# 如何开发不带 Flash API 的 Bootloader 实现在线升级

JOHNSON CHEN / EP FAE

## 摘要

在实际应用中，越来越多的产品有在线升级的需求，比如升级功能或者更新软件来解决已知软件 bug,因此需要产品在出厂后能够在线升级或者远程升级。

客户通常会开发 Bootloader 来实现升级的功能。传统的 Bootloader 开发中，Flash API 会作为代码的一部分一直存储在芯片 Flash 里面，或者在芯片的 ROM 里面。在需要在线升级的时候，Bootloader 直接调用 Flash API 来实现 Flash 的擦除，编程，校验等操作。

因此，如果代码开发不够严谨或者因为干扰等原因，有可能使得程序误调用 Flash API，从而出现非预期的 Flash 操作，导致产品无法工作。随着市场对可靠性和安全性要求的提高，如何减少这种可能性变得尤为重要。

本文将提供一种不带 Flash API 的 Bootloader 开发方案，来解决这个问题。

# Contents

# Figures

# 1 介绍

为了解决因代码开发不够严谨或者因为干扰等原因，使得程序误调用 Flash API，从而出现非预期的 Flash 操作，导致产品无法工作的潜在风险。

本文提供一种不带 Flash API 的 Bootloader 开发方案，来解决这个问题。

方案具体原理如下：
Bootloader 本身不带 Flash API，在需要在线升级的时候，Flash API 由上位机通过通信的方式发给 C2000 芯片。Bootloader 从上位机接收到 Flash API 后，将 Flash API 存放到指定的 RAM 空间。之后 Bootloader 再执行相应的 Flash 操作。

下面将以 F280025 为例，介绍如何一步步创建 Flash API 项目，如何生成 Flash API 的符号库，如何提取 Flash API 的二进制代码以及 Bootloader 项目如何调用上位机发下来的 Flash API 代码。同时还将提供基于 F28002x 参考代码。

本文例程 Flash API 项目和 Bootloader 项目都是从下面项目导入:
C:\ti\c2000\C2000Ware_4_00_00_00\driverlib\f28002x\examples\flash\flashapi_ex1_programming

本方案实现假设如下：

1. Bootloader从上位机接收到的Flash API存放到RAMLS6和RAMLS7的0xB000-0xBFEF区间。
2. Flash API使用到的全局变量，分配到RAMLS6和RAMLS7的0xBFF0-0xBFFF区间。

在实际项目开发时，用户可以根据实际需要对上面的RAM空间进行重新分配。

为了方便验证方案的可行性，测试 Bootloader 项目时省略了从上位机接收 Flash API 到芯片 RAM 的过程， 改为 Flash API 提取后，在 Bootloader 项目调试界面中，通过 CCS Memory Browser 窗口直接加载到指定的 RAM 空间。

# 2 创建和配置 Flash API 项目

下面将详细介绍如何创建和配置 Flash API 项目，如何提取 Flash API 二进制代码，如何创建 Flash API 项目的符号库。

## 2.1 创建和配置 Flash API 项目

导入 C:\ti\c2000\C2000Ware_4_00_00_00\driverlib\f28002x\examples\flash\flashapi_ex1_programming 项目后，在项目名上点击右键选择“Rename”(如图 1)将项目重命名为 F28002x_FlashAPI.

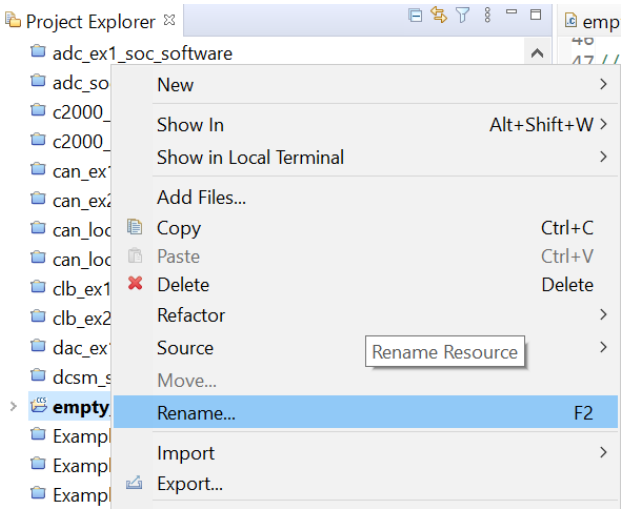


图 1. 项目重命名

更改 28002x_flash_api_lnk.cmd 分配如下：

1. 在 cmd 的 MEMORY 部分，将 RAMLS4567 部分屏蔽，替换成如下

```
// RAMLS4567          : origin = 0x0000A000, length = 0x00002000
RAMLS45               : origin = 0x0000A000, length = 0x00001000
RAMLS67               : origin = 0x0000B000, length = 0x00000FF0
RAM_API_BSS           : origin = 0x0000BFF0, length = 0x00000010
```

2. 在 cmd 的 SECTIONS 部分，将所有 RAMLS4567 部分都替换成 RAMLS45，替换完成后内容如下：

```
.bss              : > RAMLS45
.bss:output       : > RAMLS45
.data             : > RAMLS45
.sysmem           : > RAMLS45
```

```
GROUP
{
    .TI.ramfunc
    { -l FlashAPI_F28002x_FPU32.lib}
}LOAD = FLASH_BANK0_SEC1,
               RUN = RAMLS45,
               LOAD_START(RamfuncsLoadStart),
               LOAD_SIZE(RamfuncsLoadSize),
               LOAD_END(RamfuncsLoadEnd),
               RUN_START(RamfuncsRunStart),
               RUN_SIZE(RamfuncsRunSize),
               RUN_END(RamfuncsRunEnd),
               ALIGN(8)
```

3. 将 GROUP 里面的“-l FlashAPI_F28002x_FPU32.lib”内容屏蔽掉，屏蔽后内容如下：

```
{
   .TI.ramfunc
   //{ -l FlashAPI_F28002x_FPU32.lib}
}LOAD = FLASH_BANK0_SEC1,
```

4. 在 cmd 的 SECTIONS 部分，增加下面内容：

```
Flash_API_RAM   : > RAMLS67,ALIGN(8)
              {
                      -l FlashAPI_F28002x_FPU32.lib(.text)
              }
FLASH_API_BSS   : > RAM_API_BSS
               {
                      -l     FlashAPI_F28002x_FPU32.lib (.bss)
               }
```

上面 Flash_API_RAM 段的目的是把 FlashAPI_F28002x_FPU32 里的代码分配到 RAMLS67 里面， 这样可以将 Flash API 函数分配到指定的 RAM 空间，以便后面 Bootloader 项目调用，Flash_API_RAM 段名后面也会使用到。

FLASH_API_BSS 段分配的目的是把 FlashAPI_F28002x_FPU32 使用到的全局变量分配到 RAM_API_BSS 里面，在 Bootloader 项目中会保留这部分空间，从而保证这部分 RAM 空间不会被其它代码占用。

## 2.2 如何提取 Flash API 二进制代码

第一步：编译 F28002x_FlashAPI 项目。

第二步：确保板子上电，仿真器正常连接后，点击  ，对 F28002x_FlashAPI 项目进行调试。

第三步：进入调试界面后，保存 Flash API 为 FlashAPI_RAM.txt 文件。

a. 点击“View->Memory Browser”，如图 2 所示，选择“Save Memory”。

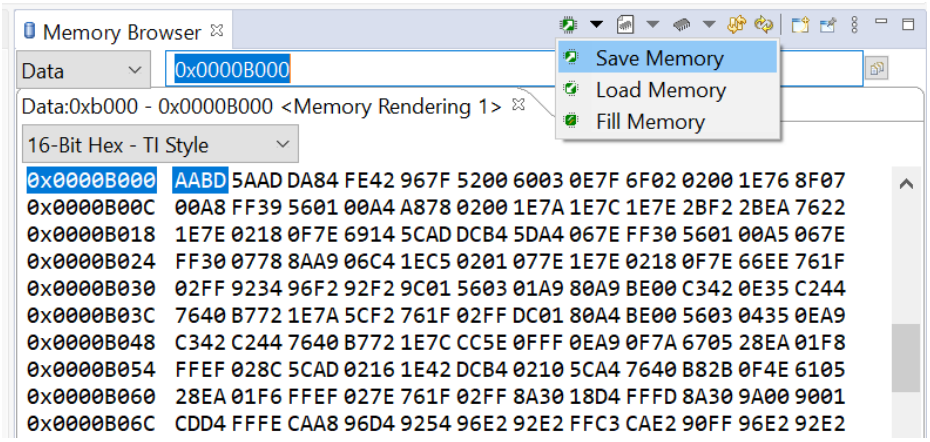


图 2. 选择 Save Memory

b. 在弹出窗口中，如图 3 所示，选择保存文件名及路径。文件格式选择 TI Data，输入保存文件名为 FlashAPI_RAM.txt。然后点击“Next”。

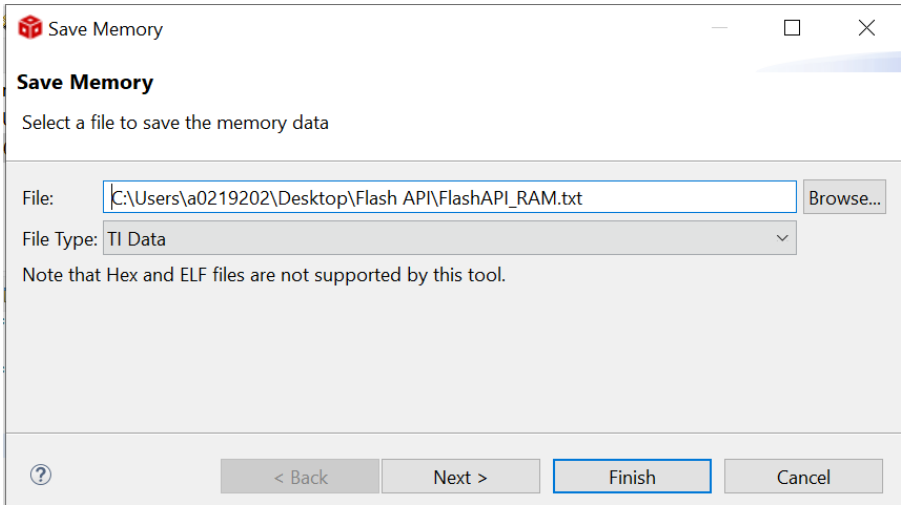


图 3. Save Memory 窗口

c. 在弹出窗口中，如图 4 所示，选择格式为 16-Bit Hex-TI Style，起始地址为 0xB000，长度为 0x1000，然后点击“Finish”完成 FlashAPI_RAM.txt 保存。

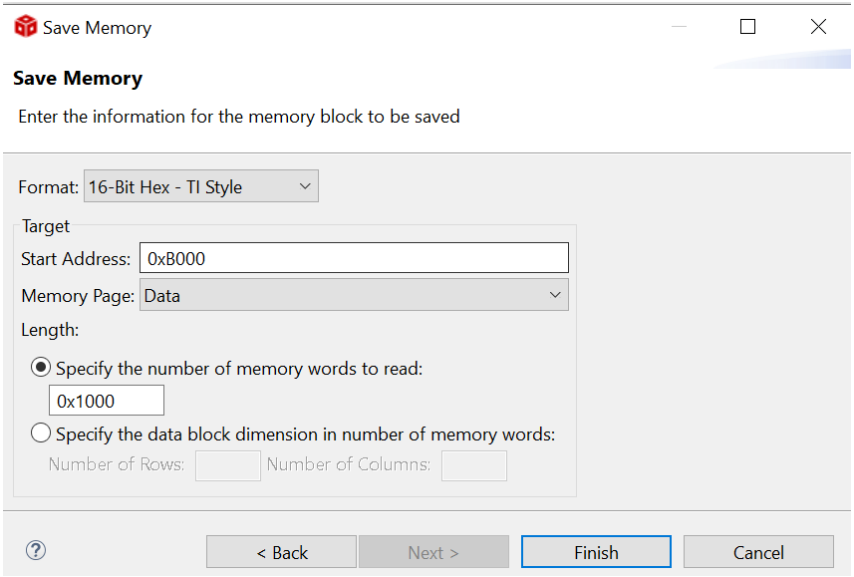


图 4. Save Memory 设置

这个 FlashAPI_RAM.txt 文件即为 Flash API 的二进制代码。上位机可以解析这个 FlashAPI_RAM.txt 文件， 然后把有效数据通过通讯方式发送给 C2000。

这个 FlashAPI_RAM.txt 文件后面测试验证时也会使用到。

## 2.3 如何创建 Flash API 项目的符号库

符号库只会包含函数的地址信息而不会包含具体二进制代码，符号库生成后，只需要把 F28002x_FlashAPI 项目的符号库添加到 Bootloader 项目进行编译就可以了。

下面为生成符号库的具体方法：

a. 如图 5 所示，在 F28002x_FlashAPI 项目中选择“Show Build Settings ->C2000 Linker -> Linker Output ”在 Detailed link information data-base into <file> (--xml_link_info) 里面输入 F28002x_FlashAPI.xml（文件名可以根据需要，自己设置）。

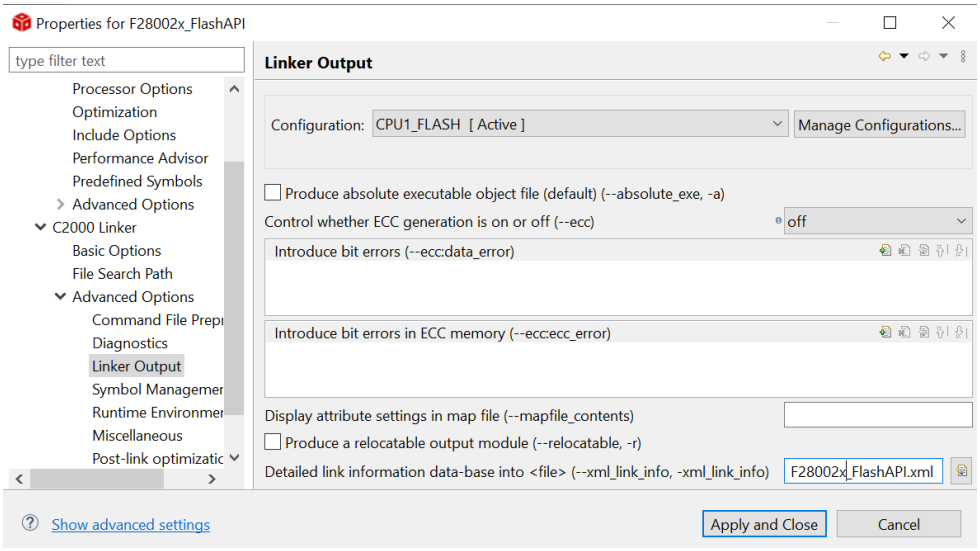


图 5. F28002x_FlashAPI 项目 Linker Output 设置

b. 然后编译 F28002x_FlashAPI 项目，编译通过后会生成 F28002x_FlashAPI.xml，F28002x_FlashAPI.xml 在后面的步骤里会用到。

c. 需要安装 Strawberry PERL 软件(free -- can download from Strawberry Perl website: https://strawberryperl.com/)。

d. 下载并安装最新的 Code Gen Tools XML Perl scripts utility: https://software-dl.ti.com/ccs/non-esd/releases/other/applications_packages/cg_xml/index.htm。

e. 创建一个文件夹，例如 test，将 F28002x_FlashAPI 项目编译生产的 F28002x_FlashAPI.xml 拷贝到这个路径下。

f. 在上面新建的文件夹下，新建一个 F28002x_FlashAPI_symbollibrary.txt 文件，将文件名改成 F28002x_FlashAPI_symbollibrary.bat。然后将下面命令行拷贝到 F28002x_FlashAPI_symbollibrary.bat 文件里。

```
PATH = "C:\ti\ti-cgt-
c2000_20.2.1.LTS\bin";C:\Strawberry\perl\bin;C:\Strawberry\c\bin;C:\Strawberry\perl\site\bin;

del *tmp*.asm

del *tmp*.obj

del *.lib

perl C:\TEST\FlashAPI\cgxml-2.61.00\map\get_rom_symbols.pl F28002x_FlashAPI.xml -s=
Flash_API_RAM

cl2000 --float_support=fpu32 --tmu_support=tmu0 --abi=eabi -g -pdsw225 -d"_DEBUG" -
d"LARGE_MODEL" -ml -v28 *.asm

ar2000 r  F28002x_FlashAPI_SymbolLibrary.lib *.obj

del *tmp*.obj *.asm
```

注意：

1. 用户实际使用的话，可以根据实际需要更改 C2000 编译器路径和文件名(F28002x_FlashAPI.xml 以及 F28002x_FlashAPI_SymbolLibrary.lib)，Flash_API_RAM 段是在前面 F28002x_FlashAPI 项目 cmd 里定义的。
2. 这个路径需要根据实际安装路径进行修改 `C:\TEST\FlashAPI\cgxml-2.61.00\map\get_rom_symbols.pl`
3. PATH 这行的后面几个路径中间不能有空格。

g.双击 F28002x_FlashAPI_symbollibrary.bat 文件，这样就生成了 Flash API 的符号库 F28002x_FlashAPI_SymbolLibrary.lib。

# 3 Bootloader 项目如何调用上位机发下来的 Flash API

在本方案中，Flash API 会从上位机发下来，因此没必要把 Flash API 的库加到 Bootloader 项目里面， Bootloader 项目只需要知道 Flash API 函数的地址就可以了。

Bootloader 项目可以通过下面方法来调用上位机发下来的 Flash API。

方法一：函数绝对地址调用

下面以 Flash API 的 Fapi_setActiveFlashBank()函数为例，介绍如何实现绝对地址调用。

我们需要在 F28002x_FlashAPI 项目的 map 文件中，查找 Fapi_setActiveFlashBank 的地址，如图 6 所示，地址为 0x0000b717。

```
3120     0000b6ae  Fapi_issueAsyncCommandWithAddress
3130     0000b2f6  Fapi_issueProgrammingCommand
3140     0000b717  Fapi_setActiveFlashBank
3150     0000b6e6  Fapi_setupBankSectorEnable
3160     0000a238  Flash_initModule
```

图 6. Fapi_setActiveFlashBank 地址

接下来我们创建一个 FlashAPI.h 文件，在 FlashAPI.h 里面，可以用下面方法定义 Fapi_setActiveFlashBank()函数绝对地址：

```
#define Fapi_setActiveFlashBank (Fapi_StatusType(*)( Fapi_FlashBankType oNewFlashBank)) 0x0000b717
```

通常我们可以通过下面方法来调用 Fapi_setActiveFlashBank()函数：

```
oReturnCheck = Fapi_setActiveFlashBank(Fapi_FlashBank0);
```

但在绝对地址调用中，我们需要使用下面函数指针的方法来调用 Fapi_setActiveFlashBank()函数：

```
oReturnCheck = (*Fapi_setActiveFlashBank)(Fapi_FlashBank0);
```

实际使用时，我们要把所有使用到的 Flash API 函数，按照上面方法一一在 FlashAPI.h 中进行绝对地址定义，然后将 FlashAPI.h 包含到源文件中，并在源文件中将要调用的 Flash API 函数一一改成函数指针的方式来调用。

绝对地址函数调用的方法，需要从编译完成的项目的 map 文件中，手动查找相应函数的具体地址，如果函数较多，这将会是不小的工作量。另外，如果源文件有改动的话，编译后函数地址会改变，因此需要用户重新去 map 文件里面手动查找并更新函数的地址，这样不利于项目维护。

方法二：使用符号库来调用

前面已经介绍如何生成符号库，使用符号库的方式，不需要对项目中的源文件进行修改。只需将前面生成的 F28002x_FlashAPI_SymbolLibrary.lib 添加到 F28002x_Bootloader 项目中进行编译即可。

使用符号库调用的方法可以自动生成函数的地址信息，因此具有更好的可维护性和灵活性，对于函数数量较多的场景来说，还可以大幅减少工作量。

下面以使用符号库方式为例进行测试验证。

# 4 方案测试验证

## 4.1 创建和配置 Bootloader 项目

第一步：导入 C:\ti\c2000\C2000Ware_4_00_00_00\driverlib\f28002x\examples\flash\flashapi_ex1_programming 项目后，在项目名上点击右键选择“Rename”将项目重命名为 F28002x_Bootloader。

第二步：在项目中把 FlashAPI_F28002x_FPU32.lib 文件删除掉。

第三步：复制 F28002x_FlashAPI 项目里的 cmd 替换掉 28002x_flash_api_lnk.cmd 里的内容。

第四步：屏蔽 cmd SECTION 里面的 Flash_API_RAM 和 FLASH_API_BSS 内容，如下：

```
//Flash_API_RAM        : > RAMLS67,ALIGN(8)
 //            {
//                    -l FlashAPI_F28002x_FPU32.lib(.text)
//              }

//FLASH_API_BSS        : > RAM_API_BSS
//            {
//                    -l     FlashAPI_F28002x_FPU32.lib (.bss)

//            }
```

第五步：把 F28002x_FlashAPI_SymbolLibrary.lib 添加到 F28002x_Bootloader 项目中进行编译。

## 4.2 Bootloader 方案测试及验证

第一步：点击  ，进入 F28002x_Bootloader 项目的调试界面。

第二步：将之前保存的 FlashAPI_RAM.txt 文件加载到 RAMLS67(请注意此步骤在用户实际项目中不需要， 而是由 Bootloader 从上位机接收 Flash API， 然后放到 RAMLS67 里面的过程来替代)。

a. 如图 7 所示，在“View->Memory Browser”窗口选择“load Memory”。

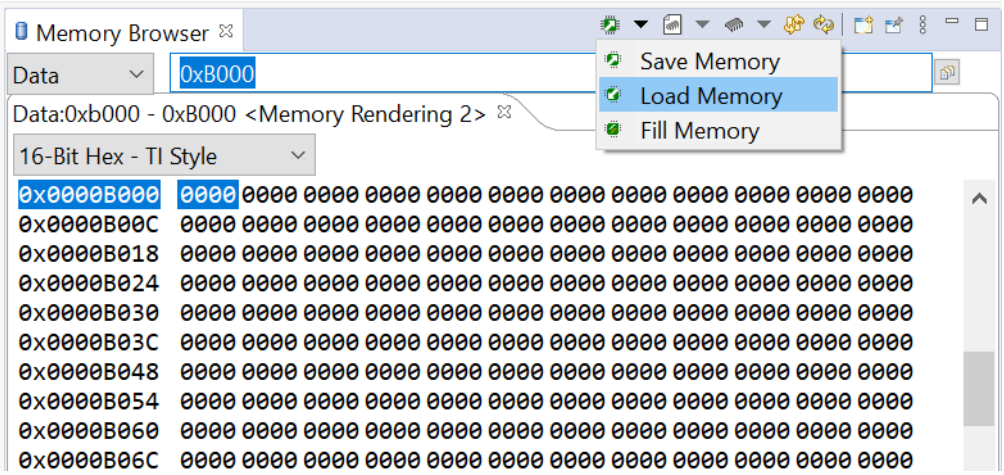


图 7. 选择 Load Memory

b. 如图 8 所示，在弹出窗口中选择之前保存的 FlashAPI_RAM.txt 文件，文件格式选 TI Data，然后点击“Next”。

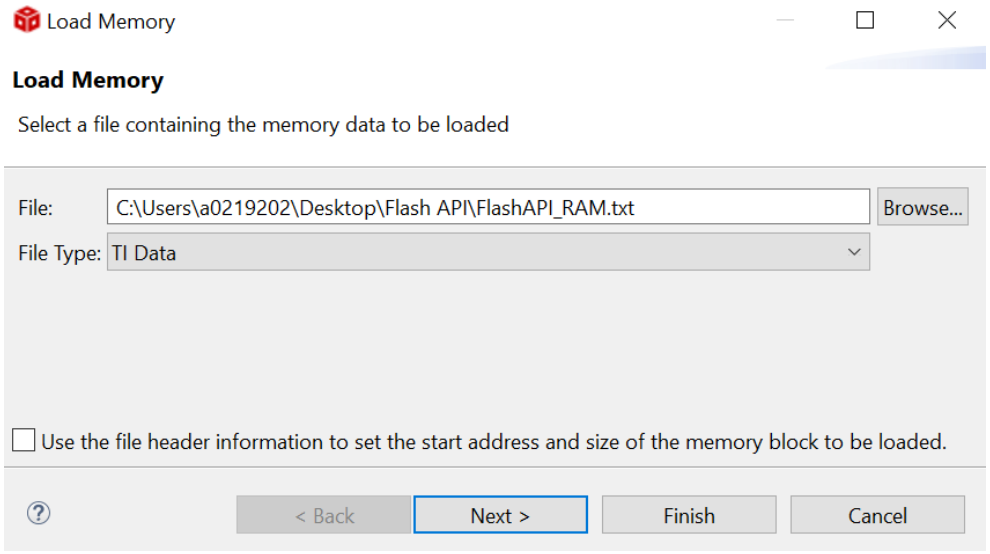


图 8. Load Memory 窗口

c. 如图 9 所示，在弹出窗口中输入起始地址为 0xB000, 长度为 0x1000，然后点击完成。

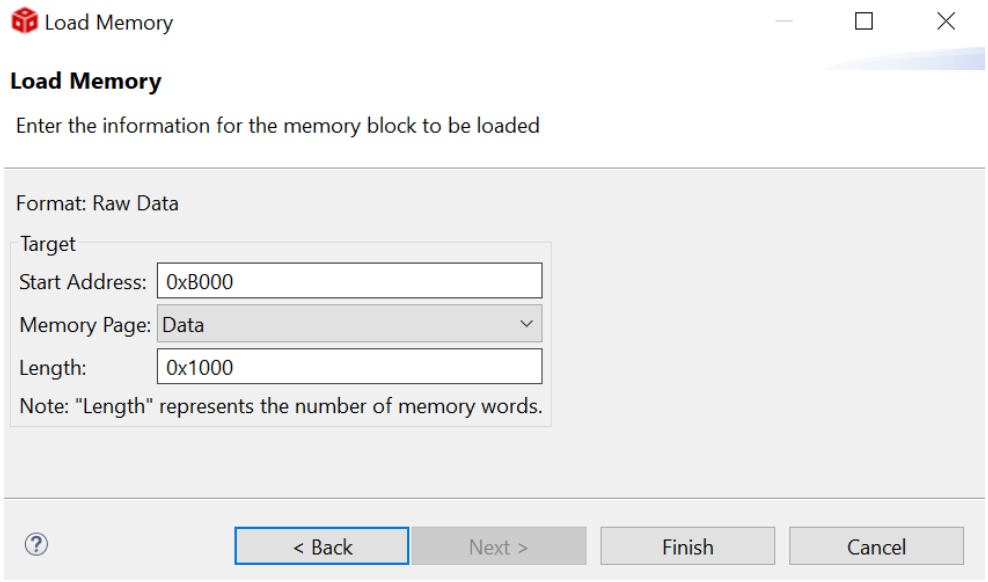


图 9. Load Memory 设置

第三步：点击  ，程序正常运行，PC 最后停在 Example_Done()里面， 说明 Flash 擦除，编程及校验等各项操作均正常完成。

因此验证了方案的可行性。

# 5 结论

本文介绍了如何一步步创建 Flash API 项目，如何生成 Flash API 的符号库，如何提取 Flash API 二进制代码以及 Bootloader 项目如何调用上位机发下来的 Flash API 代码。

基于 F280025 的例程，也验证本文方法的可行性。此方法适用于所有 C2000 产品，例如 F28003x, F28004x, F2807x, F2837x, F2838x 等。

# 6 参考文献

1. TMS320F280025 Datasheet (SPRSP45)

2. TMS320F280025 Technical Reference Manual (SPRUIN7)

# 重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265