Nuclei™ N100 系列

处理器内核指令架构手册

版权声明

联系我们

若您有任何疑问，请通过电子邮件 support@nucleisys.com 联系芯来科技。

修订历史

版本号	修订日期	修订的章节	修订的内容
1.0	2019/9/12	N/A	1. 初始版本
2.0	2020/1/22	N/A	1. 修正了部分文字说明

目录

版权声明 **0**

联系我们 **0**

修订历史 **1**

表格清单 **5**

图片清单 **6**

1. N100 系列内核指令集与 CSR 介绍 **7**

1.1. RISC-V 指令集介绍 7
1.2. N100 系列内核支持指令集 7
1.3. CSR 寄存器 7

2. N100 系列内核特权架构介绍 **8**

2.1. 总体介绍 8
2.2. 特权模式（PRIVILEGE MODES） 8
2.2.1. 机器模式（Machine Mode） *8*

3. N100 系列内核异常机制介绍 **9**

3.1. 异常概述 9
3.2. 异常屏蔽 9
3.3. 异常的优先级 9
3.4. 进入异常处理模式 9
3.4.1. 从"异常中断向量表"中的异常入口开始执行 *10*
3.4.2. 更新 CSR 寄存器 mcause *10*
3.4.3. 更新 CSR 寄存器 mepc *11*
3.4.4. 更新 CSR 寄存器 mstatus *11*
3.5. 退出异常处理模式 12
3.5.1. 从 mepc 定义的 PC 地址开始执行 *12*
3.5.2. 更新 CSR 寄存器 mstatus *12*
3.6. 异常服务程序 13
3.7. 异常嵌套 13

4. N100 系列内核中断机制介绍 **14**

4.1. 中断概述 14
4.2. 中断控制器 IRQC 14
4.3. 中断类型 14

4.3.1. 外部中断 *15*
4.3.2. 内部中断 *15*
4.4. 中断屏蔽 16
4.4.1. 中断全局屏蔽 *16*
4.4.2. 中断源单独屏蔽 *16*
4.5. 中断优先级与仲裁 16
4.6. 进入中断处理模式 16
4.6.1. 从新的 PC 地址开始执行 *17*
4.6.2. 更新 CSR 寄存器 mepc *17*
4.6.3. 更新 CSR 寄存器 mcause 和 mstatus *17*
4.7. 退出中断处理模式 18
4.7.1. 从 mepc 定义的 PC 地址开始执行 *18*
4.7.2. 更新 CSR 寄存器 mstatus *19*
4.8. 中断向量表 19
4.9. 进出中断的上下文保存和恢复 20
4.10. 中断响应延迟 20
4.11. 中断嵌套 21
4.12. 中断咬尾 21
4.13. 中断的向量处理模式 21
4.13.1. 向量处理模式 *21*

5. N100 系列内核 TIMER 和 IRQC 介绍 23

5.1. TIMER 介绍 23
5.1.1. TIMER 简介 *23*
5.1.2. TIMER 寄存器 *23*
5.1.3. 通过 mtime 寄存器进行计时 *24*
5.1.4. 通过 mstop 寄存器暂停计时器 *24*
5.1.5. 通过 mtime 和 mtimecmp 寄存器生成计时器中断 *24*
5.1.6. 通过 msip 寄存器生成软件中断 *24*
5.2. IRQC 介绍 25
5.2.1. IRQC 简介 *25*
5.2.2. IRQC 中断目标 *27*
5.2.3. IRQC 中断源 *27*
5.2.4. IRQC 中断源的编号（ID） *28*
5.2.5. IRQC 的寄存器 *28*
5.2.6. IRQC 中断源的使能位（IE） *31*
5.2.7. IRQC 中断源的等待标志位（IP） *32*
5.2.8. IRQC 中断源的电平触发（Level-Triggered） *32*

5.2.9. IRQC 中断源的边沿触发（Edge-Triggered） *32*
5.2.10. IRQC 中断源优先级（Priority） *33*
5.2.11. IRQC 中断源的向量处理（Vector Mode） *33*
5.2.12. IRQC 中断的仲裁机制 *34*
6. N100 系列内核 CSR 寄存器介绍 **35**
6.1. N100 系列内核 CSR 寄存器概述 35
6.2. N100 系列内核的 CSR 寄存器列表 35
6.3. N100 系列内核的 CSR 寄存器的访问权限 36
6.4. N100 系列内核支持的 RISC-V 标准 CSR 36
6.4.1. misa *36*
6.4.2. mvendorid *38*
6.4.3. marchid *38*
6.4.4. mimpid *38*
6.4.5. mhartid *38*
6.4.6. mstatus *38*
6.4.7. mtvec *40*
6.4.8. mepc *40*
6.4.9. mcause *41*
6.4.10. mtvt *41*
6.4.11. mcycle 和 mcycleh *42*
6.4.12. minstret 和 minstreth *42*
6.5. N100 系列内核自定义的 CSR 43
6.5.1. mcountinhibit *43*
6.5.2. mtime、mtimecmp、msip 和 mstop *43*
6.5.3. irqcip、irqcie、irqclvl、irqcedge 和 irqcedge *44*
6.5.4. sleepvalue *44*
6.5.5. txevt *45*
6.5.6. wfe *45*
7. N100 系列内核低功耗机制介绍 **47**
7.1. 进入休眠状态 47
7.2. 退出休眠状态 48
7.2.1. 中断唤醒 *48*
7.2.2. Event 唤醒 *48*
7.2.3. Debug 唤醒 *49*
7.3. WAIT FOR INTERRUPT 机制 49
7.4. WAIT FOR EVENT 机制 49

表格清单

表 3-1 MCAUSE 寄存器中的 EXCEPTION CODE 10
表 5-1 TIMER 寄存器的存储器映射地址 23
表 5-2 寄存器 MSTOP 的比特域 24
表 5-3 寄存器 MSIP 的比特域 25
表 5-4 IRQC 中断源编号和分配 28
表 5-5 IRQC 的 CSR 寄存器 28
表 5-6 寄存器 IRQCIP[I]的比特域 29
表 5-7 寄存器 IRQCIE 的比特域 29
表 5-8 寄存器 IRQCLVL 的比特域 30
表 5-9 寄存器 IRQCEDGE 的比特域 31
表 5-9 寄存器 IRQCEDGE 的比特域 31
表 6-1 N100 系列内核支持的 CSR 寄存器列表 35
表 6-2 MSTATUS 寄存器各控制位 39
表 6-3 MTVEC 寄存器各控制位 40
表 6-4 MEPC 寄存器各控制位 41
表 6-5 MCAUSE 寄存器各控制位 41
表 6-6 MTVT 对齐方式 42
表 6-7 MCOUNTINHIBIT 寄存器各控制位 43
表 6-8 SLEEPVALUE 寄存器各控制位 45
表 6-9 TXEVT 寄存器各控制位 45
表 6-10 WFE 寄存器各控制位 46

图片清单

图 4-1 中断向量表示意图 20
图 4-2 中断的向量处理模式示例 22
图 5-1 IRQC 逻辑结构示意图 26
图 5-2 IRQC 关系结构图 27
图 6-1 MISA 寄存器低 26 位各域表示的模块化指令子集 37

1. N100 系列内核指令集与 CSR 介绍

1.1. RISC-V 指令集介绍

N100 系列内核遵循的标准 RISC-V 指令集文档版本为：“指令集文档版本 2.2”（riscv-spec-v2.2.pdf）。用户可以在 RISC-V 基金会的网站上需注册便可关注并免费下载其完整原文（https://riscv.org/specifications/）。

除了 RISC-V “指令集文档版本 2.2”英文原文之外，中文用户还可以参阅中文书籍《手把手教你设计 CPU——RISC-V 处理器篇》的附录 A、附录 C~G 部分，其使用通俗易懂的中文对 RISC-V 指令集标准进行了系统讲解。

1.2. N100 系列内核支持指令集

RISC-V 指令集基于模块化设计，可以根据配置进行灵活组合。Nuclei N100 系列内核支持的是如下模块化指令集：

- RV32 架构：32 位架构通用寄存器宽度 32 位。
- E：支持 16 个通用整数寄存器。
- C：支持编码长度为 16 位的压缩指令，提高代码密度。

按照 RISC-V 架构命名规则，以上指令子集的组合可表示为 RV32EC。

1.3. CSR 寄存器

RISC-V 的架构中定义了一些控制和状态寄存器（Control and Status Register，CSR），用于配置或记录一些处理器核的运行状态。CSR 寄存器是处理器核内部的寄存器，使用其专有的 12 位地址编码空间。详情请参见第 6 章。

2. N100 系列内核特权架构介绍

2.1. 总体介绍

由于 N100 系列内核主要面向超低功耗场景的超小面积的实现，因此 N100 系列内核并没有严格遵循的标准 RISC-V 特权架构文档版本——“特权架构文档版本 1.10”（riscv-privileged-v1.10.pdf），而是对其进行了简化和删减，从而达到最小化面积和功耗的效果。

2.2. 特权模式（Privilege Modes）

N100 系列内核只支持机器模式（Machine Mode）。

2.2.1. 机器模式（Machine Mode）

N100 系列内核有关 Machine Mode 的关键要点如下：

- 处理器内核被复位后，默认处于 Machine Mode。
- 在 Machine Mode 下，程序能够访问所有的 CSR 寄存器。
- 在 Machine Mode 下，程序能够访问所有的物理地址区域。

3. N100 系列内核异常机制介绍

3.1. 异常概述

异常（Exception）机制，即处理器核在顺序执行程序指令流的过程中突然遇到了异常的事情而中止执行当前的程序，转而去处理该异常，其要点如下：

- 处理器遇到的“异常的事情”称为异常（Exception）。异常是由处理器内部事件或程序执行中的事件引起的，譬如本身硬件故障、程序故障，或者执行特殊的系统服务指令而引起的，简而言之是一种内因。
- 异常发生后，处理器会进入异常服务处理程序。

3.2. 异常屏蔽

异常是不可以被屏蔽的，也就是说一旦发生了异常，处理器一定会停止当前操作转而进入异常处理模式。

3.3. 异常的优先级

处理器内核可能存在多个异常同时发生的情形，因此异常也有优先级。异常的优先级如表 3-1 中所示，异常编号数字越小的异常优先级越高。

3.4. 进入异常处理模式

进入异常时，N100 系列内核的硬件行为可以简述如下。注意，下列硬件行为在一个时钟周期内同时完成：

- 停止执行当前程序流，转而从“异常中断向量表”中的异常入口开始执行。
- 更新相关 CSR 寄存器，分别是以下几个寄存器：
  - mcause（Machine Cause Register）

- mepc（Machine Exception Program Counter）
- mstatus（Machine Status Register）

下文将分别予以详述。

3.4.1. 从“异常中断向量表”中的异常入口开始执行

N100 系列内核遇到异常后跳入的 PC 地址由“异常中断向量表”中的异常入口指定。请参见第 6.4.7 节解“异常中断向量表”的详细介绍。

3.4.2. 更新 CSR 寄存器 mcause

N100 系列内核在进入异常时，CSR 寄存器 mcause 被同时（硬件自动）更新，以反映当前的异常种类，软件可以通过读此寄存器查询造成异常的具体原因。

mcause 寄存器的详细格式如表 6-5 所示，其中低 5 位为异常编号域，用于指示各种不同的异常类型，如表 3-1 所示。

表 3-1 mcause 寄存器中的 Exception Code

异常编号（Exception Code）	异常类型	描述
1	指令访问错误（Instruction access fault）	取指令访存错误。
2	非法指令（Illegal instruction）	非法指令。
3	断点（Breakpoint）	RISC-V 架构定义了 EBREAK 指令，当处理器执行到该指令时，会发生异常进入异常服务程序。该指令往往用于调试器（Debugger）使用，譬如设置断点
4	读存储器地址非对齐（Load address misaligned）	Load 指令访存地址非对齐。 注意：N100 系列内核不支持地址非对齐的数据存储器读写操作，因此当访问地址非对齐时会产生此异常。
5	读存储器访问错误（Load access fault）	Load 指令访存错误。

6	写存储器地址非对齐（Store address misaligned）	Store 或者 AMO 指令访存地址非对齐。注意：N100 系列内核不支持地址非对齐的数据存储器读写操作，因此当访问地址非对齐时会产生此异常。
7	写存储器访问错误（Store access fault）	Store 或者 AMO 指令访存错误。
11	机器模式环境调用（Environment call from M-mode）	Machine Mode 下执行 ecall 指令。 RISC-V 架构定义了 ecall 指令，当处理器执行到该指令时，会发生异常进入异常服务程序。该指令往往供软件使用，强行进入异常模式。

3.4.3. 更新 CSR 寄存器 mepc

N100 系列内核退出异常时的返回地址由 CSR 寄存器 mepc（Machine Exception Program Counter）保存。在进入异常时，硬件将自动更新 mepc 寄存器的值，该寄存器将作为退出异常的返回地址，在异常结束之后，能够使用它保存的 PC 值回到之前被异常停止执行的程序点。

注意：

- 出现异常时，异常返回地址 mepc 的值被更新为当前发生异常的指令 PC。
- 虽然 mepc 寄存器会在异常发生时自动被硬件更新，但是 mepc 寄存器本身也是一个可读可写的寄存器，因此软件也可以直接写该寄存器以修改其值。

3.4.4. 更新 CSR 寄存器 mstatus

mstatus 寄存器的详细格式如表 6-2 所示，N100 系列内核在进入异常时，硬件将自动更新 CSR 寄存器 mstatus（Machine Status Register）的某些域：

- mstatus.MPIE 域的值被更新为异常发生前 mstatus.MIE 域的值。mstatus.MPIE 域的作用是在异常结束之后，能够使用 mstatus.MPIE 的值恢复出异常发生之前的 mstatus.MIE 值。
- mstatus.MIE 域的值则被更新成为 0（意味着进入异常服务程序后中断被全局关闭，所有的中断都将被屏蔽不响应）。

3.5. 退出异常处理模式

当程序完成异常处理之后，最终需要从异常服务程序中退出。

由于异常处理处于 Machine Mode 下，所以退出异常时，软件必须使用 mret 指令。处理器执行 mret 指令后的硬件行为如下。注意，下列硬件行为在一个时钟周期内同时完成：

- 停止执行当前程序流，转而从 CSR 寄存器 mepc 定义的 PC 地址开始执行。
- 更新 CSR 寄存器 mstatus（Machine Status Register）。

下文将分别予以详述。

3.5.1. 从 mepc 定义的 PC 地址开始执行

在进入异常时，mepc 寄存器被同时更新，以反映当时遇到异常的指令 PC 值。通过这个机制，意味着 mret 指令执行后处理器回到了当时遇到异常的指令的 PC 地址，从而可以继续执行之前被中止的程序流。

注意：退出异常之前可能需要使用软件更新 mepc 的值。例如，如果异常由 ecall 或 ebreak 产生，由于 mepc 的值被更新为 ecall 或 ebreak 指令自己的 PC。因此在异常返回时，如果直接使用 mepc 保存的 PC 值作为返回地址，则会再次跳回 ecall 或者 ebreak 指令，从而造成死循环（执行 ecall 或者 ebreak 指令导致重新进入异常）。正确的做法是在异常处理程序中软件改变 mepc 指向下一条指令，由于现在 ecall/ebreak 都是 4 字节指令，因此改写设定 mepc=mepc+4 即可。

3.5.2. 更新 CSR 寄存器 mstatus

mstatus 寄存器的详细格式如表 6-2 所示。在执行 mret 指令后，硬件将自动更新 CSR 寄存器 mstatus 的某些域：

- mstatus.MIE 域的值被恢复为当前 mstatus.MPIE 的值。
- 当前 mstatus.MPIE 域的值则被更新为 1。

在进入异常时，mstatus.MPIE 的值曾经被更新为异常发生前的 mstatus.MIE 值。而 mret 指令执行后，将 mstatus.MIE 域的值恢复为 mstatus.MPIE 的值。通过这个机制，则意味着 mret 指令执行后，处理器的 mstatus.MIE 值被恢复成异常发生之前的值（假设之前的 mstatus.MIE 值为 1，则意味着中断被重新全局打开）。

3.6. 异常服务程序

当处理器进入异常后，即开始从 mtvec 寄存器定义的 PC 地址执行新的程序，该程序通常为异常服务程序，并且程序还可以通过查询 mcause 中的异常编号（Exception Code）决定进一步跳转到更具体的异常服务程序。譬如当程序查询 mcause 中的值为 0x2，则得知该异常是非法指令错误（Illegal Instruction）引起的，因此可以进一步跳转到非法指令错误异常服务子程序中去。

注意：由于进入异常和退出异常机制中没有硬件自动保存和恢复上下文的操作，因此需要软件明确地使用（汇编语言编写的）指令进行上下文的保存和恢复。请参见《Nuclei_N100 系列内核 SDK 使用说明》结合一个完整的异常服务程序代码示例对其进行理解。

3.7. 异常嵌套

由于 N100 系列内核主要面向超低功耗场景的超小面积的实现，因此不支持异常嵌套的复杂情形，如果发生了两级异常嵌套，属于致命错误，结果是不可预测的。

4. N100 系列内核中断机制介绍

4.1. 中断概述

中断（Interrupt）机制，即处理器内核在顺序执行程序指令流的过程中突然被别的请求打断而中止执行当前的程序，转而去处理别的事情，待其处理完了别的事情，然后重新回到之前程序中断的点继续执行之前的程序指令流。

中断的若干基本知识要点如下：

- 打断处理器执行的“别的请求”便称之为中断请求（Interrupt Request），“别的请求”的来源便称之为中断源（Interrupt Source），中断源通常来自于内核外部（称之为外部中断源），也可以来自于内核内部（成为内部中断源）。
- 处理器转而去处理的“别的事情”便称之为中断服务程序（Interrupt Service Routine，ISR）。
- 中断处理是一种正常的机制，而非一种错误情形。处理器收到中断请求之后，需要保存当前程序的现场，简称为“保存现场”。等到处理完中断服务程序后，处理器需要恢复之前的现场，从而继续执行之前被打断的程序，简称为“恢复现场”。
- 可能存在多个中断源同时向处理器发起请求的情形，需要对这些中断源进行仲裁，从而选择哪个中断源被优先处理。此种情况称为“中断仲裁”，同时可以给不同的中断分配级别和优先级以便于仲裁，因此中断存在着“中断级别”和“中断优先级”的概念。

4.2. 中断控制器 IRQC

N100 系列内核实现了一个“私有中断控制器（Interrupt Controller，简称 IRQC）”，可用于多个中断源的管理。N100 系列内核中的所有类型（除了调试中断之外）的中断都由 IRQC 统一进行管理，有关 IRQC 的详情请参见第 5.2 节。有关 N100 系列内核支持的所有中断类型的介绍请参见第 4.3 节。

4.3. 中断类型

N100 系列内核支持的中断类型包括外部中断与内部中断。下文将分别予以详述。

4.3.1. 外部中断

外部中断是指来自于处理器核外部的中断。外部中断可供用户连接外部中断源，譬如外部设备UART、GPIO 等产生的中断。

注意：N100 系列内核支持多个外部中断源，所有外部中断都由 IRQC 进行统一管理。

4.3.2. 内部中断

N100 系列内核有几种内核私有的内部中断，分别为：

- 软件中断（Software Interrupt）
- 计时器中断（Timer Interrupt）
- 存储器访问错误中断（Bus Error Interrupt）

注意：N100 系列内核的内部中断也都由 IRQC 进行统一管理。

4.3.2.1 软件中断

软件中断要点如下：

- N100 系列内核实现了一个 TIMER 单元，TIMER 单元里定义了一个 msip 寄存器，通过其可以产生软件中断，请参见第 5.1.6 节了解其详情。
- 注意：软件中断也由 IRQC 进行统一管理。

4.3.2.2 计时器中断

计时器中断要点如下：

- N100 系列内核实现了一个 TIMER 单元，TIMER 单元里定义了一个计时器，通过其可以产生计时器中断，请参见第 5.1.5 节了解其详情。
- 注意：计时器中断也由 IRQC 进行统一管理。

4.3.2.3 存储器访问错误中断

“存储器访问错误异常”转化的中断要点如下：

- 当 N100 系类处理器内核遭遇“存储器访问错误异常”时，并不会产生异常，而是会被转化成为相应的内部中断，当作一种中断来处理。
- 注意：存储器访问错误中断也由 IRQC 进行统一管理。

4.4. 中断屏蔽

4.4.1. 中断全局屏蔽

N100 系列内核的中断可以被屏蔽掉，CSR 寄存器 mstatus 的 MIE 域控制中断的全局使能。请参见第 6.4.6.1 节了解详情。

4.4.2. 中断源单独屏蔽

对于不同的中断源而言，IRQC 为每个中断源分配了各自的中断使能寄存器，用户可以通过配置 IRQC 寄存器来管理各个中断源的屏蔽，请参见第 5.2.6 节了解其详情。

4.5. 中断优先级与仲裁

当多个中断同时出现时，需要进行仲裁。对于 N100 系列内核处理器而言，IRQC 统一管理所有的中断。IRQC 的每个中断源有固定的的优先级（中断源的编号越大则优先级越高），当多个中断同时发生时，IRQC 会仲裁出优先级最高的中断。请参见第 5.2.10 节了解其详情。

4.6. 进入中断处理模式

响应中断时，N100 系列内核的硬件行为可以简述如下。注意，下列硬件行为在一个时钟周期内同时完成：

- 停止执行当前程序流，转而从新的 PC 地址开始执行。

- 进入中断不仅会让处理器跳转到上述的 PC 地址开始执行，还会让硬件同时更新其他几个 CSR 寄存器，分别是以下几个寄存器：
  - mepc（Machine Exception Program Counter）
  - mcause（Machine Cause Register）
  - mstatus（Machine Status Register）

下文将分别予以详述。

4.6.1. 从新的 PC 地址开始执行

对于 IRQC 的每个中断源而言，当该中断被处理器内核响应后，处理器直接跳入该中断的向量入口（Vector Table Entry）存储的目标地址。有关中断向量表的详细介绍，请参见第 4.8 节。

4.6.2. 更新 CSR 寄存器 mepc

N100 系列内核退出中断时的返回地址由 CSR 寄存器 mepc 指定。在进入中断时，硬件将自动更新 mepc 寄存器的值，该寄存器将作为退出中断的返回地址，在中断结束之后，能够使用它保存的 PC 值回到之前被停止执行的程序点。

注意：

- 出现中断时，中断返回地址 mepc 被指向一条指令，此指令因为中断的出现而未能完成执行。那么在退出中断后，程序便会回到之前的程序点，从 mepc 所存储的未执行完的指令开始重新执行。
- 虽然 mepc 寄存器会在中断发生时自动被硬件更新，但是 mepc 寄存器本身也是一个可读可写的寄存器，因此软件也可以直接写该寄存器以修改其值。

4.6.3. 更新 CSR 寄存器 mcause 和 mstatus

mcause 寄存器的详细格式如表 6-5 所示。N100 系列内核在进入中断时，CSR 寄存器 mcause 被同时（硬件自动）更新，详情如下：

- 当前的中断被响应后，需要有一种机制能够记录当前这个中断源的 ID 编号。
  - N100 系列内核在进入中断时，CSR 寄存器 mcause.EXCCODE 域被更新以反映当前响应的 IRQC 中断源的 ID 编号，因此软件可以通过读此寄存器查询中断源的具体 ID。并且 mcause. INTERRUPT 域被更新为 1 以反映这是中断类型。
- 当前的中断被响应后，需要有一种机制能够记录响应中断之前的中断全局使能状态和特权模式。
  - N100 系列内核在进入中断时，CSR 寄存器 mstatus.MPIE 域的值被更新为中断发生前中断的全局使能状态（mstatus.MIE 域的值）。mstatus.MIE 域的值则被更新成为 0（意味着进入中断服务程序后中断被全局关闭，所有的中断都将被屏蔽不响应）。
- 当前响应的中断如果是向量处理模式，则处理器响应中断后会直接跳入该中断的向量入口（Vector Table Entry）存储的目标地址。有关中断向量处理模式的详细介绍，请参见第 4. 13. 1 节。

4.7. 退出中断处理模式

当程序完成中断处理之后，最终需要从中断服务程序中退出，并返回主程序。由于中断处理处于 Machine Mode 下，所以退出中断时，软件必须使用 mret 指令。处理器执行 mret 指令后的硬件行为如下。注意，下列硬件行为在一个时钟周期内同时完成：

- 停止执行当前程序流，转而从 CSR 寄存器 mepc 定义的 PC 地址开始执行。
- 执行 mret 指令不仅会让处理器跳转到上述的 PC 地址开始执行，还会让硬件同时更新其他几个 CSR 寄存器，分别是以下几个寄存器：
  - mstatus（Machine Status Register）

下文将分别予以详述。

4.7.1. 从 mepc 定义的 PC 地址开始执行

在进入中断时，mepc 寄存器被同时更新，以反映当时遇到中断时的 PC 值。软件必须使用 mret 指令退出中断，执行 mret 指令后处理器将从 mepc 定义的 pc 地址重新开始执行。通过这个机制，意味着 mret 指令执行后处理器回到了当时遇到中断时的 PC 地址，从而可以继续执行之前被中止

的程序流。

4.7.2. 更新 CSR 寄存器 mstatus

执行 mret 指令后，硬件将自动更新 CSR 寄存器 mstatus 的某些域：

- 在进入中断时，mstatus.MPIE 的值曾经被更新为中断发生前的 mstatus.MIE 值。而使用 mret 指令退出中断后，硬件将 mret 指令执行后，将 mstatus.MIE 的值恢复为 mstatus.MPIE 的值。通过这个机制，则意味着退出中断后，处理器的 mstatus.MIE 值被恢复成中断发生之前的值。
- 在进入中断时，mstatus.MPP 的值曾经被更新为中断发生前的特权模式（Privilege Mode）。而使用 mret 指令退出中断后，硬件将处理器特权模式（Privilege Mode）恢复为 mstatus.MPP 的值。通过这个机制，则意味着退出中断后，处理器的特权模式（Privilege Mode）被恢复成中断发生之前的模式。

4.8. 中断向量表

如图 4-1 中所示，中断向量表是指在存储器里面开辟的一段连续的地址空间，该地址空间的每个字（Word）用于存储 IRQC 每个中断源对应的中断服务程序（Interrupt Service Routine，ISR）函数的 PC 地址。

N100 系类内核中，中断向量表的起始地址由寄存器 mtvt 指定。为了缩减面积，mtvt 被实现成了一个由宏控制的寄存器，mtvt 只能读，写会忽略。

中断向量表的作用非常重要，当处理器响应某个中断源后，硬件将通过查询中断向量表中存储的 PC 地址跳转到其对应的中断服务程序函数中去，请参见第 4.13 节了解更多详细介绍。

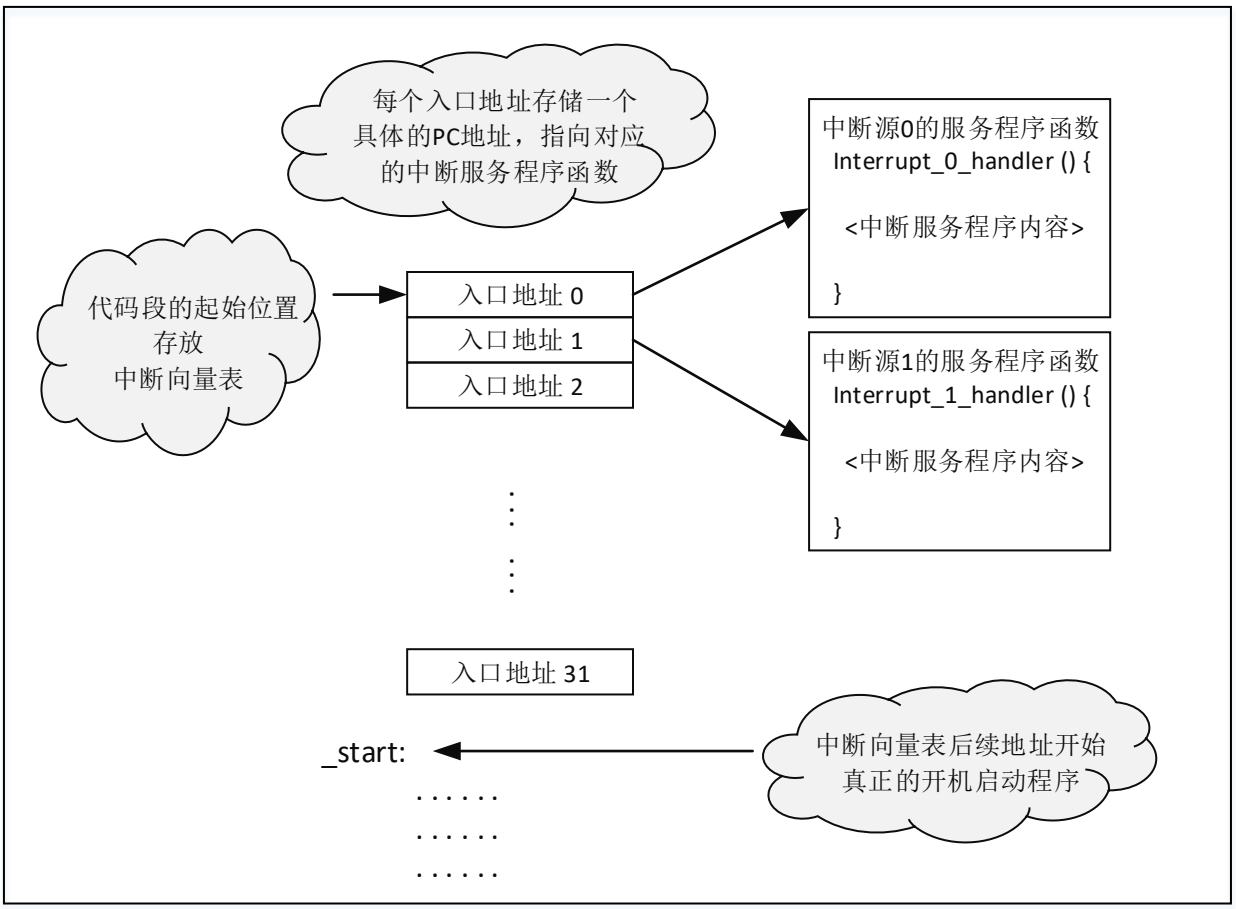

图 4-1 中断向量表示意图

4.9. 进出中断的上下文保存和恢复

RISC-V 架构的处理器在进入和退出中断处理模式时没有硬件自动保存和恢复上下文（通用寄存器）的操作，因此需要软件明确地使用（汇编语言编写的）指令进行上下文的保存和恢复，请参见第 4.13 节了解更多详细介绍。

4.10. 中断响应延迟

中断响应延迟的概念通常是指，从“外部中断源拉高”到“处理器真正开始执行该中断源对应的中断服务程序（Interrupt Service Routine，ISR）中的第一条指令”所消耗的指令周期数。因此，中断响应延迟通常会包含如下几个方面的周期开销：

- 处理器内核响应中断后进行跳转的开销

- 处理器内核保存上下文所花费的周期开销
- 处理器内核跳转到中断服务程序（Interrupt Service Routine，ISR）中去的开销。

关于中断响应延迟，请参见第 4.13 节了解更多详细介绍。

4.11. 中断嵌套

N100 系类内核主要面向超低功耗场景的超小面积的实现，为了简化设计，缩减面积，N100 系类内核不支持中断嵌套。

4.12. 中断咬尾

N100 系类内核主要面向超低功耗场景的超小面积的实现，为了简化设计，缩减面积，N100 系类内核不支持中断咬尾。

4.13. 中断的向量处理模式

N100 系类内核主要面向超低功耗场景的超小面积的实现，为了简化设计，缩减面积，N100 系类内核只支持向量处理模式中断。

4.13.1. 向量处理模式

4.13.1.1 向量处理模式的特点和延迟

如果为向量处理模式，则该中断被处理器内核响应后，处理器会直接跳入该中断的向量入口（Vector Table Entry）存储的目标地址，即该中断源的中断服务程序（Interrupt Service Routine，ISR），如图 4-2 中所示的例子。

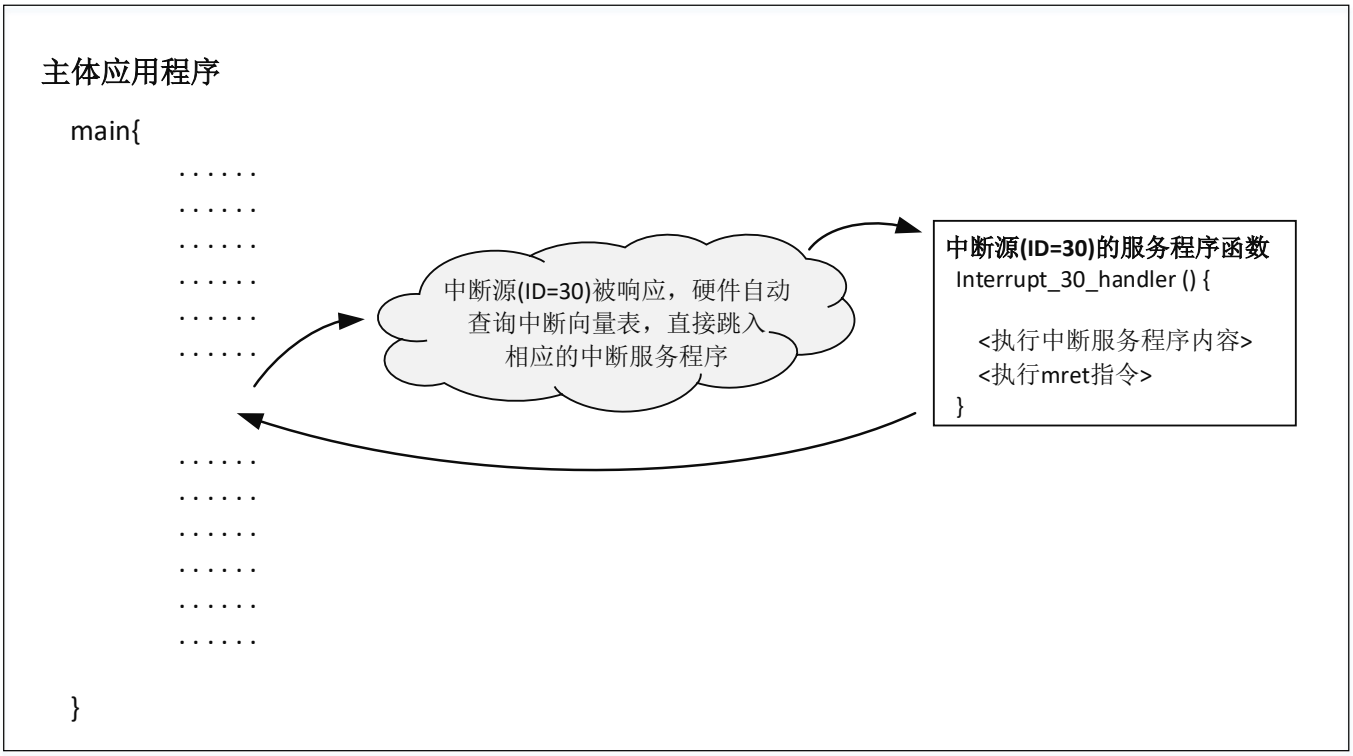

图 4-2 中断的向量处理模式示例

向量处理模式具有如下特点：

- 向量处理模式时处理器会直接跳到中断服务程序，并没有进行上下文的保存，因此，中断响应延迟非常之短，从中断源拉高到处理器开始执行中断服务程序中的第一条指令，基本上只需要硬件进行查表和跳转的时间开销，理想情况下约 6 个时钟周期。
- 对于向量处理模式的中断服务程序函数，一定要使用特殊的__attribute__((interrupt))来修饰中断服务程序函数。
- 向量处理模式时，由于在跳入中断服务程序之前，处理器并没有进行上下文的保存，因此，理论上中断服务程序函数本身不能够进行子函数的调用（即，必须是 Leaf Function）。
  - 如果中断服务程序函数不小心调用了其他的子函数（不是 Leaf Function），如果不加处理则会造成功能的错误。为了规避这种不小心造成的错误情形，只要使用了特殊的__attribute__ ((interrupt)) 来修饰该中断服务程序函数，那么编译器会自动的进行判断，当编译器发现该函数调用了其他子函数时，便会自动的插入一段代码进行上下文的保存，以保证功能的正确性。

5. N100 系列内核 TIMER 和 IRQC 介绍

5.1. TIMER 介绍

5.1.1. TIMER 简介

计时器单元（Timer Unit，TIMER），在 N100 系列内核中主要用于产生计时器中断（Timer Interrupt）和软件中断（Software Interrupt）。请参见第 4.3.2.1 节和第 4.3.2.2 节了解计时器中断与软件中断的详细信息。

5.1.2. TIMER 寄存器

TIMER 是一个 CSR 地址映射的单元：

- TIMER 单元在 N100 系列内核中的基地址请参见《Nuclei_N100 系列内核简明数据手册》中的介绍。
- TIMER 单元内寄存器和地址偏移量如表 5-1 中所示。

表 5-1 TIMER 寄存器的存储器映射地址

CSR 地址	读写属性	寄存器名称	复位默认值	功能描述
0xBD8	MRW	msip	0x00000000	生成软件中断，参见第 5.1.6 节了解其详细介绍。
0xBD9	MRW	mtimecmp	0xFFFFFFFF	配置计时器的比较值 mtimecmp，参见第 5.1.5 节了解其详细介绍。
0xBDA	MRW	mtime	0x00000000	反映计时器 mtime 的，参见第 5.1.3 节了解其详细介绍。
0xBDB	MRW	mstop	0x00000000	控制计时器的暂停，参见第 5.1.4 节了解其详细介绍。

下文对各寄存器的功能和使用进行详细描述。

5.1.3. 通过 mtime 寄存器进行计时

TIMER 可以用于实时计时，要点如下：

- TIMER 中实现了一个 32 位的 mtime 寄存器，该寄存器反映了 32 位计时器的值。计时器根据低速的输入节拍信号进行自增计数，计时器默认是打开的，因此会一直进行计数。
- 在 N100 系列内核中，此计数器的自增频率由处理器的输入信号 mtime_toggle_a 控制，请参见文档《Nuclei_N100 系列内核简明数据手册》了解该输入信号的详情。

5.1.4. 通过 mstop 寄存器暂停计时器

由于 TIMER 的计时器上电后默认会一直进行自增计数，为了在某些特殊情况下关闭此计时器计数，TIMER 中实现了一个 mstop 寄存器。如表 5-2 中所示，mstop 寄存器只有最低位为有效位，该有效位直接作为计时器的暂停控制信号，因此，软件可以通过将 mstop 寄存器设置成 1 将计时器暂停计数。

表 5-2 寄存器 mstop 的比特域

域名	比特位	属性	复位值	描述
Reserved	31:1	只读，写忽略	N/A	未使用的域，值为常数 0
TIMESTOP	0	可读可写	0	控制计时器运行或者暂停。如果该域的值为 1，则计时器暂停计数，否则正常自增计数。

5.1.5. 通过 mtime 和 mtimecmp 寄存器生成计时器中断

TIMER 可以用于生成计时器中断，要点如下：

- TIMER 中实现了一个 32 位的 mtimecmp 寄存器，该寄存器作为计时器的比较值，假设计时器的值 mtime 大于或者等于 mtimecmp 的值，则产生计时器中断。软件可以通过改写 mtimecmp 或者 mtime 的值（使得 mtimecmp 大于 mtime 的值）来清除计时器中断。

注意：计时器中断是连接到 IRQC 单元进行统一管理，有关 IRQC 的详情请参见第 5.2 节。

5.1.6. 通过 msip 寄存器生成软件中断

TIMER 可以用于生成软件中断。TIMER 中实现了一个 msip 寄存器，如表 5-3 中所示，msip 寄存器只有最低位为有效位，该有效位直接作为软件中断，因此：

- 软件写通过写 1 至 msip 寄存器产生软件中断；
- 软件可通过写 0 至 msip 寄存器来清除该软件中断。

注意：软件中断是连接到 IRQC 单元进行统一管理，有关 IRQC 的详情请参见第 5.2 节。

表 5-3 寄存器 msip 的比特域

域名	比特位	属性	复位值	描述
Reserved	31:1	只读，写忽略	N/A	未使用的域，值为常数 0
MSIP	0	可读可写	0	该域用于产生软件中断

5.2. IRQC 介绍

N100 系列内核支持私有的中断控制器（Interrupt Controller，IRQC）”，用于管理所有的中断源。

注意：

- IRQC 只服务于一个处理器内核，为该处理器内核私有。

5.2.1. IRQC 简介

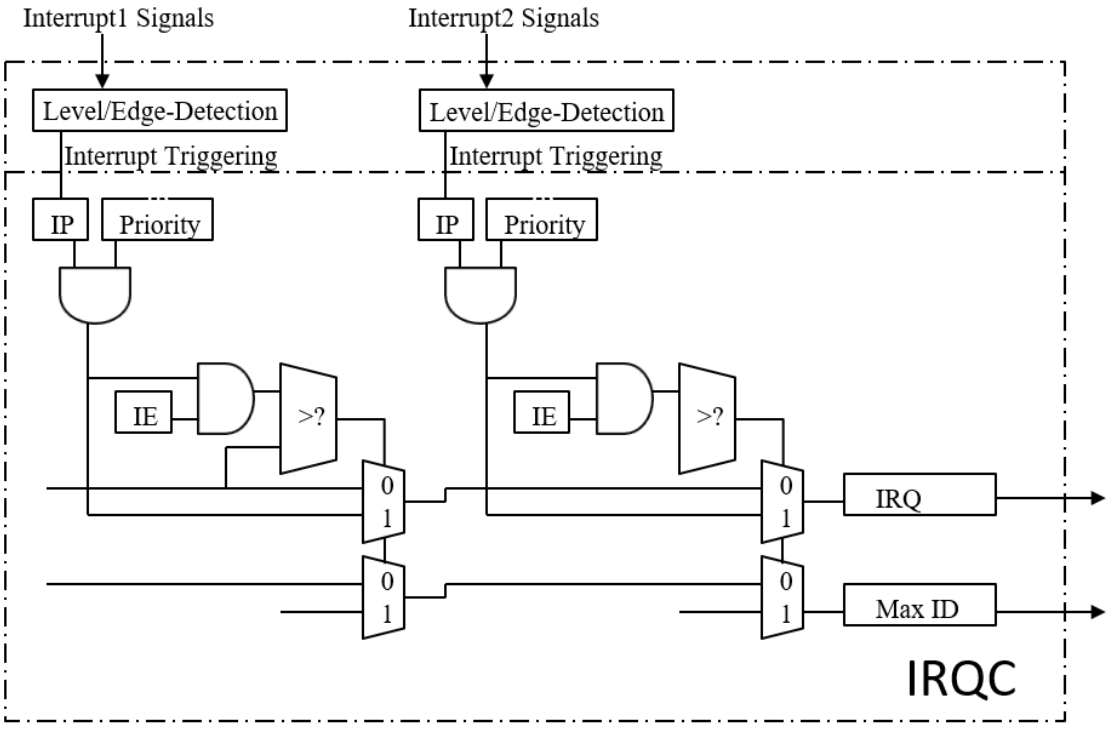

图 5-1 IRQC 逻辑结构示意图

IRQC 用于对多个内部和外部中断源进行仲裁、发送请求并支持中断嵌套。IRQC 的寄存器如表 5-5 所述，逻辑结构如图 5-1 所示，相关概念如下：

- IRQC 中断目标
- IRQC 中断源
- IRQC 中断源的编号
- IRQC 的寄存器
- IRQC 中断源的使能位
- IRQC 中断源的等待标志位
- IRQC 中断源的优先级
- IRQC 中断源的向量处理
- IRQC 中断目标的阈值级别
- IRQC 中断的仲裁机制

- IRQC 中断的响应

下文将分别予以详述。

5.2.2. IRQC 中断目标

IRQC 单元生成一根中断线，发送给处理器内核（作为中断目标），其关系结构如图 5-2 所示。

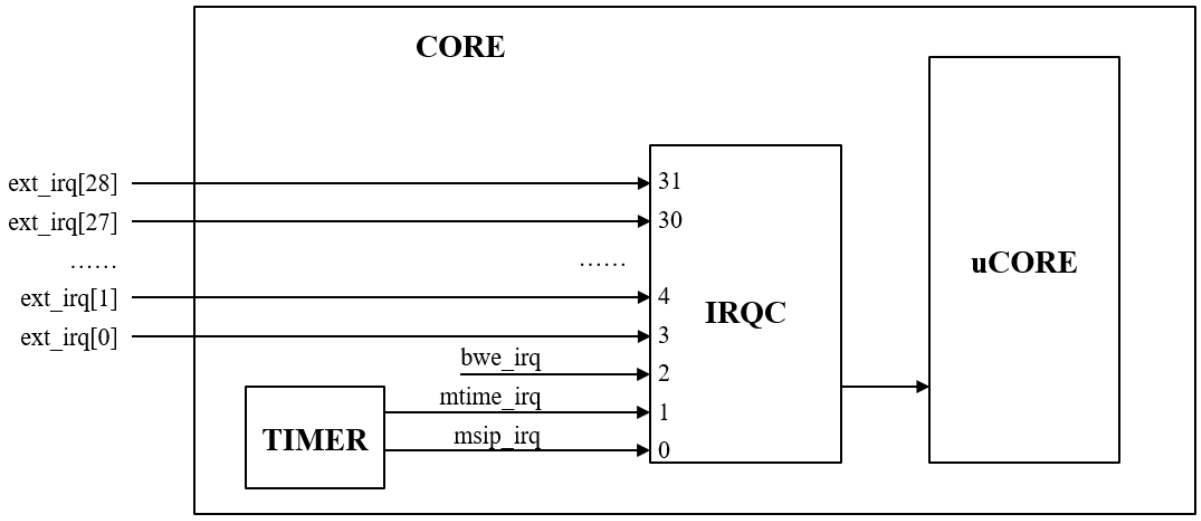

图 5-2 IRQC 关系结构图

5.2.3. IRQC 中断源

如图 5-2 所示，IRQC 理论上从编程模型上可以支持多达 32 个中断源（Interrupt Source）。IRQC 为每个中断源定义了如下特性和参数：

- 编号（ID）
- 使能位（IE）
- 等待标志位（IP）
- 电平属性（Level-Triggered）
- 边沿属性（Level Edge-Triggered）

- 优先级（Priority）
- 向量处理（Vector Mode）

下文分别予以介绍。

5.2.4. IRQC 中断源的编号（ID）

IRQC 为每个中断源分配了一个独一无二的编号（ID）。譬如，假设某 IRQC 的硬件实现真正支持 32 个 ID，则 ID 应为 0 至 31。注意：

- 在 N100 系列内核中，中断 ID 编号 0 至 2 的中断被预留作为了内核特殊的内部中断。
- 普通外部中断分配的中断源 ID 从 3 开始，用户可以用于连接外部中断源。

详细介绍如表 5-4 中所示。

表 5-4 IRQC 中断源编号和分配

IRQC 中断编号	功能	中断源介绍
0	软件中断	N100 系列内核的 TIMER 单元生成的软件中断。
1	计时器中断	N100 系列内核的 TIMER 单元生成的计时器中断。
2	存储器访问错误中断	N100 系列内核存储器访问错误转化成为内部中断。
3 ~ 31	外部中断	普通外部中断供用户连接使用。注意： ■ 虽然 IRQC 从编程模型上支持最多 32 个中断源，但是实际硬件支持的中断源数目反映在信息寄存器 irqcinfo。

5.2.5. IRQC 的寄存器

IRQC 是一个 CSR 地址映射的单元：

- IRQC 单元内的 CSR 寄存器和地址如表 5-5 中所示。

表 5-5 IRQC 的 CSR 寄存器

	属性	名称	宽度
0xBD0	MRW	irqcip	32 位

0xBD1	MRW	irqcie	32 位
0xBD2	MRW	irqclvl	32 位
0xBD3	MRW	irqcedge	32 位
0xBD4	MRW	irqcinfo	32 位

下文对各个寄存器进行详细介绍。

5.2.5.1 寄存器 irqcip

irqcip 寄存器是中断源的等待标志寄存器，其具体比特域的信息请参见表 5-6 中所示,其中 IP2-IP0 是内部中断源对应的等待标志位，IP31-IP3 是外部中断源对应的等待标志位，可参见第 5.2.7 节了解中断源等待标志位的详细介绍。

表 5-6 寄存器 irqcip[i]的比特域

域名	比特位	属性	复位值	描述
IP31	31	-	-	外部中断 28 的等待标志位，如果对应的外部中断存在，则可读可写，否则 tie 0。
...	...	...	...	...
IP4	4	-	-	外部中断 1 的等待标志位，如果对应的外部中断存在，则可读可写，否则 tie 0。
IP3	3	可读可写	0	外部中断 0 的等待标志位。
IP2	2	可读可写	0	存储器错误中断的等待标志位。
IP1	1	可读可写	0	计时器中断的等待标志位。
IP0	0	可读可写	0	软件中断的等待标志位。

5.2.5.2 寄存器 irqcie

irqcie 寄存器是中断源的使能寄存器，其具体比特域的信息请参见表 5-7 中所示，其中 IE2-IE0 是内部中断源对应的使能位，IE31-IE3 是外部中断源对应的使能位，可参见第 5.2.6 节了解中断源使能位的详细介绍。

表 5-7 寄存器 irqcie 的比特域

域名	比特位	属性	复位值	描述
IE31	31	-	-	外部中断 28 的使能位，如果对应的外部中断存在，则可读可写，否则 tie 0。
...	...	...	...	...
IE4	4	-	-	外部中断 1 的使能位，如果对应的外部中断存在，则可读可写，否则 tie 0。
IE3	3	可读可写	0	外部中断 0 的使能位。
IE2	2	可读可写	0	存储器错误中断的使能位。
IE1	1	可读可写	0	计时器中断的使能位。
IE0	0	可读可写	0	软件中断的使能位。

5.2.5.3 寄存器 irqclvl

irqclvl 寄存器是中断源的电平触发控制寄存器，其具体比特域的信息请参见表 5-8 寄存器 irqclvl 的比特域所示,其中 LVL2-LVL0 是内部中断源对应的电平触发控制位，LVL31-LVL3 是外部中断源对应的电平触发控制位，可参见第 5.2.8 节了解中断源电平触发控制位的详细介绍。

表 5-8 寄存器 irqclvl 的比特域

域名	比特位	属性	复位值	描述
LVL31	31	-	-	外部中断 28 的电平触发控制位，如果对应的外部中断存在，则可读可写，否则 tie 0。
...	...	...	...	...
LVL4	4	-	-	外部中断 1 的电平触发控制位，如果对应的外部中断存在，则可读可写，否则 tie 0。
LVL3	3	可读可写	0	外部中断 0 的电平触发控制位。
LVL2	2	可读可写	0	存储器错误中断的电平触发控制位。
LVL1	1	可读可写	0	计时器中断的电平触发控制位。
LVL0	0	可读可写	0	软件中断的电平触发控制位。

5.2.5.4 寄存器 irqcedge

irqcedge 寄存器是中断源的边沿触发控制寄存器，其具体比特域的信息请参见表 5-9 寄存器 irqcedge 的比特域中所示，其中 EDGE2-EDGE0 是内部中断源对应的边沿触发控制位，

EDGE31-EDGE3 是外部中断源对应的边沿触发控制位，可参见第 5.2.9 节了解中断源边沿触发控制位的详细介绍。

表 5-9 寄存器 irqcedge 的比特域

域名	比特位	属性	复位值	描述
EDGE31	31	-	-	外部中断 28 的边沿触发控制位，如果对应的外部中断存在，则可读可写，否则 tie 0。
...	...	...	...	...
EDGE 4	4	-	-	外部中断 1 的边沿触发控制位，如果对应的外部中断存在，则可读可写，否则 tie 0。
EDGE 3	3	可读可写	0	外部中断 0 的边沿触发控制位。
EDGE 2	2	可读可写	0	存储器错误中断的边沿触发控制位。
EDGE 1	1	可读可写	0	计时器中断的边沿触发控制位。
EDGE 0	0	可读可写	0	软件中断的边沿触发控制位。

5.2.5.5 寄存器 irqcinfo

irqcinfo 寄存器是用于软件查询支持的中断源的个数，其具体比特域的信息请参见表 5-9 寄存器 irqcedge 的比特域中所示，其中 EDGE2-EDGE0 是内部中断源对应的边沿触发控制位，EDGE31-EDGE3 是外部中断源对应的边沿触发控制位，可参见第 5.2.9 节了解中断源边沿触发控制位的详细介绍。

表 5-10 寄存器 irqcedge 的比特域

域名	比特位	属性	复位值	描述
Reserved	31:6	N/A	N/A	未使用的域为常数 0
info	5:0	只读	N/A	中断源的个数。

5.2.6. IRQC 中断源的使能位（IE）

如图 5-2 所示，IRQC 为每个中断源分配了一个中断使能位（IE）其功能如下：

- 每个中断源的中断使能位可读可写，从而使得软件可以对其编程。

- 如果 IE 被编程配置成为 0，则意味着此中断源被屏蔽。
- 如果 IE 被编程配置成为 1，则意味着此中断源被打开。

5.2.7. IRQC 中断源的等待标志位（IP）

如图 5-2 所示，IRQC 为每个中断源分配了一个中断等待标志位（IP），其功能如下：

- 如果某个中断源的 IP 位为高，则表示该中断源被触发。中断源的触发条件取决于它是电平触发还是边沿触发的属性，请参见第 5.2.8 节和第 5.2.9 节的详细介绍。
- 中断源的 IP 位软件可读可写，软件写 IP 位的行为取决于它是电平触发还是边沿触发的属性，请参见第 5.2.8 节和第 5.2.9 节的详细介绍。
- 对于边沿触发的中断源，其 IP 还可能存在硬件自清的行为，请参见第 5.2.9 节的详细介绍。

5.2.8. IRQC 中断源的电平触发（Level-Triggered）

如图 5-2 所示，IRQC 的每个中断源均可以设置为电平触发（LVL），其要点如下：

- 当 LVL == 1 时，设置该中断属性为电平触发的中断：
  - 如果该中断源被配置为电平触发，中断源的 IP 位会实时反映该中断源的电平值。
  - 如果该中断源被配置为电平触发，由于中断源的 IP 位实时反映该中断源的电平值，所以软件对该中断 IP 位的写操作会被忽略，即，软件无法通过写操作设置或者清除 IP 位的值。如果软件需要清除中断，只能通过清除中断的最终源头的方式进行。
- 当 LVL == 0 时，设置该中断属性为边沿触发的中断，请参见第 5.2.9 节的详细介绍。

5.2.9. IRQC 中断源的边沿触发（Edge-Triggered）

如图 5-2 所示，IRQC 的每个中断源均可以设置为边沿触发（EDGE），其要点如下：

- 当 LVL == 1 时，设置该中断属性为电平触发的中断，对应的 EDGE 不会生效。
- 当 LVL == 0 且 EDGE == 0 时，设置该中断属性为上升沿触发的中断：
  - 如果该中断源被配置为上升沿触发，则 IRQC 检测到该中断源的上升沿时，该中断源

在 IRQC 中被触发，该中断源的 IP 位被置高。

- 如果该中断源被配置为上升沿触发，软件对该中断 IP 位的写操作会生效，即，软件可以通过写操作设置或者清除 IP 位的值。
- 注意：对于上升沿触发的中断而言，为了能够提高中断处理的效率，当该中断被响应，处理器内核跳入中断服务程序（Interrupt Service Routines，ISR）之时，IRQC 的硬件会自动清除该中断的 IP 位，从而无需软件在 ISR 内部对该中断的 IP 位进行清除。

- 当 LVL == 0 且 EDGE == 1 时，设置该中断属性为下降沿触发的中断：
  - 如果该中断源被配置为下降沿触发，则 IRQC 检测到该中断源的下降沿时，该中断源在 IRQC 中被触发，该中断源的 IP 位被置高。
  - 如果该中断源被配置为下降沿触发，软件对该中断 IP 位的写操作会生效，即，软件可以通过写操作设置或者清除 IP 位的值。
  - 注意：对于下降沿触发的中断而言，为了能够提高中断处理的效率，当该中断被响应，处理器内核跳入中断服务程序（Interrupt Service Routines，ISR）之时，IRQC 的硬件会自动清除该中断的 IP 位，从而无需软件在 ISR 内部对该中断的 IP 位进行清除。

5.2.10. IRQC 中断源优先级（Priority）

如图 5-2 所示，IRQC 的每个中断源拥有固定的优先级，其要点如下：

- 每个中断源的中断编号（ID）即是其优先级（Priority）。
  - Priority 的数字值越大，则表示其优先级越高，注意：
    - 中断优先级（Priority）不参与中断嵌套的判断，即中断能否嵌套与中断优先级（Priority）的数值大小没有关系。
    - 多个中断同时 Pending 时，IRQC 需要仲裁决定哪个中断被发送给内核进行处理，仲裁时需要参考每个中断源的 Priority 数字值。请参见第 5.2.12 节的详细介绍。

5.2.11. IRQC 中断源的向量处理（Vector Mode）

IRQC 的每个中断源只支持向量处理模式，其要点如下：

- 该中断被处理器内核响应后，处理器直接跳入该中断的向量入口（Vector Table Entry）存储的目标地址。有关中断向量处理模式的详细介绍，请参见第 4.13 节。

5.2.12. IRQC 中断的仲裁机制

如图 5-2 所示，IRQC 对其所有中断源进行仲裁选择的原则如下：

- 只有满足下列所有条件的中断源才能参与仲裁：
  - 中断源的使能位（IE）必须为 1。
  - 中断源的等待标志位（IP）必须为 1。
- 从所有参与仲裁的中断源中进行仲裁的规则为：
  - 每个中断源的中断优先级固定，中断 ID 即是对应的中断优先级（Priority）。
  - 中断优先级（Priority）数字值越大的中断源，其仲裁优先级越高。

6. N100 系列内核 CSR 寄存器介绍

6.1. N100 系列内核 CSR 寄存器概述

RISC-V 的架构中定义了一些控制和状态寄存器（Control and Status Register， CSR）， 用于配置或者记录一些运行的状态。CSR 寄存器是处理器核内部的寄存器，使用其专有的 12 位地址编码空间。

6.2. N100 系列内核的 CSR 寄存器列表

Nuclei N100 系列支持的 CSR 寄存器列表如表 6-1 所示。

表 6-1 N100 系列内核支持的 CSR 寄存器列表

CSR 地址	读写属性	名称	全称
0xF11	MRO	mvendorid	商业供应商编号寄存器（Machine Vendor ID Register）
0xF12	MRO	marchid	架构编号寄存器（Machine Architecture ID Register）
0xF13	MRO	mimpid	硬件实现编号寄存器（Machine Implementation ID Register）
0xF14	MRO	mhartid	Hart 编号寄存器（Hart ID Register）
0x300	MRW	mstatus	异常处理状态寄存器
0x301	MRW	misa	指令集架构寄存器（Machine ISA Register）
0x305	MRW	mtvec	异常入口基地址寄存器
0x307	MRW	mtvt	异常中断向量表的基地址
0x341	MRW	mepc	异常 PC 寄存器（Machine Exception Program Counter）
0x342	MRW	mcause	异常原因寄存器（Machine Cause Register）
0xB00	MRW	mcycle	周期计数器的低 32 位（Lower 32 bits of Cycle counter）
0xB80	MRW	mcycleh	周期计数器的高 32 位（Upper 32 bits of Cycle counter）
0xB02	MRW	minstret	完成指令计数器的低 32 位（Lower 32 bits of Instructions-retired counter）
0xB82	MRW	minstreth	完成指令计数器的高 32 位（Upper 32 bits of Instructions-retired counter）
0xBD0	MRW	irqcip	中断源等待标志寄存器
0xBD1	MRW	irqcie	中断源使能控制寄存器
0xBD2	MRW	irqclvl	中断源电平触发控制寄存器

0xBD3	MRW	irqcedge	中断源边沿触发控制寄存器
0xBD4	MRW	irqcinfo	中断信息寄存器
0xBD8	MRW	msip	机器模式软件中断等待寄存器（Machine-mode Software Interrupt Pending Register）
0xBD9	MRW	mtimecmp	计时器比较寄存器（Machine-mode timer compare register）
0xBDA	MRW	mtime	计时器寄存器（Machine-mode timer register）
0xBsDB	MRW	mstop	用于停止计时器
0x811	MRW	sleepvalue	WFI 的休眠模式寄存器
0x812	MRW	txevt	发送 Event 寄存器
0x810	MRW	wfe	Wait for Event 控制寄存器

6.3. N100 系列内核的 CSR 寄存器的访问权限

Nuclei N100 系列内核对于 CSR 寄存器的访问权限规定如下：

- 如果向不存在的 CSR 寄存器地址区间进行读写操作，则会产生 Illegal Instruction Exception。
- 对 MRW 属性的 CSR 寄存器进行读写操作则一切正常。
- 对 MRO 属性的 CSR 寄存器进行读操作则一切正常。
- 如果向 MRO 的 CSR 寄存器进行写操作，则会产生 Illegal Instruction Exception。

6.4. N100 系列内核支持的 RISC-V 标准 CSR

本节介绍 N100 系列处理器核支持的 RISC-V 标准 CSR 寄存器。

6.4.1. misa

misa 寄存器用于指示当前处理器所支持的架构特性。

misa 寄存器的最高两位用于指示当前处理器所支持的架构位数：

- 如果最高两位值为 1，则表示当前为 32 位架构（RV32）。

- 如果最高两位值为 2，则表示当前为 64 位架构（RV64）。
- 如果最高两位值为 3，则表示当前为 128 位架构（RV128）。

misa 寄存器的低 26 位用于指示当前处理器所支持的 RISC-V ISA 中不同模块化指令子集，每一位表示的模块化指令子集如图 6-1 中所示。该寄存器其他未使用到的比特域为常数 0。

Bit	Character	Description
0	A	Atomic extension
1	B	*Tentatively reserved for Bit operations extension*
2	C	Compressed extension
3	D	Double-precision floating-point extension
4	E	RV32E base ISA
5	F	Single-precision floating-point extension
6	G	Additional standard extensions present
7	H	*Reserved*
8	I	RV32I/64I/128I base ISA
9	J	*Tentatively reserved for Dynamically Translated Languages extension*
10	K	*Reserved*
11	L	*Tentatively reserved for Decimal Floating-Point extension*
12	M	Integer Multiply/Divide extension
13	N	User-level interrupts supported
14	O	*Reserved*
15	P	*Tentatively reserved for Packed-SIMD extension*
16	Q	Quad-precision floating-point extension
17	R	*Reserved*
18	S	Supervisor mode implemented
19	T	*Tentatively reserved for Transactional Memory extension*
20	U	User mode implemented
21	V	*Tentatively reserved for Vector extension*
22	W	*Reserved*
23	X	Non-standard extensions present
24	Y	*Reserved*
25	Z	*Reserved*

图 6-1 misa 寄存器低 26 位各域表示的模块化指令子集

注意：misa 寄存器在 RISC-V 架构文档中被定义为可读可写的寄存器，从而允许某些处理器的设计能够动态地配置某些特性。但是在 N100 系列内核的实现中，misa 寄存器为只读寄存器，恒定地反映处理器核所支持的 ISA 模块化子集（RV32EC）。写此寄存器会会被忽略。

6.4.2. mvendorid

此寄存器是只读寄存器，用于反映该处理器核的商业供应商编号（Vendor ID）。

如果此寄存器的值为 0，则表示此寄存器未实现。

6.4.3. marchid

此寄存器是只读寄存器，用于反映该处理器核的硬件实现微架构编号（Microarchitecture ID）。

如果此寄存器的值为 0，则表示此寄存器未实现。

6.4.4. mimpid

此寄存器是只读寄存器，用于反映该处理器核的硬件实现编号（Implementation ID）。

如果此寄存器的值为 0，则表示此寄存器未实现。

6.4.5. mhartid

此寄存器是只读寄存器，用于反映当前 Hart 的编号（Hart ID）。

Hart（取“Hardware Thread”之意）表示一个硬件线程，单个处理器核中可能实现多份硬件线程，譬如硬件超线程（Hyper-threading）技术，每套线程有自己独立的寄存器组等上下文资源，但大多数的运算资源均被所有硬件线程复用，因此面积效率很高。在这样的硬件超线程处理器中，一个核内便存在着多个硬件线程（Hart）。

N100 处理器内核中 Hart 编号值受输入信号 core_mhartid 控制。注意：RISC-V 架构规定，如果在单 Hart 或者多 Hart 的系统中，起码要有一个 Hart 的编号必须是 0。

6.4.6. mstatus

mstatus 寄存器是机器模式（Machine Mode）下的状态寄存器。mstatus 寄存器中各控制位域

如表 6-2 所示。

表 6-2 mstatus 寄存器各控制位

域	位	复位值	描述
Reserved	2:0	N/A	未使用的域为常数 0
MIE	3	0	参见第 6.4.6.1 节了解其详情
Reserved	6:4	N/A	未使用的域为常数 0
MPIE	7	0	参见第 6.4.6.2 节了解其详情
Reserved	10:8	N/A	未使用的域为常数 0
Reserved	12:11	0	N100 内核中该域固定值为 b11
Reserved	31:13	0	未使用的域为常数 0

6.4.6.1 mstatus.MIE

mstatus 寄存器中的 MIE 域表示全局中断使能：

- 当 MIE 域的值为 1 时，表示中断的全局开关打开，中断能够被正常响应；
- 当 MIE 域的值为 0 时，表示全局关闭中断，中断被屏蔽，无法被响应。

注意：N100 系列处理器内核在进入异常或中断处理模式时，MIE 的值会被更新成为 0（意味着进入异常或中断处理模式后中断被屏蔽）。

6.4.6.2 mstatus.MPIE

mstatus 寄存器中的 MPIE 分别用于自动保存进入异常和中断之前 mstatus.MIE。

N100 系列处理器内核进入异常时更新 mstatus 寄存器 MPIE 域的硬件行为，请参见 3.4.4 节了解其详情。

N100 系列处理器内核退出异常时（在异常处理模式下执行 mret 指令）更新 mstatus 寄存器 MPIE 的硬件行为，请参见 3.5.2 节了解其详情。

N100 系列处理器内核进入中断时更新 mstatus 寄存器 MPIE 的硬件行为，请参见第 4.6.3 节了解其详情。

N100 系列处理器内核退出中断时（在异常处理模式下执行 mret 指令）更新 mstatus 寄存器 MPIE 域的硬件行为，请参见 4.7.2 节了解其详情。

6.4.7. mtvec

mtvec 寄存器用于配置异常中断处理程序的入口地址，在 N100 系类处理器内核中，为了缩减面积，将 mtvec 寄存器实现为了只读寄存器，通过宏控制，写 mtvec 被忽略。

- 当 mtvec 配置异常处理程序入口地址时要点如下：
  - 异常处理程序采用 4byte 对齐的 mtvec 地址（将 mtvec 的低 2bit 用 0 代替）作为入口地址。
- 当 mtvec 配置中断程序的入口地址时要点如下：
  - 必须配置 mtvec.MODE = 6'b000011，其余值均为非法值。
    - ◆ 中断处理的入口地址和要点如第 5.2.11 节中所述。

mtvec 寄存器各地址位域如表 6-3 所示。

表 6-3 mtvec 寄存器各控制位

域	位	描述
ADDR	31:6	mtvec 地址
MODE	5：0	■ MODE 域为中断处理模式控制域： ● 000011: IRCQC 中断模式（默认模式） ● Others: 不支持

6.4.8. mepc

mepc 寄存器用于保存进入异常之前处理器正在执行指令的 PC 值，作为异常的返回地址。

为了理解此寄存器，请先参见第 3 章系统地了解异常的相关信息。

注意：

- 处理器进入异常时，mepc 寄存器被同时更新以反映当前遇到异常的指令的 PC 值。

- 值得注意的是，虽然 mepc 寄存器会在异常发生时自动被硬件更新，但是 mepc 寄存器本身也是一个（在 Machine Mode 下）可读可写的寄存器，因此软件也可以直接写该寄存器以修改它的值。

mepc 寄存器各地址位域如表 6-4 所示。

表 6-4 mepc 寄存器各控制位

域	位	描述
Reserved	31：20	未使用的域为常数 0
EPC	19：1	保存异常发生前处理器正在执行的指令的 PC 值
Reserved	0	未使用的域为常数 0

6.4.9. mcause

mcause 寄存器，用于保存进入异常和中断之前的出错原因，以便于对 Trap 原因进行诊断和调试。

mcause 寄存器各地址位域如表 6-5 所示。

表 6-5 mcause 寄存器各控制位

域	位	描述
INTERRUPT	31	表示当前是异常或者中断： ■ 0：异常 ■ 1：中断
Reserved	31:12	未使用的域为常数 0
EXCCODE	11:0	异常/中断编码

6.4.10. mtvt

mtvt 寄存器用于保存中断向量表的基地址，此基地址至少为 64byte 对齐，在 N100 系类处理器内核中，为了缩减面积，将 mtvt 寄存器实现为了只读寄存器，通过宏控制，写 mtvt 被忽略。

为了提升性能减少硬件门数，硬件根据实际实现的中断的个数来决定 mtvt 的对齐方式，具体

如表 6-6 所示。

表 6-6 mtvt 对齐方式

最大中断个数	mtvt 对齐方式
0 to 16	64-byte
17 to 32	128-byte

6.4.11. mcycle 和 mcycleh

RISC-V 架构定义了一个 64 位宽的时钟周期计数器，用于反映处理器执行了多少个时钟周期。只要处理器处于执行状态时，此计数器便会不断自增计数。

mcycle 寄存器反映了该计数器低 32 位的值，mcycleh 寄存器反映了该计数器高 32 位的值。

mcycle 和 mcycleh 寄存器可以用于衡量处理器的性能，且具备可读可写属性，因此软件可以通过 CSR 指令改写 mcycle 和 mcycleh 寄存器中的值。

由于考虑到此计数器计数会消耗某些动态功耗，因此在 Nuclei N100 系列处理器的实现中，在自定义 CSR 寄存器 mcountinhibit 中额外增加了一位控制域，软件可以配置此控制域将 mcycle 和 mcycleh 对应的计数器停止计数，从而在不需要衡量性能之时停止计数器以达到省电的作用。请参见 6.5.1 节了解更多 mcountinhibit 寄存器信息。

注意：如果在调试模式下时，此计数器并不会计数，只有在正常功能模式下，计数器才会进行计数。

6.4.12. minstret 和 minstreth

RISC-V 架构定义了一个 64 位宽的指令完成计数器，用于反映处理器成功执行了多少条指令。只要处理器每成功执行完成一条指令，此计数器便会自增计数。

minstret 寄存器反映了该计数器低 32 位的值，minstreth 寄存器反映了该计数器高 32 位的值。

minstret 和 minstreth 寄存器可以用于衡量处理器的性能，且具备可读可写属性，因此软件可

以通过 CSR 指令改写 minstret 和 minstreth 寄存器中的值。

由于考虑到此计数器计数会消耗某些动态功耗，因此在 Nuclei N100 系列处理器内核的实现中，在自定义的 CSR 寄存器 mcountinhibit 中额外增加了一位控制域，软件可以配置此控制域将 minstret 和 minstreth 对应的计数器停止计数，从而在不需要衡量性能之时停止计数器以达到省电的作用。请参见 6.5.1 节了解更多 mcountinhibit 寄存器信息。

注意：如果在调试模式下时，此计数器并不会计数，只有在正常功能模式下，计数器才会进行计数。

6.5. N100 系列内核自定义的 CSR

本节介绍 N100 系列处理器核自定义的 CSR 寄存器。

6.5.1. mcountinhibit

mcountinhibit 寄存器用于控制 mcycle 和 minstret 的计数，各控制位域如表 6-7 所示。

表 6-7 mcountinhibit 寄存器各控制位

域	位	描述
Reserved	31:3	未使用的域为常数 0
IR	2	IR 为 1 时 minstret 的计数被关闭
Reserved	1	未使用的域为常数 0
CY	0	CY 为 1 时 mcycle 的计数被关闭

6.5.2. mtime、mtimecmp、msip 和 mstop

N100 系列内核定义了一个 32 位的私有计时器模块（TIMER），按照系统的低速实时时钟（Real Time Clock）频率进行计时。该计时器的值实时反映在 CSR 寄存器 mtime 中。

N100 系列内核还定义了一个 32 位的 mtimecmp 寄存器，该寄存器作为计时器的比较值，假

设计时器的值 mtime 大于或者等于 mtimecmp 的值，则产生计时器中断。

N100 系列内核还定义了一个 1 位的 msip 寄存器，用来触发软件中断。

由于考虑到计时器计数会消耗某些动态功耗，因此在 N100 系类内核的实现中，在自定义 mstop 寄存器中额外增加了一位控制域，软件可以配置此控制域将 mtime 对应的计时器停止计数，从而在不需要之时停止计时器达到省电的作用。

注意：如果在调试模式下时，此计数器并不会计数，只有在正常功能模式下，计数器才会进行计数。

6.5.3. irqcip、irqcie、irqclvl、irqcedge 和 irqcedge

N100 系列自定义了一个内核中断控制器（IRQC），用于实现中断管理。

N100 系列自定义一个 CSR 寄存器 irqcip，用于反应各个中断源的等待状态，请参见第 5.2.5.1 节了解更多详情。

N100 系列自定义一个 CSR 寄存器 irqcie，用于控制各个中断源的使能，请参见第 5.2.5.2 节了解更多详情。

N100 系列自定义一个 CSR 寄存器 irqclvl，用于控制各个中断源的电平触发，请参见第 5.2.5.3 节了解更多详情。

N100 系列自定义一个 CSR 寄存器 irqcedge，用于控制各个中断源的边沿触发，请参见第 5.2.5.4 节了解更多详情。

N100 系列自定义一个 CSR 寄存器 irqcinfo，用于软件查询支持的中断源的个数，请参见第 5.2.5.5 节了解更多详情。

6.5.4. sleepvalue

N100 系列自定义了一个 CSR 寄存器 sleepvalue 用于控制不同的休眠模式，请参见第 7.1 节了解更多详情。sleepvalue 寄存器中各控制位域如表 6-8 所示。

表 6-8 sleepvalue 寄存器各控制位

域	位	描述
SLEEPVALUE	0	控制 WFI 的休眠模式 ■ 0：浅度休眠模式（执行 WFI 后，处理器内核主工作时钟 core_clk 被关闭） ■ 1：深度休眠模式（执行 WFI 后，处理器内核主工作时钟 core_clk 和处理器内核的常开时钟 core_aon_clk 都被关闭） 此位复位默认值为 0
Reserved	31:1	未使用的域为常数 0

6.5.5. txevt

N100 系列处理器内核自定义了一个 CSR 寄存器 txevt，用于对外发送 Event。

txevt 寄存器中各控制位域如表 6-9 所示。

表 6-9 txevt 寄存器各控制位

域	位	描述
TXEVT	0	控制发送 Event： ■ 如果向此位写 1，则会触发 N100 系列处理器内核的输出信号 tx_evt 产生一个单周期脉冲信号，作为对外的 Event 信号。 ■ 该比特位为自清比特位，即，向此位写入 1 之后，下一个周期其被自清为 0。 ■ 向此位写入 0 则无任何反应和操作。 此位复位默认值为 0
Reserved	31:1	未使用的域为常数 0

6.5.6. wfe

N100 系列处理器内核自定义了一个 CSR 寄存器 wfe，用于控制 WFI 指令的唤醒条件是使用中断还是使用 Event。请参见第 7.2.2 节了解更多详情。

wfe 寄存器中各控制位域如表 6-10 所示。

表 6-10 wfe 寄存器各控制位

域	位	描述
WFE	0	控制 WFI 指令的唤醒条件是使用中断还是使用 Event。 ■ 0：处理器内核进入休眠模式时，可以被中断唤醒。 ■ 1：处理器内核进入休眠模式时，可以被 Event 唤醒。 此位复位默认值为 0。
Reserved	31:1	未使用的域为常数 0

7. N100 系列内核低功耗机制介绍

N100 系列内核可以支持休眠模式实现较低的静态功耗。

7.1. 进入休眠状态

N100 系列内核可以通过 WFI 指令进入休眠状态。当处理器执行到 WFI 指令之后，将会：

- 立即停止执行当前的指令流；
- 等待处理器内核完成任何尚未完成的滞外操作（Outstanding Transactions），譬如取指令和数据读写操作，以保证发到总线上的操作都完成；
  - 注意：如果在等待总线上的操作完成的过程中发生了存储器访问错误异常，则会进入到异常处理模式，而不会休眠。
- 当所有的滞外操作（Outstanding Transactions）都完成后，处理器会安全地进入一种空闲状态，这种空闲状态可以被称之为“休眠”状态。
- 当进入休眠模式后：
  - N100 系列内核内部的各个主要单元的时钟将会被门控关闭以节省静态功耗；
  - N100 系列内核的输出信号 core_wfi_mode 会拉高，指示此处理器核处于执行 WFI 指令之后的休眠状态；
  - N100 系列内核的输出信号 core_sleep_value 会输出 CSR 寄存器 sleepvalue 的值（注意：该信号只有在 core_wfi_mode 信号为高电平时生效；core_wfi_mode 信号为低电平时 core_sleep_value 的值一定是 0）。软件可以通过事先配置 CSR 寄存器 sleepvalue 来指示不同的休眠模式（0 或者 1）。注意：
    - 对于不同的休眠模式而言，N100 系列内核的行为完全一样。此休眠模式只是仅供 SoC 系统层面的 PMU（Power Management Unit）进行相应不同的控制。

7.2. 退出休眠状态

N100 系列内核处理器退出休眠模式的要点如下：

- N100 系列内核的输出信号 core_wfi_mode 会相应拉低。
- N100 系列内核处理器可以通过以下四种方式被唤醒：
  - 中断
  - Event
  - Debug 请求

下文将予以详细介绍 。

7.2.1. 中断唤醒

中断也可以唤醒处理器内核：

- 如果 mstatus.MIE 域被配置为 1（表示全局中断被打开），则：
  - 当 IRQC（通过将外部请求的中断进行仲裁）向处理器内核发送了中断，处理器内核被唤醒，进入到中断服务程序开始执行。
- 如果 mstatus.MIE 域被配置为 0（表示全局中断被关闭），则：
  - 如果 CSR 寄存器 wfe.WFE 域被配置为 0，则：
    - 当 IRQC（通过将外部请求的中断进行仲裁）向处理器内核发送了中断，处理器内核被唤醒，继续顺序执行之前停止的指令流（而不是进入到中断服务程序）。
  - 如果 CSR 寄存器 wfe.WFE 域被配置为 1，则等待 Event 唤醒，请参见下节描述。

7.2.2. Event 唤醒

当满足如下条件时，Event 可以唤醒处理器内核：

- 如果 mstatus.MIE 域被配置为 0（表示全局中断被关闭），且 CSR 寄存器 wfe.WFE 域被配置为 1，则：
  - 当处理器内核检测到输入信号 rx_evt（称之为 Event 信号）为高电平时，处理器内核被唤醒，继续执行之前停止的指令流（而不是进入到中断服务程序）。

7.2.3. Debug 唤醒

Debug 请求总能够唤醒处理器内核，如果调试器（Debugger）接入，也会将处理器内核唤醒而进入调试模式。

7.3. Wait for Interrupt 机制

Wait for Interrupt 机制，是指将处理器内核进入休眠模式，然后等待中断唤醒处理器内核，醒来后进入相应中断的处理函数中去。

如第 7.1 节和第 7.2 节所述，Wait for Interrupt 机制可以直接通过 WFI 指令（配合 mstatus.MIE 域被配置为 1）完成。

7.4. Wait for Event 机制

Wait for Event 机制，是指将处理器内核进入休眠模式，然后等待 Event 唤醒处理器内核，醒来后继续先前停止的程序（而不是进入中断的处理函数中去）。

如第 7.1 节和第 7.2 节所述，Wait for Event 机制可以直接通过 WFI 指令，配合如下指令序列完成：

> 第 1 步：配置 `wfe.WFE` 域为 `1`
> 第 2 步：调用 `WFI` 指令。调用此指令后处理器会进入休眠模式，当 `Event` 将其唤醒后将会继续向下执行。
> 第 3 步：恢复 `wfe.WFE` 域为 `0`