移植到 PIC32MM 单片机系列

简介

本文档介绍了 PIC32MM 器件架构的许多特性，以便于从其他器件进行移植。本文档假定读者了解自己当前正在使用的器件，因而对 PIC32MM 系列作了高度概要的介绍，而不是全面比较各种产品之间的外设差异。本文档旨在补充而不是取代系列参考手册和器件数据手册。

CPU

CPU 是 32 位的 MIPS® microAptiv™ 单相统一哈佛 RISC 架构内核，具有可在每个系统时钟完成一条指令的 5 级流水线。完成每条指令本需要 5 个时钟，但由于采用流水线， 5 条指令处于流水线的不同级，它们会同时在流水线中前进，从而使每个时钟可以完成一条指令。内核使用装载和存储架构。这意味着不是在存储器中对数据进行操作。实际上，数据先从存储器读入内核寄存器，进行操作，然后写回存储器。

实现了 MIPS MCU ASE 扩展。该扩展可缩短中断延时，并增加了原子性读和写指令。未实现 DSP ASE 扩展和浮点协处理器。

CPU 寄存器和零寄存器

CPU 没有专用的累加器寄存器；它包含 32 个 32 位的通用寄存器 （General Purpose Register， GPR），称为内核寄存器。零寄存器除外，所有其他寄存器的功能都是相同的。寄存器编号 0 （$0）被称为零寄存器。读取零寄存器将总是返回 0。对零寄存器的写操作不会更改零寄存器的内容。一些指令（如转移和链接）只能对特定的非用户可选寄存器进行操作。

根据约定，寄存器会被分配为包含值，如函数的返回值或堆栈指针。

程序计数器

MIPS CPU 没有可直接访问的程序计数器 （Program Counter， PC）。目标地址或偏移量被装入 GPR 寄存器， JUMP （J）或 CALL （JAL）指令使用寄存器的内容和当前 PC 值来生成目标地址。

CPU 配置

协处理器 0（CP0）寄存器用于配置 CPU、中断模式和类似操作。 CP0 寄存器还包含关于 CPU 实现的信息，如支持的指令集和实现的硬件乘法器的类型。这些寄存器通过在通用寄存器和CP0寄存器之间传送数据的特殊指令 MFC0 和 MTC0 进行访问。

数据宽度

PIC32MM 器件固有的数据宽度为 32 位。数学运算在 32 位字上执行，字节和半字运算通过对 32 位运算的结果进行掩码操作来执行。由于最大指令长度和数据宽度是相同的，所以装入立即数指令无法包含所有可能的 2^32 位值。因此，向寄存器装入立即数值可能需要多条指令来 “构建” 所需的值 （见 **“汇编语言”**）。 MPLAB® XC32 编译器支持 char （8 位）、 short （16 位）、 int （32 位）和 long long （64 位）数据类型。

指令集、指令宽度和正交性

仅支持 microMIPS™ 指令集。它是一种混合的双长度（16 位和 32 位）指令集，支持所有 MIPS32® 汇编助记符，但与 MIPS32 或 MIPS16e® 指令集的二进制不兼容。 microMIPS 指令集通过提供 16 位版本的常用指令来缩减代码大小。16 位版本不支持所有寄存器，并且偏移量和值范围缩小。在可能时，编译器会使用 16 位指令；否则，使用 32 位指令。指令长度会作为取操作的一部分而解码，因此 16 位和 32 位指令可以相邻，不需要像 MIPS16e 实现中一样使用调用来切换模式。MPLAB XC32 编译器会自动选择对应于该产品系列的 microMIPS 指令集。不支持 MIPS DSP ASE。

指令集是正交的。除了几个指令，所有其他指令都可以使用任意 CPU 寄存器作为源寄存器，使用任意寄存器来保存结果，包括零寄存器。

汇编语言

microMIPS 汇编语言包含了一些指令和多个伪指令来简化常见操作的编码。这些伪指令会基于操作和操作数扩展为单条或多条指令（见例 1）。

例 1：　　装入立即数伪指令的示例

```
LI $1, 0x12345678
可以由汇编器以两条 32 位指令实现：
LUI $2, 0x1234       # 0 | (0x1234<<16) -> $2
ORI $2, $0, 0x5678   #($2) | 0 | 0x5678 -> $2

当同一个宏的操作数较小时：
LI $2, 0x12
可以由汇编器以单条 16 位指令实现：
LI16 $2, 0x12        # 0x12 -> $2
```

空操作指令（NOP）

MIPS CPU 没有专用的 NOP 指令。由于具有零寄存器，有多个数学运算指令在执行后不产生任何可见效果（例如将某个值加到零寄存器），它们可以提供 NOP 功能。根据约定，SLL32 $0, $0, 0 用于 32 位 NOP（NOP32），而 MOVE16 $0, $0 用于 16 位 NOP（NOP16）。MPLAB XC32 遵循该约定，并在反汇编这些指令时显示 NOP。一些指令必须后跟 NOP32，而其他一些指令必须后跟 NOP16。

用户模式和内核模式

仅支持内核模式。在内核模式下，固件可直接访问外设。

位操作

CPU 和硬件支持原子性单位操作和多位操作。

原子性单位操作使用 ASET 和 ACLR 指令来支持。这些指令会禁止中断，执行多时钟周期的读 - 修改 - 写操作，然后重新允许中断。它们是多周期指令。由于具有原子性，它们可以用于实现信号量，但应仅用于访问 RAM 存储器，而不用于外设。

非原子性的单位操作和多位操作（如 T1CONbits.ON）会生成指令序列来执行读操作、逻辑或运算，并写回寄存器。为了在不使用读 - 修改 - 写序列的情况下以原子性操作的形式提供这种功能，大多数外设都支持硬件置 1/ 清零 / 取反寄存器，它们统称为 SCI 寄存器。这些寄存器以相对于外设的寄存器地址的偏移量（+0x4、+0x8、0xC）的形式进行寻址，并相应地命名为 xCLR、xSET 和 xINV（见例 2）。例如，可以通过 T1CONSET = (1<<15) 来将 Timer1 的 ON 位置 1。SCI 寄存器并不以寄存器形式存在，而是指示硬件使用所提供的掩码模式对所需寄存器执行操作。基于所使用的掩码值，可以同时操作多个位。SCI 寄存器是只写的；从其中读取的值是无效的。中断标志应仅使用 SCI 寄存器来清除。这是为了防止无意中清除在读 - 修改 - 写序列期间置为有效的中断。

例 2：　　LATA 的 SCI 地址偏移量和名称

```
0xBF802BE0 LATA
0xBF802BE4 LATACLR
0xBF802BE8 LATASET
0xBF802BEC LATAINV
```

移位操作和循环移位操作

移位操作和循环移位操作在 32 位上进行操作，可以对任意 CPU 寄存器执行，移位范围为 0-31 位。

乘法 / 除法单元（MDU）

CPU 具有自主的乘法 / 除法单元（Multiply and Divide Unit，MDU），可以在 CPU 执行代码时并行执行乘法或除法。乘法引擎可以在 1 个时钟周期内执行 32 位 x16 位 MAC，或在两个时钟周期内执行 32 位 x 32 位的有符号或无符号乘法。除法引擎可以在 11 至 34 个时钟周期内（取决于数据长度）执行 32 位 x 32 位的有符号或无符号除法。未实现硬件浮点；这些操作通过软件库进行仿真。

CPU 数学状态标志

MIPS CPU 未实现数学状态标志，如借位、进位或全零标志；需要使用比较指令来确定存储在寄存器中的结果是为零、大于零还是小于零（见例 3）。

例 3：　　测试结果是否等于零

```
# Sum contents of registers.($7 + $8) -> $6
ADDU $6, $7, $8
# Test and branch to EqualsZero:
BEQZ $6, EqualsZero
```

内核定时器

MIPS CPU 具有以 ½ 个 CPU 时钟速率进行计数的通用定时器（内核定时器）。内核定时器通过 CP0 寄存器进行访问 （见例 4 和例 5）。定时器可以配置为在达到指定值时产生中断。

例 4：　　设置 / 读取内核定时器

```
//Set Core Timer
unsigned int counts = 0x12345678;
asm volatile("mtc0   %0, $9": "+r"(counts));

//Read Core Timer
asm volatile("mfc0   %0, $9": "=r"(counts));
```

例 5：　　使用宏来设置 / 读取内核定时器

```
//Set Core Timer
unsigned int counts = 0x12345678;
_CP0_SET_COUNT(counts);

//Read Core Timer
counts = _CP0_GET_COUNT();
```

性能计数器

MIPS microAptiv CPU 支持两个性能计数器。这些计数器可以配置为对已完成的 CPU 指令、CPU 周期、已执行的转移指令以及其他度量进行计数，用于进行代码分析。性能计数器属于 CP0 寄存器的一部分，使用 MTC0 和 MFC0 指令或通过 MPLAB XC32 宏进行访问（见例 6）。

例 6：　　配置计数器

```
/ configure counter
#define  PCNTR0_MODE_NOPS (17<<5)
#define  PCNTR_CNT_KERNEL (1<<1)
_CP0_SET_PERFCNT0_CONTROL(PCNTR0_MODE_NOPS | PCNTR_CNT_KERNEL
);

_CP0_SET_PERFCNT0_COUNT(0);          // reset counts to 0
// user code
counts = _CP0_GET_PERFCNT0_COUNT();  // read counts
```

存储器和总线矩阵

等待状态

在整个器件工作频率范围内，闪存和 RAM 都进行零等待状态访问。未实现闪存和 RAM 等待状态控制位。

预取和高速缓存

未实现预取或高速缓存。行缓冲区用于在取操作之后存放一行闪存数据（64 位）。如果行缓冲区中存在要获取的下一条指令，则从行缓冲区中读取它，而不是产生另一个闪存取操作。

从 RAM 中执行

由于采用统一存储器映射，所以 RAM 存储器使用与闪存相同的方式进行映射。因此，可以从 RAM 中执行代码。从 RAM 中执行时，指令和数据总线必须通过仲裁获取对 RAM 存储器的访问权，这会产生内核停顿。要从 RAM 中执行代码，必须使用 EXECADDRx 位将存储器逻辑划分为指令和数据存储器 （见例 7）。默认分区方式为全为程序存储器。使用 __longramfunc__ 属性将某个函数放入 RAM 时，编译器会自动分配所需的存储器。

例 7： 手动设置 RAM 数据 / 代码分区

```
CFGCONbits.EXECADDR = 1; // 1k of RAM is
                         // dedicated to
                         // instruction memory
```

闪存和 ECC 纠错

闪存为 64 位宽，具有保留用于 ECC 纠错的附加位。闪存中的所有单位错误（Single Bit Error， SBE）都可以检测并纠正。所有双位错误（Double-Bit Error，DBE）都可以检测，但不能纠正，并会产生总线错误异常（数据装载）。由于具有 ECC 位，闪存数据必须以 64 位对齐、64 位双字或对齐行的形式写入。在未先执行擦除操作的情况下在闪存中重写数据会违反闪存规范，当读取数据时，如果数据不同于先前的数据，则会产生 ECC 错误。 ECC 错误仅在读操作期间检测。

统一存储器映射

CPU 具有 4 GB 的线性地址范围。存储器没有页或存储区选择位。统一存储器映射中包含处于公共地址范围内的闪存、RAM 和外设。该存储器映射被分为多个部分，称为段。一些段是其他段的别名，用于与支持高速缓存的器件保持兼容。

闪存编程

闪存的自编程可以通过双字写操作或行写操作执行。当闪存写操作正在进行时，CPU 或 DMA 的所有闪存访问都会暂停，直到闪存写操作完成为止。闪存控制器具有独立的硬件锁定机制（类似于 SYSKEY），用于防止意外操作。如果 CPU 从 RAM 中执行，则它可以在对闪存进行编程时继续执行。但是，如果执行行写操作，则 CPU 必须与闪存控制器通过仲裁来获取 RAM 访问权。

物理和虚拟存储器地址

物理存储器的地址范围为：0x0000_0000 至 0x7FFF_FFFF。该范围包含器件上所有可访问的存储器和外设的地址。物理地址仅由 DMA 外设 （包括闪存控制器、独立 DMA 和 USB DMA）使用。

CPU 通过虚拟寻址来访问存储器和外设。在虚拟寻址中，物理存储器映射的内容通过别名地址（称为段）进行访问。这些段可以具有基于 CPU 配置的唯一属性。FMT 可以将虚拟地址转换为物理地址。

存储器段 （ksegx）

物理存储器的内容可以通过多个地址 （称为段）访问。每个段可以具有唯一属性。虚拟地址的高 3 位用于确定段。 kuseg 地址与物理地址相同。该实现仅支持 kseg1 和 kseg0。 kseg1 是不可高速缓存的存储器， kseg0 是可高速缓存的存储器。由于未实现高速缓存，所以这些区域的功能是相同的。为了提高代码的可移植性，应遵循可高速缓存 / 不可高速缓存的约定。要同时供 DMA 和 CPU 访问的存储器应为不可高速缓存的 （kseg1），以防止一致性问题。外设应以不可高速缓存 （kseg1）的方式访问。只供 CPU 访问的存储器 （变量）应使用可高速缓存的存储器 （kseg0）。

固定映射转换 （FMT）表

FMT 用于将虚拟地址转换成物理地址，以用于 CPU 读操作和写操作。FMT 的操作对于用户是透明的。FMT 使用所提供的地址来确定所需操作的物理地址。

总线矩阵 （BMX）

总线矩阵是连接 CPU 指令总线、CPU 数据总线、外设和存储器的交叉开关。 CPU、闪存控制器、通用 DMA 和 USB DMA 被称为启动器，因为它们会启动读操作或写操作。闪存、RAM 和外设被称为目标，因为将向其中写入数据或从其中读取数据。所有外设通过单个目标访问。一些外设 （如 DMA）同时是目标和启动器。 BMX 允许多个启动器对目标执行并发访问，前提是它们不是同一目标。如果目标是相同的，则会发生仲裁。例如，DMA 可以将数据从外设传送到 RAM，同时 CPU 从闪存中取指令或数据。如果 DMA 和 CPU 尝试访问同一目标，则会发生仲裁，使 CPU 或 DMA 停顿。

中断

异常

异常是中断的通用术语。异常可以由外设中断、指令、地址错误和数学溢出产生。异常的原因包含在 Cause 寄存器 （CP0.13）中，异常的地址存储在异常返回寄存器 （EPC， CP0.14）中。

异常包括：

- 外设中断
- 地址错误异常 （装载或取指）
- 地址错误异常 （存储）
- 总线错误异常 （取指）
- 总线错误异常 （数据引用：装载或存储）
- SYSCALL 异常指令
- 软件断点异常
- 保留指令异常 （不解码为 NOP）
- 协处理器不可用异常 （仅支持 CP0）
- 算术溢出异常
- 陷阱异常 （根据 TRAP 指令的结果条件性地生成）

例 8： 异常处理程序

```
volatile unsigned int epc, cause;

void     __attribute__((naked)) _general_exception_handler(void)
{


    // determine the source of the issue
    epc = _CP0_GET_EPC();
    cause = (((_CP0_GET_CAUSE()) & 0x0000007C) >> 2);

    while(1);
}
```

中断

对于每个中断发生源，外设中断源都具有唯一的中断向量和中断请求 （Interrupt Request， IRQ）。每个 IRQ 具有用户可选的优先级和子优先级。支持多向量和单向量中断处理。

中断服务程序 （ISR）函数

编译器提供了一些属性来确定ISR函数的实现（例 9）。

例 9： 为 TIMER1 计满返回配置 ISR 的 ISR

```
unsigned int val = 0;

// Timer configuration is not shown

// set interrupt priority and enable peripheral interrupt
IPC1bits.T1IP = 7;                                  // set Timer1’s interrupt priority to 7
IPC1bits.T1IS = 0;                                  // set Timer1’s sub-priority to 0
IEC0bits.T1IE = 1;                                  // enable timer1 int

PRISS = 1<<28;                                      // pair IPL7 with Shadow Set 1

// enable of multi-vector mode is done by start-up code.The code below is not required.
asm      volatile("mfc0 %0,$13":"=r"(val));         // set the CP0 cause IV bit high
val |= 0x00800000;
asm      volatile("mtc0   %0,$13" : "+r"(val));
INTCONSET = _INTCON_MVEC_MASK;

__builtin_enable_interrupts();                      // XC32 macro to enable interrupts

void __ISR(_TIMER_1_VECTOR, IPL7SRS) TimerISR(void)  // ISR
{
IFS0CLR = (1<<17);                                  // clear the timer interrupt flag
// user code
}
```

中断优先级

每个中断源 IRQ 都具有相关的优先级、子优先级和固定的自然优先级。中断优先级用于确定中断抢占。优先级较高的中断将抢占优先级较低的中断。只有在有多个优先级相同的中断待处理时，才会使用子优先级。如果有多个优先级和子优先级相同的中断待处理，则使用自然顺序 （最低 IRQ 编号）来确定更高的优先级。有 8 个优先级和 4 个子优先级可用。IRQ 优先级和子优先级使用 IPCx 寄存器中的位来设置。假设没有会造成抢占的更高优先级中断在等待处理，则中断优先级不会影响中断延时。

允许中断

中断可以全局以及单独允许和禁止。各个中断允许位包含在 IECx 寄存器中。中断可以使用 `ei` 和 `di` 指令或通过使用编译器宏 `__builtin_enable_interrupts()` 和 `__builtin_disable_interrupts()` 全局允许和禁止。要使中断源可以产生中断，必须全局允许中断，中断的优先级必须大于零，并且必须允许相应的各个中断。

多向量、单向量中断和间距

在多向量模式下，每个 IRQ 具有用于中断处理程序的相应唯一向量地址。所有向量之间的距离，即中断服务程序 （ISR）最大长度可以由用户设置。用户可以指定要定位到向量地址处的 ISR 代码，也可以在向量地址处使用跳转目标为实际 ISR 代码的 JUMP 指令，从而减少向量表使用的空间。默认情况下，会在向量地址处创建跳转表，以减小向量表的大小。器件启动代码会使能多向量模式。

清除中断

应仅使用中断控制器的中断标志清除（IFSxCLR）寄存器来清除中断。对 IFSx 寄存器的软件读 - 修改 - 写操作会覆盖在读操作和写操作之间设置的任何中断标志，从而屏蔽中断。使用清除寄存器执行操作的速度也更快，并且使用的指令更少 （例 10）。

例 10：

```
IFS0CLR = 1<< _IFS0_CTIF_POSITION;
// Clear the Core Timer interrupt
```

影子集

有一个硬件影子集可用于中断处理。它是第二组CPU内核寄存器，在 ISR 声明和 PRISS 寄存器中使用参数与特定的中断优先级关联。当发生与影子集相关的中断时，影子集会替代内核寄存器，从而无需将内核寄存器的内容保存到堆栈中。对于不使用影子集的中断，会将内核寄存器的内容保存到堆栈中。通常情况下，影子集用于最高中断优先级，以最大程度减小中断延时。PRISS 寄存器用于将中断优先级与影子集配对。

中断延时

硬件中断延时为 8 个时钟周期。它等于从产生中断到 CPU 完成执行 ISR 中第一条指令的时间。软件延时取决于 ISR 类型。对于硬件影子集，没有对应于内核寄存器的额外延时，但中断程序可能会保存一些CP0寄存器的内容。对于软件影子集，延时可能为 20 或更多时钟，因为必须将多个内核寄存器压入堆栈。在 ISR 向量处实现跳转表会另外增加 2 个用于 JUMP 指令的时钟周期。

持久性中断和非持久性中断

使用 FIFO （以水平线作为中断源）和比较器的外设具有持久性中断 （UART 和 SPI）。该中断在中断原因得到处理之前是不可清除的。必须从 FIFO 中读取这种类型的中断数据，直到它低于水平线为止。在中断原因得到处理之后，可以清除中断标志。该中断会在超过水平线阈值时重新置为有效。

非水平线中断标志可以在读取相应寄存器之前清除。如果在读取寄存器之前清除中断标志，则在清除该标志之前可以发生第二次中断（不会被检测）的时间也会减少。

不可屏蔽中断 （NMI）

发生 NMI 时，程序执行会跳转到 NMI 处理程序。如果用户未指定一个处理程序，则会使用从 NMI 返回的弱功能默认处理程序。 NMI 向量在多个 NMI 源 （从休眠唤醒、 WDT 事件和复位）之间共用。在发生 WDT 事件时，会启动 NMI 定时器。如果 NMI 定时器在 NMI 得到处理之前计时期满，则会发生器件复位。以系统时钟表示的 NMI 定时器超时周期可以使用 RNMICON 寄存器设置。

系统

外设总线 （PB）

仅实现了单条外设总线。 PB 分频比固定为 1:1。

系统密钥 （SYSKEY）

为了防止意外访问某些寄存器、振荡器和复位，通过一种硬件锁定机制对它们进行保护。要解锁这些寄存器，必须按顺序向SYSKEY寄存器中写入密钥值0xAA996655和它的逆序值。如果在这两次写操作之间访问任何其他系统寄存器，则会中止解锁操作。写入一个非序列值也会重新锁定寄存器。为了防止这种情况，在使用解锁序列之前，应先禁止 / 暂停中断和 DMA 访问。可以通过读取 SYSKEY 寄存器来确定锁定状态。在发生任何复位之后，都会自动锁定系统。

例 11： 解锁序列

```
SYSKEY = 0x00000000;
// force lock, reset sequence

SYSKEY = 0xAA996655;   // unlock sequence
SYSKEY = 0x556699AA;
```

节能模式

有两种类型的节能模式可供使用：休眠和空闲。

存在多种休眠模式，具体取决于所需的功耗和唤醒时间要求。在休眠模式下会禁止大多数时钟。只有选定外设（包括 ADC 和电平变化通知（Change Notice，CN））可以在休眠模式下工作。

数据保持休眠是一种功耗低于休眠的模式。它使用一个专用的低压稳压器来维持状态和 RAM 的内容。由于该模式下的工作电压较低，唤醒时间会比休眠模式长。

只有一种空闲模式，但大多数外设都具有软件可写的位 SIDL，用于决定外设是否在空闲模式下处于活动状态。在运行模式下使能的时钟源会在空闲模式下继续运行。CPU 会停止执行指令，直到发生唤醒事件为止。在器件处于空闲模式时，独立 DMA 外设可以执行传输。

器件通过 WAIT 指令来进入节能模式。所进入的模式取决于 RCON 寄存器中的 SLPEN 位的状态和选定的休眠模式。从休眠或空闲模式唤醒属于 NMI 事件。在唤醒后，CPU 会跳转到 NMI 处理程序。默认 NMI 处理程序会将代码执行返回到 WAIT 之后的指令处。

硬件跟踪

PIC32MM 系列器件不支持硬件跟踪。

调试断点

支持 4 个指令断点和 2 个数据断点。数据断点可以配置为在访问特定单元或向特定单元读 / 写特定值时发生。软件断点以 BREAK16 指令的形式提供。不支持复杂断点。

外设引脚选择 （PPS）

完全 PPS 允许将选定外设的输入和 / 或输出映射到具有 RPn 功能的任意器件引脚。 PPS 外设不按照 PPS Lite 要求的那样分组。

双看门狗定时器 （WDT）和确定性

WDT 具有两个定时器；一个用于运行模式下的操作，另一个用于休眠模式下的操作。运行模式 WDT 以用户选定时钟的速率递增。休眠模式 WDT 使用 LPRC 时钟源工作。对存储器和外设访问进行仲裁的 DMA 外设可能会减少在给定时间段内执行的 CPU 指令数。这可能会使 CPU 执行表现出非确定性，因而在预期代码在周期或窗口结束附近清零 WDT 时导致意外的 WDT 超时。

振荡器和时钟 （FOSC 与 FCY）

由于 CPU FCY 的单相特性，系统时钟与 FOSC 为 1:1 关系。不使用分频器。

时钟输出

时钟输出引脚上的信号来自 PB 时钟。它与系统时钟的频率相同。

参考时钟 （REFO）

REFO 外设会使用用户选定的整数分频比对用户指定的输入时钟进行分频，产生一系列广泛的整数分频输出。该外设的输出可送至 GPIO 引脚，以及用作选定外设的时钟输入。它可以用于基于公共器件时钟 （如 48 MHz USB 时钟）生成频率。不支持小数输出分频。

软件复位

CPU 没有 RESET 指令。软件复位的执行方式是，向复位控制寄存器 RSWRST 写入一个值，然后回读（见例 12）。复位控制寄存器通过 SYSKEY 进行硬件锁定。复位序列应后跟 4 条 NOP 或一条 while(1) 指令，以防止在复位序列期间发生任何写操作。

例 12：

```
unsigned    int temp;
SYSKEY = 0x00000000;
// force lock, reset sequence

SYSKEY = 0xAA996655;    // unlock sequence
SYSKEY = 0x556699AA;
RSWRST = 1;             // write the reset bit
temp = RSWRST;
// read operation will generate a reset

while(1);               // wait for the reset
```

直接存储器访问 （DMA）和带有 DMA 的外设

DMA 引擎用于在闪存、RAM 和外设器件之间的任一方向上传输数据。独立 DMA 唯一不支持的操作是写入闪存。所有 DMA 操作都会通过系统总线矩阵，因此，需要与 CPU 和其他 DMA 一起通过仲裁来获取总线访问权。 DMA 传输配置为仅使用物理存储器地址。

通用 DMA 外设是用于在无需 CPU 干预的情况下传输数据的独立 DMA。在单个 DMA 控制器中实现了多个 DMA 通道。数据传输基于 DMA 通道触发事件和通道优先级发生。在给定时间，只能有一个 DMA 通道在传输数据。该系列中的所有器件都没有 DMA 外设。

通用 DMA 还具有可以与 DMA 通道相链接的集成可编程 CRC 引擎。

USB 外设包含专用的基于链表的 DMA 引擎。描述符和数据传输配置存储在系统 RAM 中。该系列中的所有器件都没有 USB 外设。

闪存控制器包含专用的不可编程 DMA 引擎，用于在行写操作期间将数据从系统 RAM 传输到闪存。它是唯一可以执行闪存写操作的 DMA。

循环冗余校验 （CRC）引擎

CRC 引擎是用于减轻 CPU 计算负担的可编程 CRC 计算器。 CRC 是通用 DMA 的一部分，对于没有 DMA 的器件，它是类似的独立 CRC 模块。

参考资料

《编程器的 MIPS® 架构》，*第 I-A 卷：MIPS32® 架构简介，第 5 版*

《编程器的 MIPS® 架构》，*第 II-B 卷：microMIPS32™ 指令集*

《MIPS32® M14K™ 处理器内核数据手册》

《PIC32MM0064GPL036 系列数据手册》（DS60001324B_CN）

《PIC32 系列参考手册》***第 50 章 “采用 MIPS32® microAptiv™ 和 M-Class 内核的器件的 CPU”***（DS60001192B_CN）

《MPLAB® XC32 C/C++ 编译器用户指南》**第 14 章“中断”**（DS50001686G_CN）

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。

ISBN：978-1-5224-1658-6

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

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
Tel: 1-317-536-2380

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608
Tel: 1-951-273-7800

罗利 Raleigh, NC
Tel: 1-919-844-7510

纽约 New York, NY
Tel: 1-631-435-6000

圣何塞 San Jose, CA
Tel: 1-408-735-9110
Tel: 1-408-436-4270

加拿大多伦多 Toronto
Tel: 1-905-695-1980
Fax: 1-905-695-2078

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: 86-21-3326-8000
Fax: 86-21-3326-8021

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: 886-7-213-7830

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

芬兰 Finland - Espoo
Tel: 358-9-4520-820

法国 France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

法国 France - Saint Cloud
Tel: 33-1-30-60-70-00

德国 Germany - Garching
Tel: 49-8931-9700

德国 Germany - Haan
Tel: 49-2129-3766400

德国 Germany - Heilbronn
Tel: 49-7131-67-3636

德国 Germany - Karlsruhe
Tel: 49-721-625370

德国 Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

德国 Germany - Rosenheim
Tel: 49-8031-354-560

以色列 Israel - Ra'anana
Tel: 972-9-744-7705

意大利 Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

意大利 Italy - Padova
Tel: 39-049-7625286

荷兰 Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

挪威 Norway - Trondheim
Tel: 47-7289-7561

波兰 Poland - Warsaw
Tel: 48-22-3325737

罗马尼亚 Romania - Bucharest
Tel: 40-21-407-87-50

西班牙 Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

瑞典 Sweden - Gothenberg
Tel: 46-31-704-60-40

瑞典 Sweden - Stockholm
Tel: 46-8-5090-4654

英国 UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820