TB3162

8位PIC®单片机上的向量中断控制器

作者： *June Anthony Asistio*
Microchip Technology Inc.

简介

中断是一种请求，能够使单片机暂停运行主程序，转而执行称为中断服务程序（Interrupt Service Routine，ISR）的任务。通常，中断向量由中断处理程序中包含的多个中断源共用。发生中断时，处理程序内部将扫描中断标志以确定中断源，然后调用该中断源的ISR。向量中断控制器模块提供了一种替代方案，即使用中断向量表（Interrupt Vector Table，IVT）。IVT为每个中断源提供一个中断向量。发生中断时，将直接执行中断程序，无需扫描中断源的标志位。

本技术简介的目的是演示向量中断控制器在处理和实现中断程序时的功能和灵活性。

向量中断控制器的工作原理

图1所示为向量中断控制器的向量中断状态转换图。通过将INTCON0寄存器的IPEN位置1，可以将中断分组为高优先级和低优先级两个级别。当高优先级和低优先级中断请求同时发生时，始终先处理高优先级中断（图2）。高优先级中断信号也可以抢占正在进行的低优先级ISR（图3）。

图1：　向量中断状态转换图

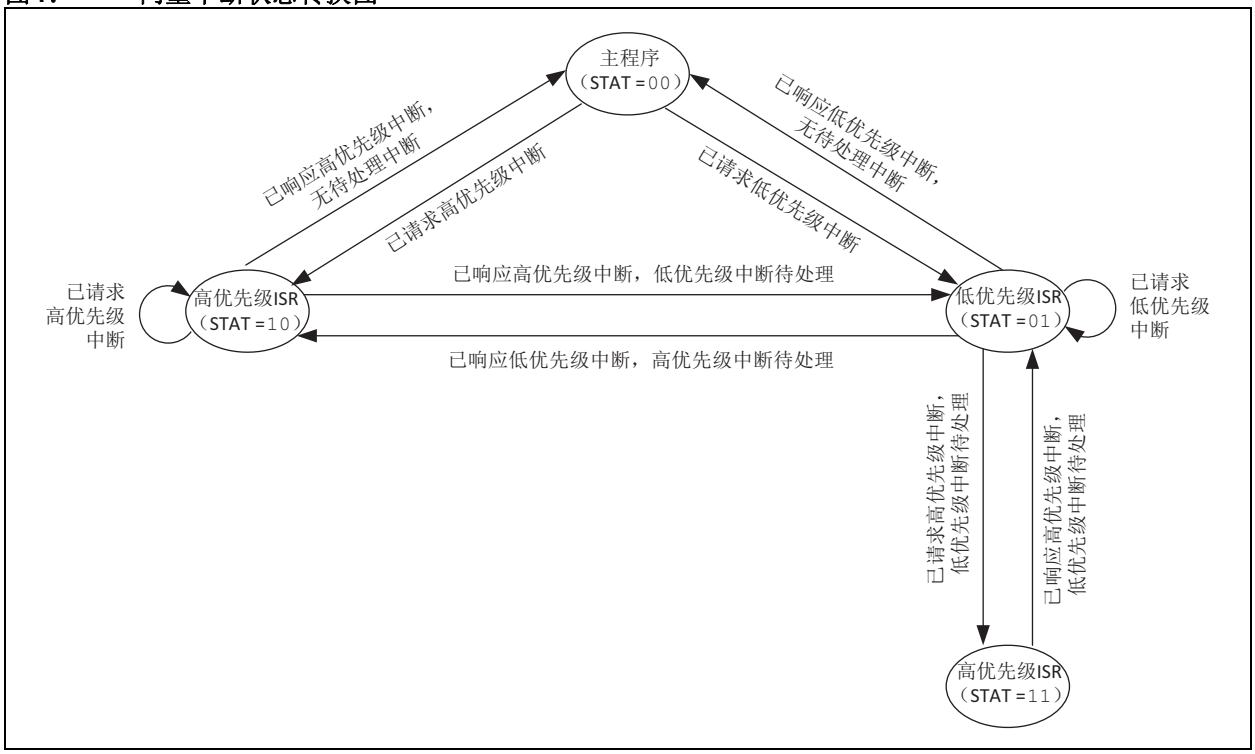


图2：　同时接收到高优先级中断和低优先级中断

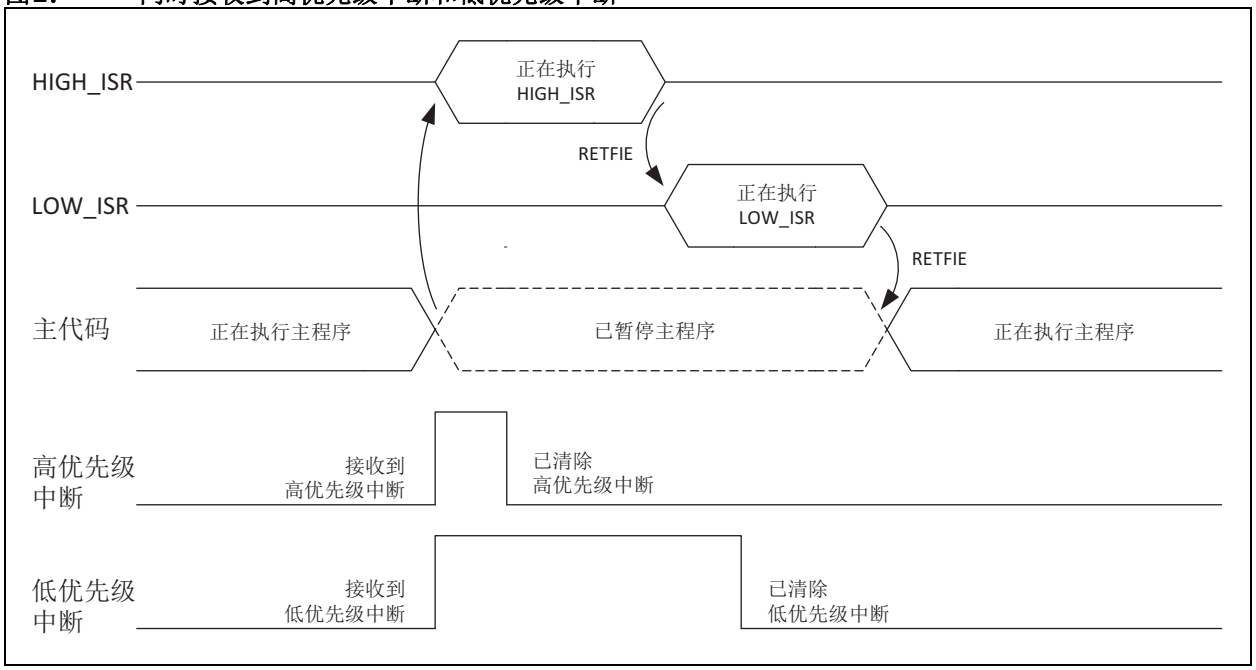
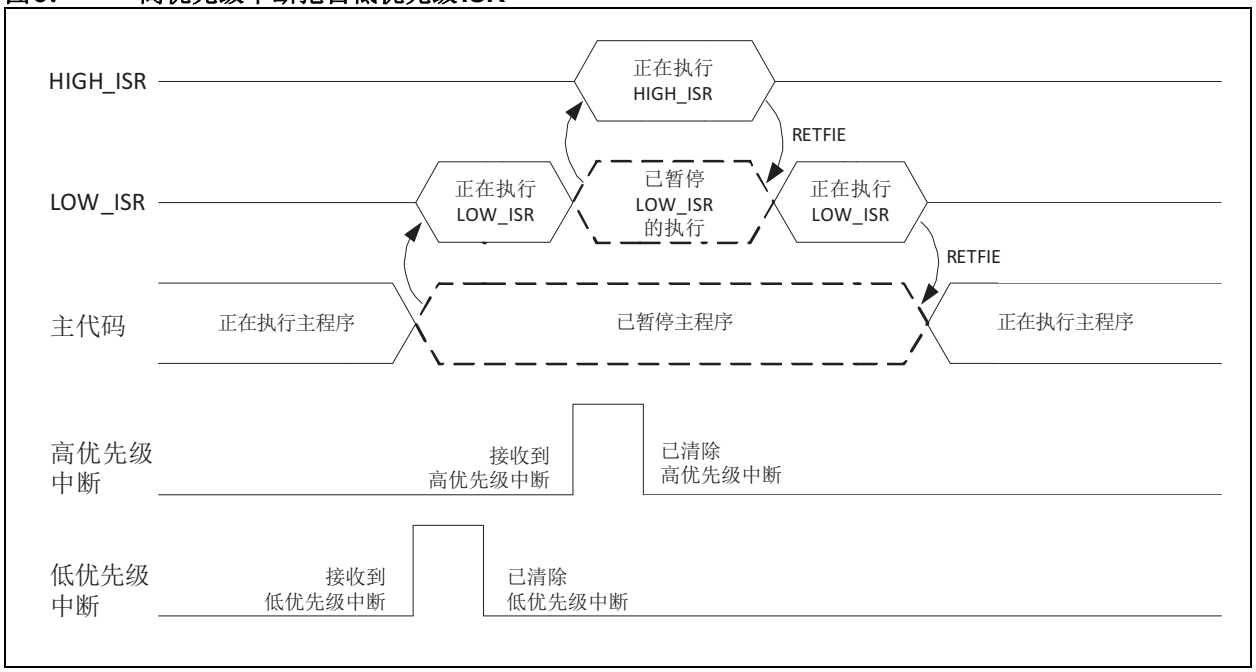


图3：　高优先级中断抢占低优先级ISR



有两种中断方法可用于向量中断控制器。第一种方法是使用中断处理程序。要使用此设置，需将CONFIG_2L寄存器的多向量使能（MVECEN）熔丝清零。可以为高优先级和低优先级中断处理程序创建两个中断向量，并将中断信号放入其中一个处理程序中。IPR寄存器为中断源分配高优先级或低优先级。通过检查中断处理程序中轮询标志位的优先级来解决相同优先级的并发中断请求。

第二种方法无需中断处理程序，而是使用IVT。其实现方式是将MVECEN熔丝置1，以便使用IVT上的多向量中断。IPR寄存器为中断源分配高优先级或低优先级。IVT为每个中断源提供一个专用的中断向量地址。IVT可为同一优先级的并发中断请求确定优先次序。首先处理IVT上最高位置（最低向量编号）的中断信号。该方法可提供比使用软件中断处理程序更优异的延时性能。无需扫描中断标志即可确定中断源。需要三个时钟周期来从主程序指向ISR。

可重定位中断向量地址

向量中断控制器还具有可编程IVTBASE寄存器，可以重定位中断向量的基址。表1所示为MVECEN熔丝禁止时，不同IVTBASE设置对应的PIC18FXXK42高优先级和低优先级处理程序的中断向量单元。

表1：　　PIC18(L)FXXK42的中断向量地址（MVECEN = 禁止）

中断优先级向量	IVTBASE（默认设置）	IVTBASE（0010h）
高优先级中断向量 = IVTBASE	0008h	0010h
低优先级中断向量 = IVTBASE + 8字	0018h	0020h

当MVECEN使能时，IVTBASE寄存器也可以改变IVT上每个中断源的向量地址的基本位置。每个中断请求对应的中断向量地址可以通过公式1计算。

公式1：　　向量地址单元

$$向量地址 = IVTBASE + 2 \cdot (向量编号)$$

表2所示为PIC18(L)FXXK42器件的IVT。该表提供IVTBASE为0008h和40F0h时，每个中断源的中断向量。

表2： PIC18(L)FXXK42的IVT（IVTBASE = 0008h和IVTBASE = 40F0h）

向量编号（n）	中断向量地址（默认IVTBASE）	中断向量地址（IVTBASE = 40F0h）	中断源
0	0008h	40F0h	软件中断
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

表2： PIC18(L)FXXK42的IVT（IVTBASE = 0008h和IVTBASE = 40F0h）（续）

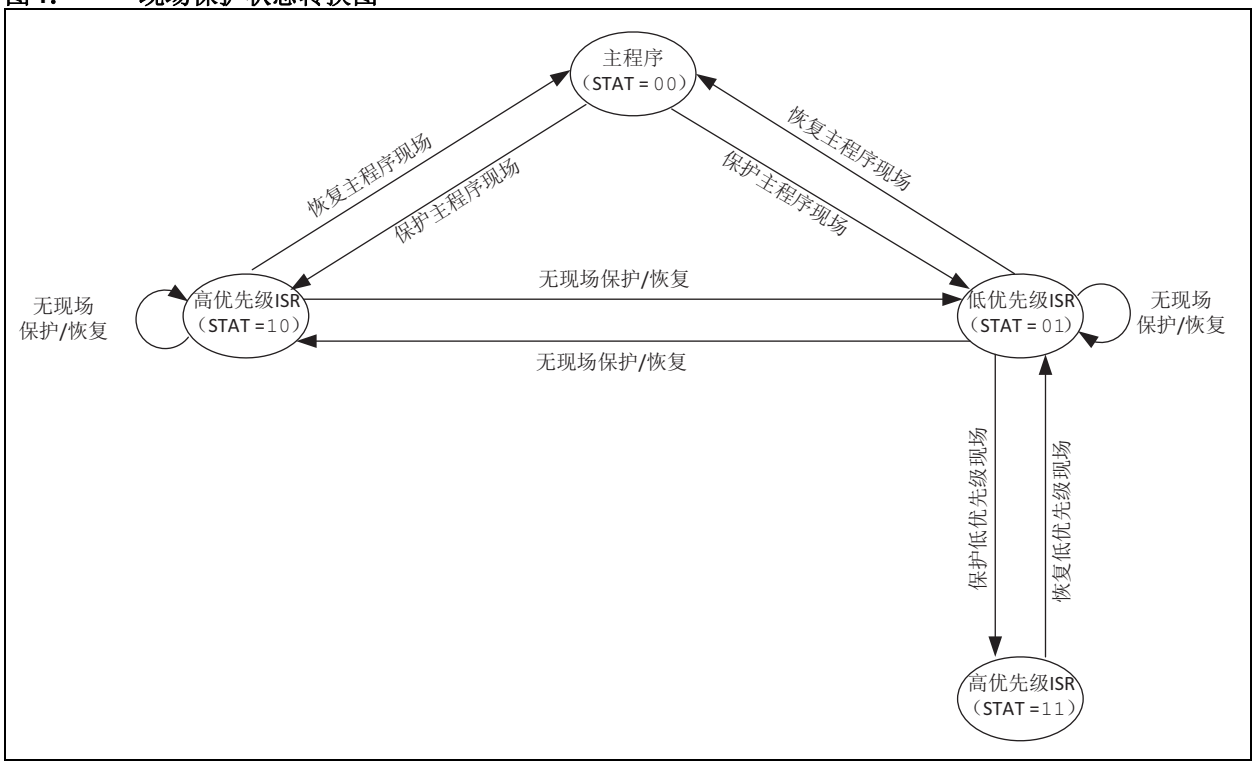
向量编号（n）	中断向量地址（默认IVTBASE）	中断向量地址（IVTBASE = 40F0h）	中断源
39	0056h	413Eh	CLC1
40	0058h	4140h	INT1
41	005Ah	4142h	CMP2
42	005Ch	4144h	DMA2SCNT
43	005Eh	4146h	DMA2DCNT
44	0060h	4148h	DMA2OR
45	0062h	414Ah	DMA2ABRT
46	0064h	414Ch	I2C2RX
47	0066h	414Eh	I2C2TX
48	0068h	4150h	I2C2
49	006Ah	4152h	I2C2E
50	006Ch	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

可重定位中断向量在编程具有自举程序的单片机时特别有用。由于自举程序占用默认中断向量所在的存储空间，因此可编程IVTBASE可以使主程序中断向量的地址单元发生偏移。另一个优点是自举程序现在可以使用自己的一组中断，并在主程序的不同位置创建一个单独的中断向量。

自动现场保护和恢复

中断控制器可以自动保存主程序和低优先级ISR的现场。STATUS、WREG、BSR、FSR0/1/2、PRODL/H和PCLATH/U将保存到其影子寄存器中。现场保护和恢复操作如图4所示。

图4：　　现场保护状态转换图



创建高优先级和低优先级中断

以下是关于如何为高优先级和低优先级条件创建中断的示例。

高优先级中断将来自以下中断源：INT0、ZCD、CMP1和CLC1。低优先级中断将来自以下定时器：TMR0、TMR1、TMR2和TMR3。下面给出了两种解决方案。第一种解决方案是为每个中断源使用专用向量中断，第二种解决方案是使用软件中断处理程序。

第一种解决方案：使用多向量中断（MVECEN = 使能）

为高优先级和低优先级中断使用IVT。例1说明了如何初始化中断控制器。IVTBASE设置为40F0h。例2给出了每个ISR的代码，其中每个ISR都有一个专用的中断向量。

例1：　　中断初始化（IPEN = 1；IVTBASE = 40F0h）

```
void  INTERRUPT_Initialize (void)
{
      // 使能中断优先级
      INTCON0bits.IPEN = 1;

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

      // 清除中断标志
      PIR1bits.INT0IF = 0;
      PIR1bits.ZCDIF = 0;
      PIR1bits.C1IF = 0;
      PIR4bits.CLC1IF = 0;
      PIR3bits.TMR0IF = 0;
      PIR4bits.TMR1IF = 0;
      PIR4bits.TMR2IF = 0;
      PIR6bits.TMR3IF = 0;

      // 允许中断
      PIE1bits.INT0IE = 1;
      PIE1bits.ZCDIE = 1;
      PIE1bits.C1IE = 1;
      PIE4bits.CLC1IE = 1;
      PIE3bits.TMR0IE = 1;
      PIE4bits.TMR1IE = 1;
      PIE4bits.TMR2IE = 1;
      PIE6bits.TMR3IE = 1;

      // 将定时器中断设置为低优先级
      IPR3bits.TMR0IP = 0;
      IPR4bits.TMR1IP= 0;
      IPR4bits.TMR2IP = 0;
      IPR6bits.TMR3IP = 0;

      // 允许中断
      INTCON0bits.GIEH = 1;
      INTCON0bits.GIEL = 1;
}
```

例2：　　多向量中断ISR（MVCEN = 使能；IVTBASE = 40F0h）

```
//MVECEN=使能时的向量中断
void __interrupt(irq(IRQ_INT0), base(0x40F0)) INT0_ISR(void)
{
    PIR1bits.INT0IF = 0;       // 清除INT0中断标志
    //在此放置代码
}
void __interrupt(irq(IRQ_ZCD), base(0x40F0)) ZCD_ISR(void)
{
    PIR1bits.ZCDIF = 0;        // 清除ZCD中断标志
    //在此放置代码
}
void __interrupt(irq(IRQ_CMP1), base(0x40F0)) C1_ISR(void)
{
    PIR1bits.C1IF = 0;         // 清除C1中断标志
    //在此放置代码
}
void __interrupt(irq(CLC1_TMR3), base(0x40F0)) CLC1_ISR(void)
{
    PIR4bits.CLC1IF = 0;       // 清除CLC1中断标志
    //在此放置代码
}
void __interrupt(irq(IRQ_TMR0), base(0x40F0)) TMR0_ISR(void)
{
    PIR3bits.TMR0IF = 0;       // 清除TMR0中断标志
    //在此放置代码
}
void __interrupt(irq(IRQ_TMR1), base(0x40F0)) TMR1_ISR(void)
{
    PIR4bits.TMR1IF = 0;       // 清除TMR1中断标志
    //在此放置代码
}
void __interrupt(irq(IRQ_TMR2), base(0x40F0)) TMR2_ISR(void)
{
    PIR4bits.TMR2IF = 0;       // 清除TMR2中断标志
    //在此放置代码
}
void __interrupt(irq(IRQ_TMR3), base(0x40F0)) TMR3_ISR(void)
{
    PIR6bits.TMR3IF = 0;       // 清除TMR3中断标志
    //在此放置代码
}
```

结果为，中断源将被置于高优先级和低优先级，如表3所示。编写中断代码时不再需要软件中断处理程序。

表3：　　高优先级和低优先级中断

高优先级中断		低优先级中断	
向量编号	中断源	向量编号	中断源
8	INT0	31	TMR0
9	ZCD	32	TMR1
12	CMP1	34	TMR2
39	CLC1	54	TMR3

第二种解决方案：使用传统模式中断（MVECEN = 禁止）

为高优先级和低优先级使用两个中断处理程序。例3所示为类似的中断初始化过程，但IVTBASE选择了0010h。两个中断处理程序如例4所示。

例3：　　中断初始化（IPEN = 1；IVTBASE = 0010h）

```
void  INTERRUPT_Initialize (void)
{
      // 使能中断优先级
      INTCON0bits.IPEN = 1;

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

      // 清除中断标志
      PIR1bits.INT0IF = 0;
      PIR1bits.ZCDIF = 0;
      PIR1bits.C1IF = 0;
      PIR4bits.CLC1IF = 0;
      PIR3bits.TMR0IF = 0;
      PIR4bits.TMR1IF = 0;
      PIR4bits.TMR2IF = 0;
      PIR6bits.TMR3IF = 0;

      // 允许中断
      PIE1bits.INT0IE = 1;
      PIE1bits.ZCDIE = 1;
      PIE1bits.C1IE = 1;
      PIE4bits.CLC1IE = 1;
      PIE3bits.TMR0IE = 1;
      PIE4bits.TMR1IE = 1;
      PIE4bits.TMR2IE = 1;
      PIE6bits.TMR3IE = 1;

      // 将定时器中断设置为低优先级
      IPR3bits.TMR0IP = 0;
      IPR4bits.TMR1IP = 0;
      IPR4bits.TMR2IP = 0;
      IPR6bits.TMR3IP = 0;

      // 允许中断
      INTCON0bits.GIEH = 1;
      INTCON0bits.GIEL = 1;
}
```

例4：　　高优先级和低优先级中断处理程序（MVCEN = 禁止；IPEN = 1；IVTBASE = 0010h）

```
// MVECEN = 禁止且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();
    }
}
```

使用向量编号实现中断

在中断期间，WREG 寄存器存储中断信号的向量编号。例5给出了MVECEN禁止时使用向量编号实现中断处理程序的另一种方式。

例5：　　使用向量编号来实现中断处理程序

```
// 中断源的向量ID编号
#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且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;
    }
}
```

结论

向量中断控制器模块可以处理最多两个优先级的中断。该模块可以通过采用中断处理程序中所轮询标志位的优先顺序，或者使用由单片机的IVT设置的自然硬件顺序来解决同一优先级的并发中断的冲突问题。该模块可灵活地重定位中断向量地址。使用IVT的多向量中断时，无需软件中断处理程序，从而可简化代码。与使用传统软件中断处理程序相比，使用IVT上的多向量中断可实现更快的响应性能和更优异的延时性能。

注：

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。 Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了 《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2009 认证。 Microchip 的 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品严格遵守公司的质量体系流程。此外，Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

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

ISBN：978-1-5224-2367-6

全球销售及服务网点

美洲

公司总部 Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://www.microchip.com/support
网址：www.microchip.com

亚特兰大 Atlanta
Duluth, GA
Tel: 1-678-957-9614
Fax: 1-678-957-1455

奥斯汀 Austin, TX
Tel: 1-512-257-3370

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Novi, MI
Tel: 1-248-848-4000

休斯敦 Houston, TX
Tel: 1-281-894-5983

印第安纳波利斯 Indianapolis
Noblesville, IN
Tel: 1-317-773-8323
Fax: 1-317-773-5453
Tel: 1-317-536-2380

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608
Tel: 1-951-273-7800

罗利 Raleigh, NC
Tel: 1-919-844-7510

纽约 New York, NY
Tel: 1-631-435-6000

圣何塞 San Jose, CA
Tel: 1-408-735-9110
Tel: 1-408-436-4270

加拿大多伦多 Toronto
Tel: 1-905-695-1980
Fax: 1-905-695-2078

亚太地区

中国 - 北京
Tel: 86-10-8569-7000

中国 - 成都
Tel: 86-28-8665-5511

中国 - 重庆
Tel: 86-23-8980-9588

中国 - 东莞
Tel: 86-769-8702-9880

中国 - 广州
Tel: 86-20-8755-8029

中国 - 杭州
Tel: 86-571-8792-8115

中国 - 南京
Tel: 86-25-8473-2460

中国 - 青岛
Tel: 86-532-8502-7355

中国 - 上海
Tel: 86-21-3326-8000

中国 - 沈阳
Tel: 86-24-2334-2829

中国 - 深圳
Tel: 86-755-8864-2200

中国 - 苏州
Tel: 86-186-6233-1526

中国 - 武汉
Tel: 86-27-5980-5300

中国 - 西安
Tel: 86-29-8833-7252

中国 - 厦门
Tel: 86-592-238-8138

中国 - 香港特别行政区
Tel: 852-2943-5100

中国 - 珠海
Tel: 86-756-321-0040

台湾地区 - 高雄
Tel: 886-7-213-7830

台湾地区 - 台北
Tel: 886-2-2508-8600

台湾地区 - 新竹
Tel: 886-3-577-8366

亚太地区

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733

印度 India - Bangalore
Tel: 91-80-3090-4444

印度 India - New Delhi
Tel: 91-11-4160-8631

印度 India - Pune
Tel: 91-20-4121-0141

日本 Japan - Osaka
Tel: 81-6-6152-7160

日本 Japan - Tokyo
Tel: 81-3-6880-3770

韩国 Korea - Daegu
Tel: 82-53-744-4301

韩国 Korea - Seoul
Tel: 82-2-554-7200

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870

菲律宾 Philippines - Manila
Tel: 63-2-634-9065

新加坡 Singapore
Tel: 65-6334-8870

泰国 Thailand - Bangkok
Tel: 66-2-694-1351

越南 Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

欧洲

奥地利 Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

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

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

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

德国 Germany - Garching
Tel: 49-8931-9700

德国 Germany - Haan
Tel: 49-2129-3766400

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

德国 Germany - Karlsruhe
Tel: 49-721-625370

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

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

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

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

意大利 Italy - Padova
Tel: 39-049-7625286

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

挪威 Norway - Trondheim
Tel: 47-7289-7561

波兰 Poland - Warsaw
Tel: 48-22-3325737

罗马尼亚 Romania - Bucharest
Tel: 40-21-407-87-50

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

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

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

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