GigaDevice Semiconductor Inc.

Arm® Cortex®-M3/4/23/33 32-bit MCU

应用笔记

AN017

目录

表索引

图索引

1. 简介

本应用笔记采用 GD32F450i-EVAL 开发板，目标芯片为 GD25Q16BS SPI nor flash 芯片，通过 J-FLASH SPI 上位机或者修改 KEIL 下载算法，将文件到 GD25Qxx SPI nor flash 中。

2. 使用 J-Flash SPI 上位机下载文件到 SPI Nor Flash

2.1. 硬件连接

JLink 支持 SPI 协议，将 JLink 中的六根线 VTref、GND、TDI（MOSI）、TMS（nCS）、TCK（CLK）、TDO（MISO）对应连接到 SPI Nor Flash 的引脚上。本应用笔记采用 GD32F450i-EVAL V1.1 开发板中的 GD25Q16BS SPI nor flash 芯片，根据开发板原理图和 JTAG 引脚图，如*图 2-1. GD25Q16BS 原理图（左）与 JTAG 引脚图（右）*所示，使用杜邦线将 Jlink 与 Flash 硬件连接，连接方法参考*表 2-1. Jlink 与 SPI Flash 硬件连接*。

图 2-1. GD25Q16BS 原理图（左）与 JTAG 引脚图（右）

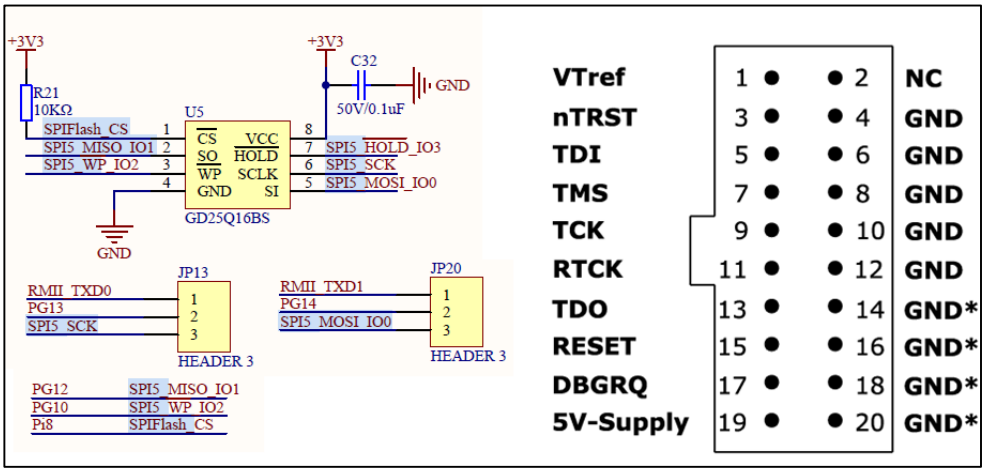


表 2-1. Jlink 与 SPI Flash 硬件连接

JTAG 引脚编号及名称	连接到 GD25Q16BS 的引脚
1(VTref)	开发板 VCC
5(TDI)	开发板 JP20 3 号引脚（MOSI）
7(TMS)	开发板 PI8 引脚(CS)
9(TCK)	开发板 JP13 3 号引脚（SCK）
13(TDO)	开发板 PG12 引脚(MISO)
4(GND)	开发板 GDN

2.2. Jflash-SPI 上位机配置与下载

首先单击打开 J-Flash SPI，如图 2-2 所示，打开界面见*图 2-2. SEGGER 中 J-Flash SPI 软件*。

图 2-2. SEGGER 中 J-Flash SPI 软件

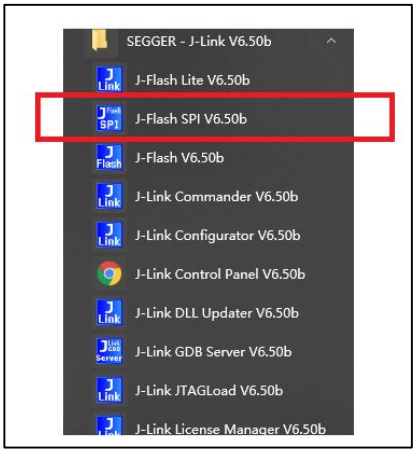
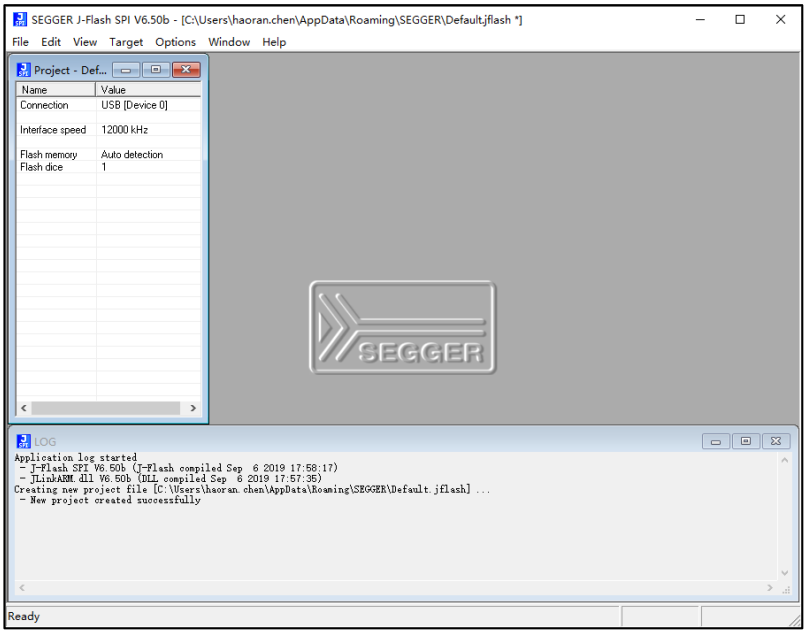
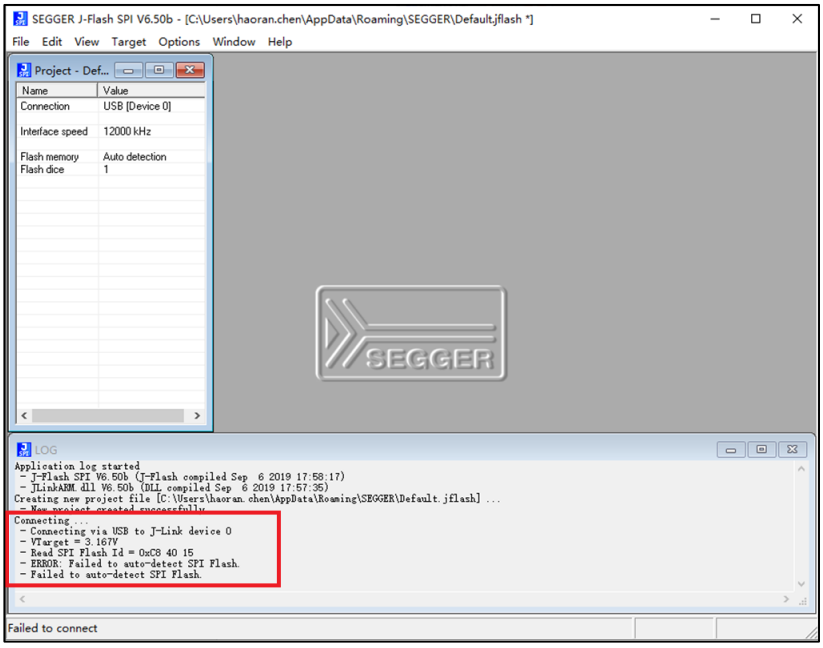


图 2-3. 打开 J-Flash SPI 软件界面



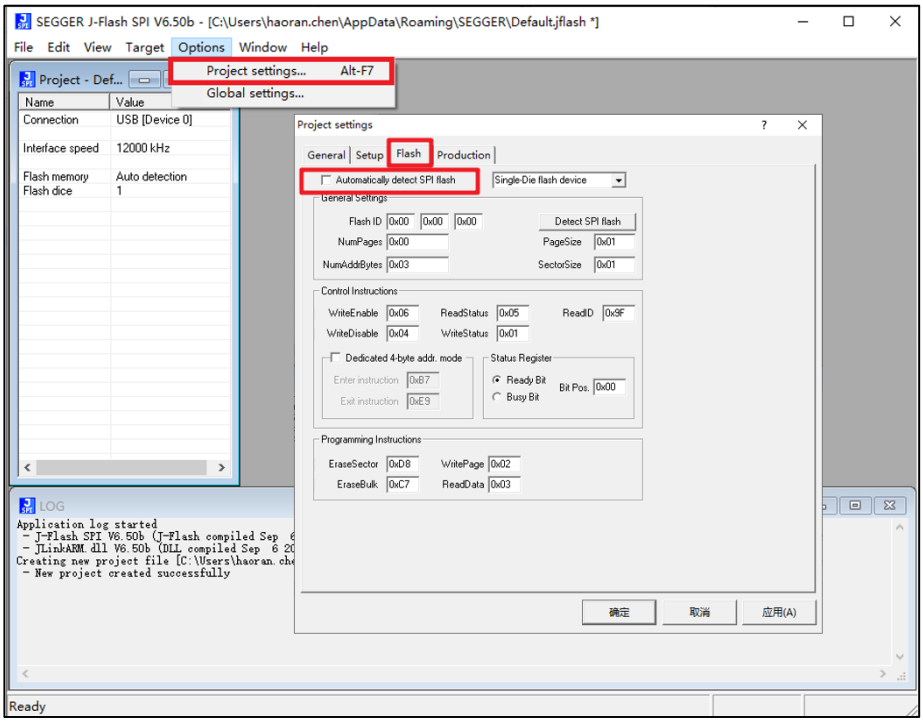
点击“Target-->Connect”，可以看到*图 2-3. 打开 J-Flash SPI 软件界面*中红色框图，此时已成功读出 SPI Flash Id，但是连接失败，接下来将配置 Flash 相关参数。

图 2-4. 连接目标 SPI Flash



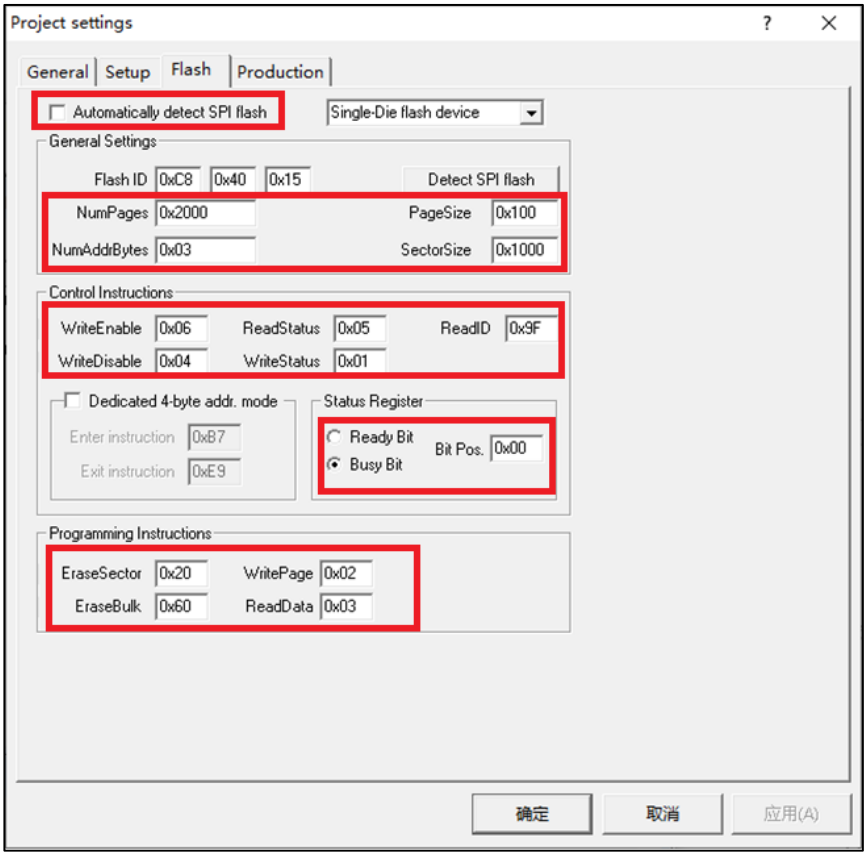
点击“Options-->Project settings”，选择 FLASH，将 Automatically detect SPI flash 取消勾选，参考界面*图 2-5. SPI Flash 配置界面*。

图 2-5. SPI Flash 配置界面



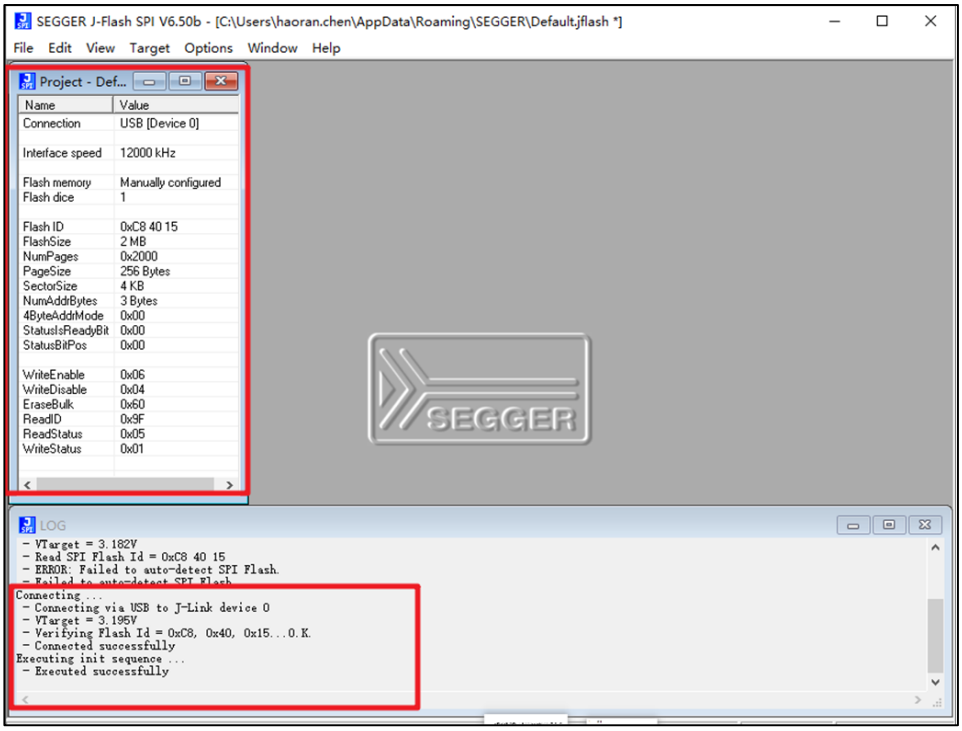
参考 GD25Q16B datasheet，填写 Flash 页大小、块大小、读写命令等相关参数，具体配置参考*图 2-6. GD25Q16B 参数配置*，配置完成后点击确定。

图 2-6. GD25Q16B 参数配置



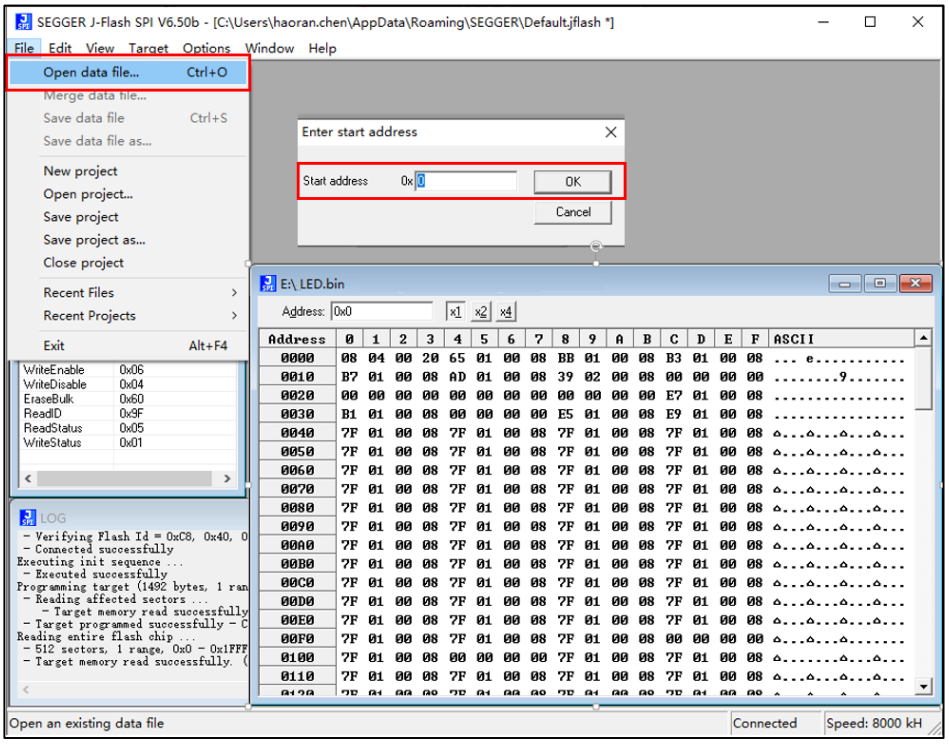
在主界面中再次点击“Target-->Connect”，可以看到*图 2-7. JLink 成功连接 SPI Flash* 所示红框，此时在左边显示相关参数，并且提示 JLink 与 Flash 连接成功。

图 2-7. JLink 成功连接 SPI Flash



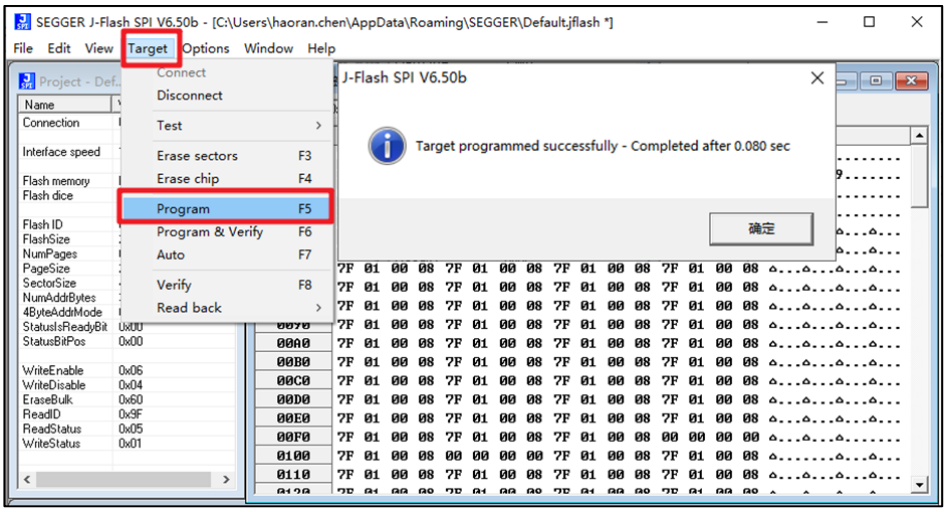
点击“File—>open data file”，打开需要下载的二进制文件，如*图 2-8. 打开下载的二进制文件*所示。

图 2-8. 打开下载的二进制文件



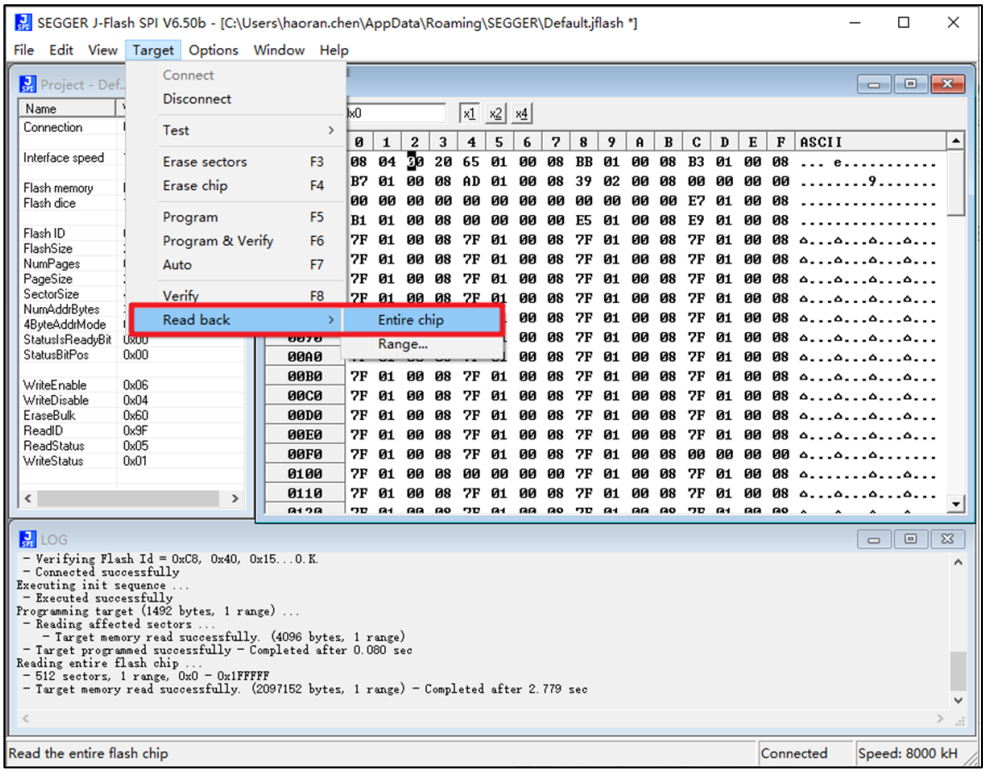
点击“Target-->Program”，如*图 2-9. 文件下载到 Flash 成功提示*所示，下载完成后，提示 Target programmed successfully。

图 2-9. 文件下载到 Flash 成功提示



为了验证是否成功下载二进制文件到 Flash 中，可以通过“Target-->Read back-->Entire chip”操作，读出所在地址的值，与源文件进行比对，如*图 2-10. 读取 Flash 中的数据*所示。

图 2-10. 读取 Flash 中的数据

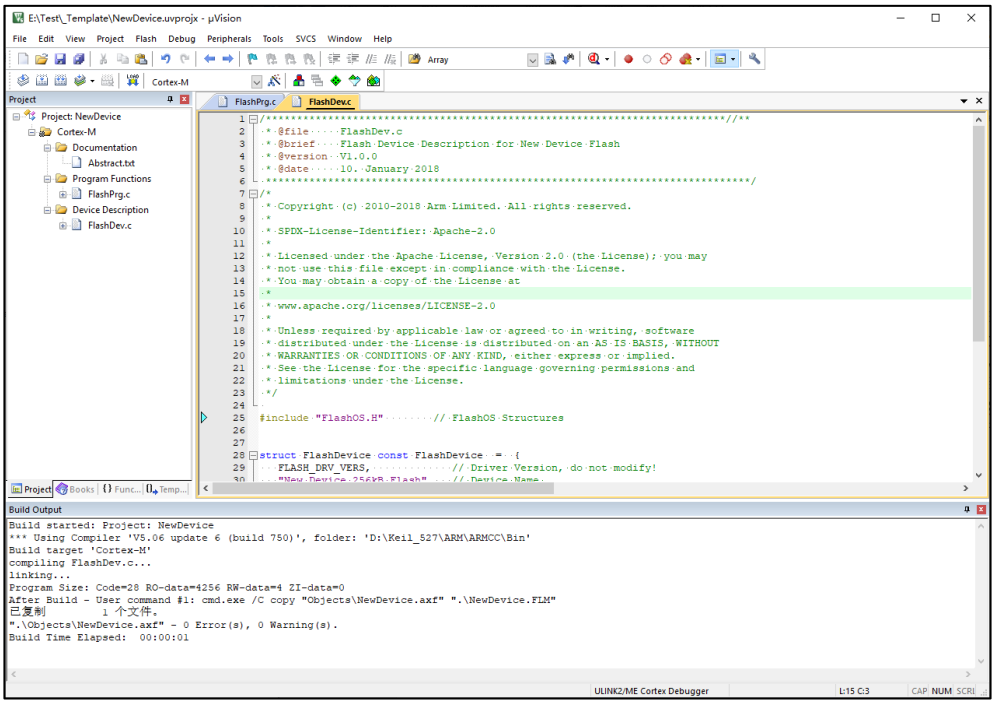


3. 使用 KEIL 下载文件到 SPI Nor Flash

3.1. 新建 FLM 工程

进入安装 KEIL 的盘符，将 Keil\ARM\Flash_Template 工程拷贝到 E 盘 Test 文件夹中（可自己设定），双击打开“NewDevice.uvprojx”工程，编译该工程，工程会报错“FlashDev.c(25): error: #5: cannot open source input file "..\FlashOS.H": No such file or directory”，再次进入 Keil\ARM\Flash 目录下找到“FlashOS.h”文件，拷贝到“E:\Test_Template”目录下，将 FlashDev.c 和 FlashPrg.c 中的 #include "../FlashOS.H" 修改为 #include "FlashOS.H"，再次编译工程，工程没有错误，并生成 NewDevicec.FLM。相关工程及编译如*图 3-1. 新建 FLM 工程*所示。

图 3-1. 新建 FLM 工程



3.2. 移植 SPI Flash 驱动代码

打开 FlashPrg.c 文件，该文件主要包含七个函数接口，如*表 3-1. FlashPrg.c 函数接口*所示

表 3-1. FlashPrg.c 函数接口

```
/* Flash Programming Functions (Called by FlashOS) */
extern          int   Init        (unsigned long adr,      /* Initialize Flash */
                                   unsigned long clk,
                                   unsigned long fnc);
```

```
extern              int   UnInit        (unsigned long fnc);      /* De-initialize Flash */
extern              int   BlankCheck    (unsigned long adr,       /* Blank Check */
                                         unsigned long sz,
                                         unsigned char pat);
extern              int   EraseChip     (void);                   /* Erase complete Device */
extern              int   EraseSector (unsigned long adr);        /* Erase Sector Function */
extern              int   ProgramPage (unsigned long adr,         /* Program Page Function */
                                         unsigned long sz,
                                         unsigned char *buf);
extern unsigned long Verify          (unsigned long adr,          /* Verify Function */
                                         unsigned long sz,
                                         unsigned char *buf);
```

这里主要实现 Init、EraseChip、EraseSector、ProgramPage 和 Verify 函数接口，函数接口实现如*表 3-2. FlashPrg.c 函数接口的实现*所示。

表 3-2. FlashPrg.c 函数接口的实现

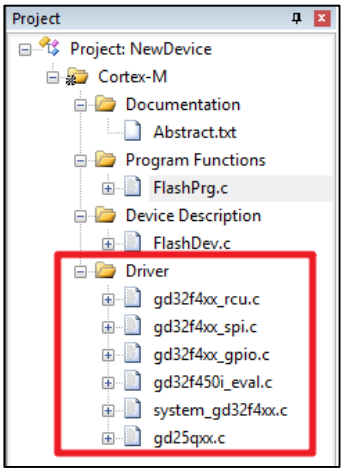
```
uint32_t base_adr;
/*
 *   Initialize Flash Programming Functions
 *      Parameter:          adr:   Device Base Address
 *                          clk:   Clock Frequency (Hz)
 *                          fnc:   Function Code (1 - Erase, 2 - Program, 3 - Verify)
 *      Return Value:     0 - OK,   1 - Failed
 */
int Init (unsigned long adr, unsigned long clk, unsigned long fnc) {

  /* Add your Code */
  spi_flash_init();
  base_adr = adr;
  return (0);                                   /* Finished without Errors */
}
/*
 *   Erase complete Flash Memory
 *      Return Value:     0 - OK,   1 - Failed
 */
int EraseChip (void) {

  /* Add your Code */
  spi_flash_bulk_erase();
  return (0);                                   /* Finished without Errors */
}
/*
 *   Erase Sector in Flash Memory
```

```
 *       Parameter:          adr:   Sector Address
 *       Return Value:     0 - OK,    1 - Failed
 */
int EraseSector (unsigned long adr) {

  /* Add your Code */
  spi_flash_sector_erase(adr);
  return (0);                                       /* Finished without Errors */
}
/*
 *   Program Page in Flash Memory
 *       Parameter:          adr:   Page Start Address
 *                           sz:     Page Size
 *                           buf:   Page Data
 *       Return Value:     0 - OK,    1 - Failed
 */

int ProgramPage (unsigned long adr, unsigned long sz, unsigned char *buf) {

  /* Add your Code */
  spi_flash_page_write(buf,adr,sz);
  return (0);                                       /* Finished without Errors */
}

unsigned long Verify (unsigned long adr, unsigned long sz, unsigned char *buf)
{
  uint8_t readbuf[256];
  uint32_t len;
  uint32_t count = 0;
  uint32_t readcount = 0;
  uint32_t readaddrs = 0;
  if((sz%256)==0)
  {
    readcount = sz/256;
  }else
  {
    readcount = sz/256 + 1;
  }
  readaddrs = (adr - base_adr);
  for(count=0;count<readcount;count++)
  {
    spi_flash_buffer_read(readbuf,(readaddrs+count*256),256);
    for(len=0;len<256;len++)
```

```
    {
      if(buf[len+count*256] != readbuf[len])
      {
        return count*256 + adr + len;
      }
    }
  }
  return adr+sz;
}
```

相关 SPI 驱动根据 GD32F4xx_Firmware_Library 和 GD25qxx.c 添加到 KEIL 工程中，所添加的文件如*图 3-2 移植 SPI 驱动和 GD25qxx 文件*所示。

图 3-2 移植 SPI 驱动和 GD25qxx 文件



3.3. 修改 FlashDevice 结构体

打开 FlashDev.c 文件，修改 FlashDevice 结构体中的相关内容，修改后代码如*表 3-3. FlashDevice 结构体实现*所示。

表 3-3. FlashDevice 结构体实现

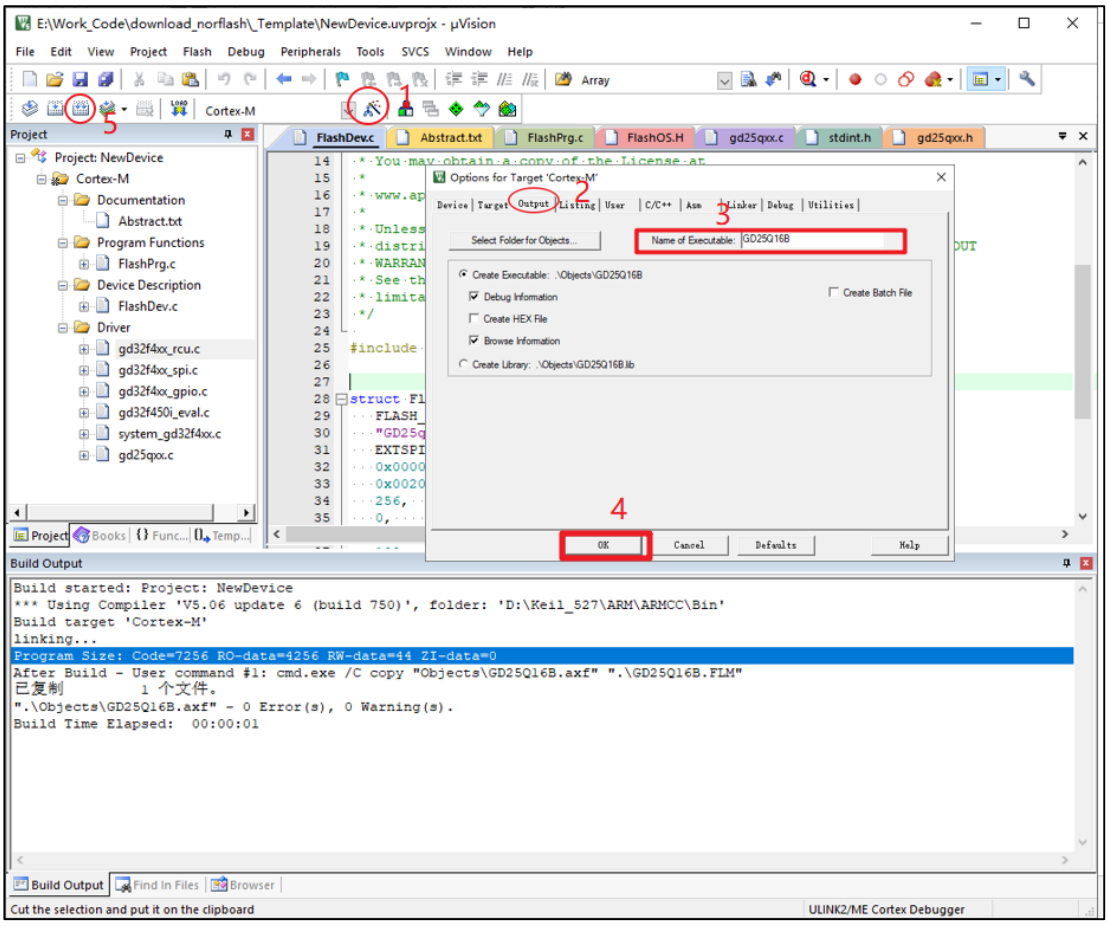
```
struct FlashDevice const FlashDevice  =  {
    FLASH_DRV_VERS,          /* Driver Version, do not modify! */
    "GD25qxx",               /* Device Name */
    EXTSPI,                  /* Device Type */
    0x00000000,              /* Device Start Address */
    0x00200000,              /* Device Size in Bytes (2M) */
    256,                     /* Programming Page Size */
    0,                       /* Reserved, must be 0 */
    0xFF,                    /* Initial Content of Erased Memory */
    100,                     /* Program Page Timeout 100 mSec */
    3000,                    /* Erase Sector Timeout 3000 mSec */
```

```
/* Specify Size and Address of Sectors */
    0x001000, 0x000000,          /* Sector Size   4kB (4096 Sectors) */
//    0x010000, 0x010000,        /* Sector Size 64kB (2 Sectors) */
//    0x002000, 0x030000,        /* Sector Size   8kB (8 Sectors) */
    SECTOR_END
};
```

3.4. 编译生成 FLM 文件

打开魔术棒，进入 Output 页面，修改 Name of Executable 为 GD25Q16B,编译工程，生成 GD25Q16B.FLM 文件。如*图 3-3. 编译生成 GD25Q16B.FLM 文件*所示。

图 3-3. 编译生成 GD25Q16B.FLM 文件



3.5. 添加算法文件到 KEIL 工程中

将编译生成好的 GD25Q16B.FLM 拷贝到 KEIL 安装目录下，D:\Keil_527\ARM\PACK\GigaDevice\GD32F4xx_DFP\2.0.0\Flash，接着返回上级目录打开 GigaDevice.GD32F4xx_DFP.pdsc 文件，将其属性修改为可读写，找到 GD32F450IK，添加代

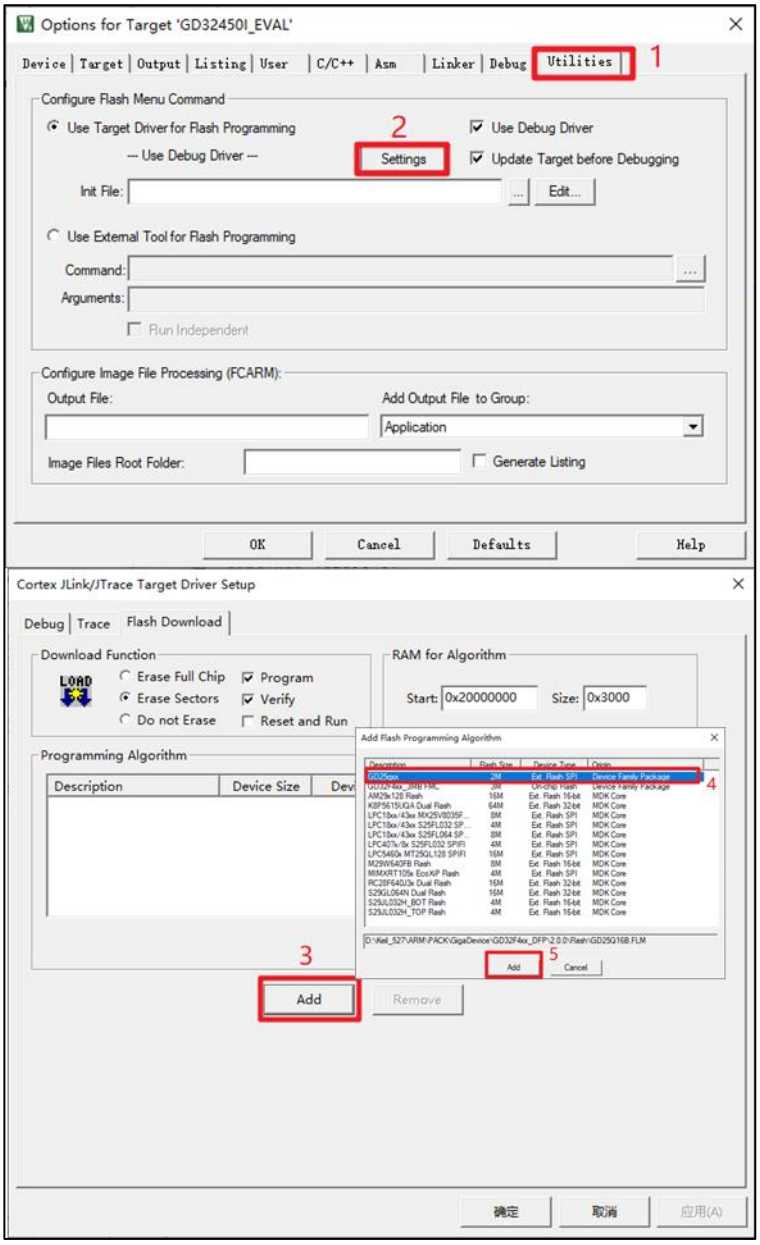
码如**表 3-4. 修改 pdsc 文件代码**红色所示：

表 3-4. 修改 pdsc 文件代码

```
<!-- ****************** Device 'GD32F450IK'   ****************** -->
<device Dname="GD32F450IK">
<memory   id="IROM1"    start="0x08000000" size="0x0300000" startup="1" default="1"/>
<memory   id="IRAM1"    start="0x20000000" size="0x030000" init ="0" default="1"/>
<memory   id="IRAM2" start="0x10000000" size="0x010000" init ="0" default="0"/>
<algorithm name="Flash/GD32F4xx_3MB.FLM" start="0x08000000" size="0x0300000" default="1"/>
<algorithm name="Flash/GD25Q16B.FLM"    start="0x00000000" size="0x01000000" default="1"/>
</device>
```

在 KEIL 工程，打开魔术棒 Utilities 页面 setting，添加 GD25qxx 算法，如***图 3-4. KEIL 中添加 GD25Qxx 下载算法***所示。

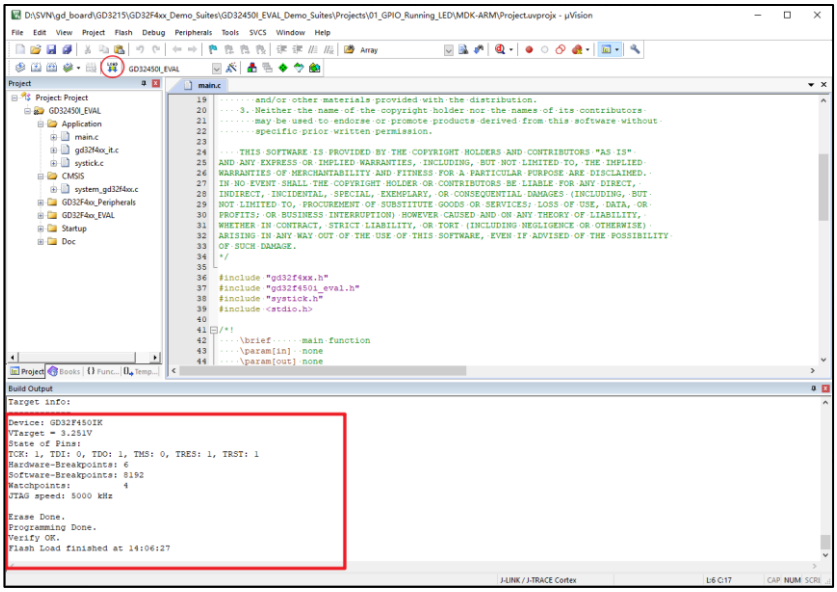
图 3-4. KEIL 中添加 GD25Qxx 下载算法



3.6. 编译和下载

在 KEIL 中编译工程，生成.axf 文件，点击 Download 按钮实现文件的下载，如*图 3-5. KEIL 中编译和下载文件到 SPI Flash* 所示，提示下载成功。

图 3-5. KEIL 中编译和下载文件到 SPI Flash



3.7. 测试和验证

为了确定文件是否成功下载到 GD25Q16BS Flash 中，参考 ***Jflash-SPI 上位机配置与下载***，通过 J-Flash SPI 上位机读取 Flash 中的数据，比较下载文件和读出文件是否相同，进行验证测试。

4. 历史版本

表 4-1. 历史版本

版本号.	描述	日期
1.0	首次发布	2021 年 04 月 30 日

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company under the intellectual property laws and treaties of the People’s Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as it’s suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as it’s suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.