说明

青稞 V3 系列微处理器是基于标准 RISC-V 指令集架构，自研的 32 位通用 MCU 微处理器。V3 系列支持 RV32IMAC 指令集扩展，支持单周期乘法和硬件除法。除此之外，还支持硬件压栈（HPE）、免表中断（VTF）、精简的两线调试接口、支持“WFE”指令等特色功能。

特点

特点	描述
指令集架构	RV32IMAC
流水线	3 级
FPU	不支持
分支预测	静态分支预测
中断	支持异常在内的共 256 个中断，支持免表中断
硬件压栈	最大支持 2 级硬件压栈
物理内存保护	不支持
低功耗模式	支持睡眠和深度睡眠模式，并支持 WFI 和 WFE 睡眠方式
扩展指令集	不支持
调试	增强两线调试接口，标准 RISC-V 调试

第 1 章 概览

青稞 V3 微处理器支持 RV32IMAC 指令集扩展，其主要特点如下表 1-1 所示。

表 1-1 青稞 V3 微处理器概览

特点 / 型号	指令集	硬件堆栈级数	中断嵌套级数	免表中断通道数	流水线	向量表模式	扩展指令（XW）	内存保护区域个数
V3A	RV32IMAC	2	2	4	3	指令	×	×

注：操作系统的任务切换一般使用软件堆栈，软件堆栈不限级数

1.1 指令集

青稞 V3 系列微处理器遵循标准的 RISC-V 指令集架构，详细的标准文档可参考 RISC-V 基金会官网的指令集手册。RISC-V 指令集架构简洁，支持模块化设计，可以根据不同的需求进行灵活的组合，青稞 V3 系列微处理器支持如下指令集扩展：

- RV32：32 位架构，通用寄存器位宽为 32 位
- I：支持整形操作，具有 32 个整形寄存器
- M：支持整形乘法和除法指令
- A：支持原子指令
- C：支持 16 位压缩指令

1.2 寄存器组

RV32I 拥有 x0-x31 共 32 个寄存器组，V3 系列不支持“F”扩展，即没有浮点寄存器组。在 RV32 中，每个寄存器均为 32 位。下表 1-2 列举了 RV32I 的寄存器及其描述。

表 1-2 RISC-V 寄存器

寄存器	ABI 名称	描述	存储者
x0	zero	硬编码 0	-
x1	ra	返回地址	Caller
x2	sp	栈指针	Callee
x3	gp	全局指针	-
x4	tp	线程指针	-
x5-7	t0-2	临时寄存器	Caller
x8	s0/fp	保存寄存器/帧指针	Callee
x9	s1	保存寄存器	Callee
x10-11	a0-1	函数参数/返回值	Caller
x12-17	a2-7	函数参数	Caller
x18-27	s2-11	保存寄存器	Callee
x28-31	t3-6	临时寄存器	Caller

上表中 Caller 属性意为被调过程不保存该寄存器值，Callee 属性意为被调过程保存该寄存器。

1.3 特权模式

标准 RISC-V 架构包括三种特权模式：机器模式、监督模式、用户模式，如下表 1-3 所示。其中机器模式为必须实现的模式，其他的模式为可选择实现的模式。详细的可以参考 RISC-V 特权架构标准文档，用户可在 RISC-V 基金会官网免费下载。

表 1-3 RISC-V 架构特权模式

编码	名称	简称
0b00	用户模式	U
0b01	监督模式	S
0b10	保留	保留
0b11	机器模式	M

青稞 V3 系列微处理器支持其中两种特权模式：

- 机器模式

  机器模式具有最高的权限，该模式下程序可以访问所有的控制和状态寄存器（Control and Status Register,CSR），同时也能够访问所有的物理地址区域。上电默认处于机器模式下，当执行 mret（机器模式返回指令）返回后，根据 CSR 寄存器 mstatus（机器模式状态寄存器）中 MPP 位，若 MPP=0b00，则退出机器模式进入用户模式，MPP=0b11，则继续保留机器模式。

- 用户模式

  用户模式具有最低权限，该模式下只能访问限定的 CSR 寄存器。当发生异常或中断时，微处理器会由用户模式进入机器模式，处理异常和中断。

1.4 CSR 寄存器

RISC-V 架构中定义了一系列 CSR 寄存器，用于控制和记录微处理器的运行状态。这些 CSR 使用内部专用的 12 位地址编码空间，可扩展 4096 个寄存器。并且使用高两位 CSR[11:10]来定义该寄存器的读写权限，0b00、0b01、0b10 表示允许读写，0b11 表示仅读。使用 CSR[9:8]两位来定义可以访问该寄存器的最低特权级别，该值和表 1-3 中定义的特权模式对应。青稞 V3 微处理器实现的 CSR 寄存器详见第 7 章。

第 2 章 异常

异常（Exception）机制，即拦截和处理“不寻常运行事件”的机制。青稞 V3 系列微处理器搭载一个异常响应系统，能够处理中断在内的多达 256 个异常。当发生异常或中断时，微处理器能够快速响应并处理异常和中断事件。

2.1 异常类型

不管发生异常或是中断，微处理器的硬件行为是一致的，微处理器暂停当前程序，转向异常或中断处理程序，处理完成后返回之前暂停的程序。广义的来说，中断也是异常的一部分。具体当前发生的是中断还是异常可以通过机器模式异常原因寄存器 mcause 来查看。其中 mcause[31]为 interrupt 域，用于表示产生异常的原因是中断还是异常，mcause[31]=1 表示为中断，mcause[31]=0 表示为异常。mcause[30:0]为异常编码，用于表示异常产生的具体原因或中断编号，如下表所示。

表 2-1 V3 微处理器异常编码

interrupt	异常编码	同步/异步	异常原因
1	0-1	-	保留
1	2	精确异步	NMI 中断
1	3-11	-	保留
1	12	精确异步	SysTick 中断
1	13	-	保留
1	14	同步	软件中断
1	15	-	保留
1	16-255	精确异步	外部中断 16-255
0	0	同步	指令地址不对齐
0	1	同步	取指令访问错误
0	2	同步	非法指令
0	3	同步	断点
0	4	同步	Load 指令访存地址不对齐
0	5	非精确异步	Load 指令访存错误
0	6	同步	Store/AMO 指令访存地址不对齐
0	7	非精确异步	Store/AMO 指令访存错误
0	8	同步	用户模式下环境调用
0	11	同步	机器模式下环境调用

表中“同步”是指可以准确地定位到某一条执行的指令，例如一条 ebreak 或 ecall 指令，每次执行到该指令均会触发进入异常。“异步”是指不能准确定位于某条指令，每次异常发生时的指令 PC 值有可能不同。其中的“精确异步”是指异常发生后可以精确地定位到某一条指令的边界，即某条指令执行之后的状态，例如外部的中断。而“非精确异步”是指无法精确定位到某一条指令的边界，有可能是某条指令执行了一半被打断后的状态，例如访存错误。访问存储器需要时间，微处理器访存时一般不会一直等待访存结束，而是继续执行指令，此时再出现访存错误异常时，微处理器已经执行了后续的指令，无法精确定位。

2.2 进入异常

当程序在正常运行过程中，若因某种原因，触发进入异常或者中断。此时微处理器的硬件行为可以概括如下：

（1）暂停当前程序流，转向执行异常或中断处理函数。

异常或中断函数的入口基地址及寻址方式由异常入口基地址寄存器 mtvec 定义，mtvec[31:2]定义了异常或中断函数的基地址。mtvec[1:0]定义了处理函数的寻址方式，当 mtvec[1:0]=0，所有异常和中断使用统一入口，即发生异常或中断时，转向 mtvec[31:2]定义的基地址处执行。具体属于哪

种类型或某个中断，需要通过 mcause 寄存器查询，并且分别处理；当 mtvec[1:0]=1，异常和中断使用向量表模式，即对每个异常和中断进行编号，根据中断编号*4 进行地址偏移，发生异常或中断时，转向 mtvec[31:2]定义的基地址+中断编号*4 处执行。中断向量表处存放是一条跳转至中断处理函数的指令，也可以是其他指令。

（2）更新 CSR 寄存器

当进入异常或中断时，微处理器会自动更新相关的 CSR 寄存器，包括机器模式异常原因寄存器 mcause、机器模式异常指针寄存器 mepc、机器模式异常值寄存器 mtval、机器模式状态寄存器 mstatus。

- 更新 mcause

如前所述，进入异常或中断后，其值反映当前的异常种类或中断的编号，软件可以读取该寄存器值查看引起异常的原因或判断中断的来源，详见表 2-1。

- 更新 mepc

标准定义退出异常或中断后微处理器的返回地址保存在 mepc 中。所以当发生异常或中断后，硬件自动更新 mepc 值为当前遇到异常时的指令 PC 值，或中断前下一条预执行的指令 PC 值。异常或中断处理结束后，微处理器使用其保存的值作为返回地址，回到中断的位置继续执行。

但值得注意的是：

1. mepc 是一个可读可写的寄存器，软件也可以修改该值，达到修改返回后运行的 PC 指针位置的目的。

2. 当发生中断时，即异常原因寄存器 mcause[31]=1 时，mepc 的值更新为中断时下一条未执行的指令 PC 值。

而发生异常时，异常原因寄存器 mcause[31]=0 时，mepc 的值更新为当前异常的指令 PC 值。因此这时异常返回时，如果直接使用 mepc 的值返回，还是继续执行之前产生异常的指令，此时还会继续进异常。通常我们处理好异常后，可以修改 mepc 的值为下一条未执行指令的值后再返回。例如我们因 ecall/ebreak 引起异常，处理异常后，由于 ecall/ebreak（c.ebreak 为 2 字节）为 4 字节指令，此时只需要软件修改 mepc 的值为 mepc+4(c.ebreak 为 mepc+2)后返回即可。

- 更新 mtval

进入异常和中断时，硬件将自动更新 mtval 的值，该值即为引起异常的值。该值一般是：

1. 如果存储器访问引起的异常，硬件会将异常时存储器访问的地址存入 mtval。
2. 如果是非法指令引起的异常，硬件会将该指令的指令编码存入 mtval。
3. 如果是硬件断点引起的异常，硬件会将断点处 PC 值存入 mtval。
4. 对于其他的异常，硬件将 mtval 的值设为 0，例如 ebreak，ecall 指令引起的异常。
5. 进入中断时，硬件将 mtval 的值设为 0。

- 更新 mstatus

进入异常和中断时，硬件会更新 mstatus 中的某些位：

1. MPIE 更新为进入异常和中断前的 MIE 值，异常和中断结束后，MPIE 用于恢复 MIE。
2. MPP 更新为进入异常和中断前的特权模式，异常和中断结束后，MPP 用于恢复之前的特权模式。
3. 青稞 V3 微处理器支持机器模式下的中断嵌套，进入异常和中断后，MIE 不会被清零。

（3）更新微处理器特权模式

发生异常和中断时，微处理器的特权模式被更新为机器模式。

2.3 异常处理函数

进入异常或中断后，微处理器从 mtvec 寄存器定义的地址和模式执行程序。当使用统一入口时，微处理器从 mtvec[31:2]定义的基地址处取一条跳转指令，转而去执行。此时异常和中断处理函数中可根据 mcause[31]的值判断引起的是异常或者中断，由异常编码判断异常的类型和原因或者对应的中断，进行相应的处理。

当使用基地址+中断编号*4 进行偏移的方式时，硬件自动根据中断编号跳转至向量表获取异常或者中断函数的入口地址，并跳转执行。

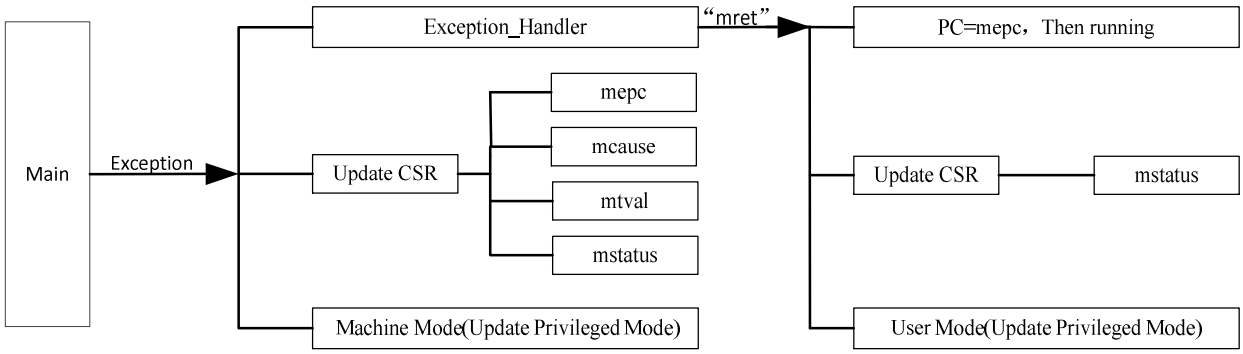
2.4 异常退出

异常或中断处理程序完成之后，需要从服务程序中退出。进入异常和中断后，微处理器由用户模式进入机器模式，异常和中断的处理也在机器模式下完成，当需要退出异常和中断时，需要使用 mret 指令进行返回。此时，微处理器硬件将自动执行如下操作：

- PC 指针恢复为 CSR 寄存器 mepc 的值，即从 mepc 保存的指令地址处开始执行。需要注意异常处理完成后对 mepc 的偏移操作。
- 更新 CSR 寄存器 mstatus，MIE 恢复为 MPIE，MPP 用于恢复之前的微处理器的特权模式。

整个异常的响应过程可由下图 2-1 描述：

图 2-1 异常响应过程示意图



第 3 章 PFIC 与中断控制

青稞 V3 微处理器内部设计了一个可编程快速中断控制器（Programmable Fast Interrupt Controller,PFIC）可管理异常在内的最多 256 个中断。其中前 16 个固定为微处理器内部中断，其余为外部中断，即最大支持扩展 240 个外部中断。其主要特征如下：

- 240个外部中断，每个中断请求都有独立的触发和屏蔽控制位，有专用的状态位
- 可编程多级中断嵌套，最大嵌套深度8级
- 特有快速中断进出机制，硬件自动压栈和恢复，最大硬件压栈深度3级，无需指令开销
- 特有免表（Vector Table Free,VTF）中断响应机制，4路可编程直达中断向量地址

注：不同型号的微处理器，中断控制器支持的最大嵌套深度和硬件压栈深度不同，具体可参考表 1-1；

中断和异常的向量表如下表 3-1 所示。

表 3-1 异常和中断向量表

编号	优先级	类型	名称	描述
0	-	-	-	-
1	-	-	-	-
2	-2	固定	NMI	不可屏蔽中断
3	-1	固定	EXC	异常中断
4-11	-	-	-	-
12	0	可编程	SysTick	系统定时器中断
13	-	-	-	-
14	1	可编程	SWI	软件中断
15	-	-	-	-
16-255	2-241	可编程	External Interrupt	外部中断 16-255

3.1 PFIC 寄存器组

表 3-2 PFIC 寄存器

名称	访问地址	访问	描述	复位值
PFIC_ISRx	0xE000E000 -0xE000E01C	RO	中断使能状态寄存器 x	0x00000000
PFIC_IPRx	0xE000E020 -0xE000E03C	RO	中断挂起状态寄存器 x	0x00000000
PFIC_ITHRESDR	0xE000E040	RW	中断优先级阈值配置寄存器	0x00000000
PFIC_VTFBADDRR	0xE000E044	RW	免表中断基地址寄存器	0x00000000
PFIC_CFGR	0xE000E048	RW	中断配置寄存器	0x00000000
PFIC_GISR	0xE000E04C	RO	中断全局状态寄存器	0x00000000
PFIC_VTFADDRRx	0xE000E060 -0xE000E06C	RW	VTF 中断 x 偏移地址寄存器	0x00000000
PFIC_IENRx	0xE000E100 -0xE000E11C	WO	中断使能设置寄存器 x	0x00000000
PFIC_IRERx	0xE000E180 -0xE000E19C	WO	中断使能清除寄存器 x	0x00000000
PFIC_IPSRx	0xE000E200 -0xE000E21C	WO	中断挂起设置寄存器 x	0x00000000

PFIC_IPRRx	0xE000E280 -0xE000E29C	WO	中断挂起清除寄存器 x	0x00000000
PFIC_IACTRx	0xE000E300 -0xE000E31C	RO	中断激活状态寄存器 x	0x00000000
PFIC_IPRIORx	0xE000E400 -0xE000E43C	RW	中断优先级配置寄存器	0x00000000
PFIC_SCTLR	0xE000ED10	RW	系统控制寄存器	0x00000000

注：1.PFIC_ISR0 寄存器的默认值为 0xC，即 NMI 和异常总是默认使能的。

2.NMI、EXC 支持中断挂起清除和设置操作，不支持中断使能清除和设置操作。

各寄存器描述如下：

中断使能状态和中断挂起状态寄存器（PFIC_ISR<0-7>/PFIC_IPR<0-7>）

名称	访问地址	访问	描述	复位值
PFIC_ISR0	0xE000E000	RO	中断 0-31 使能状态寄存器，共 32 个状态位[n]，表示#n 中断使能状态 *注：NMI 和 EXC 默认使能*	0x0000000C
PFIC_ISR1	0xE000E004	RO	中断 32-63 使能状态寄存器，共 32 个状态位	0x00000000
…	…	…	…	…
PFIC_ISR7	0xE000E01C	RO	中断 224-255 使能状态寄存器，共 32 个状态位	0x00000000
PFIC_IPR0	0xE000E020	RO	中断 0-31 挂起状态寄存器，共 32 个状态位[n]，表示#n 中断的挂起状态	0x00000000
PFIC_IPR1	0xE000E024	RO	中断 32-63 挂起状态寄存器，共 32 个状态位	0x00000000
…	…	…	…	…
PFIC_IPR7	0xE000E03C	RO	中断 244-255 挂起状态寄存器，共 32 个状态位	0x00000000

两组寄存器用于指示中断的使能状态和中断的挂起状态

中断使能设置和清除寄存器（PFIC_IENR<0-7>/PFIC_IRER<0-7>）

名称	访问地址	访问	描述	复位值
PFIC_IENR0	0xE000E100	WO	中断 0-31 使能设置寄存器，共 32 个设置位[n]，用于中断#n 使能设置 *注：NMI 和 EXC 默认使能*	0x00000000
PFIC_IENR1	0xE000E104	WO	中断 32-63 使能设置寄存器，共 32 个设置位	0x00000000
…	…	…	…	…
PFIC_IENR7	0xE000E11C	WO	中断 224-255 使能设置寄存器，共 32 个设置位	0x00000000
–	–	–	–	–
PFIC_IRER0	0xE000E180	WO	中断 0-31 使能清除寄存器，	0x00000000

			共 32 个清除位[n]，用于中断#n 使能清除 *注：NMI 和 EXC 无法操作*	
PFIC_IRER1	0xE000E184	WO	中断 32-63 使能清除寄存器，共 32 个清除位	0x00000000
…	…	…	…	…
PFIC_IRER7	0xE000E19C	WO	中断 244-255 使能清除寄存器，共 32 个清除位	0x00000000

两组寄存器用于使能和除能相应的中断。

中断挂起设置和清除寄存器（PFIC_IPSR<0-7>/PFIC_IPRR<0-7>）

名称	访问地址	访问	描述	复位值
PFIC_IPSR0	0xE000E200	WO	中断 0-31 挂起设置寄存器，共 32 个设置位[n]，用于中断#n 挂起设置	0x00000000
PFIC_IPSR1	0xE000E204	WO	中断 32-63 挂起设置寄存器，共 32 个设置位	0x00000000
…	…	…	…	…
PFIC_IPSR7	0xE000E21C	WO	中断 224-255 挂起设置寄存器，共 32 个设置位	0x00000000
–	–	–	–	–
PFIC_IPRR0	0xE000E280	WO	中断 0-31 挂起清除寄存器，共 32 个清除位[n]，用于中断#n 挂起清除	0x00000000
PFIC_IPRR1	0xE000E284	WO	中断 32-63 挂起清除寄存器，共 32 个清除位	0x00000000
…	…	…	…	…
PFIC_IPRR7	0xE000E29C	WO	中断 244-255 挂起清除寄存器，共 32 个清除位	0x00000000

当微处理器使能某个中断时，可以通过中断挂起寄存器直接设置，触发进入中断。使用中断挂起清除寄存器，清除挂起触发。

中断激活状态寄存器（PFIC_IACTR<0-7>）

名称	访问地址	访问	描述	复位值
PFIC_IACTR0	0xE000E300	RO	中断 0-31 激活状态寄存器，共 32 个状态位[n]，表示中断#n 正在执行	0x00000000
PFIC_IACTR1	0xE000E304	RO	中断 32-63 激活状态寄存器，共 32 个状态位	0x00000000
…	…	…	…	…
PFIC_IACTR7	0xE000E31C	RO	中断 224-255 激活状态寄存器，共 32 个状态位	0x00000000

每个中断都有一个活动状态位，当进入中断时，该位被置起，当 mret 返回后，该位被硬件清除。

中断优先级和优先级阈值寄存器（PFIC_IPRIOR<0-7>/PFIC_ITHRESDR）

名称	访问地址	访问	描述	复位值
PFIC_IPRIOR0	0xE000E400	RW	中断 0 优先级配置： [7:4]：优先级控制位 若配置无嵌套，无抢占位 若配置嵌套，bit7 为抢占位 [3:0]：保留，固定为 0 *注：优先级数值越小，优先级约高，同一抢占优先级中断若同时挂起，优先执行优先级高的中断。*	0x00
PFIC_IPRIOR1	0xE000E401	RW	中断 1 优先级设置，功能同 PFIC_IPRIOR0	0x00
PFIC_IPRIOR2	0xE000E402	RW	中断 2 优先级设置，功能同 PFIC_IPRIOR0	
…	…	…	…	…
PFIC_IPRIOR254	0xE000E4FE	RW	中断 254 优先级设置，功能同 PFIC_IPRIOR0	0x00
PFIC_IPRIOR255	0xE000E4FF	RW	中断 255 优先级设置，功能同 PFIC_IPRIOR0	0x00
–	–	–	–	–
PFIC_ITHRESDR	0xE000E040	RW	中断优先级阈值设置 [31:8]：保留，固定为 0 [7:4]：优先级阈值 [3:0]：保留，固定为 0 *注：优先级值≥阈值的中断，当发生挂起时不执行中断服务函数，该寄存器为 0 时，表示阈值寄存器无效。*	0x00

中断配置寄存器（PFIC_CFGR）

名称	访问地址	访问	描述	复位值
PFIC_CFGR	0xE000E048	RW	中断配置寄存器	0x00000000

其各位定义为：

位	名称	访问	描述	复位值
[31:16]	KEYCODE	WO	对应不同的目标控制位，需要同步写入相应的安全访问标识数据才能修改，读出数据固定为 0。 KEY1 = 0xFA05； KEY2 = 0xBCAF； KEY3 = 0xBEEF。	0
[15:8]	Reserved	RO	保留。	0
7	SYSRESET	WO	系统复位（同步写入 KEY3）。自动清 0。 写 1 有效，写 0 无效。	0

6	PFICRESET	WO	PFIC 模块复位，自动清 0。 写 1 有效，写 0 无效。	0
5	EXCRESET	WO	异常中断挂起清除（同步写入 KEY2）。 写 1 有效，写 0 无效。	0
4	EXCSET	WO	异常中断挂起设置（同步写入 KEY2）。 写 1 有效，写 0 无效。	0
3	NMIRESET	WO	NMI 中断挂起清除（同步写入 KEY2）。 写 1 有效，写 0 无效。	0
2	NMISET	WO	NMI 中断挂起设置（同步写入 KEY2）。 写 1 有效，写 0 无效。	0
1	NESTCTRL	RW	中断嵌套使能控制： 1：关闭；0：打开（同步写入 KEY1）。	0
0	HWSTKCTRL	RW	硬件压栈使能控制： 1：关闭；0：打开（同步写入 KEY1）。	0

中断全局状态寄存器（PFIC_GISR）

名称	访问地址	访问	描述	复位值
PFIC_CFGR	0xE000E04C	RO	中断全局状态寄存器	0x00000000

其各位定义为：

位	名称	访问	描述	复位值
[31:10]	Reserved	RO	保留。	0
9	GPENDSTA	RO	当前是否有中断处于挂起： 1：有；　　0：没有。	0
8	GACTSTA	RO	当前是否有中断被执行： 1：有；　　0：没有。	0
[7:0]	NESTSTA	RO	当前中断嵌套状态： 0x03：第 2 级中断中； 0x01：第 1 级中断中； 0x00：没有中断发生； 其他：不可能情况。	0

免表中断基址和偏移地址寄存器（PFIC_VTFBADDRR /PFIC_VTFADDRR<0-3>）

名称	访问地址	访问	描述	复位值
PFIC_VTFBADDRR	0xE000E044	RW	[31:28]：免表中断的目标地址的高 4 位。 [27:0]：保留。	0x00000000
–	–	–	–	–
PFIC_VTFADDRR0	0xE000E060	RW	[31:24]：免表中断 0 中断编号。 [23:0]：免表中断目标地址低 24 位，其中低 20 位配置有效，[23:20]固定为 0。	0x00000000
PFIC_VTFADDRR1	0xE000E064	RW	[31:24]：免表中断 1 中断编号。	0x00000000

			[23:0]：免表中断目标地址低 24 位，其中低 20 位配置有效，[23:20]固定为 0。	
PFIC_VTFADDRR2	0xE000E068	RW	[31:24]：免表中断 2 中断编号。 [23:0]：免表中断目标地址低 24 位，其中低 20 位配置有效，[23:20]固定为 0。	0x00000000
PFIC_VTFADDRR3	0xE000E06C	RW	[31:24]：免表中断 3 中断编号。 [23:0]：免表中断目标地址低 24 位，其中低 20 位配置有效，[23:20]固定为 0。	0x00000000

系统控制寄存器（PFIC_SCTLR）

名称	访问地址	访问	描述	复位值
PFIC_SCTLR	0xE000ED10	RW	系统控制寄存器	0x00000000

其各位定义如下：

位	名称	访问	描述	复位值
[31:6]	Reserved	RO	保留。	0
5	SETEVENT	WO	设置事件，可以唤醒 WFE 的情况。	0
4	SEVONPEND	RW	当发生事件或者中断挂起状态时，可以从 WFE 指令后唤醒系统,如果未执行 WFE 指令，将在下次执行该指令后立即唤醒系统。 1：启用的事件和所有中断（包括未开启中断）都能唤醒系统； 0：只有启用的事件和启用的中断可以唤醒系统。	0
3	WFITOWFE	RW	将 WFI 指令当成是 WFE 执行。 1：将之后的 WFI 指令当做 WFE 指令； 0：无作用。	0
2	SLEEPDEEP	RW	控制系统的低功耗模式： 1：deepsleep 0：sleep	0
1	SLEEPONEXIT	RW	控制离开中断服务程序后，系统状态： 1：系统进入低功耗模式； 0：系统进入主程序。	0
0	Reserved	RO	保留。	0

3.2 中断相关 CSR 寄存器

另外，下列 CSR 寄存器也对中断的处理有重大影响：

机器模式异常基地址寄存器（mtvec）

名称	CSR 地址	访问	描述	复位值
mtvec	0x305	MRW	异常基地址寄存器	0x00000000

其各位定义为：

位	名称	访问	描述	复位值
[31:2]	BASEADDR[31:2]	MRW	中断向量表基地址。	0
1	Reserved	MRO	保留	0
0	MODE0	MRW	中断或异常入口地址模式选择： 0：使用统一入口地址； 1：根据中断编号*4 进行地址偏移。	0

对于 V3 系列微处理器的 MCU，启动文件里默认配置了 MODE0 为 1，异常或中断的入口根据中断编号*4 进行偏移，注意 V3 系列微处理器向量表处存储的是一条跳转指令。

3.3 中断嵌套

配合中断配置寄存器 PFIC_CFGR 和中断优先级寄存器 PFIC_IPRIOR，可以允许中断发生嵌套。在中断配置寄存器中使能嵌套（V3 系列微处理器默认打开嵌套），并且配置对应中断的优先级。优先级数值越小，优先级越高。抢占位数值越小，抢占优先级越高。同一抢占优先级下，若有中断同时挂起，微处理器优先响应优先级数值较小（优先级高）的中断。

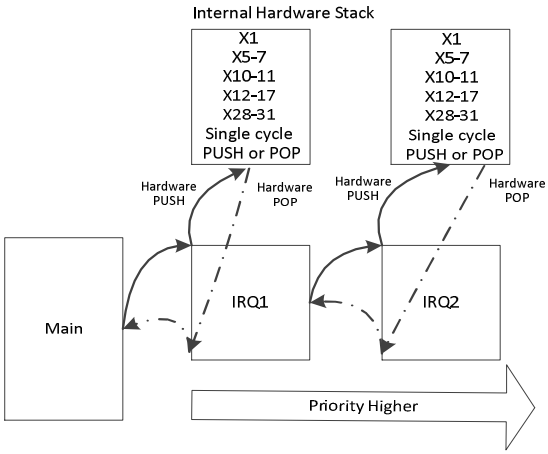
3.4 硬件压栈

当发生异常或者中断时，微处理器停止当前程序流，转向执行异常或者中断处理函数时，需要对当前程序流的现场进行保存。异常或中断返回后，需要恢复现场后，继续执行停止的程序流。对于 V3 系列微处理器，这里的“现场”指的是表 1-2 中，所有的 Caller Saved 寄存器。

V3 系列微处理器，支持硬件单周期自动保存其中 16 个整形 Caller Saved 寄存器至内部的堆栈区域，该区域对用户不可见。当异常或者中断返回后，硬件单周期自动从内部堆栈区恢复数据至 16 个整形寄存器。硬件压栈支持嵌套，嵌套深度最大为 2 级。

微处理器压栈示意图如下图所示：

图 3-1 压栈示意图



注： 1. 使用硬件压栈的中断函数需要使用 MRS 或者其提供的工具链进行编译且中断函数需要采用__attribute__((interrupt("WCH-Interrupt-fast")))声明。

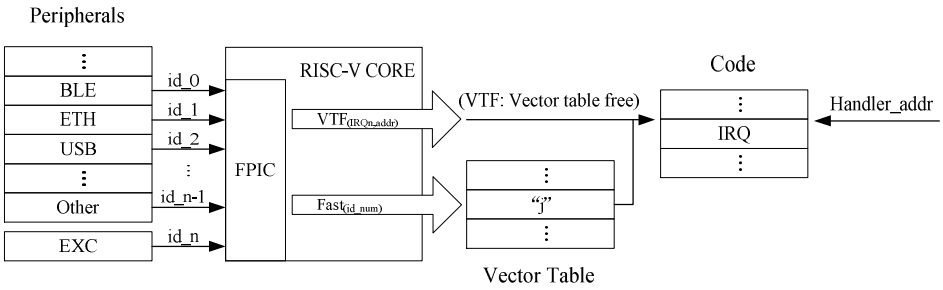
2. 使用软件压栈的中断函数采用__attribute__((interrupt()))声明。

3.5 免表中断

可编程快速中断控制器（PFIC）提供四个免表（Vector Table Free）中断通道，即不经过中断向量表的查表过程，直达中断函数入口。

正常配置一个中断函数的同时，将其中断编号、中断服务函数基地址和偏移地址写入对应的 PFIC 控制器寄存器中，即可使能 VTF 通道。其可以减小中断响应延迟，在非零等待存储器中更明显。PFIC 对于快速中断和免表中断的响应过程如下图 3-2 所示。

图 3-2 可编程快速中断控制器示意图



第 4 章 系统定时器 SysTick

青稞 V3 系列微处理器内部设计了一个 64 位加计数器（SysTick），其时钟源为系统时钟的 8 分频。可为实时操作系统提供时基，提供定时、测量时间等。定时器涉及 5 个寄存器并映射到外设地址空间，用于控制 SysTick，如下表 4-1 所示。

表 4-1 SysTick 寄存器列表

名称	访问地址	描述	复位值
STK_CTLR	0xE000F000	系统计数控制寄存器	0x00000000
STK_CNTL	0xE000F004	系统计数器低位寄存器	0x00000000
STK_CNTH	0xE000F008	系统计数器高位寄存器	0x00000000
STK_CMPLR	0xE000F00C	计数比较值低位寄存器	0x00000000
STK_CMPHR	0xE000F010	计数比较值高位寄存器	0x00000000

各寄存器详细描述如下：

系统计数控制寄存器（STK_CTLR）

表 4-2 SysTick 控制寄存器

位	名称	访问	描述	复位值
[31:1]	Reserved	RO	保留。	0
0	STE	RW	系统计数器使能控制位： 1：启动系统计数器 STK（HCLK/8 时基）； 0：关闭系统计数器 STK，计数器停止计数。	0

系统计数器低位寄存器（STK_CNTL）

表 4-3 SysTick 计数器低位寄存器

位	名称	访问	描述	复位值
[31:0]	CNTL	RW	当前计数器计数值低 32 位。 此寄存器可按 8 位/16 位/32 位读取，但是只能以 8 位进行写。	0

注：寄存器 STK_CNTL 和寄存器 STK_CNTH 共同构成了 64 位系统计数器。

系统计数器高位寄存器（STK_CNTH）

表 4-4 SysTick 计数器高位寄存器

位	名称	访问	描述	复位值
[31:0]	CNTH	RW	当前计数器计数值高 32 位。 此寄存器可按 8 位/16 位/32 位读取，但是只能以 8 位进行写。	0

注：寄存器 STK_CNTL 和寄存器 STK_CNTH 共同构成了 64 位系统计数器。

计数比较值低位寄存器（STK_CMPLR）

表 4-5 SysTick 比较值低位寄存器

位	名称	访问	描述	复位值
[31:0]	CMPL	RW	设置计数器比较值低 32 位。 当 CMP 值和 CNT 值相等时将触发 STK 中断。 此寄存器可按 8 位/16 位/32 位读取，但是只能以 8 位进行写。	0

注：寄存器 STK_CMPLR 和寄存器 STK_CMPHR 共同构成了 64 位计数器比较值。

计数比较值高位寄存器（STK_CMPHR）

表 4-6 SysTick 比较值高位寄存器

位	名称	访问	描述	复位值
[31:0]	CMPH	RW	设置计数器比较值高 32 位。 当 CMP 值和 CNT 值相等时将触发 STK 中断。 此寄存器可按 8 位/16 位/32 位读取，但是只能以 8 位进行写。	0

注：寄存器 STK_CMPLR 和寄存器 STK_CMPHR 共同构成了 64 位计数器比较值。

第 5 章 处理器低功耗设置

青稞 V3 系列微处理器支持通过 WFI(Wait For Interrupt)指令进入睡眠状态，实现较低的静态功耗。配合 PFIC 的系统控制寄存器（PFIC_SCTLR），能实现多种睡眠模式和 WFE 指令。

5.1 进入睡眠

青稞 V3 系列微处理器可以通过两种方式进入睡眠，即等待中断（Wait for Interrupt，WFI）和等待事件（Wait For Event，WFE）。WFI 方式是指微处理器进入睡眠后，等待中断来唤醒，醒来后进入相应的中断中去执行。WFE 方式是指微处理器进入睡眠后，等待事件来唤醒，醒来后继续执行之前停止的程序流。

标准的 RISC-V 支持 WFI 指令，直接执行 WFI 命令，即可通过 WFI 方式进入睡眠。而对于 WFE 方式，系统控制寄存器 PFIC_SCTLR 中 WFITOWFE 位用于控制将之后的 WFI 指令当做 WFE 处理，实现 WFE 方式进入睡眠。

根据 PFIC_SCTLR 中 SLEEPDEEP 位控制睡眠的深度：

- 若PFIC_SCTLR寄存器中的SLEEPDEEP清零，微处理器进入睡眠模式，除SysTick及部分唤醒逻辑外的内部单元时钟允许被关闭。
- 若PFIC_SCTLR寄存器中的SLEEPDEEP置位，微处理器进入深度睡眠模式，所有单元时钟均允许被关闭。

当微处理器处于调试模式下时，无法进入任何一种睡眠模式。

5.2 睡眠唤醒

青稞 V3 系列微处理器因 WFI 和 WFE 睡眠后，可采用以下方式唤醒：

- WFI方式进入睡眠后，可由以下方式唤醒：
  （1）微处理器可被中断控制器响应的中断源唤醒，唤醒后，微处理器先执行中断函数。
  （2）进入睡眠模式，调试请求可以使微处理器唤醒，进入深度睡眠，调试请求无法唤醒微处理器。

- WFE方式进入睡眠后，微处理器可被下面的方式唤醒：
  （1）内部或外部的事件，此时无需配置中断控制器，唤醒后继续执行程序。
  （2）若使能某中断源，产生中断时微处理器被唤醒，唤醒后，微处理器先执行中断函数。
  （3）若配置PFIC_SCTLR中的SEVONPEND位，中断控制器不使能中断下，但产生新的中断挂起信号时（之前产生的挂起信号不生效）也可以使微处理器唤醒，唤醒后需要手动清除相应的中断挂起标志。
  （4）进入睡眠模式调试请求可以使微处理器唤醒，进入深度睡眠，调试请求无法唤醒微处理器。

另外，可以通过配置PFIC_SCTLR中的SLEEPONEXIT位控制微处理器唤醒后的状态：

- SLEEPONEXIT置位，最后一级中断返回指令（mret）将触发WFI方式睡眠。

- SLEEPONEXIT清零，无影响。

搭载V3系列微处理器的各类MCU产品可根据PFIC_SCTLR不同配置，采取不同的睡眠方式，关闭不同的外设及时钟，执行不同的电源管理策略和唤醒方式，实现多种低功耗模式。

第 6 章 调试支持

青稞 V3 系列微处理器内含一个硬件调试模块，支持复杂的调试操作。调试模块可以暂停和恢复微处理器的运行，当微处理器暂停时，调试模块可以通过抽象命令、program buffer 部署指令等等方式访问微处理器的 GPRs、CSRs、储存器、外部设备等等。

调试模块遵循 RISC-V External Debug Support Version0.13.2 规范，详细文档可在基金会官方网站下载。

6.1 调试模块

微处理器内部的调试模块，能够执行调试主机下发的调试操作，包括：

- 通过调试接口访问寄存器
- 通过调试接口可以使微处理器复位、暂停、恢复
- 通过调试接口读写存储器、指令寄存器以及外部设备
- 通过调试接口可以部署多条任意指令
- 通过调试接口设置软件断点
- 支持单步调试

调试模块内部寄存器使用7位地址编码，青稞V3系列微处理器内部实现了以下寄存器：

表6-1 调试模块寄存器列表

名称	访问地址	描述
data0	0x04	数据寄存器 0，可用于暂存数据
data1	0x05	数据寄存器 1，可用于暂存数据
dmcontrol	0x10	调试模块控制寄存器
dmstatus	0x11	调试模块状态寄存器
hartinfo	0x12	微处理器状态寄存器
abstractcs	0x16	抽象命令状态寄存器
command	0x17	抽象命令寄存器
progbuf0-7	0x20-0x27	指令缓存寄存器 0-7
haltsum0	0x40	暂停状态寄存器

调试主机可通过配置dmcontrol寄存器控制微处理器的暂停、恢复、复位等。也可通过command寄存器触发调试模块生成抽象命令。RISC-V标准定义三种抽象命令类型：访问寄存器、快速访问、访问存储器。青稞V3微处理器支持其中两种，不支持快速访问。可以通过抽象命令实现寄存器访问，存储器的连续访问等。

调试模块内部实现了八个指令缓存寄存器progbuf0-7，调试主机可向缓冲区缓存多条指令（可以是压缩指令），可选择执行完抽象命令后继续执行指令缓存寄存器中的指令，也可直接执行缓存的指令。需要注意的是，progbufs中的最后一条指令需要是一条“ebreak”或“c.ebreak”指令。通过抽象命令和progbufs中缓存的指令，也可实现储存器、外设等的访问。

各寄存器详细描述如下：

数据寄存器 0（data0）

表6-2 data0寄存器定义

位	名称	访问	描述	复位值
[31:0]	data0	RW	数据寄存器 0，用于暂存数据	0

数据寄存器 1（data1）

表6-3 data1寄存器定义

位	名称	访问	描述	复位值
[31:0]	data1	RW	数据寄存器 1，用于暂存数据	0

调试模块控制寄存器（dmcontrol）

该寄存器可以控制微处理器的暂停、复位、恢复。调试主机向对应的字段写数据，即可实现暂停（haltreq）、复位（ndmreset）、恢复（resumereq）。各位描述如下：

表6-4 dmcontrol寄存器定义

位	名称	访问	描述	复位值
31	haltreq	WO	0：清除暂停请求 1：发出暂停请求	0
30	resumereq	W1	0：无效 1：恢复当前微处理器 *注：写 1 有效，微处理器恢复后，硬件清零*	0
29	Reserve	RO	保留	0
28	ackhavereset	W1	0：无效 1：清除微处理器的 haverest 状态位	0
[27:2]	Reserve	RO	保留	0
1	ndmreset	RW	0：清除复位 1：复位调试模块以外的整个系统	0
0	dmactive	RW	0：复位调试模块 1：调试模块正常工作	0

调试模块状态寄存器（dmstatus）

该寄存器用于指示调试模块的状态，是一个只读寄存器，各位的描述如下：

表6-5 dmstatus寄存器定义

位	名称	访问	描述	复位值
[31:20]	Reserve	RO	保留	0
19	allhavereset	RO	0：无效 1：微处理器复位	0
18	anyhavereset	RO	0：无效 1：微处理器复位	0
17	allresumeack	RO	0：无效 1：微处理器已经恢复	0
16	anyresumeack	RO	0：无效 1：微处理器已经恢复	0
[15:14]	Reserve	RO	保留	0
13	allavail	RO	0：无效 1：微处理器不可用	0
12	anyavail	RO	0：无效 1：微处理器不可用	0
11	allrunning	RO	0：无效 1：微处理器正在运行	0
10	anyrunning	RO	0：无效	0

			1：微处理器正在运行	
9	allhalted	RO	0：无效 1：微处理器处于暂停	0
8	anyhalted	RO	0：无效 1：微处理器出暂停	0
7	authenticated	RO	0：使用调试模块前需要认证 1：调试模块已经通过认证	0x1
[6:4]	Reserve	RO	保留	0
[3:0]	version	RO	调试系统支持架构版本 0010：V0.13	0x2

微处理器状态寄存器（hartinfo）

该寄存器用于向调试主机提供微处理器的信息，是一个只读寄存器，各个位描述如下：

表6-6 hartinfo寄存器定义

位	名称	访问	描述	复位值
[31:24]	Reserve	RO	保留	0
[23:20]	nscratch	RO	支持的 dscratch 寄存器数量	0x2
[19:17]	Reserve	RO	保留	0
16	dataaccess	RO	0：数据寄存器映射为 CSR 地址 1：数据寄存器映射为存储器地址	0x1
[15:12]	datasize	RO	数据寄存器数量	0x2
[11:0]	dataaddr	RO	数据寄存器 data0 偏移地址，其基地址为 0xe0000000	0x380

抽象命令控制与状态寄存器（abstractcs）

该寄存器用于指示抽象命令执行的情况，调试主机可以通过读取该寄存器，了解上一个抽象命令是否执行完毕，并且可以检查抽象命令执行过程中是否产生错误以及错误的类型，该寄存器的详细描述如下：

表6-7 abstractcs寄存器定义

位	名称	访问	描述	复位值
[31:29]	Reserve	RO	保留	0
[28:24]	progbufsize	RO	表示 program buffer 程序缓存寄存器数量	0x8
[23:13]	Reserve	RO	保留	0
12	busy	RO	0：无抽象命令执行 1：有抽象命令正在执行 *注：执行完毕后，硬件清零*	0
11	Reserve	RO	保留	0
[10:8]	cmder	RW	抽象命令错误类型 000：无错误 001：抽象命令执行时对 command、abstractcs、abstractauto 寄存器进行写或者对 data 和 progbuf 寄存器进行读写操作 010：不支持当前抽象命令 011：执行抽象命令出现异常 100：微处理器未暂停或不可用，而不能执行抽象命令	0

			101：总线出错 110：通讯时奇偶校验位错误 111：其他错误 *注：对应位写 1 用于清零*	
[7:4]	Reserve	R0	保留	0
[3:0]	datacount	R0	数据寄存器数量	0x2

抽象命令寄存器（command）

调试主机可以通过抽象命令寄存器中写入不同的配置值，对微处理器内部的 GPRs、FPRs、CSRs 寄存器进行访问。

当对寄存器访问时，command 寄存器各位定义如下：

表6-8 访问寄存器时command寄存器定义

位	名称	访问	描述	复位值
[31:24]	cmdtype	WO	抽象命令类型 0：访问寄存器 1：快速访问（未支持） 2：访问存储器（未支持）	0
23	Reserve	WO	保留	0
[22:20]	aarsize	WO	访问寄存器数据位宽 000：8 位 001：16 位 010：32 位 011：64 位（未支持） 100：128 位（未支持） *注：当访问浮点寄存器 FPRs 时，仅支持 32 位访问*	0
19	aarpostincrement	WO	0：无影响 1：访问寄存器后自动增加 regno 的值	0
18	postexec	WO	0：无影响 1：执行抽象命令后执行 progbuf 中的命令	0
17	transfer	WO	0：不执行 write 指定的操作 1：执行 write 指令的操纵	0
16	write	WO	0：从指定的寄存器拷贝数据至 data0 1：从 data0 寄存器拷贝数据至指定的寄存器	0
[15:0]	regno	WO	指定访问寄存器 0x0000-0x0fff 为 CSRs 0x1000-0x101f 为 GPRs 0x1020-0x103f 为 FPRs	0

指令缓存寄存器（progbufx）

该寄存器用于存放任意指令，部署相应的操作，包括 8 个，需要注意最后一条执行需要时“ebreak”或者“c.ebreak”。

表6-9 progbuf寄存器定义

位	名称	访问	描述	复位值
[31:0]	progbuf	RW	缓存操作的指令编码，可以包括压缩指令	0

暂停状态寄存器（haltsum0）

该寄存器用于指示微处理器是否暂停，每一位指示一个微处理器的暂停状态，当只有一个核时，仅用该寄存器最低位表示。

表6-10 haltsum0寄存器定义

位	名称	访问	描述	复位值
[31:1]	Reserve	RO	保留	0
0	haltsum0	RO	0：微处理器正常运行 1：微处理器停止	0

除了上述调试模块的寄存器外，调试功能还涉及部分 CSR 寄存器，主要是调试控制和状态寄存器 dcsr 和调试指令指针 dpc，寄存器详细描述如下：

调试控制和状态寄存器（dcsr）

表6-11 dcsr寄存器定义

位	名称	访问	描述	复位值
[31:28]	xdebugver	DRO	0000：不支持外部调试 0100：支持标准的外部调试 1111：支持外部调试，但是不符合规范	0x4
[27:16]	Reserve	DRO	保留	0
15	ebreakm	DRW	0：机器模式下的 ebreak 指令的行为如特权文档所述 1：机器模式下的 ebreak 指令能够进入调试模式	0
[14:13]	Reserve	DRO	保留	0
12	ebreaku	DRW	0：用户模式下的 ebreak 指令的行为如特权文档所述 1：用户模式下的 ebreak 指令能够进入调试模式	0
11	stepie	DRW	0：单步调试下禁止中断 1：单步调试下启用中断	0
10	Reserve	DRO	保留	0
9	stoptime	DRW	0：调试模式下系统定时器运行 1：调试模式下系统定时器停止	0
[8:6]	cause	DRO	进入调试的原因 001：以 ebreak 指令方式进入调试（优先级为 3） 010：以 trigger module 形式进入调试（优先级为 4，最高） 011：以暂停请求形式进入调试（优先级为 1） 100：以单步调试形式进入调试（优先级为 0，最低） 101：微处理器复位之后直接停止进入调试模式（优先级为 2） 其他：保留	0
[5:3]	Reserve	DRO	保留	0
2	step	DRW	0：关闭单步调试	0

			1：开启单步调试	
[1:0]	prv	DRW	特权模式 00：用户模式 01：监督模式（未支持） 10：保留 11：机器模式 *注：记录进入调试模式时特权模式，调试器可以修改该值，以修改退出调试时的特权模式*	0

调试模式程序指针（dpc）

该寄存器用于保存微处理器进入调试模式之后将要执行的下一条指令的地址，其值根据进入调试的原因不同，更新的规则也不相同。dpc 寄存器详细描述如下：

表6-12 dpc寄存器定义

位	名称	访问	描述	复位值
[31:0]	dpc	DRW	指令地址	0

寄存器的更新规则如下表所示：

表6-13 dpc更新规则

进入调试方式	dpc 更新规则
ebreak	Ebreak 指令的地址
single step	当前指令的下一条指令的指令地址
trigger module	暂不支持
halt request	进入调试时，下一条将要被执行的指令地址

6.2 调试接口

区别于标准的 RISC-V 定义的 JTAG 接口，青稞 V3 系列微处理器采用两线调试接口，遵循沁恒调试接口协议 V1.0。调试接口负责调试主机和调试模块之间的通讯，实现调试主机对调试模块寄存器的读写操作。沁恒设计了 WCH_Link，并开源其原理图和程序二进制文件，可用于所有 RISC-V 架构的微处理器调试。

具体的调试接口协议参考沁恒调试协议手册。

第 7 章 CSR 寄存器列表

RISC-V 架构中定义了一些控制和状态寄存器（Control and Status Register,CSR），用于控制和记录微处理器的运行状态。前文已经介绍到部分 CSR，本章将详细说明青稞 V3 系列微处理器实现的 CSR 寄存器。

7.1 CSR 寄存器列表

表 7-1 微处理器 CSR 寄存器列表

类型	名称	CSR 地址	访问	描述
RISC-V 标准 CSR	marchid	0xF12	MRO	架构编号寄存器
	mimpid	0xF13	MRO	硬件实现编号寄存器
	mstatus	0x300	MRW	状态寄存器
	misa	0x301	MRW	硬件指令集寄存器
	mtvec	0x305	MRW	异常基地址寄存器
	mscratch	0x340	MRW	机器模式暂存寄存器
	mepc	0x341	MRW	异常程序指针寄存器
	mcause	0x342	MRW	异常原因寄存器
	mtval	0x343	MRW	异常值寄存器
	dcsr	0x7B0	DRW	调试控制与状态寄存器
	dpc	0x7B1	DRW	调试模式程序指针寄存器
	dscratch0	0x7B2	DRW	调试模式暂存寄存器 0
	dscratch1	0x7B3	DRW	调试模式暂存寄存器 1

7.2 RISC-V 标准 CSR 寄存器

架构编号寄存器（marchid）

此寄存器为只读寄存器，用于指示当前微处理器硬件架构编号，其主要由厂商编码、架构编码、系列编码、版本编码构成。其各位定义如下：

表 7-2 marchid 寄存器定义

位	名称	访问	描述	复位值
31	Reserved	MRO	保留	1
[30:26]	Vender0	MRO	厂商编码 0 固定为字母“W”编码	0x17
[25:21]	Vender1	MRO	厂商编码 1 固定为字母“C”编码	0x03
[20:16]	Vender2	MRO	厂商编码 2 固定为字母“H”编码	0x08
15	Reserved	MRO	保留	1
[14:10]	Arch	MRO	架构编码 RISC-V 架构固定为字母“V”编码	0x16
[9:5]	Serial	MRO	系列编码 青稞 V3 系列，固定为数字“4”	0x04
[4:0]	Verision	MRO	版本编码 可以是版本“A” “B” “C” “F”等字母的编码	x

其中厂商编号、版本编号为英文字母，系列编号为数字。字母的编码表如下表所示：

表 7-3 字母映射表

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

该寄存器在任何机器实现中均为可读，青稞 V3 系列处理器中，该寄存器读返回为零。

硬件实现编号寄存器（mimpid）

该寄存器主要由厂商编码构成，其各位定义如下：

表 7-4 mimpid 寄存器定义

位	名称	访问	描述	复位值
31	Reserved	MRO	保留	1
[30:26]	Vender0	MRO	厂商编码 0 固定为字母“W”编码	0x17
[25:21]	Vender1	MRO	厂商编码 1 固定为字母“C”编码	0x03
[20:16]	Vender2	MRO	厂商编码 2 固定为字母“H”编码	0x08
15	Reserved	MRO	保留	1
[14:1]	Reserved	MRO	保留	0
0	Reserved	MRO	保留	1

该寄存器在任何机器实现中均为可读，青稞 V3 系列处理器中，该寄存器读返回为零。

机器模式状态寄存器（mstatus）

该寄存器前文已经描述部分域，其各位定位如下：

表 7-5 mstatus 寄存器定义

位	名称	访问	描述	复位值
[31:15]	Reserved	MRO	保留	0
[14:13]	Reserved	MRO	保留	0
[12:11]	MPP	MRW	进中断前特权模式	0
[10:8]	Reserved	MRO	保留	0
7	MPIE	MRW	进中断之前中断使能状态	0
[6:4]	Reserved	MRO	保留	0
3	MIE	MRW	机器模式中断使能	0
[2:0]	Reserved	MRO	保留	0

MPP 域用于保存进入异常或中断前的特权模式，用于退出异常或中断后的特权模式恢复，MIE 为全局中断使能位，当进入异常或中断时，MPIE 的值被更新为 MIE 值，需要注意的是青稞 V3 系列微处理器中，在最后一级嵌套中断前 MIE 不会被更新为 0，以保证机器模式下的中断嵌套继续执行。当退出异常或中断后，微处理器恢复为 MPP 保存的机器模式，并且 MIE 恢复为 MPIE 的值。

青稞 V3 微处理器支持机器模式和用户模式，若需要使微处理器仅工作在机器模式，可在启动文件的初始化中把 MPP 设置为 0x3，即返回后，始终保持在机器模式。

硬件指令集寄存器（misa）

此寄存器用于指示微处理器的架构和支持的指令集扩展情况，其各位描述如下：

表 7-6 misa 寄存器定义

位	名称	访问	描述	复位值
[31:30]	MXL	MRO	机器字长 1：32	1

			2：64 3：128	
[29:26]	Reserved	MR0	保留	0
[25:0]	Extensions	MR0	指令集扩展情况	x

其 MXL 用于指示微处理器的字长，青稞 V3 均为 32 位微处理器，该域固定为 1。Extensions 用于指示微处理器支持扩展指令集详情，每位表示一类扩展，其详细描述如下表所示：

表 7-7 指令集扩展详情

位	名称	描述
0	A	Atomic extension
1	B	Tentatively reserved for Bit-Manipulation 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	Hypervisor extension
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

例如青稞 V3A 微处理器，该寄存器值为 0x40101105，即支持的指令集架构为 RV32IMAC，且其具有用户模式实现。

机器模式异常基址寄存器（mtvec）

该寄存器器用于存储异常或中断处理程序的基地址，并且低两位用于配置向量表的模式和识别方式，详见 3.2 节介绍。

机器模式暂存寄存器（mscratch）

表 7-8 mscratch 寄存器定义

位	名称	访问	描述	复位值
[31:0]	mscratch	MRW	数据暂存	0

该寄存器是一个机器模式下 32 位的可读可写的寄存器，用于临时保存数据。例如，在进入异常或中断处理程序时，将用户堆栈指针 SP 存入该寄存器，并将中断栈指针赋值给 SP 寄存器。退出异常或异常后，从 mscratch 中恢复用户堆栈指针 SP 的值。即可实现中断栈和用户栈的隔离。

机器模式异常程序指针寄存器（mepc）

表 7-9 mepc 寄存器定义

位	名称	访问	描述	复位值
[31:0]	mepc	MRW	异常程序指针	0

该寄存器用于保存进入异常或中断时的程序指针，其用于在产生异常或中断时保存进入异常前的指令 PC 指针，当处理完异常或中断后，mepc 被作为返回地址，用于异常或中断返回。但是需要注意的是：

- 当发生异常时，mepc被更新为当前产生异常的的指令的PC值；
- 当发生中断时，mepc被更新为下一条指令的PC值。

固在处理完异常，需要返回时，应注意修改mepc的值，更多的细节内容可参第2章异常章节。

机器模式异常原因寄存器（mcause）

表 7-10 mcause 寄存器定义

位	名称	访问	描述	复位值
31	Interrupt	MRW	中断指示域 0：异常 1：中断	0
[30:0]	Exception Code	MRW	异常编码，详见表 2-1	0

该寄存器主要用于保存产生异常的原因或中断的中断编号，其最高位为 Interrupt 域，用于指示当前发生的是异常还是中断。低位为异常编码，用于指示具体的原因。其详细内容可以参考第 2 章异常章节。

机器模式异常值寄存器（mtval）

表 7-11 mtval 寄存器定义

位	名称	访问	描述	复位值
[31:0]	mtval	MRW	异常值	0

该寄存器用于保存发生异常时，引起异常的值。其保存的值和时间等详细的内容可以参考第 2 章异常章节。

调试模式程序指针（dpc）

该寄存器用于保存微处理器进入调试模式之后将要执行的下一条指令的地址，其值根据进入调试的原因不同，更新的规则也不相同。详细描述参考 6.1 节。

调试模式暂存寄存器（dscratch0-1）

该组寄存器用于调试模式下临时存储数据。

表 7-12 dscratch0-1 寄存器定义

位	名称	访问	描述	复位值
[31:0]	dscratch	DRW	调试模式数据暂存值	0