第 50 章 采用 MIPS32® microAptiv™ 和 M-Class 内核的器件的 CPU

本章包括下列主题：

50.1 简介 50-2
50.2 架构概述 50-4
50.3 PIC32 CPU 详细信息 50-8
50.4 写入 CP0 寄存器时的特殊注意事项 50-13
50.5 MIPS32 架构 50-14
50.6 CPU 总线 50-15
50.7 内部系统总线 50-15
50.8 置 1/ 清零 / 取反 50-16
50.9 ALU 状态位 50-16
50.10 中断和异常机制 50-17
50.11 编程模型 50-17
50.12 浮点单元 （FPU） 50-24
50.13 协处理器 0 （CP0）寄存器 50-42
50.14 协处理器 1 （CP1）寄存器 50-121
50.15 microMIPS 执行 50-132
50.16 MCU ASE 扩展 50-132
50.17 MIPS DSP ASE 扩展 50-133
50.18 存储器模型 （仅限 MCU） 50-133
50.19 存储器管理 （仅限 MPU） 50-135
50.20 L1 高速缓存 （仅限 MPU） 50-141
50.21 CPU 指令 50-145
50.22 MIPS DSP ASE 指令 50-151
50.23 CPU 初始化 50-153
50.24 复位的影响 50-154
50.25 相关应用笔记 50-155
50.26 版本历史 50-156

© 2013-2016 Microchip Technology Inc.

注： 本系列参考手册章节旨在用作对器件数据手册的补充。根据不同的器件型号，本手册章节可能并不适用于所有 PIC32 器件。

请参见当前器件数据手册中 **“CPU”** 章节开始处的注释，以查看本文档是否支持您当前使用的器件。

器件数据手册和系列参考手册章节可从 Microchip 网站下载：http://www.microchip.com

50.1 简介

根据器件系列，PIC32 器件是基于 Imagination Technologies Ltd. 的 microAptiv™ 微处理器内核或 M-Class 微处理器内核的复杂片上系统（System-on-Chip， SoC）。本文档概述采用这些微处理器内核的 PIC32 单片机的 CPU 系统架构和特性。

microAptiv 微处理器内核是 MIPS® M14KE™ 和 M14KEc™ 微处理器内核的超集。这两个内核是最新型的 32 位低功耗 RISC 处理器内核，采用了增强型 MIPS32® 版本 2 指令集架构（Instruction Set Architecture， ISA）。

M-Class 微处理器内核是 microAptiv™ 微处理器内核的超集。该 32 位、低功耗、RISC 处理器内核采用增强型 MIPS32® 版本 5 指令集架构 （ISA）。

请访问 Imagination Technologies Ltd. 网站（www.imgtec.com），了解更多有关微处理器内核的信息。

根据内核配置，将使用 MCU 或 MPU 两者中的一个，如表 50-1 所示。

表 50-1： microAptiv 和 M-Class 微处理器内核配置

MCU 特性	MPU 特性
拆分式总线架构	统一总线架构
集成 DSP ASE （见**注 1**）	集成 DSP ASE （见**注 1**）
集成 MCU™ ASE	集成 MCU ASE
microMIPS™ 代码压缩	microMIPS 代码压缩
基于 FMT 的 MMU	基于 TLB 的 MMU
两个影子寄存器集	八个影子寄存器集
EJTAG TAP 控制器	EJTAG TAP 控制器
性能计数器	性能计数器
硬件跟踪 （iFlowtrace®）	硬件跟踪 （iFlowtrace）
	一级 （L1） CPU 高速缓存

注 1： 该特性并非在所有器件上均可用，请参见具体器件数据手册中的 **“CPU”** 章节以确定可用性。

MCU 和 MPU 的主要区别是：MPU 上存在一个 L1 高速缓存和基于 TLB 的 MMU。这些特性有助于实现使用操作系统来管理虚拟存储器的 PIC32 设计。

© 2013-2016 Microchip Technology Inc.

50.1.1 采用 microAptiv 微处理器内核的所有 PIC32 器件共有的主要特性

下列主要特性为基于 microAptiv 微处理器内核的所有 PIC32 器件共有的特性：

- microMIPS 可变长度指令模式，用于紧凑型代码
- 带有最多 256 个中断源的向量中断控制器
- 可对外设寄存器执行原子级位操作 （单周期）
- 高速 Microchip ICD 端口，具有基于硬件的非侵入式数据监视和应用程序数据流功能
- EJTAG 调试端口，支持广泛的第三方调试、编程和测试工具
- 指令控制的功耗管理模式
- 5 级流水线指令执行
- 内部代码保护，以帮助保护知识产权
- 支持算术饱和及溢出处理
- 零循环开销饱和及舍入操作
- 原子级读 - 修改 - 写存储器到存储器指令
- 具有最多 4 个累加器的 MAC 指令
- 固有小数数据类型 （Q15 和 Q31），支持舍入
- 数字信号处理 （Digital Signal Processing， DSP）特定应用扩展 （Application-Specific Extension， ASE）版本 2，增加了 DSP 功能，支持强大的数据处理操作
- 乘法 / 除法单元，最高指令发出速率为每个时钟一条 32 x 32 乘法指令

50.1.2 采用 M-Class 微处理器内核的所有 PIC32 器件共有的主要特性

除采用 microAptiv 内核器件的上述特性外，基于 M-Class 微处理器内核的所有 PIC32 器件还具有以下共有的主要特性：

- 采用最新的 MIPS 版本 5 架构，包括适用于工业控制器、物联网 （Internet of Things， IoT）、可穿戴、无线通信、汽车和存储等应用的 IP 保护和可靠性
- 浮点单元 （Floating Point Unit， FPU）

50.1.3 相关的 MIPS 文档

下列相关文档可从 Imagination Technologies Ltd. 的相关产品页面下载。请注意，需要登录才能访问这些文档。

可从以下链接下载关于 microAptiv 内核的文档：

http://www.imgtec.com/mips/aptiv/microaptiv.asp

可从以下链接下载关于 M-Class 内核的文档：

http://www.imgtec.com/mips/warrior/mclass.asp

50.2 架构概述

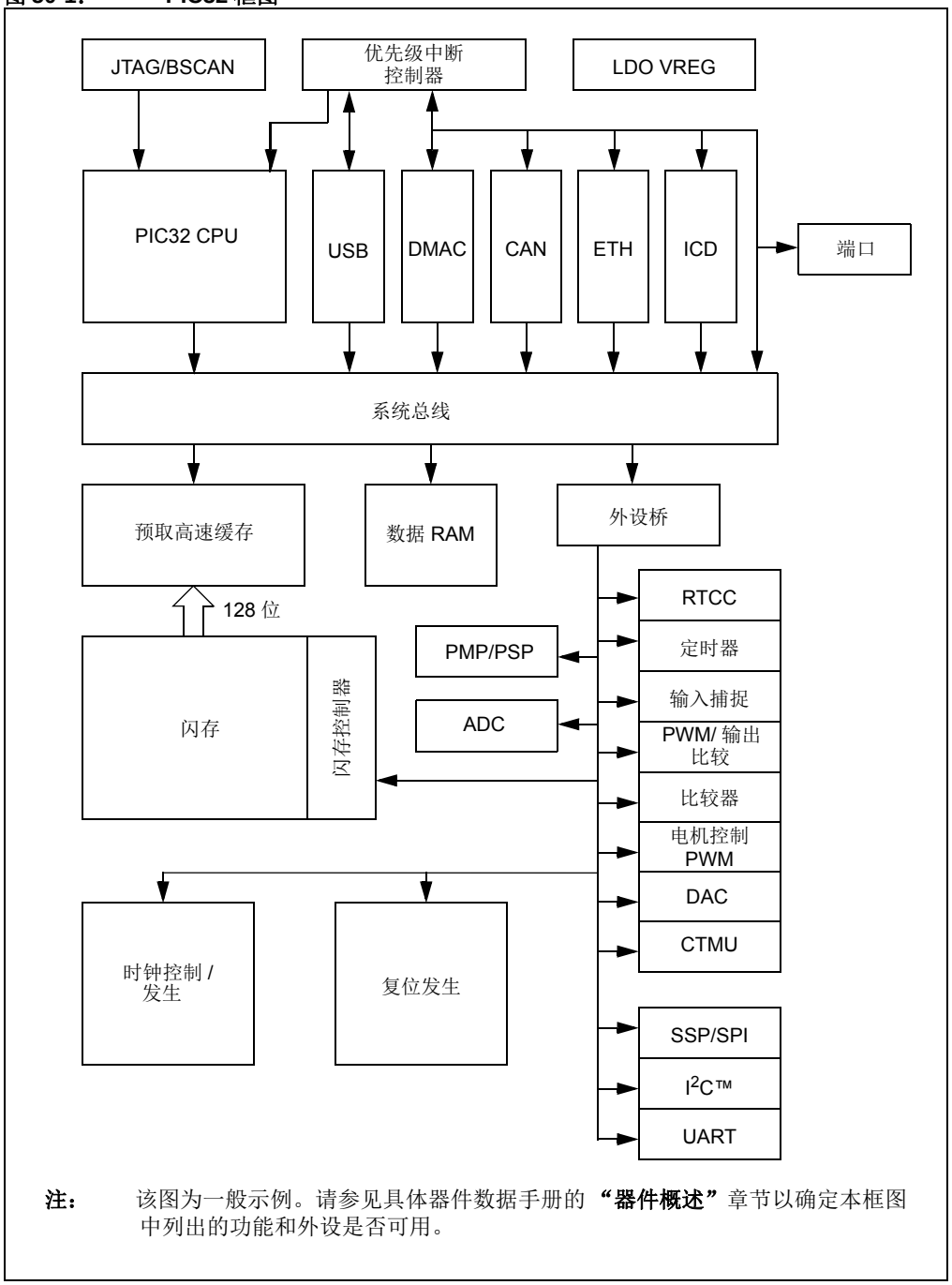
PIC32 系列器件是复杂的片上系统，具有许多特性。PIC32 系列的所有处理器中都包含了高性能 RISC CPU，可以使用 32 位、 16 位模式乃至混合模式进行编程。

采用 M-Class 内核的器件包含一个浮点单元 （FPU），利用 MIPS 版本 5 指令架构实现浮点计算。 FPU 支持 ANSI/IEEE 标准 754 （用于二进制浮点算术运算的 IEEE 标准）指定的单 / 双精度数据格式。

注： 请参见具体器件数据手册中的**“CPU”**章节，以确定FPU模块在您器件上的可用性。

PIC32 器件包含了高性能中断控制器、 DMA 控制器、 USB 控制器、在线调试器、用于对外设进行高速数据访问的高性能开关矩阵，以及用于保存数据和程序的片上数据 RAM 存储器。对于闪存，采用了可选的预取高速缓存和预取缓冲区，无需闪存访问延时，提供相当于 0 个等待状态的访问性能。

图 50-1： PIC32 框图



注： 该图为一般示例。请参见具体器件数据手册的 **“器件概述”**章节以确定本框图中列出的功能和外设是否可用。

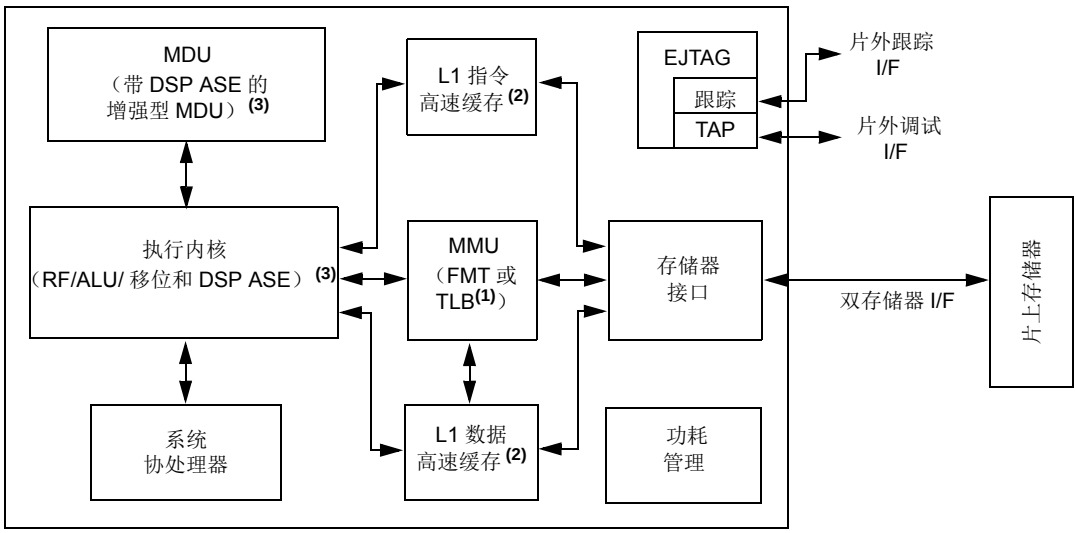
© 2013-2016 Microchip Technology Inc.

PIC32 器件外设通过一条系统总线和一系列内部总线与 CPU 连接。主外设总线通过一个或多个外设桥将大部分外设单元与系统总线进行连接。

PIC32 CPU 在程序控制下执行操作。 CPU 会同时进行指令的取指、译码和执行。指令位于闪存程序存储器或数据 RAM 存储器中。另外，采用 microAptiv 和 M-Class 内核的 PIC32 器件包含 MIPS DSP 特定应用扩展版本 2，支持许多强大的数据处理操作，可提供数字信号处理 （DSP） 功能。

PIC32 CPU 基于装载 / 存储架构，大多数操作都是针对一组内部寄存器执行的。它使用特定的装载和存储指令在这些内部寄存器和外界之间传送数据。

图 50-2： microAptiv™ 微处理器内核框图

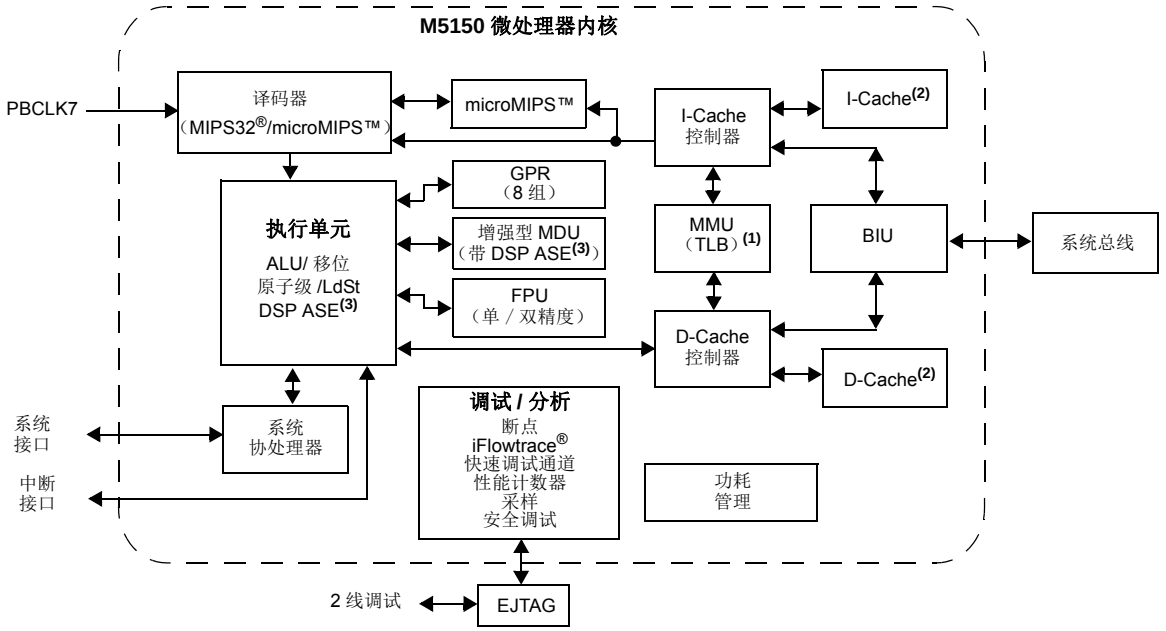


注 1： TLB 仅存在于采用 MPU 微处理器内核的器件上。

2： 一级 （L1）高速缓存仅存在于采用 MPU 微处理器内核的器件上。

3： DSP ASE 并非在所有器件上都可用。请参见具体器件数据手册中的 **“CPU”** 章节以确定可用性。

图 50-3： M-Class 微处理器内核框图



注 1： TLB 仅在采用 MPU 微处理器内核的器件上可用。

2： 一级 （L1）高速缓存仅在采用 MPU 微处理器内核的器件上可用。

3： DSP ASE 并非在所有器件上都可用。请参见具体器件数据手册中的 **“CPU”** 章节以确定可用性。

© 2013-2016 Microchip Technology Inc.

50.2.1 总线

所有 PIC32 器件使用系统总线连接 CPU 和其他总线主器件与存储器和其他目标器件。系统总线控制和仲裁不同总线主器件和目标器件之间的访问。特定 PIC32 器件的系统总线和总线仲裁类型取决于所采用的 microAptiv 或 M-Class CPU 内核。

基于 MCU 微处理器内核的 PIC32 器件采用分离式总线 CPU 架构。在该架构中，用于取指和数据装载 / 存储操作的总线是独立的。指令总线（或 I 侧总线）和数据总线（或 D 侧总线）连接到系统总线。系统总线支持同时访问不同总线主器件 （访问不同目标），且可通过仲裁算法将不同主器件对同一目标的访问序列化。由于 CPU 到系统总线具有两条不同的数据路径，所以对于系统来说， CPU 实际上是两个不同的总线主器件。从闪存中运行时，对 SRAM 和内部外设的装载和存储操作将与对闪存的取指操作并行进行。

基于 MPU 内核的 PIC32 器件使用统一总线 CPU 架构以及多层（交叉）系统总线。在该架构中，CPU 具有单个系统总线接口。系统总线使用专用链路在总线发起器和目标之间提供多个独立的数据路径。这支持在总线上进行并行数据事务。

注：	请参见具体器件数据手册的 **“存储器构成”**章节和 《PIC32 系列参考手册》的**第 3 章 “存储器构成”**（DS60001115），获取具体器件的系统总线说明。

除了 CPU 之外，根据具体的器件型号，在某些 PIC32 器件中还有其他总线主器件：

- DMA 控制器
- 在线调试器 （In-Circuit Debugger， ICD）
- USB 控制器
- CAN 控制器
- 以太网控制器

50.2.2 内核定时器

PIC32 架构包含了一个可供应用程序使用的内核定时器。该定时器通过两个协处理器寄存器实现：Count 寄存器和 Compare 寄存器。 Count 寄存器每两个系统时钟 （SYSCLK）周期递增一次。在调试模式期间，可以选择暂停计数递增。Compare 寄存器用于产生定时器中断（如需要）。当 Compare 寄存器与 Count 寄存器匹配时，将产生中断。只有在中断控制器中允许某个中断时，才会执行该中断。

关于内核定时器的更多信息，请参见 **50.13 “协处理器 0 （CP0）寄存器”**和 《PIC32 系列参考手册》中的**第 8 章 “中断”**（DS60001108）。

50.3 PIC32 CPU 详细信息

50.3.1 流水线分级

流水线包含 5 级：

- 指令 （I）级
- 执行 （E）级
- 存储 （M）级
- 对齐 （A）级
- 回写 （W）级

50.3.1.1 I 级 —— 取指

在 I 级期间：

- 从指令 SRAM 取指。
- microMIPS 指令转换为类似于 MIPS32 指令的指令

50.3.1.2 E 级 —— 执行

在 E 级期间：

- 从寄存器文件中取操作数
- M 和 A 级的操作数旁路传递到此级
- 对于寄存器 - 寄存器指令，算术逻辑单元 （Arithmetic Logic Unit， ALU）开始执行算术或逻辑运算
- 对于装载和存储指令， ALU 会计算数据虚拟地址， MMU 执行虚拟地址到物理地址的固定转换
- ALU 确定转移条件是否为真，并计算转移指令的虚拟转移目标地址
- 指令逻辑选择指令地址，并且 MMU 执行虚拟地址到物理地址的固定转换
- 所有乘法 / 除法运算都在此级开始

50.3.1.3 M 级 —— 从存储器取数据

在 M 级期间：

- 完成算术或逻辑 ALU 运算
- 对于装载和存储指令，将执行数据 SRAM 访问
- 完成阵列中的 16 x 16 或 32 x 16 乘法运算，并在 M 级停留一个时钟，以在 M 级中完成进位传输加法
- 32 x 32 乘法运算会在 M 级停留两个时钟，以在 M 级中完成阵列第二个周期和进位传输加法
- 乘法和除法计算在 MDU 中进行。如果计算在 IU 将指令推移到 M 级之前完成，那么 MDU 会将结果保存在临时寄存器中，直到 IU 将指令推移到 A 级 （因此将不会被中止）。

50.3.1.4 A 级 —— 对齐

在 A 级期间：

- 独立的对齐器将装载数据与其字边界对齐
- 乘法运算产生可供回写的结果。实际的寄存器回写在 W 级执行
- 从此级开始，装载数据或来自 MDU 的结果可通过旁路传递到 E 级

50.3.1.5 W 级 —— 回写

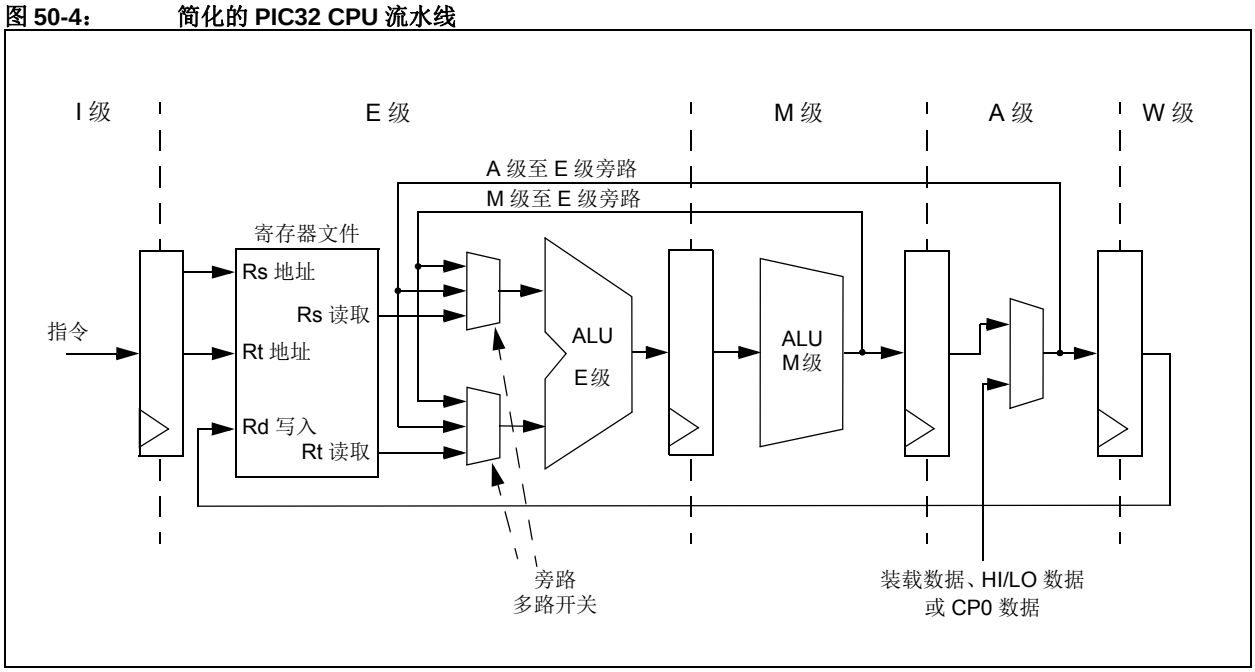
在 W 级期间：

© 2013-2016 Microchip Technology Inc.

对于寄存器 - 寄存器指令或装载指令，结果回写到寄存器文件中。

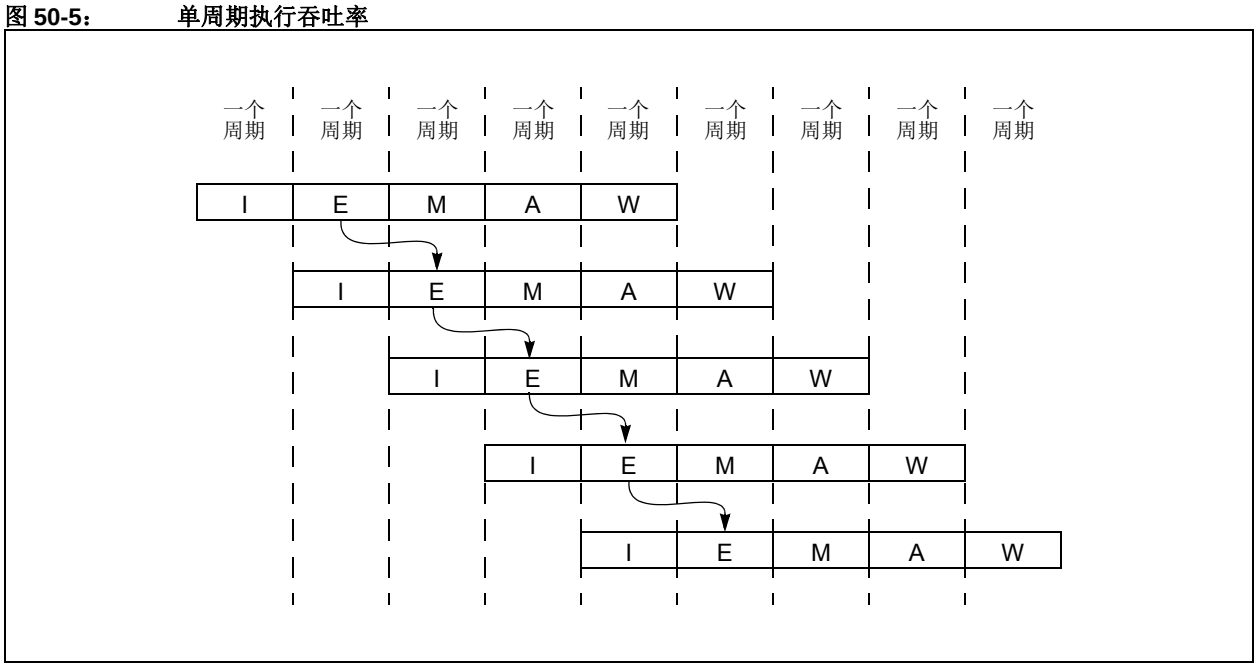
微处理器内核实现了“旁路”机制，通过该机制可以将运算结果直接送到需要它的指令处，而无需先将结果写入寄存器，然后再进行回读。

图 50-4： 简化的 PIC32 CPU 流水线



在 PIC32 内核中使用指令流水线实现了一个快速的单周期指令执行环境。

图 50-5： 单周期执行吞吐率



50.3.2 执行单元

PIC32 执行单元负责执行 MIPS 指令集大部分指令的处理。执行单元通过流水线执行的方式，为大部分指令提供了单周期吞吐率。流水线执行将复杂的操作拆分为称作 “级”的小片段。这些操作级在多个时钟周期中执行。

执行单元包含以下特性：

- 32 位加法器，用于计算数据地址
- 地址单元，用于计算下一条指令的地址
- 逻辑单元，用于进行转移判断和转移目标地址计算
- 装载对齐器
- 旁路多路开关，用于避免执行指令流时 （当数据生成指令后紧跟使用其结果的指令时）出现停顿
- 前导 0/1 检测单元，用于实现 CLZ 和 CLO 指令
- 算术逻辑单元 （ALU），用于执行位宽的逻辑运算
- 移位器和存储对齐器

50.3.3 乘法 / 除法单元 （MDU）

乘法 / 除法单元（Multiply/Divide unit， MDU）用于执行乘法和除法运算。 MDU 包含 32 x 16 乘法器、结果累加寄存器 （HI 和 LO）、乘法和除法状态机，以及执行这些功能所需的所有多路开关和控制逻辑。高性能流水线 MDU 支持在每个时钟周期执行一次 16 x 16 或 32 x 16 乘法运算；32 × 32 乘法运算可以每隔一个时钟周期发出一次。它采用适当的互锁机制来阻止发出背靠背 32 x 32 乘法运算。除法运算通过简单的每时钟 1 位的迭代算法实现，最坏情况下需要 35 个时钟周期才能完成。在除法仍在进行时任何发出后续 MDU 指令的尝试都会导致流水线停顿，直到除法运算完成为止。

微处理器内核还另外实现了一条乘法指令 MUL，该指令规定乘法结果的低 32 位放入寄存器文件中，而不是 HI/LO 寄存器对中。通过避免显式的从 LO 传送 （MFLO）指令 （使用 LO 寄存器时需要），并通过支持多个目标寄存器，可以提高乘法密集型运算的吞吐率。乘加（MADD/MADDU）和乘减（MSUB/MSUBU）这两条指令用于执行乘加和乘减运算。MADD 指令可以将两个数字相乘，然后将乘积与 HI 和 LO 寄存器的当前内容相加。类似地， MSUB 指令可以将两个操作数相乘，然后从 HI 和 LO 寄存器内容中减去乘积。MADD/MADDU 和 MSUB/MSUBU 运算常用于数字信号处理器（Digital Signal Processor， DSP）算法。

MDU 具有一个单独的流水线用于整数乘法和除法运算以及 DSP ASE 乘法指令 （见**注**）。该流水线与整数单元（Integer Unit，IU）流水线并行工作，且在 IU 流水线停顿时不停顿。这允许长时间运行的 MDU 运算被系统停顿和 / 或其他整数单元指令部分屏蔽。 MDU 支持在每个时钟周期执行一次 32 x 32 乘法或乘积累加运算。 32 位除法运算在 12-38 个时钟周期内执行。 MDU 还实现针对 HI/LO 寄存器的各种移位指令以及乘法指令，如 DSP ASE 中所定义。

注：	DSP ASE 并非在所有器件上都可用。请参见具体器件数据手册中的 **“CPU”** 章节以确定可用性。

50.3.4 影子寄存器集

PIC32 处理器实现了通用寄存器 （General Purpose Register， GPR）的一个或多个副本，供高优先级中断使用。这些额外的寄存器存储区称为影子寄存器集。当发生高优先级中断时，处理器可以无需软件干预而自动切换到影子寄存器集。这可以降低中断处理程序中的开销，并降低实际响应延时。

影子寄存器集由位于系统协处理器 （CP0）中的寄存器和位于 CPU 内核之外的中断控制器硬件控制。

关于影子寄存器集的更多信息，请参见《PIC32 系列参考手册》中的**第 8 章“中断”**（DS60001108）。

© 2013-2016 Microchip Technology Inc.

50.3.5 流水线互锁处理

当某个流水线级中的指令由于数据相依性或类似外部条件而无法前移时，流畅的流水线流程会被中断。流水线中断完全在硬件中进行处理。这些相依性称为 “互锁”。在每个周期，都会为所有执行中的指令检查互锁条件。一条指令依赖于前一条指令的结果的情况即为互锁条件。

通常， MIPS 处理器支持两种类型的硬件互锁：

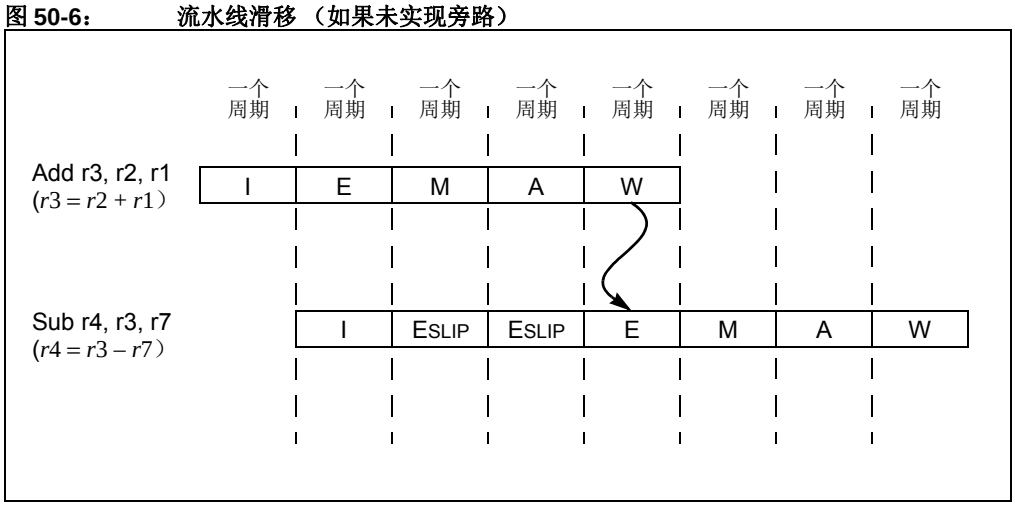
- 停顿 （Stall） —— 这种互锁通过暂停整个流水线来解决。当前正在每个流水线级中执行的所有指令都会受到停顿的影响
- 滑移 （Slip） —— 这种互锁允许流水线中有一部分前移，而流水线的其他部分保持静态

在 PIC32 处理器内核中，所有互锁都以滑移形式进行处理。为了最大程度减少滑移，通过使用称作寄存器旁路的方法 （**50.3.6 “寄存器旁路”** 将介绍该方法）来从其他流水线级抓取结果。

注：	为了说明流水线滑移的概念，图 50-6 中的示例给出了 PIC32 内核未实现寄存器旁路时所发生的情况。

如图 50-6 所示，对于寄存器 r3， sub 指令与前一条 add 指令存在源操作数相依性。 sub 指令会滑移两个时钟，等待 add 的结果回写到寄存器 r3 中。在 PIC32 系列处理器上，不会发生这种滑移情况。

图 50-6： 流水线滑移 （如果未实现旁路）



50.3.6 寄存器旁路

如前面所述， PIC32 处理器实现了一种称作寄存器旁路的机制，帮助减少执行期间的流水线滑移。当某条指令处于流水线 E 级时，只有在操作数可用的情况下，该指令才能继续。如果指令的某个源操作数需要通过执行流水线中的另一条指令计算获得，通过寄存器旁路可以产生一条快捷路径，直接从流水线获取源操作数。处于 E 级的指令可以从在流水线 M 级或 A 级中执行的另一条指令处获取源操作数。如图 50-7 所示，三条指令组成的序列存在相依性，但它们在执行期间完全未发生滑移。该示例使用了 A 级至 E 级和 M 级至 E 级的寄存器旁路。图 50-8 显示了一条利用 A 级至 E 级旁路的装载指令的操作。因为装载指令的结果直到流水线 A 级才可用，所以不需要 M 级至 E 级旁路。

寄存器旁路的性能优点是，即使存在寄存器相依性，ALU 运算的指令吞吐率也可以上升为每个时钟一条指令。

图 50-7： IU 流水线 M 级至 E 级旁路

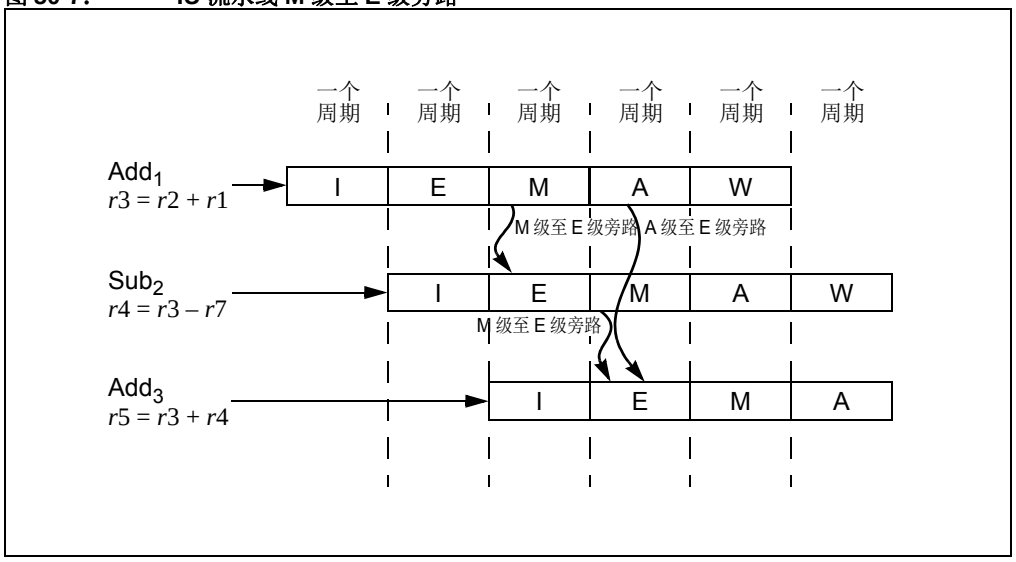
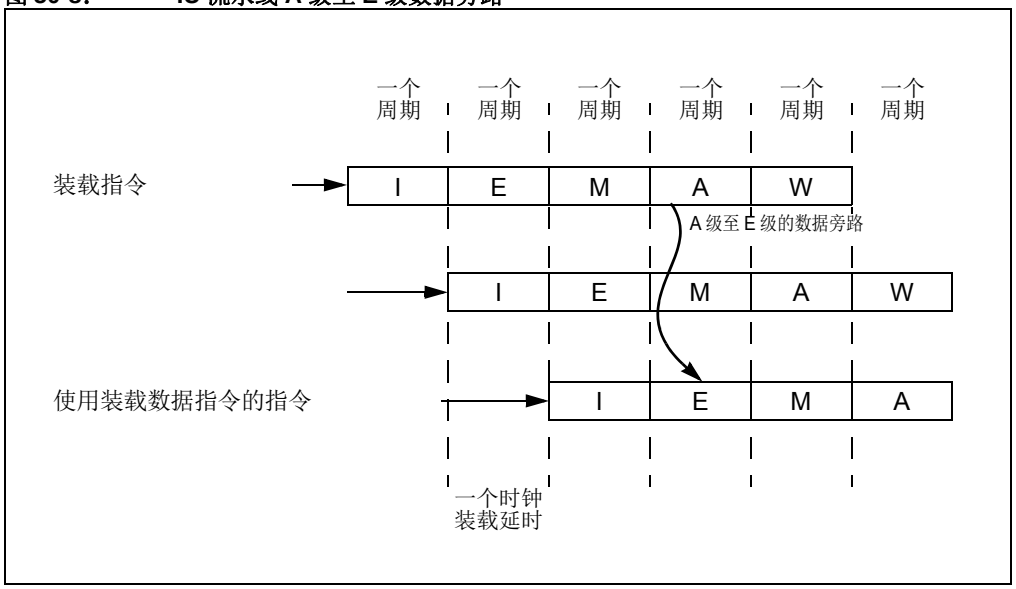


图 50-8： IU 流水线 A 级至 E 级数据旁路



© 2013-2016 Microchip Technology Inc.

50.4 写入 CP0 寄存器时的特殊注意事项

通常，PIC32 内核会确保指令按照完全顺序性的程序模型执行。程序中的每条指令都可以看到前一条指令的结果，但也存在一些背离该模型的情况。这些背离情况称为“冒险”（hazard）。

在特权软件中，存在两种类型的冒险：

- 执行冒险
- 指令冒险

50.4.1 执行冒险

执行冒险在执行一条指令时产生，由另一条指令在执行时看到。表 50-2 列出了这些执行冒险。

表 50-2： 执行冒险

产生方	看见方	冒险源自	间距（指令数）
LL	MFC0	LLAddr	1
MTC0	依赖于 CU0 位 （Status<28>）新值的协处理器指令执行	CU0 位 （Status<28>）	1
MTC0	ERET	EPC、 DEPC 和 ErrorEPC	1
MTC0、 EI 和 DI	被中断的指令	IE 位 （Status<0>）	1
MTC0	被中断的指令	IP1 和 IP0 位 （Cause<1> 和 <0>）	3
MTC0	TLBR、 TLBWI 和 TLBWR	EntryHi	1
MTC0	TLBP 和受新状态影响的装载 / 存储	ASID<7:0> 位 （EntryHi<7:0>）	1
MTC0	TLBWI 和 TLBWR	Index	1
MTC0	RDPGPR 和 WRPGPR	PSS<3:0> 位 （SRSCtl<9:6>）	1
MTC0	未看到内核定时器中断的指令	清除内核定时器中断的比较更新	4
MTC0	受更改影响的指令	任何其他 CP0 寄存器	2

50.4.2 指令冒险

指令冒险在执行一条指令时产生，由另一条指令在取指时看到。表 50-3 列出了这些指令冒险。

表 50-3： 指令冒险

产生方	看见方	冒险源自	间距（指令数）
TLBWR 和 TLBWI	使用新的 TLB 条目的取指操作	TLB 条目	3
MTC0	看到新值的取指操作 （包括对 ERL 进行更改，然后从 useg 段取指）	Status	
MTCO	看到新值的取指操作	ASID<7:0> 位 （EntryHi<7:0>）	3
MTC0	看到新值的取指操作	WatchHi 和 WatchLo	1
MTC0	被中断的指令	IP1 和 IP0 位 （Cause<1> 和 <0>）	2
通过高速缓存进行的指令流写操作	看到新指令流的取指操作	高速缓存条目	3
通过存储进行的指令流写操作	看到新指令流的取指操作	高速缓存条目	取决于系统

50.5 MIPS32 架构

MIPS32 架构基于固定长度且常规编码的指令集，使用装载 / 存储数据模型。其流水线架构支持高级语言的优化执行。算术和逻辑运算采用 3 操作数形式，使编译器可以优化复杂的表达式。提供 32 个通用寄存器，使编译器可以通过将频繁访问的数据存放于寄存器中，从而进一步优化代码生成的性能。

欲了解更多相关信息和文档，请参见 MIPS32 架构网页：

http://www.imgtec.com/mips/architectures/mips32.asp

50.5.1 版本 2 架构

具有 microAptiv 内核的 PIC32 器件采用了 MIPS32 处理器的版本 2 架构，实现了以下版本 2 特性：

- 使用外部至内核的中断控制器实现向量中断，能够使中断直接跳转到该中断的处理程序
- 可编程异常向量基址，对于在 $Status_{BEV}$ 为 0 时发生的异常，允许移动异常向量的基址。该特性使所有系统都可以将异常向量放在适合于系统环境的存储器中
- 原子级中断允许 / 禁止，增加了两条指令，可在原子级允许 / 禁止中断，并返回 Status 寄存器的先前值
- 对于功耗高度敏感的应用，可以禁止 Count 寄存器
- GPR 影子寄存器，提供了附加的 GPR 影子寄存器，并且可以将这些寄存器与向量中断或异常绑定
- Field、 Rotate 和 Shuffle 指令，另外增加了处理寄存器位域的功能
- 显式冒险管理，提供了一组指令来显式地管理冒险，代替基于周期的 SSNOP 冒险处理方法

50.5.2 版本 5 架构

采用 M-Class 内核的 PIC32 器件除了实现版本 2 中的所有特性外，还具有下列版本 5 的特性：

- 通过 Config5 寄存器 （CP0 Register 16， Select 5）访问用户模式
- 额外的 MCU ASE 指令：ASET 和 ACLR 用于置 1 和清零原子级 8 位存储单元

© 2013-2016 Microchip Technology Inc.

50.6 CPU 总线

PIC32 器件使用两种不同的 CPU 总线架构（分离式总线或数据 / 指令总线），具体取决于所实现的 CPU 内核。

50.6.1 分离式总线架构

基于 MCU 微处理器内核的 PIC32 器件具有分离式总线架构，使用两个不同的总线来提供并行指令和数据操作。装载和存储操作在执行取指操作的同时发生。这两条总线称为 I 侧总线 （用于为 CPU 输送指令）和 D 侧总线 （用于数据传输）。

在拆分式总线架构中， CPU 在流水线 I 级期间取指。取指请求发送到 I 侧总线，并由系统总线进行处理。根据地址，系统总线将执行以下操作之一：

- 将取指请求转发到预取高速缓存单元 （如果可用）
- 将取指请求转发到 DRM 单元，或
- 产生异常

无论取指地址为何，取指操作总是使用 I 侧总线。

D 侧总线负责处理由 CPU 执行的所有装载和存储操作。在执行装载或存储指令时，该请求将通过 D 侧总线传给系统总线。该操作在流水线 M 级期间发生，并传给以下几个目标器件之一：

- 数据 RAM
- 预取高速缓存 / 闪存
- 快速外设总线 （中断控制器、 DMA、调试单元、 USB、以太网和 GPIO 端口）
- 通用外设总线 （UART、 SPI、闪存控制器、 EPMP/EPSP、 TRCC 定时器、输入捕捉、 PWM/ 输出比较、 ADC、双比较、 I^2C、时钟 SIB 和复位 SIB）

50.6.2 数据 / 指令架构

基于 MPU 内核的 PIC32 器件具有一条统一的数据 / 指令总线连接到系统总线。该架构使用一条多层系统总线来提供总线发起器和目标器件之间的多个同时数据事务。

50.7 内部系统总线

PIC32 处理器内部总线将外设与系统总线相连接。系统总线会利用器件中的几条数据路径，将来自不同发起器的总线访问请求传给一组目标，以帮助消除性能瓶颈。

系统总线使用的路径中有一部分是专用路径，而其他路径则由若干个目标共用。

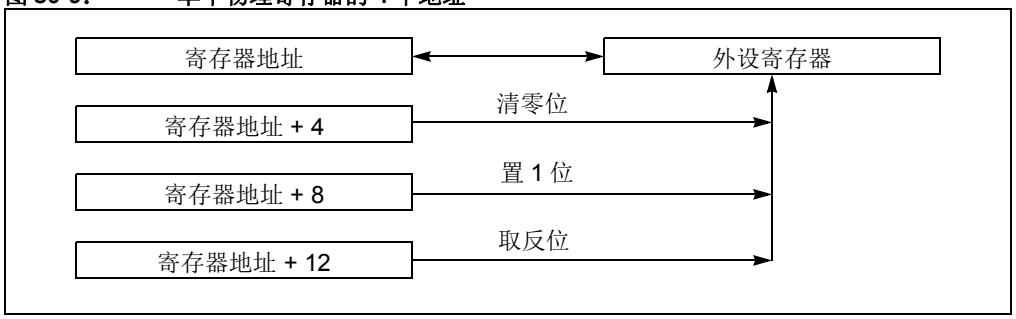
数据 RAM 和闪存读取路径是专用路径，通过它们，对于存储器资源的访问不会被外设总线活动延迟，实现很短的访问延时。高带宽外设位于高速总线上，包括中断控制器、调试单元、DMA 引擎、 USB 主机 / 外设单元和其他高带宽外设 （如， CAN 和以太网引擎）。

不需要高带宽的外设位于独立的外设总线上，以节省功耗。

50.8 置 1/ 清零 / 取反

为了对外设进行单周期位操作，可以根据外设地址，使用三种不同方式访问外设单元中的寄存器。每个寄存器都具有 4 个不同的地址。虽然这 4 个不同地址显现为不同的寄存器，但它们实际上只是同一物理寄存器的 4 种不同寻址方法。

图 50-9： 单个物理寄存器的 4 个地址



基址寄存器地址用于正常的读 / 写访问，而其他三个地址用于提供特殊的只写功能。

- 正常访问
- 置 1 位原子级 RMW 访问
- 清零位原子级 RMW 访问
- 取反位原子级 RMW 访问

外设读取必须通过每个外设寄存器的基址进行。读取置 1/ 清零 / 取反地址的含义是未定义的，对于每个外设可能不同。

对基址执行写操作时，将向外设寄存器中写入完整的值。所有的位都会被写入。例如，假设在写入 0x000000FF 之前，某个寄存器包含 0xAAAA5555。写入之后，寄存器将包含 0x000000FF（假定所有位均为读 / 写位）。

对任意外设寄存器的置 1 地址执行写操作时，只有对于写入 1 的位，目标寄存器中的对应位才会置 1。例如，假设在向置 1 寄存器地址中写入 0x000000FF 之前，某个寄存器包含 0xAAAA5555。写入置 1 寄存器地址之后，外设寄存器的值将包含 0xAAAA55FF。

对任意外设寄存器的清零地址执行写操作时，只有对于写入 1 的位，目标寄存器中的对应位才会清零。例如，假设在向清零寄存器地址中写入 0x000000FF 之前，某个寄存器包含 0xAAAA5555。写入清零寄存器地址之后，外设寄存器的值将包含 0xAAAA5500。

对任意外设寄存器的取反地址执行写操作时，只有对于写入 1 的位，目标寄存器中的对应位才会取反 （或翻转） 。例如，假设在向取反寄存器地址中写入 0x000000FF 之前，某个寄存器包含 0xAAAA5555。写入取反寄存器之后，外设寄存器的值将包含 0xAAAA55AA。

50.9 ALU 状态位

不同于大多数其他 PIC 单片机，PIC32 处理器不使用 Status 寄存器标志。许多处理器中都使用条件标志，帮助在程序执行期间执行判定操作。标志是否置 1 取决于比较操作或一些算术运算的结果。然后，这些单片机上的条件转移指令会根据一组条件代码的值作出判定。

PIC32 处理器改为使用一些指令，这些指令执行比较操作，并将标志或值存储到通用寄存器中。然后，使用该通用寄存器作为操作数来执行条件转移。

© 2013-2016 Microchip Technology Inc.

50.10 中断和异常机制

> **注：** 在本部分中，术语 “精确”和 “不精确”都是用来描述异常的。在精确异常中，可以使用 EPC （CP0，寄存器 14，选择 0）标识导致异常的指令。在不精确异常中，不能标识导致异常的指令。大多数异常均为精确异常。总线错误异常可能是不精确异常。

PIC32 系列处理器实现了高效灵活的中断和异常处理机制。中断和异常的相似之处在于，当前指令流会临时改变，以执行特殊的过程来处理中断或异常。两者的区别在于，中断通常是正常操作产生的结果，而异常则是错误条件 （例如总线错误）产生的结果。

当发生中断或异常时，处理器会执行以下操作：

1. 将要在处理程序返回之后执行的下一条指令的 PC 保存到协处理器寄存器中。
2. 更新 Cause 寄存器，以反映异常或中断的原因。
3. 将 Status 寄存器的 EXL 或 ERL 位置 1，以在内核模式执行。
4. 根据 Ebase 和 OFFSET 值计算处理程序 PC。
5. 自动中断恢复现场可将一些 COP0 状态保存在堆栈中并在准备处理中断时自动更新一些 COP0 寄存器的内容。
6. 处理器从新 PC 处开始执行。

这是对中断和异常机制的简单概述。详细信息，请参见具体器件数据手册的 **“CPU 异常和中断控制器”**章节。

50.11 编程模型

PIC32 系列处理器设计为与高级语言（例如 C 编程语言）配合使用。它支持多种数据类型，并使用高级语言所需的简单而灵活的寻址模式。它有 32 个通用寄存器，以及两个用于乘法和除法的特殊寄存器。

对于 PIC32 处理器上的机器语言指令，有三种不同的格式：

- 立即数 （I 类） CPU 指令
- 跳转 （J 类） CPU 指令，以及
- 寄存器 （R 类） CPU 指令

大多数操作都在寄存器中执行。寄存器类型的 CPU 指令具有三个操作数；即两个源操作数和一个目标操作数。

具有三个操作数和很大的寄存器集可以让汇编语言编程器和编译器高效地使用 CPU 资源。中间结果可以保留在寄存器中，不需要常常从存储器来回传送数据，从而可以得到速度更快和长度更短的程序。

立即数格式指令具有一个立即操作数、一个源操作数和一个目标操作数。跳转指令具有一个 26 位相对指令偏移字段，用于计算跳转目标。

50.11.1 CPU 指令格式

CPU 指令是单个 32 位对齐的字。以下显示了 CPU 指令格式：

- 立即数 （见图 50-10）
- 跳转 （见图 50-11）
- 寄存器 （见图 50-12）

表 50-4 列出了在这些指令中使用的字段。

表 50-4： CPU 指令格式字段

字段	说明
操作码	6 位主操作码。
rd	目标寄存器的 5 位说明符。
rs	源寄存器的 5 位说明符。
rt	目标 （源 / 目标）寄存器的 5 位说明符，或用于指定主操作码 REGIMM 中的功能。
立即数	16 位有符号立即数，用作逻辑操作数、算术有符号操作数、装载 / 存储地址字节偏移量和 PC 相对转移指令的有符号指令位移。
instr_index	26 位索引，左移 2 位来提供跳转目标地址的低 28 位。
sa	5 位移位量。
function	6 位功能字段，用于指定主操作码 SPECIAL 中的功能。

图 50-10： 立即数 （I 类） CPU 指令格式

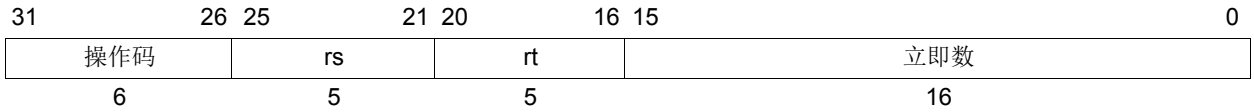


图 50-11： 跳转 （J 类） CPU 指令格式

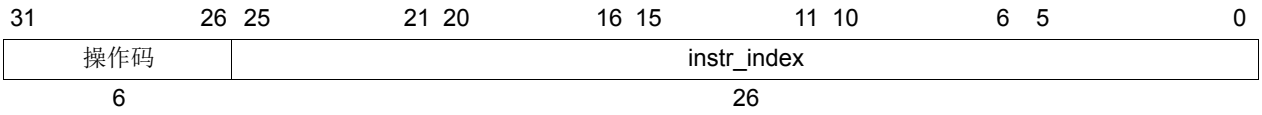
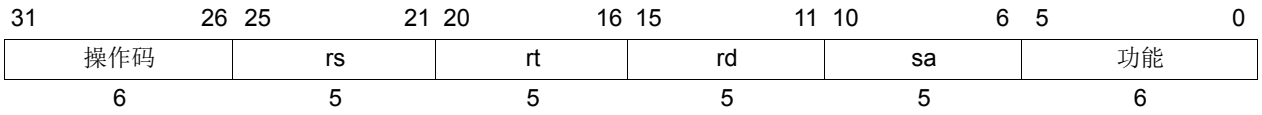


图 50-12： 寄存器 （R 类） CPU 指令格式



© 2013-2016 Microchip Technology Inc.

50.11.2 CPU 寄存器

PIC32 架构定义了以下 CPU 寄存器：

- 32 个 32 位通用寄存器 （GPR）
- 标准 MIPS32 架构定义了一对 HI/LO 累加寄存器 （AC0）。 PIC32 器件中的内核包含 DSP ASE （见**注**），它提供额外的三对 HI/LO 累加寄存器 （AC1、 AC2 和 AC3）。这些寄存器提高了独立累加程序的并行性。针对累加器的 DSP 指令使用两个指令位来指定目标累加器。
- 1 个特殊用途程序计数器 （Program Counter， PC），它只会受一些特定指令间接影响；它在架构中是一个不可见的寄存器。

注：	DSP ASE 并非在所有器件上都可用。请参见具体器件数据手册中的 **“CPU”** 章节以确定可用性。

50.11.2.1 CPU 通用寄存器

CPU 通用寄存器中的两个寄存器具有指定的功能：

- r0—— 此寄存器硬连接到值 0，可以用作结果将被丢弃的任何指令的目标寄存器。在需要值 0 时， r0 也可以用作源寄存器。
- r31—— 它是 JAL、 BLTZAL、 BLTZALL、 BGEZAL 和 BGEZALL 使用的目标寄存器，无需在指令字中明确指定。其他情况下， r31 用作一般寄存器。

其他寄存器可供一般用途使用。

50.11.2.2 寄存器约定

虽然 PIC32 架构中的大部分寄存器都指定为通用寄存器（如表 50-5 所示），但为了通过高级语言正确执行软件操作（例如 Microchip MPLAB® XC32 C/C++ 编译器），建议按下表方式使用寄存器。

表 50-5： 寄存器约定

CPU 寄存器	符号寄存器	用法
r0	zero	总是为 0 （见**注 1**）
r1	at	汇编器临时量
r2 - r3	v0-v1	函数返回值
r4 - r7	a0-a3	函数参数
r8 - r15	t0-t7	临时量 —— 调用函数不需要保留内容
r16 - r23	s0-s7	保存的临时量 —— 调用函数须保留内容
r24 - r25	t8-t9	临时量 —— 调用函数不需要保留内容
r26 - r27	k0-k1	内核临时量 —— 用于中断和异常处理
r28	gp	全局指针 —— 用于快速访问常用数据
r29	sp	堆栈指针 —— 软件堆栈
r30	s8 或 fp	保存的临时量 —— 调用函数必须保留内容，*或者* 帧指针 —— 指向堆栈中过程帧的指针
r31	ra	返回地址 （见**注 1**）

注 1： 硬件强制实现，不仅仅是约定。

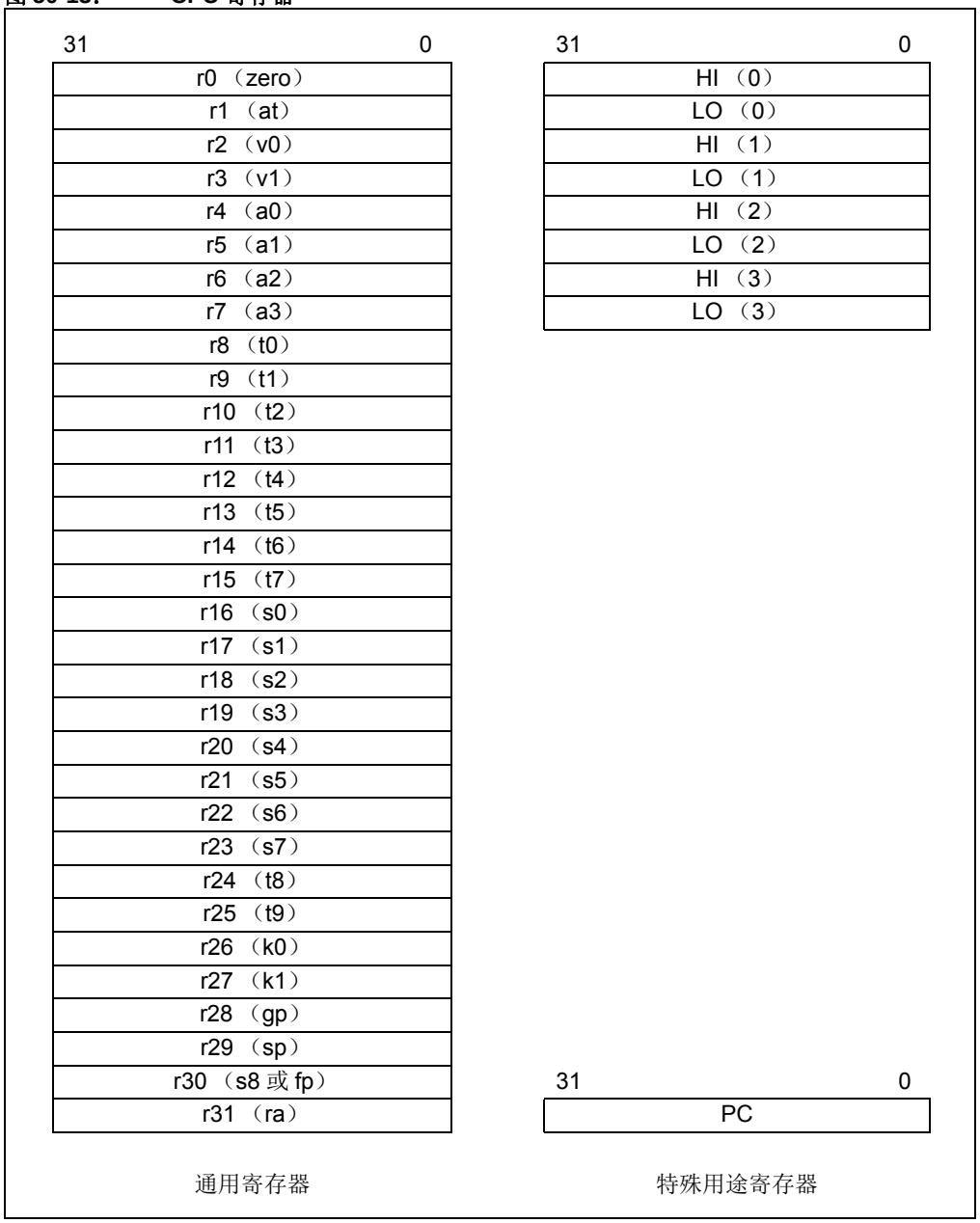
50.11.2.3 CPU 特殊用途寄存器

CPU 包含以下特殊用途寄存器：

- PC—— 程序计数器寄存器
- AC0 到 AC3——64 位累加寄存器对 （HI/LO）：
  - HI/LO—— 乘法和除法寄存器对 （高位字结果和低位字结果）：
    - 在乘法运算期间， HI 和 LO 寄存器存储整数乘法的乘积
    - 在乘加或乘减运算期间， HI 和 LO 寄存器存储整数乘加或乘减的结果
    - 在除法运算期间， HI 和 LO 寄存器存储整数除法的商 （在 LO 中）和余数 （在 HI 中）
    - 在乘法累加运算期间， HI 和 LO 寄存器存储运算的累加结果

图 50-13 给出了 CPU 寄存器的布局。

图 50-13： CPU 寄存器



© 2013-2016 Microchip Technology Inc.

表 50-6： microMIPS 16 位指令寄存器用法

16 位寄存器编码	32 位 MIPS 寄存器编码	符号名称	说明
0	16/0	s0/zero	通用寄存器
1	17	s1	通用寄存器
2	2	v0	通用寄存器
3	3	v1	通用寄存器
4	4	a0	通用寄存器
5	5	a1	通用寄存器
6	6	a2	通用寄存器
7	7	a3	通用寄存器
N/A	28	gp	microMIPS 隐式引用的通用指针寄存器
N/A	29	sp	microMIPS 隐式引用的堆栈指针寄存器
N/A	31	ra	microMIPS 隐式引用的返回地址寄存器

表 50-7： microMIPS 特殊寄存器

符号名称	用途
PC	程序计数器。 PC 相对寻址指令可以将该寄存器作为操作数进行访问。
HI	包含乘法或除法结果的高位字。
LO	包含乘法或除法结果的低位字。

50.11.3 如何实现堆栈 /MIPS 调用约定

PIC32 CPU 没有硬件堆栈。处理器依赖软件来提供此功能。由于硬件本身不会执行堆栈操作，因此系统中必须存在相关约定，让系统中的所有软件可以使用相同的机制。例如，堆栈可以向低地址方向延伸，也可以向高地址方向延伸。如果一段软件代码假定堆栈向低地址方向延伸，并调用了假定堆栈向高地址方向延伸的程序，则堆栈会损坏。

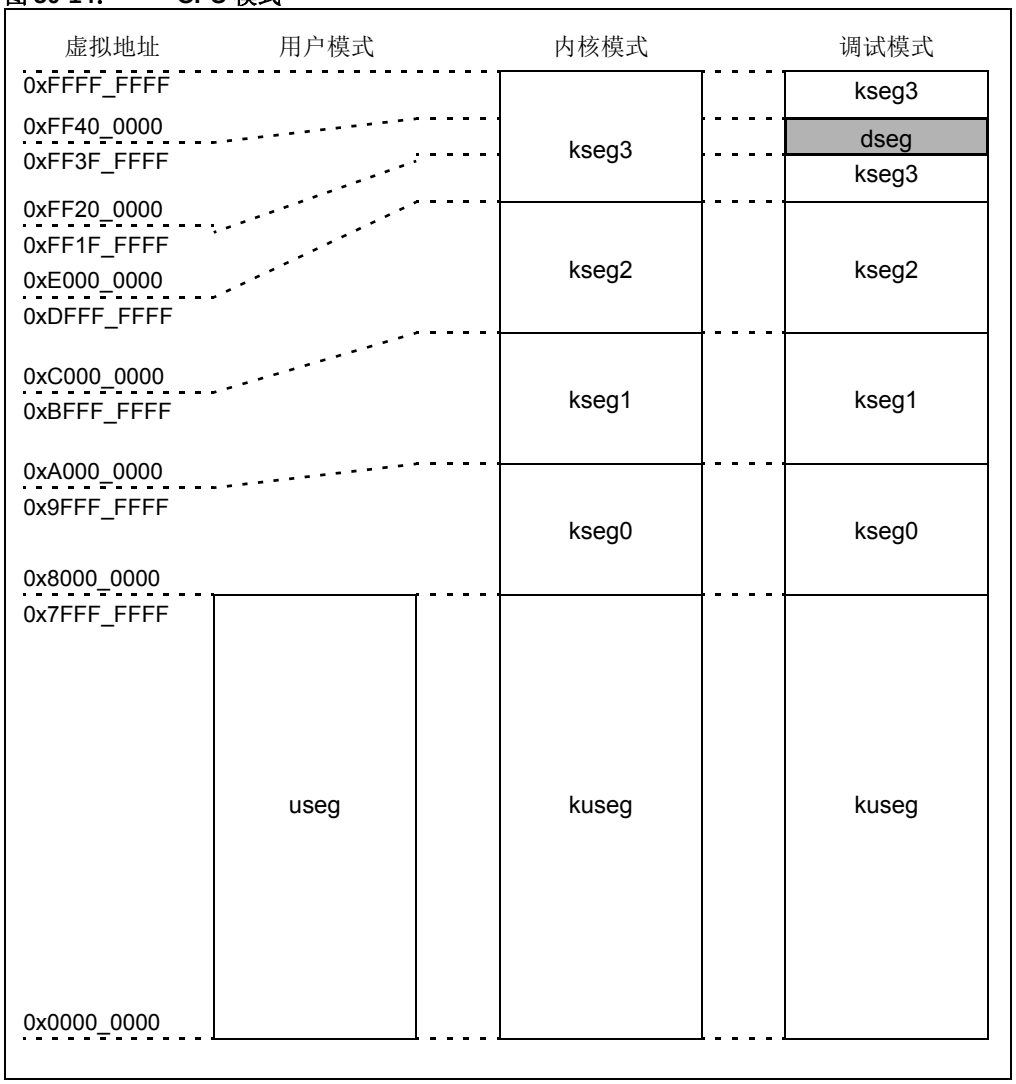
使用系统范围的调用约定可以防止发生该问题。 Microchip MPLAB® XC32 C/C++ 编译器假定堆栈向低地址方向延伸。

50.11.4 处理器模式

PIC32 系列 CPU 中有两种工作模式和一种特殊的执行模式；即用户模式、内核模式和调试模式。处理器开始时以内核模式执行，如果需要，可以停留在内核模式下进行正常操作。用户模式是一种可选模式，系统设计人员可以利用该模式来划分特权和非特权软件代码。调试模式通常仅供调试器或监视器使用。

这些工作模式之间的一个主要区别是允许软件访问的存储器地址。外设在用户模式下无法访问。图 50-14 给出了每种模式的不同存储器映射。关于处理器的存储器映射的更多信息，请参见《PIC32 系列参考手册》中的**第 3 章 “存储器构成”**（DS60001115）。

图 50-14： CPU 模式



© 2013-2016 Microchip Technology Inc.

50.11.4.1 内核模式

为了访问许多硬件资源，处理器必须工作于内核模式。在内核模式下，软件可以访问处理器的整个地址空间，以及访问特权指令。

当 Debug 寄存器中的 DM 位为 0，并且 Status 寄存器包含一个或多个以下值时，处理器工作于内核模式：

- UM = 0
- ERL = 1
- EXL = 1

检测到非调试异常时，EXL 或 ERL 将置 1，处理器将进入内核模式。在异常处理程序末尾，通常会执行异常返回 （ERET）指令。 ERET 指令会跳转到异常 PC （EPC 或 ErrorPC，具体取决于异常）、清零 ERL，并且如果 ERL= 0 则清零 EXL。

如果 UM = 1，则在 ERL 和 EXL 重新清零 0，并从异常返回之后，处理器会恢复为用户模式。

50.11.4.2 用户模式

在用户模式下执行时，软件仅限于使用部分处理器资源。在许多情况下，都希望将应用程序级别的代码保持在用户模式下运行，在此模式下可控制所发生的错误，不允许错误影响内核模式代码。

应用程序可以通过受控接口 （例如 SYSCALL 机制）访问内核模式功能。

如图 50-14 所示，用户模式软件可以访问 USEG 存储区。

要工作于用户模式， Status 寄存器必须包含以下每个位值：

- UM = 1
- EXL = 0
- ERL = 0

50.11.4.3 调试模式

调试模式是处理器的一种特殊模式，通常仅供调试器和系统监视器使用。通过调试异常进入调试模式，在此模式下可以访问所有内核模式资源，以及用于调试应用程序的所有特殊硬件资源。

当 Debug 寄存器中的 DM 位为 1 时，处理器处于调试模式。

退出调试模式的方法通常是在调试处理程序中执行 DERET 指令。

© 2013-2016 Microchip Technology Inc.

50.12 浮点单元（FPU）

采用 M-Class 内核的 PIC32 器件包含了实现 MIPS 版本 5 指令集架构的浮点单元（FPU），用于进行浮点计算。

注： 该模块并非在所有器件上都可用。要确定可用性，请参见具体器件数据手册中的**“CPU”**章节。

50.12.1 特性

该模块的一些重要特性包括：

- PIC32 实现支持 *“IEEE Standard for Binary Floating-Point Arithmetic”*（ANSI/IEEE 754 标准）的单精度和双精度数据格式。更多信息，请参见 **50.12.6.5 “IEEE 754-1985 标准”**。
- 寄存器文件和功能单元中都实现了全 64 位操作。FPU 具有 32 个 64 位浮点寄存器，用于所有的浮点操作。
- 32 位的浮点控制寄存器用于控制 FPU 的操作，以及监视条件代码和异常条件
- 单元的性能针对单精度格式进行了优化。大多数指令都具有 1 个 FPU 周期的吞吐率和 4 个 FPU 周期的延时。
- FPU 实现了在乘法功能后进行中间舍入的复合乘加 （MADD）和乘减 （MSUB）指令。其结果保证与执行 MUL 后接着执行 ADD/SUB 指令的结果相同，但指令延时、取指、分派带宽和寄存器访问总数都改善了。
- 对于一些指令，硬件支持 IEEE 非规格化输入操作数和结果。提供了快速清除为零模式，以优化 IEEE 非规格化结果的性能。快速清除为零模式必须通过 FPU 控制寄存器使能，建议在生成非规格化结果时使用该模式，以获得最佳性能。
- MIPS® 架构（见**注**）规定了其他一些 IEEE 754 标准未规定的算术运算（例如，倒数和平方根倒数）， FPU 实现了这些算术运算。为了实现低延时计数，这些指令可以满足更宽松的精度要求。

注： 关于 MIPS 架构的信息，请参见 Imagination Technologies Ltd. 网站 www.imgtec.com 。

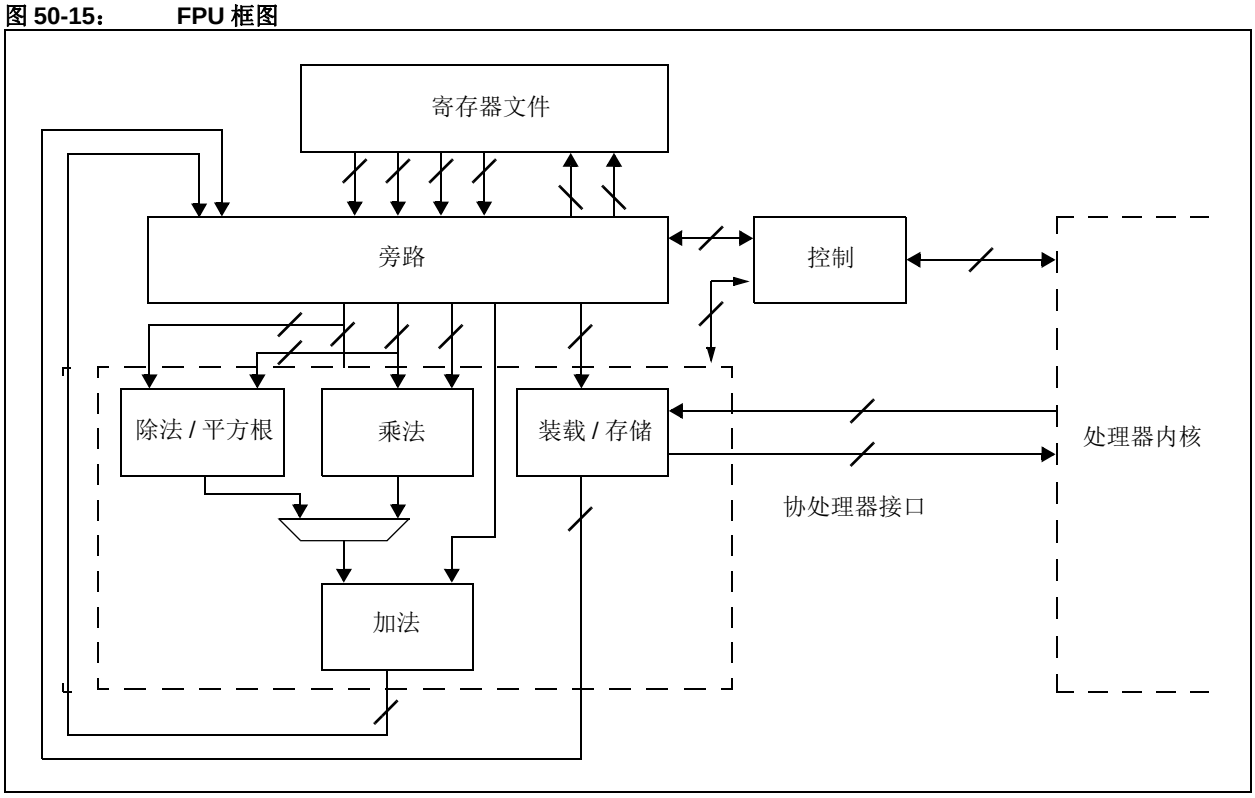
- MIPS FPU 架构设计为可以使用硬件和软件的组合来实现架构。PIC32 FPU 可以操作处于特定范围内的数字 （IEEE 规格化数字），但它依赖软件处理程序来操作 FPU 硬件无法处理的数字（IEEE 非规格化数字）。
- FPU 具有用于浮点指令执行的独立流水线。该流水线与整数内核流水线并行工作，且在整数流水线停顿时不停顿。这允许长时间运行的 FPU 运算 （例如除法或平方根）被系统停顿和 / 或其他整数单元指令部分屏蔽。
- FPU 访问是通过协处理器 1 提供的。与主处理器内核一样，协处理器 1 使用装载 / 存储指令集来编程和操作。处理器内核使用专用的协处理器接口与协处理器 1 进行通信。FPU 作为一个自治单元工作。硬件是完全互锁的，从而在编写软件时，编程人员可以不必担心在装载之后和依赖指令之间插入延时时隙。
- 算术指令总是按顺序分派与完成，但装载和存储可以不按顺序完成。异常模型始终是 “精确的”。

关于相关 FPU 寄存器的信息，请参见 **50.14 “协处理器 1 （CP1）寄存器”**。

图 50-15 给出了 PIC32 FPU 的框图。

© 2013-2016 Microchip Technology Inc.

图 50-15： FPU 框图



50.12.2 FPU 数据格式

FPU 支持 IEEE 754 标准规定的单精度和双精度浮点数据类型。

此外，还支持定点数据类型：MIPS 架构提供的有符号整数。

50.12.2.1 浮点格式

PIC32 FPU 支持以下两种浮点格式：

- 32 位单精度浮点 （类型 S，如图 50-16 所示）
- 64 位双精度浮点 （类型 D，如图 50-17 所示）

单精度和双精度浮点数据类型包含 3 个位域：符号、指数和小数。

它们表示数字值，以及以下特殊实体：

- 两种无穷大：+∞ 和 -∞
- 信号非数 （SNaN）
- 静默非数 （QNaN）
- 以下形式的数字：$(-1)^s\ 2^E$ b0.b1 b2..bp-1，其中：
  - s = 0 或 1
  - E = 介于 E_min 和 E_max 之间的任意整数 （含两端）
  - bi = 0 或 1 （高位 b0 位于二进制小数点的左侧）
  - p 是符号量值的精度

表 50-8 列出了架构支持的单精度和双精度数字的大小。

表 50-8： 浮点数据类型的参数

参数	单精度	双精度
尾数精度 p 的位数	24	53
最大指数 E_max	+127	+1023
最小指数 E_min	-126	-1022
指数偏移量	+127	+1023
指数位域 e 的位数	8	11
b0 整数位的表示形式	隐藏	隐藏
小数位域 f 的位数	23	52
以位为单位的格式总宽度	32	64
最大可表示数字的大小	3.4028234664e+38	1.7976931349e+308
最小规格化可表示数字的大小	1.1754943508e-38	2.2250738585e-308

图 50-16： 单精度浮点格式 （S）

S	指数 <0:7>	小数 <0:22>

图 50-17： 双精度浮点格式 （D）

S	指数 <0:10>	小数 <0:51>

图 50-16 和图 50-17 中的位域为：

- 1 位符号 S
- 偏移后的指数 e = E + 偏移量
- 二进制小数 f =.b1 b2..bp-1 （b0 位是隐藏的；它不会被记录）

值使用表 50-9 中列出的无偏移指数、小数和符号值以指定格式进行编码。

对于 NaN，小数位域的高位 （标识为 b1）还具有特殊的重要性。

© 2013-2016 Microchip Technology Inc.

表 50-9： 单精度或双精度浮点数据类型编码

无偏移的 E	f	s	b1	值 V	值的类型	典型单精度位模式的值（见注 1）	典型双精度位模式的值
E_max+1	≠ 0		1	SNaN	信号 NaN（$FCSR_{NAN2008}$ = 0）	0x7FFFFFFF	0x7FFFFFFF FFFFFFFF
			0	QNaN	静默 NaN（$FCSR_{NAN2008}$ = 0）	0x7FBFFFFF	0x7FF7FFFF FFFFFFFF
E_max+1	≠ 0		1	QNaN	静默 NaN（$FCSR_{NAN2008}$ = 1）	0x7FFFFFFF	0x7FFFFFFF FFFFFFFF
			0	SNaN	信号 NaN（$FCSR_{NAN2008}$ = 1）	0x7FBFFFFF	0x7FF7FFFF FFFFFFFF
E_max+1	0	1		-∞	负无穷大	0xFF800000	0xFFF00000 00000000
		0		+∞	正无穷大	0x7F800000	0x7FF00000 00000000
E_max 至 E_min		1		$-(2^E)(1.f)$	规格化负数	0x80800000 至 0xFF7FFFFF	0x80100000 00000000 至 0xFFEFFFFF FFFFFFFF
		0		$+(2^E)(1.f)$	规格化正数	0x00800000 至 0x7F7FFFFF	0x00100000 00000000 至 0x7FEFFFFF FFFFFFFF
E_min-1	≠ 0	1		$-(2^{E_min})(0.f)$	非规格化负数	0x807FFFFF	0x800FFFFF FFFFFFFF
		0		$+(2^{E_min})(0.f)$	非规格化正数	0x007FFFFF	0x000FFFFF FFFFFFFF
E_min-1	0	1		-0	负零	0x80000000	0x80000000 00000000
		0		+0	正零	0x00000000	0x00000000 00000000

注 1： NaN 和非规格化值的位模式的 “典型” 性质反映了这种情况：符号可能具有任一值 （NaN），小数位域可能具有任意非零值 （两者）。因此，所显示的位模式是表示这些特殊值的一类可能值中的一个。

50.12.2.1.1 规格化和非规格化数字

可以表示为单精度或双精度数据类型的每个非零数字值都只有一种编码。它称为规格化形式。

p 位尾数的高位 （位于二进制小数点的左侧）是 “隐藏的”，不会在小数位域中被记录。该位的值可以通过查看指数的值确定：

- 当无偏移指数处于E_min至E_max范围内（含两端）时，数字是规格化的，该隐藏位必须为1。
- 如果数字值由于指数小于 E_min 而无法规格化，则其表示形式是非规格化的，编码数字的指数为 E_min - 1，该隐藏位的值为 0。

请注意，正零和负零属于不视为非规格化值的特殊情况。

50.12.2.1.2 无穷大和 NaN

在一些特定条件下 （如未使用初始化的变量、违反数学规则或结果无法表示），浮点操作会发出属于 IEEE 754 标准一部分的异常条件。

通常，在遇到异常条件时，程序会产生陷阱。但是，可以使用另一种方法，就是在计算遇到任何这些条件时不产生陷阱而继续进行，但生成指示在计算过程中产生了异常条件的结果。为了支持这种行为，每种浮点格式都定义了 IEEE 754 标准要求的正无穷大（+∞）、负无穷大（-∞）、静默非数 （QNaN）和信号非数 （SNaN）的表示形式。关于这些值，请参见表 50-9。

50.12.2.2 无穷大算术运算

无穷大表示一个由于量值太大而无法使用给定格式表示的数字。在计算过程中，它代表量值上溢。在被零除运算和 **50.12.5 “浮点异常概述”** 所述的一些上溢情况中，将生成带有正确符号的 +∞ 或 -∞ 作为默认结果。

在作为默认结果创建时， ∞ 可以成为后续运算中的操作数。其顺序为 -∞< （每个有限数字） <+∞。对 ∞ 的算术运算是对任意大操作数的实数算术运算的极限情况，前提是存在这种极限。在这些情况下，认为对 ∞ 的算术运算是精确的，不会产生异常条件。由 ∞ 表示的超出范围指示通过后续计算传播。

某些情况下，对操作数 ∞ 的实数算术运算不存在有意义的极限情况。这些情况下会抛出无效操作异常条件，如 **50.12.3 “通用浮点寄存器”** 所述。

50.12.2.2.1 信号非数 （SNaN）

对于算术运算，SNaN 操作数会导致无效操作异常。SNaN 是用于放入未初始化变量的值。SNaN 永远不会作为结果值产生。

MIPS 架构将格式化操作数传送指令 （`MOV.fmt`、 `MOVT.fmt`、 `MOVF.fmt`、 `MOVN.fmt` 和 `MOVZ.fmt`）定为非算术指令；它们不会发出 IEEE 754 标准异常。

50.12.2.2.2 静默非数 （QNaN）

QNaN 提供从无效或不可用数据和结果传播的诊断信息。这种传播要求在算术运算和浮点格式转换过程中保留 QNaN 中包含的信息。

对 QNaN 操作数的算术运算不会发出异常。在要提供浮点结果时，QNaN 操作数会导致算术运算提供 QNaN 结果。在可行时，该 QNaN 结果是操作数 QNaN 值之一。

对于不提供浮点结果的运算 （即，比较运算）， QNaN 的效果与 SNaN 类似。

执行某些不涉及 QNaN 操作数的无效操作且未使能陷阱时，会生成一个新的 QNaN 值。表 50-10 列出了在没有输入操作数 QNaN 值可复制时生成的 QNaN 值。对于定点格式列出的值是为了满足当 QNaN 或无穷大浮点值被转换为定点值时的 IEEE 754 标准而提供的值。架构的所有其他功能都不会检测或使用这些 “整数 QNaN” 值。

表 50-10： 生成新 QNaN 时提供的值

格式	新的 QNaN 值 （FCSR$_{NAN2008}$ = 0）	新的 QNaN 值 （FCSR$_{NAN2008}$ = 1）
单精度浮点	0x7FBF FFFF	0x7FFF FFFF
双精度浮点	0x7FF7 FFFF FFFF FFFF	0x7FFF FFFF FFFF FFFF
字定点	0x7FFF FFFF	0x7FFF FFFF
长字定点	0x7FFF FFFF FFFF FFFF	0x7FFF FFFF FFFF FFFF

50.12.2.3 定点格式

PIC32 FPU 提供了两种定点数据类型，它们是由 MIPS 架构提供的有符号整数：

- 32 位字定点格式 （类型 W）
- 64 位长字定点格式 （类型 L）

定点值以二进制补码的形式保存，用于 CPU 中的有符号整数。架构不提供无符号定点数据类型；应用程序软件可以基于现有的指令和数据类型合成用于无符号整数的计算。

© 2013-2016 Microchip Technology Inc.

50.12.3 通用浮点寄存器

FPU 是带有 64 位浮点通用寄存器（Floating Point General Registers， FPGR）的 64 位浮点单元，但也支持 32 位寄存器模式，用于保持 MIPS 向后兼容性。 CP0 状态寄存器中的 FR 位（CP0<26>）决定选择哪种模式：

- 当 FR 位为 1 时，FPU 处于 FR64 模式并使用 64 位寄存器模型，它会定义 32 个 64 位寄存器，寄存器中支持所有格式。
- 当 FR 位为 0 时，FPU 处于 FR32 模式并使用 32 位寄存器模型，它会定义 32 个 32 位寄存器，双精度格式值存储在偶编号 - 奇编号寄存器对中；寄存器文件也可视为具有 16 个 64 位寄存器。以这种方式配置时，双精度操作存在几项限制：
  - 任何指定奇编号寄存器作为源或目标的双精度操作都会导致保留指令异常
  - 访问奇编号 FPU 寄存器的 MTHC1/MFHC1 指令会发出保留指令异常。

50.12.3.1 FPR 寄存器和格式化操作数的结构

对格式化操作数值进行操作的 FPU 指令会指定存放值的浮点寄存器（Floating Point Register，FPR）。只有 32 位宽（W 和 S 格式）的操作数仅使用 FPR 中的一半空间。关于 FPR 构成和操作数数据在其中的存储方式，请参见图 50-18 和图 50-19。

图 50-18： FPR 中的单精度浮点或字定点操作数

FPR<32:63>—— 未定义 / 未用	FPR<0:31>—— 数据字

图 50-19： FPR 中的双精度浮点或长字定点操作数

FPR<0:63>—— 数据双字 / 长字

50.12.3.2 FPR 寄存器中使用的值的格式

不同于 CPU， FPU 既不解析源操作数的二进制编码，也不产生每个操作结果的二进制编码。在 FPR 中存放的值具有格式或类型，它只能由操作该格式的指令使用。FPR 中的值的格式是有效数字格式之一：单精度或双精度浮点，以及字或长字定点。否则，该值是未解析或未知的。

FPR 中的值总是在当某个值如下写入寄存器时设置：

- 当数据传输指令将二进制数据写入 FPR （LWC1、 LDC1 或 MTC1 指令的装载目标）时， FPR 接收未解析的二进制值。
- 产生 *fmt* 类型结果的计算或 FP 寄存器传送指令将一个 *fmt* 类型的值放入结果寄存器。

当需要 *fmt* 格式值的指令使用具有未解析值的 FPR 作为源操作数时，二进制内容会被解析为 *fmt* 格式的编码值， FPR 中的值会变为 *fmt* 格式的值。二进制内容不能以不同的格式重新解析。

如果某个 FPR 包含 *fmt* 格式的值，则计算指令一定不能使用该 FPR 作为不同格式的源操作数。如果发生这种情况，寄存器中的值会变为未知的，指令的结果也是未知的值。使用包含未知值的 FPR 作为源操作数会产生具有未知值的结果。

当数据传输指令读取 FPR（存储，即 SWC1、SDC1 或 MFC1 指令的源操作数）时， FPR 中值的格式不变。数据传输指令会生成 FPR 中包含的值的二进制编码。如果 FPR 中的值是未知的，则由操作生成的编码二进制值是未定义的。

50.12.3.3 32 位和 64 位二进制数据传输

数据传输指令会在 FPU FPR 和系统之间传送字和双字。

图 50-20 和图 50-21 分别显示了字和双字装载和传送至指令的操作。

存储和传送自指令执行相反的操作，也即从相应的装载或传送至指令已写入的单元读取数据。

图 50-20： FPU 字装载和传送至操作

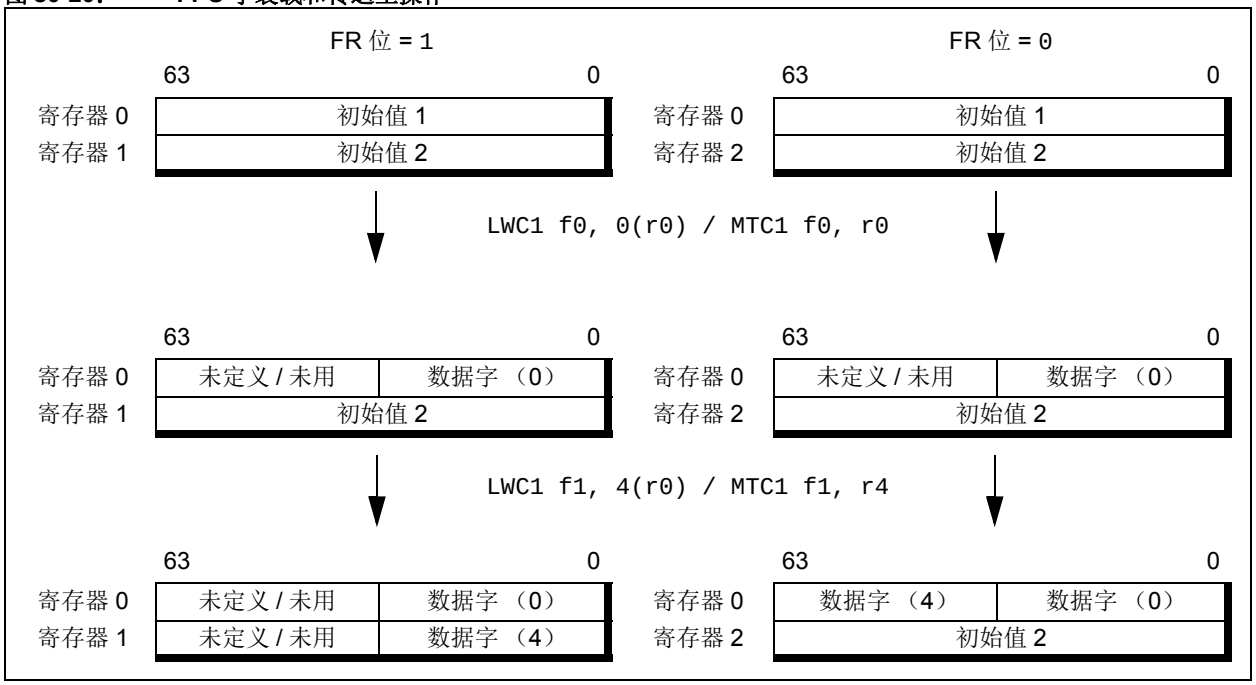
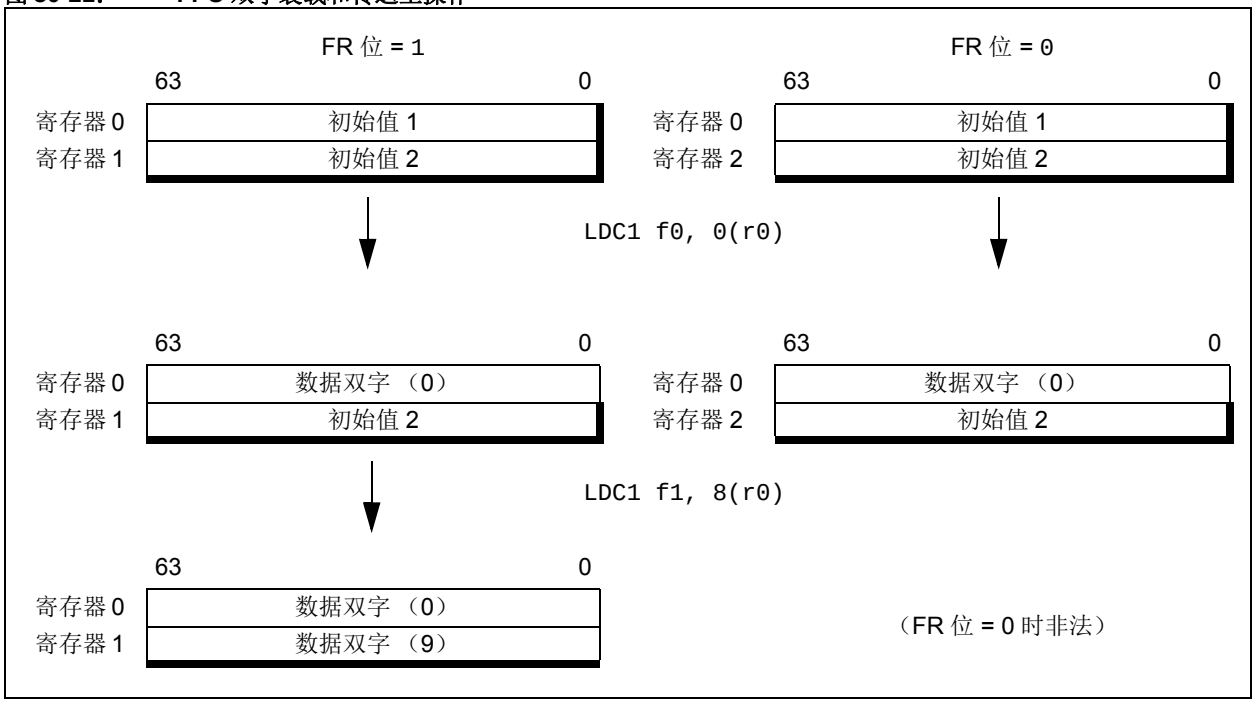


图 50-21： FPU 双字装载和传送至操作



© 2013-2016 Microchip Technology Inc.

50.12.4 浮点指令概述

FPU 指令分为以下类别：

- FPU 数据传输指令
- FPU 算术指令
- FPU 转换指令
- FPU 格式化操作数值传送指令
- FPU 条件转移指令
- 其他 FPU 指令

"MIPS32® M5150 Processor Core Family Software User's Manual" 中的**第 14 章 "M5150 Processor Core Instructions"** 详细介绍了这些指令。注册用户可以从 Imagination Technologies Ltd. 网站 （www.imgtec.com）下载该文档。

50.12.4.1 FPU 数据传输指令

FPU 具有两个寄存器集：协处理器通用寄存器 （Coprocessor General Register， CGR）和协处理器控制寄存器（Coprocessor Control Registers，CCR）。FPU 具有装载 / 存储架构：所有计算都是对协处理器通用寄存器中存放的数据执行的。

控制寄存器用于控制 FPU 操作。数据在寄存器和系统其余部分之间使用专用的装载、存储和传送指令来传输。传输的数据会被视为未格式化的二进制数据。不会执行任何格式转换，因此不会产生任何 IEEE 浮点异常。

表 50-11： FPU 数据传输指令

传输方向			传输的数据
FPU 通用寄存器	<->	存储器	字 / 双字装载 / 存储
FPU 通用寄存器	<->	CPU 通用寄存器	字传送
FPU 控制寄存器	<->	CPU 通用寄存器	字传送

所有协处理器装载和存储都是对自然对齐的数据项进行操作。尝试对数据项非自然对齐的地址执行装载或存储会导致地址错误异常。字或双字的地址是对象中的最小字节地址。对于 PIC32 架构，它是最低字节。

50.12.4.2 FPU 数据传输指令寻址

FPU 具有使用与 CPU 所用相同的寄存器 + 偏移量寻址方式的装载和存储指令。此外，仅对于 FPU，还具有使用寄存器 + 寄存器寻址方式的装载和存储指令。

表 50-12 和表 50-13 列出了 FPU 数据传输指令。

表 50-12： FPU 装载和存储指令

助记符	指令	寻址模式
LDC1	将双字装载到浮点	寄存器 + 偏移量
LWC1	将字装载到浮点	寄存器 + 偏移量
SDC1	从浮点存储双字	寄存器 + 偏移量
SWC1	从浮点存储字	寄存器 + 偏移量
LDXC1	使用变址方式将双字装载到浮点	寄存器 + 寄存器
LUXC1	使用变址方式将非对齐双字装载到浮点	寄存器 + 寄存器
LWXC1	使用变址方式将字装载到浮点	寄存器 + 寄存器
SDXC1	使用变址方式从浮点存储双字	寄存器 + 寄存器
SUXC1	使用变址方式从浮点存储非对齐双字	寄存器 + 寄存器
SWXC1	使用变址方式从浮点存储字	寄存器 + 寄存器

表 50-13： FPU 传送至和传送自指令

助记符	指令
CFC1	从浮点传送控制字
CTC1	将控制字传送到浮点
MFC1	从浮点传送字
MFHC1	从浮点的高半部分传送字
MTC1	将字传送到浮点
MTHC1	将字传送到浮点的高半部分

50.12.4.3 FPU 算术指令

算术指令对格式化数据值执行操作。大多数浮点算术运算的结果都满足 IEEE 754 标准的准确性要求。结果等于使用当前舍入模式舍入为指定格式的无限精度结果。舍入结果与精确结果的差小于 1 个最低位单位 （Unit in the Least-significant Place， ULP）。

一般情况下，对于非规格化数字，算术指令会产生未实现操作异常，但 ABS、 C 和 NEG 指令除外，它们可以处理非规格化数字。CP1 FCSR 寄存器中的 FS、FO 和 FN 位可以改写该行为，如 **50.14.6 “FS/FO/FN 位的浮点操作”** 所述。

表 50-14 列出了 FPU IEEE 兼容的算术运算。

表 50-14： FPU IEEE 算术指令

助记符	指令
ABS.fmt	浮点绝对值
ADD.fmt	浮点加法
C.cond.fmt	浮点比较
DIV.fmt	浮点除法
MUL.fmt	浮点乘法
NEG.fmt	浮点取负
SQRT.fmt	浮点平方根
SUB.fmt	浮点减法
RECIP.fmt	浮点倒数近似值。见**注 1**。
RSQRT.fmt	浮点平方根倒数近似值。见**注 2**。

注 1： 该低延时运算的准确性可能低于 IEEE 规范。RECIP 结果与精确倒数的差不超过 1 个最低位单位 （ULP）。

2： 该低延时运算的准确性可能低于 IEEE 规范。RSQRT 结果与精确平方根倒数的差不超过 2 个 ULP。

4 个复合运算指令执行不同形式的乘法累加运算：将两个操作数相乘，将结果累加到第三个操作数，产生结果。在累加之前，会先根据当前舍入模式对积进行舍入。这种模型满足 IEEE 的准确性规范；其结果在数值上等于使用乘法、加法、减法或取负指令的等效计算。

表 50-15 中列出了复合运算指令。

表 50-15： FPU 乘法累加算术指令

助记符	指令
MADD.fmt	浮点乘加
MSUB.fmt	浮点乘减
NMADD.fmt	浮点取负乘加
NMSUB.fmt	浮点取负乘减

© 2013-2016 Microchip Technology Inc.

50.12.4.4 FPU 转换指令

这些指令在浮点和定点数据类型之间执行转换。每个指令可以将值从一些操作数格式转换为特定的结果格式。一些转换指令使用在浮点控制 / 状态寄存器 （Floating Control/Status Register，FCSR）中指定的舍入模式，而其他一些则直接指定舍入模式。

一般情况下，转换指令只会对于非规格化数字产生未实现操作异常。

CP1 FCSR寄存器中的FS和FN位可以改写该行为，如**50.14.6“FS/FO/FN位的浮点操作”**所述。

表 50-16 和表 50-17 根据舍入模式列出了 FPU 转换指令。

表 50-16： 使用 FCSR 舍入模式的 FPU 转换操作指令

助记符	指令
CVT.D.fmt	浮点转换为双精度浮点
CVT.L.fmt	浮点转换为长定点
CVT.S.fmt	浮点转换为单精度浮点
CVT.W.fmt	浮点转换为字定点

表 50-17： 使用指定舍入模式的 FPU 转换操作指令

助记符	指令
CEIL.L.fmt	浮点向上舍入为长定点
CEIL.W.fmt	浮点向上舍入为字定点
FLOOR.L.fmt	浮点向下舍入为长定点
FLOOR.W.fmt	浮点向下舍入为字定点
ROUND.L.fmt	浮点舍入为长定点
ROUND.W.fmt	浮点舍入为字定点
TRUNC.L.fmt	浮点截断为长定点
TRUNC.W.fmt	浮点截断为字定点

50.12.4.5 FPU 格式化操作数值传送指令

这些指令在FPU通用寄存器值之间传送格式化操作数值。特定操作数类型必须通过处理该类型的指令传送。存在三种传送指令：

- 无条件传送
- 测试 FPU 真 / 假条件代码的条件传送
- 测试 CPU 通用寄存器是否为零的条件传送

条件传送指令会以可能非预期的方式操作。它们总是会强制将目标寄存器中的值变为指令中指定格式的值。如果在执行条件传送之前，目标寄存器不包含指定格式的操作数，其内容会变为未定义的。关于更多信息，请参见 *"MIPS32® Architecture Reference Manual, Volume II"* 和 *"microMIPS32™ Architecture Reference Manual, Volume II"* 中各自对于条件传送指令的说明。注册用户可以从 Imagination Technologies Ltd. 网站 （www.imgtec.com）下载这些文档。

表 50-18 列出了格式化操作数值传送指令。

表 50-18： FPU 格式化操作数传送指令

助记符	指令
MOV.fmt	浮点传送
MOVF.fmt	FP 为假时的浮点条件传送
MOVT.fmt	FP 为真时的浮点条件传送
MOVN.fmt	非零时的浮点条件传送
MOVZ.fmt	为零时的浮点条件传送

50.12.4.6 FPU 条件转移指令

FPU具有PC相对条件转移指令，它们会测试由FPU比较指令（C.cond.fmt）设置的条件代码。

所有转移都具有一条指令的架构性延时。在发生转移时，紧接在转移指令之后的指令称为处于转移延时时隙中。它在转换到目标指令的操作发生之前执行。

存在两种形式的条件转移，它们的区别在于未发生转移而向下执行时，对于延时时隙中指令的处理方式：

- 转移指令会执行延时时隙中的指令。
- 如果未发生转移，转移指令可能不会执行延时时隙中的指令 （称作废弃延时时隙中的指令）。

注：	虽然包含了可能转移指令，但强烈建议避免在软件中使用可能转移指令，因为 MIPS 架构的未来版本中将会删除它们。

MIPS 架构定义了 8 个在比较和转移指令中使用的条件代码。为了与 ISA 的先前版本保持向后兼容，条件代码 bit 0 和条件代码 bit 1 至 7 在 FCSR 中处于非连续位域中。

表 50-19 列出了条件转移 （转移和可能转移） FPU 指令。

表 50-19： FPU 条件转移指令

助记符	指令
BC1F	FP 为假时转移
BC1T	FP 为真时转移
BC1FL	FP 可能为假时转移。见**注 1**。
BC1FT	FP 可能为真时转移。见**注 1**。

注 1： 已弃用的 FPU 指令。软件应避免使用它们。

© 2013-2016 Microchip Technology Inc.

50.12.4.7 其他 FPU 指令

MIPS32 架构定义了其他一些基于 FPU 条件代码条件性地将一个 CPU 通用寄存器内容传送另一个寄存器的指令。

表 50-20： FPU 为真 / 假时的 CPU 条件传送指令

助记符	指令
MOVN	FP 为假时的条件传送
MOVZ	FP 为真时的条件传送

50.12.5 浮点异常概述

IEEE 754 标准定义了 5 种异常条件：

- 无效操作异常
- 被零除异常
- 下溢异常
- 上溢异常
- 不精确异常

此外，还有一个特定于 MIPS 的异常条件，即未实现操作异常，它用于指示需要用软件来仿真指令。

通常， IEEE 算术运算只会导致一个异常条件。可以同时产生两个异常的唯一情况是不精确且上溢和不精确且下溢。

在程序控制下，IEEE 异常条件可以导致陷阱或不导致陷阱。IEEE 754 标准规定了在不产生陷阱时提供的结果。每当异常条件不会导致陷阱时， FPU 都会提供这些结果。执行的默认操作取决于异常条件的类型，对于上溢和下溢，将执行当前舍入模式。

表 50-21 总结了由 FPU 提供的默认结果。

在 PIC32 FPU 架构中， FPU 异常实现为具有 FCSR 的原因、使能和标志位域。标志位用于实现 IEEE 异常状态标志，原因和使能位用于控制异常陷阱。

在每个位域中， 5 种 IEEE 异常条件各有一个对应的位。原因位域具有一个额外的异常位，即未实现操作，可以用于产生软件仿真辅助的陷阱。如果通过 FCSR 的使能位域使能了某种异常类型，则对于这种类型的异常， FPU 以精确异常模式工作。

50.12.5.1 浮点精确异常模式

在精确异常模式下，陷阱在导致该陷阱的指令之前或任何后续指令可以完成并写入其结果之前产生。所以如果需要，软件陷阱处理程序可以在处理异常后恢复执行被中断的指令流。

原因位域用于报告每个位的指令异常条件。原因位在每个浮点算术运算期间写入，用于指示运算过程中产生的任何异常条件。如果产生其相应的异常条件，则原因位设置为 1 ；否则，它会清除为 0。

每次原因位及其相应的使能位均置 1 时，会产生浮点陷阱。这种情况在执行浮点运算过程中或将值传送至 FCSR 中时发生。未实现操作异常没有使能位：该异常总是会产生陷阱。

在陷阱处理程序中，在产生陷阱的任何浮点运算过程中产生的异常条件在原因位域中报告。在从浮点中断或异常返回之前，或在传送至 FCSR 操作使原因位置 1 之前，软件必须先通过执行传送至 FCSR 操作来清零已使能的原因位，以防止错误地重新产生陷阱。

如果浮点运算只是使非使能的原因位置 1，则不会产生任何陷阱，并存储 IEEE 754 标准定义的默认结果 （见表 50-21）。在浮点运算未产生陷阱时，程序可以通过读取原因位域来监视异常条件。

标志位域是指令完成时产生的 IEEE 异常条件的累积报告；产生陷阱的指令不会更新这些标志位。如果抛出了相应的 IEEE 异常，则这些标志位会被设置为 1，否则这些位不变。未实现操作异常没有对应的标志位。浮点运算永远不会产生使这些标志位清零的附带作用，但可以通过用软件将新值传送到 FCSR 来置 1 或清零它们。

表 50-21： FPU 对于未产生陷阱的异常提供的结果

位名称	说明	默认操作
V	无效操作	提供静默 NaN。
Z	被零除	提供带正确符号的无穷大。
U	下溢	取决于舍入模式，如下所示： （RN）：提供一个零，其符号为精确结果的符号。 （RZ）：提供一个零，其符号为精确结果的符号。 （RP）： 对于正下溢值，提供 2^{E_min} （MinNorm）。 对于负下溢值，提供正零。 （RM）： 对于正下溢值，提供负零。 对于负下溢值，提供负 2^{E_min} （MinNorm）。 注：只有 FCSR FN 位清零时，该行为才是有效的。
I	不精确	提供舍入后的结果。如果由上溢引起而未使能上溢陷阱，则提供上溢后的结果。如果由下溢引起而未使能下溢陷阱，则提供下溢后的结果。
O	上溢	取决于舍入模式，如下所示： • （RN）：提供一个无穷大，其符号为精确结果的符号。 • （RZ）：提供该格式的最大有限数字，其符号为精确结果的符号。 • （RP）：对于正上溢值，提供正无穷大。对于负上溢值，提供该格式的最负有限数字。 • （RM）：对于正上溢值，提供该格式的最大有限数字。对于负上溢值，提供负无穷大。

50.12.5.2 浮点无效操作异常

对于要执行的操作，当一个操作数无效或两个操作数都无效时，会发出无效操作异常。如果产生异常条件但未产生精确陷阱，结果将为静默 NaN。

以下操作是无效的：

- 一个操作数是信号NaN或两个操作数都是信号NaN（非算术MOV.fmt、MOVT.fmt、MOVF.fmt、MOVN.fmt 和 MOVZ.fmt 指令除外）。
- 加法或减法：无穷大的量值减法，如 (+∞) + (−∞) 或 (−∞) − (−∞)。
- 乘法：0 × ∞，使用任意符号。
- 除法：0/0 或 ∞/∞，使用任意符号。
- 平方根：操作数小于 0 （-0 是有效的操作数值）。
- 将浮点数字转换为某种定点格式，但上溢或操作数值为无穷大或 NaN 会阻止正确地用该格式表示。
- 一个操作数是 QNaN 值或两个操作数都是 QNaN 值的一些比较操作。

© 2013-2016 Microchip Technology Inc.

50.12.5.3 浮点被零除异常

如果除数为零且被除数为有限非零数字，则除法运算会发出被零除异常。在未产生精确陷阱时，其结果是带正确符号的无穷大。除法运算 0/0 和 ∞/0 不会导致被零除异常。 0/0 的结果是无效操作异常。 ∞/0 的结果是带正确符号的无穷大。

50.12.5.4 浮点下溢异常

有两个相关事件可能导致下溢异常：

- 极小值：产生介于 $\pm 2^{E_min}$ 之间的极小非零结果 （由于它极小）稍后可能导致某种其他异常，例如在进行除法运算时发生上溢。 IEEE 754 标准允许在检测极小值事件方面作出一些选择。 PIC32/MIPS 架构规定，当在将指数范围当作无界的情况下计算得到的非零结果严格介于 $\pm 2^{E_min}$ 之间时，在进行舍入后检测极小值。
- 准确性损失：在使用非规格化数字近似表示此类极小数字时，会发生非正常的准确性损失。 IEEE 754 标准允许在检测准确性损失事件方面作出一些选择。PIC32/MIPS 架构规定，当提供的结果不等于在将指数范围和精度都当作无界的情况下计算得到的结果时，将准确性损失检测为不精确结果。

发出下溢异常的方式取决于是否使能下溢陷阱：

- 未使能下溢陷阱时，只有同时检测到极小值和准确性损失时，才会发出下溢异常。提供的结果可能为零、非规格化数字或 $\pm 2^{E_min}$。
- 使能下溢陷阱时 （通过 FCSR 使能位域），无论是否发生准确性损失，在检测到极小值时都发出下溢异常。

50.12.5.5 浮点上溢异常

当舍入后的浮点结果的大小（如果指数范围是无界的）大于目标格式的最大有限数字时，将发出上溢异常。

在未产生精确陷阱时，结果由舍入模式和中间结果的符号决定。

50.12.5.6 浮点不精确异常

当发生以下事件之一时，将发出不精确异常：

- 运算的舍入结果是不精确的。
- 运算的舍入结果发生上溢，但没有产生上溢陷阱。
- 将非规格化操作数清除为零时。

50.12.5.7 浮点未实现操作异常

未实现操作异常是 MIPS 定义的异常，用于提供软件仿真支持。该异常不是 IEEE 兼容的。

PIC32/MIPS 架构设计为可以使用硬件和软件的组合来实现 FPU 功能。硬件不完全支持的操作会导致未实现操作异常，会使软件可以执行操作。

该异常条件没有对应的使能位。它总是会导致陷阱 （但是在 FS = 1 时，实际上会对于所有操作屏蔽该条件）。在软件异常处理程序中完成相应的仿真或其他操作后，可以继续执行原来的指令流。

对于不支持非规格化操作数的指令，当遇到非规格化操作数或极小结果，并且这种情况未由 FS/FO/FN 位处理时，会产生未实现操作异常。

50.12.6 浮点流水线和性能

本节介绍 FPU 流水线的结构和操作。

50.12.6.1 FPU 流水线概述

FPU 具有一个 7 级流水线，整数流水线会向其分派指令：

- 解码、寄存器读取和解包 （FR 流水线级）
- 乘法树 —— 两次运算得到双精度 （M1 流水线级）
- 乘法完成 （M2 流水线级）
- 加法第一步 （A1 流水线级）
- 加法第二步即最后一步 （A2 流水线级）
- 打包为 IEEE 格式 （FP 流水线级）
- 寄存器回写 （FW 流水线级）

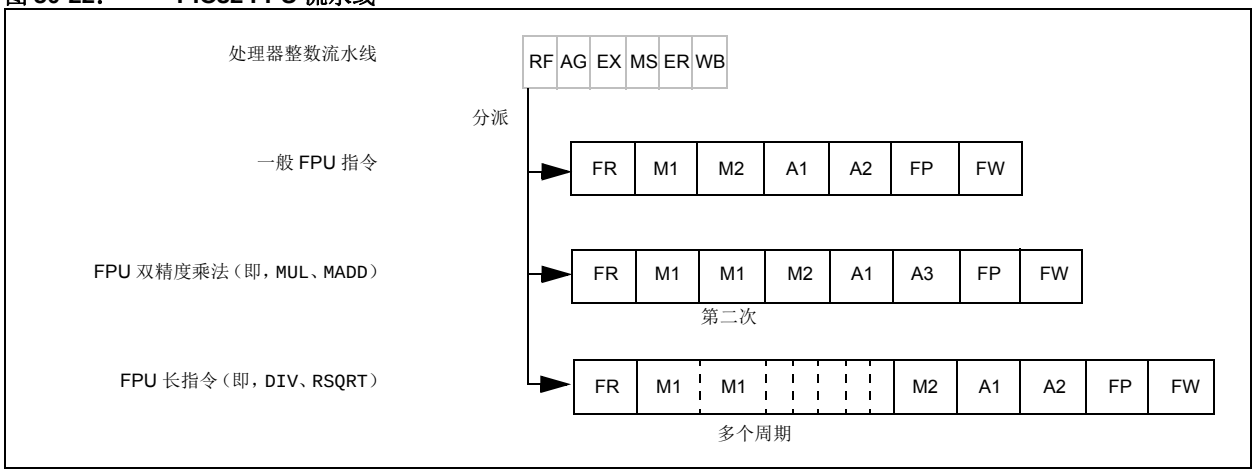
FPU 实现了旁路机制，通过该机制可以将运算结果直接转发到需要它的指令处，而无需先将结果写入 FPU 寄存器，然后再进行回读。

FPU 流水线与 PIC32 内核整数流水线并行运行。 FPU 构造为以与 PIC32 内核相同的频率运行。

FPU 流水线针对单精度指令进行了优化，从而可以在单周期吞吐率和低延时下执行基本乘法、ADD/SUB 和 MADD/MSUB 指令。执行双精度乘法和 MADD/MSUB 指令需要第二次通过 M1 级，以生成积的全部 64 位。

执行长延时指令 （如 DIV 和 RSQRT）会使 M1 流水线级延长。图 50-22 显示了 FPU 流水线。

图 50-22： PIC32 FPU 流水线



流水线级 1：FPU 流水线：FR 级 —— 解码、寄存器读取和解包

FR 级具有以下功能：

- 对分派的指令解码，以进行寄存器访问。
- 从寄存器文件中读取数据。
- 将操作数解包为内部格式。

流水线级 2：FPU 流水线：M1 级 —— 乘法树

M1 级具有以下功能：

- 为单精度数据格式乘法提供单周期乘法阵列，为双精度数据格式乘法提供两个周期。
- 长指令 （如除法和平方根）会在该流水线级循环若干个周期。
- 计算指数的和。

© 2013-2016 Microchip Technology Inc.

流水线级 3：FPU 流水线：M2 级 —— 乘法完成

M2 级具有以下功能：

- 乘法在保留进位的已编码乘积压缩为二进制时完成
- 执行舍入
- 计算用于加法路径的指数差

流水线级 4：FPU 流水线：A1 级 —— 加法第一步

该流水线级执行加法的第一步。

流水线级 5：FPU 流水线：A2 级 —— 加法第二步即最后一步

该流水线级执行加法的第二步即最后一步。

流水线级 6：FPU 流水线：FP 级 —— 结果打包

FP 级具有以下功能：

- 将来自数据路径的结果打包为 IEEE 754 标准格式，以写入 FPR 寄存器文件
- 解决上溢和下溢异常条件

流水线级 7：FPU 流水线：FW 级 —— 寄存器写入

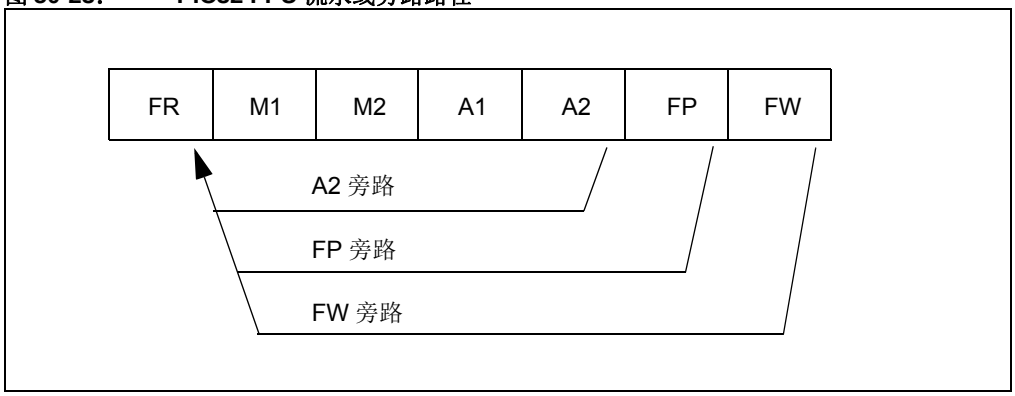
将结果写入 FPR 寄存器文件。

50.12.6.2 FPU 旁路

FPU 流水线实现了大量旁路功能，从而结果无需先写入寄存器文件再读回后才能使用，结果可以直接转发到已处于流水线中的指令。

在 32 位寄存器文件模式下工作时 （CP0 状态 <26> 寄存器中的 FR 位为 0），由于通过将偶编号 - 奇编号 32 位寄存器配对来提供 64 位寄存器，部分旁路功能会被禁止。

图 50-23： PIC32 FPU 流水线旁路路径



50.12.6.3 FPU 重复率和延时

表 50-22 列出了 FPU 指令的重复率和延时。

请注意，与浮点运算相关的周期数是以 FPU 时钟数列出的。

表 50-22： FPU 延时和重复率

操作码 （见注 1）	延时 （FPU 周期数）	重复率 （FPU 周期数）
ABS.[S,D], NEG.[S,D], ADD.[S,D],SUB.[S,D], MUL.S, MADD.S,MSUB.S, NMADD.S?MSUB.S	4	1
MUL.D, MADD.D, MSUB.D, NMADD.D, NMSUB.D	5	2
RECIP.S	13	10
RECIP.D	25	21
RSQRT.S	17	14
RSQRT.D	35	31
DIV.S, SQRT.S	17	14
DIV.D, SQRT.D	32	29
C.cond.[S,D] 至 MOVF.fmt 和 MOVT.fmt 指令 /MOVT, MOVN, BC1 指令	1/2	1
CVT.D.S、 CVT.[S,D].[W,L]	4	1
CVT.S.D	6	1
CVT.[W,L].[S,D], CEIL.[W,L].[S,D], FLOOR.[W,L].[S,D], ROUND.[W,L].[S,D], TRUNC.[W,L].[S,D]	5	1
MOV.[S,D], MOVF.[S,D], MOVN.[S,D], MOVT.[S,D], MOVZ.[S,D]	4	1
LWC1, LDC1, LDXC1, LUXC1, LWXC1	3	1
MTC1, MFC1	2	1

图注： S = 单精度； D = 双精度； W = 字； L = 长字

© 2013-2016 Microchip Technology Inc.

50.12.6.4 浮点 2008 FPU 支持

PIC32 FPU 实现了以下状态 / 控制位，以便与 2008 年发布的 IEEE 754 浮点标准更加兼容：

- Has2008 位 （FIR<23>）总是读为 1，表示实现了 2008 FPU。
- MAC2008 位 （FCSR<20>）总是读为 0，表示未实现融合乘加运算。
- ABS2008 位 （FCSR<19>）总是设置为 1，这会使 ABS 和 NEG 指令成为非算术指令。所有浮点异常都会被禁止。
- NAN2008 位（FCSR<18>）总是设置为 1，表示支持 IEEE 754-2008 标准建议的静默 NaN 和信号 NaN 编码。此外，实现了以下行为：
  - 在具有一个或多个 QNaN 操作数（没有 SNaN）的情况下，QNaN 操作数传播自其中一个输入寄存器操作数 （按优先顺序排列）：*fs*、*ft* 和 *fr* （请参见下面的**注**）。
  - 使用 SNaN 作为输入并且异常被禁止时， QNaN 是预期的输出
  - QNaN 输出不会是固定值。为了符合 IEEE，输入 NaN 应生成具有输入 NaN 有效载荷（如果其可用目标格式表示）的 NaN，其中的有效载荷定义为尾数位域去掉其最高有效位。
  - 如果 ABS2008 = 1 且 MAC2008 = 0 （在 PIC32 中总是如此）， NMADD 和 NMSUB 的符号不会反转任何 QNaN 输入的符号，符号会被保留并传播给输出。
  - 当 NaN 为输入时，输出将为输入 NaN 之一，并尽可能多地保留尾数。
  - SNaN 输入的优先级高于 QNaN 输入，然后 *fs* 的优先级高于 *ft*，后者的优先级又高于 *fr* 寄存器 （请参见下面的**注**）。
  - 所选择的 NaN 的符号会被保留。如果为输出选择的输入已经是 QNaN，则会保留整个尾数。但是，如果为输出选择的输入是 SNaN，则会补足 SNaN 尾数的最高位，以便将 SNaN 转换为 QNaN。如果这种转换为 QNaN 的操作会产生一个无穷大，则尾数的下一最高位会被置 1。
  - 对于CVT.s.d，NaN尾数的高位会被保留。对于CVT.d.s，NaN尾数的低位会被填充为0。
  - 对于乘加，如果 *fs*/*ft* 和 *fr* 寄存器都为 QNaN，则乘法会基于 *fs*/*ft* 产生一个 QNaN，并且该 QNaN 在加法运算中的优先级高于 *fr*。但是，如果 *fs*/*ft* 和 *fr* 寄存器都为 SNaN，并且未使能无效陷阱，则乘法会基于 *fs*/*ft* 产生一个 QNaN，然后将它与信号 *fr* 寄存器相加，并且信号 *fr* 优先。
  - 当输出需要一个 NaN，但没有 NaN 输入时，会产生一个正 QNaN，并且它的所有其他尾数位均置 1。

注：	寄存器 *fs*、*ft* 和 *fr* 为乘加浮点运算中出现的浮点寄存器。例如：MADD.D fd, fr, fs, ft。

50.12.6.5 IEEE 754-1985 标准

IEEE 754-1985 标准 *“IEEE Standard for Binary Floating-Point Arithmetic ”* 定义了以下方面：

- 浮点数据类型
- 基本的算术、比较和转换运算
- 计算模型

IEEE 754-1985 标准未定义具体的处理资源，也未定义指令集。关于 IEEE 754-1985 标准的更多信息，请访问 IEEE 网页 http://stdsbbs.ieee.org/ 。

50.13 协处理器 0 （CP0）寄存器

PIC32 使用特殊的寄存器接口在系统软件和 CPU 之间传送状态和控制信息。该接口称为协处理器 0 或 CP0。通过协处理器 0 可实现的 CPU 功能为：

- 页表缓冲区 （Translation Lookaside Buffer， TLB）
- 内核定时器
- 中断和异常控制
- 虚拟存储器配置
- 影子寄存器集控制
- 处理器标识
- 调试器控制
- 性能计数器

系统软件可以使用协处理器指令 （例如 MFC0 和 MTC0）访问 CP0 中的寄存器。表 50-23 列出了 PIC32 器件上的 CP0 寄存器。

表 50-23： CP0 寄存器

寄存器编号	寄存器名称	功能
0	Index	TLB 阵列的索引 （仅限 MPU）。
1	Random	随机生成的 TLB 阵列索引 （仅限 MPU）。
2	EntryLo0	偶数编号虚拟页面的 TLB 条目的低位部分 （仅限 MPU）。
3	EntryLo1	奇数编号虚拟页面的 TLB 条目的低位部分 （仅限 MPU）。
4	Context/ UserLocal	存储器中页表条目的指针 （仅限 MPU）。 可由特权软件写入并通过 RDHWR 指令读取的用户信息。
5	PageMask/ PageGrain	PageMask 控制 TLB 条目中可变页大小。 PageGrain 支持 TLB 中的 1 KB 页面 （仅限 MPU）。
6	Wired	控制固定 （即连接） TLB 条目的数量 （仅限 MPU）。
7	HWREna	使能在非特权模式下通过 RDHWR 指令访问选定的硬件寄存器。
8	BadVAddr	报告上一条发生地址相关异常的地址。
	BadInstr	报告导致上一次异常的指令。
	BadInstrP	如果上一次异常由于延时时隙导致，报告该转移指令。
9	Count	处理器周期计数。
10	EntryHi	TLB 条目的高位部分 （仅限 MPU）。
11	Compare	内核定时器中断控制。
12	Status	处理器状态和控制。
	IntCtl	向量空间中断控制。
	SRSCtl	影子寄存器集控制。
	SRSMap	影子寄存器映射控制。
	View_IPL	允许读 / 写优先级，而无需从 Status 寄存器取出该位或者将该位写入该寄存器。
	SRSMAP2	包含 2 个 4 位域，这些位域用于将向量编号映射到要在提供中断服务时使用的影子集编号。
13	Cause	描述上一次异常的原因。
	NestedExc	包含在当前异常之前存在的错误和异常级别状态位值。
	View_RIPL	使能对 Cause 寄存器中的 RIPL 位的读访问。
14	EPC	上一次异常的程序计数器。
	NestedEPC	包含在当前异常之前存在的异常程序计数器。

© 2013-2016 Microchip Technology Inc.

表 50-23： CP0 寄存器 （续）

寄存器编号	寄存器名称	功能
15	PRID	处理器标识和版本。
	Ebase	异常向量的异常基址。
	CDMMBase	通用器件存储器映射基址。
16	Config	配置寄存器。
	Config1	配置寄存器 1。
	Config2	配置寄存器 2。
	Config3	配置寄存器 3。
	Config4	配置寄存器 4。
	Config5	配置寄存器 5。
	Config7	配置寄存器 7。
17	LLAddr	装载链接地址 （仅限 MPU）。
18	WatchLo	监视点地址低位 （仅限 MPU）。
19	WatchHi	监视点地址高位 （仅限 MPU）。
20-22	保留	PIC32 内核中保留的寄存器。
23	Debug	EJTAG 调试寄存器。
	TraceControl	EJTAG 跟踪控制寄存器。
	TraceControl2	EJTAG 跟踪控制寄存器 2。
	UserTraceData1	EJTAG 用户跟踪数据寄存器 1。
	TraceBPC	EJTAG 跟踪断点寄存器。
	Debug2	调试控制 / 异常状态寄存器 1。
24	DEPC	上一次调试异常的程序计数器。
	UserTraceData2	EJTAG 用户跟踪数据寄存器 2。
25	PerfCtl0	性能计数器控制寄存器 0。
	PerfCnt0	性能计数器 0。
	PerfCtl1	性能计数器控制寄存器 1。
	PerfCnt1	性能计数器 1。
26	ErrCtl	支持对指令高速缓存和数据高速缓存的路选和数据 RAM 阵列进行软件测试 （仅限 MPU）。
27	CacheErr	记录关于高速缓存 /SPRAM 奇偶校验错误的信息。
28	TagLo/DataLo	高速缓存标记接口的低位部分 （仅限 MPU）。
29	保留	PIC32 内核中保留的寄存器。
30	ErrorEPC	上一次错误的程序计数器。
31	DeSAVE	调试处理程序的中间结果缓存寄存器。
	KScratch1	用于内核模式的中间结果暂存寄存器。
	KScratch2	用于内核模式的中间结果暂存寄存器。

50.13.1 Index 寄存器 （CP0 寄存器 0，选择 0）（仅限 MPU）

Index 寄存器是一个 32 位读 / 写寄存器，对于 TLBP、 TLBR 和 TLBWI 指令，包含用于访问 TLB 的索引。

寄存器 50-1： Index：TLB 索引寄存器； CP0 寄存器 0，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-x	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	P	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	—	—	—	Index<4:0>(1)				

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31 **P：**探针故障检测位
1 = 先前的 TLBP 指令未在 TLB 中找到匹配项
0 = 先前的 TLBP 指令在 TLB 中找到了匹配项

bit 30-5 **未实现：**读为 0

bit 4-0 **Index<4:0>：**TLB 条目的索引值位，受 TLBR 和 TLBW 指令的影响 (1)
11111 = TLB 条目 31
•
•
•
00000 = TLB 条目 0

注 1： 并非所有位均可使用，具体取决于 MMU 的配置。 MMU 所支持的 TLB 条目数可从 Config1 CP0 寄存器的 MMU Size<5:0> 域读取。

© 2013-2016 Microchip Technology Inc.

50.13.2 Random 寄存器 （CP0 寄存器 1，选择 0） （仅限 MPU）

Random 寄存器为只读寄存器，其值用于在 TLBWR 指令期间对 TLB 建立索引。

寄存器的值在上限和下限之间变化，如下所示：

- 下限值由保留为专用于操作系统的 TLB 条目数 （Wired 寄存器的内容）决定。 Wired 寄存器所索引的条目是可由 TLB 随机写操作写入的第一个条目。
- 上限值为 TLB 条目总数减 1。

Random 寄存器几乎每个时钟递减 1，当递减到等于 Wired 寄存器值时就重置为上限值。为了增加随机性并降低产生活锁条件的可能性，可使用 LFSR 寄存器来伪随机地防止数值的减少。

在复位异常并且对 Wired 寄存器进行写操作时，处理器将 Random 寄存器初始化为上限值。

寄存器 50-2： Random：随机域寄存器； CP0 寄存器 1，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	R-1	R-1	R-1	R-1
	—	—	—	—	Random<3:0>			

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-4 **未实现：**读为 0

bit 3-0 **Random<3:0>：**TLB 随机索引位

50.13.3 EntryLo0 寄存器 （CP0 寄存器 2，选择 0）和 EntryLo1 寄存器 （CP0 寄存器 3，选择 0）（仅限 MPU）

这一对 EntryLo 寄存器用作 TLB 和 TLBR、TLBWI 和 TLBWR 指令间的接口。EntryLo0 保存偶数页的条目，而 EntryLo1 保存奇数页的条目。

如果出现了地址错误、TLB 无效、TLB 修改或 TLB 重填异常，那么 EntryLo0 和 EntryLo1 寄存器的内容为未定义。

寄存器 50-3： EntryLo0：偶数页 TLB 条目寄存器； CP0 寄存器 2，选择 0 和 EntryLo1：奇数页 TLB 条目寄存器； CP0 寄存器 3，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	U-0	U-0	U-0	U-0	R/W-x	R/W-x
	RI	XI	—	—	—	—	PFN<19:18>	
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	PFN<17:10>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	PFN<9:2>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	PFN<1:0>		C<2:0>			D	V	G

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31 **RI：**读禁止位
如果该位置 1，尝试读取页面中的数据将导致产生 TLB 无效异常，即使 V （有效）位置 1 也是如此。仅当 PageGrain 寄存器的 RIE 位置 1 时，才使能 RI 位。如果 PageGrain 的 RIE 位未置 1，根据 MIPS32 规范，EntryLo0/EntryLo1 的 RI 位保留为 0。

bit 30 **XI：**执行禁止位
如果该位置 1，尝试从页面取指将导致产生 TLB 无效异常，即使 V（有效）位置 1 也是如此。仅当 PageGrain 寄存器的XIE位置1时，才使能XI位。如果PageGrain的XIE位未置1，根据MIPS32规范，EntryLo0/EntryLo1 的 XI 位保留为 0。

bit 29-26 **未实现：**读为 0

bit 25-6 **PFN<19:0>：**页帧编号位
有助于定义物理地址的高位。
PFN<19:0> 位对应物理地址的 bit <31:12>。

bit 5-3 **C<2:0>：**页面一致性属性位
111 = 保留
110 = 保留
101 = 保留
100 = 保留
011 = 可高速缓存、非一致性、回写和写分配
010 = 非高速缓存
001 = 可高速缓存、非一致性、直写和写分配
000 = 可高速缓存、非一致性、直写和无写分配

bit 2 **D：**Dirty （写使能）位
1 = 允许写入该页面
0 = 写入该页面将导致 TLB 修改异常

bit 1 **V：**有效位
1 = 允许访问该页面
0 = 访问该页面将导致 TLB 无效异常

bit 0 **G：**全局位
在对 TLB 进行写操作时， EntryLo0 和 EntryLo1 寄存器中 G 位的逻辑与运算结果作为 TLB 条目中的 G 位。如果 TLB 条目的 G 位为 1， ASID 比较将在 TLB 匹配时被忽略。在对 TLB 条目进行读操作时， EntryLo0 和 EntryLo1 的 G 位均反映了 TLB G 位的状态。

© 2013-2016 Microchip Technology Inc.

50.13.4 Context 寄存器 （CP0 寄存器 4，选择 0）（仅限 MPU）

Context 寄存器为一个读 / 写寄存器，包含一个指向页表条目（Page Table Entry，PTE）阵列中的某个条目的指针。该阵列采用操作系统数据结构，包含虚拟地址到物理地址的转换信息。当 TLB 未命中时，操作系统从 PTE 阵列读取未命中的转换地址装入 TLB。Context 寄存器复制 BadVAddr 寄存器中提供的部分信息，但使用特定的组织方式，使操作系统可以直接访问存储器中的一个 8 字节页表条目 （PTE）。

寄存器 50-4： Context：Context 寄存器； CP0 寄存器 4，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	PTEBase<8:1>							
23:16	R/W-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	PTEBase<0>	BadVPN2<19:13>						
15:8	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	BadVPN2<12:5>							
7:0	R-x	R-x	R-x	R-x	R-x	U-0	U-0	U-0
	BadVPN2<4:0>					—	—	—

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-23 **PTEBase<8:0>**：Context 寄存器 PTE 阵列指针位
这些位供操作系统使用，通常，写入这些位的值能够让操作系统将 Context 寄存器作为指针来访问存储器中的当前 PTE 阵列。

bit 22-4 **BadVPN2<19:0>**：TLB 硬件未命中状态位
这些位包含未命中虚拟地址的 VA<31:13> 的值。

bit 3-0 **未实现**：读为 0

50.13.5 UserLocal 寄存器 （CP0 寄存器 4，选择 2）

UserLocal 寄存器为一个读 / 写寄存器，不由硬件控制，而是通过 RDHWR 指令进行有条件读取。

特权软件可能会将任何信息写入该寄存器，并且通过 RDHWR 指令的寄存器 29 （ULR）将它设为可被非特权软件访问。为此，ULR位（HWREna<29>）必须设置为1来使能寄存器的非特权访问。

在某些操作环境中，UserLocal 寄存器包含一个指向线程专用存储区的指针，可通过 RDHWR 寄存器获得。

寄存器 50-5： UserLocal：UserLocal 寄存器； CP0 寄存器 4，选择 2

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	USERLOCAL<31:24>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	USERLOCAL<23:16>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	USERLOCAL<15:8>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	USERLOCAL<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **USERLOCAL<31:0>：**用户本地位

© 2013-2016 Microchip Technology Inc.

50.13.6 PageMask 寄存器 （CP0 寄存器 5，选择 0）（仅限 MPU）

PageMask 寄存器为一个读 / 写寄存器，用于读 / 写 TLB。它包含一个比较掩码，可为每个 TLB 条目设置可变页大小，如表 50-24 所示。

寄存器 50-6： PageMask：TLB 可变页大小寄存器； CP0 寄存器 5，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	—	—	—	Mask<15:11>				
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	Mask<10:3>							
15:8	R/W-x	R/W-x	R/W-x	U-0	U-0	U-0	U-0	U-0
	Mask<2:0>			—	—	—	—	—
7:0	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-29 **未实现：**读为 0

bit 28-13 **Mask<15:0>：**虚拟地址掩码位

当该位为 1 时，指示虚拟地址相应的位不参与 TLB 匹配。

bit 12-0 **未实现：**读为 0

表 50-24： PageMask 寄存器掩码位的值

页大小	寄存器位位置															
	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
4 KB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16 KB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
64 KB	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
256 KB	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
1 MB	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
4 MB	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
16 MB	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
64 MB	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
256 MB	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

50.13.7 PageGrain 寄存器 （CP0 寄存器 5，选择 1）（仅限 MPU）

在 PIC32 器件中，PageGrain 寄存器用于使能或禁止 EntryLo0 和 EntryLo1 寄存器中的读和执行禁止位。

寄存器 50-7： PageGrain：TLB 页增益使能寄存器； CP0 寄存器 5，选择 1

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	U-0	U-0	R/W-0	U-0	U-0	U-0
	RIE	XIE	—	—	IEC	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31 **RIE：**读禁止使能位
1 = 使能 EntryLo0 和 EntryLo1 寄存器的 RI 位
0 = 禁止 EntryLo0 和 EntryLo1 寄存器的 RI 位且不可写

bit 30 **XIE：**执行禁止使能位
1 = 使能 EntryLo0 和 EntryLo1 寄存器的 XI 位
0 = 禁止 EntryLo0 和 EntryLo1 寄存器的 XI 位且不可写

bit 29-28 **未实现：**必须写为 0；在读取时返回 0

bit 27 **IEC：**使能读禁止和执行禁止异常代码位
1 = 读禁止异常使用 TLBRI 异常代码。执行禁止异常使用 TLBXI 异常代码
0 = 读禁止和执行禁止异常均使用 TLBL 异常代码

bit 26-0 **未实现：**必须写为 0；在读取时返回 0

© 2013-2016 Microchip Technology Inc.

50.13.8 Wired 寄存器 （CP0 寄存器 6，选择 0）（仅限 MPU）

Wired 寄存器是一个读 / 写寄存器，指定 TLB 中固定条目和随机条目的边界。 Wired 域宽度的计算方法同上述 Index 寄存器的计算。Wired 条目是固定的、不可替换的条目，不能由 TLBWR 指令改写，但可由 TLBWI 指令改写。

Wired 寄存器由复位异常复位为零。写 Wired 寄存器将导致 Random 寄存器复位为其上限值。

寄存器 50-8： Wired：TLB 边界条目寄存器； CP0 寄存器 6，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	Wired<4:0>(1)				

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-5 **未实现：**读为 0

bit 4-0 **Wired<4:0>：**TLB 固定边界位 (1)

11111 = 条目 31 为随机，条目 0-30 为固定
·
·
·
01111 = 条目 15 为随机，条目 0-14 为固定
·
·
·
00111 = 条目 7 及以上为随机，条目 7 以下为固定
·
·
·
00000 = 所有 16 个条目均为随机

注 1： 并非所有位均可用，具体取决于 MMU 的配置。 MMU 所支持的 TLB 条目数可从 Config1 CP0 寄存器的 MMU Size<5:0> 域读取。

50.13.9 HWREna 寄存器 （CP0 寄存器 7，选择 0）

HWREna 寄存器包含的位掩码决定哪些硬件寄存器可通过 RDHWR 指令访问。

寄存器 50-9： HWREna：硬件可访问能力寄存器； CP0 寄存器 7，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
	—	—	ULR	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	—	MASK<3:0>			

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-30 **未实现：**读为 0

bit 29 **ULR：**用户本地寄存器位
1 = 使能 UserLocal 寄存器的非特权访问
0 = 禁止 UserLocal 寄存器的非特权访问
该位提供对协处理器 0 UserLocal 寄存器的读访问。

bit 28-4 **未实现：**读为 0

bit 3-0 **MASK<3:0>：**位掩码位
1 = 允许访问相应的硬件寄存器
0 = 禁止访问
这些位中的每个位用于允许 RDHWR 指令对特定硬件寄存器 （它可能不是实际的寄存器）的访问。关于有效硬件寄存器的列表，请参见 RDHWR 指令。

© 2013-2016 Microchip Technology Inc.

50.13.10 BadVAddr 寄存器 （CP0 寄存器 8，选择 0）

BadVAddr 是一个只读寄存器，它捕捉最近导致地址错误异常的虚拟地址。可导致地址错误的事件有：在未对齐地址处执行装载、存储或取指操作，或在用户模式下尝试访问内核模式地址。

对于采用 MPU 内核的器件，BadVAddr 寄存器还将捕捉最近导致 TLB 重填、TLB 无效、或 TLB 修改异常的虚拟地址。

BadVAddr 不会捕捉总线错误的地址信息，因为这不是寻址错误。

寄存器 50-10： BadVAddr：无效虚拟地址寄存器； CP0 寄存器 8，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	BadVAddr<31:24>							
23:16	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	BadVAddr<23:16>							
15:8	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	BadVAddr<15:8>							
7:0	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	BadVAddr<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **BadVAddr<31:0>**：无效虚拟地址位
捕捉导致最近地址错误异常的虚拟地址。

50.13.11 BadInstr 寄存器 （CP0 寄存器 8，选择 1）（仅限 M-Class）

BadInstr 寄存器是一个可选的只读寄存器，用于捕捉导致以下异常之一的最近指令：

- 指令有效性：协处理器不可用和保留指令
- 执行异常：整数溢出、陷阱、系统调用、断点、浮点和协处理器 2 异常
- 寻址：地址错误、 TLB 重填、 TLB 无效、 TLB 读禁止、 TLB 执行禁止和 TLB 修改

BadInstr 寄存器用于加快指令仿真速度。 BadInstr 寄存器只会由与指令同步产生的异常置 1。 BadInstr 寄存器不会由中断或由 NMI、机器检查、总线错误、高速缓存错误、监视或 EJTAG 异常置 1。

产生同步异常时，如果没有有效的指令字 （例如 TLB 重填 —— 取指），则存储在 BadInstr 中的值是不可预测的。

是否存在BadInstr寄存器由Config3BI位指示。BadInstr寄存器在MT ASE处理器中按VPE实例化。

寄存器 50-11： BadInstr：无效指令字寄存器； CP0 寄存器 8，选择 1

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	BadInstr<31:24>							
23:16	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	BadInstr<23:16>							
15:8	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	BadInstr<15:8>							
7:0	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	BadInstr<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **BadInstr<31:0>**：无效 （故障）指令字位
少于 32 位的指令字会被放入 bit 15:0， bit 31:16 则包含 0。

© 2013-2016 Microchip Technology Inc.

50.13.12 BadInstrP 寄存器 （CP0 寄存器 8，选择 2）（仅限 M-Class）

BadInstrP 寄存器是一个与 BadInstr 寄存器配合使用的可选寄存器。当故障指令处于转移延时时隙中时， BadInstrP 寄存器中包含先前的转移指令。

对于以下异常， BadInstrP 寄存器会被更新：

- 指令有效性：协处理器不可用和保留指令
- 执行异常：整数溢出、陷阱、系统调用、断点、浮点和协处理器 2 异常
- 寻址：地址错误、 TLB 重填、 TLB 无效、 TLB 读禁止、 TLB 执行禁止和 TLB 修改

BadInstrP 寄存器用于加快指令仿真速度。 BadInstrP 寄存器只会由与指令同步产生的异常置 1。 BadInstrP 寄存器不会由中断或由 NMI、机器检查、总线错误、高速缓存错误、监视或 EJTAG 异常置 1。在产生同步异常，并且故障指令不处于转移延时时隙中时，存储在 BadInstrP 中的值是不可预测的。

是否存在 BadInstrP 寄存器由 Config3BP 位指示。BadInstrP 寄存器在 MT ASE 处理器中按 VPE 实例化。

寄存器 50-12： BadInstrP：无效先前转移指令寄存器； CP0 寄存器 8，选择 2

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	BadInstrP<31:24>							
23:16	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	BadInstrP<23:16>							
15:8	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	BadInstrP<15:8>							
7:0	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	BadInstrP<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **BadInstrP<31:0>**：无效先前转移指令位
少于 32 位的指令字会被放入 bit 15:0， bit 31:16 则包含 0。

50.13.13 Count 寄存器 （CP0 寄存器 9，选择 0）

Count 寄存器用作一个定时器，无论指令是否执行、撤消，或是在流水线中有任何前移，它都以恒定速率递增。如果 Cause 寄存器中的 DC 位为 0，则计数器每隔一个时钟递增一次。

Count 可用于实现某种功能或用于进行诊断，包括在复位时使用或用于同步处理器。

通过写 Debug 寄存器中的 COUNTDM 位，可以控制在处理器处于调试模式时 Count 是否继续递增。

寄存器 50-13： Count：间隔计数器寄存器； CP0 寄存器 9，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COUNT<31:24>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COUNT<23:16>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COUNT<15:8>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COUNT<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **COUNT<31:0>：**间隔计数器位
该值每隔一个时钟周期递增一次。

© 2013-2016 Microchip Technology Inc.

50.13.14 EntryHi 寄存器 （CP0 寄存器 10，选择 0）（仅限 MPU）

EntryHi 寄存器包含用于 TLB 读、写和访问操作的虚拟地址匹配信息。

TLB 异常（TLB 重填、TLB 无效或 TLB 修改）导致将虚拟地址的位 VA31...13 写入 EntryHi 寄存器的 VPN2 域。 TLBR 指令将选中的 TLB 条目的相应域写入 EntryHi 寄存器。软件将当前地址空间标识符值写入 ASID 域，并且该域在 TLB 比较过程中用于确定 TLB 是否匹配。

由于 ASID 域由 TLBR 指令改写，因此软件必须保存和重新存储使用 TLBR 前后的 ASID 的值。这在发生 TLB 无效和 TLB 修改异常时以及在其他存储器管理软件中尤为重要。

在发生地址错误异常后， EntryHi 寄存器的 VPN2 域未定义，并且该域可在地址错误异常序列期间被硬件修改。软件写（通过 MTC0）EntryHi 寄存器不会导致 BadVAddr 或 Context 寄存器中的地址相关域发生隐式写入。

寄存器 50-14： EntryHi：TLB 地址匹配寄存器； CP0 寄存器 10，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	VPN2<18:11>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	VPN2<10:3>							
15:8	R/W-x	R/W-x	R/W-x	U-0	U-0	U-0	U-0	U-0
	VPN2<2:0>			—	—	—	—	—
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	ASID<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-13 **VPN2<18:11>：**虚拟页码位
这些位在发生 TLB 异常或进行 TLB 读取时由硬件写入，且在写 TLB 之前由软件写入。

bit 12-8 **未实现：**读为 0

bit 7-0 **ASID<7:0>：**地址空间标识符位
这些位在进行 TLB 读取时由硬件写入；通过软件写入时，可以为 TLB 写操作确立当前的 ASID 值以及与每个条目的 TLB ASID 域匹配的 TLB 引用。

50.13.15 Compare 寄存器 （CP0 寄存器 11，选择 0）

Compare 寄存器与 Count 寄存器一起用于实现定时器和定时器中断功能。 Compare 保持一个稳定值，不会自行更改。

当 Count 值等于 Compare 值时，CPU 会向系统中断控制器发出中断信号。该信号将一直保持有效，直到写入 Compare 为止。

寄存器 50-15： Compare：间隔计数比较寄存器； CP0 寄存器 11，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COMPARE<31:24>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COMPARE<23:16>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COMPARE<15:8>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COMPARE<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **COMPARE<31:0>：**间隔计数比较值位

© 2013-2016 Microchip Technology Inc.

50.13.16 Status 寄存器 （CP0 寄存器 12，选择 0）

Status 是一个读 / 写寄存器，包含处理器的工作模式、中断允许和诊断状态。该寄存器的这些位组合产生处理器的工作模式。

50.13.16.1 中断允许

当以下所有条件均为真时，将允许中断：

- IE = 1
- EXL = 0
- ERL = 0
- DM = 0

如果满足这些条件，则 IPL 位的设置将允许中断。

50.13.16.2 工作模式

如果 Debug 寄存器中的 DM 位为 1，则处理器处于调试模式；否则，处理器处于内核或用户模式。表 50-25 中所示的 CPU Status 寄存器位设置决定用户或内核模式：

表 50-25： 决定处理器模式的 CPU Status 寄存器位

模式	位 / 设置		
用户 （要求以下*所有*位和值）	UM = 1	EXL = 0	ERL = 0
内核 （要求*一个*或多个以下位值）	UM = 0	EXL = 1	ERL = 1

注： Status 寄存器的 CU0 位（Status<28>）控制协处理器 0 的可访问性。如果协处理器 0 不可用，则访问它的指令会产生异常。

寄存器 50-16： Status：状态寄存器； CP0 寄存器 12，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	R/W-0	R/W-x	R/W-0	R/W-0	R/W-x	R/W-0
	—	—	CU1(2)	CU0	RP	FR(2)	RE	MX(3)
23:16	U-0	R/W-1	R/W-0, HS, CS	R/W-1	R/W-0	R/W-x	U-0	R/W-0
	—	BEV	TS(1)	SR	NMI	IPL<7>	—	IPL<6>
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0
	IPL<5:0>						—	—
7:0	U-0	U-0	U-0	R/W-x	U-0	R/W-x	R/W-x	R/W-x
	—	—	—	UM	—	ERL	EXL	IE

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-30 **未实现：**读为 0

注 1： 该位仅在采用 MPU 内核的器件上可用。要确定可用性，请参见具体器件数据手册中的 **“CPU”** 章节。
2： 该位仅在采用 M-Class 内核的器件上可用。要确定可用性，请参见具体器件的数据手册。
3： DSP ASE 并非在所有器件上都可用。要确定可用性，请参见具体器件数据手册中的 **“CPU”** 章节。

寄存器 50-16： Status：状态寄存器； CP0 寄存器 12，选择 0 （续）

bit 29 **CU1**：协处理器 1 可用位 (2)
该位允许访问协处理器 1。只有在配置了 FPU 时才能写入该位。如果不存在 FPU，则该位读为 0。
1 = 允许访问
0 = 不允许访问

bit 28 **CU0**：协处理器 0 可用位
该位控制对协处理器 0 的访问。
1 = 允许访问
0 = 不允许访问
当处理器运行于内核模式时，无论 CU0 位的状态如何，协处理器 0 总是可用。

bit 27 **RP**：降低功耗位
1 = 使能降低功耗模式
0 = 禁止降低功耗模式

bit 26 **FR**：浮点寄存器模式 (2)
1 = 浮点寄存器可以包含任意数据类型
0 = 浮点寄存器可以包含任意 32 位数据类型。 64 位数据类型存放在偶编号 - 奇编号寄存器对中

注： 对于没有 FPU 的 CPU，必须忽略对该位的写操作并读为 0。

bit 25 **RE**：反向字节序存储器引用使能位
用于使能在处理器运行于用户模式时的反向字节序存储器引用
1 = 用户模式使用反向字节序
0 = 用户模式使用所配置的字节序
在调试模式、内核模式和监督模式下的引用不会受该位状态影响。

bit 24 **MX**：MIPS DSP 资源使能位 (3)
执行任何 DSP ASE 指令之前，必须将该位置 1。在该位清零时尝试执行 DSP ASE 指令将导致 DSP 状态禁止异常。
1 = 使能访问
0 = 禁止访问

bit 23 **未实现**：读为 0

bit 22 **BEV**：引导异常向量控制位
控制异常向量的位置。
1 = 引导
0 = 正常

bit 21 **TS**：TLB 关闭控制位 (1)
指示 TLB 已检测到多个条目存在匹配。当发出的 TLBWI 或 TLBWR 指令将导致 TLB 关闭条件（允许完成的话）时，该位也将置 1。还会产生机器检查异常。
1 = TLB 关闭事件
0 = 无 TLB 关闭事件
软件只能通过向该位写入 0 来清零该位，不能强制进行 0 到 1 的转换。

bit 20 **SR**：软复位位
指示进入复位异常向量是由于软复位。
1 = 软复位。对于 PIC32 内核的任意复位类型，该位始终置 1。
0 = 不在 PIC32 器件中使用
软件只能通过向该位写入 0 来清零该位，不能强制进行 0 到 1 的转换。

注 1： 该位仅在采用 MPU 内核的器件上可用。要确定可用性，请参见具体器件数据手册中的 **“CPU”** 章节。
2： 该位仅在采用 M-Class 内核的器件上可用。要确定可用性，请参见具体器件的数据手册。
3： DSP ASE 并非在所有器件上都可用。要确定可用性，请参见具体器件数据手册中的 **“CPU”** 章节。

© 2013-2016 Microchip Technology Inc.

寄存器 50-16： Status：状态寄存器； CP0 寄存器 12，选择 0 （续）

bit 19 **NMI**：不可屏蔽中断位
指示进入复位异常向量是由于 NMI。
1 = NMI
0 = 非 NMI （软复位或复位）
软件只能通过向该位写入 0 来清零该位，不能强制进行 0 到 1 的转换。

bit 18 **IPL<7>**：中断优先级位
该位域是当前 IPL 的编码值 （0-256）。只有所请求的 IPL 大于该值时，才会发出中断。

bit 17 **未实现**：读为 0

bit 16-10 **IPL<6:0>**：中断优先级位
该位域是当前 IPL 的编码值 （0-256）。只有所请求的 IPL 大于该值时，才会发出中断。

bit 9-5 **未实现**：读为 0

bit 4 **UM**：用户模式位
该位指示处理器的基本工作模式。该位的编码为：
1 = 基本模式为用户模式
0 = 基本模式为内核模式
如果 ERL 或 EXL 置 1，则无论 UM 位状态如何，处理器也可处于内核模式。

bit 3 **未实现**：读为 0

bit 2 **ERL**：错误级别位
当发生复位、软复位、 NMI 或高速缓存错误异常时由处理器置 1。
1 = 错误级别
0 = 正常级别

当 ERL 置 1 时：
- 处理器运行于内核模式
- 中断被禁止
- ERET 指令将使用 ErrorEPC 寄存器而非 EPC 寄存器中保存的返回地址
- kuseg 的低 2^{29} 字节被视为非映射和非高速缓存区。这使得可以在存在高速缓存错误时访问主存储器。如果 ERL 位在处理器从 kuseg 中执行指令时置 1，则处理器的操作是未定义的。

bit 1 **EXL**：异常级别位
当发生除复位、软复位或 NMI 异常之外的任何其他异常时由处理器置 1。
1 = 异常级别
0 = 正常级别

当 EXL 置 1 时：
- 处理器运行于内核模式
- 中断被禁止

如果发生另一个异常， EPC、 BD 和 SRSCtl 将不会被更新。

bit 0 **IE**：中断允许位
用作软件和硬件中断的主允许控制：
1 = 允许中断
0 = 禁止中断
该位可以通过 DI 和 EI 指令单独修改。

注 1： 该位仅在采用 MPU 内核的器件上可用。要确定可用性，请参见具体器件数据手册中的 **“CPU”** 章节。
2： 该位仅在采用 M-Class 内核的器件上可用。要确定可用性，请参见具体器件的数据手册。
3： DSP ASE 并非在所有器件上都可用。要确定可用性，请参见具体器件数据手册中的 **“CPU”** 章节。

50.13.17 IntCtl：中断控制寄存器 （CP0 寄存器 12，选择 1）

Intctl 寄存器控制 PIC32 架构的向量间距。

寄存器 50-17： IntCtl：中断控制寄存器； CP0 寄存器 12，选择 1

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1
	—	PF	ICE	STKDEC<4:0>				
15:8	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
	CLREXL	APE	USESTK	—	—	—	VS<4:3>	
7:0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
	VS<2:0>			—	—	—	—	—

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-23 **未实现：**读为 0

bit 22 **PF：**向量预取使能位
1 = 使能向量预取
0 = 禁止向量预取

bit 21 **ICE：**中断链允许位
1 = 允许中断链
0 = 禁止中断链

bit 20-16 **STKDEC<4:0>：**堆栈指针递减位
对于自动现场保护功能，这是堆栈指针值递减的 4 字节字数。
31-4 = 指定要递减的字数
3-0 = 递减 3 个字 （12 字节）

bit 15 **CLREXL：**清零 KSU/ERL/EXL 位
对于自动现场保护功能和 IRET 指令，该位在自动现场保护和 IRET 中断链期间置 1 时会清零 KSU/ERL/EXL 位。
1 = 位被这些操作清零
0 = 位不被这些操作清零

bit 14 **APE：**自动现场保护使能位
1 = 使能自动现场保护
0 = 禁止自动现场保护

bit 13 **USEKSTK：**堆栈使用位
选择在中断自动现场保护期间使用的堆栈。
1 = 在 IAP 开始时使用当前 SRS 的 r29
在具有独立的用户模式和内核模式堆栈的环境中使用。在该情况下，IAP 期间使用的 SRS 的 r29 必须由软件预先初始化以保存内核模式堆栈指针。
0 = 在 IAP 开始时将上一个 SRS 的 r29 复制到当前 SRS
用于只有 1 个堆栈的裸机环境。

bit 12-10 **未实现：**读为 0

© 2013-2016 Microchip Technology Inc.

寄存器 50-17： IntCtl：中断控制寄存器； CP0 寄存器 12，选择 1 （续）

bit 9-5 **VS<4:0>：**向量间距位
这些位指定每个中断向量之间的间距。

编码	向量之间的间距 （十六进制）	向量之间的间距 （十进制）
0x00	0x000	0
0x01	0x020	32
0x02	0x040	64
0x04	0x080	128
0x08	0x100	256
0x10	0x200	512

所有其他值均保留。如果向这些位写入保留值，则处理器的操作是未定义的。

bit 4-0 **未实现：**读为 0

50.13.18 SRSCtl 寄存器 （CP0 寄存器 12，选择 2）

SRSCtl 寄存器控制处理器中的 GPR 影子寄存器集的操作。

表 50-26： 发生异常或中断时新 CSS 的来源

异常类型	位源	条件	备注
异常	ESS	全部	—
非向量中断	ESS	IV 位 （Cause<23>） = 0	视为异常
向量 EIC 中断	EICSS	IV 位 （Cause<23>） = 1 且 VEIC 位 （Config3<6>） = 1	来源是外部中断控制器。

寄存器 50-18： SRSCtl：影子寄存器集寄存器； CP0 寄存器 12，选择 2

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	R-0	R-1	R-1	R-1	U-0	U-0
	—	—	HSS<3:0>				—	—
23:16	U-0	U-0	R-x	R-x	R-x	R-x	U-0	U-0
	—	—	EICSS<3:0>				—	—
15:8	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
	ESS<3:0>				—	—	PSS<3:2>	
7:0	R/W-0	R/W-0	U-0	U-0	R-0	R-0	R-0	R-0
	PSS<1:0>		—	—	CSS<3:0>			

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-30 **未实现：**读为 0

bit 29-26 **HSS<3:0>：**影子寄存器集高编号位

该位包含该处理器实现的最高影子寄存器集编号。这些位的值为 0000 时，指示仅实现了正常 GPR。
1111 = 保留
0111 = 存在 8 个影子寄存器集
0110 = 保留
0101 = 保留
0100 = 保留
0011 = 存在 4 个影子寄存器集
0010 = 保留
0001 = 存在 2 个影子寄存器集
0000 = 存在 1 个影子寄存器集 （正常 GPR 集）

这些位的值也代表可以写入该寄存器的 ESS<3:0>、EICSS<3:0>、PSS<3:0> 和 CSS<3:0> 位或 SRSMap 寄存器的任意位的最大值。如果向其中的任何其他位写入的值大于该位的值，则处理器的操作是未定义的。

bit 25-22 **未实现：**读为 0

bit 21-18 **EICSS<3:0>：**外部中断控制器影子集位

EIC 中断模式影子集。对于每个中断请求，该位从外部中断控制器装载，并代替 SRSMap 寄存器来选择中断的当前影子集。

bit 17-16 **未实现：**读为 0

bit 15-12 **ESS<3:0>：**异常影子集位

该位指定由于向量中断之外的任何其他异常而进入内核模式时使用的影子集。如果软件写入该位的值大于 HSS<3:0> 位的值，则处理器的操作是未定义的。

bit 11-10 **未实现：**读为 0

© 2013-2016 Microchip Technology Inc.

寄存器 50-18： SRSCtl：影子寄存器集寄存器； CP0 寄存器 12，选择 2 （续）

bit 9-6 **PSS<3:0>：**先前影子集位

由于实现了GPR影子寄存器，所以在发生异常或中断时，将从CSS位复制该位。如果BEV（Status<22>） = 0，则 ERET 指令会将该值复制回 CSS 位中。

以下情况下，该位不会被更新：发生将 ERL 位 （Status<2>）设置为 1 的异常 （即，复位、软复位、 NMI 和高速缓存错误）时，进入 EJTAG 调试模式时，或在 EXL 位 （Status<1>） = 1 或 BEV = 1 时发生异常或中断时。在 ERL = 1 时发生异常时，该位不会被更新。

如果软件写入该位的值大于 HSS<3:0> 位中的值，则处理器的操作是未定义的。

bit 5-4 **未实现：**读为 0

bit 3-0 **CSS<3:0>：**当前影子集位

由于实现了 GPR 影子寄存器，所以该位是当前 GPR 集的编号。在发生任何中断或异常时，该位会被更新为新值；在执行 ERET 后会从 PSS 位中恢复。

表 50-26 列出了在发生异常或中断时， CSS<3:0> 位的各种更新来源。

以下情况下，该位不会被更新：发生将 ERL 位 （Status<2>）设置为 1 的异常 （即，复位、软复位、 NMI 和高速缓存错误）时，进入 EJTAG 调试模式时，或在 EXL 位 （Status<1>） = 1 或 BEV = 1 时发生异常或中断时。在 ERL = 1 或 BEV = 1 时执行 ERET 之后，该位也不会被更新。在 ERL = 1 时发生异常时，该位不会被更新。

软件只能通过写 PSS<3:0> 位并执行 ERET 指令来直接更改 CSS<3:0> 位的值。

50.13.19 SRSMap：寄存器 （CP0 寄存器 12，选择 3）

SRSMap 寄存器包含 8 个 4 位域，这些位域用于将向量编号映射到要在提供中断服务时使用的影子集编号。该寄存器的值不用于非中断异常或非向量中断（IV 位（Cause<23>）= 0 或 VS<4:0> 位 （IntCtl<9:5>）= 0）。这些情况下，影子集编号来自 ESS<3:0> 位 （SRSCtl<15:12>）。

如果 HSS<3:0> 位 （SRSCtL<29:26>）为 0，则软件读或写该寄存器的结果是不可预测的。

如果写入该寄存器任何位的值大于 HSS<3:0> 位的值，则处理器的操作是未定义的。

SRSMap 寄存器包含对应于向量编号 7 至 0 的影子寄存器集编号。对于同一影子集编号可以设立多个中断向量，产生从向量到单个影子寄存器集编号的多对一映射。

寄存器 50-19： SRSMap：影子寄存器集映射寄存器； CP0 寄存器 12，选择 3

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	SSV7<3:0>				SSV6<3:0>			
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	SSV5<3:0>				SSV4<3:0>			
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	SSV3<3:0>				SSV2<3:0>			
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	SSV1<3:0>				SSV0<3:0>			

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-28 **SSV7<3:0>：**影子集向量 7 位
对应于向量编号 7 的影子寄存器集编号。

bit 27-24 **SSV6<3:0>：**影子集向量 6 位
对应于向量编号 6 的影子寄存器集编号。

bit 23-20 **SSV5<3:0>：**影子集向量 5 位
对应于向量编号 5 的影子寄存器集编号。

bit 19-16 **SSV4<3:0>：**影子集向量 4 位
对应于向量编号 4 的影子寄存器集编号。

bit 15-12 **SSV3<3:0>：**影子集向量 3 位
对应于向量编号 3 的影子寄存器集编号。

bit 11-8 **SSV2<3:0>：**影子集向量 2 位
对应于向量编号 2 的影子寄存器集编号。

bit 7-4 **SSV1<3:0>：**影子集向量 1 位
对应于向量编号 1 的影子寄存器集编号。

bit 3-0 **SSV0<3:0>：**影子集向量 0 位
对应于向量编号 0 的影子寄存器集编号。

© 2013-2016 Microchip Technology Inc.

50.13.20 View_IPL 寄存器 （CP0 寄存器 12，选择 4）

该寄存器提供对 IPL<7:0> 位的读写访问，也可通过 Status 寄存器获取。使用该寄存器允许读 / 写优先级而无需从 Status 寄存器读取该域或将该域写入 Status 寄存器。

IPL 域是 Status 寄存器中的非连续位。在该寄存器中，所有 IPL 位表示为一个连续域。

寄存器 50-20： View_IPL：查看中断优先级寄存器； CP0 寄存器 12，选择 4

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
	—	—	—	—	—	—	IPL<7:6>	
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0
	IPL<5:0>						—	—

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-10 **未实现：**读为 0

bit 9-2 **IPL<7:0>：**这些位包含当前 IPL 的编码值 （0...256）。

bit 1-0 **未实现：**读为 0

50.13.21 SRSMAP2 寄存器 （CP0 寄存器 12，选择 5）

SRSMAP2 寄存器包含 2 个 4 位域，这些位域用于将向量编号映射到要在提供中断服务时使用的影子集编号。该寄存器的值不用于非中断异常或非向量中断（IV 位（Cause<23>）= 0 或 VS<4:0> 位 （IntCtl<9:5>） = 0）。这些情况下，影子集编号来自 ESS<3:0> 位 （SRSCtl<15:12>）。

如果 HSS<3:0> 位 （SRSCtl<9:6>）为 0，则软件读或写该寄存器的结果是不可预测的。

如果写入该寄存器任何位的值大于 HSS<3:0> 位的值，则处理器的操作是未定义的。

SRSMAP2 寄存器包含对应于向量编号 9 至 8 的影子寄存器集编号。对于同一影子集编号可以设立多个中断向量，产生从向量到单个影子寄存器集编号的多对一映射。

寄存器 50-21： SRSMAP2：影子寄存器集寄存器 2 ； CP0 寄存器 12，选择 5

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	SSV9<3:0>				SSV8<3:0>			

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-8 **未实现：**读为 0

bit 7-4 **SSV9<3:0>：**影子集向量 9 位
对应于向量编号 9 的影子寄存器集编号

bit 3-0 **SSV8<3:0>：**影子集向量 8 位
对应于向量编号 8 的影子寄存器集编号。

© 2013-2016 Microchip Technology Inc.

50.13.22 Cause 寄存器 （CP0 寄存器 13，选择 0）

Cause 寄存器主要描述最近异常的原因。此外，这些位还用于控制软件中断请求，以及向哪个向量分派中断。除 IP1、 IP0、 DC 和 IV 位外， Cause 寄存器中的所有其他位都是只读的。

表 50-27： Cause 寄存器 EXCCODE<4:0> 位

异常编码值		助记符	说明
十进制	十六进制		
0	0x00	Int	中断
1	0x01	MOD(1)	TLB 修改异常
2	0x02	TLBL(1)	TLB 异常 （装载或取指）
3	0x03	TLBS(1)	TLB 异常 （存储）
4	0x04	AdEL	地址错误异常 （装载或取指）
5	0x05	AdES	地址错误异常 （存储）
6	0x06	IBE	总线错误异常 （取指）
7	0x07	DBE	总线错误异常 （数据引用：装载或存储）
8	0x08	Sys	Syscall 异常
9	0x09	Bp	断点异常
10	0x0a	RI	保留的指令异常
11	0x0B	CPU	协处理器不可用异常
12	0x0C	Ov	算术溢出异常
13	0x0D	Tr	陷阱异常
14	0x0E	—	保留
15	0x0F	FPE(2)	浮点异常
16-18	0x10-0x0x12		保留
19	0x13	TLBRI(1)I	TLB 读禁止
20	0x14	TLBEI(1)	TLB 执行禁止
21-22	0x15-0x16	—	保留
23	0x17	WATCH(1)	引用 WatchHi/WatchLo 地址
24	0x18	MCheck(1)	机器检查
25	0x19	—	保留
26	0x1A	DSPDis	DSP ASE 状态禁止异常 (3)
27-31	0x1B-0x1F	—	保留

注 1： 该特性仅在采用 microAptiv™ MPU 内核的 PIC32 器件上可用。要确定可用性，请参见具体器件数据手册中的 **“CPU”** 章节。

2： 该特性仅限采用 M-Class 内核的 PIC32 器件上可用。要确定可用性，请参见具体器件的数据手册中的 **“FPU”** 章节。

3： DSP ASE 并非在所有器件上都可用。要确定可用性，请参见具体器件数据手册中的 **“CPU”** 章节。

寄存器 50-22： Cause：异常原因寄存器； CP0 寄存器 13，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-x	R-x	R-x	R-x	R/W-0	R-0	R-x	R-x
	BD	TI	CE<1:0>		DC	PCI	IC	AP
23:16	R/W-x	R/W-x	R-x	U-0	U-0	U-0	R-x	R-x
	IV	WP(1)	FDCI	—	—	—	RIPL<7:6>	
15:8	R-x	R-x	R-x	R-x	R-x	R-x	R/W-x	R/W-x
	RIPL<5:0>						IP1	IP0
7:0	U-0	R-x	R-x	R-x	R-x	R-x	U-0	U-0
	—	EXCCODE<4:0>					—	—

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31 **BD：**转移延时位

指示上一次异常是否在转移延时时隙中发生：
1 = 处于延时时隙中
0 = 不处于延时时隙中
发生异常时，只有在 EXL 位 （Status<1>）为 0 的情况下，处理器才会更新 BD。

bit 30 **TI：**定时器中断位

定时器中断。该位指示是否有定时器中断在等待处理 （类似于其他中断类型的 IP 位）：
1 = 有定时器中断在等待处理
0 = 没有定时器中断在等待处理

bit 29-28 **CE<1:0>：**协处理器异常位

发生协处理器不可用异常时引用的协处理器单元编号。该位在每次发生异常时由硬件装载，但对于除协处理器不可用异常之外的所有其他异常，它是不可预测的。
11 = 保留
10 = 保留
01 = 保留
00 = 协处理器 0

bit 27 **DC：**禁止 Count 寄存器位

在一些对功耗敏感的应用中，将不使用 Count 寄存器，可以停止它以避免不必要的翻转计数。
1 = 禁止 Count 寄存器计数
0 = 使能 Count 寄存器计数

bit 26 **PCI：**性能计数器中断位

1 = 性能计数器中断在等待处理
0 = 无性能计数器中断在等待处理

bit 25 **IC：**中断链位

指示中断链是否在执行最后一个 IRET 指令时发生。
1 = 在执行最后一个 IRET 指令时发生了中断链
0 = 在执行最后一个 IRET 指令时未发生中断链

bit 24 **AP：**中断自动现场保护异常位

指示在中断自动现场保护期间是否发生异常。
1 = 自动现场保护操作期间发生异常
0 = 自动现场保护操作期间未发生异常

注 1： 该位仅存在于采用 MPU 内核的 PIC32 器件上。请参见具体器件数据手册的 **“CPU”** 章节确定是否存在此位。

© 2013-2016 Microchip Technology Inc.

寄存器 50-22： Cause：异常原因寄存器； CP0 寄存器 13，选择 0 （续）

bit 23 **IV：**中断向量位

指示中断异常使用的是通用异常向量还是特殊中断向量。
1 = 使用特殊中断向量 （0x200）
0 = 使用通用异常向量 （0x180）
如果 IV 位 （Cause<23>）为 1 且 BEV 位 （Status<22>）为 0，则特殊中断向量代表向量中断表的基址。

bit 22 **WP：**监视异常待处理位 **(1)**

该位指示监视异常被延迟，原因是在检测到监视异常时状态位 EXL 或 ERL 被置 1。
1 = 监视异常在等待处理
0 = 无监视异常在等待处理

该位置 1 时，指示有一个待处理监视异常，一旦 Status EXL 和 ERL 位均为零，将导致异常启动。软件必须在监视异常处理程序中将该位清零以防止出现监视异常循环。

注： 软件不应在该位的值为零 0 时向其写入 1，这样会导致 0 到 1 的转换。如果此类转换是由软件引起的，则结果将是不可预测的。

bit 21 **FDCI：**快速调试通道中断位

该位指示有一个 FDC 中断在等待处理：
1 = 快速调试通道中断在等待处理
0 = 无快速调试通道中断在等待处理

bit 20-18 **未实现：**读为 0

bit 17-10 **RIPL<7:0>：**请求的中断优先级位

该位是所请求中断的编码值 （255-0）。值为 0 指示未请求任何中断。

bit 9-8 **IP<1:0>：**软件中断请求控制位

控制软件中断请求：
1 = 请求软件中断
0 = 未请求任何中断
这些位导出到系统中断控制器，用于在 EIC 中断模式下与其他中断源一起比较优先级。

bit 7 **未实现：**读为 0

bit 6-2 **EXCCODE<4:0>：**异常编码位

请参见表 50-27 获取异常编码列表。

bit 1-0 **未实现：**读为 0

注 1： 该位仅存在于采用 MPU 内核的 PIC32 器件上。请参见具体器件数据手册的 **“CPU”**章节确定是否存在此位。

50.13.23 View_RIPL 寄存器 （CP0 寄存器 13，选择 4）

该寄存器提供对 Cause 寄存器中的 RIPL 位的读访问。使用该寄存器允许读取请求的优先级，而无需从 Cause 寄存器读取该位。

寄存器 50-23： View_RIPL：查看请求的优先级寄存器； CP0 寄存器 13，选择 4

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	R-0	R-0
	—	—	—	—	—	—	RIPL<7:6>	
7:0	R-0	R-0	R-0	R-0	R-0	R-0	R/W-0	R/W-0
	RIPL<5:0>						IP<1:0>	

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-10 **未实现：**读为 0

bit 9-2 **RIPL<7:0>：**请求的中断优先级位
如果使能 EIC 中断模式，该位指示待处理中断的当前请求优先级的编码值 （0...255）。

bit 1-0 **IP<1:0>：**软件中断待处理位
如果未使能 EIC 中断模式，将控制待处理的 SW 中断。

© 2013-2016 Microchip Technology Inc.

50.13.24 NestedExc 寄存器 （CP0 寄存器 13，选择 5）

NestedExc 寄存器为一个只读寄存器，包含接受当前异常之前的 Status<1>（EXL）和 Status<2>（ERL）的值。

寄存器 50-24： NestedExc：嵌套异常寄存器； CP0 寄存器 13，选择 5

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	U-0	U-0	U-0	U-0	U-0	R/W-x	R/W-x	U-0
	—	—	—	—	—	NERL	NEXL	—

图注：	r = 保留位		
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-3 **未实现：**读为 0

bit 2 **NERL：**嵌套错误级别位

该位包含接受当前异常之前的 ERL 位的值。该位由所有会将状态寄存器中的 EXL 位或 ERL 位置 1 的异常更新。该位不会被调试异常更新。

bit 1 **NEXL：**嵌套异常级别位

该位包含接受当前异常之前的EXL位的值。该位由当EXL位未置1时会更新异常程序计数器的异常（MCheck、中断、地址错误、所有 TLB 异常、总线错误、CopUnusable、保留的指令、溢出、陷阱、Syscall 和 FPU 等）更新。对于这些异常类型，无论 Status EXL 的值为何，该寄存器位域都会更新。该位不会被更新 ErrorEPC 的异常类型 （复位、软复位、 NMI 和高速缓存错误）更新。此外，该位不会被调试异常更新。

bit 0 **未实现：**读为 0

50.13.25 EPC 寄存器 （CP0 寄存器 14，选择 0）

异常程序计数器（Exception Program Counter，EPC）是一个读 / 写寄存器，包含处理完异常之后继续进行处理的地址。 EPC 寄存器的所有位均为有效位，均可写。

对于同步 （精确）异常， EPC 包含以下值之一：

- 直接导致异常的指令的虚拟地址。
- 紧接在 BRANCH 或 JUMP 指令之前的虚拟地址 （如果导致异常的指令处于转移延时时隙中，并且 Cause 寄存器中的转移延时位置 1）。

发生新异常时，如果 Status 寄存器中的 EXL 位置 1，则处理器不会写 EPC 寄存器；但是，仍然可以通过 MTC0 指令写该寄存器。

由于 PIC32 系列实现了 MIPS16e® 或 microMIPS ASE，所以读 EPC 寄存器（通过 MFC0）会在目标 GPR 中返回以下值：

$GPR[rt] \leftarrow ExceptionPC_{31..1} \,||\, ISAMode_0$

即，异常 PC 的高 31 位与 ISA<1:0> 位 （Config3<15:14>）的低位相结合，并写入 GPR。

类似地，写 EPC 寄存器（通过 MTC0）时会获取 GPR 的值，并将该值分配到异常 PC 和 ISA<1:0> 位 （Config3<15:14>）中，如下：

$ExceptionPC \leftarrow GPR[rt]_{31..1} \,||\, 0$
$ISAMode \leftarrow 2\#0 \,||\, GPR[rt]_0$

即， GPR 的高 31 位写入异常 PC 的高 31 位，异常 PC 的低位会被清零。 ISA<1:0> 位（Config3<15:14>）的高位会被清零，低位则装入 GPR 的低位。

寄存器 50-25： EPC：异常程序计数器寄存器； CP0 寄存器 14，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	EPC<31:24>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	EPC<23:16>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	EPC<15:8>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	EPC<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **EPC<31:0>：**异常程序计数器位

© 2013-2016 Microchip Technology Inc.

50.13.26 NestedEPC 寄存器 （CP0 寄存器 14，选择 2）

NestedEPC 寄存器是一个读 / 写寄存器，操作与 EPC 寄存器相同，但以下两种情况例外：

- NestedEPC 寄存器忽略 Status<1> （EXL）的值，因此在发生任何异常 （包括嵌套异常）时都会进行更新。
- ERET/DERET/IERET 指令不使用 NestedEPC 寄存器。要返回到 NestedEPC 中存储的地址，软件必须将 NestedEPC 寄存器的值复制到 EPC 寄存器。

寄存器 50-26： NestedEPC：嵌套异常程序计数器寄存器； CP0 寄存器 14，选择 2

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	EPC<31:24>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	EPC<23:16>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	EPC<15:8>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	EPC<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **EPC<31:0>：**嵌套异常程序计数器位

这些位由当 EXL 位未置 1 时会更新异常程序计数器的异常 （MCheck、中断、地址错误、所有 TLB 异常、总线错误、CopUnusable、保留的指令、溢出、陷阱、Syscall 和 FPU 等）更新。对于这些异常类型，无论 EXL 的值为何，该寄存器位域都会更新。这些位不会被会更新异常程序计数器的异常类型 （复位、软复位、 NMI 和高速缓存错误）更新。此外，这些位不会被调试异常更新。

50.13.27 PRID 寄存器 （CP0 寄存器 15，选择 0）

处理器标识 （Processor Identification， PRID）寄存器是一个 32 位只读寄存器，包含处理器的制造商、制造商选项、处理器标识和版本级别的标识信息。

寄存器 50-27： PRID：处理器标识寄存器； CP0 寄存器 15，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	R-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-1
	COMPANYID<23:16>							
15:8	R-1	R-0	R-0	R-1	R-1	R-1	R-1	R-0
	PROCESSORID<15:8>							
7:0	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	MAJORREV<2:0>			MINORREV<2:0>			PATCHREV<1:0>	

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-24 **未实现：**读为 0

bit 23-16 **COMPANYID<23:16>：**公司标识位
在 PIC32 器件中，这些位包含值 1，指示 Imagination Technologies Ltd. 为处理器的制造商 / 设计公司。

bit 15-8 **PROCESSORID<15:8>：**处理器标识位
这些位允许软件区分各种类型的 Imagination Technologies Ltd. 的处理器。
0x9E = 微处理器内核

bit 7-5 **MAJORREV<2:0>：**处理器主版本标识位
这些位允许软件区分相同处理器类型的版本。该数字随处理器内核的主版本而增加。

bit 4-2 **MINORREV<2:0>：**处理器次版本标识位
该数字在处理器每次版本升级时增加，并在每次产生新的主版本时复位。

bit 1-0 **PATCHREV<1:0>：**处理器补丁级标识位
如果发布了补丁来修改较旧的处理器版本，则这些位的值增加。

© 2013-2016 Microchip Technology Inc.

50.13.28 Ebase 寄存器 （CP0 寄存器 15，选择 1）

Ebase 寄存器是一个读 / 写寄存器，包含异常向量的基址 （在 BEV 位 （Status<22>）等于 0 时使用）和一个只读 CPU 编号值 （软件可以使用它来区分多处理器系统中的不同处理器）。

Ebase 寄存器使得软件可以识别多处理器系统中的特定处理器，允许每个处理器的异常向量可以不同，特别是在由异构处理器组成的系统中。当 BEV 为 0 时， Ebase 寄存器的 bit 31-12 将与 0 组合，构成异常向量的基址。当 BEV 位为 1 时，或者对于任何 EJTAG 调试异常，异常向量基址将来自固定的默认值。 Ebase 寄存器的 bit 31-12 的复位状态会将异常基址寄存器初始化为 0x80000000。

Ebase 寄存器的 bit 31 和 bit 30 固定为值 2#10，以强制使异常基址处于 kseg0 或 kseg1 非映射虚拟地址段中。

如果要更改异常基址寄存器的值，则必须在 BEV 位等于 1 时执行。如果 BEV 位 （Status<22>）为 0 时， Ebase<17:0> 位中写入其他值，则处理器的操作是未定义的。

通过组合 Ebase 寄存器的 bit 31-20，异常向量的基址可以放置在任意 4 KB 页边界处。

寄存器 50-28： Ebase：异常基址寄存器； CP0 寄存器 15，选择 1

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-1	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	EBASE<17:12>					
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	EBASE<11:4>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R-0	R-0
	EBASE<3:0>				—	—	CPUNUM<9:8>	
7:0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	CPUNUM<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31 **未实现：**读为 1

bit 30 **未实现：**读为 0

bit 29-12 **EBASE<17:0>：**异常向量基址位
当 BEV 位 （Status<22>）为 0 时，这些位与 bit 31-30 一起指定异常向量的基址。

bit 11-10 **未实现：**读为 0

bit 9-0 **CPUNUM<9:0>：**CPU 编号位
这些位指定多处理器系统中的 CPU 编号，软件可以使用它来将特定处理器与其他处理器区分开来。在单处理器系统中，该值设置为 0。

© 2013-2016 Microchip Technology Inc.

50.13.29 CDMMBase 寄存器 （CP0 寄存器 15，选择 2）

通用器件存储器映射（Common Device Memory Map，CDMM）的 36 位物理基址由该寄存器定义。由于 PIC32 是 32 位器件，因此地址的高 4 位为零。CDMM 是物理地址空间区域，保留用于映射 MIPS 处理器中的 I/O 器件配置寄存器。CDMM 帮助将各种器件映射聚集到一个区域，避免存储器地址空间碎片。在 PIC32 器件上， CDMM 包含快速调试通道 （Fast Debug Channel，FDC）控制寄存器。

寄存器 50-29： CDMMBase：通用器件存储器映射基址寄存器； CP0 寄存器 15，选择 2

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	CDMMUA<20:13>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	CDMMUA<12:5>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-0	R-0	R-0
	CDMMUA<4:0>					EN	CI	CDMMSize<8>
7:0	R-0	R-0	R-0	R-0	R-0	R-0	R-1	R-0
	CDMMSize<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-11 **CDMMUA<20:0>**：CDMM 高地址位
映射的存储器寄存器的物理基址的 bit 35:15。未实现的位忽略写操作并在读取时返回 0。

bit 10 **EN**：CDMM 使能位
1 = 使能 CDMM 区域
0 = 禁止 CDMM 区域

bit 9 **CI**：器件寄存器区域保留位
1 = CDMM 的第一个 64 字节器件寄存器区域保留为以下寄存器：用于管理 CDMM 区域操作，并且不是 IO 器件寄存器的其他寄存器
0 = CDMM 的第一个 64 字节器件寄存器区域未保留

bit 8-0 **CDMMSize<8:0>**：CDMM 大小位
这些位指示 CPU 内核中已实例化的 64 字节器件寄存器区域的数量。
111111111 = 512 个器件寄存器区域
•
•
•
000000000 = 1 个器件寄存器区域

© 2013-2016 Microchip Technology Inc.

50.13.30 Config 寄存器 （CP0 寄存器 16，选择 0）

Config 寄存器指定各种配置和功能信息。Config 寄存器中的大部分字段由硬件在复位异常过程中初始化，或者为常量。

> **注：** 内核配置寄存器位域和默认值可能因器件不同而异。关于具体器件的内核配置，请参见具体器件数据手册的 **“内核配置”** 章节。

寄存器 50-30： Config：配置寄存器； CP0 寄存器 16，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	r-1	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R-x
	—	K23<2:0>(1)			KU<2:0>(1)			ISP
23:16	R-x	R-x	R-x	R-x	U-0	R-x	R-x	R-x
	DSP	UDI	SB	MDU	—	MM<1:0>(2)		DS(1)
								BM(2)
15:8	R-0	R-0	R-0	R-0	R-0	R-1	R-0	R-x
	BE	AT<1:0>		AR<2:0>			MT<2:1>	
7:0	R-x	U-0	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x
	MT<0>	—	—	—	—	K0<2:0>		

图注：	r = 保留位		
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31 **保留：** 该位硬连线为 1，指示存在 Config1 寄存器。

bit 30-28 **K23<2:0>：** kseg2 和 kseg3 位 (1)
这些位控制 kseg2 和 kseg3 地址段的高速缓存功能。
位编码请参见表 50-28。

bit 27-25 **KU<2:0>：** kuseg 和 useg 位 (1)
这些位控制 kuseg 和 useg 地址段的高速缓存功能。
位编码请参见表 50-28。

bit 24 **ISP：** 指令中间结果暂存 RAM 位
该位指示是否实现了指令中间结果暂存 RAM。
1 = 实现了指令中间结果暂存 RAM
0 = 未实现指令中间结果暂存 RAM

bit 23 **DSP：** 数据中间结果暂存 RAM 位
该位指示是否实现了数据中间结果暂存 RAM。
1 = 实现了数据中间结果暂存 RAM
0 = 未实现数据中间结果暂存 RAM

注 1： 该位仅在采用 MCU 微处理器内核的器件上可用。在采用 MPU 微处理器内核的器件上，该位保留，且始终读为 0。请参见具体器件数据手册的 **“CPU”** 章节确定该位是否可用。

2： 该位仅在采用 MPU 微处理器内核的器件上可用。请参见具体器件数据手册的 **“CPU”** 章节确定该位是否可用。

寄存器 50-30： Config：配置寄存器； CP0 寄存器 16，选择 0 （续）

bit 22 **UDI：**用户定义位
该位指示是否实现了 CorExtend 用户定义指令。
1 = 实现了用户定义指令
0 = 未实现用户定义指令

bit 21 **SB：**SimpleBE 位
该位指示是否使能了 SimpleBE 总线模式。
1 = 内部总线接口上仅允许使能简单字节
0 = 内部总线接口上无使能的保留字节
该位硬连线为 1，指示内部总线接口上仅允许使能简单字节。

bit 20 **MDU：**乘法 / 除法单元位
1 = 迭代、高效 MDU
0 = 快速、高性能 MDU
该位硬连线为 0，指示快速、高性能 MDU。

bit 19 **未实现：**读为 0

bit 18-17 **MM<1:0>：**合并模式位 (2)
11 = 保留
10 = 允许合并
01 = 保留
00 = 不允许合并

bit 16 **DS：**双 SRAM 位 (1)
1 = 双指令 / 数据 SRAM 内部总线接口
0 = 统一的指令 / 数据 SRAM 内部总线接口

BM：突发模式位 (2)
该位硬连线为 0，指示突发顺序是连续的。

bit 15 **BE：**大尾数位
指示处理器运行的尾数模式， PIC32 总是采用小尾数模式。
1 = 大尾数
0 = 小尾数

bit 14-13 **AT<1:0>：**架构类型位
处理器实现的架构类型。该位总是为 00，以指示 MIPS32 架构。

bit 12-10 **AR<2:0>：**架构版本级别位
架构版本级别。该位总是为 001，以指示 MIPS32 版本 2。
111 = 保留
110 = 保留
101 = 保留
100 = 保留
011 = 保留
010 = 保留
001 = 版本 2
000 = 版本 1

注 1： 该位仅在采用 MCU 微处理器内核的器件上可用。在采用 MPU 微处理器内核的器件上，该位保留，且始终读为 0。请参见具体器件数据手册的 **“CPU”**章节确定该位是否可用。

2： 该位仅在采用 MPU 微处理器内核的器件上可用。请参见具体器件数据手册的 **“CPU”**章节确定该位是否可用。

© 2013-2016 Microchip Technology Inc.

寄存器 50-30： Config：配置寄存器； CP0 寄存器 16，选择 0 （续）

bit 9-7 **MT<2:0>：**MMU 类型位
采用 MCU 微处理器内核的 PIC32 器件使用固定映射 MMU。
采用 MPU 微处理器内核的 PIC32 器件使用基于 TLB 的 MMU。
111 = 保留
110 = 保留
101 = 保留
100 = 保留
011 = 固定映射
010 = 保留
001 = 标准 TLB
000 = 保留

bit 6-3 **未实现：**读为 0

bit 2-0 **K0<2:0>：**Kseg0 位
Kseg0 一致性算法。位编码请参见表 50-28。

注 1： 该位仅在采用 MCU 微处理器内核的器件上可用。在采用 MPU 微处理器内核的器件上，该位保留，且始终读为 0。请参见具体器件数据手册的 **“CPU”**章节确定该位是否可用。

2： 该位仅在采用 MPU 微处理器内核的器件上可用。请参见具体器件数据手册的 **“CPU”**章节确定该位是否可用。

表 50-28： 高速缓存一致性属性

K0<2:0> 值	高速缓存一致性属性
011	可高速缓存、非一致性、回写和写分配
010	非高速缓存
001	可高速缓存、非一致性、直写和写分配
000	可高速缓存、非一致性、直写和无写分配

50.13.31 Config1 寄存器 （CP0 寄存器 16，选择 1）

Config1 寄存器是 Config 寄存器的辅助寄存器，保存关于内核所具备功能的附加信息。 Config1 寄存器中的所有位域都是只读的。

注： 内核配置寄存器位域和默认值可能因器件不同而异。关于具体器件内核配置，请参见具体器件数据手册的 **“内核配置”**章节。

寄存器 50-31： Config1：配置寄存器 1 ； CP0 寄存器 16，选择 1

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	r-1	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	—	MMU Size<5:0>(1)						IS<2>
23:16	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	IS<1:0>(1)		IL<2:0>(1)			IA<2:0>(1)		
15:8	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	DS<2:0>(1)			DL<2:0>(1)			DA<2:1>(1)	
7:0	R-x	U-0	U-0	R-x	R-x	R-x	R-x	R-x
	DA<0>	—	—	PC	WR	CA	EP	FP

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31 **保留：**该位硬连线为 1，指示存在 Config2 寄存器。

bit 30-25 **MMU Size<5:0>：**包含 TLB 条目数减 1 的数量 (1)

11111 = 32 个 TLB 条目
·
·
·
00000 = 无 TLB 条目

bit 24-22 **IS<2:0>：**指令高速缓存集位 (1)

包含每路的指令高速缓存集数。
0x0 = 64
0x1 = 128
0x2 = 256
0x3 = 512
0x4 = 1024
0x5 = 保留
0x6 = 保留
0x7 = 保留

bit 21-19 **IL<2:0>：**指令高速缓存行位 (1)

包含指令高速缓存行大小。
0x0 = 没有指令高速缓存
0x3 = 16 字节
所有其他值均保留。

注 1： 在采用 MCU 微处理器内核的 PIC32 器件上，这些位保留，且始终读为 0。请参见具体器件数据手册的 **“CPU”**章节确定该位是否可用。

© 2013-2016 Microchip Technology Inc.

寄存器 50-31： Config1：配置寄存器 1； CP0 寄存器 16，选择 1（续）

bit 18-16 **IA<2:0>：**指令高速缓存相关性位 **(1)**
包含指令高速缓存相关性级别。
0x0 = 直接映射
0x1 = 2 路
0x2 = 3 路
0x3 = 4 路
0x4 = 保留
0x5 = 保留
0x6 = 保留
0x7 = 保留

bit 15-13 **DS<2:0>：**数据高速缓存设置位 **(1)**
包含每路的数据高速缓存设置数。
0x0 = 64
0x1 = 128
0x2 = 256
0x3 = 512
0x4 = 1024
0x5 = 保留
0x6 = 保留
0x7 = 保留

bit 12-10 **DL<2:0>：**数据高速缓存行位 **(1)**
包含数据高速缓存行大小。
0x0 = 没有指令高速缓存
0x3 = 16 字节
所有其他值均保留。

bit 9-7 **DA<2:0>：**数据高速缓存相关性位 **(1)**
包含为数据高速缓存设置的相关性的类型。
0x0 = 直接映射
0x1 = 2 路
0x2 = 3 路
0x3 = 4 路
0x4 = 保留
0x5 = 保留
0x6 = 保留
0x7 = 保留

bit 6-5 **未实现：**读为 0

bit 4 **PC：**性能计数器位
实现了性能计数器寄存器。
1 = 处理器内核包含性能计数器
0 = 处理器内核不包含性能计数器

bit 3 **WR：**监视寄存器存在位
1 = 不存在任何监视寄存器
0 = 存在一个或多个监视寄存器

bit 2 **CA：**代码压缩实现位
1 = 实现了 MIPS16e
0 = 不存在 MIPS16e

bit 1 **EP：**EJTAG 存在位
1 = 实现了 EJTAG 单元
0 = 未实现 EJTAG 单元

bit 0 **FP：**浮点单元位
1 = 实现了浮点单元
0 = 未实现浮点单元

注 1： 在采用 MCU 微处理器内核的 PIC32 器件上，这些位保留，且始终读为 0。请参见具体器件数据手册的 **“CPU”**章节确定该位是否可用。

50.13.32 Config2 （CP0 寄存器 16，选择 2）

Config2 寄存器是 Config 寄存器的辅助寄存器，保留用于保存附加功能的信息。 Config2 分配用于显示 2/3 级高速缓存的配置。这些位复位为 0，因为 PIC32 内核不支持 L2/L3 高速缓存。Config2 寄存器中的所有位都是只读的。

注： 内核配置寄存器位域和默认值可能因器件不同而异。关于具体器件内核配置，请参见具体器件数据手册的 **“内核配置”** 章节。

寄存器 50-32： Config2：配置寄存器 2 ； CP0 寄存器 16，选择 2

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	r-1	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—

图注：	r = 保留位		
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31 **保留：** 该位硬连线为 1，指示存在 Config3 寄存器。

bit 30-0 **未实现：** 读为 0

© 2013-2016 Microchip Technology Inc.

50.13.33 Config3 寄存器 （CP0 寄存器 16，选择 3）

Config3 寄存器用于保存关于附加功能的信息。 Config3 寄存器中的所有位域都是只读的。

注：	内核配置寄存器位域和默认值可能因器件不同而异。关于具体器件内核配置，请参见具体器件数据手册的 **“内核配置”** 章节。

寄存器 50-33： Config3：配置寄存器 3 ； CP0 寄存器 16，选择 3

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	r-1	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	R-0	R-x	R-0	R-0	R-0	R-x	R/W-y
	—	IPLW<1:0>		MMAR<2:0>			MCU	ISAONEXC
15:8	R-y	R-y	R-x	R-x	R-x	R-x	U-0	R-x
	ISA<1:0>		ULRI	RXI(1)	DSP2P(2)	DSPP(2)	—	ITL
7:0	U-0	R-x	R-x	R-x	R-x	U-0	U-0	R-x
	—	VEIC	VINT	SP(1)	CDDM	—	—	TL

图注：			
U = 未实现位，读为 0	y = POR 时根据 BOOTISA 配置位 （CFG0<6>）设置的值		
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31 **保留：**该位硬连线为 1，指示存在 Config4 寄存器。

bit 30-23 **未实现：**读为 0

bit 22-21 **IPLW<1:0>：**Status IPL 和 Cause RIPL 的宽度位

11 = 保留
10 = 保留
01 = IPL 和 RIPL 位宽度为 8 位
00 = IPL 和 RIPL 位宽度为 6 位

如果 IPL 位域为 8 位宽， Status 寄存器的 bit 18 和 bit 16 分别用作 IPL 的最高位和次高位。

如果 RIPL 位域为 8 位宽， Cause 寄存器的 bit 17 和 bit 16 分别用作 RIPL 的最高位和次高位。

注： PIC32 器件内核使用 8 位 IPL 和 RIPL 位域，所以这些位设置为 01。

bit 20-18 **MMAR<2:0>：**microMIPS 架构版本级别位

111 = 保留
110 = 保留
101 = 保留
100 = 保留
011 = 保留
010 = 保留
001 = 保留
000 = 版本 1

bit 17 **MCU：**MIPS MCU ASE 实现位

1 = 实现了 MCU ASE
0 = 未实现 MCU ASE

bit 16 **ISAONEXC：**异常时的 ISA 位

反映跳转到异常时使用的 ISA。影响其向量为相对 EBASE 的偏移量的异常。该位的复位值由 BOOTISA 配置位 （CFG0<6>）决定。

1 = microMIPS 用作异常向量的入口
0 = MIPS32 ISA 用作异常向量的入口

注 1： 该位仅在采用 MPU 微处理器内核的器件上可用。请参见具体器件数据手册的 **“CPU”** 章节确定该位是否可用。

2： 该位并非在所有器件上均可用。请参见具体器件数据手册的 **“CPU”** 章节确定该位是否可用。

寄存器 50-33： Config3：配置寄存器 3 ； CP0 寄存器 16，选择 3 （续）

bit 15-14 **ISA<1:0>：**指示指令集可用性
该位的复位值由 BOOTISA 配置位 （CFG0<6>）决定。
11 = MIPS32 和 microMIPS 均已实现；从复位退出时使用 microMIPS
10 = MIPS32 和 microMIPS 均已实现；从复位退出时使用 MIPS32 ISA
01 = 仅实现了 microMIPS
00 = 仅实现了 MIPS32

bit 13 **ULRI：**UserLocal 寄存器实现位
该位指示是否已实现 UserLocal 协处理器 0 寄存器。
1 = 实现了 UserLocal 寄存器
0 = 未实现 UserLocal 寄存器

bit 12 **RXI：**PageGrain 中的 RIE 和 XIE 实现位 **(1)**
1 = 实现了 RIE 和 XIE 位
0 = 未实现 RIE 和 XIE 位

bit 11 **DSP2P：**MIPS DSP ASE 版本 2 存在位 **(2)**
1 = 存在 DSP 版本 2
0 = 不存在 DSP 版本 2

bit 10 **DSPP：**MIPS DSP ASE 存在位 **(2)**
1 = 存在 DSP
0 = 不存在 DSP

bit 9 **未实现：**读为 0

bit 8 **ITL：**iFlowtrace® 硬件位
该位指示存在 iFlowtrace 硬件
1 = 已在内核中实现 iFlowtrace
0 = 未在内核中实现 iFlowtrace

bit 7 **未实现：**读为 0

bit 6 **VEIC：**外部向量中断控制器位
该位指示是否已实现对外部中断控制器 （EIC）的支持。
1 = 实现了对 EIC 中断模式的支持
0 = 未实现对 EIC 中断模式的支持
PIC32 器件内部实现了一个 MIPS “外部中断控制器”；因此该位读为 1。

bit 5 **VINT：**向量中断位
该位指示是否实现了向量中断。
1 = 实现了向量中断
0 = 未实现向量中断
在 PIC32 内核上，由于已实现向量中断，该位始终为 1。

bit 4 **SP：**小页面位 **(1)**
该位指示是否已实现对小页面 （1 KB）的支持。
1 = 实现了小页面支持
0 = 未实现小页面支持

bit 3 **CDMM：**通用器件存储器映射位
该位指示是否已实现对通用器件存储器映射的支持。
1 = 实现了 CDMM
0 = 未实现 CDMM

bit 2-1 **未实现：**读为 0

bit 0 **TL：**跟踪逻辑位
该位指示是否已实现跟踪逻辑。
1 = 实现了跟踪逻辑
0 = 未实现跟踪逻辑

注 1： 该位仅在采用 MPU 微处理器内核的器件上可用。请参见具体器件数据手册的 **“CPU”**章节确定该位是否可用。

2： 该位并非在所有器件上均可用。请参见具体器件数据手册的 **“CPU”**章节确定该位是否可用。

© 2013-2016 Microchip Technology Inc.

50.13.34 Config4 寄存器 （CP0 寄存器 16，选择 4）

Config4 寄存器是 Config 寄存器的辅助寄存器，保存关于内核所具备功能的附加信息。在 PIC32 器件上， Config4 寄存器指示存在 Config5 寄存器。 Config4 寄存器中的所有位域都是只读的。

注：	内核配置寄存器位域和默认值可能因器件不同而异。关于具体器件内核配置，请参见具体定器件数据手册的 **“内核配置”** 章节。

寄存器 50-34： Config4：配置寄存器 4 ； CP0 寄存器 16，选择 4

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	r-1	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—

图注：	r = 保留		
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31 **保留：**该位硬连线为 1，指示存在 Config5 寄存器。

bit 30-0 **未实现：**读为 0

50.13.35 Config5 寄存器 （CP0 寄存器 16，选择 5）

Config5 寄存器是 Config 寄存器的辅助寄存器，保存关于内核所具备功能的附加信息。 Config5 寄存器指示是否已实现嵌套故障功能。 Config5 寄存器中的所有位域都是只读的。

注： 内核配置寄存器位域和默认值可能因器件不同而异。关于具体器件内核配置，请参见具体器件数据手册的 **“内核配置”**章节。

寄存器 50-35： Config5：配置寄存器 5 ； CP0 寄存器 16，选择 5

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	U-0	U-0	U-0	U-0	U-0	R/W-0	U-0	R-1
	—	—	—	—	—	UFR(1)	—	NF

图注：	r = 保留		
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-3 **未实现：**读为 0

bit 2 **UFR：**用户模式访问位 (1)
该特性允许用户模式通过 CTC1 和 CFC1 指令访问 $Status_{FR}$。
1 = 允许用户模式 $Status_{FR}$ 指令
0 = 不允许用户模式 $Status_{FR}$ 指令

bit 1 **未实现：**读为 0

bit 0 **NF：**嵌套故障位
1 = 实现了嵌套故障功能
0 = 未实现嵌套故障功能

注 1： 该位仅在采用 M-Class 内核的器件上可用。请参见具体器件数据手册的 **“CPU”**章节确定该位是否可用。

© 2013-2016 Microchip Technology Inc.

50.13.36 Config7 寄存器 （CP0 寄存器 16，选择 7）

Config7 寄存器包含实现具体配置的信息。写入其中一些位以禁止微处理器内核中的特定性能增强功能。

注：	内核配置寄存器位域和默认值可能因器件不同而异。关于具体器件内核配置，请参见具体器件数据手册的 **“内核配置”** 章节。

寄存器 50-36： Config7：配置寄存器 7 ； CP0 寄存器 16，选择 7

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-1	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	WII	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31 **WII**：等待 IE 忽略位
指示该处理器将允许中断以解锁 WAIT 指令，即使 IE 在阻止中断发生时也是如此。这会避免在将 WAIT 指令用于下半部分中断服务时出现问题。对于采用微处理器内核的器件，该位始终读为 1。

bit 30-0 **未实现**：读为 0

50.13.37 LLAddr 寄存器 （CP0 寄存器 17，选择 0）（仅限 MPU）

LLAddr 寄存器包含最近链接装载（Load Linked，LL）指令所读取的物理地址。该寄存器仅用于诊断目的，且在正常工作期间不提供任何功能。

寄存器 50-37： LLAddr：链接装载地址寄存器； CP0 寄存器 17，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	R-x	R-x	R-x	R/x
	—	—	—	—	PAddr<27:24>			
23:16	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	PAddr< 23:16>							
15:8	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	PAddr<15:8>							
7:0	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	PAddr<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-28 **未实现：**读为 0

bit 27-0 **PAddr<27:0>：**链接装载指令物理地址编码位
这些位包含保存最近链接装载指令所读取的物理地址。

© 2013-2016 Microchip Technology Inc.

50.13.38 WatchLo 寄存器 （CP0 寄存器 18，选择 0-3）（仅限 MPU）

WatchLo 和 WatchHi 寄存器共同提供一个监视点调试功能接口，如果指令或数据访问的地址与这两个寄存器中指定的地址匹配，便产生一个监视异常。就此而言，表明它们复制了 EJTAG 调试方法中的一些功能。监视异常仅在 Status 寄存器中的 EXL 和 ERL 位均为 0 时才会产生。如果其中任何一位为1，Cause寄存器中的WP位将置1，监视异常将推迟到EXL和ERL位均为0时产生。

WatchLo 寄存器指定要匹配的虚拟基址和引用类型 （取指、装载或存储）。

寄存器 50-38： WatchLo：看门狗调试寄存器低位； CP0 寄存器 18，选择 0-3

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	VAddr<28:21>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	VAddr< 20:13>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	VAddr<12:5>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-0	R/W-0	R/W-0
	VAddr<4:0>					I	R	W

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-3 **VAddr<28:0>：**虚拟地址匹配位
该位域指定要匹配的虚拟地址。这是一个双字地址，原因是 bit <2:0> 用于控制匹配类型。

bit 2 **I：**取指监视地址异常使能位
1 = 为与地址匹配的取指使能监视异常
0 = 不使能监视异常

bit 1 **R：**取指监视装载异常使能位
1 = 为与地址匹配的装载使能监视异常
0 = 不使能监视异常

bit 0 **W：**取指监视存储异常使能位
1 = 为与地址匹配的存储使能监视异常
0 = 不使能监视异常

50.13.39 WatchHi 寄存器 （CP0 寄存器 19，选择 0-3）（仅限 MPU）

WatchLo 和 WatchHi 寄存器共同提供一个监视点调试功能接口，如果指令或数据访问的地址与这两个寄存器中指定的地址匹配，便产生一个监视异常。就此而言，表明它们复制了 EJTAG 调试方法中的一些功能。监视异常仅在 Status 寄存器中的 EXL 和 ERL 位均为 0 时才会产生。如果其中任何一位为1，Cause寄存器中的WP位将置1，监视异常将推迟到EXL和ERL位均为0时产生。

WatchHi 寄存器包含限定 WatchLo 寄存器中指定的虚拟地址的信息：ASID、全局 （G）位和可选地址掩码。如果 G 位为 1，任何与指定地址匹配的虚拟地址引用将导致监视异常。如果 G 位为 0，仅当虚拟地址引用满足以下条件时才导致监视异常：WatchHi 寄存器中的 ASID 值与 EntryHi 寄存器中的 ASID 值匹配。可选掩码域提供地址掩码以限定 WatchLo 中指定的地址。

寄存器 50-39： WatchHi：看门狗调试寄存器高位； CP0 寄存器 19，选择 0-3

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-x	R/W-x	U-0	U-0	U-0	U-0	U-0	U-0
	M	G	—	—	—	—	—	—
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	ASID<7:0>							
15:8	U-0	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
	—	—	—	—	Mask<8:5>			
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	W-x, HS, CS	W-x, HS, CS	W-x, HS, CS
	Mask<4:0>					I	R	W

图注：	HS = 硬件置 1 位	CS = 软件清零位	
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31 **M**：监视寄存器对检测位
指示除了该寄存器外是否还存在其他监视寄存器对

bit 30 **G**：全局位
1 = 任何与 WatchLo 寄存器中指定的值匹配的地址导致监视异常
0 = ASID<7:0> 必须与 EntryHi 寄存器的 ASID 位匹配才会导致监视异常

bit 29-24 **未实现**：读为 0

bit 23-16 **ASID<7:0>**：地址空间 ID 监视异常匹配位
在 WatchHi 寄存器中的 G 位为 0 时，需与 EntryHi 寄存器中的 ASID 值匹配的 ASID 值。

bit 15-12 **未实现**：读为 0

bit 11-3 **Mask<8:0>**：虚拟地址匹配掩码位
限定 WatchLo 寄存器中的地址的位掩码。该域中任何位置 1 时将禁止相应的地址位参与地址匹配。

bit 2 **I**：取指条件匹配位
当取指条件与监视寄存器对中的值匹配时，该位由硬件置 1。置 1 后，该位保持置 1 直至由软件清零，可通过写入 1 来完成。

bit 1 **R**：装载条件匹配位
当装载条件与监视寄存器对中的值匹配时，该位由硬件置 1。置 1 后，该位保持置 1 直至由软件清零，可通过写入 1 来完成。

bit 0 **W**：存储条件匹配位
当存储条件与监视寄存器对中的值匹配时，该位由硬件置 1。置 1 后，该位保持置 1 直至由软件清零，可通过写入 1 来完成。

© 2013-2016 Microchip Technology Inc.

50.13.40 Debug 寄存器 （CP0 寄存器 23，选择 0）

Debug 寄存器用于控制调试异常，并提供关于调试异常原因的信息，以及何时由于调试模式下的一般异常而重新进入调试异常向量处。只读信息位在以下情况下更新：每次发生调试异常时，或在已处于调试模式时发生一般异常。

在非调试模式下读取时，只有 DM 位和 VER<2:0> 位有效；所有其他位和位域的值是不可预测的。如果在非调试模式下写 Debug 寄存器，则处理器的操作是未定义的。

一些位和位域只有在发生调试异常和 / 或调试模式下的异常时才会更新，如下所示：

- DSS、 DBP、 DDBL、 DDBS、 DIB 和 DINT 在发生调试异常和调试模式下的异常时更新
- DEXCCODE<4:0> 在发生调试模式下的异常时更新，发生调试异常之后值是未定义的
- HALT 和 DOZE 在发生调试异常时更新，在发生调试模式下的异常之后值是未定义的
- DBD 在发生调试异常和调试模式下的异常时更新

在正常模式下读取时，除 VER<2:0> 和 DM 位外，所有其他位都是未定义的。

寄存器 50-40： Debug：调试异常寄存器； CP0 寄存器 23，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-0	R-0	R-0	R/W-0	R-x	R-x	R/W-1	R/W-0
	DBD	DM	NODCR	LSNM	DOZE	HALT	COUNTDM	IBUSEP
23:16	U-0	U-0	R/W-0	R/W-0	R-0	R-0	R-1	R-0
	—	—	DBUSEP	IEXI	DDBSIMPR	DDBLIMPR	VER<2:1>	
15:8	R-1	R-x	R-x	R-x	R-x	R-x	R-0	R/W-0
	VER<0>	DEXCCODE<4:0>					NOSST	SST
7:0	U-0	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	—	DIBIMPR	DINT	DIB	DDBS	DDBL	DBP	DSS

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31 **DBD**：转移延时调试异常位
指示上一次调试异常或调试模式下的异常是否是在转移延时时隙中发生的：
1 = 处于延时时隙中
0 = 不处于延时时隙中

bit 30 **DM**：调试模式位
指示处理器是否工作于调试模式：
1 = 处理器工作于调试模式
0 = 处理器工作于非调试模式

bit 29 **NODCR**：调试控制寄存器位
指示 dseg 存储器段是否存在，以及调试控制寄存器是否可访问：
1 = 不存在 dseg
0 = 存在 dseg

bit 28 **LSNM**：装载 / 存储访问控制位
控制 dseg 和主存储器之间的装载 / 存储访问：
1 = dseg 地址范围内的装载 / 存储访问至主存储器
0 = dseg 地址范围内的装载 / 存储访问至 dseg

寄存器 50-40： Debug：调试异常寄存器； CP0 寄存器 23，选择 0 （续）

bit 27 **DOZE**：低功耗模式调试异常位
指示发生调试异常时处理器是否处于某种类型的低功耗模式。
1 = 发生调试异常时处理器处于低功耗模式
0 = 发生调试异常时处理器未处于低功耗模式

bit 26 **HALT**：系统总线时钟停止位
指示发生调试异常时内部系统总线时钟是否停止。
1 = 内部系统总线时钟在运行
0 = 内部系统总线时钟已停止

bit 25 **COUNTDM**：Count 寄存器操作位
指示 Count 寄存器在调试模式下的操作。
1 = Count 寄存器在调试模式下运行
0 = Count 寄存器在调试模式下停止

bit 24 **IBUSEP**：取指总线错误异常等待处理位
在发生取指总线错误事件时或软件向该位写入 1 时置 1。在处理器发生取指总线错误异常时清零，或通过复位清零。如果 IBUSEP 在 IEXI 清零时置 1，则处理器会发生取指总线错误异常，并清零 IBUSEP。

bit 23-22 **未实现**：读为 0

bit 21 **DBUSEP**：数据访问总线错误异常等待处理位
涵盖数据访问时发生的非精确总线错误，操作类似于 IBUSEP 在取指时发生非精确总线错误的操作。

bit 20 **IEXI**：非精确错误异常禁止控制位
控制由于非精确错误指示而发生的异常。在处理器发生调试异常或调试模式下的异常时置 1。通过执行 DERET 指令清零；另外，可以通过调试模式软件修改。当 IEXI 置 1 时，由于总线错误（在取指或访问数据时发生）、高速缓存错误或机器检查而产生的非精确错误异常会被禁止和推迟，直到该位清零为止。

bit 19 **DDBSIMPR**：调试数据中断存储异常位
指示发生非精确调试数据中断存储异常。在 PIC32 内核上，所有数据中断都是精确的，所以该位将总是读为 0。

bit 18 **DDBLIMPR**：调试数据中断装载异常位
指示发生非精确调试数据中断装载异常。在 PIC32 内核上，所有数据中断都是精确的，所以该位将总是读为 0。

bit 17-15 **VER<2:0>**：EJTAG 版本位
包含 EJTAG 版本号。

bit 14-10 **DEXCCODE<4:0>**：调试模式下的最近异常位
指示调试模式下的最近异常的原因。对于在调试模式下可能发生的那些一般异常，该字段的编码与 Cause 寄存器中 EXCCODE<4:0> 位相同。发生调试异常之后的值是未定义的。

bit 9 **NOSST**：单步功能控制位
指示可通过 SST 位控制的单步功能在此实现中是否可用。
1 = 单步功能不可用
0 = 单步功能可用

bit 8 **SST**：调试单步控制位
控制是否使能调试单步异常。
1 = 使能调试单步异常
0 = 不使能调试单步异常

bit 7 **未实现**：读为 0

bit 6 **DIBIMPR**：非精确调试指令中断异常位
指示发生非精确调试指令中断异常 （由于复杂断点）。发生调试模式下的异常时清零。

© 2013-2016 Microchip Technology Inc.

寄存器 50-40： Debug：调试异常寄存器； CP0 寄存器 23，选择 0 （续）

bit 5 **DINT**：调试中断异常位
指示发生了调试中断异常。发生调试模式下的异常时清零。
1 = 发生了调试中断异常
0 = 未发生调试中断异常

bit 4 **DIB**：调试指令中断异常位
指示发生了调试指令中断异常。发生调试模式下的异常时清零。
1 = 发生了调试指令异常
0 = 未发生调试指令异常

bit 3 **DDBS**：存储时发生调试数据中断异常位
指示在存储时发生了调试数据中断异常。发生调试模式下的异常时清零。
1 = 在存储时发生了调试数据异常
0 = 在存储时未发生调试数据异常

bit 2 **DDBL**：装载时发生调试数据中断异常位
指示在装载时发生了调试数据中断异常。发生调试模式下的异常时清零。
1 = 在装载时发生了调试数据异常
0 = 在装载时未发生调试数据异常

bit 1 **DBP**：调试软件断点异常位
指示发生了调试软件断点异常。发生调试模式下的异常时清零。
1 = 发生了调试软件断点异常
0 = 未发生调试软件断点异常

bit 0 **DSS**：调试单步异常位
指示发生了调试单步异常。发生调试模式下的异常时清零。
1 = 发生了调试单步异常
0 = 未发生调试单步异常

50.13.41 TraceControl 寄存器 （CP0 寄存器 23，选择 1）

TraceControl 寄存器使能软件跟踪控制，且仅在具有 EJTAG 跟踪功能的器件上实现。

寄存器 50-41： TraceControl：跟踪控制寄存器； CP0 寄存器 23，选择 1

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-x	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
	TS	UT	—	—	TB	IO	D(1)	E(1)
23:16	R/W-x	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	K(1)	—	U(1)	ASID_M<7:3>				
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	ASID_M<2:0>			ASID<7:3>				
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-0
	ASID<2:0>			G	MODE<2:0>			ON

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31 **TS：**跟踪选择位
该位用于在硬件和软件跟踪控制位间选择。
1 = 选择跟踪控制位
0 = 选择外部硬件跟踪电路信号

bit 30 **UT：**用户类型选择位
该位用于指示用户触发跟踪记录的类型。
1 = 用户类型 2
0 = 用户类型 1

bit 29-28 **未实现：**读为 0

bit 27 **TB：**跟踪转移位
1 = 跟踪所有发生的转移分支的 PC 值
0 = 跟踪将跳转到不可预测静态地址的转移分支的 PC 值

bit 26 **IO：**禁止溢出位
该信号用来通知内核跟踪逻辑，需要缓慢但完整的跟踪。
1 = 禁止 FIFO 溢出，或丢弃跟踪数据
0 = 允许 FIFO 溢出，或丢弃跟踪数据

bit 25 **D：**调试模式跟踪使能位 (1)
1 = 使能调试模式下的跟踪
0 = 禁止调试模式下的跟踪

bit 24 **E：**异常模式跟踪使能 (1)
1 = 使能异常模式下的跟踪
0 = 禁止异常模式下的跟踪

bit 23 **K：**内核模式跟踪使能位 (1)
1 = 使能内核模式下的跟踪
0 = 禁止内核模式下的跟踪

bit 22 **未实现：**读为 0

注 1： ON 位必须设置为 1，才能使能跟踪。

© 2013-2016 Microchip Technology Inc.

寄存器 50-41： TraceControl：跟踪控制寄存器； CP0 寄存器 23，选择 1（续）

bit 21 **U**：用户模式跟踪使能位 **(1)**
1 = 使能用户模式下的跟踪
0 = 禁止用户模式下的跟踪

bit 20-13 **ASID_M<7:0>**：地址空间 ID 掩码值位
该域中的任何位为 1 将禁止相应的 ASID 位参与匹配。
该域值为 0 时将与 ASID 的所有位进行比较。

bit 12-5 **ASID<7:0>**：地址空间 ID 值位
当 G 设置为 0 时，这些位必须与地址空间 ID 匹配。
当 G 设置为 1 时，将忽略这些位。

bit 4 **G**：全局位
1 = 将对所有进程使能跟踪，前提是其他功能也已使能
0 = 不使能跟踪

bit 3-1 **MODE<2:0>**：跟踪模式控制位
111 = 跟踪 PC，以及装载和存储地址及数据
110 = 跟踪 PC，以及存储地址和数据
101 = 跟踪 PC，以及装载地址和数据
100 = 跟踪 PC 和装载数据
110 = 跟踪 PC，以及装载和存储地址
010 = 跟踪 PC 和存储地址
001 = 跟踪 PC 和装载地址
000 = 跟踪 PC

bit 0 **ON**：主器件跟踪使能位
1 = 当另一个跟踪使能位设置为 1 时，使能跟踪
0 = 禁止跟踪

注 1： ON 位必须设置为 1，才能使能跟踪。

50.13.42 TraceControl2 寄存器 （CP0 寄存器 23，选择 2）

TraceControl2 寄存器提供额外的控制和状态信息。请注意，TraceControl2 寄存器中的一些位域是只读的，它们的复位状态为 “未定义” 。这是因为这些位的值都是从跟踪控制模块 （Trace Control Block，TCB）装载的。因此，TraceControl2 寄存器中这些位域的值在 TCB 使之生效之前一直处于无效状态。

寄存器 50-42： TraceControl2：跟踪控制寄存器 2 ； CP0 寄存器 23，选择 2

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	U-0	R-1	R-0	R-0	R-1	R-x	R-x	R-x
	—	VALIDMODES<1:0>		TBI	TBU	SYP<2:0>		

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-7 **未实现：**读为 0

bit 6-5 **VALIDMODES<1:0>：**有效跟踪模式选择位
11 = 保留
10 = 跟踪 PC、装载和存储地址，及装载和存储数据
01 = 仅跟踪 PC 以及装载和存储地址
00 = 仅跟踪 PC

bit 4 **TBI：**跟踪缓冲区实现位
1 = 由 TCB 实现片内和片外跟踪缓冲区
0 = 仅实现一个跟踪缓冲区

bit 3 **TBU：**跟踪缓冲区使用位
1 = 跟踪数据被发送至片外跟踪缓冲区
0 = 跟踪数据被发送至片内跟踪缓冲区

bit 2-0 **SYP<2:0>：**同步周期位
当跟踪数据被写入片内跟踪缓冲区 （即 TBU 位 = 0）时，使用 “片内” 列中的值。相反，当跟踪数据被写入片外跟踪缓冲区 （即 TBU 位 = 1）时，使用 “片外” 列中的值。

位设置	片内	片外
111 =	2^2	2^7
110 =	2^3	2^8
101 =	2^4	2^9
100 =	2^5	2^{10}
011 =	2^6	2^{11}
010 =	2^7	2^{12}
001 =	2^8	2^{13}
000 =	2^9	2^{14}

© 2013-2016 Microchip Technology Inc.

50.13.43 UserTraceData1 寄存器 （CP0 寄存器 23，选择 3）

软件写入 UserTraceData1 寄存器中的任何位都将触发写跟踪记录，指明类型 1 用户格式。类型基于 TraceControl 寄存器中的 UT 位。不能在连续的周期写该寄存器。如果在连续的周期写该寄存器，则跟踪输出数据将不可预测。

该寄存器仅在具有 EJTAG 跟踪功能的器件中实现。

寄存器 50-43： UserTraceData1：用户跟踪数据寄存器 1 ； CP0 寄存器 23，选择 3

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DATA<31:24>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DATA<23:10>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DATA<15:6>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DATA<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **DATA<31:0>：**软件可读 / 可写数据位

写入时，该寄存器触发来自 PDtrace 接口的用户格式跟踪记录，该接口将数据位发送至跟踪存储器。

50.13.44 TraceBPC 寄存器 （CP0 寄存器 23，选择 4）

该寄存器用于采用 EJTAG 硬件断点控制跟踪的启动和停止。然后可将硬件断点设置为触发源，也可选择作为调试异常断点。

该寄存器仅在具有硬件断点和 EJTAG 跟踪功能的器件中实现。

寄存器 50-44： TraceBPC：跟踪断点控制寄存器； CP0 寄存器 23，选择 4

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	DE	—	—	—	—	—	—	—
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DBPO7(1)	DBPO6(1)	DBPO5(1)	DBPO4(1)	DBPO3(1)	DBPO2(1)	DBPO1	DBPO0
15:8	R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	IE	—	—	—	—	—	—	—
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	IBPO7(1)	IBPO6(1)	IBPO5	IBPO4	IBPO3	IBPO2	IBPO1	IBPO0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31 **DE：**EJTAG 数据断点触发选择位
1 = 使能来自数据断点的触发信号
0 = 禁止来自数据断点的触发信号

bit 30-24 **未实现：**读为 0

bit 23 **DBPO7：**数据断点 7 位 (1)
1 = 使能相应的数据指令断点触发以启动跟踪
0 = 禁止采用触发信号进行跟踪

bit 22 **DBPO6：**数据断点 6 位 (1)
1 = 使能相应的数据指令断点触发以启动跟踪
0 = 禁止采用触发信号进行跟踪

bit 21 **DBPO5：**数据断点 5 位 (1)
1 = 使能相应的数据指令断点触发以启动跟踪
0 = 禁止采用触发信号进行跟踪

bit 20 **DBPO4：**数据断点 4 位 (1)
1 = 使能相应的数据指令断点触发以启动跟踪
0 = 禁止采用触发信号进行跟踪

bit 19 **DBPO3：**数据断点 3 位 (1)
1 = 使能相应的数据指令断点触发以启动跟踪
0 = 禁止采用触发信号进行跟踪

bit 18 **DBPO2：**数据断点 2 位 (1)
1 = 使能相应的数据指令断点触发以启动跟踪
0 = 禁止采用触发信号进行跟踪

bit 17 **DBPO1：**数据断点 1 位
1 = 使能相应的数据指令断点触发以启动跟踪
0 = 禁止采用触发信号进行跟踪

注 1： 该位仅存在于采用 MPU 内核的 PIC32 器件上。

© 2013-2016 Microchip Technology Inc.

寄存器 50-44： TraceBPC：跟踪断点控制寄存器； CP0 寄存器 23，选择 4 （续）

bit 16 **DBPO0**：数据断点 0 位
1 = 使能相应的数据指令断点触发以启动跟踪
0 = 禁止采用触发信号进行跟踪

bit 15 **IE**：EJTAG 指令断点选择位
1 = 使能来自指令断点的触发信号
0 = 禁止来自指令断点的触发信号

bit 14-8 **未实现**：读为 0

bit 5 **IBPO7**：指令断点 7 位 (1)
1 = 使能相应的指令断点触发以启动跟踪
0 = 禁止采用触发信号进行跟踪

bit 4 **IBPO6**：指令断点 6 位 (1)
1 = 使能相应的指令断点触发以启动跟踪
0 = 禁止采用触发信号进行跟踪

bit 5 **IBPO5**：指令断点 5 位
1 = 使能相应的指令断点触发以启动跟踪
0 = 禁止采用触发信号进行跟踪

bit 4 **IBPO4**：指令断点 4 位
1 = 使能相应的指令断点触发以启动跟踪
0 = 禁止采用触发信号进行跟踪

bit 3 **IBPO3**：指令断点 3 位
1 = 使能相应的指令断点触发以启动跟踪
0 = 禁止采用触发信号进行跟踪

bit 2 **IBPO2**：指令断点 2 位
1 = 使能相应的指令断点触发以启动跟踪
0 = 禁止采用触发信号进行跟踪

bit 1 **IBPO1**：指令断点 1 位
1 = 使能相应的指令断点触发以启动跟踪
0 = 禁止采用触发信号进行跟踪

bit 0 **IBPO0**：指令断点 0 位
1 = 使能相应的指令断点触发以启动跟踪
0 = 禁止采用触发信号进行跟踪

注 1： 该位仅存在于采用 MPU 内核的 PIC32 器件上。

50.13.45 Debug2 寄存器 （CP0 寄存器 23，选择 6）

该寄存器用来保存关于复杂断点异常的其他信息。该寄存器仅在存在复杂硬件断点时实现。

寄存器 50-45： Debug2：调试断点异常寄存器； CP0 寄存器 23，选择 6

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	r-1	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	U-0	U-0	U-0	U-0	R-x	R-x	R-x	R-x
	—	—	—	—	PRM	DQ	TUP	PACO

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31 **保留**：读为 1

bit 30-4 未实现：读为 0

bit 3 **PRM**：初始位
指示上次调试异常时是否观察到带有效初始条件的复杂断点。

bit 2 **DQ**：数据限定位
指示上次调试异常时是否观察到带有效数据限定符的复杂断点。

bit 1 **TUP**：元组断点位
指示上次调试异常时是否观察到元组断点。

bit 0 **PACO**：通过计数器位
指示上次调试异常时是否观察到带有效通过计数器的复杂断点。

© 2013-2016 Microchip Technology Inc.

50.13.46 DEPC 寄存器 （CP0 寄存器 24，选择 0）

调试异常程序计数器 （Debug Exception Program Counter， DEPC）寄存器是一个读 / 写寄存器，包含在处理完调试异常或调试模式下的异常之后继续进行处理的地址。

对于同步 （精确）调试异常和调试模式下的异常， DEPC 寄存器包含以下值之一：

- 直接导致调试异常的指令的虚拟地址，或者
- 紧接在转移或跳转指令之前的虚拟地址 （如果导致调试异常的指令处于转移延时时隙中，并且 Debug 寄存器中的调试转移延时 （Debug Branch Delay， DBD）位置 1）。

对于异步调试异常 （调试中断）， DEPC 寄存器包含一条指令的虚拟地址，在调试处理程序代码执行完毕之后将在该地址处继续执行。

由于 PIC32 系列实现了 MIPS16e 或 microMIPS ASE，所以读 DEPC 寄存器（通过 MFC0）会在目标 GPR 中返回以下值：

$$GPR[rt] = DebugExceptionPC_{31..1} \ || \ ISAMode_0$$

即，调试异常 PC 的高 31 位与 ISA<1:0> 位 （Config3<15:14>）的低位组合，并写入 GPR。

类似地，写 DEPC 寄存器 （通过 MTC0）时会获取 GPR 的值，并将该值分配到调试异常 PC 和 ISA<1:0> 位 （Config3<15:14>）中，如下：

$$DebugExceptionPC = GPR[rt]_{31..1} \ || \ 0$$
$$ISAMode = 2\#0 \ || \ GPR[rt]_0$$

即， GPR 的高 31 位写入调试异常 PC 的高 31 位，调试异常 PC 的低位会被清零。ISA<1:0> 位（Config3<15:14>）的高位会被清零，低位中则装入 GPR 的低位。

寄存器 50-46： DEPC：调试异常程序计数器寄存器； CP0 寄存器 24，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DEPC<31:24>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DEPC<23:16>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DEPC<15:8>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DEPC<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **DEPC<31:0>**：调试异常程序计数器位

DEPC 寄存器会更新为导致调试异常的指令的虚拟地址。如果指令处于转移延时时隙中，则紧接在转移或跳转指令之前的虚拟地址会被放入该寄存器中。

执行 DERET 指令会导致跳转到 DEPC 寄存器中的地址处。

50.13.47 UserTraceData2 （CP0 寄存器 24，选择 3）

软件写入 UserTraceData2 寄存器中的任何位都将触发写跟踪记录，指明类型 2 用户格式。类型基于 TraceControl 寄存器中的 UT 位。不能在连续的周期写该寄存器。如果在连续的周期写该寄存器，则跟踪输出数据将不可预测。

该寄存器仅在具有 EJTAG 跟踪功能的器件中实现。

寄存器 50-47： UserTraceData2：用户跟踪数据寄存器 2 ； CP0 寄存器 24，选择 3

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DATA<31:24>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DATA<23:16>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DATA<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	DATA<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **DATA<31:0>：**软件可读 / 可写数据位

写入时，该寄存器触发来自 PDtrace 接口的用户格式跟踪记录，该接口将 DATA 位发送至跟踪存储器。

© 2013-2016 Microchip Technology Inc.

50.13.48 PerfCtlx 寄存器 （CP0 寄存器 25，选择 0/3）

microAptiv™ 微处理器内核定义了两个性能计数器 PerfCnt0 和 PerfCnt1（见寄存器 50-49）以及两个相关的控制寄存器 PerfCtl0 和 PerfCtl1（这两个寄存器映射到 CP0 寄存器 25）。MTC0/MFC0 指令的选择位用于选择指令访问的特定寄存器，如表 50-29 所示。

表 50-29： 性能计数器寄存器选择

Select<2:0>	寄存器
0	控制寄存器 0
1	计数寄存器 0
2	控制寄存器 1
3	计数寄存器 1

每个计数器都是一个 32 位读 / 写寄存器，每当发生可计数的事件时该计数器的值加 1，该事件由相应的控制寄存器指定。每个计数器一次只能独立对一类事件进行计数。

每个计数器的 bit 31 和相应控制寄存器的中断允许位（IE）进行“与”运算，确定是否发送性能计数器中断信号。然后这两个位再进行“或”运算，产生一个性能计数器中断输出。该输出向内核发送一个中断信号。计数操作不受中断指示的影响。当计数器值归零时该输出清零，也可以在软件中通过向性能计数器计数寄存器的 bit 31 写入 0 来清零。

注： 性能计数器寄存器和一个时钟相连，该时钟在处理器处于休眠模式时停止工作。在这段时间内绝大多数事件是不会运行的，但有些事件，特别是周期计数事件会继续运行。当对系统进行分析测量时需要考虑该操作。

寄存器 50-48： PerfCtlx：性能计数器控制寄存器； CP0 寄存器 25，选择 0/3

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	r-x	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
	—	—	—	—	EVENT<6:3>			
7:0	R/W-x	R/W-x	R/W-x	R/W-0	R-0	U-0	R/W-x	R/W-x
	EVENT<2:0>			IE	U	—	K	EXL

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31 **保留：**对于 PerfCtl0 读为 1，而对于 PerfCtl1 读为 0

bit 30-12 **未实现：**读为 0

bit 11-5 **EVENT<6:0>：**
使能该计数器的计数事件。表 50-30 列出了可能的事件。

bit 4 **IE：**计数中断允许位
该位屏蔽相应计数寄存器的 bit 31 产生中断异常请求输出。

bit 3 **U：**在用户模式下计数位
当该位置 1 时，在用户模式下对指定事件计数。

寄存器 50-48： PerfCtlx：性能计数器控制寄存器； CP0 寄存器 25，选择 0/3 （续）

bit 2 **未实现：**读为 0

bit 1 **K：**在内核模式下计数位
当该位置 1，且 EXL 和 ERL 均为 0 时，在内核模式下对事件计数。

bit 0 **EXL：**EXL 置 1 时计数位
当该位置 1，且 EXL = 1 和 ERL = 0 时对事件计数。

表 50-30 描述了这两个性能计数器可计数的事件。“模式”列指示事件计数是否受模式位（U、K 和 EXL）的影响。对于指定为保留的事件，计数器的操作是不可预测的。

当 ERL = 1 时，性能计数器不会在调试模式下计数。

表 50-30： 性能可计数事件

事件编号	计数器 0	模式	计数器 1	模式
0	周期数	否	周期数	否
1	已完成指令数	是	已完成指令数	是
2	转移指令数	是	保留	N/A
3	JR r31 （返回）指令数	是	保留	N/A
4	JR （非 r31）指令数	是	保留	N/A
5	ITLB 访问数 (1)	是	ITLB 未命中数 (1)	是
6	DTLB 访问数 (1)	是	DTLB 未命中数 (1)	是
7	JTLB 指令访问数 (1)	是	JTLB 指令未命中数 (1)	是
8	JTLB 数据访问数 (1)	是	JTLB 数据未命中数 (1)	是
9	指令高速缓存访问数 (1)	是	指令高速缓存未命中数 (1)	是
10	数据高速缓存访问数 (1)	是	数据高速缓存回写数 (1)	是
11	数据高速缓存未命中数 (1)	是	数据高速缓存未命中数 (1)	是
12	保留	N/A	保留	N/A
13	保留	N/A	保留	N/A
14	已完成的整数指令数	是	保留	N/A
15	已完成的装载数	是	已完成的存储数	是
16	已完成的 J/JAL	是	已完成的 microMIPS 指令数	是
17	已完成的空操作数	是	已完成的整数乘 / 除法	是
18	暂停周期数	否	保留	N/A
19	已完成的 SC 指令数	是	已失败的 SC 指令数	是
20	已完成的预取指令数	是	已完成的高速缓存命中预取指令数 (1)	是
21	保留	N/A	保留	N/A
22	保留	N/A	保留	N/A
23	已发生的异常数	是	保留	N/A
24	保留	N/A	保留	N/A
25	IFU 暂停周期数 (1)	否	ALU 暂停周期数	否
26	保留	N/A	保留	N/A
27	保留	N/A	保留	N/A
28	保留	N/A	特定实现的 CP2 事件	是
29	保留	N/A	保留	N/A
30	保留	N/A	保留	N/A

注 1： 该事件仅存在于采用 MPU 内核的器件上。请参见具体器件数据手册的 **“CPU”**章节确定该事件是否存在。

© 2013-2016 Microchip Technology Inc.

表 50-30： 性能可计数事件 （续）

事件编号	计数器 0	模式	计数器 1	模式
31	保留	N/A	保留	N/A
32	保留	N/A	保留	N/A
33	非高速缓存装载数	N/A	非高速缓存存储数	N/A
34	保留	N/A	保留	N/A
35	已完成的 CP2 算术指令数	是	已完成的 CP2 传送指令数	是
36	保留	N/A	保留	N/A
37	指令高速缓存未命中暂停周期数 (1)	是	数据高速缓存未命中暂停周期数 (1)	是
38	保留	N/A	保留	N/A
39	数据高速缓存未命中周期数 (1)	否	保留	N/A
40	非高速缓存暂停周期数	是	保留	N/A
41	MDU 暂停周期数	是	保留	N/A
42	CP2 暂停周期数	是	保留	N/A
43	保留	N/A	保留	N/A
44	CACHE 指令暂停周期数 (1)	否	保留	N/A
45	装载以使用暂停周期数	是	保留	N/A
46	其他互锁暂停周期数	是	保留	N/A
47	保留	N/A	保留	N/A
48	保留	N/A	保留	N/A
49	EJTAG 指令触发点数	是	EJTAG 数据触发点数	是
50 -51	保留	N/A	保留	N/A
52	LFQ< 四分之一满 (1)	否	LFQ 四分之一到二分之一满 (1)	否
53	LFQ> 二分之一满 (1)	否	LFQ 完整流水线暂停周期数 (1)	否
54	WBB< 四分之一满 (1)	否	WBB 四分之一到二分之一满 (1)	否
55	WBB> 二分之一满 (1)	否	WBB 完整流水线暂停周期数 (1)	否
56-63	保留	N/A	保留	N/A

注 1： 该事件仅存在于采用 MPU 内核的器件上。请参见具体器件数据手册的 **“CPU”** 章节确定该事件是否存在。

表 50-31： 事件说明

事件名称	计数器	事件编号	说明
周期数	0/1	0	总周期数。性能计数器由顶级门控时钟提供时钟。如果内核是使用存在门控的时钟构建的，那么在时钟停止时 （例如，由于 WAIT 指令）两个计数器均不递增。
完成指令 下面的事件指示完成的各类型指令			
指令数	0/1	1	完成的指令总数。
转移指令数	0	2	计数所有已完成的转移指令。
JR R31 （返回）指令数	0	3	计数所有已完成的 JR R31 指令。
JR （非 R31）	0	4	计数所有 JR rxx （非 r31）和 JALR 指令 （间接跳转指令）
整数指令数	0	14	非浮点指令数
装载数	0	15	包含整数和协处理器装载数。
存储数	1	15	包含整数和协处理器存储数。
J/JAL	0	16	直接跳转 （和链接）指令。
microMIPS	1	16	所有 microMIPS 指令数。
空操作数	0	17	这包括正常写入 GPR 的所有指令，但目标寄存器应设置为 r0。
整数乘法 / 除法	1	17	计数所有整数乘法 / 除法指令 （MULxx、 DIVx、 MADDx 和 MSUBx）。
SC	0	19	计数条件存储，无论它们是否成功。
PREF	0	20	请注意，此事件仅计数实际尝试的 PREF。不包括指向非高速缓存的地址或存在转换错误的 PREF。
非高速缓存装载数 (1)	0	33	包含非高速缓存和非高速缓存加速 CCA。
非高速缓存存储数 (1)	1	33	
指令执行事件			
ITLB 访问数 (1)	0	5	计数 ITLB 访问，该访问产生于流水线操作 IF 阶段的取指操作，并且没有使用固定地址映射或者未发生在未映射的空间中。如果在流水线中取某个地址两次 （如高速缓存未命中的情况），则该指令将使 ITLB 访问被计数两次。另外，由于每次取指返回 2 条指令，因此会使用双字标记每次访问。
ITLB 未命中数 (1)	1	5	计数 ITLB 中所有未命中数，那些紧跟在一个未命中后出现的未命中事件除外。我们无法处理那些紧连的未命中事件，因此忽略了这些事件。另外还忽略某种形式的地址错误。
DTLB 访问数 (1)	0	6	计数 DTLB 访问，包括那些对未映射地址空间的访问。
DTLB 未命中数 (1)	1	6	计数 DTLB 未命中数。紧连的两个未命中数只会引起重填一个 DTLB 条目，因此记作一次未命中。
JTLB 指令访问数 (1)	0	7	指令 JTLB 访问的计数值和 ITLB 未命中的计数值完全相同。
JTLB 指令未命中数 (1)	1	7	计数导致无匹配或者匹配无效转换的指令 JTLB 访问。

注 1： 该事件仅存在于采用 MPU 内核的器件上。请参见具体器件数据手册的 **“CPU”** 章节确定该事件是否存在。

© 2013-2016 Microchip Technology Inc.

表 50-31： 事件说明 （续）

事件名称	计数器	事件编号	说明
JTLB 数据访问数 (1)	0	8	数据 JTLB 访问数。
JTLB 数据未命中数 (1)	1	8	计数导致无匹配或匹配无效转换的数据 JTLB 访问。
指令高速缓存访问数 (1)	0	9	计数每次对指令高速缓存的访问。所有重复浪费的取指操作等都考虑在内。例如，紧跟在转移指令之后，即使进行了预测，失败的访问也考虑在内。
指令高速缓存未命中数 (1)	1	9	计数所有引起总线请求的指令高速缓存未命中数。
数据高速缓存访问数 (1)	0	10	计数高速缓存装载和存储。
数据高速缓存回写数 (1)	1	10	计数由于替换或高速缓存操作而写回存储器的高速缓存行。
数据高速缓存未命中数 (1)	0/1	11	计数高速缓存中未命中的装载和存储。
已失败的 SC 指令数	1	19	不更新存储器的 SC 指令。 **注：** 当该事件和 SC 指令计数事件可配置为在指定工作模式下计数时，事件的时序会有很大的不同，并且观察到的工作模式可在两者之间切换，引起测得的比率有误差。
已完成的高速缓存命中的 PREF(1)	1	20	计数命中高速缓存的 PREF 指令。
已发生的异常数	0	23	出现的任何类型的异常。
EJTAG 指令触发数	0	49	满足计数 EJTAG 指令触发点条件的次数。
EJTAG 数据触发数	1	49	满足计数 EJTAG 数据触发点条件的次数。
流水线			
高速缓存修复 (1)	0	24	计数下面情况下的周期数：DCC 正在修复并且不能从 ALU 接收新的指令。当一条指令需要重新访问高速缓存或 DTLB 时，DCC 的修复会重复进行。
通用暂停			
IFU 暂停周期数 (1)	0	25	计数下面情况下的周期数：取指单元没有向 ALU 提供有效的指令。
ALU 暂停周期数	1	25	计数下面情况下的周期数：ALU 流水线无法前移。
暂停周期数	0	18	计数下面情况下的总周期数：SRAM 没有向 ALU 发送任何指令 （RF 级无法前移）。这种情况包含了上面两种事件。然而，该计数值并不是前两者之和，因为两种暂停同时出现时相应周期只会计数一次。
特定暂停 这些事件将计数由于特定暂停引起的损失周期数。这将包括流水线中由于重复访问而引起的气泡。如果同时有多个暂停源处于活动状态，每个活动事件的计数器均会递增 1。			
指令高速缓存未命中暂停周期数 (1)	0	37	由于指令高速缓存未命中而导致 ICC 没有任何可运行的指令，致使 ICC 暂停时的周期数。由于 ITLB 未命中以及一个重定位之后的四个周期引起的暂停都会被忽略。

注 1： 该事件仅存在于采用 MPU 内核的器件上。请参见具体器件数据手册的 **“CPU”** 章节确定该事件是否存在。

表 50-31： 事件说明 （续）

事件名称	计数器	事件编号	说明
数据高速缓存未命中暂停周期数 (1)	1	37	计数以下情况下的所有周期数：由于数据高速缓存未命中而导致整数流水线等待装载返回数据。
数据高速缓存未命中周期周期数 (1)	0	39	数据高速缓存未命中尚未解决，但并不必阻塞流水线。该周期数与数据高速缓存未命中暂停周期数的差可以表示非阻塞高速缓存未命中时间的周期数增量。
非高速缓存暂停周期数	0	40	当处理器由于非高速缓存的取指、装载或存储操作而暂停的周期数。
MDU 暂停周期数	0	41	计数整数流水线等待 MDU 返回数据期间的所有周期数。
CACHE 指令暂停周期数 (1)	0	44	计数由于 CACHE 指令导致流水线暂停期间的所有周期数。包括 CACHE 指令本身在 ALU 内暂停的周期数以及 CACHE 指令引起后续指令暂停的周期数。
装载以使用暂停周期数	0	45	计数整数流水线等待装载操作返回数据期间的所有周期数。
其他互锁暂停周期数	0	46	计数整数流水线等待 MFC0 和 RDHWR 指令返回数据期间的所有周期数。
LFB 完整流水线暂停周期数	1	53	由于 DCC 的装载填充缓冲区 （Load Fill Buffer， LFB）已满导致的流水线暂停的周期数。
直写缓冲区满暂停周期数 (1)	1	55	由于 BIU 的直写缓冲区已满导致的流水线暂停的周期数。

注 1： 该事件仅存在于采用 MPU 内核的器件上。请参见具体器件数据手册的 **“CPU”** 章节确定该事件是否存在。

© 2013-2016 Microchip Technology Inc.

50.13.49 PerfCntx 寄存器 （CP0 寄存器 25，选择 1/3）

微处理器内核定义了两个性能计数器 PerfCnt0 和 PerfCnt1 及两个相关的控制寄存器 PerfCtl0 和 PerfCtl1 （见寄存器 50-48，这两个寄存器映射到 CP0 寄存器 25）。 MTC0/MFC0 指令的选择位用于选择指令访问的特定寄存器，如表 50-29 所示。

性能计数器复位为低功耗状态，在该状态期间，直到软件通过 MTC0 指令对性能计数器控制寄存器使能事件计数后，计数器才能开始对事件计数。

寄存器 50-49： PerfCntx：性能计数器计数寄存器； CP0 寄存器 25，选择 1/3 （x = 0 或 1）

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COUNTER<31:24>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COUNTER<23:16>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COUNTER<15:8>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	COUNTER <7:0>							

图注：

R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **COUNTER<31:0>：**事件计数器位
对该计数器使能事件计数。

50.13.50 ErrCtl 寄存器 （CP0 寄存器 26，选择 0）（仅限 MPU）

ErrCtl 寄存器提供对路选和 RAM 的软件测试。

通过将 WST 位置 1 来使能路选 RAM 测试模式。它改变 CACHE 索引装载标记和索引存储标记操作的功能，从而使这些操作修改路选 RAM 而保持标记 RAM 不变。该位置 1 时，TagLo 寄存器中 PA 域的低六位用作索引装载标记和索引存储标记 CACHE 操作的源和目标。

WST 位还可使能数据 RAM 测试模式。该位置 1 时，使能索引存储数据 CACHE 指令。该 CACHE 操作将 DataLo 寄存器的内容写入索引和字节地址所指示的数据阵列的字中。

寄存器 50-50： ErrCtl：奇偶校验保护控制寄存器； CP0 寄存器 26，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
	—	—	WST	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-28 **未实现：**读为 0

bit 29 **WST：**路选 / 标记阵列位

指示在执行索引装载 / 存储标记 CACHE 指令时是读 / 写标记阵列还是路选阵列。该位还可使能索引存储数据 CACHE 指令，会将 DataLo 的内容写入数据阵列。

1 = 执行索引装载 / 存储标记 CACHE 指令时读 / 写路选阵列
0 = 执行索引装载 / 存储标记 CACHE 指令时读 / 写标记阵列

bit 27-0 **未实现：**读为 0

© 2013-2016 Microchip Technology Inc.

50.13.51 CacheErr 寄存器 （CP0 寄存器 27，选择 0）（仅限 M-Class）

CacheErr 寄存器用于提供与高速缓存错误检测逻辑的接口。当发出高速缓存 /SPRAM 奇偶校验错误异常时，会相应地设置该寄存器的位域。

寄存器 50-51： CacheErr：高速缓存错误寄存器； CP0 寄存器 27，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R-0	R-0	R-0	R-0	R-0	U-0	R-0	R-0
	ER	EC	ED	ET	ES	—	EB	EF
23:16	R-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0
	SP	EW	Way<1:0>		Index<19:16>			
15:8	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	Index<15:8>							
7:0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	Index<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31 **ER：**错误引用位
指示遇到错误的引用的类型。
1 = 遇到数据引用
0 = 遇到指令引用

bit 30 **EC：**高速缓存级别错误检测位
指示检测到错误的高速缓存级别。
1 = 检测到非主高速缓存级别
0 = 检测到主高速缓存级别

bit 29 **ED：**错误数据位
指示数据 RAM 错误。
1 = 检测到数据 RAM 错误
0 = 未检测到数据 RAM 错误

bit 28 **ET：**错误标记位
指示标记 RAM 错误。
1 = 检测到标记 RAM 错误
0 = 未检测到标记 RAM 错误

bit 27 **ES：**错误源位
指示错误是内部处理器或还是外部侦听请求导致的。
1 = 在外部请求时发生错误
0 = 在内部请求时发生错误

bit 26 **未实现：**读为 0

bit 25 **EB：**双错误位
指示除了指令高速缓存 /SPRAM 奇偶校验错误之外，还发生了数据高速缓存 /SPRAM 奇偶校验错误。在外加发生数据高速缓存 /SPRAM 奇偶校验错误时，该寄存器中其余的位将根据指令高速缓存 /SPRAM 奇偶校验错误来设置。
1 = 外加发生了数据高速缓存 /SPRAM 奇偶校验错误
0 = 未外加发生数据高速缓存 /SPRAM 奇偶校验错误

寄存器 50-51： CacheErr：高速缓存错误寄存器； CP0 寄存器 27，选择 0 （续）

bit 24 **EF：**致命错误位
指示发生了致命的高速缓存错误。
在一些情况下，软件无法从 CacheErr 寄存器获取关于高速缓存错误的所有信息。这些情况是致命的，因为软件无法确定哪些存储单元受到了错误的影响。为使软件能够检测到这些情况，在 CacheErr 寄存器中添加了 EF 位。
以下情况会用 EF 位指示为致命高速缓存错误：
- 脏受害者 （dirty victim）中出现脏奇偶校验错误 （脏位清零）
- 脏受害者中出现标记奇偶校验错误
- 脏受害者中出现数据奇偶校验错误
- 所请求索引处出现 WB 存储未命中和 EW 错误

此外，同时发生指令高速缓存错误和数据高速缓存错误（由 EB 位指示）会导致数据高速缓存错误的相关信息不可用。但是，这种情况不通过 EF 位指示。

bit 23 **SP：**中间结果暂存器位
1 = 检测到中间结果暂存 RAM 错误
0 = 未检测到中间结果暂存 RAM 错误

bit 22 **EW：**错误高速缓存路位
1 = 检测到路选 RAM 错误
0 = 未检测到路选 RAM 错误

bit 21-20 **Way<1:0>：**高速缓存路位
指明检测到错误的高速缓存路。如果检测到标记 RAM 错误 （ET = 1）或检测到中间结果暂存 RAM 错误（SP = 1），则它是无效的。

bit 19-0 **Index<19:0>：**索引位
指明检测到错误的双字的高速缓存或中间结果暂存 RAM 索引。发生故障的高速缓存路由硬件写入 Way 位域。软件必须将在该寄存器中读取的高速缓存路和索引与 Config1 寄存器中的高速缓存配置信息相结合，以获得可以在索引高速缓存指令中使用的索引，用于访问故障高速缓存数据或标记。请注意，索引会被对齐为字节索引，所以软件不需要先对它进行移位即可在索引高速缓存指令中使用它。索引位 4-3 在发生标记 RAM 错误时是未定义的，高于实际用于高速缓存索引的 MSB 的索引位也是未定义的。位 19-16 仅用于指示中间结果暂存 RAM 中的错误。

© 2013-2016 Microchip Technology Inc.

50.13.52 WST （ErrCtl<29>）= 0 时的 TagLo 寄存器 （CP0 寄存器 28，选择 0）（仅限 MPU）

TagLo 寄存器用作高速缓存标记阵列的接口。 CACHE 指令的索引存储标记和索引装载标记操作使用 TagLo 寄存器作为标记信息源。

当 ErrCtl 寄存器的 WST 位有效时，该寄存器成为路选 RAM 的接口。在该模式下，重新定义该域以提供对 WS 阵列 （而不是标记阵列）内容的适当访问。

注： 寄存器 50-53 描述了 WST = 1 时 TagLo 寄存器中的域。

寄存器 50-52： TagLo：高速缓存标记阵列接口寄存器； CP0 寄存器 28，选择 0（WST = 0 时）

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	PA<21:14>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	PA<13:6>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0
	PA<5:0>						—	—
7:0	R/W-x	R/W-x	R/W-x	U-0	U-0	U-0	U-0	U-0
	V	D	L	—	—	—	—	—

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-10 **PA<21:0>：**高速缓存行物理地址位
这些位包含高速缓存行的物理地址。 PA<21> 对应物理地址的 bit 31，而 PA<0> 对应物理地址的 bit 10。

bit 9-8 **未实现：**读为 0

bit 7 **V：**有效高速缓存行状态位
该位指示高速缓存行是否有效。

bit 6 **D：**高速缓存行 Dirty 状态位
该位指示高速缓存行是否 dirty。该位仅在 bit 7 （V）也置 1 时置 1。

bit 5 **L：**高速缓存标记锁定状态位
该位指定高速缓存标记的锁定位。当该位和 bit 7 （V）均置 1 时，相应的高速缓存行将不会被高速缓存替换算法替换。

bit 4-0 **未实现：**读为 0

© 2013-2016 Microchip Technology Inc.

50.13.53 WST （ErrCtl<29>） = 1 时的 TagLo 寄存器 （CP0 寄存器 28，选择 0）（仅限 MPU）

TagLo 寄存器用作高速缓存标记阵列的接口。 CACHE 指令的索引存储标记和索引装载标记操作使用 TagLo 寄存器作为标记信息源。

当 ErrCtl 寄存器的 WST 位有效时，该寄存器成为路选 RAM 的接口。在该模式下，重新定义该域以提供对 WS 阵列 （而不是标记阵列）内容的适当访问。

注： 寄存器 50-52 描述了 WST = 0 时 TagLo 寄存器中的域。

寄存器 50-53： TagLo：高速缓存标记阵列接口寄存器； CP0 寄存器 28，选择 0（WST = 1 时）

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
	—	—	—	—	WSD<3:0>			
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0
	WSLRU<5:0>						—	—
7:0	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-20 **未实现：** 读为 0

bit 19-16 **WSD<3:0>：** 路选 dirty 状态位
这些位包含执行 CACHE 索引装载 WS 操作后，从 WS 阵列读取的值。它用于在 CACHE 索引存储 WS 操作期间存储 WS 阵列。

bit 15-10 **WSLRU<5:0>：** 路选最近使用位
这些位包含从 WS 阵列读取的值或要存储到 WS 阵列的值。

bit 9-0 **未实现：** 读为 0。

© 2013-2016 Microchip Technology Inc.

50.13.54 DataLo 寄存器 （CP0 寄存器 28，选择 1）（仅限 MPU）

DataLo 寄存器充当高速缓存数据阵列的接口且仅用于诊断操作。 CACHE 指令的索引装载标记操作将相应的数据值读取到 DataLo 寄存器。如果 ErrCtl 寄存器中的 WST 位置 1，可通过执行索引存储数据 CACHE 指令将 DataLo 的内容写入高速缓存数据阵列。

寄存器 50-54： DataLo：高速缓存数据阵列接口寄存器； CP0 寄存器 28，选择 1

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DATA<31:24>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DATA<23:16>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DATA<15:8>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DATA<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **DATA<31:0>**：从高速缓存数据阵列读取低位数据位

50.13.55 ErrorEPC （CP0 寄存器 30，选择 0）

ErrorEPC 寄存器是一个类似于 EPC 寄存器的读 / 写寄存器，只是 ErrorEPC 是在发生错误异常时使用。 ErrorEPC 寄存器的所有位均为有效位，必须为可写。它也在发生复位、软复位和不可屏蔽中断 （Nonmaskable Interrupt， NMI）异常时用于存储程序计数器。

ErrorEPC 寄存器包含一个虚拟地址，在处理完错误之后在该地址处继续处理指令。该地址可能为：

- 导致异常的指令的虚拟地址
- 紧接在转移或跳转指令之前的虚拟地址 （当导致错误的指令处于转移延时时隙中时）

不同于 EPC 寄存器， ErrorEPC 寄存器不存在相应的转移延时时隙指示。

由于 PIC32 系列实现了 MIPS16e® 或 microMIPS，所以读 ErrorEPC 寄存器（通过 MFC0）会在目标 GPR 中返回以下值：

$GPR[rt] = ErrorExceptionPC_{31..1} \;||\; ISAMode_0$

即，错误异常 PC 的高 31 位与 ISA<1:0> 位 （Config3<15:14>）的低位相结合，并写入 GPR。

类似地，写 ErrorEPC 寄存器 （通过 MTC0）时会获取 GPR 的值，并将该值分配到错误异常 PC 和 ISA<1:0> 位 （Config3<15:14>）中，如下：

$ErrprExceptionPC = GPR[rt]_{31..1} \;||\; 0$
$ISAMode = 2\#0 \;||\; GPR[rt]_0$

即， GPR 的高 31 位写入错误异常 PC 的高 31 位，错误异常 PC 的低位会被清零。 ISA<1:0> 位（Config3<15:14>）的高位会被清零，低位中则装入 GPR 的低位。

寄存器 50-55： ErrorEPC：错误异常程序计数器寄存器； CP0 寄存器 30，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	ErrorEPC<31:24>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	ErrorEPC<23:16>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	ErrorEPC<15:8>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	ErrorEPC<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **ErrorEPC<31:0>：**错误异常程序计数器位

© 2013-2016 Microchip Technology Inc.

50.13.56 DeSAVE 寄存器 （CP0 寄存器 31，选择 0）

DeSAVE 寄存器是一个读 / 写寄存器，用作简单的存储单元。调试异常处理程序使用该寄存器来替代一个 GPR，然后将剩余的现场信息保存到预先确定的存储区（例如保存在 EJTAG 探针中）。无法断定是否存在用于保存现场信息的有效堆栈时，可以通过该寄存器，对异常处理程序和其他类型的代码安全地进行调试。

寄存器 50-56： DeSAVE：调试异常保存寄存器； CP0 寄存器 31，选择 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DESAVE<31:24>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DESAVE<23:16>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DESAVE<15:8>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	DESAVE<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **DESAVE<31:0>：**调试异常保存位

由调试异常代码使用的中间结果暂存寄存器。

50.13.57 KScratch*n* 寄存器 （CP0 寄存器 31，选择 2-3）（仅限 M-Class）

KScratch*n* 寄存器是可选的读 / 写寄存器，可由内核模式软件用于中间结果暂存。对于 32 位处理器，这些寄存器的宽度为 32 位，对于 64 位处理器则为 64 位。

是否存在这些寄存器由 Config4 寄存器中的 KScrExist 位域指示。KScrExist 位域指示对于哪些选择会包含一个内核中间结果暂存寄存器。

调试模式软件不应使用这些寄存器；而是使用 DeSave 寄存器。如果实现了 EJTAG，则不应使用选择 0 来获得一个 KScratch 寄存器。选择 1 保留供将来调试使用，不应使用它来获取一个 KScratch 寄存器。

寄存器 50-57： KScratch*n*：内核模式中间结果暂存寄存器； CP0 寄存器 31，选择 2-3

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	Data<31:24>							
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	Data<23:16>							
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	Data<15:8>							
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	Data<7:0>							

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-0 **Data<31:0>**：由内核软件保存的中间结果暂存数据位

© 2013-2016 Microchip Technology Inc.

50.14 协处理器 1 （CP1）寄存器

共有 5 个用于控制 FPU 的协处理器 1 （CP1）寄存器，也称为 FPU 控制寄存器 （FPU Control Register，FCR）。这些寄存器都为 32 位宽：FIR、FCCR、FEXR、FENR 和 FCSR。其中 3 个寄存器：FCCR、 FEXR 和 FENR 将选择 FCSR （浮点控制 / 状态寄存器）的子集。

> **注：** 协处理器 1 （CP1）寄存器仅在采用 M-Class 内核的器件上可用。要确定可用性，请参见具体器件数据手册中的 **“CPU”** 章节。

表 50-32 中汇总了 CP1 控制寄存器。

表 50-32： 协处理器 1 寄存器汇总

寄存器编号	寄存器名称	功能
0	FIR	浮点实现寄存器。包含标识 FPU 的信息。
25	FCCR	浮点条件代码寄存器。
26	FEXR	浮点异常寄存器。
28	FENR	浮点使能寄存器。
31	FCSR	浮点控制和状态寄存器。

50.14.1 浮点寄存器 （FIR， CP1 控制寄存器 0）（仅限 M-Class）

浮点实现寄存器 （Floating Point Implementation Register， FIR）是一个 32 位的只读寄存器，包含标识 FPU 功能的信息、浮点处理器标识和 FPU 版本级别。

寄存器 50-58： FIR：浮点实现寄存器； CP1 控制寄存器 0

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	R-1	U-0	U-0	U-0	R-1
	—	—	—	UFRP	—	—	—	FC
23:16	R-1	R-1	R-1	R-1	R-1	R-1	R-1	R-1
	Has2008	F64	L	W	3D	PS	D	S
15:8	R-1	R-1	R-1	R-0	R-0	R-1	R-1	R-1
	ProcessorID<7:0>							
7:0	R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
	Revision<7:0>							

图注：	W = 可写位	U = 未实现位，读为 0	
R = 可读位	HS = 硬件置 1 位	HC = 硬件清零位	r = 保留
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-29 **未实现：** 读为 0

bit 28 **UFRP：** 用户模式 $Status_{FR}$ 开关指令支持状态位
1 = 支持用户模式 $Status_{FR}$ 开关指令。Config5 寄存器（CP0 寄存器 16，选择 5）中的 UFR 位指定是否使能用户访问模式指令。

bit 27-25 **未实现：** 读为 0

bit 24 **FC：** 完整转换范围实现位
1 = 该位总是为 1，表示实现了完整转换范围。这意味着，所有数字都可以通过 FPU 转换为另一种类型（但是，如果 FCSR 中的 FS 位未置 1，则对于非规格化操作数仍然可以产生未实现操作异常）。

bit 23 **Has2008：** IEEE-754-2008 实现位
1 = 该位总是设置为 1，表示 FCSR 寄存器中存在 MAC2008、ABS2008 和 NAN2008 位（见寄存器 50-62）

bit 22 **F64：** 64 位 FPU 位
1 = 该位总是为 1，表示这是 64 位 FPU。

bit 21 **L：** 长定点数据类型实现位
1 = 该位总是为 1，表示实现了长定点数据类型。

bit 20 **W：** 字定点数据类型实现位
1 = 该位总是为 1，表示实现了字定点数据类型。

bit 19 **3D：** MIPS-3D ASE 实现位
0 = 该位总是为 0，表示未实现 MIPS-3D。

bit 18 **PS：** 配对单精度实现位
0 = 该位总是为 0，表示未实现配对单精度浮点数据类型。

bit 17 **D：** 双精度实现位
1 = 该位总是为 1，表示实现了双精度浮点数据类型。

bit 16 **S：** 单精度实现位
1 = 该位总是为 1，表示实现了单精度浮点数据类型。

bit 15-8 **ProcessorID<7:0>：** 处理器 ID 位
该值与 PRID 寄存器 （CP0 寄存器 15，选择 0）的相应位域匹配。

bit 7-0 **Revision<7:0>：** FPU 版本号位
指定 FPU 的版本号。该位域使软件可以区分相同浮点处理器类型的不同版本。

© 2013-2016 Microchip Technology Inc.

50.14.2 浮点条件代码寄存器 （FCCR， CP1 控制寄存器 25）（仅限 M-Class）

浮点条件代码寄存器 （Floating Point Condition Codes Register， FCCR）提供了一种替代方式来读写也会出现在 FCSR 寄存器中的浮点条件代码值。不同于 FCSR，在 FCCR 寄存器中，所有 8 个 FCC 位是连续的。

寄存器 50-59： FCC：浮点条件代码寄存器； CP1 控制寄存器 25

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	FCC<7:0>							

图注：	W = 可写位	U = 未实现位，读为 0	
R = 可读位	HS = 硬件置 1 位	HC = 硬件清零位	r = 保留
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-8 **未实现：**读为 0

bit 7-0 **FCC<7:0>：**FPU 浮点条件代码位

请参见寄存器 50-62：“浮点控制和状态寄存器 （FCSR， CP1 控制寄存器 31）”。

50.14.3 浮点异常寄存器 （FEXR， CP1 控制寄存器 26）（仅限 M-Class）

浮点异常寄存器（Floating Point Exceptions Register， FEXR）提供了一种替代方式来读写也会出现在 FCSR 寄存器中的原因和标志位。

寄存器 50-60： FEXR：浮点异常寄存器； CP1 控制寄存器 26

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	R/W-x	R/W-x
	—	—	—	—	—	—	Cause<5:4>	
15:8	R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0	U-0	U-0
	Cause<3:0>				—	—	—	—
7:0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0
	—	Flags<4:0>					—	—

图注：	W = 可写位	U = 未实现位，读为 0	
R = 可读位	HS = 硬件置 1 位	HC = 硬件清零位	r = 保留
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-18 **未实现：**读为 0

bit 17-12 **Cause<5:0>：**浮点原因位
原因位为 E、 V、 Z、 O、 U 和 I。关于这些位的定义，请参见表 50-33 和寄存器 50-62。

bit 11-7 **未实现：**读为 0

bit 6-2 **Flags<4:0>：**浮点标志位
标志位为 V、 Z、 O、 U 和 I。关于这些位的定义，请参见表 50-33 和寄存器 50-62。

bit 1-0 **未实现：**读为 0

表 50-33： 原因、使能和标志定义

位名称	说明
E	未实现操作 （只有原因位域中存在该位）。
V	无效操作。
Z	被零除。
O	上溢。
U	下溢。
I	不精确。

© 2013-2016 Microchip Technology Inc.

50.14.4 浮点使能寄存器 （FENR， CP1 控制寄存器 28）（仅限 M-Class）

浮点使能寄存器 （FENR）提供了一种替代方式来读写也会出现在 FCSR 寄存器中的使能、 FS 和 RM 位。

寄存器 50-61： FENR：浮点使能寄存器； CP1 控制寄存器 28

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	—	Enables<4:1>			
7:0	R/W-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
	Enables<0>	—	—	—	—	FS	RM<1:0>	

图注：	W = 可写位	U = 未实现位，读为 0	
R = 可读位	HS = 硬件置 1 位	HC = 硬件清零位	r = 保留
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-12 **未实现：** 读为 0

bit 11-7 **Enables<4:0>：** 浮点使能位
使能位为 V、 Z、 O、 U 和 I。关于这些位的定义，请参见表 50-33 和寄存器 50-62。

bit 6-3 **未实现：** 读为 0

bit 2 **FS：** 清除为零位
1 = 使能清除为零
0 = 禁止清除为零
关于清除为零操作的详细信息，请参见寄存器 50-62。

bit 1-0 **RM<1:0>：** 舍入模式位
这些位指示用于大多数浮点运算的舍入模式。
关于舍入模式的详细信息，请参见表 50-34 和寄存器 50-62。

50.14.5 浮点控制和状态寄存器 （FCSR， CP1 控制寄存器 31）（仅限 M-Class）

浮点控制和状态寄存器 （FCSR）控制浮点单元的操作。

它用于浮点单元的控制和状态报告：

- 选择 FPU 算术运算的默认舍入模式
- 使能 FPU 异常条件的陷阱
- 控制非规格化数字处理选项
- 报告在执行浮点指令期间产生的任何 IEEE 异常
- 报告在已完成的浮点指令中累积产生的任何 IEEE 异常
- 指示浮点比较指令的条件代码结果

对 FCSR 的访问不需要特权；它可以由有权访问 FPU 的任何程序读取或写入 （通过状态寄存器（CP0 寄存器 12，选择 0）中的协处理器使能位）。

寄存器 50-62： FCSR：浮点控制和状态寄存器； CP1 控制寄存器 31

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	FCC<7:1>							FS
23:16	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	FCC<0>	FO	FN	MAC2008	ABS2008	NAN2008	Causes<5:4>	
15:8	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	Causes<3:0>				Enables<4:1>			
7:0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
	Enables<0>	Flags<4:0>					RM<1:0>	

图注：	W = 可写位	U = 未实现位，读为 0	
R = 可读位	HS = 硬件置 1 位	HC = 硬件清零位	r = 保留
-n = POR 值	1 = 置 1	0 = 清零	x = 未知

bit 31-25 **FCC<7:1>：**浮点条件代码位

这些位用于记录浮点比较的结果，并通过测试它们来进行条件转移和条件传送。

FCC 位的使用在比较、转移或条件传送指令中指定。

bit 24 **FS：**清除为零位

该位控制非规格化操作数和极小结果的处理。关于这些位的操作的详细信息，请参见 **50.14.6 “FS/FO/FN 位的浮点操作”**。

bit 23 **FCC<0>：**浮点条件代码位

该位控制非规格化操作数和极小结果的处理。关于该位的操作的详细信息，请参见 **50.14.6 “FS/FO/FN 位的浮点操作”**。

bit 22 **FO：**清除改写位

关于该位的操作的详细信息，请参见 **50.14.6 “FS/FO/FN 位的浮点操作”**。

bit 21 **FN：**清除为最接近值位

关于该位的操作的详细信息，请参见 **50.14.6 “FS/FO/FN 位的浮点操作”**。

© 2013-2016 Microchip Technology Inc.

寄存器 50-62： FCSR：浮点控制和状态寄存器； CP1 控制寄存器 31 （续）

bit 20 **MAC2008：**融合乘加模式位

0 = 非融合乘加

IEEE 754-2008 标准的融合乘加运算在无界范围和精度下执行乘法和加法，在得到目标格式过程中仅进行一次舍入。

PIC32 内核不支持融合乘加。

PIC32 FPU 实现了非融合乘加，它会将中间乘法结果舍入为目标格式。

该位域适用于 MADD.fmt、 NMADD.fmt、 MSUB.fmt 和 NMSUB.fmt 指令。

bit 19 **ABS2008：**ABS 和 NEG 指令兼容位

1 = ABS 和 NEG 可接受 QNaN 输入，而不会产生陷阱。

IEEE 754-2008 标准要求 ABS 和 NEG 函数接受 QNaN 输入，而不产生陷阱。

该位在 PIC32 内核中总是置 1，表示支持 IEEE 754-2008 标准。

bit 18 **NAN2008：**静默 NaN 和信号 NaN 编码位

1 = IEEE 754-2008 NaN 编码

正如 IEEE 754-2008 标准所建议，静默 NaN 编码为小数的第一位为 1，信号 NaN 编码为小数域的第一位为 0。

该位在 PIC32 内核中总是置 1，表示支持 IEEE 754-2008 标准编码。

bit 17-12 **Cause<3:0>：**原因位

1 = 在指令执行过程中产生了相应的异常条件

0 = 在指令执行过程中未产生相应的异常条件

这些位指示在执行 FPU 算术指令的过程中产生的异常条件。

通过读取寄存器，可以确定先前的 FPU 算术指令导致的异常条件。

关于原因位的定义，请参见表 50-33。

bit 11-7 **Enables<4:0>：**使能陷阱位

1 = 相应的原因位置 1 时产生陷阱

0 = 相应的原因位置 1 时不产生陷阱

这些位控制对于 5 种条件 （V、 Z、 O、 U 和 I）中的任意一种条件，是否在出现浮点异常条件时产生陷阱。在 FPU 算术运算过程中或将某个值传送到 FCSR 或其某个替代表示形式中时，如果某个使能位及其相应的原因位均置 1，就会产生陷阱。

原因位 E 没有相应的使能位； MIPS 架构将不属于 IEEE 标准的未实现操作异常定义为总是使能。

关于使能位的定义，请参见表 50-33。

bit 6-2 **Flags<4:0>：**异常标志位

1 = 对于已完成的 FPU 指令，产生了相应的异常条件

0 = 对于已完成的 FPU 指令，未产生相应的异常条件

该位域指示自软件上次复位标志以来，对于已完成的指令产生的任何异常条件。当 FPU 算术运算抛出不会导致浮点异常的 IEEE 异常条件时 （使能位关闭），标志位域中的相应位会置 1，而其他位保持不变。

会导致浮点异常的算术运算 （使能位开启）不会更新标志位域。

注： 硬件永远不会复位该位域；软件必须显式地复位该位域。

关于标志位的定义，请参见表 50-33。

bit 1-0 **RM<1:0>：**舍入模式位

该位域指示用于大多数浮点运算的舍入模式 （一些运算使用特定的舍入模式）。

关于舍入模式的详细信息，请参见表 50-34。

表 50-34：　舍入模式定义

RM 位域编码	RM 位域说明
0	RN—— 舍入为最接近值 将结果舍入为最接近的可表示值。当两个可表示值的距离相等时，结果舍入为最低位为零的值 （偶数）。
1	RZ—— 向零舍入 将结果舍入为最接近但大小不大于结果的值。
2	RP—— 向正无穷大舍入 将结果舍入为最接近但不小于结果的值。
3	RM—— 向负无穷大舍入 将结果舍入为最接近但不大于结果的值。

50.14.6　FS/FO/FN 位的浮点操作

CP1 FCSR 寄存器中的 FS、 FO 和 FN 位控制非规格化操作数和极小结果 （即介于 $\pm 2^{E_min}$ 之间的非零结果）的处理，由此 FPU 可以直接处理这些情况，而不用依赖于速度慢得多的软件处理程序。其代价是损失 IEEE 兼容性和准确性 （使用 FO 位除外），因为 FPU 提供的是极小规格化或零结果，而不是软件处理程序提供的更准确的非规格化结果。其好处是可以显著提高性能和精度。

对于下面列出的指令，使用 FS、 FO 和 FN 位会影响非规格化浮点数字和极小结果的处理：

表 50-35：　FS/FO/FN 位浮点操作

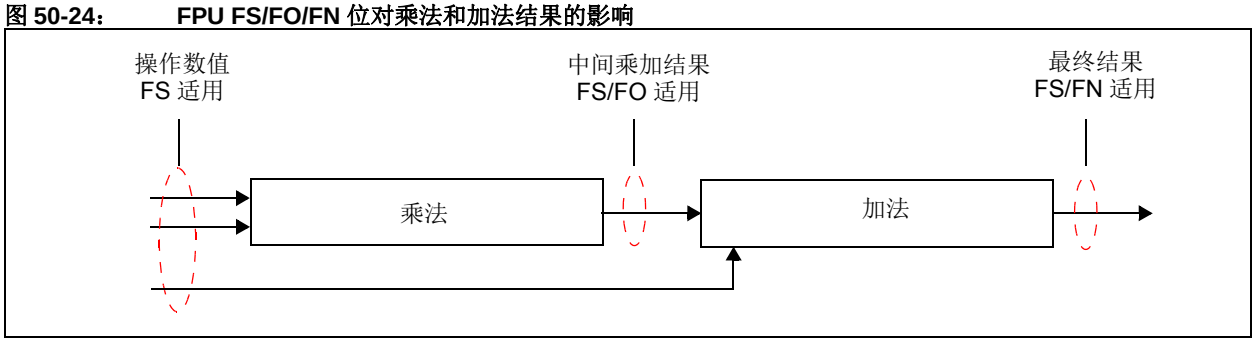
FS/FO/FN 位	受影响的指令
FS 和 FN	ADD、 CEIL、 CVT、 DIV、 FLOOR、 MADD、 MSUB、 MUL、 NMADD、 NMSUB、 RECIP、ROUND、RSQRT、SQRT、TRUNC、SUB、ABS、C.cond 和 NEG。见**注 1**。
FO	MADD、 MSUB、 NMADD 和 NMSUB

注　1： 对于 ABS、 C.cond 和 NEG，当 FS = 0 时，非规格化输入操作数或极小结果不会导致未实现异常。无论如何，FS = 1 时都会实现清除为零，从而使这些运算返回与等效的算术 FPU 运算序列相同的结果。

对于操作数或结果中的非规格化数字，表 50-35 中未列出的指令不会导致未实现操作异常。

图 50-24 说明了 FS、FO 和 FN 位如何控制非规格化数字的处理。对于不属于乘法或加法类型的指令 （如 DIV），只有 FS 和 FN 位适用。

图 50-24：　FPU FS/FO/FN 位对乘法和加法结果的影响



© 2013-2016 Microchip Technology Inc.

50.14.6.1 FPU 清除为零位

当清除为零（Flush-To-Zero，FS）位置 1 时，非规格化输入操作数会被清除为零。极小结果将根据舍入模式设置来清除为零或所采用格式的最小规格化数字 （MinNorm）。表 50-36 列出了极小结果的清除行为。

表 50-36： 极小结果的清除为零

舍入模式	负极小结果	正极小结果
RN （RM = 0）	-0	+0
RZ （RM = 1）	-0	+0
RP （RM = 2）	-0	+MinNorm
RM （RM = 3）	-MinNorm	+0

结果清除基于通过舍入使用无界指数范围的尾数计算得到的中间结果；即，在进行舍入之前，极小数字不会通过移入前导零而规格化到所支持的指数范围。

非规格化操作数值和极小结果的处理取决于 FS 位的设置，如表 50-37 所示。

表 50-37： 基于 FS 位设置的非规格化操作数值和极小结果处理

FS 位设置	非规格化操作数值的处理
0	产生未实现操作异常
1	操作数会被清除为零，而极小结果会被强制设为零或 MinNorm，而不是导致未实现操作异常。

50.14.6.2 FPU 清除改写位

当清除改写 （Flush Override， FO）位置 1 时，将不会根据 FS 位清除任何乘加类型指令的极小中间结果。中间结果保持为内部的规格化格式，以提高准确性。

FO 仅适用于乘加类型指令的中间结果。

极小中间结果的处理取决于 FO 和 FS 位，如表 50-38 所示。

表 50-38： 基于 FO 和 FS 位设置的极小中间结果处理

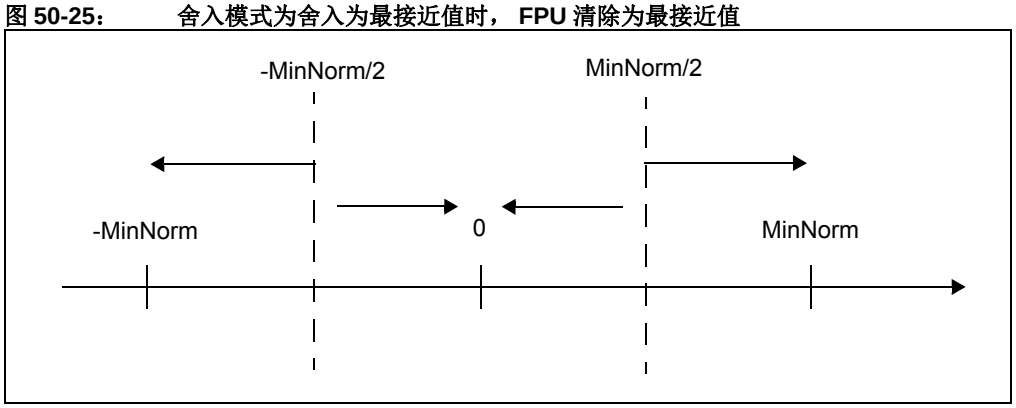
FO 位设置	FS 位设置	极小结果值的处理
0	0	产生未实现操作异常。
0	1	中间结果会被强制设为对于未产生陷阱的下溢异常提供的值 （见表 50-21：“FPU 对于未产生陷阱的异常提供的结果”），而不是导致未实现操作异常。
1	x	中间结果会保持为内部格式 （它可以视为具有通常的尾数精度，但具有无限的指数精度），而不强制设为特定值或产生异常。

50.14.6.3 FPU 清除为最接近值位

当清除为最接近值（Flush-to-Nearest，FN）位置 1 并且舍入模式为舍入为最接近值（Round to Nearest， RN）时，极小最终结果会被清除为零或 2E_min （MinNorm）。如果极小数字严格小于 MinNorm/2，则将结果清除为零；否则，将它清除为 MinNorm （见图 50-25）。清除后的结果与清除前的结果具有相同的符号。请注意， FN 位优先于 FS 位。

对于舍入为最接近值 （RN）之外的所有其他舍入模式，将 FN 位置 1 会导致最终结果被清除为零或 MinNorm，就如同 FS 位置 1。

图 50-25： 舍入模式为舍入为最接近值时， FPU 清除为最接近值



极小最终结果的处理取决于 FN 和 FS 位，如表 50-39 所示。

表 50-39： FPU 基于 FN 和 FS 位设置的极小最终结果处理

FN 位设置	FS 位设置	极小结果值的处理
0	0	产生未实现操作异常。
0	1	最终结果会被强制设为对于未产生陷阱的下溢异常提供的值 （见表 50-21：“FPU 对于未产生陷阱的异常提供的结果”），而不是导致未实现操作异常。
1	x	在处于舍入为最接近值 （RN）舍入模式时，最终结果舍入为零或 2^{E_min} （MinNorm），取两个值中最接近的。对于其他舍入模式，给出的最终结果与 FS 设置为 1 时相同。

50.14.6.4 建议的 FPU FS/FO/FN 位设置

表 50-40 总结了 FPU FS/FO/FN 位的建议设置。

表 50-40： FPU FS/FO/FN 位的建议设置

FS 位设置	FO 位设置	FN 位设置	备注
0	0	0	IEEE 兼容模式。对于非规格化操作数和极小结果的性能低
1	0	0	常规的嵌入式应用程序。对于非规格化操作数和极小结果的性能高
1	1	1	最高准确性和性能的配置。 注：在该模式下， MADD 可能会返回与等效的 MUL 和 ADD 运算序列不同的结果。

50.14.6.5 FPU FCSR 原因位更新流程

50.14.6.5.1由 CTC1 指令触发的 FPU 异常

无论所针对的控制寄存器如何，协处理器 CTC1 指令都会导致对 FCSR 的使能和原因位域进行检查，以确定是否抛出异常。

© 2013-2016 Microchip Technology Inc.

50.14.6.6 FPU 计算一般流程

计算分两个步骤执行：

1. 计算具有无界指数范围的舍入后尾数。
2. 如果来自步骤 1 的结果上溢或极小，则清除为默认结果 （对于支持非规格化结果的指令（如 MOV），不会对非规格化结果发生清除操作）。

原因位域在执行这两个步骤中的每一个之后更新。在这两个步骤中检测到任何已使能的异常都会导致陷阱，并且后续步骤不会再进一步更新原因位域。

步骤 1 可以置 1 原因位 I、 U、 O、 Z、 V 和 E。

E 的优先级高于 V； V 的优先级高于 Z； Z 的优先级高于 U 和 O。因而，当 E、V 或 Z 在步骤 1 中置 1 时，其他任何原因位都不可以置 1。但是请注意，如果非规格化操作数被清除 （FS = 1），则 I 和 V 都可以置 1。I、U 和 O 可以单独或成对（IU 或 IO）置 1。在步骤 1 中，U 和 O 永远不可以同时置 1。如果计算得到的无界指数超出规格化 IEEE 格式支持的指数范围，则 U 和 O 置 1。

如果生成默认结果，步骤 2 可以将 I 置 1。

50.14.6.7 FPU 乘加流程

对于乘加类型的指令，计算多了两个步骤：

1. 为乘法计算具有无界指数范围的舍入后尾数。
2. 如果来自步骤 1 的结果上溢或极小，则清除为默认结果 （如果 FO = 1，则不会对极小结果发生清除操作）。
3. 为加法计算具有无界指数范围的舍入后尾数。
4. 如果来自步骤 3 的结果上溢或极小，则清除为默认结果。

原因位域在执行这 4 个步骤中的每一个之后更新。在这 4 个步骤中检测到任何已使能的异常都会导致陷阱，并且后续步骤不会再进一步更新原因位域。

步骤 1 和步骤 3 可将原因位置 1，如 **50.14.6.6 “FPU 计算一般流程”** 中对于步骤 1 所述。

如果生成默认结果，步骤 2 和步骤 4 可以将 I 置 1。

虽然 U 和 O 永远不可以在步骤 1 或步骤 3 中同时置 1，但 U 和 O 可能在步骤 3 中执行乘加之后都置 1，因为 U 可能在步骤 1 中置 1， O 可能在步骤 3 中置 1。

50.14.6.8 输入操作数的 FPU 原因更新流程

当 FS = 1 时，步骤 1 或步骤 3 的非规格化输入操作数总是会使原因位 I 置 1。例如，如果到达了步骤 3， SNaN+DeNorm （非规格化）会使 I（和 V）置 1 （对于乘加类型的指令）。

与输入操作数直接相关的条件（例如，I/E 由于 DeNorm 置 1，V 由于 SNaN 和 QNaN 传播置 1）在逻辑使用操作数的步骤中检测。例如，对于乘加类型的指令，输入操作数 *fr* 导致的异常条件在步骤 3 中检测。

注：	*fr* 寄存器是在 MADD.fmt、 NMADD.fmt、 MSUB.fmt 和 NMSUB.fmt FPU 指令中出现的 FPU 寄存器。

50.14.6.9 未实现操作的 FPU 原因更新流程

请注意，原因位 E 是一个特殊的位；它会清除在先前步骤中完成的任何原因更新。例如，如果步骤 3 导致 E 置 1，则在步骤 1 或步骤 2 中完成的任何 I、 U 或 O 原因更新都会被清除。当产生未实现操作陷阱时，原因位域中只有 E 会置 1。

50.15 microMIPS 执行

microMIPS™ 最大程度地降低了应用程序的代码资源占用量，从而降低了存储器成本（嵌入式存储器的成本相当高）。同时保持了 MIPS 内核的高性能。采用该技术可实现最佳结果，且无需花时间规划应用。较小的代码资源占用量通常会使每次执行任务时的功耗较低，因为存储器访问数较少。

MIPS32 版本 3.0 架构支持 MIPS32 指令集和 microMIPS 增强型 MIPS32 指令集。

50.16 MCU ASE 扩展

MCU ASE 为 microMIPS 和 MIPS32 架构扩展了一组旨在进行单片机市场开发的新特性。

MCU ASE 在以下几个领域进行了增强：中断传送和中断延迟。

50.16.1 中断传送

MCU ASE将中断硬件输入数量从63扩展到255（外部中断控制器（External Interrupt Controller，EIC）模式），且均具有单独的优先级和向量。

50.16.2 中断延迟降低

MCU ASE 对 microMIPS 和 MIPS32 进行了一组扩展，这些扩展降低了处理器对发出的中断作出响应所需的延迟。

50.16.2.1 中断向量预取

通常在 MIPS 架构处理器上，当发出中断或异常时，必须先清除执行流水线才能取中断 / 异常处理程序。需要这样做以避免混合中断 / 故障程序和异常处理程序的现场。 MCU ASE 引入了硬件机制，即每当中断输入信号改变时就预取中断异常向量。预取存储器事务与流水线清除和指定异常优先级并行进行。这降低了执行中断处理程序的第一条指令的整体延迟。

50.16.2.2 自动中断现场保护

影子寄存器集的使用避免了在处理中断前必须保存通用寄存器的软件步骤。

MCU ASE 增加了其他硬件逻辑，这些逻辑用来在堆栈中自动保存一些 CP0 状态并自动更新一些 CP0 寄存器以为中断处理做准备。

50.16.2.3 自动中断现场恢复

自动中断现场恢复为自动现场保护的镜像操作，该特性可自动从堆栈恢复一些CP0 寄存器并准备一些 CP0 寄存器以返回非异常模式。该特性在 ASE 中引入的 `IRET` 指令内实现。

50.16.2.4 中断链

是自动中断现场恢复的可选特性，该特性允许在处理主中断后处理第二次中断，而无需返回非异常模式 （和通常有必要进行的相关流水线清除）。

© 2013-2016 Microchip Technology Inc.

50.17 MIPS DSP ASE 扩展

> **注：** DSP ASE 并非在所有器件上都可用。要确定可用性，请参见具体器件数据手册中的**“CPU”**章节。

MIPS DSP 应用特定扩展版本 2 是 MIPS32 架构的扩展。该扩展包含新的整数指令和状态（包括新的 HI/LO 累加寄存器对和一个 DSP 控制寄存器）。该扩展在广泛的 DSP、多媒体和涉及音频和视频处理应用的类似于 DSP 的算法中至关重要。该扩展支持固有小数格式数据类型运算和寄存器单指令多数据（Single Instruction Multiple Data，SIMD）运算，如加、减、乘和移位运算。此外，该扩展包含以下实质上使 DSP 算法计算更有效率的特性：

- 支持复杂操作数的乘法
- 可变位插入和抽取
- 实现和使用虚拟循环缓冲区
- 支持算术饱和及溢出处理功能
- 零循环开销饱和及舍入操作

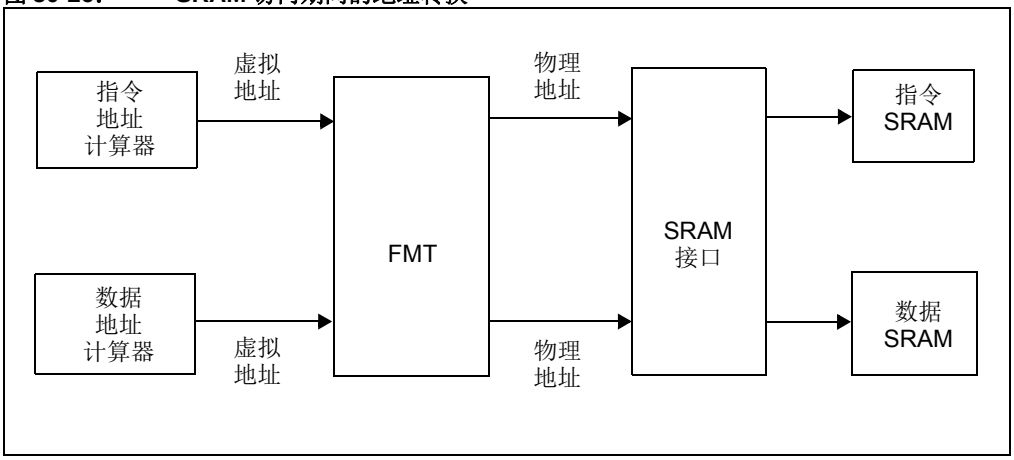
关于该扩展的信息，请参见 *“MIPS32® Architecture for Programmers VolumeIV-e: The MIPS® DSP Application-Specific Extension to the MIPS32® Architecture”* —— MD00374 中关于该扩展的信息。该文档可从 Imagination Technologies Ltd. 网站下载：http://www.imgtec.com/mips/architectures/dsp.asp。

50.18 存储器模型 （仅限 MCU）

软件使用的虚拟地址在发送给 CPU 总线之前，会通过存储器管理单元 （Memory Management Unit，MMU）转换为物理地址。对于该转换，基于 microAptiv MCU 微处理器内核的 PIC32 器件使用固定映射方式。

关于系统存储器模型的更多信息，请参见 《PIC32 系列参考手册》中的**第 3 章 “存储器构成”**（DS60001115）。

图 50-26： SRAM 访问期间的地址转换



50.18.1 高速缓存功能

CPU 使用取指、装载或存储操作的虚拟地址来确定是否访问高速缓存。 kseg0 或 useg/kuseg 中的存储器访问可以进行高速缓存，而 kseg1 中的访问不能进行高速缓存。 CPU 使用 Config 寄存器中的 CCA 位来确定存储器段的高速缓存功能。如果相应的 CCA = 011_2，则说明存储器访问可高速缓存。关于高速缓存操作的更多信息，请参见 《PIC32 系列参考手册》中的**第 4 章 “预取高速缓存模块”**（DS60001119）。

50.18.1.1 小尾数字节顺序

在以字节分辨率对存储器进行寻址的 CPU 上，对于多字节数据项存在一种约定，指定高字节到低字节的顺序。大尾数字节顺序规定最低地址包含 MSB。小尾数顺序规定最低地址包含多字节数据的 LSB。 PIC32 CPU 支持小尾数字节顺序。

图 50-27： 大尾数字节顺序

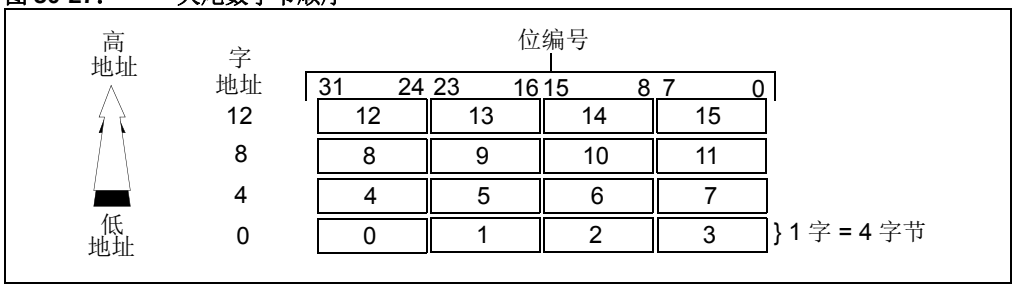
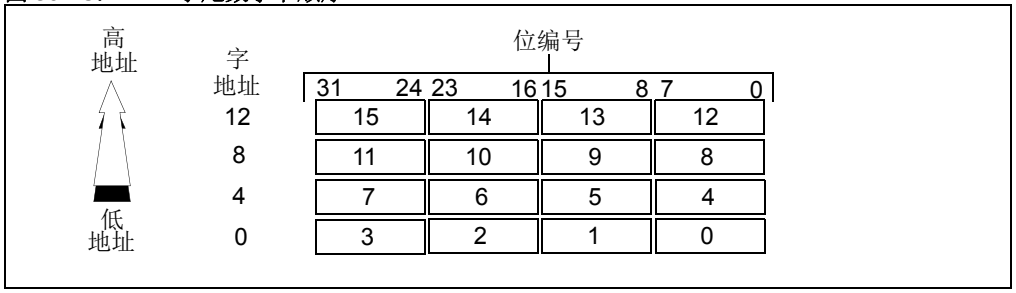


图 50-28： 小尾数字节顺序



© 2013-2016 Microchip Technology Inc.

50.19 存储器管理 （仅限 MPU）

采用 MPU 内核的 PIC32 器件包含一个存储器管理单元（MMU），该单元使用页表缓冲区（TLB）将虚拟页地址转换为物理页地址。操作系统使用该特性来管理在同一虚拟存储器中运行的多项任务，方法是将任务映射到单独的物理存储单元。 MMU 还可提供对物理存储器区域的保护以及定义高速缓存协议。采用 MPU 内核的 PIC32 器件支持的页大小范围为 4 KB 到 1 MB。

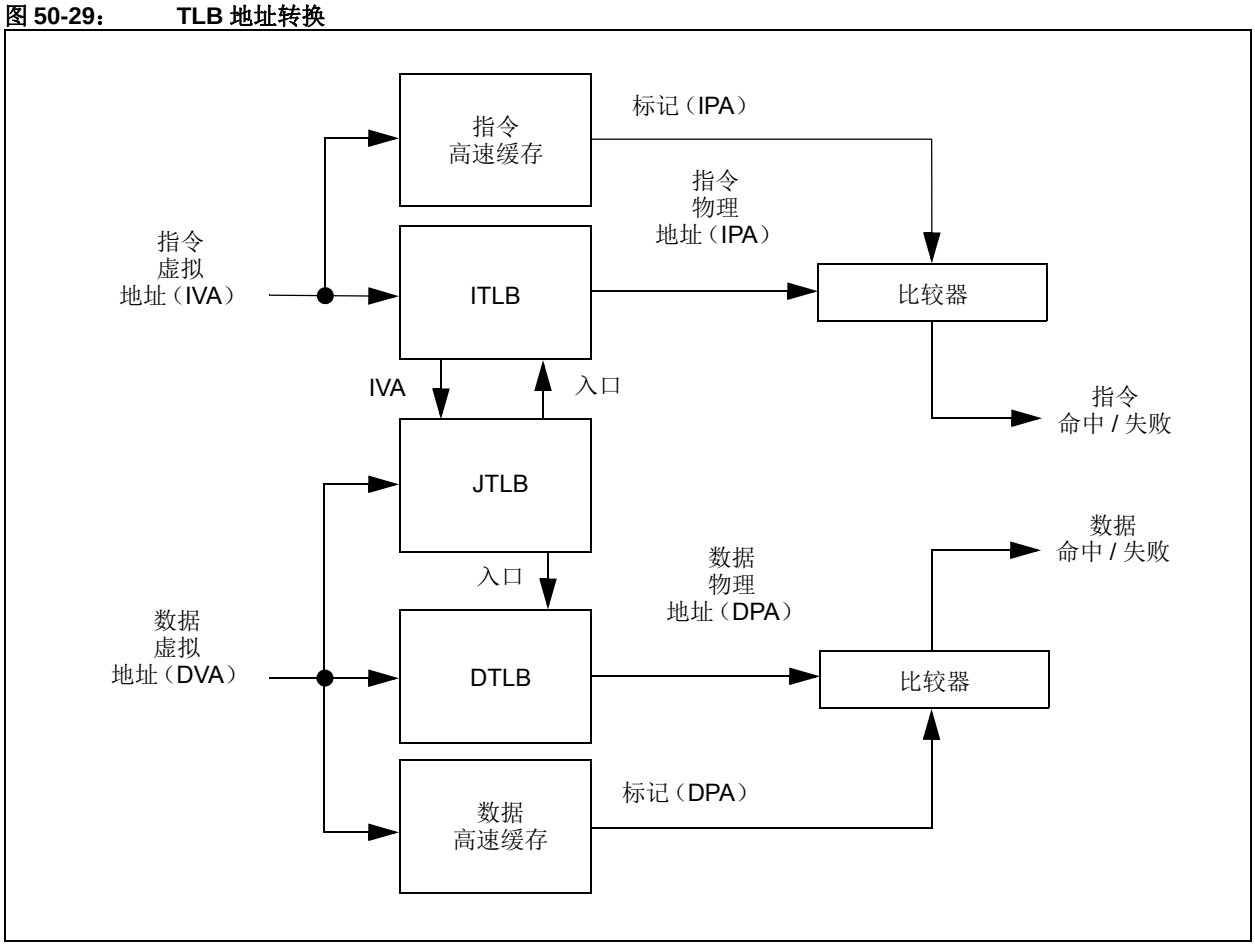
MMU 的内核为 TLB。 TLB 包含 3 个地址转换缓冲区：

- 16 个双输入完全相关的联合 TLB （Joint TLB， JTLB）
- 4 输入指令微型 TLB （Instruction micro-TLB， ITLB）
- 4 输入数据微型 TLB （Data micro-TLB， DTLB）

转换页地址时，首先访问 ITLB 或 DTLB。如果未在微型 TLB 中找到转换，将访问 JTLB。如果未在 JTLB 中找到转换，将发生异常。

图 50-29 显示了采用 MPU 内核的 PIC32 器件中 TLB 如何与高速缓存访问交互。

图 50-29： TLB 地址转换



50.19.1 虚拟存储器和工作模式

所有 PIC32 器件支持三种工作模式：

- 用户模式
- 内核模式
- 调试模式

内核在复位或识别到异常时进入内核模式。在内核模式下，软件可访问整个 4 GB 地址空间以及 CP0 寄存器。

用户模式限制为访问首个 2 GB 地址空间（0x00000000 到 0x7FFFFFFF），且不能访问 CP0 功能。在用户模式下访问 0x7FFFFFFF 以上的虚拟地址将导致异常。

发生调试异常时进入调试模式。在调试模式下，软件可访问所有内核模式地址和功能以及调试段 dseg，该段将覆盖部分内核段 kseg3。

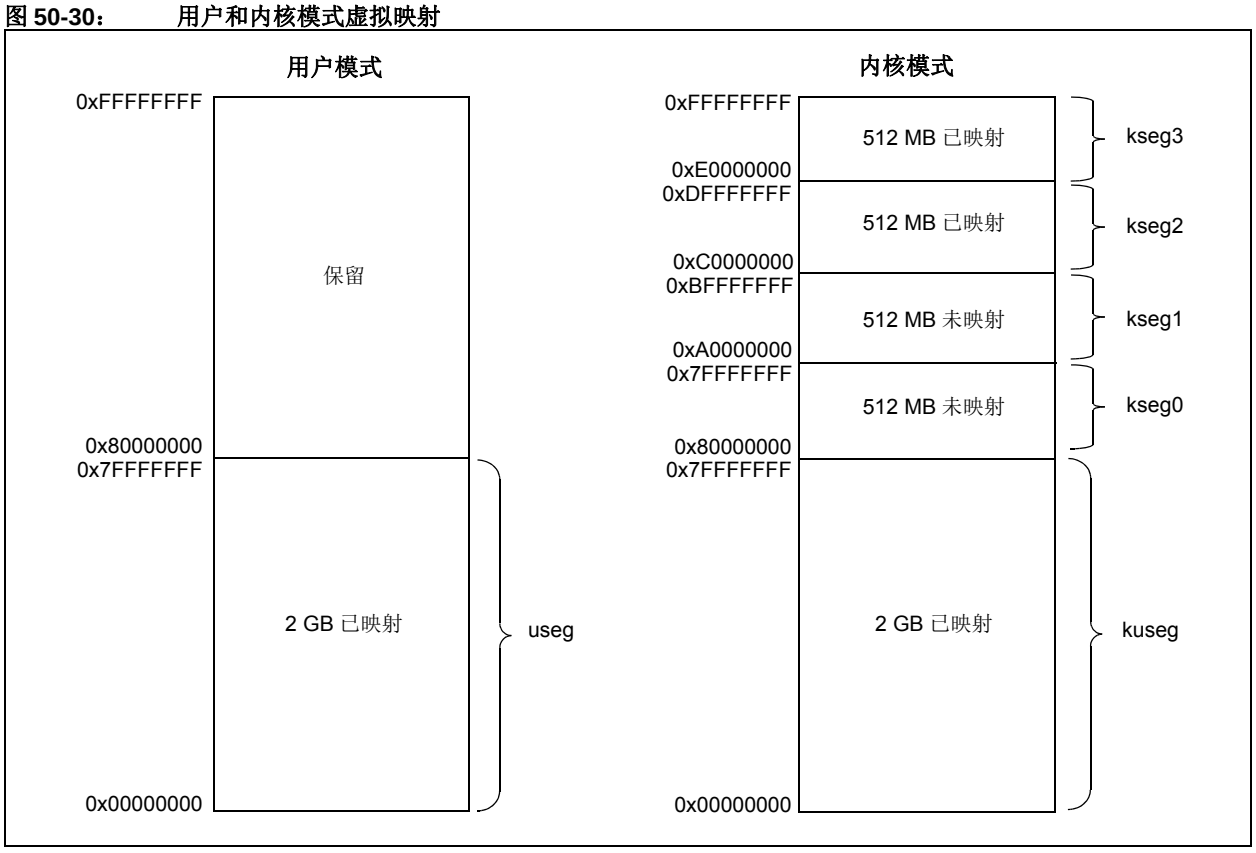
虚拟存储段因操作模式而异。关于 PIC32 虚拟存储器映射，请参见 **50.11.4 “处理器模式”** 中的图 50-14。

50.19.1.1 映射和未映射段

使用 MMU 将虚拟页地址转换为物理存储单元的存储器段被视为已映射。未使用 MMU 的那些段被视为未映射。未映射段具有虚拟地址到物理地址的固定转换。复位时，必须从未映射存储器执行直到 TLB 设定为执行地址转换。

除了 kseg0，未映射段始终为非高速缓存。kseg0 的高速缓存功能在 CP0 寄存器的 K0 域中设置。映射段的高速缓存功能在 CP0 寄存器的 K23 和 KU 域设置。图 50-30 给出了内核模式和用户模式的映射和未映射虚拟存储器区域。当工作在调试模式下时，dseg 区域未映射。所有其他区域的映射与内核模式相同。

图 50-30： 用户和内核模式虚拟映射



© 2013-2016 Microchip Technology Inc.

50.19.2 页表缓冲区 （TLB）

TLB 由一个联合和两个微型地址转换缓冲区组成：

- 16 个双输入完全相关的联合 （JTLB）
- 4 输入完全相关的指令微型 TLB （ITLB）
- 4 输入完全相关的数据微型 TLB （DTLB）

50.19.2.1 联合 TLB

JTLB 将 32 个虚拟页地址映射到其对应的物理地址。此 TLB 的主要任务是将虚拟页地址以及与其相对应的地址空间 ID （Address Space ID， ASID）转换为物理存储地址。操作系统通常为每个用户程序或进程分配一个 ASID。 TLB 可帮助操作系统将每个用户进程分离到其自己的物理存储空间。

通过把虚拟地址的高位以及 ASID 位与每个 JTLB 标记条目进行比较就可以完成虚拟地址到物理地址的转换。因为 JTLB 可用于对指令和数据与虚拟页地址进行转换，因此，称为“联合”TLB。

JTLB 由包含页地址转换的奇数和偶数条目对构成，页大小在 4 KB 到 1 MB 之间。每个虚拟标记条目对应两个物理数据条目：一个偶数页条目和一个奇数页条目。不参与标记比较的虚拟地址最高位用于确定使用两个数据条目中的哪一个。图 50-31 给出了 JTLB 中一个具有两个条目的具体结构。

图 50-31： JTLB 条目 （标记和数据）

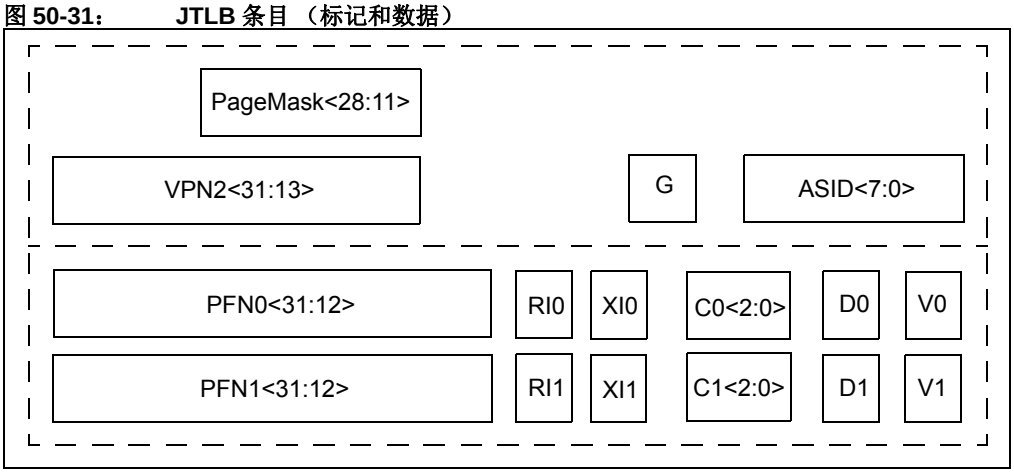


表 50-41 描述了 JTLB 条目的每个域。

表 50-41： TLB 标记条目域

位名称	说明
PageMask<28:11>	页掩码值。页掩码通过屏蔽相应的 VPN2 位来定义页大小。它还确定用来判断奇偶页（PFN0-PFN1）的地址位。 **PageMask** / **页大小** / **奇 / 偶存储区选择位** 00 0000 0000 0000 0011 / 4 KB / VAddr<12> 00 0000 0000 0000 1111 / 16 KB / VAddr<14> 00 0000 0000 0011 1111 / 64 KB / VAddr<16> 00 0000 0000 1111 1111 / 256 KB / VAddr<18> 00 0000 0011 1111 1111 / 1 MB / VAddr<20>
VPN2<31:13>	虚拟页号除以 2。该域包含虚拟页号除以 2 的高位。由于它表示 TLB 页对，因此将被除以 2。Bit <31:25> 始终参与 TLB 查找比较。Bit <24:13> 是否参与取决于页大小，由 PageMask 定义。
G	全局位。置 1 时，表示本条目对所有地址空间都有效，从而在 TLB 查找比较中将禁止使用 ASID。
ASID<7:0>	地址空间标识符。用来标识与该 TLB 条目相关的进程或线程。
PFN0<31:12> PFN1<31:12>	物理帧号。定义物理地址的高位。对于大于 4 KB 的页大小，实际使用的仅是这些位中的一部分。
C0<2:0> C1<2:0>	高速缓存功能。指示高速缓存功能属性并确定该页是否应置入高速缓存。 **C<2:0>** / **一致性属性** 000 / 可高速缓存、非一致性、直写和无写分配 001 / 可高速缓存、非一致性、直写和写分配 010 / 非高速缓存 011 / 可高速缓存、非一致性、回写和写分配 100-111 / 保留
RI0 RI1	读禁止位。指示该页受读保护。如果该位置 1，且 PageGrain 寄存器的 IEC 位置 1，任何尝试读该页的操作将导致 TLB 读禁止异常。
XI0 XI1	执行禁止位。指示该页受执行保护。如果该位置 1，且 PageGrain 寄存器的 IEC 位置 1，任何尝试从该页进行取指的操作将导致 TLB 执行禁止异常。
D0 D1	dirty 位。指示该页已被写入和 / 或可写入。若置 1，允许写入该页。若该位未置 1，写该页将导致 TLB 修改异常。
V0 V1	有效位。指示 TLB 条目有效。若置 1，允许访问该页。若该位未置 1，访问该页将导致 TLB 无效异常。

使用 TLBWI 或 TLBWR 指令填充表条目。执行任一指令前，以下 CP0 寄存器必须使用写入 TLB 条目的信息更新：

- PageMask 寄存器 （CP0 寄存器 5，选择 0）中的 PageMask 置 1。
- EntryHi 寄存器 （CP0 寄存器 10，选择 0）中的 VPN2、 VPN2X 和 ASID 置 1
- EntryLo0 寄存器 （CP0 寄存器 2，选择 0）中的 PFN0、 C0、 D0、 V0 和 G 置 1
- EntryLo1 寄存器 （CP0 寄存器 3，选择 0）中的 PFN1、 C1、 D1、 V1 和 G 置 1

注： 全局位由EntryLo0和EntryLo1寄存器共同决定。写入JTLB中的G位的值是EntryLo0和 EntryLo1 寄存器中 G 位进行逻辑与运算的结果。

© 2013-2016 Microchip Technology Inc.

50.19.2.2 微型 TLB （ITLB 和 DTLB）

ITLB 是一个专用于指令流的小型（即微型）TLB。DTLB 是一个小型 TLB，用于为装载 / 存储地址提供比 JTLB 更快的转换速度。这两个 TLB 均为 4 输入完全相关且完全由硬件管理的。

首先通过 ITLB 处理取指地址转换。如果取指未命中 ITLB，下一个时钟周期将访问 JTLB。如果成功，该条目将被复制到 ITLB。然后再次访问 ITLB，并且地址成功转换。

数据转换并行访问 ITLB 和 JTLB。如果 DTLB 未命中而 JTLB 命中，可在同一时钟周期内重新装载 DTLB。接着在下一个时钟周期访问 DTLB。

50.19.3 虚拟地址到物理地址的转换

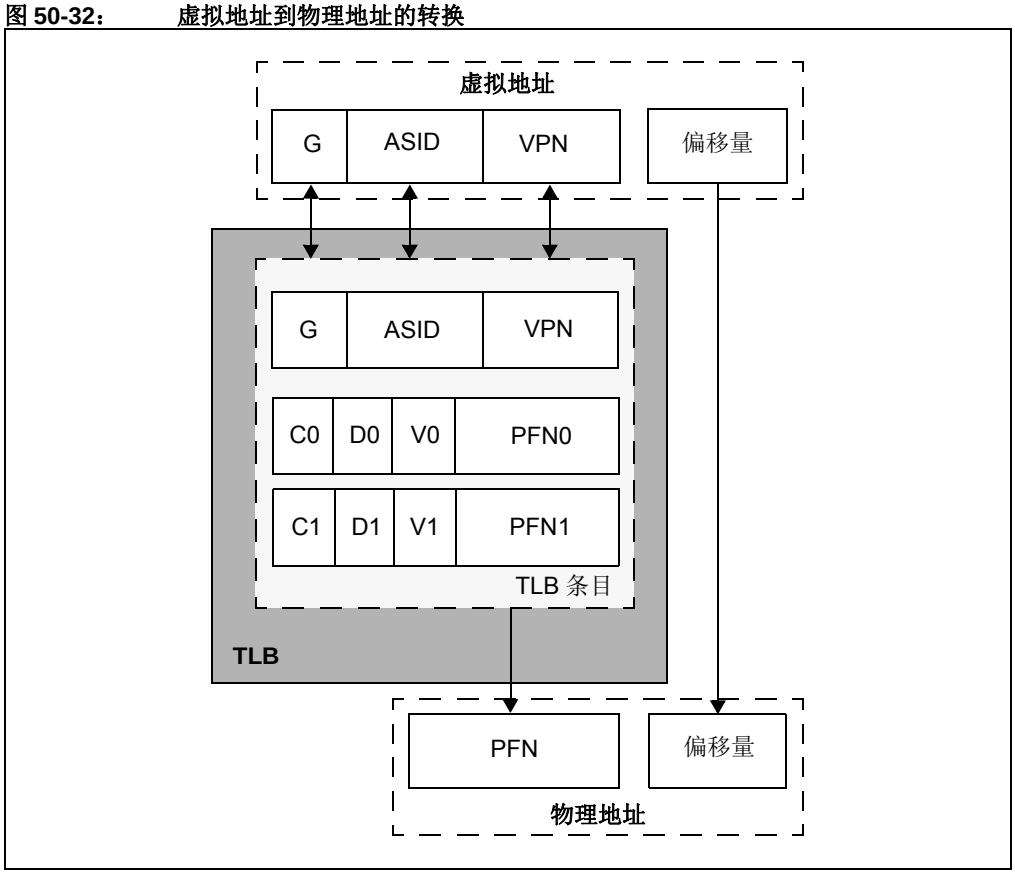
TLB 本质上相当于页表条目的高速缓存。将虚拟地址转换为物理地址时，地址的虚拟页号（Virtual Page Number，VPN）（地址的高位）与 TLB 中存储的虚拟页号相比较。当满足以下条件时将发生 TLB 命中：

- 虚拟地址的 VPN 与 TLB 中存储的 VPN 匹配且：
  - TLB 条目的奇数页和偶数页的全局 （G）位均置 1，或
  - 虚拟地址的 ASID 域与 TLB 条目的 ASID 域相同

如果未满足这两个条件，处理器就产生一个 TLB 未命中异常，同时允许软件根据存储器中虚拟 / 物理地址页表重填 TLB。

图 50-32 给出了虚拟地址到物理地址的转换。

图 50-32： 虚拟地址到物理地址的转换



每个 JTLB 条目都包含一个标记和两个数据域。TLB 命中时，虚拟地址的高位替换成相应的数据域中存储的页帧号（Page Frame Number，PFN）。TLB 未命中时，将产生一个异常，同时软件根据存储器中存储的页表重新填充 TLB。

50.19.4 TLB 条目替换

通常， TLB 未命中时，会通过随机替换算法替换 JTLB 条目。 CPU 还提供了另一种功能，通过 CP0 Wired 寄存器将一个可设定数量的映射锁入 TLB （详细信息，请参见寄存器 50-8）。

JTLB 支持不同大小的页，范围为 4 KB 到 1 MB，但必须是 4 的整数倍。可以以条目为单位对页大小进行配置，方法是将所需页大小装载到 CP0 PageMask 寄存器，然后再写入新条目。通常使用该特性将较大的存储器区 （例如帧缓冲器）映射为单个 TLB 条目。

50.19.5 TLB 指令

表 50-42 列出了 PIC32 支持的 TLB 相关指令。关于这些指令的详细信息，请参见 **50.21.5.5“TLB 指令 （仅限 MPU）”**。

表 50-42： TLB 指令

指令	说明
TLBP	检测页表缓冲区
TLBR	读页表缓冲区
TLBWI	索引写入页表缓冲区
TLBWR	随机写入页表缓冲区

© 2013-2016 Microchip Technology Inc.

50.20 L1 高速缓存 （仅限 MPU）

采用 MPU 微处理器内核的 PIC32 器件具有独立的指令和数据高速缓存。使用独立的高速缓存允许同时进行指令和数据引用。这两个高速缓存均为虚索引实标记 （Virtually Index, Physically-tagged， VIPT），这允许高速缓存访问与虚拟到物理地址转换并行进行。

50.20.1 高速缓存配置

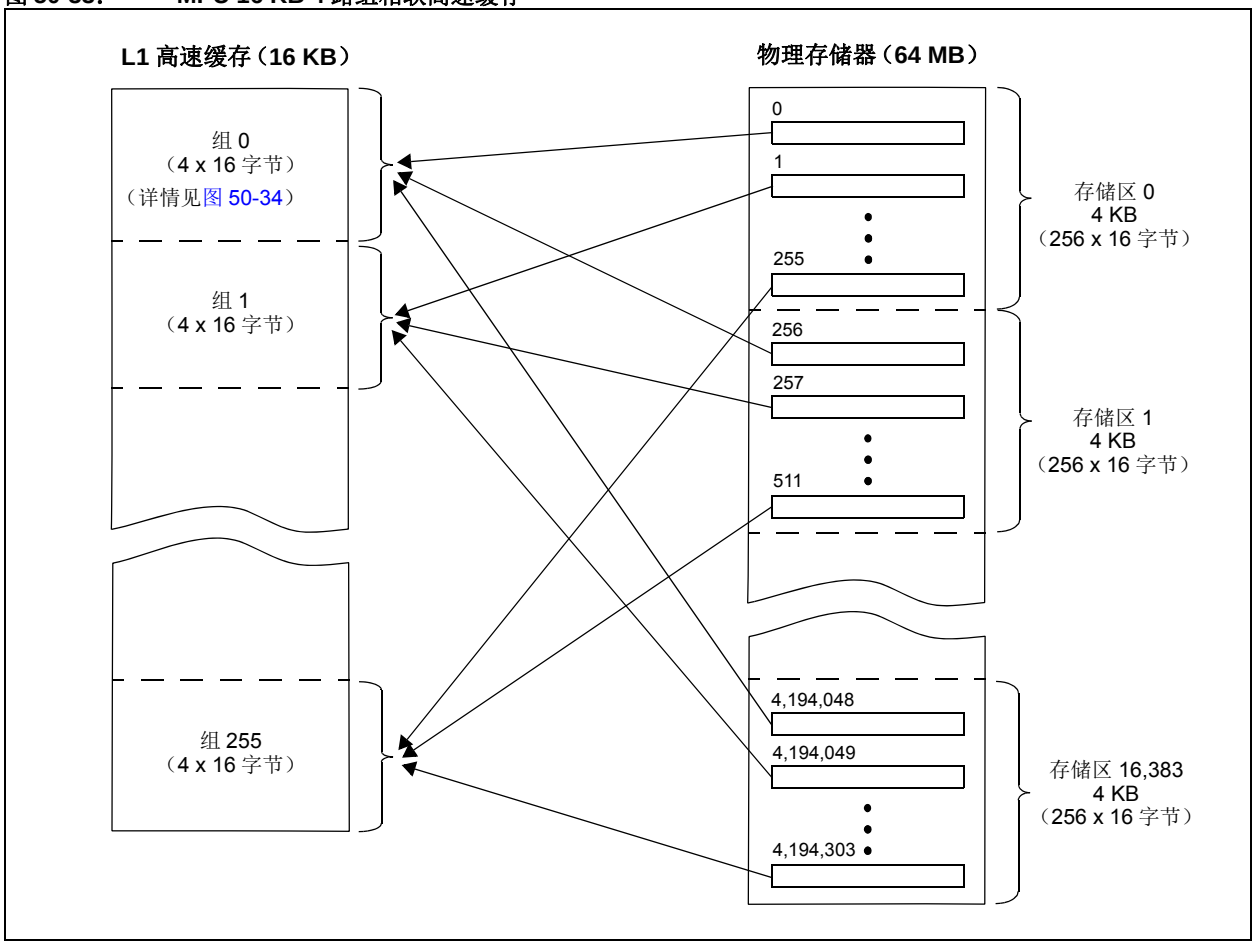
指令和数据高速缓存可能具有不同的大小和相联性。例如，指令高速缓存可以为 16 KB 大小，具有 4 路组相联结构，而数据高速缓存可以为 4 KB，具有 2 路组相联结构。组相联高速缓存是直接映射和全相联高速缓存的折衷。

在直接映射高速缓存中，任何存储单元只能映射到高速缓存中的单个位置。直接映射高速缓存很容易实现和寻址，但是不够灵活且在两个频繁使用的存储单元共用一个映射时导致颠簸。

全相联高速缓存允许任何存储单元映射到高速缓存中的任何位置。这可最大程度地减少冲突，但实现成本比较高。

在组相联高速缓存中，高速缓存被分为叫做组的行组。每个组的行数称为相联度。每个存储单元映射到一个组，且可高速缓存在组中的任何行中。例如，在 16 KB、每行 16 字节、4 路组相联高速缓存中，高速缓存被分为 256 组，每组 4 行。如图 50-33 所示，任何可高速缓存 （16 字节）存储单元映射到单个组中，且可映射到该组中的任一四行。在具有 64 MB 可高速缓存存储器的系统中，每个组由 16,384 （256 x 16 字节）存储区共用。

图 50-33： MPU 16 KB 4 路组相联高速缓存



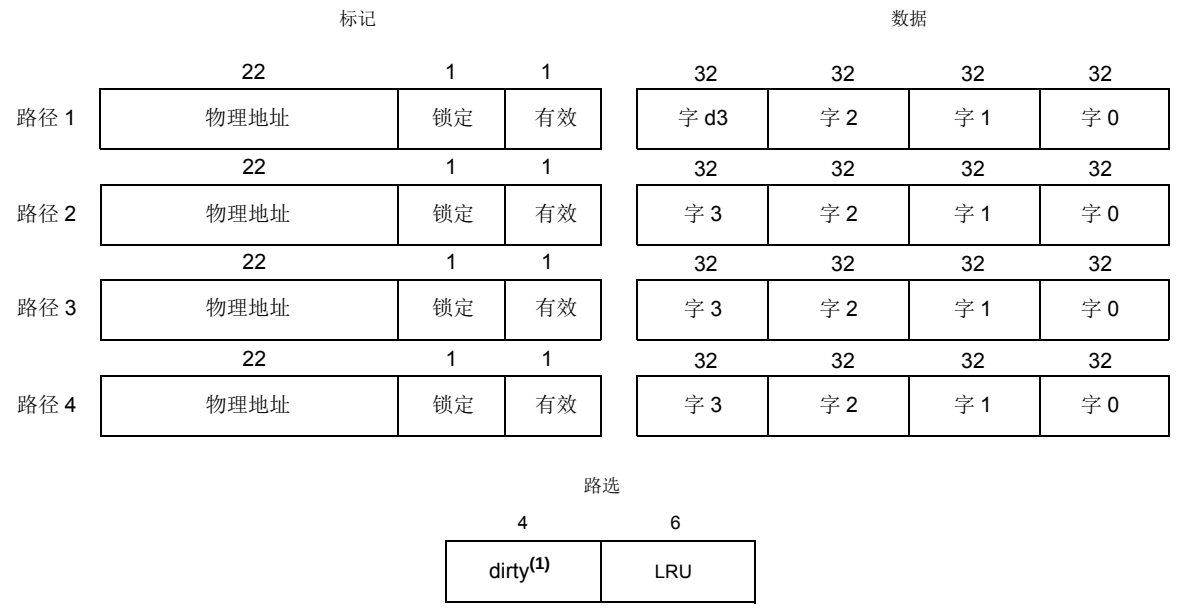
50.20.2 高速缓存构成

每个高速缓存行分成 3 个阵列：

- 标记
- 数据
- 路选

标记阵列保存高速缓存存储单元的物理地址，而数据阵列保存高速缓存的 （16 字节）的指令或数据。标记和数据阵列保存每组的四行信息，对应 4 路组相联的高速缓存。路选阵列保存最近最少使用 （Least Recently Used， LRU）位，对这些位译码以根据 LRU 算法选择要替换的路径。在数据高速缓存中，该阵列还保存 dirty 位，该位指示替换高速缓存中的数据前是否需要将数据回写到存储器。

图 50-34： 高速缓存阵列格式



注 1： 仅用于数据高速缓存。

图 50-34 的定义如下：

- **物理地址：**物理地址的高 22 位 （bit <31:10>）
- **锁定：**使用 CACHE 指令将该位置 1 或清零。置 1 时，高速缓存行被 “锁定”（即，在高速缓存未命中时无法选择该行作为替换）。
- **有效：**指示高速缓存行中的数据是否有效
- **字 0-3：**存储器的指令或数据字。每个高速缓存行存储四个 32 位字。
- **写使能：**仅存在于数据高速缓存阵列中，每个路径有一个 dirty 位。当高速缓存行中的数据被修改时， dirty 位置 1。这可让 CPU 知道在高速缓存行在进行高速缓存填充而被替换前应将其回写到存储器。
- **LRU：**这些位指示根据最近最少使用 （LRU）算法要替换的路径

© 2013-2016 Microchip Technology Inc.

50.20.3 高速缓存功能属性

可使用 CP0 Config1 寄存器设置 kseg0 的高速缓存功能属性。高速缓存功能选项有：

- **非高速缓存：**非高速缓存存储区域中的地址从主存储器读取，而不是高速缓存。写入非高速缓存的存储区域将直接写入主存储器，而不改变高速缓存内容。
- **具有写分配的回写：**装载和取指首先搜索高速缓存，仅当所需的数据未位于高速缓存中时才读取主存储器。执行数据存储操作时，首先搜索高速缓存以查看目标地址是否位于高速缓存中。如果是，将更新高速缓存内容，但不写入主存储器。如果高速缓存查找在存储时未命中，将读主存储器以将行写入高速缓存并将其与新存储数据合并。因此，高速缓存未命中时的分配策略是读分配和写分配。数据存储将更新路选阵列中相应的 dirty 位以指示行包含修改数据。当包含 dirty 数据的行在高速缓存中被替换时，将被回写到存储器。
- **不具有写分配的直写：**装载和取指首先搜索高速缓存，仅当所需的数据未位于高速缓存中时才读取主存储器。执行数据存储操作时，首先搜索高速缓存以查看目标地址是否位于高速缓存中。如果是，将更新高速缓存内容，同时还写入主存储器。如果在存储时高速缓存查找未命中，则仅写入主存储器。因此，高速缓存未命中时的分配策略仅是读分配。
- **具有写分配的直写：**装载和取指首先搜索高速缓存，仅当所需的数据未位于高速缓存中时才读取主存储器。执行数据存储操作时，首先搜索高速缓存以查看目标地址是否位于高速缓存中。如果是，将更新高速缓存内容，同时还写入主存储器。如果高速缓存查找在存储时未命中，将读主存储器以将行写入高速缓存并将其与新存储数据合并。此外，存储数据还写入主存储器。因此，高速缓存未命中时的分配策略是读分配和写分配。

50.20.4 高速缓存替换策略

高速缓存替换策略是指在高速缓存未命中时高速缓存如何确定填充组中的哪个路径。在通常的高速缓存未命中情况下，锁定和 LRU 标记位用于确定被填充的路径。如果所有路径均有效，将不会替换任何锁定路径。如果所有路径均锁定，填充数据将不会填充到高速缓存，且回写存储变为直写或写分配存储。如果要替换的路径为 dirty， 16 字节行在被填充之前将回写到存储器。

路选阵列中的 LRU 域在以下几个方面进行更新：

- 高速缓存命中时，相关路径更新为最近最多使用
- 高速缓存填充时，填充路径更新为最近最多使用
- 执行 CACHE 指令时， LRU 位的更新取决于以下操作：
  - 索引 （回写）无效：最近最少使用
  - 索引装载标记：不更新
  - 索引存储标记 （CP0 ErrCtl<WST> = 0）：CP0 TagLo<V> = 1 时为最近最多使用；CP0 TagLo<V> = 0 时为最近最少使用
  - 索引存储标记 （CP0 ErrCtl<WST> = 1）：更新 CP0 TagLo<LRU> 的内容
  - 索引存储数据：不更新
  - 命中无效：如果产生命中，则为最近最少使用；否则不更新
  - 填充：最近最多使用
  - 命中回写：不更新
  - 取指和锁定：最近最多使用

50.20.5 高速缓存指令

标记、数据和路选阵列的内容由用户使用 CACHE 指令进行控制。用户可通过置 1 或清零标记和数据阵列中的位来填充和锁定各个高速缓存行及使其无效。关于如何使用 CACHE 指令的详细信息，请参见 **50.21.7 “MPU Cache 指令”**。

50.20.6 高速缓存一致性

由于高速缓存中存有存储器数据的副本，因此其他总线主器件可修改高速缓存存储单元，以表示高速缓存的数据过期。同样，CPU 可更新高速缓存内容，以表示相应的存储器数据过期，直至回写相应的存储器数据。PIC32 CPU 没有硬件支持来维持高速缓存和存储器的一致性，因此必须由软件处理。大多数操作系统自动管理高速缓存一致性问题。不依靠操作系统运行的程序必须自己管理高速缓存一致性。

在直写模式下，所有数据写操作都写入主存储器。在回写模式下，数据仅写入高速缓存；高速缓存可能仅包含数据的有效副本，直到其被行重新填充或 CACHE 指令写入主存储器。

50.20.7 高速缓存初始化

PIC32 L1 高速缓存标记和数据阵列上电后处于未知状态且不受复位影响。因此，在使用高速缓存前必须先将其初始化。这通常通过引导代码将全零值写入标记阵列来完成。请注意，引导代码从非高速缓存存储器 （kseg1）运行。

© 2013-2016 Microchip Technology Inc.

50.21 CPU 指令

CPU 指令分为以下功能组：

- 装载和存储
- 计算
- 跳转和转移
- 其他
- 协处理器

每条指令的长度为 32 位。

50.21.1 CPU 装载和存储指令

MIPS 处理器使用装载 / 存储架构；所有操作均针对保存在处理器寄存器中的操作数执行，而主存储器仅通过装载和存储指令进行访问。

50.21.1.1 装载和存储的类型

有几种不同类型的装载和存储指令，分别针对不同用途而设计：

- 传送各种大小的位域 （例如， LB 和 SW）
- 将传送的数据交换为有符号或无符号整数 （例如， LHU）
- 访问未对齐位域 （例如， LWR 和 SWL）
- 原子级存储器更新 （读 - 修改 - 写：例如， LL/SC）

50.21.1.2 CPU 装载和存储指令的列表

CPU 装载和存储指令可以传送以下数据大小 （在 AccessLength 字段中定义）：

- 字节
- 半字
- 字

装载指令支持不同大小的有符号和无符号整数，它们可以对装载到寄存器中的数据进行符号扩展或零扩展。

未对齐字和双字可以通过使用一对特殊指令只在两条指令中进行装载或存储。对于装载，一条 LWL 指令与一条 LWR 指令配对。装载指令从对齐字处读取左字节或右字节 （寄存器的左侧或右侧），并将它们合并为对应于目标寄存器的正确字节。

50.21.1.3 用于原子级更新的装载和存储

配对指令 （链接装载和条件存储）可以用于对字或双字高速缓存的存储单元执行原子级读 - 修改 - 写操作。这些指令在经过仔细编码的序列中使用，用以提供多种同步原语中的一种，包括测试并设置 （test-and-set）、位级别锁定、信号量，以及排序器和事件计数。

50.21.1.4 协处理器装载和存储

如果特定协处理器未使能，则无法执行对该处理器的装载和存储操作，尝试执行的装载或存储操作会导致协处理器不可用异常。使能协处理器是一种特权操作，通过系统控制协处理器 CP0 提供。

50.21.2 计算指令

对于以二进制补码表示的整数，将执行二进制补码算法。以下是有符号形式的运算：

- 加法
- 减法
- 乘法
- 除法

标为“无符号”的加法和减法运算实际上是不带溢出检测的取模运算。

此外，还有无符号形式的乘法和除法运算，以及全部的移位和逻辑运算。逻辑运算对于寄存器宽度不敏感。

MIPS32 提供了 32 位整数和 32 位算术运算。

50.21.2.1 移位指令

ISA 定义了两种类型的移位指令：

- 从指令字中的 5 位域获取固定移位量的指令 （例如， SLL 和 SRL）
- 从通用寄存器的低位获取移位量的指令 （例如， SRAV 和 SRLV）

50.21.2.2 乘法和除法指令

乘法指令执行 32 位 x 32 位乘法，产生 64 位或 32 位结果。除法指令将 64 位值除以 32 位值，产生 32 位结果。除一条指令外，其他所有指令均将它们的结果传到 HI 和 LO 特殊寄存器中。MUL 指令会将结果的低半部分直接传到 GPR。

- 乘法会产生宽度为输入操作数两倍的全宽度乘积；低半部分装入 LO，高半部分装入 HI。
- 乘加和乘减产生宽度为输入操作数两倍的全宽度乘积，并将 HI 和 LO 的相连值加上或减去乘积。加法结果的低半部分装入 LO，高半部分装入 HI。
- 除法会产生一个商 （装入 LO）和一个余数 （装入 HI）。

结果通过在 HI/LO 和通用寄存器之间传送数据的指令进行访问。

50.21.3 跳转和转移指令

50.21.3.1 ISA 定义的跳转和转移指令的类型

本架构中定义了以下跳转和转移指令：

- PC 相对条件转移
- PC 区域无条件跳转
- 绝对 （寄存器）无条件跳转
- 在通用寄存器中记录返回链接地址的一组过程调用。

50.21.3.2 转移延时和转移延时时隙

所有转移都具有一条指令的架构性延时。紧接在转移指令之后的指令处于转移延时时隙中。如果转移或跳转指令位于转移延时时隙中，则这两种指令的操作都是未定义的。

根据约定，如果发生异常或中断，导致无法完成转移延时时隙中的指令，则会通过重新执行转移指令来继续指令流。为了允许这一点，转移必须可重新启动；过程调用不能使用存储返回链接的寄存器 （通常为 GPR 31）来确定转移目标地址。

© 2013-2016 Microchip Technology Inc.

50.21.3.3 转移和可能转移

存在两种形式的条件转移；它们的区别在于未发生转移而向下执行时，对于延时时隙中指令的处理方式。

- 转移指令会执行延时时隙中的指令
- 如果未发生转移，可能转移指令不会执行延时时隙中的指令 （称作废弃延时时隙中的指令）

注：	虽然本规范中包含了可能转移指令，但强烈建议避免在软件中使用可能转移指令，因为 MIPS 架构的未来版本中将会删除它们。

50.21.4 microMIPS 指令

microMIPS ISA 引入了几个新指令，能够从存储器装载多个 16 位或 32 位字或将多个 16 位或 32 位字存储到存储器。关于 microMIPS ISA 的完整列表和说明，请参见 *“MIPS® Architecture for Programmers Volume II-B: The microMIPS32™ Instruction Set”* —— MD00582。该文档可从 Imagination Technologies Ltd. 网站：http://www.imgtec.com/mips/architectures/mips32.asp 获取。

50.21.5 其他指令

50.21.5.1 指令序列化 （SYNC 和 SYNCI）

在正常操作中，本架构并未规定对于执行处理器之*外*的观察者，其存储器装载和存储访问的顺序是如何的 （例如，在多处理器系统中）。

SYNC 指令可用于在执行指令流中建立一个同步点，该同步点可以决定一些装载和存储操作的相对顺序：在SYNC之前执行的装载和存储操作完成之后，SYNC之后的装载和存储操作才能开始。

SYNCI 指令可以将处理器高速缓存与先前写操作或对于指令流的其他修改进行同步。

50.21.5.2 异常指令

异常指令会将控制转移给内核中的软件异常处理程序。存在两种类型的异常：条件和无条件异常。它们由以下指令产生：`syscall`、`trap` 和 `break`。

陷阱指令会根据比较结果产生条件异常。系统调用和断点指令会产生无条件异常。

50.21.5.3 条件传送指令

MIPS32 中包含了一些可以根据第三个通用寄存器值而条件性地将一个 CPU 通用寄存器的内容传送到另一个通用寄存器的指令。

50.21.5.4 NOP 指令

`NOP` 指令实际上编码为一条全零指令。 MIPS 处理器会将该编码特别处理为不执行任何操作，并优化指令的执行。此外，在任意处理器（包括架构的超标量实现形式）上，`SSNOP` 指令都占用一个指令发出周期。

50.21.5.5 TLB 指令 （仅限 MPU）

`TLBR`、`TLBWI` 和 `TLBWR` 指令可用于读或写 TLB 条目。 Index 寄存器装入 TLB 条目的 VPN，而标记内容读取到 EntryLo0、 EntryLo1 和 PageMask 寄存器中。使用 `TLBWI` 和 `TLBWR` 指令以类似方式写入 TLB 条目。

50.21.6 协处理器指令

50.21.6.1 协处理器的作用

协处理器是一些备用执行单元，具有独立于 CPU 的寄存器文件。简单来说，MIPS 架构最多提供 4 个协处理器单元，编号为 0 至 3。 ISA 的每一级都定义了一些协处理器。协处理器 0 总是用于系统控制，协处理器 1 和 3 用于浮点单元。协处理器 2 保留用于特定于实现的用途。PIC32 器件仅实现了协处理器 0。

协处理器可能具有两种不同的寄存器集：

- 协处理器通用寄存器
- 协处理器控制寄存器

每个寄存器集最多包含 32 个寄存器。协处理器计算指令可以使用任一寄存器集中的寄存器。

50.21.6.2 系统控制协处理器 0 （CP0）

所有 MIPS 处理器的系统控制器都实现为协处理器 0 （CP0），即系统控制协处理器。它提供了处理器控制、存储器管理和异常处理功能。

50.21.6.3 协处理器装载和存储指令

对于 CP0，未定义任何显式的装载和存储指令；必须使用协处理器的传送指令来读写 CP0 寄存器。

50.21.7 MPU Cache 指令

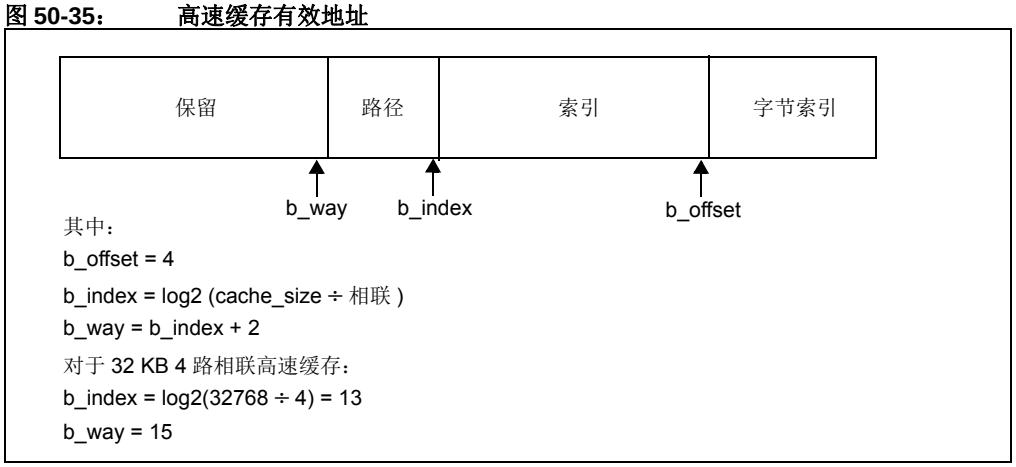
CACHE 指令可用于读或写数据、标记或路选阵列。它有两个参数：一个 5 位操作域和一个 16 位偏移量域。该格式与装载和存储指令类似，偏移量域由基址寄存器和 16 位有符号立即数偏移量组成。

```
cache OP, offset(base)
```

16 位偏移量进行符号扩展并与基址寄存器的内容相加构成有效地址。有效地址可为以下两种类型之一：

- **地址：**有效地址是一些指令或数据的虚拟存储单元，如同普通高速缓存访问一样进行处理。
- **索引：**有效地址包含高速缓存虚拟索引、高速缓存行中的字节位置以及路径。每个值的精确大小和边界取决于高速缓存大小和配置。通常，索引有效地址具有以下格式，如图 50-35 所示。

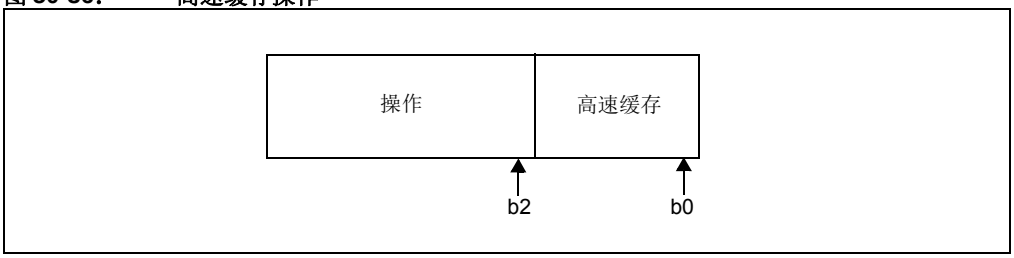
图 50-35： 高速缓存有效地址



© 2013-2016 Microchip Technology Inc.

操作域包含要在其上执行操作和其本身的高速缓存，如图 50-36 所示。

图 50-36： 高速缓存操作



操作域的低两位指定要对其执行操作的高速缓存，如表 50-43 所示。

表 50-43： 高速缓存指令 OP 位解码 （高速缓存）

OP<1:0> 位	高速缓存
‘0b00	指令高速缓存
‘0b01	数据高速缓存
‘0b10	保留
‘0b11	保留

操作域的其余位指定要执行的操作，如表 50-44 所示。

表 50-44： 高速缓存指令 OP 位解码

OP<4:2> 位	高速缓存	名称	类型	操作
‘0b000	指令	索引无效	索引	将指定地址处的高速缓存块的状态设置为无效。通常用于在启动时初始化指令高速缓存。
	数据			将指定地址处的高速缓存块的状态设置为无效。如果无效且 dirty，首先回写存储器。不应用于在启动时初始化数据高速缓存。
‘0b001	—	—	—	保留。
‘0b010	两者	索引存储标记	索引	使用 TagLo 寄存器值设置高速缓存标记。通常用于在启动时初始化数据高速缓存，方法是将 TagLo 设置为零。
‘0b011	—	—	—	保留。
‘0b100	两者	命中无效	地址	如果高速缓存块包含指定地址，则设置为无效。dirty 时不回写。
‘0b101	指令	填充	地址	从指定地址填充高速缓存。类似于高速缓存未命中。
	数据			如果高速缓存块包含指定地址，则设置为无效。dirty 时写回。是使运行中高速缓存中的高速缓存行无效的建议方法。
‘0b110	—	—	—	保留。
‘0b111	—	—	—	保留。

CACHE 索引装载标记和索引存储标记指令可用于通过将 ErrCtl CP0 寄存器中的 WST 位置 1 读 / 写指令或数据高速缓存路选阵列中的 LRU 位。表 50-45 给出了有效 LRU 域值。

表 50-45： LRU 位路选解码

选择顺序	LRU<5:0> 位	选择顺序	LRU<5:0> 位	选择顺序	LRU<5:0> 位	选择顺序	LRU<5:0> 位
0123	000000	1023	000100	2013	100010	3012	011001
0132	000001	1032	000101	2031	110010	3021	011011
0213	000010	1203	100100	2103	100110	3102	011101
0231	010010	1230	101100	2130	101110	3120	111101
0312	010001	1302	001101	2301	111010	3201	111011
0321	010011	1320	101101	2310	111110	3210	111111

该顺序通过将最近最少使用的路径列在左边而最近最多使用的路径列在右边来指示。请注意，并非所有值均有效。

© 2013-2016 Microchip Technology Inc.

50.22 MIPS DSP ASE 指令

> **注：** DSP ASE 并非在所有器件上都可用。要确定可用性，请参见具体器件数据手册中的 **“CPU”** 章节。

MIPS DSP ASE 版本 2 指令可分类为以下子类：

- 算术
- 基于 GPR 的移位
- 乘法
- 位操作
- 比较选择
- DSP 控制访问
- 索引装载
- 转移

MIPS DSP ASE 新增了四个寄存器。需要软件来识别 MIPS DSP ASE 是否存在以及在现场保护和恢复操作中包含这些附加寄存器。

3 个附加的 HI/LO 寄存器可用于创建共 4 个累加寄存器。许多常用 DSP 计算涉及到累加（例如，卷积）。针对累加器的 MIPS DSP ASE 版本 2 指令使用两位来指定目标累加器，零值表示 MIPS 架构的原始累加器。

DSPControl 为新的控制寄存器，用于保存有效支持新指令所需的额外状态位。寄存器 50-63 描述了该寄存器中的位。

50.22.1 DSPControl 寄存器

DSPControl 寄存器用于保存有效支持 MIPS DSP ASE 版本 2 指令集所需的额外状态位。

> **注：** 该寄存器并非在所有器件上都可用。要确定可用性，请参见具体器件数据手册中的 **“CPU”** 章节。

寄存器 50-63： DSPControl：MIPS DSP ASE 控制寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	—	CCOND<3:0>			
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	OUFLAG<7:0>							
15:8	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	EFI	C	SCOUNT<5:1>				
7:0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	SCOUNT<0>	—	POS<5:0>					

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 31-28 **未实现：** 读为 0

bit 27-24 **CCOND<3:0>：** 条件代码位

这些位由向量比较指令设置，且被 PICK 指令组用作源选择器。向量元素大小决定被比较置 1 的位数 （1、2 或 4）。比较后未置 1 的位是不可预测的。

寄存器 50-63： DSPControl：MIPS DSP ASE 控制寄存器 （续）

bit 23-16 **OUFLAG<7:0>：**上溢 / 下溢指示位

溢出标志位 OUFLAG<7:0> 由下表中列出的指令置 1。这些位保持不变且仅在显式写入该寄存器中的这些位 （使用 WRDSP 指令）时复位。

位号	说明
23	该位在目标为累加器 （HI-LO 对） 0 时置 1，且发生上溢或下溢操作。这些指令是：DPAQ_S、 DPAQ_SA、 DPSQ_S、 DPSQ_SA、 DPAQX_S、 DPAQX_SA、 DPSQX_S、 DPSQX_SA、 MAQ_S、 MAQ_SA 和 MULSAQ_S。
23	当目标为累加器 （HI-LO 对） 1 时置 1，指令同上。
21	当目标为累加器 （HI-LO 对） 2 时置 1，指令同上。
20	当目标为累加器 （HI-LO 对） 3 时置 1，指令同上。
19	在发生上溢 / 下溢时置 1 该位的指令：ABSQ_S、 ADDQ、 ADDQ_S、 ADDU、 ADDU_S、 ADDWC、 SUBQ、 SUBQ_S、 SUBU 和 SUBU_S。
18	在发生上溢 / 下溢时置 1 该位的指令：MUL、MUL_S、MULEQ_S、MULEU_S、MULQ_RS 和 MULQ_S。
17	在发生上溢 / 下溢时置 1 该位的指令：PRECRQ_RS、 SHLL、 SHLL_S、 SHLLV 和 SHLLV_S。
16	在发生上溢 / 下溢时置 1 该位的指令：EXTR、 EXTR_S、 EXTR_RS、 EXTRV 和 EXTRV_RS。

bit 15 **未实现：**读为 0

bit 14 **EFI：**抽取失败指示符位

当其中一个抽取指令 （EXTP、 EXTPV、 EXTPDP 或 EXTPDPV）失败时，该位设置为 1。当要抽取的位不够时 （即 POS<5:0> 位的值小于指令中指定的大小参数时），将发生故障。

bit 13 **C：**进位位

该位由特殊的加法指令置 1 和使用，以在 microMIPS32 实现中的两个 GPR 之间实现 64 位加法。指令 ADDSC 置 1 该位，而指令 ADDWC 使用该位。

bit 12-7 **SCOUNT<5:0>：**大小计数位

该位由 INSV 指令用来指定要插入的位域的大小。

bit 6 **未实现：**读为 0

bit 5-0 **POS<5:0>：**插入 / 抽取位置位

这些位由可变插入指令 INSV 使用，指定插入位的位置。它还用于指示 EXTP、 EXTPV、 EXTPDP 或 EXTPDPV 指令的抽取位置。

© 2013-2016 Microchip Technology Inc.

50.23 CPU 初始化

发生复位事件之后，软件需要初始化器件的以下部分：

50.23.1 通用寄存器

CPU 寄存器文件在上电时处于未知状态； r0 除外，它总是为 0。为了让硬件正确工作，并不需要初始化其他寄存器文件。但是根据软件环境，可能需要初始化几个寄存器。其中有：

- sp—— 堆栈指针
- gp—— 全局指针
- fp—— 帧指针

50.23.2 协处理器 0 状态

在退出引导代码之前，还需要初始化一些 CP0 状态。有一些异常会被 ERL = 1 或 EXL = 1 阻止，但复位时并不会清除这些异常。在退出引导代码时，可以将它们清除，以避免捕捉到虚假的异常。

表 50-46： CPU 初始化

CP0 寄存器	操作
Cause	WP （监视待处理）、 SW0/1 （软件中断）应清除。
Config	通常， K0、 KU 和 K23 位域应先设置为所需的高速缓存一致性算法（Cache Coherency Algorithm， CCA）值，然后再访问相应的存储区。
Count(1)	如果使用定时器中断，则应设置为已知值。
Compare(1)	如果使用定时器中断，则应设置为已知值。写入 Compare 寄存器也会清除所有待处理的定时器中断 （所以，应先设置 Count 寄存器，再设置 Compare 寄存器，以避免任何意外中断）。
Status	应设置所需的器件状态。
其他 CP0 状态	其他寄存器应先写入后再读取。一些寄存器并不是显式可写，它们的更新只是执行指令或捕捉异常的副产品。未初始化的位应在读取这些寄存器之后通过掩码操作去除。

注 1： 当 Count 寄存器等于 Compare 寄存器时，将会发出定时器中断。中断控制器中存在一个用于禁止向 CPU 传递该中断的屏蔽位 （如果需要）。

50.23.3 系统总线

在切换为用户模式之前或从 DRM 中执行之前，应先初始化系统总线。写入系统总线的值取决于要运行的应用程序的存储器布局。

50.23.4 高速缓存 （仅限 MPU）

高速缓存标记和数据阵列在上电时处于未知状态，且不受复位影响。应使用 CACHE 指令 （索引无效函数）将高速缓存阵列中的每个标记初始化为无效状态。

50.24 复位的影响

50.24.1 主复位

硬件复位不会完全初始化 PIC32 内核。只有很少的一部分处理器状态会被清零。在非映射和非高速缓存代码空间中运行时，这已足以启动内核。软件可以在此后初始化所有其他处理器状态。上电复位会将器件置为一种已知状态。软复位可以通过将主复位（MCLR）引脚置为有效来强制产生。实现该特性是为了与其他 MIPS 处理器保持兼容。在实际中，两种复位的处理方式是相同的。

50.24.1.1 协处理器 0 状态

许多硬件初始化都在协处理器 0 中进行，如表 50-47 所述。

表 50-47： 复位清零或置 1 的位

寄存器名称	位名称	清零或置 1	值	清零或置 1 的原因
Status	BEV	置 1	1	复位或软复位
	TS	清零	0	复位或软复位
	SR	置 1	1	软复位
	SR	清零	0	复位
	NMI	清零	0	复位或软复位
	ERL	置 1	1	复位或软复位
	RP	清零	0	复位或软复位
所有配置寄存器： Config Config1 Config2 Config3 Config4 Config5 Config7	**注：** CP0 配置寄存器的复位状态因器件而异。详情请参见具体器件数据手册的 **“CPU”** 章节。			
Debug	DM	清零	0	复位或软复位 (1)
	LSNM	清零	0	复位或软复位
	IBUSEP	清零	0	复位或软复位
	IEXI	清零	0	复位或软复位
	SSt	清零	0	复位或软复位

注 1： 除非使用 EJTAGBOOT 选项来引导进入调试模式。

50.24.1.2 总线状态机

当捕捉到复位或软复位异常时，所有待处理总线事务都会被中止，并且 SRAM 接口单元中的状态机会复位。

50.24.2 取指地址

在复位 / 软复位时，除非使用了 EJTAGBOOT 选项，否则取指操作会指向 VA 0xBFC00000（PA 0x1FC00000）。该地址位于 kseg1 中，它是非映射和非高速缓存的。

50.24.3 WDT 复位

在 WDT 事件之后， CPU 寄存器的状态取决于发生 WDT 事件之前 CPU 的操作模式。

如果器件先前不是处于休眠模式， WDT 事件会将寄存器强制设为复位值。

© 2013-2016 Microchip Technology Inc.

50.25 相关应用笔记

本节列出了与手册本章内容相关的应用笔记。这些应用笔记可能并不是专为 PIC32 器件系列而编写的，但其概念是相近的，通过适当修改即可使用，但在使用时会受到一定限制。当前与采用 MIPS32® microAptiv™ 和 M-Class 内核的器件的 PIC32 CPU 相关的应用笔记包括：

标题	应用笔记编号
目前没有相关的应用笔记。	N/A

注： 如需获取更多 PIC32 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

50.26 版本历史

版本 A （2013 年 4 月）

这是本文档的初始版本。

版本 B （2015 年 7 月）

该版本包含以下更新：

- 文档标题改名为：“采用 MIPS32® microAptiv™ 和 M-Class 内核的器件的 CPU”
- 在全文中更新了与 M-Class 内核的相关内容
- 将所有对 MIPS Technologies 的引用修改为 Imagination Technologies Ltd.
- 将 Debug2 寄存器的选择编号从 5 改为 6 （见寄存器 50-45）
- 新增以下寄存器：
  - BadInstr：无效指令字寄存器； CP0 寄存器 8，选择 1 （寄存器 50-11）
  - BadInstrP：无效先前转移指令寄存器； CP0 寄存器 8，选择 2 （寄存器 50-12）
  - CacheErr：高速缓存错误寄存器； CP0 寄存器 27，选择 0 （寄存器 50-51）
  - KScratchn：内核模式中间结果暂存寄存器； CP0 寄存器 31，选择 2-3 （寄存器 50-57）
- 新增 **50.5.2 “版本 5 架构”**
- 新增 **50.12 “浮点单元 （FPU）”**
- 新增 **50.14 “协处理器 1 （CP1）寄存器”**，其中包括以下寄存器：
  - FIR：浮点实现寄存器； CP1 控制寄存器 0 （寄存器 50-58）
  - FCC：浮点条件代码寄存器； CP1 控制寄存器 25 （寄存器 50-59）
  - FEXR：浮点异常寄存器； CP1 控制寄存器 26 （寄存器 50-60）
  - FENR：浮点使能寄存器； CP1 控制寄存器 28 （寄存器 50-61）
  - FCSR：浮点控制和状态寄存器； CP1 控制寄存器 31 （寄存器 50-62）
- 对全文的内容和格式进行了少量更新

© 2013-2016 Microchip Technology Inc.

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了 《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc. 及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc. 的英文原版文档。

本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。**Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。** Microchip 对因这些信息及使用这些信息而引起的后果不承担任何责任。如果将 Microchip 器件用于生命维持和 / 或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切伤害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任，并加以赔偿。除非另外声明，在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2009 认证。Microchip 的 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品严格遵守公司的质量体系流程。此外，Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949═

商标

Microchip 的名称和徽标组合、 Microchip 徽标、 AnyRate、dsPIC、FlashFlex、flexPWR、Heldo、JukeBlox、KeeLoq、KeeLoq 徽标、 Kleer、 LANCheck、 LINK MD、 MediaLB、MOST、MOST 徽标、MPLAB、OptoLyzer、PIC、PICSTART、PIC32 徽标、 RightTouch、 SpyNIC、 SST、 SST 徽标、SuperFlash 及 UNI/O 均为 Microchip Technology Inc. 在美国和其他国家或地区的注册商标。

ClockWorks、 The Embedded Control Solutions Company、ETHERSYNCH、 Hyper Speed Control、 HyperLight Load、IntelliMOS、 mTouch、 Precision Edge 和 QUIET-WIRE 均为 Microchip Technology Inc. 在美国的注册商标。

Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、BodyCom、chipKIT、chipKIT 徽标、CodeGuard、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、EtherGREEN、 In-Circuit Serial Programming、 ICSP、 Inter-Chip Connectivity、 JitterBlocker、 KleerNet、 KleerNet 徽标、MiWi、motorBench、MPASM、MPF、MPLAB Certified 徽标、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、 PICDEM、 PICDEM.net、 PICkit、 PICtail、PureSilicon、 RightTouch 徽标、 REAL ICE、 Ripple Blocker、Serial Quad I/O、SQI、SuperSwitcher、SuperSwitcher II、Total Endurance、TSHARC、USBCheck、VariSense、ViewSpan、WiperLock、Wireless DNA 和 ZENA 均为 Microchip Technology Inc. 在美国和其他国家或地区的商标。

SQTP 为 Microchip Technology Inc. 在美国的服务标记。

Silicon Storage Technology 为 Microchip Technology Inc. 在除美国外的国家或地区的注册商标。

GestIC 为 Microchip Technology Inc. 的子公司 Microchip Technology Germany II GmbH & Co. & KG 在除美国外的国家或地区的注册商标。

在此提及的所有其他商标均为各持有公司所有。

© 2013-2016, Microchip Technology Inc. 版权所有。

ISBN：978-1-5224-0633-4

全球销售及服务网点

美洲

公司总部 Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://www.microchip.com/support
网址：www.microchip.com

亚特兰大 Atlanta
Duluth, GA
Tel: 1-678-957-9614
Fax: 1-678-957-1455

奥斯汀 Austin, TX
Tel: 1-512-257-3370

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

克里夫兰 Cleveland
Independence, OH
Tel: 1-216-447-0464
Fax: 1-216-447-0643

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Novi, MI
Tel: 1-248-848-4000

休斯敦 Houston, TX
Tel: 1-281-894-5983

印第安纳波利斯 Indianapolis
Noblesville, IN
Tel: 1-317-773-8323
Fax: 1-317-773-5453

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

纽约 New York, NY
Tel: 1-631-435-6000

圣何塞 San Jose, CA
Tel: 1-408-735-9110

加拿大多伦多 Toronto
Tel: 905-695-1980
Fax: 905-695-2078

亚太地区

亚太总部 Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

中国 - 北京
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

中国 - 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 - 重庆
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

中国 - 东莞
Tel: 86-769-8702-9880

中国 - 广州
Tel: 86-20-8755-8029

中国 - 杭州
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

中国 - 南京
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

中国 - 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 - 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 - 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 - 深圳
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

中国 - 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 - 西安
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

中国 - 厦门
Tel: 86-592-238-8138
Fax: 86-592-238-8130

中国 - 香港特别行政区
Tel: 852-2943-5100
Fax: 852-2401-3431

亚太地区

中国 - 珠海
Tel: 86-756-321-0040
Fax: 86-756-321-0049

台湾地区 - 高雄
Tel: 886-7-213-7828

台湾地区 - 台北
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

台湾地区 - 新竹
Tel: 886-3-5778-366
Fax: 886-3-5770-955

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

印度 India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

印度 India - Pune
Tel: 91-20-3019-1500

日本 Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

日本 Japan - Tokyo
Tel: 81-3-6880-3770
Fax: 81-3-6880-3771

韩国 Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

韩国 Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

菲律宾 Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

新加坡 Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

泰国 Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

欧洲

奥地利 Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

丹麦 Denmark-Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

法国 France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 Germany - Dusseldorf
Tel: 49-2129-3766400

德国 Germany - Karlsruhe
Tel: 49-721-625370

德国 Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

意大利 Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

意大利 Italy - Venice
Tel: 39-049-7625286

荷兰 Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

波兰 Poland - Warsaw
Tel: 48-22-3325737

西班牙 Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

瑞典 Sweden - Stockholm
Tel: 46-8-5090-4654

英国 UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

06/23/16

© 2013-2016 Microchip Technology Inc.