中断与异常2

程晨闻
东南大学电气工程学院

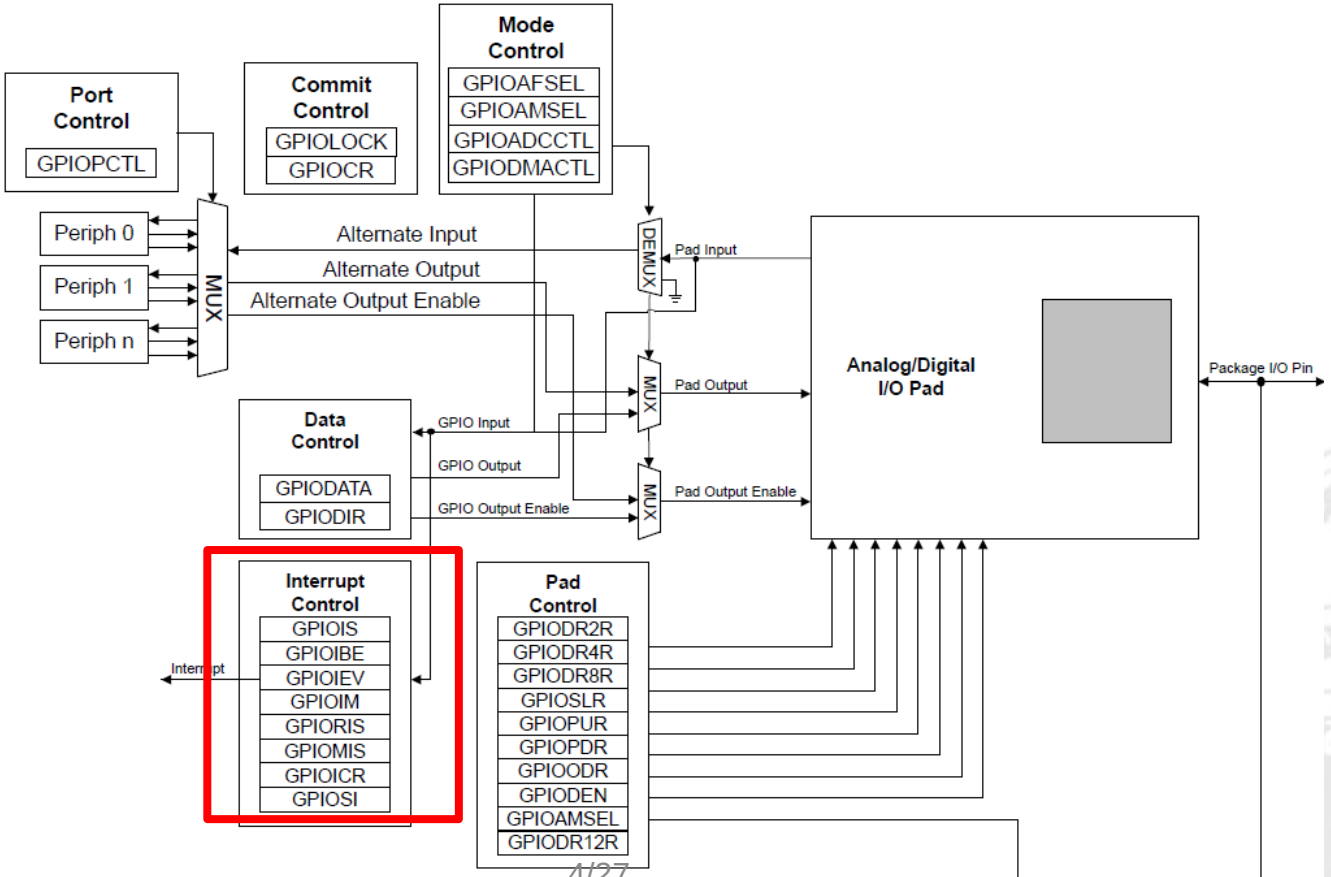
- **中断和异常基本概念**
  - 内部异常（1~15），外设中断（>16）
  - 中断向量表
- **中断的响应流程**
  - 外设级，NVIC级，CPU级
  - 嵌套，尾链，延迟到达
- **中断控制寄存器的访问**
  - IntMasterEnable() (PRIMASK,CPU级)
  - IntEnable() (ENn, NVIC级)
  - IntDisable() (DISn, NVIC级)

- GPIO的中断源管理
- 设置GPIO中断和操作

GPIO的中断源管理

外设		NVIC	CPU
产生中断	发出中断	接收中断	接收中断

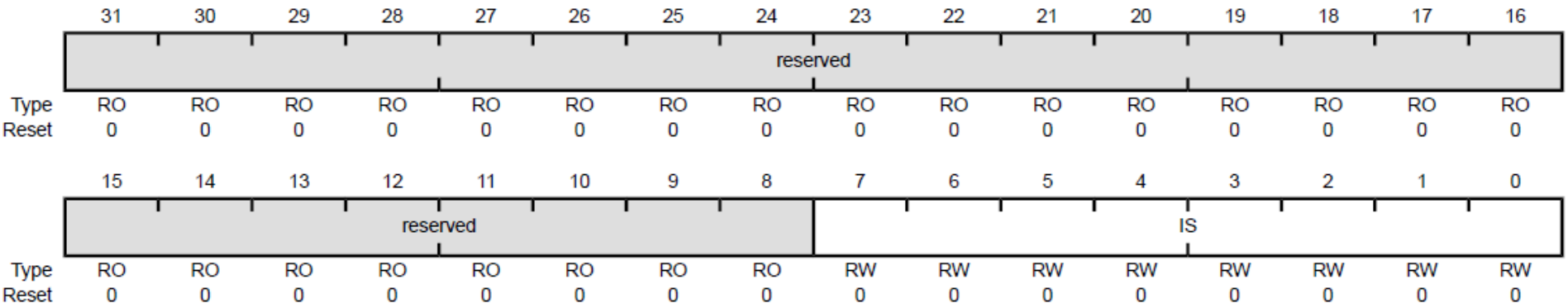
- GPIO的中断，由一组寄存器管理
- 设置中断的触发条件：电平触发（高电平、低电平）、边沿触发（上升沿、下降沿）
- 中断的状态和设置：中断状态、中断使能和屏蔽，中断清除



➢ GPIO中断感知寄存器（GPIOIS）

– 0：边沿触发

– 1：电平触发



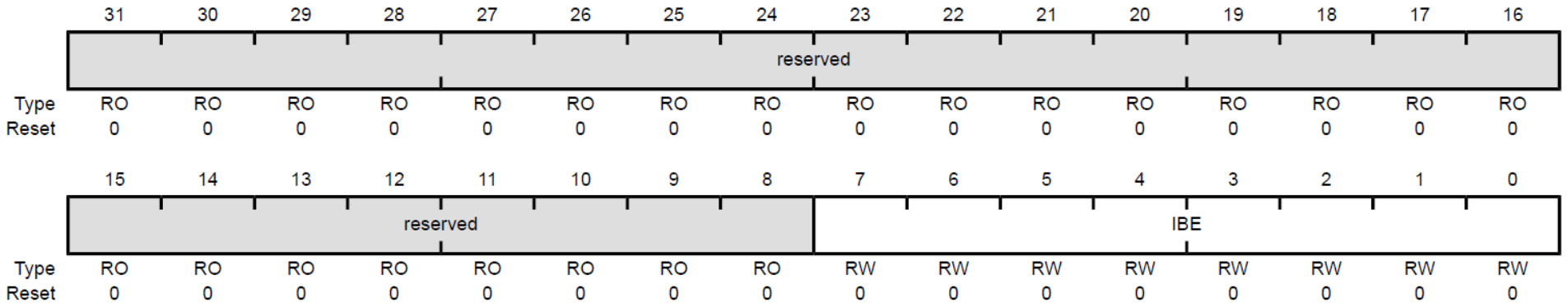
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IS	RW	0x00	GPIO Interrupt Sense

Value	Description
0	The edge on the corresponding pin is detected (edge-sensitive).
1	The level on the corresponding pin is detected (level-sensitive).

➢ GPIO中断边沿控制（GPIOIBE）

– 0：由GPIOIEV决定中断触发边沿

– 1：上升沿、下降沿都触发中断

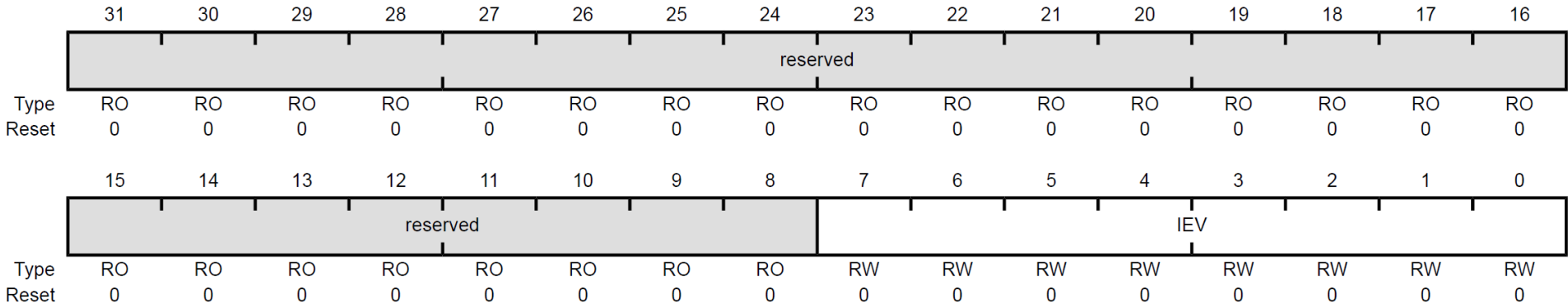


Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IBE	RW	0x00	GPIO Interrupt Both Edges

Value	Description
0	Interrupt generation is controlled by the **GPIO Interrupt Event (GPIOIEV)** register (see page 763).
1	Both edges on the corresponding pin trigger an interrupt.

➢ 中断事件寄存器（GPIOIEV）

- 0：下降沿或低电平触发
- 1：上升沿或高电平触发



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IEV	RW	0x00	GPIO Interrupt Event

Value	Description
0	A falling edge or a Low level on the corresponding pin triggers an interrupt.
1	A rising edge or a High level on the corresponding pin triggers an interrupt.

➢ 中断屏蔽寄存器（GPIOIM）

– 0：相应位的中断被屏蔽

– 1：相应位的中断使能

– 配置中断时，一定要先将中断屏蔽

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved							DMAIME	IME							
Type	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

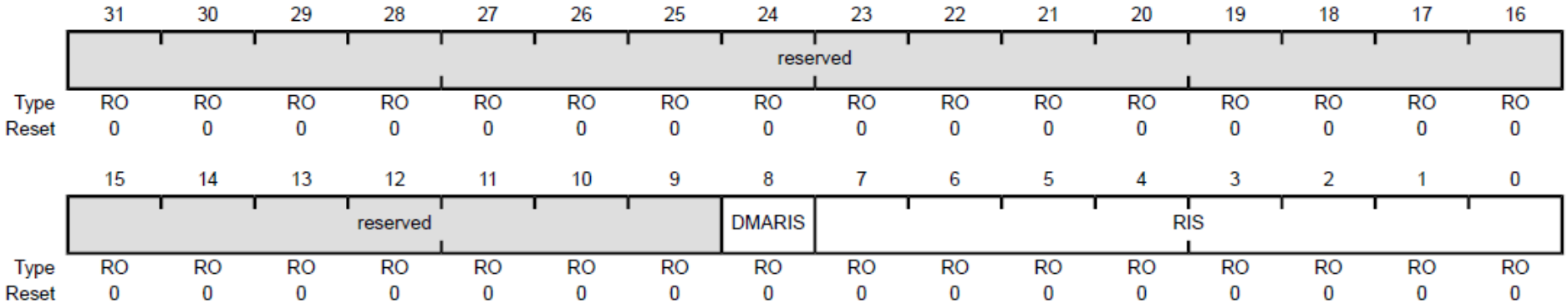
7:0	IME	RW	0x00	GPIO Interrupt Mask Enable

Value	Description
0	The interrupt from the corresponding pin is masked.
1	The interrupt from the corresponding pin is sent to the interrupt controller.

➢ 原始中断状态寄存器（GPIORIS）

– 0：相应的位不满足触发条件

– 1：相应的位满足触发条件



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved							DMARIS	RIS							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7:0	RIS	RO	0x00	GPIO Interrupt Raw Status

Value	Description
0	An interrupt condition has not occurred on the corresponding pin.
1	An interrupt condition has occurred on the corresponding pin.

For edge-detect interrupts, this bit is cleared by writing a 1 to the corresponding bit in the **GPIOICR** register.

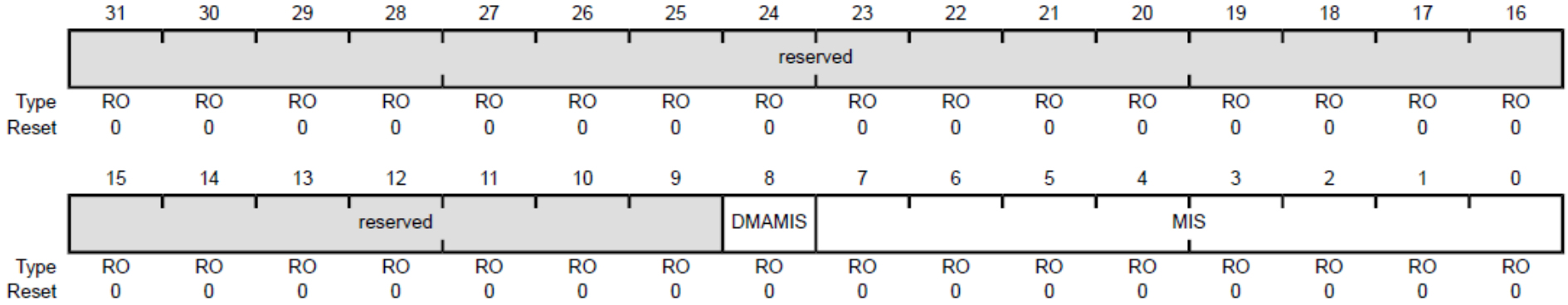
For a GPIO level-detect interrupt, the bit is cleared when the level is deasserted.

如果是电平触发的中断，中断条件消失后，相应的位会自动清零
如果是边沿触发的中断，必须通过GPIOICR寄存器清除中断

➢ 屏蔽中断状态寄存器（GPIOMIS）

– 0：相应的位不满足触发条件或满足条件但中断被屏蔽

– 1：相应的位满足触发条件且中断未被屏蔽



Bit/Field	Name	Type	Reset	Description
7:0	MIS	RO	0x00	GPIO Masked Interrupt Status

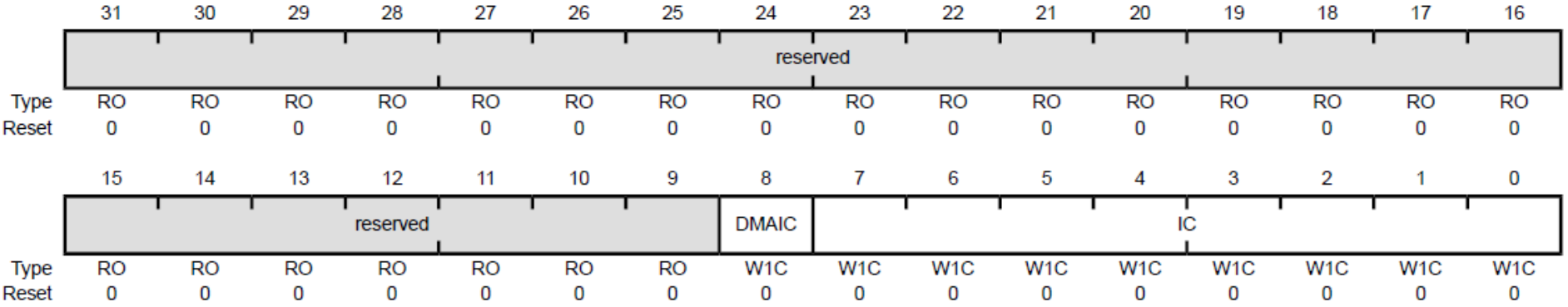
Value	Description
0	An interrupt condition on the corresponding pin is masked or has not occurred.
1	An interrupt condition on the corresponding pin has triggered an interrupt to the interrupt controller.

For edge-detect interrupts, this bit is cleared by writing a 1 to the corresponding bit in the **GPIOICR** register.

For a GPIO level-detect interrupt, the bit is cleared when the level is deasserted.

➢ 中断清除寄存器（GPIOICR）

– 在相应位写1清除中断

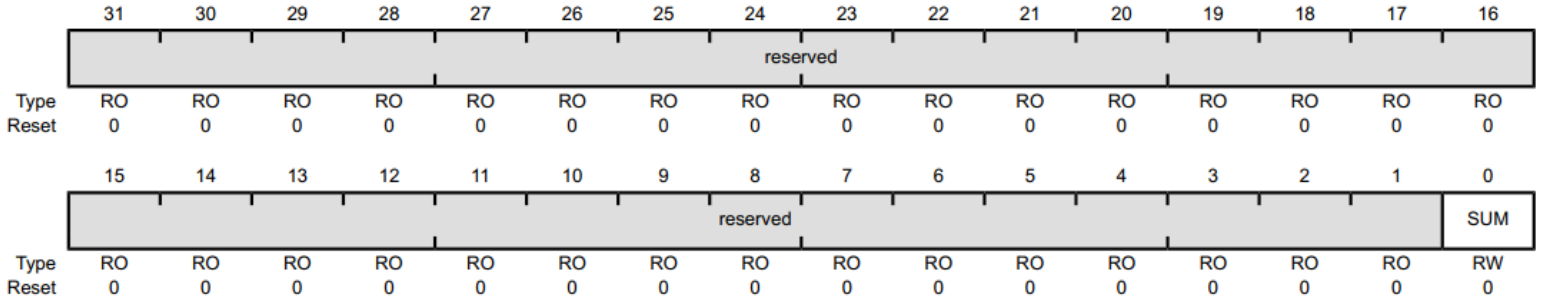


7:0	IC	W1C	0x00	GPIO Interrupt Clear

Value	Description
0	The corresponding interrupt is unaffected.
1	The corresponding interrupt is cleared.

➢ 中断选择寄存器（GPIOSI）

- PORTP和PORTQ每个管脚都有相应的中断向量
- GPIOSI.SUM选择是为每个管脚发送中断信号，还是汇总后通过0号管脚发送（SUM=0，汇总；SUM=1，每个管脚单独发送）



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															SUM
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```
IntDefaultHandler,                      // GPIO Port P (Summary or P0)
IntDefaultHandler,                      // GPIO Port P1
IntDefaultHandler,                      // GPIO Port P2
IntDefaultHandler,                      // GPIO Port P3
IntDefaultHandler,                      // GPIO Port P4
IntDefaultHandler,                      // GPIO Port P5
IntDefaultHandler,                      // GPIO Port P6
IntDefaultHandler,                      // GPIO Port P7
IntDefaultHandler,                      // GPIO Port Q (Summary or Q0)
IntDefaultHandler,                      // GPIO Port Q1
IntDefaultHandler,                      // GPIO Port Q2
IntDefaultHandler,                      // GPIO Port Q3
IntDefaultHandler,                      // GPIO Port Q4
IntDefaultHandler,                      // GPIO Port Q5
IntDefaultHandler,                      // GPIO Port Q6
IntDefaultHandler,                      // GPIO Port Q7
IntDefaultHandler,                      // GPIO Port R
IntDefaultHandler,                      // GPIO Port S
```

GPIO中断流程

- 配置GPIO中断条件：电平 or 边沿？高（上升沿） or 低（下降沿）？
- 使能中断 GPIOIM；
- 配置NVIC和CPU相关寄存器；
- 响应中断，清除中断状态。

外设		NVIC	CPU		外设
产生中断	发出中断	接收中断	接收中断		清除中断
设置要操作的引脚的中断的产生条件：电平还是边沿、高（上升）或低（下降）	GPIOIM中使能中断			中断响应 判断中断的具体来源	GPIOICR

➢ 设置GPIO中断

– 库函数提供了GPIOIntEnable和GPIOIntTypeSet使能中断并设置通断的触发条件

```
GPIOIntTypeSet(GPIO_PORTJ_AHB_BASE,GPIO_INT_PIN_0,GPIO_RISING_EDGE);

GPIOIntEnable(GPIO_PORTJ_AHB_BASE,GPIO_INT_PIN_0);
```

➢ 设置GPIO中断

GPIOIntTypeSet

Sets the interrupt type for the specified pin(s).

Prototype:

```
void
GPIOIntTypeSet(uint32_t ui32Port,
               uint8_t ui8Pins,
               uint32_t ui32IntType)
```

✓ 为特定的引脚进行中断触发的设置

✓ 可以设置位电平触发或者边沿触发

➢ 设置GPIO中断

Parameters:

ui32Port is the base address of the GPIO port.
ui8Pins is the bit-packed representation of the pin(s).
ui32IntType specifies the type of interrupt trigger mechanism.

Description:

This function sets up the various interrupt trigger mechanisms for the specified pin(s) on the selected GPIO port.

One of the following flags can be used to define the *ui32IntType* parameter:

- **GPIO_FALLING_EDGE** sets detection to edge and trigger to falling
- **GPIO_RISING_EDGE** sets detection to edge and trigger to rising
- **GPIO_BOTH_EDGES** sets detection to both edges
- **GPIO_LOW_LEVEL** sets detection to low level
- **GPIO_HIGH_LEVEL** sets detection to high level

In addition to the above flags, the following flag can be OR'd in to the *ui32IntType* parameter:

- **GPIO_DISCRETE_INT** sets discrete interrupts for each pin on a GPIO port.

The **GPIO_DISCRETE_INT** is not available on all devices or all GPIO ports, consult the data sheet to ensure that the device and the GPIO port supports discrete interrupts.

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Note:

In order to avoid any spurious interrupts, the user must ensure that the GPIO inputs remain stable for the duration of this function.

Returns:

None.

GPIOIntTypeSet

```
#define GPIO_FALLING_EDGE       0x00000000  // Interrupt on falling edge
#define GPIO_RISING_EDGE        0x00000004  // Interrupt on rising edge
#define GPIO_BOTH_EDGES         0x00000001  // Interrupt on both edges
#define GPIO_LOW_LEVEL          0x00000002  // Interrupt on low level
#define GPIO_HIGH_LEVEL         0x00000006  // Interrupt on high level
#define GPIO_DISCRETE_INT       0x00010000  // Interrupt for individual pins
```

```
void GPIOIntTypeSet(uint32_t ui32Port, uint8_t ui8Pins,
                    uint32_t ui32IntType)
{
    HWREG(ui32Port + GPIO_O_IBE) = ((ui32IntType & 1) ?
                                    (HWREG(ui32Port + GPIO_O_IBE) | ui8Pins) :
                                    (HWREG(ui32Port + GPIO_O_IBE) & ~(ui8Pins)));
    HWREG(ui32Port + GPIO_O_IS) = ((ui32IntType & 2) ?
                                   (HWREG(ui32Port + GPIO_O_IS) | ui8Pins) :
                                   (HWREG(ui32Port + GPIO_O_IS) & ~(ui8Pins)));
    HWREG(ui32Port + GPIO_O_IEV) = ((ui32IntType & 4) ?
                                    (HWREG(ui32Port + GPIO_O_IEV) | ui8Pins) :
                                    (HWREG(ui32Port + GPIO_O_IEV) & ~(ui8Pins)));

    HWREG(ui32Port + GPIO_O_SI) = ((ui32IntType & 0x10000) ?
                                   (HWREG(ui32Port + GPIO_O_SI) | 0x01) :
                                   (HWREG(ui32Port + GPIO_O_SI) & ~(0x01)));
}
```

GPIOIBE

0：由GPIOIEV决定中断触发边沿

1：上升沿、下降沿都触发中断

GPIOIS

0：边沿触发

1：电平触发

GPIOIEV

0：下降沿或低电平触发

1：上升沿或高电平触发

➢ GPIOIntEnable

GPIOIntEnable

Enables the specified GPIO interrupts.

Prototype:

```
void
GPIOIntEnable(uint32_t ui32Port,
              uint32_t ui32IntFlags)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui32IntFlags is the bit mask of the interrupt sources to enable.

Description:

This function enables the indicated GPIO interrupt sources. Only the sources that are enabled can be reflected to the processor interrupt; disabled sources have no effect on the processor.

The *ui32IntFlags* parameter is the logical OR of any of the following:

- **GPIO_INT_PIN_0** - interrupt due to activity on Pin 0.
- **GPIO_INT_PIN_1** - interrupt due to activity on Pin 1.
- **GPIO_INT_PIN_2** - interrupt due to activity on Pin 2.
- **GPIO_INT_PIN_3** - interrupt due to activity on Pin 3.
- **GPIO_INT_PIN_4** - interrupt due to activity on Pin 4.
- **GPIO_INT_PIN_5** - interrupt due to activity on Pin 5.
- **GPIO_INT_PIN_6** - interrupt due to activity on Pin 6.
- **GPIO_INT_PIN_7** - interrupt due to activity on Pin 7.
- **GPIO_INT_DMA** - interrupt due to DMA activity on this GPIO module.

GPIOIntEnable

```
void
GPIOIntEnable(uint32_t ui32Port, uint32_t ui32IntFlags)
{
    // Check the arguments.
    ASSERT(_GPIOBaseValid(ui32Port));
    // Enable the interrupts.
    HWREG(ui32Port + GPIO_O_IM) |= ui32IntFlags;
}
```

- **中断屏蔽寄存器（GPIOIM）**
  - 0：相应位的中断被屏蔽
  - 1：相应位的中断使能

输入参数在gpio.h中定义：

```
#define GPIO_INT_PIN_0          0x00000001
#define GPIO_INT_PIN_1          0x00000002
#define GPIO_INT_PIN_2          0x00000004
#define GPIO_INT_PIN_3          0x00000008
#define GPIO_INT_PIN_4          0x00000010
#define GPIO_INT_PIN_5          0x00000020
#define GPIO_INT_PIN_6          0x00000040
#define GPIO_INT_PIN_7          0x00000080
```

➢库函数提供了GPIOIntClear函数，用来清除GPIO的中断

GPIOIntClear

Clears the specified interrupt sources.

Prototype:

```
void
GPIOIntClear(uint32_t ui32Port,
             uint32_t ui32IntFlags)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui32IntFlags is the bit mask of the interrupt sources to disable.

Description:

Clears the interrupt for the specified interrupt source(s).

The *ui32IntFlags* parameter is the logical OR of the **GPIO_INT_*** values.

Note:

Because there is a write buffer in the Cortex-M processor, it may take several clock cycles before the interrupt source is actually cleared. Therefore, it is recommended that the interrupt source be cleared early in the interrupt handler (as opposed to the very last action) to avoid returning from the interrupt handler before the interrupt source is actually cleared. Failure to do so may result in the interrupt handler being immediately reentered (because the interrupt controller still sees the interrupt source asserted).

Returns:

None.

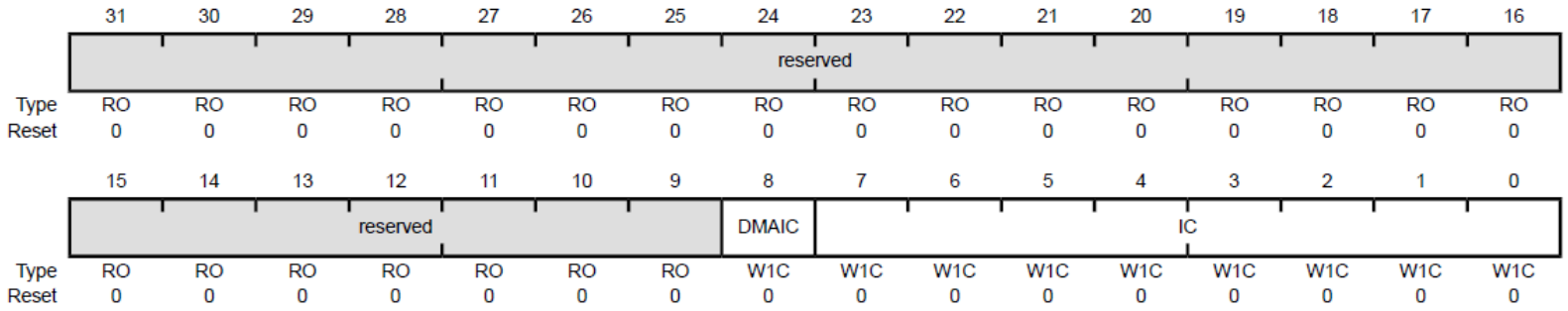
➢ GPIOIntClear

TivaWare_C_Series-2.1.4.178\driverlib\gpio.c

```
GPIOIntClear(uint32_t ui32Port, uint32_t ui32IntFlags)
{
    // Check the arguments.
    ASSERT(_GPIOBaseValid(ui32Port));
    // Clear the interrupts.
    HWREG(ui32Port + GPIO_O_ICR) = ui32IntFlags;
}
```

➢ 中断清除寄存器（GPIOICR）

– 在相应位写1清除中断



➢ 库函数提供了GPIOIntStatus查看中断的具体来源

GPIOIntStatus

Gets interrupt status for the specified GPIO port.

Prototype:

```
uint32_t
GPIOIntStatus(uint32_t ui32Port,
              bool bMasked)
```

Parameters:

ui32Port is the base address of the GPIO port.

bMasked specifies whether masked or raw interrupt status is returned.

Description:

If *bMasked* is set as **true**, then the masked interrupt status is returned; otherwise, the raw interrupt status is returned.

Returns:

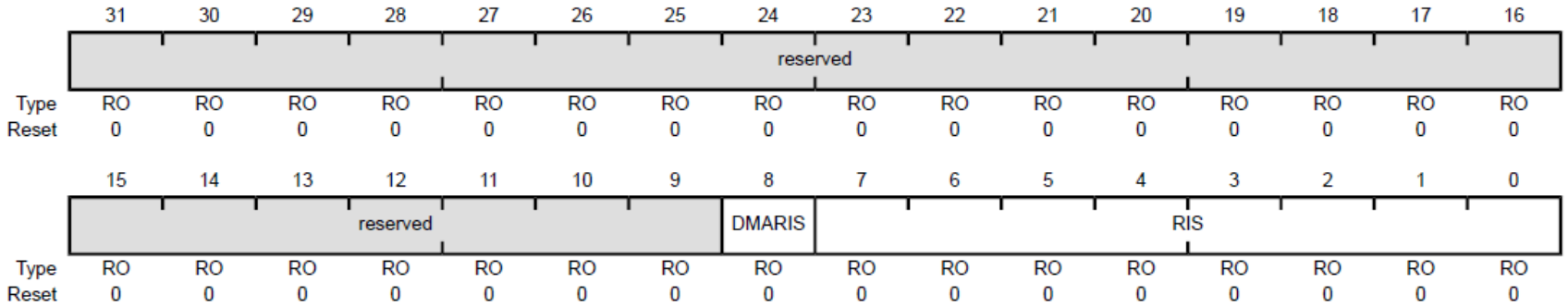
Returns the current interrupt status for the specified GPIO module. The value returned is the logical OR of the **GPIO_INT_*** values that are currently active.

➢ GPIOIntStatus

```
uint32_t
GPIOIntStatus(uint32_t ui32Port, bool bMasked)
{
    ASSERT(_GPIOBaseValid(ui32Port));
    // Return the interrupt status.
    if(bMasked)
    {
        return(HWREG(ui32Port + GPIO_O_MIS));
    }
    else
    {
        return(HWREG(ui32Port + GPIO_O_RIS));
    }
}
```

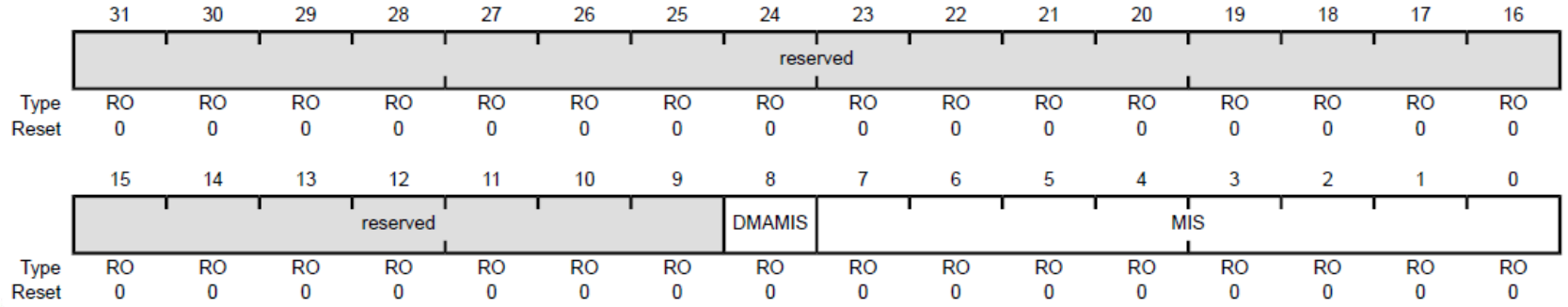
➢ 原始中断状态寄存器（GPIORIS）

- 0：相应的位不满足触发条件
- 1：相应的位满足触发条件



➢ 屏蔽中断状态寄存器（GPIOMIS）

- 0：相应的位不满足触发条件或满足条件但中断被屏蔽
- 1：相应的位满足触发条件且中断未被屏蔽



小节

- 中断和异常基本概念
- 中断的响应流程
- 中断控制寄存器的访问
- GPIO的中断源管理
- 设置GPIO中断

- **以下为一段中断的初始化程序，每行代码分别影响了哪些寄存器？这行代码执行过程中，是怎样操作这些寄存器的？**

- IntMasterEnable();
- IntEnable(INT_GPIOJ);　　// INT_GPIOJ=67

- GPIOIntTypeSet(GPIO_PORTJ_AHB_BASE,GPIO_INT_PIN_1,GPIO_RISING_EDGE);

- GPIOIntClear(GPIO_PORTJ_AHB_BASE,GPIO_INT_PIN_1);
- GPIOIntEnable(GPIO_PORTJ_AHB_BASE,GPIO_INT_PIN_1);

谢谢！