GigaDevice Semiconductor Inc.

Arm® Cortex®-M3/4/23/33 32-bit MCU

Application Note
AN017

Table of Contents

List of Figures

List of Table

1. Introduction

This application note uses the GD32F450i-EVAL board, the target chip is GD25Q16BS SPI nor flash, and the file is downloaded to the GD25Qxx SPI nor flash through the J-FLASH SPI host computer or modified KEIL download algorithm.

2. Use J-Flash SPI host computer to download files to SPI Nor Flash

2.1. Hardware connection

JLink supports the SPI protocol and connects the six wires: VTref, GND, TDI (MOSI), TMS (nCS), TCK (CLK), and TDO (MISO) in JLink to the pins of SPI Nor Flash. This application note uses the GD25Q16BS SPI nor flash chip in the GD32F450i-EVAL V1.1 development board. According to the schematic diagram of the development board and the JTAG pin diagram, as shown in ***Figure 2-1. GD25Q16BS schematic diagram (left) and JTAG pin diagram (right)***, use the DuPont cable to connect the Jlink and the Flash hardware. The method is shown in ***Table 2-1. Jlink and SPI Flash hardware connection***.

Figure 2-1. GD25Q16BS schematic diagram (left) and JTAG pin diagram (right)

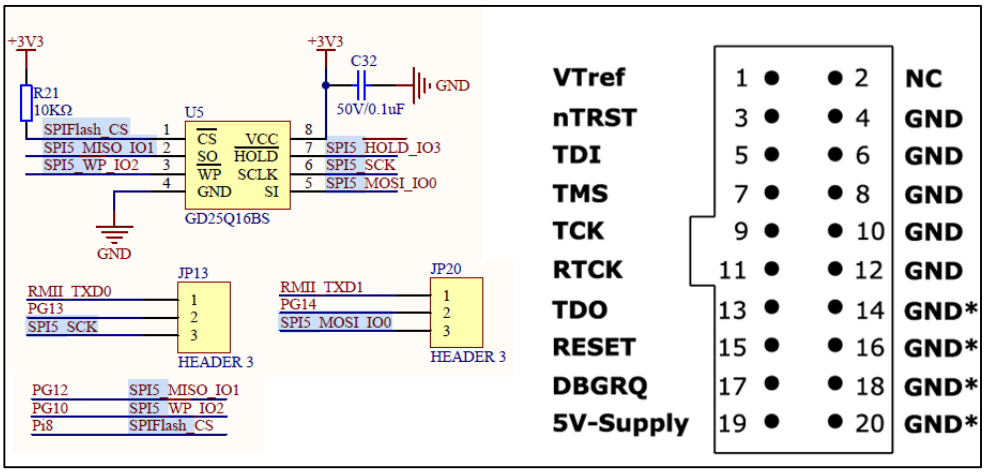


Table 2-1. Jlink and SPI Flash hardware connection

JTAG pin number and name	Connect to the pins of GD25Q16BS
1(VTref)	Board VCC
5(TDI)	Board JP20 No. 3 pin (MOSI)
7(TMS)	Board PI8 pin(CS)
9(TCK)	Board JP13 No. 3 pin (SCK)
13(TDO)	Board PG12 pin(MISO)
4(GND)	Board GND

2.2. Jflash-SPI host computer configuration and download

First, double-click to open J-Flash SPI, as shown in ***Figure 2-2. J-Flash SPI software in SEGGER***, and open the interface as shown in ***Figure 2-3. Open the J-Flash SPI software interface***.

Figure 2-2. J-Flash SPI software in SEGGER

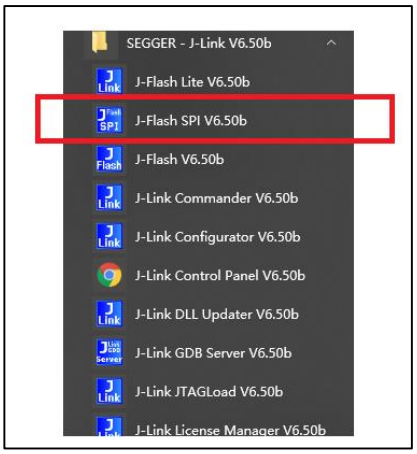
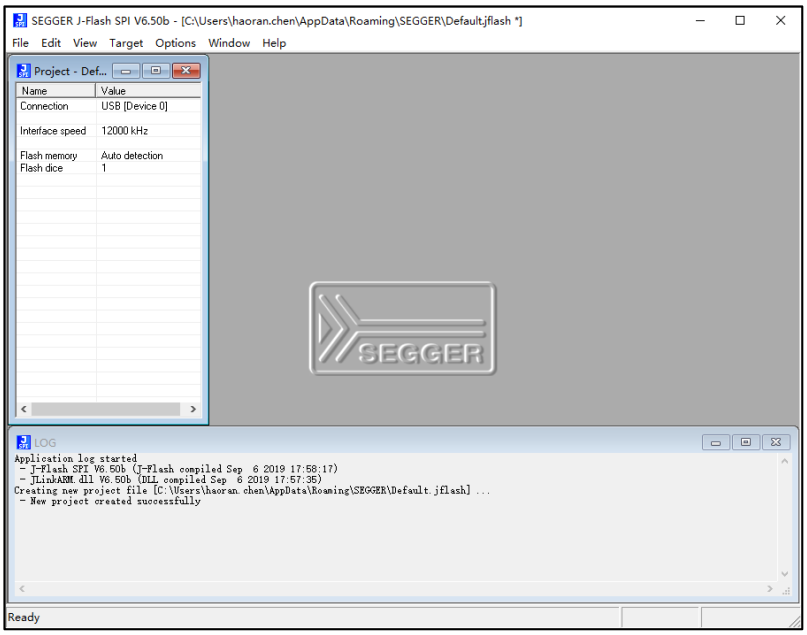
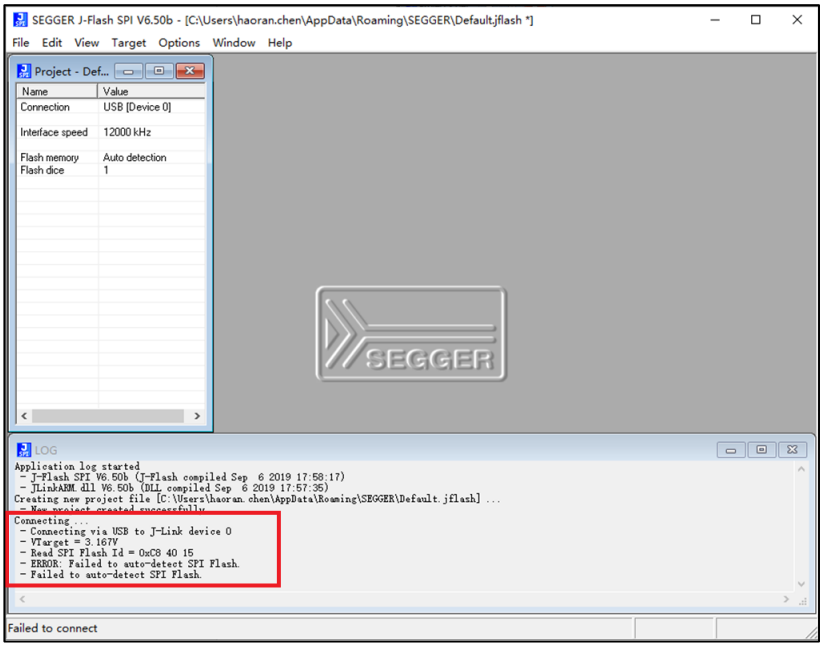


Figure 2-3. Open the J-Flash SPI software interface



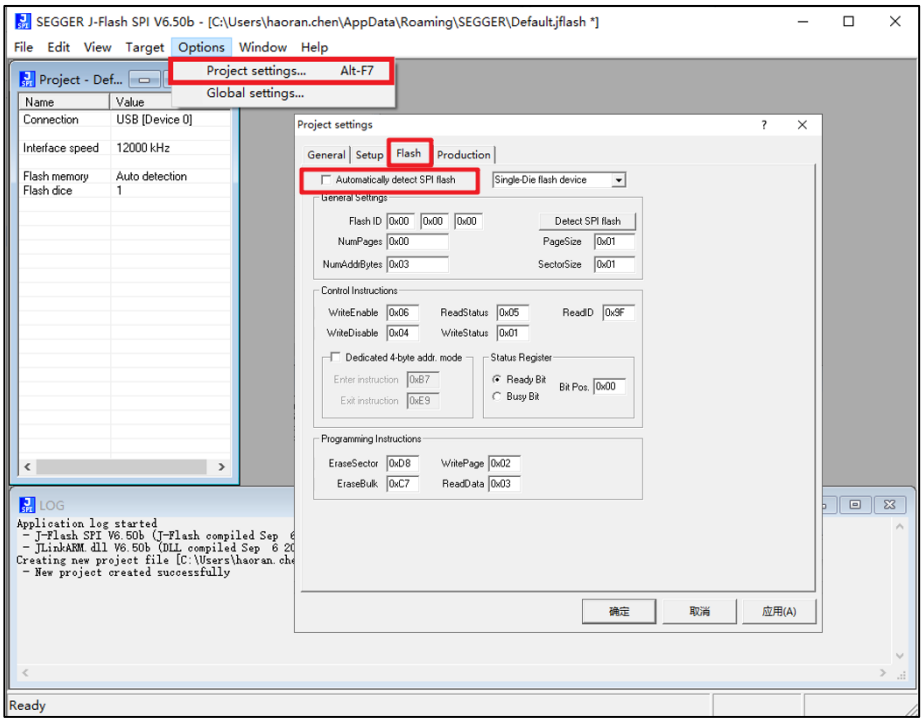
Click "Target-->Connect", the result is shown in the red block diagram in ***Figure 2-4. Connect target SPI Flash***. At this time, the SPI Flash Id has been successfully read, but the connection failed. Next, the Flash related parameters will be configured.

Figure 2-4. Connect target SPI Flash



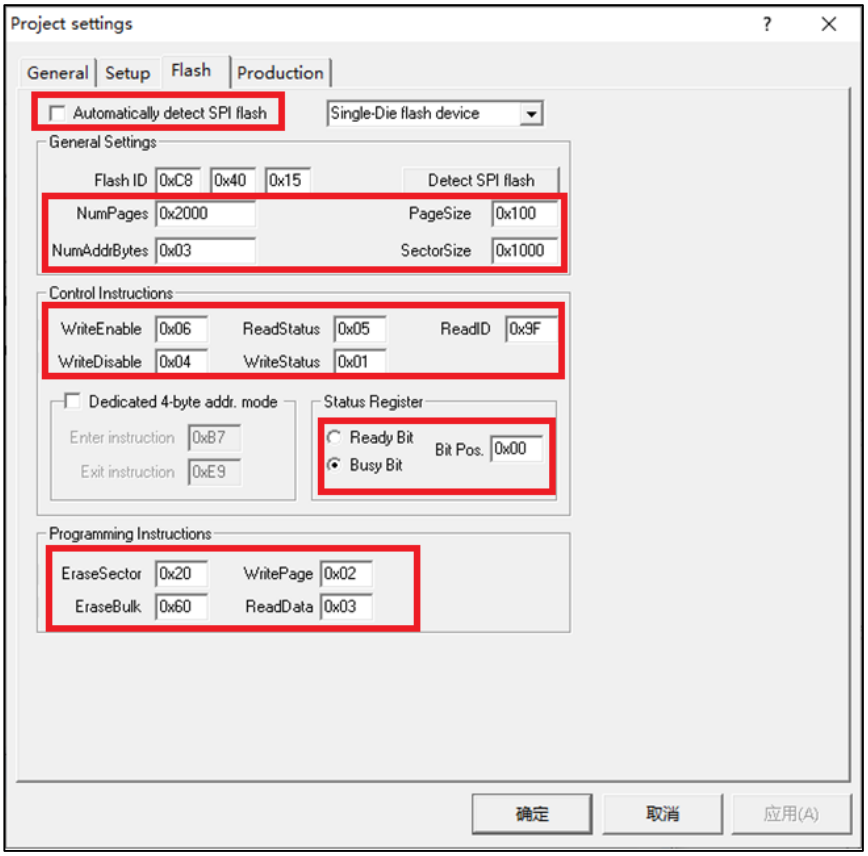
Click "Options-->Project settings", select FLASH, and uncheck automatically detect SPI flash. The reference interface is shown in ***Figure 2-5. SPI Flash configuration interface***.

Figure 2-5. SPI Flash configuration interface



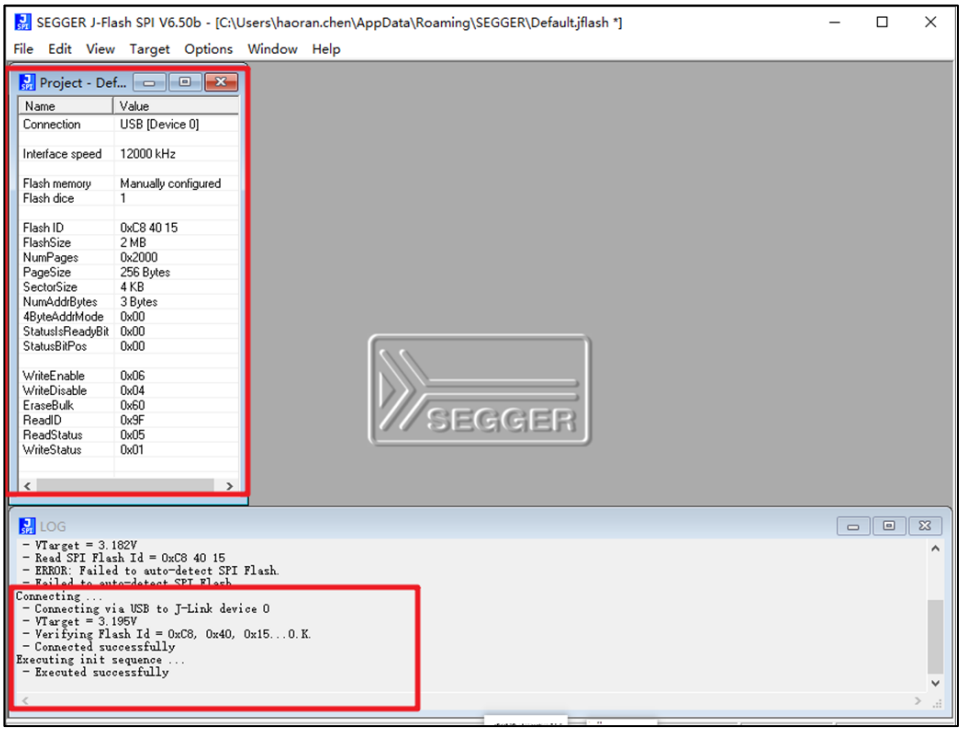
Refer to the GD25Q16B datasheet and fill in the relevant parameters such as the Flash page size, block size, read and write commands, and the specific configuration is shown in ***Figure 2-6. GD25Q16B parameter configuration***. After the configuration is complete, click ”OK”.

Figure 2-6. GD25Q16B parameter configuration



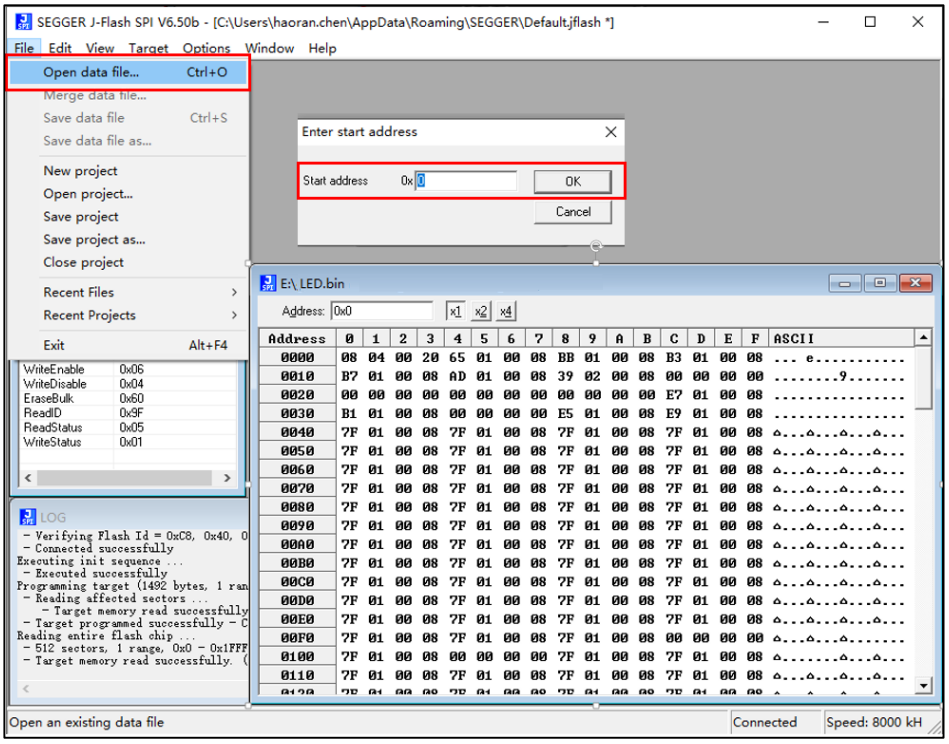
Click "Target-->Connect" again in the main interface, the result is shown in the red block diagram in ***Figure 2-7. JLink successfully connected to SPI Flash***, and the relevant parameters are displayed on the left, and it prompts that JLink and Flash are successfully connected.

Figure 2-7. JLink successfully connected to SPI Flash



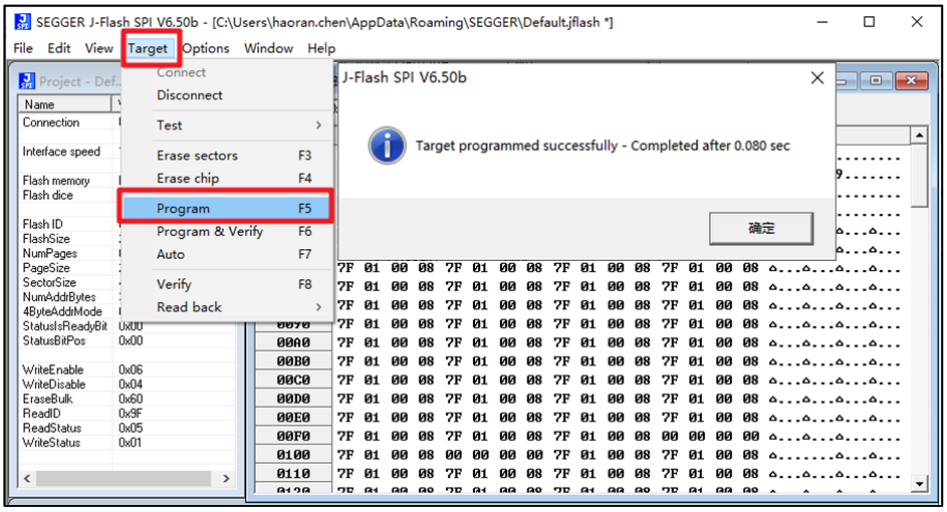
Click "File—>open data file" to open the binary file to be downloaded, as shown in ***Figure 2-8. Open the downloaded binary file***.

Figure 2-8. Open the downloaded binary file



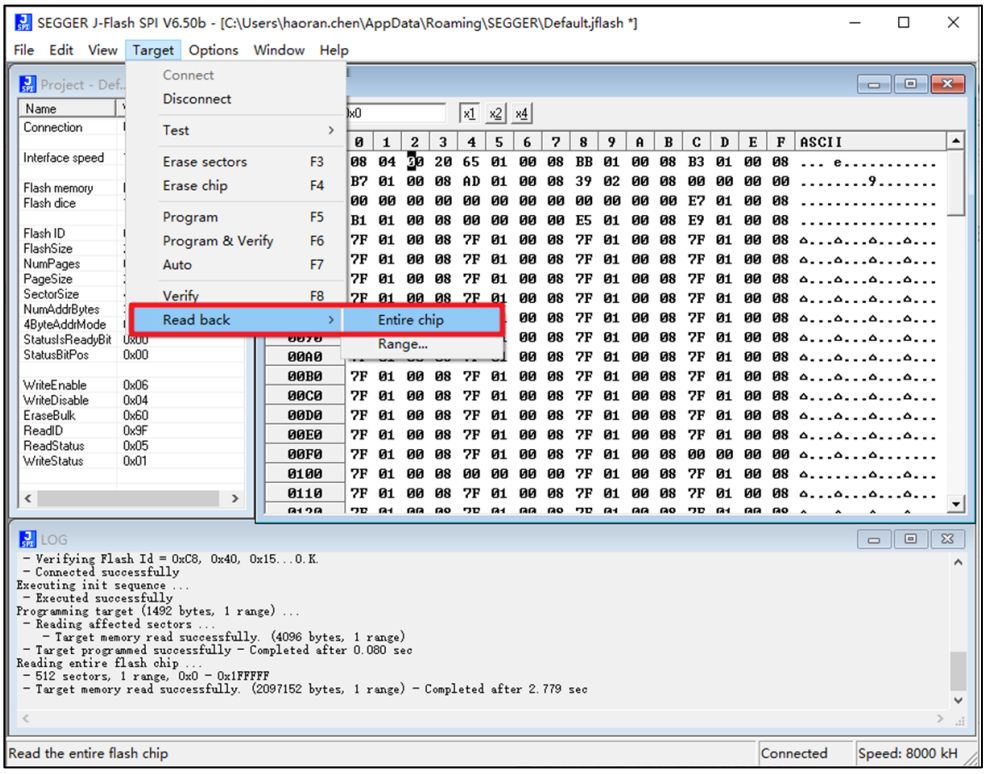
Click "Target-->Program", as shown in ***Figure 2-9. File download to Flash successful prompt***, after the download is complete, it will prompt Target programmed successfully.

Figure 2-9. File download to Flash successful prompt



In order to verify whether the binary file is successfully downloaded to the Flash, through the "Target-->Read back-->Entire chip" operation, read the value of the address and compare it with the source file, as shown in ***Figure 2-10. Read data in Flash***.

Figure 2-10. Read data in Flash

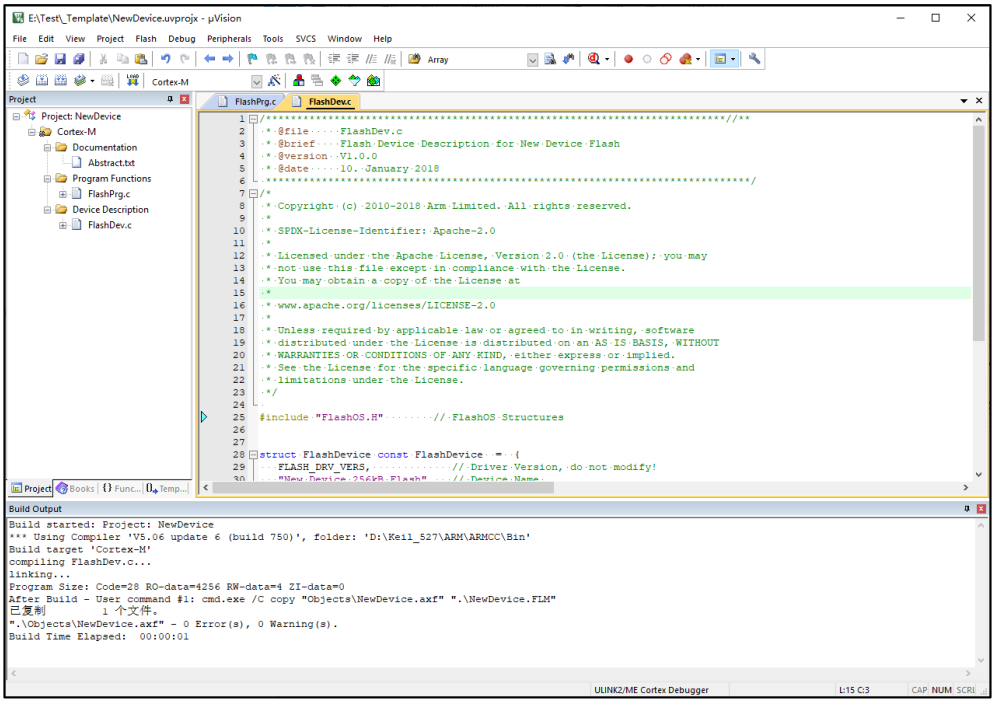


3. Use KEIL to download files to SPI Nor Flash

3.1. New FLM project

Enter the drive letter where KEIL is installed, copy the Keil\ARM\Flash_Template project to the Test folder of Disk E (the folder location can be modified as needed), double-click to open the "NewDevice.uvprojx" project, compile the project, the project will report an error "FlashDev.c(25): error: #5: cannot open source input file "..\FlashOS.H": No such file or directory", enter againFind the "FlashOS.h" file in the Keil\ARM\Flash directory, copy it to the "E:\Test_Template" directory, and change the #include "../FlashOS.H" in FlashDev.c and FlashPrg.c to #include "FlashOS.H", compile the project again, there is no error in the project, and generate NewDevicec.FLM. Related projects and compilation are shown in ***Figure 3-1. New FLM project.***

Figure 3-1. New FLM project



3.2. Porting SPI Flash driver code

Open the FlashPrg.c file, which mainly contains seven function interfaces, as shown in ***Table 3-1. FlashPrg.c function interface***.

Table 3-1. FlashPrg.c function interface

/* 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);
```

It mainly implements Init, EraseChip, EraseSector, ProgramPage and Verify function interfaces. The function interface implementation is shown in ***Table 3-2. Implementation of FlashPrg.c Function Interface***.

Table 3-2. Implementation of FlashPrg.c Function Interface

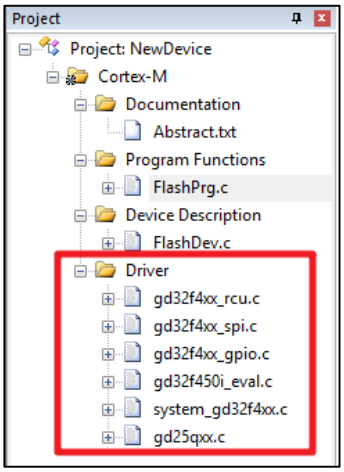
```
uint32_t base_adr;
/*
 *   Initialize Flash Programming Functions
 *      Parameter:          adr:   Device Base Address
 *                          clk:   Clock Frequency (Hz)
 *                          fnc:   Function Code (1 - Erase, 2 - Program, 3 - Verify)
 *      Return Value:     0 - OK,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

The related SPI driver is added to the KEIL project according to GD32F4xx_Firmware_Library and GD25qxx.c, and the added files are shown in ***Figure 3-2. Porting SPI driver and GD25qxx file***.

Figure 3-2. Porting SPI driver and GD25qxx file



3.3. Modify FlashDevice structure

Open the FlashDev.c file and modify the relevant content in the FlashDevice structure. The modified code is shown in ***Table 3-3. FlashDevice structure realization***.

Table 3-3. FlashDevice structure realization

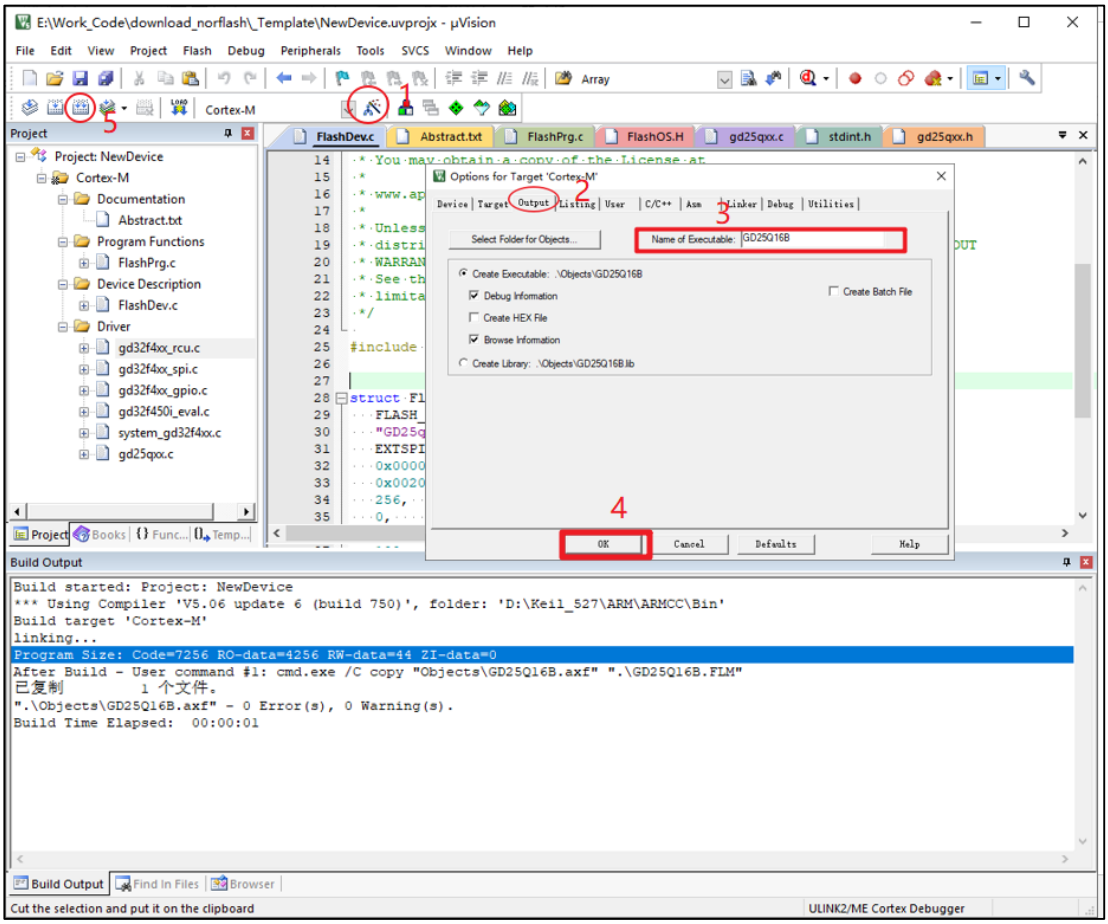
```
struct FlashDevice const FlashDevice  =  {
    FLASH_DRV_VERS,          /* Driver Version, do not modify!
    "GD25qxx",               /* Device Name */
    EXTSPI,                  /* Device Type */
    0x00000000,              /* Device Start Address */
    0x00200000,              /* Device Size 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. Compile and generate FLM file

Open the magic wand, enter the Output page, modify the Name of Executable to GD25Q16B, compile the project, and generate the GD25Q16B.FLM file. As shown in ***Figure 3-3. Compile and generate GD25Q16B.FLM file***.

Figure 3-3. Compile and generate GD25Q16B.FLM file



3.5. Add algorithm file to KEIL project

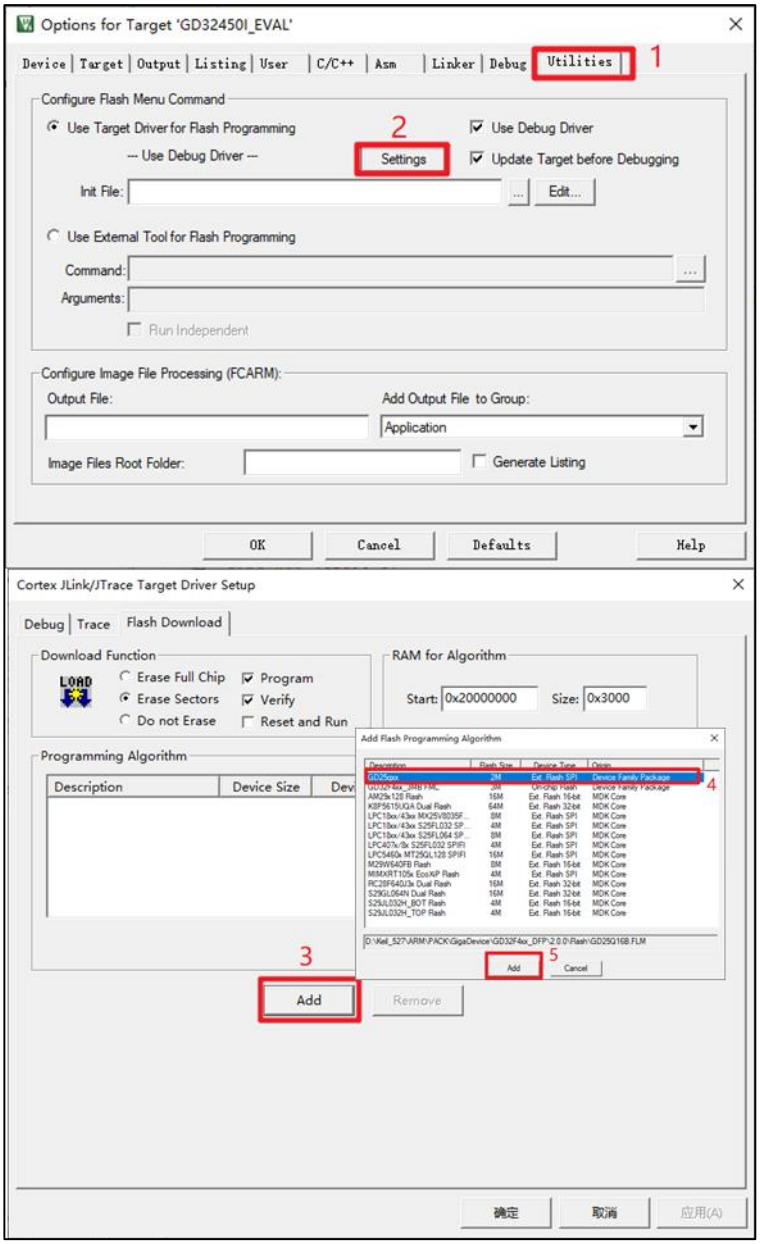
Copy the compiled GD25Q16B.FLM to the KEIL installation directory, D:\Keil_527\ARM\PACK\GigaDevice\GD32F4xx_DFP\2.0.0\Flash, and then return to the upper-level directory to open the GigaDevice.GD32F4xx_DFP.pdsc file and modify its attributes To read and write, find GD32F450IK, add the code as shown in red in ***Table 3-4. Modify the pdsc file code***.

Table 3-4. Modify the pdsc file code

```
<!-- ****************** 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>
```

In the KEIL project, open the magic wand Utilities page setting, and add the GD25qxx algorithm, as shown in ***Figure 3-4. Add GD25Qxx download algorithm to KEIL.***

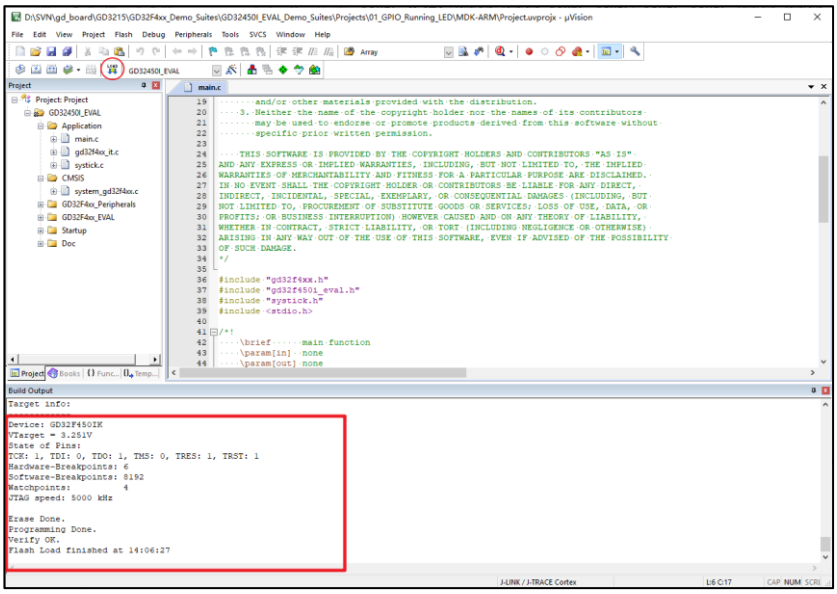
Figure 3-4. Add GD25Qxx download algorithm to KEIL



3.6. Compile and download

Compile the project in KEIL, generate the .axf file, click the Download button to download the file, as shown in ***Figure 3-5. Compile and download files in KEIL to SPI Flash***, indicating that the download is successful.

Figure 3-5. Compile and download files in KEIL to SPI Flash



3.7. Testing and verification

In order to determine whether the file is successfully downloaded to the GD25Q16BS Flash, refer to ***Jflash-SPI host computer configuration and download***, read the data in Flash through the J-Flash SPI host computer, compare whether the downloaded file and the read file are the same, and perform a verification test.

4. Revision history

Table 4-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Apr.30, 2021

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.