

AN0003: UART 引导装载程序



本应用说明专供 EFM32 UART 引导装载程序用户使用。借助该引导装载程序，用户无需调试器即可通过 UART 对 EFM32 系列 0、EZR32 系列 0 和 EFM32 系列 1 设备编程。

除了启动用户应用，EFM32/EZR32 引导装载程序还提供破坏性写入模式，允许用户覆盖引导装载程序，以便将整个闪存空间用于用户应用。在 EFM32 系列 1 设备中，引导装载程序驻留在闪存的保留区，因此这些设备的引导装载程序不提供破坏性写入模式。然而，可以通过设置锁定位页中的位来禁用 EFM32 系列 1 设备的引导装载程序。

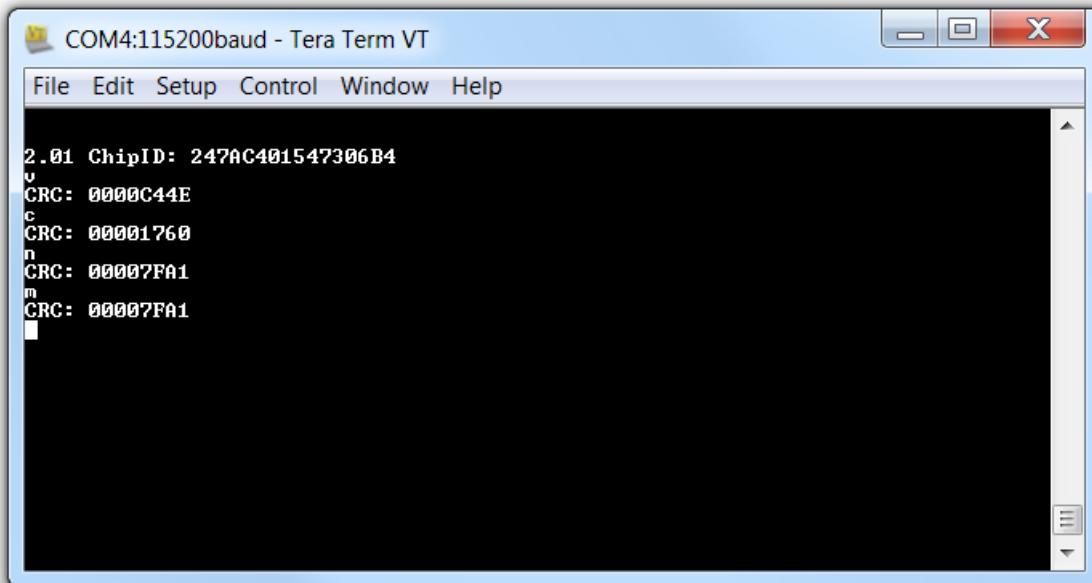
在所有设备中，可通过 CRC 校验和来验证闪存内容，并可启用调试锁定以保护 IP。由于引导装载程序使用已建立的 XMODEM-CRC 协议来上传数据，因此可使用任何串行终端程序与引导装载程序通信。

此应用说明包括以下内容：

- 本 PDF 文档
- 源文件 (zip)
 - 完整的引导加载程序源代码
 - IAR EW 项目文件
- 应用的 IAR 链接器文件
- 引导加载程序的二进制图像

内容要点

- 所有 EFM32 设备均使用引导装载程序预编程，实现：
 - 可与客户应用同时保留以支持现场升级。
 - 可被覆写以最大程度地增加可用闪存空间（仅针对 EFM32 系列 0 和 EZR32 系列 0）或可被锁定位页的锁定位禁用（仅针对 EFM32 系列 1）。
 - 使用 UART 接口进行通信（有关部件特定外围配置和所支持波特率的信息，请参见第 1.2 节）。
 - 支持单字符命令：对 EFM32 设备进行编程（上传/覆写）、验证（计算校验和）、保护（写入保护、锁定调试端口）等。
 - 接受使用 XMODEM-CRC 协议进行文件传输。
 - 在复位以后被调用，同时 DBG_SWCLK 被拉到高电平。



1. 设备兼容性

本应用说明支持多个设备系列，某些功能因设备不同而有所差异。

UART 引导装载程序已在所有 EFM32 系列 0 设备、所有 EZR32 系列 0 设备和部分 EFM32 系列 1 设备中预编程。带有 USB 外围设备的 EFM32 系列 0 和 EZR32 系列 0 设备还内置了 USB 引导装载程序。对于已启用 USB 的设备，引导装载程序包含在应用说明 “AN0042: USB UART 引导装载程序” 中，通过 Simplicity Studio (www.silabs.com/simplicity) 或 Silicon Labs 网站 www.silabs.com 提供 www.silabs.com。

以下是支持的设备列表：

EFM32 系列 0:

- EFM32 Gecko (EFM32G)
- EFM32 Giant Gecko (EFM32GG)
- EFM32 Wonder Gecko (EFM32WG)
- EFM32 Leopard Gecko (EFM32LG)
- EFM32 Tiny Gecko (EFM32TG)
- EFM32 Zero Gecko (EFM32ZG)
- EFM32 Happy Gecko (EFM32HG)

EZR32 系列 0:

- EZR32 Wonder Gecko (EZR32WG)
- EZR32 Leopard Gecko (EZR32LG)
- EZR32 Happy Gecko (EZR32HG)

EFM32 系列 1:

- EFM32 Pearl Gecko (EFM32PG1/EFM32PG12 修订版 C 及更高版本)
- EFM32 Jade Gecko (EFM32JG1/EFM32JG12 修订版 C 及更高版本)
- EFM32 Giant Gecko GG11 (EFM32GG11)
- EFM32 Giant Gecko GG12 (EFM32GG12)
- EFM32 Tiny Gecko 11 (EFM32TG11)

Note: IOVDD 不应由 EFM32xG11/12 设备上的直流转换器供电。复位时，直流转换器默认进入未配置安全状态，同时其输出悬空，以使连接电路保持未通电状态，直至固件进行必要的配置。出厂时，空白设备会运行引导装载程序，在尝试通过 BOOT_RX 和 BOOT_TX 引脚与主机通信时会失败（无 IOVDD 电源）。在此情况下，使用调试接口 (DBG_SWCLKTCK 和 DBG_SWDIOTMS) 进行初始固件下载同样也会失败。

Note: 该引导装载程序软件不支持 EFM32 系列 1。

EFR32 系列 1 该引导装载程序不支持 EFR32xG1/EFR32xG12/EFR32xG13/EFR32xG14 设备。这些设备的引导装载程序包含在协议栈软件中。这些设备包括：

- EFR32 Blue Gecko (EFR32BGxx)
- EFR32 Flex Gecko (EFR32FGxx)
- EFR32 Mighty Gecko (EFR32MGxx)

2. 启动 UART 引导装载程序

2.1 进入引导装载程序模式

如果要进入引导装载程序模式，必须将 DBG_SWCLK 拉到高电平并将 EFM32 系列 0、EZR32 系列 0 或 EFM32 系列 1 设备复位。如果 DBG_SWCLK 在复位时为低电平，引导装载程序会检查闪存的应用空间。如果应用空间中包含有效应用并且 DBG_SWCLK 为低电平，引导装载程序将运行该应用。如果不存在有效的应用，引导装载程序将处于 EM2 睡眠模式以节省电力，并会定期检查引导装载程序引脚。

Note: DBG_SWCLK 具有内部下拉电阻。如未连接该引脚，将不会在复位时调用引导装载程序模式。

Note: 引导装载程序的早期版本使用 DBG_SWDIO 和 DBG_SWCLK 引脚进入引导装载程序。您仍可通过将 DBG_SWDIO 线拉至低电平来进入引导装载程序，但调试锁定将不起作用。

2.2 利用 UART 引导装载程序初始化通信

有对于 EFM32 系列 0 和 EZR32 系列 0 设备，UART 引导装载程序通常使用 GPIO 引脚 PE11 (RX) 和 PE10 (TX) 进行 UART 通信（例外情况请参见以下注释）。有关引脚位置，请查看具体设备数据表。

Note: 对于 EFM32G、EFM32GG、EFM32LG 和大多数 EFM32TG 部件，引导装载程序使用位置 0 处的 USART0 进行通信。然而，一些设备没有 USART0 外围设备，另一些设备有 USART0 外围设备但不包含位置 0 选项。出于上述及其他原因，EFM32ZG、EFM32HG 及一些 EFM32TG（尤其是 EFM32TG108Fxx 和 EFM32TG110Fxx）部件均使用 LEUART0、位置 3。请注意，该位置与常规 SWD 端口重叠，如上所述，该端口用于进入引导装载程序。因此，在使用这些部件时，应在 DBG_SWCLK 上使用 4 kΩ 上拉电阻。对于 EFM32 系列 1 设备，引导装载程序在端口 F0 (DBG_SWCLK) 和 F1 (DBG_SWDIO) 上使用 USART 连接。这些设备也应在 DBG_SWCLK 上使用 4 kΩ 上拉电阻。

UART 使用 1 个停止位、8 个数据位、无校验位。此外，引导装载程序还使用自动波特率来启用各种不同终端。自动波特率功能通过终端程序感应所用的波特率并进行相应调节。打开设备串行连接后立即向引导装载程序发送大写字母 “U”，完成初始化。引导装载程序可感应位之间的时序，并调节其自己的预分频器来匹配所感应的波特率。在 AN0003 软件包中导航至 [an0003_efm32_uart_bootloader]>[二进制文件]>[notes.txt]（可在 <https://www.silabs.com/support/resources> 网站上获取该软件包），以查看与自动波特率算法兼容的部件特定波特率范围。

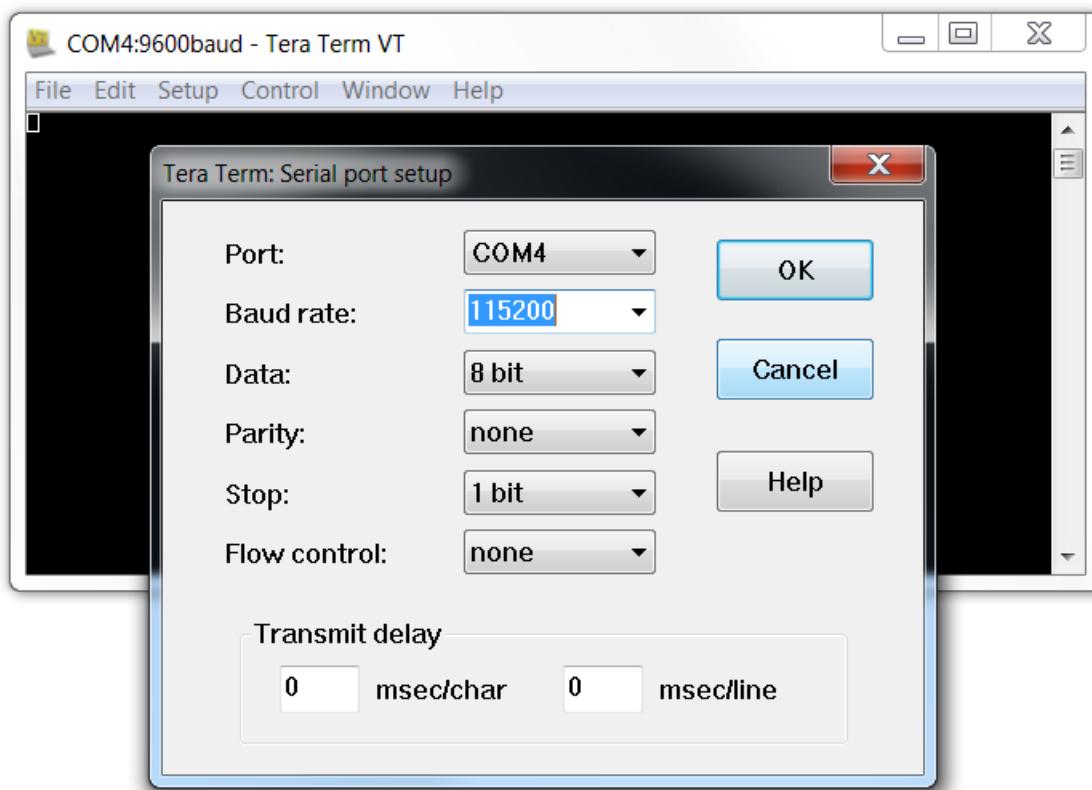


Figure 2.1. 在 Windows 7 上的 Tera Term 中配置 UART 引导装载程序的串行端口

引导装载程序成功初始化之后，即会打印引导装载程序版本和芯片唯一 ID：

1.40 Chip ID: F08AB6000B153525

Note: 1.40 版之前的引导装载程序版本不能打印引导装载程序版本，只能打印芯片唯一 ID。

Note: 在 AN0003 软件包中导航至 [an0003_efm32_uart_bootloader]>[二进制文件]>[bootloader-matrix.txt]（可在 <https://www.silabs.com/support/resources> 网站上获取该软件包），以查看每个产品系列的最新引导装载程序版本。

2.3 命令行接口

命令行接口使用单个字母字符作为命令。以下是命令行接口支持的单一小写字母字符命令：

u

上传应用。

EFM32 系列 0 和 EZR32 系列 0: 该命令允许用户将应用上传到闪存，同时保持引导装载程序完好无损。为了使应用正常工作，必须使用将 EFM32G、EFM32TG、EFM32ZG 和 EFM32HGF 部件的应用起始地址放在 0x800 以及将 EFM32GG、EFM32LG 和 EFM32WG 部件的应用起始地址放在 0x1000 的链接器文件。使用 XMODEM-CRC 协议传输应用。

EFM32 系列 1: 该命令允许用户将应用上传到闪存。无需对用户应用进行特殊修改，因为引导装载程序驻留在闪存保留区，无法覆写。使用 XMODEM-CRC 协议传输应用。

d

破坏性上传（仅针对 EFM32 系列 0 和 EZR32 系列 0）。该命令允许用户将应用上传到闪存，从而覆写引导装载程序。无需修改用户应用的起始地址。使用 XMODEM-CRC 协议传输应用。EFM32 系列 1 设备上没有破坏性上传选项。

t

上传到用户页。该命令允许用户写入用户信息页。使用 XMODEM-CRC 协议上传数据。

p

上传到锁定页。该命令允许用户写入锁定位信息页。使用 XMODEM-CRC 协议上传数据。

b

启动应用。该命令将启动已上传的应用。

l

调试锁定。该命令用于设置锁定页的调试锁定位。EFM32 系列 0、EZR32 系列 0 或 EFM32 系列 1 将被锁定以进行调试。

v

验证闪存校验和。该命令将计算整个闪存的 CRC-16 校验和并打印。该命令适合与 [d] 命令 (EFM32 系列 0 和 EZR32 系列 0) 或 [u] 命令 (EFM32 系列 1) 结合使用。请注意，对于 EFM32 系列 1 设备，[v] 命令和 [c] 命令得到同样的结果。

c

验证应用校验和。该命令将计算应用的 CRC-16 校验和并打印。该命令适合与 [u] 命令结合使用。

请注意，对于 EFM32 系列 1 设备，[c] 命令和 [v] 命令得到同样的结果。

n

验证用户页校验和。该命令将计算用户页的 CRC-16 校验和并打印。该命令适合与 [t] 命令结合使用。

m

验证锁定页校验和。该命令将计算锁定页的 CRC-16 校验和并打印。该命令适合与 [p] 命令结合使用。

r

复位 EFM32 系列 0、EZR32 系列 0 或 EFM32 系列 1 设备。

3. 上传应用

如果要将应用上传到 EFM32 系列 0、EZR32 系列 0 和 EFM32 系列 1，必须使用 [u]（所有设备）或 [d]（仅针对 EFM32 系列 0 和 EZR32 系列 0）命令。按下键以后，使用内置支持 XMODEM-CRC 的终端软件传输文件。可以使用支持 XMODEM-CRC 传输的任何终端软件。

如果要在 Tera Term 中通过 XMODEM-CRC 发送文件，请导航至 [文件]>[传输]>[XMODEM]>[发送...]。下图所示为使用支持内置传输的 Tera Term 传输文件的示例。

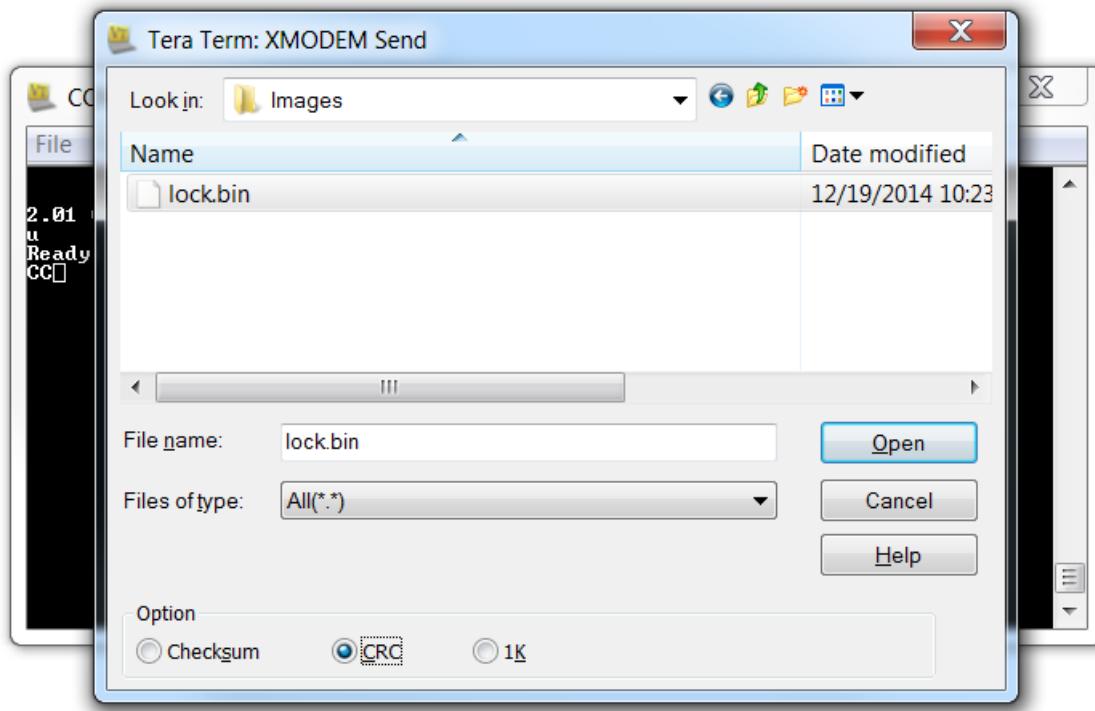


Figure 3.1. 通过 Tera Term 使用 XMODEM-CRC 传输文件

3.1 创建适用于引导装载程序 - EFM32 系列 0 和 EZR32 系列 0 的应用

Note: 有关创建适用于 EFM32 系列 1 设备的应用的信息，请参见 3.2 创建适用于引导装载程序 - EFM32 系列 1 的应用。

使用适用于 EFM32 系列 0 和 EZR32 系列 0 设备的引导装载程序上传应用时，存在破坏性和定期上传两种可能。破坏性上传会覆盖引导装载程序。在本例中，创建应用时无需采取额外的步骤。定期上传会保留引导装载程序，从而允许未来使用引导装载程序进行升级。不过，必须对应用进行设置，才能实现这一功能。为了使应用适用于引导装载程序，对于 EFM32G、EFM32TG、EFM32ZG 和 EFM32HG 部件，必须使用起始地址 0x800 创建应用；对于 EFM32GG、EFM32LG 和 EFM32WG 部件，必须使用起始地址 0x1000 创建应用。原因在于，引导装载程序本身分别占用 0x0 与 0x7FF 或 0xFFFF 之间的闪存区。为了使应用与引导装载程序共存，必须更改应用链接器文件的默认闪存起始地址 0x0。

Note:

如果在使用任一链接器文件时需要调试应用，则必须在代码中明确设定矢量表的位置。可通过以下方式完成：

```
SCB->VTOR=0x800; // EFM32G, EFM32TG, EFM32ZG and EFM32HG parts
```

或者

```
SCB->VTOR=0x1000; // EFM32GG, EFM32LG and EFM32WG parts
```

在发布的应用中，没有必要如此，因为在应用启动之前，VTOR 由引导装载程序本身设定。有关详情，请参见 Boot.c。

3.1.1 使用 IAR 创建应用

如果要使用 IAR 创建应用, 请在 AN0003 软件包中导航至 [an0003_efm32_usb_uart_bootloader]>[iar_linker_files] 在 AN0003 软件包 (可在 <https://www.silabs.com/support/resources> 网站上获取该软件包), 并使用项目的部件特定链接器文件。这将为二进制文件设置正确的起始地址。另一种方法是, 右键单击 IAR 项目并选择 [选项...], 导航至 [链接器] 选项卡, 并确保选中 [覆盖默认设置] 选项。

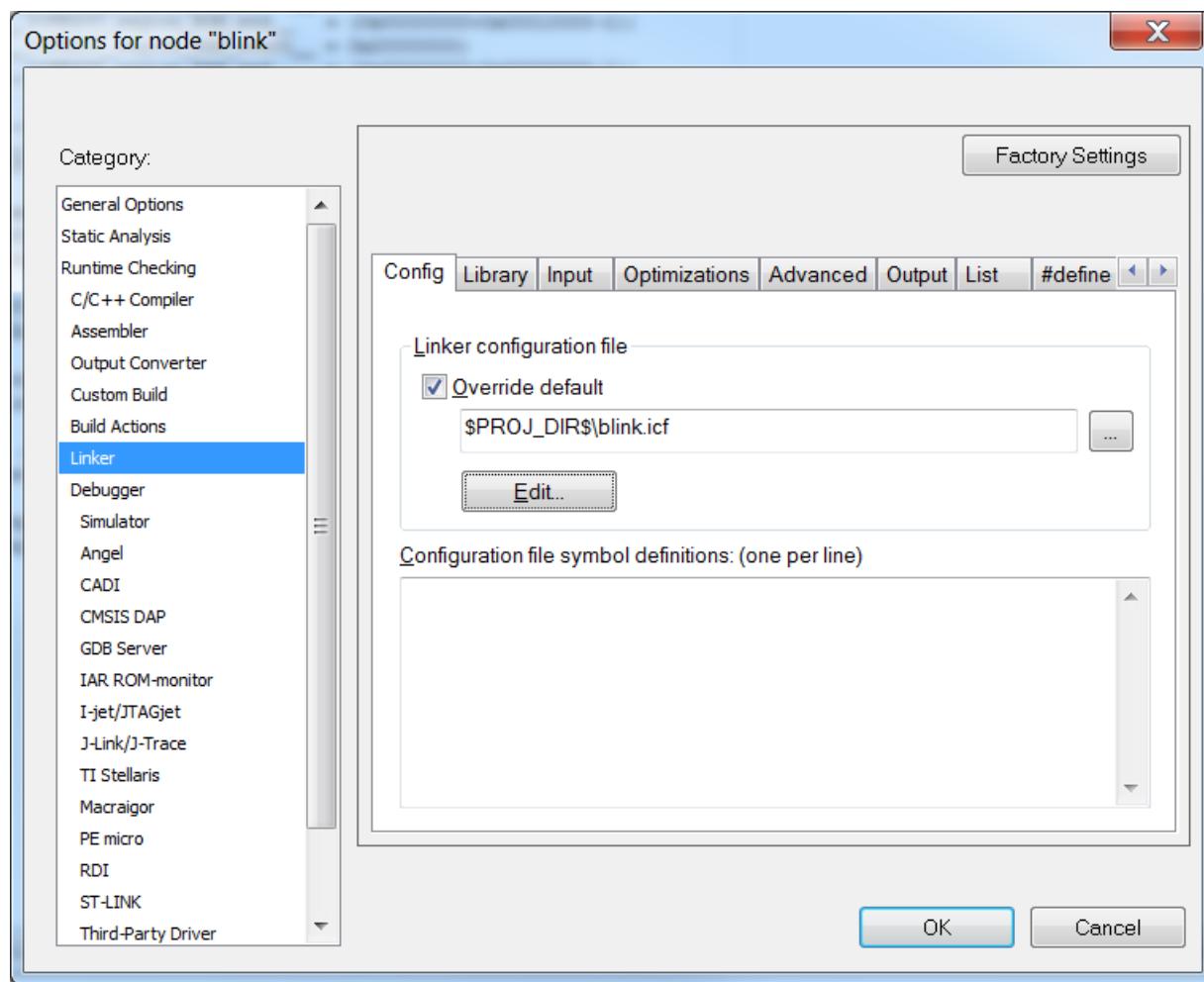


Figure 3.2. IAR 项目选项

在项目目录中创建 .icf 文件, 然后单击 [编辑]。在 [矢量表] 选项卡中将矢量表的起始地址设置为 [0x0800] 或者 [0x1000] (视部件而定)。在 [内存区域选项卡中 将 ROM 的起始地址设置为] 选项卡中将矢量表的起始地址设置为 [0x0800] 或者 [0x1000] (视部件而定)。单击 [完成后单击保存。] when done. Click [在] in the [选项] 窗口中单击确定。

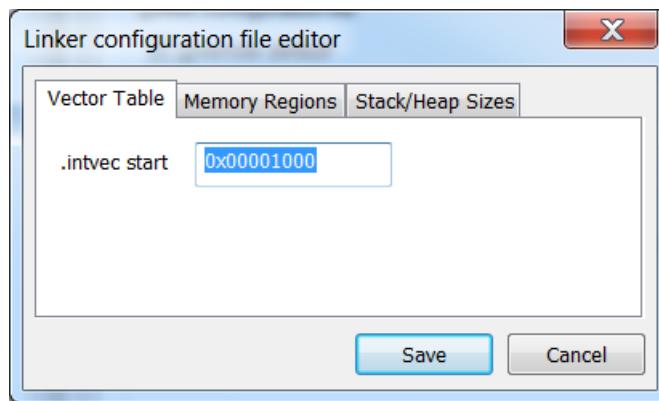


Figure 3.3. 更改矢量表地址

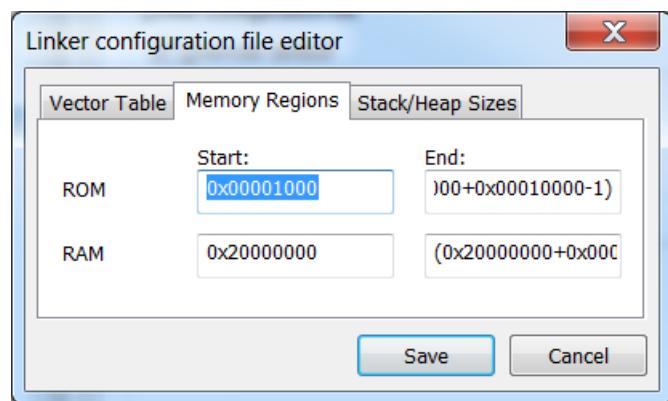


Figure 3.4. 更改 ROM 起始地址

在项目选项菜单中，选择 [输出转换器] 和 [生成额外输出]。选择 [二进制] 输出格式。所得的二进制文件可用于 UART 引导装载程序。

3.1.2 使用 Keil uVision 4/MDK-ARM 创建应用

如果要使用 Keil uVision 4/MDK-ARM 创建应用，必须先更改项目的目标设置。在选项对话框中，将 IROM1 更改为起始于 0x800 或 0x1000 并从大小字段中减去 0x800 或 0x1000。（为 0x800 还是 0x1000 取决于使用哪个部件）。

如果要生成二进制输出文件，可使用命令行实用工具“fromelf.exe”，该实用工具通常安装在 C:\Keil\ARM\BIN40\fromelf.exe 下。有关详情，请参见“uVision 帮助”中的 *Realview 实用工具指南*。

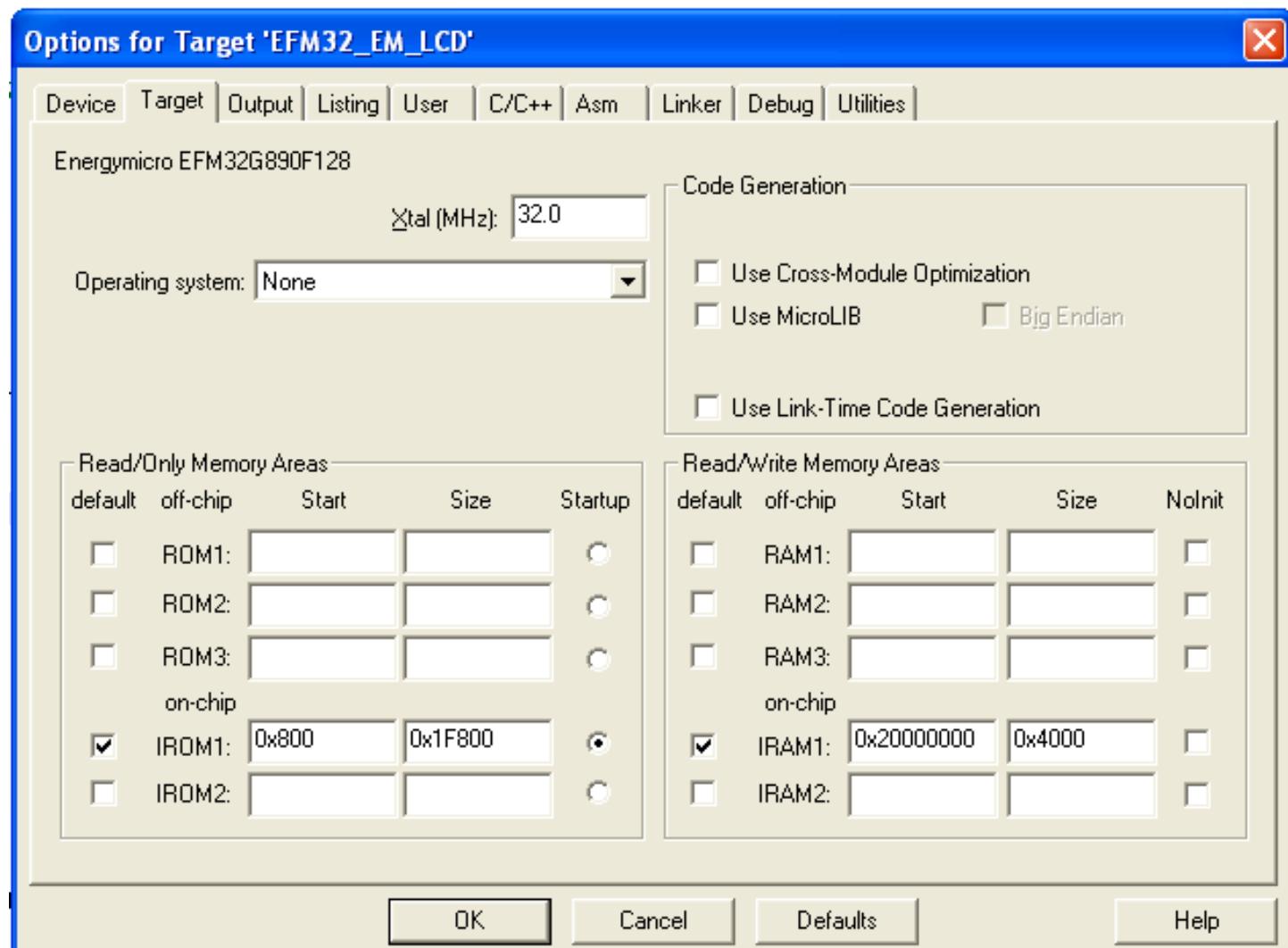


Figure 3.5. 设置 Keil uVision 4/MDK-ARM

3.1.3 使用 Eclipse/GCC/Sourcery CodeBench 创建应用

如果要使用与引导装载程序同时运行的 Eclipse、GCC 或 Sourcery CodeBench 创建应用，需要修改链接器文件。对于应用说明和示例项目，链接器文件在软件项目包含的生成文件中指定。在链接器文件的 MEMORY 命令中，将 ROM ORIGIN 更改为 0x00000800 或 0x00001000，并对长度作相应更改，如下图所示。

```

1  /* Linker script for Energy Micro EFM32G
2  *
3  * Version: Sourcery G++ 4.4-139 - Preview
4  * Support: https://support.codesourcery.com/GNUToolchain/
5  *
6  * Copyright (c) 2007, 2008, 2009, 2010 CodeSourcery, Inc.
7  *
8  * The authors hereby grant permission to use, copy, modify, distribute,
9  * and license this software and its documentation for any purpose, provided
10 * that existing copyright notices are retained in all copies and that this
11 * notice is included verbatim in any distributions. No written agreement,
12 * license, or royalty fee is required for any of the authorized uses.
13 * Modifications to this software may be copyrighted by their authors
14 * and need not follow the licensing terms described here, provided that
15 * the new terms are clearly indicated on the first page of each file where
16 * they apply.
17 */
18 OUTPUT_FORMAT ("elf32-littlearm", "elf32-bigarm", "elf32-littlearm")
19 ENTRY(__cs3_reset)
20 SEARCH_DIR(.)
21 GROUP(-lgcc -lc -lcs3 -lcs3unhosted)
22
23 MEMORY
24 {
25     rom (rx) : ORIGIN = 0x00000800, LENGTH = 129024
26     ram (rwx) : ORIGIN = 0x20000000, LENGTH = 16384
27 }
28
29 /* These force the linker to search for particular symbols from
30 * the start of the link process and thus ensure the user's
31 * overrides are picked up
32 */

```

length:7460 lines:255 Ln:25 Col:50 Sel:0 Dos\Windows ANSI INS

Figure 3.6. Eclipse/gcc/cs 链接器文件中的应用起始地址

Note: 如果在使用任一链接器文件时需要调试应用，则必须在代码中明确设定矢量表的位置。可通过以下方式完成：

SCB->VTOR=0x800; // EFM32G、EFM32TG、EFM32ZG 和 EFM32HG 部件

或

SCB->VTOR=0x1000; // EFM32GG、EFM32LG 和 EFM32WG 部件

在发布的应用中，没有必要如此，因为在应用启动之前，VTOR 由引导装载程序本身设定（有关详情，请参见 Boot.c）。

3.1.4 使用 Simplicity Studio 创建应用

如果要使用与 EFM32 系列 0 或 EZR32 系列 0 引导装载程序同时运行的 Simplicity Studio 创建应用，必须更改项目属性，告知链接器在内存中哪个位置锁定应用。如果要从 Simplicity Studio IDE 执行此项操作，请单击 [项目] > [属性]。在 [属性] 窗口中，选择 [工具设置] 选项卡，然后选择 [内存布局]。选择 [覆盖默认闪存选项]，然后在 [原始] 字段中输入 0x800（EFM32G、EFM32TG、EFM32ZG 和 EFM32HG）或 0x1000（EFM32GG、EFM32LG 和 EFM32WG），相应调整 [长度] 值（即分别从原始长度中减去 0x800 或 0x1000），如 Figure 3.7 在 Simplicity Studio 中更改应用起始地址和内存长度 on page 11 中所示。由于链接器文件由 Simplicity Studio 自动生成，因此无需手动更改链接器文件中的这些内存设置，您的更改将在下次构建时被覆盖。

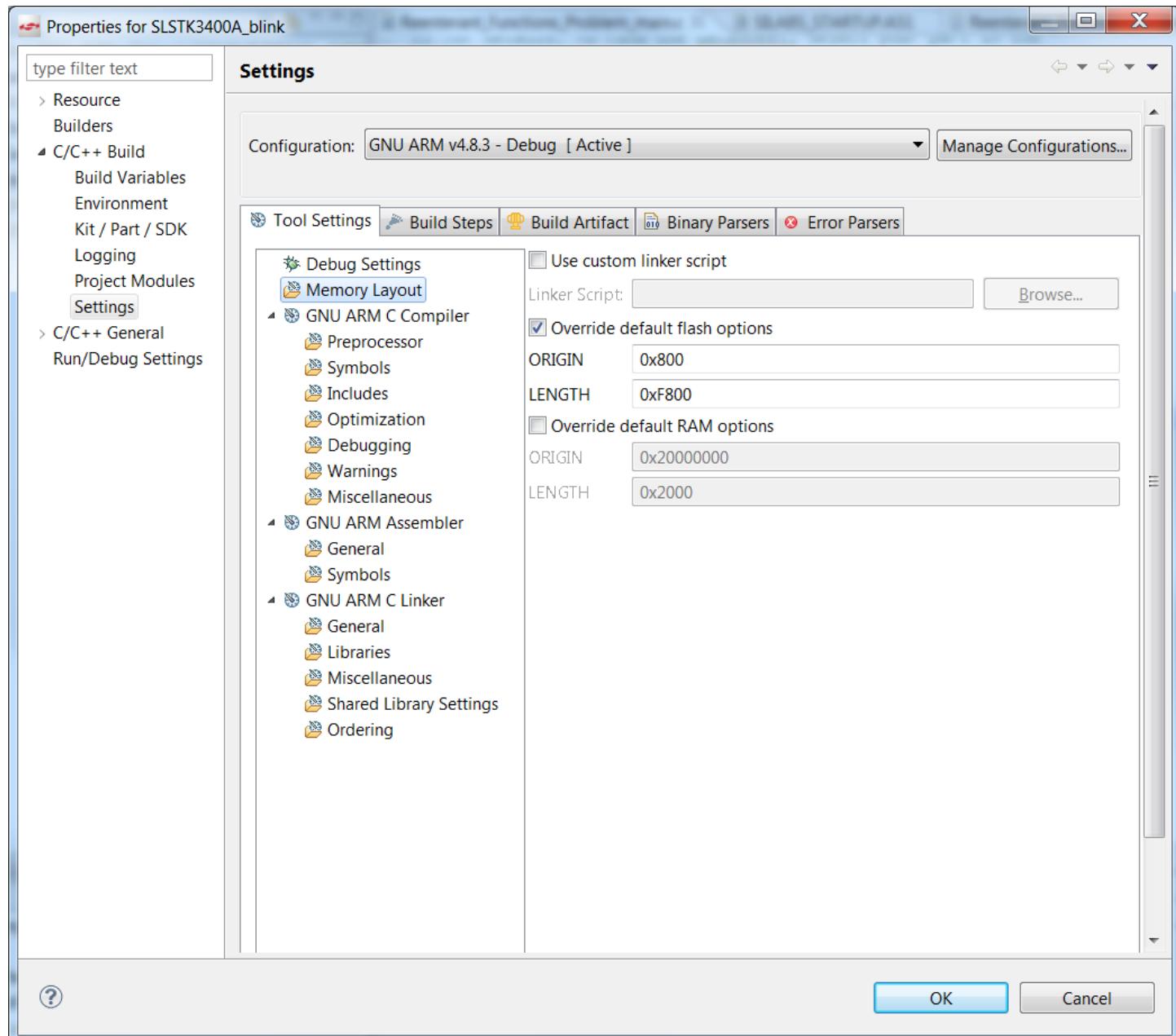


Figure 3.7. 在 Simplicity Studio 中更改应用起始地址和内存长度

Note: 如果在更改以后需要调试应用，则必须在代码中明确设定矢量表的位置。可通过以下方式完成：

```
SCB->VTOR=0x800; // EFM32G、EFM32TG、EFM32ZG 和 EFM32HG 部件
```

或

```
SCB->VTOR=0x1000; // EFM32GG、EFM32LG 和 EFM32WG 部件
```

在发布的应用中，没有必要如此，因为在应用启动之前，VTOR 由引导装载程序本身设定（有关详情，请参见 Boot.c）。

3.2 创建适用于引导装载程序 - EFM32 系列 1 的应用

由于引导加载程序驻留在专用引导加载程序区域，因此除了 EFx32xG1 设备外，所有 EFM32 系列 1 设备的引导加载程序都不会被应用覆盖。因此，为具有专用引导加载程序的这些设备创建用户应用不需要其他步骤。可通过清除锁定位页中的位 122 (CLW0) 来禁用这些设备上的引导装载程序。有关锁定位页配置的更多信息，请参见设备参考手册。对于 EFx32xG1 设备，请遵循 [3.1 创建适用于引导装载程序 - EFM32 系列 0 和 EZR32 系列 0 的应用](#) 一节。

3.3 上传应用

[u] 命令将上传应用。使用终端软件将应用二进制文件传输至芯片。上传完成之后，您可能希望通过计算已上传二进制文件的 CRC-16 来验证正确性。可通过“验证应用校验和”命令来完成（参见 [4.1 验证应用校验和](#)）。如果要从引导装载程序启动应用，则使用“启动”命令（[b]—参见 [5.1 启动应用](#)）。

3.4 破坏性上传 - 仅针对 EFM32 系列 0 和 EZR32 系列 0

[d] 命令将在 EFM32 系列 0 和 EZR32 系列 0 设备上启动破坏性上传。EFM32 系列 1 设备不提供此选项。使用终端软件将二进制文件传输至芯片。破坏性上传与定期上传的不同之处在于其会覆盖引导装载程序，从而允许上传另一引导装载程序；或者，如果不需引导装载程序，则回收引导装载程序占用的闪存。上传完成之后，您可能希望通过计算 CRC-16 校验和来验证正确性。可通过“验证闪存内容”命令来完成（参见 [4.2 验证闪存内容](#)）。如果要启动应用，可使用“复位”命令（[r]—参见 [5.2 复位设备](#)）。

3.5 写入用户信息页

[t] 命令允许将数据写入用户信息页。使用终端软件将用户数据传输到用户信息页。

3.6 写入锁定位信息页

[p] 命令允许将数据写入锁定位信息页。使用终端软件将用户数据传输到用户信息页。该命令允许锁定闪存中的页，使其无法写入和擦除，但不能保护内容。有关锁定位详情，请参见参考手册。

4. 验证上传

Note: XMODEM-CRC 传输 128 字节的数据块。如果二进制文件大小不是 128 字节的倍数，终端程序将填补剩余字节。有关详情，请参见终端程序的文档。

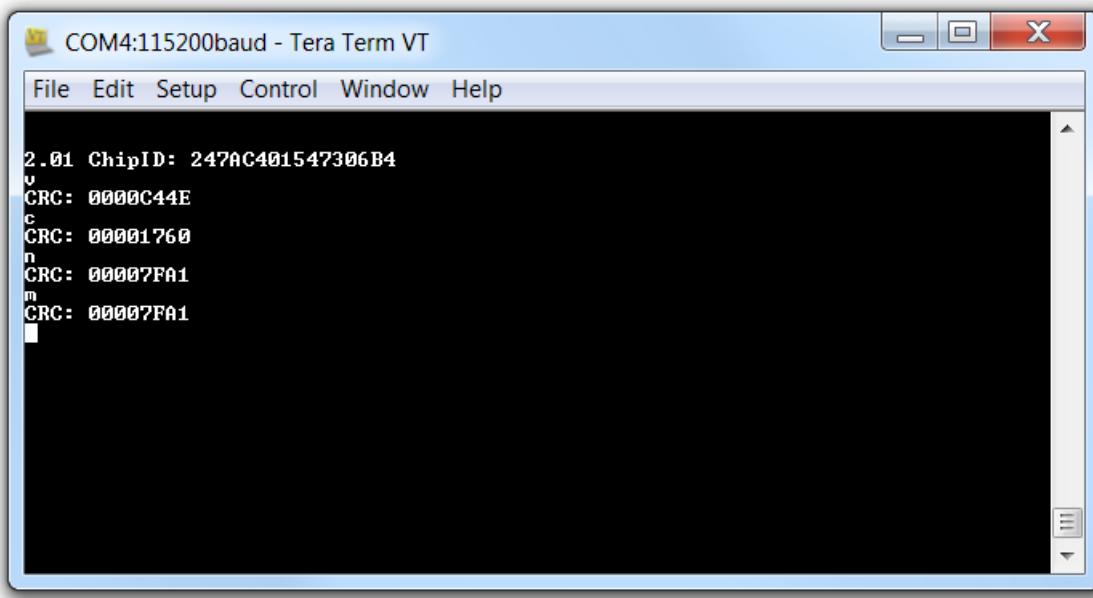


Figure 4.1. 使用引导装载程序校验和命令的示例 Tera Term 会话（未显示起始字符“U”）。

4.1 验证应用校验和

[c] 命令将计算并打印从基本 0x800 或 0x1000（应用起点）到闪存空间末端的闪存的 CRC-16 校验和。

4.2 验证闪存内容

[v] 命令将计算并打印从基本 0x0（闪存空间起点）到闪存空间末端的闪存的 CRC-16 校验和。

4.3 验证用户页校验和

[n] 命令将计算并打印用户数据 (UD) 页（信息闪存块的第 0 页）的 CRC-16 校验和。

4.4 验证锁定页校验和

[m] 命令将计算并打印锁定定位 (LB) 页（信息闪存块的第 1 页）的 CRC-16 校验和。

5. 其他命令

5.1 启动应用

虽然 [b] 命令将启动已上传的应用，其方式类似于不通过拉高调试引脚电平来启用引导装载程序。引导装载程序的工作方式为首先将处理器的矢量表设置为应用基础。然后读出新矢量表中的首字，对 SP 进行相应设置。最后将 PC 设置为复位矢量定义的值，执行矢量复位。

Note: 引导装载程序在正常运行中配置 TIMER、USART、CMU 和 GPIO。使用该命令启动应用时这些设置将保留。然而，如果不是通过使引导装载程序引脚有效而进入引导装载程序，这些寄存器将不会更改。这是典型的情况。

Note: 使用 UART 引导加载程序时，字母 [b] 不会打印在串行终端上，但应用在用户按下 [b] 命令后应该会启动。

5.2 复位设备

[r] 命令将复位设备。如果在破坏性上传以后发出该命令，将启动新的二进制文件。如果在定期上传以后发出该命令并且未将调试引脚拉到高电平，应用将启动。否则，引导装载程序将重新启动。

5.3 调试锁定

[l] 命令将锁定调试接口。锁定之后，将无法访问定期调试工具；仅能通过调试接口进行设备擦除。

Note: 在锁定调试接口之前，必须将设备复位。如果锁定成功，该命令将返回“确定”，否则将返回“失败”。如果调试锁定失败，请确保 DBG_SWDIO 未连接并且 DBG_SWCLK 处于高电平。

6. 编译引导装载程序

除了本应用说明，还随附引导装载程序自身的源代码。可以使用该源代码编译您自己的引导装载程序。使用该源代码时需要特别注意几个方面。

- 编译后的引导装载程序必须适合闪存配置区的大小。对于 Gecko、Tiny Gecko、Zero Gecko 和 Happy Gecko，大小为 0x800 字节。对于 Leopard Gecko、Giant Gecko 和 Wonder Gecko，大小为 0x1000 字节。
- 源项目仅适用于 IAR。可将源文件用于其他 IDE，但必须确保编译后的引导装载程序适合分配的大小。
- 在 IAR 中，必须在发布配置中编译源代码。如果在调试配置中编译，编译后的程序可能过大。勾选“大小高度优化”并启用“多文件编译”。“放弃使用的公共区”也是非常有用的功能，但要先通过在发布配置中编译来启用。
- 对于 Tiny Gecko，共有两个项目。EFM32TG108Fxx 和 EFM32TG110Fxx 设备应使用 bootloader-tg-small。其他 Tiny Gecko 设备应使用 bootloader-tg。

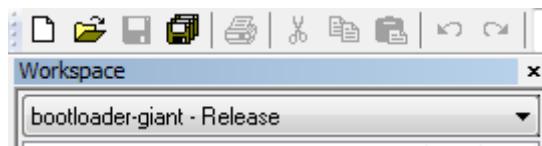


Figure 6.1. 在发布模式下编译

7. 版本历史

修订版 1.77

2022 年 3 月

- 将 EFM32GG12 添加到了 [1. 设备兼容性](#)。

修订版 1.76

2020 年 7 月

- 在 首页增加了关于应用说明随附文件的信息。
- 阐明了 [2.3 命令行接口](#) 和 [3.2 创建适用于引导装载程序 - EFM32 系列 1 的应用](#) 这两节中的一些措辞。
- 整个文档中的其他变更。

修订版 1.75

2020 年 5 月

- 阐明了 [3.2 创建适用于引导装载程序 - EFM32 系列 1 的应用](#) 一节中的一些措辞。

修订版 1.74

2018 年 3 月

- 更改了设备兼容性，以包括 EFM32GG11 和 EFM32TG11 设备。
- 从 EFM32 系列 1 列表中删除了 EFM32PG13 和 EFM32JG13。
- 在 EFR32 系列 1 列表中增加了 EFR32XG14
- 删除了关于引导加载程序对 EFM32GG11-X 工程状态设备的支持的说明。

修订版 1.73

2017 年 10 月

- 更改了设备兼容性，以包括系列 1 设备。
- 更新了与自动波特率算法兼容的波特率。
- 更新了关于在 Teraterm 中进行 XMODEM 传输的措辞。
- 增加了关于使用 UART 引导加载程序时串行终端显示差异的说明。
- 增加了关于显示最新版本引导加载程序的文本文件位置的说明。
- 增加了一些截图，以详细说明如何使用 IAR 创建应用。

修订版 1.72

2017 年 6 月

- 将兼容性列表更名为 [1. 设备兼容性](#) 并添加 EFM32GG11。

修订版 1.71

2017 年 2 月

- 修改了设备参考，以反映系列 0 和系列 1 的命名方案。

修订版 1.70

2015 年 11 月

- 增加了 EFM32PG 和 EFM32JG 设备。
- 增加了关于在 Simplicity Studio 中更改内存设置的说明。
- 区分了 EFM32 系列 0 和 EZR32 系列 0 设备和 EFM32 系列 1 设备的说明。

修订版 1.69

2015 年 3 月

- 增加了 EFM32HG 设备。
- 更新了格式。
- 澄清了各个系列引导装载程序 UART 配置。
- 增加了 IAR 编译指南。

修订版 1.68

2014 年 5 月

- 增加了 EFM32 Wonder Gecko 二进制和链接器文件
- 将 CodeSourcery 和 Eclipse 参考替换为通用 GCC 参考
- 更改为 Silicon Labs 许可证

修订版 1.67

2013 年 10 月

- 新封面布局

修订版 1.66

2013 年 7 月

- 明确说明了哪些部件使用 LEUART0。

修订版 1.65

2013 年 5 月

- 增加了关于编译引导装载程序的章节

修订版 1.64

2012 年 11 月

- 修复了表述错误的说明；EFM32TG110 不包含 USART0。

修订版 1.63

2012 年 4 月

- 修复了程序集文件中缺失的注释 ‘’。

修订版 1.62

2012 年 4 月

- 增加了新的预构建二进制文件，此前的版本不适用于闪存小于 32k 的设备。
- 针对新外围设备库命名和 CMSIS_V3 调整了软件项目。
- 增加了关于如何修改 Eclipse/GCC/Sourcery CodeBench 应用代码链接器文件的章节。

修订版 1.61

2011 年 11 月

- 增加了缺失的 EFM32LG 二进制文件。
- 重命名了应用说明。

修订版 1.60

2011 年 9 月

- 增加了对 EFM32LG 和 EFM32LG 部件的支持。
- 其他引导装载程序无功能差异。

修订版 1.50

2011 年 6 月

- 引导装载程序 USART 从 TG110 和 TG108 设备移动到 F0、F1，与引导装载程序引脚重叠。
- 其他引导装载程序无功能差异。

修订版 1.40

2011 年 4 月

- 引导装载程序引脚从 SWDIO 和 SWDCLK 变更为仅 SWDCLK。
- 修复了调试锁定。
- 增加了对 Tiny 设备的支持。
- 打印版本号与芯片 ID。
- 增加了链接器文件，以自动约束闪存和 RAM 使用量。
- 将部件特定的配置移动到 config.h。
- 将某些功能移动到闪存，以减小 SRAM 的大小。

修订版 1.30

2010 年 9 月

- 使用 DMA 提升了下载速度。
- 改进了高比特率处理。
- 增加了测试用的二进制文件。

修订版 1.22

2010 年 9 月

- 更新了首页主标题，代码未变更。
- 更新了引导装载程序应用说明适用于哪些设备的参考。

修订版 1.21

2010 年 9 月

- 更新了首页主标题，代码未变更。

修订版 1.20

2010 年 8 月

- 提高了编程速度。修复了应用在使用 RTC 时崩溃的问题。该问题是在 1.10 版中引入的。

修订版 1.10

2010 年 4 月

- 引导装载程序当前在空闲时使用 EM2。

修订版 1.00

2010 年 1 月

- 原始版本。

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!



IoT Portfolio

www.silabs.com/IoT



SW/HW

www.silabs.com/simplicity



Quality

www.silabs.com/quality



Support & Community

www.silabs.com/community

Disclaimer

Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice to the product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Without prior notification, Silicon Labs may update product firmware during the manufacturing process for security or reliability reasons. Such changes will not alter the specifications or the performance of the product. Silicon Labs shall have no liability for the consequences of use of the information supplied in this document. This document does not imply or expressly grant any license to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any FDA Class III devices, applications for which FDA premarket approval is required or Life Support Systems without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons. Silicon Labs disclaims all express and implied warranties and shall not be responsible or liable for any injuries or damages related to use of a Silicon Labs product in such unauthorized applications.

Note: This content may contain offensive terminology that is now obsolete. Silicon Labs is replacing these terms with inclusive language wherever possible. For more information, visit www.silabs.com/about-us/inclusive-lexicon-project

Trademark Information

Silicon Laboratories Inc.[®], Silicon Laboratories[®], Silicon Labs[®], SiLabs[®] and the Silicon Labs logo[®], Bluegiga[®], Bluegiga Logo[®], EFM[®], EFM32[®], EFR, Ember[®], Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Redpine Signals[®], WiSeConnect, n-Link, ThreadArch[®], EZLink[®], EZRadio[®], EZRadioPRO[®], Gecko[®], Gecko OS, Gecko OS Studio, Precision32[®], Simplicity Studio[®], Telegesis, the Telegesis Logo[®], USBXpress[®], Zentri, the Zentri logo and Zentri DMS, Z-Wave[®], and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. Wi-Fi is a registered trademark of the Wi-Fi Alliance. All other products or brand names mentioned herein are trademarks of their respective holders.