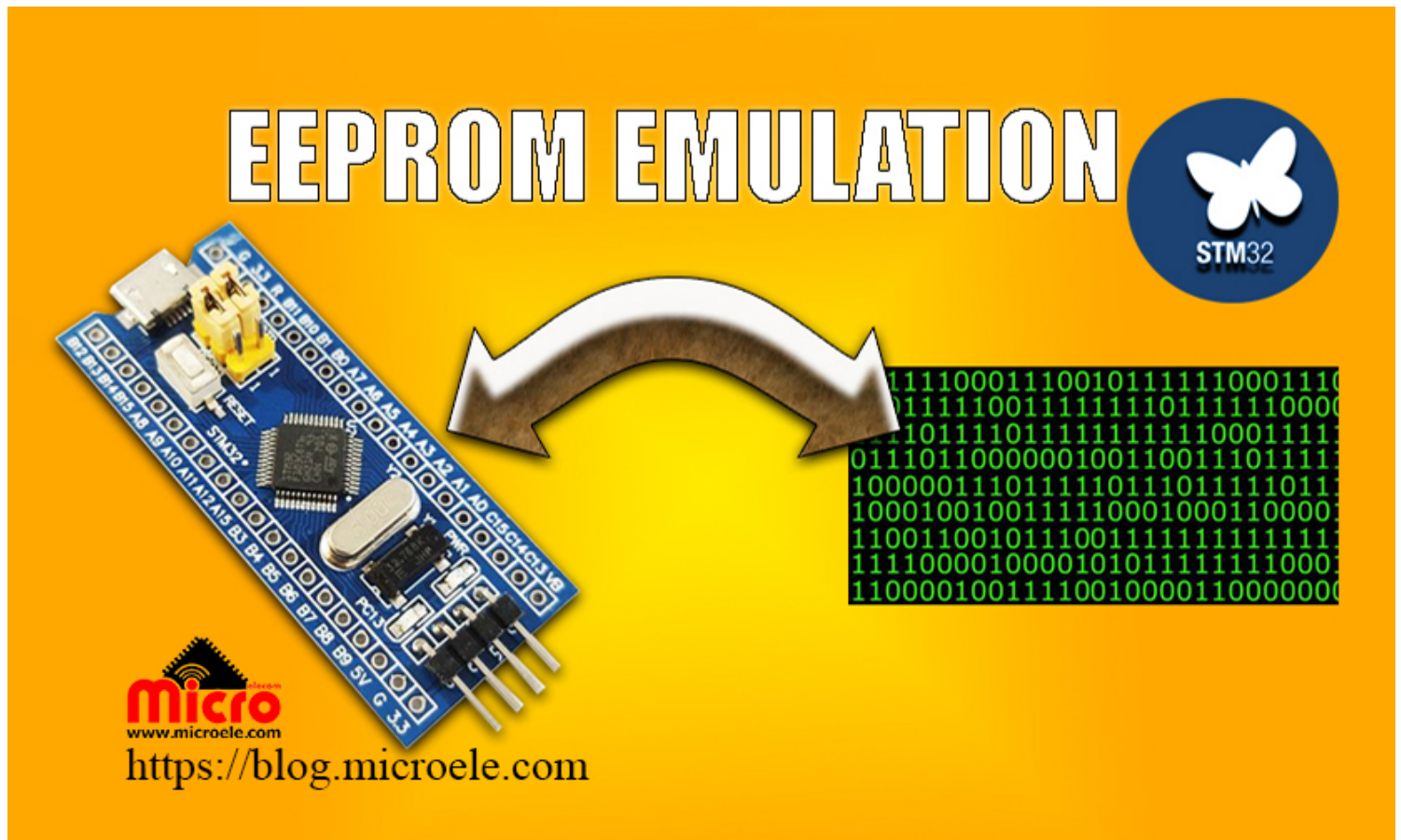




تبدیل حافظه FLASH به EEPROM داخلی در میکروکنترلر های STM32



تاریخ انتشار ۱۶ فروردین، ۱۴۰۰ توسط آرش فتاحی

در این مطلب قصد داریم نحوه تبدیل بخشی از حافظه FLASH در میکروکنترلرهای STM32 سری های F0, F1 و F3 را به حافظه EEPROM بررسی کنیم. مهم ترین کاربرد EEPROM ها، در ذخیره و نگهداری دیتایی است که می‌خواهیم همواره وجود داشته و با قطع، وصل یا خاموش شدن سیستم از بین نرود.



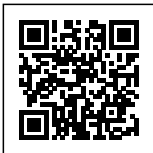
معرفی حافظه ها و کاربرد EEPROM

معمولا در میکروکنترلرها از سه نوع حافظه FLASH برای نگهداری برنامه، EEPROM برای ذخیره اطلاعات ماندگار و SRAM برای ذخیره اطلاعات میانی در یک برنامه استفاده می شود. یکی از حافظه های پرکاربرد در میکروکنترلرها، حافظه های EEPROM هستند که غیر فرار بوده و می توان به دفعات بر روی آنها اطلاعات مورد نیاز را ذخیره و در صورت نیاز حتی پس از قطع منبع تغذیه مجدداً آن اطلاعات را بازیابی نمود. این نوع از حافظه ها به صورت پیش فرض در برخی میکروکنترلرهای 8 بیتی همچون AVR وجود دارند. همچنین می توان از EEPROM های خارجی مانند سری آییسی های AT24Cx نیز نام برد.

EEPROM یا Electrically Erasable Programmable Read-Only Memory (حافظه فقط خواندنی قابل برنامه ریزی و پاکسازی به صورت الکتریکی)، قابلیت برنامه ریزی و پاک کردن اطلاعات را به صورت الکتریکی بر روی خود فراهم می کنند. اطلاعاتی که بر روی EEPROM نوشته می شود، با قطع منبع تغذیه از میکرو پاک نشده و در حافظه به صورت پایدار باقی خواهد ماند و با هر بار روشن شدن مدار، امکان بازیابی این اطلاعات از روی حافظه EEPROM امکان پذیر می باشد.



آیسی EEPROM خارجی AT24C32D



نمونه ای از کاربرد EEPROM در میکروکنترلرها

به عنوان یک نمونه از کاربرد حافظه EEPROM، فرض کنید می‌خواهیم سامانه‌ای جهت کنترل دمای یک اتاق طراحی کنیم. این سامانه دمای مد نظر را از کاربر دریافت کرده و در صورتی که دما از مقدار دریافت شده توسط کاربر، بالاتر یا پایین تر برود، سامانه سرمایشی یا گرمایشی جهت رساندن دمای محیط به مقدار تعیین شده توسط کاربر فعال می‌گردد. از آن جایی که می‌خواهیم در هر بار روشن شدن سامانه کنترل دما، نیازی به وارد کردن دمای مد نظر توسط کاربر نباشد، در برنامه نوشته شده، این مقدار را در حافظه EEPROM میکرو یا EEPROM های خارجی ذخیره می‌کنیم. به این ترتیب با هر بار روشن شدن و راه اندازی مدار، نیازی به تنظیم مجدد و وارد کردن اطلاعات توسط کاربر نمی‌باشد.

EEPROM در STM32

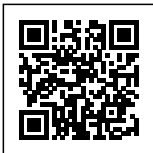
از آن جایی که میکروکنترلرهای STM32 فاقد این نوع از حافظه ها می‌باشند در نگاه اول چاره ای جز استفاده از EEPROM های خارجی به صورت مجزا دیده نمی‌شود. اما این قابلیت در میکروهای سری STM32 وجود دارد تا بتوان بخشی از حافظه FLASH در این نوع از میکروها را تبدیل به EEPROM کرده و به نوعی EEPROM را در حافظه FLASH شبیه سازی نمود.

حافظه های فلش حافظه‌هایی هستند که برنامه اصلی با پروگرامر بر روی آن‌ها ریخته می‌شود و معمولاً توسط برنامه، برای نوشتن دیتا قابل دسترس نمی‌باشد. اما در میکروهای STM32 دسترسی به این حافظه، توسط خود میکرو و برنامه نوشته شده امکان پذیر است.

کلیات آموزش

در این آموزش قصد داریم یک شمارنده باینری را بر روی میکروکنترلر STM32 پیاده سازی کنیم که با هر بار قطع و وصل شدن تغذیه از مدار، آخرین مقدار شمرده شده بر روی LED های روی برد برد، مجدد نمایش داده شود و شمارش از همان نقطه‌ای که تغذیه قطع شده بود ادامه یابد.

همچنین برای راحتی کاربران، یک کتابخانه آماده جهت شبیه سازی EEPROM در میکروکنترلرهای STM32 جهت دانلود قرار گرفته است. می‌توانید برای دریافت فایل کتابخانه از [این لینک](#) استفاده نمایید.



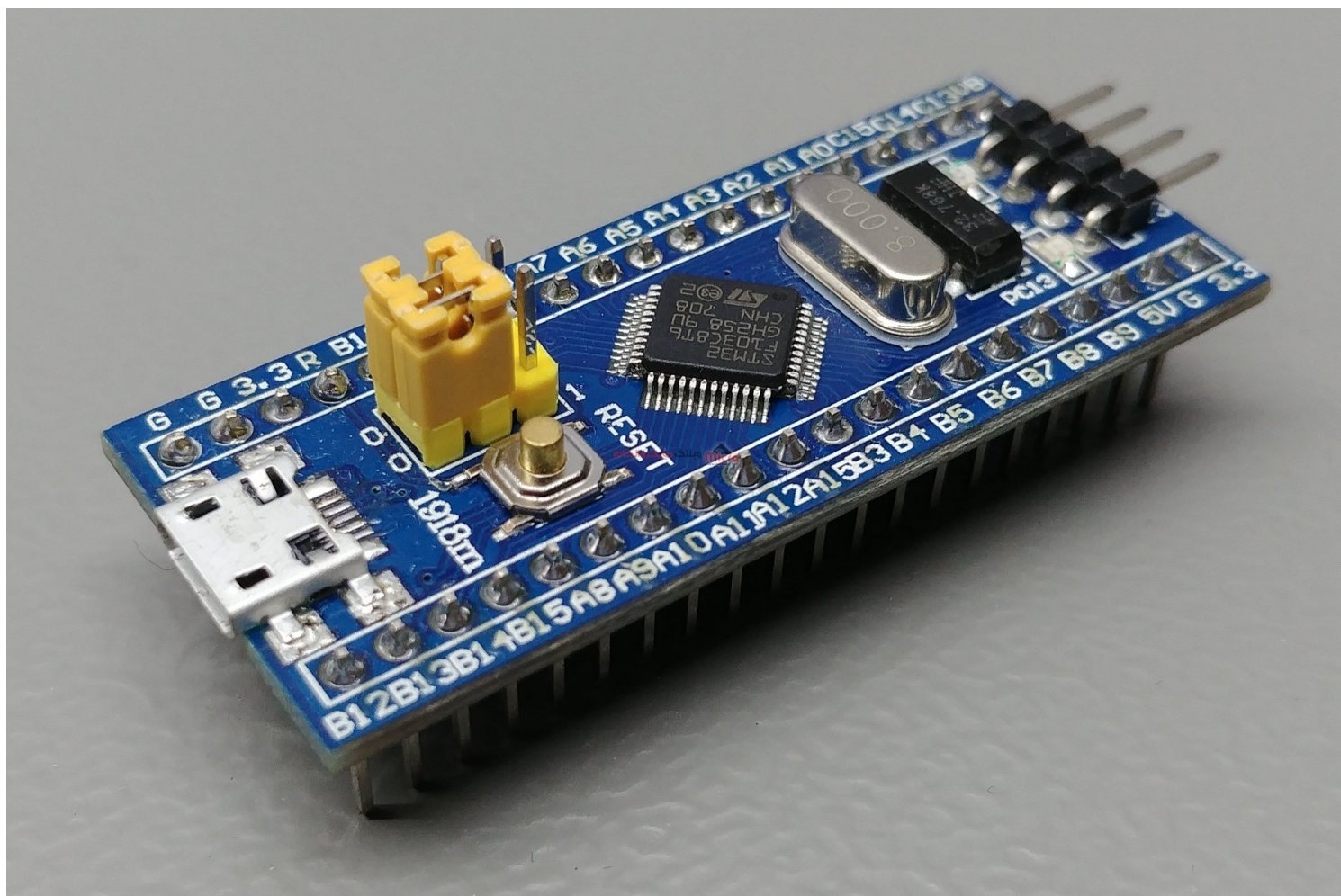
پروژه از طریق STM32CUBEMX تولید و سپس در Keil IDE کدنویسی شده و اضافه کردن کتابخانه‌ها در این محیط انجام خواهد گرفت. البته روند کار در محیط IDE های دیگر همچون IAR و CUBEIDE یکسان بوده و می توان از روش گفته شده برای IDE های استاندارد دیگر نیز استفاده نمود.

تعریف پروژه

برای اجرای این آزمایش، از یک برد Bluepill با میکروی STM32f103C8T6، یک کلید و چهار LED استفاده شده است. کاربر با هر بار فشردن کلید، مقدار شمارنده باینری را افزایش داده و عدد بر روی LED ها نمایش داده می‌شود. با قطع کردن تغذیه مدار و وصل کردن مجدد آن، آخرین مقدار شمارش شده بر روی LED ها بارگذاری شده و شمارش از آن نقطه ادامه می‌یابد.

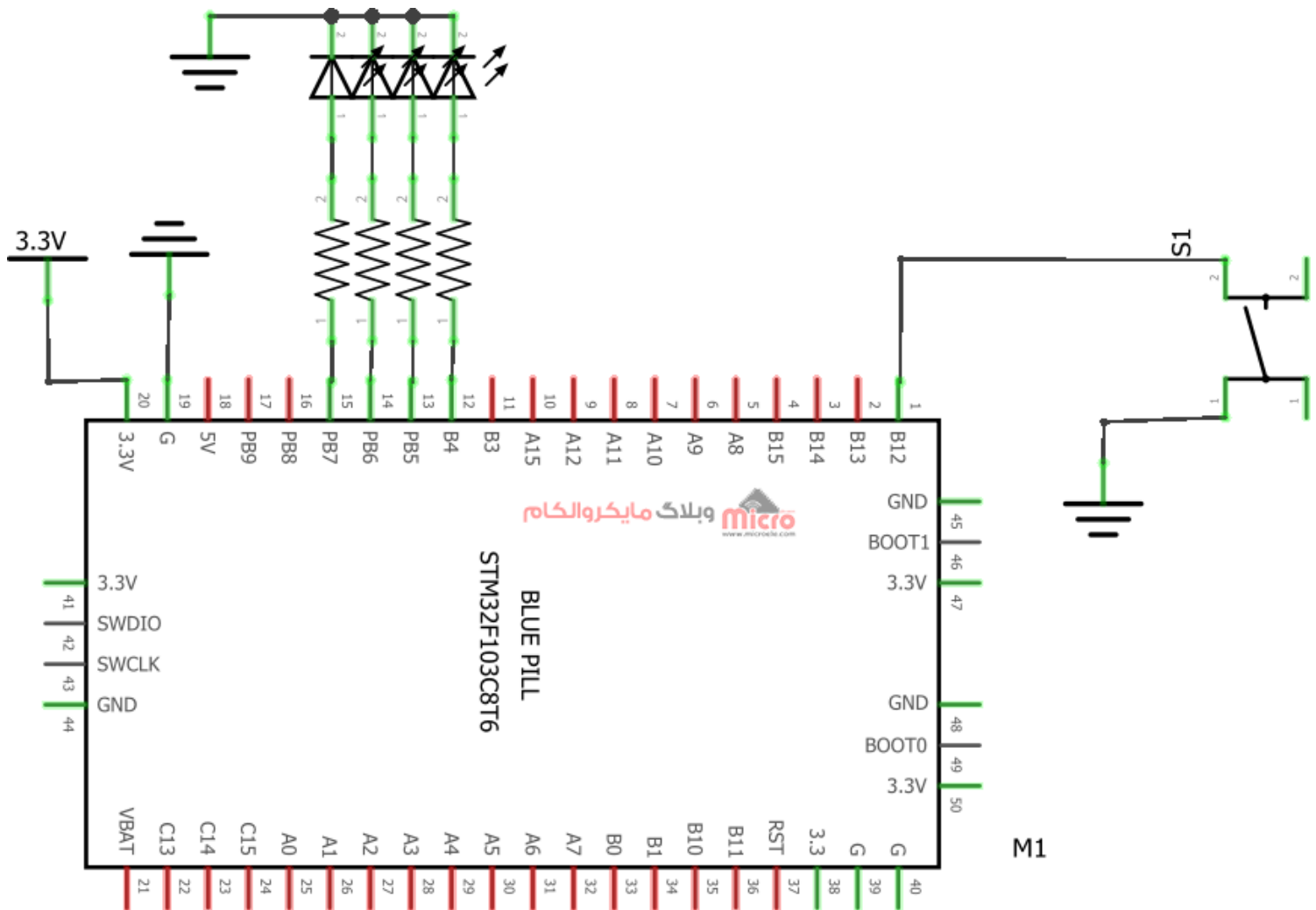
وسایل مورد نیاز

- برد Bluepill با میکروی STM32F103C8T6
- برد بورد
- LED
- سیم برد بردی
- پروگرامر St-link V2



STM32F103C8T6

شماتیک مدار



Schematic

مراحل انجام آزمایش

ابتدا نرم افزار STM32CUBEMX را باز کرده و بعد از انتخاب گزینه ACCESS TO MCU SELECTOR، میکرووی STM32f103C8T6 را انتخاب می‌کنیم (شما می‌توانید میکرو مورد نظر خود را از سری F0، F1 یا F3 انتخاب کنید).



MX STM32CubeMX Untitled



File

Window

Help



Home

Existing Projects

Recent Opened Projects

EE_EMU_F1.ioc

MX

Last modified date : 31/03/2021 1:03:49

ds1307.ioc

MX

Last modified date : 26/03/2021 13:34:53

SLSTM32.ioc

MX

Last modified date : 26/03/2021 13:32:51

Slave.ioc

MX

Last modified date : 26/03/2021 12:52:53

Other Projects



New Project

I need to :

Start My project from MCU

ACCESS TO MCU SELECTOR

Start My project from ST B...

ACCESS TO BOARD SELECTOR

Start My project from Exam...

ACCESS TO EXAMPLE SELECTOR

Manage software installations

Check for STM32CubeMX a...

CHECK FOR UPDATES

Install or remove embedded ...

INSTALL / REMOVE



About STM32

External Tools

STM32 CubeMX



MX New Project from a MCU/MPU

MCU/MPU Selector | Board Selector | Example Selector | Cross Selector

MCU/MPU Filters

Part Number: stm32f103c8

Core >

Series >

Line >

Package >

Other >

Peripheral >

Features | Block Diagram | Docs & Resources | Datasheet | Buy | Sta

☆ STM32F1 Series

STM32F103C8 Mainstream Performance line, Arm Cortex-M3 MCU with 64 Kbytes of Flash memory, 72 MHz CPU, motor control, USB and CAN

ACTIVE Active Product is in mass production

Unit Price for 10kU (US\$): 1.999

 LQFP48

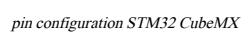
The STM32F103xx medium-density performance line family incorporates the high-performance

MCUs/MPUs List: 1 item + Display similar items Export

	Part No	Referen...	Unit Pri...	Board	Package	Flash	RAM	IO	Freq.
☆	STM32F103C8	STM32... A...	1.999		LQFP48	64 kBy...	20 kByt...	37	72 M...

STM32 CubeMX

در محیط CUBE، پایه PB12 را برای کلید، ورودی (input) و پایه های PB6، PB5، PB4 و PB7 را برای اتصال LED ها، خروجی (output) تعریف می کنیم.



blog.microele.com



STM32CubeMX GPIO Mode and Configuration

Categories: A-Z

System Core

- DMA
- GPIO**
- IWDG
- NVIC
- ✓ RCC
- ▲ SYS
- WWDG

Analog >

Timers >

Connectivity >

Computing >

Middleware >

Configuration

Group By Peripherals

✓ GPIO ✓ RCC ✓ SYS

Search Signals

Search (Ctrl+F)

☐ Show only Modified Pins

Pin	Signal	GPIO Mode	GPIO Type	GPIO Pull-up/Pull-down	Maximum Frequency	User Label	Modified
PB4	n/a	Low	Output	No pull-up	Low		☐
PB5	n/a	Low	Output	No pull-up	Low		☐
PB6	n/a	Low	Output	No pull-up	Low		☐
PB7	n/a	Low	Output	No pull-up	Low		☐
PB12	n/a	n/a	Input mode	Pull-up	n/a		☒
PC13-T...	n/a	Low	Output	No pull-up	Low		☐

PB12 Configuration :

GPIO mode: Input mode

GPIO Pull-up/Pull-down: **Pull-up**

User Label:

pin configuration STM32 CubeMX

سپس به سربرگ Project Manager رفته و در آن جا نام پروژه، محل ذخیره سازی و IDE مد نظر خود را برای توسعه برنامه انتخاب می‌کنیم.



STM32CubeMX Untitled*: STM32F103C8Tx

STM32CubeMX Home STM32F103C8Tx > **Untitled - Project Manager** > GENERATE CODE

Pinout & Configuration Clock Configuration **Project Manager** Tools

Project

Project Settings

Project Name
EE_EMU_F1

Project Location
F:\microele\EE_EMULATION\PRJ\ Browse

Application Structure
Advanced ☐ Do not generate the ma...

Toolchain Folder Location
F:\microele\EE_EMULATION\PRJ\EE_EMU_F1\

Toolchain / IDE
MDK-ARM Min Version
V5.27 ☐ Generate Under ...

Code Generator

Advanced Settings

Linker Settings

Minimum Heap Size 0x200

Minimum Stack Size 0x400

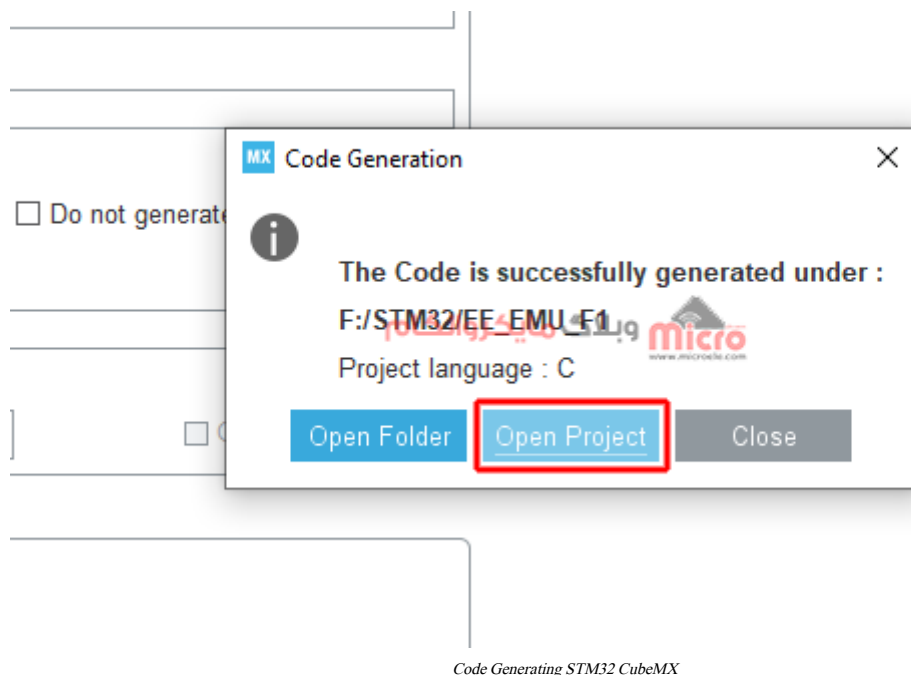
Project Manager STM32 CubeMX

بر روی GENERATE CODE کلیک کرده و منتظر مانده تا پروژه تولید شود.

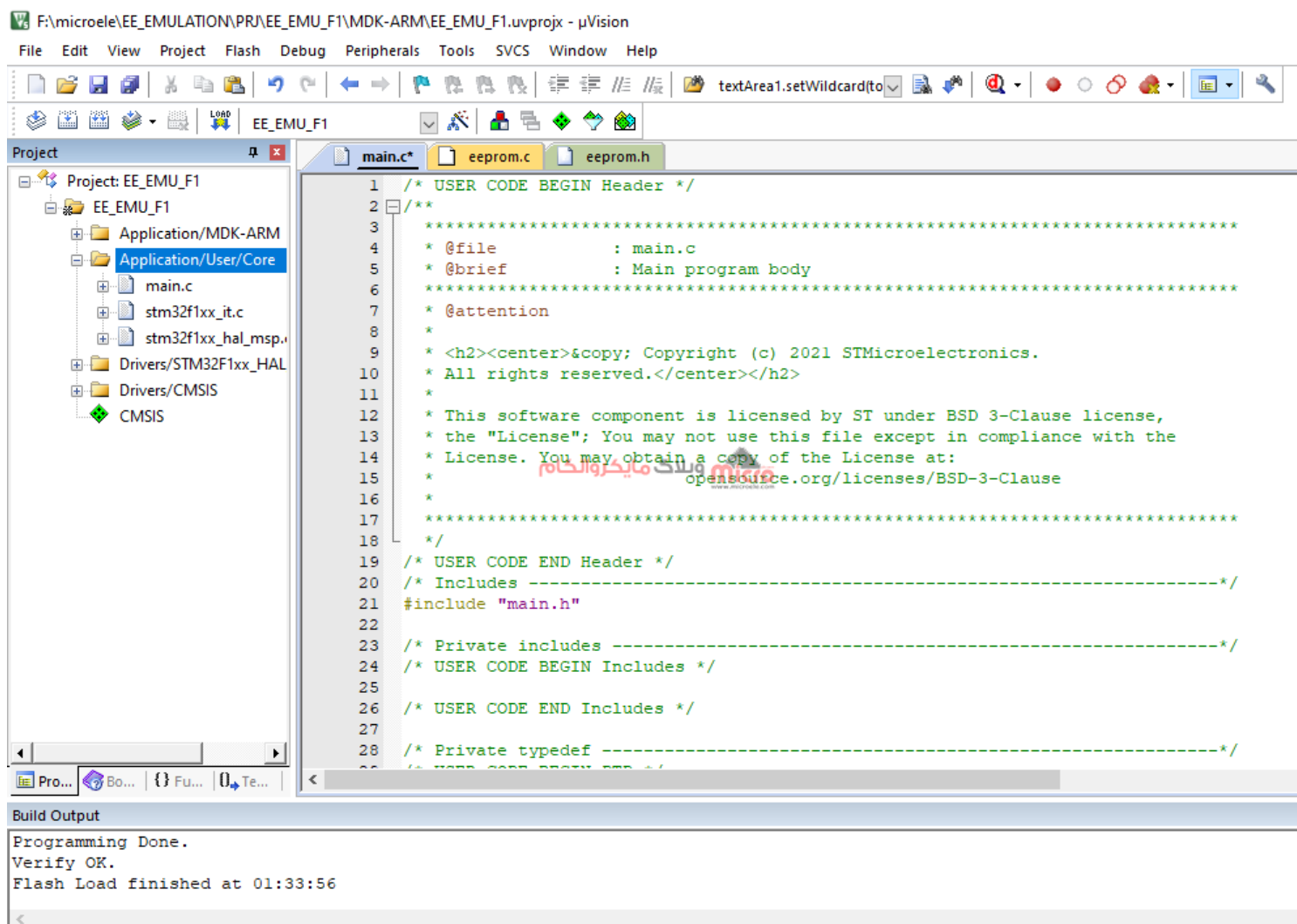


Project Manager STM32 CubeMX

بعد از تولید کد، یک دیالوگ باکس با پیغام موفقیت آمیز بودن تولید کد ظاهر می شود که با کلیک بر روی Open project، مستقیماً به محیط IDE انتخاب شده برای کد نویسی، که در اینجا برای ما محیط Keil می باشد خواهیم رفت.

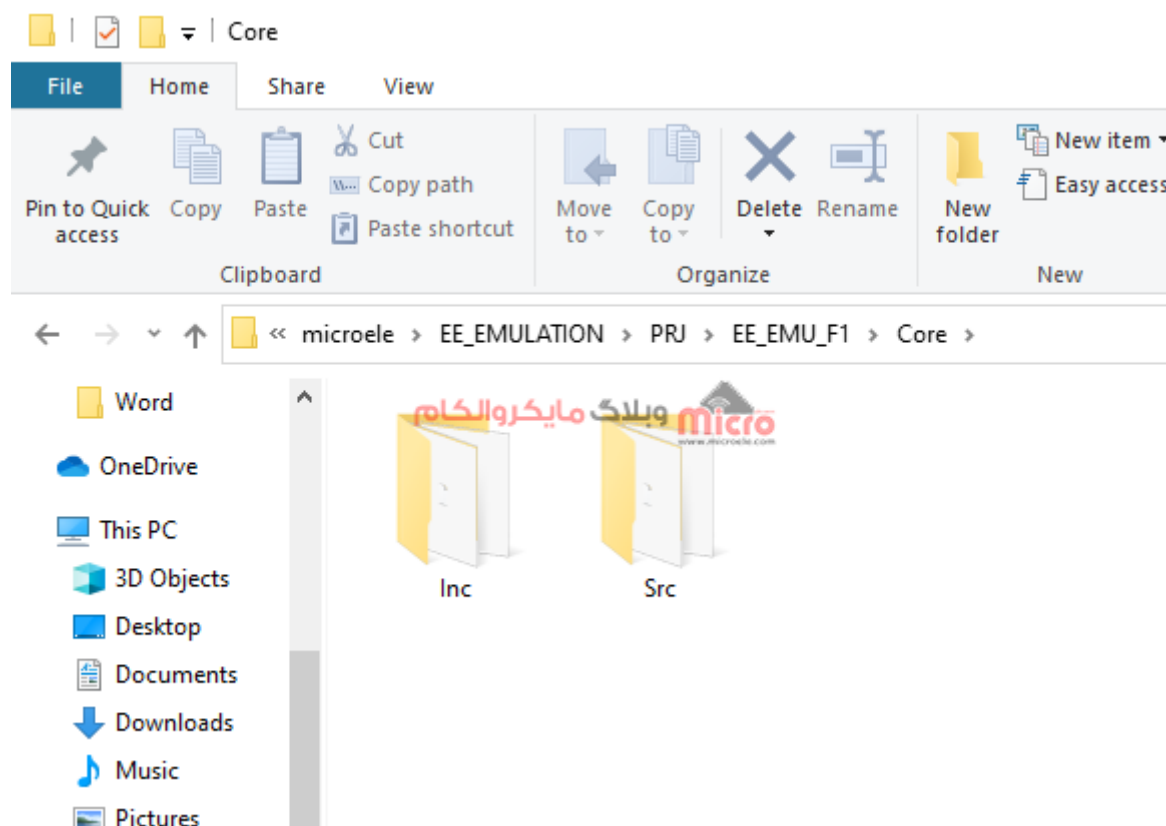


در محیط Keil از قسمت Application/User/Core فایل main.c را باز می‌کنیم.



Keil

حال باید فایل های کتابخانه مربوط به eeprom را به پروژه اضافه کنیم. این کتابخانه شامل یک فایل ".c" و دو فایل ".h" می باشد که باید در مسیر پوشه Core و در داخل پوشه های Src و Inc کپی شوند. توجه شود که در نسخه های قدیمی CUBEMX بعد از تولید کد، پوشه های Src و Inc درون پوشه اصلی پروژه قرار دارند. بعد از دانلود و دریافت فایل های کتابخانه، فایل eeprom.c را در پوشه Src و فایل های eeprom.h و eepromConfig.h را درون پوشه Inc قرار دهید.



کتابخانه EEPROM



File Explorer view showing the directory structure of a project. The path is: <microele> EE_EMULATION > PRJ > EE_EMU_F1 > Core > Inc. The files listed in the 'Inc' folder are:

Name	Date modified	Type	Size
eeprom.h	۱۴۰۰/۰۱/۱۱ ق.ظ ۰۱:۲۵	C/C++ Header File	1 KB
eepromConfig.h	۱۳۹۶/۱۲/۱۱ ق.ظ ۰۴:۰۹	C/C++ Header File	4 KB
main.h	۱۴۰۰/۰۱/۱۱ ق.ظ ۰۱:۰۳	C/C++ Header File	3 KB
stm32f1xx_hal_conf.h	۱۴۰۰/۰۱/۱۱ ق.ظ ۰۱:۰۳	C/C++ Header File	16 KB
stm32f1xx_it.h	۱۴۰۰/۰۱/۱۱ ق.ظ ۰۱:۰۳	C/C++ Header File	3 KB

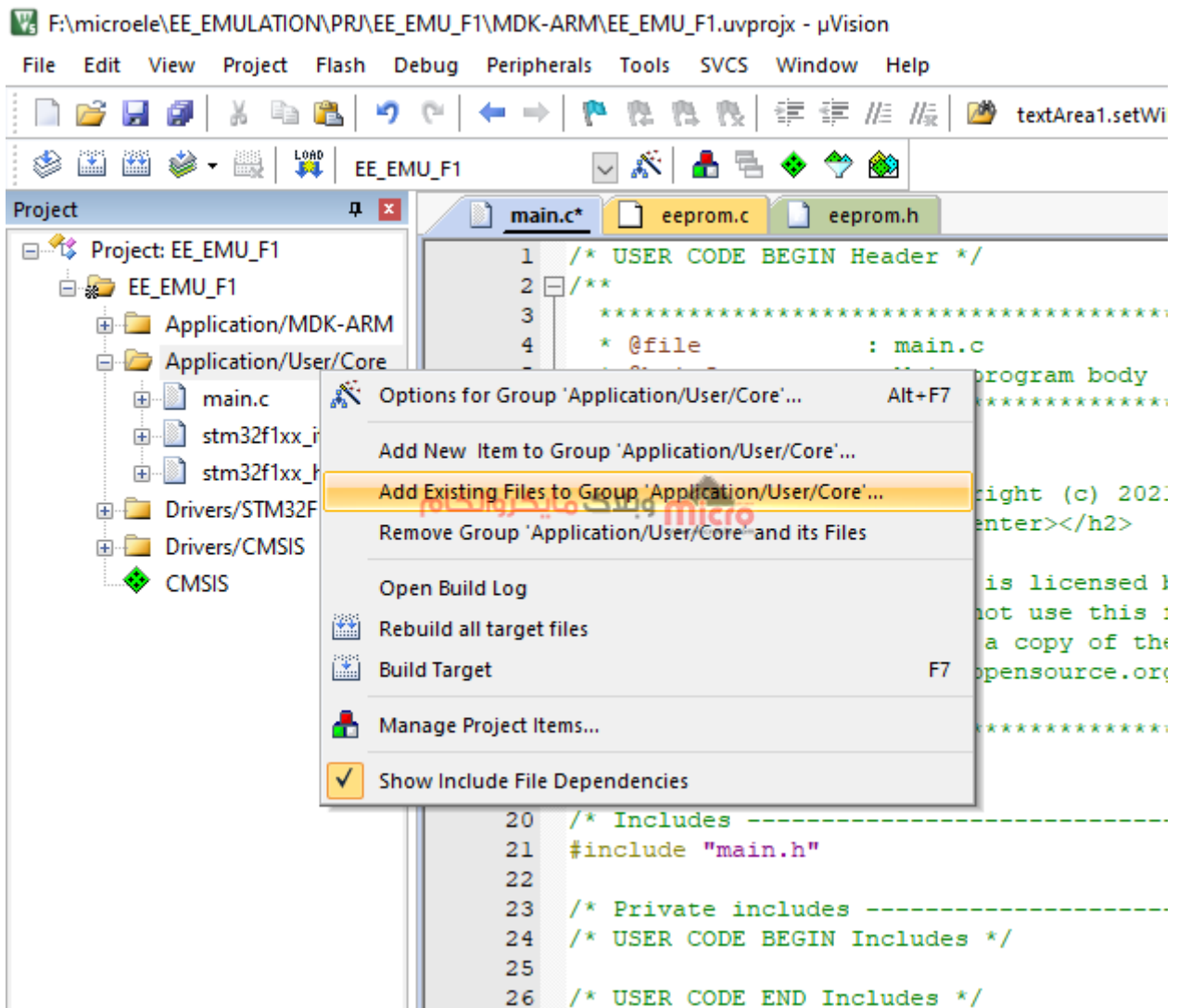
اضافه کردن کتابخانه EEPROM



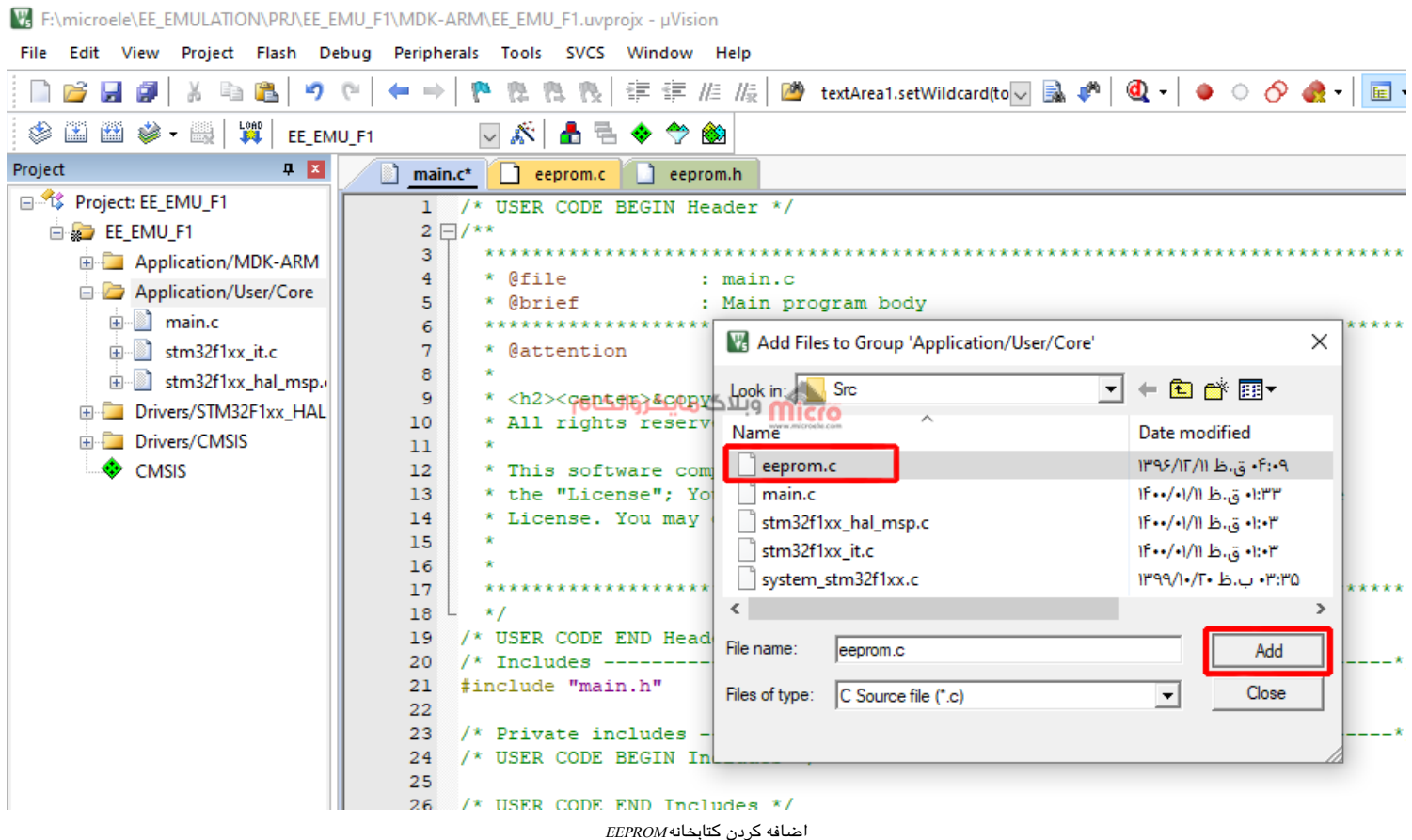
File Explorer window showing the directory structure: `microele > EE_EMULATION > PRJ > EE_EMU_F1 > Core > Src`. The file `eeeprom.c` is highlighted in the file list.

Name	Date modified	Type	Size
eeeprom.c	۱۳۹۶/۱۲/۱۱ ق.ظ ۰۴:۰۹	C Source File	9 KB
main.c	۱۴۰۰/۰۱/۱۱ ق.ظ ۰۱:۳۳	C Source File	7 KB
stm32f1xx_hal_msp.c	۱۴۰۰/۰۱/۱۱ ق.ظ ۰۱:۰۳	C Source File	3 KB
stm32f1xx_it.c	۱۴۰۰/۰۱/۱۱ ق.ظ ۰۱:۰۳	C Source File	6 KB
system_stm32f1xx.c	۱۳۹۹/۱۰/۲۰ ب.ظ ۰۳:۳۵	C Source File	15 KB

اضافه کردن کتابخانه EEPROM



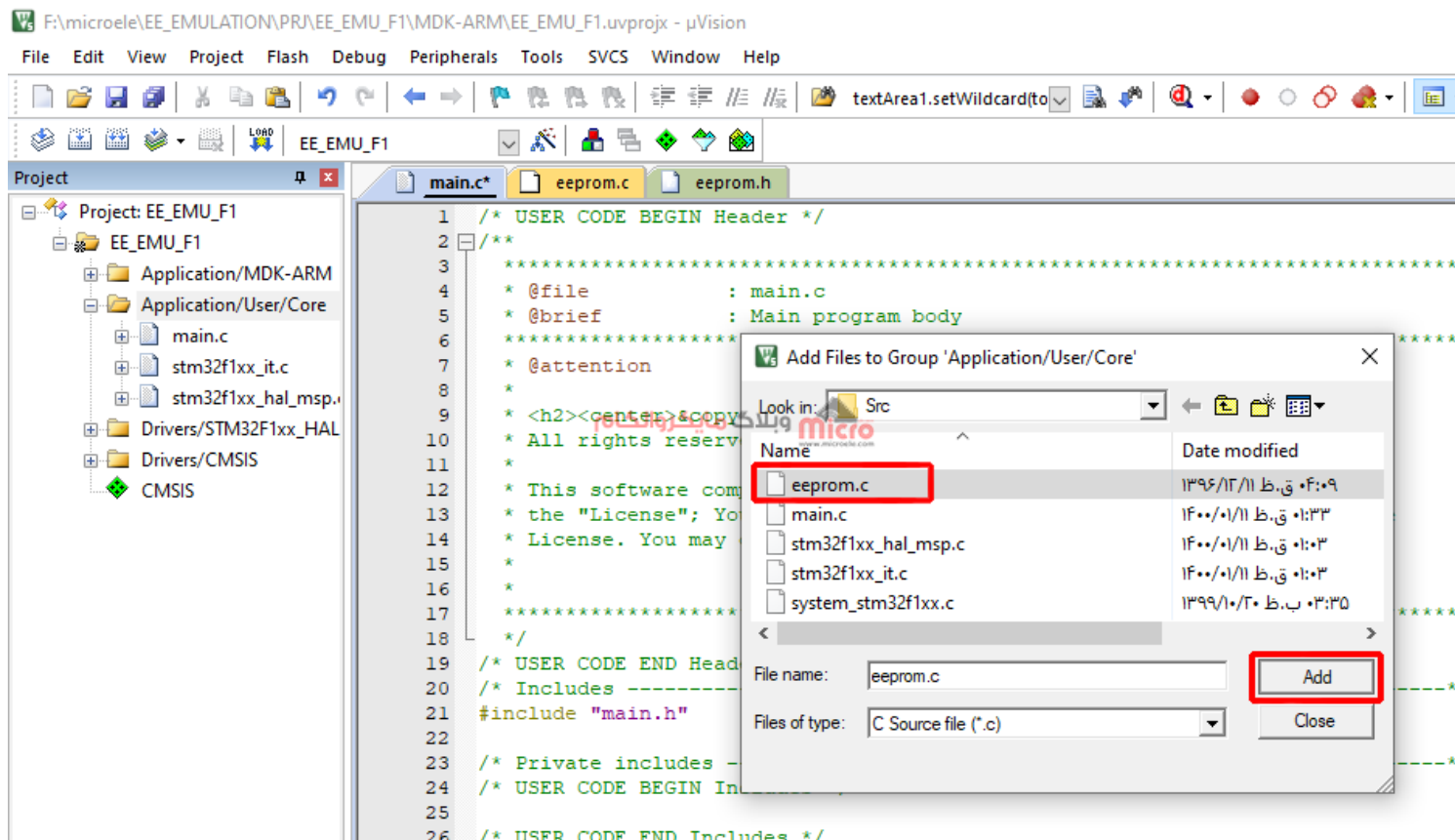
اضافه کردن کتابخانه EEPROM



اضافه کردن کتابخانه EEPROM

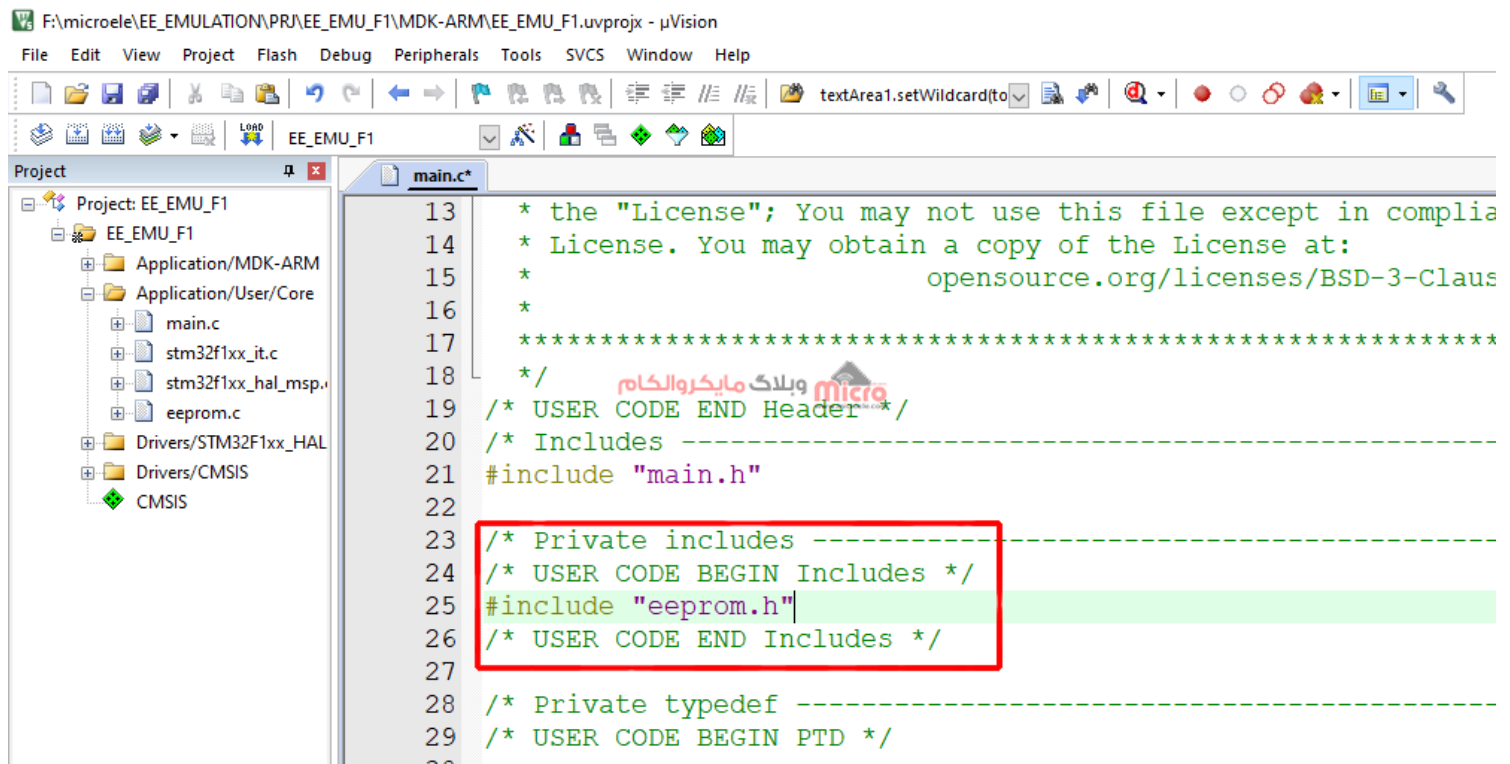
بر روی Application/User/Core کلیک راست کرده و گزینه Add Existing Files to Group را انتخاب و فایل eeprom.c را اضافه کنید.





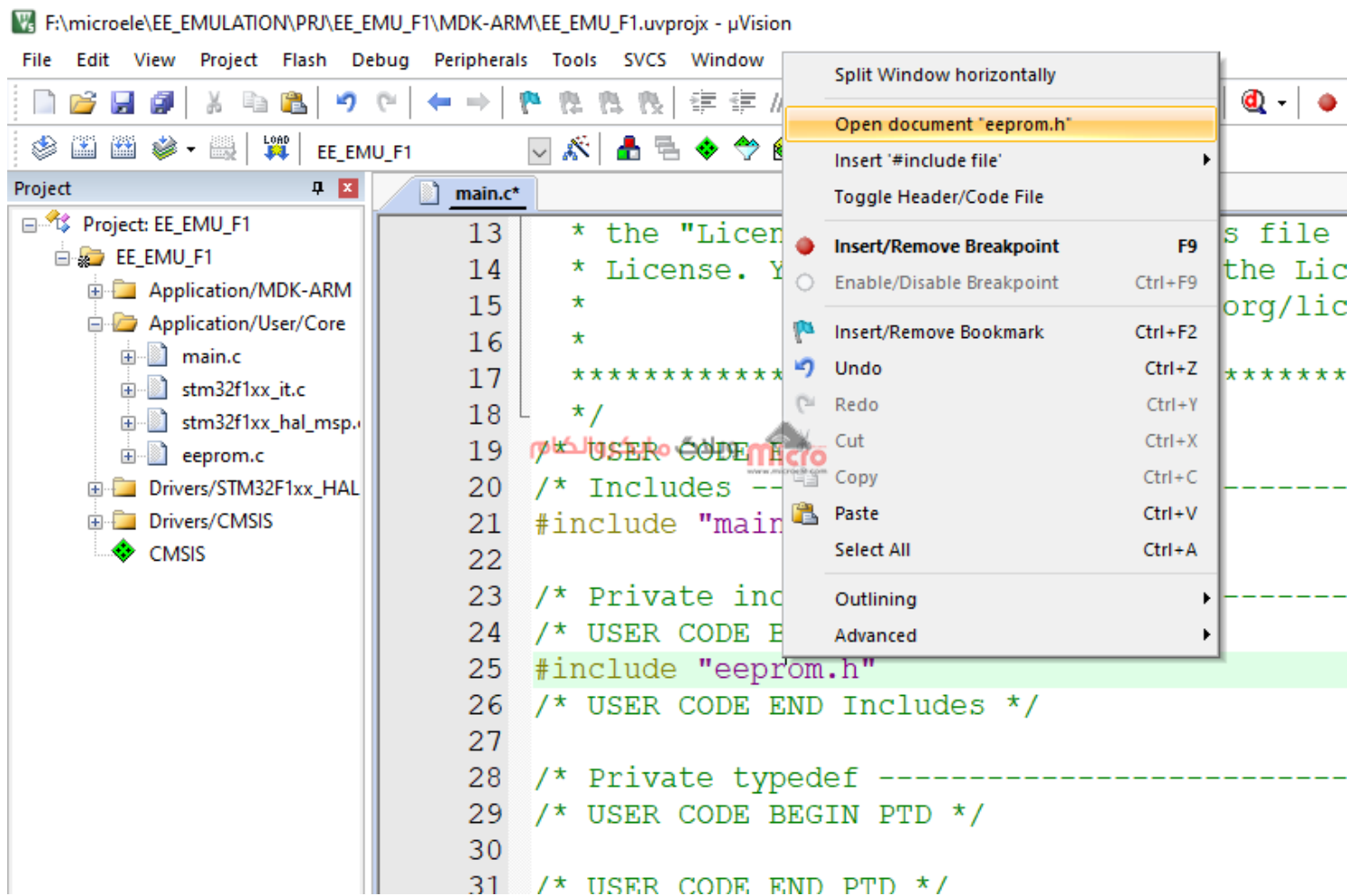
کد نویسی

حال به سراغ کدنویسی می‌رویم. ابتدا در قسمت Private includes، کتابخانه eeprom.h را اضافه کنید.



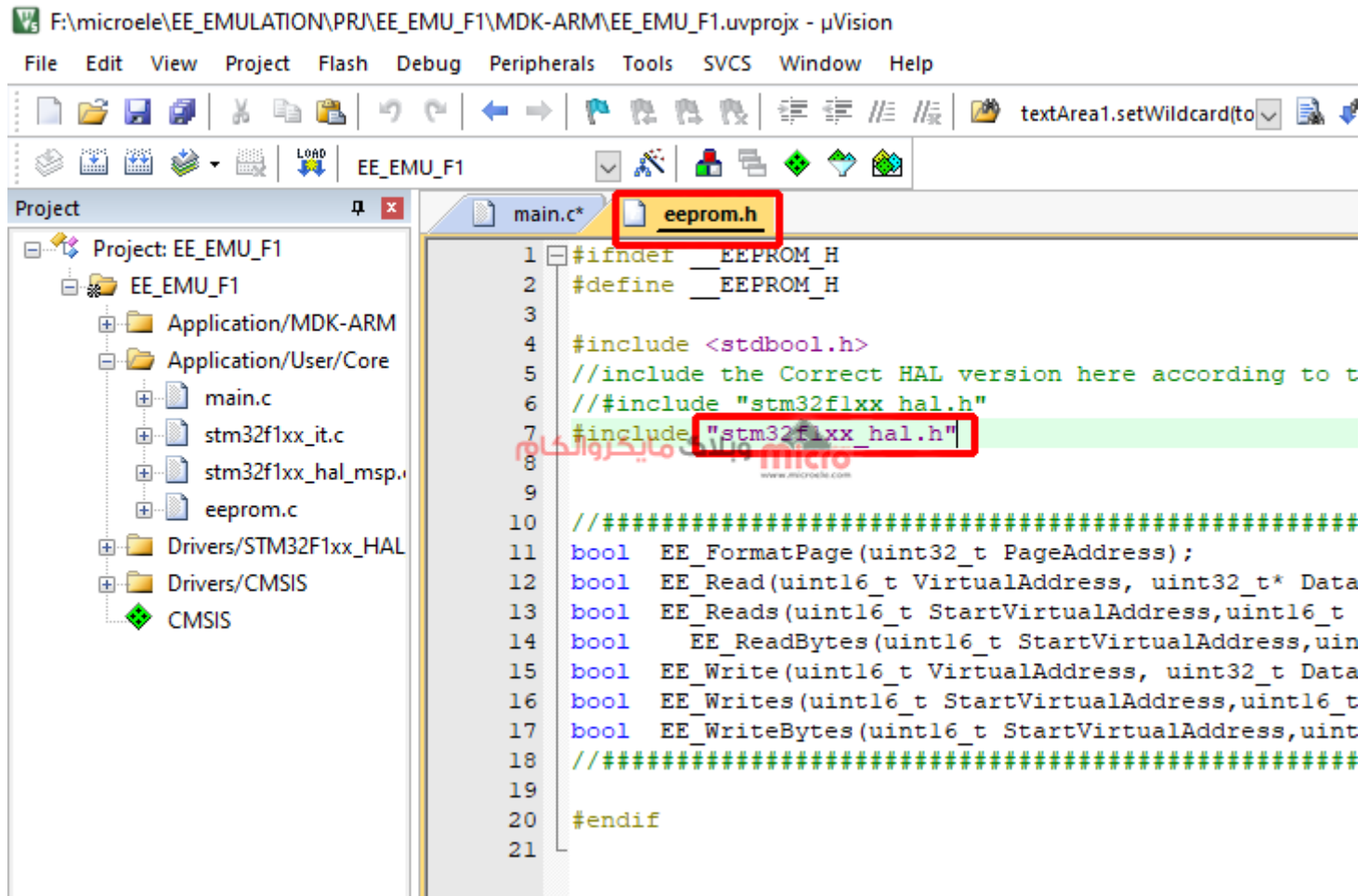
اضافه کردن کتابخانه EEPROM

در صورتی که می‌خواهید سری میکرووی مورد نظر خود را تغییر دهید، می‌توانید با کلیک راست بر روی eeprom.h و انتخاب گزینه open document فایل کتابخانه را باز کرده و هدر میکرووی مدنظر خود را جایگزین کنید.



اضافه کردن کتابخانه EEPROM به پروژه، در محیط Keil

کتابخانه به صورت پیش فرض بر روی stm32f1 تنظیم شده است که با تغییر عدد جلوی f (f0، f3) می‌توانید از این کتابخانه برای سری‌های دیگر نیز استفاده کنید.



اضافه کردن کتابخانه EEPROM به Keil

برای نوشتن بدنه اصلی برنامه، مجدد به فایل main.c بر می گردیم.

توابع مورد استفاده، برای ذخیره سازی مقادیر مورد نیاز و خواندن اطلاعات در این آزمایش به شرح زیر می باشد:

```
EE_Write(n,m);
```

تابع EE_Write برای نوشتن دیتا در حافظه فلش میکرو مورد استفاده قرار می گیرد که در آن n شماره خانه ای از حافظه، و m دیتایی است که می خواهیم ذخیره کنیم.



```
EE_Read(n,q);
```

تابع EE_Read برای خواندن دیتا از حافظه به کار می رود که در آن n شماره خانه و q آرایه ای است که دیتا خوانده شده از حافظه فلش به آن منتقل می گردد.

برای شروع کار ابتدا یک آرایه از نوع unit32_t به صورت گلوبال و همچنین یک متغیر از نوع int به نام cnt با مقدار اولیه صفر تعریف کنید.



F:\microele\EE_EMULATION\PRJ\EE_EMU_F1\MDK-ARM\EE_EMU_F1.uvprojx - µVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

textArea1.setWildcard(to

EE_EMU_F1

Project

Project: EE_EMU_F1

- EE_EMU_F1
 - Application/MDK-ARM
 - Application/User/Core
 - main.c
 - stm32f1xx_it.c
 - stm32f1xx_hal_msp.c
 - eeeprom.c
 - Drivers/STM32F1xx_HAL
 - Drivers/CMSIS
 - CMSIS

main.c*

```
37 /* Private macro -----
38 /* USER CODE BEGIN PM */
39
40 /* USER CODE END PM */
41
42 /* Private variables -----
43
44 /* USER CODE BEGIN PV */
45 uint32_t eeeprom_buffer[2];
46 int cnt=0;
47 /* USER CODE END PV */
48
49 /* Private function prototypes -----
50 void SystemClock_Config(void);
51 static void MX_GPIO_Init(void);
52 /* USER CODE BEGIN PFP */
53
54 /* USER CODE END PFP */
55
56 /* Private user code -----
57 /* USER CODE BEGIN 0 */
```

Build Output

Programming Done.
Verify OK

EEPROM در میکروکنترلر های سری STM32

```
/* USER CODE BEGIN PV */
int cnt=0;
uint32_t eeeprom_buffer[2];
```



```
/* USER CODE END PV */
```

حال برنامه به صورت زیر تکمیل می‌شود:

```
88  MX_GPIO_Init();
89  /* USER CODE BEGIN 2 */
90  EE_Read(0,eeprom_buffer);
91  cnt = eeprom_buffer[0];
92  HAL_GPIO_WritePin(GPIOB,cnt<<4,GPIO_PIN_SET);
93  /* USER CODE END 2 */
94
95  /* Infinite loop */
96  /* USER CODE BEGIN WHILE */
97  while (1)
98  {
99      /* USER CODE END WHILE */
100
101      /* USER CODE BEGIN 3 */
102      if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_12) == 0)
103      {
104          cnt++;
105          if(cnt>15) cnt=0;
106          HAL_GPIO_WritePin(GPIOB,0xf<<4,GPIO_PIN_RESET);
107          HAL_GPIO_WritePin(GPIOB,cnt<<4,GPIO_PIN_SET);
108          EE_Write(0,cnt);
109          HAL_Delay(200);
110      }
111  }
```

Build Output

Programming Done.
Verify OK.
Flash Load finished at 01:33:56

افزودن EEPROM در میکروکنترلر های سری STM32

-در قسمت 1 ابتدا وضعیت کلید خوانده می‌شود. کلید از قبل Pullup شده و با هر باز زدن آن، پایه مربوطه صفر می‌



گردد. مقدار cnt زیاد شده و در نهایت با دستور HAL_GPIO_WritePin عدد داخل متغیر cnt بر روی پایه‌های مورد نظر دیده خواهد شد و از طریق تابع EE_Write مقدار cnt در خانه صفرم حافظه ذخیره می‌گردد.

در قسمت 2 با استفاده از تابع EE_Read، خانه صفرم خوانده شده و در آرایه eeprom_buffer که قبل تر تعریف کرده بودیم ریخته می‌شود. این مقدار به صورت پیش فرض در خانه صفرم آرایه ذخیره شده است که آن را به cnt منتقل می‌کنیم (cnt = eeprom_buffer[0]).

سپس مقدار cnt را مجدداً با دستور HAL_GPIO_WritePin بر روی LED ها نمایش می‌دهیم.

```
/* USER CODE BEGIN 2 */
EE_Read(0,eeprom_buffer);
cnt = eeprom_buffer[0];
HAL_GPIO_WritePin(GPIOB,cnt<<4,GPIO_PIN_SET);
/* USER CODE END 2 */

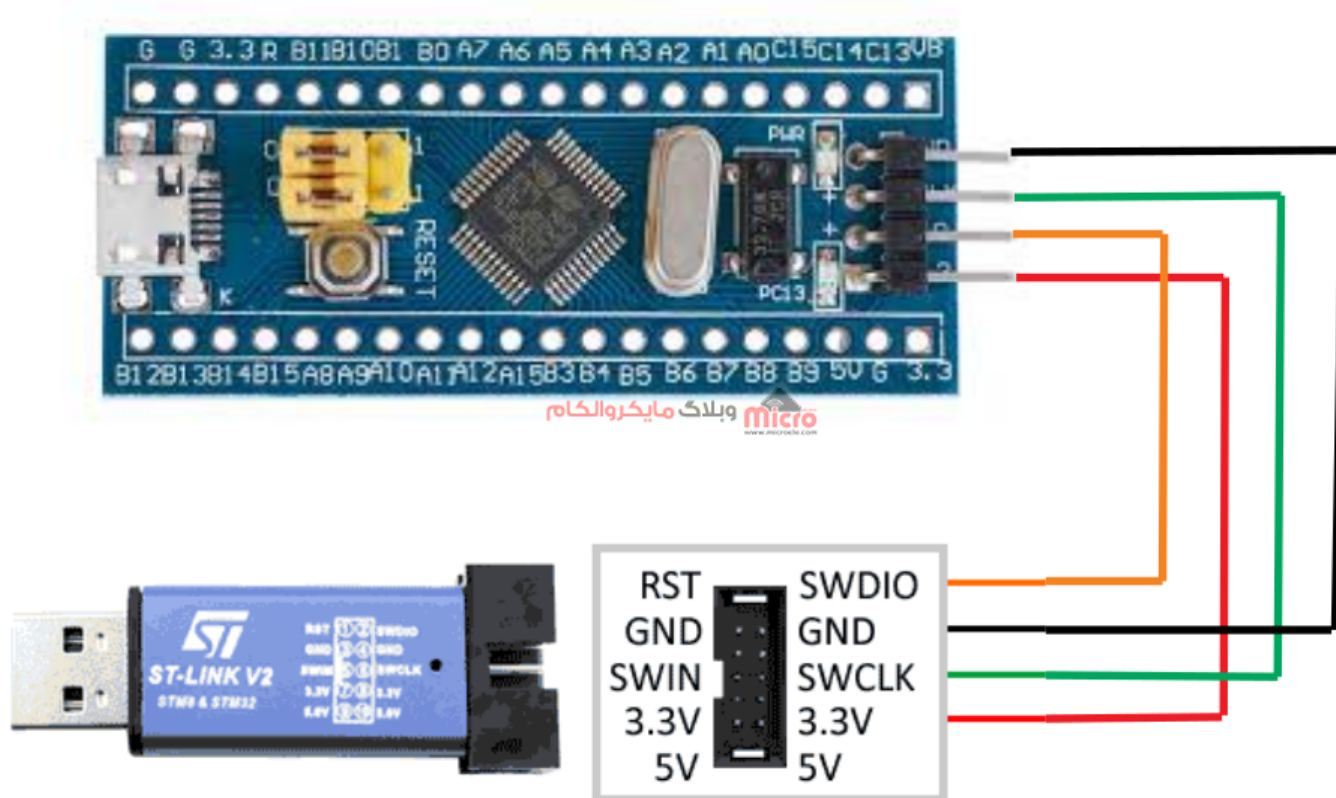
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    if(HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_12) == 0)
    {
        cnt++;
        if(cnt>15) cnt=0;
        HAL_GPIO_WritePin(GPIOB,0xf<<4,GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOB,cnt<<4,GPIO_PIN_SET);
        EE_Write(0,cnt);
        HAL_Delay(300);
    }
}
```

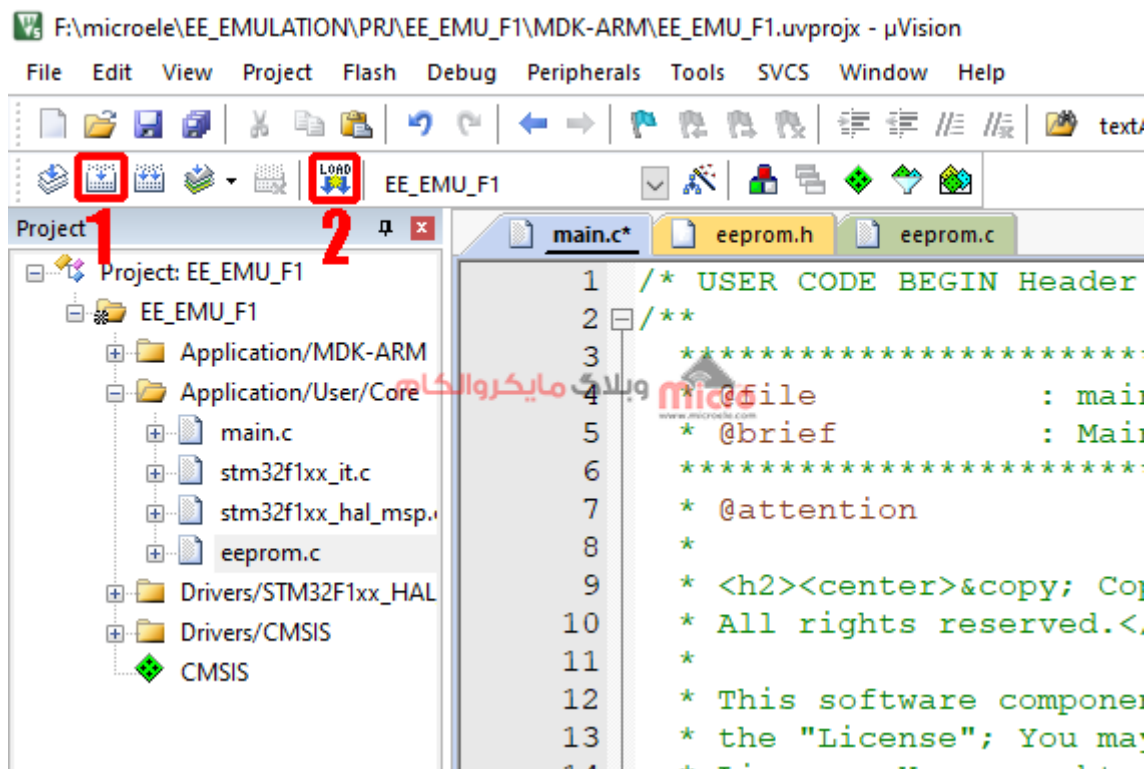


```
/* USER CODE END 3 */
```

حال برنامه را Build کرده و با اتصال پروگرامر ST-LINK به برد، برنامه را LOAD می‌کنیم.



اتصال Bluepill به پروگرامر ST-LINK V2



کامپایل و LOAD کردن برنامه در میکرووی STM32

اکنون با فشردن کلید عدد روی LED ها افزایش خواهد یافت. حال اگر تغذیه مدار را قطع کنیم و مجدد تغذیه را متصل نماییم، مشاهده خواهد شد که آخرین عدد دوباره روی LED ها به نمایش در می آید و با فشردن کلید، شمارش از همان جایی که تغذیه مدار قطع شده بود ادامه خواهد یافت. همچنین می توانید نتیجه کار را در ویدیوی زیر مشاهده کنید:

نتیجه گیری

در این مطلب حافظه EEPROM را بر روی حافظه FLASH در میکروکنترلر های STM32، بدون نیاز به اضافه کردن EEPROM خارجی ایجاد کردیم تا امکان ذخیره سازی مقادیر مهمی را که نیاز است توسط مصرف کننده در سامانه درج شود را در حافظه خود میکرو ذخیره سازی کنیم در نهایت با هر بار روشن شدن مدار، مقادیر مربوطه در سامانه بارگذاری شده و نیاز به تنظیم مجدد آن ها توسط کاربر مرتفع می گردد.



امیدوارم از این آموزش کمال بهره را برده باشید. در صورتی که هرگونه نظر یا سوال داشتید درباره این آموزش لطفاً اون رو در انتهای همین صفحه در قسمت دیدگاه ها قرار بدید. در کوتاه ترین زمان ممکن به اون ها پاسخ خواهم داد. اگر این مطلب براتون مفید بود، اون رو حتماً با دوستانتون به اشتراک بگذارید. همینطور میتونید اون رو پس از اجرای عملی توی اینستاگرام با هشتگ #microelecom به اشتراک بگذارید و پیج مایکروالکام (@microelecom) رو هم منشن کنید.