MICROCHIP

Section 28. Interrupts (Part 2)

HIGHLIGHTS

This section of the manual contains the following topics:

28.1 Introduction

The dsPIC® DSC SMPS (Switched Mode Power Supply) Interrupt controller reduces numerous peripheral interrupt requests to a single interrupt request signal to the CPU, and has the following features:

- Up to eight processor exceptions and software traps
- Eight user-selectable priority levels
- Interrupt Vector Table (IVT) with up to 62 vectors
- A 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

28.1.1 Interrupt Vector Table

The Interrupt Vector Table (IVT) is shown in Figure 28-1. The IVT resides in program memory, starting at location 0x000004. The IVT contains 62 vectors consisting of eight non-maskable trap vectors plus up to 54 sources of interrupt. In general, each interrupt source has its own vector. Each interrupt vector contains a 24-bit address. The value programmed into each interrupt vector location is the starting address of the associated Interrupt Service Routine (ISR).

28.1.2 Alternate Vector Table

The Alternate Interrupt Vector Table (AIVT) is located after the IVT. 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 efforts 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.

28.1.3 Reset Sequence

A device Reset is not a true exception because the interrupt controller is not involved in the reset process. The dsPIC DSC SMPS 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 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 28-1: Interrupt Vector Table

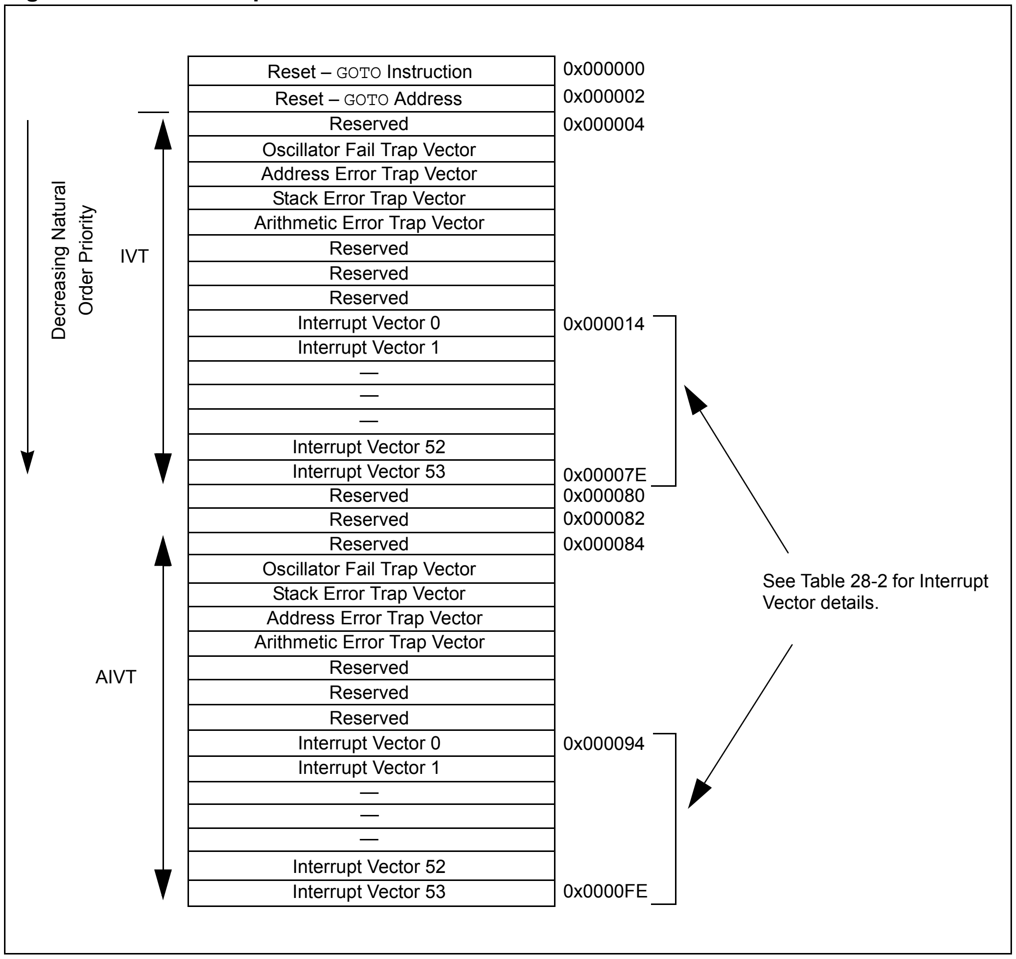


Table 28-1: Trap Vector Details

Vector Number	IVT Address	AIVT Address	Trap Source
0	0x000004	0x000084	Reserved
1	0x000006	0x000086	Oscillator Failure
2	0x000008	0x000088	Address Error
3	0x00000A	0x00008A	Stack Error
4	0x00000C	0x00008C	Arithmetic Error
5	0x00000E	0x00008E	Reserved
6	0x000010	0x000090	Reserved
7	0x000012	0x000092	Reserved

Table 28-2: Interrupt Vector Details

Vector Number	IVT Address	AIVT Address	Interrupt Source
8	0x000014	0x000094	INT0 – External Interrupt 0
9	0x000016	0x000096	IC1 – Input Compare 1
10	0x000018	0x000098	OC1 – Output Compare 1
11	0x00001A	0x00009A	T1 – Timer 1
12	0x00001C	0x00009C	Reserved
13	0x00001E	0x00009E	OC2 – Output Compare 2
14	0x000020	0x0000A0	T2 – Timer 2
15	0x000022	0x0000A2	T3 – Timer 3
16	0x000024	0x0000A4	SPI1
17	0x000026	0x0000A6	U1RX – UART1 Receiver
18	0x000028	0x0000A8	U1TX – UART1 Transmitter
19	0x00002A	0x0000AA	ADC – ADC Convert Done
20	0x00002C	0x0000AC	NVM – NVM Write Complete
21	0x00002E	0x0000AE	SI2C – I^{2}C™ Slave Event
22	0x000030	0x0000B0	MI2C – I^{2}C Master Event
23	0x000032	0x0000B2	Reserved
24	0x000034	0x0000B4	INT1 – External Interrupt 1
25	0x000036	0x0000B6	INT2 – External Interrupt 2
26	0x000038	0x0000B8	PWM Special Event Trigger
27	0x00003A	0x0000BA	PWM Gen#1
28	0x00003C	0x0000BC	PWM Gen#2
29	0x00003E	0x0000BE	PWM Gen#3
30	0x000040	0x0000C0	PWM Gen#4
31	0x000042	0x0000C2	Reserved
32	0x000044	0x0000C4	Reserved
33	0x000046	0x0000C6	Reserved
34	0x000048	0x0000C8	Reserved
35	0x00004A	0x0000CA	CN – Input Change Notification
36	0x00004C	0x0000CC	Reserved
37	0x00004E	0x0000CE	Analog Comparator 1
38	0x000050	0x0000D0	Analog Comparator 2
39	0x000052	0x0000D2	Analog Comparator 3
40	0x000054	0x0000D4	Analog Comparator 4
41	0x000056	0x0000D6	Reserved
42	0x000058	0x0000D8	Reserved
43	0x00005A	0x0000DA	Reserved
44	0x00005C	0x0000DC	Reserved
45	0x00005E	0x0000DE	ADC Pair 0 Conversion Done
46	0x000060	0x0000E0	ADC Pair 1 Conversion Done
47	0x000062	0x0000E2	ADC Pair 2 Conversion Done
48	0x000064	0x0000E4	ADC Pair 3 Conversion Done
49	0x000066	0x0000E6	ADC Pair 4 Conversion Done
50	0x000068	0x0000E8	ADC Pair 5 Conversion Done
51	0x00006A	0x0000EA	Reserved
52	0x00006C	0x0000EC	Reserved
53-61	0x00006E-0x00007E	0x0000EE-0x0000FE	Reserved
Lowest Natural Order Priority			

28.1.4 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 level 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 28.2 "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 Status (IPL<2:0>) bits located in the Status (SR<7:5>) register
- CPU Interrupt Priority Level Status (IPL3) bit located in the Core Control (CORCON<3>) register

The IPL<2:0> status bits are readable and writable, so 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, 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 28.2.4.2 "Interrupt Nesting"** for more information.

28.1.5 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 Interrupt Priority Control (IPCx) 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 (the lowest priority) and end at level 7 (the highest priority). If the IPC bits associated with an interrupt source are all cleared, the interrupt source is effectively disabled.

Note: If the application program re-configures the interrupt priority levels on the fly, it must disable the interrupts while doing so. Failure to disable interrupts can produce unexpected results.

Since more than one interrupt request source can 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. Table 28-2 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-assigned priority of that source in the Interrupt Priority Control (IPCx) register, and 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, which become pending after the exception process begins, 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, Timer2 can be given a priority of 7 and the External Interrupt 0 (INT0) can be assigned to priority level 1, thus giving it a very low effective priority.

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

28.2 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 that will reset the device. Otherwise, the user application programs the trap vector with the address of a service routine that corrects the trap condition.

The dsPIC DSC SMPS devices have four implemented sources of non-maskable traps:

- Oscillator Failure Trap
- Stack Error Trap
- Address Error Trap
- Arithmetic 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. In addition, trap sources are classified into two distinct categories: hard traps and soft traps.

28.2.1 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.

28.2.1.1 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 that is 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 in the Interrupt Control Register 1 (INTCON1<2>). To avoid re-entering the trap service routine, the STKERR status flag must be cleared in software before exiting the stack error trap.

28.2.1.2 Arithmetic Error Trap (Soft Trap, Level 11)

Any of the following events will generate a math error trap:

- Accumulator A overflow
- Accumulator B overflow
- Catastrophic accumulator overflow
- Divide by zero
- Shift Accumulator (`SFTAC`) operation that exceeds ±16 bits

Three bits in the INTCON1 register enable the three types of accumulator overflow traps:

- The Accumulator A Overflow Trap Flag (OVATE) control bit (INTCON1<10>) enables traps for an Accumulator A overflow event.
- The Accumulator B Overflow Trap Flag (OVBTE) control bit (INTCON1<9>) enables traps for an Accumulator B overflow event.
- The Catastrophic Overflow Trap Enable (COVTE) control bit (INTCON1<8>) enables traps for a catastrophic overflow of either accumulator. When this trap is detected, these corresponding ERROR bits are set in the INTCON1 register:
  - Accumulator A Overflow Trap Flag (OVAERR)
  - Accumulator B Overflow Trap Flag (OVBERR)
  - Accumulator A Catastrophic Overflow Trap Enable (COVAERR)
  - Accumulator B Catastrophic Overflow Trap Enable (COVBERR)

An Accumulator A or Accumulator B overflow event is defined as a carry-out from bit 31. No accumulator overflow can occur if the 31-bit Saturation mode is enabled for the accumulator. A catastrophic accumulator overflow is defined as a carry-out from bit 39 of either accumulator. No catastrophic overflow can occur if accumulator saturation (31-bit or 39-bit) is enabled.

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.

Accumulator shift traps cannot be disabled. The `SFTAC` instruction can be used to shift the accumulator by a literal value or a value in one of the W registers. If the shift value exceeds ±16 bits, an arithmetic trap is generated and the Shift Accumulator Error Status (SFTAERR) bit (INTCON1<7>) is set. The `SFTAC` instruction executes, but the results of the shift are not written to the target accumulator.

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 before exiting the math error trap. Before the MATHERR status bit can be cleared, all conditions that caused the trap to occur must also be cleared. If the trap was due to an accumulator overflow, the OA and OB status bits (SR<15:14>) must be cleared. The OA and OB status bits are read-only, so the user software must perform a dummy operation on the overflowed accumulator (such as adding '`0`'), which will cause the hardware to clear the OA or OB status bit.

28.2.2 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.

28.2.2.1 Trap Priority and Hard Trap Conflicts

If a higher-priority trap occurs while any lower-priority trap is in progress, processing of the trap with lower priority is suspended. The trap with higher priority is acknowledged and processed. The trap with lower priority remains pending until the higher priority trap is processed.

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 (RCON<15>) register in the Reset module is set when the Reset occurs, so that the condition can be detected in software.

28.2.2.2 Oscillator Failure Trap (Hard Trap, Level 14)

An oscillator failure trap event is generated for any of the following 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 before exiting the oscillator failure trap.

28.2.2.3 Address Error Trap (Hard Trap, Level 13)

Operating conditions that can generate an address error trap include the following:

- 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 dsPIC DSC SMPS device 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 Address Error Trap Status (ADDRERR) bit (INTCON1<3>). To avoid re-entering the Trap Service Routine, the ADDRERR status flag must be cleared in software before exiting the address error trap.

Note:	In the `MAC` class of instructions, the data space is split into X and Y spaces. In these instructions, unimplemented X space includes all of Y space, and unimplemented Y space includes all of X space.

28.2.3 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.

28.2.4 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 '`1`' in an Interrupt Flag Status (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.

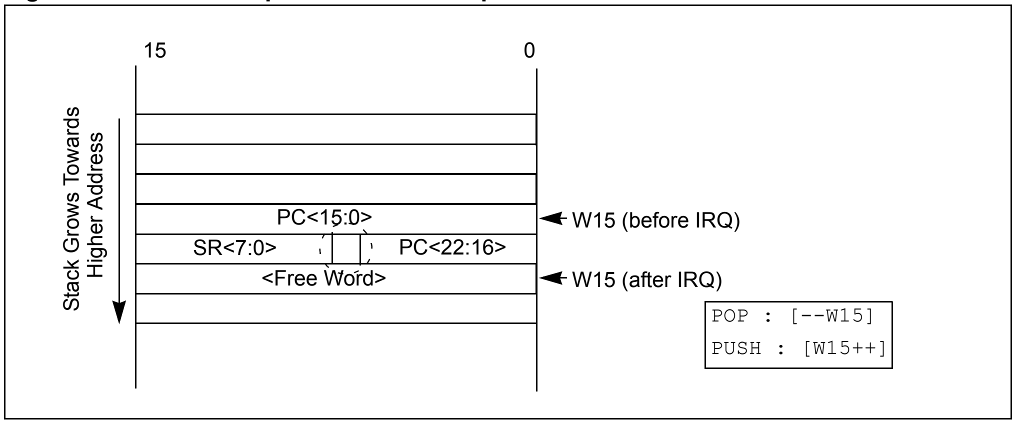
If there is a pending IRQ with a user-assigned priority level greater than the current processor priority level, indicated by the IPL<2:0> status bits (SR<7:5>), an interrupt is presented to the processor. The processor then saves the following information on the software stack:

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

These three values allow the return PC 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` (Return from Interrupt) instruction.

Figure 28-2: Stack Operation for Interrupt Event



28.2.4.1 Return from Interrupt

The `RETFIE` (Return from Interrupt) instruction clears the stack of the PC return address, the IPL3 status bit and the SRL register to return the processor to the state and priority level that existed before the interrupt sequence.

28.2.4.2 Interrupt Nesting

Interrupts, by default, can be nested. Any ISR in progress can be interrupted by another source of interrupt with a higher user-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-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.

28.2.5 Wake-Up From Sleep and Idle

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

1. 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.
2. If the user-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 CPU priority level for the interrupt must be assigned to CPU priority level 1 or greater.

28.2.6 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.

28.2.7 External Interrupt Support

The dsPIC DSC SMPS device supports up to three external interrupt pin sources (INT0-INT2). Each external interrupt pin has edge detection circuitry to detect the interrupt event. The INTCON2 register has three 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.

28.3 Interrupt Processing Timing

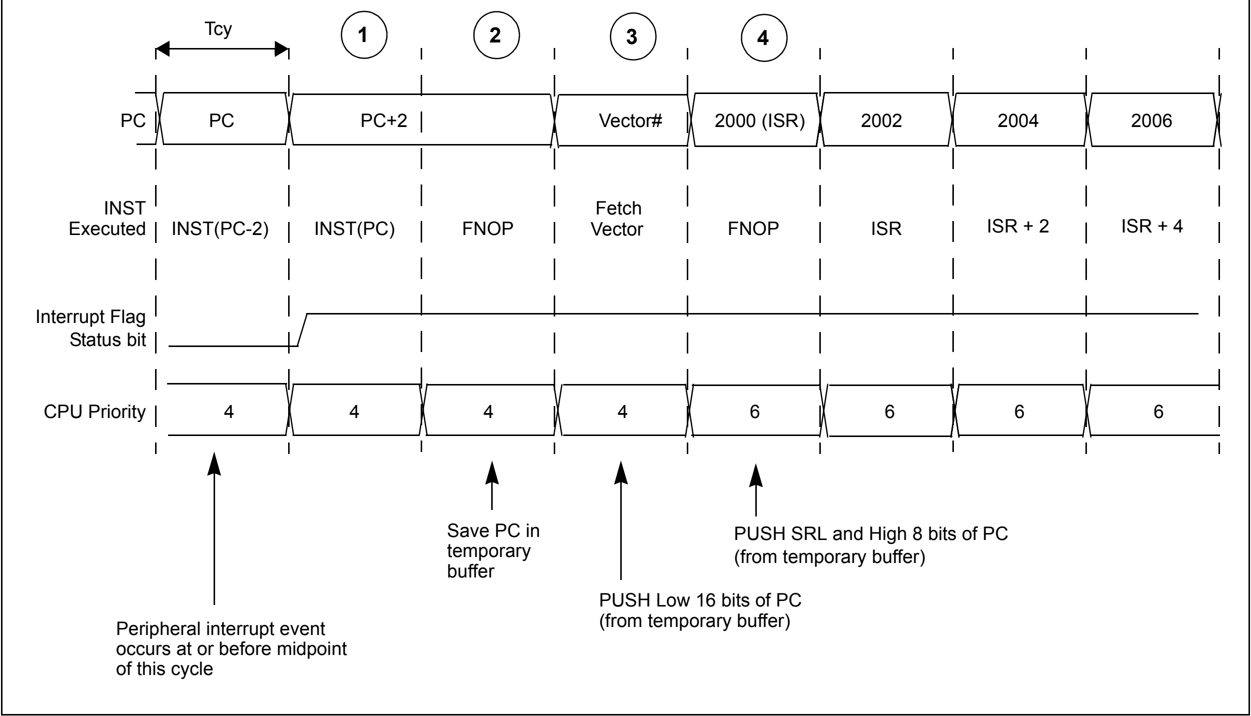
The following sections describe interrupt processing timing.

28.3.1 Interrupt Latency for One-Cycle Instructions

Figure 28-3 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 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 28.3.2 "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 28-3: Interrupt Timing During a One-Cycle Instruction


Tcy
1
2
3
4
PC
PC
PC+2
Vector#
2000 (ISR)
2002
2004
2006
INST Executed
INST(PC-2)
INST(PC)
FNOP
Fetch Vector
FNOP
ISR
ISR + 2
ISR + 4
Interrupt Flag Status bit
CPU Priority
4
4
4
4
6
6
6
6
Save PC in temporary buffer
PUSH SRL and High 8 bits of PC (from temporary buffer)
PUSH Low 16 bits of PC (from temporary buffer)
Peripheral interrupt event occurs at or before midpoint of this cycle

28.3.2 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 cycles of the interrupt process allow the two-cycle instruction to complete execution. The timing diagram in Figure 28-4 illustrates the peripheral interrupt event occurring in the instruction cycle prior to execution of the two-cycle instruction.

Figure 28-5 illustrates 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 28.3.1 "Interrupt Latency for One-Cycle Instructions"**).

Figure 28-4: Interrupt Timing During a Two-Cycle Instruction

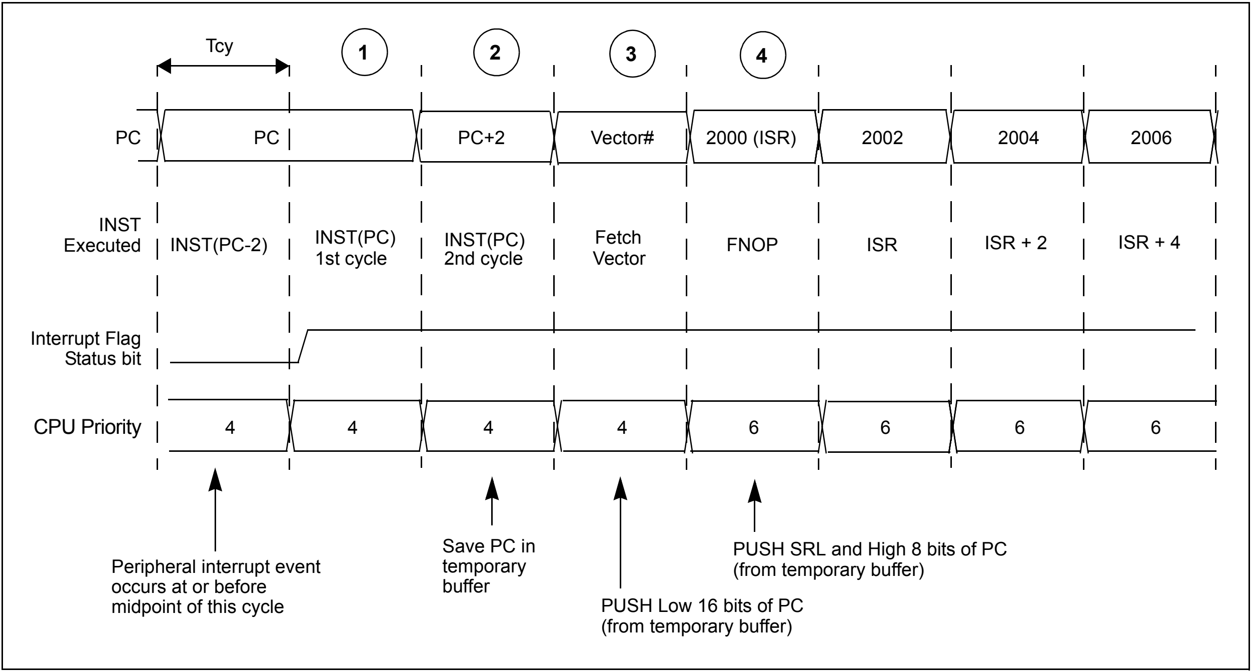
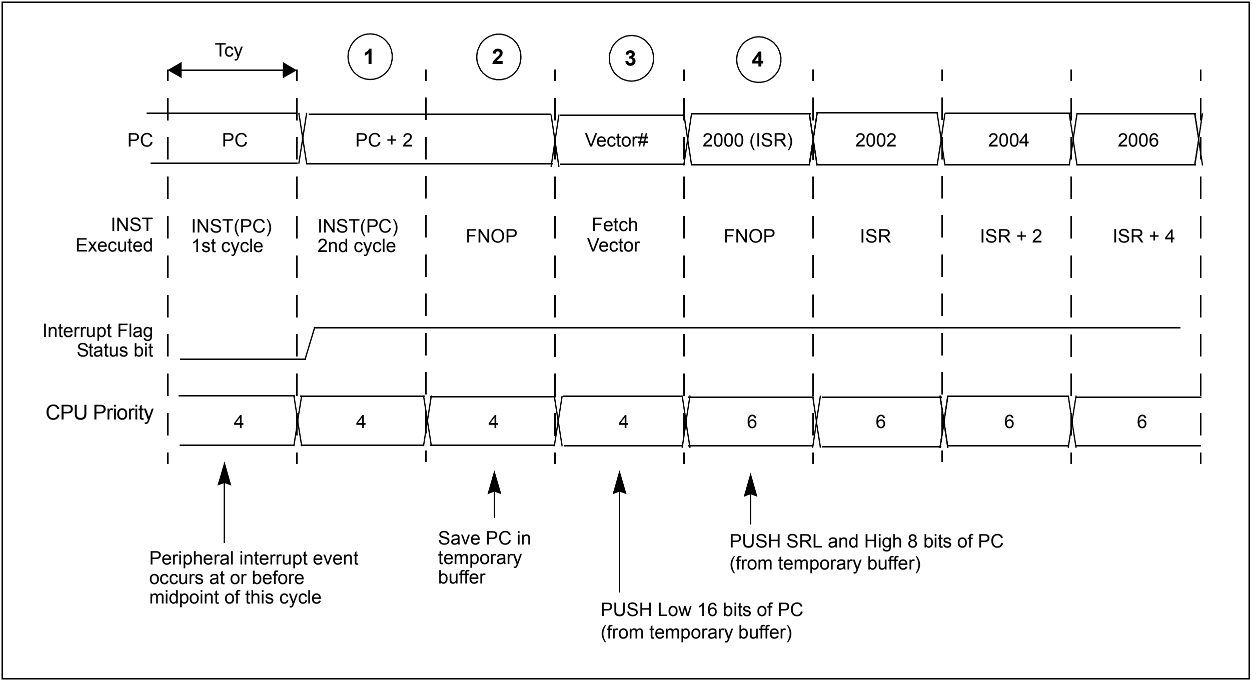

Tcy
1
2
3
4
PC
PC
PC+2
Vector#
2000 (ISR)
2002
2004
2006
INST Executed
INST(PC-2)
INST(PC) 1st cycle
INST(PC) 2nd cycle
Fetch Vector
FNOP
ISR
ISR + 2
ISR + 4
Interrupt Flag Status bit
CPU Priority
4
4
4
4
6
6
6
6
Peripheral interrupt event occurs at or before midpoint of this cycle
Save PC in temporary buffer
PUSH Low 16 bits of PC (from temporary buffer)
PUSH SRL and High 8 bits of PC (from temporary buffer)

Figure 28-5: Interrupt Timing, Interrupt Occurs During 1st Cycle of a Two-Cycle Instruction

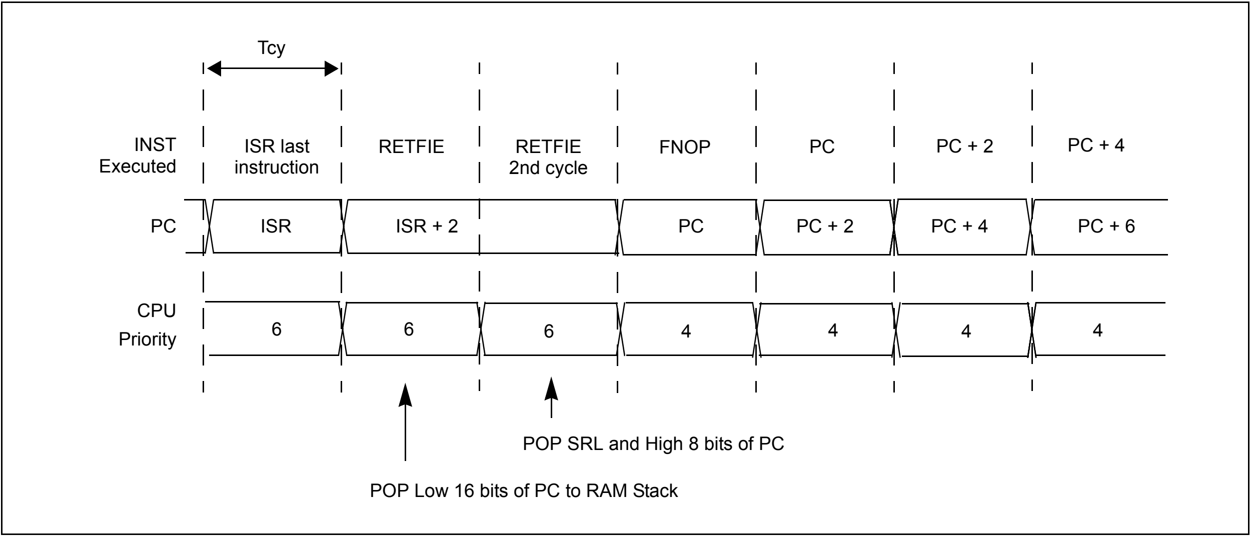

Tcy
1
2
3
4
PC
PC
PC + 2
Vector#
2000 (ISR)
2002
2004
2006
INST Executed
INST(PC) 1st cycle
INST(PC) 2nd cycle
FNOP
Fetch Vector
FNOP
ISR
ISR + 2
ISR + 4
Interrupt Flag Status bit
CPU Priority
4
4
4
4
6
6
6
6
Peripheral interrupt event occurs at or before midpoint of this cycle
Save PC in temporary buffer
PUSH Low 16 bits of PC (from temporary buffer)
PUSH SRL and High 8 bits of PC (from temporary buffer)

28.3.3 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 28-6: Return from Interrupt Timing



28.3.4 Special Conditions for Interrupt Latency

The dsPIC DSC SMPS device 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 the following 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

28.4 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.
- **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 signals and is cleared using 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 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.
- **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 will depend on the device variant. Refer to the specific device data sheet for further details.

28.4.1 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 28-2. For example, the INT0 (External Interrupt 0) is shown as having vector number and a natural order priority of ‘0’. Thus, the INT0IF Status bit is found in IFS0<0>. The INT0 interrupt uses bit 0 of the IEC0 register as its Enable bit and the IPC0<2:0> bits assign the interrupt priority level for the INT0 interrupt.

Register 28-1: SR: Status Register (in CPU)

Upper Byte:

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
OA	OB	SA	SB	OAB	SAB	DA	DC
bit 15							bit 8

Lower Byte:

R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
IPL<2:0>			RA	N	OV	Z	C
bit 7							bit 0

bit 15-8 Not used by the Interrupt Controller
(See the "*dsPIC30F/33F Programmer's Reference Manual*" (DS70157) for descriptions of the SR bits.)

bit 7-5 **IPL<2:0>:** CPU Interrupt Priority Level Status bits
`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>).

bit 4-0 Not used by the Interrupt Controller
(See the *"dsPIC30F/33F Programmer's Reference Manual*" (DS70157) for descriptions of the SR bits.)

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

Register 28-2: CORCON: Core Control Register

Upper Byte:

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

Lower Byte:

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

bit 15-4 Not used by the Interrupt Controller
(See the "*dsPIC30F/33F Programmer's Reference Manual*" (DS70157) for descriptions of the CORCON register 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 1: 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 the "*dsPIC30F/33F Programmer's Reference Manual*" (DS70157) for descriptions of the CORCON register bits.)

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

Register 28-3: INTCON1: Interrupt Control Register 1

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
NSTDIS	OVAERR	OVBERR	COVAERR	COVBERR	OVATE	OVBTE	COVTE
bit 15							bit 8

Lower Byte:							
R/W-0-0	R/W-0-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
SFTACERR	DIV0ERR	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—
bit 7							bit 0

bit 15	**NSTDIS:** Interrupt Nesting Disable bit 1 = Interrupt nesting is disabled 0 = Interrupt nesting is enabled
bit 14	**OVAERR:** Accumulator A Overflow Trap Flag bit 1 = Trap was caused by overflow of Accumulator A 0 = Trap was not caused by overflow of Accumulator A
bit 13	**OVBERR:** Accumulator B Overflow Trap Flag bit 1 = Trap was caused by overflow of Accumulator B 0 = Trap was not caused by overflow of Accumulator B
bit 12	**COVAERR:** Accumulator A Catastrophic Overflow Trap Flag bit 1 = Trap was caused by catastrophic overflow of Accumulator A 0 = Trap was not caused by catastrophic overflow of Accumulator A
bit 11	**COVBERR:** Accumulator B Catastrophic Overflow Trap Flag bit 1 = Trap was caused by catastrophic overflow of Accumulator B 0 = Trap was not caused by catastrophic overflow of Accumulator B
bit 10	**OVATE:** Accumulator A Overflow Trap Enable bit 1 = Trap overflow of Accumulator A 0 = Trap disabled
bit 9	**OVBTE:** Accumulator B Overflow Trap Enable bit 1 = Trap overflow of Accumulator B 0 = Trap disabled
bit 8	**COVTE:** Catastrophic Overflow Trap Enable bit 1 = Trap on catastrophic overflow of Accumulator A or B enabled 0 = Trap disabled
bit 7	**SFTACERR:** Shift Accumulator Error Status bit 1 = Math error trap was caused by an invalid accumulator shift 0 = Math error trap was not caused by an invalid accumulator shift
bit 6	**DIV0ERR:** Math Error Status bit 1 = Math error trap was caused by an invalid accumulator shift 0 = Math error trap was not caused by an invalid accumulator shift
bit 5	**Unimplemented:** Read as '0'
bit 4	**MATHERR:** Arithmetic Error 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

Register 28-3: INTCON1: Interrupt Control Register 1 (Continued)

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`'

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

Register 28-4: INTCON2: Interrupt Control Register 2

Upper Byte:

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

Lower Byte:

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

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

bit 14 **DISI:** Disable Interrupts (`DISI`) Instruction Status bit
1 = `DISI` instruction is active
0 = `DISI` 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

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

Register 28-5: IFS0: Interrupt Flag Status Register 0

Upper Byte:

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF
bit 15							bit 8

Lower Byte:

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
T3IF	T2IF	OC2IF	—	T1IF	OC1IF	IC1IF	INT0IF
bit 7							bit 0

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

bit 14 **MI2CIF:** I^{2}C Master Events Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 13 **SI2CIF:** I^{2}C Slave Events Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 12 **NVMIF:** Non-Volatile Memory Write Complete Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 11 **ADIF:** ADC Conversion Complete Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 10 **U1TXIF:** UART1 Transmitter Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 9 **U1RXIF:** UART1 Receiver Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 8 **SPI1IF:** SPI1 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 7 **T3IF:** Timer3 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 6 **T2IF:** Timer2 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 5 **OC2IF:** Output Compare 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 28-5: 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

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

Register 28-6: IFS1: Interrupt Flag Status Register 1

Upper Byte:							
R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0	U0	U-0
AC3IF	AC2IF	AC1IF	—	CNIF	—	—	—
bit 15							bit 8

Lower Byte:							
U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	PWM4IF	PWM3IF	PWM2IF	PWM1IF	PSEMIF	INT2IF	INT1IF
bit 7							bit 0

bit 15 **AC3IF:** Analog Comparator #3 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 14 **AC2IF:** Analog Comparator #2 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 13 **AC1IF:** Analog Comparator #1 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 12 **Unimplemented:** Read as ‘0’

bit 11 **CNIF:** Input Change Notification Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 10-7 **Unimplemented:** Read as ‘0’

bit 6 **PWM4IF:** Pulse-Width Modulation Generator #4 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 5 **PWM3IF:** Pulse-Width Modulation Generator #3 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 4 **PWM2IF:** Pulse-Width Modulation Generator #2 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 3 **PWM1IF:** Pulse-Width Modulation Generator #1 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 2 **PSEMIF:** PWM Special Event Match Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

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

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

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

Register 28-7: IFS2: Interrupt Flag Status Register 2

Upper Byte:							
U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	ADCP5IF	ADCP4IF	ADCP3IF
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	R/W-0
ADCP2IF	ADCP1IF	ADCP0IF	—	—	—	—	AC4IF
bit 7							bit 0

bit 15-11	**Unimplemented:** Read as '0'
bit 10	**ADCP5IF:** ADC Pair 5 Conversion Done Interrupt Flag Status bit 1 = Interrupt request has occurred 0 = Interrupt request has not occurred
bit 9	**ADCP4IF:** ADC Pair 4 Conversion Done Interrupt Flag Status bit 1 = Interrupt request has occurred 0 = Interrupt request has not occurred
bit 8	**ADCP3IF:** ADC Pair 3 Conversion Done Interrupt Flag Status bit 1 = Interrupt request has occurred 0 = Interrupt request has not occurred
bit 7	**ADCP2IF:** ADC Pair 2 Conversion Done Interrupt Flag Status bit 1 = Interrupt request has occurred 0 = Interrupt request has not occurred
bit 6	**ADCP1IF:** ADC Pair 1 Conversion Done Interrupt Flag Status bit 1 = Interrupt request has occurred 0 = Interrupt request has not occurred
bit 5	**ADCP0IF:** ADC Pair 0 Conversion Done Interrupt Flag Status bit 1 = Interrupt request has occurred 0 = Interrupt request has not occurred
bit 4-1	**Unimplemented:** Read as '0'
bit 0	**AC4IF:** Analog Comparator #4 Interrupt Flag Status bit 1 = Interrupt request has occurred 0 = Interrupt request has not occurred

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

Register 28-8: IEC0: Interrupt Enable Control Register 0

Upper Byte:

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	MI2CIE	SI2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE
bit 15							bit 8

Lower Byte:

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
T3IE	T2IE	OC2IE	—	T1IE	OC1IE	IC1IE	INT0IE
bit 7							bit 0

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

bit 14 **MI2CIE:** I^{2}C Master Event Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 13 **SI2CIE:** I^{2}C Slave Event Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 12 **NVMIE:** Non-Volatile Memory Write Complete Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 11 **ADIE:** ADC Conversion Complete Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 10 **U1TXIE:** UART1 Transmitter Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 9 **U1RXIE:** UART1 Receiver Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 8 **SPI1IE:** SPI1 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 7 **T3IE:** Timer3 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 6 **T2IE:** Timer2 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 5 **OC2IE:** Output Compare 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 28-8: 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

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

Register 28-9: IEC1: Interrupt Enable Control Register 1

Upper Byte:

R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0	U-0	U-0
AC3IE	AC2IE	AC1IE	—	CNIE	—	—	—
bit 15							bit 8

Lower Byte:

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	PWM4IE	PWM3IE	PWM2IE	PWM1IE	PSEMIE	INT2IE	INT1IE
bit 7							bit 0

bit 15 **AC3IE:** Analog Comparator #3 Interrupt Enable bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 14 **AC2IE:** Analog Comparator #2 Interrupt Enable bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 13 **AC1IE:** Analog Comparator #1 Interrupt Enable bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

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

bit 11 **CNIE:** Input Change Notification Enable bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 10-7 **Unimplemented:** Read as '0'

bit 6 **PWM4IE:** Pulse-Width Modulation Generator #4 Interrupt 2 Enable bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 5 **PWM3IE:** Pulse-Width Modulation Generator #3 Interrupt 2 Enable bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 4 **PWM2IE:** Pulse-Width Modulation Generator #2 Interrupt 2 Enable bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 3 **PWM1IE:** Pulse-Width Modulation Generator #1 Interrupt 2 Enable bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 2 **PSEMIE:** PWM Special Event Match Interrupt Enable bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 1 **INT2IE:** External Interrupt 2 Enable bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 0 **INT1IE:** External Interrupt 1 Enable bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

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

Register 28-10: IEC2: Interrupt Enable Control Register 2

Upper Byte:							
U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	ADCP5IE	ADCP4IE	ADCP3IE
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	R/W-0
ADCP2IE	ADCP1IE	ADCP0IE	—	—	—	—	AC4IE
bit 7							bit 0

bit 15-11 **Unimplemented:** Read as '0'

bit 10 **ADCP5IE:** ADC Pair 5 Conversion Done Interrupt Enable bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 9 **ADCP4IE:** ADC Pair 4 Conversion Done Interrupt Enable bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 8 **ADCP3IE:** ADC Pair 3 Conversion Done Interrupt Enable bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 7 **ADCP2IE:** ADC Pair 2 Conversion Done Interrupt Enable bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 6 **ADCP1IE:** ADC Pair 1 Conversion Done Interrupt Enable bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 5 **ADCP0IE:** ADC Pair 0 Conversion Done Interrupt Enable bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 4-1 **Unimplemented:** Read as '0'

bit 0 **AC4IE:** Analog Comparator #4 Interrupt Enable bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

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

Register 28-11: IPC0: Interrupt Priority Control Register 0

Upper Byte:							
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

Lower Byte:							
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

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

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

Register 28-12: IPC1: Interrupt Priority Control Register 1

Upper Byte:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	T3IP<2:0>			—	T2IP<2:0>		
bit 15							bit 8

Lower Byte:							
U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	OC2IP<2:0>			—	—	—	—
bit 7							bit 0

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

bit 14-12 **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

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

bit 10-8 **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 7 **Unimplemented:** Read as '0'

bit 6-4 **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 3-0 **Unimplemented:** Read as '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

Register 28-13: IPC2: Interrupt Priority Control Register 2

Upper Byte:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	ADIP<2:0>			—	U1TXIP<2:0>		
bit 15							bit 8

Lower Byte:							
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 7							bit 0

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

bit 14-12 **ADIP<2:0>:** ADC Conversion Complete 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 **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

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

bit 6-4 **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 3 **Unimplemented:** Read as '0'

bit 2-0 **SPI1IP<2:0>:** SPI1 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

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

Register 28-14: IPC3: Interrupt Priority Control Register 3

Upper Byte:							
U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-0	R/W-0
—	—	—	—	—	MI2CIP<2:0>		
bit 15							bit 8

Lower Byte:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	SI2CIP<2:0>			—	NVMIP<2:0>		
bit 7							bit 0

bit 15-11 **Unimplemented:** Read as ‘`0`’

bit 10-8 **MI2CIP<2:0>:** I^{2}C Master Events 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 **SI2CIP<2:0>:** I^{2}C Slave 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 **NVMIP<2:0>:** Non-Volatile Memory Write Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
•
•
•
`001` = Interrupt is priority 1
`000` = Interrupt source is disabled

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

Register 28-15: IPC4: Interrupt Priority Control Register 4

Upper Byte:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	PWM1IP<2:0>			—	PSEMIP<2:0>		
bit 15							bit 8

Lower Byte:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	INT2IP<2:0>			—	INT1IP<2:0>		
bit 7							bit 0

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

bit 14-12 **PWM1IP<2:0>:** PWM Generator #1 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 **PSEMIP<2:0>:** PWM Special Event Match 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 **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 **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

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

Register 28-16: IPC5: Interrupt Priority Control Register 5

Upper Byte:							
U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-0	R/W-0
—	—	—	—	—	PWM4IP<2:0>		
bit 15							bit 8

Lower Byte:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	PWM3IP<2:0>			—	PWM2IP<2:0>		
bit 7							bit 0

bit 15-11 **Unimplemented:** Read as '0'

bit 10-8 **PWM4IP<2:0>:** PWM Generator #4 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 **PWM3IP<2:0>:** PWM Generator #3 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 **PWM2IP<2:0>:** PWM Generator #2 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
•
•
•
`001` = Interrupt is priority 1
`000` = Interrupt source is disabled

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

Register 28-17: IPC6: Interrupt Priority Control Register 6

Upper Byte:

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

Lower Byte:

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

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 11-0 **Unimplemented:** Read as ‘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

Register 28-18: IPC7: Interrupt Priority Control Register 7

Upper Byte:

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	AC3IP<2:0>			—	AC2IP<2:0>		
bit 15							bit 8

Lower Byte:

U-0	R/W-1	R/W-0	R/W-0	U-0	U-1	U-0	U-0
—	AC1IP<2:0>			—	—	—	—
bit 7							bit 0

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

bit 14-12 **AC3IP<2:0>:** Analog Comparator 3 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 **AC2IP<2:0>:** Analog Comparator 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 **AC1IP<2:0>:** Analog Comparator 1 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'

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

28

Interrupts

Register 28-19: IPC8: Interrupt Priority Control Register 8

Upper Byte:

U-0	U-0	U-0	U-0	U-0	U0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

Lower Byte:

U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-0	R/W-0
—	—	—	—	—	AC4IP<2:0>		
bit 7							bit 0

bit 15-3 **Unimplemented:** Read as ‘0’

bit 2-0 **AC4IP<2:0>:** Analog Comparator 4 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

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

Register 28-20: IPC9: Interrupt Priority Control Register 9

Upper Byte:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	ADCP2IP<2:0>			—	ADCP1IP<2:0>		
bit 15							bit 8

Lower Byte:							
U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	ADCP0IP<2:0>			—	—	—	—
bit 7							bit 0

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

bit 14-12 **ADCP2IP<2:0>:** ADC Pair 2 Conversion Done 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 **ADCP1IP<2:0>:** ADC Pair 1 Conversion Done 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 **ADCP0IP<2:0>:** ADC Pair 0 Conversion Done 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'

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

Register 28-21: IPC10: Interrupt Priority Control Register 10

Upper Byte:							
U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-0	R/W-0
—	—	—	—	—	ADCP5IP<2:0>		
bit 15							bit 8

Lower Byte:							
U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	ADCP4IP<2:0>			—	ADCP3IP<2:0>		
bit 7							bit 0

bit 15-11 **Unimplemented:** Read as ‘0’

bit 10-8 **ADCP5IP<2:0>:** ADC Pair 5 Conversion Done 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 **ADCP4IP<2:0>:** ADC Pair 4 Conversion Done 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 **ADCP3IP<2:0>:** ADC Pair 3 Conversion Done Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

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

Register 28-22: INTTREG: Interrupt Control and Status Register

Upper Byte:

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

Lower Byte:

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

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

bit 11-8 **ILR<3:0>:** 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<6:0>:** 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

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

28.5 Interrupt Setup Procedures

The following sections describe the interrupt setup procedures.

28.5.1 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 the user assigned priority level for the interrupt source by writing the control bits in the appropriate IPCx Control register. The priority level will depend on the specific application and type of interrupt source. If multiple priority levels are not desired, the IPCx register control bits for all enabled interrupt sources may be programmed to the same non-zero value.

> **Note:** At a device Reset, the IPC 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 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.

28.5.2 Interrupt Service Routine

The method that is used to declare an ISR and initialize the IVT with the correct vector address will depend on the programming language (i.e., C or assembler) and the language development tool suite that is used to develop the application. In general, the user must clear the interrupt flag in the appropriate IFSx 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.

28.5.3 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.

28.5.4 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.

28.5.5 Code Example

The following code in Example 28-1 enables nested interrupts and sets up Timer1, Timer2, Timer3 and Change Notice peripherals to priority levels 2, 5, 3 and 4 respectively. It also illustrates how interrupts can be enabled and disabled via the Status Register. Sample ISRs illustrate interrupt clearing.

Example 28-1: Nested Interrupts 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 3 (level 7 is highest) */
IPC1bits.T3IP = 3;
/* Set Timer2 interrupt priority to 5 */
IPC1bits.T2IP = 5;
/* Set Change Notice interrupt priority to 4 */
IPC6bits.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;
/* Enable Timer1 interrupt */
IEC0bits.T1IE = 1;
/* Enable Timer2 interrupt (PWM time base) */
IEC0bits.T2IE = 1;
/* Enable Timer3 interrupt (Replacement for Timer2)*/
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;
}
```

Example 28-1: Nested Interrupts Code Example (Continued)

```
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 28-3: Special Function Registers Associated with the Interrupt Controller

SFR Name	ADR	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	Reset State
INTCON1	0080	NSTDIS	OVAERR	OVBERR	COVAERR	COVBERR	OVATE	OVBTE	COVTE	SFTACERR	DIV0ERR	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—	0000 0000 0000 0000
INTCON2	0082	ALTIVT	DISI	—	—	—	—	—	—	—	—	—	—	—	INT2EP	INT1EP	INT0EP	0000 0000 0000 0000
IFS0	0084	—	MI2CIF	SI2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	—	T1IF	OC1IF	IC1IF	INT0IF	0000 0000 0000 0000
IFS1	0086	AC3IF	AC2IF	AC1IF	—	CNIF	—	—	—	—	PWM4IF	PWM3IF	PWM2IF	PWM1IF	PSEMIF	INT2IF	INT1IF	0000 0000 0000 0000
IFS2	0088	—	—	—	—	—	ADCP5IF	ADCP4IF	ADCP3IF	ADCP2IF	ADCP1IF	ADCP0IF	—	—	—	—	AC4IF	0000 0000 0000 0000
IEC0	0094	—	MI2CIE	SI2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	—	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IEC1	0096	AC3IE	AC2IE	AC1IE	—	CNIE	—	—	—	—	PWM4IE	PWM3IE	PWM2IE	PWM1IE	PSEMIE	INT2IE	INT1IE	0000 0000 0000 0000
IEC2	0098	—	—	—	—	—	ADCP5IE	ADCP4IE	ADCP3IE	ADCP2IE	ADCP1IE	ADCP0IE	—	—	—	—	AC4IE	0000 0000 0000 0000
IPC0	00A4	—	T1IP<2:0>			—	OC1IP<2:0>			—	IC1IP<2:0>			—	INT0IP<2:0>			0100 0100 0100 0100
IPC1	00A6	—	T31P<2:0>			—	T2IP<2:0>			—	OC2IP<2:0>			—	—	—	—	0100 0100 0100 0000
IPC2	00A8	—	ADIP<2:0>			—	U1TXIP<2:0>			—	U1RXIP<2:0>			—	SPI1IP<2:0>			0100 0100 0100 0100
IPC3	00AA	—	—	—	—	—	MI2CIP<2:0>			—	SI2CIP<2:0>			—	NVMIP<2:0>			0000 0000 0000 0100
IPC4	00AC	—	PWM1IP<2:0>			—	PSEMIP<2:0>			—	INT2IP<2:0>			—	INT1IP<2:0>			0100 0100 0100 0100
IPC5	00AE	—	—	—	—	—	PWM4IP<2:0>			—	PWM3IP<2:0>			—	PWM2IP<2:0>			0000 0000 0000 0100
IPC6	00B0	—	CNIP<2:0>			—	—	—	—	—	—	—	—	—	—	—	—	0100 0000 0000 0000
IPC7	00B2	—	AC3IP<2:0>			—	AC2IP<2:0>			—	AC1IP<2:0>			—	—	—	—	0100 0100 0100 0000
IPC8	00B4	—	—	—	—	—	—	—	—	—	—	—	—	—	AC4IP<2:0>			0000 0000 0000 0100
IPC9	00B6	—	ADCP2IP<2:0>			—	ADCP1IP<2:0>			—	ADCP0IP<2:0>			—	—	—	—	0100 0100 0100 0000
IPC10	00B8	—	—	—	—	—	ADCP5IP<2:0>			—	ADCP4IP<2:0>			—	ADCP3IP<2:0>			0000 0100 0100 0100
INTTREG	00E0	—	—	—	—	ILR<3:0>				—	VECNUM<6:0>							0000 0100 0100 0100

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.

28.6 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.

28.7 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 SMPS dsPIC DSC 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 include the following:

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 SMPS dsPIC DSC family of devices.

28.8 Revision History

Revision A (January 2007)

This is the initial released revision of this document.

Revision B (February 2008)

Corrected name of Interrupt Vector 35 to CN – Input Change Notice (see Table 28-2). Added note to disable interrupts while changing IPL on the fly (see Section 28.1.5).