2	V64IP1	V64IP0
IPC17	—	V71IP2	V71IP1	V71IP0	—	V70IP2	V70IP1	V70IP0	—	V69IP2	V69IP1	V69IP0	—	V68IP2	V68IP1	V68IP0
IPC18	—	V75IP2	V75IP1	V75IP0	—	V74IP2	V74IP1	V74IP0	—	V73IP2	V73IP1	V73IP0	—	V72IP2	V72IP1	V72IP0
IPC19	—	V79IP2	V79IP1	V79IP0	—	V78IP2	V78IP1	V78IP0	—	V77IP2	V77IP1	V77IP0	—	V76IP2	V76IP1	V76IP0
IPC20	—	V83IP2	V83IP1	V83IP0	—	V82IP2	V82IP1	V82IP0	—	V81IP2	V81IP1	V81IP0	—	V80IP2	V80IP1	V80IP0
IPC21	—	V87IP2	V87IP1	V87IP0	—	V86IP2	V86IP1	V86IP0	—	V85IP2	V85IP1	V85IP0	—	V84IP2	V84IP1	V84IP0
IPC22	—	V91IP2	V91IP1	V91IP0	—	V90IP2	V90IP1	V90IP0	—	V89IP2	V89IP1	V89IP0	—	V88IP2	V88IP1	V88IP0
IPC23	—	V95IP2	V95IP1	V95IP0	—	V94IP2	V94IP1	V94IP0	—	V93IP2	V93IP1	V93IP0	—	V92IP2	V92IP1	V92IP0
IPC24	—	V99IP2	V99IP1	V99IP0	—	V98IP2	V98IP1	V98IP0	—	V97IP2	V97IP1	V97IP0	—	V96IP2	V96IP1	V96IP0
IPC25	—	V103IP2	V103IP1	V103IP0	—	V102IP2	V102IP1	V102IP0	—	V101IP2	V101IP1	V101IP0	—	V100IP2	V100IP1	V100IP0
IPC26	—	V107IP2	V107IP1	V107IP0	—	V106IP2	V106IP1	V106IP0	—	V105IP2	V105IP1	V105IP0	—	V104IP2	V104IP1	V104IP0
IPC27	—	V111IP2	V111IP1	V111IP0	—	V110IP2	V110IP1	V110IP0	—	V109IP2	V109IP1	V109IP0	—	V108IP2	V108IP1	V108IP0
IPC28	—	V115IP2	V115IP1	V115IP0	—	V114IP2	V114IP1	V114IP0	—	V113IP2	V113IP1	V113IP0	—	V112IP2	V112IP1	V112IP0
IPC29	—	—	—	—	—	—	—	—	—	V117IP2	V117IP1	V117IP0	—	V116IP2	V116IP1	V116IP0

Note: All interrupt sources and their associated control bits may not be available on a particular device. Refer to the device data sheet for details.

8.7 DESIGN TIPS

Question 1: ***What happens when two sources of interrupt become pending at the same time and have the same user-assigned priority level?***

Answer: The interrupt source with the highest natural order priority will take precedence. The natural order priority is determined by the Interrupt Vector Table (IVT) address for that source. Interrupt sources with a smaller IVT address have a higher natural order priority.

Question 2: ***Can the `DISI` instruction be used to disable all sources of interrupt and traps?***

Answer: The `DISI` instruction does not disable traps or priority level 7 interrupt sources. However, the `DISI` instruction can be used as a convenient way to disable all interrupt sources if no priority level 7 interrupt sources are enabled in the user's application.

8.8 RELATED APPLICATION NOTES

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the PIC24F device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the Interrupts are:

Title	Application Note #
No related application notes at this time.	

Note: Please visit the Microchip web site (www.microchip.com) for additional application notes and code examples for the PIC24F family of devices.

8.9 REVISION HISTORY

Revision A (April 2006)

This is the initial released revision of this document.

NOTES:

