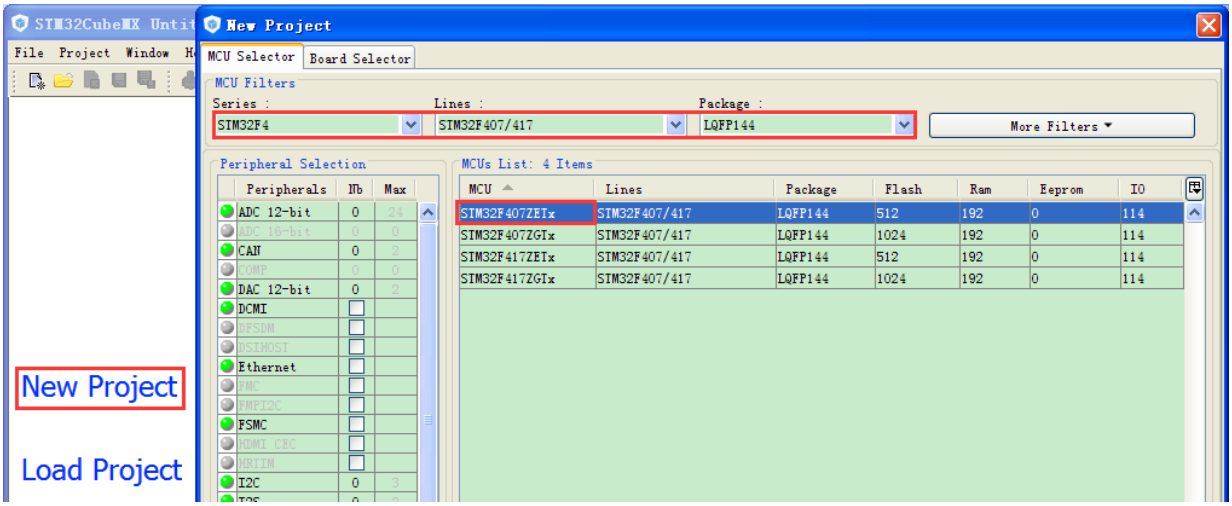
STM32Cube 学习之十三：FLASH 读写

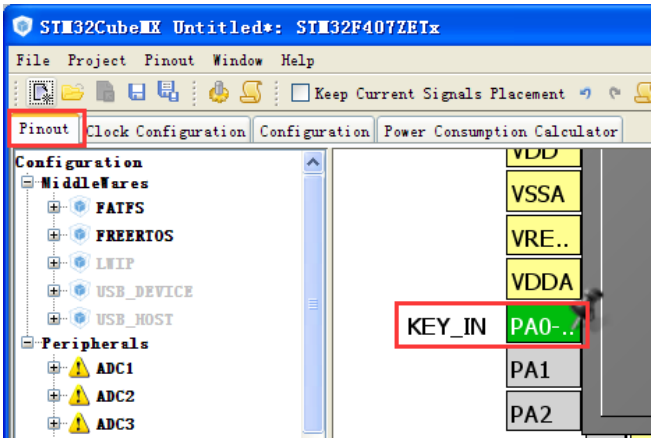
假设已经安装好 STM32CubeMX 和 STM32CubeF4 支持包。

Step1.打开 STM32CubeMX，点击“New Project”，选择芯片型号，STM32F407ZETx。

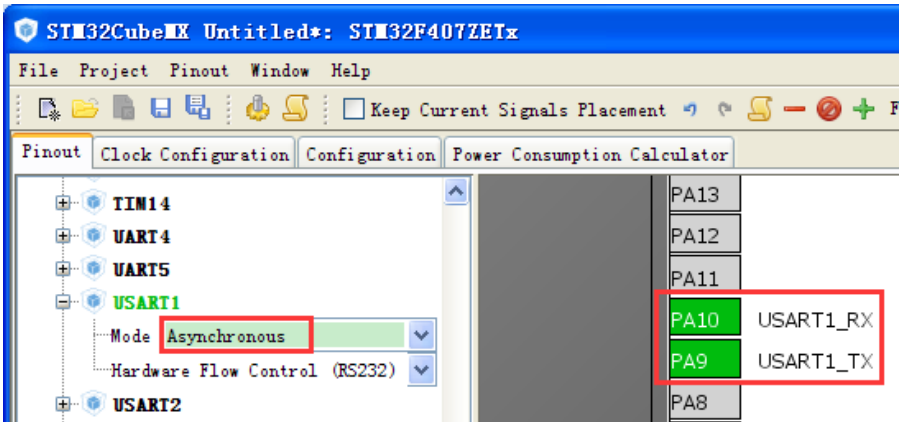


Step2. 在 Pinout 界面下配置引脚功能。

根据电路使用 PA0 作为按键输入，将 PA0 的功能配置为 GPIO_Input，并把用户标签改为 KEY_IN。

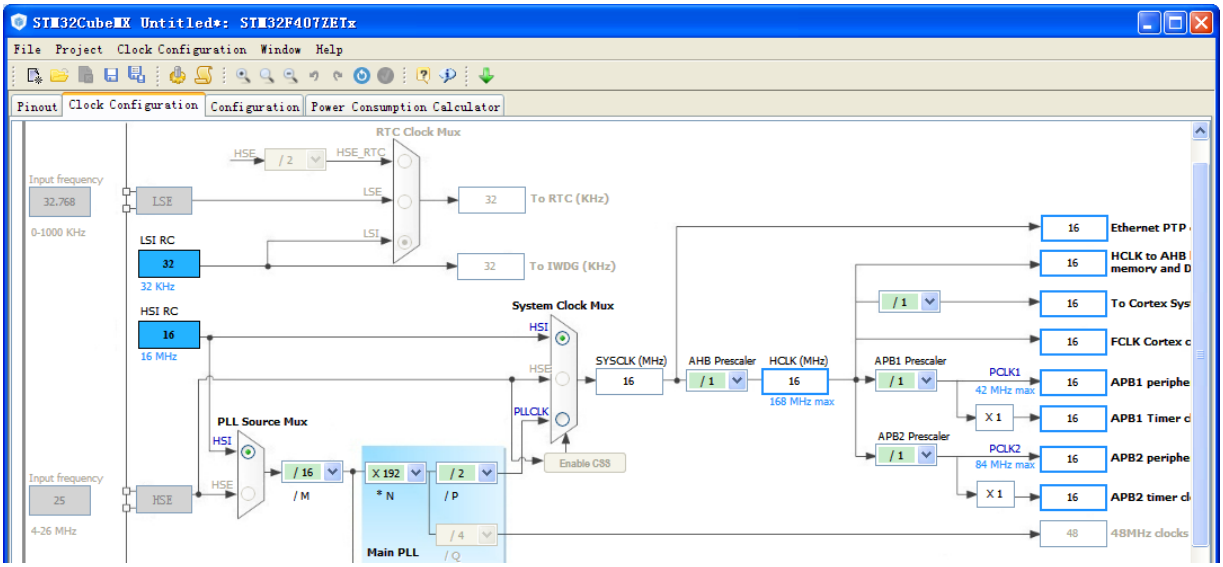


配置串口，作为信息输出接口。



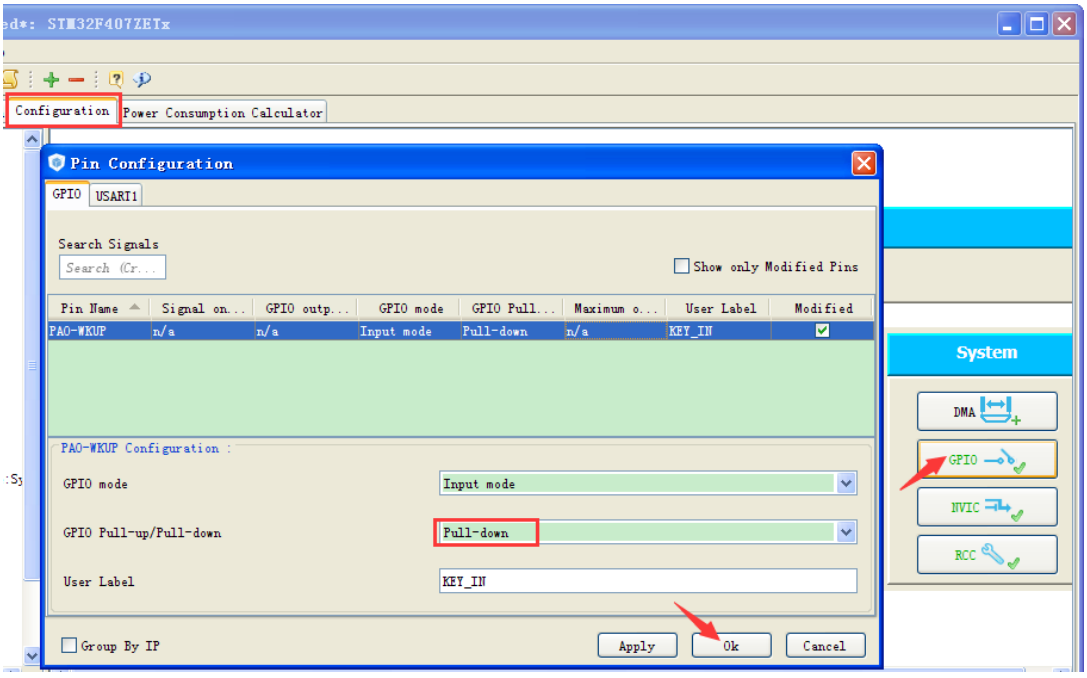
Step3.在 Clock Configuration 界面配置时钟源。

配置时钟树，在此使用默认值， 内部 16MHz。

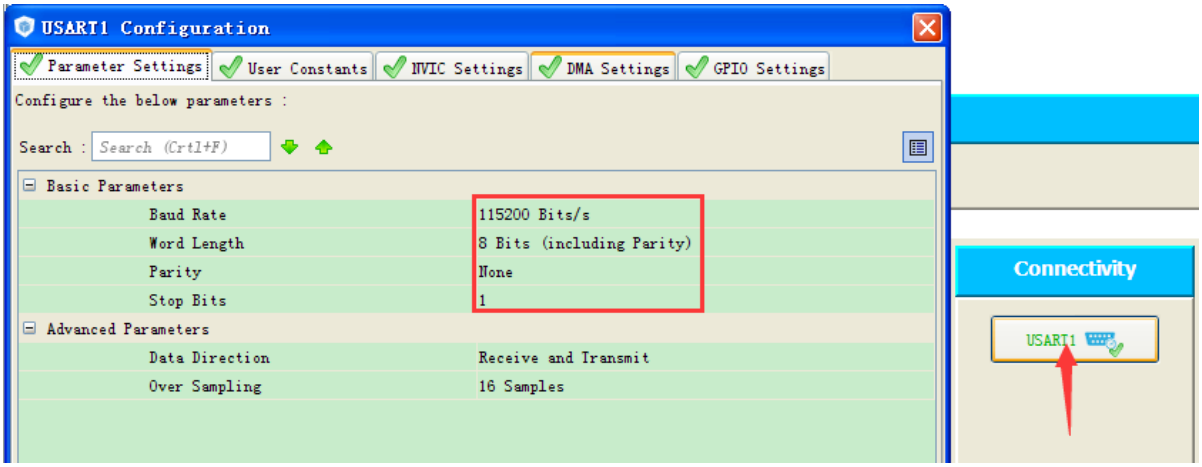


Step4.配置外设参数。

GPIO：根据电路连接，将 KEY_IN 脚设置下拉。

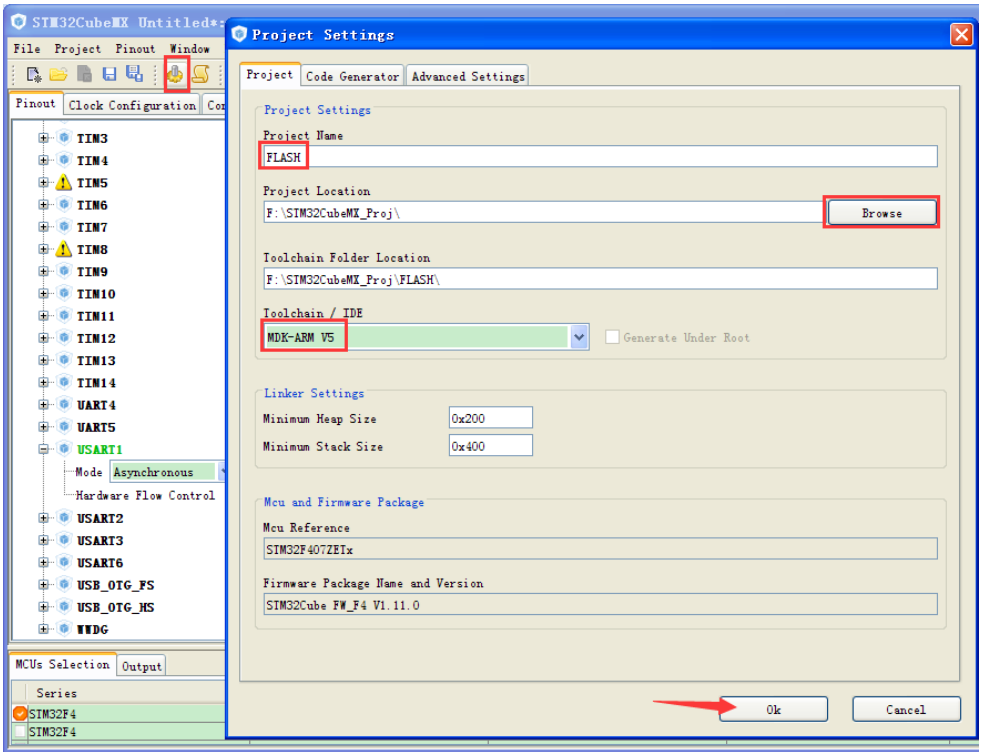


USART：串口参数默认即可。

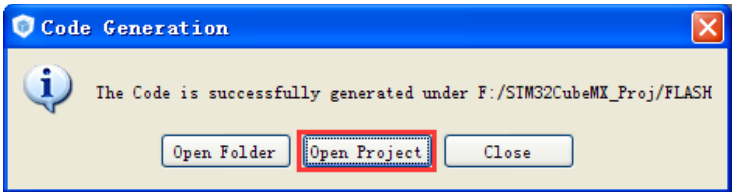


Step5.生成源代码。

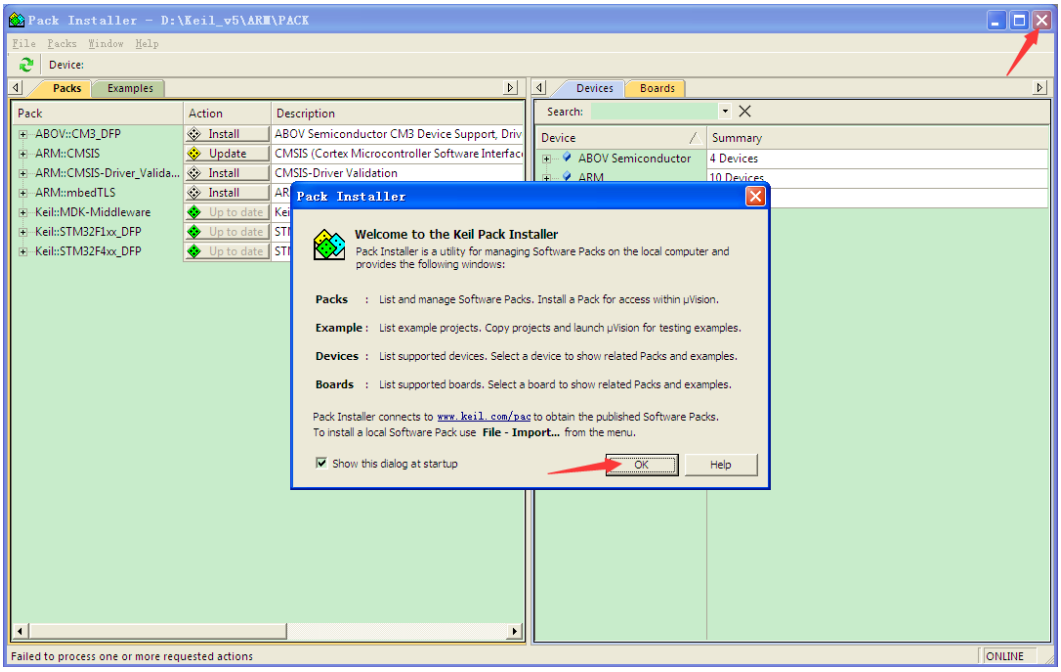
点击生成源代码按钮。在设置界面中输入工程名，保存路径，工程 IDE 类型，点 OK 即可。



生成代码完成后可直接打开工程。

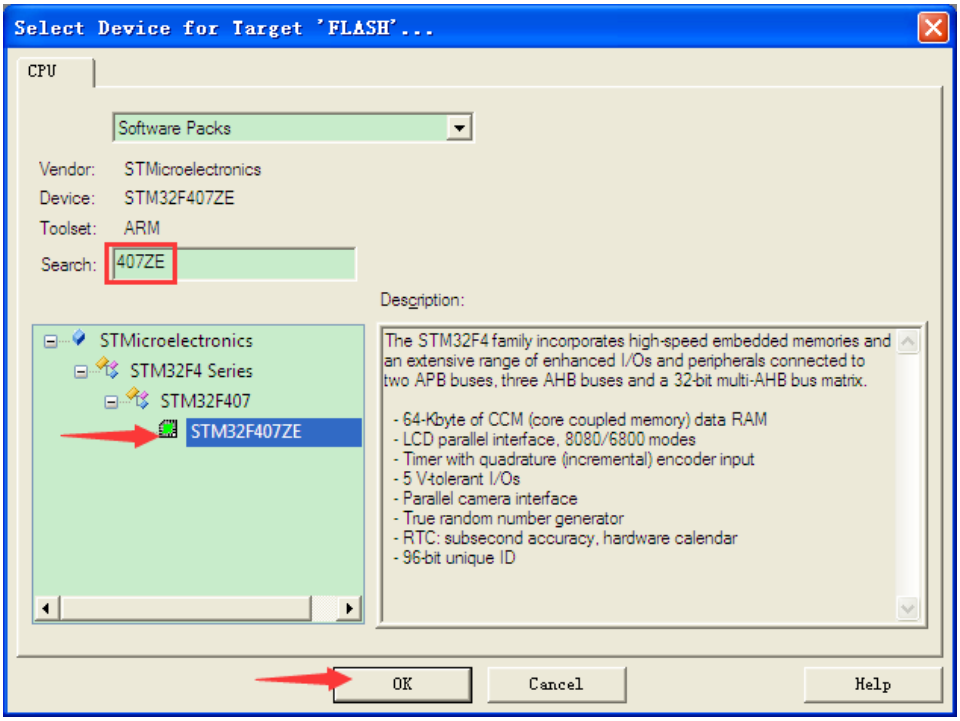


弹出如下对话框时，如果已经安装了 F4 的支持包，则点击 OK 关闭。如果没有安装，则点击界面中的 www.keil.com/...链接，找到芯片的支持包，然后安装。



关闭后面的界面。

点击“是”，然后选择芯片型号。可以在搜索框中输入关键字，加快选择速度。



Step6.添加功能代码。

先在 main.c 文件用户代码区输入包含标准输入输出头文件。

```
33  /* Includes ------------------------------------------------------------------*/
34  #include "stm32f4xx_hal.h"
35
36  /* USER CODE BEGIN Includes */
37  #include <stdio.h>
38  /* USER CODE END Includes */
39
```

在用户代码区 4 实现标准输出 printf()的底层驱动函数 fputc()，功能是在 UART1 输出一个字符。

```
171 /* USER CODE BEGIN 4 */
172 int fputc(int ch, FILE *f)
173 {
174     HAL_UART_Transmit(&huart1, (uint8_t*)&ch, 1, 10);
175     return ch;
176 }
177 /* USER CODE END 4 */
```

在/* USER CODE BEGIN PV */和/* USER CODE END PV */用户代码，定义一个 const 数组 data_flash[]。注意，在此将数组位置指定为 FLASH 扇区 2，大小为 16k 字节，即该数组占据整个扇区。数组前面的宏定义，参考的是 FLASH_EraseProgram 例程。

```
/* USER CODE BEGIN PV */
#define ADDR_FLASH_SECTOR_0     ((uint32_t)0x08000000) /* Base @ of Sector 0, 16 Kbytes */
#define ADDR_FLASH_SECTOR_1     ((uint32_t)0x08004000) /* Base @ of Sector 1, 16 Kbytes */
#define ADDR_FLASH_SECTOR_2     ((uint32_t)0x08008000) /* Base @ of Sector 2, 16 Kbytes */
#define ADDR_FLASH_SECTOR_3     ((uint32_t)0x0800C000) /* Base @ of Sector 3, 16 Kbytes */
#define ADDR_FLASH_SECTOR_4     ((uint32_t)0x08010000) /* Base @ of Sector 4, 64 Kbytes */
#define ADDR_FLASH_SECTOR_5     ((uint32_t)0x08020000) /* Base @ of Sector 5, 128 Kbytes */
#define ADDR_FLASH_SECTOR_6     ((uint32_t)0x08040000) /* Base @ of Sector 6, 128 Kbytes */
#define ADDR_FLASH_SECTOR_7     ((uint32_t)0x08060000) /* Base @ of Sector 7, 128 Kbytes */
#define ADDR_FLASH_SECTOR_8     ((uint32_t)0x08080000) /* Base @ of Sector 8, 128 Kbytes */
#define ADDR_FLASH_SECTOR_9     ((uint32_t)0x080A0000) /* Base @ of Sector 9, 128 Kbytes */
#define ADDR_FLASH_SECTOR_10    ((uint32_t)0x080C0000) /* Base @ of Sector 10, 128 Kbytes */
#define ADDR_FLASH_SECTOR_11    ((uint32_t)0x080E0000) /* Base @ of Sector 11, 128 Kbytes */

#define FLASH_SECTOR0_SIZE     ((uint32_t)0x04000) /* 16 Kbytes */
#define FLASH_SECTOR1_SIZE     ((uint32_t)0x04000) /* 16 Kbytes */
#define FLASH_SECTOR2_SIZE     ((uint32_t)0x04000) /* 16 Kbytes */
#define FLASH_SECTOR3_SIZE     ((uint32_t)0x04000) /* 16 Kbytes */
#define FLASH_SECTOR4_SIZE     ((uint32_t)0x10000) /* 64 Kbytes */
#define FLASH_SECTOR5_SIZE     ((uint32_t)0x20000) /* 128 Kbytes */
#define FLASH_SECTOR6_SIZE     ((uint32_t)0x20000) /* 128 Kbytes */
#define FLASH_SECTOR7_SIZE     ((uint32_t)0x20000) /* 128 Kbytes */
#define FLASH_SECTOR8_SIZE     ((uint32_t)0x20000) /* 128 Kbytes */
#define FLASH_SECTOR9_SIZE     ((uint32_t)0x20000) /* 128 Kbytes */
#define FLASH_SECTOR10_SIZE    ((uint32_t)0x20000) /* 128 Kbytes */
#define FLASH_SECTOR11_SIZE    ((uint32_t)0x20000) /* 128 Kbytes */
/* Private variables ---------------------------------------------------------*/
const uint8_t data_flash[FLASH_SECTOR2_SIZE] __attribute__((at(ADDR_FLASH_SECTOR_2))) = {
    0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0xAA,0xBB,0xCC,0xDD,0xEE,0xFF,
    0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0xAA,0xBB,0xCC,0xDD,0xEE,0xFF,
};
/* USER CODE END PV */
```

在/* USER CODE BEGIN PFP */和/* USER CODE END PFP */用户代码添加两个用户函数。参考 FLASH_EraseProgram 例程。

```
/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------
/*
功能: 向FLASH扇区2填充指定数据;
输入: dat,要填充的数据,按32bit;
返回: 操作成功返回HAL_OK,失败返回HAL_ERROR;
*/
HAL_StatusTypeDef flash_write_sector2(uint32_t dat)
{
    FLASH_EraseInitTypeDef EraseInitStruct;
    uint32_t FirstSector = 0, NbOfSectors = 0, Address = 0;
    uint32_t SectorError = 0;

    HAL_FLASH_Unlock();

    FirstSector = FLASH_SECTOR_2;
    NbOfSectors = 1;

    EraseInitStruct.TypeErase = FLASH_TYPEERASE_SECTORS;
    EraseInitStruct.VoltageRange = FLASH_VOLTAGE_RANGE_3;
    EraseInitStruct.Sector = FirstSector;
    EraseInitStruct.NbSectors = NbOfSectors;
    if(HAL_FLASHEx_Erase(&EraseInitStruct, &SectorError) != HAL_OK) {
        HAL_FLASH_Lock();
        return HAL_ERROR;
    }

    __HAL_FLASH_DATA_CACHE_DISABLE();
    __HAL_FLASH_INSTRUCTION_CACHE_DISABLE();

    __HAL_FLASH_DATA_CACHE_RESET();
    __HAL_FLASH_INSTRUCTION_CACHE_RESET();

    __HAL_FLASH_INSTRUCTION_CACHE_ENABLE();
    __HAL_FLASH_DATA_CACHE_ENABLE();

    Address = ADDR_FLASH_SECTOR_2;
    while (Address < ADDR_FLASH_SECTOR_3) {
        if (HAL_FLASH_Program(FLASH_TYPEPROGRAM_WORD, Address, dat) == HAL_OK) {
            Address = Address + 4;
        } else {
            HAL_FLASH_Lock();
            return HAL_ERROR;
        }
    }

    HAL_FLASH_Lock();
    return HAL_OK;
}
```

```
/*
功能: 读取FLASH扇区2数据,并通过UART打印输出.
*/
void flash_read_sector2(void)
{
    uint32_t Address = 0;
    uint32_t data32 = 0;

    Address = ADDR_FLASH_SECTOR_2;

    while (Address < ADDR_FLASH_SECTOR_3) {
        if (0 == (Address%32)) {
            printf("\r\n"); // 每32字节换一行
        }
        data32 = *(__IO uint32_t*)Address;
        printf("0x%08X, ",data32);
        Address = Address + 4;
    }
    printf("\r\n\r\n");
}
/* USER CODE END PFP */
```

在 while(1)中的用户代码区 3，添加按键识别和 FLASH 操作代码。

```
  while (1)
  {
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
      if (GPIO_PIN_SET == HAL_GPIO_ReadPin(KEY_IN_GPIO_Port, KEY_IN_Pin)) {
        HAL_Delay(20);
        if (GPIO_PIN_SET == HAL_GPIO_ReadPin(KEY_IN_GPIO_Port, KEY_IN_Pin)) {
            flash_read_sector2();               // 读取FLASH扇区2的数据
            flash_write_sector2(0x12345678);    // 用0x12345678填充FLASH扇区2
            flash_read_sector2();               // 再读取FLASH扇区2的数据
            flash_write_sector2(0x89ABCDEF);    // 用0x89ABCDEF填充FLASH扇区2
            flash_read_sector2();               // 再读取FLASH扇区2的数据
        }
      }
  }
  /* USER CODE END 3 */
```

至此，工程完成。

功能是，按一次按键，将依次执行：

读取 FLASH 扇区 2 的数据发送到串口；

用 0x12345678 填充扇区 2；

再读取扇区 2 数据发送到串口；

用 0x89ABCDEF 填充扇区 2；

再读取扇区 2 数据发送到串口。

官方例程请参考 stm32cubef4.zip 解压后 STM32Cube_FW_F4_V1.9.0\Projects\STM324xG_EVAL\Examples\FLASH\FLASH_EraseProgram 目录下的工程。

S.D.Lu 于 深圳

2016 年 10 月