

# 第六章 中断与中断控制

6.1 概述

6.2 CPU中断向量与优先级

6.3 可屏蔽中断

6.4 非屏蔽中断

6.5 外设中断扩展模块PIE

6.6 外部中断

6.7 中断控制寄存器

- 中断的重要性：
- 微处理器的灵魂之一
- 学习和掌握的关键内容—重中之重
- 内容包括：
- 中断的总架构
- 中断屏蔽
- 中断标志
- 中断的优先级
- 中断响应
- 中断标志的清除
- 中断使用的经验

## 6.1 概述

中断是指能引起C28x CPU停止当前程序的运行而执行另一个子程序的硬件或软件信号，

中断可由软件触发，例如INTR、OR IFR或TRAP等指令，

也可由硬件触发，如一个引脚、片外设备或片上硬件/外设等。

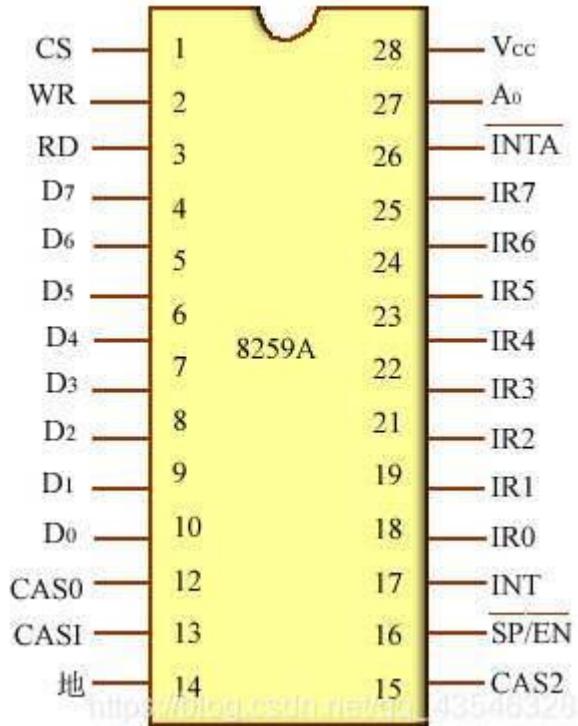
当同一时刻多个中断被触发时，会根据中断优先级来响应它们。

# 中断的架构

- 中断分类：非屏蔽中断（NMI） + 可屏蔽中断
- 内部中断和外设中断
- 中断管理：
- 两级架构：CPU级管理 + 单个中断管理
- 三级架构：CPU级 + 中断组 + 单个中断
- 2802X：改进的三级管理架构
- 趋势：两级架构

## 中断结构（改进的三级架构）：

- 在CPU级，28x CPU支持1个非可屏蔽中断（NMI）和16个可屏蔽的优先中断请求（INT1 – INT14, RTOSINT和DLOGINT）。
- 28x器件有许多外设，每个外设能够产生一个或更多中断来响应外设级的许多事件。
- 由于CPU没有足够的能力在CPU级处理所有的外设中断请求，所以需要一个集中式外设中断扩展（PIE）控制器来判断来自各个源（例如，外设和其它外部引脚）的中断请求。



- 外设中断扩展（PIE）模块将许多中断源复用成一个较小的中断输入集合。
- PIE向量表用来保存系统内每个中断服务程序（ISR）的地址（向量）。
- 每个中断源（包括所有多路复用和非多路复用的中断在内）有一个向量。
- 用户在器件初始化过程中填充向量表，并可以在操作过程中更新它。

## 6.2 CPU中断向量与优先级

C28x DSP支持32个CPU中断向量，其中包含复位向量。

**中断向量**是指一个中断所对应的中断服务程序的入口地址，该地址是22位的，存储在两个相邻存储单元。

低位地址单元存储中断向量低16位，高位地址单元低6位存储中断向量的高6位，其高10位是无效的。

一旦中断被响应，22位的中断矢量被读取。

表6-1 中断向量及优先级

| 中断向量    | 地址        |           | 硬件优先级  | 说明          |
|---------|-----------|-----------|--------|-------------|
|         | VMAP=0    | VMAP=1    |        |             |
| RESET   | 0x00 0000 | 0x3F FFC0 | 1（最高）  | 复位          |
| INT1    | 0x00 0002 | 0x3F FFC2 | 5      | 可屏蔽中断1      |
| INT2    | 0x00 0004 | 0x3F FFC4 | 6      | 可屏蔽中断2      |
| INT3    | 0x00 0006 | 0x3F FFC6 | 7      | 可屏蔽中断3      |
| INT4    | 0x00 0008 | 0x3F FFC8 | 8      | 可屏蔽中断4      |
| INT5    | 0x00 000A | 0x3F FFCA | 9      | 可屏蔽中断5      |
| INT6    | 0x00 000C | 0x3F FFCC | 10     | 可屏蔽中断6      |
| INT7    | 0x00 000E | 0x3F FFCE | 11     | 可屏蔽中断7      |
| INT8    | 0x00 0010 | 0x3F FFD0 | 12     | 可屏蔽中断8      |
| INT9    | 0x00 0012 | 0x3F FFD2 | 13     | 可屏蔽中断9      |
| INT10   | 0x00 0014 | 0x3F FFD4 | 14     | 可屏蔽中断10     |
| INT11   | 0x00 0016 | 0x3F FFD6 | 15     | 可屏蔽中断11     |
| INT12   | 0x00 0018 | 0x3F FFD8 | 16     | 可屏蔽中断12     |
| INT13   | 0x00 001A | 0x3F FFDA | 17     | 可屏蔽中断13     |
| INT14   | 0x00 001C | 0x3F FFDC | 18     | 可屏蔽中断14     |
| DLOGINT | 0x00 001E | 0x3F FFDE | 19（最低） | 可屏蔽数据日志中断   |
| RTOSINT | 0x00 0020 | 0x3F FFE0 | 4      | 可屏蔽实时操作系统中断 |

## 表6-1 中断向量及优先级

| 中断向量     | 地址        |           | 硬件优先级 | 说明        |
|----------|-----------|-----------|-------|-----------|
|          | VMAP=0    | VMAP=1    |       |           |
| Reserved | 0x00 0022 | 0x3F FFE2 | 2     | 保留        |
| NMI      | 0x00 0024 | 0x3F FFE4 | 3     | 非屏蔽中断     |
| ILLEGAL  | 0x00 0026 | 0x3F FFE6 |       | 非法指令TRAP  |
| USER1    | 0x00 0028 | 0x3F FFE8 |       | 用户定义的软件中断 |
| USER2    | 0x00 002A | 0x3F FFEA |       | 用户定义的软件中断 |
| USER3    | 0x00 002C | 0x3F FFEC |       | 用户定义的软件中断 |
| USER4    | 0x00 002E | 0x3F FFEE |       | 用户定义的软件中断 |
| USER5    | 0x00 0030 | 0x3F FFF0 |       | 用户定义的软件中断 |
| USER6    | 0x00 0032 | 0x3F FFF2 |       | 用户定义的软件中断 |
| USER7    | 0x00 0034 | 0x3F FFF4 |       | 用户定义的软件中断 |
| USER8    | 0x00 0036 | 0x3F FFF6 |       | 用户定义的软件中断 |
| USER9    | 0x00 0038 | 0x3F FFF8 |       | 用户定义的软件中断 |
| USER10   | 0x00 003A | 0x3F FFFA |       | 用户定义的软件中断 |
| USER11   | 0x00 003C | 0x3F FFFC |       | 用户定义的软件中断 |
| USER12   | 0x00 003E | 0x3F FFFE |       | 用户定义的软件中断 |

表6-1只给出中断向量表可以映射至程序空间的顶部或底部，这取决于状态寄存器ST1中向量映射位VMAP的值，

但实际上中断向量可映射到存储器四个不同的位置，具有四种模式。

表6-2中断向量映射

| 向量映射   | 向量读取块    | 地址范围              | VMAP | M0M1MAP | ENPIE |
|--------|----------|-------------------|------|---------|-------|
| M1向量   | M1 SARAM | 0x000000~0x00003F | 0    | 0       | x     |
| M0向量   | M0 SARAM | 0x000000~0x00003F | 0    | 1       | x     |
| BROM向量 | 引导ROM    | 0x3FFFC0~0x3FFFFF | 1    | x       | 0     |
| PIE向量  | PIE      | 0x000D00~0x000DFF | 1    | x       | 1     |

**M1和M0向量映射保留，仅供TI测试使用。**

当使用其它向量映射时， M0和M1存储器模块被当做SARAM模块，可以不受限制地自由使用。

## 实际上， 28xx器件只使用PIE向量表映射

- 在复位和引导结束之后，该由用户代码来初始化PIE向量表。
- 然后，应用代码使能PIE向量表。
- 从这时起，中断向量从PIE向量表中提取。
- **注：当复位出现时，复位向量总是从表6-2中提取。**
- 复位后PIE向量表总是被禁能。

## 6.3 可屏蔽中断

INT1~INT14、数据日志中断DLOGINT和实时操作系统中断RTOSINT为16个可屏蔽中断，

INT1~INT14为通用中断， DLOGINT和RTOSINT为2个用于仿真的中断，

这些可屏蔽中断由三个专用的寄存器和状态寄存器ST1控制。

三个专用的寄存器为：

CPU中断标志寄存器**IFR**、

CPU中断允许寄存器**IER**

CPU调试中断允许寄存器**DBGIER**。

**16位的IFR**中的每一位表示其对应的中断是否有中断请求。

**16位的IER**和**DBGIER**中的每一位用来允许或禁止某一个可屏蔽中断。要在**IER**中允许某一个中断，用户可将**IER**中对应的位置1；要在**DBGIER**中允许同一个中断，用户可将**DBGIER**中对应的位置1。

**ST1**的第0位为中断总的屏蔽位**INTM**，它用于允许或禁止所有的中断：

当**INTM=0**时，所有可屏蔽中断**被允许**；

当**INTM=1**时，所有可屏蔽中断**被禁止**。

## 6.4 非屏蔽中断

非屏蔽中断请求**不能被任何使能位所阻止**，C28x的CPU允许这类中断被立即响应并转移到对应的中断服务程序中。

只有一个例外，就是当C28x当前正处于仿真停止状态，此时任何中断请求包括非屏蔽中断请求都不会被CPU响应。

C28x的非屏蔽中断包括四种：

硬件复位中断（RESET）、

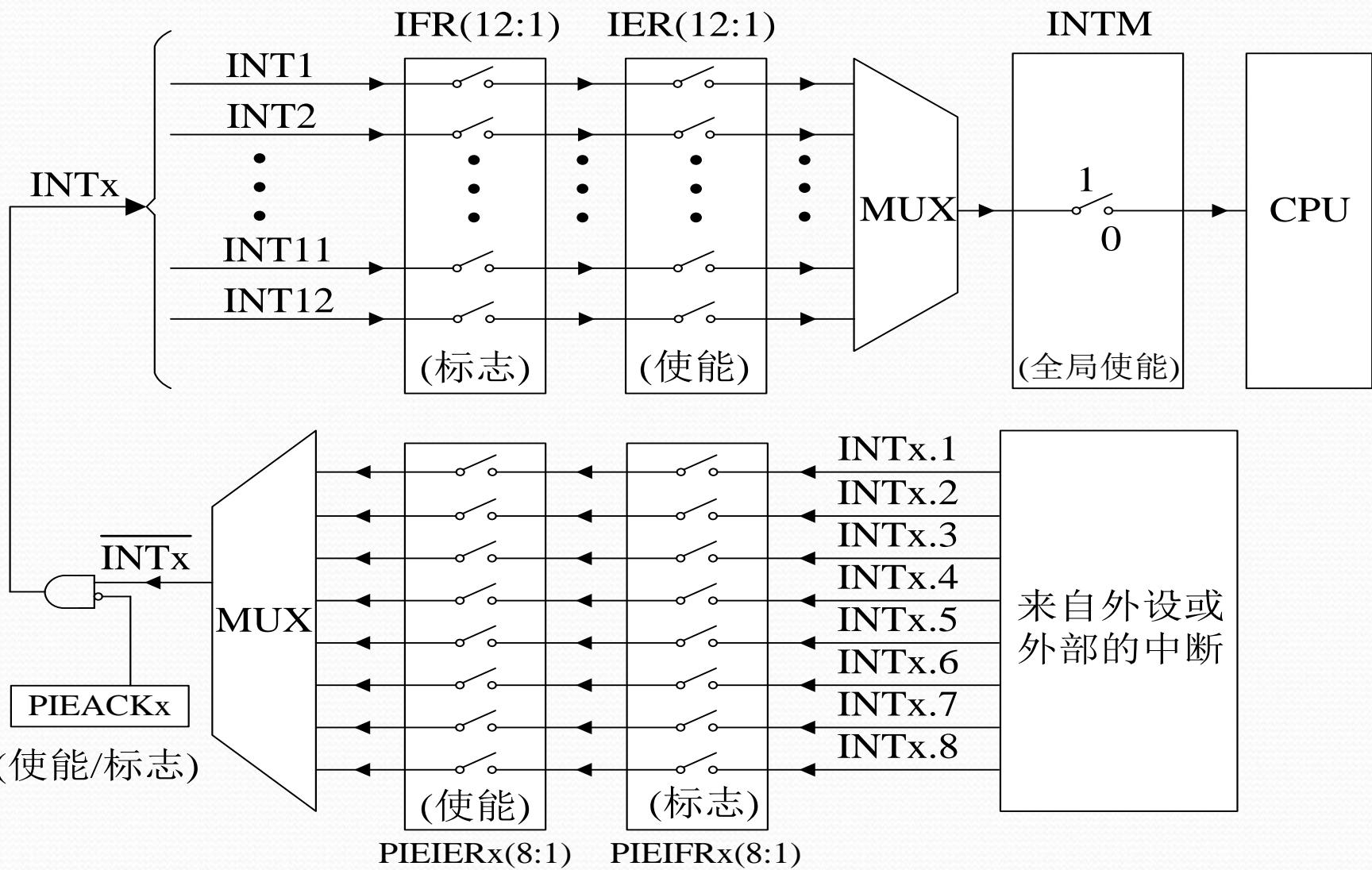
硬件NMI中断、

软件中断（INTR和TRAP指令）

非法指令陷阱。

## 6.5 外设中断扩展PIE模块

- PIE模块可以支持96个单个的中断，这些中断组成8个块。
- 每组中断馈送到12个内核中断线（INT1~INT12）中的一条。
- 96个中断中的每一个都各自被保存在专有RAM模块中的向量支持着，用户可以修改这个RAM模块。
- CPU在服务中断时自动提取相应的中断向量。取出向量并保存重要的CPU寄存器需要花费9个CPU时钟周期。因此，CPU可以立刻响应中断事件。
- 中断的优先顺序由硬件和软件控制。
- 每个单个的中断可以在PIE模块中被使能/禁能。



## 中断发生和响应过程：

外设级 → PIE级 → CPU级

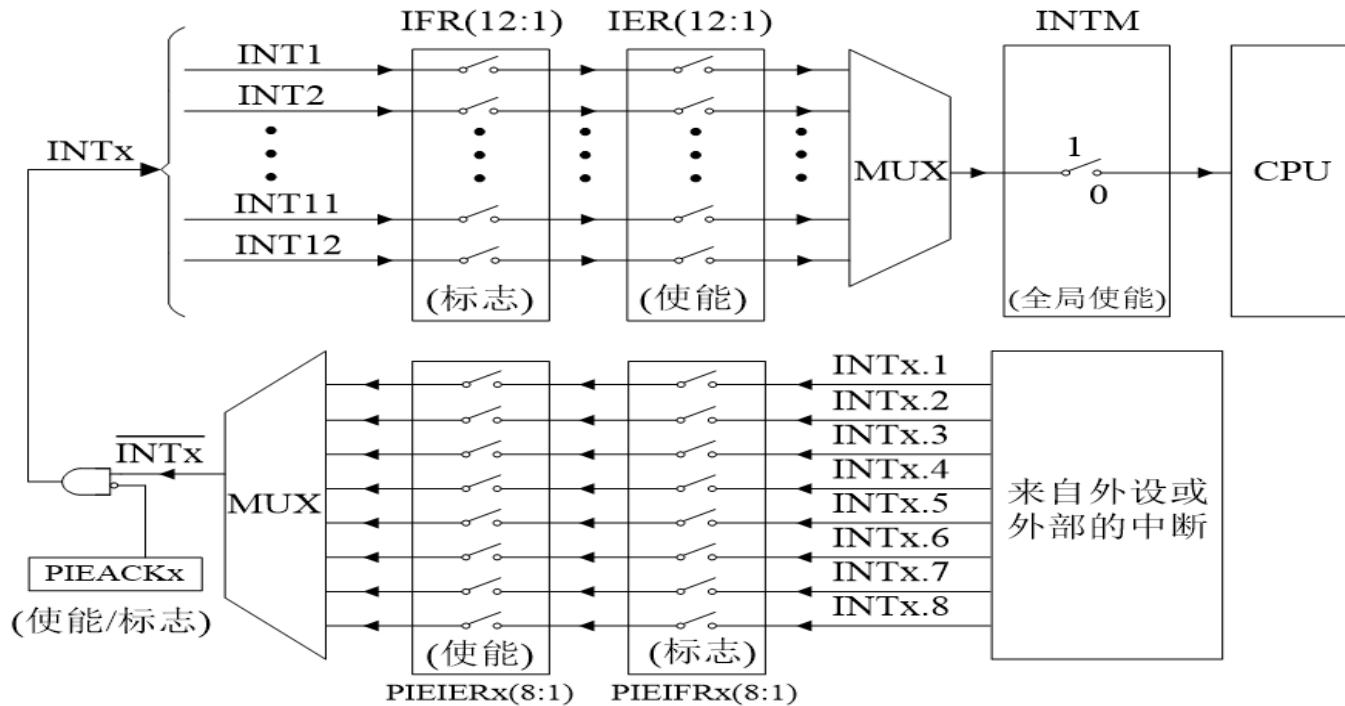
- 外设级：

- 外设中出现一个中断产生的事件。该事件对应的中断标志位（IF）在寄存器中被置位。
- 如果相应的中断使能（IE）位被置位，则外设向PIE控制器产生一个中断请求。如果中断在外设级未被使能，IF就保持置位，直到被软件清除。如果中断稍后被使能，并且中断标志仍然置位，中断请求就提交到PIE。
- 外设寄存器中的中断标志必须手动清除。

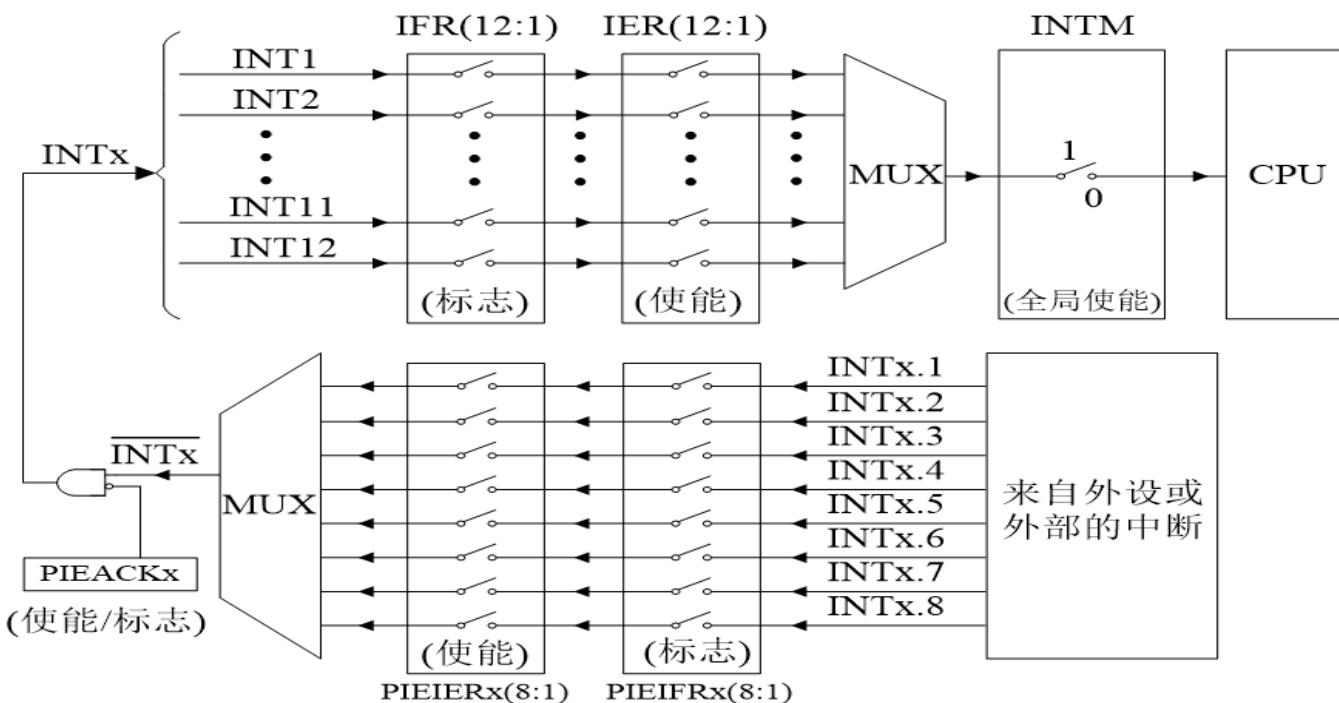
## ● PIE级：

- PIE模块将8个外设和外部引脚中断多路复用成一个CPU中断。
- 这些中断分成12组：PIE组1 – PIE组12。
- 一个组内的中断多路复用成一个CPU中断。例如，**PIE组1被多路复用成CPU中断1（INT1）**，**PIE组12被复用成CPU中断12（INT12）**。
- 连接到剩余CPU中断的中断源不复用。
- 对于非多路复用的中断，**PIE直接将中断请求传递给CPU**。

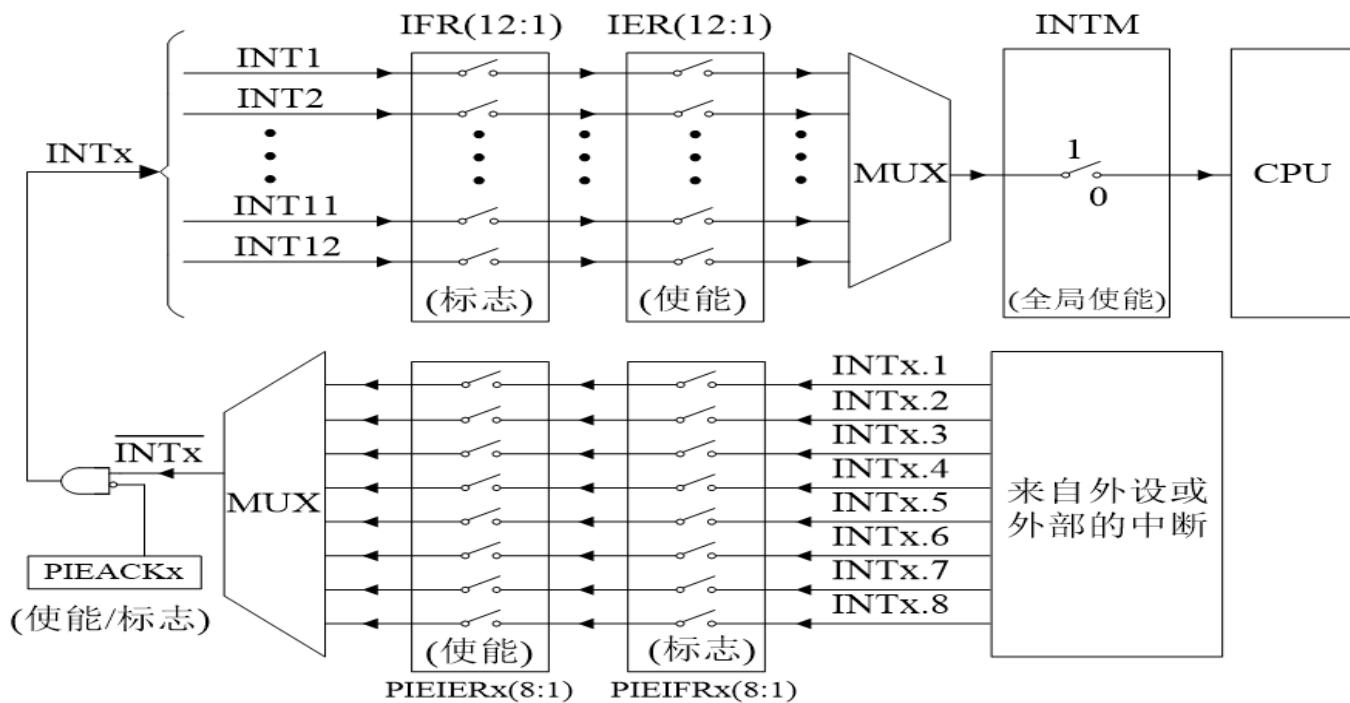
## ● PIE级:



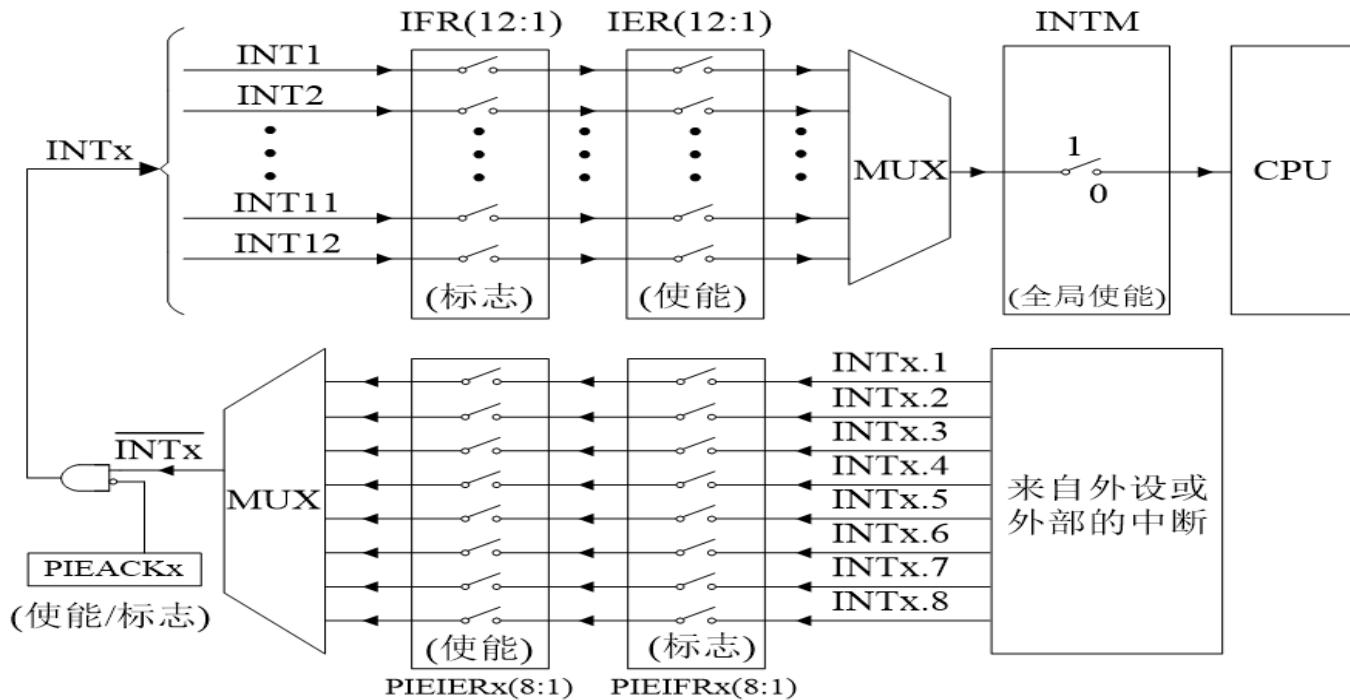
- PIE模块将8个外设和外部引脚中断多路复用成一个CPU中断INTx。
- 这些中断分成12组: PIE组1 – PIE组12。
- 一个组内的中断多路复用成一个CPU中断。例如, PIE组1被多路复用成CPU中断1 (INT1), PIE组12被复用成CPU中断12 (INT12)。
- 连接到剩余CPU中断的中断源不复用。
- 对于非多路复用的中断, PIE直接将中断请求传递给CPU。



- 对于多路复用的中断源，**PIE**模块中的每个中断有一个相关的标志寄存器（**PIEIFRx**）和使能（**PIEIERx**）寄存器（ $x = \text{PIE组1} - \text{PIE组12}$ ）。
- 每个位（表示为y）对应组内8个多路复用中断中的一个。这样，**PIEIFRx.y**和**PIEIERx.y**对应**PIE**组x（ $x = 1-12$ ）的中断y（ $y = 1-8$ ）。
- 另外，每个**PIE**中断组还有一个应答位（**PIEACK**），称为**PIEACKx**（ $x = 1-12$ ）。



- 当外设或外部中断发生，就向**PIE**控制器作出请求。外设级
- **PIE**控制器一旦收到请求，相应**PIE**中断标志（**PIEIFRx.y**）位就被置位。
- 如果**PIE**中断使能（**PIEIERx.y**）位也为了指定的中断而置位，**PIE**就检查对应的**PIEACKx**位来确定**CPU**是否准备好处理来自该组的中断。
- 如果该组的**PIEACKx**位被清除，**PIE**就将中断请求发送给**CPU**。
- 如果**PIEACKx**被置位，**PIE**就等到它被清零才将**INTx**的请求发送出去。



## ● CPU级：

- 一旦请求发送到CPU，对应INTx的CPU级中断标志（**IFR**）就被设置。
- 在标志锁存到**IFR**中以后，并不服务相应的中断，直至**CPU中断使能（IER）寄存器或调试中断使能寄存器（DBGIER）**
- 以及**全局中断屏蔽（INTM）位将中断使能**。

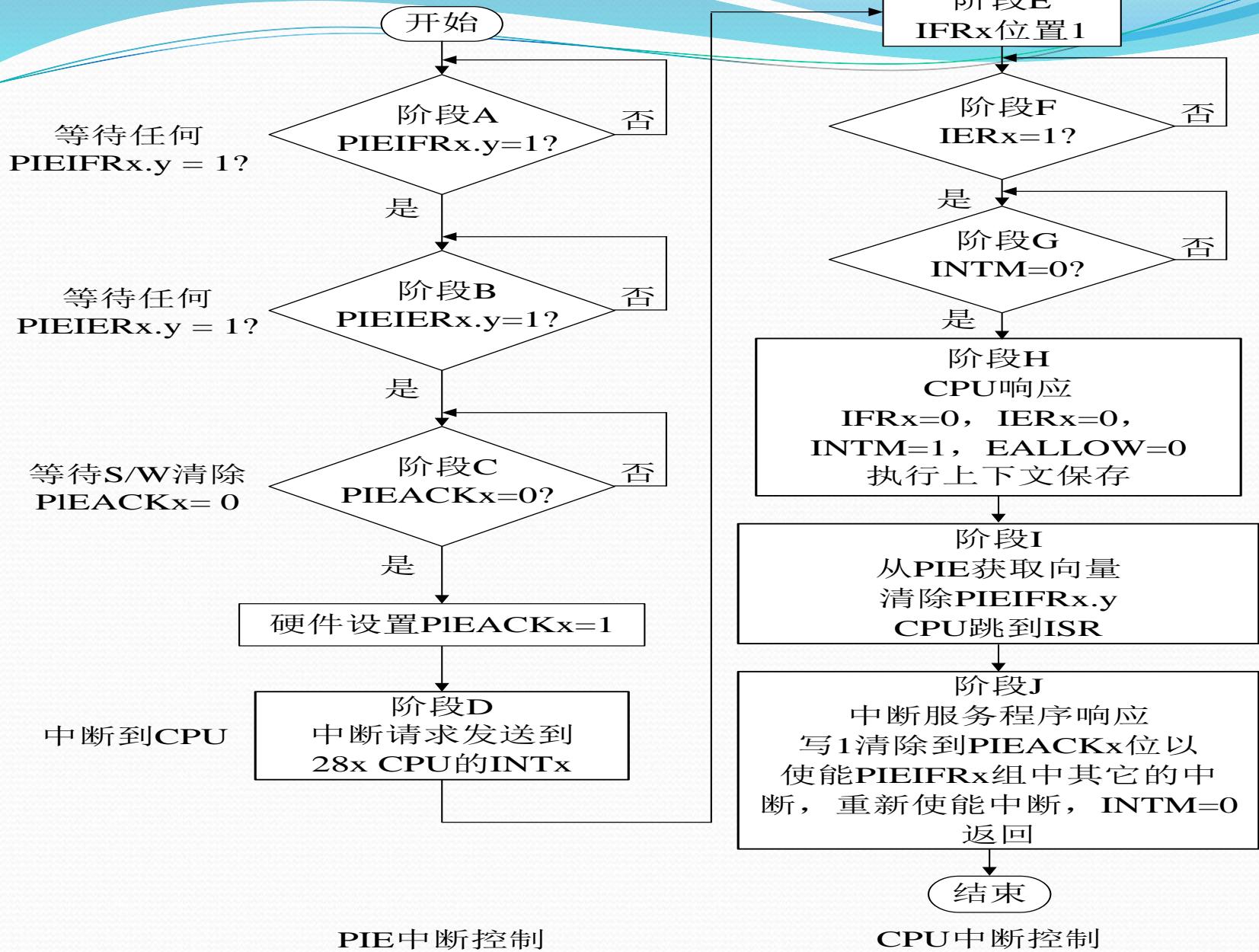


图6.4 外设中断的响应流程

- 使能**CPU**级可屏蔽中断的要求取决于正在使用的中断处理进程。
- 在标准进程中（绝大部分时间都使用该进程），不使用**DBGIER**寄存器。
- 当**28x**处于实时仿真模式且**CPU**被停止时，使用另一个不同的进程。在特殊的情况下，使用**DBGIER**，但忽略**INTM**位。
- 如果**DSP**处于实时模式并且**CPU**正在运行，就运用标准中断处理进程。

| 中断处理进程          | 如果出现以下条件就使能中断                 |
|-----------------|-------------------------------|
| 标准              | $\text{INTM} = 0$ 且IER中相应的位为1 |
| DSP处于实时模式且CPU停止 | IER中的位为1且 <b>DBGIER</b> 为1    |

## CPU准备服务中断：

阶段H  
CPU响应  
IFRx=0, IERx=0,  
INTM=1, EALLOW=0  
执行上下文保存

- 在准备过程中，清除相应的**CPU IFR**和**IER**位，清除**EALLOW**和**LOOP**，设置**INTM**和**DBG**M，清空管道，保存返回地址，并自动执行上下文保护。

阶段I  
从PIE获取向量  
清除PIEIRx.y  
CPU跳到ISR

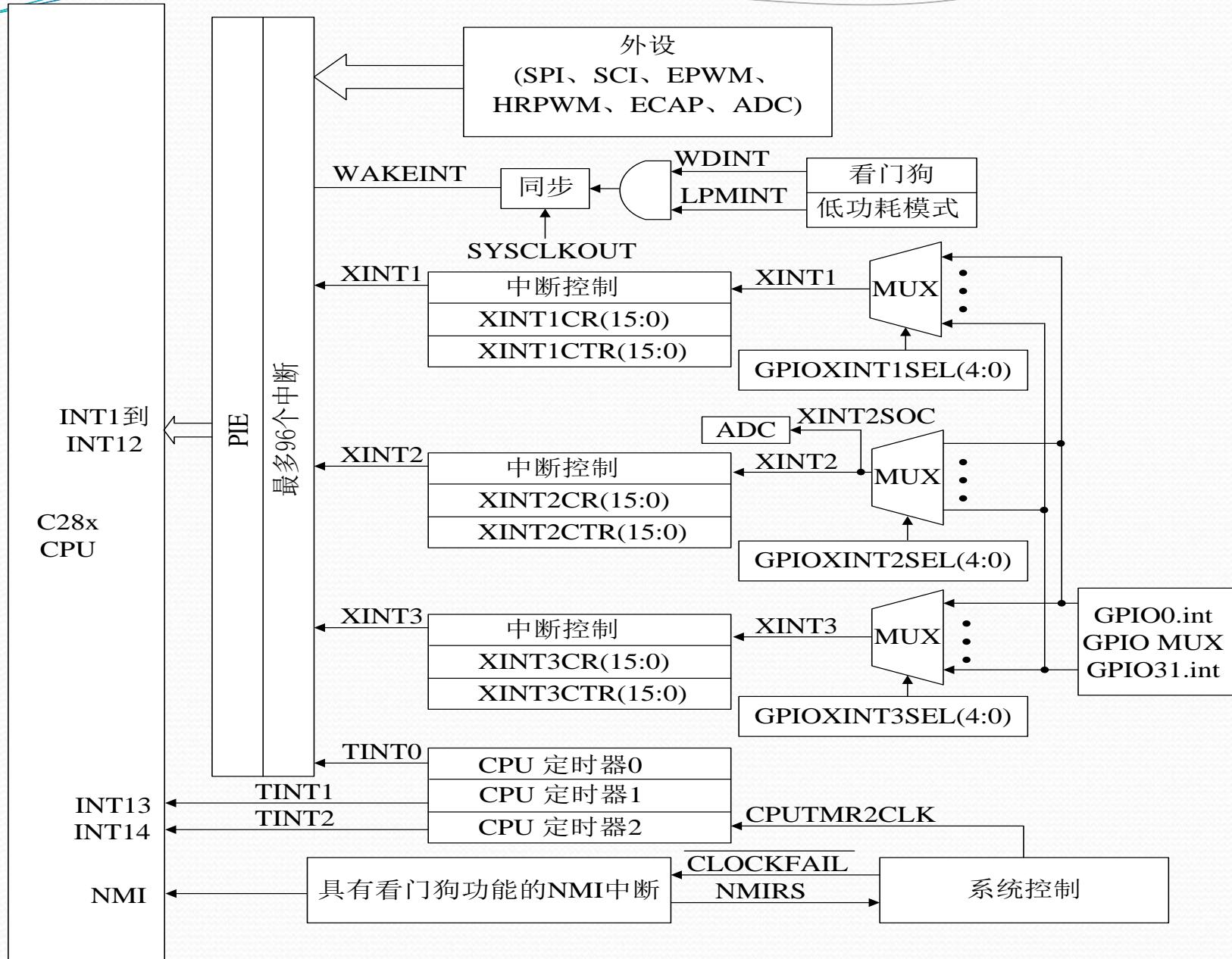
- 然后从**PIE**模块中取出**ISR**的向量。如果中断请求来自一个多路复用的中断，**PIE**模块就使用组**PIEIERx**和**PIEIRx**寄存器来解码需要服务哪个中断。

↓  
阶段I  
从PIE获取向量  
清除PIEIFRx.y  
CPU跳到ISR

- 执行的中断服务程序的地址直接从**PIE**中断向量表中提取。
- **PIE**中96个可能的中断，每一个中断都有一个**32位**的向量。
- 当中断向量被提取时，**PIE**模块内的中断标志（**PIEIFRx.y**）自动被清除。
- 但是，当准备接收来自**PIE**组的更多中断时，必须手动清除给定中断组的**PIE**应答位。

↓  
阶段J  
中断服务程序响应  
写1清除到**PIEACKx**位以  
使能**PIEIFRx**组中其它的中  
断，重新使能中断，**INTM=0**  
返回

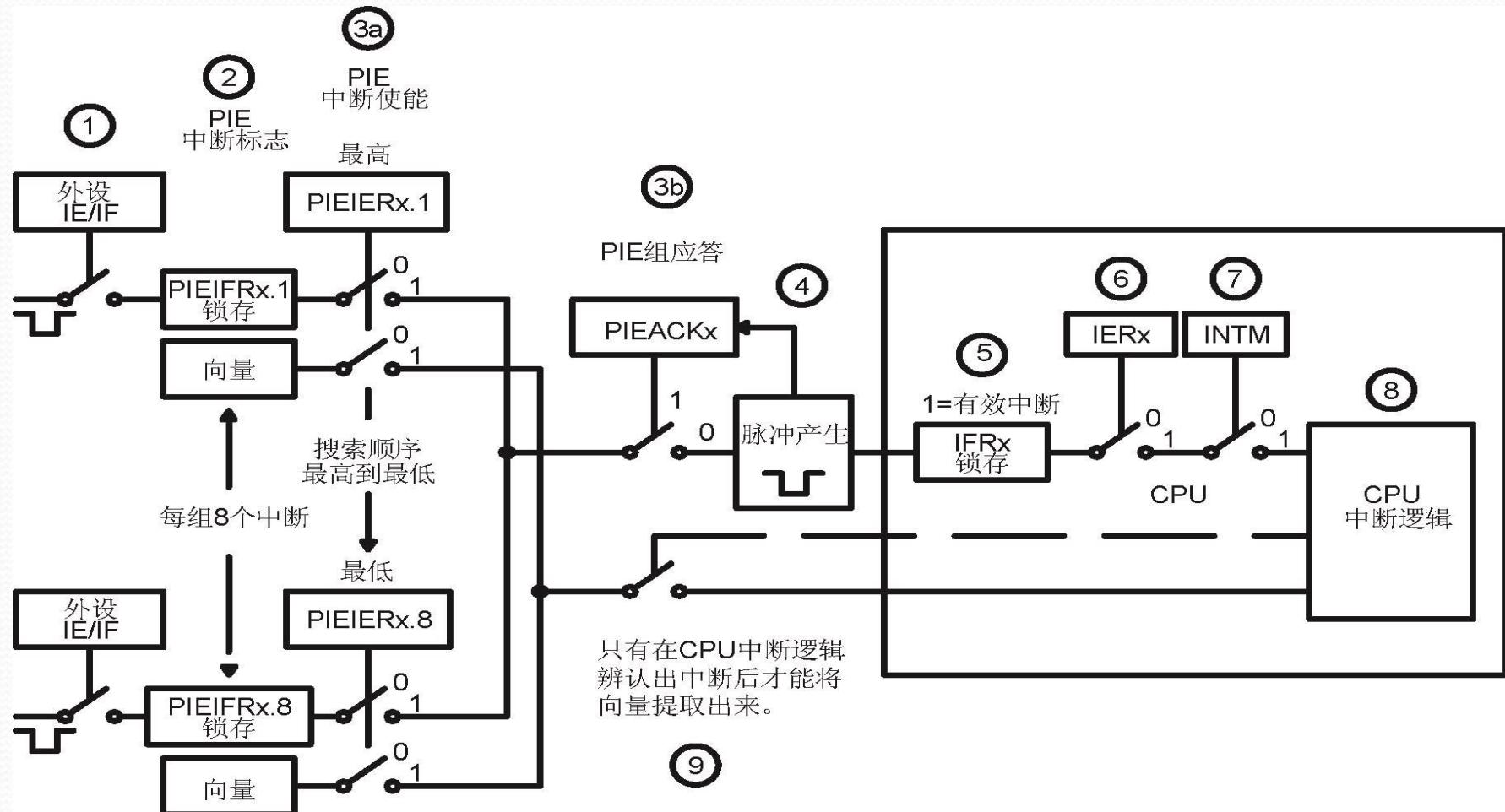
## 6.5.2 F2802x的中断源



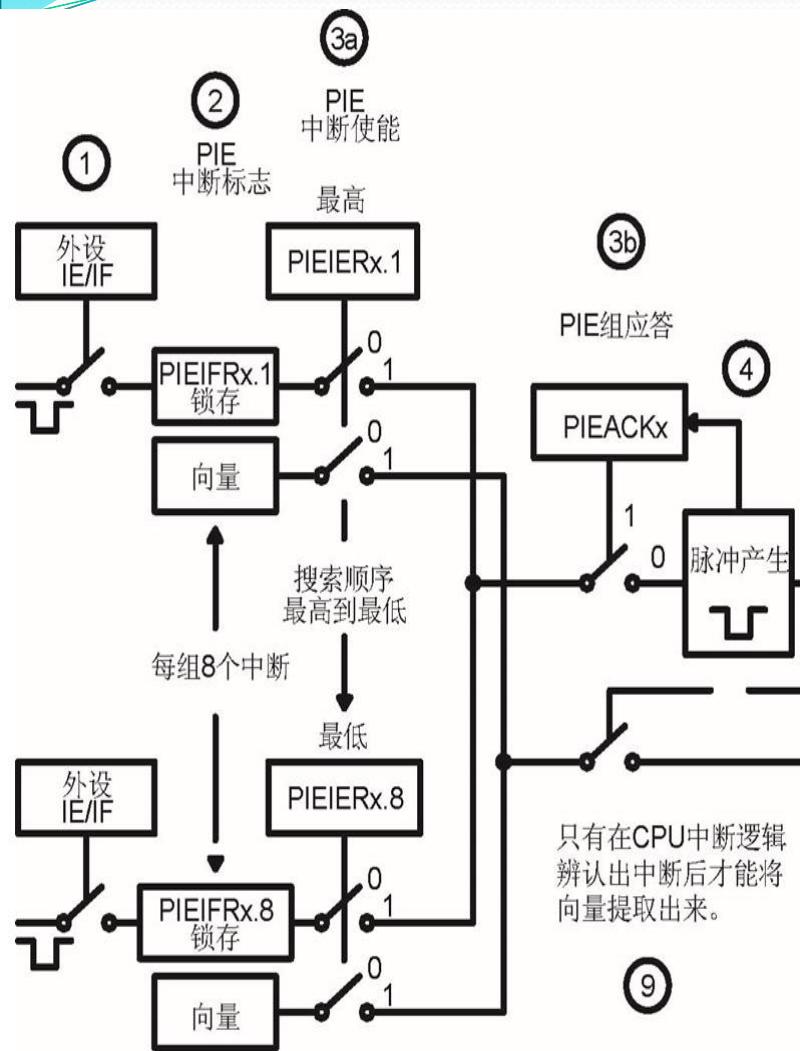
# 处理多路复用中断的方法

- **PIE**模块将8个外设和外部引脚中断多路复用为一个**CPU**中断。这些中断被分成12组：**PIE组1 – PIE组12**。
- 每组有一个相关的使能**PIEIER**寄存器和标志**PIEIFR**寄存器。
- 这两个寄存器用来控制到**CPU**的中断流。
- **PIE**模块也使用**PIEIER**和**PIEIFR**寄存器来解码**CPU**应该跳转到哪个中断服务程序。

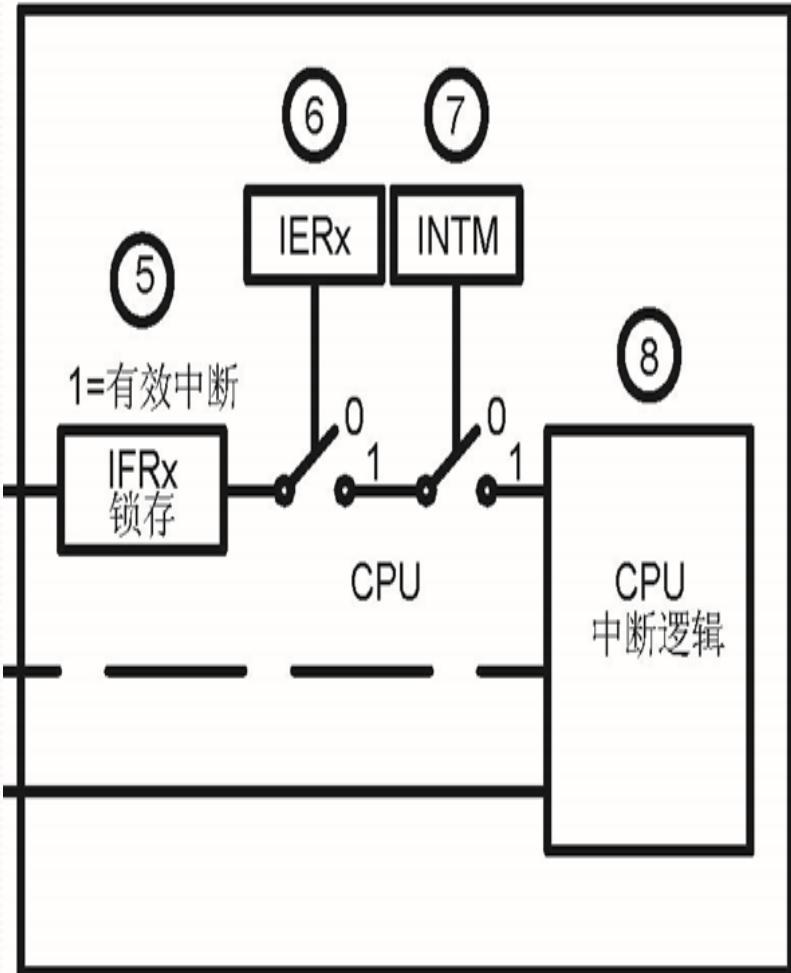
# 从一个外设到CPU的多路复用中断请求的流程



多路复用中断请求流程图



1. **PIE**组内的任何外设或外部中断产生一个中断。如果中断在外设模块中被使能，中断请求就被发送到**PIE**模块。
2. **PIE**模块确认**PIE**组x内的中断 (**INTx.y**) 已经提交一个中断并且相应的**PIE**中断标志位被锁存: **PIEIERx.y = 1**。
3. 中断请求要从**PIE**到**CPU**, 必须满足条件
  - a. 相应的使能位必须置位**PIEIERx.y = 1**
  - b. 组的**PIEACKx**位必须被清除。
4. 如果**3a**和**3b**的条件都满足, 中断请求就被发送到**CPU**, 并且再次置位应答位 **PIEACKx = 1**。**PIEACKx**位将保持置位, 直到你将其清除 (表明组的其它中断可以从**PIE**发送到**CPU**)。



5. 置位**CPU**中断标志位 (**CPU IFR<sub>x</sub> = 1**)，表明**CPU**级有一个挂起的中断<sub>x</sub>。
6. 如果**CPU**中断被使能 (**CPU IER<sub>x</sub> = 1**, 或者**DBGIER<sub>x</sub> = 1**)，并且全局中断屏蔽被清除 (**INTM = 0**)，那么**CPU**将服务**INT<sub>x</sub>**。
7. **CPU**确认中断并执行自动上下文保存，清除**IER**位，置位**INTM**以及清除**EALLOW**。
8. 然后**CPU**将从**PIE**中请求合适的向量。

9. 对于多路复用的中断，**PIE**模块使用**PIEIERx**和**PIEIFRx**寄存器的当前值来译码应该使用哪个向量地址。

有两种可能的情形：

a. 组内最高优先级中断的向量（该中断在**PIEIERx**寄存器中被使能，并且在**PIEIFRx**中标识为挂起）被提取出来用作跳转地址。这样，即使在第7步以后有一个更高优先级的已经使能的中断被标识出来，它也先被服务。

b. 如果组内没有标识出来的中断被使能，**PIE**将用该组内最高优先级中断的向量来响应。该向量用作**INTx.1**的跳转地址。这个操作对应**28x TRAP**或**INT**指令。

## 6.5.3 PIE向量表

- PIE向量表由一个 $256 \times 16$ 的SRAM块组成，如果PIE模块不使用，它也可以用作RAM（只在数据区域内）。复位时未定义PIE向量表的内容。
- CPU决定INT1~INT12的中断优先级。例如，如果INT1. 1和INT8. 1同时出现，PIE模块将两个中断同时提交给CPU，CPU先服务INT1. 1。
- PIE控制每组8个中断的优先级。如果INT1. 1和INT1. 8同时出现，INT1. 1先发送到CPU，然后再发送INT1. 8给CPU。在中断处理的向量提取过程中对中断的优先级进行划分。

- 在96种可能的多路复用中断中有43个中断目前被使用
- 剩余的中断保留供未来的器件使用。
- 这些保留的中断可以用作软件中断，只要它们在PIEIFRx级被使能，前提是组内没有中断正在被外设使用。否则，来自外设的中断可能因为在修改PIEIFR时中断标志被意外地清除而丢失。

在下面两种安全的情况下保留的中断可用作软件中断：

1. 组内没有外设正在提交中断。
2. 没有外设中断被指定到组。例如，PIE组11和12没有任何外设和它们相连。

表6-4 PIE外设中断源

## 表6-5 PIE向量表

| 名称             | 向量ID | 地址          | 大小(x16) | 说明   | CPU优先级         | PIE优先级 |
|----------------|------|-------------|---------|--|----------------|--------|
| <b>Reset</b>   | 0    | 0x0000 0D00 | 2       | Reset总是从0x003F FFC0读取中断矢量                          | 1<br>(Highest) |        |
| <b>INT1</b>    | 1    | 0x0000 0D02 | 2       | 未使用  | 5              |        |
| <b>INT2</b>    | 2    | 0x0000 0D04 | 2       | 未使用  | 6              |        |
| <b>INT3</b>    | 3    | 0x0000 0D06 | 2       | 未使用  | 7              |        |
| <b>INT4</b>    | 4    | 0x0000 0D08 | 2       | 未使用  | 8              |        |
| <b>INT5</b>    | 5    | 0x0000 0D0A | 2       | 未使用  | 9              |        |
| <b>INT6</b>    | 6    | 0x0000 0D0C | 2       | 未使用  | 10             |        |
| <b>INT7</b>    | 7    | 0x0000 0D0E | 2       | 未使用  | 11             |        |
| <b>INT8</b>    | 8    | 0x0000 0D10 | 2       | 未使用  | 12             |        |
| <b>INT9</b>    | 9    | 0x0000 0D12 | 2       | 未使用  | 13             |        |
| <b>INT10</b>   | 10   | 0x0000 0D14 | 2       | 未使用  | 14             |        |
| <b>INT11</b>   | 11   | 0x0000 0D16 | 2       | 未使用  | 15             |        |
| <b>INT12</b>   | 12   | 0x0000 0D18 | 2       | 未使用  | 16             |        |
| <b>INT13</b>   | 13   | 0x0000 0D1A | 2       | External Interrupt 13<br>(XINT13)<br>or CPU-Timer1 | 17             |        |
| <b>INT14</b>   | 14   | 0x0000 0D1C | 2       | CPU-Timer2 (for TI-RTOS use)                       | 18             |        |
| <b>DATALOG</b> | 15   | 0x0000 0D1E | 2       | CPU 数据日志中断   | 19<br>(Lowest) |        |

## 表6-5 PIE向量表

| 名称      | 向量ID | 地址          | 大小(x16) | 说明                      | CPU优先级 | PIE优先级 |
|---------|------|-------------|---------|-------------------------|--------|--------|
| RTOSINT | 16   | 0x0000 0D20 | 2       | CPU RTOS中断              | 4      |        |
| EMUINT  | 17   | 0x0000 0D22 | 2       | CPU Emulation Interrupt | 2      |        |
| NMI     | 18   | 0x0000 0D24 | 2       | 非屏蔽中断NMI                | 3      |        |
| ILLEGAL | 19   | 0x0000 0D26 | 2       | 非法指令TRAP中断              |        |        |
| USER1   | 20   | 0x0000 0D28 | 2       | 用户定义软件中断                |        |        |
| USER2   | 21   | 0x0000 0D2A | 2       | 用户定义软件中断                |        |        |
| USER3   | 22   | 0x0000 0D2C | 2       | 用户定义软件中断                |        |        |
| USER4   | 23   | 0x0000 0D2E | 2       | 用户定义软件中断                |        |        |
| USER5   | 24   | 0x0000 0D30 | 2       | 用户定义软件中断                |        |        |
| USER6   | 25   | 0x0000 0D32 | 2       | 用户定义软件中断                |        |        |
| USER7   | 26   | 0x0000 0D34 | 2       | 用户定义软件中断                |        |        |
| USER8   | 27   | 0x0000 0D36 | 2       | 用户定义软件中断                |        |        |
| USER9   | 28   | 0x0000 0D38 | 2       | 用户定义软件中断                |        |        |
| USER10  | 29   | 0x0000 0D3A | 2       | 用户定义软件中断                |        |        |
| USER11  | 30   | 0x0000 0D3C | 2       | 用户定义软件中断                |        |        |
| USER12  | 31   | 0x0000 0D3E | 2       | 用户定义软件中断                |        |        |

# 表6-5 PIE向量表

| 名称                                       | 向量ID | 地址          | 大小<br>(x16) | 说明          |              | CPU<br>优先级 | PIE<br>优先级 |
|--|------|-------------|-------------|-------------|--------------|------------|------------|
| <b>PIE Group 1 Vector – 合并到 CPU INT1</b> |      |             |             |             |              |            |            |
| INT1.1                                   | 32   | 0x0000 0D40 | 2           | ADCINT1     | (ADC)        | 5          | 1(Highest) |
| INT1.2                                   | 33   | 0x0000 0D42 | 2           | ADCINT2     | (ADC)        | 5          | 2          |
| INT1.3                                   | 34   | 0x0000 0D44 | 2           | Reserved    |              | 5          | 3          |
| INT1.4                                   | 35   | 0x0000 0D46 | 2           | XINT1       |              | 5          | 4          |
| INT1.5                                   | 36   | 0x0000 0D48 | 2           | XINT2       |              | 5          | 5          |
| INT1.6                                   | 37   | 0x0000 0D4A | 2           | ADCINT9     | (ADC)        | 5          | 6          |
| INT1.7                                   | 38   | 0x0000 0D4C | 2           | TINT0       | (CPU-Timer0) | 5          | 7          |
| INT1.8                                   | 39   | 0x0000 0D4E | 2           | WAKEINT     | (LPM/WD)     | 5          | 8(Lowest)  |
| <b>PIE Group 2 Vector – 合并到 CPU INT2</b> |      |             |             |             |              |            |            |
| INT2.1                                   | 40   | 0x0000 0D50 | 2           | EPWM1_TZINT | (EPWM1)      | 6          | 1(Highest) |
| INT2.2                                   | 41   | 0x0000 0D52 | 2           | EPWM2_TZINT | (EPWM2)      | 6          | 2          |
| INT2.3                                   | 42   | 0x0000 0D54 | 2           | EPWM3_TZINT | (EPWM3)      | 6          | 3          |
| INT2.4                                   | 43   | 0x0000 0D56 | 2           | EPWM4_TZINT | (EPWM4)      | 6          | 4          |
| INT2.5                                   | 44   | 0x0000 0D58 | 2           | Reserved    |              | 6          | 5          |
| INT2.6                                   | 45   | 0x0000 0D5A | 2           | Reserved    |              | 6          | 6          |
| INT2.7                                   | 46   | 0x0000 0D5C | 2           | Reserved    |              | 6          | 7          |
| INT2.8                                   | 47   | 0x0000 0D5E | 2           | Reserved    |              | 6          | 8(Lowest)  |

# 表6-5 PIE向量表

| 名称                                       | 向量ID | 地址          | 大小<br>(x16) | 说明        |         | CPU<br>优先级 | PIE<br>优先级 |
|--|------|-------------|-------------|-----------|---------|------------|------------|
| <b>PIE Group 3 Vector – 合并到 CPU INT3</b> |      |             |             |           |         |            |            |
| INT3.1                                   | 48   | 0x0000 0D60 | 2           | EPWM1_INT | (EPWM1) | 7          | 1(Highest) |
| INT3.2                                   | 49   | 0x0000 0D62 | 2           | EPWM2_NT  | (EPWM2) | 7          | 2          |
| INT3.3                                   | 50   | 0x0000 0D64 | 2           | EPWM3_INT | (EPWM3) | 7          | 3          |
| INT3.4                                   | 51   | 0x0000 0D66 | 2           | EPWM4_INT | (EPWM4) | 7          | 4          |
| INT3.5                                   | 52   | 0x0000 0D68 | 2           | Reserved  |         | 7          | 5          |
| INT3.6                                   | 53   | 0x0000 0D6A | 2           | Reserved  |         | 7          | 6          |
| INT3.7                                   | 54   | 0x0000 0D6C | 2           | Reserved  |         | 7          | 7          |
| INT3.8                                   | 55   | 0x0000 0D6E | 2           | Reserved  |         | 7          | 8(Lowest)  |
| <b>PIE Group 4 Vector – 合并到 CPU INT4</b> |      |             |             |           |         |            |            |
| INT4.1                                   | 56   | 0x0000 0D70 | 2           | ECAP1_INT | (ECAP1) | 8          | 1(Highest) |
| INT4.2                                   | 57   | 0x0000 0D72 | 2           | Reserved  |         | 8          | 2          |
| INT4.3                                   | 58   | 0x0000 0D74 | 2           | Reserved  |         | 8          | 3          |
| INT4.4                                   | 59   | 0x0000 0D76 | 2           | Reserved  |         | 8          | 4          |
| INT4.5                                   | 60   | 0x0000 0D78 | 2           | Reserved  |         | 8          | 5          |
| INT4.6                                   | 61   | 0x0000 0D7A | 2           | Reserved  |         | 8          | 6          |
| INT4.7                                   | 62   | 0x0000 0D7C | 2           | Reserved  |         | 8          | 7          |
| INT4.8                                   | 63   | 0x0000 0D7E | 2           | Reserved  |         | 8          | 8(Lowest)  |

# 表6-5 PIE向量表

| 名称                                       | 向量ID | 地址          | 大小<br>(x16) | 说明        |         | CPU<br>优先级 | PIE<br>优先级 |
|--|------|-------------|-------------|-----------|---------|------------|------------|
| <b>PIE Group 5 Vector – 合并到 CPU INT5</b> |      |             |             |           |         |            |            |
| INT5.1                                   | 64   | 0x0000 0D80 | 2           | EQEP1_INT | (EQEP1) | 9          | 1(Highest) |
| INT5.2                                   | 65   | 0x0000 0D82 | 2           | EQEP2_INT | (EQEP2) | 9          | 2          |
| INT5.3                                   | 66   | 0x0000 0D84 | 2           | Reserved  |         | 9          | 3          |
| INT5.4                                   | 67   | 0x0000 0D86 | 2           | Reserved  |         | 9          | 4          |
| INT5.5                                   | 68   | 0x0000 0D88 | 2           | Reserved  |         | 9          | 5          |
| INT5.6                                   | 69   | 0x0000 0D8A | 2           | Reserved  |         | 9          | 6          |
| INT5.7                                   | 70   | 0x0000 0D8C | 2           | Reserved  |         | 9          | 7          |
| INT5.8                                   | 71   | 0x0000 0D8E | 2           | Reserved  |         | 9          | 8(Lowest)  |
| <b>PIE Group 6 Vector – 合并到 CPU INT6</b> |      |             |             |           |         |            |            |
| INT6.1                                   | 72   | 0x0000 0D90 | 2           | SPIRXINTA | (SPI-A) | 10         | 1(Highest) |
| INT6.2                                   | 73   | 0x0000 0D92 | 2           | SPITXINTA | (SPI-A) | 10         | 2          |
| INT6.3                                   | 74   | 0x0000 0D94 | 2           | Reserved  |         | 10         | 3          |
| INT6.4                                   | 75   | 0x0000 0D96 | 2           | Reserved  |         | 10         | 4          |
| INT6.5                                   | 76   | 0x0000 0D98 | 2           | Reserved  |         | 10         | 5          |
| INT6.6                                   | 77   | 0x0000 0D9A | 2           | Reserved  |         | 10         | 6          |
| INT6.7                                   | 78   | 0x0000 0D9C | 2           | Reserved  |         | 10         | 7          |
| INT6.8                                   | 79   | 0x0000 0D9E | 2           | Reserved  |         | 10         | 8(Lowest)  |

# 表6-5 PIE向量表

| 名称                                       | 向量ID | 地址          | 大小(x16) | 说明       |         | CPU优先级 | PIE优先级     |
|--|------|-------------|---------|----------|---------|--------|------------|
| <b>PIE Group 7 Vector – 合并到 CPU INT7</b> |      |             |         |          |         |        |            |
| INT7.1                                   | 80   | 0x0000 0DA0 | 2       | Reserved |         | 11     | 1(Highest) |
| INT7.2                                   | 81   | 0x0000 0DA2 | 2       | Reserved |         | 11     | 2          |
| INT7.3                                   | 82   | 0x0000 0DA4 | 2       | Reserved |         | 11     | 3          |
| INT7.4                                   | 83   | 0x0000 0DA6 | 2       | Reserved |         | 11     | 4          |
| INT7.5                                   | 84   | 0x0000 0DA8 | 2       | Reserved |         | 11     | 5          |
| INT7.6                                   | 85   | 0x0000 0DAA | 2       | Reserved |         | 11     | 6          |
| INT7.7                                   | 86   | 0x0000 0DAC | 2       | Reserved |         | 11     | 7          |
| INT7.8                                   | 87   | 0x0000 0DAE | 2       | Reserved |         | 11     | 8(Lowest)  |
| <b>PIE Group 8 Vector – 合并到 CPU INT8</b> |      |             |         |          |         |        |            |
| INT8.1                                   | 88   | 0x0000 0DB0 | 2       | I2CINT1A | (I2C-A) | 12     | 1(Highest) |
| INT8.2                                   | 89   | 0x0000 0DB2 | 2       | I2CINT2A | (I2C-A) | 12     | 2          |
| INT8.3                                   | 90   | 0x0000 0DB4 | 2       | Reserved |         | 12     | 3          |
| INT8.4                                   | 91   | 0x0000 0DB6 | 2       | Reserved |         | 12     | 4          |
| INT8.5                                   | 92   | 0x0000 0DB8 | 2       | Reserved |         | 12     | 5          |
| INT8.6                                   | 93   | 0x0000 0DBA | 2       | Reserved |         | 12     | 6          |
| INT8.7                                   | 94   | 0x0000 0DBC | 2       | Reserved |         | 12     | 7          |
| INT8.8                                   | 95   | 0x0000 0DBE | 2       | Reserved |         | 12     | 8(Lowest)  |

# 表6-5 PIE向量表

| 名称   | 向量ID | 地址          | 大小<br>(x16) | 说明        |         | CPU<br>优先级 | PIE<br>优先级 |
|--|------|-------------|-------------|-----------|---------|------------|------------|
| <b>PIE Group 9 Vector – 合并到 CPU INT9</b>   |      |             |             |           |         |            |            |
| INT9.1                                     | 96   | 0x0000 0DC0 | 2           | SCIRXINTA | (SCI-A) | 13         | 1(Highest) |
| INT9.2                                     | 97   | 0x0000 0DC2 | 2           | SCITXINTA | (SCI-A) | 13         | 2          |
| INT9.3                                     | 98   | 0x0000 0DC4 | 2           | Reserved  |         | 13         | 3          |
| INT9.4                                     | 99   | 0x0000 0DC6 | 2           | Reserved  |         | 13         | 4          |
| INT9.5                                     | 100  | 0x0000 0DC8 | 2           | Reserved  |         | 13         | 5          |
| INT9.6                                     | 101  | 0x0000 0DCA | 2           | Reserved  |         | 13         | 6          |
| INT9.7                                     | 102  | 0x0000 0DCC | 2           | Reserved  |         | 13         | 7          |
| INT9.8                                     | 103  | 0x0000 0DCE | 2           | Reserved  |         | 13         | 8(Lowest)  |
| <b>PIE Group 10 Vector – 合并到 CPU INT10</b> |      |             |             |           |         |            |            |
| INT10.1                                    | 104  | 0x0000 0DD0 | 2           | ADCINT1   | (ADC)   | 14         | 1(Highest) |
| INT10.2                                    | 105  | 0x0000 0DD2 | 2           | ADCINT2   | (ADC)   | 14         | 2          |
| INT10.3                                    | 106  | 0x0000 0DD4 | 2           | ADCINT3   | (ADC)   | 14         | 3          |
| INT10.4                                    | 107  | 0x0000 0DD6 | 2           | ADCINT4   | (ADC)   | 14         | 4          |
| INT10.5                                    | 108  | 0x0000 0DD8 | 2           | ADCINT5   | (ADC)   | 14         | 5          |
| INT10.6                                    | 109  | 0x0000 0DDA | 2           | ADCINT6   | (ADC)   | 14         | 6          |
| INT10.7                                    | 110  | 0x0000 0DDC | 2           | ADCINT7   | (ADC)   | 14         | 7          |
| INT10.8                                    | 111  | 0x0000 0DDE | 2           | ADCINT8   | (ADC)   | 14         | 8(Lowest)  |

# 表6-5 PIE向量表

| 名称   | 向量ID | 地址          | 大小(x16) | 说明       | CPU优先级 | PIE优先级     |
|--|------|-------------|---------|----------|--------|------------|
| <b>PIE Group 11 Vector – 合并到 CPU INT11</b> |      |             |         |          |        |            |
| INT11.1                                    | 112  | 0x0000 0DE0 | 2       | Reserved | 15     | 1(Highest) |
| INT11.2                                    | 113  | 0x0000 0DE2 | 2       | Reserved | 15     | 2          |
| INT11.3                                    | 114  | 0x0000 0DE4 | 2       | Reserved | 15     | 3          |
| INT11.4                                    | 115  | 0x0000 0DE6 | 2       | Reserved | 15     | 4          |
| INT11.5                                    | 116  | 0x0000 0DE8 | 2       | Reserved | 15     | 5          |
| INT11.6                                    | 117  | 0x0000 0DEA | 2       | Reserved | 15     | 6          |
| INT11.7                                    | 118  | 0x0000 0DEC | 2       | Reserved | 15     | 7          |
| INT11.8                                    | 119  | 0x0000 0DEE | 2       | Reserved | 15     | 8(Lowest)  |
| <b>PIE Group 12 Vector – 合并到 CPU INT12</b> |      |             |         |          |        |            |
| INT12.1                                    | 120  | 0x0000 0DF0 | 2       | XINT3    | 16     | 1(Highest) |
| INT12.2                                    | 121  | 0x0000 0DF2 | 2       | Reserved | 16     | 2          |
| INT12.3                                    | 122  | 0x0000 0DF4 | 2       | Reserved | 16     | 3          |
| INT12.4                                    | 123  | 0x0000 0DF6 | 2       | Reserved | 16     | 4          |
| INT12.5                                    | 124  | 0x0000 0DF8 | 2       | Reserved | 16     | 5          |
| INT12.6                                    | 125  | 0x0000 0DFA | 2       | Reserved | 16     | 6          |
| INT12.7                                    | 126  | 0x0000 0DFC | 2       | Reserved | 16     | 7          |
| INT12.8                                    | 127  | 0x0000 0DFE | 2       | Reserved | 16     | 8(Lowest)  |

## 6.6 外部中断控制寄存器：

1. 支持3个外部中断，**XINT1 - XINT3**。
2. 每个外部中断可以选择负边沿或正边沿触发，也可以被使能或禁能。
3. 屏蔽的中断还包含一个16位自由运行的递增计数器，在检测到一个有效的中断沿时复位为0。这个计数器可以用来精确地计时中断。
4. 非典型的外部中断

## 中断控制和计数器寄存器（不受EALLOW保护）

| 名称       | 地址范围                      | 大小 (x16) | 描述           |
|----------|---------------------------|----------|--------------|
| XINT1CR  | 0x0000 7070               | 1        | XINT1 配置寄存器  |
| XINT2CR  | 0x0000 7071               | 1        | XINT2 配置寄存器  |
| XINT3CR  | 0x0000 7072               | 1        | XINT3 配置寄存器  |
| 保留       | 0x0000 7073 - 0x0000 7077 | 5        |              |
| XINT1CTR | 0x0000 7078               | 1        | XINT1 计数器寄存器 |
| XINT2CTR | 0x0000 7079               | 1        | XINT2 计数器寄存器 |
| XINT3CTR | 0x0000 707A               | 1        | XINT3 计数器寄存器 |
| 保留       | 0x0000 707B - 0x0000 707E | 5        |              |

# 外部中断n控制寄存器 (XINTnCR)

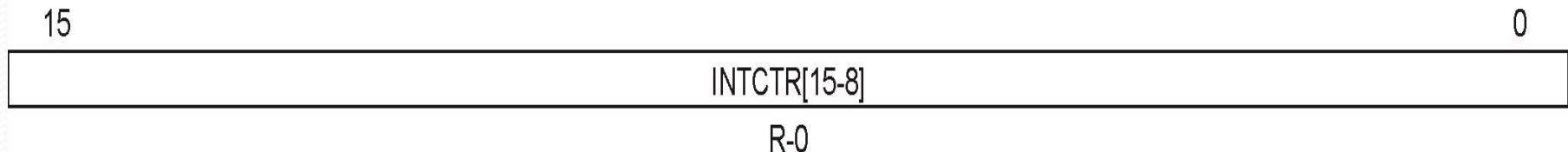
| 15  | 4 | 3        | 2   | 1      | 0 |
|-----|---|----------|-----|--------|---|
| 保留  |   | Polarity | 保留  | Enable |   |
| R-0 |   | R/W-0    | R-0 | R/W-0  |   |

图例: R/W=读/写; R=只读; -n=复位后的值

## 外部中断n控制寄存器 (XINTnCR) 的域描述

| 位    | 域        | 值  | 描述   |
|------|----------|----|--|
| 15-4 | 保留       |    | 读这些位返回 0; 写这些位没有任何影响。                              |
| 3-2  | Polarity | 00 | 这个读/写位确定中断在引脚信号的上升沿还是下降沿产生。<br>中断在下降沿 (高到低的跳变) 产生。 |
|      |          | 01 | 中断在上升沿 (低到高的跳变) 产生。                                |
|      |          | 10 | 中断在下降沿 (高到低的跳变) 产生。                                |
|      |          | 11 | 中断在下降沿以及上升沿 (高到低的跳变以及低到高的跳变) 产生。                   |
| 1    | 保留       |    | 读该位返回 0; 写该位没有任何影响。                                |
| 0    | Enable   | 0  | 这个读/写位使能或禁能外部中断 XINTn。<br>禁能中断。                    |
|      |          | 1  | 使能中断。  |

对于**XINT1/XINT2/XINT3**来说，还有一个**16位的计数器**，只要检测到一个中断边沿，计数器就复位为**0x000**。这些计数器可以用来精确计时中断的出现。



图例: R/W=读/写; R=只读; -n=复位后的值

外部中断n计数器 (XINTnCTR)

## 外部中断n计数器（**XINTnCTR**）的域描述

| 位    | 域      | 描述  |
|------|--------|---|
| 15-0 | INTCTR | 这是一个自由运行的 16 位递增计数器，它以 SYSCLKOUT 的速率被计时。计数器的值在检测到一个有效的中断边沿时复位到 0x0000，然后继续计数，直至检测到下个有效的中断边沿。当中断被禁能时，计数器停止运行。计数器是一个自由运行的计数器，达到最大值时返回到 0。计数器是一个只读寄存器，只有一个有效的中断边沿或复位能将其复位到零。 |

# 6.7 中断控制寄存器

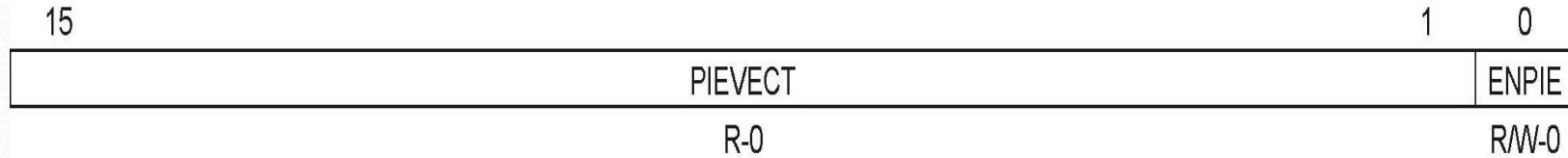
## 6.7.1 PIE配置寄存器

PIE配置寄存器控制着PIE模块的功能

PIE配置和控制寄存器

| 名称      | 地址            | 大小 (x16) | 描述               |
|---------|---------------|----------|------------------|
| PIECTRL | 0x0000 – 0CE0 | 1        | PIE, 控制寄存器       |
| PIEACK  | 0x0000 – 0CE1 | 1        | PIE, 应答寄存器       |
| PIEIER1 | 0x0000 – 0CE2 | 1        | PIE, INT1 组使能寄存器 |
| PIEIFR1 | 0x0000 – 0CE3 | 1        | PIE, INT1 组标志寄存器 |
| PIEIER2 | 0x0000 – 0CE4 | 1        | PIE, INT2 组使能寄存器 |
| PIEIFR2 | 0x0000 – 0CE5 | 1        | PIE, INT2 组标志寄存器 |
| PIEIER3 | 0x0000 – 0CE6 | 1        | PIE, INT3 组使能寄存器 |
| PIEIFR3 | 0x0000 – 0CE7 | 1        | PIE, INT3 组标志寄存器 |
| PIEIER4 | 0x0000 – 0CE8 | 1        | PIE, INT4 组使能寄存器 |
| PIEIFR4 | 0x0000 – 0CE9 | 1        | PIE, INT4 组标志寄存器 |

|          |               |   |                  |
|----------|---------------|---|------------------|
| PIEIER5  | 0x0000 – 0CEA | 1 | PIE, INT5组使能寄存器  |
| PIEIFR5  | 0x0000 – 0CEB | 1 | PIE, INT5组标志寄存器  |
| PIEIER6  | 0x0000 – 0CEC | 1 | PIE, INT6组使能寄存器  |
| PIEIFR6  | 0x0000 – 0CED | 1 | PIE, INT6组标志寄存器  |
| PIEIER7  | 0x0000 – 0CEE | 1 | PIE, INT7组使能寄存器  |
| PIEIFR7  | 0x0000 – 0CEF | 1 | PIE, INT7组标志寄存器  |
| PIEIER8  | 0x0000 – 0CF0 | 1 | PIE, INT8组使能寄存器  |
| PIEIFR8  | 0x0000 – 0CF1 | 1 | PIE, INT8组标志寄存器  |
| PIEIER9  | 0x0000 – 0CF2 | 1 | PIE, INT9组使能寄存器  |
| PIEIFR9  | 0x0000 – 0CF3 | 1 | PIE, INT9组标志寄存器  |
| PIEIER10 | 0x0000 – 0CF4 | 1 | PIE, INT10组使能寄存器 |
| PIEIFR10 | 0x0000 – 0CF5 | 1 | PIE, INT10组标志寄存器 |
| PIEIER11 | 0x0000 – 0CF6 | 1 | PIE, INT11组使能寄存器 |
| PIEIFR11 | 0x0000 – 0CF7 | 1 | PIE, INT11组标志寄存器 |
| PIEIER12 | 0x0000 – 0CF8 | 1 | PIE, INT12组使能寄存器 |
| PIEIFR12 | 0x0000 – 0CF9 | 1 | PIE, INT12组标志寄存器 |



图例: RW=读/写; R=只读; -n=复位后的值

### PIECTRL 寄存器 (地址CEO)

#### PIECTRL寄存器地址域描述

| 位    | 域       | 值          | 描述   |
|------|---------|------------|--|
| 15-1 | PIEVECT |            | <p>这些位指示提取向量的 PIE 向量表中的地址。地址的最低位被忽略，只显示地址的位 1~15。你可以读取向量值来确定哪个中断产生的向量提取。</p> <p>例如：如果 PIECTRL = 0x0D27，地址 0x0D26 处的向量（非法操作）被提取。</p>  |
| 0    | ENPIE   | 0<br><br>1 | <p>使能从 PIE 向量表中提取向量。</p> <p>注：复位向量切勿从 PIE 中提取，即使被使能。它总是从 Boot ROM 中提取。</p> <p>如果这个位被设为 0，PIE 模块就被禁能，向量从 Boot ROM 中的 CPU 向量表中提取。即使 PIE 模块被禁能，所有的 PIE 模块寄存器（PIEACK、PIEIFR、PIEIER）仍能被访问。</p> <p>当 ENPIE 被设为 1 时，除复位向量之外的所有向量都从 PIE 向量表中提取。复位向量总是从 Boot ROM 中提取。</p> |
|      |         |            |  |

| 15  | 12 | 11     | 0 |
|-----|----|--------|---|
| 保留  |    | PIEACK |   |
| R-0 |    | RW1C-1 |   |

图例: RW1C=读/写1来清除; R=只读; -n=复位后的值

### PIE中断应答寄存器 (PIEACK) (地址CE1)

#### PIE中断应答寄存器 (PIEACK) 的域描述

| 位     | 域      | 值                             | 描述   |
|-------|--------|-------------------------------|--|
| 15-12 | 保留     |                               | 保留   |
| 11-0  | PIEACK | 位 x=0 <sup>(1)</sup><br>位 x=1 | PIEACK 中的每个位对应一个特定的 PIE 组。位 0 对应多路复用到 INT1 的 PIE 组 1 的中断, ..., 依此类推, 位 11 对应多路复用到 INT12 的 PIE 组 12 的中断。<br>如果该位读出为 0, 表明 PIE 可以将一个中断从相应的组发送到 CPU。<br>写 0 被忽略。<br>读出 1 表明来自一个组的中断已经发送到 CPU, 所有来自该组的其它中断当前被屏蔽。<br>如果某个组的中断被挂起, 写 1 到相应的中断位将清除该位并使能 PIE 模块驱动一个脉冲到 CPU 中断输入。 |

# PIE中断标志寄存器

有12个PIEIFR寄存器，每个寄存器对应PIE模块使用的一个CPU中断（INT1 – INT12）。

| 15    | 保留    |       |       |       |       |       |       |        | R-0    |        |        |        |        |        |        |
|-------|-------|-------|-------|-------|-------|-------|-------|--------|--------|--------|--------|--------|--------|--------|--------|
| R-0   |       |       |       |       |       |       |       |        |        |        |        |        |        |        |        |
| 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     | INTx.8 | INTx.7 | INTx.6 | INTx.5 | INTx.4 | INTx.3 | INTx.2 | INTx.1 |
| R/W-0 |        |        |        |        |        |        |        |        |

图例：RW=读/写；R=只读；-n=复位后的值

PIEIFRx寄存器（x=1~12）

# PIEIFRx寄存器的域描述

| 位    | 域      | 描述  |
|------|--------|---|
| 15-8 | 保留     | 保留  |
| 7    | INTx.8 | 这些位指示中断当前是否有效。它们的行为非常像 CPU 中断标志寄存器。当一个中断有效时，相应的寄存器位被置位。该位在中断被服务时清除，或者通过写 0 到寄存器位来将其清除。也可以读取这个寄存器来确定哪些中断正有效或正挂起。x=1~12。INTx 指代 CPU INT1~INT12。 |
| 6    | INTx.7 |   |
| 5    | INTx.6 |   |
| 4    | INTx.5 |   |
| 3    | INTx.4 | PIEIFR 寄存器位在中断处理的中断向量提取过程中被清除。  |
| 2    | INTx.3 | 硬件优先于 CPU 对 PIEIER 寄存器进行访问。   |
| 1    | INTx.2 |   |
| 0    | INTx.1 |   |

**注：**

切勿清除一个**PIEIFR**位；  
中断可能在读-修改-写操作过程中被丢失。

# PIE中断使能寄存器

有12个PIEIER寄存器，每个寄存器对应PIE模块使用的一个CPU中断（INT1 – INT12）。

| 保留     |        |        |        |        |        |        |        |
|--------|--------|--------|--------|--------|--------|--------|--------|
| R-0    |        |        |        |        |        |        |        |
| 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
| INTx.8 | INTx.7 | INTx.6 | INTx.5 | INTx.4 | INTx.3 | INTx.2 | INTx.1 |

图例：RW = 读/写； R=只读； -n=复位后的值

PIEIERx寄存器（x=1~12）

## PIEIERx寄存器（x=1~12）的域描述

| 位    | 域      | 描述   |
|------|--------|--|
| 15-8 | 保留     | 保留   |
| 7    | INTx.8 | 这些寄存器位分别使能组内的相应中断，其行为非常像内核中断使能寄存器。                                       |
| 6    | INTx.7 | 将一个位设置为 1 来使能服务相应的中断。将一个位设置为 0 来禁止相应的中断服务。x=1~12。INTx 指代 CPU INT1~INT12。 |
| 5    | INTx.6 |  |
| 4    | INTx.5 |  |
| 3    | INTx.4 |  |
| 2    | INTx.3 |  |
| 1    | INTx.2 |  |
| 0    | INTx.1 |  |

注：在正常操作过程中清除PIEIER位时必须十分小心。

## 6.7.2 CPU中断控制寄存器

### CPU中断标志寄存器（IFR）

- CPU中断标志寄存器（IFR）是一个16位的CPU寄存器，用来识别和清除挂起的中断。
- IFR包含所有CPU级可屏蔽中断（INT1-INT14，DLOGINT 和RTOSINT）的标志位。
- 当PIE被使能时，PIE模块复用INT1-INT12的中断源。
- 当请求一个可屏蔽中断时，相应外设控制寄存器中的标志位被设置为1。如果对应的屏蔽位也为1，中断请求就发送到CPU，设置IFR中的相应标志。这表明中断正在挂起或等待响应。

下面的事件也可以清除**IFR**标志：

- CPU响应中断。
- 28x器件被复位。

**注：**

- 1.要清除一个**CPU IFR**位，你必须向该位写入**0**，而不是**1**。
- 2.当一个可屏蔽中断被响应时，只有**IFR**位自动清除。相应外设控制寄存器中的标志位不清除。如果应用要求清除控制寄存器标志，相应的位必须通过软件来清除。
- 3.当**INTR**指令请求一个中断并且相应的**IFR**位被置位时，**CPU**不自动清除**IFR**位。如果应用要求清除**IFR**位，必须通过软件来清除。
- 4.**IER**和**IFR**是内核级中断固有的寄存器。所有外设在相应的控制/配置寄存器中有自己的中断屏蔽和标志位。注：几个外设中断被分组为一个内核级中断。

|         |         |       |       |       |       |       |       |
|---------|---------|-------|-------|-------|-------|-------|-------|
| 15      | 14      | 13    | 12    | 11    | 10    | 9     | 8     |
| RTOSINT | DLOGINT | INT14 | INT13 | INT12 | INT11 | INT10 | INT9  |
| R/W-0   | R/W-0   | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| 7       | 6       | 5     | 4     | 3     | 2     | 1     | 0     |
| INT8    | INT7    | INT6  | INT5  | INT4  | INT3  | INT2  | INT1  |
| R/W-0   | R/W-0   | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

图例: R/W=读/写; R=只读; -n=复位后的值

### 中断标志寄存器 (IFR) —— CPU寄存器

## 中断使能寄存器（IER和调试中断使能寄存器（DBGIER）

IER是一个16位的CPU寄存器。IER包含所有可屏蔽CPU中断级（INT1-INT14，RTOSINT和DLOGINT）的使能位。IER不包含NMI或XRS；因此，IER对于这两个中断没有影响。

可以读IER来识别使能或禁能的中断级，写IER来使能或禁能中断级。当一个中断被禁能时，它不被响应，不管INTM位的值是什么。当一个中断被使能时，如果相应的IFR位为1且INTM位为0，中断就被响应。

当使用**OR IER**和**AND IER**指令修改**IER**位时，确保它们不修改位15（RTOSINT）的状态，除非存在一个实时操作系统。

- 当一个硬件中断被服务或执行一条**INTR**指令时，相应的**IER**位被自动清除。
- 当**TRAP**指令请求中断时，**IER**位不自动清除。在**TRAP**指令请求中断的情况下，如果需要清除**IER**位，必须由中断服务程序来完成。
- 复位时，所有**IER**位均被清零，禁能所有可屏蔽的CPU级中断。

# 中断使能寄存器 (IER) ——CPU寄存器

|         |         |       |       |       |       |       |       |
|---------|---------|-------|-------|-------|-------|-------|-------|
| 15      | 14      | 13    | 12    | 11    | 10    | 9     | 8     |
| RTOSINT | DLOGINT | INT14 | INT13 | INT12 | INT11 | INT10 | INT9  |
| R/W-0   | R/W-0   | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| 7       | 6       | 5     | 4     | 3     | 2     | 1     | 0     |
| INT8    | INT7    | INT6  | INT5  | INT4  | INT3  | INT2  | INT1  |
| R/W-0   | R/W-0   | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

图例: R/W=读/写; R=只读; -n=复位后的值

# 中断使能寄存器 (IER) ——CPU寄存器

|         |         |       |       |       |       |       |       |
|---------|---------|-------|-------|-------|-------|-------|-------|
| 15      | 14      | 13    | 12    | 11    | 10    | 9     | 8     |
| RTOSINT | DLOGINT | INT14 | INT13 | INT12 | INT11 | INT10 | INT9  |
| R/W-0   | R/W-0   | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| 7       | 6       | 5     | 4     | 3     | 2     | 1     | 0     |
| INT8    | INT7    | INT6  | INT5  | INT4  | INT3  | INT2  | INT1  |
| R/W-0   | R/W-0   | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |

图例: R/W=读/写; R=只读; -n=复位后的值

# v125\DSP2802x\_headers\include

| 名称                        | 修改日期            | 类型              | 大小    |
|---------------------------|-----------------|-----------------|-------|
| DSP2802x_Adc.h            | 2015/4/23 21:17 | C++ Header File | 17 KB |
| DSP2802x_BootVars.h       | 2015/4/23 21:17 | C++ Header File | 2 KB  |
| DSP2802x_Comp.h           | 2015/4/23 21:17 | C++ Header File | 3 KB  |
| DSP2802x_CpuTimers.h      | 2015/4/23 21:17 | C++ Header File | 6 KB  |
| DSP2802x_DevEmu.h         | 2015/4/23 21:17 | C++ Header File | 3 KB  |
| DSP2802x_Device.h         | 2015/4/23 21:17 | C++ Header File | 6 KB  |
| DSP2802x_ECap.h           | 2015/4/23 21:17 | C++ Header File | 6 KB  |
| DSP2802x_EPwm.h           | 2015/4/23 21:17 | C++ Header File | 22 KB |
| DSP2802x_Gpio.h           | 2015/4/23 21:17 | C++ Header File | 12 KB |
| DSP2802x_I2c.h            | 2015/4/23 21:17 | C++ Header File | 7 KB  |
| DSP2802x_Nmlntrupt.h      | 2015/4/23 21:17 | C++ Header File | 4 KB  |
| <u>DSP2802x_PieCtrl.h</u> | 2015/4/23 21:17 | C++ Header File | 6 KB  |
| <u>DSP2802x_PieVect.h</u> | 2015/4/23 21:17 | C++ Header File | 7 KB  |
| DSP2802x_Sci.h            | 2015/4/23 21:17 | C++ Header File | 8 KB  |
| DSP2802x_Spi.h            | 2015/4/23 21:17 | C++ Header File | 7 KB  |
| DSP2802x_SysCtrl.h        | 2015/4/23 21:17 | C++ Header File | 16 KB |
| DSP2802x_Xlntrupt.h       | 2015/4/23 21:17 | C++ Header File | 2 KB  |

# v125\DSP2802x\_common\source

| 名称                                 | 修改日期            | 类型               | 大小    |
|------------------------------------|-----------------|------------------|-------|
| DSP2802x_Adc.c                     | 2015/4/23 21:17 | C Source         | 11 KB |
| DSP2802x_CodeStartBranch.asm       | 2015/4/23 21:17 | Assembler Source | 4 KB  |
| DSP2802x_Comp.c                    | 2015/4/23 21:17 | C Source         | 4 KB  |
| DSP2802x_CpuTimers.c               | 2017/5/1 10:00  | C Source         | 5 KB  |
| DSP2802x_CSMPasswords.asm          | 2015/4/23 21:17 | Assembler Source | 3 KB  |
| DSP2802x_DBGIER.asm                | 2015/4/23 21:17 | Assembler Source | 1 KB  |
| DSP2802x_DefaultIsr.c              | 2017/5/1 10:36  | C Source         | 20 KB |
| DSP2802x_DisInt.asm                | 2015/4/23 21:17 | Assembler Source | 2 KB  |
| DSP2802x_ECap.c                    | 2015/4/23 21:17 | C Source         | 3 KB  |
| DSP2802x_EPwm.c                    | 2015/4/23 21:17 | C Source         | 9 KB  |
| DSP2802x_Gpio.c                    | 2015/4/23 21:17 | C Source         | 3 KB  |
| DSP2802x_I2C.c                     | 2015/4/23 21:17 | C Source         | 4 KB  |
| DSP2802x_MemCopy.c                 | 2015/4/23 21:17 | C Source         | 2 KB  |
| DSP2802x_OscComp.c                 | 2015/4/23 21:17 | C Source         | 5 KB  |
| DSP2802x_PieCtrl.c                 | 2015/4/23 21:17 | C Source         | 3 KB  |
| DSP2802x_PieVect.c                 | 2015/4/23 21:17 | C Source         | 7 KB  |
| DSP2802x_Sci.c                     | 2015/4/23 21:17 | C Source         | 4 KB  |
| DSP2802x_Spi.c                     | 2015/4/23 21:17 | C Source         | 4 KB  |
| DSP2802x_SWPrioritizedDefaultIsr.c | 2015/4/23 21:17 | C Source         | 30 KB |
| DSP2802x_SWPrioritizedPieVect.c    | 2015/4/23 21:17 | C Source         | 9 KB  |
| DSP2802x_SysCtrl.c                 | 2015/4/23 21:17 | C Source         | 16 KB |
| DSP2802x_TempSensorConv.c          | 2015/4/23 21:17 | C Source         | 3 KB  |
| DSP2802x_usDelay.asm               | 2015/4/23 21:17 | Assembler Source | 3 KB  |

v125\DSP2802x\_headers\include\DSP2802x\_PieCtrl.h  
v125\DSP2802x\_common\source\DSP2802x\_PieCtrl.c

```
void InitPieCtrl(void)
{
    // Disable Interrupts at the CPU level:
    DINT;
    // Disable the PIE
    PieCtrlRegs.PIECTRL.bit.ENPIE = 0;
    // Clear all PIEIER registers:
    PieCtrlRegs.PIEIER1.all = 0; ..... PieCtrlRegs.PIEIER12.all = 0;
    // Clear all PIEIFR registers:
    PieCtrlRegs.PIEIFR1.all = 0; ..... PieCtrlRegs.PIEIFR12.all = 0;
}
void EnableInterrupts()
{
    // Enable the PIE
    PieCtrlRegs.PIECTRL.bit.ENPIE = 1;
    // Enables PIE to drive a pulse into the CPU
    PieCtrlRegs.PIEACK.all = 0xFFFF;
    // Enable Interrupts at the CPU level
    EINT;
}
```

v125\DSP2802x\_headers\include\DSP2802x\_PieVect.h  
v125\DSP2802x\_common\source\DSP2802x\_PieVect.c

```
void InitPieVectTable(void)
{
    int16      i;
    Uint32 *Source = (void *) &PieVectTableInit;
    Uint32 *Dest = (void *) &PieVectTable;
    // Do not write over first 3 32-bit locations (these locations are
    // initialized by Boot ROM with boot variables)
    Source = Source + 3;
    Dest = Dest + 3;
    EALLOW;
    for(i=0; i < 125; i++) *Dest++ = *Source++;
    EDIS;

    // Enable the PIE Vector Table
    PieCtrlRegs.PIECTRL.bit.ENPIE = 1;
}
```

## v125\DSP2802x\_common\source\DSP2802x\_Defaultlsr.c

```
#####
// FILE:    DSP2802x_Defaultlsr.c
// TITLE:   DSP2802x Device Default Interrupt Service Routines.
// This file contains shell ISR routines for the 2802x PIE vector table.
// Typically these shell ISR routines can be used to populate the entire PIE
// vector table during device debug. In this manner if an interrupt is taken
// during firmware development, there will always be an ISR to catch it.
//
// As development progresses, these ISR routines can be eliminated and replaced
// with the user's own ISR routines for each interrupt. Since these shell ISRs
// include infinite loops they will typically not be included as-is in the final
// production firmware.
//
#####

// INT1.7
interrupt void TINT0_ISR(void)      // CPU-Timer 0
{
    // Insert ISR Code here

    // To receive more interrupts from this PIE group, acknowledge this interrupt
    // PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;

    // Next two lines for debug only to halt the processor here
    // Remove after inserting ISR Code
    asm ("    ESTOP0");
    for(;;)
}
```

## 6.8 示例程序

### 定时器中断应用示例

```
// DESCRIPTION:  
//  
// This example configures CPU Timer0, 1, & 2 and increments  
// a counter each time the timer asserts an interrupt.  
  
#include "DSP28x_Project.h"    // Device Headerfile and Examples Include File  
  
// Prototype statements for functions found within this file.  
interrupt void cpu_timer0_isr(void);  
interrupt void cpu_timer1_isr(void);  
interrupt void cpu_timer2_isr(void);  
  
void main(void)  
{  
  
// Step 1. Initialize System Control:PLL, WatchDog, enable Peripheral Clocks  
// This example function is found in the DSP2802x_SysCtrl.c file.  
InitSysCtrl();
```

## 6.8 示例程序

```
void InitSysCtrl(void)
{
    DisableDog();
    EALLOW;
    SysCtrlRegs.PCLKCR0.bit.ADCENCLK = 1; //(*Device_cal)();
    SysCtrlRegs.PCLKCR0.bit.ADCENCLK = 0; //EDIS;
    Int0sc1Sel();
    InitP11(DSP28_PLLCR, DSP28_DIVSEL);
    InitPeripheralClocks();
    InitFlash();
}
```

**// Step 2. Initialize GPIO:This example function is found in the DSP2802x\_Gpio.c //**  
**file and illustrates how to set the GPIO to it's default state.**

// InitGpio(); // Skipped for this example

**// Step 3. Clear all interrupts and initialize PIE vector table: Disable CPU interrupts**

DINT;

// Initialize the PIE control registers to their default state.The default state is all PIE  
// interrupts disabled and flags are cleared.

// This function is found in the DSP2802x\_PieCtrl.c file.

InitPieCtrl();

// Disable CPU interrupts and clear all CPU interrupt flags:

IER = 0x0000;

IFR = 0x0000;

// Initialize the PIE vector table with pointers to the shell Interrupt Service Routines

// (ISR).This will populate the entire table, even if the interruptis not used in this  
// example. This is useful for debug purposes.The shell ISR routines are found in

// DSP2802x\_Defaultlsr.c.This function is found in DSP2802x\_PieVect.c.

InitPieVectTable();

```
// Interrupts that are used in this example are re-mapped to
// ISR functions found within this file.
EALLOW; // This is needed to write to EALLOW protected registers
PieVectTable.TINT0 = &cpu_timer0_isr;
PieVectTable.TINT1 = &cpu_timer1_isr;
PieVectTable.TINT2 = &cpu_timer2_isr;
EDIS; // This is needed to disable write to EALLOW protected registers

// Step 4. Initialize the Device Peripheral. This function can be
// found in DSP2802x_CpuTimers.c
InitCpuTimers(); // For this example, only initialize the Cpu Timers

#if (CPU_FRQ_60MHZ)
// Configure CPU-Timer 0, 1, and 2 to interrupt every second:
// 60MHz CPU Freq, 1 second Period (in uSeconds)

ConfigCpuTimer(&CpuTimer0, 60, 1000000);
ConfigCpuTimer(&CpuTimer1, 60, 1000000);
ConfigCpuTimer(&CpuTimer2, 60, 1000000);
#endif
```

```
#if (CPU_FRQ_50MHZ)
// Configure CPU-Timer 0, 1, and 2 to interrupt every second:
// 50MHz CPU Freq, 1 second Period (in uSeconds)
    ConfigCpuTimer(&CpuTimer0, 50, 1000000);
    ConfigCpuTimer(&CpuTimer1, 50, 1000000);
    ConfigCpuTimer(&CpuTimer2, 50, 1000000);
#endif
#ifndef (CPU_FRQ_40MHZ)
// Configure CPU-Timer 0, 1, and 2 to interrupt every second:
// 40MHz CPU Freq, 1 second Period (in uSeconds)
    ConfigCpuTimer(&CpuTimer0, 40, 1000000);
    ConfigCpuTimer(&CpuTimer1, 40, 1000000);
    ConfigCpuTimer(&CpuTimer2, 40, 1000000);
#endif
// To ensure precise timing, use write-only instructions to write to the entire register.
// Therefore, if any of the configuration bits are changed in ConfigCpuTimer and
// InitCpuTimers (in DSP2802x_CpuTimers.h), the below settings must also be
// updated.
    CpuTimer0Regs.TCR.all = 0x4001; // Use write-only instruction to set TSS bit = 0
    CpuTimer1Regs.TCR.all = 0x4001; // Use write-only instruction to set TSS bit = 0
    CpuTimer2Regs.TCR.all = 0x4001; // Use write-only instruction to set TSS bit = 0
```

**// Step 5. User specific code, enable interrupts:**

```
// Enable CPU int1 which is connected to CPU-Timer 0, CPU int13  
// which is connected to CPU-Timer 1, and CPU int 14, which is connected  
// to CPU-Timer 2:
```

```
IER |= M_INT1;  
IER |= M_INT13;  
IER |= M_INT14;
```

```
// Enable TINT0 in the PIE: Group 1 interrupt 7
```

```
PieCtrlRegs.PIEIER1.bit.INTx7 = 1;
```

```
// Enable global Interrupts and higher priority real-time debug events:
```

```
EINT; // Enable Global interrupt INTM  
ERTM; // Enable Global realtime interrupt DBGM
```

**// Step 6. IDLE loop. Just sit and loop forever (optional):**

```
for(;;)
```

```
}
```

```
interrupt void cpu_timer0_isr(void)
{
    CpuTimer0.InterruptCount++;

    // Acknowledge this interrupt to receive more interrupts from group 1
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
}

interrupt void cpu_timer1_isr(void)
{
    CpuTimer1.InterruptCount++;
    // The CPU acknowledges the interrupt.
    EDIS;
}

interrupt void cpu_timer2_isr(void)
{
    EALLOW;
    CpuTimer2.InterruptCount++;
    // The CPU acknowledges the interrupt.
    EDIS;
}
```

# 总结：DSP程序框架

1. DSP程序包括主程序和中断服务程序两部分；

2. 主程序主要包括初始化和主循环：

    初始化：时钟初始化，FLASH初始化，GPIO初始化

        中断初始化一：CPU级和PIE级IE/IF清零

            PIE向量初始化

            用到中断向量的写入

            外设或者功能的初始化，

        中断初始化二：PIE和CPU的IE使能

            INTM = 0

    主循环：主要完成实时性要求不高的功能模块的处理

        包括：逻辑、通信、人机交互等

# 总结：DSP程序框架

1. 中断服务程序主要完成实时性要求比较高的功能模块；
2. 中断服务程序：
  - 现场保护（通过函数属性interrupt申明来体现）
  - 功能模块
  - 现场恢复：**PIE**中断的**ACK**处理
  - 其他由函数属性实现