RISC-V 高级中断架构

版本 1.0

原文: John Hauser
jh.riscv@jhauser.us

2023 年 6 月 30 日

译者：Song Cunjie
songcunjie@163.com
仅供学习交流使用

本规范所有版本的贡献者按字母顺序排列（请联系编辑提出更正建议）：Krste Asanovi´c、Paul Donahue、Greg Favor、John Hauser、James Kenney、David Kruckemyer、Shubu Mukherjee、Stefan O'Rear、Vernon Pang、Anup Patel、Josh Scheid、Ved Shanbhogue 和 Andrew Waterman。

序言

本文档描述了 RISC-V 系统的高级中断架构。本规范于 2023 年 6 月获得 RISC-V 国际协会批准。

下表显示了本文档中哪些章节规定了 RISC-V ISA（指令集架构）的扩展，哪些章节规定了非 ISA。

章节	ISA?
1.导言	-
2.为 Harts 添加控制和状态寄存器（CSR）	是
3.Incoming MSI 控制器（IMSIC）	是
4.高级平台级中断控制器（APLIC）	没有
5.Machine 和 supervisor-level 中断	是
6.虚拟机中断（VS 级）	是
7.处理器间中断（IPI）	没有
8.IOMMU 对虚拟机 MSI 的支持	没有

对批准的 1.0 版本的修改

解决了第 2 章中关于何时引发虚拟指令异常与非法指令异常的不一致问题。

RC5 的变化（批准候选 5）

使间接访问寄存器的规则与 hypervisor extension 和即将推出的 Smcsrind/Sscsrind 扩展更加一致。特别是当 vsiselect 具有保留值时，从虚拟机（VS 或 VU-模式）访问 sireg 的尝试最好引发非法指令异常，而不是虚拟指令异常。

添加了对第 8 章中使用的 *IOMMU* 一词的说明。

添加了关于 MSI 写入被 MRIF 更新替换以及更新后 MSI 发送通知的说明。

针对 RC4 的更改

为了与其他即将推出的 RISC-V ISA 扩展保持一致，间接访问 CSR、miselect、mireg、siselect、sireg、vsiselect 和 vsireg 的宽度都改为当前的 XLEN，而不是与各自的权限级别挂钩（以前的 MXLEN 用于 miselect 和 mireg，SXLEN 用于 siselect 和 sireg，VSXLEN 用于 vsiselect 和 vsireg）。

修改了 *high-half* CSR 及其伙伴 CSR 的描述（但不是实际功能），以符合最新的 RISC-V 特权 ISA 规范。(high-half CSR 的示例是 miph，其伙伴是 mip）。

RC3 的更改

将仍在起草中的 Duo-PLIC 章节移至另一份文件。

为 RAS 事件信号分配了主要中断 35 和 43（第 5.1 节）。

在第 5.3 节中，增加了 CSR mvien 中可写入第 1 位和第 9 位的选项，并指定了设置每个位的效果。

将第 8 章（"IOMMU 支持"）升级到*冻结*状态。

RC2 的更改

澄清了 CSR hvictl 的字段 IID 必须支持该字段实现的所有无符号整数位数，并且写入 hvictl 时总是以最直接的方式设置 IID。

在第 7 章中添加了一条注释，警告当 IPI 通过向其他硬盘的 IMSIC 写入 MSI 发送到这些硬盘时，可能需要 FENCE 指令。

目录

RISC-V 高级中断架构..... 1

序言 3

对批准的 1.0 版本的修改..... 3

RC5 的变化（批准候选 5）..... 3

针对 RC4 的更改..... 2

RC3 的更改 2

RC2 的更改 2

目录 3

1.导言 9

1.1目标 9

1.2限制 2

1.3主要组件概览..... 3

1.3.1 无 IMSIC 的外部中断 3

1.3.2 使用 IMSIC 的外部中断..... 5

1.3.3 其他中断..... 5

1.4Hart 的中断标识..... 6

1.5选择接收中断的 Harts **7**

1.6ISA 扩展名 Smaia 和 Ssaia **8**

2.添加到 Harts 的控制和状态寄存器 (CSR) 9

2.1Machine-level CSR **9**

2.2Supervisor-level CSR **10**

2.3Hypervisor 和 VS CSR **11**

2.4虚拟指令异常 **13**

2.5通过 state-enable CSR 进行访问控制 **14**

3.Incoming MSI 控制器（IMSIC） 16

3.1中断文件和中断标识 **16**

3.2MSI 编码 **17**

3.3中断优先级 **18**

3.4重置和显示状态 **18**

3.5中断文件的内存区域 **18**

3.6多个中断文件的内存区域的安排 **19**

3.7通过 IMSIC 发起外部中断相关的 CSR **21**

3.8间接访问的中断文件寄存器 **22**

3.8.1 外部中断发送使能寄存器（eidelivery） **22**

3.8.2 外部中断使能阈值寄存器（eithreshold） **22**

3.8.3 外部中断挂起寄存器（eip0-eip63）.....22

3.8.4 外部中断使能寄存器（eie0-eie63）.....23

3.9Top 外部中断 CSR（mtopei、stopei、vstopei）.....23

3.10 中断交付和处理.....24

4.高级平台级中断控制器（APLIC）.....25

4.1中断源和标识.....26

4.2中断域.....26

4.3Hart 索引编号.....29

4.4单域中断控制概述.....29

4.5中断域的内存映射控制区.....29

4.5.1 域配置 (domaincfg).....31

4.5.2 源配置 (sourcecfg[1]-sourcecfg[1023]).....32

4.5.3 Machine MSI 地址配置（mmsiaddrcfg 和 mmsiaddrcfgh）.....33

4.5.4 Supervisor MSI 地址配置（smsiaddrcfg 和 smsiaddrcfgh）.....35

4.5.5 设置中断挂起位（setip[0]-setip[31].....36

4.5.6 按编号设置中断挂起位（setipnum）.....36

4.5.7 校正后输入值，清除中断挂起位（in_clrip[0]-in_clrip[31]）.....36

4.5.8 按编号清除中断挂起位（clripnum）.....36

4.5.9 设置中断使能位（setie[0]-setie[31]）.....37

4.5.10 按数字设置中断使能位（**setienum**） **37**

4.5.11 清除中断使能位（**clrie[0]-clrie[31]** **37**

4.5.12 按编号清除中断使能位 **(clrienum)** **37**

4.5.13 按数字设置中断挂起位，小数点后一位（**setipnum_le**） **37**

4.5.14 按数字设置中断挂起位，**big-endian**（**setipnum_be**） **37**

4.5.15 生成 **MSI (genmsi)** **38**

4.5.16 中断目标（**target[1]-target[1023]**） **38**

4.6 重置 **39**

4.7 对中断挂起位的精确影响 **40**

4.8 由 **APLIC** 直接中断交付 **40**

4.8.1 中断发送控制（**IDC**）结构 **41**

4.8.2 中断交付和处理 **42**

4.9MSI 的中断转发 **43**

4.9.1 发送 **MSI** 的地址和数据 **43**

4.9.2 对电平敏感中断源的特殊考虑 **44**

4.9.3 实现 **Hart** 与 **APLIC** 之间的同步互动 **44**

5.Machine 和 Supervisor level 中断 45

5.1 已定义的主要中断和默认优先级 **45**

5.2Machine-level 中断 **58**

5.2.1 在 machine 层面配置主要中断的优先级 58

5.2.2 Machine top 中断 CSR (mtopi) 60

5.3 用于 supervisor-level 的中断过滤和虚拟中断 61

5.4 Supervisor-level 中断 63

5.4.1 在 supervisor-level 配置主要中断的优先级 63

5.4.2 Supervisor 最高中断 CSR (stopi) 64

5.5 WFI（等待中断）指令 65

6.虚拟机中断（VS 级） 67

6.1 使用 guest 中断文件的 VS 级外部中断 67

6.1.1 Guest 操作系统直接控制设备 68

6.1.2 将虚拟主机迁移到不同的 guest 中断文件 68

6.2 无需 guest 中断文件的 VS 级外部中断 69

6.3 VS 级中断 69

6.3.1 在 VS 级配置主要中断的优先级 69

6.3.2 VS 级虚拟中断 71

6.3.3 虚拟监控器最高中断 CSR (vstopi) 73

6.3.4 中断陷阱转为 VS 模式 73

7.处理器间中断 (IPI) 79

8.IOMMU 对虚拟机 MSI 的支持 81

8.1 IOMMU 的设备上下文 82

8.2 设备 MSI 地址转换 82

8.3 内存驻留中断文件 83

8.3.1 内存驻留中断文件格式 84

8.3.2 将收到的 MSI 记录到内存驻留中断文件中 85

8.3.3 使用内存驻留中断文件进行原子更新 86

8.3.4 使用内存驻留中断文件，无需原子更新 87

8.3.5 为接收通知 MSI 分配 guest 中断文件 87

8.4 识别虚拟机中断文件的页地址 88

8.5 MSI 页表 88

8.5.1 MSI PTE，基本转换模式 89

8.5.2 MSI PTE，MRIF 模式 90

第 1 章

1.导言

本文件规定了 RISC-V 的高级中断体系结构，包括：(a) 《RISC-V 指令集手册》第二卷中规定的 RISC-V 硬件标准特权体系结构的扩展；(b) RISC-V 系统的两个标准中断控制器，即高级平台级中断控制器 (APLIC) 和传入消息信号中断控制器 (IMSIC)；以及 (c) 有关中断的其他系统组件的要求。

关于我们的设计决定、实现方案和应用的评论，格式如本段所示，如果读者只对规范本身感兴趣，可以跳过。

1.1 目标

RISC-V 高级中断架构具有这些目标：

- 以 RISC-V 专用架构的中断处理功能为基础，尽量减少对现有功能的替换。
- 除了基本的有线中断外，还为 RISC-V 系统提供直接使用 PCI Express 和其他设备标准所采用的消息信号中断 (MSI) 的功能。
- 对于有线中断，定义一个新的平台级中断控制器（高级 PLIC 或 APLIC），该控制器为每一级权限（如 RISC-V machine-level 和 supervisor-level）提供独立的控制接口，并可将有线中断转换为支持 MSI 的系统的 MSI。
- 扩展 RISC-V 核心的本地中断框架。
- 可选择允许软件配置 RISC-V 硬件的所有中断源（包括标准定时器和软件中断等）的相对优先级，而不是仅限于由独立的中断控制器来确定外部中断的优先级。

- 当 harts 实现特权架构的 hypervisor extension 时，应为虚拟机虚拟化这些相同的中断设施提供足够的帮助。
- 借助用于重定向 MSI 的 IOMMU（I/O 内存管理单元），最大限度地提高在虚拟机中运行的 guest 操作系统直接控制设备的机会和能力，最大限度地减少 hypervisor 的参与。
- 避免让中断硬件成为虚拟机数量的限制因素。
- 在速度、效率和实现灵活性之间做出最佳妥协，实现上述所有目标。

高级中断体系结构的初始版本主要针对大型高性能 RISC-V 系统的需求。目前尚未定义对以下中断处理功能的支持，这些功能在所谓的 "实时 "系统中对最大限度缩短中断响应时间非常有用，但对高速处理器内核却不太合适：

- 为每个中断源提供单独的陷阱入口地址；
- 在进入中断陷阱时自动堆叠寄存器值，并在退出时恢复；以及
- 根据优先级自动抢占（嵌套）中断。

针对较小和/或实时系统的优化功能可作为后续扩展进行开发，可单独开发，也可作为本文件中断架构未来版本的一部分。

1.2 限制

在当前版本中，RISC-V 高级中断架构可支持多达 16,384 个硬件单元的 RISC-V 对称多处理（SMP）系统。如果 Hart 为 64 位 (RV64)，并实现了 hypervisor extension，而且高级中断架构的所有功能也已完全实现，那么每个物理 Hart 可能有多达 63 个活动的虚拟 Hart 和数千个额外的空闲（交换出）虚拟 Hart，其中每个虚拟 Hart 可直接控制一个或多个物理设备。

表 1.1 概述了高级中断架构支持的物理和虚拟 Harts 数量以及不同中断标识数量的主要限制。

我们假定，任何一台 RISC-V 计算机（或集群或分布式系统中的任何单个节点）如果有成千上万个物理 Harts，都可能需要一个中断基础架构来适应机器的特定组织结构，而我们并不试图预测这一点。

	最大	要求
物理 harts	16,384	
对于每一个物理 hart，可直接控制一台设备的活跃虚拟 harts	31 for RV32 63 for RV64	RISC-V hypervisor extension 带有 guest 中断文件的 IMSIC；以及一个 IOMMU
对于每一个物理 hart，可直接控制一台设备的闲置（已交换）的虚拟Hart	可能成千上万	支持内存驻留中断文件的 IOMMU
单个 APLIC 的有线中断	1023	
每个 hart（物理 hart 或虚拟 hart）的 MSI 可使用的不同的标识	2047	IMSIC

表 1.1：表 1.1：系统中 Hart 和中断标识数量的绝对限制。个别实现可能有更小的限制。

1.3 主要组件概览

RISC-V 系统发出中断信号的整体架构取决于它是主要为消息信号中断（MSI）而建，还是为更传统的有线中断而建。在完全支持 MSI 的系统中，每个 hart都有一个 *Incoming MSI 控制器*（IMSIC），作为 hart 自己的专用中断控制器来处理外部中断。相反，在主要基于传统有线中断的系统中，hart没有 IMSIC。大型系统，尤其是那些带有 PCI 设备的系统，有望通过为 hart 提供 IMSIC 来完全支持 MSI，而许多小型系统的最佳选择仍然是有线中断和不带 IMSIC 的简单 hart。

1.3.1 无 IMSIC 的外部中断

当 RISC-V 硬件没有 Incoming MSI 控制器时，外部中断会通过专用线路向硬件发出信号。在这种情况下，*高级平台级中断控制器* (APLIC) 充当了传统的中断中心枢纽，如图 1.1 所示，为每个 Hart 路由外部中断并确定其优先级。中断可有选择地路由到 machine-level，或路由到每个 hart 的 supervisor level。APLIC 在第 4 章中有具体说明。

由于没有 IMSIC，当前的高级中断架构不支持向虚拟机直接发送外部中断信号，即使 RISC-V 硬件实现了特权架构的 hypervisor extension 也是如此。相反，中断必须发送到相关的 hypervisor，然后 hypervisor 可以选择向虚拟机注入虚拟中断。

如果 Harts 实现了 hypervisor extension，那么是否应允许 APLIC 将外部中断路由为 hypervisor extension 的 guest 外部中断，从而允许直接向虚拟机发送中断，而无需在 hypervisor 级别处理每个信号中断，这是一个正在研究的课题。目前，我们假定需要直接向虚拟机发送外部中断信号的系统都将使用 IMSIC。

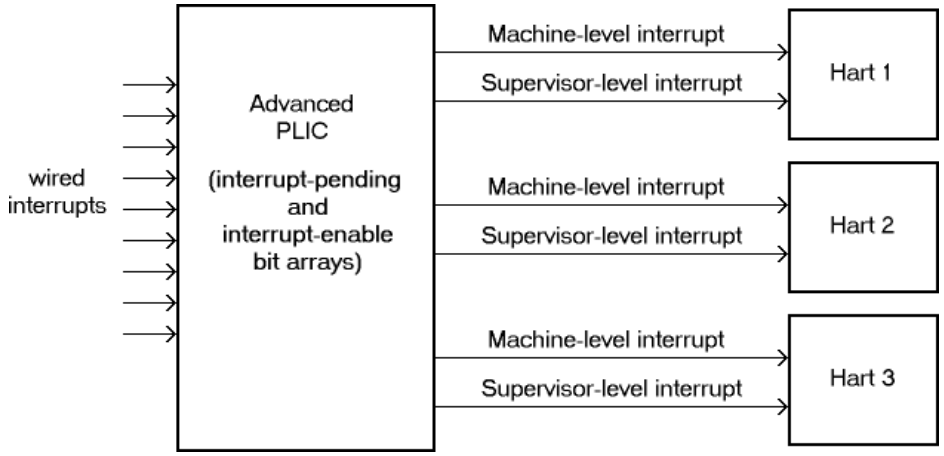

图 1.1：向不支持 MSI 的 Harts 发送有线中断的传统方式。

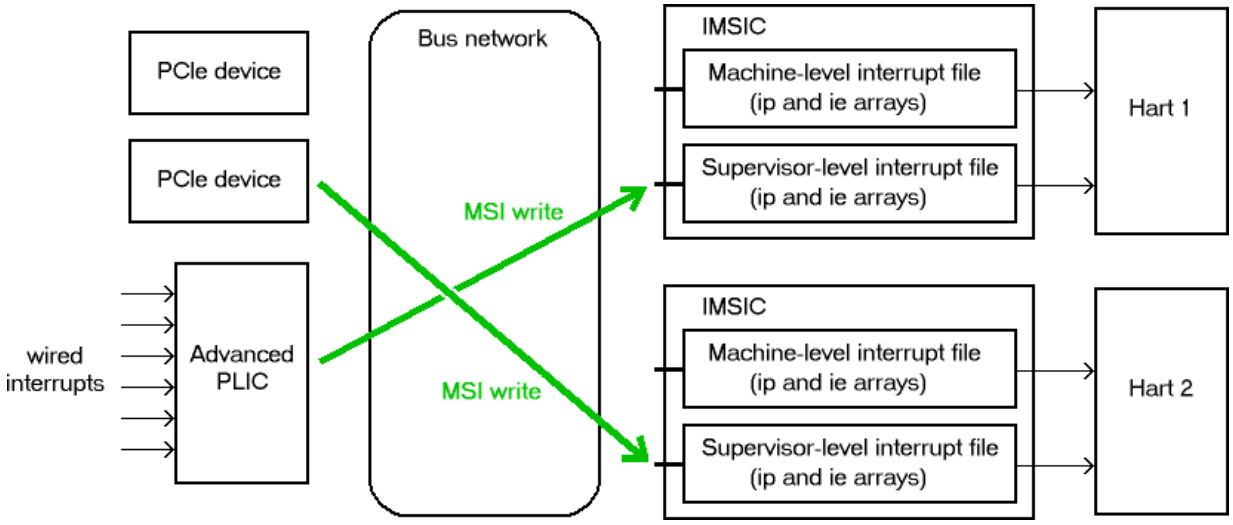

图 1.2：当 Harts 有 IMSIC 用于接收中断时，MSI 发送的中断。

1.3.2 使用 IMSIC 的外部中断

如图 1.2 所示，为了能够接收消息信号中断 (MSI)，每个 RISC-V 硬件都必须有一个Incoming MSI 控制器 (IMSIC)。从根本上说，消息信号中断只是向特定地址的内存写入，硬件将其视为指示中断。为此，每个 IMSIC 都会在 machine 的地址空间中分配一个或多个不同的地址，当以预期格式向其中一个地址写入时，接收的 IMSIC 会将该写入解释为相应 hart 的外部中断。

由于所有 IMSIC 在 machine 的物理地址空间中都有唯一的地址，因此每个 IMSIC 都可以从任何有写入权限的代理（hart 或设备）接收 MSI 写入。IMSIC 为针对 machine 和 supervisor-level 的 MSI 提供单独的地址，部分原因是为了通过控制不同地址的写入权限来分别授予或拒绝每个权限级别的中断信号能力，部分原因是为了更好地支持虚拟性（假装一个权限级别是更高级别）。针对特定权限级别 hart 的 MSI 记录在 IMSIC 的*中断文件*中，该文件主要由一个中断挂起位数组和一个中断使能位数组组成，后者表示 hart 当前准备接收哪些单个中断。

第 3 章对 IMSIC 单元进行了全面定义。RISC-V 高级中断架构使用的 MSI 格式在该章第 3.2 节中进行了描述。

当 RISC-V 系统中的 hart 具有 IMSIC 时，系统通常仍包含 APLIC，但其作用有所改变。APLIC 不再像图 1.1 中那样直接通过信号线向 hart 阵列发送中断信号，而是将传入的有线中断信号转换为 MSI 写入信号，通过 IMSIC 单元发送给 hart 阵列。每个 MSI 都会根据软件设置的 APLIC 配置发送到一个目标 Hart。

如果 RISC-V 硬件实现了特权架构的 hypervisor extension，IMSIC 可能会有额外的 *guest 中断文件，*用于向虚拟机提供中断。除有关 IMSIC 的第 3 章外，请参见第 6 章，该章专门介绍了向虚拟机提供中断的问题。如果系统还包含一个 IOMMU，用于执行 I/O 设备内存访问的地址转换，那么来自这些设备的 MSI 可能需要特殊处理。第 8 章 "IOMMU 对虚拟机 MSI 的支持 "将讨论这一主题。

1.3.3 其他中断

除了来自 I/O 设备的外部中断外，RISC-V 高级架构还为 Harts 指定了其他几类主要中断。尽管图 1.1 和图 1.2 中没有显示，但特权体系结构的定时器中断仍然得到全面支持，软件中断也至少得到部分支持。有关软件中断的详细信息，请参阅第 7 章 "处理器间中断 (IPI)"。

高级中断体系结构增加了对 hart *本地中断*的大量支持，hart 可根据异步事件（通常是错误）自行中断。本地中断仍包含在 hart（或接近 hart）中，因此与标准 RISC-V 定时器和软件中断一样，它们不会通过 APLIC 或 IMSIC。

1.4 Hart 的中断标识

RISC-V 特权架构为每个中断原因提供了一个不同的*主要标识号*，即在中断陷阱中自动写入 CSR mcause 或 scause 的异常代码。由特权架构标准化的中断原因的主要标识号范围为 0-15，而 16 及更高的标识号则可正式用于平台标准或定制用途。高级中断架构对 16-23 和 32-47 范围内的标识号拥有更多权限，而 24-31 范围内的标识号和 48 及以上的所有主要标识号仍可自由定制使用。表 1.2 列出了具有此扩展功能的所有主要中断标识。

主要标识	次要标识	
0	-	*由特权架构保留*
1	-	Supervisor 软件中断
2	-	Virtual supervisor软件中断
3	-	Machine 软件中断
4	-	*由特权架构保留*
5	-	Supervisor 计时器中断
6	-	Virtual supervisor 定时器中断
7	-	Machine 定时器中断
8	-	*由特权架构保留*
9	由外部中断控制器决定	Supervisor 外部中断
10		Virtual supervisor 外部中断
11		Machine 外部中断
12	-	Supervisor guest 外部中断
13	-	计数器溢出中断
14-15	-	*由特权架构保留*
16-23	-	*保留用于标准本地中断*
24-31	-	*指定用于定制用途*
32-34	-	*保留用于标准本地中断*
35	-	低优先级 RAS 事件中断
36-42	-	*保留给标准本地中断*
43	-	高优先级 RAS 事件中断
44-47	-	*保留给标准本地中断*
≥ 48	-	*指定用于定制用途*

表 1.2： Hart 所有中断原因的主要和次要标识。主要标识 0-15 属于 RISC-V 特权架构的权限范围。

大多数 I/O 设备的中断都是由 Hart 的*外部中断控制器*（即 Hart 的 IMSIC（图 1.2）或 APLIC（图 1.1））传送到 Hart 的。如表 1.2 所示，特定权限级别的外部中断都共享一个主要标识：machine-level 为 11，supervisor level 为 9，VS 级为 10。来自不同原因的外部中断可通过外部中断控制器提供的*次要标识*相互区分。

除外部中断外，其他中断原因也可能有自己的次要标识。不过，本文只需要讨论外部中断的次要标识。

高级中断架构定义的本地中断及其处理方法主要在第 5 章 "machine 和 supervisor-level 中断 "中介绍。

1.5 选择接收中断的 Harts

每个发出信号的中断只发送到一个权限级别的 Hart，通常由软件以某种方式决定。与其他一些体系结构不同，RISC-V 高级中断体系结构不提供向多个 Hart 广播或多播中断的标准硬件机制。

对于本地中断，以及软件注入到 Hart 中较低权限级别的任何 "虚拟 "中断，中断完全是 Hart 的本地事务，其他 Hart 永远看不到。RISC-V 特权体系结构的定时器中断也是与单个中断绑定的。至于其他中断，则是由 Hart 从 Hart 外部接收的，每个中断信号（无论是通过线或 MSI 发送）都由软件配置为只发送到单个 Hart。

要向多个 Hart 发送处理器间中断 (IPI)，始发 Hart 只需执行一个循环，向每个目标 Hart 发送单个 IPI 即可。有关向单个目标 hart 发送 IPI 的信息，请参见第 7 章。

与接收站处理这些中断的综合工作量相比，源站向多个目的地发送单个 IPI 所耗费的工作量无疑相形见绌。因此，为 IPI 多播提供自动机制最多只能适度减少系统的总工作量。在有大量 Harts 的情况下，IPI 组播的硬件机制必须解决软件在每次使用时如何准确指定目标集的问题，此外，IPI 的实际物理传输可能与软件版本差别不大。

我们不排除未来为组播 IPI 提供可选硬件机制的可能性，但前提是必须在实际使用中证明其显著优势。据观察，截至 2020 年，Linux 即使在拥有组播 IPI 硬件的系统上也无法使用该硬件。

在极少数情况下，如果需要将来自 I/O 设备的单个中断发送给多个 Hart，则必须将中断发送给单个 Hart，然后由该 Hart 通过 IPI 向其他 Hart 发出信号。

我们认为，需要将 I/O 中断发送给多个 Harts 的情况非常罕见，因此在这种情况下，没有理由将多播的硬件支持标准化。

除了组播交付外，其他架构还支持 "1-of-N "中断交付选项，即硬件从一组配置好的 N 个 Hart 中选择一个目标 Hart，目的是在 Hart 之间自动平衡中断处理的负载。2010 年代的实验对 1-of-N 模式在实践中的实用性提出了质疑，实验表明，软件在负载平衡方面往往比实际芯片中实现的硬件算法做得更好。因此，Linux 被修改为即使在有 1-of-N 中断交付的系统上也不再使用。

对于中断处理的硬件负载平衡可能有利于某些专业市场（如网络）的论点，我们持开放态度。不过，迄今为止在这方面提出的主张并不能证明要求所有 RISC-V 服务器都支持 1-of-N 传输是合理的。随着更多证据的出现，某些 1-N 交付机制可能会成为未来的一种选择。

RISC-V 的原始平台级中断控制器（PLIC）是可配置的，因此每个中断源都会向任何子集（可能是所有子集）的 Harts 发出外部中断信号。当多个 Hart 从 PLIC 的单一中断源接收到外部中断时，第一个在 PLIC 请求中断的 Hart 将负责为该中断提供服务。这通常会引发一场竞赛，即被配置为接收组播中断的 Harts 子集都会同时接收外部中断陷阱，并争先在 PLIC 请求中断。这样做的目的是提供一种 1-of N 的中断交付形式。然而，对于所有未能赢得请求的 Harts 来说，中断陷阱就成了白费力气。

由于上述原因，高级 PLIC 只支持将每个信号中断发送到软件选择的一个 Hart，而不是多个 Hart。

1.6 ISA 扩展名 Smaia 和 Ssaia

高级中断架构（AIA）为 RISC-V 指令集架构（ISA）的扩展定义了两个名称，一个用于 machine-level 执行环境，另一个用于 supervisor-level 环境。对于 machine-level 环境，扩展 **Smaia** 包括所有新增的 CSR，以及对 AIA 指定的所有权限级别的中断响应行为的所有修改。对于 supervisor-level 环境，扩展 **Ssaia** 与 Smaia 基本相同，只是不包括 supervisor-level 无法直接看到的 machine-level CSR 和行为。

扩展 Smaia 和 Ssaia 仅涵盖那些对 ISA 有根本影响的 AIA 功能。尽管本文档将以下内容作为 AIA 的一部分进行描述或讨论，但由于这些组件被归类为非 ISA，因此 Smaia 或 Ssaia 并不暗示这些内容：APLIC、IOMMU 以及除写入 IMSIC 之外的任何启动处理器间中断的机制。

正如后续章节所揭示的，AIA 添加的 CSR 和行为的确切集合，以及 Smaia 或 Ssaia 的含义，取决于基础 ISA 的 XLEN（RV32 或 RV64）、是否实现了 S 模式和 hypervisor extension，以及 hart 是否具有 IMSIC。但是，AIA 并没有为每个可能的有效子集提供单独的扩展名。相反，不同的组合可以从所标示的特征交叉点（如 RV64I + S-mode + Smaia，但没有 hypervisor extension）中推断出来。

编译器和汇编器等软件开发工具不必关心是否存在 IMSIC，而只需在出现 Smaia 或 Ssaia 时允许尝试访问 IMSIC CSR（见第 2 章和第 3 章）。如果没有实际的 IMSIC，这种尝试可能会陷入陷阱，但这对开发工具来说不是问题。

第二章

2.添加到 Harts 的控制和状态寄存器 (CSR)

对于 RISC-V hart 可以接受中断陷阱的每个权限级别，高级中断架构都会增加用于中断控制和处理的 CSR。

2.1 Machine-level CSR

表 2.1 列出了新增的 machine-level CSR 和现有的 machine-level CSR，其大小因高级中断架构而改变。现有 CSR mie、mip 和 mideleg 扩展到 64 位，以支持总共 64 个中断原因。

对于 RV32，表中列出的 *high-half* CSR 允许访问寄存器 mideleg、mie、mvien、mvip 和 mip 的高 32 位。高级中断架构要求 RV32 存在这些 high-half CSR，但它们访问的位可能都只是只读 0。

CSR miselect 和 mireg 为访问表 2.1 中 CSR 以外的多个寄存器提供了窗口。miselect 是一个 **WARL 寄存器**，它必须支持最小值范围，具体取决于实现的功能。在未实现 IMSIC 时，miselect 必须至少能够保存 0 至 0x3F 范围内的任何 6 位值。实现 IMSIC 时，miselect 必须能够保存 0 至 0xFF 范围内的任何 8 位值。0 至 0xFF 范围内的 miselect 值目前按以下子范围分配：

0x00-0x2F 保留
0x30-0x3F 主要中断优先级
0x40-0x6F 保留
0x70-0xFF 外部中断（仅适用于 IMSIC）

miselect 也可以支持 0x00-0xFF 范围之外的值，但目前没有为 0xFF 以上的值分配标准寄存器。

编号	权限	宽度	名称	说明
间接访问寄存器的 machine-level 窗口				
0x350	MRW	XLEN	miselect	machine 间接寄存器选择
0x351	MRW	XLEN	mireg	machine 间接寄存器别名
machine-level 中断				
0x304	MRW	64	mie	machine 中断使能位
0x344	MRW	64	mip	machine 中断挂起位
0x35C	MRW	MXLEN	mtopei	machine top 外部中断（仅适用于 IMSIC）
0xFB0	MRO	MXLEN	mtopi	machine top 中断
supervisor-level 委托中断和虚拟中断				
0x303	MRW	64	mideleg	machine 中断委托
0x308	MRW	64	mvien	machine 虚拟中断使能
0x309	MRW	64	mvip	machine 虚拟中断挂起位
machine-level high-half CSR（仅限 RV32）				
0x313	MRW	32	midelegh	mideleg 的高 32 位（仅适用于 S 模式）
0x314	MRW	32	mieh	mie 的高 32 位
0x318	MRW	32	mvienh	mvien 的高 32 位（仅限 S 模式）
0x319	MRW	32	mviph	mvip 的高 32 位（仅适用于 S 模式）
0x354	MRW	32	miph	mip 的高 32 位

表 2.1：高级中断架构增加或扩大的 machine-level CSR。

最高有效位为1（位 XLEN - 1 = 1）的 miselect 值指定给用户使用，可能是用于通过 mireg 访问自定义寄存器。如果 XLEN 发生变化，miselect 的最高有效位将移动到新位置，并保留之前的值。执行程序无需支持 miselect 的任何自定义值。

当 miselect 是一个保留范围内的数字（目前为 0x00-0x2F、0x40-0x6F 或 0xFF 以上未指定为自定义使用的数字）时，尝试访问 mireg 通常会引发非法指令异常。

通常情况下，外部中断区域 0x70-0xFF 仅在实现 IMSIC 时填入；否则，当 miselect 位于该区域时，试图访问 mireg 也会导致非法指令异常。外部中断区域的内容在有关 IMSIC 的第 3 章中有详细说明。

CSR mtopei 也只有在执行 IMSIC 时才存在，因此与间接访问的 IMSIC 寄存器一起记录在第 3 章中。

CSR mtopi 报告 5.2.2 节中规定的 machine-level 的挂起并且使能的最高优先级中断。

实现了 S 模式时，CSR mvien 和 mvip 支持中断过滤和 supervisor level 虚拟中断。第 5.3 节将解释这些功能。

如果还实现了扩展 Smcsrind，那么当 miselect 的值范围在 0x30-0x3F 或 0x70-0xFF 之间时，访问别名 CSR mireg2 至 mireg6 的尝试会引发非法指令异常。

2.2 Supervisor-level CSR

表 2.2 列出了新增的 supervisor-level CSR 和现有的 CSR，如果 hart 实现了 S 模式，这些 CSR 的位数将扩大到 64 位。这些寄存器的功能都与 machine-level 寄存器一致。

编号	权限	宽度	名称	说明

间接访问寄存器的 supervisor-level 窗口				
0x150	SRW	XLEN	siselect	Supervisor 间接寄存器选择
0x151	SRW	XLEN	sireg	Supervisor 间接寄存器别名
supervisor-level 中断				
0x104	SRW	64	sie	Supervisor 中断使能位
0x144	SRW	64	sip	Supervisor 中断挂起位
0x15C	SRW	SXLEN	stopei	Supervisor top 外部中断（only with an IMSIC）
0xDB0	SRO	SXLEN	stopi	Supervisor top 中断
supervisor-level high-half CSR（仅限 RV32）				
0x114	SRW	32	sieh	sie 的高 32 位
0x154	SRW	32	siph	sip 的高 32 位

表 2.2：高级中断架构增加或扩大的 supervisor-level CSR。

通过 siselect/sireg 窗口访问的寄存器空间与 machine-level 的寄存器空间是分开的，但与 machine-level 的寄存器空间类似，都是用于 supervisor-level 中断而非 machine-level 中断。在 0 至 0xFF 范围内的 siselect 分配值也是如此：

0x00-0x2F 保留
0x30-0x3F 主要中断优先级
0x40-0x6F 保留
0x70-0xFF 外部中断（仅适用于 IMSIC）

为获得最大的兼容性，建议 siselect 至少支持 9 位范围，0 至 0x1FF，无论是否存在 IMSIC。

由于 VS CSR vsiselect*（第 2.3 节）始终至少有 9 位，而且与其他 VS CSR 一样，当在虚拟机（VS 模式或 VU 模式）中执行时，*vsiselect *会替代* siselect*，因此为* siselect *设置一个较小的范围可以让软件发现自己不是在虚拟机中运行。*

与 miselect 类似，最高有效位为1（位 XLEN - 1 = 1）的 siselect 值可用于自定义用途。如果 XLEN 发生变化，siselect 的最高有效位将移动到新的位置，并保留其之前的值。执行程序无需支持 siselect 的任何自定义值。

当 siselect 为保留范围内的数字（目前为 0x00-0x2F、0x40-0x6F，或 0xFF 以上未指定为自定义使用的数字），或 0x70-0xFF 范围内没有 IMSIC 时，访问 sireg 的尝试最好引发非法指令异常（除非在虚拟机中执行，下一节将介绍）。

请注意，siselect 和 sireg 的宽度始终是当前的 XLEN 而不是 SXLEN。因此，举例来说，如果 MXLEN = 64，SXLEN = 32，那么当当前权限模式为 M（运行 RV64 代码）时，这些寄存器的宽度为 64 位，而当权限模式为 S（运行 RV32 代码）时，这些寄存器的宽度为 32 位。

CSR stopei 与 IMSIC 的相关介绍见第 3 章。

寄存器 stopi 报告挂起的最高优先级中断，并根据第 5.4.2 节的规定启用 supervisor-level 。

如果还实现了扩展名 Sscsrind，那么当 siselect 的值范围在 0x30-0x3F 或 0x70-0xFF 之间时，尝试访问别名 CSR sireg2 至 sireg6 会引发非法指令异常（除非在虚拟机中执行，下一节将介绍）。

2.3 Hypervisor 和 VS CSR

如果 hart 实现了特权架构的 hypervisor extension，那么表 2.3 中列出的 hypervisor 和 VS CSR 也会增加或扩大到 64 位。

表中新的虚拟机 hypervisor CSR（hvien、hvictl、hviprio1 和 hviprio2）增强了 hvip 的功能，可将中断注入 VS 层。这些寄存器的使用将在第 6 章 "虚拟机中断 "中介绍。

新的 VS CSR（vsiselect、vsireg、vstopei 和 vstopi）都与监控 CSR 相匹配，在虚拟机（VS 模式或 VU 模式）中执行时可替代这些监控 CSR。

CSR vsiselect 要求至少支持 0 至 0x1FF 的 9 位范围，无论是否实现了 IMSIC。与 siselect 一样，最高有效位为 1（位 XLEN - 1 = 1）的 vsiselect 值指定用于自定义用途。如果 XLEN 发生变化，vsiselect 的最高有效位将移动到新位置，并保留之前的值。

与 siselect 和 sireg 一样，vsiselect 和 vsireg 的宽度始终是当前的 XLEN，而不是 VSXLEN。因此，举例来说，如果 HSXLEN = 64，VSXLEN = 32，那么这些寄存器在 HS 模式（运行 RV64 代码）下被 hypervisor 访问时为 64 位，而在 VS 模式（运行 RV32 代码）下被 guest 操作系统访问时为 32 位。

编号	权限	宽度	名称	说明
VS 级的委托中断和虚拟中断、中断优先级				
0x603	HRW	64	hideleg	Hypervisor 中断授权
0x608	HRW	64	hvien	Hypervisor 虚拟中断使能
0x609	HRW	HSXLEN	hvictl	Hypervisor 虚拟中断控制
0x645	HRW	64	hvip	Hypervisor 虚拟中断挂起位
0x646	HRW	64	hviprio1	Hypervisor VS 级中断优先级
0x647	HRW	64	hviprio2	Hypervisor VS 级中断优先级
间接访问寄存器的 VS 级窗口				
0x250	HRW	XLEN	vsiselect	Virtual supervisor 间接寄存器选择
0x251	HRW	XLEN	vsireg	Virtual supervisor 间接寄存器别名
VS 级中断				
0x204	HRW	64	vsie	Virtual supervisor 中断使能位
0x244	HRW	64	vsip	Virtual supervisor 中断挂起位
0x25C	HRW	VSXLEN	vstopei	Virtual supervisor top 外部中断（only with an IMSIC）
0xEB0	HRO	VSXLEN	vstopi	Virtual supervisor top 中断
hypervisor 和 VS 级 high-half CSR（仅限 RV32）				
0x613	HRW	32	hidelegh	hideleg 的高 32 位
0x618	HRW	32	hvienh	hvien 的高 32 位
0x655	HRW	32	hviph	hvip 的高 32 位
0x656	HRW	32	hviprio1h	hviprio1 的高 32 位
0x657	HRW	32	hviprio2h	hviprio2 的高 32 位
0x214	HRW	32	vsieh	vsie 的高 32 位
0x254	HRW	32	vsiph	vsip 的高 32 位

表 2.3： 高级中断架构增加或拓宽的 hypervisor 和 VS CSR。(参数 HSXLEN 只是 SXLEN 的另一个名称，用于 hypervisor extension S 模式）。

与 machine 和 supervisor-level 相比，vsiselect 可选择的寄存器空间更为有限：

0x000-0x02F 保留
0x030-0x03F 不可访问
0x040-0x06F 保留
0x070-0x0FF 外部中断（仅限 IMSIC），或无法访问
0x100-0x1FF 保留

对于别名 CSR sireg 和 vsireg，Hypervisor 扩展关于何时引发虚拟指令异常（基于指令是否*符合 HS 条件*）的通常规则并不适用。本节针对 sireg 和 vsireg 所给出的规则反而适用，除非被第 2.5 节的要求所覆盖，当扩展 Smstateen 也被实现时，这些要求优先于本节。

如果尝试从 VS 模式或 VU 模式直接访问 vsireg，或尝试从 VU 模式访问 sireg，都会引发虚拟指令异常。

当 vsiselect 具有保留值（包括 0x1FF 以上未指定为自定义用途的值）时，从 M 模式或 HS 模式访问 vsireg 或从 VS 模式访问 sireg（实际上是 vsireg）的尝试最好引发非法指令异常。

当 vsiselect 具有*不可访问*寄存器的编号时，从 M 模式或 HS 模式访问 vsireg 的尝试会引发非法指令异常，而从 VS 模式访问 sireg（实际上是 vsireg）的尝试会引发虚拟指令异常。

要求 vsiselect *的范围为* 0-0x1FF*，即使大部分或全部空间都被保留或无法访问，也允许 hypervisor 在实现的范围内模拟间接访问的寄存器，包括将来可能标准化的位置为* 0x100-0x1FF 的*寄存器。*

外部中断间接访问寄存器（编号 0x70-0xFF）只有在 hstatus 的字段 VGEIN 为已执行的 guest 外部中断编号而非 0 时才能访问。如果 VGEIN 不是已执行的 guest 外部中断的编号（包括未实现 IMSIC 的情况），则所有在 0x030-0x03F 和 0x070-0x0FF 范围内的间接寄存器编号都指定为 VS 级不可访问寄存器。

同样，当 hstatus.VGEIN 不是已实现的 guest 外部中断的编号时，从 M 模式或 HS 模式访问 CSR vstopei 的尝试会引发非法指令异常，而从 VS 模式访问 stopei 的尝试会引发虚拟指令异常。

如果还实现了扩展名 Sscsrind，那么当 vsiselect 的值范围在 0x30-0x3F 或 0x70-0xFF 之间时，从 M 模式或 HS 模式访问别名 CSR vsireg2 至 vsireg6 的尝试会引发非法指令异常，而从 VS 模式访问 sireg2 至 sireg6 的尝试会引发虚拟指令异常。

2.4 虚拟指令异常

根据 hypervisor extension 的默认规则，从 VS 模式直接访问除 vsireg 之外的 hypervisor 或 VS CSR，或从 VU 模式访问除 sireg 或 vsireg 之外的任何 supervisor-level CSR（包括 hypervisor 和 VS CSR），通常不会引发非法指令异常，而是虚拟指令异常。有关详情，请参阅 RISC-V 特权架构。

读/写 CSR stopei 或 vstopei 的指令被视为*符合 HS 标准*，除非以下条件全部为真：hart 具有 IMSIC，实现了扩展 Smstateen，且 mstateen0 的第 58 位为 0。(有关 mstateen0，请参见下一节 2.5）。

关于 sireg 和 vsireg，请参阅上一节 2.3 和下一节 2.5，了解何时需要虚拟指令异常而不是非法指令异常。

2.5 通过 state-enable CSR 进行访问控制

如果扩展 Smstateen 与高级中断架构（AIA）一起实现，状态使能寄存器 mstateen0 的三个位将控制从权限低于 M 模式的特权模式访问 AIA 附加状态：

位 60　CSR siselect、sireg、vsiselect 和 vsireg
位 59　AIA 添加的、不受比特 60 和 58 控制的所有其他状态
位 58　所有 IMSIC 状态，包括 CSR stopei 和 vstopei

如果在 mstateen0 中这些位中有一位为 0，则试图从权限小于 M 模式的权限模式访问相应状态时，会出现非法指令陷阱。与往常一样，状态使能 CSR 不影响在 M 模式下对任何状态的访问，只影响在权限较低模式下的访问。更多解释，请参阅扩展 Smstateen 的文档。

第 59 位控制对 AIA CSR siph、sieh、stopi、hidelegh、hvien/hvienh、hviph、hvictl、hviprio1/hviprio1h、hviprio2/hviprio2h、vsiph、vsieh 和 vstopi 的访问，以及对通过 siselect + sireg（第 5.4.1 节的 iprio 数组）访问的 supervisor-level 中断优先级的访问。

只有当 hart 实现 IMSIC 时，才在 mstateen0 中执行第 58 位。如果同时实现了 hypervisor extension，则该位不会影响 hypervisor CSR hgeip 和 hgeie 的行为或访问性，也不会影响 hstatus 的字段 VGEIN。特别是，即使 mstateen0 的第 58 位为 0，来自 IMSIC 的 guest 外部中断仍可在 hgeip 中的 HS 模式中看到。

Smstateen 的早期批准前草案指出，当 mstateen0 *的第 58 位为零时，寄存器* hgeip *和* hgeie *以及* hstatus *的字段 VGEIN 均为只读零。这种说法已不再正确。*

如果 Hart 没有 IMSIC，则 mstateen0 的第 58 位只读为 0，但 Smstateen 对访问不存在的 IMSIC 状态的尝试没有影响。

这尤其意味着，当 hart 没有 IMSIC 时，尽管 mstateen0 *的第 58 位为 0，下面的指令将引发第 2.3 节所述的虚拟指令异常，而不是非法指令异常：*

- *当* vsiselect *的值在* 0x70-0xFF *范围内时，尝试从 VS 模式访问* sireg *（实际上是* vsireg*）；以及*
- *尝试从 VS 模式访问* stopei *（实际上是* vstopei*）。*

如果 mstateen0 的第 60 位为 1，则无论其他任何 mstateen 位（包括 mstateen0 的第 58 和 59 位）如何，对于所有从 VS 模式或 VU 模式直接访问 vsireg 的尝试，以及所有从 VU 模式访问 sireg 的尝试，都会引发第 2.3 节所述的虚拟指令异常。只有当 mstateen0 的第 60 位为 0 时，该行为才会被覆盖。

如果实现了 hypervisor extension，则在 hypervisor CSR hstateen0 中也定义了同样的三个位，但只涉及在特权模式 VS 和 VU 下执行的虚拟机可能访问的状态：

位 60 CSRs siselect 和 sireg（实际上是 vsiselect 和 vsireg）
位 59　CSR siph 和 sieh（仅限 RV32）以及 stopi（实际为 vsiph、vsieh 和 vstopi）
位 58 IMSIC guest 中断文件的所有状态，包括 CSR stopei（实际为 vstopei）

如果这些位中的一位在 hstateen0 中为 0，而在 mstateen0 中为 1，那么从 VS 或 VU 模式访问相应状态的尝试将引发虚拟指令异常。(但请注意，对于 high-half CSR siph 和 sieh，这只适用于 XLEN = 32 的情况。当 XLEN > 32 时，尝试访问 siph 或 sieh 会像往常一样引发非法指令异常，而不是虚拟指令异常）。

如果 mstateen0 中的第 60 位为 1，而 hstateen0 中的第 60 位为 0，则无论 vsiselect 或任何其他 mstateen 位的值如何，VS 或 VU 模式访问 siselect 或 sireg 的所有尝试都会引发虚拟指令异

常，而不是非法指令异常。

只有当 hart 具有 IMSIC 时，才会在 hstateen0 中执行第 58 位。此外，即使有 IMSIC，如果 IMSIC 没有用于 guest 外部中断的 guest 中断*文件*，hstateen0 中的第 58 位也可能（或不可能）只读为零（第 3 章）。当该位为 0 时（无论是只读为 0 还是设置为 0），虚拟机将无法访问 hart 的 IMSIC，这与 hstatus.VGEIN = 0 时的情况相同。

扩展 Ssstateen 被定义为 Smstateen 的 supervisor-level 视图。因此，Ssaia 和 Ssstateen 的组合包含了上文定义的 hstateen0 的位，但不包括 mstateen0 的位，因为 machine-level CSR 对 supervisor-level 不可见。

第三章

3.Incoming MSI 控制器（IMSIC）

Incoming MSI 控制器（IMSIC）是一个可选的 RISC-V 硬件组件，与 hart 紧密相连，每个 hart 有一个 IMSIC。IMSIC 接收并记录 hart的传入消息信号中断 (MSI)，并在有挂起和已启用的中断时向 hart 发出信号。

IMSIC 在 machine 地址空间中有一个或多个内存映射寄存器，用于接收 MSI。除了这些内存映射寄存器外，软件与 IMSIC 的交互主要通过附加核心的几个 RISC-V CSR。

3.1 中断文件和中断标识

在 RISC-V 系统中，MSI 不仅会被定向到特定 hart，而且会被定向到特定 hart 的特定权限级别，如 machine 或 supervisor-level。此外，当 hart 实现了 hypervisor extension 时，IMSIC 可以选择允许 MSI 定向到 virtual supervisor level（VS 级）的特定虚拟 hart。

对于每个权限级别和每个可向其发送 MSI 的虚拟 hart，hart的 IMSIC 都包含一个单独的*中断文件*。假定一个 hart 实现了 supervisor 模式，那么它的 IMSIC 至少有两个中断文件，一个用于 machine-level，另一个用于 supervisor-level 。如果 Hart 还实现了 hypervisor extension，其 IMSIC 可能会有额外的中断文件，用于虚拟 Hart，即 *guest 中断文件*。IMSIC 为虚拟机提供的 guest 中断文件数量正好是 *GEILEN*，即 RISC-V 专用体系结构为 hypervisor extension 定义的受支持的 guest 外部中断数量。

每个独立的中断文件主要由两个大小相同的比特数组组成，一个数组用于记录已到达但尚未服务的 MSI（中断挂起比特），另一个数组用于指定 hart 当前接受的中断（中断使能比特）。这两个数组中的每个位都对应不同的中断*标识号，*不同来源的 MSI 可通过该*标识号*在中断文件中进行区分。由于 IMSIC 是 hart 的外部中断控制器，因此中断文件的中断标识将成为所附 hart 外部中断的*次要标识*。

中断文件支持的中断标识数（以及每个数组中的有效位数）是比 64 的倍数小于 1 的数，最小为 63，最大为 2047。

平台标准可能会增加每个中断文件必须执行的最小中断标识数。

当中断文件支持 *N 个*不同的中断标识时，有效标识号介于 1 和 *N 之间*。在此范围内的标识号在中断文件中已实现；在此范围外的标识号未实现。0 绝不是有效的中断标识。

IMSIC 硬件不认为一个中断文件的中断标识号与另一个中断文件的中断标识号有任何联系。软件通常会为不同中断文件的不同 MSI 源分配相同的中断标识号，而不会在不同中断文件之间进行协调。因此，系统中可单独区分的 MSI 信号源总数可能是单个中断文件的中断标识数乘以系统中所有中断文件总数的乘积。

系统中的所有中断文件并不一定大小相同（大小相同表示实现的中断标识数量相同）。对于给定的 hart，guest 外部中断的中断文件大小必须相同，但 machine-level 和 supervisor-level 的中断文件大小可能与 guest 外部中断的中断文件不同，也可能彼此不同。同样，不同 Hart 的中断文件也可能大小不同。

平台可为软件提供配置 IMSIC 中中断文件数量和/或其大小的方法，例如允许用 machine-level 的较小中断文件换取 supervisor-level 的较大中断文件，反之亦然。任何此类可配置性都超出了本规范的范围。不过，建议只赋予 machine-level 改变 IMSIC 中中断文件数量和大小的权力。

3.2 MSI 编码

既定标准（特别是 PCI 和 PCI Express 标准）规定，来自设备的单个消息信号中断 (MSI) 采用设备自然对齐的 32 位写入形式，地址和值均由软件在设备（或设备控制器）上配置。根据设备或控制器所符合的标准版本，地址可能被限制在低 4 GB（32 位）范围内，而写入的值可能被限制在 16 位范围内，高 16 位始终为零。

当 RISC-V Hart 具有 IMSIC 时，来自设备的 MSI 通常会直接发送到由软件选择来处理中断的单个 Hart（可能是基于某种中断亲和性策略）。MSI 将通过存在于接收单元 IMSIC 中的相应中断文件定向到特定权限级别或特定虚拟单元。MSI 写地址是与目标中断文件物理连接的特定字长寄存器的物理地址。MSI 写入的数据只是该中断文件中挂起中断的标识号（最终成为外部中断的次要标识）。

通过在设备上配置 MSI 的地址和数据，系统软件可完全控制：(a) 接收特定设备中断的目标单元，(b) 目标权限级别或虚拟单元，以及 (c) 在目标中断文件中代表 MSI 的标识号。要素 a 和 b 由 MSI 地址所针对的中断文件决定，而要素 c 则由 MSI 数据传达。

由于 IMSIC 可支持的最大中断标识号为 2047，因此 MSI 数据值的 16 位限制并不存在问题。

在实现 hypervisor extension 且设备由 guest 操作系统直接管理时，设备的 MSI 地址最初是 guest 物理地址，因为它们是由 guest 操作系统在设备上配置的。这些 guest 地址必须由 IOMMU 转换，IOMMU 由 hypervisor 配置，以便将这些 MSI 重定向到中断文件，用于正确的传递 guest 外部中断。有关此主题的更多信息，请参阅第 8 章。

3.3 中断优先级

在单个中断文件中，中断优先级直接由中断标识号决定。标识号越小，优先级越高。

由于 MSI 可让软件完全控制中断文件中标识号的分配，因此软件可自由选择反映中断所需的相对优先级的标识号。

诚然，如果中断文件中包含一个优先级数组，可以为每个中断标识分配优先级，那么软件就可以更动态地调整中断优先级。但我们认为，这种额外的灵活性不会经常使用，不足以证明额外的硬件支出是合理的。事实上，对于目前使用 MSI 的许多系统来说，软件通常会完全忽略中断优先级，并将所有中断视为具有相同的优先级。

一个中断文件的最低标识号被赋予了最高优先级，而不是相反的顺序，因为只有最高优先级的中断才可能需要仔细管理优先级顺序，而在所有系统中保证存在的是低标识号，即 1 到 63（或者 1 到 255）。例如，一个中断文件中优先级最高的中断（可能是时间最紧迫的中断）总是标识号 1。如果优先级顺序颠倒，最高优先级的中断在不同的 machine 上就会有不同的标识号，这取决于中断文件实现了多少个标识。如果软件能够为最高优先级的中断分配固定的标识号，那么中断优先级顺序与自然数顺序相反所带来的任何不适都是值得的。

3.4 重置和显示状态

重置 IMSIC 后，其中断文件的所有状态都会变得有效和一致，但其他状态未作说明，machine-level 和 supervisor-level 中断文件的 eidelivery 寄存器除外，具体说明见第 3.8.1 节。

如果 IMSIC 中包含 supervisor-level 中断文件，且所附 hart 上的软件启用了先前禁用的 S 模式（例如，通过将 CSR misa 的 S 位从 0 改为 1），则 supervisor-level 中断文件的所有状态均有效且一致，但未指定其他状态。同样，如果 IMSIC 中包含 guest 中断文件，而所连接的 hart 上的软件启用了先前禁用的 hypervisor extension （例如，将 misa 的 H 位从 0 改为 1），则 IMSIC guest 中断文件的所有状态都有效且一致，但在其他方面未指定。

3.5 中断文件的内存区域

IMSIC 中的每个中断文件都有一个或两个内存映射 32 位寄存器，用于接收 MSI 写入。这些内存映射寄存器位于中断文件物理地址空间中自然对齐的 4-KiB 区域（页）内，即每个中断文件一个页。

中断文件内存区域的布局为

偏移	尺寸	寄存器名
0x000	4 个字节	seteipnum_le
0x004	4 个字节	seteipnum_be

中断文件 4-KiB 内存区域中的所有其他字节都是保留字节，必须以只读 0 的方式执行。

在中断文件的内存区域内，只支持自然对齐的 32 位简单读写。对只读字节的写入将被忽略。对于其他形式的访问（其他大小、错位访问或 AMO），IMSIC 实现最好报告访问故障或总线错误，否则必须忽略访问。

如果 *i* 是一个已实现的中断标识号，以 little-endian 字节顺序向 seteipnum_le（按编号设置外部

中断挂起位，little-endian）写入值 *i 会*将中断 *i 的*挂起位设为 1。如果写入的值不是按小端顺序执行的中断标识号，则对 seteipnum_le 的写入将被忽略。

对于支持 big-endian 字节序的系统，如果 *i* 是已实现的中断标识号，则以 big-endian 字节序向 seteipnum_be（按编号设置外部中断挂起位，Big-Endian）写入值 *i* 会将中断 *i* 的挂起位设为 1。如果写入的值不是按 big-endian 字节顺序执行的中断标识号，对 seteipnum_be 的写入将被忽略。只支持小端顺序的系统可选择忽略所有写入 seteipnum_be 的操作。

在大多数系统中，seteipnum_le 是指向该中断文件的 MSI 的写入端口。对于主要为 big-endian 字节序而构建的系统，seteipnum_be 可以作为某些设备指向该中断文件的 MSI 的写入端口。

在任何情况下，读取 seteipnum_le 或 seteipnum_be 都会返回 0。

如果没有被忽略，对中断文件内存区域的写入将保证最终反映在中断文件中，但不一定是立即反映。对于单个中断文件，对其内存区域的多次写入（存储）的影响虽然会任意延迟，但其发生顺序总是与 RISC-V 非特权 ISA 所定义的存储的*全局内存顺序*相同。

在大多数情况下，从完成对中断文件内存区域的写入到写入对中断文件的影响之间的任何延迟，都与内存系统中的其他延迟没有区别。但是，如果 Hart 向其 IMSIC 的 seteipnum_le *或* seteipnum_be *寄存器写入数据，则 Hart 可能会看到存储指令完成与中断文件中的中断挂起位被设置之间的延迟。*

3.6 多个中断文件的内存区域的安排

如上一节所述，IMSIC 实现的每个中断文件都有自己的内存区域，正好占用一个 4-KiB 的 machine 地址空间页。在可行的情况下，所有 IMSIC 的 machine-level 中断文件的内存页应一起位于地址空间的一部分，而所有 supervisor-level 和 guest 中断文件的内存页同样应一起位于地址空间的另一部分，具体规则如下。

将 machine-level 中断文件与地址空间中的其他中断文件分开的主要原因是，实现物理内存保护（PMP）的 Harts 只需使用一个 PMP 表项，就能向所有 supervisor-level 和 guest 级中断文件授予 supervisor-level 访问权限。如果将 machine-level 中断文件的内存页与权限较低的中断文件的内存页交错排列，那么向所有非 machine-level 中断文件授予 supervisor-level 访问权限所需的 PMP 表项数量可能与系统中的 Harts 数量相当。

如果 machine 的结构要求将 Harts 细分为若干组，每组在地址空间中各占一部分，那么最好的办法是将每组 Harts 的 machine-level 中断文件分别放在一起，同样地，将每组 Harts 的 supervisor-level 和 guest 级中断文件分别放在一起。下文将进一步讨论这种情况。

系统可能会在地址空间中将中断组划分为若干组，因为每一组都存在于单独的芯片（或多芯片模块中的芯片组）上，将多个芯片的地址空间编织在一起是不切实际的。在这种情况下，授予 supervisor-level 访问所有非 machine-level 中断文件的权限，每个组只需一个 PMP 表项。

为了在地址空间中定位中断文件的内存页，假设每个 hart（或组中的每个 hart）都有一个唯一的 hart 编号，该编号可能与 RISC-V 特权架构分配给 hart 的唯一 hart 标识符（"hart ID"）有关，也可能无关。

为方便寻址，所有 machine-level 中断文件（或单个中断文件组的所有中断文件）的内存页应这样安排：对于某个整数常量 A 和 *C*，hart 号 *h* 的 machine-level 中断文件地址由公式 $A + h \times 2^C$ 给出。如果最大的 hart 号是 h_{max}，则让 $k = \lceil \log_2(h_{max} + 1) \rceil$，即表示任何 hart 号所需的位数。那么

基地址 A 应对齐 2^{k+C} 地址边界，因此 $A + h \times 2^C$ 总是等于 $A \mid (h \times 2^C)$，其中竖条 (|) 表示位逻辑 OR。

C 的最小值是 12，2^C 是一个 4KB 页面的大小。如果 $C > 12$，则每个 machine-level 中断文件的内存页起点不仅要与 4KB 页对齐，还要与更严格的 2^C 地址边界对齐。在 2^{k+C} 大小的地址范围 A 至 $A + 2^{k+C} - 1$ 内，未被 machine-level 中断文件占用的每个 4-KiB 页都应填满 32 位只读 0 字，这样，读取对齐字时返回 0，写入对齐字时被忽略。

所有 supervisor-level 中断文件的内存页（或单组 *hart 的*所有内存页）也应进行类似的排列，以便 hart 编号为 h 的 supervisor-level 中断文件的地址为 $B + h \times 2^D$ （取某个整数常数 B 和 D），其中基地址 B 对齐到 2^{k+D} 地址边界。

如果 IMSIC 实现了 guest 中断文件，则 IMSIC 的 supervisor-level 中断文件和 guest 中断文件的内存页应是连续的，从地址最低的 supervisor-level 中断文件开始，然后是 guest 中断文件，按 guest 中断编号排序。按计划，内存页应按以下方式连续排序

$$S, G_1, G_2, G_3, \ldots$$

其中，S 是 supervisor-level 中断文件的页，每个 G_i 是 guest 中断编号 i 的中断文件页。因此，常数 D 的最小值为 $\lceil \log_2 (\text{maximum GEILEN} + 1) \rceil + 12$，记得每个 IMSIC 的 GEILEN 是 IMSIC 实现的 guest 中断文件数。

在 2^{k+D} 大小的地址范围 B 至 $B + 2^{k+D} - 1$ 内，每一个未被中断文件（supervisor-level 或 guest）占用的 4KB 页都应填满 32 位只读 0 字。

当系统将 hart 分成若干组，每个组都有自己独立的地址空间时，对于 machine-level 中断文件，中断文件的内存页地址应遵循如下公式 $g \times 2^E + A + h \times 2^C$；对于 supervisor-level 中断文件，应遵循如下公式 $g \times 2^E + B + h \times 2^D$，其中 g 是*组号*，h 是相对于组的 hart 号，E 是另一个整数常数 $\geq k + \max(C, D)$，但通常要大得多。如果最大的分组数是 g_{max}，让 $j = \lceil \log_2 (g_{max} + 1) \rceil$，即表示任何分组数所需的比特数。除了分别是 2^{k+C} 和 2^{k+D} 的倍数外，A 和 B *的*选择应符合以下条件

$$(2^j - 1) \times 2^E \,\&\, a = 0 \qquad \text{和} \qquad (2^j - 1) \times 2^E \,\&\, b = 0$$

其中，"&"表示位逻辑 AND。这样可以确保

$$g \times 2^E + A + h \times 2^C \quad \text{总是等于} \quad (g \times 2^E) \mid A \mid (h \times 2^C) \text{、} \quad \text{和}$$
$$g \times 2^E + B + h \times 2^D \quad \text{总是等于} \quad (g \times 2^E) \mid B \mid (h \times 2^D).$$

只在每个组内填充只读为零的页面，而不是在不同组之间填充。具体来说，如果 g 是 *0 和 2^j - 1*（*包括 2^j* - 1）之间的任意整数，那么在地址范围内：

$g \times 2^E + A$ through $g \times 2^E + A + 2^{k+C} - 1$、 和
$g \times 2^E + B$ through $g \times 2^E + B + 2^{k+D} - 1$、

中断文件未占用的页面应为只读 0。

APLIC 可以使用默认算法来确定传出 MSI 的目标地址，即 IMSIC 中断文件的地址，另请参见第 4.9.1 节。

3.7 通过 IMSIC 发起外部中断相关的 CSR

软件主要通过第 2 章介绍的 CSR 访问 Hart 的 IMSIC。每个可接收中断的权限级别都有一套独立的 CSR。machine-level CSR 与 IMSIC 的 machine-level 中断文件交互，而如果实现了 supervisor 模式，则 supervisor-level CSR 与 IMSIC 的 supervisor-level 中断文件交互。当 IMSIC 具有 guest 中断文件时，VS CSR 与单个 guest 中断文件交互，guest 中断文件由 CSR hstatus 的 VGEIN 字段选择。

对于 machine-level，相关的 CSR 包括 miselect、mireg 和 mtopei。在实现 supervisor 模式时，supervisor-level CSR 的集合与 machine-level 的一致：siselect、sireg 和 stopei。而在实现 hypervisor extension 时，有三个相应的 VS CSR：vsiselect、vsireg 和 vstopei。

如第 2 章所述，寄存器 miselect 和 mireg 提供了对其他 machine-level 寄存器的间接访问。同样，supervisor-level 寄存器 siselect 和 sireg 以及 VS 级寄存器 vsiselect 和 vsireg 也是如此。在每种情况下，**iselect CSR*（miselect、siselect 或 vsiselect）的值范围为 0x70-0xFF，用于选择相应 IMSIC 中断文件的寄存器，即 machine-level 中断文件（miselect）、 supervisor-level 中断文件（siselect）或 guest 中断文件（vsiselect）。

各级中断文件的作用相同。对于给定的权限级别，*iselect CSR 在 0x70-0xFF 范围内的值会选择相应中断文件的这些寄存器：

0x70 eidelivery
0x72 eithreshold
0x80 eip0
0x81 eip1
. . .. ..
0xBF eip63
0xC0 eie0
0xC1 eie1
. . .. ..
0xFF eie63

寄存器编号 0x71 和 0x73-0x7F 为保留值。当 *iselect CSR 具有其中一个值时，从匹配的 **ireg CSR*（mireg、sireg 或 vsireg）读取将返回 0，写入 *ireg CSR 将被忽略。(对于 vsiselect 和 vsireg，所有访问都取决于 hstatus.VGEIN 是否为 guest 中断文件的有效编号）。

寄存器 eip0 至 eip63 包含所有已执行中断标识的挂起位，统称为 *eip 数组*。寄存器 eie0 至 eie63 包含相同中断标识的使能位，统称为 *eie 数组*。

间接访问的中断文件寄存器和 CSR mtopei、stopei 和 vstopei 将在接下来的两节中详细介绍。

3.8 间接访问的中断文件寄存器

本节将介绍中断文件中的寄存器，这些寄存器可以通过一个 *iselect CSR（miselect、siselect 或 vsiselect）及其伙伴 *ireg CSR（mireg、sireg 或 vsireg）进行间接访问。这些间接访问的宽度始终是当前的 XLEN，对于 RV32 代码是 32 位，对于 RV64 代码是 64 位。

3.8.1 外部中断发送使能寄存器（eidelivery）

eidelivery 是一个 **WARL** 寄存器，用于控制中断文件中的中断是否从 IMSIC 发送到所连接的 hart，以便在 hart 的 mip 或 hgeip CSR 中显示为挂起外部中断。寄存器 eidelivery 也可选择支持将 PLIC（平台级中断控制器）或 APLIC（高级 PLIC）的中断直接发送到所连接的 hart。目前为 eidelivery 定义了三种可能的值：

0 = 禁用中断传送
1 = 使能从中断文件发送中断
0x40000000 = 使能从 PLIC 或 APLIC 的中断传送（可选）

如果 eidelivery 支持值为 0x40000000，则系统中的特定 PLIC 或 APLIC 可充当所附 hart 的备用外部中断控制器，其权限级别与该中断文件相同。当 eidelivery 值为 0x40000000 时，中断文件的功能与 eidelivery 值为 0 时相同，PLIC 或 APLIC 将取代中断文件，为 hart 提供该权限级别的挂起外部中断。

Guest 中断文件不支持用于 eidelivery 的值 0x40000000。

如果支持 0x40000000，重置后会将 eidelivery 初始化为 0x40000000；否则，重置后 eidelivery 将具有未指定的有效值（0 或 1）。

---

eidelivery *值* 0x40000000 *支持对 IMSIC 视而不见的系统软件，它认为外部中断控制器是 PLIC 或 APLIC。这类软件的存在可能是由于它早于 IMSIC 的存在，也可能是由于绕过 IMSIC 被认为可以减少编程工作量。*

3.8.2 外部中断使能阈值寄存器（eithreshold）

eithreshold 是一个 **WLRL** 寄存器，用于确定允许从该中断文件向附加 hart 发出中断信号的最小中断优先级（最大中断标识号）。如果 *N* 是该中断文件的最大执行中断标识号，则 eithreshold 必须能够保持 0 至 *N* 之间的所有值。

当 eithreshold 为非零值 *P* 时，无论 eie 数组中相应的中断使能位如何设置，中断标识 *P* 及更高的中断标识都不参与发起中断，就像这些中断标识未被使能一样。当 eithreshold 为零时，所有已使能的中断标识都会从中断文件中发出中断信号。

3.8.3 外部中断挂起寄存器（eip0-eip63）

当当前 XLEN = 32 时，寄存器 eipk 包含标识号为 *k* × 32 至 *k* × 32 + 31 的中断的挂起位。对于该范围内已执行的中断标识 *i*，中断 *i* *的挂起位*是寄存器 *eipk* *的*位（*i* mod 32）。

当当前 XLEN = 64 时，奇数寄存器 eip1、eip3、...eip63 不存在。在这种情况下，如果 *iselect CSR 是 0x81-0xBF 范围内的奇数值，试图访问匹配的 *ireg CSR 会引发非法指令异常，除非在 VS 模式下进行，否则会引发虚拟指令异常。对于偶数 *k*，寄存器 *eipk* 包含标识号为 *k* × 32 至 *k* × 32 + 63 的中断的挂起位。对于该范围内已执行的中断标识 *i*，中断 *i* *的挂起*位是寄存器 *eipk* *的*位（*i* mod 64）。

有效 *eipk* 寄存器中与支持的中断标识不对应的位位置（如 eip0 的第 0 位）为只读 0。

3.8.4 外部中断使能寄存器（eie0-eie63）

当当前 XLEN = 32 时，寄存器 *eiek* 包含标识号为 *k* × 32 至 *k* × 32 + 31 的中断的使能位。对于该范围内已执行的中断标识 *i*，中断 *i 的*使能位是寄存器 *eiek* 的位（*i* mod 32）。

当当前 XLEN = 64 时，奇数寄存器 eie1、eie3、...eie63 不存在。在这种情况下，如果 *iselect CSR 是 0xC1-0xFF 范围内的奇数值，试图访问匹配的 *ireg CSR 会引发非法指令异常，除非在 VS 模式下进行，否则会引发虚拟指令异常。对于偶数 *k*，寄存器 *eiek* 包含身份号为 *k* × 32 至 *k* × 32 + 63 的中断的使能位。对于该范围内已执行的中断标识 *i*，中断 *i 的*使能位是寄存器 *eiek* 的位（*i* mod 64）。

有效 *eiek* 寄存器中与支持的中断标识不对应的位位置（如 eie0 的第 0 位）为只读 0。

3.9 Top 外部中断 CSR（mtopei、stopei、vstopei）

CSR mtopei 直接与 IMSIC 的 machine-level 中断文件交互。如果实现了 supervisor 模式，CSR stopei 会直接与 supervisor-level 中断文件交互。如果实现了 hypervisor extension，且 hstatus 的字段 VGEIN 是已执行的 guest 中断文件的编号，则 vstopei 会与所选的 guest 中断文件交互。

**topei CSR*（mtopei、stopei 或 vstopei）的值表示中断文件当前最高优先级的挂起并且使能的中断，如果 eithreshold 不为零，该中断还超过了 eithreshold 寄存器指定的优先级阈值。标识号较小的中断优先级较高。

如果在中断文件的 eip 数组中没有挂起的中断，在 eie 数组中也没有启用的中断，或者如果 eithreshold 不为零，且没有挂起和启用的中断的标识号小于 eithreshold 的值，则读取 *topei CSR 的返回值为零。否则，读取 *topei 返回的值格式如下：

位 26:16　中断标识
位 10:0　中断优先级（与标识相同）

所有其他位均为 0。

在 *topei CSR 中报告的中断标识是 hart 外部中断的次要标识。

---

*从 *topei *CSR 中读取的冗余值与高级 PLIC 一致，后者以上述相同格式返回中断标识号及其优先级，但这两部分相互独立。*

对 *topei CSR 的写入会清除中断文件中的挂起位，从而 *claim* 所报告的中断标识。写入的值将被忽略；相反，寄存器的当前可读值将决定哪个中断挂起位被清除。具体来说，当写入 *topei CSR 时，如果寄存器值 26:16 位的中断标识为 *i*，那么中断文件中中断 *i 的*挂起位将被清零。当 *topei CSR 的值为零时，对寄存器的写入不起作用。

如果通过一条 CSR 指令（CSRRW、CSRRS 或 CSRRC）同时读写 *topei CSR，则读取返回的值表示被清零的挂起位。

---

*如果不同时读取**topei寄存器，就写入**topei寄存器，几乎总是错误的。特别要注意的是，如果对**topei寄存器的读取和随后对该寄存器的写入是由两条不同的CSR指令完成的，那么在这两条指令之间，中断文件中可能会有一个优先级更高的中断被挂起并启用，从而导致写入指令清除了新中断的挂起位，而不是读取指令所报告的中断。一旦新中断的待清除位被清除，中断就会丢失。*

*如果需要先读取 *topei *CSR，然后再单独 claim 中断，则可以通过清除* eip *数组中的挂起*

*位来安全地申请中断，*而不是写入 *topei。

3.10 中断交付和处理

IMSIC 的中断文件向连接的 hart 提供*外部中断*信号，每个中断文件提供一个中断信号。来自 machine-level 中断文件的中断信号在 CSR mip 中显示为 MEIP，而来自 supervisor-level 中断文件的中断信号在 mip 和 sip 中显示为位 SEIP。来自任何 guest 中断文件的中断信号在 hypervisor CSR hgeip 中显示为活动位。

当中断文件的 eidelivery 寄存器禁用中断发送（eidelivery = 0）时，来自中断文件的中断信号将保持非有效（假）状态。启用中断文件的中断传送（eidelivery = 1）后，如果且仅当中断文件有一个挂起且已启用的中断，且该中断也超过了 eithreshold 指定的优先级阈值（如果不为零）时，其中断信号才会被确认。

通过 IMSIC 编写一个专门用于外部中断的陷阱处理程序大致如下：

保存处理器寄存器
i = 读取 CSR mtopei 或 stopei，同时写入以申请（claim）中断
i = i>>16
调用外部中断 i（次要标识）的中断处理程序
恢复处理器寄存器
复归（return from trap）

第二步中 mtopei 或 stopei 的组合读写可由一条 CSRRW machine 指令完成、

csrrw *rd* , mtopei/stopei, x0

其中，*rd* 是值 i 的目的寄存器。

第 4 章

4.高级平台级中断控制器（APLIC）

在 RISC-V 系统中，平台级中断控制器（PLIC）负责处理通过信号线而非 MSI 发出信号的外部中断。当系统中的 RISC-V 硬件没有 IMSIC 时，硬件本身也不支持 MSI，此类硬件的所有外部中断都必须通过 PLIC。但是，即使在有 IMSIC 且大多数中断都通过 MSI 通信的计算机中，某些设备中断仍通过专用线路发出信号的情况也很常见。特别是对于不需要在系统中启动总线事务的设备（或设备控制器）来说，支持 MSI 的成本尤其高昂，因此有线中断是一种省钱的替代方案。与 MSI 不同的是，有线中断继续得到当前所有计算机平台的普遍支持，这也是许多商品设备或控制器选择有线中断而非 MSI 的另一个原因，除非实现的标准（如 PCI Express）要求使用 MSI。

本章介绍*高级 PLIC*（APLIC），它与早期的 RISC-V PLIC 并不向后兼容。完全符合高级中断体系结构需要 APLIC。不过，如果只对 Harts（而非 MSI）进行有线中断，也可以用较早的 PLIC 代替，建立一个可行的系统。

我们打算最终提供一个用可移植 SystemVerilog 编写的 APLIC 的免费参数化实现示例，预计无需修改即可适用于许多 RISC-V 系统。

目前已有一份 Duo-PLIC *规范草案，该规范可通过软件配置来充当原始 RISC-V PLIC 或 APLIC。但目前看来，RISC-V 国际协会不太可能将 Duo-PLIC 规范批准为标准。*

在没有 IMSIC 的机器中，每个 RISC-V hart 都接受来自一个 PLIC 或 APLIC *的*中断，该 PLIC 或 APLIC 是hart 的*外部中断控制器*。Hart 的外部中断控制器（PLIC 或 APLIC）通过专用连接（通常是信号线）向 Hart 发送中断信号，以满足 Hart 可以接收中断的每个权限级别。(回顾第 4 页图 1.1）。

没有 IMSIC 的系统通常只有一个 PLIC 或 APLIC，作为所有 RISC-V 硬件的外部中断控制器。

由于每个不带 IMSIC 的 RISC-V hart 都有一个 PLIC 或 APLIC 作为外部中断控制器，因此具有多个 APLIC 的系统必须将 hart 划分为不同的子集，使每个 APLIC 成为单独子集 hart 的外部中断控制器。虽然这种安排并未被禁止，但其效率可能低于所有 Harts 共享一个 APLIC。

采用 IMSIC 作为外部中断控制器的 RISC-V 硬件只能以 MSI 的形式接收外部中断。在这种情况下，APLIC 的作用就是将有线中断转换为 MSI 供 Harts 使用。(回顾第 4 页的图 1.2）可以说 APLIC 通过发送 MSI 将传入的有线信号中断转发给 harts。

当 Harts 有支持 MSI 的 IMSIC 时，系统很容易包含多个 APLIC，用于将有线中断转换为 MSI，每个 APLIC 转发来自不同设备子集的中断。当设备组之间物理距离较远，甚至可能位于不同芯片上（包括多芯片模块中的芯片组）时，可能更容易出现多个 APLIC。

4.1 中断源和标识

单个 APLIC 支持固定数量的*中断源*，与 APLIC 的物理输入中断线完全对应。通常情况下，每个中断源的输入线都与单个设备或设备控制器的输出中断线相连。(对于电平敏感型中断，可将多个设备或控制器的中断输出组合起来，以驱动 APLIC 上单个中断源的输入线）。举例来说，如果中断源始终被配置为 "分离"，那么中断源的输入线也可以简单地绑定为高电平或低电平。有关*源模式*的说明，请参见第 4.5.2 节）。

APLIC 的每个中断源都有一个固定的唯一*标识号*，范围从 1 到 *N*，其中 *N* 是 APLIC 的中断源总数。在 APLIC 中，0 不是有效的中断标识号。APLIC 最多可支持 1023 个中断源。

当 APLIC 直接向给定权限级别的 Harts 发送中断（而不是作为 MSI 监控中断）时，APLIC 就是该权限级别 Harts 的外部中断控制器，APLIC 的中断标识直接成为 Harts 外部中断*的次要标识*。

另一方面，当 APLIC 通过 MSI 转发中断时，软件会为每个源的出站 MSI 配置一个新的中断间标识号。因此，在这种情况下，特定 APLIC 的源标识号只能区分 APLIC 的传入中断，与 APLIC 外部无关。

4.2 中断域

APLIC 支持一个或多个*中断域*，每个中断域与一个权限级别（machine 或 supervisor-level）的 RISC-V Harts 子集相关联。一个中断域内的 Harts 是该域在相应权限级别下可以中断的 Harts。每个域在机器地址空间中都有自己的内存映射控制区，看似控制一个完整、独立的 APLIC，但实际上所有域接口都一起访问一个组合中断控制器。

图 4.1 至图 4.3 描述了 RISC-V 系统中 APLIC 可能实现的中断域层次结构。

第一张图表示一个最小的系统，该系统只有一个不支持 supervisor 模式的 Hart，该 Hart 上的 machine-level 只有一个中断域。下图 4.2 显示的是为对称多处理（SMP）设计的较大系统的基本安排，该系统有多个均执行 supervisor 模式的 hart。在这种情况下，如图所示，APLIC 通常会为 supervisor-level 提供一个单独的中断域。这种 supervisor-level 中断域允许在多处理器上以 S 模式运行的操作系统直接控制其接收的中断，而无需调用 M 模式来行使控制权。

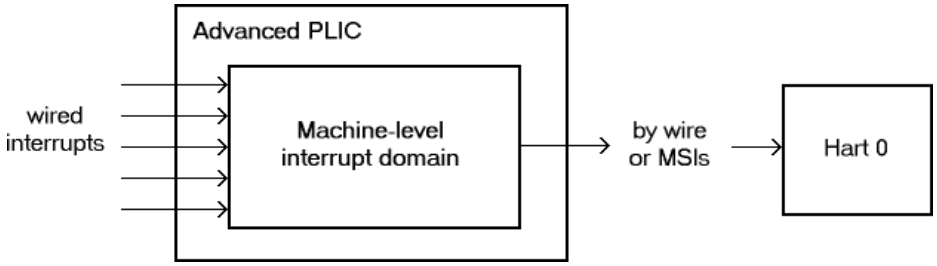

图 4.1：一个 RISC-V 系统的示例，该系统有一个只执行 M 模式的 Hart，该 Hart 有一个 machine-level 中断域。

APLIC 的中断域以树状层次结构排列，根域始终处于 machine-level。传入的中断线首先到达根域。然后，每个域可选择性地将全部或部分中断源委托给其在层次结构中的子域。在给定的 APLIC 中，所有域的中断源编号都是不变的，因此源标识号 *i* 在每个域中总是指相同的源，与传入线编号 *i* 相对应。

图 4.3 显示了三个中断域的层次结构，其中两个在 machine-level，一个在 supervisor-level。图中的安排与 PMP（物理内存保护）相结合，使 machine-level 软件可以隔离一些中断，专供 Hart 0 使用，即使在 machine-level 也不受四个应用 Hart 的影响。

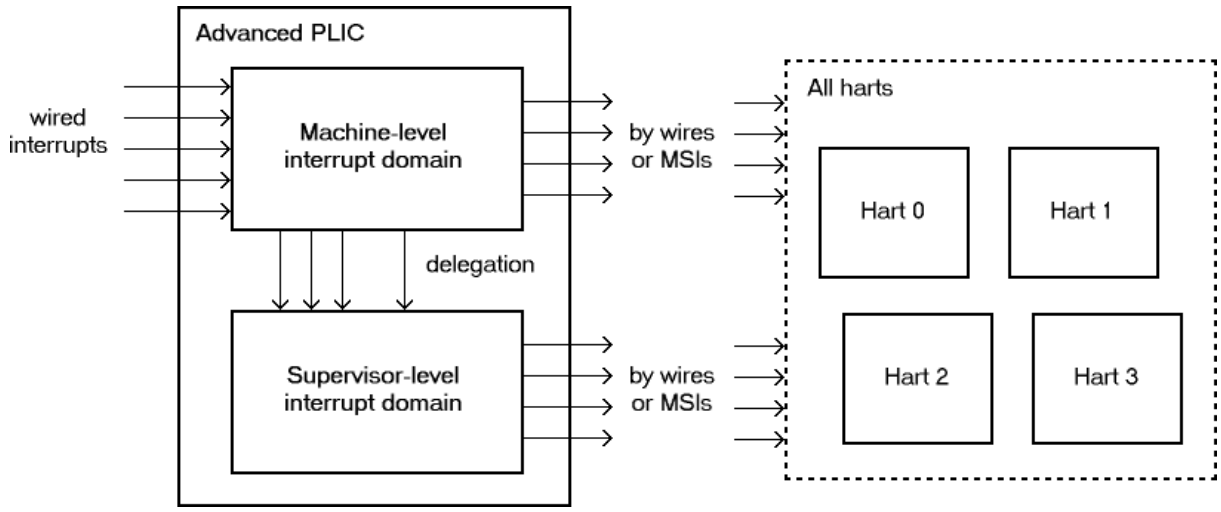

图 4.2: 一个示例系统，有四个实现 M 模式和 S 模式的 Harts，有两个 APLIC 中断域，machine-level 和 supervisor-level 各一个。

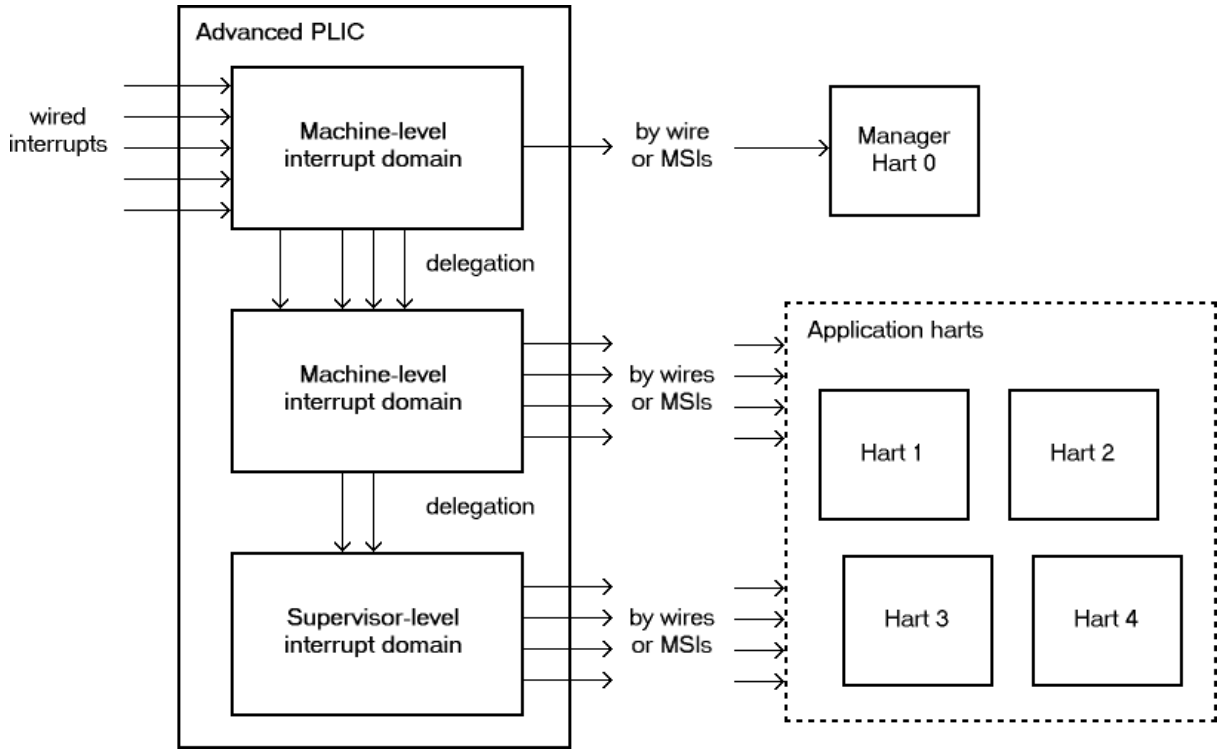

图 4.3：RISC-V 系统对图 4.2 中的示例进行了扩展，增加了第五个仅有 M 模式的 "管理器 "Hart，在其他域之上有一个独立的 machine-level 中断域。

为了让中断域内的 Harts 直接控制来自该中断域的中断，Harts 必须由相同权限级别的软件共同控制。特别是，单个操作系统应控制与 supervisor-level 中断域相关的所有 Harts。在图 4.2 和图 4.3 的示例中，APLIC 的 supervisor-level 中断域的控制权无法在多个独立操作系统之间安全分割。

考虑到图中描述的域层次结构，如果有必要为多个操作系统划分application harts，machine-level 软件将需要防止操作系统直接访问 Supervisor 级中断域，而是为控制 APLIC 中断提供 SBI 服务，或者模拟不同 Supervisor 级中断域的控制接口，每个操作系统一个。需要注意的是，这种仿真仍可使用 APLIC 的物理 supervisor-level 中断域，但由 machine-level 软件控制。

APLIC 的中断域层次结构符合这些规则：

- 根域位于 machine-level。
- 任何 supervisor-level 中断域的父域都是一个 machine-level 域，它至少包括相同的 harts（但显然是 machine-level 的）。父域在 machine-level 可能有更多的中断点。
- 对于每个中断域，来自该中断域的中断信号都是通过线或 MSI 以相同的方式发送给所有 Harts，而不是通过 Harts 之间的混合方式发送。

当 RISC-V hart 的外部中断控制器是 APLIC 而不是 IMSIC 时，hart 在每个权限级别只能处于该 APLIC 的一个中断域内。

另一方面，使用 IMSIC 作为外部中断控制器的 hart，在每个权限级别都可能处于多个 APLIC 中断域中，甚至是同一 APLIC 的中断域中，并有可能接收来自 machine 中多个不同 APLIC 的 MSI。

平台可能会为软件提供一种方法，在任何给定 APLIC 的多个中断域层次结构之间进行选择。任何此类可配置性都超出了本规范的范围，而应只在 machine-level 上提供。

4.3 Hart 索引编号

在给定的中断域中，域的每个中断组都有一个唯一的*索引号*，其范围从 0 到 2^{14} - 1 (= 16,383)。域与 hart 相关联的索引号可能与 RISC-V 特权架构分配给 hart 的唯一 hart 标识符（"hart ID"）有任何关系，也可能没有任何关系。两个不同的中断域可能会为同一组 hart 使用完全不同的索引号。但是，如果 APLIC 的任何中断域都能通过 MSI 转发中断，那么 APLIC 的所有 machine-level 域都能共享索引号与 hart 的共同映射。

为提高效率，实现时应优先选择小整数作为 Hart 索引号。

4.4 单域中断控制概述

APLIC 实现的每个中断域都有自己独立的物理控制接口，该接口在机器的地址空间中进行内存映射，这样就可以通过 PMP（物理内存保护）和基于页面的地址转换轻松调节对每个域的访问。所有中断域的控制接口都有一个共同的结构。在大多数情况下，每个域在软件看来都像是一个根域，在层次结构中看不到它上面的域。

对于 APLIC 的每个中断源，单个中断域由以下部分组成：

- 源配置。这决定了特定源在域中是否处于活动状态，如果处于活动状态，则决定了传入线缆的解释方式，如电平敏感或边沿敏感。对于在域中处于非活动状态的源，源配置可控制对子域的任何委托。
- 中断挂起位和中断使能位。对于非活动源，这两个位为只读 0。否则，挂起位将记录已到达但尚未发出信号或转发的中断，而启用位则决定当前是否应发送来自该中断源的中断，还是应保持挂起状态。
- 目标选择。对于活动源，目标选择决定了接收中断的 Hart 以及中断的优先级或作为 MSI 转发时的新中断标识。

对于直接将中断发送到 hart 而不是由 MSI 转发的中断域，该域有一组控制中断发送到 hart 的最终组件，域中的每个 hart 都有一个实例。

虽然具有多个中断域的 APLIC 看起来会重复上面列出的每个中断源的状态（中断源配置等），但实际上，APLIC 的实现可以利用每个中断源最终只在一个中断域中处于活动状态这一事实。在未授权特定中断源的所有域中，与中断源相关的状态显示为只读零，不需要物理寄存器位。

4.5 中断域的内存映射控制区

APLIC 支持的每个中断域都有一个专用的内存映射控制区，用于管理该域中的中断。该控制区域的大小是 4 KiB 的倍数，并与 4 KiB 地址边界对齐。最小的有效控制区域为 16 KiB。中断域的控制区域由一组 32 位寄存器组成。前 16 KiB 包含表 4.1 所列的寄存器。

从偏移量 0x4000 开始，中断域的控制区域可选择包含一个*中断发送控制*（IDC）结构阵列，在 0 到某个最大值（至少与中断域的最大 hart 索引号一样大）的范围内，每个潜在 hart 索引号对应一

个 IDC 结构。

Offset	size	register name
0x0000	4 bytes	domaincfg
0x0004	4 bytes	sourcecfg[1]
0x0008	4 bytes	sourcecfg[2]
. . .	. . .	
0x0FFC	4 bytes	sourcecfg[1023]
0x1BC0	4 bytes	mmsiaddrcfg (machine-level interrupt domains only)
0x1BC4	4 bytes	mmsiaddrcfgh ”
0x1BC8	4 bytes	smsiaddrcfg ”
0x1BCC	4 bytes	smsiaddrcfgh ”
0x1C00	4 bytes	setip[0]
0x1C04	4 bytes	setip[1]
. . .	. . .	
0x1C7C	4 bytes	setip[31]
0x1CDC	4 bytes	setipnum
0x1D00	4 bytes	in_clrip[0]
0x1D04	4 bytes	in_clrip[1]
. . .	. . .	
0x1D7C	4 bytes	in_clrip[31]
0x1DDC	4 bytes	clripnum
0x1E00	4 bytes	setie[0]
0x1E04	4 bytes	setie[1]
. . .	. . .	
0x1E7C	4 bytes	setie[31]
0x1EDC	4 bytes	setienum
0x1F00	4 bytes	clrie[0]
0x1F04	4 bytes	clrie[1]
. . .	. . .	
0x1F7C	4 bytes	clrie[31]
0x1FDC	4 bytes	clrienum
0x2000	4 bytes	setipnum_le
0x2004	4 bytes	setipnum_be
0x3000	4 bytes	genmsi
0x3004	4 bytes	target[1]
0x3008	4 bytes	target[2]
. . .	. . .	
0x3FFC	4 bytes	target[1023]

表 4.1：中断域内存映射控制区前 16 KiB 的寄存器

IDC 结构仅在中断域配置为直接向 Harts 发送中断而不是由 MSI 转发时使用。如果中断域只支持由 MSI 转发中断，而不支持由 APLIC 直接发送中断，则其控制区域中不需要 IDC 结构。

第一个 IDC 结构（如果有）用于索引号为 0 的 hart；第二个 IDC 结构用于索引号为 1 的 hart；以此类推。每个 IDC 结构由 32 个字节组成，并具有以下定义的寄存器：

偏移	尺寸	注册名
0x00	4 个字节	idelivery
0x04	4 个字节	iforce
0x08	4 个字节	ithreshold
0x18	4 个字节	topi
0x1C	4 个字节	claimi

IDC 结构是连续打包的，每个结构 32 字节，因此从中断域控制区域的起点到第二个 IDC 结构（hart 索引 1）（如果存在）的偏移量是 0x4020；到第三个 IDC 结构（hart 索引 2）（如果存在）的偏移量是 0x4040；等等。

IDC 结构阵列中可能包括一些*潜在的* Hart 索引号，但这些索引号并不是域中*实际的* Hart 索引号。例如，第一个 IDC 结构总是用于 hart 索引 0，但 0 不一定是域中任何 hart 的有效索引号。对于数组中不对应域中有效 hart 索引号的每个 IDC 结构，IDC 结构的寄存器可能（也可能不）都是只读 0。

除了表 4.1 中的寄存器和上面列出的 IDC 结构寄存器外，中断域控制区域中的所有其他字节都是保留字节，并以只读 0 字节的形式实现。

在中断域的控制区域内，只支持自然对齐的 32 位简单读写。对只读字节的写入将被忽略。对于其他形式的访问（其他大小、不对齐访问或 AMO），实现时最好报告访问故障或总线错误，否则必须忽略访问。

中断域控制区前 16 KiB 的寄存器（除 IDC 结构外的所有寄存器）将在下文中逐一介绍。IDC 结构将在后面的第 4.8 节 "由 APLIC 直接发送中断 "中介绍。

4.5.1 域配置 (domaincfg)

域控制寄存器的格式如下：

位 31:24	只读 0x80
位 8	IE
位 7	只读 0
位 2	DM (**WARL**)
位 0	BE (**WARL**)

所有其他寄存器位均为保留位，读作 0。

位 IE（中断使能）是该中断域中所有活动中断源的全局使能。只有当 IE = 1 时，挂起并启用的中断才会实际发出信号或转发给 Harts。

字段 DM（发送模式）为 **WARL**，决定了该中断域向 Harts 发送中断的方式。DM 的两个可能值是

0 = 直接传送模式
1 = MSI 输送模式

在*直接传送模式下*，中断由 APLIC 自身确定优先级并直接向各分支机构发出信号。在 *MSI 发送模式下*，APLIC 将中断作为 MSI 转发到中断中心，以便这些中断中心的 IMSIC 进一步处理。给定的 APLIC 实现可为每个中断域支持上述两种或其中一种传递模式。

如果中断域的 Harts 具有 IMSIC，那么除非这些 IMSIC 的相关中断文件支持寄存器 eidelivery 值 0x40000000，否则将 DM 设置为零（直接交付模式）与将 IE 设置为零的效果相同。参见第 3.8.1 节和第 4.8.2 节。

BE（Big-Endian）是一个 **WARL** 字段，用于确定中断域内存映射控制区中大多数寄存器的字节顺序。如果 BE = 0，字节顺序为小端；如果 BE = 1，字节顺序为大端。对于只支持 little-endian 的 RISC-V 系统，BE 可以是只读的 0，而对于只支持 big-endian 的系统，BE 可以是只读的 1。对于双显系统，BE 是可写的。

字段 BE 与中断域控制区域内的其他寄存器一样，会影响访问 domaincfg 寄存器本身的字节顺序。为了解决这个问题，domaincfg 最高有效字节 31:24 位的只读值有两个作用。首先，在对 domaincfg 进行任何读取时，都可以通过获得的四个字节值轻松确定该寄存器的正确字节顺序：按照正确的字节顺序解释时，第 31 位为 1，而按照错误的顺序解释时，第 31 位为 0。其次，如果 BE 的值不确定（大概是在软件初始化中断域之前），可以通过写入 (*x* << 24) | *x* 将 8 位值 *x* 安全地写入 domaincfg，其中 << 24 表示左移 24 位，竖条 (|) 表示位逻辑 OR。写入一次 domaincfg 后，BE 的值就应该是已知的了，因此以后的写入就不需要重复同样的技巧了。

在系统复位时，包括 IE 在内的 domaincfg 中的所有可写位都被初始化为零。如果某个实现支持 APLIC 的其他复位形式，那么这些其他复位会如何影响 domaincfg，则由实现定义（也可能由平台定义）。

4.5.2 源配置 (sourcecfg[1]-sourcecfg[1023])

对于每个可能的中断源 *i*，寄存器 sourcecfg[*i*] 控制该中断域中源 *i* *的源模式*，以及将源委托给子域的任何授权。当中断源 *i* *未被*实现或在本中断域中未被实现时，寄存器 sourcecfg[*i*] 为只读 0。如果源 *i* 未委托给本域，但随后（在父域）被更改为委托给本域，则 sourcecfg[*i*] 将保持为零，直到成功写入非零值。

sourcecfg[*i*] 的第 10 位是一个 1 位字段，称为 D（委托）。如果 D = 1，则表示源 *i* 已委托给子域；如果 D = 0，则表示未委托给子域。sourcecfg[*i*] 其他内容的解释取决于字段 D。

当中断源 *i* 被委托给子域时，sourcecfg[*i*] 的格式如下：

位 10	D, = 1
位 9:0	子域索引（**WLRL**）

所有其他寄存器位均为保留位，读作 0。

子域索引（Child Index）是一个 **WLRL** 字段，用于指定此源代码所对应的中断域。对于具有 *C* 个子域的中断域，该字段必须能够容纳范围在 0 到 *C* - 1 之间的整数值。每个中断域都有一个从这些索引号到子域的固定映射。

如果中断域在域层次结构中没有子域，则该域的任何源代码控制寄存器中的位 D 都不能置 1。对于这样的叶域，如果试图写入一个源代码控制寄存器，而该寄存器的值为第 10 位 = 1，则整个寄存器的值将被设置为 0。

当中断源 *i* 未委托给子域时，sourcecfg[*i*] 的格式如下：

位 10　　D, = 0
位 2:0　　SM (**WARL**)

所有其他寄存器位均为保留位，读作 0。

SM（源模式）字段为 **WARL** 字段，用于控制中断源在此域中是否处于活动状态，如果处于活动状态，传入线路上的哪些值或转换将被解释为中断。表 4.2 列出了 SM 的允许值及其含义。字段 SM 始终支持非活动（零）。对于每个中断源，实现者可自由选择 SM 支持的其他值。

值	名称	说明
0	Inactive	在本领域内不活跃（未授权）
1	Detached	激活，脱离源信号线
2-3	-	*保留*
4	Edge1	有效，边沿敏感；上升沿时拉起中断
5	Edge0	有效，边沿敏感；下降沿时拉起中断
6	Level1	有效，电平敏感；高电平时拉起中断
7	Level0	有效，电平敏感；低电平时拉起中断

表 4.2：当 D 位 = 0 时，源代码配置寄存器 SM（源模式）字段的编码

如果中断源被授权给子中断域（D = 1）或未被授权（D = 0）且 SM 为非活动状态，则该中断源在中断域中处于非活动状态。当中断源 *i* 在中断域中处于非活动状态时，域中相应的中断挂起位和中断使能位将为只读 0，寄存器 target[*i*] 也将为只读 0。如果中断源 *I* 从非激活模式变为激活模式时，中断源的挂起位和使能位保持为零，除非由于本节后面或第 4.7 节中指定的原因而自动设置，并且 target[*i*] 的定义子字段获得未指定的值。

当一个信号源被配置为 "Detached" 模式时，其信号线输入将被忽略；不过，中断挂起位仍可通过写入 setip 或 setipnum 寄存器来设置。(例如，这种模式可用于接收 MSI）。

边沿敏感源可配置为在上升沿（低电平到高电平的转换）或下降沿（高电平到低电平的转换）识别进入的中断。当配置为下降沿（模式 Edge0）时，信号源被称为*反相*。

电平敏感源可配置为将信号线上的高电平（1）或低电平（0）解释为中断信号。当配置为低电平（模式 Level0）时，信号源被称为*反相*。

对于配置为边沿敏感或电平敏感的中断源，定义

校正后的输入值 = (incoming wire value) XOR (source is inverted).

对于未激活或 Detached 的信号源，*校正后的输入值*为零。

如果在新的源模式下校正后的输入值为高（= 1），对源控制寄存器的任何写入都可能（或不可能）导致相应的中断挂起位被置 1。对源控制寄存器的写入本身不会导致挂起位被清零，除非源处于非活动状态。(但请参见第 4.7 节）。

4.5.3 Machine MSI 地址配置（mmsiaddrcfg 和 mmsiaddrcfgh）

对于 machine-level 中断域，寄存器 mmsiaddrcfg 和 mmsiaddrcfgh 可选择提供参数，用于确定写

入外发 MSI 的地址。

如果 APLIC 的任何中断域都不支持 MSI 发送模式（domaincfg.DM 对所有域都是只读零），则这两个寄存器对任何域都不起作用。否则，这两个寄存器将在根域执行，其他 machine-level 域可能执行，也可能不执行。对于非 machine-level 域，这两个寄存器永远不会被执行。当一个域不执行 mmsiaddrcfg 和 mmsiaddrcfgh 时，这两个寄存器位置上的 8 个字节与其他保留字节一样，都是只读 0。

寄存器 mmsiaddrcfg 和 mmsiaddrcfgh 只能在根域中写入。对于实现这两个寄存器的所有其他 machine-level 域，它们都是只读寄存器。

实现时，mmsiaddrcfg 的格式如下：

位 31:0　Low Base PPN (**WARL**)

mmsiaddrcfgh 的格式如下：

位 31　L
位 28:24　HHXS（**WARL**）
位 22:20　LHXS（**WARL**）
位 18:16　HHXW (**WARL**)
位 15:12　LHXW (**WARL**)
位 11:0　High Base PPN (**WARL**)

mmsiaddrcfgh 的所有其他位均为保留位，读作 0。

来自 mmsiaddrcfgh 的字段 High Base PPN 和来自 mmsiaddrcfg 的字段 Low Base PPN 连接起来形成 44 位 Base PPN（物理页码）。该值和字段 HHXS（高 Hart 索引移位）、LHXS（低 Hart 索引移位）、HHXW（高 Hart 索引宽度）和 LHXW（低 Hart 索引宽度）用于确定 MSI 的目标地址，稍后将在第 4.9.1 节中介绍。

当 mmsiaddrcfg 和 mmsiaddrcfgh 可写入（仅限根域）时，除 L 外的所有字段均为 **WARL**。实现者可自由选择支持哪些值。通常情况下，某些位是可写的，而其他位则是只读常量。在极端情况下，所有字段的值都可能是固定不变的。

如果 mmsiaddrcfgh 中的位 L 设置为 1，mmsiaddrcfg 和 mmsiaddrcfgh 将被*锁定*，对寄存器的写入将被忽略，寄存器实际上成为只读寄存器。当 L = 1 时，mmsiaddrcfg 和 mmsiaddrcfgh 中的其他字段可选择全部读为 0。在这种情况下，如果在根域中首次设置 L 时，这些其他字段的值为非零，则 APLIC 将保留其内部值，但在读取 mmsiaddrcfg 和 mmsiaddrcfgh 时将不再可见。

将 mmsiaddrcfgh.L 设置为 1 也会锁定下一小节中描述的寄存器 smsiaddrcfg 和 smsiaddrcfgh（如果这些寄存器也已实现）。

对于根域，L 在系统重置时被初始化为 0 或 1，以适合特定 APLIC 实现的方式为准。如果重置时将 L 初始化为 1，要么 APLIC 将其他字段硬连接为常量，要么 APLIC 在本标准之外采用其他方法确定外发 MSI 写入地址。在后一种情况下，mmsiaddrcfg 和 mmsiaddrcfgh 中的其他字段可能全部读为 0，因此寄存器 mmsiaddrcfg 和 mmsiaddrcfgh 分别只有只读值 0 和 0x80000000。如果 mmsiaddrcfg 或 mmsiaddrcfgh 的值不同（分别不是 0 或 0x80000000），则根据第 4.9.1 节的规定，向 machine-level 写入的 MSI 外发地址必须可以从这些寄存器的可见值推导出来。

对于非根域的 machine-level 域，如果执行了这些寄存器，则 L 位始终为 1，其他字段要么是根域 mmsiaddrcfg 和 mmsiaddrcfgh 的只读副本，要么全为 0。

如果允许软件任意确定发送 MSI 的地址（即使只允许在 machine-level 使用），就可以绕过物理内存保护 (PMP)。对于支持 MSI 发送模式的 APLIC，如果可行，建议 APLIC 在内部硬连接所有目标 IMSIC 的物理地址，使软件无法更改这些地址。不过，并非所有 APLIC 实现都能遵循这一建议。

预计大多数系统都会将目标 IMSIC 的物理地址与 hart 索引号进行简单的线性对应。(参见第 3.6 节。）寄存器 mmsiaddrcfg *和* mmsiaddrcfgh *（以及下一小节中的* smsiaddrcfg *和* smsiaddrcfgh*）允许可信度足够高的 machine-level 软件在系统重置后尽早配置目标 IMSIC 的物理地址 pattern，然后锁定该配置，防止后续篡改。*

实际内部硬连接 IMSIC 地址的 APLIC 可以简单地将这些寄存器设置为只读寄存器，值为 0 和 0x80000000*。或者，如果 IMSIC 地址必须由软件计算，但计算公式过于复杂，寄存器* mmsiaddrcfg *和* mmsiaddrcfgh *无法处理，也可以将这些寄存器简单地作为只读寄存器来实现，其值为 0 和* 0x80000000*，并提供一个单独的自定义机制来配置 IMSIC 地址。*

如果 APLIC 除系统复位外还支持其他形式的复位，那么这些其他复位将如何影响根域中的 mmsiaddrcfg 和 mmsiaddrcfgh（以及 smsiaddrcfg 和 smsiaddrcfgh），则由实现定义（或可能由平台定义）。不过，权限不足的软件不可能通过将位 L 改回 0 来使用局部重置解锁这些寄存器。因此，可能只有系统完全重置才会影响这些寄存器，其他任何重置都不会。

4.5.4 Supervisor MSI 地址配置（smsiaddrcfg 和 smsiaddrcfgh）

对于 machine-level 中断域，寄存器 smsiaddrcfg 和 smsiaddrcfgh 可选择提供参数，供 supervisor level域确定写出 MSI 的地址。

如果某个域实现了 mmsiaddrcfg 和 mmsiaddrcfgh，并且 APLIC 至少有一个 supervisor-level 中断域，那么该域就能实现寄存器 smsiaddrcfg 和 smsiaddrcfgh。如果寄存器未执行，则其位置上的 8 个字节与其他保留字节一样，都是只读 0。

与 mmsiaddrcfg 和 mmsiaddrcfgh 寄存器一样，寄存器 smsiaddrcfg 和 smsiaddrcfgh 也只能在根域中写入。对于实现这两个寄存器的所有其他 machine-level 域，它们都是只读寄存器。

实现时，smsiaddrcfg 的格式如下：

位 31:0	Low Base PPN (**WARL**)

和 smsiaddrcfgh 都是这种格式：

位 22:20	LHXS (**WARL**)
位 11:0	High Base PPN (**WARL**)

smsiaddrcfgh 的所有其他位均为保留位，读作 0。

来自 smsiaddrcfgh 的字段 High Base PPN 和来自 smsiaddrcfg 的字段 Low Base PPN 合并形成 44 位 Base PPN（物理页码）。该值和字段 LHXS（低Hart 索引位移）用于确定 MSI 的目标地址，稍后将在第 4.9.1 节中介绍。

当 smsiaddrcfg 和 smsiaddrcfgh 可写入（仅限根域）时，所有字段均为 **WARL**。与 mmsiaddrcfg 和 mmsiaddrcfgh 一样，执行者可自由选择支持的值。

如果域的寄存器 mmsiaddrcfgh 的 L 位设置为 1，则 smsiaddrcfg 和 smsiaddrcfgh 与 mmsiaddrcfg 和 mmsiaddrcfgh 一起被*锁定*为只读。当 mmsiaddrcfgh.L = 1 时，如果 mmsiaddrcfg 和 mmsiaddrcfgh 的可读值分别为 0 和 0x80000000（因为它们的其他字段被隐藏），则 smsiaddrcfg 和 smsiaddrcfgh 也会被隐藏并读作 0。

仅对于根域，如果 mmsiaddrcfgh.L = 1 且 MSI 地址配置字段被隐藏（因此 mmsiaddrcfgh 读作 0x80000000，寄存器 mmsiaddrcfg、smsiaddrcfg 和 smsiaddrcfgh 都读作 0），则无论

smsiaddrcfg 和 smsiaddrcfgh 在 mmsiaddrcfgh.L 首次设置时的值是多少，APLIC 内部都会保留这些值，尽管读取寄存器时已无法看到这些值。L 首次设置时的任何值，APLIC 内部都会保留，但读取寄存器时已无法看到这些值。另外，如果在根域中系统复位初始化了 mmsiaddrcfgh.L = 1，并且如果所有 MSI 地址配置字段从未出现过 0 以外的值，那么 APLIC 实现就有其他可能是非标准的方法来确定传出 MSI 的地址，这在上一小节 4.5.3 中讨论过。

如果 mmsiaddrcfg 和 mmsiaddrcfgh 分别不是只读的 0 和 0x80000000，则必须按照第 4.9.1 节的规定，根据寄存器 mmsiaddrcfgh、smsiaddrcfg 和 smsiaddrcfgh 的可见值推导出指向 Supervisor 的 MSI 写入地址。

对于非根域的 machine-level 域，如果执行了 smsiaddrcfg 和 smsiaddrcfgh 且它们不是只读零，那么它们就是根域中相同寄存器的只读副本。

4.5.5 设置中断挂起位（setip[0]-setip[31]

读取或写入寄存器 setip[*k*]，可读取或修改中断源 *k* × 32 至 *k* × 32 + 31 的挂起位。对于该范围内已执行的中断源 *i*，*中断源* *i* 的挂起位与寄存器位（*i* mod 32）相对应。

读取 setip 寄存器会返回相应中断源的挂起位。结果值中与已实现中断源不对应的位位置（如 setip[0] 的第 0 位）为零。

在写入 setip 寄存器时，如果写入的 32 位值中的每一位为 1，而该位位置对应的是活动中断源，则该中断源的中断挂起位将被置 1（如有可能）。有关写入 setip 寄存器可设置挂起位的具体时间，请参见第 4.7 节。

4.5.6 按编号设置中断挂起位（setipnum）

如果 *i* 是域中的活动中断源编号，则向寄存器 setipnum 写入 32 位值 i，可将中断源 *i* 的挂起位置 1。具体何时可通过写入 setipnum 设置挂起位，请参见第 4.7 节。

如果写入的值不是域中的活动中断源编号，则对 setipnum 的写入将被忽略。对 setipnum 的读取总是返回 0。

4.5.7 校正后输入值，清除中断挂起位（in_clrip[0]-in_clrip[31]）

读取 in_clrip[*k*] 中的寄存器会返回中断源 *k* × 32 至 *k* × 32 + 31 的校正后输入值（第 4.5.2 节），而写入 in_clrip[*k*] 则可能会修改相同中断源的挂起位。对于指定范围内的已实现中断源 *i*，*中断源* *i* 与寄存器位（*i* mod 32）相对应。

读取 in_clrip 寄存器会返回相应中断源的校正后输入值。结果值中与已执行的中断源不对应的位位置（如 in_clrip[0] 的第 0 位）为零。

在写入 in_clrip 寄存器时，如果写入的 32 位值中的每一位为 1，而该位对应的是活动中断源，则该中断源的中断挂起位将被清除（如有可能）。具体何时可通过写入 in_clrip 寄存器清除挂起位，请参见第 4.7 节。

4.5.8 按编号清除中断挂起位（clripnum）

如果 *i* 是域中的活动中断源编号，向寄存器 clripnum 写入 32 位值 i 将清除源 *i* 的挂起位。具体何时可通过向 clripnum 写入清除挂起位，请参见第 4.7 节。

如果写入的值不是域中的活动中断源编号，则对 clripnum 的写入将被忽略。对 clripnum 的读取总是返回 0。

4.5.9 设置中断使能位（setie[0]-setie[31]）

读写寄存器 setie[*k*] 可读取或修改中断源 *k* × 32 至 *k* × 32 + 31 的使能位。对于该范围内已实现的中断源 *i*，*中断*源 *i* 的使能位与寄存器位（*i* mod 32）相对应。

读取 setie 寄存器会返回相应中断源的使能位。结果值中与已执行的中断源不对应的位位置（如 setie[0] 的第 0 位）为零。

在写入设置寄存器时，如果写入的 32 位值中的每个位为 1，则该位位置对应的是活动中断源，则该中断源的中断使能位被设置为 1。

4.5.10 按数字设置中断使能位（setienum）

如果 *i* 是域中的活动中断源编号，则向寄存器 setienum 写入 32 位值 *I* 会将信号源 *i* 的使能位设置为 1。

如果写入的值不是域中的活动中断源编号，则对 setienum 的写入将被忽略。对 setienum 的读取总是返回 0。

4.5.11 清除中断使能位（**clrie**[0]**-clrie**[31]

写入寄存器 clrie[*k*] 可能会修改中断源 *k* × 32 至 *k* × 32 + 31 的使能位。对于该范围内已实现的中断源 *i*，*中断*源 *i* *的*使能位与寄存器位（*i* mod 32）相对应。

在写入 clrie 寄存器时，写入的 32 位值中每有一位为 1，就会清除该中断源的中断使能位。

对 clrie 寄存器的读取总是返回 0。

4.5.12 按编号清除中断使能位 **(**clrienum**)**

如果 *i* 是域中的活动中断源编号，则将 32 位值 *i* 写入寄存器 clrienum 会导致源 *i* 的使能位被清零。

如果写入的值不是域中的活动中断源编号，则对 clrienum 的写入将被忽略。对 clrienum 的读取总是返回 0。

4.5.13 按数字设置中断挂起位，小数点后一位（setipnum_le）

寄存器 setipnum_le 的作用与 setipnum 相同，只是字节顺序总是小端，就像寄存器 domaincfg 的 BE 字段（大端）为零一样。

对于只使用 big-endian 的系统（domaincfg.BE 硬连接为 1），则无需实现 setipnum_le，在这种情况下，该偏移量处的四个字节与其他保留字节一样，只是只读 0。

setipnum_le 可用作 MSI 的写端口。

4.5.14 按数字设置中断挂起位，**big-endian**（setipnum_be）

寄存器 **setipnum_be** 的作用与 setipnum 相同，只是字节顺序总是大端，就像寄存器 domaincfg 的字段 BE（大端）是 1。

对于只使用小endian 的系统（domaincfg.BE 硬连接为 0），无需实现 setipnum_be，在这种情况下，该偏移量处的四个字节与其他保留字节一样，只是只读的 0。

对于主要为 big-endian 字节序构建的系统，setipnum_be 可以作为某些设备 MSI 的写入端口。

4.5.15 生成 MSI (genmsi)

当中断域配置为 MSI 发送模式（domaincfg.DM = 1）时，寄存器 genmsi 可用于从 APLIC 向 Hart 发送*临时* MSI。该功能的主要目的是协助在 hart 向 APLIC 寄存器的写入和从 APLIC 向 hart 发送 MSI 之间建立一个临时的已知顺序，详见第 4.9.3 节。

出于其他目的，向 hart发送 MSI 通常最好直接写入 hart的 IMSIC，而不是使用 APLIC 作为中介。应尽量减少 genmsi *寄存器的使用，以避免其成为瓶颈。*

寄存器 genmsi 的格式是这样的：

位 31:18	Hart Index（**WLRL**）
位 12	Busy（**只读**）
位 10:0	EIID (**WARL**)

所有其他寄存器位均为保留位，读作 0。

Busy 位通常为 0（假），但写入 genmsi 会导致 "Busy" 位变为 1（真），表明外部 MSI 正在等待。Hart Index 字段指定目标 Hart，EIID（外部中断标识）指定 MSI 的数据值。Hart Index 和 EIID 字段的格式和行为与目标寄存器中的相同，详见下一小节 4.5.16.。对于 machine-level 中断域，外部 MSI 在 machine-level 发送给目标 hart；对于 supervisor-level 中断域，外部 MSI 在 supervisor-level 发送给目标 hart。

APLIC 在发送待发的外部 MSI 时应尽量减少延迟。一旦它离开 APLIC，且 APLIC 能够为另一个临时 MSI 接受新的写入 genmsi，Busy 将恢复为 false。之前从该 APLIC 发送到同一 hart 的所有 MSI 必须在 hart 的 IMSIC 上可见，然后才会在 hart 的 IMSIC 上可见临时 MSI。

如果 Busy 为真，则写入 genmsi 的内容会被忽略。

临时 MSI 不受域的 domaincfg 寄存器 IE 位的影响。即使 domaincfg.IE = 0，也会发送临时 MSI。

当中断域配置为直接发送模式（domaincfg.DM = 0）时，寄存器 genmsi 为只读零。

4.5.16 中断目标（target[1]-target[1023]）

如果中断源 *i* 在该域中处于非活动状态，寄存器 target[*i*] 则为只读 0。如果中断源 *i* 处于活动状态，寄存器 target[*i*] 将决定向哪个中断源发出信号或转发中断。target[*i*] 的确切解释取决于寄存器 domaincfg 的 DM 字段配置的发送模式。

如果更改 domaincfg.DM，域内所有活动中断源的目标寄存器将在为新交付模式定义的所有字段中获得未指定的值。

活动的源，直接传输模式

对于活动中断源 *i*，如果域配置为直接交付模式（domaincfg.DM = 0），则寄存器 target[*i*] 的格式

为

位 31:18　Hart Index (**WLRL**)
位 7:0　IPRIO (**WARL**)

所有其他寄存器位均为保留位，读作 0。

Hart Index 是一个 **WLRL** 字段，用于指定该中断源将向哪个 Hart 发送中断。

字段 IPRIO（中断优先级）指定中断源的*优先级编号*。该字段是 *IPRIOLEN* 位的 **WARL** 无符号整数，其中 IPRIOLEN 是给定 APLIC 的常量参数，范围为 1 至 8。IPRIO 的值只允许为 1 至 $2^{IPRIOLEN}$-1，不允许为 0。写入目标寄存器后，IPRIO 将等于所写入 32 位值的位（IPRIOLEN - 1）:0，除非这些位全为 0，在这种情况下，优先级编号将被设置为 1。(如果 IPRIOLEN = 1，这些规则将导致 IPRIO 有效为只读，值为 1）。

优先级数字越小，表示优先级越高。当中断源的优先级相同时，标识号最小的中断源优先级最高。

中断优先级采用整数编码，数字越小优先级越高，以匹配 IMSIC 的优先级编码。

活动的源，MSI 交付模式

对于活动中断源 *i*，如果域配置为 MSI 发送模式（domaincfg.DM = 1），那么寄存器 target[*i*] 的格式就是这样：

位 31:18　Hart Index (**WLRL**)
位 17:12　Guest Index（**WLRL**）
位 10:0　EIID (**WARL**)

第 11 位为保留位，读数为 0。

Hart Index（Hart 索引）字段指定了该中断源将转发到的 Hart。

如果中断域处于 supervisor-level，并且域的 Harts 执行了 RISC-V Privileged Architecture 的 hypervisor extension，那么 Guest Index 就是一个 **WLRL** 字段，必须能够容纳 0 到 GEILEN 范围内的所有整数值（参数 GEILEN 由 Privileged Architecture 的 hypervisor extension 定义）。（参数 *GEILEN* 由特权架构的 hypervisor extension 定义）。否则，Guest Index 字段将为只读 0。对于 supervisor-level 中断域，非零的 Guest Index 是目标 hart guest 中断文件的编号，MSI 将被发送到该文件。当 Guest Index 为零时，来自 supervisor-level 域的 MSI 将被转发到 supervisor-level 的目标 hart。对于 machine-level 域，Guest Index 为只读 0，MSI 将始终在 machine-level 转发给目标 hart。

寄存器 target[*i*] 的 Hart Index 和 Guest Index 字段共同决定了为中断源 *i* 转发的 MSI 的地址。其余字段 EIID（外部中断标识）指定了这些 MSI 的数据值，最终成为目标 hart 外部中断的次要标识。

如果中断域的 harts 具有实现 *N 个*不同中断标识的 IMSIC 中断文件（第 3.1 节），则 EIID 是一个 k *位*无符号整数字段，其中$\lceil \log_2 N \rceil \le k \le 11$。因此，EIID 至少能保存 0 到 *N 的*值。写入目标寄存器后，EIID 的 *k 个*执行位就等于写入的 32 位值中最不重要的 *k* 个位。

4.6　重置

APLIC 重置后，其所有状态都会变得有效和一致，但其他未指定的状态除外：

- 每个中断域的 domaincfg 寄存器（第 4.5.1 节）；
- 可能是 machine-level 中断域的 MSI 地址配置寄存器（第 4.5.3 和 4.5.4 节）；以及
- 每个中断域的 genmsi 寄存器（如果存在）的忙位（第 4.5.15 节）。

4.7 对中断挂起位的精确影响

通过写入中断域控制区寄存器来设置或清除中断源挂起位的尝试可能会成功，也可能不会成功，这取决于相应的源模式、中断域的发送模式以及源的校正输入值状态（定义见第 4.5.2 节）。下面列举了在给定源模式下设置或清除挂起位的所有情况。

如果源模式为 "Detached"：

- 只有向 setip 或 setipnum 寄存器写入相关内容后，挂起位才会被置 1。
- 当 APLIC 要求中断或 MSI 转发中断，或向 in_clrip 寄存器或 clripnum 写入相关内容时，挂起位清零。

如果源模式为 Edge1 或 Edge0：

- 校正后的输入值从低到高的转换，或对 setip 或 setipnum 寄存器的相关写入，都会将挂起位置 1。
- 当 APLIC 要求中断或 MSI 转发中断，或向 in_clrip 寄存器或 clripnum 写入相关内容时，挂起位清零。

如果源模式为 Level1 或 Level0，且中断域配置为直接交付模式（domaincfg.DM = 0）：

- 只要校正后的输入值为高，挂起位就会被置 1。写入 setip 或 setipnum 寄存器时不能设置挂起位。
- 当校正后的输入值为低时，挂起位清零。挂起位不会因 APLIC 中断请求而清零，也不会因对 in_clrip 寄存器或 clripnum 的写入而清零。

如果源模式为 Level1 或 Level0，且中断域配置为 MSI 发送模式（domaincfg.DM = 1）：

- 当校正后的输入值从低到高转换时，挂起位被置 1。当校正后的输入值为高时，挂起位也可通过写入 setip 或 setipnum 寄存器来设置，但当校正后的输入值为低时，挂起位不会被设置。
- 每当校正后输入值为低时、MSI 转发中断时、或向 in_clrip 寄存器或 clripnum 写入相关内容时，挂起位清零。

当中断域处于直接发送模式时，电平敏感源的挂起位始终只是校正后输入值的副本。即使在 MSI 发送模式下，当校正后的输入值为低时，电平敏感源的挂起位也永远不会被置位（= 1）。

除上述规则外，根据第 4.5.2 节的规定，写入源 cfg 寄存器可导致源的中断挂起位置 1。

4.8 由 APLIC 直接中断交付

当中断域处于直接交付模式（domaincfg.DM = 0）时，中断会通过一个唯一的信号（通常是一条专用线路）从 APLIC 发送到每个挂接点。在这种情况下，域的内存映射控制区末尾包含一个中断发送控制（IDC）结构数组，每个潜在的挂起索引有一个 IDC 结构。第一个 IDC 结构用于索引为 0 的 hart；第二个 IDC 结构用于索引为 1 的 hart；等等。

4.8.1 中断发送控制（IDC）结构

每个 IDC 结构有 32 个字节（自然对齐到 32 字节地址边界），并有这些定义的寄存器：

偏移	尺寸	寄存器名
0x00	4 个字节	idelivery
0x04	4 个字节	iforce
0x08	4 个字节	ithreshold
0x18	4 个字节	topi
0x1C	4 个字节	claimi

如果 IDC 结构的 hart 索引号不适用于中断域中的任何实际 hart，则这些寄存器可选择为只读 0。否则，这些寄存器将在下文中逐一说明。

在不完全了解系统在每个中断域中将使用的一组分组索引号的情况下，特定的 APLIC 可能被构建为支持最大数量的分组。在这种情况下，对于未使用的 Hart 索引号，APLIC 可能具有在 APLIC 内起作用（而不是只读 0）的 IDC 结构，但只是不与任何物理 Hart 连接。

4.8.1.1 中断交付启用（idelivery）

idelivery 是一个 **WARL** 寄存器，用于控制是否将针对相关 hart 的中断发送到 hart，以便在 hart 的 mip CSR 中显示为挂起中断。目前只为 idelivery 定义了两个值：

0 = 禁用中断发送
1 = 启用中断传送

如果 IDC 结构用于不存在的中断组（即对应的中断组索引号对中断域中的任何实际中断组无效），则将 idelivery 设为 1 不会向任何中断组发送中断。

4.8.1.2 强制中断（iforce）

iforce 是一个 **WARL** 寄存器，用于测试。只允许使用 0 和 1 的值。设置 iforce = 1 时，只要 domaincfg 的 IE 字段为 1，且 idelivery 寄存器向 hart 启用了中断传送功能，就会强制向相应的 hart 发出中断。当 topi 为 0 时，会为 hart 产生一个*虚假*的*外部中断*。

当读取寄存器 claimi 返回的中断标识为零时（表示虚假中断），iforce 将自动清零。

4.8.1.3 中断使能阈值（ithreshold）

ithreshold 是一个 WLRL 寄存器，用于确定向相应 hart 发出中断信号的最小中断优先级（最大优先级号）。寄存器 ithreshold 恰好实现了 IPRIOLEN 位，因此能够保存从 0 到 $2^{IPRIOLEN} - 1$ 的所有优先级。

当 ithreshold 为非零值 P 时，优先级为 P 或更高的中断源不会向 Hart 发送信号中断，就像这些中断源未启用一样，与中断使能位的设置无关。当 ithreshold 为零时，所有启用的中断源都可以

向 Hart 发送信号中断。

4.8.1.4 Top 中断 (topi)

topi 是一个只读寄存器，其值表示当前针对此 hart 的最高优先级的挂起且已启用的中断，如果不为零，该中断也超过了 ithreshold 指定的优先级阈值。

读取 topi 返回值时，如果没有以该中断为目标的中断处于挂起和启用状态，或者如果 ithreshold 不为零，且没有以该中断为目标的挂起和启用中断的优先级小于 ithreshold 值，则返回值为零。否则，读取 topi 返回的值格式如下：

位 25:16　　中断标识（源编号）

位 7:0　　中断优先级

所有其他位均为零。

topi 中报告的中断标识是目标 hart 外部中断的次要标识。

对 topi 的写入将被忽略。

4.8.1.5 Claim top 中断 (claimi)

寄存器 claimi 的值与 topi 相同。当该值不为零时，读取 claimi 会同时清除所报告中断标识的挂起位（如果可能）。有关读取 claimi 时清除挂起位的具体时间，请参见第 4.7 节。

从 claimi 读取返回值为零的数据，会同时产生设置 iforce 寄存器为零。

写入 claimi 的数据将被忽略。

4.8.2 中断交付和处理

当中断域配置为由 APLIC 直接向 harts 发送中断信号（domaincfg 的 DM 字段为零）时，APLIC 将按照域的权限级别为域内的所有 harts 提供*外部中断*信号，只要以下情况之一为真：(a) harts 没有 IMSIC，或 (b) 相关 IMSIC 中断文件的 eidelivery 寄存器设置为 0x40000000（第 3.8.1 节）。对于 machine-level 域，来自 APLIC 的中断信号在每个 hart 的 mip CSR 中显示为位 MEIP（machine 外部中断挂起）。对于 supervisor-level 域，中断信号在每个 hart 的 mip 和 sip CSR 中显示为位 SEIP（supervisor-level 外部中断挂起）。每个中断信号从 APLIC 发送到相应的 hart 时，都可以任意延迟。

在 APLIC 中，向 hart发送的每个中断信号都来自寄存器 domaincfg 的 IE 字段以及该域内存映射控制区中 hart IDC 结构的当前状态。如果 domaincfg.IE = 0 或 idelivery 寄存器禁用了向 hart 发送中断信号（idelivery = 0），则中断信号保持为无效状态。当 domaincfg.IE = 1 且中断传送已启用（idelivery = 1）时，只要寄存器 iforce 或 topi 不为零，中断信号就会被确认。

由于 APLIC 与 Hart 之间的通信可能会出现延迟，因此可能会发生外部中断捕获，但在实际读取 Hart 的 claimi 寄存器时，Hart 并没有挂起的中断，也没有启用中断。在这种情况下，claim 报告的中断标识将为零，从而导致 APLIC 发出明显的*假中断*。便携式软件必须为 APLIC 可能出现的假中断做好准备，这些假中断可以被安全地忽略，而且应该很少发生。为测试目的，可通过将 IDC 结构的 iforce 寄存器设置为 1 来触发 HART 的假中断。

仅通过 APLIC 处理外部中断的陷阱处理程序可大致编写如下：

保存处理器寄存器
i = 从 APLIC 的Hart IDC 结构中读取寄存器 claimi
i = i>>16
调用外部中断 i（次要标识）的中断处理程序 恢复处理器寄存器
复归

为了考虑虚假中断，本伪代码假定 "外部中断 0 "有一个中断处理程序，但它什么也不做。

4.9 MSI 的中断转发

在 MSI 发送模式（domaincfg.DM = 1）下，中断域通过 MSI 将中断转发给目标 Harts。

只有当特定信号源的相应挂起位和使能位均为 1，且寄存器 domaincfg 的 IE 字段也为 1 时，才会为该信号源发送 MSI。如果发送了 MSI，源的中断挂起位将被清零。

4.9.1 发送 MSI 的地址和数据

要通过 MSI 转发中断，APLIC 必须知道每个中断的 MSI 目标地址。对于任何给定系统，这些地址都是固定的，如果可能，应将其硬连接到 APLIC 中。不过，有些 APLIC 实现可能要求软件提供 MSI 目标地址。在这种情况下，根域寄存器 mmsiaddrcfg、mmsiaddrcfgh、smsiaddrcfg 和 smsiaddrcfgh（第 4.5.3 和 4.5.4 节）可用于配置所有中断域的 MSI 地址。另外，也可以通过本标准之外的自定义方法配置 MSI 地址。如果必须通过软件配置 MSI 目标地址，则只能在具有适当权限的执行模式下进行，通常只需一次，即在系统复位后的早期。

对于 machine-level 中断域，如果 MSI 目标地址由 mmsiaddrcfg 和 mmsiaddrcfgh 确定，则中断源 *i* 的外发 MSI 地址由这些寄存器和寄存器 target[*i*] 的 Hart 索引字段计算得出，计算公式如下：

g = (Hart Index>>LHXW) & (2^{HHXW} - 1)
h = Hart Index & (2^{LHXW} - 1)
MSI 地址 = Base PPN | (*g*<<(HHXS + 12))| (h<<LHXS) <<12

这里，<< *k* 和 >> *k* 表示向左和向右移动 *k* 位，双引号 (&) 表示按位逻辑 AND，竖杠 (|) 表示按位逻辑 OR。假设按照第 3.6 节中的建议在机器地址空间中安排 IMSIC 中断文件，则 *g* 值表示 Hart 组的编号（如果 HHXW = 0，则始终为0），而 *h* 值表示该组中目标 Hart 的编号。按照第 3.6 节的说法，HHXW = *j*，LHXW = *k*，HHXS = *E* - 24，LHXS = *C* - 12，Base PPN = *A*>>*12*。

对于 supervisor-level 域，如果 MSI 目标地址是由根域的配置寄存器（smsiaddrcfg 及其他）决定的，那么要为中断源 *i* 构建传出 MSI 的地址，必须首先将寄存器 target[*i*] 中的 Hart 索引转换为 machine-level 域用于相同 hart 的索引号（这些数字通常相同，但也可能不同）。(然后，利用 machine-level Hart 索引、来自 smsiaddrcfg 和 smsiaddrcfgh 的 Base PPN 和 LHXS 值、来自 mmsiaddrcfgh 的其他字段（HHXW、LHXW 和 HHXS）以及来自 target[*i*] 的 Guest 索引，计算出 MSI 的地址，如下所示：

g = (machine-level Hart Index >>LHXW) & (2^{HHXW} - 1)
h = machine-level Hart Index & (2^{LHXW} - 1)
MSI 地址 = Base PPN | (*g*<<(HHXS + 12))| (h<<LHXS) | guest Index <<12

用第 3.6 节中的术语表示，HHXW = *j*，LHXW = *k*，HHXS = *E* - 24，LHXS = *D* - 12，Base PPN = B>>*12*。

外发 MSI 写入的数据取自目标[*i*]的 EIID 字段，零扩展为 32 位。无论域的 domaincfg 寄存器的 BE 字段如何，MSI 的 32 位数据始终按小端顺序写入。

4.9.2 对电平敏感中断源的特殊考虑

一旦通过 MSI 转发了电平敏感型中断，APLIC 就会清除中断源的挂起位，然后忽略中断源，直到其传入信号被取消。在发送 MSI 时清除挂起位显然是必要的，这样可以避免 APLIC 为同一中断向目标机发送大量重复的 MSI。然而，在中断服务例程处理了为中断找到的原因后，尽管 APLIC 中的中断挂起位已被清除，而且在没有软件干预的情况下，传入的中断线仍可能因其他原因而在 APLIC 中保持挂起状态。如果中断服务例程随后退出而没有采取进一步行动，那么来自该中断源的持续中断可能永远不会受到关注。

为避免以这种方式丢弃中断，电平敏感中断的中断服务例程可在退出前执行以下操作之一：

第一种方法是通过读取 APLIC 中相应的 in_clrip 寄存器来测试进入 APLIC 的中断线是否仍 asserted。如果进入的中断仍 asserted，则可重复中断服务例程的主体，以便在再次测试源线之前找到并处理其他中断原因。一旦观察到输入线未 asserted，中断服务例程即可安全退出，因为任何新的中断 asserted 都将导致挂起位被置位，并向 Hart 发送新的 MSI。

第二种方法是，中断服务例程在退出前将 APLIC 的中断源标识号写入域的 setipnum 寄存器。如果中断源仍在 asserted 中断，这将导致中断的挂起位再次置 1，但如果中断源未 asserted 中断，则不会置 1。

4.9.3 实现 Hart 与 APLIC 之间的同步互动

当 APLIC 向 Hart 发送 MSI 时，在 Hart 的 IMSIC 观察到 MSI 之前会有一个未指定的传输延迟。因此，在通过写入 APLIC 寄存器改变 APLIC 的配置后，电车可能会在写入 APLIC 寄存器之前的一段时间内继续看到来自 APLIC 的 MSI。

有时，有必要知道何时不再有这些迟来的 MSI 到达。例如，如果要关闭（"关机"） hart 阵列，就必须将指向该 hart 阵列的所有中断重定向到其他 hart 阵列，这可能需要重新配置一个或多个 APLIC。即使重新配置了 APLIC，在确定没有更多的 MSI 目标地址之前，仍不能安全地关闭 Hart。

genmsi 寄存器（第 4.5.15 节）允许软件确定何时所有先前的 MSI 都已到达 Hart。要将 genmsi 用于此目的，软件可以在每个微调单元的 IMSIC 中断文件中指定一个外部中断标识，专门用于 APLIC 同步。假定有多个 hart，APLIC 的 genmsi 寄存器也应受到标准互斥锁的保护。然后，可以使用以下序列在 APLIC 和特定 hart 之间进行同步：

1. 在 hart 的 *IMSIC 中*，清除专门用于 APLIC 同步的特定次要中断标识的挂起位。
2. 获取 APLIC genmsi 寄存器的共享锁。
3. 写入 genmsi，向 hart 生成中断标识为 *i* 的 MSI。
4. 重复读取 genmsi，直到 Busy 位为假。
5. 为 genmsi 松开锁。
6. 在 hart 的 IMSIC 中重复读取小中断标识 *i 的*挂起位，直到发现该位被设置。

第 4 步和第 6 步的循环通常很快就会成功，往往在第一次或第二次尝试时就会成功。当这一步骤完成时，所有先前从 APLIC 发送的 MSI 也一定已经到达 hart 的 IMSIC。

第 5 章

5.Machine 和 Supervisor level 中断

RISC-V 特权体系结构在 0-15 范围内为中断定义了几个主要标识，包括 machine-level 和 supervisor-level 外部中断（编号 11 和 9）、 machine-level 和 supervisor-level 定时器中断（编号 7 和 5）以及 machine-level 和 supervisor-level 软件中断（编号 3 和 1）。除了这些主要标签外，APLIC 或 IMSIC 等*外部*中断控制器还为每个权限级别的外部中断提供次要标识，以区分来自不同设备或原因的中断。第 3 章和第 4 章介绍了外部中断的次要标识，具体说明了 IMSIC 和 APLIC 组件。

高级中断架构还为其他*本地中断*（通常用于报告错误）保留了 24 个主要中断标识，这些*中断*出现在 Hart 内部或与 Hart 非常接近的*地方*。此外还定义了一种机制，允许软件有选择地将本地中断和自定义中断委托给下一个较低权限级别，或在某些情况下将完全虚拟的中断注入到较低权限级别。

最后，软件还可以通过一项可选功能为主要中断（如定时器和软件中断以及任何本地中断）分配优先级，以便与 PLIC、APLIC 或 IMSIC 为外部中断设置的优先级相混合。

5.1 已定义的主要中断和默认优先级

表 5.1 列出了目前为符合高级中断架构（AIA）的 RISC-V 硬件定义的所有主要中断。除了 RISC-V 高级体系结构指定的主要中断外，AIA 还增加了中断号 35 和 43，作为低优先级和高优先级 *RAS 事件*的本地中断。

在特权体系结构控制的主要中断（编号 0-15）中，AIA 将计数器溢出中断（代码 13）归类为*本地中断*。此外，还假定将来对保留中断编号 14 和 15 的定义也将是本地中断。除两个 RAS 中断外，AIA 还在 16-23 范围内保留了主要中断号

默认的 优先级顺序	主要中断编号	说明
最高	43	本地中断：高优先级 RAS 事件
	11, 3, 7 9, 1, 5 12 10, 2, 6 13	Machine 中断：外部、软件、定时器 Supervisor 中断：外部、软件、定时器 Supervisor guest 外部中断 VS 中断：外部、软件、定时器 本地中断：计数器溢出
最低	35	本地中断：低优先级 RAS 事件

表 5.1： 按默认优先级顺序列出的标准主要中断代码。

主要中断编号	类别
0-12 13-15	非本地中断 本地中断 } 由特权架构分配
16-23 24-31 32-47 ≥ 48	本地中断 *指定用于定制用途* 本地中断 *指定用于定制用途*

表 5.2： 当前和未来主要中断的分类。

而 32-47 则是其他 RISC-V 扩展可能定义的标准本地中断。其余分配给特权体系结构的主要中断（0-12 号）被归类为非本地中断。总的来说，表 5.2 总结了 AIA 对所有主要中断标识的分类。

RAS 是*可靠性*、*可用性和可维护性*的缩写。通常情况下，RAS 事件与损坏数据的检测（如软或硬错误）和/或此类数据的使用相对应。例如，高优先级 RAS 事件本地中断可能是发生紧急未纠正错误的信号，需要 RAS 错误处理程序采取行动来控制错误，并在可能的情况下恢复错误。例如，低优先级 RAS 事件本地中断可由非紧急延迟或已纠正错误触发。

AIA 本身并不要求检测到的 RAS 事件必须触发为此定义的两个本地中断之一。系统可以自由地以其他方式报告任何或所有 RAS 事件，如通过 APLIC 或 IMSIC 路由的外部中断，或自定义中断。

报告特定 RAS 事件的方法很可能取决于系统中检测到该事件的位置。AIA 为 RAS 事件定义了本地中断号，这样系统在本地检测到此类事件时就有了标准的报告方法，而不必完全依赖外部中断或自定义中断。

一如既往，平台标准可能会进一步限制系统报告事件的方式，无论是 RAS 事件还是其他事件。

对于 RISC-V 高级体系结构未定义的标准本地中断（编号 16-23 和 32-47），目前的计划是按照本表所示顺序分配默认优先级：

最高	*47, 23, 46, 45, 22, 44, 43, 21, 42, 41, 20, 40*	
	11, 3, 7 *9, 1, 5* *12* *10, 2, 6* *13*	*Machine 中断：外部、软件、定时器* *Supervisor 中断：外部、软件、定时器* *Supervisor guest 外部中断* *VS 中断：外部、软件、定时器* *计数器溢出中断*
最低	*39, 19, 38, 37, 18, 36, 35, 17, 34, 33, 16, 32*	

在 16-23 号中断中，较高的中断编号表示较高的默认优先级，32-47 号中断也是如此。这两组中断按完整顺序交错排列，而特权架构的标准中断（0-15）则插入序列中间。这种建议的默认优先级顺序安排使 0-31 中断有可能单独成为 32 位 RISC-V 系统的一个适当子集。

实际上，未来的 RISC-V 扩展可能会、也可能不会在其定义的中断默认优先级顺序上坚持这一计划。

除了表 5.1 中现有的主要中断外，我们还初步提出了以下本地中断，这些中断按默认优先级递减的顺序排列：

23 总线或系统错误
45 每核高功率或过温事件
17 调试/跟踪中断

这些本地中断将由其他 RISC-V 扩展指定。请注意，此列表并非最终列表，可能会随着相关扩展的开发和批准而发生变化。

如果未来版本的 RISC-V 专用体系结构定义了中断 0，高级中断体系结构需要其默认优先级低于某些外部中断。请参见第 5.2.2 和 5.4.2 节中的 CSR mtopi *和* stopi。

24-31 号和 48 号及更高的中断号都被指定为自定义用途。如果 hart 实现了任何自定义中断，则必须记录这些中断在默认优先级顺序中的位置。

虽然许多标准寄存器（如 mip *和* mie*）只为 0-63 范围内的主要中断提供空间，但如果增加了自定义支持，则可以实现 64 及以上的自定义中断。CSR* mtopi*（第 5.2.2 节）和* stopi*（第 5.4.2 节）允许的主要中断编号可能高达 4095。*

当 hart 支持软件任意配置中断优先级时（将在后面的章节中介绍），当两个中断源被分配了相同的优先级时，默认优先级顺序仍然适用于打破并列关系。

5.2 Machine-level 中断

无论执行的是哪种标准的本地中断，CSR mip 和 mie 中的相应位都必须是可写的，而 mideleg 中的相应位（如果因为执行了 supervisor 模式而存在该 CSR）都必须是可写的或硬连线为 0。发生本地中断事件时，mip 中的中断挂起位会被置 1。然后，该位将保持设置状态，直至被软件清除。

根据 RISC-V 特权体系结构的规定，只要以下情况全部为真，中断就会跳转到 M-模式：(a) 要么当前特权模式为 M-模式，且 mstatus 的 MIE 位已启用 machine-level 中断，要么当前特权模式的特权小于 M-模式；(b) mip 和 mie 中的匹配位均为 1；(c) 如果存在 mideleg，则 mideleg 中的相应位为 0。

当多个中断原因准备同时触发时，首先触发的中断由优先级顺序决定，优先级顺序可以是上一节（5.1）中指定的默认顺序，也可以是软件配置的修改顺序。

5.2.1 在 machine 层面配置主要中断的优先级

主要中断 0-63 的 machine-level 优先级可通过一组寄存器配置，这些寄存器可通过第 2 章介绍的 miselect 和 mireg CSR 访问。当 XLEN = 32 时，这些寄存器中有 16 个已定义，下面列出了它们的 miselect 地址：

0x30 iprio0
0x31 iprio1
. . .. ..
0x3F iprio15

每个寄存器控制四个中断的优先级，每个中断一个 8 位字节。对于范围在 0-15 之间的数字 *k*，寄存器 *iprio*k 控制 *k* × 4 至 *k* × 4 + 3 的中断优先级，格式如下：

位 7:0　　中断 *k* × 4 的优先级编号
位 15:8　　*中断 k* × 4 + 1 的优先级编号
位 23:16　　*中断 k* × 4 + 2 的优先级编号
位 31:24　　中断 *k* × 4 + 3 的优先级编号

当 XLEN = 64 时，只存在偶数寄存器：

0x30 iprio0
0x32 iprio2
. . .. ..
0x3E iprio14

每个寄存器控制八个中断的优先级。 对于 0-14 范围内的偶数 *k*，寄存器 *iprio*k 控制 *k* × 4 至 *k* × 4 + 7 中断的优先级，格式如下：

位 7:0　　中断 *k* × 4 的优先级编号
位 15:8　　*中断 k* × 4 + 1 的优先级编
位 23:16　　*中断 k* × 4 + 2 的优先级编号
位 31:24　　中断 *k* × 4 + 3 的优先级编号
位 39:32　　*中断 k* × 4 + 4 的优先级编号
位 47:40　　*中断 k* × 4 + 5 的优先级编号
位 55:48　　*中断 k* × 4 + 6 的优先级编号
位 63:56　　中断 *k* × 4 + 7 的优先级编号

当 XLEN = 64 且 miselect 为 0x31-0x3F 范围内的奇数值时，尝试访问 mireg 引发非法指令异常。

有效寄存器 iprio0-iprio15 统称为 machine-level 的 *iprio 数组*。

外部中断的优先级数宽度为 *IPRIOLEN*。该参数受 hart 的主外部中断控制器（PLIC、APLIC 或 IMSIC）影响。

对于 APLIC，IPRIOLEN 的范围为 1-8，具体请参见 APLIC 第 4 章。

对于 IMSIC，IPRIOLEN 为 6、7 或 8。只有当 IMSIC 执行的外部中断标识数为 63 时，IPRIOLEN 才可以为 6。只有当 IMSIC 实现的外部中断标识数不超过 127 时，IPRIOLEN 才可以为 7。对于任何 IMSIC，IPRIOLEN 都可以为 8，而与外部中断标识的数量无关。

有效 *ipriok* 寄存器的每个字节要么是只读零，要么是 **WARL** 无符号整数字段，正好实现 IPRIOLEN 位。对于给定的中断编号，如果 mie 中的相应位为只读零，则 iprio 数组中的中断优先级编号也必须为只读零。Machine-level 外部中断的优先级编号（寄存器 iprio2 的第 31:24 位）也必须为只读零。除这两项限制外，实现者可自由选择哪些优先级编号字段可设置，哪些为只读零。如果 iprio 数组中的所有字节都是只读零，则只能为外部中断配置优先级，而不能为任何其他中断配置优先级。

平台标准可能要求针对某些中断原因配置优先级。

通过 miselect 和 mireg 访问的 iprio 数组仅在中断陷入到 M 模式时才会影响中断的优先级。当中断在数组中的优先级编号为零（只读为零或设置为零）时，其优先级采用第 5.1 节中的默认顺序。对于默认优先级高于 machine 外部中断的主要中断，将其优先级设置为非零值会*降低*其优先级。对于默认优先级低于 machine 外部中断的主要中断，将其优先级编号设置为非零值可*提高*其优先级。当两个中断原因的标称优先级相同时，将根据默认优先级顺序打破平局。表 5.3 总结了优先级编号对中断优先级的影响。

当 hart 的 IMSIC 支持 255 个以上的外部中断次要标识时，可为其他中断配置的非默认优先级只能是与外部中断标识 1-255 相对应的优先级，而不是标识 256 或更高的优先级。

	默认优先级高于 machine 外部中断的中断	Machine 外部中断	默认优先级低于 machine 外部中断的中断
优先级顺序	Machine-level iprio 数组中的优先级编号	来自中断控制器（APLIC 或 IMSIC）的优先级编号	Machine-level iprio 数组中的优先级编号
最高	0		
	1 2 - - - 254 255	1 2 - - - 254 255	1 2 - - - 254 255
		256 及以上 (仅限 IMSIC）	
最低			0

表 5.3：Machine-level iprio 数组对 M 模式中断优先级的影响。对于优先级相同的中断，以第 5.1 节的默认顺序为准。

要实现本节所述的优先级可配置性，RISC-V hart 的外部中断控制器不仅要向 hart 发送挂起并已启用的外部中断信息，还要发送中断的优先级编号。通常情况下，这意味着向 Hart 发送外部中断信号的连接宽度不仅是通常的单线，而是现在的 IPRIOLEN + 1 线。

预计许多系统将放弃主要中断的优先级配置，而只是让 iprio *数组全部为只读 0。需要这种优先级可配置性的系统可以尝试将每个中断控制卡的外部中断控制器安排在离中断控制卡相对较近的位置，例如，将系统限制在最多几个与 APLIC 相连的小型内核上，或者让每个中断控制卡都有自己的 IMSIC。*

如果受支持，将 supervisor 级外部中断（iprio2 的第 15:8 位）的优先级设置为非零值 *p*，则整个 supervisor 级外部中断的优先级与优先级为 *p* 的 machine 外部中断的优先级相同。

(由于在实现 hypervisor extension 时，需要将 supervisor guest 外部中断和 VS 级外部中断委托给 supervisor-level，因此这些中断的 machine-level 优先级编号始终会被忽略，并且应为只读 0）。

如果系统具有原始 PLIC，以便与旧版软件向后兼容，则重置时应将 machine-level iprio 数组初始化为全零。

5.2.2 Machine top 中断 CSR (mtopi)

Machine-level CSR mtopi 为只读，宽度为 MXLEN。读取 mtopi 会以这种格式返回 machine-level 最高优先级的挂起且启用中断的信息：

位 27:16　IID
位 7:0　IPRIO

mtopi 的所有其他位均为保留位，读作 0。

mtopi 的值为零，除非在 mip 中有未授权给较低权限级别的挂起中断，且在 mie 中已启用。当 machine-level 有一个挂起并启用的主要中断时，字段 IID（中断标识）是优先级最高的中断的主要标识号，字段 IPRIO 表示其优先级。

如果 machine-level iprio 数组的所有字节都是只读 0，则可以简化字段 IPRIO 的实现，只要 mtopi 不为 0，其值就始终为 1。

否则，当 mtopi 不为零时，如果报告中断的优先级编号在 1 至 255 之间，IPRIO 就是该编号。如果中断的优先级编号为 0 或大于 255，IPRIO 将按如下方式设置为 0 或 255：

- 如果中断的优先级大于 255，则 IPRIO 为 255（最低优先级）。
- 如果中断的优先级编号为 0，且中断编号 IID 的默认优先级高于 machine 外部中断，则 IPRIO 为 0（最高优先级）。
- 如果中断的优先级编号为 0，且中断编号 IID 的默认优先级低于 machine 外部中断，则 IPRIO 为 255（最低优先级）。

为确保当中断挂起并启用 machine-level 时 mtopi *永远不为零，如果主要中断 0 可以捕获到 M 模式，则其默认优先级必须低于 machine 外部中断。*

mtopi 的值不受 CSR mstatus 中全局中断使能 MIE 的影响。

RISC-V 特权架构确保，当 mtopi 的值不为零时，如果当前特权模式为 M 且 mstatus.MIE 为 1，或者当前特权模式的特权小于 M-模式，则会对字段 IID 所指示的中断采取陷阱 M-模式。陷阱本身不会导致 mtopi 的值发生变化。

下面的伪代码展示了 machine-level 陷阱处理程序如何读取 mtopi，以避免在处理另一个陷阱（同步异常或先前的中断）期间发生中断时，重复恢复和保存处理器寄存器：

```
保存处理器寄存器
i = read CSR mcause
如果 (i >= 0) {
    处理同步异常 i
    必要时恢复 mstatus
}
如果 (mstatus.MPIE == 1) {
    loop {
        i = read CSR mtopi
        if (i == 0) exit loop
        i = i>>16
        调用主要中断 i 的中断处理程序
    }
}
恢复处理器寄存器
从陷阱返回
```

(这个例子可以进一步优化，但复杂程度会增加）。

5.3 用于 supervisor-level 的中断过滤和虚拟中断

在实现了 supervisor 模式时，高级中断架构利用新的 CSR mvien（machine 虚拟中断使能）和 mvip（machine 虚拟中断挂起位），增加了软件过滤中断和虚拟中断的功能。*中断过滤功能*允许 supervisor-level 中断（SEI 或 SSI）或本地中断或自定义中断捕获到 M 模式，然后由软件有选择地委托给 supervisor-level 中断，即使 mideleg 中的相应位仍为零。在适当的情况下，同样的硬件还允许 machine-level 向 supervisor-level 发出与任何实际中断事件无关的*虚拟中断*。

与 CSR mip、mie 和 mideleg 一样，寄存器 mvien 和 mvip 的每一位都与 0-63 范围内的中断编号相对应。当 mideleg 中的某位为 0，而 mvien 中的匹配位为 1 时，sip 中的相同位就是 mvip 中相应位的别名。当 mideleg 和 mvien 中的相应位都为零时，sip 中的某位只读为零。表 5.4 总结了 mideleg 和 mvien 对 sip 和 sie 的综合影响。

mideleg[*n*]	mvien[*n*]	sip[*n*]	sie[*n*]
0	0	只读 0	只读 0
0	1	mvip[*n*] 的别名	可写
1	-	mip[*n*] 的别名	mie[*n*] 的别名

表 5.4：mideleg 和 mvien 对 sip 和 sie 的影响（hypervisor extension 的 VS 级中断除外，它们出现在 hip 和 hie 中，而不是 sip 和 sie 中）。只有主要中断 1、9 和 13-63 才能将 mvien 中的位设置为 1。对于中断 0-12，sip 中的别名可能是只读的，如 RISC-V 特权架构所规定。

CSR mvien *的名称不是 "*mvie*"，因为该寄存器的功能更类似于* mcounteren*，而不是* mie*。*
mvien *的位控制寄存器* mvip *中的虚拟中断挂起位是否处于活动状态，并在 supervisor-level 可见。这与通常的中断使能位（如* mie *中的中断使能位）屏蔽挂起中断的方式不同。*

当且仅当 mideleg 或 mvien 中的相应位被设置时，sie 中的位才是可写的。当中断由 mideleg 委托时，sie 中的可写位是 mie 中相应位的别名，否则就是独立的可写位。通常，sie 中不可写的位必须是只读 0。

如果 mideleg 中的某位为 0，而 mvien 中的相应位从 0 变为 1，那么 sie 中的匹配位的值就会变得不明确。同样，如果 mvien 中的某位为 1，而 mideleg 中的相应位从 1 变为 0，则 sie 中的匹配位的值再次变为未指定。

对于中断号 13-63，实现者可自由选择 mvien 的哪些位可写，哪些位只读为 0 或 1。如果 mvien 中的某个位只读为 0（防止启用虚拟中断），则 mvip 中的相同位也应只读为 0。中断 13-63 的所有其他位在 mvip 中必须是可写的。

平台标准或其他扩展可能会要求某些中断原因的 mvien *位可写，或只读为 0 或 1。*

mvien 中用于Supervisor 软件中断（代码 1）和Supervisor 外部中断（代码 9）的位均为可写或只读 0，不能为只读 1。中断 0-12 的 mvien 其他位为保留位，必须为只读 0。

强烈建议 mvien 的第 9 位可以写入。此外，如果 mip 的第 1 位（SSIP）可以由中断控制器自动设置，而不仅仅是明确写入 mip 或 sip，则强烈建议 mvien 的第 1 位也是可写的。

当 mvien 的第 1 位为 0 时，mvip 的第 1 位是 mip 相同位（SSIP）的别名。但当 mvien 的第 1 位为 1 时，mvip 的第 1 位是独立于 mip.SSIP 的可写位。当 mvien 第 1 位的值从 0 变为 1 时，mvip 第 1 位的值将变得不明确。

当 mip 中的 STIP 位可写入时，mvip 的第 5 位是 mip 中相同位（STIP）的别名。当 STIP 在 mip 中不可写时（如 menvcfg.STCE = 1），mvip 的第 5 位为只读 0。

当 mvien 的第 9 位为 0 时，mvip 的第 9 位是软件可写的 mip 第 9 位（SEIP）的别名。但当 mvien 的第 9 位为 1 时，mvip 的第 9 位是独立于 mip.SEIP 的可写位。与位 1 不同，改变 mvien 第 9 位的值不会影响 mvip 第 9 位的值。

当 mvien *的第 9 位为零时，* mvip *的第 9 位会使* mip *的软件可写 SEIP 位直接被自己访问。*

除上述第 1、5 和 9 位外，mvip 在 12:0 范围内的位数为保留位，必须为只读零。

mvien 第 9 位的值对Supervisor 外部中断有一些额外影响：

- 当 mvien 的第 9 位为 0 时，软件可写 SEIP 位（mvip 的第 9 位）将按照 RISC-V 特权架构规定的方式与 mip 的读写进行交互。特别是，在大多数情况下，mvip 第 9 位的值与 mip.SEIP 的可读值进行逻辑 OR。但当 mvien 的第 9 位为 1 时，mip 中的 SEIP 位是只读的，不包括 mvip 第 9 位的值。相反，mip.SEIP 的值只是来自 hart 外部中断控制器（APLIC 或 IMSIC）的上位外部中断信号。
- 如果 hart 具有 IMSIC，那么当 mvien 的第 9 位为 1 时，从 S 模式明确访问 supervisor-level 中断文件的尝试将引发非法指令异常。访问 CSR stopei 或访问 sireg（当 siselect 的值在 0x70-0xFF 范围内）的尝试都会引发异常。访问 guest 中断文件（通过 vstopei 或 vsiselect + vsireg）不会受到影响。

在实现 hypervisor extension 时，如果 mideleg 和 mvien 中同一位置的某位为 0，则 hideleg 中的该位为只读 0（此外，sip、sie、hip 和 hie 中的该位也为只读 0）。但是，如果中断 13-63 的某个位在 mideleg 或 mvien 中都是 1，那么 hideleg 中的相同位可能是可写的，也可能是只读为

零的，具体取决于实现方式。hideleg 中的任何位都不是只读 1。RISC-V 特权架构进一步限制了 hideleg 的第 12:0 位。

在执行监督模式时，mvien 和 mvip 的最低执行要求是所有位都为只读零，除了 mvip 的第 1 位和第 9 位，有时还有第 5 位，每个位都是 mip 中现有可写位的别名。(不过，如前所述，强烈建议 mvien 的第 9 位也是可写的）。不执行监督模式时，寄存器 mvien 和 mvip 不存在。

5.4 Supervisor-level 中断

如果 sip 中的标准本地中断成为挂起（= 1），则 sip 中的位可写，并将保持设置状态，直到软件清除为止。

与 machine-level 中断一样，supervisor-level 中断的捕获与 RISC-V 特权架构规定的基本相同。只要以下情况全部为真，中断就会进入 S-模式（或 HS-模式）：(a) 要么当前权限模式为 S-模式，且 sstatus 的 SIE 位启用了 supervisor-level 中断，要么当前权限模式的权限小于 S-模式；(b) sip 和 sie 中的匹配位均为 1，或者，如果实现了 hypervisor extension，则 hip 和 hie 中的匹配位均为 1；(c) 如果实现了 hypervisor extension，则 hideleg 中的相应位为 0。

5.4.1 在 supervisor-level 配置主要中断的优先级

通过 siselect 和 sireg 访问的 supervisor-level *ipriok* 寄存器阵列中，可对主要中断 0-63 的 supervisor-level 优先级进行配置。当 XLEN = 32 或 64 时，该数组的结构与 machine-level iprio 数组相同。总之，当 XLEN = 32 时，有 16 个 32 位寄存器具有这些 siselect 地址：

0x30 iprio0
0x31 iprio1
. . .. ..
0x3F iprio15

每个寄存器控制四个中断的优先级，每个中断一个 8 位字节。当 XLEN = 64，则只存在偶数寄存器：

0x30 iprio0
0x32 iprio2
. . .. ..
0x3E iprio14

每个寄存器控制八个中断的优先级。如果 XLEN = 64 且 siselect 为 0x31-0x3F 范围内的奇数值，则尝试访问 sireg 会引发非法指令异常。

有效寄存器 iprio0-iprio15 统称为 supervisor-level *的 iprio 数组*。有效*寄存器 ipriok* 的每个字节都是只读 0 或 **WARL** 无符号整数字段，精确执行 IPRIOLEN 位。

对于给定的中断编号，如果 sie 中的相应位只读为 0，则Supervisor 级 iprio 数组中的中断优先级编号也必须只读为 0。 supervisor-level 外部中断的优先级编号（iprio2 的第 15:8 位）也必须为只读零。除这两项限制外，实现者可自由选择哪些优先级编号字段可设置，哪些为只读 0。

与往常一样，平台标准可能要求针对某些中断原因配置优先级。

---

*预计许多高端系统将不支持本节所述的在 supervisor-level 配置主要中断优先级的功能。尤其是 Linux 系统，即使提供了此类功能，也无法利用。*iprio *数组必须可以访问，但可以是只读的 0。*

通过 siselect 和 sireg 访问的 supervisor-level iprio 数组仅在中断陷波到 S 模式时才会影响中断的

优先级。当数组中的中断优先级编号为零（只读为零或设置为零）时，其优先级采用第 5.1 节中的默认顺序。将中断的优先级编号设置为非零值 *p*，则该中断的优先级与优先级编号为 *p* 的 Supervisor 级外部中断的优先级相同。对于默认优先级低于上级外部中断的中断，将其优先级编号设置为非零值可提高其优先级。当两个中断原因的标称优先级相同时，将根据默认优先级顺序打破平局。表 5.5 总结了优先级编号对中断优先级的影响。

如果支持将 VS 级外部中断的优先级（iprio2 的第 23:16 位）设为非零值 *p*，则当 VS 外部中断陷到 S 模式时，整个 VS 外部中断类别的优先级名义上与优先级为 *p* 的Supervisor 外部中断相同。

	默认优先级高于 Supervisor 外部中断的中断	外部 Supervisor 中断	默认优先级低于 Supervisor 外部中断的中断
优先顺序	优先序号 supervisor-level iprio阵列	优先序号从中断控制器（APLIC 或 IMSIC）	优先序号 supervisor-level iprio阵列
最高	0		
	1 2 - - - 254 255	1 2 - - - 254 255	1 2 - - - 254 255
		256 及以上 (仅限 IMSIC）	
最低			0

表 5.5： supervisor-level iprio 数组对 S 模式中断优先级的影响。对于优先级相同的中断，以第 5.1 节的默认顺序为准。

如果系统具有原始 PLIC，以便与旧版软件向后兼容，则重置时应将Supervisor 级 iprio 阵列初始化为全零。

5.4.2 Supervisor 最高中断 CSR (stopi)

Supervisor 级 CSR stopi 为只读，宽度为 SXLEN。读取 stopi 后，将以这种格式返回最高优先级的挂起并已启用的 supervisor-level 中断信息：

位数 27:16 IID
位 7:0 IPRIO

stopi 的所有其他位均为保留位，读作 0。

stopi 的值为零，除非：(a) 有一个中断在 sip 中等 pending，并在 sie 中启用，或者，如果实现了 hypervisor extension，有一个中断在 hip 中等 pending，并在 hie 中启用；以及(b) 有一个中断在 sip 中等 pending，并在 sie 中启用。
(b) 中断没有委托给较低的权限级别（通过隐藏脚本，如果 hypervisor extension 了实现）。当有一个挂起并启用的 supervisor-level 主要中断时，字段 IID 是优先级最高的中断的主要标识号，字段 IPRIO 表示其优先级。

如果Supervisor 级 iprio 数组的所有字节都是只读 0，则可以简化字段 IPRIO 的实现，即只要 stopi 不为 0，其值就始终为 1。

否则，当 stopi 不为零时，如果报告中断的优先级编号在 1 至 255 之间，IPRIO 就是该编号。如果中断的优先级编号为 0 或大于 255，IPRIO 将按如下方式设置为 0 或 255：

- 如果中断的优先级大于 255，则 IPRIO 为 255（最低优先级）。
- 如果中断的优先级编号为 0，且中断编号 IID 的默认优先级高于Supervisor 外部中断，则 IPRIO 为 0（最高优先级）。
- 如果中断的优先级编号为 0，且中断编号 IID 的默认优先级低于Supervisor 外部中断，则 IPRIO 为 255（最低优先级）。

---

*为确保当中断挂起并启用了 supervisor-level 时，*stopi *永远不为零，如果主要中断 0 可以捕获到 S 模式，则其默认优先级必须低于 supervisor-level 外部中断。*

stopi 的值不受 CSR sstatus 中全局中断使能 SIE 的影响。

RISC-V 特权架构确保，当 stopi 的值不为零时，如果当前特权模式为 S 且 sstatus.SIE 为 1，或者当前特权模式的特权小于 S-模式，则会对字段 IID 所指示的中断执行 S-模式陷阱。陷阱本身不会导致 stopi 的值发生变化。

下面的伪代码展示了当处理另一个陷阱（同步异常或先前的中断）期间发生中断时，Supervisor 级陷阱处理程序如何读取 stopi，以避免重复恢复和保存处理器寄存器：

```
保存处理器寄存器
i = read CSR scause if
(i >= 0) {
    处理同步异常 i
    必要时恢复 sstatus
}
如果 (sstatus.SPIE == 1) {
    loop {
        i = read CSR stopi
        if (i == 0) exit loop
        i = i>>16
        调用主要中断 i 的中断处理程序
    }
}
恢复处理器寄存器
从陷阱返回
```

(这个例子可以进一步优化，但复杂程度会增加）。

5.5 WFI（等待中断）指令

RISC-V 高级体系结构规定，指令 WFI（等待中断）可暂停在中断点执行，直到该中断点等待中断。高级中断架构（AIA）重新定义了 WFI 之后必须恢复执行的时间。

根据 RISC-V 特权架构，只要任何中断在 CSR mip 和 mie 中处于挂起和启用状态，就必须从 WFI 恢复指令执行，而不考虑 mideleg 所指示的任何委托。使用 AIA 后，由于 AIA 增加了虚拟中断机制，这一简明规则不再适用。取而代之的是，只要有任何权限级别的中断挂起（无论中断权限级别高于或低于 hart 的当前权限模式），就必须从 WFI 恢复执行。

如果寄存器 mtopi 不为零，则在 machine-level 挂起中断。如果实现了 S 模式，寄存器 stopi 不为零，则在 supervisor level 挂起中断。如果实现了 hypervisor extension，则如果 vstopi（第 6.3.3 节）不为零，则在 VS 层挂起中断。

当出现以下情况时，AIA 的 WFI 规则与特权架构的规则具有相同的行为，当 mvien = *0，如果实现了 hypervisor extension，还需要* hvien = *0 和* hvictl.*VTI = 0，从而禁用* mip *中不可见的所有虚拟中断。(AIA 的 hypervisor 寄存器将在下一章"虚拟机中断（VS 级）"中介绍）。*

第 6 章

6.虚拟机中断（VS 级）

实现 hypervisor extension 后，hart 可能的权限模式集包括用于托管虚拟 hart 的虚拟*supervisor*（VS）和*虚拟 user*（VU）模式。高级中断架构为 hypervisor extension 增加了新的中断设施，与前面描述的 hypervisor 级中断设施保持一致。

如第 2 章所述，新增了多个 hypervisor 和 VS CSR：hvien、hvictl、hviprio1、hviprio2、vsiselect、vsireg、vstopei 和 vstopi。(对于 RV32，还增加了以下 high-half CSR：hidelegh、hvienh、hviph、hviprio1h、hviprio2h、vsiph 和 vsieh）。一如既往，在 VS 模式或 VU 模式下执行时，VS CSR 将替代相应的 supervisor CSR。

为了让在虚拟机中运行的软件看起来就像在真实机器上执行一样，该机器在 supervisor-level 实现了高级中断体系结构，hypervisor 软件和本章所述的硬件设施共同承担责任。有些中断可以直接由硬件处理，有些则需要 hypervisor 进行大量仿真，有时还需要硬件协助。

6.1 使用 guest 中断文件的 VS 级外部中断

当一个 hart 实现了 hypervisor extension 时，建议该 hart 也有一个带有 guest 中断文件的 IMSIC。如果有 guest 中断文件，每个文件都可以分配给物理 hart 的一个虚拟 hart，作为该虚拟 hart 的 supervisor-level 中断文件。如果有 *N* 个 guest 中断文件，那么该物理 hart 上的 *N* 个虚拟 hart 都可以拥有一个物理 guest 中断文件，作为其（虚拟）supervisor-level 中断文件。CSR hstatus 的 VGEIN 字段总是指示当前虚拟 hart 的 guest 中断文件。如果 VGEIN 不是 guest 中断文件的有效编号，则当前虚拟分组没有 guest 中断文件作为其 supervisor-level 中断文件。

当 hstatus.VGEIN 为 guest 中断文件的有效编号时，范围为 0x70-0xFF 的 vsiselect 值将选择 guest 中断文件的寄存器，就像同一范围内的 siselect 值选择 IMSIC 真正的 supervisor-level 中断文件的寄存器一样。通过 vsiselect 和 vsireg 间接访问的中断文件寄存器，以及 IMSIC 专用 CSR vstopei，将在有关 IMSIC 的第 3 章中详细介绍。由于所有 IMSIC 中断文件的作用完全相同，因此虚拟 Hart 通过 CSR siselect、sireg 和 stopei 访问的 guest 中断文件与 S 模式（或 HS 模式）下的真正 supervisor-level 中断文件没有区别。

除了每个 hart上的 IMSIC 外，虚拟机可能还需要看到 PLIC 或 APLIC。然而，与 IMSIC 为虚拟机提供物理 guest 中断文件的功能不同，PLIC 或 APLIC 必须由 hypervisor 为虚拟机进行模拟。

高级中断体系结构目前不包括虚拟 APLIC 的硬件支持。对于数量较少的 Harts，此类硬件将比 IMSIC 实现 guest 中断文件所需的硬件大得多。我们假设，大多数高性能 I/O 可通过可直接向 guest 中断文件发送 MSI 的设备（如通过 PCI Express 互连的设备）完成。对于中断必须通过（虚拟）APLIC 的设备类型，仿真 APLIC 的开销成本预计不会太大。

当虚拟主机因分配了 guest 中断文件而被视为拥有 IMSIC 时，所有针对虚拟主机的外部中断（无论是真实中断还是仿真中断）都必须通过这个感知到的 IMSIC。Hypervisor 可以通过设置通过 vsiselect 和 vsireg 间接访问的中断挂起数组中的一个位，轻松地将仿真外部中断注入 hstatus.VGEIN 选择的 guest 中断文件。当虚拟机有 guest 中断文件时，hypervisor 通常不会设置 CSR hvip 中的 VSEIP 位。

在虚拟机的仿真 APLIC 具有与真实 APLIC 的实际中断源等价的有线中断源的特殊情况下，如果在该虚拟机中运行的软件将其虚拟 APLIC 配置为将来自该中断源的中断作为 MSI 转发到特定虚拟 Hart，则 hypervisor 可将真实 APLIC 配置为将实际中断作为 MSI 直接转发到虚拟 Hart 的 guest 中断文件。这样，尽管 hypervisor 必须捕获和仿真虚拟机的内存访问，以配置虚拟 APLIC 的中断转发，但中断本身可自动转换为 guest 中断文件的真正 MSI，而无需为每个到达的中断调用 hypervisor。

6.1.1 Guest 操作系统直接控制设备

为确保对中断的正确支持，在 hypervisor 允许虚拟机中运行的 guest 操作系统直接控制发送 MSI 的物理设备之前，必须满足两个条件：首先，每个虚拟机都必须有一个分配给它的 guest 中断文件，使每个虚拟机都有自己明显的 IMSIC。其次，来自设备的中断必须通过 APLIC 发出信号，该 APLIC 可以将这些中断转换为 MSI，或者系统必须有一个 IOMMU，该 IOMMU 可以转换设备本身写入 MSI 内存的地址。

如果 guest 操作系统直接控制一个能够发送 MSI 的设备，它自然会在该设备上使用操作系统为其虚拟 Harts 的 IMSIC 所看到的 guest 物理地址配置 MSI，而不知道这些地址只是虚拟的。 当设备为 MSI 执行内存写入操作时，IOMMU 必须使用 hypervisor 提供的转换表，将写入操作的目标地址从 guest OS 分配的 guest 物理地址转换为目标 guest 中断文件的真实物理地址。

从设计上讲，IOMMU 必须对设备 MSI 进行的地址转换，与 IOMMU 必须对同一设备的其他内存访问进行的地址转换（将 guest 物理地址转换为真正的物理地址）没有本质区别。由于每个虚拟 Hart 都分配了一个专用的物理 guest 中断文件，该文件与真正的 supervisor-level 中断文件没有区别，因此 MSI 写入的数据无需翻译，它指定了目标中断文件中的中断标识号。

6.1.2 将虚拟主机迁移到不同的 guest 中断文件

当需要将虚拟 hart 从一个物理 hart 移动到另一个物理 hart 时，如果虚拟 hart 使用 guest 中断文件，分配给它的特定 guest 中断文件必须从旧物理 hart 使用的文件更改为新物理 hart使用的不同文件。由于每个 guest 中断文件都与单个物理 hart 物理绑定，因此虚拟 hart 在移动时不能将其 guest 中断文件带走。

将虚拟 hart 从一个 guest 中断文件迁移到另一个 guest 中断文件的过程，要比迁移虚拟 hart 持有的其他大多数状态复杂得多。在新物理 hart 选择了目标 guest 中断文件后，建议采取以下步骤：

1. 在旧的中断文件中，将寄存器 eidelivery 和 eithreshold 的值保存到内存中，并设置 eidelivery = 0。
2. 在新的中断文件中，设置 eidelivery = 0，并将所有已执行的中断挂起位（eip 数组）清零。
3. 修改所有 IOMMU 的相关转换表，以便将该虚拟中断文件的 MSI 发送到新的物理中断文件。同样，如果 APLIC 中的任何中断被 MSI 转发到旧的中断文件，则重新配置 APLIC，将其发送到新的中断文件。根据需要，与所有 IOMMU 和 APLIC 同步，以确保在此步骤后不会有滞留的 MSI 到达旧中断文件。使用第 4.9.3 节中的算法可实现与 APLIC 的同步。
4. 在旧中断文件中，将所有已执行的中断挂起位和中断使能位（eip 和 eie 数组）转储到内存中。完成此步骤后，旧中断文件将不再使用。
5. 在新的中断文件中，将步骤 4 中保存的中断挂起位逻辑 OR 到新的中断文件中，使用指令 CSRS 写入 eip 数组。同时，将步骤 4 中保存的中断使能位加载到 eie 数组中。
6. 在新的中断文件中，将寄存器 eithreshold 和 eidelivery 装入步骤 1 中保存的值。

在整个中断文件完全迁移之前，不建议在新的物理中断中恢复虚拟中断的执行。

在中断文件完全迁移之前恢复虚拟机的执行，可能会让虚拟机中运行的软件看到来自单个设备的多个 MSI，而这些 MSI 的顺序是不应该发生的。虽然这种情况在实际操作中很少发生，但却有可能使依赖于有效事件顺序的设备驱动程序（也许是无意中）陷入困境。

6.2 无需 guest 中断文件的 VS 级外部中断

虽然建议实现 hypervisor extension 的虚拟机也要有带有 guest 中断文件的 IMSIC，但这并不是必须的。即使假设存在 guest 中断文件，也可能会出现这样的情况，即物理 Hart 上的虚拟 Hart 多于 guest 中断文件，导致某些虚拟 Hart 没有 guest 中断文件。无论哪种情况，hypervisor 都必须为虚拟 hart 模拟外部中断控制器，而不能利用分配给虚拟 hart 的 guest 中断文件。

在为虚拟主机模拟外部中断控制器时，如果虚拟主机除外部中断外不支持可配置的中断优先级，则只需设置 hvip 中的位 VSEIP，即可将外部中断 asserted 为 VS 级，如 RISC-V 特权架构所定义。不过，要同时模拟外部中断控制器和非外部中断的优先级可配置性，hypervisor 必须使用 CSR hvictl（hypervisor 虚拟中断控制），下一节将详细介绍。

6.3 VS 级中断

6.3.1 在 VS 级配置主要中断的优先级

与 supervisor-level 中断一样，高级中断架构允许通过软件配置主要 VS 级中断，使其优先级与 VS 级外部中断相混合。如第 5.4 节所述，通过 CSR siselect 和 sireg 间接访问的 iprio 阵列可配

置 supervisor-level 中断的优先级。iprio 阵列寄存器的 siselect 地址为 0x30-0x3F。

VS 级有自己的 vsiselect 和 vsireg，但与 supervisor-level 不同的是，vsiselect 地址 0x30-0x3F 处没有寄存器。当 vsiselect 的值在 0x30-0x3F 范围内时，VS 模式访问 sireg（实际上是 vsireg）的尝试会导致虚拟指令异常。为了让虚拟机产生通过 siselect 和 sireg 访问 iprio 寄存器数组的错觉，当 VS 模式访问 sireg 引起虚拟指令陷阱时，hypervisor 必须模拟 VS 级 iprio 数组。

使用 hypervisor CSR hviprio1 和 hviprio2，为 VS 级计算中断子集优先级提供了单独的硬件机制，而不是物理 VS 级 iprio 阵列。可通过硬件配置优先级的主要中断子集如下：

1	监控软件中断
5	监控计时器中断
13	计数器溢出中断
14-23	*为标准本地中断保留*

对于定向到 VS 层的中断，硬件不支持软件可配置优先级，不支持 32-48 范围内的标准本地中断。

对于自定义中断，可通过自定义 CSR 在硬件中支持优先级可配置性，扩展标准中断的 hviprio1 *和* hviprio2。

寄存器 hviprio1 和 hviprio2 具有这些格式：

hviprio1：

位 7:0	*为中断 0 的优先级号保留；读作 0*
位 15:8	中断 1 的优先级编号，Supervisor 软件中断
位 23:16	*为中断 4 保留的优先级号；读作 0*
位 31:24	中断 5 的优先级编号，Supervisor 定时器中断
位 39:32	*为中断 8 的优先级号保留；读作 0*
位 47:40	中断 13（计数器溢出中断）的优先级编号
位 55:48	中断 14 的优先级编号
位 63:56	中断 15 的优先级编号

hviprio2：

位 7:0	中断 16 的优先级编号
位 15:8	中断 17 的优先级编号
位 23:16	中断 18 的优先级编号
位 31:24	中断 19 的优先级编号
位 39:32	中断 20 的优先级编号
位 47:40	中断 21 的优先级编号
位 55:48	中断 22 的优先级编号
位 63:56	中断 23 的优先级编号

hviprio1 和 hviprio2 中的每个优先级编号都是一个 **WARL** 无符号整数字段，要么只读为零，要么至少实现 IPRIOLEN 位或 6 位（以较大者为准），最好是全部 8 位。实现者可自由选择哪些优先级数字字段为只读零，但所有其他字段必须实现相同的整数位数。这些 CSR 的最低实现要求是它们都是只读零。

虚拟机 hypervisor 可以选择使用寄存器 hviprio1 和 hviprio2 来模拟通过 siselect 和 sireg（实际上是 vsiselect 和 vsireg）间接访问的（虚拟） supervisor-level iprio 阵列。对于不在 hviprio1 和 hviprio2 支持子集中的中断，模拟的 iprio 数组中的优先级编号字节可以是只读 0。

在 VS 级仅为一组主要中断提供可配置优先级的硬件支持是一种妥协。当 M 模式和 HS 模式的所有捕获（包括中断和同步异常）都具有绝对优先级时，当每个虚拟 Hart 还可能与其他虚拟

Hart 争夺资源时，能够在 VS 层控制中断优先级的效用就显得有些虚无缥缈了。尽管如此，优先级可配置性还是为最有可能的中断子集提供了可能，同时最大限度地减少了必须在虚拟 Hart 交换机上交换的新增 CSR 数量。

优先级可配置子集之外的主要中断仍可定向到 VS 级，但其优先级将只是第 5.1 节中定义的默认顺序。

如果 hypervisor 确实必须模拟 hviprio1 和 hviprio2 支持的中断子集之外的中断优先级配置，可以通过设置 CSR hvictl 的第 VTI 位来实现，下一小节将对此进行介绍。

6.3.2 VS 级虚拟中断

假设虚拟主机不需要对 hviprio1 和 hviprio2 硬件支持的子集之外的主要中断配置优先级，那么 hypervisor 可以使用 CSR hvien（hypervisor 虚拟中断使能）和 hvip（hypervisor 虚拟中断挂起位）向虚拟主机发出中断。这些 CSR 对 VS 级中断的影响与 mvien 和 mvip 对 supervisor-level 中断的影响大致相同，详见第 5.3 节。

寄存器 hvien 和 hvip 的每一位都与 0-63 范围内的中断编号相对应。hvien 的 12:0 位为保留位，必须为只读 0，而 hvip 的 12:0 位则由 RISC-V 特权架构定义。具体来说，hvip 的第 10、6 和 2 位是可写位，分别对应 VS 级外部中断 (VSEIP)、VS 级定时器中断 (VSTIP) 和 VS 级软件中断 (VSSIP)。

以下内容仅适用于中断号 13-63 的 CSR 位：当 hideleg 中的某位为 1 时，vsip 中的相同位位置是 sip 中相应位的别名。否则，当 hideleg 中的某位为 0 且 hvien 中的匹配位为 1 时，vsip 中的相同位位置就是 hvip 中相应位的别名。当 hideleg 和 hvien 中的相应位都为零时，vsip 中的某位只读为零。表 6.1 总结了 hideleg 和 hvien 对 vsip 和 vsie 的综合影响。

hideleg[*n*]	hvien[*n*]	vsip[*n*]	vsie[*n*]
0	0	只读 0	只读 0
0	1	hvip[*n*] 的别名	可写
1	-	sip[*n*] 的别名	sie[*n*] 的别名

表 6.1：对于主要中断标识 13-63，hideleg 和 hvien 对 vsip 和 vsie 的影响

对于 13-63 号中断，当且仅当 hideleg 或 hvien 中的相应位被设置时，vsie 中的位才是可写的。当一个中断由 hideleg 授权时，vsie 中的可写位是 sie 中相应位的别名；否则，它就是一个独立的可写位。特权架构规定，vsie 的 12:0 位是 hie 中位的别名。通常，vsie 中不可写的位必须是只读 0。

如果 hideleg 中的某一位为 0，而 hvien 中的相应位从 0 变为 1，那么 vsie 中的匹配位的值就会变得不明确。同样，如果 hvien 中的某位为 1，而 hideleg 中的相应位从 1 变为 0，那么 vsie 中的匹配位的值也会变得不明确。

对于中断号 13-63，实现者可自由选择 hvien 中哪些位可写，哪些位只读为 0 或 1。如果 hvien 中的某个位只读为 0（防止启用虚拟中断），则 hvip 中的相同位也应只读为 0。中断 13-63 的所有其他位在 hvip 中必须是可写的。

CSR hvictl（Hypervisor Virtual Interrupt Control，hypervisor 虚拟中断控制）为在 VS 层注入中断提供了进一步的灵活性，以应对迄今为止所描述的设施无法完全支持的情况，但这需要 hypervisor 更积极的参与。出现以下情况时，hypervisor 必须使用 hvictl：

- 为 VS 级 asserted hvien 和 hvip 不支持的主要中断；
- 在 VS 级对 hviprio1 和 hviprio2 支持之外的主要中断的优先级进行配置；或
- 在不使用 IMSIC guest 中断文件的情况下，为虚拟主机模拟外部中断控制器，同时还支持外部中断和虚拟主机主要中断的可配置优先级。

hvictl 的格式为：

位 30	VTI
位 27:16	IID (**WARL**)
位 9	DPR
位 8	IPRIOM
位 7:0	IPRIO

hvictl 的所有其他位均为保留位，读作 0。

当位 VTI（虚拟陷阱中断控制）= 1 时，从 VS 模式明确访问 CSR sip 和 sie（或仅 RV32 的 siph 和 sieh）的尝试会导致虚拟指令异常。此外，对于任何给定的 CSR，如果在某种情况下对寄存器的写入可能导致 vsip 的某个位从 1 变为 0（不包括用于外部中断（SEIP）的第 9 位），那么当 VTI = 1 时，guest 写入该寄存器的任何尝试也会引发虚拟指令异常。在确定是否引发异常时，写入 CSR 的值和 vsip 的值（之前或之后）都将被忽略。（因此，写入寄存器实际上不需要将 vsip 的某个位从 1 变为 0 就会引发异常）。特别是，如果寄存器 vstimecmp 被实现（来自扩展 Sstc），那么当 VTI = 1 时，从 VS 模式写入 stimecmp（或仅针对 RV32 的 stimecmph）的尝试将导致虚拟指令异常。

---

*对于标准本地中断（主要特性 13-23 和 32-47）和软件中断（SSI），*vsip *中相应的中断挂起位被定义为 "粘性"，这意味着 guest 只能通过直接向* sip*（真正的* vsip*）写入值来清除它们。在标准定义的中断中，只有定时器中断（STI）可以通过向* vstimecmp *写入新值来清除。*

所有 hvictl 字段都会影响 CSR vstopi（Virtual Supervisor top 中断）的值，因此当中断捕获到 VS 模式时，vscause 中报告的中断标识也会受到影响。IID 是一个 **WARL** 无符号整数字段，至少有 6 位，而 IPRIO 始终是完整的 8 位。如果为 IID 设置了 *k* 位，则支持从 0 到 2^k - 1 的所有值，写入 hvictl 后，IID 将等于所写值的 (15 + *k*):16 位。

对于 hvictl 为 VS 级指定的虚拟中断，如果 VTI = 1 且 IID≠ 9，则字段 DPR（默认优先级）决定了该中断相对于主要标识 9 的（虚拟）上 级外部中断（SEI）的假定默认优先级顺序，如下所示：

0 = 中断的默认优先级高于 SEI
1 = 中断的默认优先级低于 SEI

当 hvictl.IID = 9 时，DPR 被忽略。

6.3.3 虚拟监控器最高中断 CSR (vstopi)

只读 CSR vstopi 的位宽为 VSXLEN 位，格式与 stopi 相同：

位 27:16 IID
位 7:0 IPRIO

vstopi 返回从这些候选中断（前缀为 + 号）中找到的 VS 级最高优先级中断的信息：

- 如果 vsip 和 vsie 中的第 9 位均为 1，则 hstatus.VGEIN 是 guest 中断文件的有效编号，且 vstopei 不为 0：

  + 一个优先级由 vstopei 表示的监督器外部中断（代码 9）；

- 如果 vsip 和 vsie 中的第 9 位均为 1，则 hstatus.VGEIN = 0，hvictl 字段 IID = 9 和 IPRIO≠ 0：

  + 一个优先级为 hvictl.IPRIO 的监督器外部中断（代码 9）；

- 如果 vsip 和 vsie 中的第 9 位都为 1，且前两种情况都不适用：

  + 一个优先级为 256 的监控器外部中断（代码 9）；

- 如果 hvictl.VTI = 0：

  使用由 hviprio1 和 hviprio2 分配的优先级编号，+ 除监控程序外部中断（代码 9）外，由 vsip 和 vsie 指示的优先级最高的挂起并启用的主要中断；

- 如果 hvictl 场 VTI = 1 和 IID≠ 9：

  + hvictl 字段 IID、DPR 和 IPRIO 指定的主要中断。

在上述列表中，所有 " supervisor "外部中断都是虚拟的，指向 VS 级，主要代码 9 位于 VS 级。

通过观察可以发现，候选中断列表的前三项是相互排斥的，而其余两项也是相互排斥的，因此可以比较容易地将候选中断列表缩减为两个最终候选中断。

当 hvictl.*VTI = 1 时，只能通过设置* hvictl.*IID = 9 来表示 VS 电平没有中断。软件可能希望在* hvictl.VTI = 1 时*使用 IID = 9、IPRIO = 0 表示*无中断。

当没有候选中断满足上述条件时，vstopi 为 0。否则，vstopi 字段 IID 和 IPRIO 由候选中断中优先级最高的中断决定。如第 66 页表 5.5 所述，supervisor-level 的优先级顺序通常适用，但优先级数字取自上述候选列表，而非 supervisor-level iprio 数组。除非 hvictl 字段 VTI = 1 且 IID≠ 9（上述候选列表中的最后一项），否则名义优先级中的并列关系将按照第 5.1 节中的默认优先级顺序打破，在这种情况下，默认优先级顺序完全由 hvictl.DPR 决定。如果 hvictl 的 IPRIOM（IPRIO 模式）位为 0，则 vstopi 中的 IPRIO 为 1；否则，如果优先级最高的候选码的优先级在 1 至 255 之间，则 IPRIO 为该值；否则，IPRIO 将按照第 5.4.2 节所述 stopi 的方式设置为 0 或 255。

6.3.4 中断陷阱转为 VS 模式

高级中断架构修改了标准 RISC-V 特权架构，当且仅当 vstopi 不为零时，VS 级中断才处于挂起状态。CSR vsip 和 vsie 本身并不决定 VS 级中断是否挂起，但它们可能通过对 vstopi 的影响间接决定 VS 级中断是否挂起。

当 vstopi 不为 0 时，如果当前权限模式为 VS 模式且 CSR vsstatus 中的 SIE 位为 1，或者当前权限模式为 VU 模式，则会对 vstopi 字段 IID 所指示的中断进行 VS 模式捕获。

vscause 的 "异常代码 "字段必须至少有足够多的比特来表示 vstopi 的 IID 字段对给定 hart 的最大值。

7.处理器间中断 (IPI)

默认情况下，除非一个平台有不同的处理器间中断（IPI）机制，否则 RISC-V 专用体系结构规定，具有多个 hart 的 machine 必须为每个 hart 提供一个实现定义的内存地址，该地址可被写入以在该 hart 发出 machine-level *软件中断*信号（主要代码 3）。因此，machine-level IPI 可以作为 machine-level 软件中断发送到任何 hart。

RISC-V 软件中断只起到最基本的 "门铃 "信号作用。接收端的软件负责将收到的软件中断识别为 IPI，并进一步解码其目的，通常是利用发送方存储在普通内存中的附加数据。

同样的机制（但内存地址集不同）可能存在，也可能不存在，用于向远程 hart 发送 supervisor-level 软件中断信号（主要代码 1）。如果不直接支持这种方式，通常会通过从 supervisor 模式到 machine 模式的环境调用，将 supervisor-level 软件中断发送到另一个 Hart。因此，在 S 模式下运行的操作系统会调用特定的 SBI 功能，将软件中断发送到另一个 hart，从而导致始发 hart 上的 machine-level 软件向目标 hart 发送 machine-level IPI，然后软件在 CSR mip 中设置 supervisor-level 软件中断挂起位 (SSIP)。

当 hart 具有 IMSIC 时，IPI 可通过写入目标 hart 的 IMSIC 发送到 hart，而不是使用特权架构的机制向远程 hart 发送软件中断信号，这与常规的消息信号中断 (MSI) 相同。在这种情况下，传入的 IPI 将作为通过 IMSIC 路由的*外部中断*出现在目标 hart，而不是像以前那样作为软件中断出现。不过，只要 IPI 的两个端点（源端和目的端）由相同的软件（如操作系统或 machine 监控器）控制，目的端就没有理由误解代表 IPI 的传入外部中断的目的。

如果微处理器没有 IMSIC，则假定 IPI 使用 RISC-V 特权架构指定的方法，在目标微处理器上发出软件中断信号。另一方面，当 Harts 具有 IMSIC 时，在远程 Harts 触发软件中断的机制与 IMSIC 的功能是多余的，因此它从一个必要条件降级为一个可选项，也许只是为了在一系列 RISC-V 系统（无论是否具有 IMSIC）中提供软件兼容性。

如果 machine 实现的是 IMSIC，而不是早期的软件中断机制，那么用于 machine-level 软件中断 MSIP 和 MSIE 的 CSR mip 和 mie 位将在 harts 中硬连接为 0。

如果一台 machine 实现了 IMSIC，但没有软件中断机制，那么后者仍可在 S 模式或 VS 模式的 supervisor-level 上完全仿真，方法是在写入特殊内存地址时进行捕获，这些地址应在远程 hart 上发出 supervisor-level 软件中断信号。在出现这种陷阱时，软件可通过 IMSIC 向目标 hart 发送高级 IPI，然后高级软件可在目标权限级别（S 或 VS）的 sip *中设置 SSIP 位。*

同样，在 SBI 环境中，发送 IPI 的要求可以很容易地继续得到支持，而 guest 完全不会意识到在 Harts 之间传送 IPI 的底层硬件发生了变化。

当软件通过向其他 Hart 的 IMSIC 写入 MSI 来发送 IPI 时，程序员还应考虑在每条写入 MSI 的存储指令之前执行 FENCE 指令的必要性。在没有 FENCE 的情况下，许多系统只能保证在单个设备之间，而不是在多个设备之间保持 Hart 的加载和存储顺序，在访问主存储器时更是如此。在这种系统中，必须记住每个 IMSIC 都可能被视为众多设备中的一个独立设备。例如，如果 Hart A 想要通知 Hart B 它已经完成了一项涉及访问某些 I/O 设备的任务，Hart A 可能需要在向 B 的 IMSIC 发送 MSI 之前执行 FENCE，以确保在 MSI 到达 B 之前，A 对该设备的所有访问都已实际完成。

8.IOMMU 对虚拟机 MSI 的支持

系统中存在 IOMMU 后，在虚拟机中运行的 guest 操作系统就可以直接控制 I/O 设备，而只需 hypervisor 进行最少的干预。直接控制设备的 guest 操作系统将使用 guest 物理地址对设备进行编程，因为操作系统只知道这些地址。当设备使用这些地址执行内存访问时，IOMMU 将负责参考 hypervisor 提供的地址转换数据结构，将这些 guest 物理地址转换为 machine 物理地址。

要处理由 guest 操作系统控制的设备发出的 MSI，IOMMU 必须能够将这些 MSI 重定向到 IMSIC 中的 guest 中断文件。没有带有 guest 中断文件的 IMSIC 的系统不需要实现本章所述的功能。

由于来自设备的 MSI 只是简单的内存写入，因此它们自然会受到 IOMMU 对其他内存写入所适用的相同地址转换。不过，高级中断架构要求 IOMMU 特殊处理指向虚拟机的 MSI，部分原因是为了简化软件，部分原因是为了允许可选支持*内存驻留中断文件*。

本章使用的 *IOMMU* 一词是泛指 IOMMU，包括虚拟化设备访问所需的所有翻译和转换处理服务，仅涉及 IOMMU 如何识别和处理指向虚拟机的 MSI。IOMMU 的大多数其他功能和细节超出了本标准的范围，必须在其他地方加以说明。

RISC-V IOMMU 架构规范详细描述了 IOMMU 架构，将翻译和事务处理功能划分为 IOMMU、IO Bridge 等模块，并描述了这些模块如何集成到系统中。

如果单个物理 I/O 设备可由多个独立的设备驱动程序细分控制，则此处将每个子设备称为一个设备。

8.1 IOMMU 的设备上下文

系统中的 IOMMU 假设如下：

- 对于通过 IOMMU 连接到系统的每个 I/O 设备，软件可以在 IOMMU 上配置一个*设备上下文*，该上下文与设备的特定虚拟地址空间和 IOMMU 可能支持的其他每个设备参数相关联。通过在 IOMMU 上为每个设备提供独立的设备上下文，每个设备都可以单独配置为独立的操作系统，可以是 guest 操作系统，也可以是主（主机）操作系统。在设备发起的每一次内存访问中，硬件都会通过某种形式的唯一设备标识符向 IOMMU 指出发起访问的设备，IOMMU 利用这些标识符在软件提供的数据结构中找到相应的设备上下文。例如，对于 PCI，可通过 PCI 总线编号、设备编号和功能编号的三重唯一标识符来识别初始设备。

- IOMMU 可选择使用地址转换数据结构（通常是由软件通过相应的设备上下文指定的页表）转换设备内存访问的地址。所有 IOMMU 实现的最小地址转换粒度都不大于 4-KiB 页，与标准 RISC-V 地址转换页表的粒度一致。(事实上，IOMMU 可以采用与 RISC-V 高级体系结构定义的基于页的地址转换格式相同的页表，但这并非必需）。

高级中断架构会根据需要在设备上下文中添加这些字段：

- *MSI 地址掩码*和*地址 pattern*，共同用于识别 guest 物理地址（gpa）空间中作为 MSI 目的地的页面；以及

- *MSI 页表*的实际物理地址，用于控制设备的 MSI 转换和/或翻译。

MSI 地址掩码和地址 pattern 均为无符号整数，宽度与 guest 物理页码相同，即比 guest 物理地址的最大支持宽度窄 12 位。第 8.4 节 "虚拟机中断文件页面地址的识别 "将解释它们的用途。

设备上下文的 MSI 页表与通常的地址转换数据结构是分开的，后者用于转换来自同一设备的其他内存访问。MSI 页表的形式和功能是本章其余大部分内容的主题。

为设备上下文提供独立的 MSI 页表有两个原因：

首先，在 Linux 或类似操作系统下运行的 hypervisor 可以受益于 MSI 译码的单独控制，以帮助简化虚拟机从一个物理机迁移到另一个物理机的情况。如第 6.1.2 节所述，当虚拟 hart 的中断文件映射到真实计算机中的 guest 中断文件时，虚拟 hart的迁移会导致这些虚拟中断文件底层的物理 guest 中断文件发生变化。然而，由于在其他系统（非 RISC-V）上，虚拟中断的迁移并不影响从 guest 物理地址到实际物理地址的映射，因此执行这种迁移的 Linux 内部功能并没有设置为修改 IOMMU 的地址转换表，以适应 RISC-V 虚拟中断文件物理位置的变化。让 hypervisor 控制 IOMMU 上单独的 MSI 转换表，可以绕过这一限制。Hypervisor 和/或管理中断的子系统可以随意修改 MSI 页表，而无需与其他许多与常规地址转换相关的操作系统组件进行协调。

*其次，指定单独的 MSI 页表有利于使用*内存驻留中断文件（*MRIF*），*第 8.3 节将介绍 MRIF。专用的 MSI 页表可以轻松支持 MRIF 的特殊表项格式（第 8.5.2 节），这种格式完全是外来的，很难改装成其他地址转换数据结构。*

8.2 设备 MSI 地址转换

如图 8.1 所示，为了支持将 MSI 从 I/O 设备直接传输到 RISC-V 虚拟机，而无需 hypervisor 的干预，IOMMU 必须能够将 MSI 的 guest 物理地址转换为 IMSIC 在虚拟机中的 guest 中断文件的实

际物理地址。这种地址转换由 IOMMU 在相应设备上下文中配置的 MSI 页表控制。由于每个中断文件，无论是真实的还是虚拟的，都占用一个自然对齐的 4-KiB 页面地址空间，因此所需的地址转换是从虚拟（guest）页面地址到物理页面地址，与常规 RISC-V 基于页面的地址转换所支持的相同。

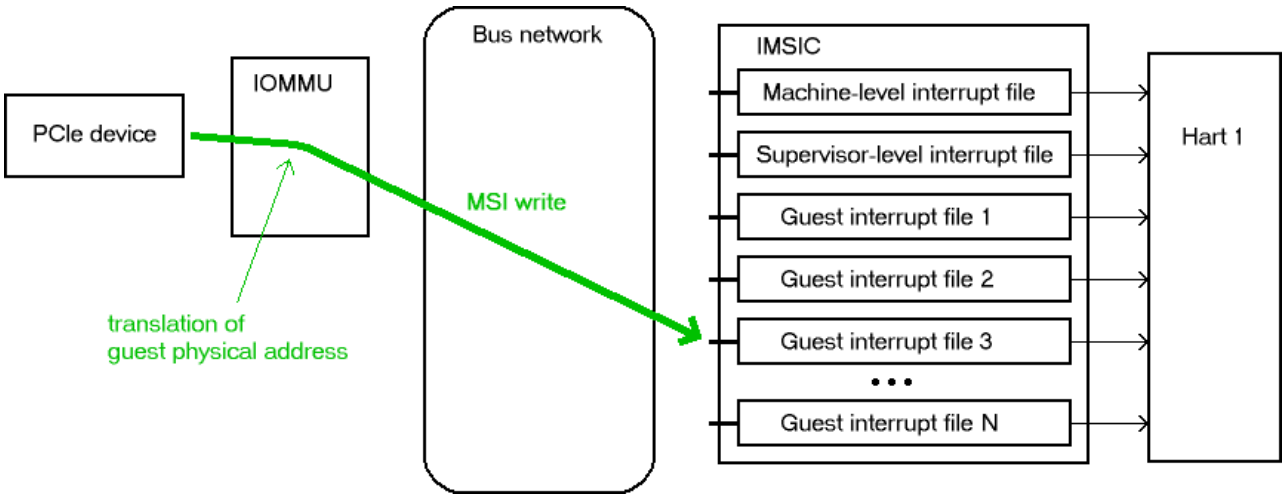

图 8.1：Guest OS 将设备源 MSI 转换为操作系统虚拟机中的（虚拟）IMSIC 中断文件。IOMMU 参考控制 hypervisor 提供的 MSI 页表，将 MSI 重定向到真实机器的 guest 中断文件。

根据写入的目标地址，可将来自设备的内存写入识别为 MSI。如果 IOMMU 确定 32 位写入的是相关虚拟机中（虚拟）中断文件的位置，则该写入被视为虚拟机中的 MSI，否则就不是。识别 MSI 的确切公式见第 8.4 节。

虽然 MSI 的翻译由其独立的页表控制，但 MSI 翻译与常规 RISC-V 地址翻译的页粒度相同，这意味着 IOMMU 中的地址翻译缓存几乎不需要修改就能同时缓存 MSI 翻译。只有在翻译缓存未命中时，IOMMU 才需要将 MSI 与来自同一设备的其他内存访问区别对待，以选择正确的翻译表，并正确访问和解释该表。

8.3 内存驻留中断文件

IOMMU 可以选择支持内存驻留中断文件（MRIF）。如果使用内存驻留中断文件，可以大大增加系统中可直接控制一个或多个物理设备的虚拟 Harts 的数量，前提是系统的其他部分仍能处理增加的负载。

如果没有内存驻留中断文件，可以直接从设备接收 MSI 的虚拟 RISC-V hart 的数量就会受到系统中所有 IMSIC 实现的 guest 中断文件总数的限制，因为 RISC-V hart 的所有 MSI 都必须通过 IMSIC。对于单个 RISC-V hart，guest 中断文件的数量是特权架构定义的 *GEILEN* 参数，RV32 最多为 31，RV64 最多为 63。

另一方面，使用内存驻留中断文件后，能够接收设备 MSI 的虚拟 RISC-V Harts 总数几乎不受任何限制，仅受实际物理内存量和处理这些中断所需的额外处理时间的限制。正如其名称所示，驻留在内存中的中断文件位于内存中，而不是 IMSIC 中。图 8.2 描述了 IOMMU 如何在 MRIF 中记录传入的 MSI。如果 hypervisor 配置得当，IOMMU 就会识别出某些传入的 MSI 是为特定虚拟中

断文件准备的，并通过设置存储在普通内存 MRIF 数据结构中的中断挂起位来记录每个此类 MSI。在 MRIF 中记录每个 MSI 后，IOMMU 还会向 hypervisor 发送一个*通知 MSI*，告知 MRIF 内容可能已更改。

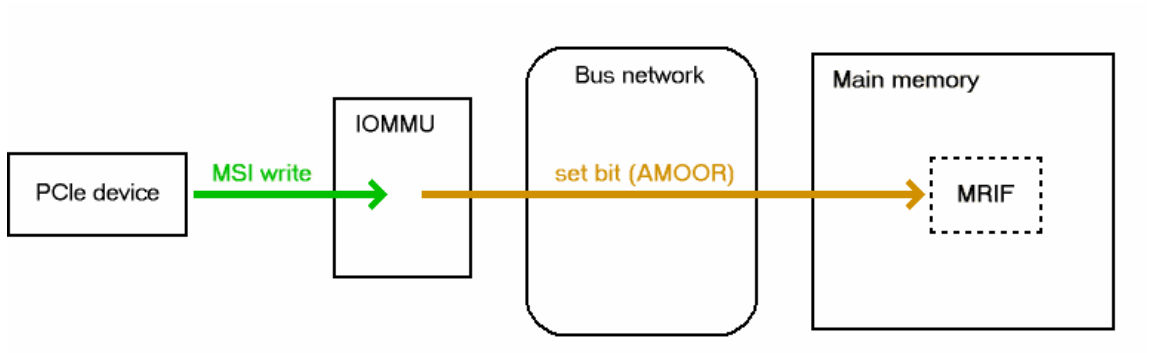

图 8.2：将传入的 MSI 记录到内存驻留中断文件 (MRIF) 中，而不是像图 8.1 中那样将其发送到 guest 中断文件中。

虽然内存驻留中断文件提供了记录 MSI 的位置，但它无法像 IMSIC 的 guest 中断文件那样直接中断 Hart。Hypervisor 收到的 MSI 通知只表明虚拟 hart *可能*需要中断；hypervisor 负责每次检查 MRIF 内容，以确定是否真的要中断虚拟 hart。此外，IMSIC 的 guest 中断文件可以直接充当虚拟 hart 的 supervisor-level 中断文件，而在虚拟 hart 执行时将虚拟 hart 的中断文件保存在 MRIF 中，则需要 hypervisor 为虚拟 hart 模拟 supervisor-level 中断文件，隐藏底层 MRIF。根据虚拟 hart 接触其中断文件的频率和实现对 MRIF 的支持程度，这种模拟的成本可能会很高。

因此，MRIF 最常用于那些由于闲置或几乎闲置而被 "换出 "物理 hart 的虚拟 hart。当虚拟机 hypervisor 确定进入 MRIF 的 MSI 应唤醒某个闲置的虚拟机时，可在 IMSIC 中为该虚拟机分配一个 guest 中断文件，并在恢复虚拟机之前将其中断文件从 MRIF 移入该 guest 中断文件。当然，为新唤醒的虚拟 hart 分配 guest 中断文件的过程可能会迫使另一个虚拟 hart 的中断文件被驱逐到自己的 MRIF 中。

并非所有系统都需要容纳大量闲置的虚拟工作线程。例如，许多批处理服务器会努力使所有虚拟工作线程从开始到结束都尽可能忙碌，只受 I/O 延迟和处理资源限制的制约。在这种环境下，只要参数 GEILEN 不是太小，对 MRIF 的支持可能并无用处。

IOMMU 对内存驻留中断文件的支持可分为三个级别：

- 没有内存驻留中断文件；
- 内存驻留中断文件，无原子更新；或
- 内存驻留中断文件的原子更新。

当内存系统支持与 RISC-V 指令 AMOAND 和 AMOOR 相对应的逻辑原子内存操作 (AMO) 时，内存驻留中断文件的效率最高。内存驻留中断文件*的原子更新需要* AMOAND 和 AMOOR 操作。如果不使用 AMO，仅依靠基本的内存读写，支持水平可能会降低。

8.3.1 内存驻留中断文件格式

驻留内存的中断文件占用 512 字节内存，自然对齐到 512 字节地址边界。这 512 字节是由 32 对 64 位双字组成的数组，总共 64 个双字。每个双字以小端字节顺序排列（即使在所有字节都是大

端的系统中也是如此）。

使用 MRIF 的 Big-endian 配置 Harts 预计将执行由标准 RISC-V 扩展 Zbb 定义的 REV8 字节反转指令，或使用一系列指令承担字节序转换的成本。

在这种排列中，成对的双字包含外部中断标识 1-2047 的中断挂起位和中断使能位：

偏移	尺寸	内容
0x000	8 个字节	ID 1-63 的中断挂起位
0x008	8 个字节	ID 1-63 的中断使能位
0x010	8 个字节	ID 64-127 的中断挂起位
0x018	8 个字节	ID 64-127 的中断使能位
.. .		.. .
0x1F0	8 个字节	ID 1984-2047 的中断挂起位
0x1F8	8 个字节	ID 1984-2047 的中断使能位

一般来说，对于整数 *k*，地址偏移 k × 16 和 *k* × 16 + 8 的一对双字包含外部中断小标识的中断挂起位和中断使能位，范围为 *k* × 64 至 *k* × 64 + 63。对于该范围内的标识 *i*，第一个（偶数）双字的位（*i* mod 64）是中断挂起位，第二个（奇数）双字的相同位是中断使能位。

中断挂起位和中断使能位通过双字交错存储在 MRIF 中，以方便 IOMMU 检查相关使能位，确定是否在更新挂起位后发送通知 MSI，而不是在更新后始终发送通知 MSI 而不考虑中断使能位的默认行为。只有在不支持原子更新的 MRIF 时，内存安排才会有影响。

MRIF 第一个双字的第 0 位存储了不存在的中断 0 的假中断挂起位。如果来自 I/O 设备的写入似乎是应存储在 MRIF 中的 MSI，但要写入的数据（中断标识）为 0，则 IOMMU 将 0 视为有效的中断标识，设置目标 MRIF 第一个双字的第 0 位，并像往常一样发送通知 MSI。

所有 MRIF 的大小均可容纳 2047 个有效中断标识，这是 IMSIC 中断文件允许的最大值。如果系统的实际 IMSIC 中断文件仅实现 *N* 个中断标识（*N* < 2047），那么软件可能会忽略 MRIF 中大于 *N* 的标识内容。然而，IOMMU 会将每个 MRIF 视为 0-2047 范围内的所有中断标识，即使软件忽略了无效标识 0 和大于 *N* 的所有标识。

对于小于 2047 个有效中断标识的 MRIF，没有必要向 IOMMU 指定所需的大小 N。IOMMU 对该信息的唯一用途是丢弃任何显示中断标识大于 N 的 MSI。如果设备由软件正确配置，这种错误的 MSI 应该不会发生；但即使发生了，软件在 MRIF 中记录虚假中断标识后将其忽略，与 IOMMU 在 MRIF 中记录虚假中断标识前将其丢弃一样有效。

同样，IOMMU 也没有必要检查并丢弃显示无效中断标识为 0 的 MSI。

8.3.2 将收到的 MSI 记录到内存驻留中断文件中

MSI 写入的数据组件指定了要在目标中断文件中触发的中断标识（回顾第 3.2 节）。该数据可以是小端顺序或大端顺序。如果 IOMMU 支持内存驻留中断文件，则可将与 machine IMSIC 所接受的字节序相同的中断标识存储到 MRIF 中。所有 IMSIC 中断文件都必须接受以小端顺序写入内存映射寄存器 seteipnum_le 的 MSI（第 3.5 节）。如果寄存器 seteipnum_be 与 seteipnum_le 同时执行，IMSIC 中断文件也可以接受大端顺序的 MSI。

如果 MSI 数据指示的中断标识（按正确字节顺序解释时）在 0-2047 范围内，则 IOMMU 通过将 MRIF 中该标识的中断挂起位置 1，将 MSI 存储到 MRIF 中。如果 MRIF 支持原子更新，则通过 AMOOR 操作设置挂起位，否则通过非原子读-修改-写序列设置挂起位。在 MRIF 中的中断挂起

位被设置后，IOMMU 将发送软件为 MRIF 配置的 MSI 通知。

将 MSI 存储到 MRIF 的具体过程在第 8.5.2 节中有更精确的说明，该节涉及以 *MRIF 模式*配置的 MSI 页表项。

IOMMU 是否可以选择性地检查目标 MRIF 中匹配的中断使能位，以决定是否在设置中断挂起位后发送通知 MSI，这是一个未决问题。目前，要求 IOMMU 在将 MSI 存储到 MRIF 后始终发送通知 MSI，即使中断标识的相应使能位为零。

8.3.3 使用内存驻留中断文件进行原子更新

要使用支持原子更新的内存驻留中断文件，除了第 8.3.1 节中的 MRIF 结构本身外，软件还必须有内存位置来保存 IMSIC 中断文件的 eidelivery 和 eithreshold 寄存器。

将虚拟 Hart 中断文件从 IMSIC 移入 MRIF 涉及以下步骤：

1. 准备 MRIF，将其所有中断挂起位（偶数双字）清零，并将 IMSIC 中断文件的 eie 数组复制到 MRIF 的中断使能位（奇数双字）。
2. 将 IMSIC 中断文件寄存器 eidelivery 和 eidelivery 的现有值保存到内存中。并设置 eidelivery = 0。
3. 修改 IOMMU 的所有相关翻译表，使该虚拟中断文件的 MSI 现在存储在 MRIF 中。如有必要，与所有 IOMMU 同步，以确保在此步骤后不会有滞留的 MSI 到达 IMSIC 中断文件。
4. 使用 AMOOR 操作，将 IMSIC 中断文件 eip 数组的内容与 MRIF 的中断挂起位进行逻辑 OR。

该序列完成后，IMSIC 中断文件将不再使用。

每次有 MSI 通知到达，表明 MRIF 中存储了 MSI 时，控制 hypervisor 都应扫描 MRIF 的中断挂起位和中断使能位，以确定是否有任何启用的中断现在处于挂起和启用状态，从而中断虚拟机。

有了 MRIF 的原子更新，只要虚拟机 hypervisor 为虚拟机 hypervisor 模拟一个适当的 IMSIC 中断间文件来隐藏底层 MRIF，虚拟机 hypervisor 就可以在 MRIF 中保存中断文件的情况下继续执行。Hypervisor 软件可以使用 AMOOR 和 AMOAND 操作安全地设置和清除 MRIF 的中断挂起位和中断使能位，即使 IOMMU 可能将传入的 MSI 存储到同一 MRIF 中。

如果 IOMMU 被配置为检查 MRIF 的中断使能位以决定是否发送通知 MSI，那么修改这些启用位通常需要与 IOMMU 协调。但只要 IOMMU 像目前假设的那样忽略中断使能位，软件就可以毫无风险地更改这些位。

将同一中断文件从 MRIF 移回 IMSIC：

1. 在新的 IMSIC 中断文件中，设置 eidelivery = 0，并将 eip 数组清零。
2. 修改 IOMMU 的所有相关翻译表，以便将该虚拟中断文件的 MSI 发送到 IMSIC 中断文件。如有必要，与所有 IOMMU 同步，以确保在此步骤后，MRIF 中不会存储滞后的 MSI。
3. 使用 CSRS 指令写入 eip 数组，将 MRIF 中的中断挂起位逻辑 OR 到 IMSIC 中断文件中。同时，将 MRIF 中的中断使能位复制到 IMSIC 中断文件的 eie 数组中。
4. 用之前保存的值加载 IMSIC 中断文件的寄存器 eithreshold 和 eidelivery。

8.3.4 使用内存驻留中断文件，无需原子更新

在不支持原子更新的情况下，内存驻留中断文件的使用与上一小节的原子更新情况类似，但增加了一些复杂性。

首先，如果虚拟 hart 控制的 I/O 设备位于多个 IOMMU 之后，那么就需要多个 MRIF 结构，每个 IOMMU 一个，而不仅仅是一个 MRIF 结构。此外，除了用于存储 eidelivery 和 eithreshold 的位置外，软件还需要在 MRIF 之外放置中断文件执行的 eip 数组的完整副本。当虚拟中断文件在内存中时，其中断挂起位将被分割到所有 MRIF 和保存的 eip 阵列中。中断使能位可能只存在于 MRIF 中。

将虚拟 Hart 中断文件从 IMSIC 移入内存，每个 IOMMU 一个 MRIF：

1. 将所有 MRIF 的中断挂起位（偶数双字）清零，并将 IMSIC 中断文件的 eie 数组复制到 MRIF 的中断使能位（奇数双字），从而为所有 MRIF 做好准备。
2. 将 IMSIC 中断文件寄存器 eidelivery 和 eidelivery 的现有值保存到内存中。并设置 eidelivery = 0。
3. 在每个 IOMMU，修改所有相关的转换表，使该虚拟中断文件的 MSI 现在存储在与 IOMMU 匹配的各个 MRIF 中。如有必要，同步所有 IOMMU，以确保在这一步骤之后，IMSIC 中断文件中不会出现滞留的 MSI。
4. 将 IMSIC 中断文件的 eip 数组转存到 MRIF 之外的独立位置。

该序列完成后，IMSIC 中断文件将不再使用。

当虚拟 hart 的中断文件保留在内存中时，中断标识的真正挂起位是其在所有 MRIF 中的位和在保存的 eip 阵列中的位的逻辑 OR。MRIF 中的所有挂起位开始时都是 0，但当该虚拟 Hart 的 MSI 到达 IOMMU 并存储在相应的 MRIF 中时，中断就可能挂起了。

如果不对 MRIF 进行原子更新，就很难清除 MRIF 中的中断挂起位。(清除一个 MRIF 中的单个挂起位需要分配和初始化一个新的 MRIF，并重新配置相应的 IOMMU，以便将 MSI 存储到新的 MRIF 中）。因此，在内存中保留一个中断文件的同时执行虚拟 Hart 可能可行，也可能不可行。当 MRIF 记录了一个应该唤醒虚拟 hart 的中断时，最简单的策略是在恢复执行虚拟 hart 之前，始终将中断文件移回 IMSIC 的 guest 中断文件中。

将中断文件从内存传回 IMSIC：

1. 在新的 IMSIC 中断文件中，设置 eidelivery = 0，并将 eip 数组清零。
2. 修改 IOMMU 的所有相关翻译表，以便将该虚拟中断文件的 MSI 发送到 IMSIC 中断文件。如有必要，与所有 IOMMU 同步，以确保在此步骤后 MRIF 中不会存储滞留的 MSI。
3. 通过比特逻辑 OR 合并所有 MRIF 的中断挂起位和保存的 eip 数组，并将这些合并位逻辑 OR 到 IMSIC 中断文件中，使用 CSRS 指令写入 eip 数组。同时，将其中一个 MRIF 中的中断使能位复制到 IMSIC 中断文件的 eie 数组中。
4. 用之前保存的值加载 IMSIC 中断文件的寄存器 eithreshold 和 eidelivery。

8.3.5 为接收通知 MSI 分配 guest 中断文件

通过为每个 MRIF 分配单独的中断标识，可以最大限度地减少 hypervisor 为响应通知 MSI 而进行的处理，因此通知 MSI 中编码的标识总是显示哪个 MRIF 可能发生了变化。但是，如果 MRIF 数量非常多（可能达到数千个），hypervisor 可能会出现 IMSIC 中可用的 supervisor-level 中断文件中的中断标识不足的情况。在这种情况下，hypervisor 可以将一个或多个 IMSIC 的 guest 中断文件分配给自己，以接收 MSI 通知，从而增加中断标识的供应。

虽然 guest 中断文件的存在主要是为了充当虚拟 Harts 的 supervisor-level 中断文件，但 IMSIC 硬件并没有明确规定软件如何使用这些文件。

8.4 识别虚拟机中断文件的页地址

当 I/O 设备由 guest 操作系统直接配置时，来自设备的 MSI 预计会针对 guest 操作系统虚拟机中的虚拟 IMSIC，使用对真实机器不合适和不安全的 guest 物理地址。IOMMU 必须将来自此类设备的某些写入识别为 MSI，并根据实际机器的需要进行转换。(回顾图 8.1）。

来自单个设备的需要转换的 MSI 预计已由运行在一个 RISC-V 虚拟机中的单个 guest 操作系统在该设备上进行了配置。假设虚拟机本身符合高级中断架构，那么 MSI 将通过写入虚拟 IMSIC 中断文件的内存映射寄存器发送到虚拟机内的虚拟 Harts。每个虚拟中断文件都占用虚拟机 guest 物理地址空间中单独的 4-KiB 页面，这与真实中断文件在真实 machine 物理地址空间中的占用情况相同。因此，如果写入的是虚拟机中虚拟 IMSIC 的中断文件所占用的页面，则对 guest 物理地址的写入可被识别为对虚拟机的 MSI。

设备上下文（第 8.1 节）中指定的 MSI 地址掩码和地址 pattern 用于识别相关虚拟机 guest 物理地址空间中虚拟中断文件的 4-KiB 页。如果目标 guest 物理页面在所提供的地址掩码中所有为 0 的位位置上都与所提供的地址 pattern 相匹配，则设备进行的 32 位写入将被识别为对虚拟中断文件的 MSI 写入。具体来说，在以下情况下，对 guest 物理地址 *A* *的*内存访问就是对虚拟中断文件内存映射页的访问。

(*A* >> 12) & ~地址掩码 = (地址 pattern & ~地址掩码)

其中，>> 12 表示向右移动 12 位，"&"表示位逻辑 AND，"~ 地址掩码 "是地址屏蔽的位逻辑补码。

当发现内存访问是对虚拟中断文件的访问时，*就会*从原始 guest 物理地址中提取*中断文件编号，*即

中断文件号 = extract（*A* >> 12，地址掩码）

这里，extract(*x*, *y*) 是一种 "比特提取"，它丢弃 *x* 中与掩码 *y* *中*相同位置的匹配比特为零的所有比特，并将 x 中的剩余比特连续打包到结果的最末端，保持与 *x* *相同的比特*顺序，并将结果最高有效端的其他比特填充为零。例如，如果 *x* 和 *y* 是

x = a b c d e f g h
y = 1 0 1 0 0 1 1 0

那么 extract(*x*, *y*) 的值的位数为 0 0 0 0 a c f g。

8.5 MSI 页表

当 IOMMU 确定内存访问是对上一节规定的虚拟中断文件的访问时，将通过查询为设备配置的 MSI 页表对访问进行翻译或转换，而不是使用适用于同一设备所有其他内存访问的常规翻译数据

结构。

MSI 页表是由 MSI 页表条目（MSI PTE）组成的平面阵列，每个条目有 16 个字节。MSI 页表没有像普通 RISC-V 页表那样的多级层次结构。相反，每个 MSI PTE 都是一个叶子条目，指定对虚拟中断文件在相关虚拟机中占用（或可能占用）的特定 4-KiB guest 物理页的访问的转换。要从 MSI 页表中选择单个 MSI PTE，PTE 数组的索引是根据上一节公式从传入内存访问的目标 guest 物理地址中提取的中断文件编号。每个 MSI PTE 可以指定一个真实 guest 中断文件的地址，以替代目标虚拟中断文件（如图 8.1 所示），也可以指定一个内存驻留中断文件，用于存储虚拟中断文件的传入 MSI（如图 8.2 所示）。

MSI 页表的条目数为 2^k，其中 k 是用于从目标 guest 物理地址中提取中断文件号的 MSI 地址掩码中为 1 的位数。如果 MSI 页表的条目数少于或等于 256 个，则页表的起始位置将与实际物理内存中的 4-KiB 页地址对齐。如果 MSI 页表有 $2^k > 256$ 个条目，则页表必须自然对齐到 $2^k \times 16$ 字节地址边界。如果 MSI 页表未按要求对齐，则表中的所有条目在 IOMMU 看来都是未指定的，而且 IOMMU 可能计算并用于从表中读取单个 MSI PTE 的任何地址也是未指定的。

每个 16 字节的 MSI PTE 被解释为两个 64 位双字。如果 IOMMU 也引用 RISC-V 特权体系结构定义的标准 RISC-V 页表进行常规地址翻译，那么内存中两个双字节（小端或大端）的字节顺序应与为同一设备上下文配置的常规 RISC-V 页表的大小端相同。否则，MSI PTE 双字节的字节序由实现定义。

MSI PTE 第一个双字的第 0 位是字段 V（有效）。当 V = 0 时，PTE 无效，IOMMU 将忽略这两个双字的所有其他位，以便软件自由使用。

如果 V = 1，则第一个双字的第 63 位为字段 C（自定义），指定用于自定义用途。如果 MSI PTE 的 V = 1 和 C = 1，则 PTE 其余部分的解释由执行定义。

如果 V = 1，自定义使用位 C = 0，则第一个双字的第 2:1 位包含字段 M（模式）。如果 M = 3，则 MSI PTE 指定了访问页面的*基本转换模式*；如果 M = 1，则指定了 *MRIF 模式*。M 的 0 和 2 值为保留值。下面两个小节将进一步详细说明 MSI PTE 对两种定义模式的解释。

8.5.1 MSI PTE，基本转换模式

当 MSI PTE 的字段为 V = 1、C = 0 和 M = 3（基本转换模式）时，PTE 的完整格式为

第一个双字：	位 63	C, = 0
	位 53:10	PPN
	位 2:1	M, = 3
	位 0	V, = 1
第二个双字：	忽略	

第一个双字的所有其他位都是保留位，必须由软件设置为零。第二个双字被 IOMMU 忽略，因此软件可以自由使用。

在 MSI PTE 所覆盖的页面内进行内存访问时，会用 PTE 中的字段 PPN（物理页码）替换访问的原始地址位 12 及以上（guest 物理页码），同时保留原始地址位 11:0（页内偏移）。翻译后的地址将根据需要进行零扩展或上端剪切，使其与机器的实际物理地址宽度一致。然后，设备的原始内存访问将以新地址传递给内存系统。

基本转换模式下的 MSI PTE 允许 hypervisor 将用于虚拟中断文件的 MSI 写入转发到机器中真实

IMSIC 的 guest 中断文件。

同时采用标准 RISC-V 页表进行常规地址转换的 IOMMU 可以最大限度地重叠处理 MSI PTE 和常规 RISC-V 叶 PTE，具体如下：

对于 RV64，基本转换模式下 MSI PTE 的第一个双字与 Sv39、Sv48、Sv57、Sv39x4、Sv48x4 或 Sv57x4 基于页面地址翻译的常规 RISC-V 叶 PTE 的编码相同，PTE 字段 D、A、G、U 和 X 均为 0，W = R = 1。因此，MSI PTE 的第一个双字与授予读写权限（R = W = 1）但不授予执行权限（X = 0）的常规 PTE 相同。除了 PTE 的访问（A）、脏（D）和用户（U）位全部为 0 外，这种相同编码的常规 PTE 将与实际的 MSI PTE 一样翻译 MSI 写入。对于 MSI PTE 和普通 RV64 leaf PTE，IOMMU 只需对这三个位进行不同处理。

用于从常规 RISC-V 页表中选择 PTE 的地址计算必须进行修改，以便从 MSI 页表中选择 MSI PTE 的第一个双字。然而，从 guest 物理地址中提取中断文件号，以获得访问 MSI 页表的索引，已经在 PTE 寻址中造成了不可避免的差异。

对于 RV32，MSI PTE 第一个双字的低 32 位字的格式与 Sv32 或 Sv32x4 基于页的地址转换的叶 PTE 的格式相同，但 PTE 位 A、D 和 U 必须区别对待。

8.5.2 MSI PTE，MRIF 模式

如果支持内存驻留中断文件，且 MSI PTE 的字段 V = 1、C = 0 和 M = 1（MRIF 模式），则 PTE 的完整格式为

第一个双字：	位 63	C, = 0
	位 53:7	MRIF 地址[55:9] 位
	2:1	M, = 1
	位 0	V, = 1
第二个双字：	位 60	NID[10］
	位 53:10	NPPN
	位 9:0	NID[9:0］

所有其他 PTE 位均为保留位，必须由软件设置为零。

PTE 的 MRIF 地址字段提供了内存驻留中断文件物理地址的第 55:9 位，用于存储传入的 MSI，称为*目标 MRIF*。由于每个内存驻留中断文件都自然对齐到 512 字节地址边界，因此目标 MRIF 地址的第 8:0 位必须为零，且不在 PTE 中指定。

字段 NPPN（通知物理页码）和两个 NID（通知标识符）字段共同指定了*通知 MSI 的*目的地和值，每次目的地 MRIF 更新后，都会发送*通知* MSI，作为查询此 PTE 以存储传入 MSI 的结果。

通常情况下，NPPN 是 IMSIC 中断文件在实际机器中的页地址，NID 是中断文件中待定的中断标识，以表明目标 MRIF 可能已发生变化。不过，NPPN 并非必须是有效的中断文件地址，IOMMU 也不得试图将其限制为有效地址。NPPN 必须接受任何页面地址。

I/O 设备对 MRIF 模式 PTE 所覆盖页面内地址的内存访问由 IOMMU 处理，而不是传递给内存系统。如果内存访问（读取或写入）的数据不是 32 位，或者访问地址未与 4 字节边界对齐（包括跨越页面边界的访问），则应将该访问作为不支持的访问而中止。对于自然对齐的 32 位读取，IOMMU 最好返回 0 作为读取值，但也可以放弃访问。自然对齐的 32 位写入要么被解释为 MSI，导致目标 MRIF 的更新，要么被丢弃。

如果系统中的 IMSIC 中断文件实现了内存映射寄存器 seteipnum_be，用于接收 big-endian 字节序的 MSI（第 3.5 节），那么 IOMMU 必须能够将小端和大端字节序的 MSI 都存储到目标 MRIF

中。如果系统中的 IMSIC 中断文件不执行寄存器 seteipnum，则 IOMMU 通常只能向目标 MRIF 存储小端序的 MSI。如果目标地址的第 2 位为 0，则假定传入的 MSI 数据为小端顺序；如果目标地址的第 2 位为 1，则假定传入的 MSI 数据为大端顺序。

如果自然对齐的 32 位写入是写入 MRIF 模式 PTE 所覆盖页面内的 guest 物理地址 *A*，并且如果写入数据按 *A* 的第 2 位所示字节顺序解释为 *D*，则写入将按以下方式处理：如果 *A*[11:3] 或 *D*[31:11] 均不为零，或者如果 *A* 的第 2 位为 1 且不支持大端 MSI，则接受写入但丢弃。否则，原始写入将被识别为 MSI，并由以下内存访问之一取代，同时将目标 MRIF 中与中断标识 *D* *相对*应的中断挂起位设置为 1：

- 如果支持原子更新，则执行原子 AMOOR 操作；或
- 如果不支持原子更新，则使用非原子读-修改-写序列。

一旦系统中的所有代理都能看到 MRIF 更新操作，11 位的 NID 值就会被归零扩展到 32 位，并以小端字节顺序写入地址 NPPN <<12（即物理页码 NPPN，页面偏移量为零）。

虽然 IOMMU 通常会缓存以基本转换模式（M = 3）配置的 MSI PTE，但它们可能不会缓存以 MRIF 模式（M = 1）配置的 PTE。不缓存 MRIF 模式下的 MSI PTE 有两个原因：首先，将 MSI 存储到 MRIF 所需的信息和操作远不同于普通的地址转换；其次，根据其性质，MSI 到 MRIF 的发生频率较低。因此，IOMMU 可能仅将 MRIF 模式处理作为 cache-miss PTW 的扩展来执行，从而使其地址转换缓存对 MRIF 模式 MSI PTE 视而不见。