

---

## Section 8. Interrupts

---

This section of the manual contains the following topics:

8.1	Introduction .....	8-2
8.2	Control Registers .....	8-3
8.3	Operation .....	8-13
8.4	Single Vector Mode .....	8-14
8.5	Multi-Vector Modes .....	8-15
8.6	Interrupt Vector Address Calculation .....	8-17
8.7	Interrupt Priorities .....	8-19
8.8	Interrupts and Register Sets .....	8-20
8.9	Interrupt Processing .....	8-21
8.10	External Interrupts .....	8-21
8.11	Temporal Proximity Interrupt Coalescing .....	8-22
8.12	Software Generated Non-Maskable Interrupt .....	8-23
8.13	Effects of Interrupts After Reset .....	8-24
8.14	Operation in Power-Saving and Debug Modes .....	8-24
8.15	Related Application Notes .....	8-25
8.16	Revision History .....	8-26

**Note:** This family reference manual section is meant to serve as a complement to device data sheets. Depending on the device variant, this manual section may not apply to all PIC32 devices.

Please consult the note at the beginning of the “**Interrupt Controller**” chapter in the current device data sheet to check whether this document supports the device you are using.

Device data sheets and family reference manual sections are available for download from the Microchip Worldwide Web site at: <http://www.microchip.com>

## 8.1 INTRODUCTION

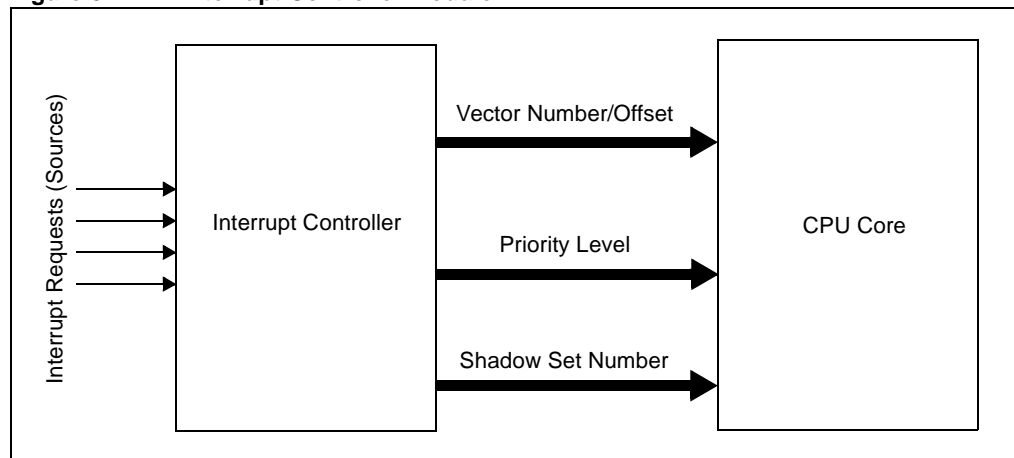
The PIC32 device generates interrupt requests in response to interrupt events from peripheral modules. The Interrupt Controller module exists external to the CPU logic and prioritizes the interrupt events before presenting them to the CPU.

The PIC32 Interrupts Controller module includes the following features:

- Up to 256 interrupt sources
- Single and Multi-Vector mode operations
- Up to five external interrupts with edge polarity control
- Interrupt proximity timer
- Seven user-selectable priority levels for each vector
- Four user-selectable subpriority levels within each priority
- User-configurable shadow set based on priority level (this feature is not available on all devices; refer to the “**Interrupt Controller**” chapter in the specific device data sheet for availability)
- Software can generate any interrupt
- User-configurable Interrupt Vector Table (IVT) location
- User-configurable interrupt vector spacing

Figure 8-1 shows the block diagram of the Interrupt Controller module.

**Figure 8-1: Interrupt Controller Module**



**Note:** Several of the registers cited in this section are not in the Interrupt Controller module. These registers (and bits) are associated with the CPU. Refer to **Section 2. “CPU”** (DS60001113) for more details.

To avoid confusion, a typographic distinction is made for registers in the CPU. The register names in this section, and all other sections of this manual, are signified by uppercase letters only (except for cases in which variables are used). CPU register names are signified by upper and lowercase letters. For example, INTSTAT is an Interrupts register; whereas, IntCtl is a CPU register.

## 8.2 CONTROL REGISTERS

**Note:** Each PIC32 device may have one or more interrupt sources, and depending on the device variant, the number of sources may be different. An 'x' used in the names of control/status bits and registers denotes that there are multiple registers that have the same function, which can define these interrupt sources. Refer to the specific device data sheet for more details.

The Interrupts Controller module consists of the following Special Function Registers (SFRs):

- **INTCON: Interrupt Control Register**

This register controls the interrupt vector spacing, Single Vector or Multi-Vector modes, Interrupt Proximity, and external Interrupt edge detection.

- **PRISS: Priority Shadow Select Register**

This register controls which CPU shadow register set is used by which interrupt priority, or in Single Vector mode, whether a shadow register set is used.

- **INTSTAT: Interrupt Status Register**

This read-only register provides the status of the interrupt priority level or vector that is presented to the CPU.

- **IPTMR: Interrupt Proximity Timer Register**

This register controls a timing window in which interrupts are held off from being presented to the CPU.

- **IFSx: Interrupt Flag Status Register**

These registers contain the flags that indicate the status of the various interrupt sources, and also clears those interrupts sources.

- **IECx: Interrupt Enable Control Register**

These registers contain the flags that enable or disable the interrupt sources from triggering an interrupt of the CPU.

- **IPCx: Interrupt Priority Control Register**

These registers control the priority and sub-priority levels of the various interrupt sources.

- **OFFx: Interrupt Vector Address Offset Register**

These registers contain the offset from EBASE that the CPU will jump to when the corresponding interrupt occurs.

[Table 8-1](#) summarizes all Interrupts-related registers. Corresponding registers appear after the summary, followed by a detailed description of each register.

Table 8-1: Interrupts Register Summary

Register Name	Bit Range	Bit 31/15	Bit 30/14	Bit 29/13	Bit 28/12	Bit 27/11	Bit 26/10	Bit 25/9	Bit 24/8	Bit 23/7	Bit 22/6	Bit 21/5	Bit 20/4	Bit 19/3	Bit 18/2	Bit 17/1	Bit 16/0	
INTCON <sup>(1)</sup>	31:16	NMIKEY<7:0> <sup>(2)</sup>										VS<6:1> <sup>(2)</sup>						VS<0> <sup>(2)</sup>
	15:0	—	—	—	MVEC	—	TPC<2:0>			—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP		
PRISS <sup>(1)</sup>	31:16	PRI7SS<3:0> <sup>(2)</sup>					PRI6SS<3:0> <sup>(2)</sup>				PRI5SS<3:0> <sup>(2)</sup>				PRI4SS<3:0> <sup>(2)</sup>			
	15:0	PRI3SS<3:0> <sup>(2)</sup>					PRI2SS<3:0> <sup>(2)</sup>				PRI1SS<3:0> <sup>(2)</sup>				—	—	—	SS0 <sup>(2)</sup>
INTSTAT <sup>(1)</sup>	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
	15:0	—	—	—	—	—	SRIPL<2:0>			—	—	VEC<5:0> <sup>(2)</sup>						
IPTMR <sup>(1)</sup>	31:16	IPTMR<31:16>																
	15:0	IPTMR<15:0>																
IFSx <sup>(1)</sup>	31:16	IFS31	IFS30	IFS29	IFS28	IFS27	IFS26	IFS25	IFS24	IFS23	IFS22	IFS21	IFS20	IFS19	IFS18	IFS17	IFS16	
	15:0	IFS15	IFS14	IFS13	IFS12	IFS11	IFS10	IFS09	IFS08	IFS07	IFS06	IFS05	IFS04	IFS03	IFS02	IFS01	IFS00	
IECx <sup>(1)</sup>	31:16	IEC31	IEC30	IEC29	IEC28	IEC27	IEC26	IEC25	IEC24	IEC23	IEC22	IEC21	IEC20	IEC19	IEC18	IEC17	IEC16	
	15:0	IEC15	IEC14	IEC13	IEC12	IEC11	IEC10	IEC09	IEC08	IEC07	IEC06	IEC05	IEC04	IEC03	IEC02	IEC01	IEC00	
IPCx <sup>(1)</sup>	31:16	—	—	—	IP03<2:0>				IS03<1:0>			—	—	—	IP02<2:0>			IS02<1:0>
	15:0	—	—	—	IP01<2:0>				IS01<1:0>			—	—	—	IP00<2:0>			IS00<1:0>
OFFx <sup>(1)</sup>	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	VOFFx<17:16>		
	15:0	VOFFx<15:1>																—

**Legend:** — = unimplemented, read as '0'.

- Note 1:** This register has an associated Clear, Set, and Invert register at an offset of 0x4, 0x8, and 0xC bytes, respectively. These registers have the same name with CLR, SET, or INV appended to the end of the register name (e.g., INTCONCLR). Writing a '1' to any bit position in these registers will clear valid bits in the associated register. Reads from these registers should be ignored.
- Note 2:** These bits are not available on all devices. Refer to the “**Interrupt Controller**” chapter in the specific device data sheet for availability.

## Section 8. Interrupts

**Register 8-1: INTCON: Interrupt Control Register**

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NMIKEY<7:0> <sup>(1)</sup>							
23:16	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	VS<6:1> <sup>(1)</sup>						VS<0> <sup>(1)</sup>
								SS0 <sup>(1)</sup>
15:8	U-0	U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
	—	—	—	MVEC	—	TPC<2:0>		
7:0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP

**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 31-24 **NMIKEY<7:0>**: Non-Maskable Interrupt Key bits<sup>(1)</sup>

Write a value of 0x4E to this field to trigger a software-generated non-maskable interrupt (NMI) event.

bit 23 **Unimplemented**: Read as '0'

bit 22-16 **VS<6:0>**: Vector Spacing bits<sup>(1)</sup>

Spacing Between Vectors		
Bit Value	Hexadecimal	Decimal
1000000	0x200	512
0100000	0x100	256
0010000	0x080	128
0001000	0x040	64
0000100	0x020	32
0000010	0x010	16
0000001	0x001	8
0000000	0x000	0

**Note:** All other values are Reserved.

bit 16 **SS0**: Single Vector Shadow Register Set bit<sup>(1)</sup>

1 = Single vector is presented with a shadow register set

0 = Single vector is not presented with a shadow register set

bit 15-13 **Unimplemented**: Read as '0'

bit 12 **MVEC**: Multi Vector Configuration bit

1 = Interrupt Controller configured for multi vectored mode

0 = Interrupt Controller configured for single vectored mode

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

bit 10-8 **TPC<2:0>**: Interrupt Proximity Timer Control bits

111 = Interrupts of group priority 7 or lower start the Interrupt Proximity timer

110 = Interrupts of group priority 6 or lower start the Interrupt Proximity timer

101 = Interrupts of group priority 5 or lower start the Interrupt Proximity timer

100 = Interrupts of group priority 4 or lower start the Interrupt Proximity timer

011 = Interrupts of group priority 3 or lower start the Interrupt Proximity timer

010 = Interrupts of group priority 2 or lower start the Interrupt Proximity timer

001 = Interrupts of group priority 1 start the Interrupt Proximity timer

000 = Disables Interrupt Proximity timer

**Note 1:** This bit is not available on all devices. Refer to the “Interrupt Controller” chapter in the specific device data sheet for availability.

## Register 8-1: INTCON: Interrupt Control Register (Continued)

- bit 7-5    **Unimplemented:** Read as '0'
- bit 4    **INT4EP:** External Interrupt 4 Edge Polarity Control bit  
1 = Rising edge  
0 = Falling edge
- bit 3    **INT3EP:** External Interrupt 3 Edge Polarity Control bit  
1 = Rising edge  
0 = Falling edge
- bit 2    **INT2EP:** External Interrupt 2 Edge Polarity Control bit  
1 = Rising edge  
0 = Falling edge
- bit 1    **INT1EP:** External Interrupt 1 Edge Polarity Control bit  
1 = Rising edge  
0 = Falling edge
- bit 0    **INT0EP:** External Interrupt 0 Edge Polarity Control bit  
1 = Rising edge  
0 = Falling edge

**Note 1:** This bit is not available on all devices. Refer to the “**Interrupt Controller**” chapter in the specific device data sheet for availability.

**Register 8-2: PRISS: Priority Shadow Select Register**

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PRI7SS<3:0>				PRI6SS<3:0>			
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PRI5SS<3:0>				PRI4SS<3:0>			
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PRI3SS<3:0>				PRI2SS<3:0>			
7:0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	R/W-0
	PRI1SS<3:0>				—	—	—	SS0

**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 31-4 **PRIxSS<3:0>**: Priority Shadow Select bits

If the MVEC bit in the INTCON register = 1:

1111 = Shadow register set 15 is used for priority level x

1110 = Shadow register set 14 is used for priority level x

•  
•  
•

0001 = Shadow register set 1 is used for priority level x

0000 = Shadow register set 0 is used for priority level x

bit 3-1 **Unimplemented**: Read as '0'

bit 0 **SS0**: Single Vector Shadow Register Set bit

1 = Single vector is presented with a shadow register set

0 = Single vector is not presented with a shadow register set

**Note:** This register is not available on all devices and is cleared on all forms of Reset. Refer to the “**Interrupt Controller**” chapter in the specific device data sheet for availability.

# PIC32 Family Reference Manual

**Register 8-3: INTSTAT: Interrupt Status Register**

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —
23:16	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —
15:8	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	R-0	R-0	R-0
7:0	U-0 —	U-0 —	R-0	R-0	R-0	R-0	R-0	R-0
	SIRQ<7:0> <sup>(2)</sup>							

**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 31-11 **Unimplemented:** Read as '0'

bit 10-8 **SRIPL<2:0>:** Requested Priority Level bits for Single Vector Mode bits<sup>(1)</sup>  
 000–111 = The priority level of the latest interrupt presented to the CPU

bit 7-6 **Unimplemented:** Read as '0'

bit 7-0 **SIRQ<7:0>:** Last Interrupt Request Serviced Status bits<sup>(2)</sup>  
 These bits are used to determine the last interrupt request number serviced by the CPU.

bit 5-0 **VEC<5:0>:** Interrupt Vector bits<sup>(1,2)</sup>  
 11111–00000 = The interrupt vector that is presented to the CPU

- Note 1:** This value should only be used when the Interrupt Controller is configured for Single Vector mode.  
**Note 2:** These bits are not available on all devices. Refer to the “**Interrupts Controller**” chapter in the specific device data sheet for availability.

**Register 8-4: IPTMR: Interrupt Proximity Timer Register**

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IPTMR<31:24>								
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IPTMR<23:16>								
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IPTMR<15:8>								
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IPTMR<7:0>								

**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 31-0 **IPTMR<31:0>:** Interrupt Proximity Timer Reload bits  
 Used by the Interrupt Proximity Timer as a reload value when the Interrupt Proximity Timer is triggered by an interrupt event.



## Section 8. Interrupts

**Register 8-5: IFSx: Interrupt Flag Status Register**

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	IFS31	IFS30	IFS29	IFS28	IFS27	IFS26	IFS25	IFS24
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	IFS23	IFS22	IFS21	IFS20	IFS19	IFS18	IFS17	IFS16
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	IFS15	IFS14	IFS13	IFS12	IFS11	IFS10	IFS09	IFS08
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	IFS07	IFS06	IFS05	IFS04	IFS03	IFS02	IFS01	IFS00

**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 31-0 **IFS31-IFS00:** Interrupt Flag Status bits

1 = Interrupt request has occurred  
 0 = No interrupt request has occurred

**Note:** This register represents a generic definition of the IFSx register. Refer to the “Interrupt Controller” chapter in the specific device data sheet to learn exact bit definitions.

**Register 8-6: IECx: Interrupt Enable Control Register**

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	IEC31	IEC30	IEC29	IEC28	IEC27	IEC26	IEC25	IEC24
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	IEC23	IEC22	IEC21	IEC20	IEC19	IEC18	IEC17	IEC16
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	IEC15	IEC14	IEC13	IEC12	IEC11	IEC10	IEC09	IEC08
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	IEC07	IEC06	IEC05	IEC04	IEC03	IEC02	IEC01	IEC00

**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 31-0 **IEC31-IEC00:** Interrupt Enable Control bits

1 = Interrupt is enabled  
 0 = Interrupt is disabled

**Note:** This register represents a generic definition of the IFSx register. Refer to the “Interrupt Controller” chapter in the specific device data sheet to learn exact bit definitions.

# PIC32 Family Reference Manual

**Register 8-7: IPCx: Interrupt Priority Control Register**

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	IP03<2:0>			IS03<1:0>	
23:16	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	IP02<2:0>			IS02<1:0>	
15:8	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	IP01<2:0>			IS01<1:0>	
7:0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	IP00<2:0>			IS00<1:0>	

**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 31-29 **Unimplemented:** Read as '0'

bit 28-26 **IP03<2:0>**: Interrupt Priority bits

111 = Interrupt priority is 7  
110 = Interrupt priority is 6  
101 = Interrupt priority is 5  
100 = Interrupt priority is 4  
011 = Interrupt priority is 3  
010 = Interrupt priority is 2  
001 = Interrupt priority is 1  
000 = Interrupt is disabled

bit 25-24 **IS03<1:0>**: Interrupt Subpriority bits

11 = Interrupt subpriority is 3  
10 = Interrupt subpriority is 2  
01 = Interrupt subpriority is 1  
00 = Interrupt subpriority is 0

bit 23-21 **Unimplemented:** Read as '0'

bit 20-18 **IP02<2:0>**: Interrupt Priority bits

111 = Interrupt priority is 7  
110 = Interrupt priority is 6  
101 = Interrupt priority is 5  
100 = Interrupt priority is 4  
011 = Interrupt priority is 3  
010 = Interrupt priority is 2  
001 = Interrupt priority is 1  
000 = Interrupt is disabled

bit 17-16 **IS02<1:0>**: Interrupt Subpriority bits

11 = Interrupt subpriority is 3  
10 = Interrupt subpriority is 2  
01 = Interrupt subpriority is 1  
00 = Interrupt subpriority is 0

bit 15-13 **Unimplemented:** Read as '0'

**Note:** This register represents a generic definition of the IPCx register. Refer to the “Interrupt Controller” chapter in the specific device data sheet to learn exact bit definitions.

### Register 8-7: IPCx: Interrupt Priority Control Register (Continued)

bit 12-10 **IP01<2:0>**: Interrupt Priority bits

111 = Interrupt priority is 7  
110 = Interrupt priority is 6  
101 = Interrupt priority is 5  
100 = Interrupt priority is 4  
011 = Interrupt priority is 3  
010 = Interrupt priority is 2  
001 = Interrupt priority is 1  
000 = Interrupt is disabled

bit 9-8 **IS01<1:0>**: Interrupt Subpriority bits

11 = Interrupt subpriority is 3  
10 = Interrupt subpriority is 2  
01 = Interrupt subpriority is 1  
00 = Interrupt subpriority is 0

bit 7-5 **Unimplemented**: Read as '0'

bit 4-2 **IP00<2:0>**: Interrupt Priority bits

111 = Interrupt priority is 7  
110 = Interrupt priority is 6  
101 = Interrupt priority is 5  
100 = Interrupt priority is 4  
011 = Interrupt priority is 3  
010 = Interrupt priority is 2  
001 = Interrupt priority is 1  
000 = Interrupt is disabled

bit 1-0 **IS00<1:0>**: Interrupt Subpriority bits

11 = Interrupt subpriority is 3  
10 = Interrupt subpriority is 2  
01 = Interrupt subpriority is 1  
00 = Interrupt subpriority is 0

**Note:** This register represents a generic definition of the IPCx register. Refer to the “Interrupt Controller” chapter in the specific device data sheet to learn exact bit definitions.

# PIC32 Family Reference Manual

**Register 8-8: OFFx: Interrupt Vector Address Offset Register**

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —
23:16	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	R/W-0	R/W-0
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	VOFFx<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
	VOFFx<7:1>							—

**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 31-18 **Unimplemented:** Read as '0'

bit 17-1 **VOFFx<17:1>:** Interrupt Vector Address Even Offset bits. Even byte aligned.

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

**Note:** This register is not available on all devices. Refer to the “Interrupt Controller” chapter in the specific device data sheet for availability.

## 8.3 OPERATION

The Interrupt Controller is responsible for preprocessing an Interrupt Request (IRQ) from a number of on-chip peripherals and presenting them in the appropriate order to the processor.

The Interrupt Controller is designed to receive up to 256 IRQs from the processor core, on-chip peripherals capable of generating interrupts, and five external inputs. All IRQs are sampled on the rising edge of the SYSCLK and latched in associated IFSx registers. A pending IRQ is indicated by the flag bit being equal to '1' in an IFSx register. The pending IRQ will not cause further processing if the corresponding IECx bit in the Interrupt Enable register is clear. The IECx bits act to mask the interrupt flag. If the interrupt is enabled, all IRQs are encoded into a vector number. Since there are more IRQs than available vector numbers, some IRQs share common vector numbers. Each vector number is assigned an interrupt-priority-level and a shadow-set number. The priority level is determined by the IPCx register setting of associated vector. In Multi-Vector mode, the user can select a priority level to receive a dedicated shadow register set. In Single Vector mode, all interrupts may receive a dedicated shadow set. The Interrupt Controller selects the highest priority IRQ among all the pending IRQs and presents the associated vector number, priority-level and shadow-set number to the processor core.

The processor core samples the presented vector information between the "E" and "M" stages of the pipeline. If the vector's priority level presented to the core is greater than the current priority indicated by the CPU Interrupt Priority bits, IPL<2:0> (Status<12:10>), the interrupt is serviced; otherwise, it will remain pending until the current priority is less than the priority of the interrupt. When servicing an interrupt, the processor core pushes the Program Counter into the Exception Program Counter (EPC) register in the CPU and sets the Exception Level (EXL) bit (Status<1>) in the CPU. The EXL bit disables further interrupts until the application explicitly re-enables them by clearing the EXL bit, and then it branches to the vector address calculated from the presented vector number.

The INTSTAT register contains the interrupt request location and SRIPL<2:0> bits (INTSTAT<10:8>) of the current pending interrupt. This may not be the same as the interrupt that caused the core to diverge from normal execution.

The processor returns to the previous state when the Exception Return (ERET) instruction is executed. The ERET instruction clears the EXL bit, restores the Program Counter, and reverts the current shadow set to the previous one.

The PIC32 Interrupt Controller can be configured to operate in one of following modes:

- Single Vector mode – all interrupt requests will be serviced at one vector address (mode out of reset)
- Multi-Vector mode – interrupt requests will be serviced at the calculated vector address

**Notes:** Reconfiguring the Interrupt Controller module from Vector to Multi-Vector mode (or conversely), during run-time, is strongly discouraged. Changing Interrupt Controller modes after initialization may result in an undefined behavior.

The PIC32 processor core supports several different interrupt processing modes. The Interrupt Controller is designed to work in External Interrupt Controller mode.

There are two types of interrupts found in PIC32 devices: persistent and non-persistent. Persistent interrupts will remain active and the associated interrupt flag set until the issue causing the interrupt is serviced. An example would be an interrupt declaring data in a UART receive buffer. Until this data is read, the interrupt flag will remain set even if the flag is cleared in software. ISRs for persistent interrupts should clear the interrupt flag after removing the condition that caused the interrupt to ensure that the interrupt flag actually clears.

In non-persistent interrupts, the interrupt is recorded once to the Interrupt Controller that presents it to the CPU. The CPU is only interrupted when a new interrupt has occurred.

For information on an interrupt type located in a device, refer to the Interrupt IRQ, Vector, and Bit Location table in the "Interrupt Controller" chapter of the specific device data sheet. In this table, there is a column that states whether or not an interrupt is persistent.

## 8.4 SINGLE VECTOR MODE

On any form of reset, the Interrupt Controller initializes to Single Vector mode. When the MVEC bit (INTCON<12>) is '0', the Interrupt Controller operates in Single Vector mode. In this mode, the CPU always vectors to the same address.

**Note:** Users familiar with the MIPS32® architecture must note that the core in PIC32 devices is still operating in External Interrupt Controller (EIC) mode. The PIC32 device achieves Single Vector mode by forcing all IRQs to use a vector number of 0x00. Because the core always operates in EIC mode, the single vector behavior through "Interrupt Compatibility mode" as defined by the MIPS32 architecture is not recommended.

To configure the CPU in Single Vector mode, the following CPU registers (Cause and Status) and the INTCON register must be configured as follows:

- EBase ≠ 00000
- IV bit (Cause<23>) = 1
- MVEC bit (INTCON<12>) = 0
- IE bit (Status<0>) = 1
- Bev = 1

### Example 8-1: Single Vector Mode Initialization

```
/*
  Set the CP0 registers for single-vector interrupt
  Place EBASE at 0x9D01F000

  This code example uses MPLAB C32 intrinsic functions to access CP0 registers.
  Check your compiler documentation to find equivalent functions or use inline assembly
*/
unsigned int temp_CP0;           // Temporary register for CP0 register storing

asm volatile("di");             // Disable all interrupts
asm volatile("ehb");            // Disable all interrupts

_CP0_SET_EBASE(0x9D01F000);      // Set an EBase value of 0x9D01F000

temp_CP0 = _CP0_GET_CAUSE();     // Get Cause
temp_CP0 |= 0x00800000;         // Set IV
_CP0_SET_CAUSE(temp_CP0);       // Update Cause

INTCONCLR = _INTCON_MVEC_MASK;  // Clear the MVEC bit
```

## 8.5 MULTI-VECTOR MODES

### 8.5.1 Computed Offset

If the OFFx registers are not present on the device, a computed offset Multi-Vector mode is used. When the MVEC bit (INTCON<12>) is '1', the Interrupt Controller operates in Multi-Vector mode. In this mode, the CPU vectors to the address for each vector number. Each vector is located at a specific offset, with respect to a base address specified by the Exception Base (EBase) register in the CPU. The individual vector address offset is determined by the vector space that is specified by the VS<6:0> bits (IntCtl<9:5> or INTCON<20:16>). The EBase and IntCtl registers are CPU registers. For more information on the CPU registers, refer to **Section 2. "CPU"** (DS60001113).

To configure the CPU in Computed Offset mode, the following CPU registers (IntCtl, Cause and Status) and the INTCON register must be configured as follows:

- EBase ≠ 00000
- VS<6:0> bits (IntCtl<9:5> or INTCON<20:16>) ≠ 00000

**Note:** If the VS<6:0> bits are present in the INTCON register, they must be used instead of the IntCtl CPU register.

- IV bit (Cause<23>) = 1
- MVEC bit (INTCON<12>) = 1
- IE bit (Status<0>) = 1
- Bev = 1

#### Example 8-2: Computed Offset Mode Initialization

```
/* Set the CP0 registers for multi-vector interrupt
   Place EBASE at 0x9D01F000

   This code example uses MPLAB C32 intrinsic functions to access CP0 registers.
   Check your compiler documentation to find equivalent functions or use inline assembly */

unsigned int temp_CP0;           // Temporary register for CP0 reg storing

asm volatile("di");             // Disable all interrupts
asm volatile("ehb");            // Disable all interrupts

_CP0_SET_EBASE(0x9D01F000);      // Set an EBase value of 0x9D01F000

temp_CP0 = _CP0_GET_CAUSE();     // Get Cause
temp_CP0 |= 0x00800000;          // Set IV
_CP0_SET_CAUSE(temp_CP0);       // Update Cause

INTCONCLR = _INTCON_MVEC_MASK;  // Clear the MVEC bit
```

## 8.5.2 Variable Offset

If the OFFx registers are present in the device, a variable offset is used for vector spacing. Variable Offset mode is similar to Computed Offset mode, but the interrupt vector spacing is now configurable. A unique interrupt vector offset can be set for each vector using its associated OFFx register. This mode also provides more controller flexibility and may free required memory space from Interrupt Service Routines (ISRs). This mode is not available on all devices. Refer to the “**Interrupt Controller**” chapter in the specific device data sheet for availability.

To configure the CPU in Variable Offset mode, the following CPU registers (Cause and Status) and the OFFx registers must be configured as follows:

- EBase  $\neq$  00000
- OFFx  $\neq$  0
- IV bit (Cause<23>) = 1
- MVEC bit (INTCON<12>) = 1
- IE bit (Status<0>) = 1
- Bev = 1

### Example 8-3: Variable Offset Mode Initialization

```
/* Set the CP0 registers for multi-vector interrupt
   Place EBASE at 0x9D01F000

This code example uses MPLAB C32 intrinsic functions to access CP0 registers.
Check your compiler documentation to find equivalent functions or use inline assembly */

unsigned int temp_CP0;           // Temporary register for CP0 register storing

asm volatile("di");             // Disable all interrupts
asm volatile("ehb");            // Disable all interrupts

_CP0_SET_EBASE(0x9D01F000);      // Set an EBase value of 0x9D01F000

OFF0 = 0xfe;                    // Set a vector offset of 254 Bytes for VOFF0

temp_CP0 = _CP0_GET_CAUSE();     // Get Cause
temp_CP0 |= 0x00800000;          // Set IV
_CP0_SET_CAUSE(temp_CP0);        // Update Cause

INTCONSET = _INTCON_MVEC_MASK;   // Set the MVEC bit
```



## 8.6 INTERRUPT VECTOR ADDRESS CALCULATION

The vector address for a particular interrupt depends on how the Interrupt Controller is configured. If the Interrupt Controller is configured for Single Vector mode (see 8.4 “Single Vector Mode”), all interrupt vectors use the same vector address. When it is configured for Multi-Vector mode (see 8.5 “Multi-Vector Modes”), each interrupt vector has a unique vector address.

The vector address of a given interrupt is calculated using the Exception Base register (EBase<31:12>), which provides a 4 KB page-aligned base address value located in the kernel segment (KSEG) address space.

### 8.6.1 Multi-Vector Modes Address Calculation

The Computed Offset mode address is calculated by using the EBase and VS<6:0> (IntCtl<9:5> or INTCON<20:16>) values. The VS<6:0> bits provide the spacing between adjacent vector addresses. Modifications to EBase and VS<6:0> values are only allowed when the BEV bit (Status<22>) is '1' in the CPU. Equation 8-1 shows the formula for calculating the vector address in Computed Offset mode and Example 8-4 shows how a computed offset address is calculated for Timer4 (vector 16).

**Note:** The Multi-Vector mode address calculation depends on the interrupt vector number. Each PIC32 device family may have its own set of vector numbers depending on its feature set. For vector numbers associated with each interrupt source, refer to the “Interrupt Controller” chapter in the specific device data sheet.

#### Equation 8-1: Computed Offset Mode Vector Address Calculation Formula

$$\text{Computed Offset Vector Address} = \text{Vector Spacing}(\text{VS}<6:0>) + \text{EBase} + 0x200$$

#### Example 8-4: Vector Address for Vector Number 16 (Computed Offset)

```
Exception Base is 0xBD000000
Vector Spacing(VS) is 64(0x40)
vector address(T4) = 0x10 X 0x40 + 0x200 + 0xBD000000
vector address(T4) = 0xBD000600
```

If the OFFx register is present in the device, the vector address is calculated by adding the vector offset to EBase. Equation 8-2 shows the formula for calculating the vector address in Variable Offset mode. The vector address of Timer4 (vector 16) is being calculated, as shown in Example 8-5.

#### Equation 8-2: Variable Offset Mode Vector Address Calculation Formula

$$\text{Variable Offset Vector Address} = \text{EBase} + \text{Vector Offset}(\text{OFFx})$$

#### Example 8-5: Vector Address for Vector Number 16 (Variable Offset)

```
Exception Base is 0x80000000
Vector offset(OFF8) is 0x0600
vector address(T4) = 0x80000000 + 0x600
vector address(T4) = 0x80000600
```

## 8.6.2 Single Vector Mode Address Calculation

The Single Vector mode address is calculated by using the EBase<17:0> bits (EBase<29:12>). In Single Vector mode, the Interrupt Controller always presents a vector number of '0'. The formula for Single Vector mode is shown in [Equation 8-3](#). [Example 8-6](#) shows how the single vector address is calculated.

### Equation 8-3: Single Vector Mode Address Calculation

$$\text{Single Vector Address} = \text{EBase} + 0x200$$

### Example 8-6: Single Vector Address

```
Exception Base is 0x80000000  
vector address = 0x80000200
```

## 8.7 INTERRUPT PRIORITIES

### 8.7.1 Interrupt Group Priority

The user can assign a group priority to each of the interrupt vectors. The group priority level bits are located in the IPCx register. Each IPCx register contains group priority bits for four interrupt vectors. The user-selectable priority levels range from 1 (lowest) to 7 (highest). If an interrupt priority is set to zero, the interrupt vector is disabled for both interrupt and wake-up purposes. Interrupt vectors with a higher priority level preempt lower priority interrupts. The user must move the RIPL<2:0> bits (Cause<12:10>) into the IPL<2:0> bits (Status<12:10>) before re-enabling interrupts. For more information on the Cause and Status registers, refer to **Section 2. “CPU”** (DS60001113). This action will disable all lower priority interrupts until the completion of the Interrupt Service Routine (ISR).

**Note:** The Interrupt Service Routine must clear the associated interrupt flag in the IFSx register before lowering the interrupt priority level to avoid recursive interrupts.

#### Example 8-7: Setting Group Priority Level

```
/*
The following code example will set the priority to level 2.
Multi-Vector initialization must be performed (See Example 8-2)
*/
IPC0CLR = 0x0000001C;    // clear the priority level
IPC0SET = 0x00000008;    // set priority level to 2
```

### 8.7.2 Interrupt Subpriority

The user can assign a subpriority level within each group priority. The subpriority will not cause preemption of an interrupt in the same priority; rather, if two interrupts with the same priority are pending, the interrupt with the highest subpriority will be handled first. The subpriority bits are located in the IPCx register. Each IPCx register contains subpriority bits for four of the interrupt vectors. These bits define the subpriority within the priority level of the vector. The user-selectable subpriority levels range from 0 (lowest) to 3 (highest).

#### Example 8-8: Setting Subpriority Level

```
/*
The following code example will set the subpriority to level 2.
Multi-Vector initialization must be performed (See Example 8-2)
*/
IPC0CLR = 0x00000003;    // clear the subpriority level
IPC0SET = 0x00000002;    // set the subpriority to 2
```

### 8.7.3 Interrupt Natural Priority

When multiple interrupts are assigned to same group priority and subpriority, they are prioritized by their natural priority. The natural priority is a fixed priority scheme, where the highest natural priority starts at the lowest interrupt vector, meaning that interrupt vector 0 is the highest natural priority. Refer to the Interrupt Vector Table (IVT) in the “**Interrupt Controller**” chapter of the specific device data sheet to determine the natural priority order of each IRQ.

## 8.8 INTERRUPTS AND REGISTER SETS

The PIC32 family of devices employs two register sets, a primary register set for normal program execution and a shadow register set for highest priority interrupt processing. Register set selection is automatically performed by the Interrupt Controller. The exact method of register set selection varies by the Interrupt Controller modes of operation.

In Single Vector and Multi-Vector modes of operation, the CSS bit in the SRSCtl register provides the current number of the register set in use, while the PSS bit provides the number of the previous register set. The SRSCtl register is a CPU register, refer to **Section 2. “CPU”** (DS60001113) for details. This information is useful to determine if the Stack and Global Data Pointers should be copied to the new register set, or not. If the current and previous register set are different, the interrupt handler prologue may need to copy the Stack and Global Data Pointers from one set to another. Most C compilers supporting the PIC32 family of devices automatically generate the necessary interrupt prologue code to handle this operation.

### 8.8.1 Shadow Register Set Selection in Single Vector Mode

In Single Vector mode, the SS0 bit (INTCON<16> or PRISS<0>) determines which register set to be used. If SS0 bit is set to '1', the Interrupt Controller will instruct the CPU to use a shadow register set for all interrupts. If SS0 bit is set to '0', the Interrupt Controller will instruct the CPU to use only the first register set. Unlike Multi-Vector mode, there is no linkage between register set and interrupt priority. The application decides whether the shadow set will be used at all.

### 8.8.2 Shadow Register Set Selection in Multi-Vector Mode

When a priority level interrupt matches a shadow set priority, the Interrupt Controller instructs the CPU to use the shadow set.

Depending on the device, there are different ways that a priority level can match a shadow set. For example, this can be done in the fuses of the configuration bits, it can be set to only one priority level, or if the PRISS register is present, each priority level can be given its own shadow set register. When using the PRISS register, if a value for a shadow set is given which is not available, no shadow set will be used. Refer to the **“Interrupt Controller”** chapter in the specific device data sheet for more information.

For all other interrupt priorities, the Interrupt Controller instructs the CPU to use the primary register set. The interrupt priority that uses the shadow set will not need to perform any context save and restore. This results in increased code throughput and decreases interrupt latency.

### 8.9 INTERRUPT PROCESSING

When the priority of a requested interrupt is greater than the current CPU priority, the interrupt request is taken and the CPU branches to the vector address associated with the requested interrupt. Depending on the priority of the interrupt, regardless of whether or not a shadow set will be used, the prologue and epilogue of the interrupt handler must perform certain tasks before executing any useful code.

The interrupt handler routine must generate a prologue and an epilogue to configure, save and restore all of the core registers, along with General Purpose Registers. At a worst case, all of the modifiable General Purpose Registers must be saved and restored by the prologue and the epilogue, which is usually performed by the compiler.

### 8.10 EXTERNAL INTERRUPTS

The Interrupt Controller supports up to five external interrupt-request signals (INT4-INT0). These inputs are edge sensitive, they require a low-to-high or a high-to-low transition to create an interrupt request. The INTCON register has the INTxEP bits that select the polarity of the edge detection circuitry.

**Note:** Changing the external interrupt polarity may trigger an interrupt request. It is recommended that before changing the polarity, the user disables that interrupt, changes the polarity, clears the interrupt flag and re-enables the interrupt.

#### Example 8-9: Setting External Interrupt Polarity

```
/*
The following code example will set INT3 to trigger on a high-to-low
transition edge. The CPU must be set up for either multi or single vector
interrupts to handle external interrupts
*/
IEC0CLR = 0x00008000;    // disable INT3
INTCONCLR = 0x00000008;  // clear the bit for falling edge trigger
IFS0CLR = 0x00008000;    // clear the interrupt flag
IEC0SET = 0x00008000;    // enable INT3
```

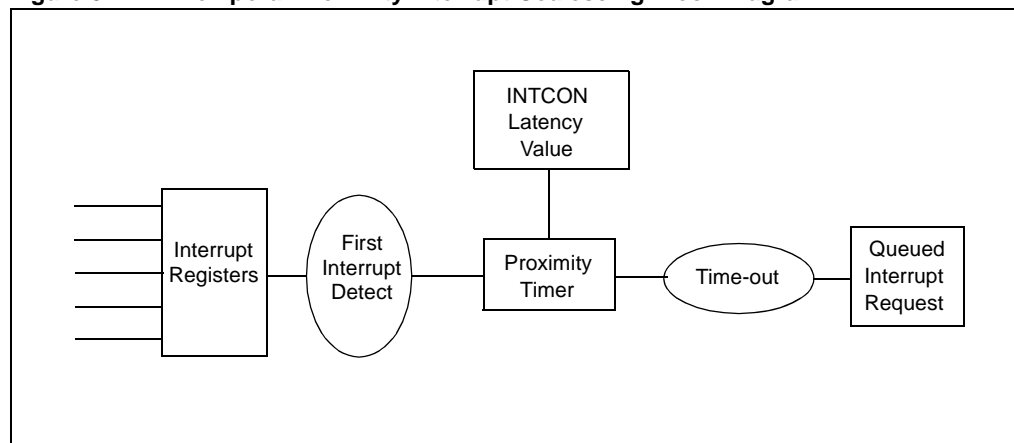
## 8.11 TEMPORAL PROXIMITY INTERRUPT COALESCING

The PIC32 CPU responds to interrupt events as if they are all immediately critical because the Interrupt Controller asserts the interrupt request to the CPU when the interrupt request occurs. The CPU immediately recognizes the interrupt if the current CPU priority is lower than the pending priority. Entering and exiting an ISR consumes clock cycles for saving and restoring context. Events are asynchronous with respect to the main program and have a limited possibility of occurring simultaneously or close together in time. This prevents the ability of a shared ISR to process multiple interrupts at a time.

The Temporal Proximity Interrupt uses the interrupt proximity timer, IPTMR, to create a temporal window in which a group of interrupts of the same, or lower priority will be held off. This provides an opportunity to queue these interrupt requests and process them using tail-chaining multiple IRQs in a single ISR.

Figure 8-2 shows a block diagram of the temporal proximity interrupt coalescing. The interrupt priority group level that triggers the temporal proximity timer is set up in the TPC<2:0> bits (INTCON<10:8>). The TPC bits select the interrupt group priority value, and values below that will trigger the temporal proximity timer to be reset and loaded with the value in the IPTMR register. After the timer is loaded with the value in the IPTMR register, reads to the IPTMR will indicate the current state of the timer. The timer decrements to zero on the rising edge of the System Clock, SYSCLK. When the timer decrements to zero, the queued interrupt requests are serviced if IPL<2:0> bits (Status<12:10>) are less than RIPL<2:0> bits (Cause<12:10>).

**Figure 8-2: Temporal Proximity Interrupt Coalescing Block Diagram**



The user can activate temporal proximity interrupt coalescing by performing the following steps:

1. Set the TPC to the preferred priority level (setting the TPC to zero will disable the proximity timer).
2. Load the preferred 32-bit value to the IPTMR register.

The interrupt proximity timer will trigger when an interrupt request of a priority equal, lower, matches the TPC value.

### Example 8-10: Temporal Proximity Interrupt Coalescing Example

```

/*
The following code example will set the Temporal Proximity Coalescing to
trigger on interrupt priority level of 3 or below and the temporal timer to
be set to 0x12345678.
*/

INTCONCLR = 0x00000700;    // clear TPC
IPTMRCLR = 0xFFFFFFFF;    // clear the timer
INTCONSET = 0x00000300;    // set TPC->3
IPTMR = 0x12345678;       // set the timer to 0x12345678
  
```

### 8.12 SOFTWARE GENERATED NON-MASKABLE INTERRUPT

A software-triggerable non-maskable Interrupt (NMI) (see **Note**) is provided from the Interrupt Controller. Unlike the SWNMI bit in the RNMICON register, this interrupt is triggered by writing a specific value to the NMIKEY<7:0> bits in the INTCON register.

When a value of 0x4E is written to NMIKEY<7:0> bits (NVMCON<31-24>), a non-maskable interrupt is sent to the Reset module. Within that module, the GNMI bit in the RNMICON register will be set, indicating that this was a global NMI event. The processor will go to the NMI handler, and this event can be handled from that point.

<p><b>Note:</b> This feature is not available on all devices. Refer to the “<b>Interrupt Controller</b>” and “<b>Reset</b>” chapters in the specific device data sheet to determine availability and for information on the RNMICON register.</p>
---

## 8.13 EFFECTS OF INTERRUPTS AFTER RESET

### 8.13.1 Device Reset

All Interrupt Controller registers are forced to their reset states upon a device Reset.

### 8.13.2 Power-on Reset

All Interrupt Controller registers are forced to their reset states upon a Power-on Reset.

### 8.13.3 Watchdog Timer Reset

All Interrupt Controller registers are forced to their reset states upon a Watchdog Timer Reset.

## 8.14 OPERATION IN POWER-SAVING AND DEBUG MODES

### 8.14.1 Interrupt Operation in Sleep Mode

During Sleep mode, the Interrupt Controller will only recognize interrupts from peripherals that can operate in Sleep mode. Peripherals such as RTCC, Change Notice, External Interrupts, ADC, and SPI Slave can continue to operate in Sleep mode and interrupts from these peripherals can be used to wake-up the device. An interrupt with its Interrupt Enable bit set may switch the device to either Run or Idle mode, subject to its Interrupt Enable bit status and priority level. An interrupt event with its Interrupt Enable bit cleared or a priority of zero will not be recognized by the Interrupt Controller and cannot change device status. If the priority of the interrupt request is higher than the current processor priority level, the device will switch to Run mode and processor will execute the corresponding interrupt request. If the proximity timer is enabled and the pending interrupt priority is less than the temporal proximity priority, the processor does not remain in sleep. It transitions to idle and then goes to run, once the temporal proximity timer times out. If the priority of the interrupt request is less than or equal to the current processor priority level, the device will switch to Idle mode and the processor will remain halted.

### 8.14.2 Interrupt Operation in Idle Mode

During Idle mode, interrupt events, with their respective Interrupt Enable bits set, may switch the device to Run mode subject to its Interrupt Enable bit status and priority level. An interrupt event with its Interrupt Enable bit cleared or a priority of zero will not be recognized by the Interrupt Controller and cannot change device status. If the priority of the interrupt request is higher than the current CPU priority level, the device will switch to Run mode and the CPU will execute the corresponding interrupt request. If the proximity timer is enabled and the pending interrupt priority is less than the temporal proximity priority, the device will remain in Idle mode and the processor will not take the interrupt until after the proximity time has expired. If the priority of the interrupt request is less than or equal to the current CPU priority level, the device will remain in Idle mode. The corresponding Interrupt Flag bits will remain set and the interrupt request will remain pending.

### 8.14.3 Interrupt Operation in Debug Mode

While the CPU is executing in Debug Exception mode (i.e., the application is halted), all interrupts, regardless of their priority level, are not taken and they will remain pending. Once the CPU exits Debug Exception mode, all pending interrupts will be taken in their order of priority.



### 8.15 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 PIC32 device 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.	N/A

<p><b>Note:</b> Please visit the Microchip web site (<a href="http://www.microchip.com">www.microchip.com</a>) for additional application notes and code examples for the PIC32 family of devices.</p>
--

## 8.16 REVISION HISTORY

### Revision A (August 2007)

This is the initial released version of this document.

### Revision B (October 2007)

Updated document to remove Confidential status.

### Revision C (April 2008)

Revised status to Preliminary; Revised U-0 to r-x.

### Revision D (June 2008)

Revise Register 8-1, FRZ note; Revise Examples 8-1 and 8-2; Change Reserved bits from "Maintain as" to "Write".

### Revision E (July 2009)

This revision includes the following updates:

- Minor updates to text and formatting have been implemented throughout the document
- Interrupts Register Summary (Table 8-1):
  - Removed all references to the Clear, Set and Invert registers
  - Added the Address Offset column
  - Added Notes 1, 2 and 3, which describe the Clear, Set and Invert registers
- Added Notes describing the Clear, Set and Invert registers to the following registers:
  - INTCON
  - INTSTAT
  - IPTMR
  - IFSx
  - IPCx
- Updated the note at the beginning of Section 8.2 "Control Registers"
- Updated the second sentence of the second paragraph in Section 8.3 "Operation" to clarify the IRQ sources
- Updated the first paragraph of Section 8.8.2 "Register Set Selection in Multi-Vector Mode"
- Updated the answer to Question 2 in Section 8.14 "Design Tips"

### Revision F (July 2011)

This revision includes the following updates:

- Added a Note at the beginning of the section, which provides information on the complementary documentation
- Changed all occurrences of PIC32MX to PIC32
- Updated all r-x bits as U-0 bits in Register 8-1 through Register 8-7
- Updated the RIPL bit as the SRIPL bit in Register 8-3
- Updated Example 8-1 and Example 8-2
- Updated Temporal Proximity Timer register (TPTMR) as Interrupt Proximity Timer register (IPTMR) in Register 8-4
- Added a sentence in the third paragraph of section 8.11 "**Temporal Proximity Interrupt Coalescing**" about timer decrementing to zero on the rising edge of the SYSCLK
- Modifications to register formatting and minor updates have been made throughout the document
- Removed Section 8.14 "Design Tips"

### Revision G (April 2012)

This revision includes the following updates:

- Updated the maximum number interrupt sources from 96 to 256 (see **8.1 “Introduction”**)
- Added the Vector Spacing bits (VS<6:0>) to the Interrupt Control register (see Table 8-1 and Register 8-1)
- Added the Priority Shadow Select register (see Table 8-1 and Register 8-2)
- Added the Interrupt Offset register (see Table 8-1 and Register 8-8)
- Minor updates to text and formatting were incorporated throughout the document

### Revision H (JULY 2015)

This revision includes the following updates:

- Register definitions were added (see **8.2 “Control Registers”**)
- The NMIKEY<7:0> bits were added to the INTCON register (see [Table 8-1](#) and [Register 8-1](#))
- **8.12 “Software Generated Non-Maskable Interrupt”** was added
- Minor updates to text and formatting were incorporated throughout the document

NOTES:

---

**Note the following details of the code protection feature on Microchip devices:**

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

---

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

#### **Trademarks**

The Microchip name and logo, the Microchip logo, dsPIC, FlashFlex, flexPWR, JukeBlox, KEELOQ, KEELOQ logo, Klear, LANCheck, MediaLB, MOST, MOST logo, MPLAB, OptoLyzer, PIC, PICSTART, PIC<sup>32</sup> logo, RightTouch, SpyNIC, SST, SST Logo, SuperFlash and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

The Embedded Control Solutions Company and mTouch are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, BodyCom, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, ECAN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, KlearNet, KlearNet logo, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, RightTouch logo, REAL ICE, SQI, Serial Quad I/O, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2007-2015, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-63277-580-1

**QUALITY MANAGEMENT SYSTEM**  
**CERTIFIED BY DNV**  
**= ISO/TS 16949 =**

*Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*

## Worldwide Sales and Service

### AMERICAS

**Corporate Office**  
2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 480-792-7200  
Fax: 480-792-7277  
Technical Support:  
[http://www.microchip.com/  
support](http://www.microchip.com/support)  
Web Address:  
[www.microchip.com](http://www.microchip.com)

**Atlanta**  
Duluth, GA  
Tel: 678-957-9614  
Fax: 678-957-1455

**Austin, TX**  
Tel: 512-257-3370

**Boston**  
Westborough, MA  
Tel: 774-760-0087  
Fax: 774-760-0088

**Chicago**  
Itasca, IL  
Tel: 630-285-0071  
Fax: 630-285-0075

**Cleveland**  
Independence, OH  
Tel: 216-447-0464  
Fax: 216-447-0643

**Dallas**  
Addison, TX  
Tel: 972-818-7423  
Fax: 972-818-2924

**Detroit**  
Novi, MI  
Tel: 248-848-4000

**Houston, TX**  
Tel: 281-894-5983

**Indianapolis**  
Noblesville, IN  
Tel: 317-773-8323  
Fax: 317-773-5453

**Los Angeles**  
Mission Viejo, CA  
Tel: 949-462-9523  
Fax: 949-462-9608

**New York, NY**  
Tel: 631-435-6000

**San Jose, CA**  
Tel: 408-735-9110

**Canada - Toronto**  
Tel: 905-673-0699  
Fax: 905-673-6509

### ASIA/PACIFIC

**Asia Pacific Office**  
Suites 3707-14, 37th Floor  
Tower 6, The Gateway  
Harbour City, Kowloon  
Hong Kong  
Tel: 852-2943-5100  
Fax: 852-2401-3431

**Australia - Sydney**  
Tel: 61-2-9868-6733  
Fax: 61-2-9868-6755

**China - Beijing**  
Tel: 86-10-8569-7000  
Fax: 86-10-8528-2104

**China - Chengdu**  
Tel: 86-28-8665-5511  
Fax: 86-28-8665-7889

**China - Chongqing**  
Tel: 86-23-8980-9588  
Fax: 86-23-8980-9500

**China - Dongguan**  
Tel: 86-769-8702-9880

**China - Hangzhou**  
Tel: 86-571-8792-8115  
Fax: 86-571-8792-8116

**China - Hong Kong SAR**  
Tel: 852-2943-5100  
Fax: 852-2401-3431

**China - Nanjing**  
Tel: 86-25-8473-2460  
Fax: 86-25-8473-2470

**China - Qingdao**  
Tel: 86-532-8502-7355  
Fax: 86-532-8502-7205

**China - Shanghai**  
Tel: 86-21-5407-5533  
Fax: 86-21-5407-5066

**China - Shenyang**  
Tel: 86-24-2334-2829  
Fax: 86-24-2334-2393

**China - Shenzhen**  
Tel: 86-755-8864-2200  
Fax: 86-755-8203-1760

**China - Wuhan**  
Tel: 86-27-5980-5300  
Fax: 86-27-5980-5118

**China - Xian**  
Tel: 86-29-8833-7252  
Fax: 86-29-8833-7256

### ASIA/PACIFIC

**China - Xiamen**  
Tel: 86-592-2388138  
Fax: 86-592-2388130

**China - Zhuhai**  
Tel: 86-756-3210040  
Fax: 86-756-3210049

**India - Bangalore**  
Tel: 91-80-3090-4444  
Fax: 91-80-3090-4123

**India - New Delhi**  
Tel: 91-11-4160-8631  
Fax: 91-11-4160-8632

**India - Pune**  
Tel: 91-20-3019-1500

**Japan - Osaka**  
Tel: 81-6-6152-7160  
Fax: 81-6-6152-9310

**Japan - Tokyo**  
Tel: 81-3-6880-3770  
Fax: 81-3-6880-3771

**Korea - Daegu**  
Tel: 82-53-744-4301  
Fax: 82-53-744-4302

**Korea - Seoul**  
Tel: 82-2-554-7200  
Fax: 82-2-558-5932 or  
82-2-558-5934

**Malaysia - Kuala Lumpur**  
Tel: 60-3-6201-9857  
Fax: 60-3-6201-9859

**Malaysia - Penang**  
Tel: 60-4-227-8870  
Fax: 60-4-227-4068

**Philippines - Manila**  
Tel: 63-2-634-9065  
Fax: 63-2-634-9069

**Singapore**  
Tel: 65-6334-8870  
Fax: 65-6334-8850

**Taiwan - Hsin Chu**  
Tel: 886-3-5778-366  
Fax: 886-3-5770-955

**Taiwan - Kaohsiung**  
Tel: 886-7-213-7828

**Taiwan - Taipei**  
Tel: 886-2-2508-8600  
Fax: 886-2-2508-0102

**Thailand - Bangkok**  
Tel: 66-2-694-1351  
Fax: 66-2-694-1350

### EUROPE

**Austria - Wels**  
Tel: 43-7242-2244-39  
Fax: 43-7242-2244-393

**Denmark - Copenhagen**  
Tel: 45-4450-2828  
Fax: 45-4485-2829

**France - Paris**  
Tel: 33-1-69-53-63-20  
Fax: 33-1-69-30-90-79

**Germany - Dusseldorf**  
Tel: 49-2129-3766400

**Germany - Munich**  
Tel: 49-89-627-144-0  
Fax: 49-89-627-144-44

**Germany - Pforzheim**  
Tel: 49-7231-424750

**Italy - Milan**  
Tel: 39-0331-742611  
Fax: 39-0331-466781

**Italy - Venice**  
Tel: 39-049-7625286

**Netherlands - Drunen**  
Tel: 31-416-690399  
Fax: 31-416-690340

**Poland - Warsaw**  
Tel: 48-22-3325737

**Spain - Madrid**  
Tel: 34-91-708-08-90  
Fax: 34-91-708-08-91

**Sweden - Stockholm**  
Tel: 46-8-5090-4654

**UK - Wokingham**  
Tel: 44-118-921-5800  
Fax: 44-118-921-5820