快速链接:

.

👉👉👉 ARMv8/ARMv9架构入门到精通-[目录] 👈👈👈

- 付费专栏-付费课程 **【购买须知】:**
- 联系方式-加入交流群 ----**联系方式-加入交流群**
- 个人博客笔记导读目录(全部)

目录

1、gic的版本

2、GICv3/gicv4的模型图

3、gic中断号的划分

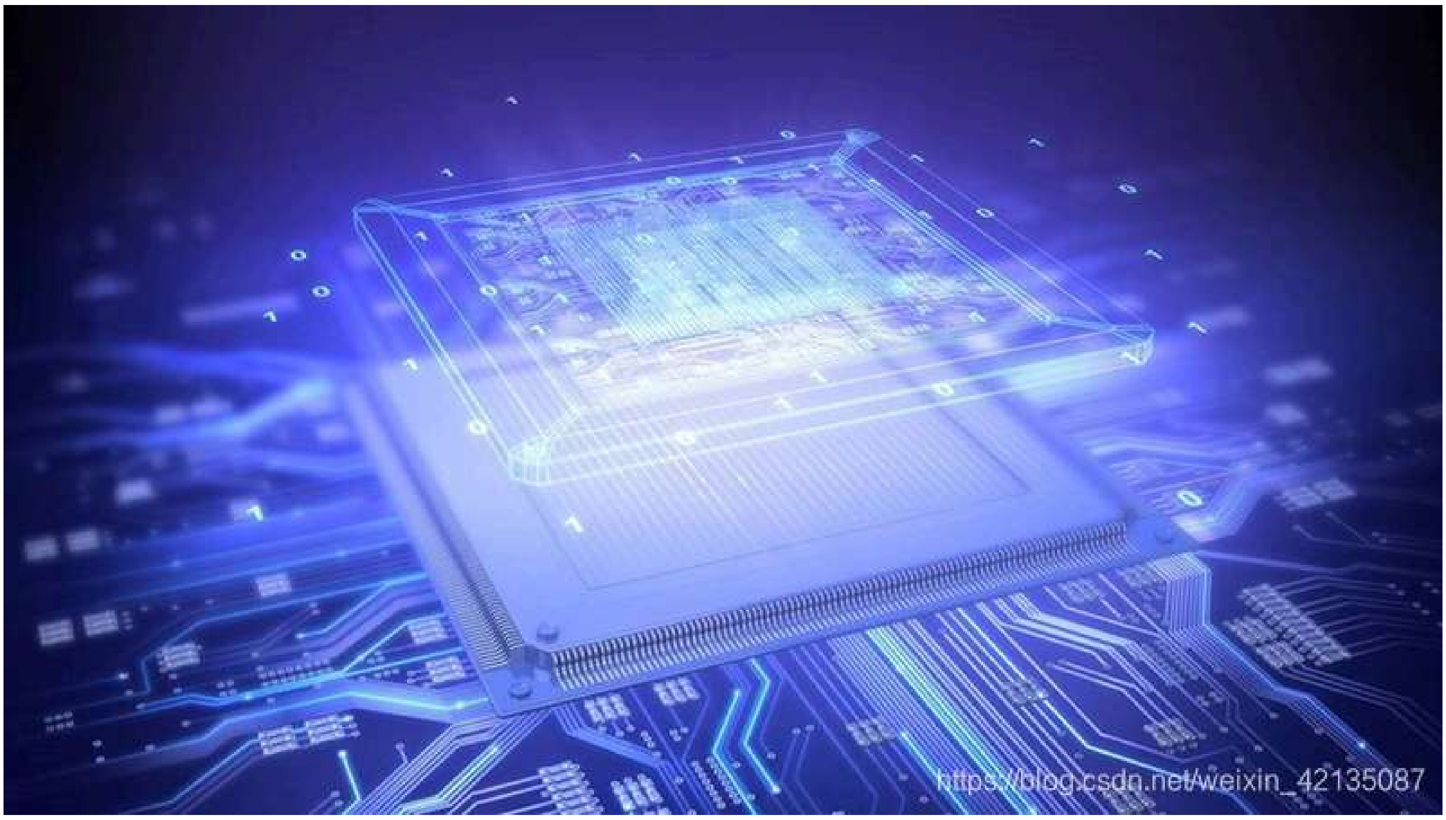
4、GIC连接方式

5、gic的状态

6、gic框架

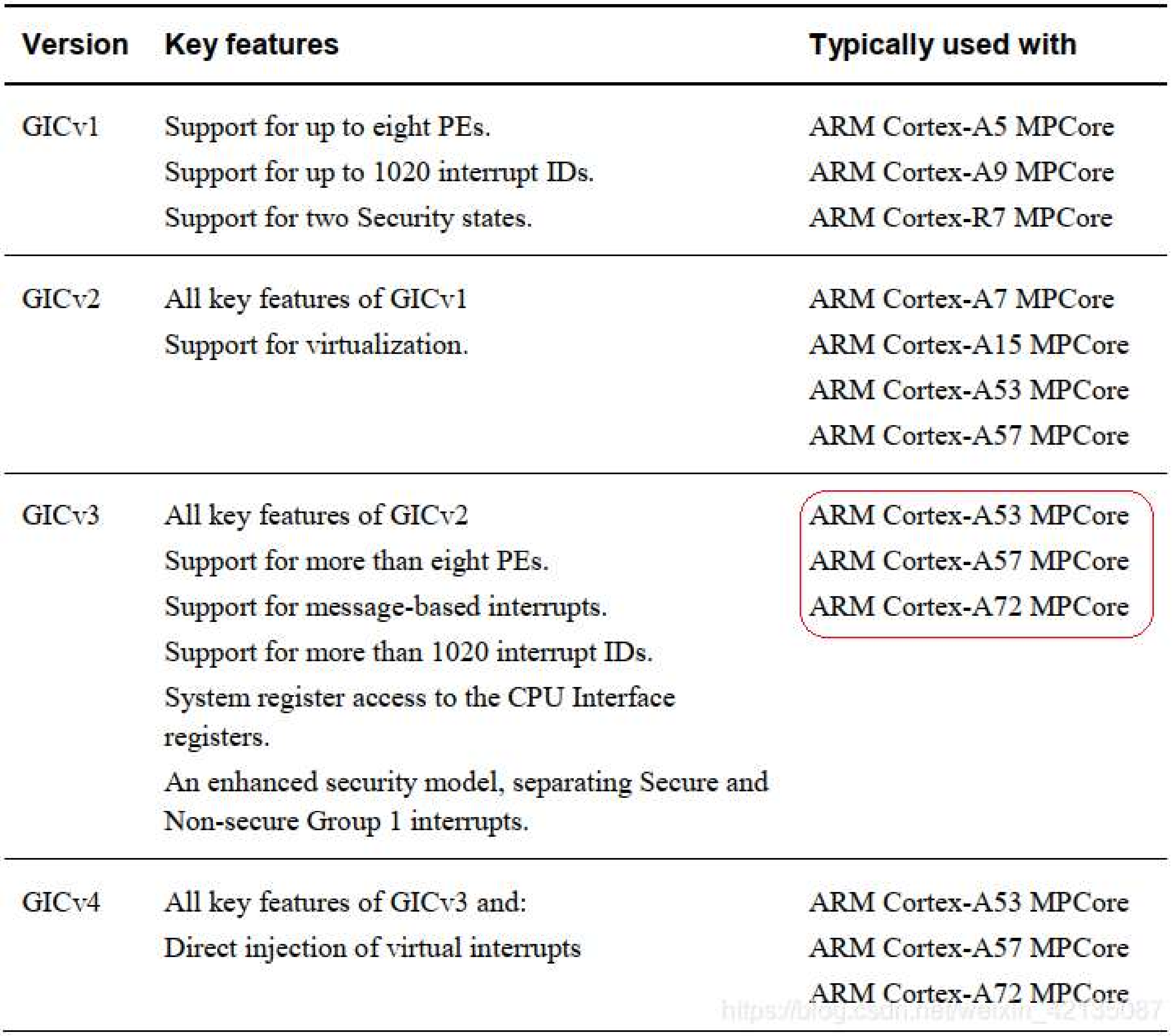
7、gic Configuring

• When ARE==0, affinity routing is disabled (legacy operation) • When ARE==1, affinity routing is enabled (GICv3 operation) ![在这里插入图片描述](https://i-blog.csdnimg.cn/blog_migrate/6830a59b75857b3e3453a26db2742af0.png)



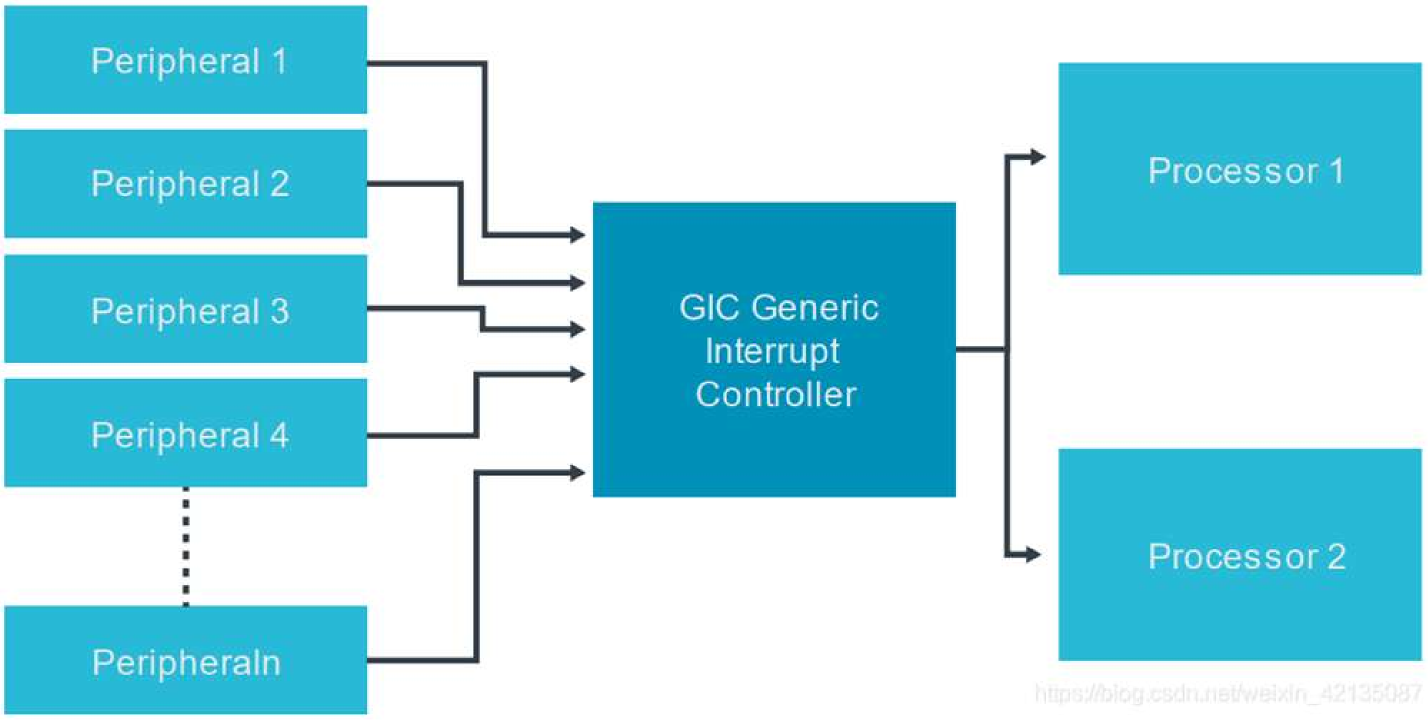
1、gic的版本

GIC是一个为Cortex-A和Arm Cortex-R设计的标准的中断控制器



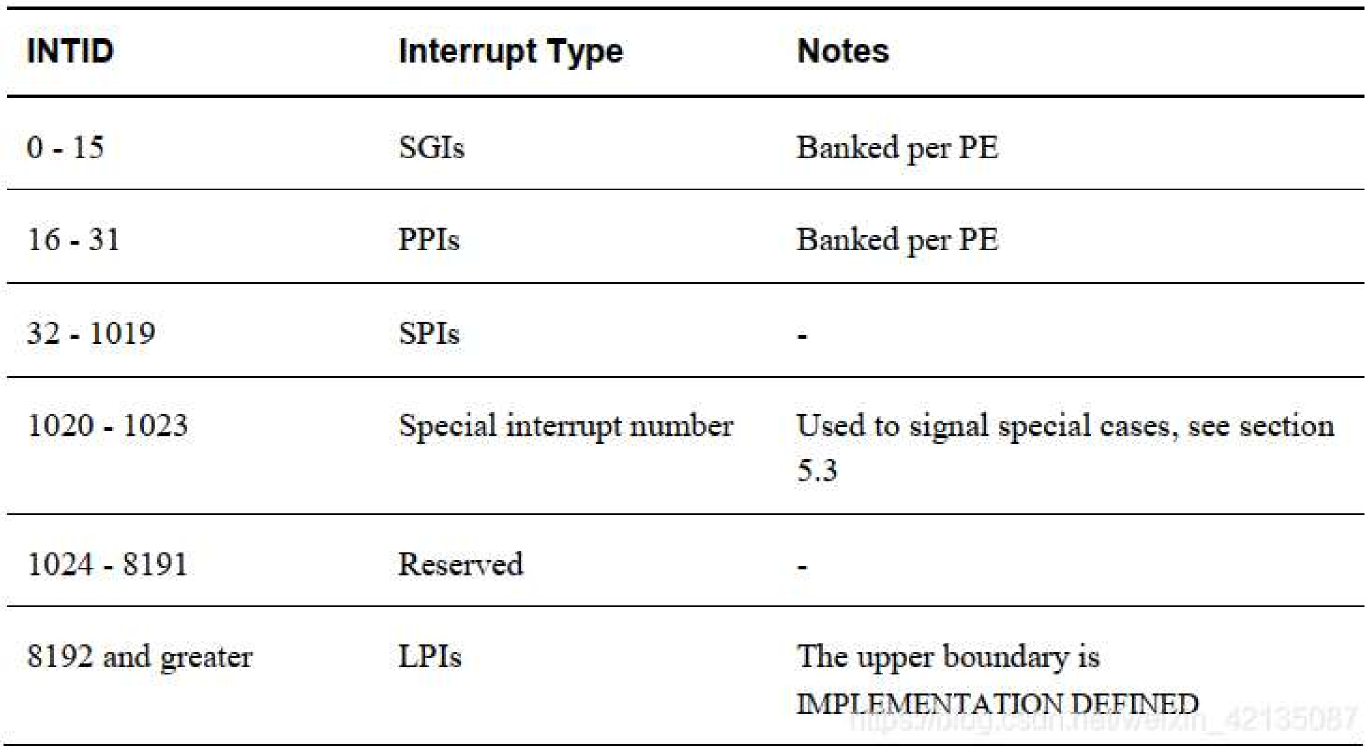
Version	Key features	Typically used with
GICv1	Support for up to eight PEs. Support for up to 1020 interrupt IDs. Support for two Security states.	ARM Cortex-A5 MPCore ARM Cortex-A9 MPCore ARM Cortex-R7 MPCore
GICv2	All key features of GICv1 Support for virtualization.	ARM Cortex-A7 MPCore ARM Cortex-A15 MPCore ARM Cortex-A53 MPCore ARM Cortex-A57 MPCore
GICv3	All key features of GICv2 Support for more than eight PEs. Support for message-based interrupts. Support for more than 1020 interrupt IDs. System register access to the CPU Interface registers. An enhanced security model, separating Secure and Non-secure Group 1 interrupts.	ARM Cortex-A53 MPCore ARM Cortex-A57 MPCore ARM Cortex-A72 MPCore
GICv4	All key features of GICv3 and: Direct injection of virtual interrupts	ARM Cortex-A53 MPCore ARM Cortex-A57 MPCore ARM Cortex-A72 MPCore

2、GICv3/gicv4的模型图



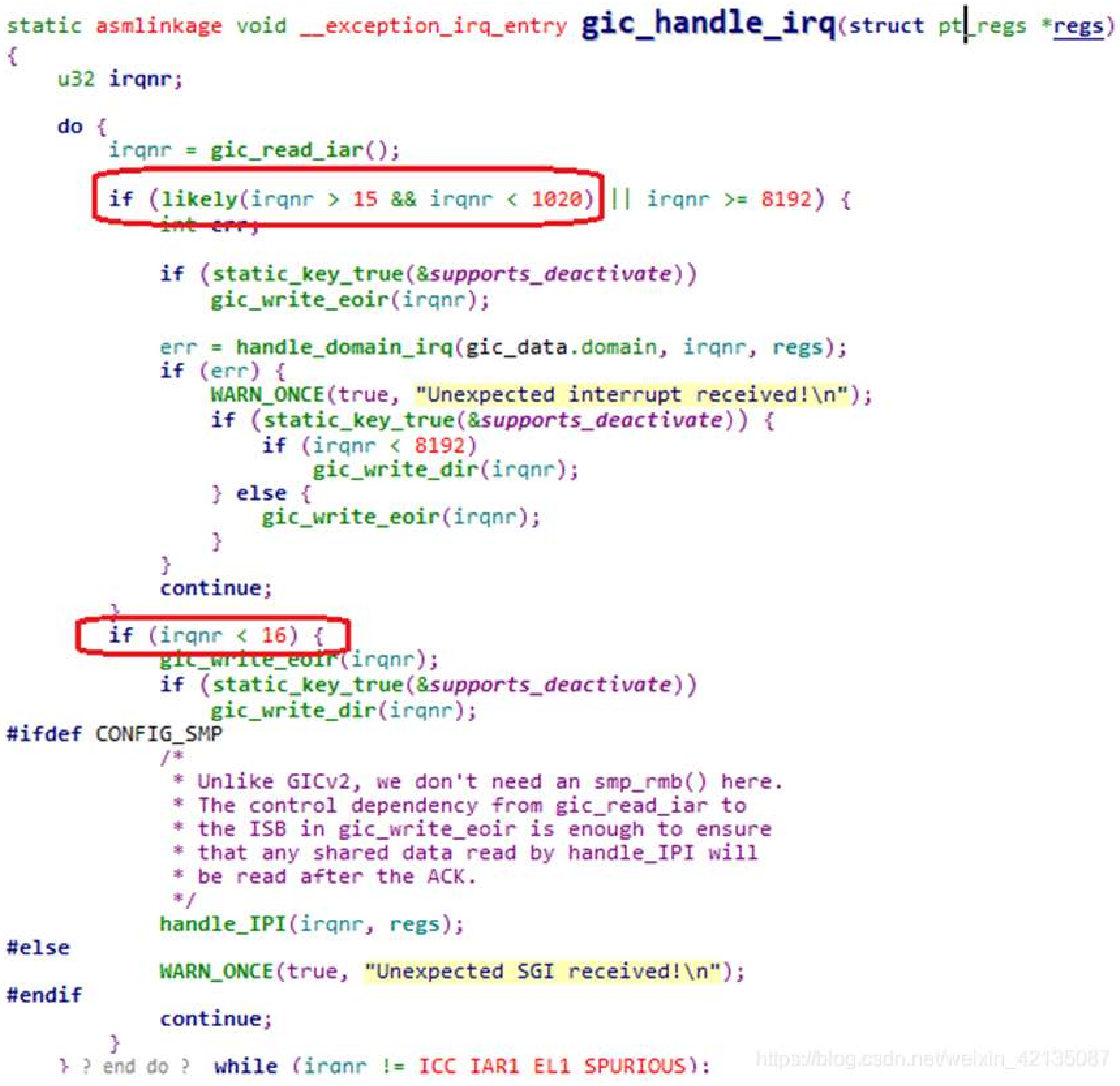
3、gic中断号的划分

- Shared Peripheral Interrupt (SPI)
- Private Peripheral Interrupt (PPI)
- Software Generated Interrupt (SGI)
- Locality-specific Peripheral Interrupt (LPI)



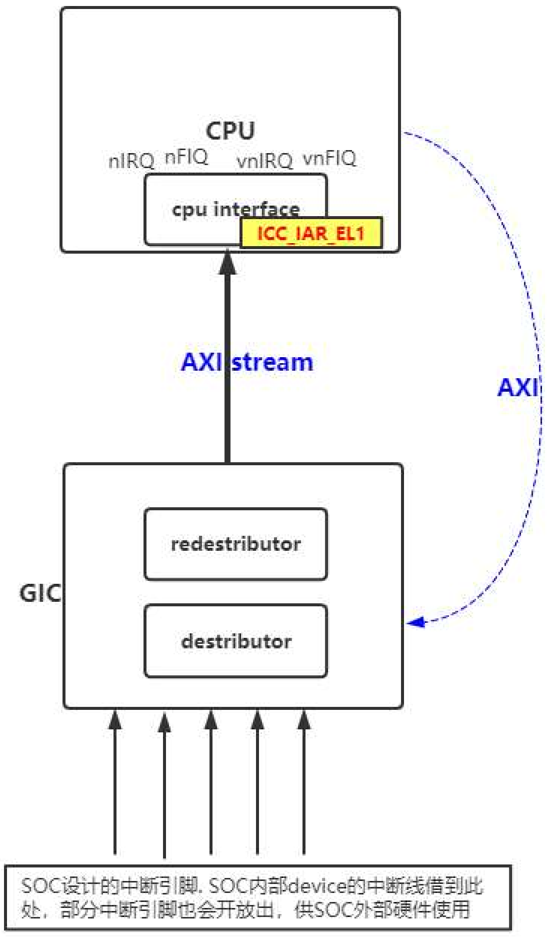
INTID	Interrupt Type	Notes
0 - 15	SGIs	Banked per PE
16 - 31	PPIs	Banked per PE
32 - 1019	SPIs	-
1020 - 1023	Special interrupt number	Used to signal special cases, see section 5.3
1024 - 8191	Reserved	-
8192 and greater	LPIs	The upper boundary is IMPLEMENTATION DEFINED

(使用示例)

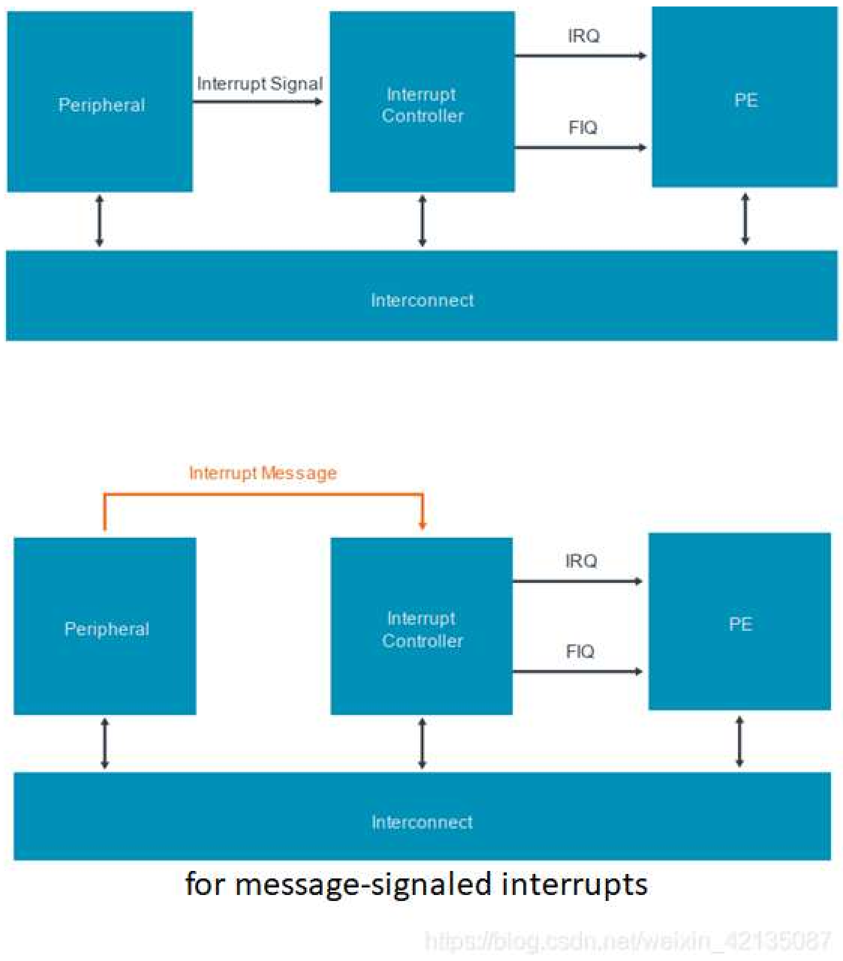


```
static asmlinkage void __exception_irq_entry gic_handle_irq(struct pt_regs *regs)
{
    u32 irqnr;

    do {
        irqnr = gic_read_iar();

        if (likely(irqnr > 15 && irqnr < 1020) || irqnr >= 8192) {
            int err;

            if (static_key_true(&supports_deactivate))
                gic_write_eoir(irqnr);

            err = handle_domain_irq(gic_data.domain, irqnr, regs);
            if (err) {
                WARN_ONCE(true, "Unexpected interrupt received!\n");
                if (static_key_true(&supports_deactivate)) {
                    if (irqnr < 8192)
                        gic_write_dir(irqnr);
                } else {
                    gic_write_eoir(irqnr);
                }
            }
            continue;
        }
        if (irqnr < 16) {
            gic_write_eoir(irqnr);
            if (static_key_true(&supports_deactivate))
                gic_write_dir(irqnr);
#ifdef CONFIG_SMP
            /*
             * Unlike GICv2, we don't need an smp_rmb() here.
             * The control dependency from gic_read_iar to
             * the ISB in gic_write_eoir is enough to ensure
             * that any shared data read by handle_IPI will
             * be read after the ACK.
             */
            handle_IPI(irqnr, regs);
#else
            WARN_ONCE(true, "Unexpected SGI received!\n");
#endif
            continue;
        }
    } ? end do ? while (irqnr != ICC_IAR1_EL1_SPURIOUS);
```

4、GIC连接方式

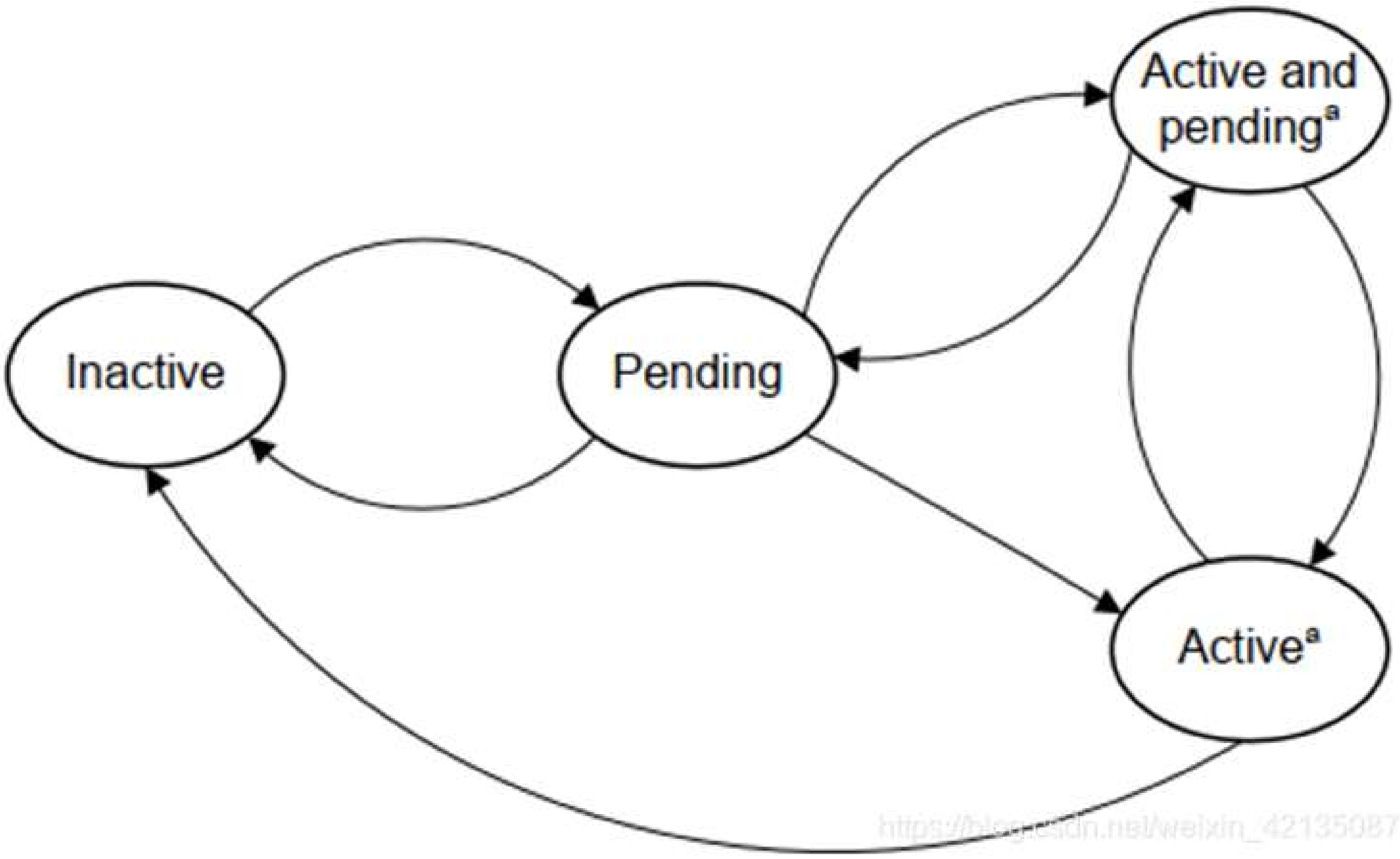


CPU
nIRQ nFIQ vnIRQ vnFIQ
cpu interface
ICC_IAR_EL1
AXI stream
AXI
GIC
redestributor
destributor
SOC设计时的中断引脚, SOC内部device的中断线借到此处，部分中断引脚也会开放出，供SOC外部硬件使用



for message-signaled interrupts

5、gic的状态

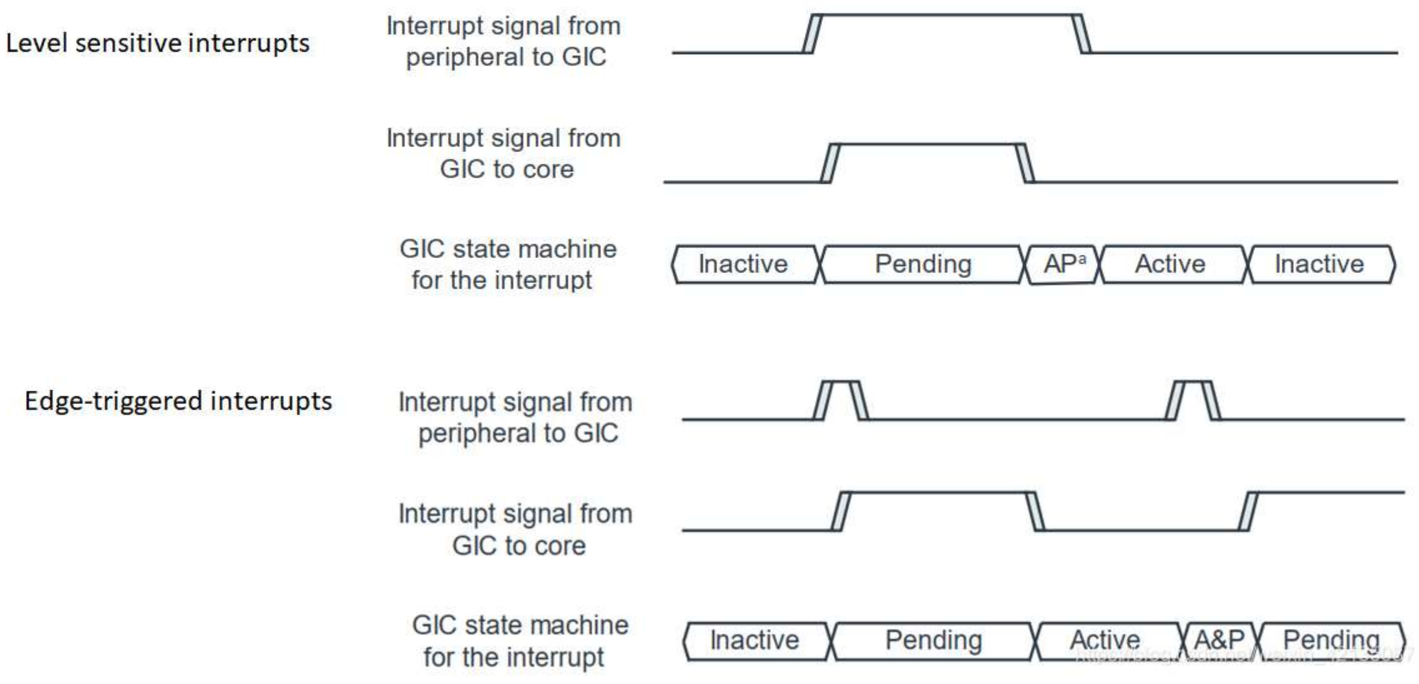


中断的生命周期:

对于电平触发的中断(level-sensitive interrupts)，一个上升沿输入，将中断变成pending，中断信号线

保持高电平直到PE断言该中断信号.

对于边沿触发的中断(edge-sensitive interrupts)，一个上升沿输入，将中断变成pending，中断信号线不会保持高电平.

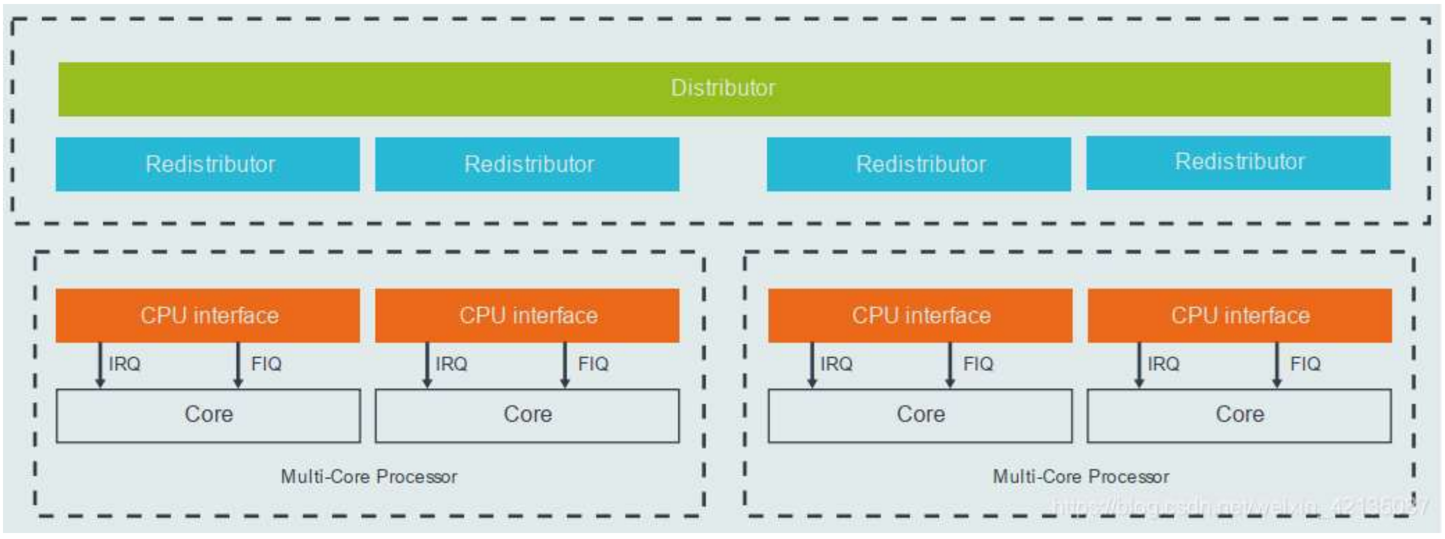

Level sensitive interrupts
Interrupt signal from peripheral to GIC
Interrupt signal from GIC to core
GIC state machine for the interrupt
Inactive
Pending
AP[a]
Active
Inactive
Edge-triggered interrupts
Interrupt signal from peripheral to GIC
Interrupt signal from GIC to core
GIC state machine for the interrupt
Inactive
Pending
Active
A&P
Pending

6、gic框架

• Distributor interface

• Redistributor interface

• CPU interface



Distributor (GICD_*) for SPIs

• Interrupt prioritization and distribution of SPIs

• Enable and disable SPIs

• Set the priority level of each SPI

• Route information for each SPI

• Set each SPI to be level-sensitive or edge-triggered

• Generate message-signaled SPIs

• Control the active and pending state of SPIs
• Determine the programmer’s model that is used in each Security state: affinity routing or legacy

Redistributors (GICR_*)
• Enable and disable SGIs and PPIs
• Set the priority level of SGIs and PPIs
• Set each PPI to be level-sensitive or edge-triggered
• Assign each SGI and PPI to an interrupt group
• Control the state of SGIs and PPIs
• Control the base address for the data structures in memory that support the associated interrupt properties and pending state for LPIs
• Provide power management support for the connected PE

CPU interfaces (ICC_*_ELn)
• Provide general control and configuration to enable interrupt handling
• Acknowledge an interrupt
• Perform a priority drop and deactivation of interrupts
• Set an interrupt priority mask for the PE
• Define the preemption policy for the PE
• Determine the highest priority pending interrupt for the PE

In Arm CoreLink GICv3, the CPU Interface registers are accessed as System registers: ICC_*_ELn.

7、gic Configuring

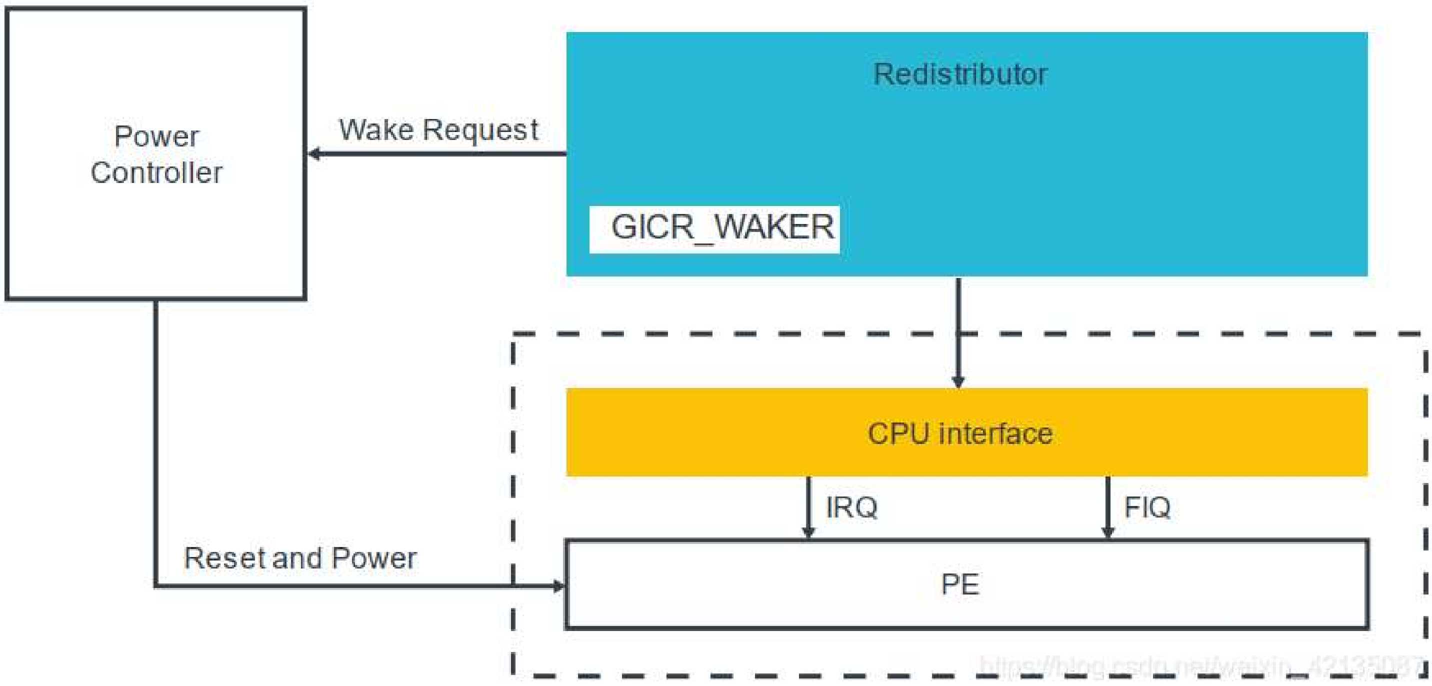
全局配置
GICD_CTLR.ARE: Enable Affinity routing (ARE bits), 1-使用gicv3 mode，0-使用legacy mode(gicv2 mode). 默认为1
GICD_CTLR.EnableGrp1S
GICD_CTLR.EnableGrp1NS
GICD_CTLR.EnableGrp0

注意在GIC-600 does not support legacy operation

(Redistributor)Settings for each PE
Redistributor中包含了一个GICR_WAKER寄存器，用于记录connected PE的状态是onLine还是offline. 如果让PE变成online，软件则必需这样做：
• Clear GICR_WAKER.ProcessorSleep to 0.
• Poll GICR_WAKER.ChildrenAsleep until it reads 0

如果PE is offline (GICR_WAKER.ProcessorSleep==1)时，来了一个中断target到该PE上，将产一个wake request信号，这个信号连接PE的power controller，该controller将会打开PE。然后PE clear the

ProcessorSleep bit



CPU interfaces (ICC_*_ELn)

SRE bit— enable cpu interface
注：有些处理器可能不支持legacy operation，SRE比特位也是固定为1，那么软件就不需要处理该比特了

Set Priority Mask and Binary Point registers
ICC_PMR_EL1、ICC_BPRn_EL1

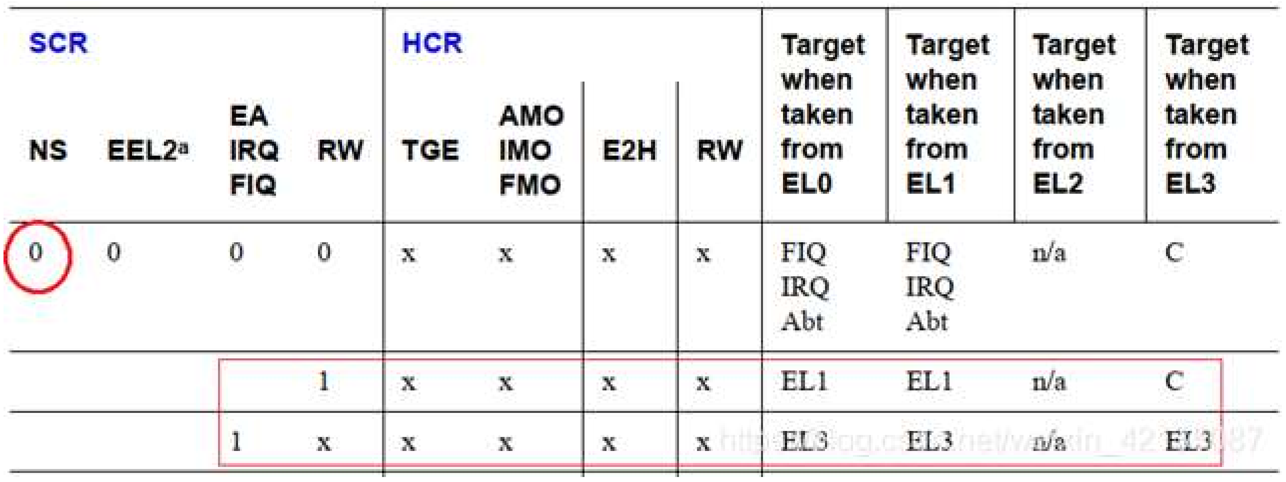
Set EOI mode （EOI:End of interrupt）
ICC_CTLR_EL1 and ICC_CTLR_EL3

Enable signaling of each interrupt group
ICC_IGRPEN1_EL1 (banked by Security state)
ICC_IGRPEN0_EL1

PE configuration

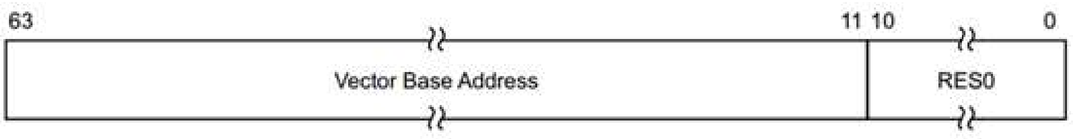
- Routing controls - SCR_EL3 、 HCR_EL2
- Interrupt masks - PSTATE

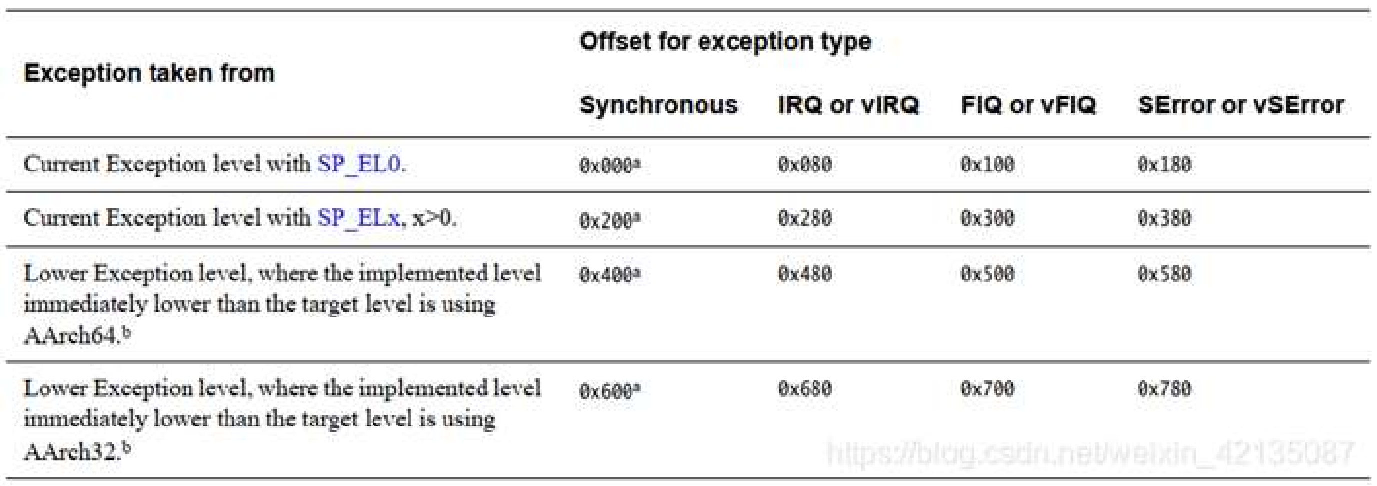
- Vector table - VBAR_ELn



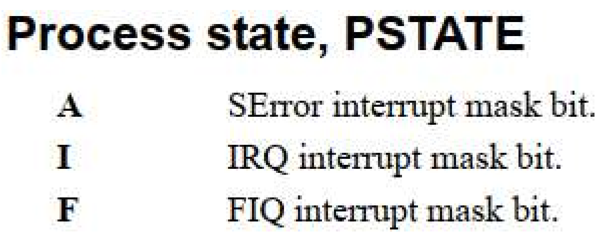
SCR				HCR							
NS	EEL2[a]	EA IRQ FIQ	RW	TGE	AMO IMO FMO	E2H	RW	Target when taken from EL0	Target when taken from EL1	Target when taken from EL2	Target when taken from EL3
0	0	0	0	x	x	x	x	FIQ IRQ Abt	FIQ IRQ Abt	n/a	C
			1	x	x	x	x	EL1	EL1	n/a	C
		1	x	x	x	x	x	EL3	EL3	n/a	EL3

VBAR_EL1, Vector Base Address Register (EL1)





Exception taken from	Offset for exception type			
	Synchronous	IRQ or vIRQ	FIQ or vFIQ	SError or vSError
Current Exception level with SP_EL0.	0x000[a]	0x080	0x100	0x180
Current Exception level with SP_ELx, x>0.	0x200[a]	0x280	0x300	0x380
Lower Exception level, where the implemented level immediately lower than the target level is using AArch64.[b]	0x400[a]	0x480	0x500	0x580
Lower Exception level, where the implemented level immediately lower than the target level is using AArch32.[b]	0x600[a]	0x680	0x700	0x780



Process state, PSTATE

A	SError interrupt mask bit.
I	IRQ interrupt mask bit.
F	FIQ interrupt mask bit.

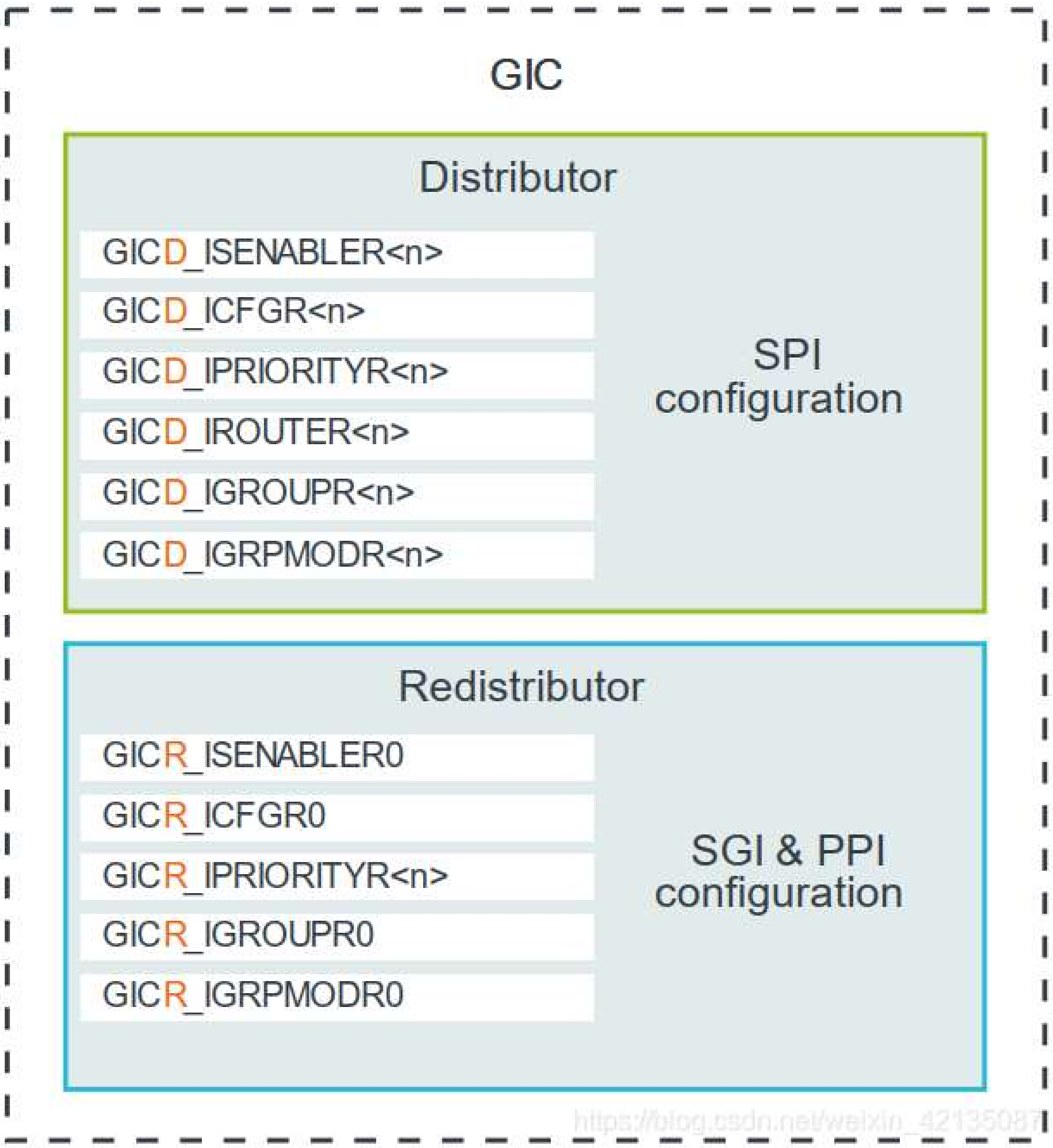
interrupt sources configuration

- SPIs are configured through the Distributor, using the GICD_* registers.

- PPIs and SGIs are configured through the individual Redistributors, using the GICR_* registers
-

对于每一个中断，软件必需配置的：

- Priority: GICD_IPRIORITYn, GICR_IPRIORITYn
- Group: GICD_IGROUPn, GICD_IGRPMODn, GICR_IGROUPn, GICR_IGRPMODn
- Edge-triggered or level-sensitive: GICD_ICFGRn, GICR_ICFGRn
- Enable: GICD_ISENABLERn, GICD_ICENABLER, GICR_ISENABLERn, GICR_ICENABLERn

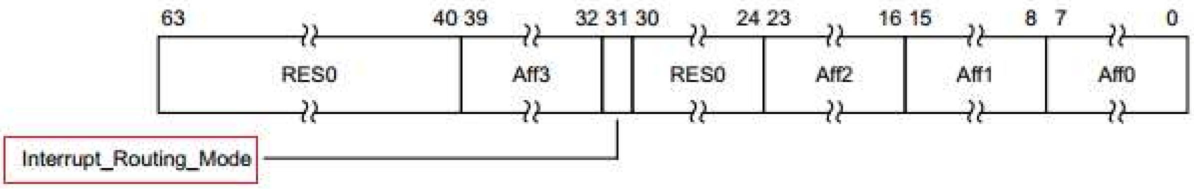


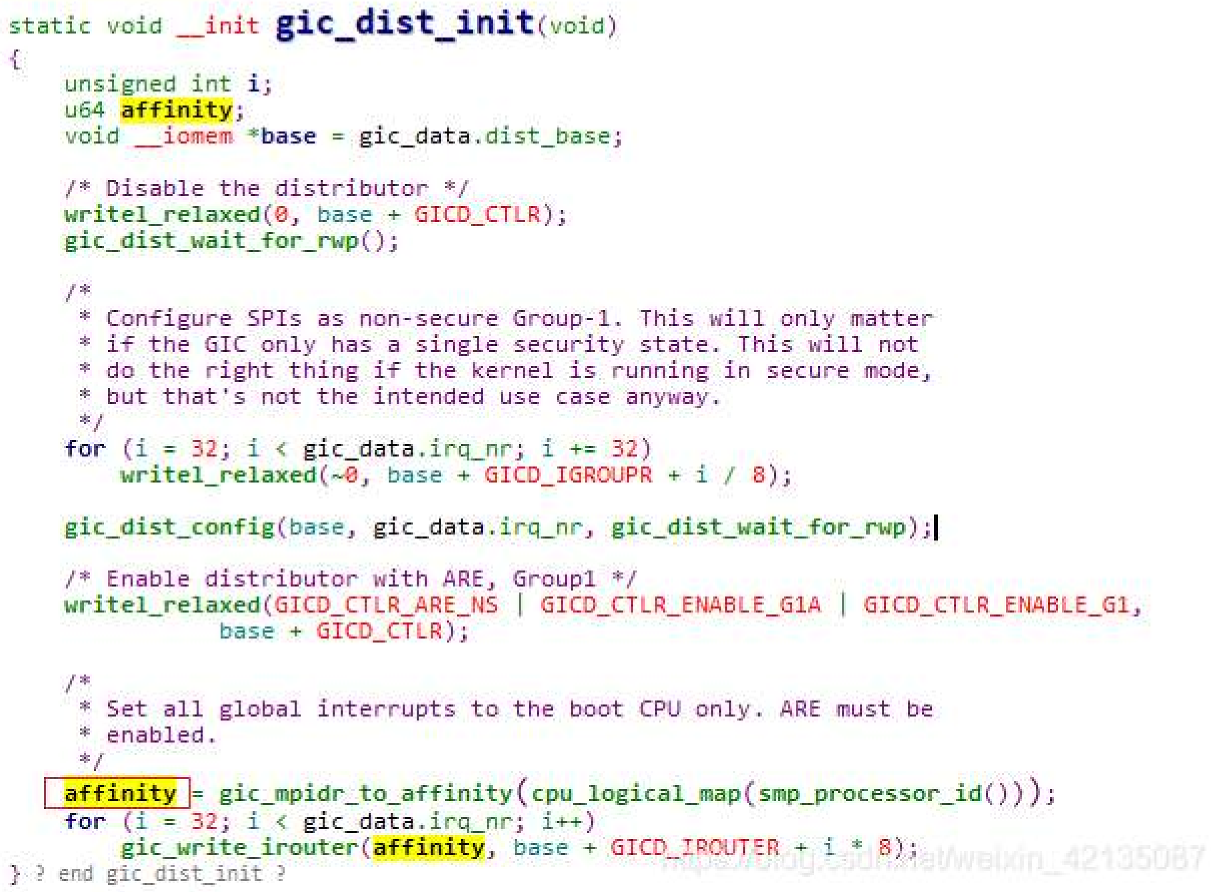
Setting the target PE for SPIs

- GICD_IROUTERn.Interrupt_Routing_Mode == 0 rounting到制定的PE

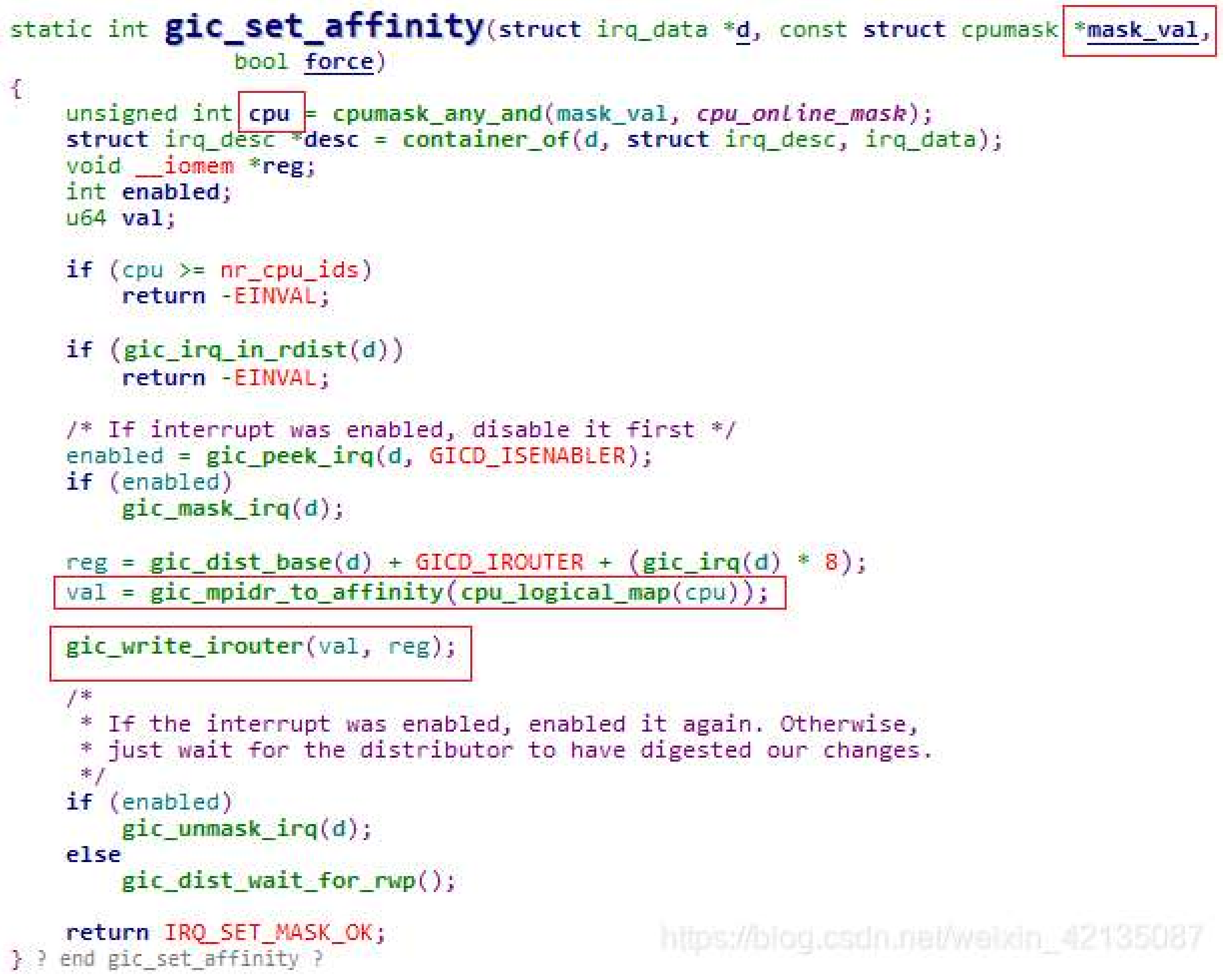
- GICD_IROUTERn.Interrupt_Routing_Mode == 1 Distributor硬件会自动选择一个PE，可以是0-n
  A PE can opt out of receiving 1-of-N interrupts. This is controlled by the DPG1S, DPG1NS and DPG0 bits in GICR_CTLR.

GICD_IROUTER<n>, Interrupt Routing Registers, n = 32 - 1019





```
static void __init gic_dist_init(void)
{
    unsigned int i;
    u64 affinity;
    void __iomem *base = gic_data.dist_base;

    /* Disable the distributor */
    writel_relaxed(0, base + GICD_CTLR);
    gic_dist_wait_for_rwp();

    /*
     * Configure SPIs as non-secure Group-1. This will only matter
     * if the GIC only has a single security state. This will not
     * do the right thing if the kernel is running in secure mode,
     * but that's not the intended use case anyway.
     */
    for (i = 32; i < gic_data.irq_nr; i += 32)
        writel_relaxed(~0, base + GICD_IGROUPR + i / 8);

    gic_dist_config(base, gic_data.irq_nr, gic_dist_wait_for_rwp);

    /* Enable distributor with ARE, Group1 */
    writel_relaxed(GICD_CTLR_ARE_NS | GICD_CTLR_ENABLE_G1A | GICD_CTLR_ENABLE_G1,
                   base + GICD_CTLR);

    /*
     * Set all global interrupts to the boot CPU only. ARE must be
     * enabled.
     */
    affinity = gic_mpidr_to_affinity(cpu_logical_map(smp_processor_id()));
    for (i = 32; i < gic_data.irq_nr; i++)
        gic_write_irouter(affinity, base + GICD_IROUTER + i * 8);
} ? end gic_dist_init ?
```

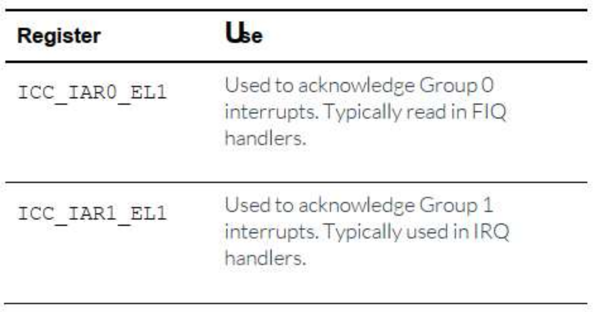


```
static int gic_set_affinity(struct irq_data *d, const struct cpumask *mask_val,
                            bool force)
{
    unsigned int cpu = cpumask_any_and(mask_val, cpu_online_mask);
    struct irq_desc *desc = container_of(d, struct irq_desc, irq_data);
    void __iomem *reg;
    int enabled;
    u64 val;

    if (cpu >= nr_cpu_ids)
        return -EINVAL;

    if (gic_irq_in_rdist(d))
        return -EINVAL;

    /* If interrupt was enabled, disable it first */
    enabled = gic_peek_irq(d, GICD_ISENABLER);
    if (enabled)
        gic_mask_irq(d);

    reg = gic_dist_base(d) + GICD_IROUTER + (gic_irq(d) * 8);
    val = gic_mpidr_to_affinity(cpu_logical_map(cpu));

    gic_write_irouter(val, reg);

    /*
     * If the interrupt was enabled, enabled it again. Otherwise,
     * just wait for the distributor to have digested our changes.
     */
    if (enabled)
        gic_unmask_irq(d);
    else
        gic_dist_wait_for_rwp();

    return IRQ_SET_MASK_OK;
} ? end gic_set_affinity ?
```

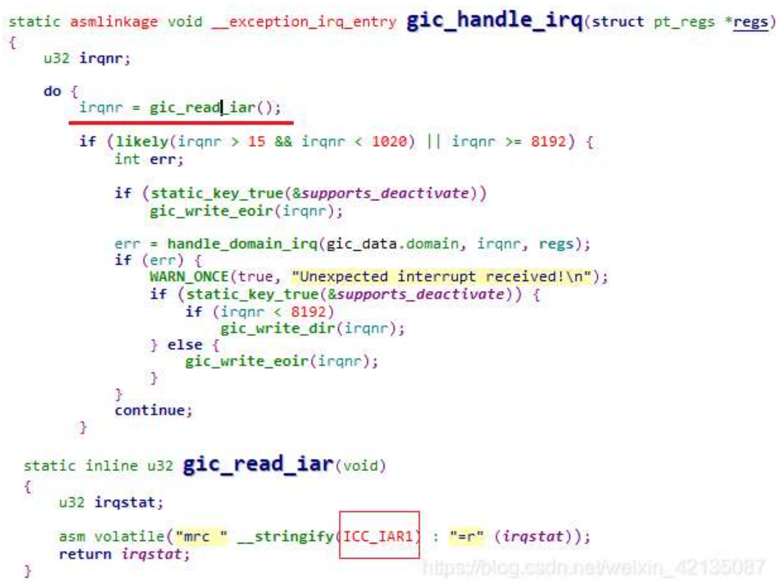
Routing a pending interrupt to a PE

- Check that the group associated with the interrupt is enabled
- Check that the interrupt is enabled
- Check the routing controls to decide which PEs can receive the interrupt.
  routing is controlled by GICD_IROUTERn，An SPI can target one specific PE, or any one of the connected PEs
- Check the interrupt priority and priority mask to decide which PEs are suitable to handle the interrupt
  Each PE has a Priority Mask register, ICC_PMR_EL1, in its CPU interface
- Check the running priority to decide which PEs are available to handle the interrup
  Only an interrupt with a higher priority than the running priority can preempt the current interrupt

软件读取中断号

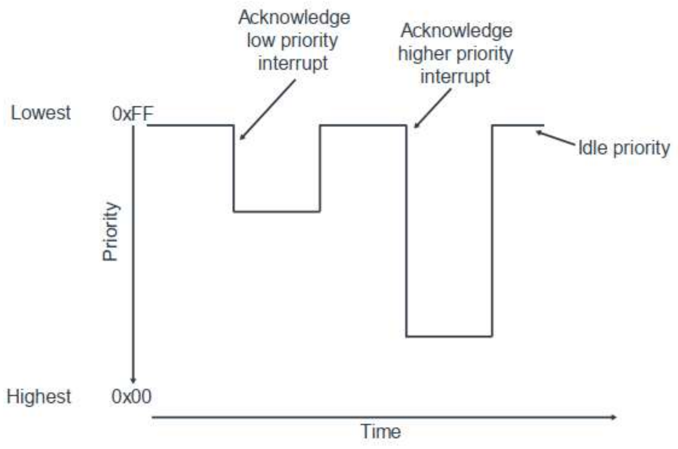


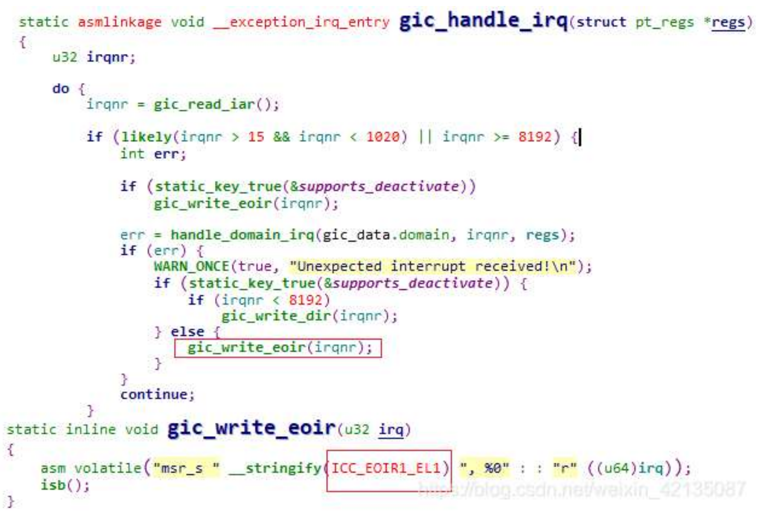
Register	Use
ICC_IAR0_EL1	Used to acknowledge Group 0 interrupts. Typically read in FIQ handlers.
ICC_IAR1_EL1	Used to acknowledge Group 1 interrupts. Typically used in IRQ handlers.



```
static asmlinkage void __exception_irq_entry gic_handle_irq(struct pt_regs *regs)
{
    u32 irqnr;

    do {
        irqnr = gic_read_iar();

        if (likely(irqnr > 15 && irqnr < 1020) || irqnr >= 8192) {
            int err;

            if (static_key_true(&supports_deactivate))
                gic_write_eoir(irqnr);

            err = handle_domain_irq(gic_data.domain, irqnr, regs);
            if (err) {
                WARN_ONCE(true, "Unexpected interrupt received!\n");
                if (static_key_true(&supports_deactivate)) {
                    if (irqnr < 8192)
                        gic_write_dir(irqnr);
                } else {
                    gic_write_eoir(irqnr);
                }
            }
            continue;
        }

static inline u32 gic_read_iar(void)
{
    u32 irqstat;

    asm volatile("mrc " __stringify(ICC_IAR1) : "=r" (irqstat));
    return irqstat;
}
```

中断优先级


Acknowledge low priority interrupt
Acknowledge higher priority interrupt
Lowest
0xFF
Idle priority
Priority
Highest
0x00
Time



```
static asmlinkage void __exception_irq_entry gic_handle_irq(struct pt_regs *regs)
{
    u32 irqnr;

    do {
        irqnr = gic_read_iar();

        if (likely(irqnr > 15 && irqnr < 1020) || irqnr >= 8192) {
            int err;

            if (static_key_true(&supports_deactivate))
                gic_write_eoir(irqnr);

            err = handle_domain_irq(gic_data.domain, irqnr, regs);
            if (err) {
                WARN_ONCE(true, "Unexpected interrupt received!\n");
                if (static_key_true(&supports_deactivate)) {
                    if (irqnr < 8192)
                        gic_write_dir(irqnr);
                } else {
                    gic_write_eoir(irqnr);
                }
            }
            continue;
        }
static inline void gic_write_eoir(u32 irq)
{
    asm volatile("msr_s " __stringify(ICC_EOIR1_EL1) ", %0" : : "r" ((u64)irq));
    isb();
}
```

中断结束End of interrupt

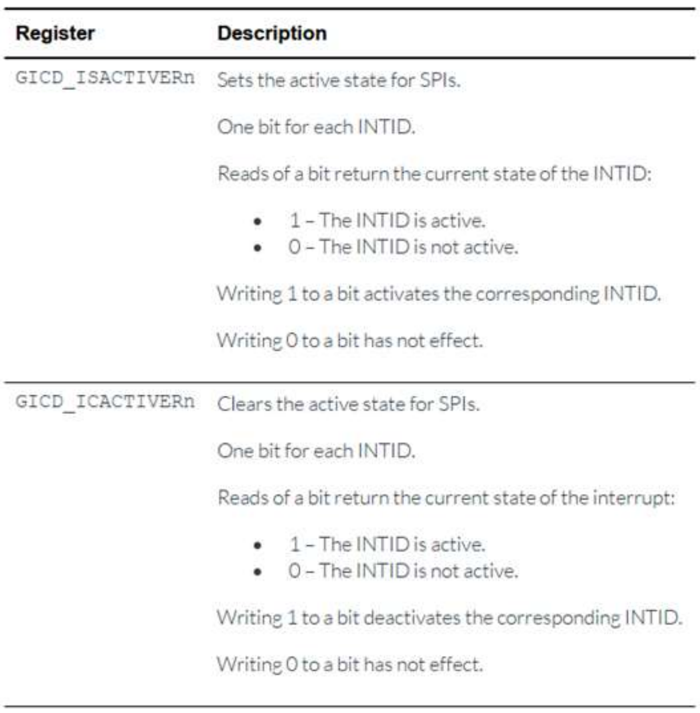
- Priority drop - 将中断优先级降到中断产生之前的值
- Deactivation - 将中断从active变成inactive

在gicv3中，drop和deactivation通常是一起打开的。

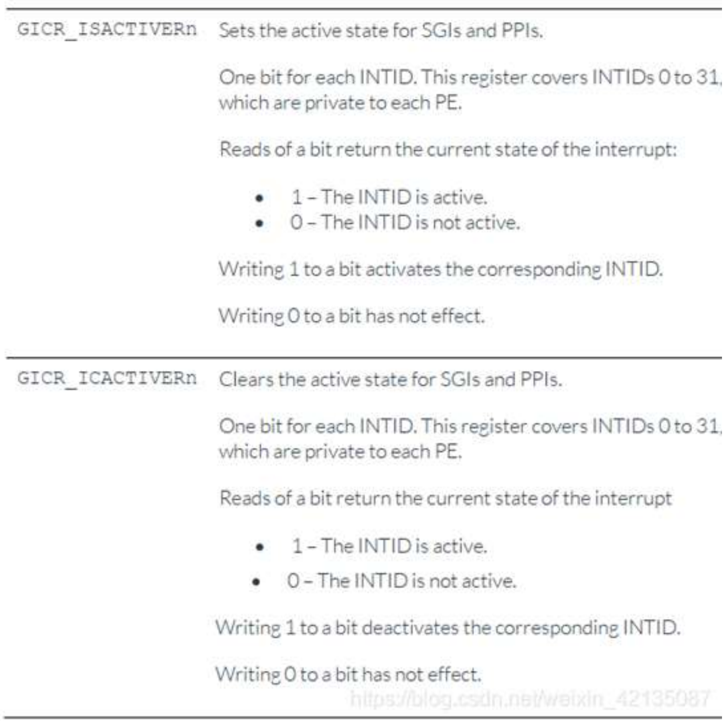
ICC_CTLR_ELn.EOImode = 1: 通过写ICC_EOIR0_EL1、ICC_EOIR1_EL1让drop and deactivation同时生效
ICC_CTLR_ELn.EOImode = 0: 通过写ICC_EOIR0_EL1、ICC_EOIR1_EL1让drop生效，写ICC_DIR_EL1让deactivation生效，这在虚拟化中会用到.

大多数的软件系统中 EOIMode0，而下hypervisor的系统中 EOIMode1

中断号的状态

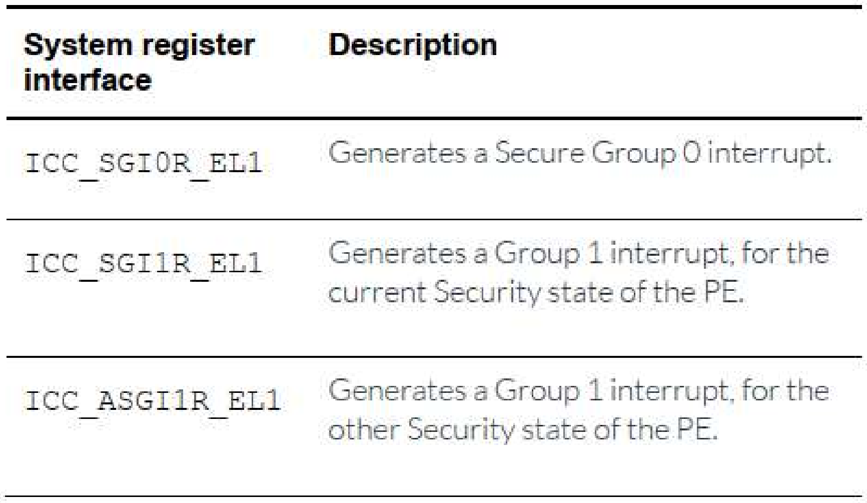


Register	Description
GICD_ISACTIVERn	Sets the active state for SPIs. One bit for each INTID. Reads of a bit return the current state of the INTID: • 1 – The INTID is active. • 0 – The INTID is not active. Writing 1 to a bit activates the corresponding INTID. Writing 0 to a bit has not effect.
GICD_ICACTIVERn	Clears the active state for SPIs. One bit for each INTID. Reads of a bit return the current state of the interrupt: • 1 – The INTID is active. • 0 – The INTID is not active. Writing 1 to a bit deactivates the corresponding INTID. Writing 0 to a bit has not effect.

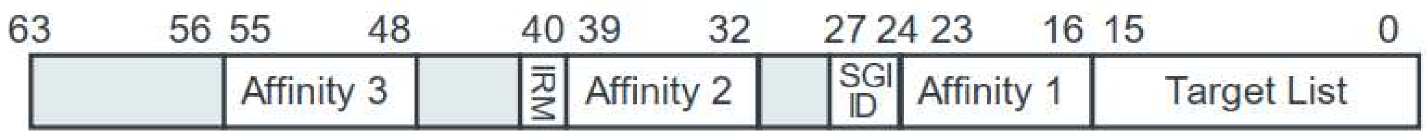


GICR_ISACTIVERn	Sets the active state for SGIs and PPIs. One bit for each INTID. This register covers INTIDs 0 to 31, which are private to each PE. Reads of a bit return the current state of the interrupt: • 1 – The INTID is active. • 0 – The INTID is not active. Writing 1 to a bit activates the corresponding INTID. Writing 0 to a bit has not effect.
GICR_ICACTIVERn	Clears the active state for SGIs and PPIs. One bit for each INTID. This register covers INTIDs 0 to 31, which are private to each PE. Reads of a bit return the current state of the interrupt • 1 – The INTID is active. • 0 – The INTID is not active. Writing 1 to a bit deactivates the corresponding INTID. Writing 0 to a bit has not effect.

产生SGI中断



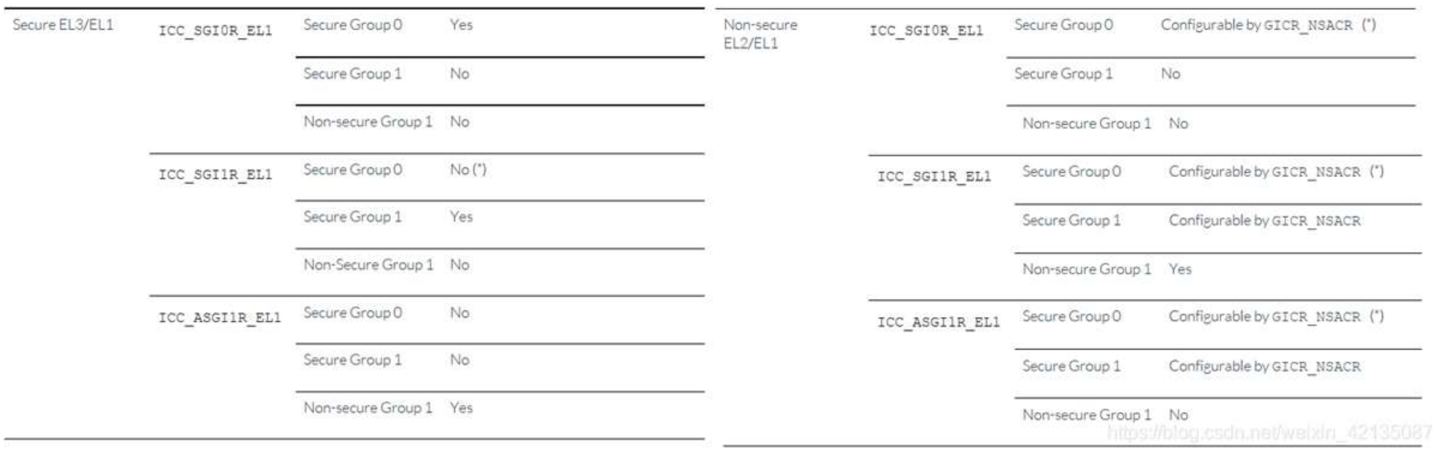
System register interface	Description
ICC_SGI0R_EL1	Generates a Secure Group 0 interrupt.
ICC_SGI1R_EL1	Generates a Group 1 interrupt, for the current Security state of the PE.
ICC_ASGI1R_EL1	Generates a Group 1 interrupt, for the other Security state of the PE.



IRM (Interrupt Routing Mode)
IRM = 0 rounting到Affinity指定的PE;
IRM = 1 routing到所有PE(除当前PE之外)

PE在secure执行时，可以产生secure和non-secure的SGI;
PE在non-secure执行时，也是可以产生secure的SGI，但是取决于GICR_NSACR寄存器的配置，该寄

存器只能在secure中读写



Secure EL3/EL1	ICC_SGI0R_EL1	Secure Group 0	Yes
		Secure Group 1	No
		Non-secure Group 1	No
	ICC_SGI1R_EL1	Secure Group 0	No (*)
		Secure Group 1	Yes
		Non-Secure Group 1	No
	ICC_ASGI1R_EL1	Secure Group 0	No
		Secure Group 1	No
		Non-secure Group 1	Yes
Non-secure EL2/EL1	ICC_SGI0R_EL1	Secure Group 0	Configurable by GICR_NSACR (*)
		Secure Group 1	No
		Non-secure Group 1	No
	ICC_SGI1R_EL1	Secure Group 0	Configurable by GICR_NSACR (*)
		Secure Group 1	Configurable by GICR_NSACR
		Non-secure Group 1	Yes
	ICC_ASGI1R_EL1	Secure Group 0	Configurable by GICR_NSACR (*)
		Secure Group 1	Configurable by GICR_NSACR
		Non-secure Group 1	No

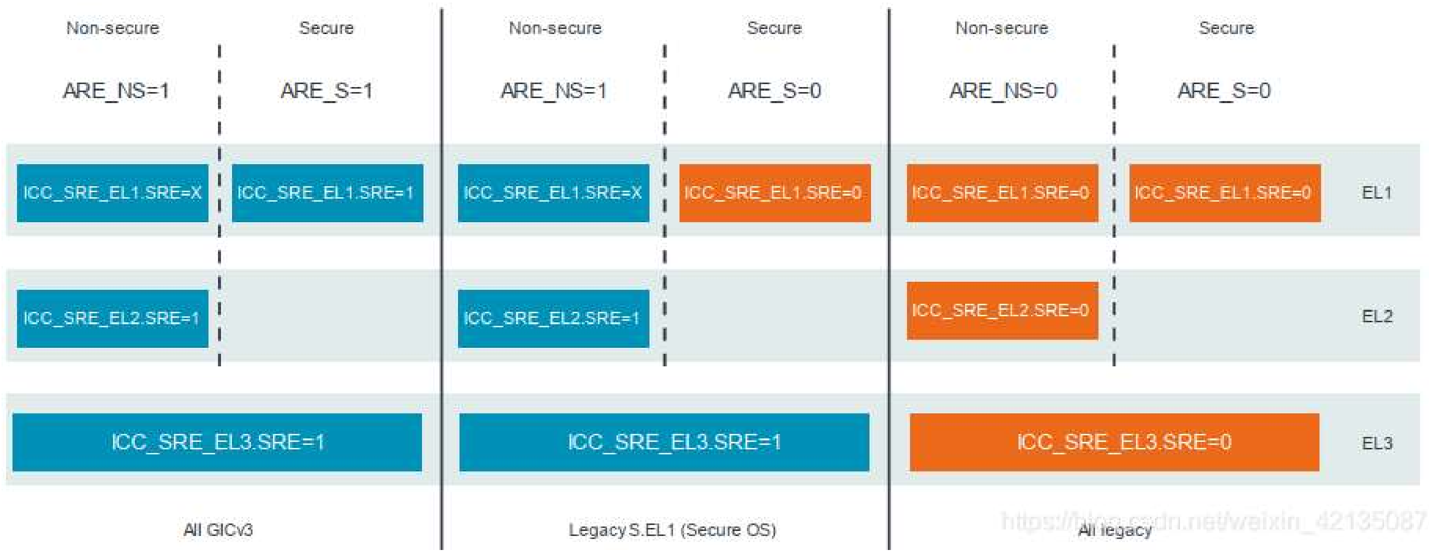
比较GICv3和GICv2

在gicv2中，SGI INTIDs对于originating PE和the target PE是banked
在gicv3中，SGI仅仅对target PE是banked

在gicv2中同时收到两个SGI=5中断，两个中断都会被PE处理。
而在gicv3上，由于originating不是banked，所有前一个SGI=5中断将会丢失。PE只能收到一个

Legacy operation

- **When ARE0, affinity routing is disabled (legacy operation)**
- **When ARE1, affinity routing is enabled (GICv3 operation)**



社群

ARM-TEE-ATF-SOC群(一)	T03-Arm二期课程交流群
ARM-TEE-ATF-SOC群(六)	T02-Arm一期课程交流群
ARM-TEE-ATF-SOC群(七)	T04-Trustzone/TEE课程交流群
ARM-TEE-ATF-SOC群(三)	T06-Secureboot课程交流群1299
ARM-TEE-ATF-SOC群(五)	T01-Arm一期课程交流群CSDN
ARM-TEE-ATF-SOC群(四)	T05-Secureboot课程交流群499
ARM-TEE-ATF-SOC群(二)	PAC&BTI&MTE三剑客课程群

（想进社区的加v：arm2023，备注：CSDN进群）

Armv8/Armv9架构从入门到精通，Armv8/Armv9架构从入门到精通（一期），Armv8/Armv9架构从入门到精通（二期）Armv8/Armv9架构从入门到精通（三期），Arm一期、Arm二期、学习资料、免费、下载，全套资料，Secureboot从入门到精通，secureboot训练营，ATF架构从入门到精通、optee系统精讲、secureboot精讲，Trustzone/TEE/安全快速入门班，Trustzone/TEE/安全标准版，Trustzone/TEE/安全高配版。全套资料。周贺贺，baron，代码改变世界，coding_the_world，Arm精选，arm_2023，安全启动，加密启动

optee、ATF、TF-A、Trustzone、optee3.14、MMU、VMSA、cache、TLB、arm、armv8、armv9、TEE、安全、内存管理、页表，Non-cacheable,Cacheable, non-shareable,inner-shareable,outer-shareable, optee、ATF、TF-A、Trustzone、optee3.14、MMU、VMSA、cache、TLB、arm、armv8、armv9、TEE、安全、内存管理、页表…