AN3399

保护闪存自写操作

作者： *Justin O'Shea*
Microchip Technology Inc.

简介

许多单片机都能够在运行时对自身的闪存进行重新编程。该功能可通过自举程序实用程序对程序存储器进行现场更新，并将数据存储在非易失性存储器中。以下讨论和示例基于dsPIC33和PIC24产品；但是，其中许多技术和注意事项适用于所有单片机。可通过以下两种方式之一对闪存程序存储器进行编程：使用外部硬件编程器和运行时自编程（Run-Time Self-Programming，RTSP）进行在线串行编程（In-Circuit Serial Programming™，ICSP™）。以下讨论针对用固件实现的自擦/写（RTSP）。

本应用笔记旨在提供有关如何防止闪存意外擦写操作（可能导致轻微到灾难性现场故障）的指南和最佳实践。在固件中添加闪存编程保护功能有助于降低发生问题的风险，确保稳健的现场更新。以下内容通过了解潜在问题来提高固件的稳健性，并提供了避免这些问题的方法。

了解潜在风险

为了防止意外擦写损坏，需要了解这些类型的问题在应用中是如何发生的。大多数问题通常由以下原因引起：

- 硬件问题/指令执行错误
- 违反操作规范
- 概念和架构缺陷

本应用笔记将针对可能导致不合规操作的硬件问题以及概念和架构问题。尽管软件/代码缺陷也可能导致自写问题，但一般的良好编码实践不在本文档的讨论范围内。

硬件问题

所有电子设计都容易受硬件相关问题的影响，这些问题可能由多种来源引起。本文讨论的重点是防止电源和时钟电路引起指令执行错误。

指令执行错误

在从闪存中取程序指令的过程中，当单片机内核中未满足信号时序时，就会发生指令执行错误的情况。这通常是由低电压条件或超出规范的时钟脉冲引起的。以在常用的C函数末尾执行RETURN指令时的情况为例。在dsPIC33和PIC24架构中，RETURN指令编码为0000 0110 0000 0000 0000 0000。如果在从闪存中取指令的过程中发生时序违例，则RETURN指令可能会被错误解析为0100 0110 0000 0000 0000 0000。尽管该值仅变化一位，但却完全改变了指令的含义，即变为ADD指令。

在C程序中，如果执行了ADD或其他指令操作码来代替预期的RETURN指令，则可能导致一个C函数实际转到闪存中紧随其后的下一个C函数。如例1所示，意外的代码流情形可能会意外地调用C函数（例如闪存擦除函数）。

除了RETURN指令，CALL、GOTO或与程序计数器相关的其他转移指令也经常出现在代码中，并可能导致到达意外的代码。如果在从闪存读取或者译码和执行期间目标地址发生位翻转，则程序计数器可能仍指向实现的闪存，而不是预期的地址。

例1：　　代码流

```
void    ExampleFunction(void)
{
    //...在此处写一些代码...
}   //编译器将在此处放置一条RETURN指令。如果取指错误，程序流将
继续执行到下一个闪存存储单元，这可能是UnlockAndProg()函数

void    UnlockAndProg(void)
{
    __builtin_write_NVM();  //注：此函数执行解锁和编程序列
}
```

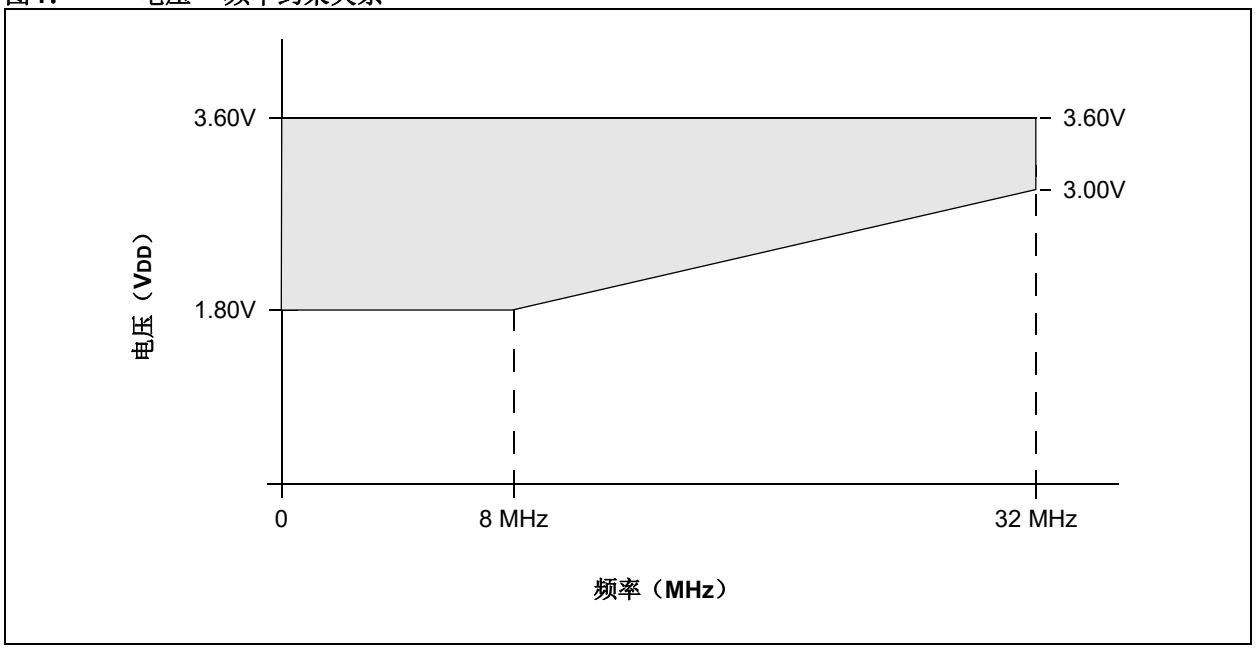
电源注意事项

为了避免指令执行错误，需要在单片机相应的电气规范范围内为其供电，以确保它们正常工作。有很多因素会导致问题，包括：

- 开关/按钮、电池触点、继电器和电源线缆上导致VDD中断的机械触点抖动
- 稳压器容差
- 温度、环境和机械老化引起的变化
- 不佳的电路板布线实践
- 由于较差的隔离或EMI敏感度导致较高的电源噪声

除了稳态（正常）操作外，单片机还必须能够正确处理上电和掉电事件。这些瞬态条件可能会违反与之相关的电压和频率时序要求。图1给出了典型的电压—频率关系图。要以给定速度在整个温度范围内工作，必须提供一定的电源电压。考虑这样一个场景：系统的电源尚未完全达到其标称值，但单片机已配置为全速运行。这种低电压条件可能导致单片机中的信号时序违例以及指令执行错误。掉电期间，当单片机全速运行而电源电压衰减时，可能会发生相同的情况。

图1：　　电压—频率约束关系



建议特别注意应用的上电和掉电行为并加以分析。上电和掉电斜坡的时间对人而言可能很短（微秒至毫秒），但能够执行几十万条指令。

机械元件（例如开关和连接器）可能对单片机的电源造成潜在的问题。如果板上电源或VDD电容无法充分滤除触点抖动，则这些抖动会强制为预期的单个上电状态快速连续地多次重复上电和掉电斜坡。在部署到现场多年并经过许多次启动后，与新系统相比，机械系统表现出截然不同的电气性能，这进一步加剧了这种问题。如果长时间空闲，表面的氧化物和污染物可能会积聚并进一步降低性能。当使用旨在断开电源的机械系统时，放置在开关元件之后的电容可以减少单片机检测到的VDD不稳定性，但也应进行评估，以确保开关导通时的峰值浪涌电流不会超过元件额定值或过早地磨损接触元件。

欠压复位（BOR）

dsPIC33和PIC24单片机具有BOR电路，可缓解欠压条件并将CPU置于复位状态，以防止意外动作。尽管能够防止会引起指令执行错误的典型VDD瞬变，但不应将BOR电路视为可以防止所有情况的解决方案。

BOR硬件是使用模拟电路实现的，这些模拟电路必须使用未知且可能不稳定的VDD产生自己的内部参考电压来自行供电。当单片机处于休眠模式时，它们还必须平衡精确电压检测、快速响应时间和低静态功耗的需求。

为了满足竞争要求，大多数BOR设计都实现了多种工作模式。例如，BOR在VDD上升时将系统保持在复位状态，然后在VDD接近BOR释放阈值时进入更精确、更快速的响应模式。之后，它将在系统开始执行且VDD漂移至远离BOR跳变点后恢复为低电流状态。随后，当VDD处于衰减状态或可能在休眠期间进入更低电流的工作状态时，它可能会反转状态转换。BOR可以检测每种状态下的功率损耗，但是由于每种模式都有一定的稳定延时，因此，如果VDD围绕BOR的跳变点快速上下振荡，则准确的触发电压和响应延时可能有所变化。

此外，BOR仅可防止低电压条件。它无法检测或防止选择或使用接近BOR限值的过快时钟配置的情况。功率事件通常与某些电路中产生的额外或异常的电噪声同时发生。例如，如果VDD由直流-直流开关电源（Switching Mode Power Supply，SMPS）提供，开启电源可能需要在几毫秒的过程中对VDD上的所有电容进行充电。这意味着VDD将足够高以供BOR释放系统来进行操作，同时电源仍以高电流值开关以对所有VDD电容进行充电。可以借此机会执行数千条指令，但这是一段存在较大噪声的不理想时间，存在让您的RTSP代码意外执行的风险。

如果以较低的速度（例如FRC）启动单片机并在稍后将时钟切换为高速时钟，则可以通过降低启动时的峰值电流需求并同时减少在VDD达到稳态之前将执行的指令数来提高稳健性。

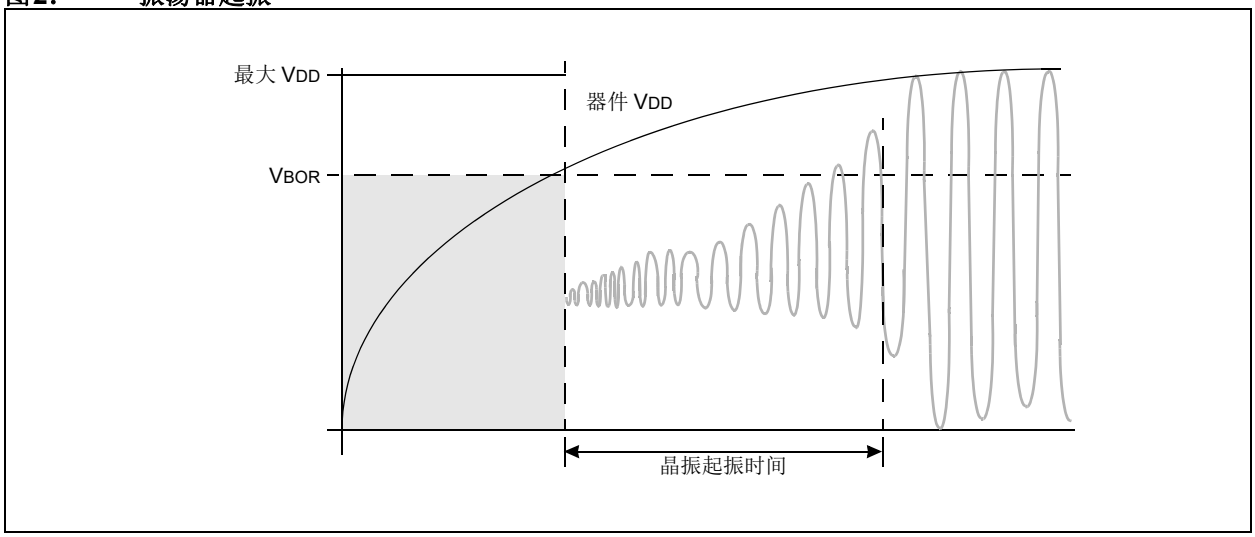
外部电压监控器

外部电压监控器可提供额外保护，以防止欠压和不合规条件。它们检测电源电压，并且可以通过控制$\overline{\text{MCLR}}$引脚来将单片机置于复位状态。外部监控器对于电源允许不合规操作（例如，不会一直降至0V的电压下降）的系统特别有用。当电源衰减到0.1V至0.7V之间的亚阈值电压时，单片机内部的某些BOR和POR电路无法生成已知的输出，因为其自身的工作电压不足。某些器件规定该条件不合规，要求VDD在回升至最小VDD值之前恢复0V。有关详细信息和限制，请参见器件特定数据手册的电气规范。这在电池供电的应用中尤其会带来一些问题。

时钟注意事项

干净且合规的时钟对于单片机正常工作至关重要。过短脉冲、偏移占空比、噪声和其他违反最小时钟宽度的异常即使仅持续一个周期，也可能导致指令执行错误。振荡器起振是一个特例，应在示波器上观察整个温度范围内的起振情况，以确保有效工作。如图2所示，有可能会在时钟达到足够的幅值和稳定的周期之前就使用时钟启动系统。使用内部FRC启动单片机并在外部振荡器稳定后切换到外部振荡器可帮助提高稳健性。如果使用PLL来提高时钟速度，请确保其在约束条件内运行，并且在使用时不要修改关键PLL设置。有关PLL限制的详细信息，请参见器件特定数据手册。

图2：　　振荡器起振



典型晶振或陶瓷谐振器电路具有半正弦振荡器波形，与方波相比，它们的边沿压摆率相对较低。这些低压摆率的边沿更容易受到外部噪声的影响，从而有可能使VDD/2交叉点附近的噪声脉冲表现为额外的意外时钟边沿。此外，由于振荡器波形本质上是正弦波，因此如果单片机使用施密特触发输入缓冲器将模拟波形转换为数字方波时钟，则将损坏精确的占空比和边沿时序信息。

建议特别小心，确保振荡器的输入和输出不受外部产生的噪声源（例如电力电子应用中功率MOSFET的开关瞬变）影响。可以通过PCB布线缓解这种情况：确保振荡器网络/走线与任何产生噪声的走线保持物理隔离，并在振荡器电路周围添加接地的屏蔽走线。

防止指令执行错误

下面总结了可防止指令执行错误的措施：

1. 使用FRC启动器件并在外部振荡器稳定后切换到外部振荡器，或者在可能的情况下，使用内部FRC或FRC+PLL以保守的频率实现所有RTSP操作，以降低IDD并延长断电时的VDD衰减时间。
2. 在启动时添加软件延时。
3. 使能BOR。如果电压值可配置，请将其设置为足以提供充分保护的大小。
4. 确保存在旁路电容且大小符合数据手册的要求。验证电源电压在上电、运行和掉电时不受振荡影响。
5. 对于带有VCAP引脚的器件，确保存在大小合适的电容。
6. 使用外部电压监控器来确保单片机在VDD不合规时无法执行代码。
7. 确保正确实现了PLL初始化和运行时时钟切换代码。
8. 在尝试编程或擦除操作之前，使用ADC验证电源电压是否足够高。
9. 在最终PCB上的温度范围内验证振荡器电路。
10. 如果可配置，请使能并使用上电延时定时器（Power-up Timer，PWRT）。

概念和架构注意事项

可以执行RTSP操作的应用程序软件的架构应最大程度降低意外擦/写和损坏闪存的风险。以下各部分将讨论架构概念和添加保护措施的方法。可使应用程序不易受意外闪存擦/写损坏影响的架构概念包括：

1. 在启动时添加软件延时。
2. 稳健的自举程序进入。
3. 使用动态密钥进行硬件解锁。
4. 解锁序列的单个实例。
5. 软件解锁序列。
6. 清零写使能位。
7. 验证在擦/写操作之前VDD是否足够。
8. 验证擦/写地址。
9. 将擦/写地址指定为安全存储单元。

在启动时添加软件延时

某些应用程序需要在应用程序开始时将数据写入非易失性存储器，但此操作很危险，可能会导致意外操作。也许最有效的RTSP保护措施是在启动时添加软件延时。一旦代码在退出复位之后开始执行，无论是否计划好RTSP操作，正确的做法都是进入软件延时循环，以将主执行延迟几十或几百毫秒。这段延时将使PCB上的所有其他元件达到稳态，包括VDD电容，该电容应达到其预期工作电压（例如：3.3V）而不是其最低工作电压（例如：3.0V），达到最低工作电压时，器件刚好超过POR/BOR限值。

确保软件在启动后的一段时间内不执行任何操作的另一个好处是，它可以抑制开发过程中的意外ICSP™编程故障和明显的RTSP擦除/编程故障。首次实现RTSP代码时，通常的想法是将NVM擦写测试程序直接置于`main()`中，然后尝试对闪存执行ICSP读回，以查看代码是否按预期执行。MPLAB® X IDE ICSP编程工具读取DEVID、批量擦除闪存、编程十六进制内容和最终执行闪存的读回验证时，可能会出现争用情况。在编程步骤和读回步骤之间，ICSP工具可能会释放MCLR并允许RTSP代码执行几毫秒。如果没有软件延时，这可能会导致IDE中出现虚假读回验证故障。

对于使用MPLAB XC16 C编译器实现的软件，可以通过在`main()`函数的开头调用`__delay32()`函数来创建可预测的启动延时。有关`__delay32()`函数的更多信息，请参见www.microchip.com上的《16位语言工具函数库》（DS51456D_CN）。

如果与上电/掉电无关的深度休眠类型复位事件要求快速启动，则可以根据POR或BOR状态标志（通常位于RCON SFR中）的状态有条件地执行`__delay32()`调用。通常，这两位必须在经过测试后用软件清零，并且只能在发生真正的针对电源的复位时用硬件复位。

稳健的自举程序进入

除了在电源事件后延迟软件执行之外，稳健的自举程序进入方法对于保护应用程序也很重要。建议不要使用过于简单的进入方法，例如仅检测一个UART字节。包含1和0位组合的32位或更长的检测序列不太可能被意外地从随机通信噪声中解码。此外，还建议应在自举协议中实现闪存解锁命令，而不是在应用程序软件中进行硬编码。主机应用程序不仅应负责发送进入自举程序模式的命令，还应负责发送单独的擦/写操作解锁命令，然后才允许发送实际修改闪存内容的其他命令。

用于硬件解锁的动态密钥

建议为NVM擦/写硬件解锁序列使用动态密钥。命令解锁所需的特定密钥应从主机传递，并在使用后销毁，以进一步防止意外操作。dsPIC33和PIC24器件的典型硬件解锁密钥为0x55/0xAA，不得自行在擦/写程序中进行硬编码。可以在安全的自举程序进入序列期间将解锁密钥存储在RAM中，以供以后在擦/写程序中使用。操作完成后，应销毁密钥，以防在尝试下一次安全自举程序进入之前进行其他擦/写操作。这样可以防止意外的代码流执行解锁序列。

XC16编译器包括一些用于解锁序列的内置函数。这些函数将写入NVMKEY寄存器，并将写入位置1以启动擦/写序列。建议使用安全版本`__builtin_write_NVM_secure()`来代替`__builtin_write_NVM()`函数。安全版本支持将动态密钥传递给函数。

此外，还应注意确保编译器不会通过在闪存中存储的实际指令中用立即数值0x55和0xAA替换变量来优化代码。通常，只要变量的内容读取自或来源于通信SFR并且未在代码中显式指定为任何类型的常量，便可防止这种情况。

解锁序列的单个实例

dsPIC33和PIC24单片机在允许闪存擦/写操作之前，需要执行一个解锁序列。可以对应用程序代码进行结构化，使解锁序列硬编码并在多个不同的软件函数中复制，从而用于不同类型的操作，包括页擦除、字写入和行写入。

如果不使用动态密钥，则硬件解锁序列只能在一个位置作为专用函数的一部分执行。由于硬件解锁密钥和NVM操作在意外执行时会导致器件无法运行，因此组织代码时应确保解锁密钥和NVM操作启动代码仅出现在单个函数中，而不会在项目中的其他任何位置重复出现。该函数应接收NVM操作值作为参数，这样一来，如果程序计数器由于指令执行错误而意外访问了解锁密钥代码，则该参数将无效并且不会造成损坏。通过将解锁序列代码单独放在一个位置，仅需要一组附加保护代码来减少安全漏洞。

软件解锁序列

使用解锁密钥或序列防止意外操作的概念也适用于应用程序代码。任何直接擦/写闪存的函数或调用执行此类操作的函数都应有一个传递给它的解锁密钥输入参数。建议使用32位或更大的变量实现软件解锁密钥，以提供最佳保护。下面的例2提供了实现此概念的一些代码，另外还进行地址和V_{DD}检查。

例2：　　使用软件解锁密钥

```
#define  SOFTWARE_UNLOCK_KEY_VALUE  (uint32_t)0x600D92FE
#define  NVM_MIN_WRITEABLE_ADDRESS  (uint32_t)0x4000     //应用特定值，请在代码中更改该值。
#define  NVM_MAX_WRITEABLE_ADDRESS  (uint32_t)0x4FFF     //应用特定值，请在代码中更改该值。
#define  NVM_MIN_VOLTAGE_MILLIVOLTS (uint16_t)3100       //应用特定值，请在代码中更改该值。

bool     NVM_OpStart(uint32_t address, uint32_t softwareUnlockKey, uint16_t key1, uint16_t key2)
{
    uint16_t VDDMilliVolts;

    //检查以确保传递给该函数的softwareUnlockKey正确
    if(softwareUnlockKey != SOFTWARE_UNLOCK_KEY_VALUE)
    {
         //错误！调用函数未将正确的参数传递给该函数。
         return false;
    }

    //确保指向的地址对擦 / 写操作有效。
    if((address < NVM_MIN_WRITEABLE_ADDRESS) || (address > NVM_MAX_WRITEABLE_ADDRESS))
    {
         //地址超出范围。
         return false;
    }

    //确保VDD足够高以进行安全的擦 / 写操作
    VDDMilliVolts = ADC_MeasureVDD();
    if(VDDMilliVolts < NVM_MIN_VOLTAGE_MILLIVOLTS)
    {
         return false;
    }

    //所有检查均已通过，启动擦 / 写操作。
    __builtin_write_NVM_secure(key1, key2);

    return true;
}
```

写使能位的安全处理

单片机硬件通常会实现某种写使能位（即WREN），在执行解锁序列来擦/ 写闪存之前必须先将该位置1。通用做法是在需要擦/ 写闪存时才将WREN置1，然后在擦除或编程操作完成后立即将其清零。这样做的理由是，硬件互锁断开的持续时间越短，发生意外的可能性就越低。但是这有意外的副作用：执行擦除或编程操作的代码表现为原子单元，如果意外进入之前的擦除/编程准备代码链，则会继续执行实际的NVM擦除/编程操作。这是不可取的，因为如果程序计数器跳转到或陷入链中的任何一环，则整个RTSP重编程链中的所有代码都将存在风险且不可预测。如果WREN按需即时断开互锁，实际上不提供任何附加保护。

为了使WREN提供实际的保护，应采用一种策略，即仅在构成RTSP重编程链的所有迭代代码循环之外的一个代码位置将WREN置1一次。例如，仅当VDD正常且从通信伙伴处接收到有效的32位或更长的自举程序“Hello”，或者从菜单选择中实际发出用户输入命令时，才将WREN置1。随后，应在代码中检测到不可恢复错误或发生普通RTSP代码终止的多个位置将WREN清零。例如，在通信超时或发生损坏事件时将WREN清零，在自举程序将完整的应用程序映像编程到闪存中后将WREN清零，在所有陷阱处理程序中将WREN清零以及在自举程序将执行移交给现有应用程序时将WREN清零。在所有其他情况下，WREN的状态都应保持不变。这种做法将抑制意外进入RTSP算法链的行为产生任何永久性后果，而与意外进入的算法链中的位置无关。

验证在擦/写操作之前VDD是否足够高

在进行擦/写操作过程中，不得复位单片机。如果由于MCLR事件和时钟监控器故障等导致发生复位，则大多数单片机将完成已经启动的NVM操作，因此各个闪存字很少会出现未完全擦除/未完全编程的中间状态。但是，因BOR等低电压条件而导致发生复位将立即终止擦/写操作，因此应通过在擦/写操作之前测量VDD从算法上避免此类风险。这可以通过在闪存擦/写解锁序列之前实现用于测量应用VDD的代码来完成，如例2所示。如果VDD过低，代码应避免启动擦/写操作。精选的dsPIC和PIC24单片机实现了与带隙参考连接的ADC通道。这可用于反向计算目前的运行时VDD电压。一些单片机还实现了高/低电压检测（HLVD或LVD）模块，可用于在运行时测量应用的VDD电压。

验证擦/写地址

在执行解锁功能来启动擦/写操作之前，建议始终检查预期的目标擦/写地址。这样做可确保它指向应用为擦/写操作保留的有效区域。例如，自举程序通常应执行检查以确保自举程序不尝试擦除自身的一部分（除非明确设计为这样做），并且EEPROM仿真代码应验证擦/写区域是否仅包含为仿真的EEPROM分配的闪存。检查的结构应当仅允许合法的目标地址，并且确保目标地址处于上下边界范围内。例如，不应仅检查目标地址是否“高于自举程序”，而应检查请求的擦/写目标地址是否高于自举程序，是否低于第一个未实现的程序存储器地址，以及最低有效对齐位是否为零，以匹配适合于所请求操作的硬件擦/写寻址粒度。应将任何偏差视为将WREN清零的持续锁定条件和/或其他使能标准。

将NVM擦/写地址指定为安全存储单元

在设置擦/写操作的代码与用于解锁密钥和最终触发NVM操作的代码之间始终存在某种程度的去耦。设置序列的一部分通常将写入NVMADR寄存器，以指示即将进行的擦除/编程操作的目标地址。只需将NVMADR指定为未实现的目标地址，即可防止因意外进入这两个代码序列之间的执行路径而造成损坏。如果使用无效地址进行了意外的擦/写操作，则硬件将忽略这种情况，存储器将不进行任何更改。这可以通过在复位后以及每次擦/写事件后将NVMADR设置为任何无效地址（例如0xFFFFFE）来实现。

总结

自写应用程序可以通过多种方式保护自己免受闪存损坏的影响。在设计具有自写功能的应用程序时，建议分析硬件设计和软件实现，以确保它们遵循本文档中概述的最佳实践和建议。

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规范。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品非常安全。
- 目前，仍存在着用恶意、甚至是非法的方法来试图破坏代码保护功能的行为。我们确信，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。这种试图破坏代码保护功能的行为极可能侵犯 Microchip 的知识产权。
- 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。

有关 Microchip 质量管理体系的更多信息，请访问 www.microchip.com/quality。

ISBN：978-1-5224-7246-9

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

中国 - 北京
Tel: 86-10-8569-7000

中国 - 成都
Tel: 86-28-8665-5511

中国 - 重庆
Tel: 86-23-8980-9588

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

中国 - 南京
Tel: 86-25-8473-2460

中国 - 青岛
Tel: 86-532-8502-7355

中国 - 上海
Tel: 86-21-3326-8000

中国 - 沈阳
Tel: 86-24-2334-2829

中国 - 深圳
Tel: 86-755-8864-2200

中国 - 苏州
Tel: 86-186-6233-1526

中国 - 武汉
Tel: 86-27-5980-5300

中国 - 西安
Tel: 86-29-8833-7252

中国 - 厦门
Tel: 86-592-238-8138

中国 - 香港特别行政区
Tel: 852-2943-5100

中国 - 珠海
Tel: 86-756-321-0040

台湾地区 - 高雄
Tel: 886-7-213-7830

台湾地区 - 台北
Tel: 886-2-2508-8600

台湾地区 - 新竹
Tel: 886-3-577-8366

亚太地区

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733

印度 India - Bangalore
Tel: 91-80-3090-4444

印度 India - New Delhi
Tel: 91-11-4160-8631

印度 India - Pune
Tel: 91-20-4121-0141

日本 Japan - Osaka
Tel: 81-6-6152-7160

日本 Japan - Tokyo
Tel: 81-3-6880-3770

韩国 Korea - Daegu
Tel: 82-53-744-4301

韩国 Korea - Seoul
Tel: 82-2-554-7200

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870

菲律宾 Philippines - Manila
Tel: 63-2-634-9065

新加坡 Singapore
Tel: 65-6334-8870

泰国 Thailand - Bangkok
Tel: 66-2-694-1351

越南 Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

欧洲

奥地利 Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

丹麦 Denmark - Copenhagen
Tel: 45-4485-5910
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

德国 Germany - Garching
Tel: 49-8931-9700

德国 Germany - Haan
Tel: 49-2129-3766400

德国 Germany - Heilbronn
Tel: 49-7131-72400

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: 47-7288-4388

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