TB3162

Vectored Interrupt Controller on 8-Bit PIC® Microcontrollers

Author: *June Anthony Asistio*
Microchip Technology Inc.

INTRODUCTION

An interrupt is a request that temporarily stops a microcontroller from running the main routine to execute a task, called the Interrupt Service Routine (ISR). Typically, an interrupt vector is shared by a number of interrupt sources which are contained inside an interrupt handler. When an interrupt occurs, the interrupt flags are scanned inside the handler to determine the source of interrupt, and then the ISR for that interrupt source is called. The vectored interrupt controller module offers an alternative approach by using the Interrupt Vector Table (IVT). The IVT provides each interrupt source an interrupt vector. When interrupt occurs, the interrupt routine is executed directly, without the need to scan the flag bits of the interrupt sources.

The purpose of this Technical Brief is to demonstrate the functionality and flexibility of the Vectored Interrupt Controller in handling and implementing interrupt routines.

VECTORED INTERRUPT CONTROLLER OPERATION

Figure 1 shows the vectored interrupt state transition diagram of the Vectored Interrupt Controller. Interrupts can be grouped up to two levels of priority, a high and low-priority level, by setting the IPEN bit of the INTCON0 register. For concurrent high and low-priority interrupt requests, the high-priority interrupt is always serviced first (Figure 2). A high-priority interrupt signal can also preempt an ongoing low-priority ISR (Figure 3).

FIGURE 1: VECTORED INTERRUPTS STATE TRANSITION DIAGRAM

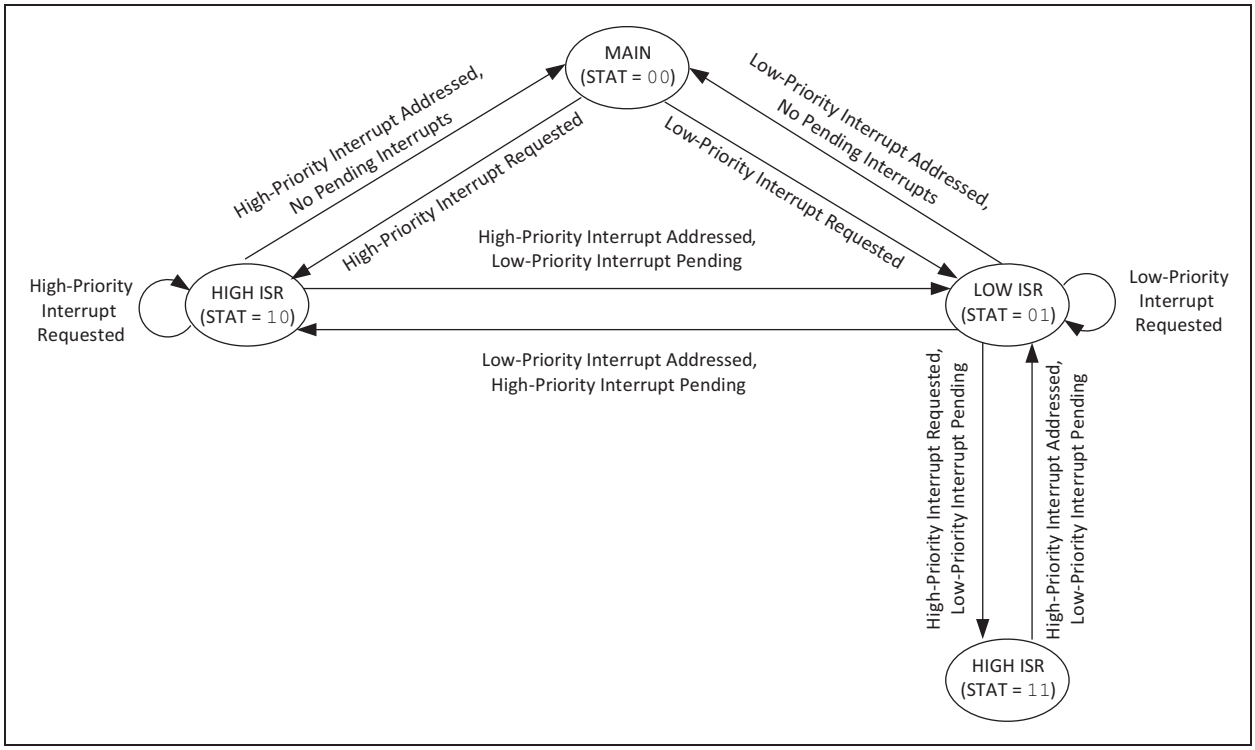


FIGURE 2: HIGH AND LOW-PRIORITY INTERRUPTS RECEIVED SIMULTANEOUSLY

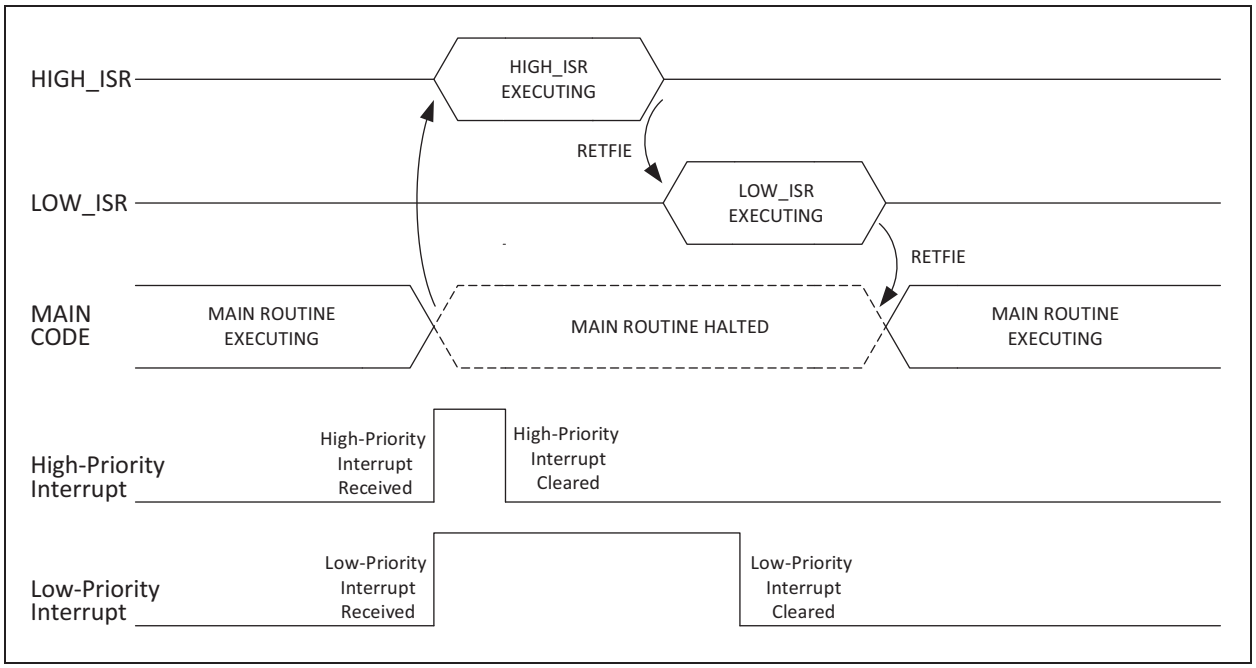
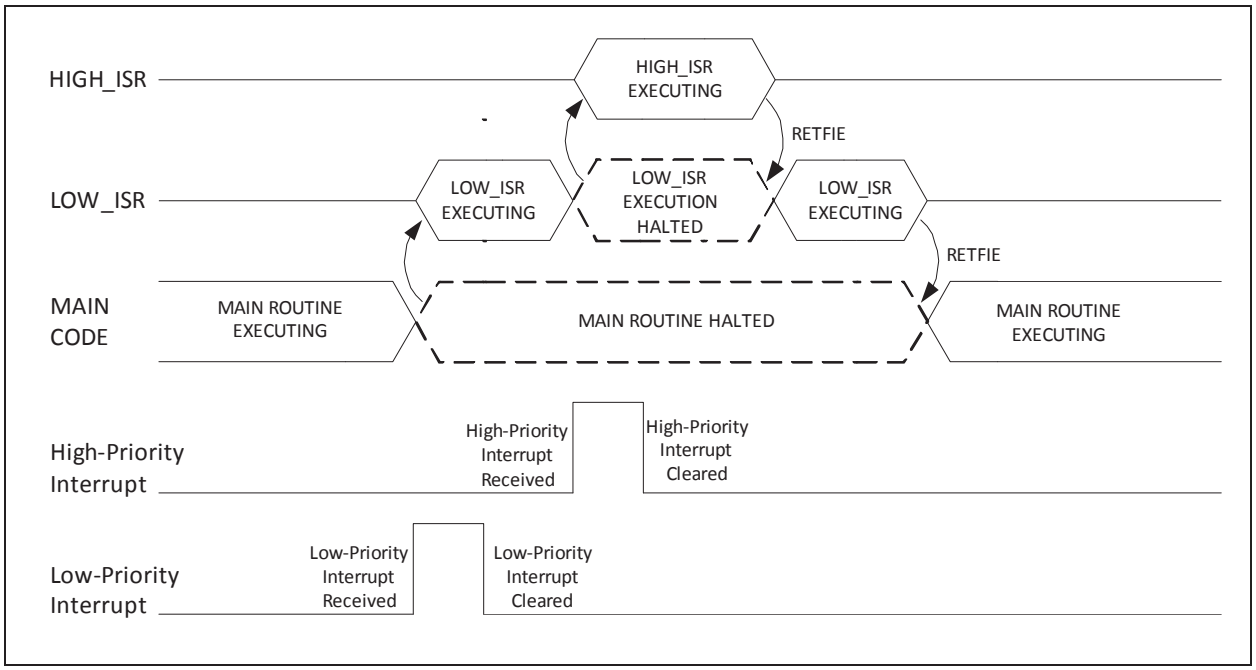


FIGURE 3: A HIGH-PRIORITY INTERRUPT PREEMPTING A LOW ISR



There are two interrupt implementations that are available for the Vectored Interrupt Controller. The first approach uses the interrupt handlers. The Multi-Vector Enable (MVECEN) fuse of the CONFIG_2L register is cleared to use this setting. Two interrupt vectors for the high and low-priority interrupt handlers can be created, and the interrupt signals can be placed into one of these handlers. The IPR registers assign the interrupt source to high or low-priority. Concurrent interrupt requests on the same priority level are resolved by checking the precedence of the polled flag bits inside the interrupt handler.

The second method eliminates the interrupt handlers, but instead, uses the IVT. This can be implemented by setting the MVECEN fuse, which enables the multi-vector interrupts on the IVT. The IPR register assigns the interrupt source to high or low-priority. The IVT gives each interrupt source a dedicated interrupt vector address. The IVT determines the priority for concurrent interrupt requests on the same priority level. The interrupt signal with the highest position (lowest vector number) on the IVT is serviced first. This approach provides better latency than using a software interrupt handler. There is no need to scan the interrupt flags in order to determine the source of the interrupt. Three clock cycles are required to vector to the ISR from the main routine.

Relocatable Interrupt Vector Address

The Vectored Interrupt Controller also features a programmable IVTBASE register that can relocate the base address of the interrupt vectors. Table 1 shows the interrupt vector locations for the high and low-priority handlers of the PIC18FXXK42 for different IVTBASE settings when the MVECEN fuse is off.

TABLE 1: INTERRUPT VECTOR ADDRESSES OF PIC18(L)FXXK42 (MVECEN = OFF)

Interrupt Priority Vectors	IVTBASE (Default)	IVTBASE (0010h)
High-Priority Interrupt Vector = IVTBASE	0008h	0010h
Low-Priority Interrupt Vector = IVTBASE + 8 Words	0018h	0020h

When the MVECEN is on, the IVTBASE register can also change the base location of the vector address of each interrupt source on the IVT. The interrupt vector address that corresponds to each interrupt request can be computed from Equation 1.

EQUATION 1: VECTOR ADDRESS LOCATION

$$Vector\ Address = IVTBASE + 2 \cdot (Vector\ Number)$$

Table 2 shows the IVT for the PIC18(L)FXXK42 device. It shows the interrupt vectors for each interrupt source when the IVTBASE is 0008h and 40F0h.

TABLE 2: IVT OF PIC18(L)FXXK42 FOR IVTBASE = 0008h AND IVTBASE = 40F0h

Vector Number (n)	Interrupt Vector Address (Default IVTBASE)	Interrupt Vector Address (IVTBASE = 40F0h)	Interrupt Source
0	0008h	40F0h	Software Interrupt
1	000Ah	40F2h	HLVD
2	000Ch	40F4h	OSF
3	000Eh	40F6h	CSW
4	0010h	40F8h	NVM
5	0012h	40FAh	SCAN
6	0014h	40FCh	CRC
7	0016h	40FEh	IOC
8	0018h	4100h	INT0
9	001Ah	4102h	ZCD
10	001Ch	4104h	AD
11	001Eh	4106h	ADT
12	0020h	4108h	CMP1
13	0022h	410Ah	SMT1
14	0024h	410Ch	SMU1PRA
15	0026h	410Eh	SMU1PWA
16	0028h	4110h	DMA1SCNT
17	002Ah	4112h	DMA1DCNT
18	002Ch	4114h	DMA1OR
19	002Eh	4116h	DMA1A
20	0030h	4118h	SPI1RX
21	0032h	411Ah	SPI1TX
22	0034h	411Ch	SPI1
23	0036h	411Eh	I2C1RX
24	0038h	4120h	I2C1TX
25	003Ah	4122h	I2C1
26	003Ch	4124h	I2C1E
27	003Eh	4126h	U1R
28	0040h	4128h	U1T
29	0042h	412Ah	U1E
30	0044h	412Ch	U1G
31	0046h	412Eh	TMR0
32	0048h	4130h	TMR1
33	004Ah	4132h	TMR1G
34	004Ch	4134h	TMR2
35	004Eh	4136h	CCP1
36	0050h	4138h	—
37	0052h	413Ah	NCO1
38	0054h	413Ch	CWG1

TABLE 2: IVT OF PIC18(L)FXXK42 FOR IVTBASE = 0008h AND IVTBASE = 40F0h (CONTINUED)

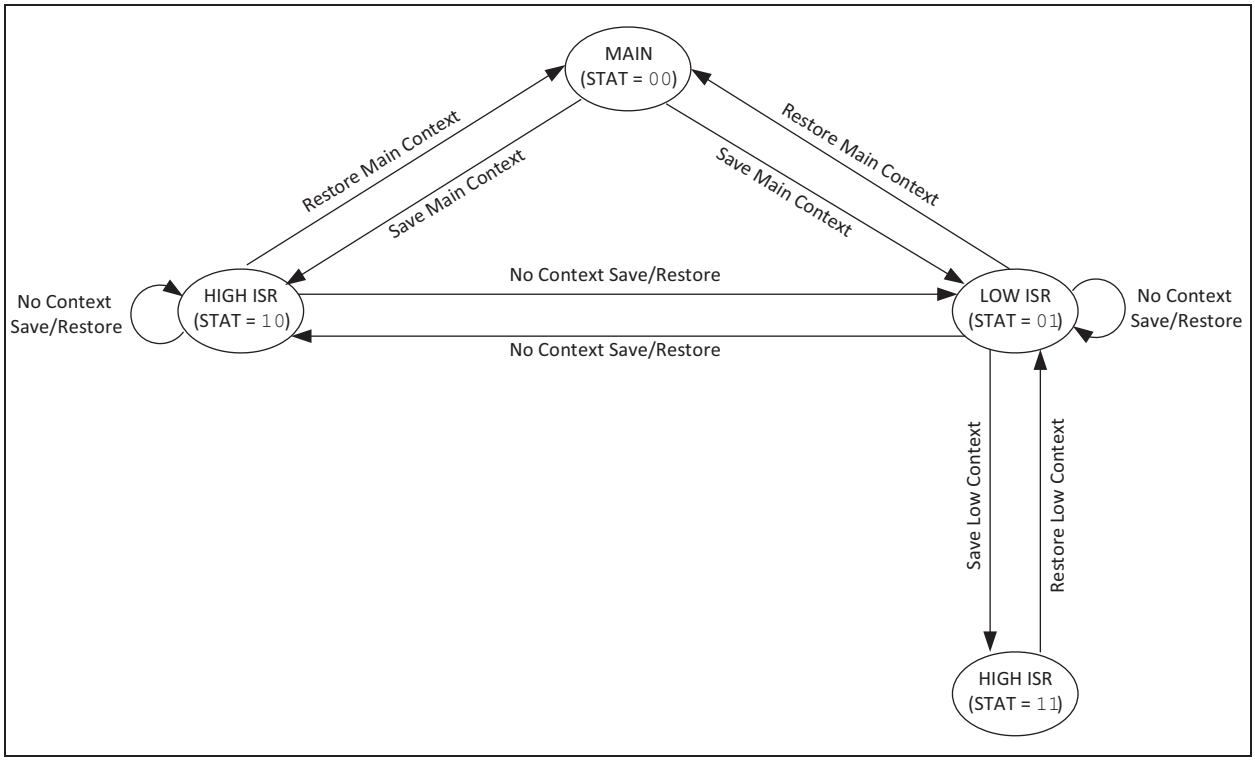
Vector Number (n)	Interrupt Vector Address (Default IVTBASE)	Interrupt Vector Address (IVTBASE = 40F0h)	Interrupt Source
39	0056h	413Eh	CLC1
40	0058h	4140h	INT1
41	005Ah	4142h	CMP2
42	005Ch	4144h	DMA2SCNT
43	005Eh	4146h	DMA2DCNT
44	0060h	4148h	DMA2OR
45	0062h	414Ah	DMA2ABRT
46	0064h	414Ch	I2C2RX
47	0066h	414Eh	I2C2TX
48	0068h	4150h	I2C2
49	006Ah	4152h	I2C2E
50	006Ch	4154h	U2R
51	006Eh	4156h	U2T
52	0070h	4158h	U2E
53	0072h	415Ah	U2G
54	0074h	415Ch	TMR3
55	0076h	415Eh	TMR3G
56	0078h	4160h	TMR4
57	007Ah	4162h	CCP2
58	007Ch	4164h	—
59	007Eh	4166h	CWG2
60	0080h	4168h	CLC2
61	0082h	416Ah	INT2
62	0084h	416Ch	—
63	0086h	416Eh	—
64	0088h	4170h	—
65	008Ah	4172h	—
66	008Ch	4174h	—
67	008Eh	4176h	—
68	0090h	4178h	—
69	0092h	417Ah	—
70	0094h	417Ch	TMR5
71	0096h	417Eh	TMR5G
72	0098h	4180h	TMR6
73	009Ah	4182h	CCP3
74	009Ch	4184h	CWG3
75	009Eh	4186h	CLC3
76	00A0h	4188h	—
77	00A2h	418Ah	—
78	00A4h	418Ch	—
79	00A6h	418Eh	—
80	00A8h	4190h	CCP4
81	00AAh	4192h	CLC4

A relocatable interrupt vector is particularly useful when programming microcontrollers that have a bootloader. Since the bootloader occupies the memory space where the default interrupt vector resides, a programmable IVTBASE can offset the address location for the interrupt vectors of the main routine. Another advantage is that the bootloader can now use its own set of interrupts, and then create a separate interrupt vector, placed on a different location for the main program.

Automatic Context Save and Restore

The interrupt controller can save both the main routine context and the low ISR context automatically. The STATUS, WREG, BSR, FSR0/1/2, PRODL/H and PCLATH/U are saved to their shadow registers. Context save and restore operation is illustrated in Figure 4.

FIGURE 4: CONTEXT SAVE STATE TRANSITION DIAGRAM



CREATING HIGH-PRIORITY AND LOW-PRIORITY INTERRUPTS

Here is an example on how to create interrupts for high-priority and low-priority conditions.

The high-priority interrupts will come from the following sources: INT0, ZCD, CMP1 and CLC1. The low-priority interrupts will come from the timers: TMR0, TMR1, TMR2 and TMR3. Two solutions are presented. The first one uses a dedicated vectored interrupt for each interrupt source and the second solution uses the software interrupt handlers.

1st Solution: Using Multi-Vector Interrupts to (MVECEN = ON)

The IVT are used for the high- and low-priority interrupts. Example 1 shows how to initialize the interrupt controller. The IVTBASE is set at 40F0h. Example 2 shows the code for each ISR, in which each ISR has a dedicated interrupt vector.

EXAMPLE 1: INTERRUPT INITIALIZATION (IPEN = 1; IVTBASE = 40F0h)

```
void  INTERRUPT_Initialize (void)
{
      // Enable priority in interrupts
      INTCON0bits.IPEN = 1;

      // Set IVTBASE
      IVTBASEU = 0x00;
      IVTBASEH = 0x40;
      IVTBASEL = 0xF0;

      //Clear interrupt flags
      PIR1bits.INT0IF = 0;
      PIR1bits.ZCDIF = 0;
      PIR1bits.C1IF = 0;
      PIR4bits.CLC1IF = 0;
      PIR3bits.TMR0IF = 0;
      PIR4bits.TMR1IF = 0;
      PIR4bits.TMR2IF = 0;
      PIR6bits.TMR3IF = 0;

      //Enable interrupts
      PIE1bits.INT0IE = 1;
      PIE1bits.ZCDIE = 1;
      PIE1bits.C1IE = 1;
      PIE4bits.CLC1IE = 1;
      PIE3bits.TMR0IE = 1;
      PIE4bits.TMR1IE = 1;
      PIE4bits.TMR2IE = 1;
      PIE6bits.TMR3IE = 1;

      //Make timer interrupts low priority
      IPR3bits.TMR0IP = 0;
      IPR4bits.TMR1IP= 0;
      IPR4bits.TMR2IP = 0;
      IPR6bits.TMR3IP = 0;

      // Enable interrupts
      INTCON0bits.GIEH = 1;
      INTCON0bits.GIEL = 1;
}
```

EXAMPLE 2: MULTI-VECTOR INTERRUPTS ISR (MVCEN = ON; IVTBASE = 40F0h)

```
//Vectored Interrupts for MVECEN=ON
void __interrupt(irq(IRQ_INT0), base(0x40F0)) INT0_ISR(void)
{
    PIR1bits.INT0IF = 0;       // Clear INT0 interrupt flag
    //place code here
}
void __interrupt(irq(IRQ_ZCD), base(0x40F0)) ZCD_ISR(void)
{
    PIR1bits.ZCDIF = 0;        // Clear ZCD interrupt flag
    //place code here
}
void __interrupt(irq(IRQ_CMP1), base(0x40F0)) C1_ISR(void)
{
    PIR1bits.C1IF = 0;         // Clear C1 interrupt flag
    //place code here
}
void __interrupt(irq(CLC1_TMR3), base(0x40F0)) CLC1_ISR(void)
{
    PIR4bits.CLC1IF = 0;       // Clear CLC1 interrupt flag
    //place code here
}
void __interrupt(irq(IRQ_TMR0), base(0x40F0)) TMR0_ISR(void)
{
    PIR3bits.TMR0IF = 0;       // Clear TMR0 interrupt flag
    //place code here
}
void __interrupt(irq(IRQ_TMR1), base(0x40F0)) TMR1_ISR(void)
{
    PIR4bits.TMR1IF = 0;       // Clear TMR1 interrupt flag
    //place code here
}
void __interrupt(irq(IRQ_TMR2), base(0x40F0)) TMR2_ISR(void)
{
    PIR4bits.TMR2IF = 0;       // Clear TMR2 interrupt flag
    //place code here
}
void __interrupt(irq(IRQ_TMR3), base(0x40F0)) TMR3_ISR(void)
{
    PIR6bits.TMR3IF = 0;       // Clear TMR3 interrupt flag
    //place code here
}
```

As a result, the interrupts sources are placed to high- and low-priority as shown in Table 3. The software interrupt handler is no longer needed in writing the interrupt code.

TABLE 3: HIGH AND LOW-PRIORITY INTERRUPT

High-Priority Interrupt		Low-Priority Interrupt	
Vector Number	Interrupt Source	Vector Number	Interrupt Source
8	INT0	31	TMR0
9	ZCD	32	TMR1
12	CMP1	34	TMR2
39	CLC1	54	TMR3

2nd Solution: Using Legacy Mode Interrupt (MVECEN = OFF)

Two interrupt handlers will be used for high and low priority. Example 3 shows a similar interrupt initialization, except that the IVTBASE is selected at 0010h. The two interrupt handlers are shown in Example 4.

EXAMPLE 3: INTERRUPT INITIALIZATION (IPEN = 1; IVTBASE = 0010h)

```
void  INTERRUPT_Initialize (void)
{
      // Enable priority in interrupts
      INTCON0bits.IPEN = 1;

      // Set IVTBASE
      IVTBASEU = 0x00;
      IVTBASEH = 0x00;
      IVTBASEL = 0x10;

      //Clear interrupt flags
      PIR1bits.INT0IF = 0;
      PIR1bits.ZCDIF = 0;
      PIR1bits.C1IF = 0;
      PIR4bits.CLC1IF = 0;
      PIR3bits.TMR0IF = 0;
      PIR4bits.TMR1IF = 0;
      PIR4bits.TMR2IF = 0;
      PIR6bits.TMR3IF = 0;

      //Enable interrupts
      PIE1bits.INT0IE = 1;
      PIE1bits.ZCDIE = 1;
      PIE1bits.C1IE = 1;
      PIE4bits.CLC1IE = 1;
      PIE3bits.TMR0IE = 1;
      PIE4bits.TMR1IE = 1;
      PIE4bits.TMR2IE = 1;
      PIE6bits.TMR3IE = 1;

      //Make timer interrupts low priority
      IPR3bits.TMR0IP = 0;
      IPR4bits.TMR1IP = 0;
      IPR4bits.TMR2IP = 0;
      IPR6bits.TMR3IP = 0;

      // Enable interrupts
      INTCON0bits.GIEH = 1;
      INTCON0bits.GIEL = 1;
}
```

EXAMPLE 4: HIGH AND LOW-PRIORITY INTERRUPT HANDLERS (MVCEN = OFF; IPEN = 1; IVTBASE = 0010h)

```
// MVECEN = OFF and IPEN = 1
void __interrupt() highPriorityInterrupt010(void)
{
     if(INTCON0bits.GIE == 1 && PIE1bits.INT0IE == 1 && PIR1bits.INT0IF == 1)
    {
        PIR1bits.INT0IF = 0;
        INT0_ISR();

    }
    if(INTCON0bits.GIE == 1 && PIE1bits.ZCDIE == 1 && PIR1bits.ZCDIF == 1)
    {
        PIR1bits.ZCDIF = 0;
        ZCD_ISR();
    }
    if(INTCON0bits.GIE == 1 && PIE1bits.C1IE == 1 && PIR1bits.C1IF == 1)
    {
        PIR1bits.C1IF = 0;
        C1_ISR();
    }
    if(INTCON0bits.GIE == 1 && PIE4bits.CLC1IE == 1 && PIR4bits.CLC1IF == 1)
    {
        PIR4bits.CLC1IF = 0;
        CLC1_ISR();
    }
}

void __interrupt(low_priority) lowPriorityInterrupt020 (void)
{
    if(INTCON0bits.GIE == 1 && PIE6bits.TMR3IE == 1 && PIR6bits.TMR3IF == 1)
    {
        PIR6bits.TMR3IF = 0;
        TMR3_ISR();
    }
    if(INTCON0bits.GIE == 1 && PIE4bits.TMR2IE == 1 && PIR4bits.TMR2IF == 1)
    {
         PIR4bits.TMR2IF = 0;
         TMR2_ISR();
    }
    if(INTCON0bits.GIE == 1 && PIE4bits.TMR1IE == 1 && PIR4bits.TMR1IF == 1)
    {
         PIR4bits.TMR1IF = 0;
         TMR1_ISR();
    }
    if(INTCON0bits.GIE == 1 && PIE3bits.TMR0IE == 1 && PIR3bits.TMR0IF == 1)
    {
         PIR3bits.TMR0IF = 0;
         TMR0_ISR();
    }
}
```

IMPLEMENTING INTERRUPTS USING THE VECTOR NUMBER

During interrupt, the WREG register stores the vector number of the interrupt signal. Example 5 shows another way of implementing the interrupt handler using the vector number when MVECEN is off.

EXAMPLE 5: USING THE VECTOR NUMBER TO IMPLEMENT THE INTERRUPT HANDLER

```
// Vector ID number of the Interrupt Sources
#define INT0  8
#define ZCD   9
#define CMP1  12
#define CLC1  39
#define TMR0  31
#define TMR1  32
#define TMR2  34
#define TMR3  54

// MVECEN = OFF and IPEN = 1
void __interrupt() highPriorityInterrupt010(void)
{
    uint8_t VectorID_High = WREG;
    switch (VectorID_High)
    {
        case INT0:
            PIR1bits.INT0IF = 0;
            INT0_ISR();
            break;
        case ZCD:
            PIR1bits.ZCDIF = 0;
            ZCD_ISR();
            break;
        case CMP1:
            PIR1bits.C1IF = 0;
            C1_ISR();
            break;
        case CLC1:
            PIR4bits.CLC1IF = 0;
            CLC1_ISR();
            break;
        default:
            break;
    }
}

void __interrupt(low_priority) lowPriorityInterrupt020 (void)
{
    uint8_t VectorID_Low = WREG;
    switch (VectorID_Low)
    {
        case TMR3:
            PIR6bits.TMR3IF = 0;
            TMR3_ISR();
            break;
        case TMR2:
            PIR4bits.TMR2IF = 0;
            TMR2_ISR();
            break;
        case TMR1:
            PIR4bits.TMR1IF = 0;
            TMR1_ISR();
            break;
        case TMR0:
            PIR3bits.TMR0IF = 0;
            TMR0_ISR();
            break;
        default:
            break;
    }
}
```

CONCLUSION

The Vectored Interrupt Controller module can handle interrupts up to two priority levels. The module can resolve conflict of concurrent interrupts that are on the same priority level either by taking the precedence of the polled flag bits inside the interrupt handlers, or by using the natural hardware order set by the IVT of the microcontroller. The module gives the flexibility of relocating the interrupt vector addresses. By using multi-vector interrupts of the IVT, the software interrupt handlers can be eliminated, thereby simplifying the code. Enabling the multi-vector interrupts on the IVT gives faster response and better latency than using legacy software interrupt handlers.

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.

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.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Trademarks

ISBN: 978-1-5224-1455-1

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
Web Address:
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

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
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

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 - Guangzhou
Tel: 86-20-8755-8029

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: 86-21-3326-8000
Fax: 86-21-3326-8021

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: 886-7-213-7830

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

Finland - Espoo
Tel: 358-9-4520-820

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

France - Saint Cloud
Tel: 33-1-30-60-70-00

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

Germany - Karlsruhe
Tel: 49-721-625370

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

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

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

Italy - Padova
Tel: 39-049-7625286

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

Norway - Trondheim
Tel: 47-7289-7561

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

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

Sweden - Gothenberg
Tel: 46-31-704-60-40

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

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