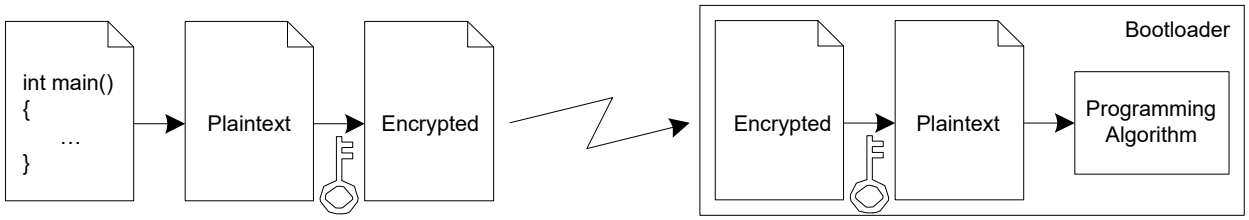
AVR231：AES 自举程序

简介

本应用笔记介绍如何在具有自举程序功能的 AVR®单片机上安全地更新固件。该方法采用高级加密标准（Advanced Encryption Standard，AES）对固件进行加密。

图 1. 概述



特性

- 适用于具有自举程序功能和至少 1 KB SRAM 的 AVR 单片机
- 支持将固件和敏感数据安全地传输到基于 AVR 的应用
- 包括易于使用的可配置应用程序示例：
  - 加密二进制文件和数据
  - 创建目标自举程序
  - 将加密文件下载到目标
- 采用高级加密标准（AES）：
  - 128 位、192 位和 256 位密钥
- AES 自举程序可适应 2 KB 存储空间
- 64 KB 应用程序、115200 波特率和 3.69 MHz 目标频率条件下的典型更新时间：
  - AES128：27s
  - AES192：30s
  - AES256：33s
- 可以在 ATmega328PB Xplained Mini 上直接对应用进行评估
- 经测试，固件只需少量更改甚至无需更改即可应用于以下器件：
  - ATmega 8/16/162/169/32/64/128/256
  - ATmega168PA
  - ATmega328PB

目录

1. 说明

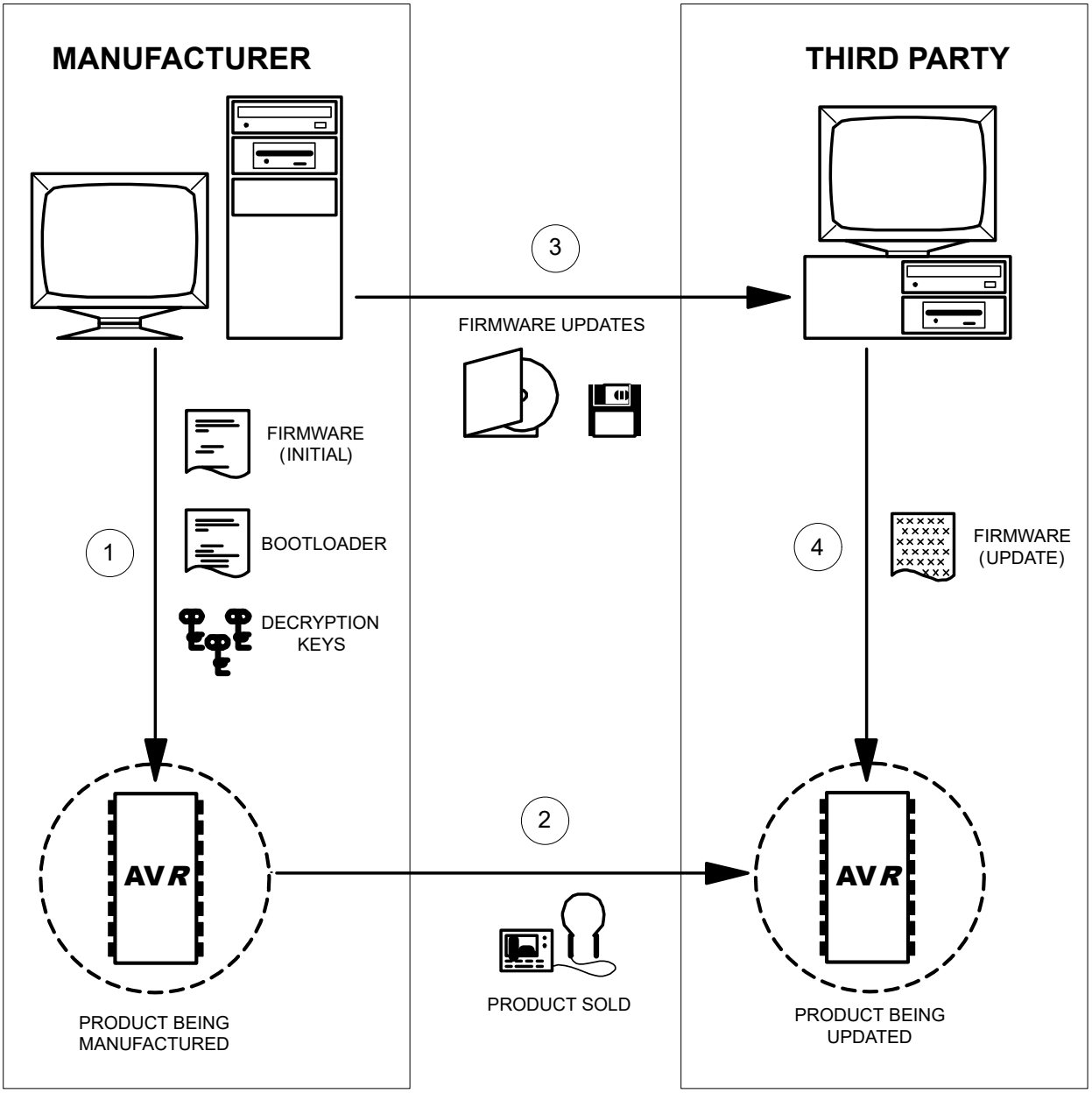
无论是便携式音乐播放器、吹风机还是缝纫机，只要是采用单片机的电子设备通常都需要集成固件。这是因为随着众多电子设备的迅速发展，越来越多的制造商希望能够对已出厂或出售的产品进行更新。想要更改硬件可能比较困难（毕竟产品已送交到最终客户手里），但对于基于闪存单片机（例如 AVR）的产品而言，更新固件却非常简单。因此，许多 AVR 单片机都会通过相应的配置来实现自举程序，从而确保能够接收固件更新内容，并根据需要对闪存进行再编程。程序存储空间分为两段：自举程序段（Bootloader Section，BLS）和应用程序段。这两个存储段都有专用的读/写保护锁定位，这样既能够在 BLS 中保护自举程序代码，又能够在应用程序区域中更新代码。这种方法可以轻松保护 BLS 中的更新算法，防止有人从外部访问。

但是，在将固件编程到闪存中并设置锁定位之前，固件仍然面临着安全问题。这意味着如果需要在现场更新固件，一旦固件离开编程平台或制造商的工作环境，便无法防止未经授权的访问。

本应用笔记说明如何通过加密方法确保数据始终安全地传输到闪存和 EEPROM 存储器。基本原理是在数据离开编程平台之前对其进行加密，并仅在其下载到目标 AVR 之后才对其进行解密。该过程不会防止未经授权的固件复制操作，但如果没有正确的解密密钥，加密信息实际上毫无用处。解密密钥仅存储在编程环境之外的一个位置——AVR 内部。该密钥无法通过加密数据重新生成。获得数据访问权限的惟一方法就是使用正确的密钥。

下图举例说明了从产品最初制造，装载初始固件，出售，一直到更新固件的全过程。

图 1-1. 产品制造和更新过程典型示例



注：

1. 在制造阶段，先在单片机中集成自举程序、解密密钥和应用固件。自举程序负责接收实际的应用程序并将其编程到闪存中，而密钥用于解密输入的数据。最后设置锁定位以保护 AVR 内部的固件。
2. 随后将产品发给代理商或出售给最终客户。此时，锁定位设置保持不变，继续保护 AVR 内部的固件。
3. 新版固件已完成，需要对已分销的产品进行更新。因此固件会被加密并发送给代理商。如果没有解密密钥，加密的固件将毫无用处，因此即使是软件的本地副本（例如，存储在代理商的硬盘上）也不会构成安全隐患。
4. 代理商对库存产品以及客户退回的产品（例如，正在维修的产品）进行升级。加密的固件将下载到 AVR，并在单片机内部解密一次。此时，锁定位设置保持不变，继续保护 AVR 内部已更新的固件。

2. 术语

BLS	自举程序段（Bootloader Section）
EEPROM	电可擦除 PROM（Electrically Erasable PROM）
AES	高级加密标准（Advanced Encryption Standard）
RAM	随机访问存储器（Random Access Memory）
CBC	密码块链接（Cipher Block Chaining）
PC	个人计算机（Personal Computer）
CRC	循环冗余校验（Cyclic Redundancy Check）
USART	通用同步/异步收发器（Universal Synchronous Asynchronous Receiver Transmitter）
KB	千字节（Kilobytes）
LCD	液晶显示屏（Liquid Crystal Display）
IDE	集成开发环境（Integrated Development Environment）

3. 准备工作

本文档中讨论的解决方案需要您对以下内容有基本的了解：

- Atmel® Studio 7 或更高版本
- ATmega328PB Xplained Mini
- AES 概念
- 自举程序概念及其在 AVR 中的实现

本应用笔记仅对高级加密标准（AES）做基本的介绍。如果用户希望深入了解 AES 概念，则需要进一步阅读相关文献。

4. 加密概述

术语“加密”是指使用密钥将信息锁定，必须通过正确的密钥解锁才能使用。

基于加密密钥的算法分为对称和非对称两类。对称算法使用相同的密钥进行加密和解密，而非对称算法则使用不同的密钥。AES 是一种对称密钥算法。

4.1 加密

加密是一种对消息或数据进行编码以使其内容对外隐藏的方法。原始格式的明文消息或数据可能包含作者或代理商想要保密的信息，比如单片机的固件。在现场更新单片机时，可能很难保护固件免遭非法复制和逆向工程。但通过对固件进行加密，可以确保其在被解密之前毫无用处。

4.2 解密

解密是一种检索原始消息或数据的方法，在不知道正确密钥的情况下通常无法执行解密。密钥可以存储在单片机的自举程序中，以便器件接收加密的数据，对数据进行解密，然后对闪存或 EEPROM 存储器的特定部分重新编程。如果对锁定位进行了相应的编程，就无法从加密数据中检索解密密钥，也无法从 AVR 单片机中读取解密密钥。

5. AES 概述

5.1 AES 实现

本节将略过关于 AES 算法或其历史的详细信息，重点介绍算法各个部分中的 AVR 特定实现。由于存储器是嵌入式应用中的稀缺资源，因此如何节省代码存储空间一直是重点所在。自举程序永远不会与主代码同时运行，因此只要数据存储空间（RAM）需求不超过单片机的存储容量，就无需节省存储空间。

以下各小节将介绍一些基本的数学运算及其 AVR 特定实现。请注意，文中引用了数学领域的有限域理论。阅读本文档不需要了解有限域知识，但感兴趣的读者应当掌握 AES 规范。

注： 如果读者对 AES 实现已有足够的了解，可以直接跳到 6. 软件实现与使用，不会影响到文章的连贯性。

5.1.1 字节加法

在 AES 算法中，字节加法的定义是各个位之间的无进位加法。这与标准异或运算完全相同。由于异或运算与其自身互为逆运算，因此字节减法与 AES 算法中的加法完全相同。异或运算可在 AVR 上轻松实现。

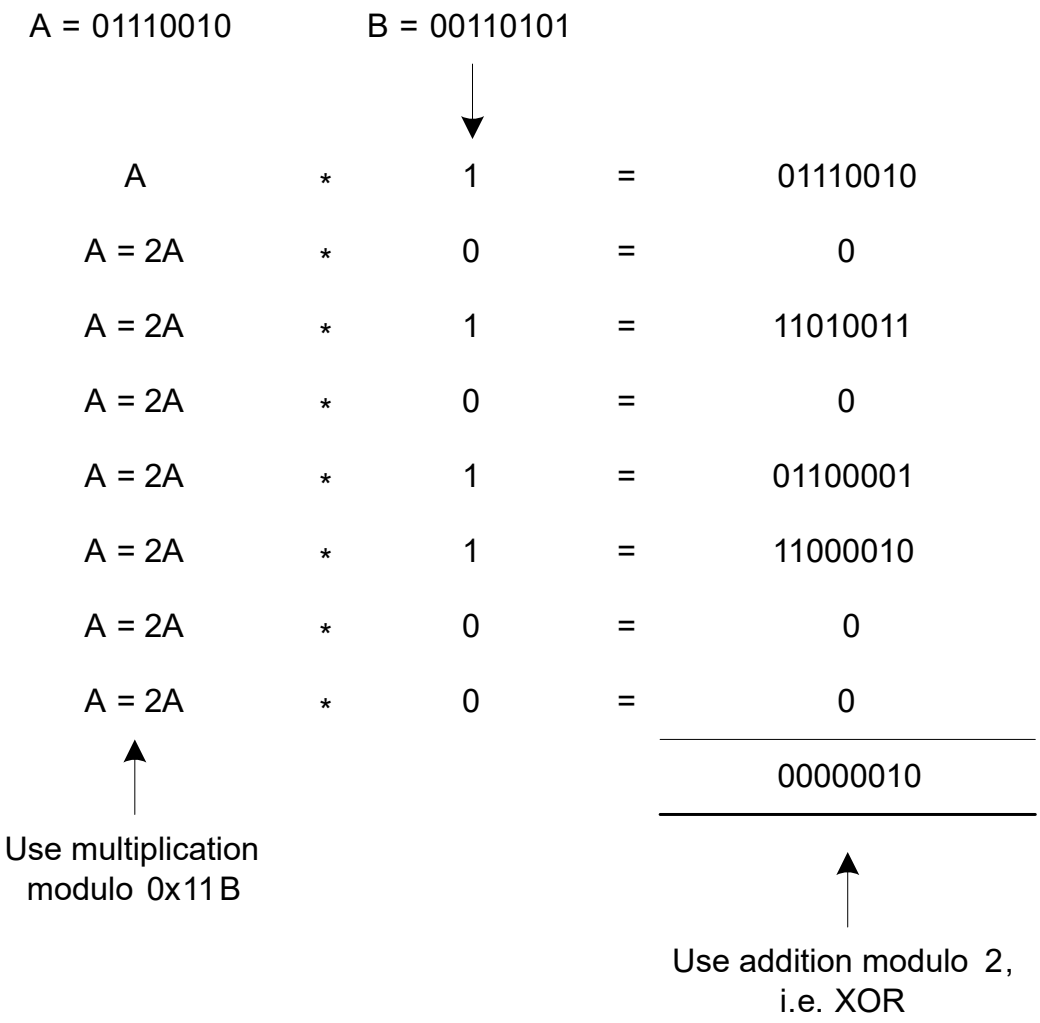
5.1.2 字节乘法

在 AES 算法中，字节乘法的定义是模数为 0x11B（二进制 1 0001 1011）的有限域乘法。推荐的实现方法是将第一个因子反复乘以 2（模数为 0x11B），并将第二个因子中值为 1 的每个位的中间结果相加。

示例：如果第二个因子是 0x1A（二进制 0001 1010），则应将第一个、第三个和第四个中间结果相加。

下图给出了另一个示例。这种方法占用的存储空间极少，非常适合 8 位单片机。

图 5-1. 字节乘法



字节乘法可以通过以下伪代码来描述：

```
bitmask = 1
tempresult = 0
tempfactor = firstfactor
while bitmask < 0x100
    if bitmask AND secondfactor <> 0
        add tempfactor to tempresult using XOR
    end if
    shift bitmask left once
    multiply tempfactor by 2 modulo 0x11B
end while
return tempresult
```

5.1.3 乘法逆元

为了能够计算有限域乘法逆元（即 1/x），该实现中利用了一项技巧。使用相同底数的幂和对数时，可以利用以下恒等式：

使用幂和对数计算 1/x。

$$a^{-\log_a x} = \frac{1}{x}$$

本例中选择的底数是 3，因为它是最简单的原根。通过使用有限域乘法来计算指数和对数，可以轻松实现乘法逆元。为避免每次都计算指数和对数，可使用两个查找表。由于乘法逆元仅在准备 5.1.4 S-Box 所述的 S-box 时使用，因此一旦 S-box 准备完毕，两个查找表所占用的存储空间便可用于其他用途。

查找表计算可以通过以下伪代码来描述：

```
tempexp = 0
tempnum = 1
do
    exponentiation_table[ tempexp ] = tempnum
    logarithm_table[ tempnum ] = tempexp
    increase tempexp
    multiply tempnum by 3 modulo 0x11B
loop while tempexp < 256
```

5.1.4 S-Box

AES 算法使用替换表（即 S-box）的概念。该算法的其中一个步骤是对字节应用可逆变换。S-box 是对所有可能的字节值应用这一变换后的预计算结果。该变换包括两个步骤：(1) 5.1.3 乘法逆元所述的乘法逆元运算；(2)遵循以下公式的线性变换，其中 a_i 是结果的位，b_i 是步骤 1 的结果的位。

S-box 中使用的线性变换：

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

仔细观察矩阵可以发现，运算可以通过对原始字节、右侧向量以及原始字节循环左移一次、两次、三次和四次后的结果求和（使用异或加法）来实现。该方法非常适合 8 位单片机。

用于解密的逆 S-box 具有类似的结构，也可搭配使用异或加法和循环来实现。相应的矩阵请参见 AES 规范；有关实现的详细信息，请参见源代码。

5.1.5 “State”（体）

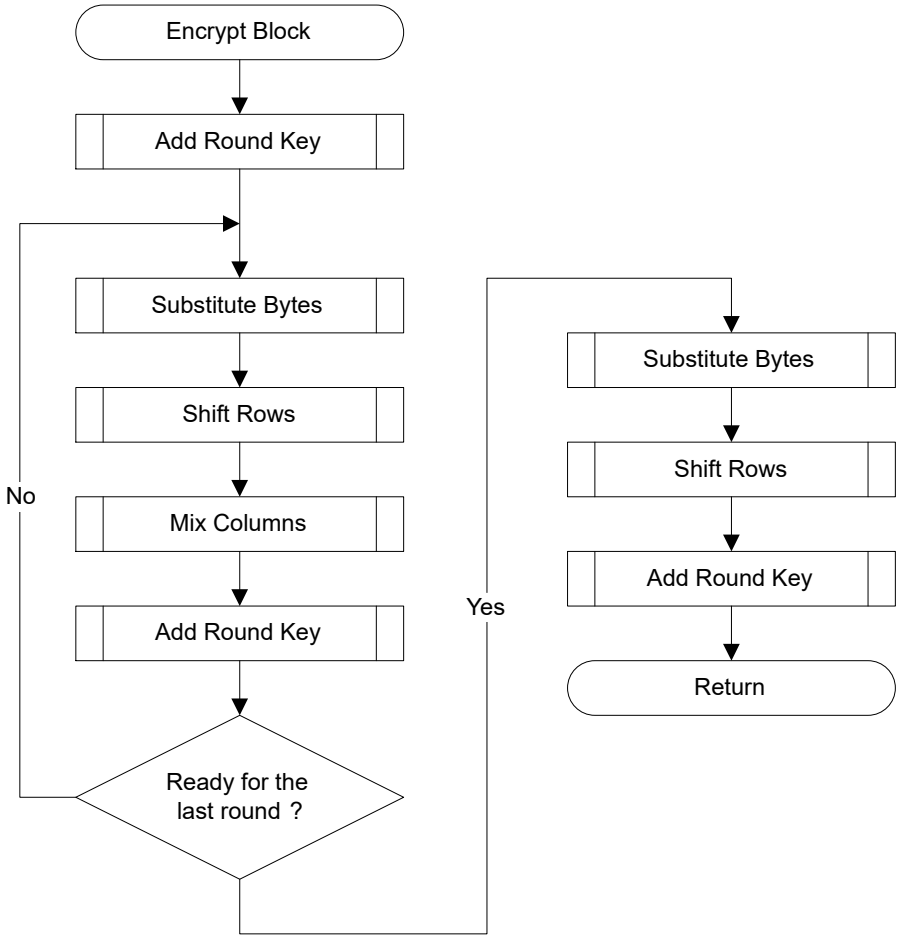
AES 算法是分组密码，这意味着数据以块的形式进行管理。对于 AES 密码，块大小为 16 字节。AES 块通常采用 4x4 的数组结构（称为“State”或“State 数组”）。State 的最左列包含块的前四个字节，从上到下，依此类推。另外请读者注意，在 AES 规范中，四个连续的字节称为一个字。

5.2 AES 加密

在讨论加密过程的步骤之前，需要先引入“加密轮数”这一概念。大多数分组密码是由一系列循环执行多次的运算组成。每个循环迭代使用不同的加密密钥。每个迭代中至少有一次运算取决于密钥。循环迭代次数即称为加密轮数，在这些轮次中使用的一系列密钥称为密钥编排方案。加密轮数取决于密钥大小。

下图给出了加密过程的流程图。以下各小节将分别介绍该过程中的不同步骤。为了方便起见，每个步骤均以子程序的形式实现。使用优化编译器可删除不必要的函数调用，从而节省代码存储空间。

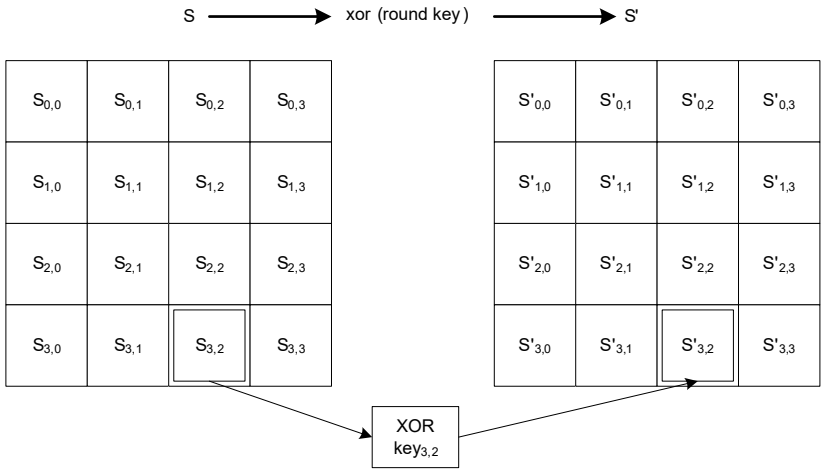
图 5-2. 加密流程图



5.2.1 添加轮密钥

该步骤使用异或加法将当前轮密钥添加到当前 State 数组中。轮密钥的大小与 State 相同，即 16 个字节（4 个字）。该运算以 16 步循环的形式实现。

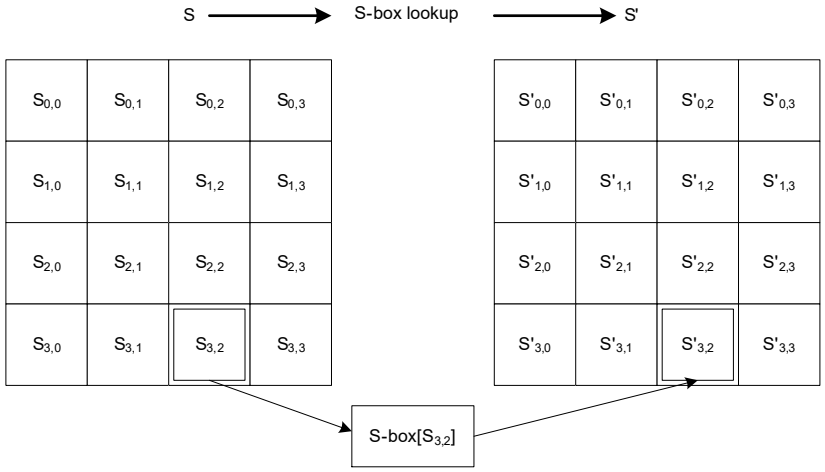
图 5-3. 将轮密钥添加到当前 State



5.2.2 替换字节

该步骤使用预先计算好的 S-box 查找表来替换 State 中的字节。与上一节相同，该步骤也以 16 步循环的形式实现。

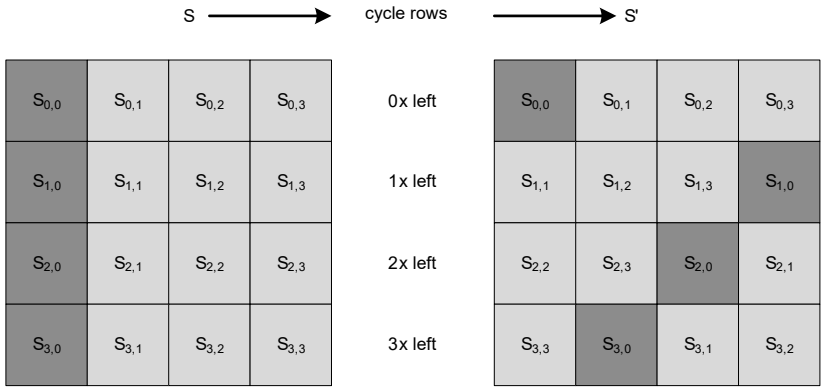
图 5-4. 替换当前 State 的字节



5.2.3 移动行

该步骤对当前 State 的行进行运算。第一行保持不变，而后三行分别循环左移一次、两次和三次。要循环左移一次，需将最左边的字节移到最右边的列，然后将其余三个字节左移一列。该过程如下图所示。

图 5-5. 循环移动当前 State 的行

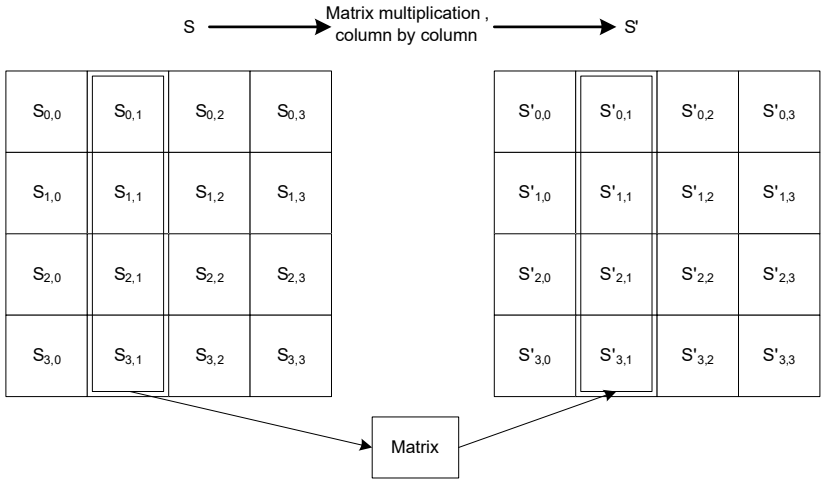


传统的实现方案是首先编写一个可将行循环左移一次的子程序，然后在每一行上调用所需的次数。但测试表明，直接执行字节改组（而不采用任何循环或子程序）反而能够以 3 倍的速度快速实现，只是代码长度会略微增大。因此，我们选择了直接实现方案。有关详细信息，请参见源代码中的 **ShiftRows()**函数。

5.2.4 混合列

该步骤在 State 上逐列进行运算。每列均被视为一个多字节向量，与固定矩阵相乘后可得到修改后 State 的列。

图 5-6. 混合当前 State 的列



该运算可通过以下公式描述，其中包含混合列的字节以及原始列的字节。

图 5-7. 混合一列时的矩阵乘法

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

该步骤直接实现，无需任何辅助函数调用。从矩阵公式可以看出，混合列的每个字节均为原始字节与其对应项的组合。有关详细信息，请参见源代码中的 **MixColumns()**函数。

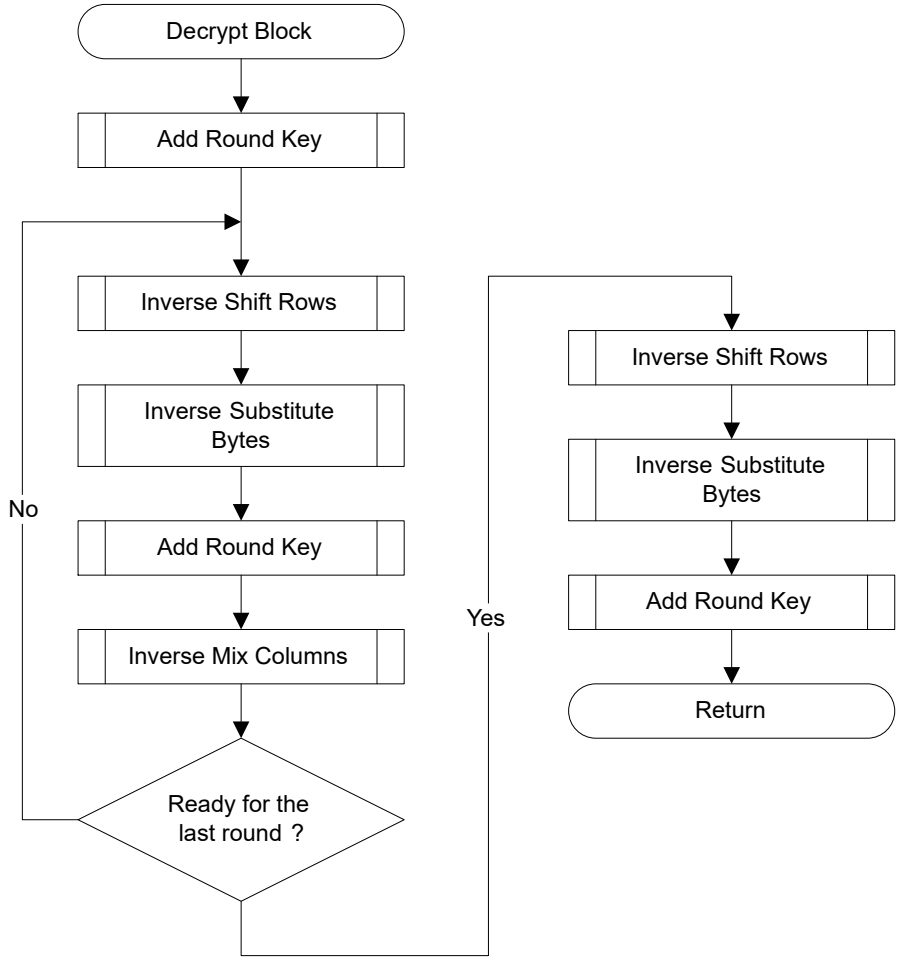
注： 使用 5.1.1 字节加法和 5.1.2 字节乘法中的异或加法和有限域乘法。

5.3 AES 解密

该过程与加密过程非常相似，惟一的不同之处在于步骤的顺序。除了“添加轮密钥”之外，所有步骤均具有其对应的逆步骤。“逆移动行”将使行循环右移而不是循环左移。“逆替换字节”使用逆 S-box。

“逆混合列”同样使用逆变换。相应的矩阵请参见 AES 规范；有关实现的详细信息，请参见源代码。下图给出了解密过程的流程图。

图 5-8. 解密流程图

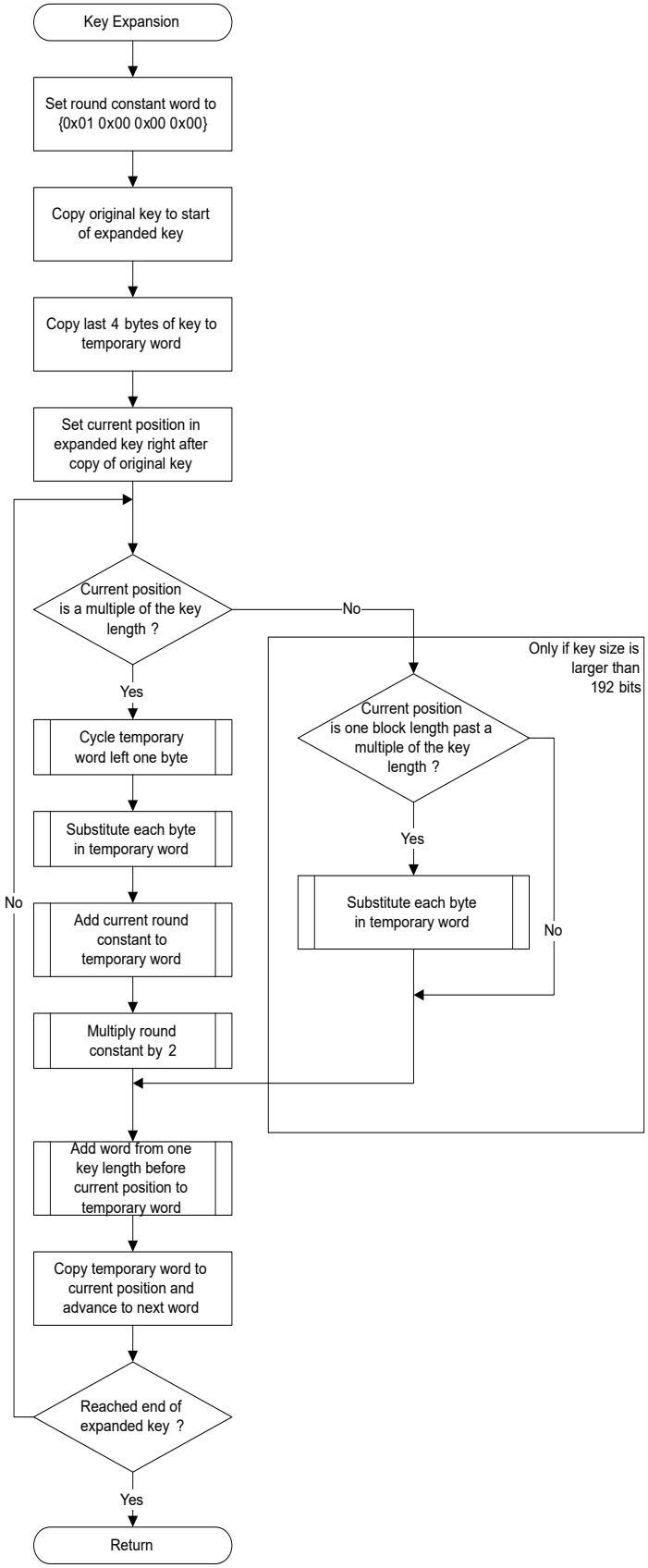


注： 解密所使用的密钥编排方案与加密相同，只是顺序相反。

5.4 密钥扩展

密钥扩展是根据原始的 128 位、196 位或 256 位加密密钥生成密钥编排方案的过程。密钥扩展的流程图如下所示：

图 5-9. 密钥扩展流程图


Key Expansion
Set round constant word to {0x01 0x00 0x00 0x00}
Copy original key to start of expanded key
Copy last 4 bytes of key to temporary word
Set current position in expanded key right after copy of original key
Current position is a multiple of the key length ?
No
Yes
Only if key size is larger than 192 bits
Current position is one block length past a multiple of the key length ?
Yes
Substitute each byte in temporary word
No
Cycle temporary word left one byte
Substitute each byte in temporary word
No
Add current round constant to temporary word
Multiply round constant by 2
Add word from one key length before current position to temporary word
Copy temporary word to current position and advance to next word
Reached end of expanded key ?
Yes
Return

该算法使用的运算上文均已介绍，例如异或加法、有限域乘法、替换和字循环。有关详细信息，请参见源代码。

注： 加密和解密的密钥扩展是相同的。因此，即使只需要解密，也会用到加密所用的 S-box。在 AVR 实现中，进行密钥扩展之前应先计算常规 S-box 所需的存储空间，之后便可直接为逆 S-box 重复使用相同的存储空间，而不必重新计算。

5.5 密码块链接——CBC

AES 是分组密码，这意味着该算法基于固定大小的数据块进行运算。加密密钥用于加密 16 字节的数据块。对于已知的输入块和常数加密密钥（尽管未知），输出块将始终相同。密码系统的攻击者可能会借此获取有用的信息。

有一些常用的方法可将相同的明文块加密为不同的密文块。其中一种称为密码块链接（CBC）。

CBC 方法可连接多个密码块，以此让前面的块影响所有后面的块。具体实现方式是先对当前明文块和上一个密文块执行异或运算，然后对异或结果（而非明文块）进行加密。这样可增加一个密文位所依赖的明文位数。

6. 软件实现与使用

本章首先探讨关于提高系统安全性的一些重要主题。这些主题是后续软件设计中做出众多决策的重要依据。

6.1 目的

本应用笔记将介绍可保护设计免遭外部入侵的相关技术。虽然在安全性方面做不到万无一失，但在构建设计时可以尽可能地提高攻破安全系统的难度。一个只要掌握基本工程技能就可轻松复制的非安全设计与只有少数高水平入侵者才能攻破的设计相比，差距非常明显。非安全设计容易遭到复制甚至逆向工程，进而导致制造商的知识产权受到侵犯，设计的市场潜力也会大打折扣。而加密设计极难攻破，令大多数入侵者望而却步，最终只能选择自食其力开发自己的产品。

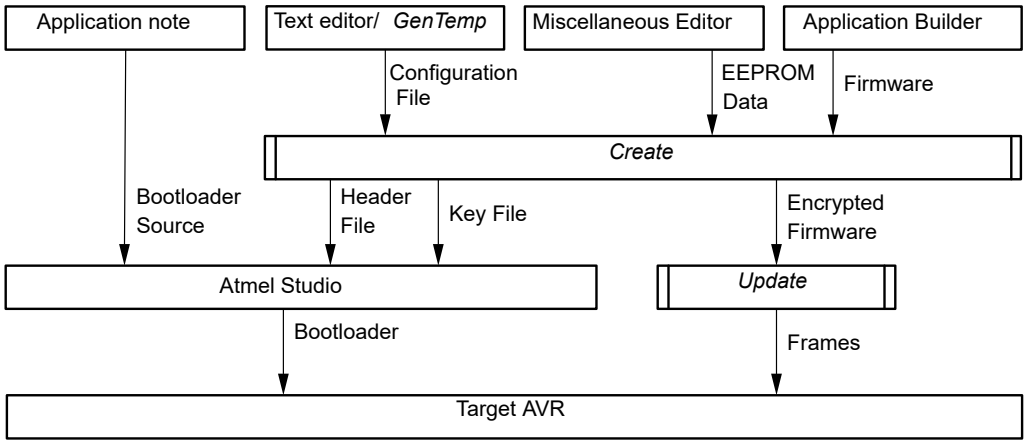
构建安全系统只有一条通用规则：尽可能地让系统设计难以攻破。入侵者在攻击系统时，会不择手段地来规避安全机制。下面例举了一些必须考虑的安全因素。

- 如果在更新固件的过程中发生断电会怎样？恢复供电后，单片机处于何种状态？锁定位和复位向量是否始终保持正确设置？
- 是否可以通过某些途径推断出明文数据的格式？为了攻破 AES，必须找到一种模式。攻击软件将必须配置为搜索已知模式，例如程序存储器开始处的中断向量以及用 0 或 1 填充的存储区等。
- 在解密过程中是否会获得反馈？任何此类反馈都会使攻击者有机可乘。例如，如果自举程序内部的解密算法为处理的每个块提供 OK/Not-OK 类型的信号，则该信号可能会反馈给攻击者。
- 是否应该以另一种顺序来发送加密帧？如果发送到自举程序的第一个帧始终包含加密文件的第一个块，则攻击者可以据此做出一些假设。例如，可以假设第一个帧映射从地址 0 开始的程序数据，或包含中断向量表。该信息可帮助攻击者细化密钥搜索。为了提高系统安全性，应以随机顺序发送帧（解密后的帧无论如何都会映射到正确的地址）。

6.2 使用概述

本小节及后续各小节将介绍如何使用和配置应用程序。该过程如下图所示。

图 6-1. 项目流程概览



主要步骤如下：

- 为目标 AVR 创建一个应用程序。如果需要，请在单独的文件中创建 EEPROM 布局。
- 创建一个包含项目相关信息的配置文件。可使用名为 **GenTemp** 的应用程序创建文件框架。
- 运行名为 **Create** 的应用程序，以创建头文件、密钥文件和加密文件。
- 使用 Atmel Studio 7 或更高版本，为目标 AVR 配置并编译自举程序
- 将自举程序下载到目标 AVR 并设置锁定和熔丝位
- 现在可以随时将加密的固件下载到 AVR

6.3 配置文件

配置文件包含用于配置项目的参数列表。下表对这些参数进行了说明。

表 6-1. 配置文件选项汇总

参数	说明	默认值	必需项
PAGE_SIZE	AVR 闪存页的大小（以十进制字节为单位）。该参数取决于器件。请参见数据手册。	N/A	是
KEY1	十六进制加密密钥的第一部分（128 位）。应为 16 个随机字节，每 8 位后插入奇校验位，因此共 18 个字节。	无：无加密	否，但强烈建议使用
KEY2	十六进制加密密钥的第二部分（64 位）。应为 8 个随机字节，每 8 位后插入奇校验位，因此共 9 个字节。如果省略，将使用 AES128。	无：使用 AES128	否，但建议使用
KEY3	十六进制加密密钥的第三部分（64 位）。应为 8 个随机字节，每 8 位后插入奇校验位，因此共 9 个字节。如果省略，将使用 AES128 或 AES192。	无：使用 AES128 或 AES192	否，但建议使用
INITIAL_VECTOR	用于链接密码块。应为 16 个十六进制随机字节。	0	否，但强烈建议使用
SIGNATURE	十六进制帧验证数据。可以是任意四个字节，但是建议随机选择这些值。	0	否
ENABLE_CRC	启用 CRC 校验：YES 或 NO。如果启用，则整个应用程序段都将被改写，并且应用程序必须先通过 CRC 校验才能启动。	否	否，但建议使用
MEM_SIZE	目标 AVR 中应用程序段的大小（以十进制字节为单位）。	N/A	是，使用 CRC 时

可以为配置文件指定任何有效的文件名。该名称稍后将作为参数提供给用于创建项目文件的应用程序。以下是 ATmega328PB 的示例配置文件。KEY1 参数是插入了奇偶校验位的 128 位密钥示例（十六进制 0123456789ABCDEF0123456789ABCDEF）。

```
PAGE_SIZE      = 128
MEM_SIZE       = 30720
CRC_ENABLE     = YES
KEY1           = EC38ECC4C11DBC16151A5E346A872AADAD36
INITIAL_VECTOR = F24D994D5DD3E9F1EEE897616C425028
SIGNATURE      = 89DBF334
```

如果不是特别了解目标 AVR，某些参数可能无法设置。下表总结了具有自举程序功能的一些现有 AVR 单片机的特性。对于下表中未列出的器件，请参见器件的数据手册。

表 6-2. AVR 特性汇总

特性	M8	M16	M162	M169	M32	M64	M128	M328PB
闪存大小（字节）	8192	16384	16384	16384	32768	65536	131072	32768
闪存页大小（字节）	64	128	128	128	128	256	256	128
闪存页	128	128	128	128	256	256	512	256
BLS（最大值）（字节）	2048	2048	2048	2048	4096	8192	8192	4096
BLS（最大值）（页）	32	16	16	16	32	32	32	32

.....（续）

特性	M8	M16	M162	M169	M32	M64	M128	M328PB
MEM_SIZE（最小值）（字节）	6144	14336	14336	14336	28672	57344	122880	28672
PAGE_SIZE（字节）	64	128	128	128	128	256	256	128

6.4 PC 应用程序——GenTemp

该应用程序用于生成配置文件模板。该应用程序将生成随机加密密钥和初始向量，而其他参数则留待用户填写（例如闪存页大小）。建议始终先使用该应用程序创建模板。

该应用程序的用法如下：

```
gentemp FileName.ext
```

FileName.ext 是要创建的配置文件的名称。生成文件后，可以使用任何纯文本编辑器对其进行编辑。

6.5 PC 应用程序——Create

该应用程序从配置文件中读取信息，然后为自举程序生成密钥和头文件。此外，还用于为固件加密。通常，该应用程序至少运行两次：

1. 为自举程序生成密钥和头文件。
2. 为新固件加密。

注： 在生成项目文件和对固件进行编码时，务必使用相同的加密信息（配置文件）。否则，自举程序可能没有正确的加密密钥集，无法解密数据。另应注意，可以使用配置文件中的信息来解密加密的固件。因此，必须确保配置文件始终保持安全，并且首次使用后不得修改。

6.5.1 命令行参数

下表列出了可用的命令行参数。

表 6-3. 命令行参数汇总

参数	说明
-c <filename.ext>	配置文件的路径。
-d	设置后，将在写入之前删除每个闪存页的内容。如果没有具体的写入操作，将保留先前的数据。
-e <filename.ext>	EEPROM 文件（写入 EEPROM 的数据）的路径。
-f <filename.ext>	闪存文件（写入应用程序段的代码）的路径。
-h <filename.ext>	输出头文件的名称。该文件稍后包含在自举程序中。
-k <filename.ext>	输出密钥文件的名称。该文件稍后包含在自举程序中。
-l [BLB12] [BLB11] [BLB02] [BLB01]	要设置的锁定位。这些锁定位在所有数据传输完毕之后，控制信号传输到更新后的应用程序之前设置。
-n	无意义。将随机数量的无意义记录添加到加密文件中。由于自举程序会忽略无意义的记录，因此该设置不会影响应用程序，只会影响输出文件的可预测性。
-o <filename.ext>	输出文件名。这是加密的文件，需要更新时可分发和发送给目标。

6.5.2 首次运行

首次运行时通常仅生成自举程序的密钥和头文件。使用命令行参数请求生成密钥和头文件。例如：

```
create -c Config.txt -h BootLdr.h -k AESKeys.inc
```

密钥和头文件必须复制到自举程序的项目目录中，并包含在自举程序代码中。

注： 自举程序项目文件默认使用上述文件名，即 BootLdr.h 和 AESKeys.inc。建议不要更改这些文件名。

6.5.3 后续运行

后续运行时，将使用应用程序对固件进行编码。在加密之前，必须对源文件进行编译，汇编并链接到一个代码段文件和/或一个 EEPROM 段文件。文件必须为 Intel®十六进制类型。

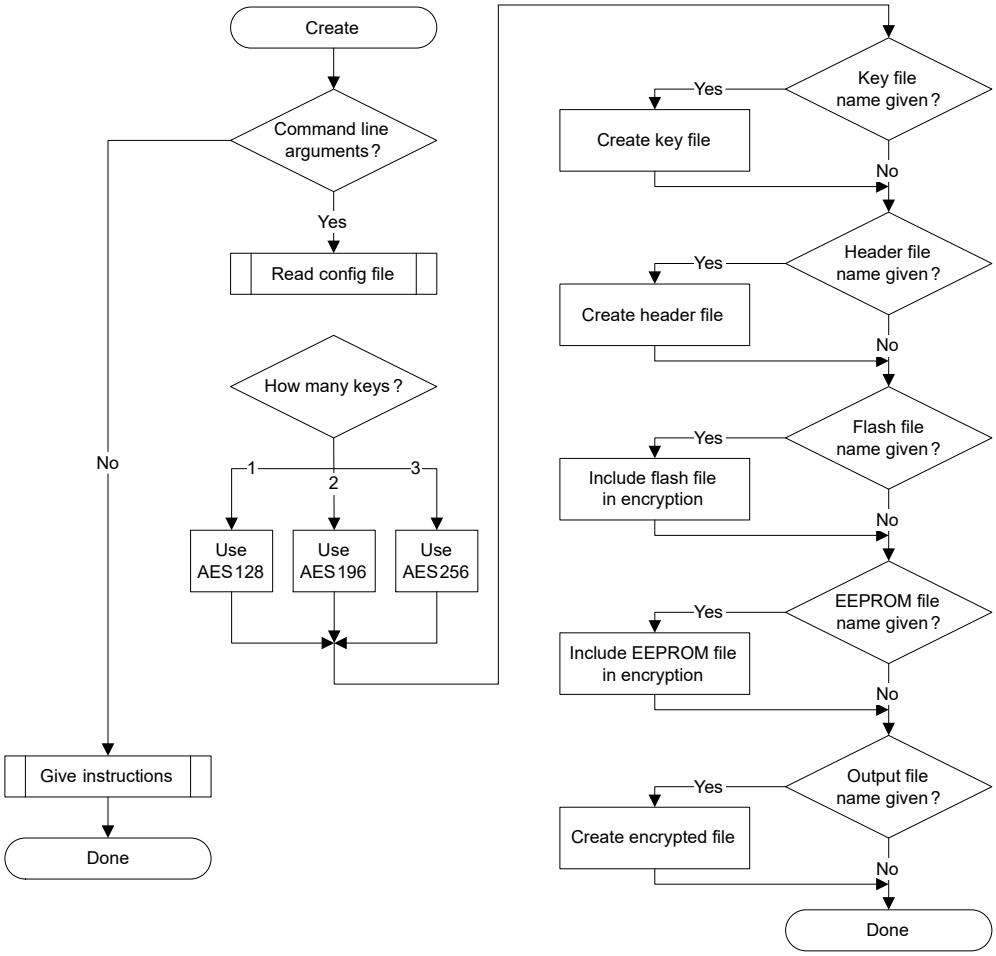
文件名通过命令行设置。加密文件将根据配置文件中的数据生成。例如：

```
create -c Config.txt -e EEPROM.hex -f Flash.hex -o Update.enc -l BLB11 BLB12
```

应用软件和 EEPROM 数据文件将合并为一个加密文件。

6.5.4 程序流

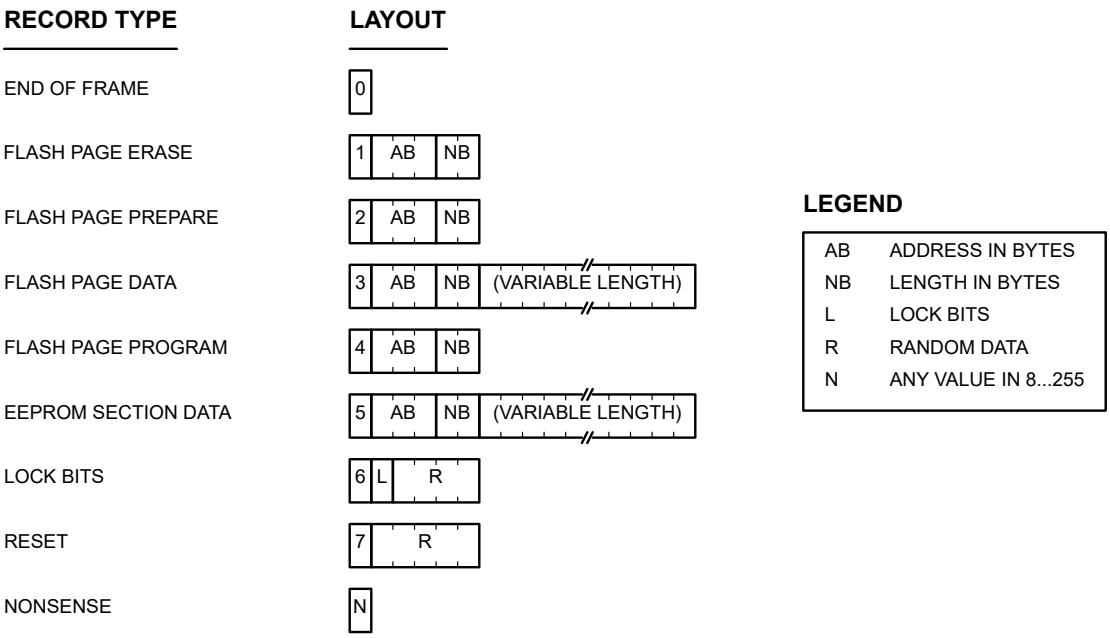
图 6-2. Create 应用程序的流程图



6.5.5 加密文件

闪存和 EEPROM 文件经过加密后存储在一个目标文件中。但在加密之前，数据会被分为多个记录。记录共有七种类型，如下图所示。

图 6-3. 加密文件的记录类型

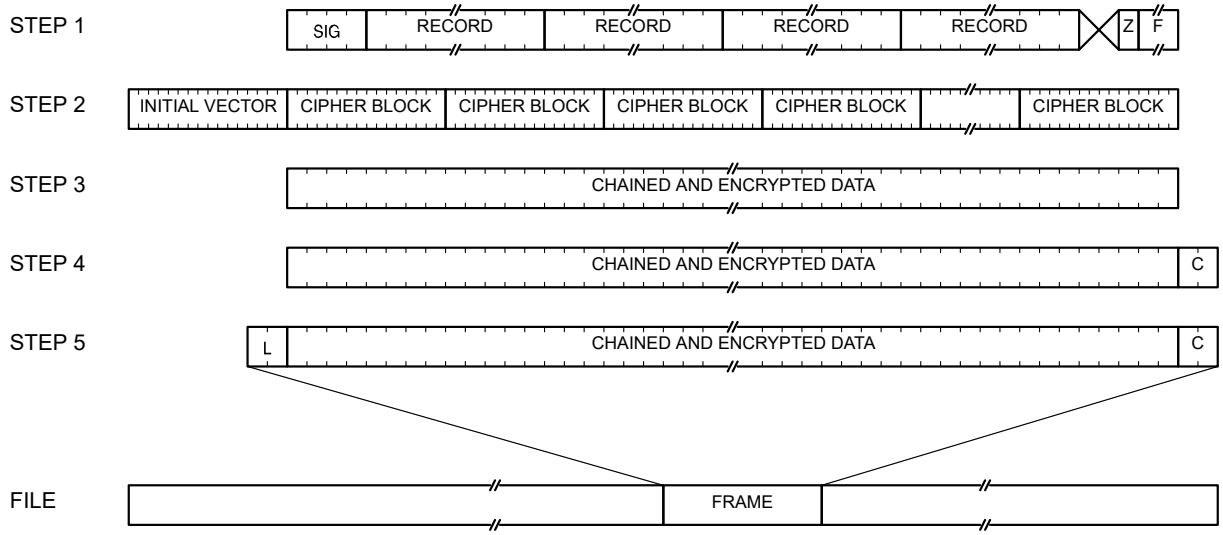


记录中的第一个字节为记录类型。应用程序数据被细分为记录类型 1、2、3 和 4（即擦除、准备、装载和将缓冲区页写入闪存）。EEPROM 段的数据为记录类型 5。锁定位以记录类型 6 的形式发送。记录类型 0 和 7 分别用于结束帧和进行传输。

所有其他记录（即标识符大于 7 的记录）均为无意义类型。启用该选项后（见 Create 工具），会将随机数量的无意义记录置于文件中的随机位置。

输出文件的创建过程如下图所示。

图 6-4. 创建加密文件



具体步骤如下（步骤编号对应于上图）：

1. 将数据格式化为记录，这些记录排列在帧签名（SIG）之后。添加 0（Z）来标记帧末尾，并用随机数据（F）填充帧以创建一个大小为 16 字节倍数的帧。

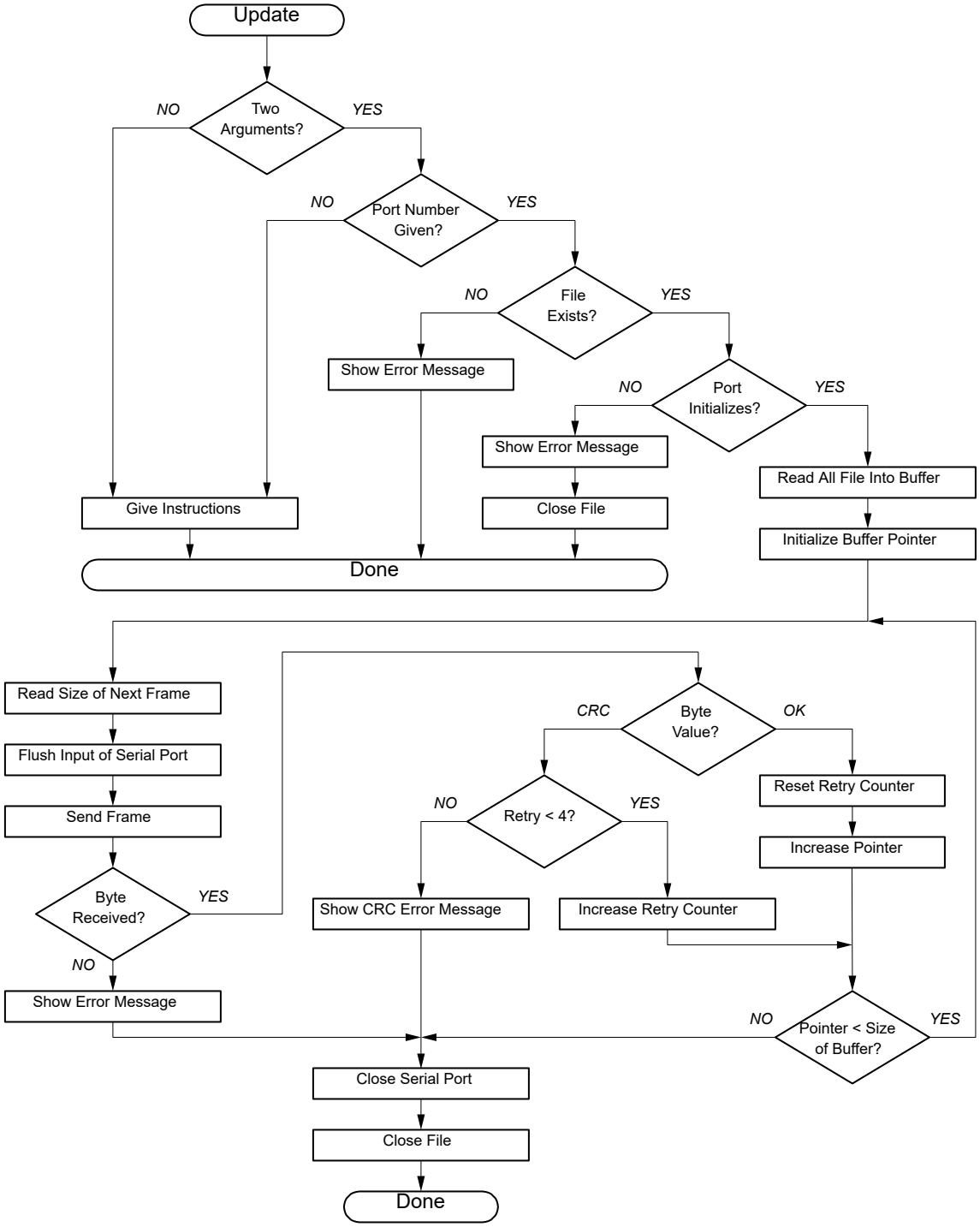
2. 将初始向量附加到帧上。在第一个帧中，向量等于配置文件中给定的向量。在随后的帧中，初始向量等于前一个帧的最后一个密码块。
3. 对初始向量和密码块进行链接和加密。随后从帧中删除初始向量。
4. 计算 CRC-16 校验和（C）并将其添加到帧中。
5. 计算帧的长度（L），并将其保存在帧的开头（不包括长度信息）。

帧将写入输出文件，然后重复上述过程，直到处理完所有数据为止。

6.6 PC 应用程序——Update

该应用程序用于将加密文件发送到目标。可以通过 PC 上的串行端口直接将数据发送到目标硬件上的 USART。程序流如下所示。

图 6-5. 项目流程概览



Update 应用程序用于读入使用 Create 应用程序生成的文件。该文件由一个或多个串连加密数据帧组成。该应用程序一次发送一个数据帧，并且每次都会暂停片刻以等待自举程序的应答。只有收到应答后才会发送下一个帧，否则应用程序将重新发送当前帧或关闭通信。

Update 应用程序通过命令行运行。下表列出了命令提示符参数。

表 6-4. Update 应用程序的命令行参数

参数	说明
<filename.ext>	要传输的加密文件的路径
-COM*n*	串行端口，其中 *n* 是串行端口号
-baudrate	波特率，其中*波特率*是实际波特率值

例如：

```
update blinky.ext -COM1 -115200
```

注意，更新系统仅更新应用程序和 EEPROM 文件中指示的闪存和 EEPROM 部分。如果启用了应用程序段的 CRC 校验，或在 Create 工具中选择了擦除选项，则在编程前会清空所有应用程序存储器。

7. 硬件设置

必须正确设置目标硬件，然后才能将加密的固件发送到自举程序。在本应用笔记中，假设使用 ATmega328PB Xplained Mini 板进行评估。以下各节介绍了有关配置、编程步骤和调试的详细信息，其中仅涵盖启动和运行 ATmega328PB Xplained Mini 所需的基本信息。更多信息，请参见 ATmega328PB Xplained Mini User Guide。

7.1 连接 ATmega328PB Xplained Mini 工具包

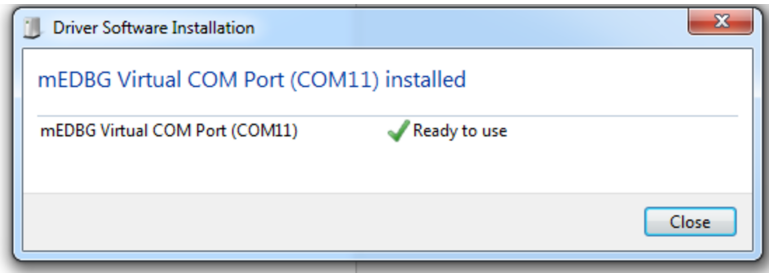
本节旨在帮助用户通过 Atmel Studio 7 连接 ATmega328PB Xplained Mini。

1. 下载并安装 Atmel Studio 7 或更高版本。
2. 启动 Atmel Studio。
3. 将 ATmega328PB Xplained Mini 连接到 USB 端口，随后它将显示在 Atmel Studio 中。

7.1.1 自动识别 Xplained Mini 工具包

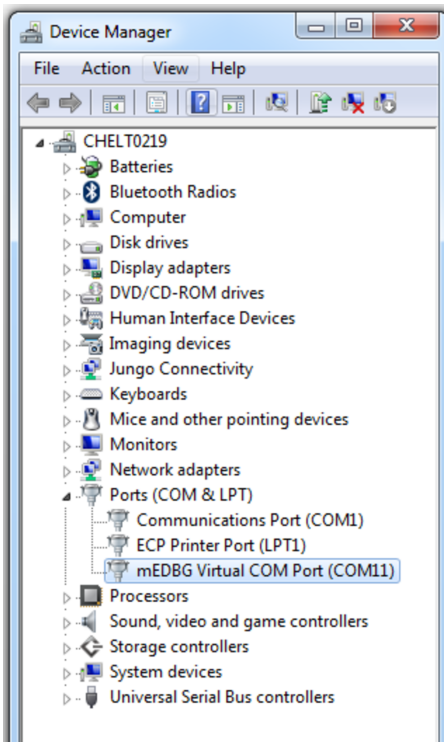
- 将 ATmega328PB Xplained Mini 工具包连接到 PC 后，Windows®任务栏将弹出一条消息，如下所示：

图 7-1. ATmega328PB Xplained Mini 驱动程序安装



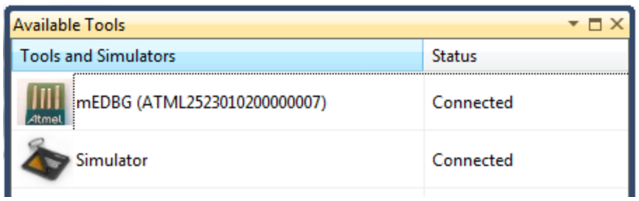
- 如果驱动程序安装成功，设备管理器中将列出 EDBG，如下所示：

图 7-2. mEDBG 驱动程序安装成功



- 打开 Atmel Studio，然后转到 **View → Available Tools**（视图 → 可用工具）。EDBG 应在工具中以 mEDBG 形式列出，并且工具状态应显示为 **Connected**（已连接）。这表明该工具正在与 Atmel Studio 正常通信。

图 7-3. Available Tools 下的 mEDBG



7.1.2 将 ATmega328PB Xplained Mini UART 连接到 mEDBG COM 端口

1. 将 mEDBG USB 连接到 PC。
2. 使用设备管理器查找 COM 端口号。
3. 默认的 COM 端口设置为 9600 波特率 N81。可使用设备管理器根据应用程序来更改 COM 端口设置。该应用程序已针对 115200 波特率进行了测试。

7.2 编程和调试

本节将介绍如何使用 mEDBG 对 ATmega328PB Xplained Mini 工具包进行编程和调试。

7.2.1 使用 mEDBG 对 ATmega328PB Xplained Mini 进行编程

1. 将 mEDBG USB 连接到 PC。
2. 转到 Atmel Studio：单击 **Tools**（工具），选择 **Device Programming**（器件编程），然后选择连接的 mEDBG 作为**工具**，ATmega328PB 作为**器件**，ISP 作为**接口**，最后单击 **Apply**（应用）。
3. 选择 **Memories**（存储器）并找到.hex 或.elf 源文件，然后单击 **Program**（编程）。
4. 如果源文件包含熔丝设置，应转到 **Production file**（生产文件），然后上传.elf 文件并对熔丝进行编程。

注： 如果 ISP 编程失败，则可能是因为启用了 debugWIRE。有关如何禁止 debugWIRE 模式的信息，请参见 7.2.2 使用 mEDBG 对 ATmega328PB Xplained Mini 进行调试。

7.2.2 使用 mEDBG 对 ATmega328PB Xplained Mini 进行调试

1. 启动 **Atmel Studio**。
2. 将 mEDBG USB 连接到 PC。
3. 打开您的项目。
4. 在 **Project**（项目）菜单中，选择项目属性页。选择 **Tools** 选项卡，然后选择 mEDBG 作为调试器，debugWIRE 作为接口。
5. 在 **Debug**（调试）菜单中，单击 **Start Debugging and Break**（开始调试并中断）。
6. 如果未启用 ATmega328PB 中的 DWEN 熔丝，则 Atmel Studio 将显示一条错误消息，请单击 YES（是），允许 Studio 通过 ISP 接口设置熔丝。
7. 调试会话会先运行到 main 函数中的中断点，然后就可以开始调试。
8. 退出调试模式时，请选择 **Debug** 菜单中的 **Disable debugWIRE and Close**（禁止 debugWIRE 并关闭）。这将禁止 DWEN 熔丝。

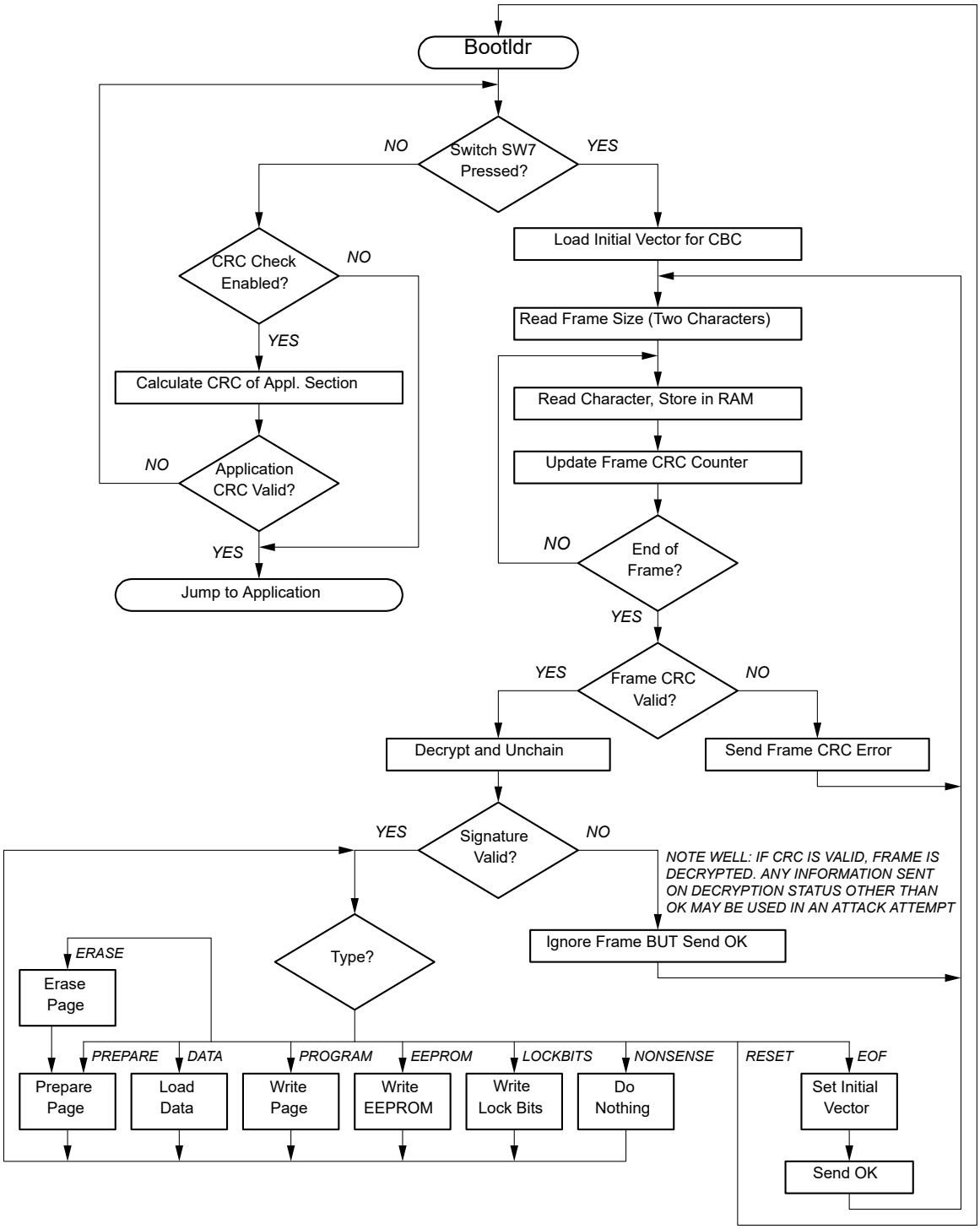
注：

1. 如果未选择 **Debug** 菜单中的 **Disable debugWIRE and Close** 退出调试模式，DWEN 熔丝将启用，目标仍处于调试模式，这样就无法使用 SPI（ISP）接口对目标进行编程。
2. 本应用笔记随附的自举程序代码已在代码长度方面进行了优化，因此可能无法进行调试。取消优化将增大代码长度，可能会超出自举程序段的限制。因此，建议仅使用编程模式。

8. AVR 自举程序

必须先将自举程序存放在目标 AVR 中，然后才能使用加密的固件来更新器件。自举程序将与 PC 通信，并且能够对 EEPROM 以及闪存应用程序区域进行编程。本应用笔记随附的自举程序是使用 Atmel Studio 7 创建的。程序流如下图所示。

图 8-1. AVR 自举程序的流程图



8.1 密钥和头文件

在编译自举程序之前，需要设置一些参数。首先，必须将 PC 应用程序 Create 生成的加密密钥和目标头文件复制到自举程序目录中。在自举程序源代码中使用**#include** 伪指令进行引用时，将包含这些文件。

8.2 项目文件

本应用笔记随附 ATmega328PB 的器件特定项目文件。

对于其他 AVR 器件，请按照下节中给出的步骤对项目文件进行修改。

8.3 Atmel Studio 和 IAR 设置

本节列出了 Atmel Studio 和 IAR™的常用设置。

1. 基本熔丝设置：
   1.1. 必须启用引导复位向量。
   1.2. 将低熔丝字节设置为使用 16 MHz 的外部时钟，以确保从 Atmel START 下载的 ATmega328PB 示例（波特率 38400）正常工作。
2. 必须使用 reg.h 文件中的 SPMREG（见下表）宏来定义 SPMCSR 寄存器地址。如果寄存器位于间接存储区中，则该寄存器地址将发生变化。有关该寄存器的地址，请参见数据手册的寄存器汇总部分。

图 8-2. 在 reg.h 中设置 SPMREG 地址

```
#ifndef REG_H_
#define REG_H_

// Bit values macros
#define EEMWE 2 // position 2 and value 4   in EEMCR
#define EEWE 1  // position 1 and value 2    in EEMCR

// Bits and their position in SPMREG
#define SPMEN 0 // position zero in SPMCR
#define BLBSET 3
#define PGERS 1
#define PGWRT 2

#ifdef __RAMPZ__
#define _RAMPZ_ 0x3B
#endif

#define SPMREG 0x37
#define SREG 0x3F
#define EEARL 0x21
#define EEARH 0x22
#define EECR 0x1F
#define EEDR 0x20

#endif /* INCFILE1_H_ */
```

3. 必须根据器件数据手册对 reg.h 文件中定义的其他寄存器地址进行修改。
4. 对于闪存容量大于 64 KB 的器件，必须在编译器和汇编器中定义符号**__RAMPZ__**（见下表）。
5. 对于 **SPMCSR** 寄存器位于直接存储器访问区域之上（即，SPMCSR 寄存器位于 0x3F 之上）的器件，必须在汇编器中定义符号**__MEMSPM__**（见下表）。例如，ATmega128 的 SPMCSR 位于 0x68 处，因此需要定义符号才能对其进行访问。

6. 对于闪存容量大于 64 KB 的器件，必须在编译器中定义符号 **BOOT_ADDR=<bootloader section start address>**（见下表）。

表 8-1. 符号设置汇总

设置	M8	M16	M32	M64	M128	M256	M162	M169	M328PB	M168PA
__RAMPZ__	无	无	无	无	有	有	无	无	无	无
__MEMSPM__	无	无	无	无	有	无	无	无	无	无
BOOT_ADDR	无	无	无	无	0x1E000	0x3E000	无	无	无	无
BOOT_START(IAR)	0x1800	0x3800	0x7000	0xE000	0x1E000	0x3E000	0x3800	0x3800	0x7000	0x3800
.text(Atmel Studio)	0x1800	0x3800	0x7000	0xE000	0x1E000	0x3E000	0x3800	0x3800	0x7000	0x3800
SPMREG	0x37	0x37	0x37	0x37	0x68	0x37	0x37	0x37	0x37	0x37

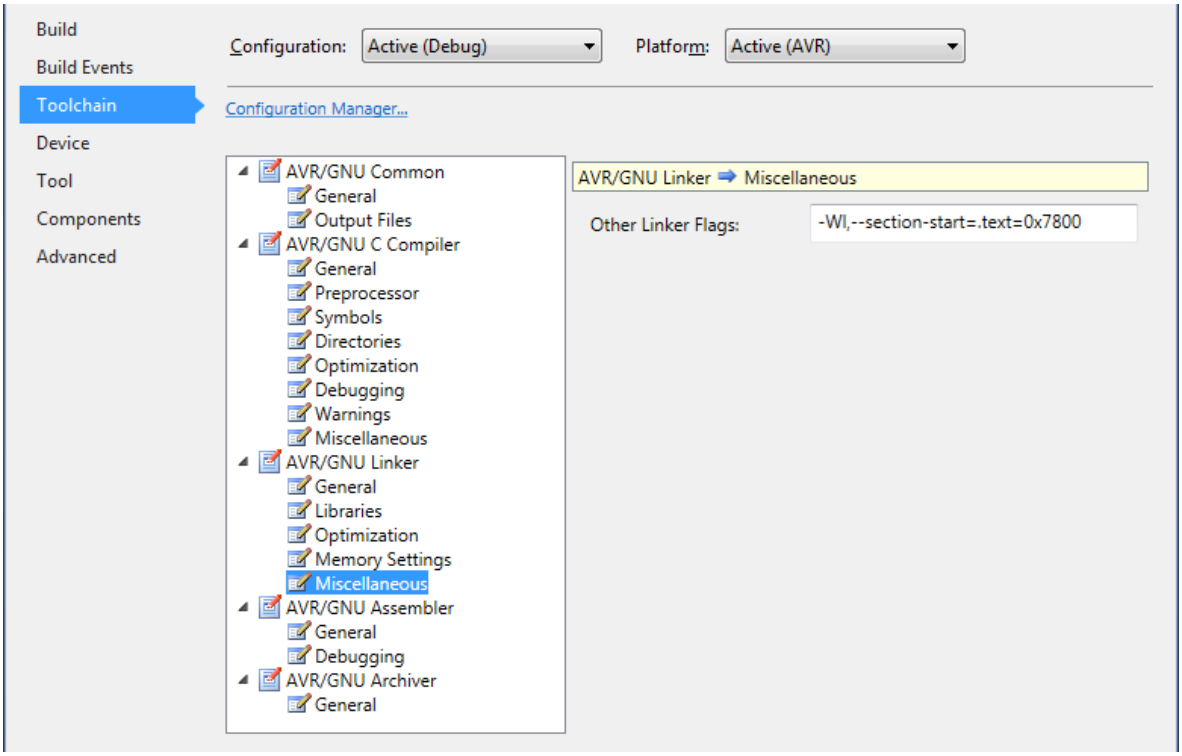
注： 上表中的 BOOT_ADDR、BOOT_START 和.text 的值是引导区域的最大值。可根据实际的引导区域大小来更改这些值。

8.3.1 Atmel Studio 设置

在 Atmel Studio 中配置特定器件时，需要进行如下所示的编译器、链接器和汇编器设置。

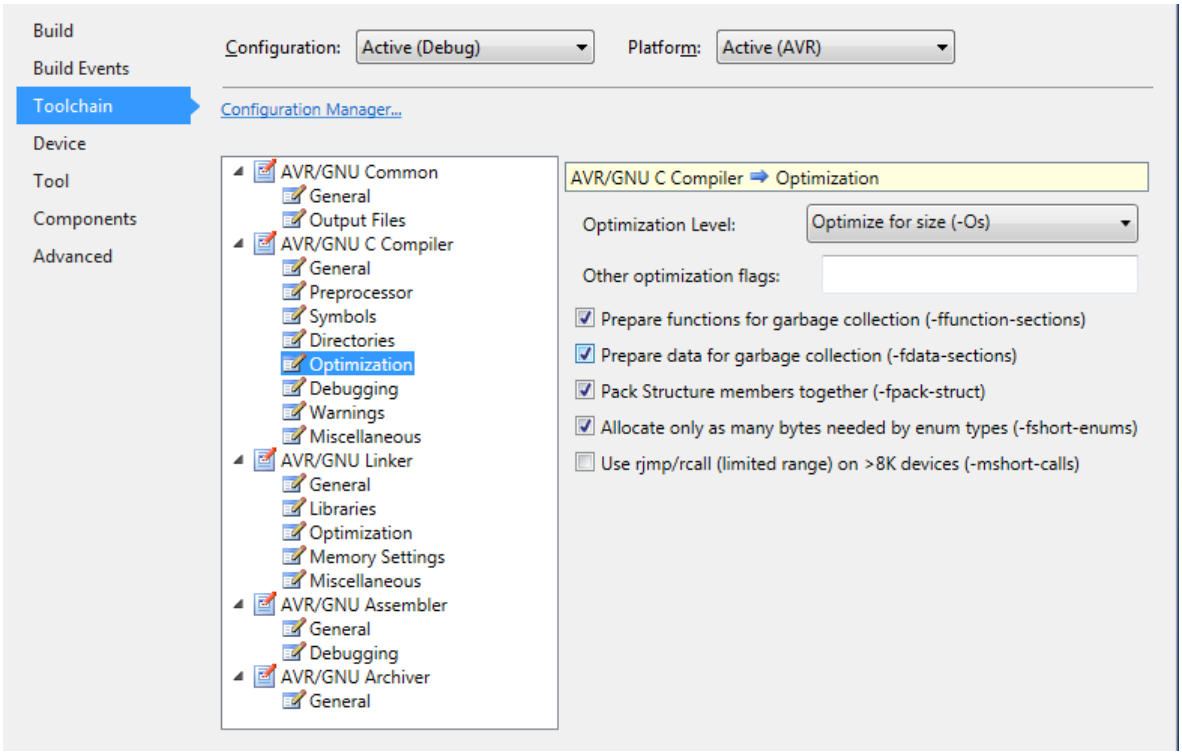
1. 必须在链接器设置中定义自举程序起始地址.text（见表 8-1），以将代码链接到自举程序起始地址（如下图所示）。起始地址应以字节地址（字地址乘以 2）的形式指定。例如，如果 ATmega328PB 的自举程序字地址为 0x3C00，则对应的字节地址为 0x3C00 * 2 = 0x7800。

图 8-3. 设置自举程序地址代码链接地址



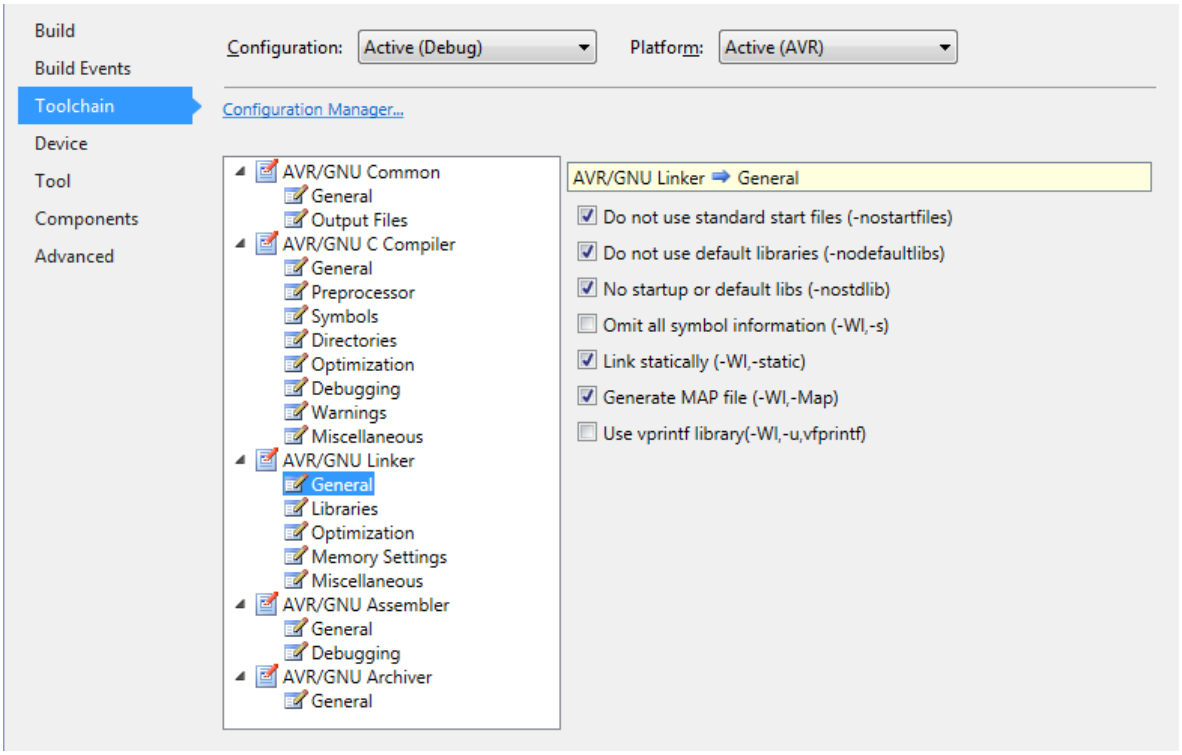
2. 优化设置（如下图所示）。

图 8-4. 优化设置



3. 其他链接器设置（如下图所示）。

图 8-5. 其他链接器设置

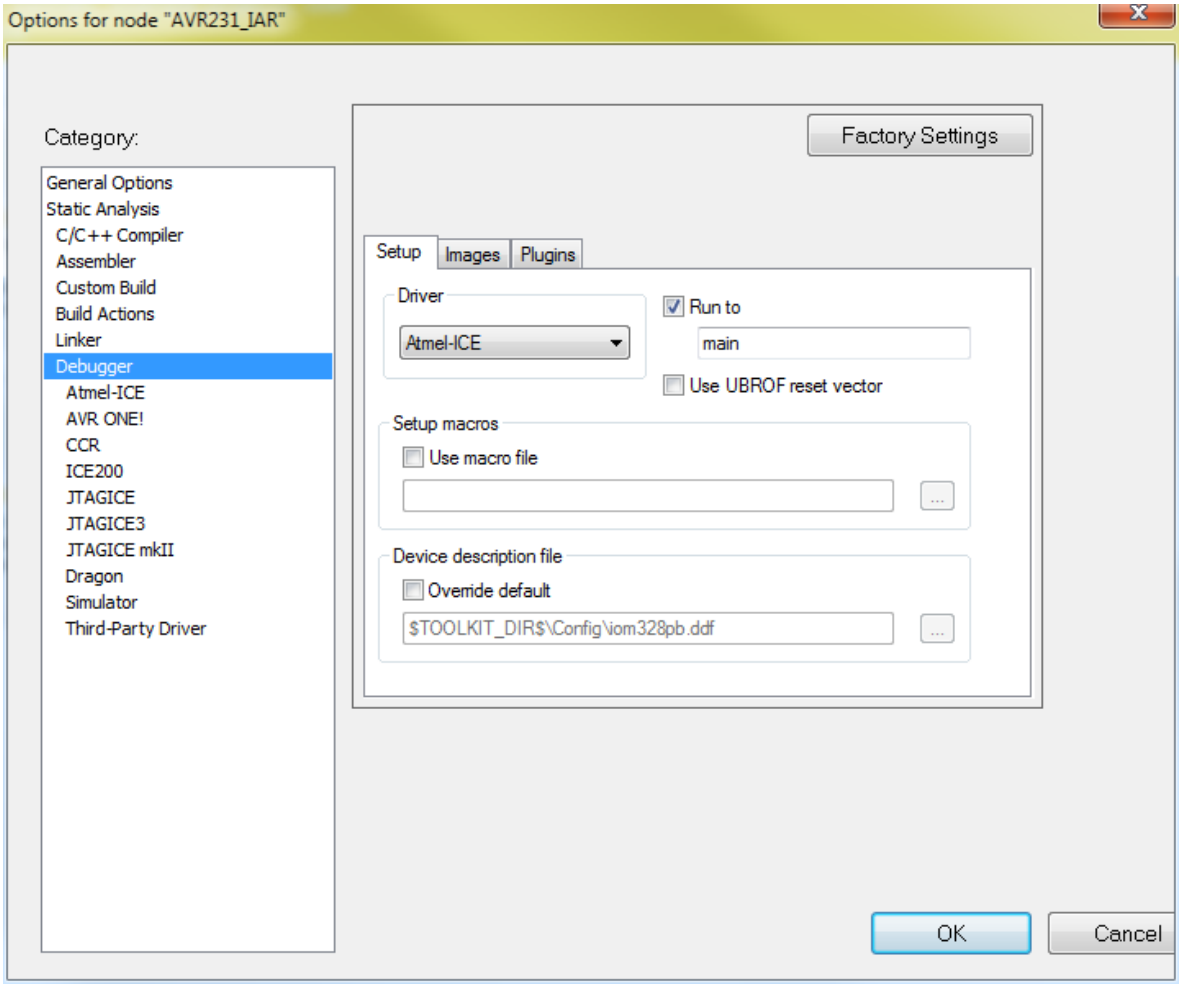


8.3.2 IAR 设置

在 IAR 中配置特定器件时，需要进行如下所示的编译器、链接器和汇编器设置。

1. 在 ATmega328PB Xplained Mini 工具包中使用 mEDBG 时，应在 Debugger Driver（调试器驱动程序）设置中选择 Atmel-ICE（如下图所示）。

图 8-6. 调试器驱动程序



2. 应使用用户定义的.xcl 文件覆盖默认链接器文件。请按照以下步骤操作，以确保链接器正常工作：

   2.1. 根据器件将模板链接器文件（例如，C:\Program Files (x86)\IAR Systems\Embedded Workbench 7.4\avr\src\template\cfgm328pb.xcl）复制到项目中，然后按照个人偏好为其重命名。

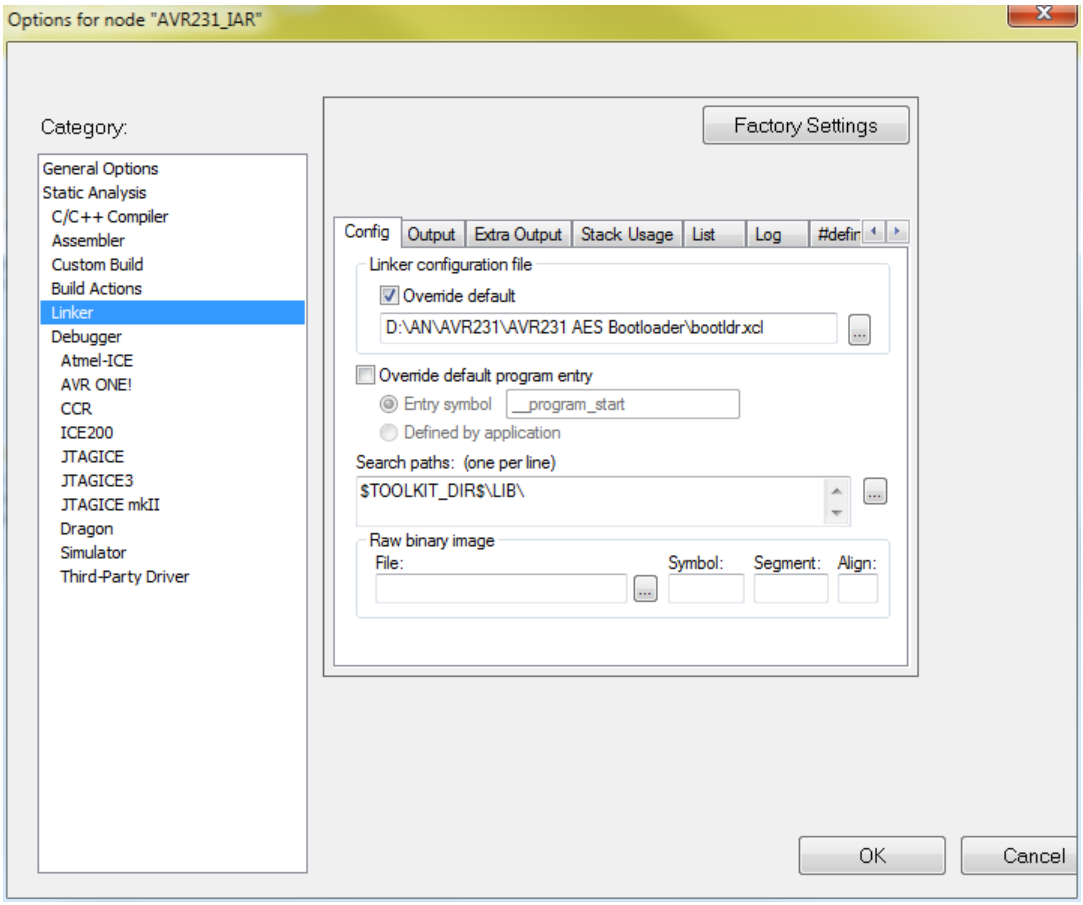
   2.2. 打开该文件，将以下内容粘贴到原始内容下方并保存。

```
-ca90
-w29


//=============================================================================
// Interrupt vectors

-Z(CODE)INTVEC=BOOT_START-(BOOT_START+_..X_INTVEC_SIZE-1)
//-H1895
-h(CODE)BOOT_START-(BOOT_START+_..X_INTVEC_SIZE-1)

//=============================================================================
// Code memory

-Z(CODE)NEAR_F,HUGE_F,SWITCH,INITTAB,DIFUNCT,CODE=BOOT_START-_..X_FLASH_END
-Z(FARCODE)FAR_F,FARCODE=BOOT_START-_..X_FLASH_END


//=============================================================================
// RAM
-Z(DATA)NEAR_I,NEAR_Z=_..X_SRAM_BASE-_..X_SRAM_END
```

```
-Z(DATA)RSTACK+_..X_RSTACK_SIZE=(_..X_SRAM_END-_..X_RSTACK_SIZE+1)-_..X_SRAM_END
-Z(DATA)CSTACK+_..X_CSTACK_SIZE=(_..X_SRAM_END-_..X_RSTACK_SIZE-_..X_CSTACK_SIZE
+1)-(_..X_SRAM_END-_..X_RSTACK_SIZE)
//-Z(DATA)RSTACK+40=(_..X_SRAM_END-40+1)-_..X_SRAM_END
//-Z(DATA)CSTACK+300=(_..X_SRAM_END-40-300+1)-(_..X_SRAM_END-40)
//-Z(DATA)TINY_I,TINY_Z,TINY_N=RAM_BASE-FF
//-Z(DATA)TINY_I,TINY_Z,TINY_N=RAM_BASE-100
```

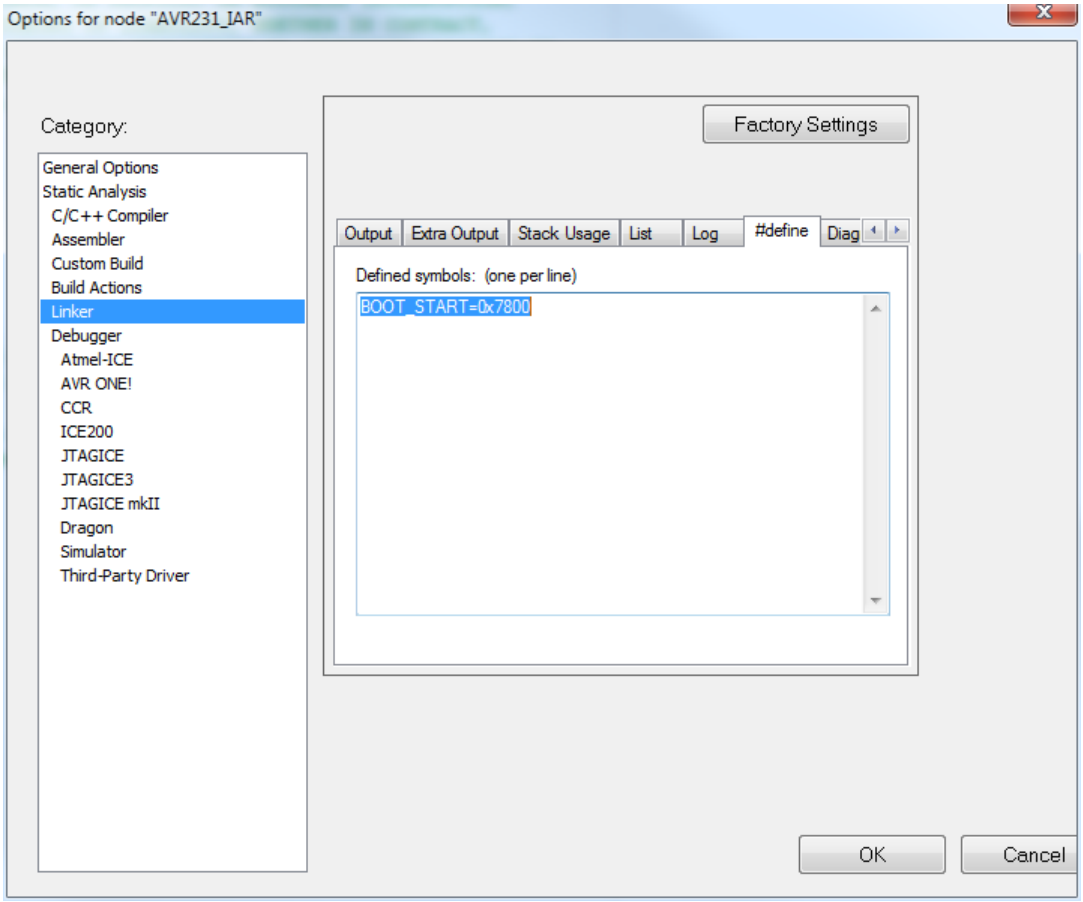
2.3. 使用该文件覆盖默认链接器文件（如下图所示）。

图 8-7. 覆盖默认链接器文件



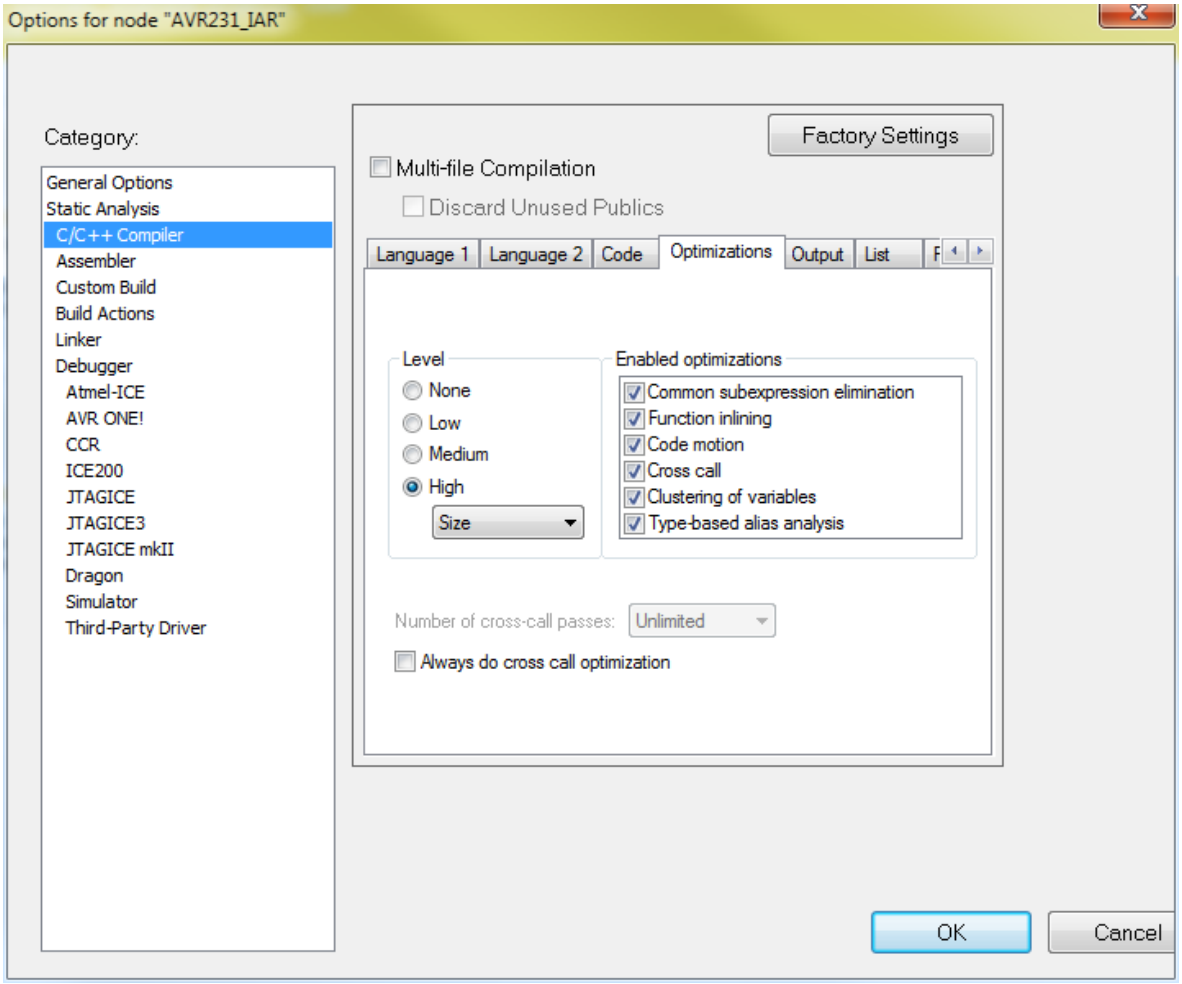
2.4. 根据器件定义 BOOT_START（见表 8-1），如下图所示。

图 8-8. 链接器#define



3. 优化设置（如下图所示）。

图 8-9. 优化设置



8.4 安装自举程序

使用 Atmel Studio 7 或更高版本来编译自举程序，然后将其下载到目标。在运行自举程序之前，必须配置以下熔丝位：

- 自举程序段的大小。设置熔丝位，使段大小与 BOOT_SIZE 设置匹配（如前文所述）。请注意，BLS 通常以字为单位，但 BOOT_SIZE 参数以字节为单位。
- 引导复位向量。必须启用引导复位向量。
- 振荡器选项。振荡器熔丝位取决于器件，可能需要进行配置（影响 USART）。

注： 请特别注意要正确设置振荡器选项。调整稍有不当都可能会导致通信失败。

下表列出了推荐的熔丝位设置。有关器件相关熔丝位的详细说明，请参见器件数据手册。

表 8-2. 推荐的熔丝位

	M8、M8515、M8535、M16、M162、M169、M32 和 M64	M128	328PB
BOOTSZ1:0	0:0	0:1	0:1
BOOTRST	0	0	0

注： “0”表示已编程，“1”表示未编程。

建议通过编程锁定位来保护应用程序存储器和自举程序，但前提是熔丝位已完成设置。可以使用 Microchip IDE（Atmel Studio 7 或更高版本）对锁定位进行编程。如果加密固件时已将 BLS 锁定位定义为命令行参数，则在更新固件时也会设置 BLS 锁定位。推荐的锁定位设置如下：

- 存储器锁定位：应设置为防止对存储器进行未授权访问。请注意，当存储器被锁定后，如果不擦除器件就无法通过系统编程进行访问。
- 自举程序段的保护模式：SPM 和 LPM 不得写入或读取 BLS。这样可以防止应用程序段中的固件损坏自举程序，并保护解密密钥。
- 应用程序段的保护模式：访问应用程序段时，不得为 SPM 或 LPM 设置任何限制；否则，自举程序无法对其进行编程。

注： 请注意，如果器件未正确锁定，就可以通过 ISP 接口访问存储器，加密固件就变得毫无意义。

下表列出了推荐用于当前 AVR MCU 的锁定位设置。有关锁定位的详细说明，请参见器件数据手册。

表 8-3. 推荐的锁定位

	M8、M8515、M8535、M16、M162、M169、M32、M64 和 M128	328PB
BLB12 : BLB11	0 0	0 0
BLB02 : BLB01	1 1	1 1
LB2 : LB1	0 0	0 0

8.5 性能

以下各小节概述了有关执行时间和代码长度方面的系统性能。

8.5.1 执行时间

目标器件接收、解码和编程数据所需的时间取决于以下因素：

- 文件大小。数据越多，所需的时间越长。
- 波特率。传输速度越高，传输时间越短。
- 目标 AVR 的速度。时钟频率越高，解码时间越短。
- 闪存页的编程时间。这是器件常数，无法更改。
- 密钥大小。AES128 的解密速度快于 AES256。但 AES192 慢于 AES256，这是因为 192 不是 2 的幂。
- 其他设置。例如，应用程序段的 CRC 校验需要很短的时间。

8.5.2 代码长度

使用编译器的最高优化设置时，自举程序将会很好地适应 2 KB 闪存。

注意，如果未提供加密密钥，则将在没有 AES 的情况下编译自举程序。本应用笔记所述的自举程序随后将作为标准自举程序系统执行，并且可在任何支持自举程序的 AVR 上使用。

9. 总结

本应用笔记介绍了如何将数据安全地传输到具有自举程序功能的 AVR 单片机。文中还重点介绍了构建安全系统时应采用的相关技术。为了提高 AVR 设计的安全性，应考虑以下问题。

实现支持以加密形式进行下载的自举程序。首次安装自举程序时（制造阶段），必须配备解密密钥（用于将来更新固件）。之后可采用加密形式分发固件，保护内容免受外界攻击。

使用 AVR 锁定位来保护应用程序段和自举程序段。只要设置锁定位，即使读取器件也无法检索存储器内容。如果未设置锁定位，固件加密将毫无用处。

分发固件之前先加密。由于没有相应的解密密钥，加密的软件一旦离开单片机就变得毫无价值。

保护加密密钥。加密密钥只能存储在以下两个位置：已由锁定位保护的自举程序以及制造商的固件开发平台。

链接加密数据。链接数据时，每个加密块都依赖于前一个块。因此，即使是相同的明文块也会产生不同的加密输出。

避免在固件中使用标准、可预测的模式。大多数程序都具有通用框架，而所有可预测的模式（例如，从跳转到低地址开始的中断向量表）都只会为入侵者提供便利。另外，应避免用常数来填充未使用的存储区。

隐藏方法。无需提及使用的算法或密钥长度。让入侵者对系统了解得越少越好。有些人或许会说，了解加密方法可以抵御某些攻击者，但是对加密方法一无所知会增加攻破难度，令更多的攻击者知难而退。

如果需要，还可以使用自举程序擦除应用程序段。许多攻击者会尝试使器件脱离其正常工作环境，然后在黑客平台上为其供电。例如，当自举程序检测到 LCD 缺失或存储器中存在 CRC 错误时，就会启动对整个存储器（包括自举程序段和解密密钥）的完全擦除操作。

在无法或不可能使用外部通信通道进行更新的应用中，可以将固件存入其中一个 CryptoMemory®器件。存储器可以采用可移除智能卡，这种智能卡可在需要升级时轻松插入器件的插槽中。单片机可以在启动时检查是否存在 CryptoMemory，并根据需要检索固件升级。

使用安全的硬件。如果硬件存在结构性缺陷，再强大的加密协议也都毫无用处。这种情况下，系统不会报告 AVR 单片机存在安全性问题。

提高安全性的方法还有很多，但最终目的都是为了给设计者指引一个正确的方向。千万不要低估对手的智商或耐心。

10. 从 Atmel | START 获取源代码

示例代码可通过 Atmel | START 获得，Atmel | START 是一种基于 Web 的工具，可通过图形用户界面（Graphical User Interface，GUI）配置应用程序代码。可以通过下面提供的直接示例代码链接或 Atmel | START 起始页上的 *Browse examples*（浏览示例）按钮，下载 Atmel Studio 和 IAR Embedded Workbench®对应的代码。

Atmel | START 网页： http://start.atmel.com/

示例代码

- AVR231 AES 自举程序：
  - http://start.atmel.com/#example/Atmel:AVR231_AES_Bootloader:0.0.1::Application:AVR231_AES_Bootloader:

有关详细信息和示例项目的相关信息，请单击 Atmel | START 中的 *User guide*（用户指南）。*User guide* 按钮可以在该网页中找到，方法是在 Atmel | START 项目配置器中的仪表板视图中单击项目名称。

Atmel Studio

在 Atmel | START 的示例浏览器中单击 *Download selected example*（下载所选示例），下载 Atmel Studio 对应的代码并保存为`.atzip` 文件。要从 Atmel | START 下载文件，单击 *Export project*（导出项目），然后单击 *Download pack*（下载数据包）。

双击下载的`.atzip` 文件，将项目导入到 Atmel Studio 7.0。

IAR Embedded Workbench

有关如何在 IAR Embedded Workbench 中导入项目的信息，请打开 Atmel | START User Guide（Atmel | START 用户指南），选择 *Using Atmel Start Output in External Tools*（使用外部工具中的 Atmel Start 输出），然后选择 *IAR Embedded Workbench*。单击 Atmel | START 起始页右上角的 *Help*（帮助）或项目配置器中右上角的 *Help And Support*（帮助和支持），均可找到 Atmel | START 用户指南的链接。

11. 参考资料

- ATmega328PB data sheet（http://www.microchip.com/wwwproducts/en/atmega328pb）
- ATmega328PB Xplained Mini kit（http://www.microchip.com/developmenttools/productdetails.aspx?partno=atmega328pb-xmini）
- Atmel Studio（http://www.atmel.com/tools/atmelstudio.aspx?tab=overview）
- Atmel START（http://start.atmel.com）
- AT10764（http://ww1.microchip.com/downloads/en/appnotes/atmel-42508-software-library-for-aes-128-encryption-and-decryption_applicationnote_at10764.pdf）
- AVR230 DES Bootloader（http://ww1.microchip.com/downloads/en/appnotes/doc2541.pdf）
- Handbook of Applied Cryptography（http://cacr.uwaterloo.ca/hac）
- AES Specification（http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf）

12. 版本历史

文档版本	日期	备注
B	2018 年 4 月	增加了“从 Atmel START 获取源代码”部分，其中包含 Atmel START 示例代码的链接。
A	2017 年 6 月	转换为 Microchip 格式，并替换了 Atmel 文档编号 2589。 更新了文档，并针对 ATmega328PB 器件测试了应用程序代码。
2589E	2012 年 3 月	更新了模板。 更新了“PC 应用程序”部分。 修正了 ATmega32 项目中错误的中断向量大小。 删除了对数据的周期依赖性。
2589D	2006 年 8 月	进行了少量的修正。

Microchip 网站

Microchip 网站（http://www.microchip.com/）为客户提供在线支持。客户可通过该网站方便地获取文件和信息。我们的网站提供以下内容：

- **产品支持**——数据手册和勘误表、应用笔记和示例程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及归档软件
- **一般技术支持**——常见问题解答（FAQ）、技术支持请求、在线讨论组以及 Microchip 设计伙伴计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

产品变更通知服务

Microchip 的产品变更通知服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请访问 http://www.microchip.com/pcn，然后按照注册说明进行操作。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师（ESE）
- 技术支持

客户应联系其代理商、代表或 ESE 寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过 http://www.microchip.com/support 获得网上技术支持。

Microchip 器件代码保护功能

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 愿意与关心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。

法律声明

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，否则在 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。

商标

Microchip 的名称和徽标组合、Microchip 徽标、Adaptec、AnyRate、AVR、AVR 徽标、AVR Freaks、BesTime、BitCloud、chipKIT、chipKIT 徽标、CryptoMemory、CryptoRF、dsPIC、FlashFlex、flexPWR、HELDO、IGLOO、JukeBlox、KeeLoq、Kleer、LANCheck、LinkMD、maXStylus、maXTouch、MediaLB、megaAVR、Microsemi、Microsemi 徽标、MOST、MOST 徽标、MPLAB、OptoLyzer、PackeTime、PIC、picoPower、PICSTART、PIC32 徽标、PolarFire、Prochip Designer、QTouch、SAM-BA、SenGenuity、SpyNIC、SST、SST 徽标、SuperFlash、Symmetricom、SyncServer、Tachyon、TempTrackr、TimeSource、tinyAVR、UNI/O、Vectron 及 XMEGA 均为 Microchip Technology Incorporated 在美国和其他国家或地区的注册商标。

APT、ClockWorks、The Embedded Control Solutions Company、EtherSynch、FlashTec、Hyper Speed Control、HyperLight Load、IntelliMOS、Libero、motorBench、mTouch、Powermite 3、Precision Edge、ProASIC、ProASIC Plus、ProASIC Plus 徽标、Quiet-Wire、SmartFusion、SyncWorld、Temux、TimeCesium、TimeHub、TimePictra、TimeProvider、Vite、WinPath 和 ZL 均为 Microchip Technology Incorporated 在美国的注册商标。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、BlueSky、BodyCom、CodeGuard、CryptoAuthentication、CryptoAutomotive、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、EtherGREEN、In-Circuit Serial Programming、ICSP、INICnet、Inter-Chip Connectivity、JitterBlocker、KleerNet、KleerNet 徽标、memBrain、Mindi、MiWi、MPASM、MPF、MPLAB Certified 徽标、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PowerSmart、PureSilicon、QMatrix、REAL ICE、Ripple Blocker、SAM-ICE、Serial Quad I/O、SMART-I.S.、SQI、SuperSwitcher、SuperSwitcher II、Total Endurance、TSHARC、USBCheck、VariSense、ViewSpan、WiperLock、Wireless DNA 和 ZENA 均为 Microchip Technology Incorporated 在美国和其他国家或地区的商标。

SQTP 为 Microchip Technology Incorporated 在美国的服务标记。

Adaptec 徽标、Frequency on Demand、Silicon Storage Technology 和 Symmcom 均为 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。

ISBN：978-1-5224-5819-7

质量管理体系

有关 Microchip 的质量管理体系的信息，请访问 http://www.microchip.com/quality。

全球销售及服务网点

美洲	亚太地区	亚太地区	欧洲
公司总部 2355 West Chandler Blvd. Chandler, AZ 85224-6199 电话：480-792-7200 传真：480-792-7277 技术支持： http://www.microchip.com/support 网址： http://www.microchip.com **亚特兰大** 德卢斯，佐治亚州 电话：678-957-9614 传真：678-957-1455 **奥斯汀，德克萨斯州** 电话：512-257-3370 **波士顿** 韦斯特伯鲁，马萨诸塞州 电话：774-760-0087 传真：774-760-0088 **芝加哥** 艾塔斯卡，伊利诺伊州 电话：630-285-0071 传真：630-285-0075 **达拉斯** 阿迪森，德克萨斯州 电话：972-818-7423 传真：972-818-2924 **底特律** 诺维，密歇根州 电话：248-848-4000 **休斯顿，德克萨斯州** 电话：281-894-5983 **印第安纳波利斯** 诺布尔斯维尔，印第安纳州 电话：317-773-8323 传真：317-773-5453 电话：317-536-2380 **洛杉矶** 米慎维荷，加利福尼亚州 电话：949-462-9523 传真：949-462-9608 电话：951-273-7800 **罗利，北卡罗来纳州** 电话：919-844-7510 **纽约，纽约州** 电话：631-435-6000 **圣何塞，加利福尼亚州** 电话：408-735-9110 电话：408-436-4270 **加拿大 - 多伦多** 电话：905-695-1980 传真：905-695-2078	**澳大利亚 - 悉尼** 电话：61-2-9868-6733 **中国 - 北京** 电话：86-10-8569-7000 **中国 - 成都** 电话：86-28-8665-5511 **中国 - 重庆** 电话：86-23-8980-9588 **中国 - 东莞** 电话：86-769-8702-9880 **中国 - 广州** 电话：86-20-8755-8029 **中国 - 杭州** 电话：86-571-8792-8115 **中国 - 香港特别行政区** 电话：852-2943-5100 **中国 - 南京** 电话：86-25-8473-2460 **中国 - 青岛** 电话：86-532-8502-7355 **中国 - 上海** 电话：86-21-3326-8000 **中国 - 沈阳** 电话：86-24-2334-2829 **中国 - 深圳** 电话：86-755-8864-2200 **中国 - 苏州** 电话：86-186-6233-1526 **中国 - 武汉** 电话：86-27-5980-5300 **中国 - 西安** 电话：86-29-8833-7252 **中国 - 厦门** 电话：86-592-2388138 **中国 - 珠海** 电话：86-756-3210040	**印度 - 班加罗尔** 电话：91-80-3090-4444 **印度 - 新德里** 电话：91-11-4160-8631 **印度 - 浦那** 电话：91-20-4121-0141 **日本 - 大阪** 电话：81-6-6152-7160 **日本 - 东京** 电话：81-3-6880-3770 **韩国 - 大邱** 电话：82-53-744-4301 **韩国 - 首尔** 电话：82-2-554-7200 **马来西亚 - 吉隆坡** 电话：60-3-7651-7906 **马来西亚 - 槟榔屿** 电话：60-4-227-8870 **菲律宾 - 马尼拉** 电话：63-2-634-9065 **新加坡** 电话：65-6334-8870 **台湾地区 - 新竹** 电话：886-3-577-8366 **台湾地区 - 高雄** 电话：886-7-213-7830 **台湾地区 - 台北** 电话：886-2-2508-8600 **泰国 - 曼谷** 电话：66-2-694-1351 **越南 - 胡志明市** 电话：84-28-5448-2100	**奥地利 - 韦尔斯** 电话：43-7242-2244-39 传真：43-7242-2244-393 **丹麦 - 哥本哈根** 电话：45-4485-5910 传真：45-4485-2829 **芬兰 - 埃斯波** 电话：358-9-4520-820 **法国 - 巴黎** 电话：33-1-69-53-63-20 传真：33-1-69-30-90-79 **德国 - 加兴** 电话：49-8931-9700 **德国 - 哈恩** 电话：49-2129-3766400 **德国 - 海尔布隆** 电话：49-7131-72400 **德国 - 卡尔斯鲁厄** 电话：49-721-625370 **德国 - 慕尼黑** 电话：49-89-627-144-0 传真：49-89-627-144-44 **德国 - 罗森海姆** 电话：49-8031-354-560 **以色列 - 若那那市** 电话：972-9-744-7705 **意大利 - 米兰** 电话：39-0331-742611 传真：39-0331-466781 **意大利 - 帕多瓦** 电话：39-049-7625286 **荷兰 - 德卢内市** 电话：31-416-690399 传真：31-416-690340 **挪威 - 特隆赫姆** 电话：47-72884388 **波兰 - 华沙** 电话：48-22-3325737 **罗马尼亚 - 布加勒斯特** 电话：40-21-407-87-50 **西班牙 - 马德里** 电话：34-91-708-08-90 传真：34-91-708-08-91 **瑞典 - 哥德堡** 电话：46-31-704-60-40 **瑞典 - 斯德哥尔摩** 电话：46-8-5090-4654 **英国 - 沃金厄姆** 电话：44-118-921-5800 传真：44-118-921-5820