Section 29. Interrupts (Part Two)

HIGHLIGHTS

This section of the manual contains the following major topics:

29.1 INTRODUCTION

This section contains device-specific information for the following devices:

- PIC24HJ12GP201
- PIC24HJ12GP202

The PIC24H Interrupt Controller module reduces the numerous peripheral interrupt request signals to a single interrupt request signal to the PIC24H CPU. The features of this module include:

- Up to eight processor exceptions and software traps
- Seven user-selectable priority levels
- Interrupt Vector Table (IVT) with up to 126 vectors
- Unique vector for each interrupt or exception source
- Fixed priority within a specified user priority level
- Alternate Interrupt Vector Table (AIVT) for debugging support
- Fixed interrupt entry and return latencies

29.1.1 Interrupt Vector Table

The Interrupt Vector Table (IVT), shown in Figure 29-1, resides in program memory starting at location 0x000004. The IVT contains 126 vectors consisting of eight non-maskable trap vectors plus up to 118 sources of interrupt. 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 Route (ISR).

29.1.2 Alternate Vector Table

The Alternate Interrupt Vector Table (AIVT) is located after the IVT, as shown in Figure 29-1. Access to the AIVT is provided by the Enable Alternate Interrupt Vector Table (ALTIVT) control bit in Interrupt Control Register 2 (INTCON2<15>). If the ALTIVT bit is set, all interrupt and exception processes 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 by providing a means to switch between an application and a support environment without requiring the interrupt vectors to be reprogrammed. 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.

29.1.3 Reset Sequence

A device Reset is not a true exception because the interrupt controller is not involved in the Reset process. The PIC24H device clears its registers in response to a Reset, which forces the Program Counter (PC) to zero. The processor then begins program execution at location 0x000000. The user application programs a `GOTO` instruction at the Reset address, which redirects program execution to the appropriate start-up routine.

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 29-1: Interrupt Vector Table

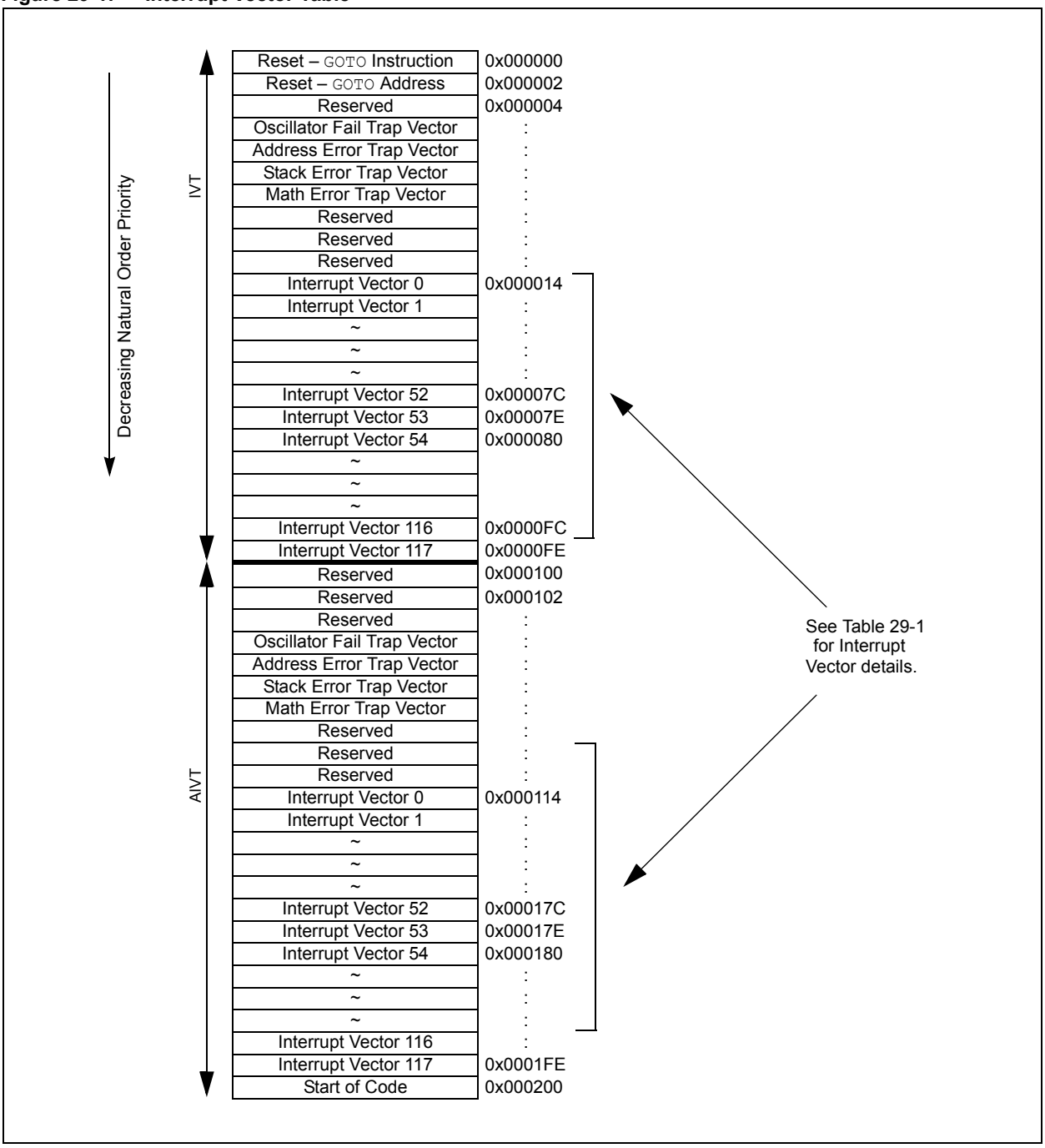


Table 29-1: Interrupt Vectors

Vector Number	IVT Address	AIVT Address	Interrupt 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	0x000014	0x000114	INT0 – External Interrupt 0
9	0x000016	0x000116	IC1 – Input Compare 1
10	0x000018	0x000118	OC1 – Output Compare 1
11	0x00001A	0x00011A	T1 – Timer1
12	0x00001C	0x00011C	Reserved
13	0x00001E	0x00011E	IC2 – Input Capture 2
14	0x000020	0x000120	OC2 – Output Compare 2
15	0x000022	0x000122	T2 – Timer2
16	0x000024	0x000124	T3 – Timer3
17	0x000026	0x000126	SPI1E – SPI1 Error
18	0x000028	0x000128	SPI1 – SPI1 Transfer Done
19	0x00002A	0x00012A	U1RX – UART1 Receiver
20	0x00002C	0x00012C	U1TX – UART1 Transmitter
21	0x00002E	0x00012E	AD1 – ADC1 Convert Done
22	0x000030	0x000130	Reserved
23	0x000032	0x000132	Reserved
24	0x000034	0x000134	SI2C1 – I2C1 Slave Events
25	0x000036	0x000136	MI2C1 – I2C1 Master Events
26	0x000038	0x000138	Reserved
27	0x00003A	0x00013A	CN – Change Notification Interrupt
28	0x00003C	0x00013C	INT1 – External Interrupt 1
29	0x00003E	0x00013E	Reserved
30	0x000040	0x000140	IC7 – Input Capture 7
31	0x000042	0x000142	IC8 – Input Capture 8
32	0x000044	0x000144	Reserved
33	0x000046	0x000146	Reserved
34	0x000048	0x000148	Reserved
35	0x00004A	0x00014A	Reserved
36	0x00004C	0x00014C	Reserved
37	0x00004E	0x00014E	INT2 – External Interrupt 2
38	0x000050	0x000150	Reserved
39	0x000052	0x000152	Reserved
40	0x000054	0x000154	Reserved
41	0x000056	0x000156	Reserved
42	0x000058	0x000158	Reserved
43	0x00005A	0x00015A	Reserved
44	0x00005C	0x00015C	Reserved
45	0x00005E	0x00015E	Reserved
46	0x000060	0x000160	Reserved
47	0x000062	0x000162	Reserved

Table 29-1: Interrupt Vectors (Continued)

Vector Number	IVT Address	AIVT Address	Interrupt Source
48	0x000064	0x000164	Reserved
49	0x000066	0x000166	Reserved
50	0x000068	0x000168	Reserved
51	0x00006A	0x00016A	Reserved
52	0x00006C	0x00016C	Reserved
53	0x00006E	0x00016E	Reserved
54	0x000070	0x000170	Reserved
55	0x000072	0x000172	Reserved
56	0x000074	0x000174	Reserved
57	0x000076	0x000176	Reserved
58	0x000078	0x000178	Reserved
59	0x00007A	0x00017A	Reserved
60	0x00007C	0x00017C	Reserved
61	0x00007E	0x00017E	Reserved
62	0x000080	0x000180	Reserved
63	0x000082	0x000182	Reserved
64	0x000084	0x000184	Reserved
65	0x000086	0x000186	Reserved
66	0x000088	0x000188	Reserved
67	0x00008A	0x00018A	Reserved
68	0x00008C	0x00018C	Reserved
69	0x00008E	0x00018E	Reserved
70	0x000090	0x000190	Reserved
71	0x000092	0x000192	Reserved
72	0x000094	0x000194	Reserved
73	0x000096	0x000196	U1E – UART1 Error
74	0x000098	0x000198	Reserved
75	0x00009A	0x00019A	Reserved
76	0x00009C	0x00019C	Reserved
77	0x00009E	0x00019E	Reserved
78	0x0000A0	0x0001A0	Reserved
79	0x0000A2	0x0001A2	Reserved
80	0x0000A4	0x0001A4	Reserved
81	0x0000A6	0x0001A6	Reserved
82	0x0000A8	0x0001A8	Reserved
83-125	0x0000AA-0x0000FE	0x0001AA-0x0001FE	Reserved

29.1.4 CPU Priority Status

The CPU can operate at one of 16 priority levels, 0-15. An interrupt or trap source must have a priority level greater than the current CPU priority to initiate an exception process. The peripheral and external interrupt sources for levels 0-7can be programmed. 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 29.2 "Non-Maskable Traps"**). The priority level for each trap source is fixed. Only one trap is assigned to a priority level. 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 status bits:

- CPU Interrupt Priority Level (IPL<2:0>) status bits in the CPU Status Register (SR<7:5>)
- CPU Interrupt Priority Level 3 (IPL3) status bit in the Core Control (CORCON<3>) register.

Because the IPL<2:0> status bits are readable and writable, the user application can modify these bits to disable all sources of interrupts below a given priority level. For example, if IPL<2:0> = 3, 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 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 application. In some applications, you might need 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 29.2.4.2 "Interrupt Nesting"** for more information.

29.1.5 Interrupt Priority

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

More than one interrupt request source can be assigned to a specific priority level. To resolve priority conflicts within a given user application-assigned level, each source of interrupt has a natural order priority based on its location in the IVT. Table 29-1 shows the location of each interrupt source in the IVT. The lower numbered interrupt vectors have higher natural priority, while the higher numbered vectors have lower natural priority. The overall priority level for any pending source of interrupt is determined first by the user application-assigned priority of that source in the IPCx 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 application-assigned priority level. Once the priority conflict is resolved and the exception process begins, the CPU can be interrupted only by a source with higher user application-assigned priority. Interrupts with the same user application-assigned priority but a higher natural order priority that become pending during the exception process remain pending until the current exception process completes.

Assigning each interrupt source to one of seven priority levels enables the user application to give an interrupt with a low natural order priority, a very high overall priority level. For example, Timer 2 can be given a priority of 7 and the External Interrupt 0 (INT0) can be assigned to priority level 1, giving it a very low effective priority.

Note:	The peripherals and sources of interrupt available in the IVT vary depending on the specific PIC24H device. The sources of interrupt shown in this document represent a comprehensive listing of all interrupt sources found on PIC24H devices. Refer to the specific device data sheet for further details.

29.2 NON-MASKABLE TRAPS

Traps are non-maskable, nestable interrupts that adhere to a fixed priority structure. Traps provide a means to correct erroneous operation during debugging and operation of the application. If the user application does not intend to correct a trap error condition, these vectors must be loaded with the address of a software routine to reset the device. Otherwise, the user application programs the trap vector with the address of a service routine that corrects the trap condition.

The PIC24H has four implemented sources of non-maskable traps:

- Oscillator Failure Trap
- Stack Error Trap
- Address Error Trap
- Math Error Trap

For many of the trap conditions the instruction that caused the trap is allowed to complete before exception processing begins. Therefore, the user application 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 a math error trap has the lowest priority (see Figure 29-1). In addition, trap sources are classified into two distinct categories: ‘hard’ traps and ‘soft’ traps.

29.2.1 Soft Traps

The math 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 two cycles to be sampled and acknowledged prior to exception processing. Therefore, additional instructions may be executed before a soft trap is acknowledged.

29.2.1.1 STACK ERROR TRAP (SOFT TRAP, LEVEL 12)

The stack is initialized to 0x0800 during a Reset. A stack error trap is generated if the stack pointer address is less than 0x0800.

A Stack Limit (SPLIM) register associated with the stack pointer is uninitialized at Reset. The stack overflow check is not enabled until a word is written to the SPLIM register.

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

A stack error can be detected in software by polling the Stack Error Trap (STKERR) status bit (INTCON1<2>). To avoid re-entering the trap service routine, the STKERR status flag must be cleared in software with a `RETFIE` (Return from Interrupt) instruction before the program returns from the trap.

29.2.1.2 MATH ERROR TRAP (SOFT TRAP, LEVEL 11)

The following event will generate a math error trap:

- Divide by zero

Divide-by-zero traps cannot be disabled. The divide-by-zero check is performed during the first iteration of the REPEAT loop that executes the divide instruction. The Math Error Status (DIV0ERR) bit (INTCON1<6>) is set when this trap is detected.

A math error trap can be detected in software by polling the Math Error Status (MATHERR) bit (INTCON1<4>). To avoid re-entering the trap service routine, the MATHERR status flag must be cleared in software with a `RETFIE` instruction before the program returns from the trap. Before the MATHERR status bit can be cleared, all conditions that caused the trap to occur must also be cleared.

29.2.2 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 are 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 does not resume until the trap has been acknowledged and processed.

29.2.2.1 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 is suspended. The higher-priority trap is acknowledged and processed. The lower-priority trap remains pending until processing of the higher-priority trap completes.

Each hard trap that occurs must be acknowledged before code execution of any type can continue. If a lower-priority hard trap occurs while a higher-priority trap is pending, acknowledged or is being processed, a hard-trap 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 Trap Reset Flag (TRAPR) status bit in the RESET Control Register (RCON<15> in the Reset module) is set when the Reset occurs so that the condition can be detected in software.

29.2.2.2 OSCILLATOR FAILURE TRAP (HARD TRAP, LEVEL 14)

An oscillator failure trap event is generated for any of these reasons:

- The Fail-Safe Clock Monitor (FSCM) is enabled and has detected a loss of the system clock source.
- A loss of PLL lock has been detected during normal operation using the PLL.
- The FSCM is enabled and the PLL fails to achieve lock at a Power-on Reset (POR).

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

29.2.2.3 ADDRESS ERROR TRAP (HARD TRAP, LEVEL 13)

Operating conditions that can generate an address error trap include:

- A misaligned data word fetch is attempted. This condition occurs when an instruction performs a word access with the Least Significant bit (LSb) of the effective address set to '`1`'. The PIC24H CPU requires all word accesses to be aligned to an even address boundary.
- A bit manipulation instruction uses the Indirect Addressing mode with the LSb of the effective address set to '`1`'.
- A data fetch is attempted from unimplemented data address space.
- Execution of a `BRA #literal` instruction or a `GOTO #literal` instruction, where `literal` is an unimplemented program memory address.
- Execution of instructions after the Program Counter has been modified to point to unimplemented program memory addresses. The Program Counter can be modified by loading a value into the stack and executing a `RETURN` instruction.

When an address error trap occurs, data space writes are inhibited 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 with a `RETFIE` instruction before the program returns from the trap.

29.2.3 Disable Interrupts Instruction

The DISI (disable interrupts) instruction can disable interrupts for up to 16384 instruction cycles. This instruction is useful for executing time-critical code segments. The DISI instruction only disables interrupts with priority levels 1-6. Priority level 7 interrupts and all trap events can still interrupt the CPU when the DISI instruction is active.

The DISI instruction works in conjunction with the Disable Interrupts Count (DISICNT) register in the CPU. 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 zero, priority level 1-6 interrupts are re-enabled. The value specified in the DISI instruction includes all cycles due to PSV accesses, instruction stalls, etc.

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

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 application can extend the interrupt disable time by modifying the contents of DISICNT.

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

The DISI Instruction (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.

29.2.4 Interrupt Operation

All interrupt event flags are sampled during each instruction cycle. A pending Interrupt Request (IRQ) is indicated by the flag bit = 1 in an IFSx register. The IRQ causes an interrupt if the corresponding bit in the Interrupt Enable (IECx) 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 is aborted when the CPU responds to the IRQ. The instruction in progress when the IRQ is sampled is completed before the ISR is executed.

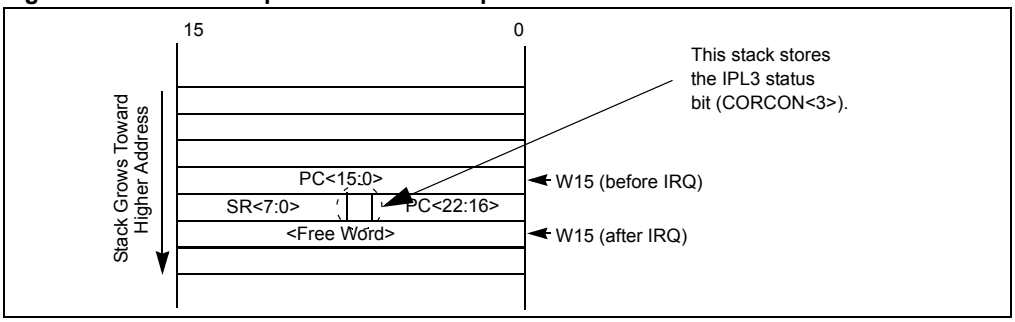
If there is a pending IRQ with a user-assigned priority level greater than the current processor priority level, indicated by the IPL<2:0> status bits (SR<7:5>), an interrupt is presented to the processor. The processor then saves the following information on the software stack:

- Current PC value
- Low byte of the Processor Status register (SRL)
- IPL3 status bit (CORCON<3>)

These three values allow the return Program Counter address value, MCU status bits and the current processor priority level to be automatically saved.

After this 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 disables all interrupts of lower or equal priority until the ISR is terminated using the RETFIE instruction.

Figure 29-2: Stack Operation for Interrupt Event



29.2.4.1 RETURN FROM INTERRUPT

The `RETFIE` instruction unstacks the PC return address, IPL3 status bit and SRL register to return the processor to the state and priority level that existed before the interrupt sequence.

29.2.4.2 INTERRUPT NESTING

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

The IPL<2:0> bits (SR<7:5>) 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.

29.2.5 Wake-up from Sleep and Idle

Any source of interrupt that is individually enabled, using its corresponding control bit in the IECx 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 by the corresponding bit in the IEC Control registers, a wake-up signal is sent to the PIC24H CPU. When the device wakes from Sleep or Idle mode, one of two actions occur:

- If the interrupt priority level for that source is greater than the current CPU priority level, the processor will process the interrupt and branch to the ISR for the interrupt source.
- If the user application-assigned interrupt priority level for the source is lower than or equal to the current CPU priority level, the processor will 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 user application must assign the CPU priority level for the interrupt to level 1 or greater.

29.2.6 A/D Converter External Conversion Request

The INT0 external interrupt request pin is shared with the A/D converter as an external conversion request signal. The INT0 interrupt source has programmable edge polarity, which is also available to the A/D converter external conversion request feature.

29.2.7 External Interrupt Support

The PIC24H supports up to five external interrupt pin sources (INT0-INT2). Each external interrupt pin has edge detection circuitry to detect the interrupt event. The INTCON2 register has five control bits (INT0EP-INT2EP) that select the polarity of the edge detection circuitry. Each external interrupt pin can be programmed to interrupt the CPU on a rising edge or falling edge event. See Register 29-4 for further details.

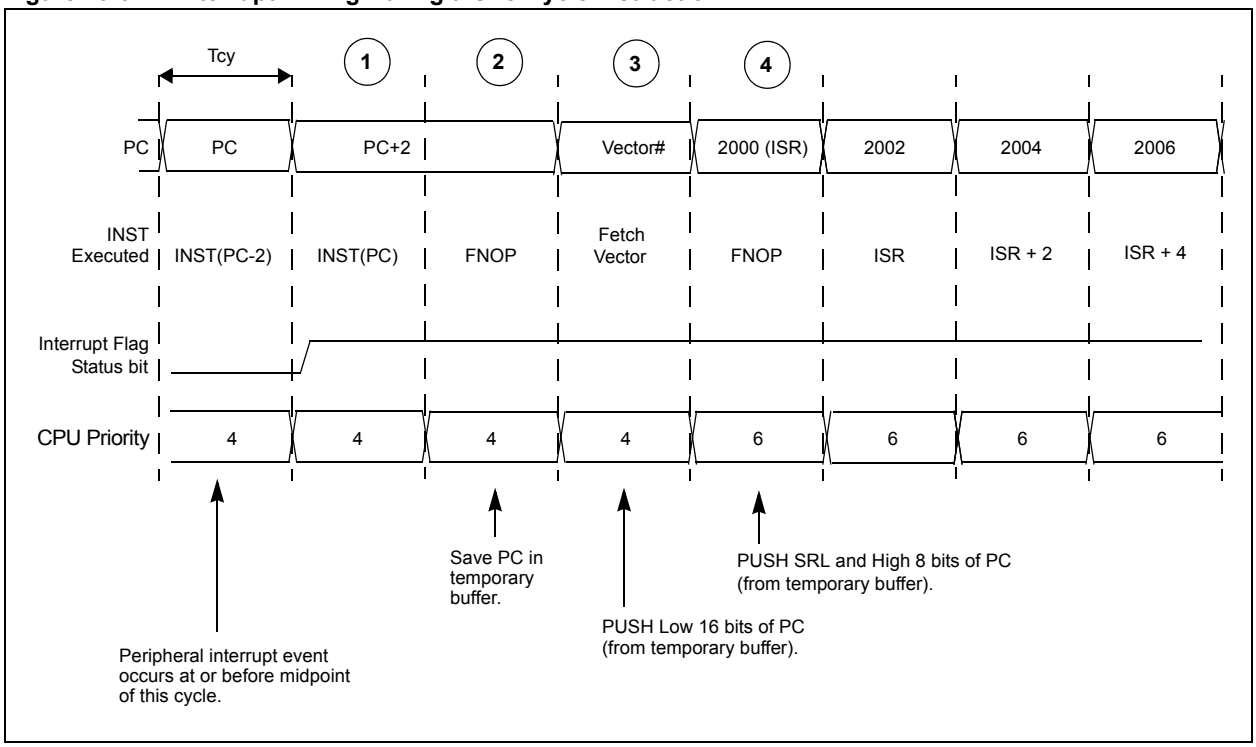
29.3 INTERRUPT PROCESSING TIMING

29.3.1 Interrupt Latency for One-Cycle Instructions

Figure 29-3 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 29-3 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 Program Counter (PC) and Lower-Byte Status (SRL) registers are saved into a temporary buffer register. The second cycle of the interrupt process is executed as a NOP instruction to maintain consistency with the sequence taken during a two-cycle instruction (see **Section 29.3.2 "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 instruction while the first instruction in the ISR is fetched.

Figure 29-3: Interrupt Timing During a One-Cycle Instruction



29.3.2 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 29-4 shows the peripheral interrupt event occurring in the instruction cycle prior to execution of the two-cycle instruction.

Figure 29-5 shows the timing when a peripheral interrupt coincides with the first cycle of a two-cycle instruction. In this case, the interrupt process completes as for a one-cycle instruction (see **Section 29.3.1 "Interrupt Latency for One-Cycle Instructions"**).

Figure 29-4: Interrupt Timing During a Two-Cycle Instruction

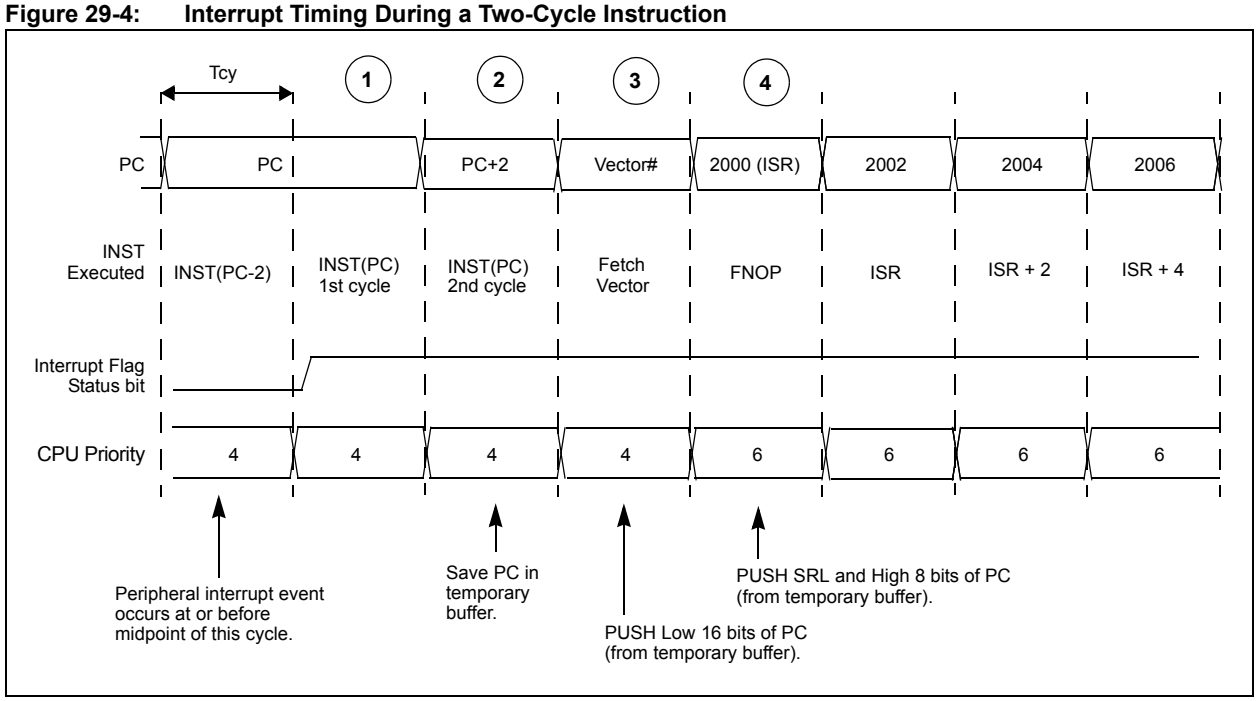
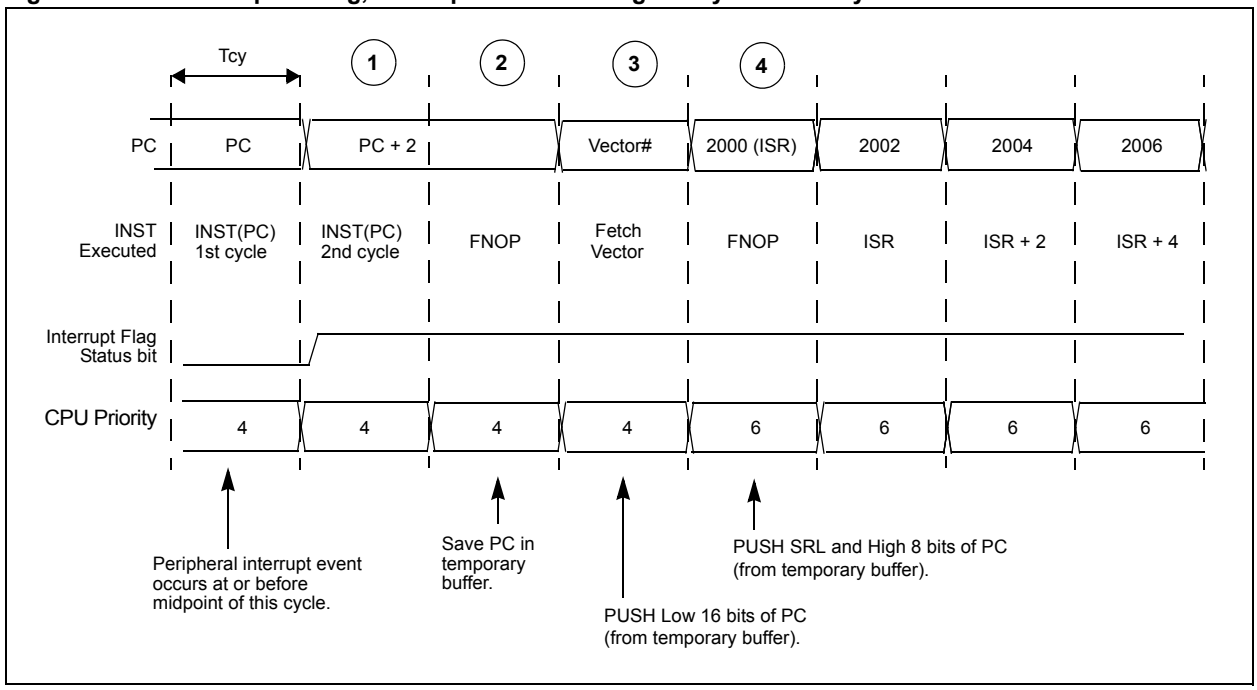


Figure 29-5: Interrupt Timing, Interrupt Occurs During 1st Cycle of a 2-Cycle Instruction

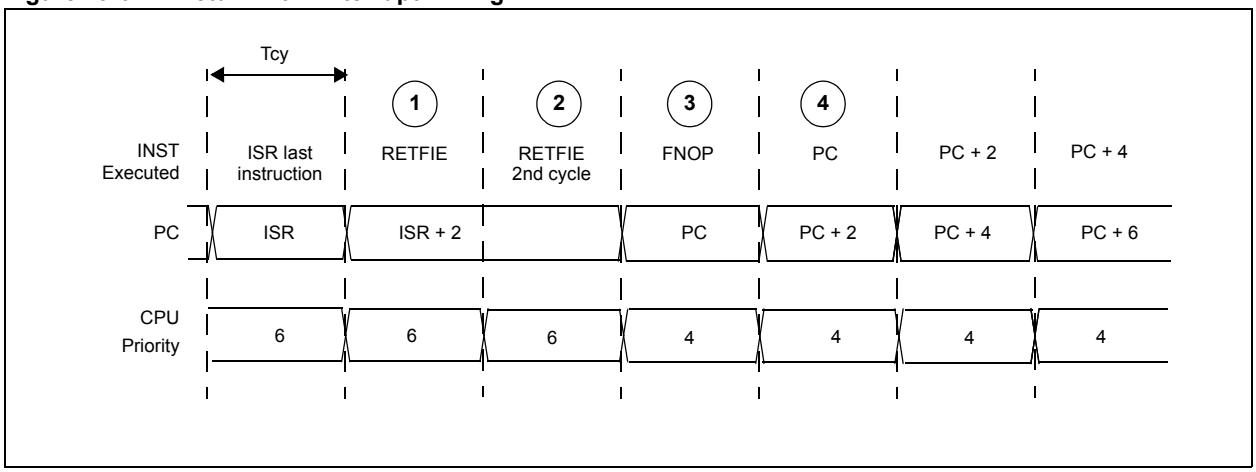


29.3.3 Returning from Interrupt

To return from an interrupt, the program must call the RETFIE instruction.

During the first two cycles of a RETFIE instruction, the contents of the PC and the SRL register are popped from the stack. The third instruction cycle is used to fetch the instruction addressed by the updated program counter. This cycle executes as a NOP instruction. On the fourth cycle, program execution resumes at the point where the interrupt occurred.

Figure 29-6: Return from Interrupt Timing



29.3.4 Special Conditions for Interrupt Latency

The PIC24H allows the current instruction to complete when a peripheral interrupt source becomes pending. The interrupt latency is the same for both one- and two-cycle instructions. However, certain conditions can increase interrupt latency by one cycle, depending on when the interrupt occurs. If a fixed latency is critical to the application, you should avoid these conditions:

- Executing a MOV.D instruction that uses PSV to access a value in program memory space
- Appending an instruction stall cycle to any two-cycle instruction
- Appending an instruction stall cycle to any one-cycle instruction that performs a PSV access
- A bit test and skip instruction (BTSC, BTSS) that uses PSV to access a value in the program memory space

29.4 INTERRUPT CONTROL AND STATUS REGISTERS

The following registers are associated with the interrupt controller:

- **INTCON1, INTCON2 Registers**

  These two registers control global interrupt functions:
  - INTCON1 contains the Interrupt Nesting Disable (NSTDIS) bit, as well as the control and status flags for the processor trap sources
  - INTCON2 controls external interrupt request signal behavior and use of the alternate vector table
- **IFSx: Interrupt Flag Status Registers**

  All interrupt request flags are maintained in the IFSx registers, where 'x' denotes the register number. Each source of interrupt has a status bit, which is set by the respective peripherals or external signal and cleared by software.
- **IECx: Interrupt Enable Control Registers**

  All Interrupt Enable Control bits are maintained in the IECx registers, where 'x' denotes the register number. These control bits are used to individually enable interrupts from the peripherals or external signals.
- **IPCx: Interrupt Priority Control Registers**

  Each user interrupt source can be assigned to one of eight priority levels. The IPC registers 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 application can 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 application.
- **INTTREG: Interrupt Control and Status Register**

  The INTTREG register contains the associated interrupt vector number and the new CPU interrupt priority level, which are latched into Vector Number (VECNUM<6:0>) and Interrupt level (ILR<3:0>) bit fields in the INTTREG register. The new interrupt priority level is the priority of the pending interrupt

Each register is described in detail in the following sections.

Note:	The total number and type of interrupt sources depend on the device variant. Refer to the specific device data sheet for further details.

29.4.1 Assignment of Interrupts to Control Registers

The interrupt sources are assigned to the IFSx, IECx and IPCx registers in the same sequence that they are listed in Table 29-1. For example, the INT0 (External Interrupt 0) source has vector number and natural order priority 0. Thus, the External Interrupt 0 Flag Status (INT0IF) bit is found in IFS0<0>. The INT0 interrupt uses bit 0 of the IEC0 register as its Enable bit. The IPC0<2:0> bits assign the interrupt priority level for the INT0 interrupt.

Register 29-1: SR: CPU STATUS Register

R-0	R-0	R/C-0	R/C-0	R-0	R/C-0	R -0	R/W-0
OA	OB	SA	SB	OAB	SAB	DA	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<2:0>			RA	N	OV	Z	C
bit 7							bit 0

Legend:		
C = Clear only bit	R = Readable bit	U = Unimplemented bit, read as '0'
S = Set only bit	W = Writable bit	-n = Value at POR
'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-8	Not used by Interrupt Controller. (See the *"dsPIC30F/33F Programmer's Reference Manual"* (DS70157) for descriptions of SR bits)
bit 7-5	**IPL<2:0>:** 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)
bit 4-0	Not used by Interrupt Controller. (See the *"dsPIC30F/33F Programmer's Reference Manual"* (DS70157) for descriptions of SR bits)

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. User interrupts are disabled when IPL<3> = 1.

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

Register 29-2: CORCON: Core Control Register

U-0	U-0	U-0	R/W-0	R/W-0	R-0	R-0	R-0
—	—	—	US	EDT	DL<1:0>		
bit 15							bit 8

R/W-0	R/W-0	R/W-1	R/W-0	R/C-0	R/W-0	R/W-0	R/W-0
SATA	SATB	SATDW	ACCSAT	**IPL3**	PSV	RND	IF
bit 7							bit 0

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

bit 15-4	**Not used by the Interrupt Controller** (See "*dsPIC30F/33F Programmer's Reference Manual* (DS70157)" for descriptions of CORCON bits)
bit 3	**IPL3:** CPU Interrupt Priority Level Status bit 3 1 = CPU interrupt priority level is greater than 7 0 = CPU interrupt priority level is 7 or less **Note:** The IPL3 bit is concatenated with the IPL<2:0> bits (SR<7:5>) to form the CPU interrupt priority level.
bit 2-0	**Not used by the Interrupt Controller** (See "*dsPIC30F/33F Programmer's Reference Manual* (DS70157)" for descriptions of CORCON bits)

Register 29-3: 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	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
—	DIV0ERR	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'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 14-7 **Unimplemented:** Read as '0'

bit 6 **DIV0ERR:** Arithmetic Error Status bit
1 = Math error trap was caused by a divide by zero
0 = Math error trap was not caused by a divide by zero

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

bit 4 **MATHERR:** Arithmetic Error Status bit
1 = Math error trap has occurred
0 = Math error 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 29-4: 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	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	INT2EP	INT1EP	INT0EP
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'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 (disable interrupts) Instruction Status bit
1 = DISI instruction is active
0 = DISI instruction is not active

bit 13-3 **Unimplemented:** Read as '0'

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 29-5: IFS0: Interrupt Flag Status Register 0

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	AD1IF	U1TXIF	U1RXIF	SPI1IF	SPI1EIF	T3IF
bit 15							bit 8

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
T2IF	OC2IF	IC2IF	—	T1IF	OC1IF	IC1IF	INT0IF
bit 7							bit 0

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

bit 15-14 **Unimplemented:** Read as '0'

bit 13 **AD1IF:** ADC1 Conversion Complete Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 12 **U1TXIF:** UART1 Transmitter Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 11 **U1RXIF:** UART1 Receiver Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 10 **SPI1IF:** SPI1 Event Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 9 **SPI1EIF:** SPI1 Fault Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 8 **T3IF:** Timer3 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 7 **T2IF:** Timer2 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 6 **OC2IF:** Output Compare Channel 2 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 5 **IC2IF:** Input Capture Channel 2 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 4 **Unimplemented:** Read as '0'

bit 3 **T1IF:** Timer1 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 2 **OC1IF:** Output Compare Channel 1 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

Register 29-5: IFS0: Interrupt Flag Status Register 0 (Continued)

bit 1 **IC1IF:** Input Capture Channel 1 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 0 **INT0IF:** External Interrupt 0 Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

Register 29-6: IFS1: Interrupt Flag Status Register 1

U-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
—	—	INT2IF	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
IC8IF	IC7IF	—	INT1IF	CNIF	—	MI2C1IF	SI2C1IF
bit 7							bit 0

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

bit 15-14 **Unimplemented:** Read as '0'

bit 13 **INT2IF:** External Interrupt 2 Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 12-8 **Unimplemented:** Read as '0'

bit 7 **IC8IF:** Input Capture Channel 8 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 6 **IC7IF:** Input Capture Channel 7 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

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

bit 4 **INT1IF:** External Interrupt 1 Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 3 **CNIF:** Input Change Notification Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 2 **Unimplemented:** Read as '0'

bit 1 **MI2C1IF:** I2C1 Master Events Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 0 **SI2C1IF:** I2C1 Slave Events Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

Register 29-7: IFS4: Interrupt Flag Status Register 4

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	U-0	U-0	U-0	U-0
—	—	—	—	—	—	U1EIF	—
bit 7							bit 0

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

bit 15-2 **Unimplemented:** Read as '0'

bit 1 **U1EIF:** UART1 Error Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

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

Register 29-8: IEC0: Interrupt Enable Control Register 0

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	AD1IE	U1TXIE	U1RXIE	SPI1IE	SPI1EIE	T3IE
bit 15							bit 8

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
T2IE	OC2IE	IC2IE	—	T1IE	OC1IE	IC1IE	INT0IE
bit 7							bit 0

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

bit 15-14 **Unimplemented:** Read as '0'

bit 13 **AD1IE:** ADC1 Conversion Complete Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 12 **U1TXIE:** UART1 Transmitter Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 11 **U1RXIE:** UART1 Receiver Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 10 **SPI1IE:** SPI1 Event Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 9 **SPI1EIE:** SPI1 Event Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 8 **T3IE:** Timer3 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 7 **T2IE:** Timer2 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 6 **OC2IE:** Output Compare Channel 2 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 5 **IC2IE:** Input Capture Channel 2 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 4 **Unimplemented:** Read as '0'

bit 3 **T1IE:** Timer1 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 2 **OC1IE:** Output Compare Channel 1 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

Register 29-8: IEC0: Interrupt Enable Control Register 0 (Continued)

bit 1 **IC1IE:** Input Capture Channel 1 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 0 **INT0IE:** External Interrupt 0 Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

Register 29-9: IEC1: Interrupt Enable Control Register 1

U-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
—	—	INT2IE	—	—	—	—	—
bit 15							bit 8

R/W-0	R/W-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
IC8IE	IC7IE	—	INT1IE	CNIE	—	MI2C1IE	SI2C1IE
bit 7							bit 0

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

bit 15-14 **Unimplemented:** Read as '0'

bit 13 **INT2IE:** External Interrupt 2 Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 12-8 **Unimplemented:** Read as '0'

bit 7 **IC8IE:** Input Capture Channel 8 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 6 **IC7IE:** Input Capture Channel 7 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

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

bit 4 **INT1IE:** External Interrupt 1 Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 3 **CNIE:** Input Change Notification Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 2 **Unimplemented:** Read as '0'

bit 1 **MI2C1IE:** I2C1 Master Events Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 0 **SI2C1IE:** I2C1 Slave Events Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

Register 29-10: IEC4: Interrupt Enable Control Register 4

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	U-0	U-0	R/W-0	U-0
—	—	—	—	—	—	U1EIE	—
bit 7							bit 0

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

bit 15-2 **Unimplemented:** Read as '0'

bit 1 **U1EIE:** UART1 Error Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

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

Register 29-11: IPC0: Interrupt Priority Control Register 0

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	T1IP<2:0>			—	OC1IP<2:0>		
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
—	IC1IP<2:0>			—	INT0IP<2:0>		
bit 7							bit 0

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

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

bit 14-12 **T1IP<2:0>:** Timer1 Interrupt Priority bits
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 **OC1IP<2:0>:** Output Compare Channel 1 Interrupt Priority bits
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 **IC1IP<2:0>:** Input Capture Channel 1 Interrupt Priority bits
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 **INT0IP<2:0>:** External Interrupt 0 Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

Register 29-12: IPC1: Interrupt Priority Control Register 1

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	T2IP<2:0>			—	OC2IP<2:0>		
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	U-1	U-0	U-0
—	IC2IP<2:0>			—	—	—	—
bit 7							bit 0

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

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

bit 14-12 **T2IP<2:0>:** Timer2 Interrupt Priority bits
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 **OC2IP<2:0>:** Output Compare Channel 2 Interrupt Priority bits
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 **IC2IP<2:0>:** Input Capture Channel 2 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

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

Register 29-13: IPC2: Interrupt Priority Control Register 2

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	U1RXIP<2:0>			—	SPI1IP<2:0>		
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
—	SPI1EIP<2:0>			—	T3IP<2:0>		
bit 7							bit 0

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

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

bit 14-12 **U1RXIP<2:0>:** UART1 Receiver Interrupt Priority bits
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 **SPI1IP<2:0>:** SPI1 Event Interrupt Priority bits
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 **SPI1EIP<2:0>:** SPI1 Error Interrupt Priority bits
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 **T3IP<2:0>:** Timer3 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

Register 29-14: IPC3: Interrupt Priority Control Register 3

U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-0	R/W-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
—	AD1IP<2:0>			—	U1TXIP<2:0>		
bit 7							bit 0

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

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

bit 6-4 **AD1IP<2:0>:** ADC1 Conversion Complete Interrupt Priority bits

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 **U1TXIP<2:0>:** UART1 Transmitter Interrupt Priority bits

111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

Register 29-15: IPC4: Interrupt Priority Control Register 4

U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	CNIP<2:0>			—	—	—	—
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
—	MI2C1IP<2:0>			—	SI2C1IP<2:0>		
bit 7							bit 0

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

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

bit 14-12 **CNIP<2:0>:** Change Notification Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 11-7 **Unimplemented:** Read as '0'

bit 6-4 **MI2C1IP<2:0>:** I2C1 Master Events Interrupt Priority bits
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 **SI2C1IP<2:0>:** I2C1 Slave Events Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

Register 29-16: IPC5: Interrupt Priority Control Register 5

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	IC8IP<2:0>			—	IC7IP<2:0>		
bit 15							bit 8

U-0	U-1	U-0	U-0	U-0	R/W-1	R/W-0	R/W-0
—	—	—	—	—	INT1IP<2:0>		
bit 7							bit 0

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

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

bit 14-12 **IC8IP<2:0>:** Input Capture Channel 8 Interrupt Priority bits
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 **IC7IP<2:0>:** Input Capture Channel 7 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 7-3 **Unimplemented:** Read as '0'

bit 2-0 **INT1IP<2:0>:** External Interrupt 1 Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

Register 29-17: IPC7: Interrupt Priority Control Register 7

U-0	U-1	U-0	U-0	U-0	U-1	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	INT2IP<2:0>			—	—	—	—
bit 7							bit 0

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

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

bit 6-4 **INT2IP<2:0>:** External Interrupt 2 Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

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

Register 29-18: IPC16: Interrupt Priority Control Register 16

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	U-0	U-0	U-0
—	U1EIP<2:0>			—	—	—	—
bit 7							bit 0

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

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

bit 6-4 **U1EIP<2:0>:** UART1 Error Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

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

Register 29-19: INTTREG: Interrupt Control and Status Register

U-0	U-0	U-0	U-0	R-0	R-0	R-0	R-0
—	—	—	—	ILR<3:0>			
bit 15							bit 8

U-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
—	VECNUM<6:0>						
bit 7							bit 0

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

bit 15-12 **Unimplemented:** Read as '0'

bit 11-8 **ILR:** New CPU Interrupt Priority Level bits
1111 = CPU Interrupt Priority Level is 15
•
•
•
0001 = CPU Interrupt Priority Level is 1
0000 = CPU Interrupt Priority Level is 0

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

bit 6-0 **VECNUM:** Vector Number of Pending Interrupt bits
0111111 = Interrupt Vector pending is number 135
•
•
•
0000001 = Interrupt Vector pending is number 9
0000000 = Interrupt Vector pending is number 8

29.5 INTERRUPT SETUP PROCEDURES

29.5.1 Initialization

To configure an interrupt source:

1. Set the NSTDIS control bit (INTCON1<15>) if you do not plan to use nested interrupts.
2. Select the user-assigned priority level for the interrupt source by writing the control bits in the appropriate IPCx Control register. The priority level depends on the specific application and type of interrupt source. If you do not plan to use multiple priority levels, program the IPCx register control bits for all enabled interrupt sources to the same non-zero value.

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

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

29.5.2 Interrupt Service Routine

The method used to declare an Interrupt Service Routine (ISR) and initialize the Interrupt Vector Table with the correct vector address depends on the programming language (C or Assembler) and the language development tool suite used to develop the application. In general, the user application must clear the interrupt flag in the appropriate IFSx register for the source of interrupt that the ISR handles. Otherwise, the application will immediately re-entered the ISR after it exits the routine. If you code the ISR in Assembler, it must be terminated using a `RETFIE` instruction to unstack the saved PC value, SRL value and old CPU priority level.

29.5.3 Trap Service Routine

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

29.5.4 Interrupt Disable

To disable interrupts:

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 interrupts, you can use the `POP` instruction to restore the previous SR value.

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

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

29.5.5 Code Example

Example 29-1 illustrates code that enables nested interrupts and sets up Timer1, Timer2, Timer3, and change notice peripherals to priority levels 2, 5, 6 and 4, respectively. It also illustrates how interrupts can be enabled and disabled using the Status Register. Sample ISRs illustrate interrupt clearing.

Example 29-1: Nested Interrupt Code Example

```
void enableInterrupts(void)
{
    /* Set CPU IPL to 0, enable level 1-7 interrupts */
    /* No restoring of previous CPU IPL state performed here */
    SRbits.IPL = 0;

    return;
}

void disableInterrupts(void)
{
    /* Set CPU IPL to 7, disable level 1-7 interrupts */
    /* No saving of current CPU IPL setting performed here */
    SRbits.IPL = 7;

    return;
}

void initInterrupts(void)
{
    /* Interrupt nesting enabled here */
    INTCON1bits.NSTDIS = 0;

    /* Set Timer3 interrupt priority to 6 (level 7 is highest) */
    IPC2bits.T3IP = 6;

    /* Set Timer2 interrupt priority to 5 */
    IPC1bits.T2IP = 5;

    /* Set Change Notice interrupt priority to 4 */
    IPC4bits.CNIP = 4;

    /* Set Timer1 interrupt priority to 2 */
    IPC0bits.T1IP = 2;

    /* Reset Timer1 interrupt flag */
    IFS0bits.T1IF = 0;

    /* Reset Timer2 interrupt flag */
    IFS0bits.T2IF = 0;

    /* Reset Timer3 interrupt flag */
    IFS0bits.T3IF = 0;

    /* Enable CN interrupts */
    IEC1bits.CNIE = 1;
```

Example 29-1: Interrupt Setup Code Example (Continued)

```
    /* Enable Timer1 interrupt */
    IEC0bits.T1IE = 1;

    /* Enable Timer2 interrupt (PWM time base) */
    IEC0bits.T2IE = 1;

    /* Enable Timer3 interrupt */
    IEC0bits.T3IE = 1;

    /* Reset change notice interrupt flag */
    IFS1bits.CNIF = 0;

    return;
}

void __attribute__((__interrupt__)) _T1Interrupt(void)
{
    /* Insert ISR Code Here*/

    /* Clear Timer1 interrupt */
    IFS0bits.T1IF = 0;
}

void __attribute__((__interrupt__)) _T2Interrupt(void)
{
    /* Insert ISR Code Here*/

    /* Clear Timer2 interrupt */
    IFS0bits.T2IF = 0;
}

void __attribute__((__interrupt__)) _T3Interrupt(void)
{
    /* Insert ISR Code Here*/

    /* Clear Timer3 interrupt */
    IFS0bits.T3IF = 0;
}

void __attribute__((__interrupt__)) _CNInterrupt(void)
{
    /* Insert ISR Code Here*/

    /* Clear CN interrupt */
    IFS1bits.CNIF = 0;
}
```


Table 29-2: Interrupt Controller Register Map

SFR Name	SFR Addr	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	All Resets
INTCON1	0080	NSTDIS	—	—	—	—	—	—	—	—	DIV0ERR	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—	0000
INTCON2	0082	ALTIVT	DISI	—	—	—	—	—	—	—	—	—	—	—	INT2EP	INT1EP	INT0EP	0000
IFS0	0084	—	—	AD1IF	U1TXIF	U1RXIF	SPI1IF	SPI1EIF	T3IF	T2IF	OC2IF	IC2IF	—	T1IF	OC1IF	IC1IF	INT0IF	0000
IFS1	0086	—	—	INT2IF	—	—	—	—	—	IC8IF	IC7IF	—	INT1IF	CNIF	—	MI2C1IF	SI2C1IF	0000
IFS4	008C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	U1EIF	—	0000
IEC0	0094	—	—	AD1IE	U1TXIE	U1RXIE	SPI1IE	SPI1EIE	T3IE	T2IE	OC2IE	IC2IE	—	T1IE	OC1IE	IC1IE	INT0IE	0000
IEC1	0096	—	—	INT2IE	—	—	—	—	—	IC8IE	IC7IE	—	INT1IE	CNIE	—	MI2C1IE	SI2C1IE	0000
IEC4	009C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	U1EIE	—	0000
IPC0	00A4	—	T1IP<2:0>			—	OC1IP<2:0>			—	IC1IP<2:0>			—	INT0IP<2:0>			4444
IPC1	00A6	—	T2IP<2:0>			—	OC2IP<2:0>			—	IC2IP<2:0>			—	—	—	—	4444
IPC2	00A8	—	U1RXIP<2:0>			—	SPI1IP<2:0>			—	SPI1EIP<2:0>			—	T3IP<2:0>			4444
IPC3	00AA	—	—	—	—	—	—	—	—	—	AD1IP<2:0>			—	U1TXIP<2:0>			4444
IPC4	00AC	—	CNIP<2:0>			—	—	—	—	—	MI2C1IP<2:0>			—	SI2C1IP<2:0>			4444
IPC5	00AE	—	IC8IP<2:0>			—	IC7IP<2:0>			—	—	—	—	—	INT1IP<2:0>			4444
IPC7	00B2	—	—	—	—	—	—	—	—	—	INT2IP<2:0>			—	—	—	—	4444
IPC16	00C4	—	—	—	—	—	—	—	—	—	U1EIP<2:0>			—	—	—	—	4444
INTTREG	00E0	—	—	—	—	ILR<3:0>>				—	VECNUM<6:0>							0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

29.6 DESIGN TIPS

Question 1: ***What happens when two sources of interrupt become pending at the same time and have the same user application-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 lower 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.

29.7 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 PIC24H Product Family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the Interrupts module 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 PIC24H Family of devices.

29.8 REVISION HISTORY

Revision A (February 2007)

This is the initial release of this document.

