Table of contents

1. Introduction
2. Specifications to toggle led based on arm-cortex-m4
3. Source code

3.1 – main application code

3.2 – startup code with c

3.3 -linker script

3.4- make file

4- map file

5- run application on uvision keil tool with feature from edx course (shape the world).

6– Debug with uvision keil tool with feature from edx course (shape the world).

1- Introduction:

Setting stack top by location counter in linker script file and extern it in startup with c done before but, in this report, I will make in startup.c stack top as a variable not a symbol with no need a linking

I will Execute this simple application on a virtual board using uvision tool with feature from edx course (shape the world).

2- Specifications to toggle led based on arm-cortex-m4 :

-led is connected to GPIO port F3

- to make a GPIO toggling in stm4c123,you need to work with two peripherals:

- SYSCTL\_RCGC2 (System control register)

-GPIO F (general purpose i/o)

- to access SYSCTL\_RCGC2 (0x400FE108)set to (0x20)

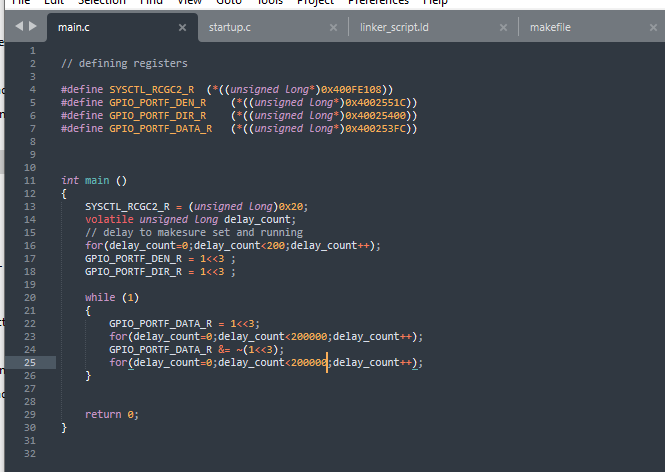
- to enable GPIOF (0x4002551c)set to 1 bit 3

- to enable Direction (0x40025400)set to 1 bit 3

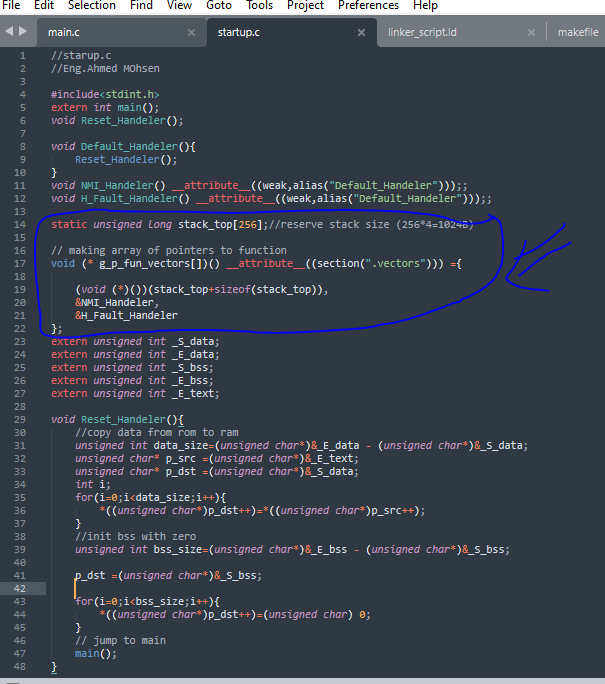
- to write GPIOF (0x400253fc)

3- Source code

3.1 – main application code



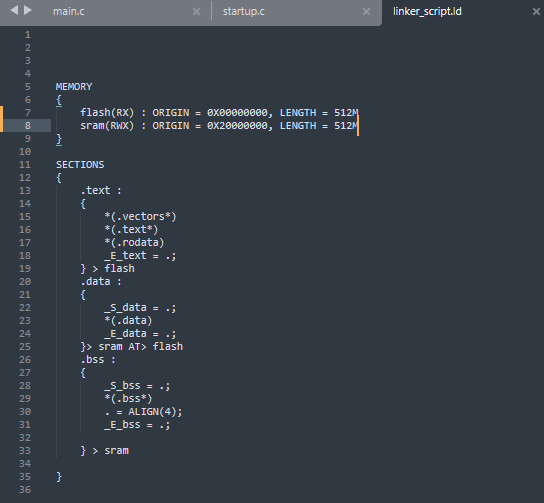
3.2- startup with c :



I know startup should be written in assembly to set stack pointer and branch label to it then branch label to main but in cortex m4 stack is labeled when power is applied to MCU the (pc) value will be 0 which mapped to (0x00000000) and will start at the same address which point to stack .

In this startup.c , I defined an array of pointers to function which holds every handlers and entry (sP) according to (IVT),and set size of stack to be 1024 byte as I defined it to uninsilzed array to be in .bss ,and put it in vectors section, and I defined handlers to be weak and alias to override in user code and cause declaration to be emitted for another symbol, and I copy data from rom to ram and initialize bss in ram then jump to main.

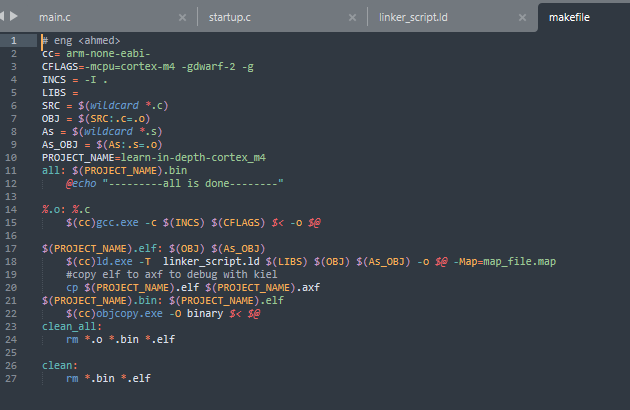
3.3- linker script

 :

In this linker script file I defined two memory flash and sram with its addresses and lengths, then I made sections like (.text)

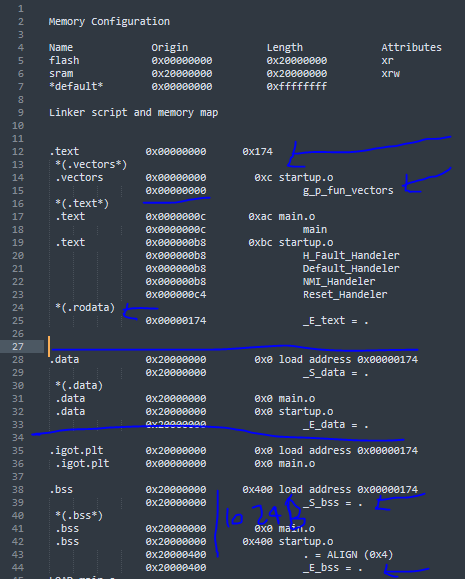
Which contain (. vectors,.text,rodata) and mapped it flash ,and then made (.data) section of all initialized data and (.bss) for all uninitialized data , and I made a locator counter to count addresses to end of .bss which I reference to top of stack in starting of .bss .

3.4 – make file :

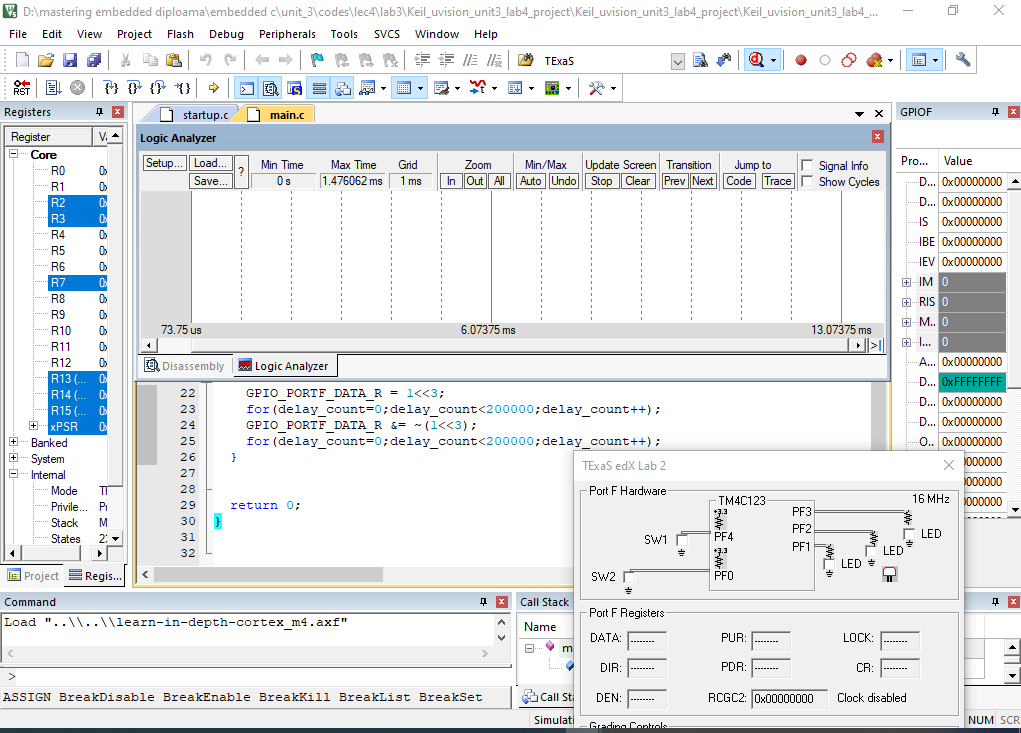


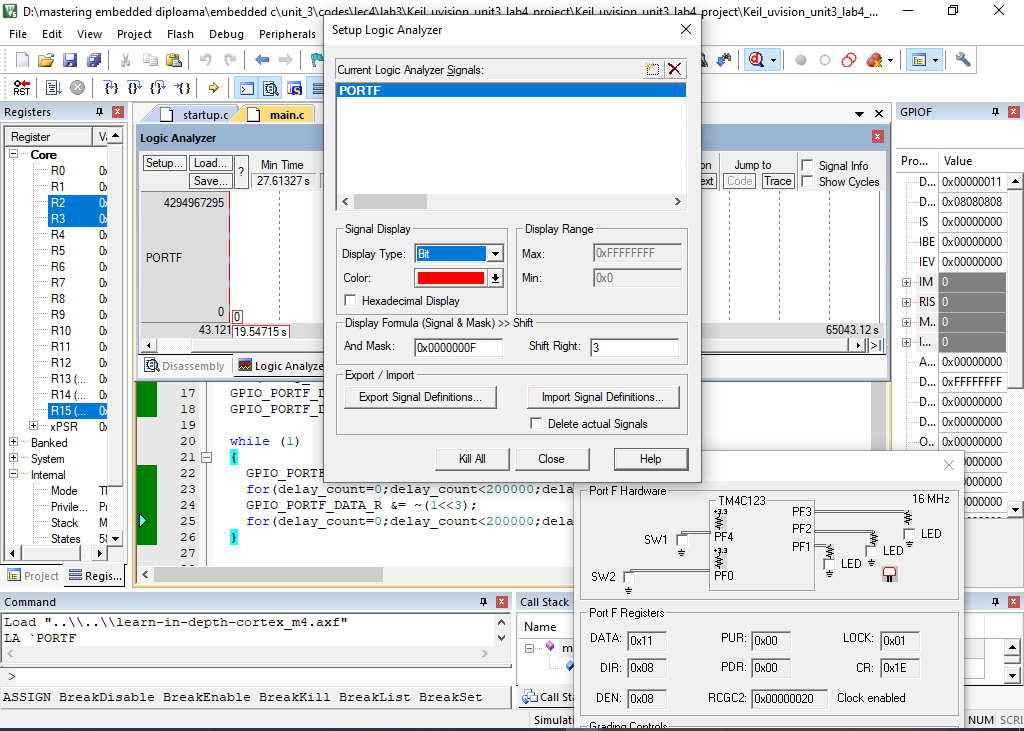
This make file is optimize compiling the program , so I used some make feature to do it like simplifaction dry and wildcards.

4-map file



5- run application on uvision keil tool with feature from edx course (shape the world).





6- Debug application on uvision keil tool with feature from edx course (shape the world).

