第 8 章 中断

目录

本章包括下列主题：

8.1 简介 8-2
8.2 控制寄存器 8-3
8.3 工作原理 8-12
8.4 单向量模式 8-14
8.5 多向量模式 8-15
8.6 中断向量地址计算 8-16
8.7 中断优先级 8-17
8.8 中断和寄存器集 8-18
8.9 中断处理 8-19
8.10 外部中断 8-23
8.11 时间接近中断聚合 8-24
8.12 复位之后中断的影响 8-25
8.13 节能和调试模式下的操作 8-25
8.14 设计技巧 8-26
8.15 相关应用笔记 8-27
8.16 版本历史 8-28

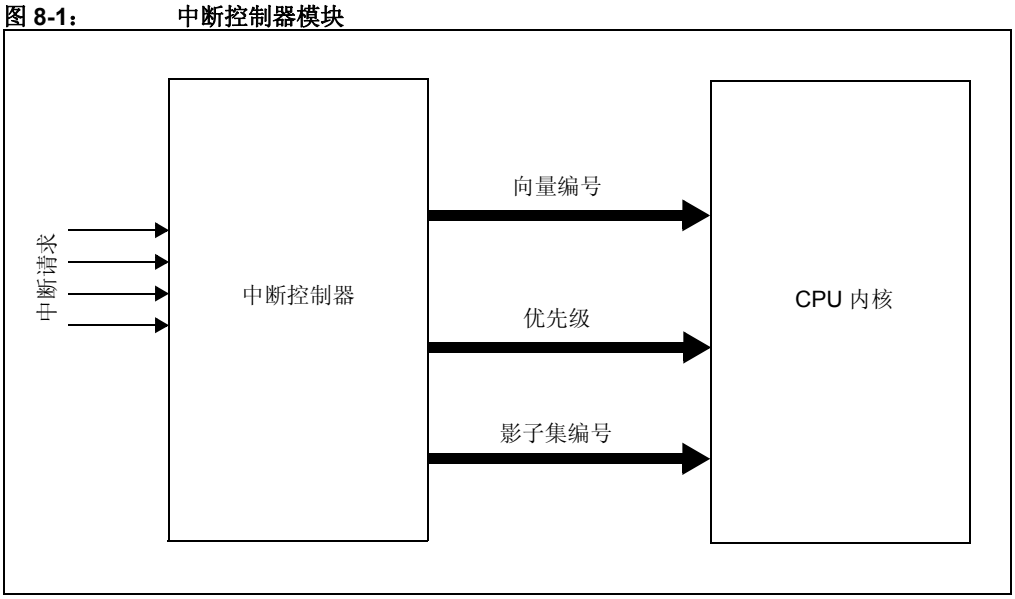
8.1 简介

PIC32MX 器件产生中断请求以响应来自外设模块的中断事件。中断模块处于 CPU 逻辑之外，并且在将中断事件预发送到 CPU 之前为其设置优先级顺序。

PIC32MX 中断模块具有以下特性：

- 最多 96 个中断源
- 最多 64 个中断向量
- 单向量工作模式和多向量工作模式
- 5 个具有边沿极性控制功能的外部中断
- 中断接近定时器
- 调试模式下模块冻结
- 每个向量有 7 个用户可选择的优先级
- 每个优先级内有 4 个用户可选择的子优先级
- 基于优先级的用户可配置影子集 （并非所有器件都提供该功能；要了解器件是否提供该功能，请参见具体器件数据手册）
- 软件可产生任何中断
- 用户可配置的中断向量表存储单元
- 用户可配置的中断向量空间

图 8-1： 中断控制器模块



注： 在本章提到的寄存器中，有几个寄存器不位于中断控制器模块中。这些寄存器 （和位）与 CPU 相关联。更多详细信息，请参见**第 2 章 “MCU”**（DS61113）。

为了避免混淆，对于 CPU 中的寄存器，使用了不同的排字方式。本章以及本手册所有其他章节中的寄存器名称仅使用大写字母表示（使用变量的情况除外）。 CPU 寄存器名称使用大写和小写字母表示。例如，INTSTAT 是中断寄存器；而 IntCtl 是 CPU 寄存器。

8.2 控制寄存器

注： 每个 PIC32MX 器件型号可能具有一个或多个中断源，并且根据器件型号，中断源数量可能不同。在控制 / 状态位和寄存器名称中使用的“x”表示存在多个可以定义这些中断源的寄存器，它们具有相同的功能。更多详细信息，请参见具体器件数据手册。

中断模块包含以下特殊功能寄存器（Special Function Register， SFR）：

- INTCON：中断控制寄存器
- INTSTAT：中断状态寄存器
- TPTMR：时间接近定时器寄存器
- IFSx：中断标志状态寄存器
- IECx：中断允许控制寄存器
- IPCx：中断优先级控制寄存器

表 8-1 简要汇总了相关的中断模块寄存器。该汇总表之后列出了相应的寄存器，并且每个寄存器均附有详细的说明。

表 8-1： 中断寄存器汇总

地址偏移	名称	位范围	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
0x00	INTCON(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	SS0
		15:8	—	FRZ	—	MVEC	—	TPC<2:0>		
		7:0	—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP
0x10	INTSTAT(1,2,3)	31:24	—	—	—	—	—	—	—	—
		23:16	—	—	—	—	—	—	—	—
		15:8	—	—	—	—	—	RIPL<2:0>		
		7:0	—	—	VEC<5:0>					
0x20	TPTMR(1,2,3)	31:24	TPTMR<31:24>							
		23:16	TPTMR<23:16>							
		15:8	TPTMR<15:8>							
		7:0	TPTMR<7:0>							
0x30-0x50	IFSx(1,2,3)	31:24	IFS31	IFS30	IFS29	IFS28	IFS27	IFS26	IFS25	IFS24
		23:16	IFS23	IFS22	IFS21	IFS20	IFS19	IFS18	IFS17	IFS16
		15:8	IFS15	IFS14	IFS13	IFS12	IFS11	IFS10	IFS09	IFS08
		7:0	IFS07	IFS06	IFS05	IFS04	IFS03	IFS02	IFS01	IFS00
0x60-0x80	IECx(1,2,3)	31:24	IEC31	IEC30	IEC29	IEC28	IEC27	IEC26	IEC25	IEC24
		23:16	IEC23	IEC22	IEC21	IEC20	IEC19	IEC18	IEC17	IEC16
		15:8	IEC15	IEC14	IEC13	IEC12	IEC11	IEC10	IEC09	IEC08
		7:0	IEC07	IEC06	IEC05	IEC04	IEC03	IEC02	IEC01	IEC00
0x90-0x180	IPCx(1,2,3)	31:24	—	—	—	IP03<2:0>			IS03<1:0>	
		23:16	—	—	—	IP02<2:0>			IS02<1:0>	
		15:8	—	—	—	IP01<2:0>			IS01<1:0>	
		7:0	—	—	—	IP00<2:0>			IS00<1:0>	

图注： — = 未实现，读为 0。地址偏移值以十六进制显示。

注 1： 该寄存器具有关联的清零寄存器，位于 0x4 字节偏移处。这些清零寄存器的命名方式是在关联寄存器的名称末尾附加 CLR （例如，INTCONCLR）。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2： 该寄存器具有关联的置 1 寄存器，位于 0x8 字节偏移处。这些置 1 寄存器的命名方式是在关联寄存器的名称末尾附加 SET （例如，INTCONSET）。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3： 该寄存器具有关联的取反寄存器，位于 0xC 字节偏移处。这些取反寄存器的命名方式是在关联寄存器的名称末尾附加 INV （例如，INTCONINV）。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

寄存器 8-1： INTCON：中断控制寄存器 (1,2,3)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	R/W-0
—	—	—	—	—	—	—	SS0
bit 23							bit 16

r-x	R/W-0	r-x	R/W-0	r-x	R/W-0	R/W-0	R/W-0
—	FRZ	—	MVEC	—	TPC<2:0>		
bit 15							bit 8

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-17 **保留：**写入 0；忽略读操作

bit 16 **SS0：**单向量影子寄存器集位
1 = 单向量，具有影子寄存器集
0 = 单向量，不具有影子寄存器集

bit 15 **保留：**写入 0；忽略读操作

bit 14 **FRZ：**调试异常模式冻结位
1 = 在 CPU 处于调试异常模式时停止工作
0 = 即使在 CPU 处于调试异常模式时也继续工作

注： FRZ 仅在调试异常模式下可写，在正常模式下强制为 0。

bit 13 **保留：**写入 0；忽略读操作

bit 12 **MVEC：**多向量配置位
1 = 中断控制器配置为多向量模式
0 = 中断控制器配置为单向量模式

bit 11 **保留：**写入 0；忽略读操作

注 1： 该寄存器具有关联的清零寄存器（INTCONCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2： 该寄存器具有关联的置 1 寄存器（INTCONSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3： 该寄存器具有关联的取反寄存器（INTCONINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

寄存器 8-1：　INTCON：中断控制寄存器 (1,2,3) （续）

bit 10-8　**TPC<2:0>：**时间接近 （Temporal Proximity， TP）控制位
111 = 组优先级为 7 或更低的中断启动 TP 定时器
110 = 组优先级为 6 或更低的中断启动 TP 定时器
101 = 组优先级为 5 或更低的中断启动 TP 定时器
100 = 组优先级为 4 或更低的中断启动 TP 定时器
011 = 组优先级为 3 或更低的中断启动 TP 定时器
010 = 组优先级为 2 或更低的中断启动 TP 定时器
001 = 组优先级为 1 的中断启动 TP 定时器
000 = 禁止接近定时器

bit 7-5　**保留：**写入 0 ；忽略读操作

bit 4　**INT4EP：**外部中断 4 边沿极性控制位
1 = 上升沿
0 = 下降沿

bit 3　**INT3EP：**外部中断 3 边沿极性控制位
1 = 上升沿
0 = 下降沿

bit 2　**INT2EP：**外部中断 2 边沿极性控制位
1 = 上升沿
0 = 下降沿

bit 1　**INT1EP：**外部中断 1 边沿极性控制位
1 = 上升沿
0 = 下降沿

bit 0　**INT0EP：**外部中断 0 边沿极性控制位
1 = 上升沿
0 = 下降沿

注　1：该寄存器具有关联的清零寄存器（INTCONCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（INTCONSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（INTCONINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

寄存器 8-2：　INTSTAT：中断状态寄存器 (1,2,3)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	r-x	r-x	r-x	r-x	R-0	R-0	R-0
—	—	—	—	—	RIPL<2:0>		
bit 15							bit 8

r-x	r-x	R-0	R-0	R-0	R-0	R-0	R-0
—	—	VEC<5:0>					
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-11　**保留：**写入 0；忽略读操作

bit 10-8　**RIPL<2:0>：**请求优先级位

000-111 = 送入 CPU 的最新中断的优先级

注：　只有中断控制器配置为单向量模式时，才应使用该值。

bit 7-6　**保留：**写入 0；忽略读操作

bit 5-0　**VEC<5:0>：**中断向量位

00000-11111 = 送入 CPU 的中断向量

注：　只有中断控制器配置为单向量模式时，才应使用该值。

注　1：该寄存器具有关联的清零寄存器（INTSTATCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2：该寄存器具有关联的置 1 寄存器（INTSTATSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3：该寄存器具有关联的取反寄存器（INTSTATINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

寄存器 8-3： TPTMR：时间接近定时器寄存器 (1,2,3)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TPTMR<31:24>							
bit 31							bit 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
TPTMR<23:16>							
bit 23							bit 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
TPTMR<15:8>							
bit 15							bit 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
TPTMR<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-0 **TPTMR<31:0>：**时间接近定时器重载位
在中断事件触发时间接近定时器时，时间接近定时器使用它作为重载值。

注 1： 该寄存器具有关联的清零寄存器（TPTMRCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

2： 该寄存器具有关联的置 1 寄存器（TPTMRSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

3： 该寄存器具有关联的取反寄存器（TPTMRINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

寄存器 8-4：　IFSx：中断标志状态寄存器 (1,2,3,4)

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
bit 31							bit 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
IFS23	IFS22	IFS21	IFS20	IFS19	IFS18	IFS17	IFS16
bit 23							bit 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
IFS15	IFS14	IFS13	IFS12	IFS11	IFS10	IFS09	IFS08
bit 15							bit 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
IFS07	IFS06	IFS05	IFS04	IFS03	IFS02	IFS01	IFS00
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-0　**IFS31-IFS00**：中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

注　1：该寄存器代表 IFSx 寄存器的通用定义。要了解确切的位定义，请参见具体器件数据手册中的**“中断”**章节。

2：这些寄存器具有关联的清零寄存器（IFSxCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

3：这些寄存器具有关联的置 1 寄存器（IFSxSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

4：这些寄存器具有关联的取反寄存器（IFSxINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

寄存器 8-5： IECx：中断允许控制寄存器 (1,2,3,4)

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
bit 31							bit 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
IEC23	IEC22	IEC21	IEC20	IEC19	IEC18	IEC17	IEC16
bit 23							bit 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
IEC15	IEC14	IEC13	IEC12	IEC11	IEC10	IEC09	IEC08
bit 15							bit 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
IEC07	IEC06	IEC05	IEC04	IEC03	IEC02	IEC01	IEC00
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-0 **IEC31-IEC00**：中断允许位
1 = 允许中断
0 = 禁止中断

注 1：该寄存器代表 IECx 寄存器的通用定义。要了解确切的位定义，请参见具体器件数据手册中的**“中断”**章节。

2：这些寄存器具有关联的清零寄存器（IECxCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

3：这些寄存器具有关联的置 1 寄存器（IECxSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

4：这些寄存器具有关联的取反寄存器（IECxINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

寄存器 8-6：　IPCx：中断优先级控制寄存器 (1,2,3,4)

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	IP03<2:0>			IS03<1:0>	
bit 31							bit 24

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	IP02<2:0>			IS02<1:0>	
bit 23							bit 16

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	IP01<2:0>			IS01<1:0>	
bit 15							bit 8

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	IP00<2:0>			IS00<1:0>	
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	P = 可编程位	r = 保留位
U = 未实现位	-n = POR 时的值：（0， 1， x = 未知）		

bit 31-29　**保留：**写入 0；忽略读操作

bit 28-26　**IP03<2:0>：**中断优先级位
111 = 中断优先级为 7
110 = 中断优先级为 6
101 = 中断优先级为 5
100 = 中断优先级为 4
011 = 中断优先级为 3
010 = 中断优先级为 2
001 = 中断优先级为 1
000 = 禁止中断

bit 25-24　**IS03<1:0>：**中断子优先级位
11 = 中断子优先级为 3
10 = 中断子优先级为 2
01 = 中断子优先级为 1
00 = 中断子优先级为 0

bit 23-21　**保留：**写入 0；忽略读操作

注　1：该寄存器代表 IPCx 寄存器的通用定义。要了解确切的位定义，请参见具体器件数据手册中的**“中断”**章节。

2：这些寄存器具有关联的清零寄存器（IPCxCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

3：这些寄存器具有关联的置 1 寄存器（IPCxSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

4：这些寄存器具有关联的取反寄存器（IPCxINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

寄存器 8-6：　IPCx：中断优先级控制寄存器 (1,2,3,4)（续）

bit 20-18	**IP02<2:0>：**中断优先级位 111 = 中断优先级为 7 110 = 中断优先级为 6 101 = 中断优先级为 5 100 = 中断优先级为 4 011 = 中断优先级为 3 010 = 中断优先级为 2 001 = 中断优先级为 1 000 = 禁止中断
bit 17-16	**IS02<1:0>：**中断子优先级位 11 = 中断子优先级为 3 10 = 中断子优先级为 2 01 = 中断子优先级为 1 00 = 中断子优先级为 0
bit 15-13	**保留：**写入 0；忽略读操作
bit 12-10	**IP01<2:0>：**中断优先级位 111 = 中断优先级为 7 110 = 中断优先级为 6 101 = 中断优先级为 5 100 = 中断优先级为 4 011 = 中断优先级为 3 010 = 中断优先级为 2 001 = 中断优先级为 1 000 = 禁止中断
bit 9-8	**IS01<1:0>：**中断子优先级位 11 = 中断子优先级为 3 10 = 中断子优先级为 2 01 = 中断子优先级为 1 00 = 中断子优先级为 0
bit 7-5	**保留：**写入 0；忽略读操作
bit 4-2	**IP00<2:0>：**中断优先级位 111 = 中断优先级为 7 110 = 中断优先级为 6 101 = 中断优先级为 5 100 = 中断优先级为 4 011 = 中断优先级为 3 010 = 中断优先级为 2 001 = 中断优先级为 1 000 = 禁止中断
bit 1-0	**IS00<1:0>：**中断子优先级位 11 = 中断子优先级为 3 10 = 中断子优先级为 2 01 = 中断子优先级为 1 00 = 中断子优先级为 0

注　1：该寄存器代表IPCx寄存器的通用定义。要了解确切的位定义，请参见具体器件数据手册中的**“中断”**章节。

2：这些寄存器具有关联的清零寄存器（IPCxCLR），位于 0x4 字节偏移处。向清零寄存器的任意位写入 1 时，会将关联寄存器中的有效位清零。对清零寄存器的读操作将被忽略。

3：这些寄存器具有关联的置 1 寄存器（IPCxSET），位于 0x8 字节偏移处。向置 1 寄存器的任意位写入 1 时，会将关联寄存器中的有效位置 1。对置 1 寄存器的读操作将被忽略。

4：这些寄存器具有关联的取反寄存器（IPCxINV），位于 0xC 字节偏移处。向取反寄存器的任意位写入 1 时，会将关联寄存器中的有效位取反。对取反寄存器的读操作将被忽略。

8.3 工作原理

中断控制器负责对来自一些片上外设的中断请求 （IRQ）进行预处理，并按相应顺序将它们送入处理器。

图 8-2 给出了 PIC32MX 中的中断处理的图示。中断控制器设计为最多可从处理器内核、能够产生中断的片上外设和 5 个外部输入处接收 96 个 IRQ。所有 IRQ 都在 SYSCLK 下降沿进行采样，并锁存到关联的 IFSx 寄存器中。待处理的 IRQ 由 IFSx 寄存器中的标志位等于 1 来指示。如果中断允许 （IECx）寄存器中的相应位清零，则待处理的 IRQ 不会得到进一步处理。 IECx 位用于对中断标志进行门控。如果允许中断，则所有 IRQ 将编码为 5 位宽的向量编号。5 位向量可产生编号为 0 至 63 的唯一中断向量编号。由于 IRQ 数量多于可用向量编号，所以一些 IRQ 共用公共的向量编号。每个向量编号都会分配一个中断优先级和影子集编号。优先级由关联向量的 IPCx 寄存器设置决定。在多向量模式下，用户可以对接收专用影子寄存器集选择优先级。在单向量模式下，所有中断都可能接收到专用影子集。中断控制器会在所有待处理 IRQ 中选择优先级最高的 IRQ，并将关联的向量编号、优先级和影子集编号送入处理器内核。

处理器内核会在流水线的 “E”和 “M”级之间采样送入的向量信息。如果送入内核的向量的优先级大于 CPU 中断优先级位 IPL （Status<15:10>）指示的当前优先级，则会对中断进行处理；否则，它将保持待处理状态，直到当前优先级小于中断的优先级。在处理中断时，处理器内核会将程序计数器压入 CPU 中的异常程序计数器（Exception Program Counter，EPC）寄存器，并将 CPU 中的异常级别（Exception Level，EXL）位（Status<1>）置 1。EXL 位会禁止进一步的中断，直到应用程序通过清零 EXL 位明确地重新允许为止。下一步，它会跳转到根据送入向量编号计算的向量地址处。

INTSTAT 寄存器包含当前待处理中断的中断向量编号（VEC）位（INTSTAT<5:0>）和请求中断优先级 （Requested Interrupt Priority， RIPL）位 （INTSTAT<10:8>）。这可能会与导致内核离开正常执行顺序的中断不同。

执行 ERET （异常返回）指令之后，处理器会恢复为先前状态。 ERET 会清零 EXL 位、恢复程序计数器，并将当前影子集回复为先前影子集。

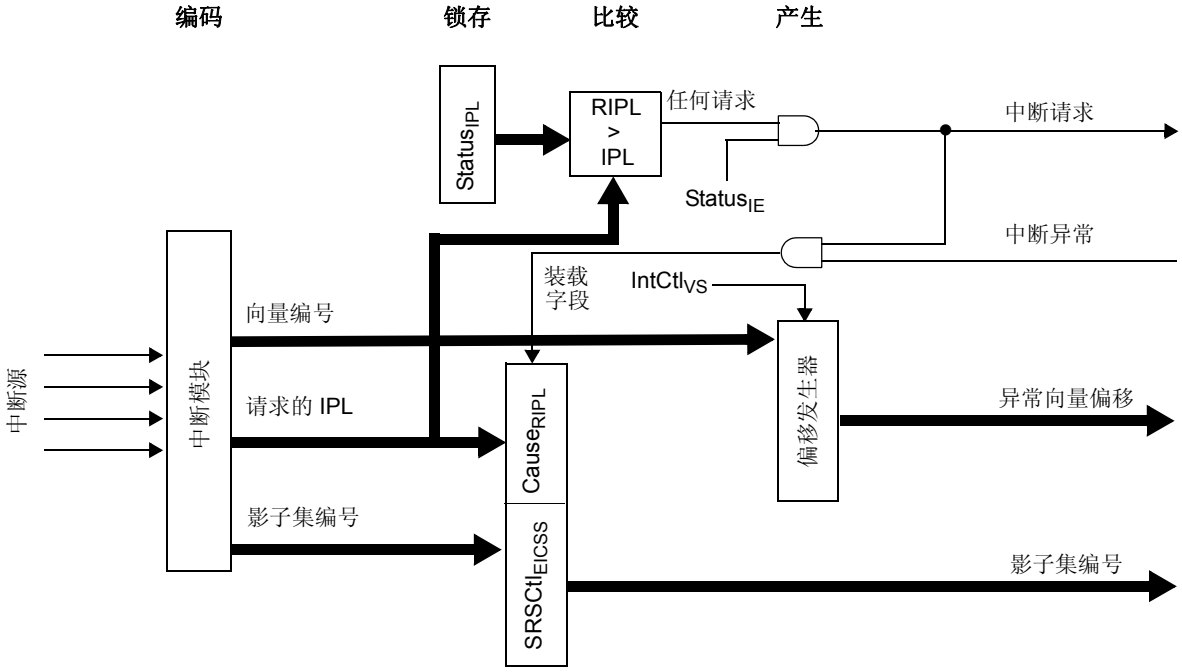
PIC32MX 的中断控制器可以配置为工作于两种模式之一：

- 单向量模式——在一个向量地址处处理所有中断请求 （复位之后的模式）。
- 多向量模式——在所计算的向量地址处处理中断请求。

注： 虽然用户可以在运行时将中断控制器从单向量模式重新配置为多向量模式 （或反之），但强烈建议用户不要如此操作。在初始化之后更改中断控制器模式可能导致未定义的行为。

M4K 内核支持几种不同的中断处理模式。中断控制器设计为可工作于外部中断控制器模式。

图 8-2： 中断处理



注： SRSCtl、Cause、Status 和 IntCtl 寄存器是 CPU 寄存器，在**第 2 章“MCU”**（DS61113）中有说明。

8.4 单向量模式

发生任意形式的复位之后，中断控制器会初始化为单向量模式。当 MVEC（INTCON<12>）位为 0 时，中断控制器工作于单向量模式。在该模式下， CPU 总是转到同一地址。

> **注：** 熟悉 MIPS32 架构的用户必须注意， PIC32MX 中的 M4K 内核仍然工作于外部中断控制器（External Interrupt Controller，EIC）模式。PIC32MX 通过强制所有 IRQ 使用向量编号 0x00 来实现单向量模式。由于 PIC32MX 中的 M4K 内核总是工作于 EIC 模式，所以建议不要通过 MIPS32 架构定义的“中断兼容性模式”产生单向量行为。

要将 CPU 配置为 PIC32MX 单向量模式，必须如下配置以下 CPU 寄存器 （IntCtl、 Cause 和 Status）和 INTCON 寄存器：

- EBase ≠ 00000
- VS （IntCtl<9:5>）≠ 00000
- IV （Cause<23>） = 1
- EXL （Status<1>） = 0
- BEV （Status<22>） = 0
- MVEC （INTCON<12>） = 0
- IE （Status<0>） = 1

例 8-1： 单向量模式的初始化

```
/*
    Set the CP0 registers for multi-vector interrupt
    Place EBASE at 0xBD000000

    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;

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

    temp = mips_getsr();                 // Get Status
    temp |= 0x00400000;                  // Set BEV bit
    mips_setsr(temp);                    // Update Status

    _mips_mtc0(C0_EBASE,   0xBD000000);  // Set an EBase value of 0xBD000000
    _mips_mtc0(C0_INTCTL,  0x00000020);  // Set the Vector Spacing to non-zero value

    temp = mips_getcr();                 // Get Cause
    temp |= 0x00800000;                  // Set IV
    mips_setcr(temp);                    // Update Cause

    temp = mips_getsr();                 // Get Status
    temp &= 0xFFBFFFFD;                  // Clear BEV and EXL
    mips_setsr(temp);                    // Update Status

    INTCONCLR = 0x800;                   // Clear MVEC bit

    asm volatile("ie");                  // Enable all interrupts
```

8.5 多向量模式

当 MVEC （INTCON<12>）位为 1 时，中断控制器工作于多向量模式。在该模式下， CPU 会转到每个向量编号的唯一地址处。每个向量都位于特定偏移处（相对于 CPU 中的 EBase（异常基址）寄存器指定的基址）。各个向量地址偏移由 IntCtl 寄存器中的 VS 位指定的向量空间决定。（IntCtl 寄存器位于 CPU 中；更多详细信息，请参见**第 2 章 “MCU”**（DS61113）。）

要将 CPU 配置为 PIC32MX 多向量模式，必须如下配置以下 CPU 寄存器 （IntCtl、 Cause 和 Status）和 INTCON 寄存器：

- EBase ≠ 00000
- VS （IntCtl<9:5>）≠ 00000
- IV （Cause<23>） = 1
- EXL （Status<1>） = 0
- BEV （Status<22>） = 0
- MVEC （INTCON<12>） = 1
- IE （Status<0>） = 1

例 8-2： 多向量模式的初始化

```
/*
    Set the CP0 registers for multi-vector interrupt
    Place EBASE at 0xBD000000 and Vector Spacing to 32 bytes

    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;

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

    temp = mips_getsr();                  // Get Status
    temp |= 0x00400000;                   // Set BEV bit
    mips_setsr(temp);                     // Update Status

    _mips_mtc0(C0_EBASE,   0xBD000000);   // Set an EBase value of 0xBD000000
    _mips_mtc0(C0_INTCTL,  0x00000020);   // Set the Vector Spacing of 32 bytes
    temp = mips_getcr();                  // Get Cause
    temp |= 0x00800000;                   // Set IV
    mips_setcr(temp);                     // Update Cause

    temp = mips_getsr();                  // Get Status
    temp &= 0xFFBFFFFD;                   // Clear BEV and EXL
    mips_setsr(temp);                     // Update Status

    INTCONSET = 0x800;                    // Set MVEC bit

    asm volatile("ie");                   // Enable all interrupts
```

8.6 中断向量地址计算

特定中断的向量地址取决于中断控制器的配置方式。如果中断控制器配置为单向量模式（见**第 8.4 节 “单向量模式”**），则所有中断向量使用同一向量地址。如果配置为多向量模式（见**第 8.5 节 “多向量模式”**），则每个中断向量具有唯一的向量地址。

发生任何形式的复位之后，处理器会进入引导模式，且控制位 BEV（Status<22>）置 1。（Status 寄存器位于 CPU 中；更多详细信息，请参见**第 2 章 “MCU”**（DS61113）。）当处理器处于引导模式时，所有中断都会被禁止，并且所有一般异常都会被重定向到一个中断向量地址 0xBFC00380 处。将中断控制器配置为所需的工作模式时，必须先将几个寄存器设置为特定值（见**第 8.4 节 “单向量模式”**和**第 8.5 节 “多向量模式”**），然后再清零 BEV 位。

给定中断的向量地址使用异常基址（EBase<31:12>）寄存器进行计算，该寄存器提供一个位于内核段（KSEG）地址空间中的 4 KB 页对齐的基址值。（EBase 是 CPU 寄存器。）

8.6.1 多向量模式地址计算

多向量模式地址通过使用 EBase 和 VS（IntCtl<9:5>）值进行计算。（IntCtl 和 Status 寄存器位于 CPU 中。）VS 位用于提供相邻向量地址之间的间距。允许的向量间距值为 32、64、128、256 和 512 字节。只有 CPU 中的 BEV（Status<22>）位为 1 时，才允许修改 EBase 和 VS 值。例 8-3 说明了如何为给定向量计算多向量地址。

注： 多向量模式地址计算取决于中断向量编号。每个 PIC32MX 器件系列可能具有自己的一组向量编号，具体取决于器件的功能集。要确定与每个中断源关联的向量编号，请参见相应的器件数据手册。

例 8-3： 向量编号 16 的向量地址

```
vector address = vector number X (VS << 5) + 0x200 + vector base.

Exception Base is 0xBD000000
Vector Spacing(VS) is 2, which is 64(0x40)
vector address(T4) = 0x10 X 0x40 + 0x200 + 0xBD000000
vector address(T4) = 0xBD000600
```

8.6.2 单向量模式地址计算

单向量模式地址通过使用异常基址（EBase<31:12>）寄存器值进行计算。在单向量模式下，中断控制器总是送入向量编号 0。单向量模式的准确公式如下：

公式 8-1： 单向量模式地址计算

$$单向量地址 = EBase + 0x200$$

8.7 中断优先级

8.7.1 中断组优先级

用户可以指定每个中断向量的组优先级。组优先级位位于 IPCx 寄存器中。每个 IPCx 寄存器包含 4 个中断向量的组优先级位。用户可选择的优先级范围为从 1（最低优先级）到 7（最高优先级）。如果中断优先级设置为 0，则中断向量会被禁止用于中断和唤醒。优先级较高的中断向量会抢占优先级较低的中断。在重新允许中断之前，用户必须先将 Cause 寄存器的请求中断优先级 （RIPL）位（Cause<15:10>）传送到 Status 寄存器的中断优先级（IPL）位（Status<15:10>）中。（Cause 和 Status 寄存器位于 CPU 中；更多详细信息，请参见**第 2 章 “MCU”**（DS61113）。）该操作将禁止所有低优先级中断，直到中断服务程序 （Interrupt Service Routine， ISR）完成为止。

注：	在降低中断优先级之前，中断服务程序必须先清零 IFSx 寄存器中关联的中断标志，以避免产生递归中断。

例 8-4： 设置组优先级

```
/*
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 中断子优先级

用户可以指定每个组优先级中的子优先级。子优先级不会导致抢占优先级相同的中断；但是，如果有两个优先级相同的中断待处理，则会先处理子优先级最高的中断。子优先级位位于 IPCx 寄存器中。每个 IPCx 寄存器包含 4 个中断向量的子优先级位。这些位用于定义向量优先级中的子优先级。用户可选择的子优先级范围为从 0 （最低子优先级）到 3 （最高子优先级）。

例 8-5： 设置子优先级

```
/*
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 中断自然优先级

当多个中断分配相同的组优先级和子优先级时，它们按照其自然顺序划分优先级。自然优先级是固定的优先级方案，其中，最小的中断向量的自然优先级最高，这意味着中断向量 0 的自然优先级最高，中断向量 63 的自然优先级最低。要了解每个 IRQ 的自然优先级顺序，请参见相应器件数据手册中的中断向量表。

8.8 中断和寄存器集

PIC32MX 系列器件采用了两个寄存器集，主寄存器集用于正常程序执行，影子寄存器集用于最高优先级中断处理。寄存器集选择由中断控制器自动执行。寄存器选择的确切方法因中断控制器的工作模式而异。

在单向量和多向量工作模式下，SRSCtl 寄存器中的 CSS 字段提供当前使用的寄存器集编号，而 PSS 字段提供先前的寄存器集编号。（SRSCtl 是 CPU 寄存器，详情请参见**第 2 章 “MCU”**（DS61113）。）该信息对于确定是否应将堆栈和全局数据指针复制到新寄存器集中非常有用。如果当前寄存器集和先前寄存器集不同，则中断处理程序前言（prologue）代码可能需要将堆栈和全局数据指针从一个寄存器集复制到另一个中去。大多数支持 PIC32MX 系列器件的 C 编译器都会自动生成必需的中断前言代码来处理该操作。

8.8.1 单向量模式下的寄存器集选择

在单向量模式下，SS0（INTCON<16>）位决定将使用哪一个寄存器集。如果 SS0 位为 1，中断控制器将指示 CPU 对于所有中断使用第二个寄存器集。如果 SS0 位为 0，中断控制器将指示 CPU 使用第一个寄存器集。不同于多向量模式，寄存器集和中断优先级之间不存在任何联系。由应用程序决定究竟是否使用第二个影子集。

8.8.2 多向量模式下的寄存器集选择

当中断优先级与影子集优先级匹配时，中断控制器会指示 CPU 使用影子集。对于所有其他中断优先级，中断控制器会指示 CPU 使用主寄存器集。使用影子集的中断优先级不需要执行任何现场信息保存和恢复操作。这可以提高代码吞吐率和降低中断响应延时。

8.9 中断处理

当所请求中断的优先级大于当前 CPU 优先级时，将会接受中断请求，并且 CPU 会跳转到与所请求中断关联的向量地址处。根据中断的优先级，中断处理程序的前言和结语 （epilogue）代码必须在执行任何有用代码之前执行某些特定任务。以下示例给出了建议的前言和结语代码。

8.9.1 单向量模式下的中断处理

当中断控制器配置为单向量模式时，所有中断请求都在同一向量地址处进行处理。中断处理程序必须生成前言和结语代码，以正确配置、保存和恢复所有内核寄存器以及通用寄存器。在最坏情况下，前言和结语代码必须保存和恢复所有可修改的通用寄存器。

8.9.1.1 单向量模式前言代码

在进入中断处理程序时，中断控制器必须先在堆栈中保存分别来自中断优先级 （IPL）位（Status<15:10>）和 ErrorEPC 寄存器的当前优先级和异常 PC 计数器。（Status 和 ErrorEPC 是 CPU 寄存器。）如果处理程序接收到新的寄存器集，则必须将先前寄存器集的堆栈寄存器复制到当前寄存器集的堆栈寄存器中。然后，可以将来自请求中断优先级（RIPL）位（Cause<15:10>）的请求优先级存储到 IPL 中，并清零 Status 寄存器中的异常级别 （EXL）位和错误级别 （ERL）位 （Status<1> 和 Status<2>），将主中断允许位 （Status<0>）置 1。最后将通用寄存器保存在堆栈中。（Cause 和 Status 寄存器位于 CPU 中。）

例 8-6： 单向量中断处理程序前言代码 （汇编代码）

```
rdpgpr  sp, sp
mfc0    k0, Cause
mfc0    k1, EPC
srl     k0, k0, 0xa
addiu   sp, sp, -76
sw      k1, 0(sp)
mfc0    k1, Status
sw      k1, 4(sp)
ins     k1, k0, 10, 6
ins     k1,zero, 1, 4
mtc0    k1, Status
sw      s8, 8(sp)
sw      a0, 12(sp)
sw      a1, 16(sp)
sw      a2, 20(sp)
sw      a3, 24(sp)
sw      v0, 28(sp)
sw      v1, 32(sp)
sw      t0, 36(sp)
sw      t1, 40(sp)
sw      t2, 44(sp)
sw      t3, 48(sp)
sw      t4, 52(sp)
sw      t5, 56(sp)
sw      t6, 60(sp)
sw      t7, 64(sp)
sw      t8, 68(sp)
sw      t9, 72(sp)
addu    s8, sp, zero

// start interrupt handler code here
```

8.9.1.2 单向量模式结语代码

完成中断处理程序的所有有用代码之后，必须将已保存到堆栈中的 Status 和 EPC 寄存器，以及通用寄存器恢复到原始状态。

例 8-7： 单向量中断处理程序结语代码 （汇编代码）

```
// end of interrupt handler code

addu    sp, s8, zero
lw      t9, 72(sp)
lw      t8, 68(sp)
lw      t7, 64(sp)
lw      t6, 60(sp)
lw      t5, 56(sp)
lw      t4, 52(sp)
lw      t3, 48(sp)
lw      t2, 44(sp)
lw      t1, 40(sp)
lw      t0, 36(sp)
lw      v1, 32(sp)
lw      v0, 28(sp)
lw      a3, 24(sp)
lw      a2, 20(sp)
lw      a1, 16(sp)
lw      a0, 12(sp)
lw      s8, 8(sp)
di
lw      k0, 0(sp)
mtc0    k0, EPC
lw      k0, 4(sp)
mtc0    k0, Status
eret
```

8.9.2 多向量模式下的中断处理

当中断控制器配置为多向量模式时，中断请求在所计算的向量地址处进行处理。中断处理程序必须生成前言和结语代码，以正确配置、保存和恢复所有内核寄存器以及通用寄存器。在最坏情况下，前言和结语代码必须保存和恢复所有可修改的通用寄存器。如果中断优先级设置为接收它自己的通用寄存器集，则前言和结语代码不需要保存或恢复任何可修改的通用寄存器，从而产生最短的中断响应延时。

8.9.2.1 多向量模式前言代码

在进入中断处理程序时，中断服务程序必须先在堆栈中保存分别来自中断优先级 （IPL）位（Status<15:10>）和 ErrorEPC 寄存器的当前优先级和异常 PC 计数器。如果处理程序接收到新的寄存器集，则必须将先前寄存器集的堆栈寄存器复制到当前寄存器集的堆栈寄存器中。然后，可以将来自请求中断优先级 （RIPL）位 （Cause<15:10>）的请求优先级存储到 IPL 中，并清零 Status 寄存器中的异常级别 （EXL）位和错误级别 （ERL）位 （Status<1> 和 Status<2>），将主中断允许位 （Status<0>）置 1。如果送入中断处理程序的不是新的通用寄存器集，则这些寄存器将保存到堆栈中。（Cause 和 Status 是 CPU 寄存器；更多详细信息，请参见**第 2 章“MCU”**（DS61113）。）

例 8-8：　　不使用专用通用寄存器集时的前言代码 （汇编代码）

```
rdpgpr  sp, sp
mfc0    k0, Cause
mfc0    k1, EPC
srl     k0, k0, 0xa
addiu   sp, sp, -76
sw      k1, 0(sp)
mfc0    k1, Status
sw      k1, 4(sp)
ins     k1, k0, 10, 6
ins     k1,zero, 1, 4
mtc0    k1, Status
sw      s8, 8(sp)
sw      a0, 12(sp)
sw      a1, 16(sp)
sw      a2, 20(sp)
sw      a3, 24(sp)
sw      v0, 28(sp)
sw      v1, 32(sp)
sw      t0, 36(sp)
sw      t1, 40(sp)
sw      t2, 44(sp)
sw      t3, 48(sp)
sw      t4, 52(sp)
sw      t5, 56(sp)
sw      t6, 60(sp)
sw      t7, 64(sp)
sw      t8, 68(sp)
sw      t9, 72(sp)
addu    s8, sp, zero

    // start interrupt handler code here
```

例 8-9：　　使用专用通用寄存器集时的前言代码 （汇编代码）

```
rdpgpr  sp, sp
mfc0    k0, Cause
mfc0    k1, EPC
srl     k0, k0, 0xa
addiu   sp, sp, -76
sw      k1, 0(sp)
mfc0    k1, Status
sw      k1, 4(sp)
ins     k1, k0, 10, 6
ins     k1,zero, 1, 4
mtc0    k1, Status
addu    s8, sp, zero

    // start interrupt handler code here
```

8.9.2.2 多向量模式结语代码

完成中断处理程序的所有有用代码之后，必须将已保存到堆栈中的 Status 和 ErrorEPC 寄存器，以及通用寄存器恢复到原始状态。（Status 和 ErrorEPC 寄存器位于 CPU 中；更多详细信息，请参见**第 2 章 “MCU”**（DS61113）。）

例 8-10： 不使用专用通用寄存器集时的结语代码 （汇编代码）

```
// end of interrupt handler code

addu    sp, s8, zero
lw      t9, 72(sp)
lw      t8, 68(sp)
lw      t7, 64(sp)
lw      t6, 60(sp)
lw      t5, 56(sp)
lw      t4, 52(sp)
lw      t3, 48(sp)
lw      t2, 44(sp)
lw      t1, 40(sp)
lw      t0, 36(sp)
lw      v1, 32(sp)
lw      v0, 28(sp)
lw      a3, 24(sp)
lw      a2, 20(sp)
lw      a1, 16(sp)
lw      a0, 12(sp)
lw      s8, 8(sp)
di
lw      k0, 0(sp)
mtc0    k0, EPC
lw      k0, 4(sp)
mtc0    k0, Status
eret
```

例 8-11： 使用专用通用寄存器集时的结语代码 （汇编代码）

```
// end of interrupt handler code

addu    sp, s8, zero
di
lw      k0, 0(sp)
mtc0    k0, EPC
lw      k0, 4(sp)
mtc0    k0, Status
eret
```

8.10 外部中断

中断控制器支持 5 个外部中断请求信号 （INT4-INT0）。这些输入是边沿敏感的，它们需要从低至高或从高至低的跳变来产生中断请求。 INTCON 寄存器有 5 个位用于选择边沿检测电路的极性：INT4EP （INTCON<4>）、 INT3EP （INTCON<3>）、 INT2EP （INTCON<2>）、 INT1EP （INTCON<1>）和 INT0EP （INTCON<0>）。

注：	更改外部中断极性可能会触发中断请求。建议用户在更改极性之前先禁止该中断，然后再更改极性、清零中断标志和重新允许中断。

例 8-12： 设置外部中断极性

```
/*
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
```

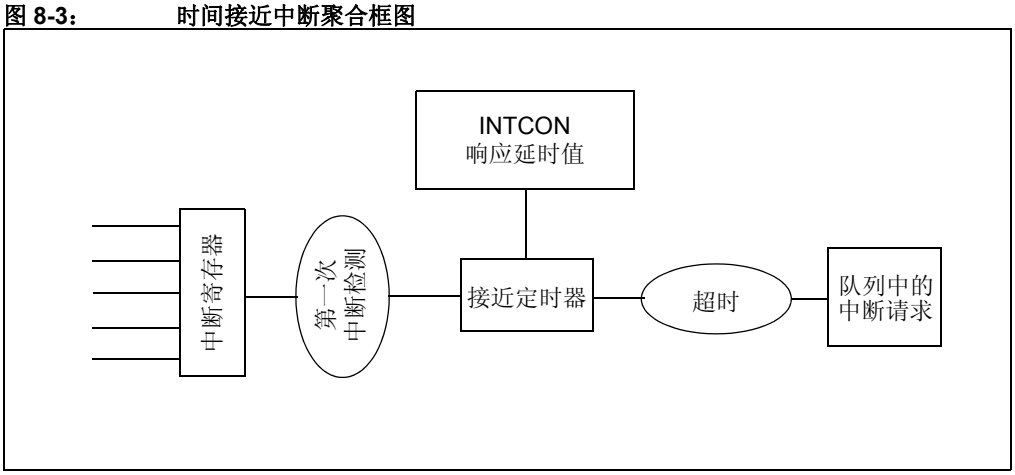
8.11 时间接近中断聚合

发生中断事件时，PIC32MX 的 CPU 会将它们全部作为紧急事件进行响应，因为中断控制器会在出现中断请求时将中断请求送入 CPU。如果当前 CPU 优先级小于待处理中断的优先级，则 CPU 会立即接受中断。进入和退出 ISR 需要一些时钟周期来保存和恢复现场信息。事件对于主程序来说是异步发生的，同时或在相近时间内一起发生的可能性有限。这可以防止发生共用 ISR 一次处理多个中断的情况。

时间接近中断使用中断接近定时器 TPTMR 来产生一个临时窗口，用于推延一组优先级相同或较低的中断。通过这种方式，可以将这些中断请求排入队列，并在单个 ISR 中以尾链技术链接多个 IRQ 的方式来处理。

图 8-3 给出了时间接近中断聚合的框图。触发时间接近定时器的中断组优先级在 TPC 位（INTCON<10:8>）中设置。TPC 用于选择一个组优先级值，如果中断的组优先级值小于等于该值，则会触发时间接近定时器复位并装入 TPTMR 的值。在定时器装入 TPTMR 的值之后，读 TPTMR 将指示定时器的当前状态。当定时器递减至 0 时，如果 IPL（Status<15:10>）小于 RIPL（Cause<15:10>），则会处理队列中的中断请求。

图 8-3： 时间接近中断聚合框图



用户可以通过执行以下步骤来激活时间接近中断聚合：

1. 将 TPC 设置为所需的优先级。（将 TPC 设置为 0 会禁止接近定时器。）
2. 将所需的 32 位值装入 TPTMR。

当中断请求的优先级与 TPC 值匹配 （小于等于 TPC 值）时，将会触发中断接近定时器。

例 8-13： 时间接近中断聚合示例

```
/*
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
TPTMPCLR = 0xFFFFFFFF;     // clear the timer
NTCONSET = 0x00000300;     // set TPC->3
TPTMR = 0x12345678;        // set the timer to 0x12345678
```

8.12 复位之后中断的影响

8.12.1 器件复位

在发生器件复位时，所有中断控制器寄存器会被强制设为它们的复位状态。

8.12.2 上电复位

在发生上电复位时，所有中断控制器寄存器会被强制设为它们的复位状态。

8.12.3 看门狗定时器复位

在发生看门狗定时器复位时，所有中断控制器寄存器会被强制设为它们的复位状态。

8.13 节能和调试模式下的操作

8.13.1 休眠模式下的中断操作

在 Sleep （休眠）模式期间，中断控制器只会接受可工作于 Sleep （休眠）模式的外设的中断。诸如 RTCC、电平变化通知、外部中断、ADC 和 SPI 从器件之类的外设可以在 Sleep（休眠）模式下继续工作，来自这些外设的中断可用于唤醒器件。中断允许位置 1 的中断可以将器件切换为 Run （运行）或 Idle （空闲）模式，具体取决于其中断允许位状态和优先级。中断允许位清零或优先级为 0 的中断事件不会被中断控制器接受，也无法更改器件状态。如果中断请求的优先级大于当前处理器优先级，器件将切换为 Run （运行）模式，并且处理器会执行相应的中断请求。如果使能了接近定时器，并且待处理中断优先级小于时间接近优先级，则处理器不会保持在休眠模式。它会在 TPT 发生超时时切换为空闲模式，然后切换为运行模式。如果中断请求的优先级小于或等于当前处理器优先级，器件会切换为 Idle （空闲）模式，处理器将保持暂停。

8.13.2 空闲模式下的中断操作

在空闲模式期间，中断事件 （相应的中断允许位置 1）可能会将器件切换为 Run （运行）模式，具体取决于其中断允许位状态和优先级。中断允许位清零或优先级为 0 的中断事件不会被中断控制器接受，也无法更改器件状态。如果中断请求的优先级大于当前 CPU 优先级，器件将切换为 Run （运行）模式，并且 CPU 会执行相应的中断请求。如果使能了接近定时器，并且待处理中断的优先级小于时间接近优先级，则器件会保持在 Idle （空闲）模式，只有接近时间结束之后，处理器才会处理中断。如果中断请求的优先级小于或等于当前 CPU 优先级，则器件会保持在 Idle （空闲）模式。相应的中断标志位将保持置 1，中断请求将保持待处理状态。

8.13.3 调试模式下的中断操作

当 CPU 在调试异常模式 （即，应用程序暂停）下执行时，所有中断 （不论它们的优先级如何）都不会被处理，将保持待处理状态。当 CPU 退出调试异常模式时，所有待处理中断将按它们的优先级顺序进行处理。

注：	只有 CPU 在调试异常模式下执行时，FRZ 位才可读写。在所有其他模式下，FRZ 位读为 0。如果 FRZ 位在 Debug （调试）模式期间发生改变，则只有退出当前调试异常模式并重新进入该模式之后，新值才会生效。在调试异常模式期间，在进入 Debug （调试）模式时 FRZ 位会读取外设状态。

8.14 设计技巧

问 1： *是否只要在 IEC 寄存器中允许中断，就可以开始接收中断请求？*

答 1： 不行，要处理任何中断请求，必须先允许内核的系统中断。然后，在 IEC 寄存器中允许中断，并在 IPS 寄存器中分配非零优先级后，才会接收到中断请求。

问 2： *在中断处理程序中应何时清零中断请求标志？*

答 2： 应在处理中断条件之后再清零中断请求标志。例如，如果发生了 UART 接收中断，处理程序应读取接收缓冲区，然后清除 UART 接收 IRQ。

问 3： *在接近定时器倒计数完毕之后，将处理哪一个中断请求？*

答 3： 当接近定时器达到 0 时，将处理优先级最高的中断请求。

8.15 相关应用笔记

本节列出了与手册本章内容相关的应用笔记。这些应用笔记可能并不是专为 PIC32MX 器件系列而编写的，但其概念是相近的，通过适当修改并受到一定限制即可使用。当前与中断模块相关的应用笔记有：

标题	应用笔记编号
目前没有相关的应用笔记。	N/A

注： 如需获取更多 PIC32MX 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

8.16 版本历史

版本 A （2007 年 8 月）

这是本文档的初始版本。

版本 B （2007 年 10 月）

更新了文档 （删除了 “机密” 状态）。

版本 C （2008 年 4 月）

将状态修改为 “初稿”；将 U-0 修改为 r-x。

版本 D （2008 年 6 月）

修改了寄存器 8-1，FRZ 的注释；修改了例 8-1 和 8-2 ；将保留位从“保持为”更改为“写入”。

版本 E （2009 年 7 月）

该版本包括以下更新：

- 对整篇文档的文字和格式进行了少量更新
- 中断寄存器汇总 （表 8-1）：
  - 删除了对清零、置 1 和取反寄存器的所有引用
  - 增加了 “地址偏移” 栏
  - 增加了介绍清零、置 1 和取反寄存器的 “注 1”、“注 2” 和 “注 3”
- 在以下寄存器中增加了对清零、置 1 和取反寄存器的注释：
  - INTCON
  - INTSTAT
  - TPTMR
  - IFSx
  - IPCx
- 更新了**第 8.2 节 “控制寄存器”** 开始处的注释
- 更新了**第 8.3 节 “工作原理”** 中第二段的第二句，以声明 IRQ 中断源
- 更新了**第 8.8.2 节 “多向量模式下的寄存器集选择”** 的第一段
- 更新了**第 8.14 节 “设计技巧”** 中问题 2 的回答

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。

ISBN: 978-1-60932-094-2

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

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-16949:2002 认证。公司在 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方面的质量体系流程均符合 ISO/TS-16949:2002。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。

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://support.microchip.com
网址：www.microchip.com

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

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

克里夫兰 Cleveland
Independence, OH
Tel: 216-447-0464

Fax: 216-447-0643

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

底特律 Detroit
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

科科莫 Kokomo
Kokomo, IN
Tel: 1-765-864-8360
Fax: 1-765-864-8387

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

圣克拉拉 Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

加拿大多伦多 Toronto
Mississauga, Ontario,
Canada
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

亚太总部 Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200

Fax: 852-2401-3431

中国 – 北京
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

中国 – 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 – 重庆
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

中国 – 香港特别行政区
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 – 南京
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

中国 – 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 – 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 – 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 – 深圳
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

中国 – 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 – 西安
Tel: 86-29-8833-7252

Fax: 86-29-8833-7256

中国 – 厦门
Tel: 86-592-238-8138

Fax: 86-592-238-8130

中国 – 珠海
Tel: 86-756-321-0040
Fax: 86-756-321-0049

台湾地区 – 高雄
Tel: 886-7-536-4818

Fax: 886-7-536-4803

台湾地区 – 台北
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

亚太地区

台湾地区 – 新竹
Tel: 886-3-6578-300
Fax: 886-3-6578-370

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

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

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

印度 India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

日本 Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

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

韩国 Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

菲律宾 Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

新加坡 Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

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

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

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

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

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

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

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

英国 UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

01/05/10