4/26/2023

Mostafa Mohamed Edrees

learn-in-depth

Lab2

Embedded C Lesson 3

**Lab2**

**Required:**

WRITE BAREMETAL SW ON ARM CORTEX-M3 32-BIT MICROCONTROLLER STM32F103C8T6 CHIP TO TOGGLE LED.

**Physical Board:**

STM32F103C6

**Processor:**

Arm CORTEX-M3

**Name:**

Mostafa Mohamed Edrees

**Supervisor:**

Eng. Keroles Shenouda

Table of Contents

[**Introduction** 3](#_Toc133547339)

[**ARM CORTEX-M3** 3](#_Toc133547340)

[**Steps** 3](#_Toc133547341)

[**Application File** 4](#_Toc133547342)

[Write app.h file 5](#_Toc133547343)

[Write app.c file 6](#_Toc133547344)

[**Lab2 Version1** 7](#_Toc133547345)

[**Makefile** 7](#_Toc133547346)

[**Startup.s** 8](#_Toc133547347)

[**Linker\_script.ld** 9](#_Toc133547348)

[**Build the project** 10](#_Toc133547349)

[**Run the project on Proteus** 10](#_Toc133547350)

[**Lab2 Version2** 10](#_Toc133547351)

[**Makefile** 11](#_Toc133547352)

[**startup.c** 12](#_Toc133547353)

[**Linker\_script.ld** 13](#_Toc133547354)

[**Build the project** 14](#_Toc133547355)

[**Run the project** 15](#_Toc133547356)

[**Debug the project** 16](#_Toc133547357)

[**Analyze Mapfile** 17](#_Toc133547358)

[**Conclusion** 19](#_Toc133547359)

# **Introduction**

Hello everyone, we will write a bare metal SW on STM32F103C6 Board and ARM CORTEX-M3 Processor to toggle led.

We will write this lab from scratch so we will write app.c, app.h, startup, linker\_script.ld and Makefile and we will run this lab on Proteus and debug it on Proteus too.

## **ARM CORTEX-M3**

We should know important information about this processor at first.

This processor when it is starts to work the PC (program counter) will point to the entry point and the entry point should contain the address of SP (stack pointer) so this processor initialize SP by itself without we write assembly code to do that so we can write startup code by C and put this code at the next address after the address that we put SP in it.

So we will do this lab with two versions:

1st Version: do this lab with **startup.s**

2nd Version: do this lab with **startup.c**

## **Steps**

Steps to do this lab:

* Write application file.
* Write startup.c or startup.s.
* Write linker\_script.
* Write makefile.
* Build the project by using makefile.
* Run the project on proteus.
* Debug the project on proteus.

Notes:

In the two versions we do the same thing so:

The application file will become common between the two versions.

So let’s start to write application file.

## **Application File**

We want to write application to connect led to **GPIO\_PortA pin13** in **STM32F103C6** to toggle this led.

To Toggle Led we need to work with two peripherals:

* RCC (reset and clock control).
* GPIO (general purpose input/output).

We should open datasheet and get the base address of each peripheral:

* Base address of RCC is **0x40021000**
* Base address of GPIO PortA is **0x40010800**

***RCC:***

It has a bit2 (IOPAEN ) in APB2ENR Register used to enable PortA.

Offset of APB2ENR Register is **0x18**

Bit Number of IOPAEN is **2**

***GPIO\_PortA:***

It has two regiseters:

* CRH Register

We should write 2 “0010” on it from bit20 to bit24.

Offset of CRH Register is **0x04**

* ODR Register

It has pin13 that will connect led with it so we can turn on/off the led by sending

1/0 to this pin.

Offset of ODR Register is **0x0C**

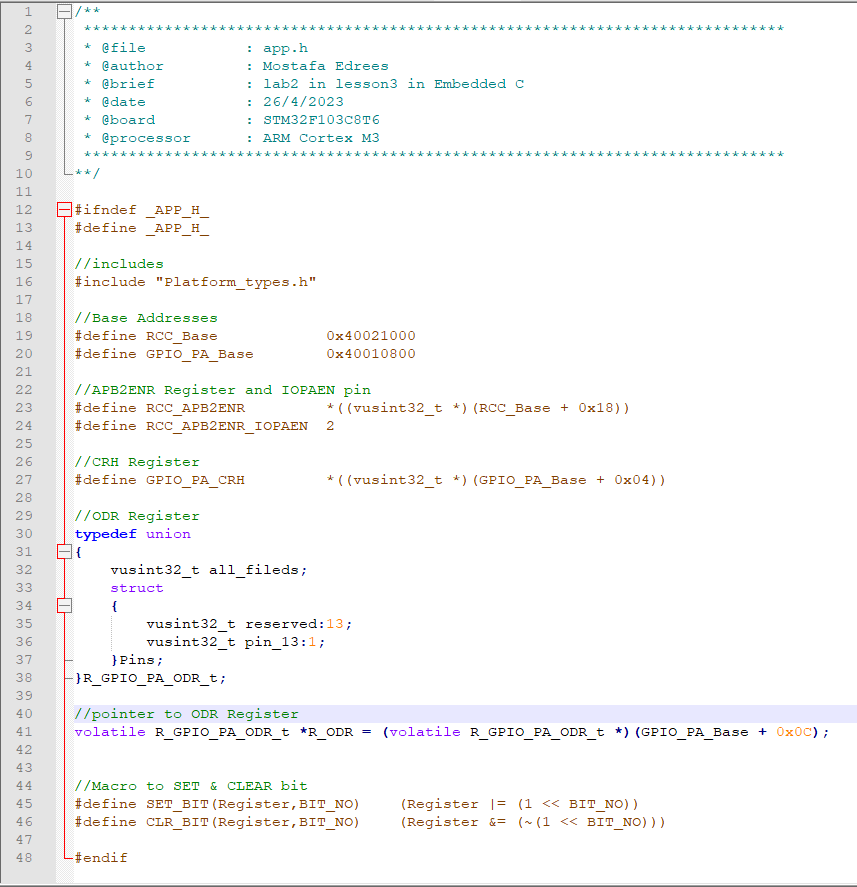
We will put a delay between turn on and turn off the led to notice the toggle.

We do the delay by using for loop.

**Delay**

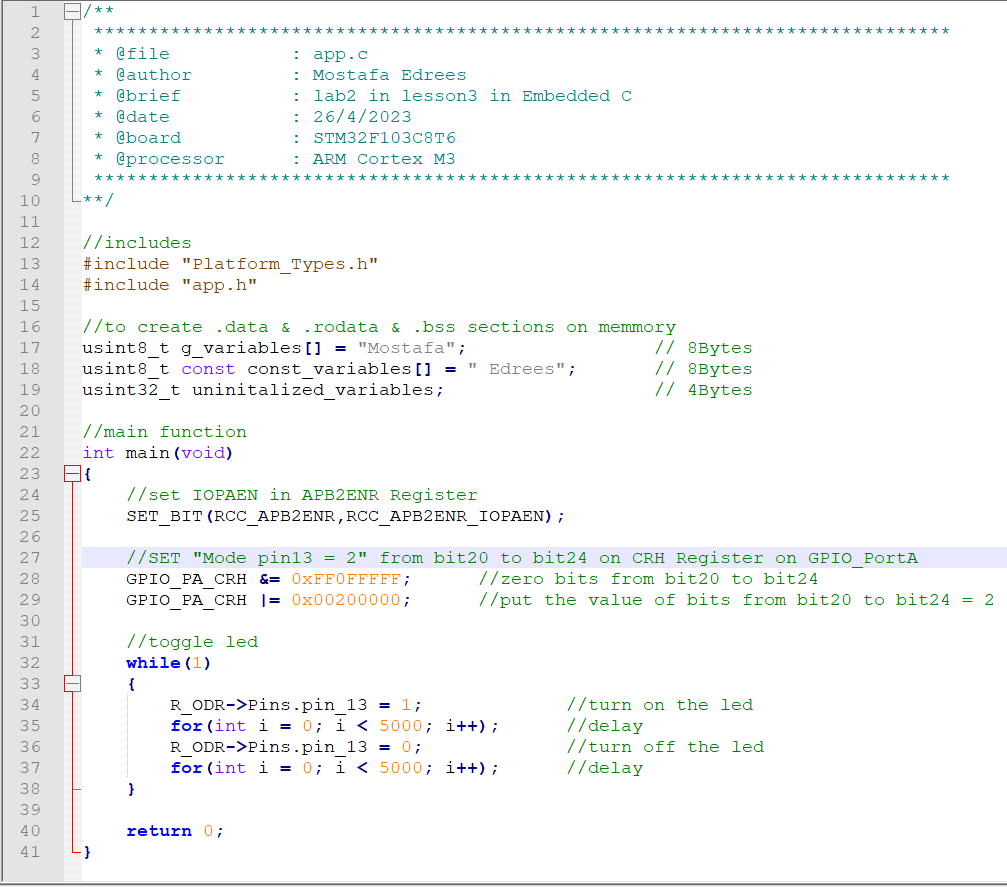
### Write app.h file

In this file we will put the definition of base addresses and registers.

**app.h**

### Write app.c file

In this file we will implement the application to toggle the led.

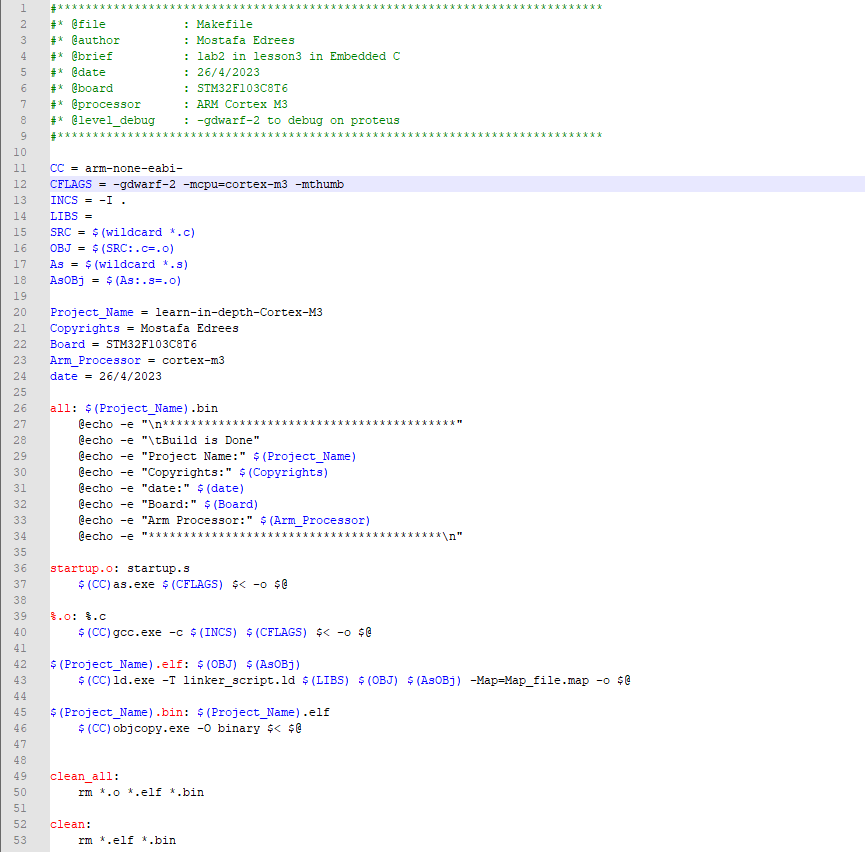
**app.c**

# **Lab2 Version1**

We will write Makefile & startup.s & linker\_script.ld files for this version then build the project by using make file then run it on proteus.

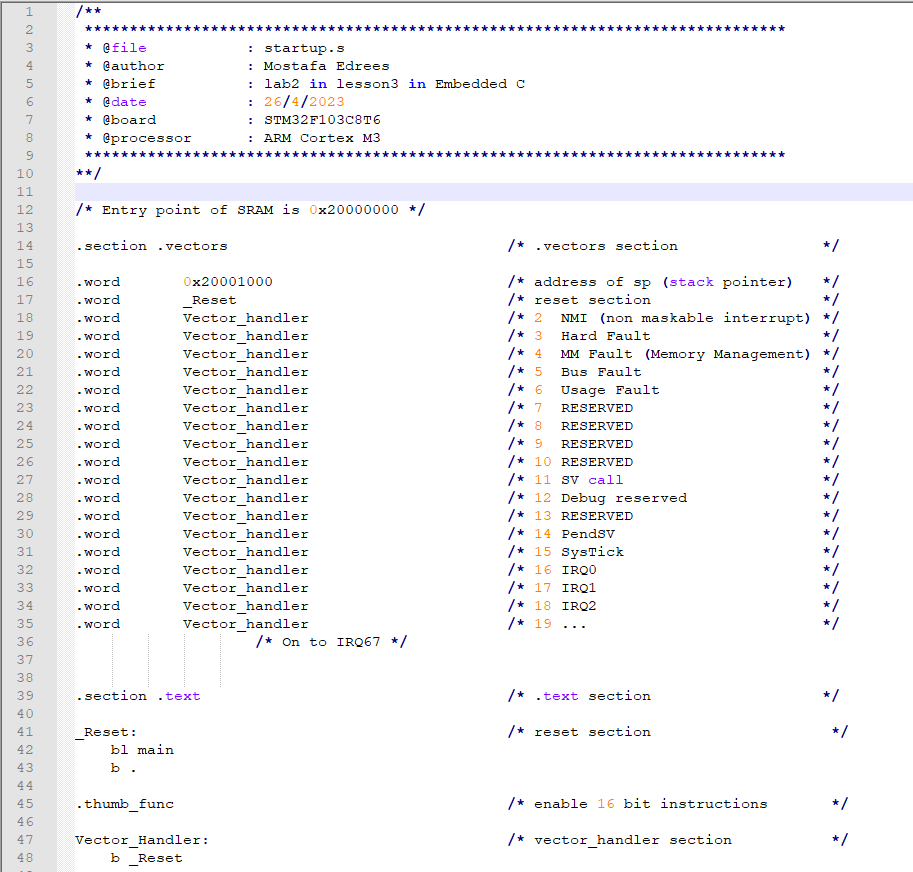
## **Makefile**

In this file we write makefile to auto build the lab.

**Makefile**

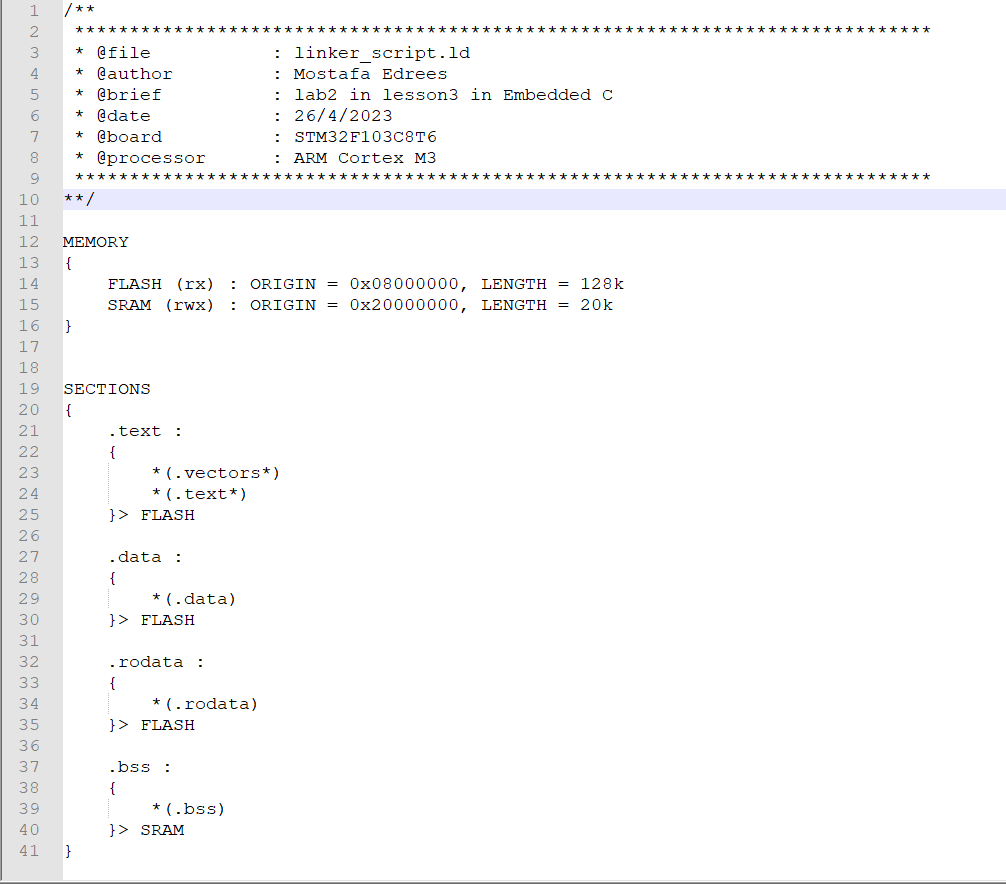
## **Startup.s**

In this file we write startup code with assembly language.

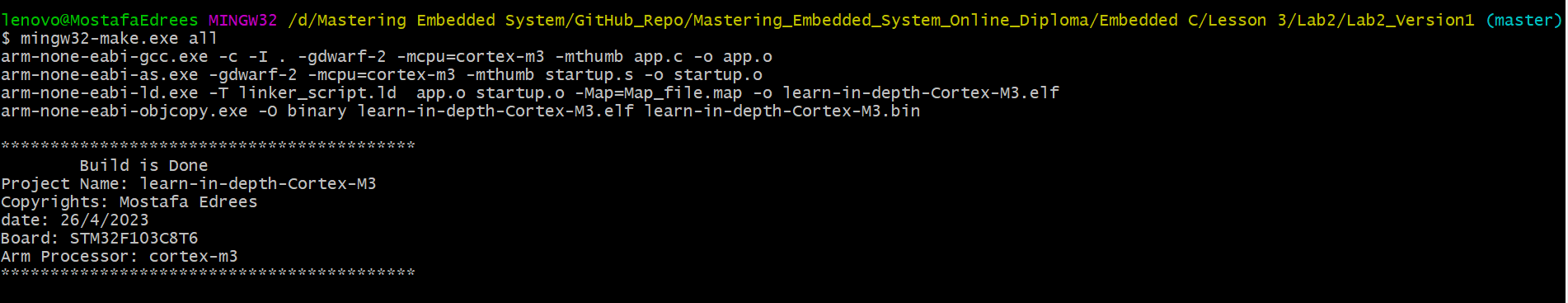
**startup.s**

## **Linker\_script.ld**

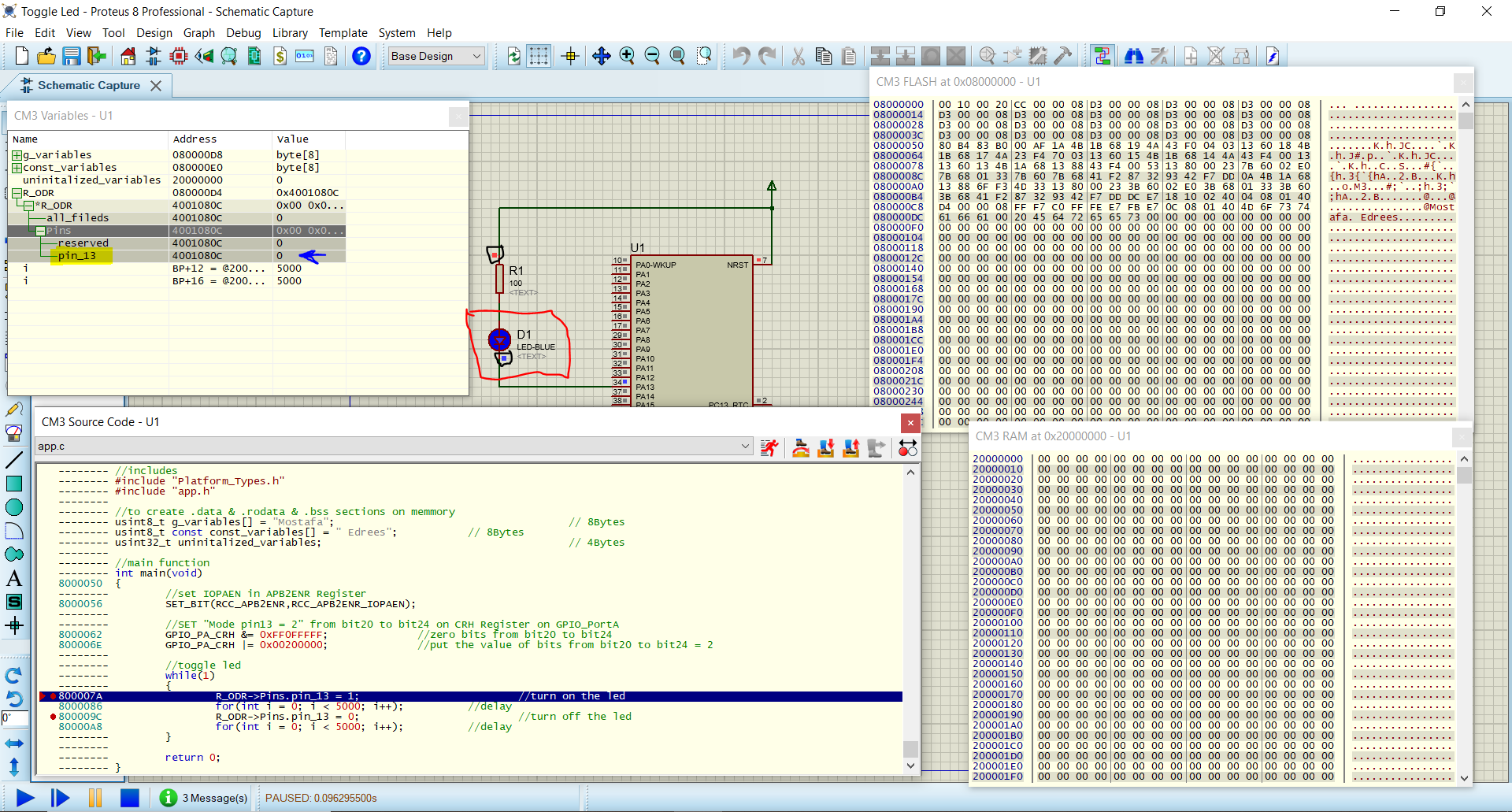
In this file we write linker\_script but in this file we will but .data section in FLASH and don’t copy it to RAM and we don’t initialize .bss section in RAM with zero we will do this in Lab2\_Version2.

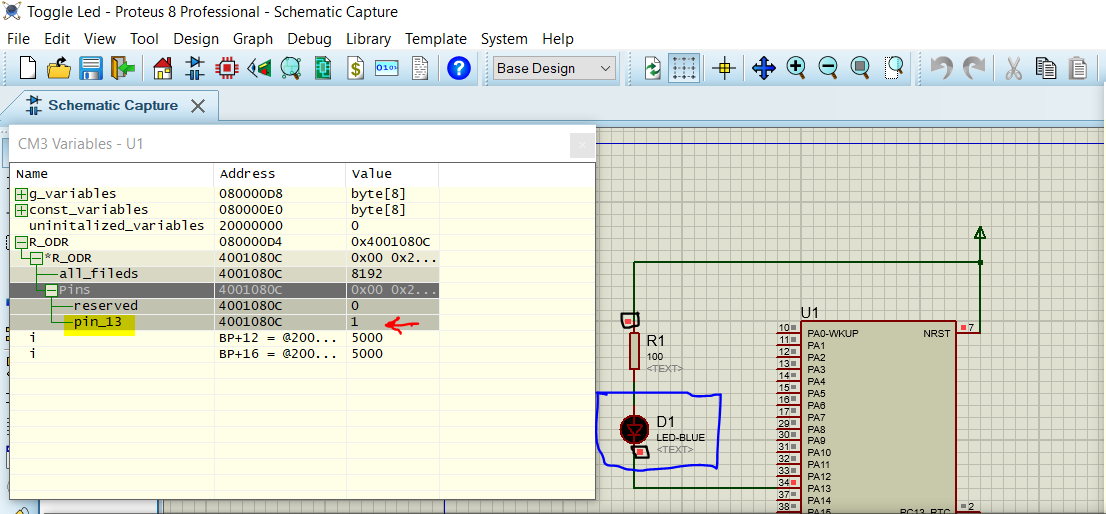
**linker\_script.ld**

## **Build the project**

We will build the project by using Makefile.

## **Run the project on Proteus**

**led is turn on:**

**Led is turn off:**

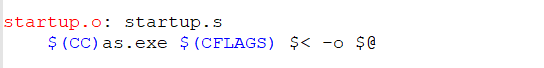
You can see the video of running project from this link:

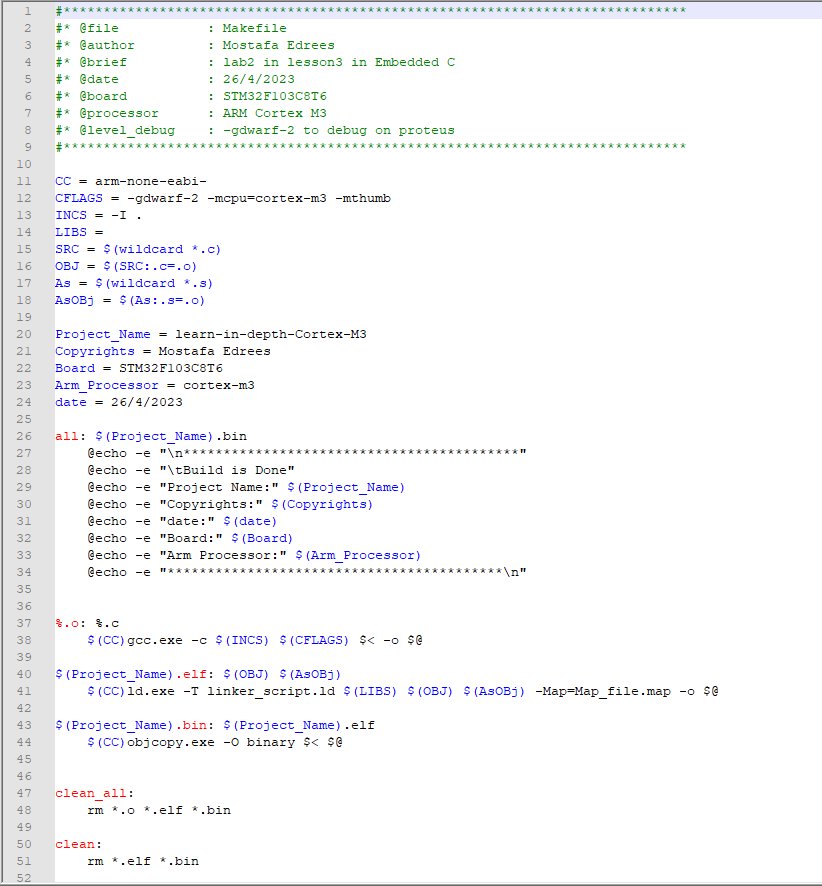
**[Video](https://drive.google.com/drive/u/2/folders/1pEAbWjwYVH2gblY3OoKVsGTH_oFWfrr-)**

# **Lab2 Version2**

We will modify something in Makefile and write startup.c & linker\_script.ld files for this version then build the project by using make file then run it on proteus and debug it on proteus.

## **Makefile**

We will delete this two lines from Makefile because we will write startup code with C.

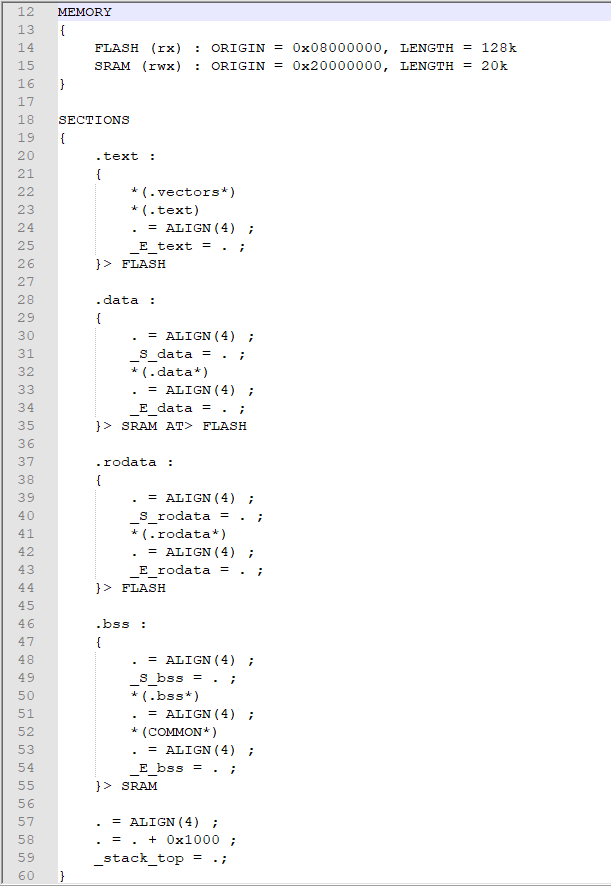
**Makefile**

## **startup.c**

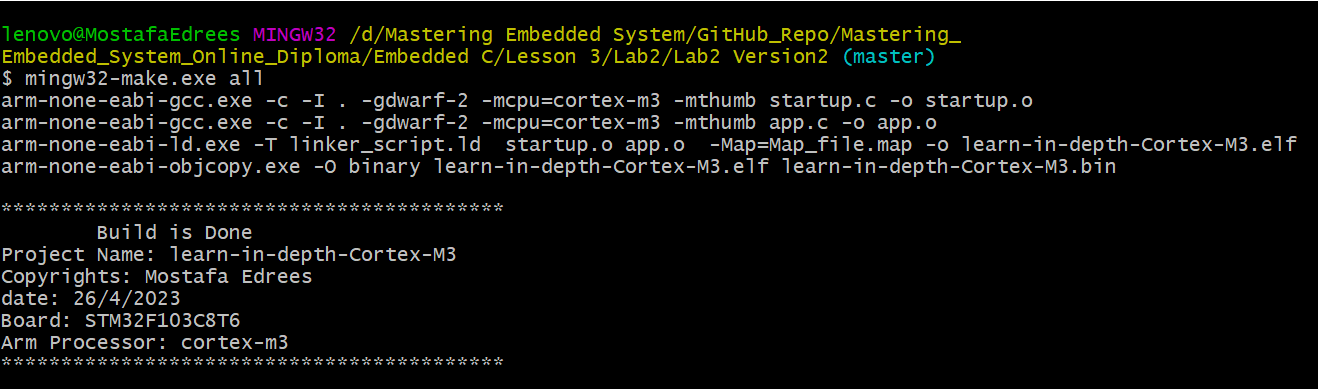
In startup.c:

* We put the address of SP (stack pointer) at the entry point of flash memory.
* We copy .data section from FLASH to RAM.
* We locate .bss section in RAM and initialize it with zero.

## **Linker\_script.ld**

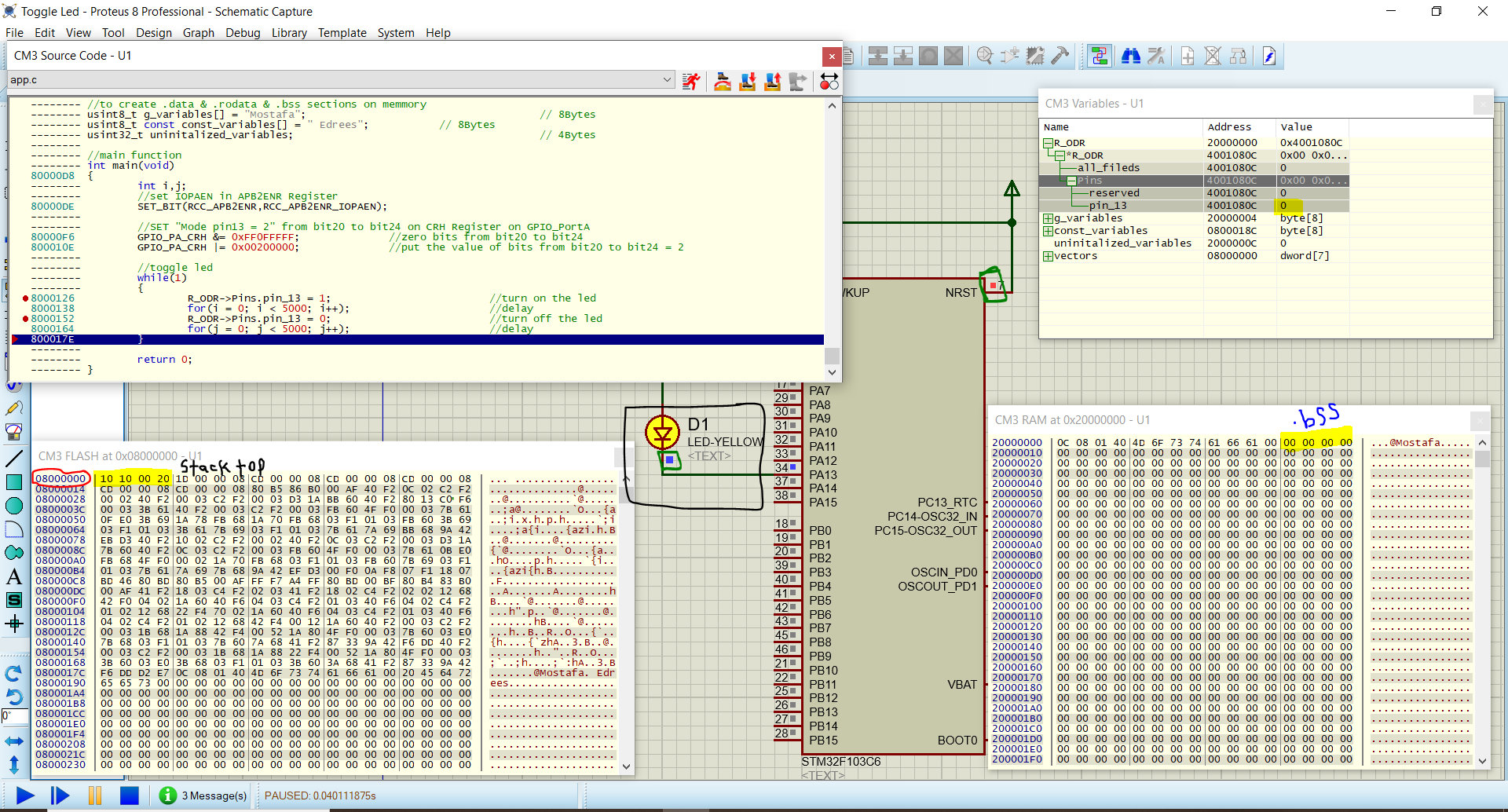
In this file we put symbols to define start and end of each section to use it to copy .data section from FLASH to RAM and locate .bss section in RAM.

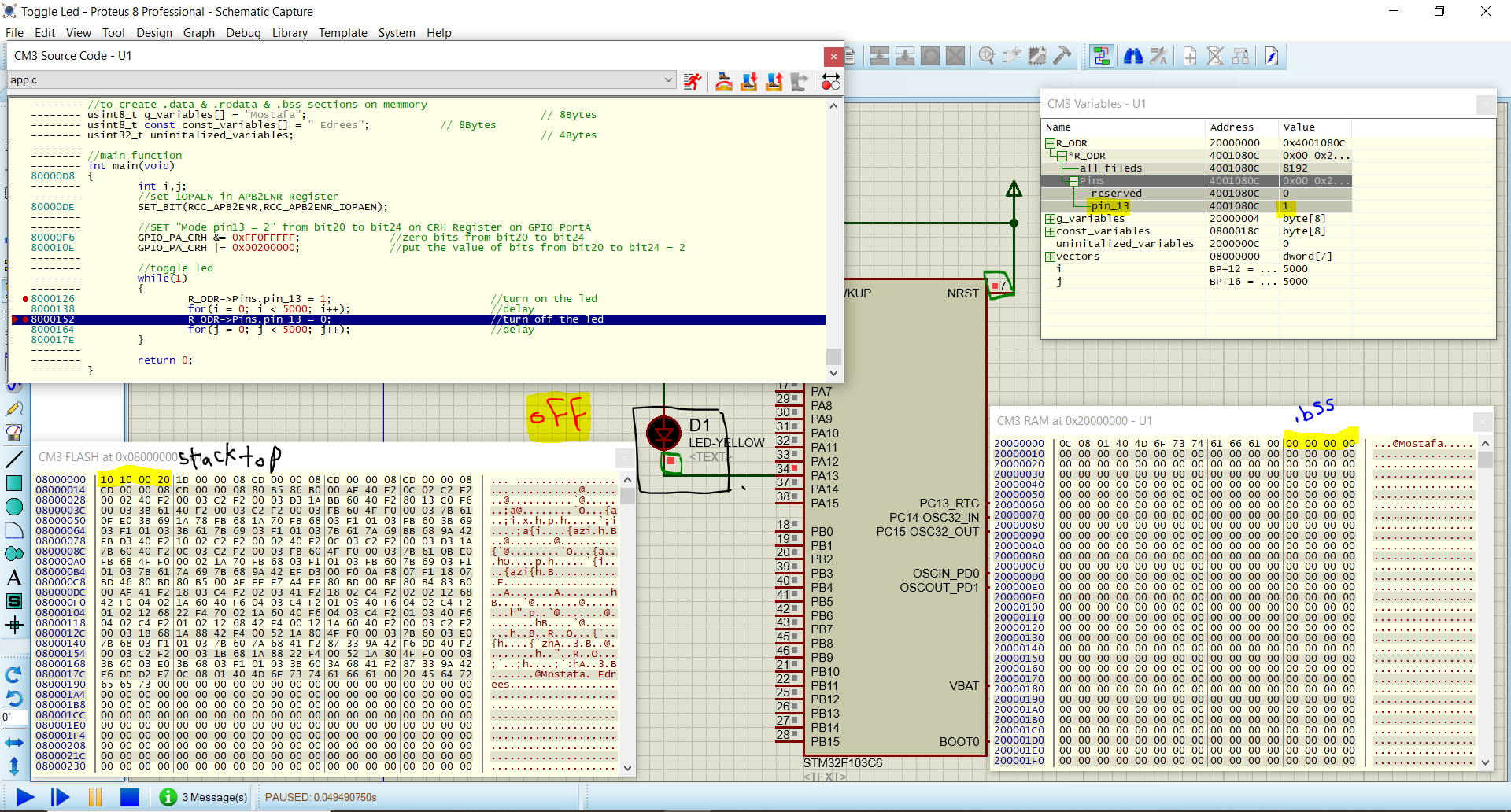
## **Build the project**

We will build the project by using Makefile.

**After that we will run & debug lab on Proteus.**

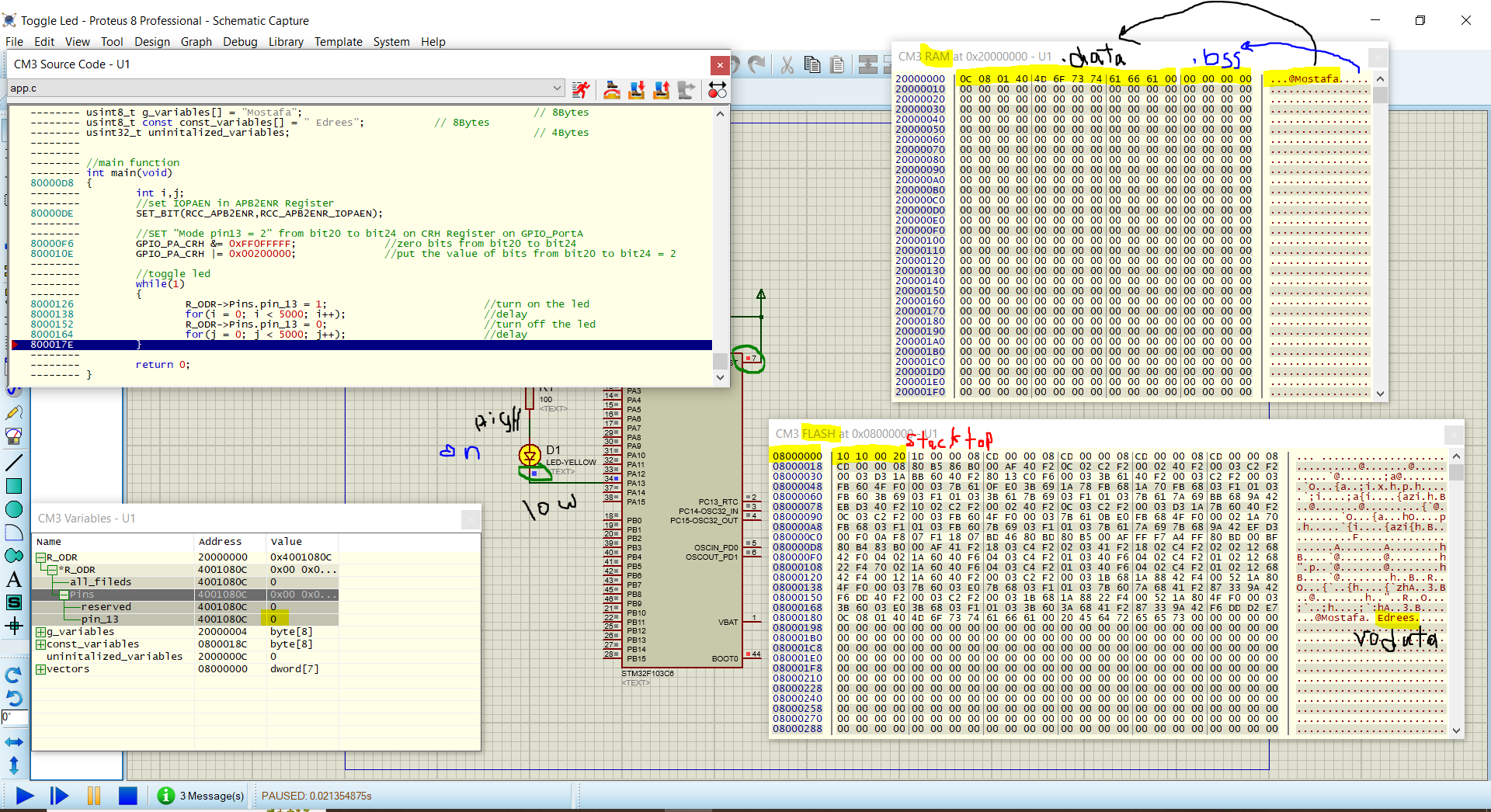
## **Run the project**

**Turn on the led**

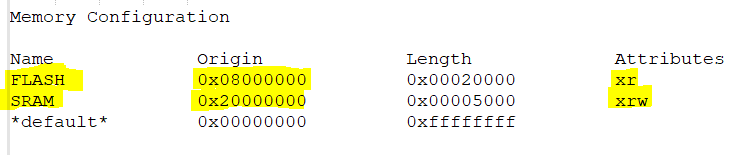
**Turn off the led**

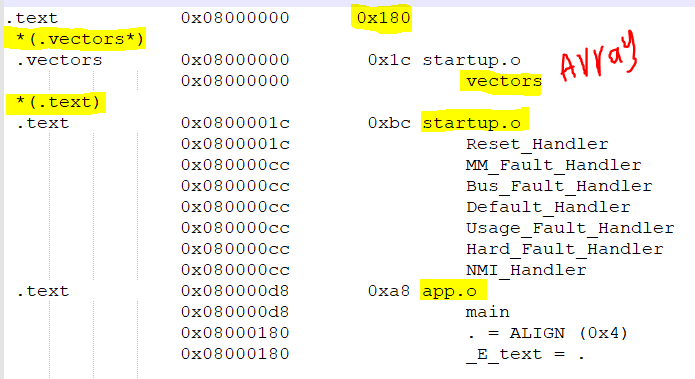
## **Debug the project**

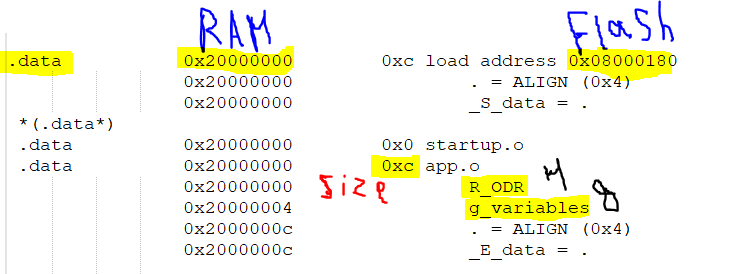
You can see the debug of the lab on proteus in this video:

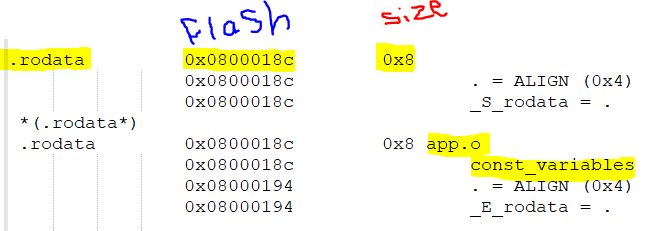
[**Video**](https://drive.google.com/drive/u/2/folders/1d4_mvPQkq_B-fYGQUnpfoneGoYT6DIOW)

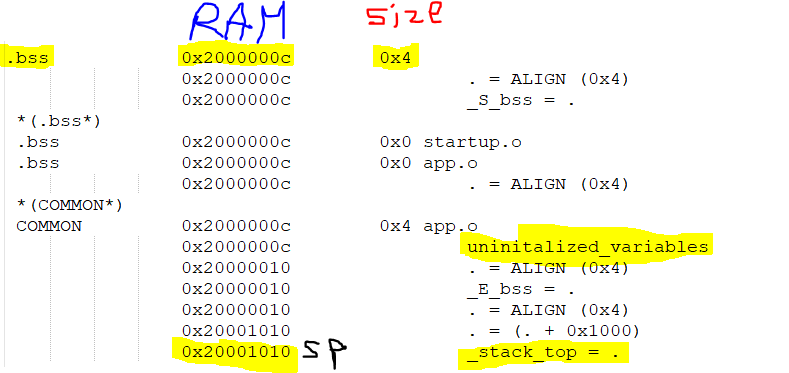
## **Analyze Mapfile**

**Memory**

**.text section**

**.data section**

**.rodata section**

**.bss section**

# **Conclusion**

After doing this lab we know important note:

**“We can write startup code with C not only assembly.”**

**Explanation:**

In some processor like CORTEX-M3 we should put address of SP (stack pointer) at the entry point of the program so the processor define SP without we write assembly code to do that so after this operation the CPU will understand C so we can write startup with C and put it after the address containing the address of SP if the processor don’t have this feature we will write startup code with assembly and we will define SP by using assembly before we jump to main function.



**Good Luck**