Table of contents

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)

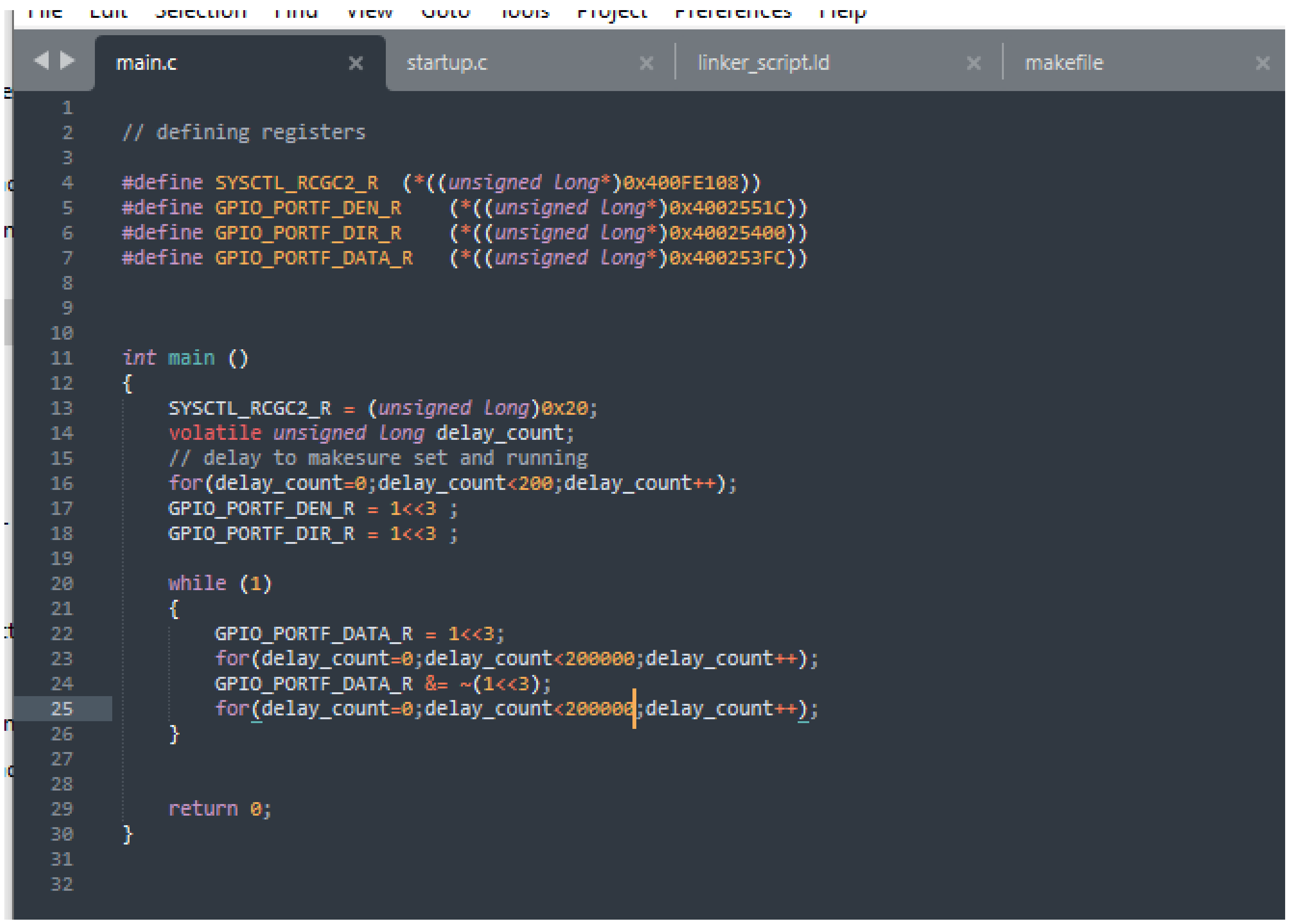
- to enable GPIOF (0x4002551c)set to 1 bit 3

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

- to write GPIOF (0x400253fc)

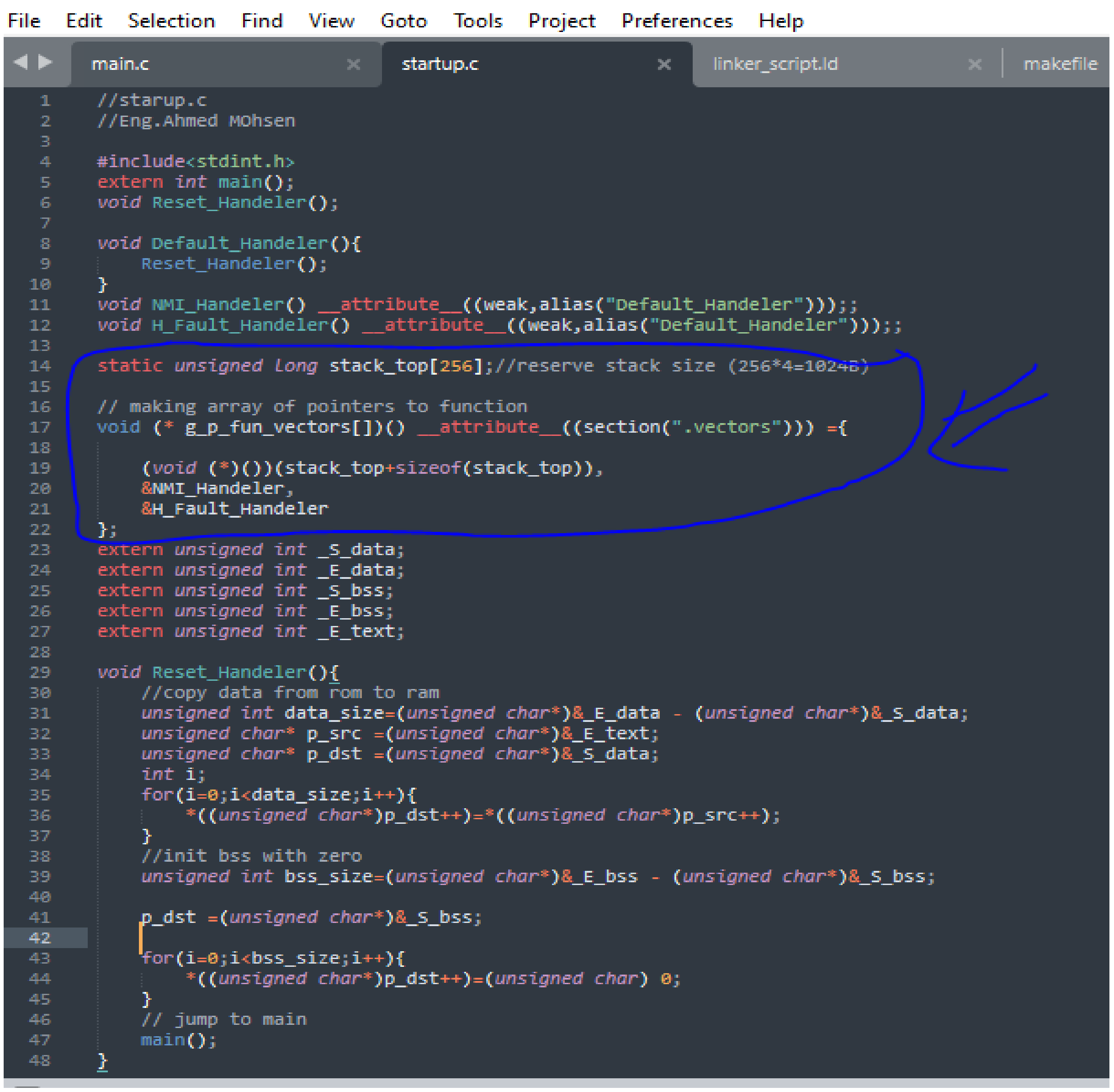
3- Source code

3.1 – main application code



```c

// defining registers

#define SYSCTL_RCGC2_R  (*((unsigned Long*)0x400FE108))
#define GPIO_PORTF_DEN_R    (*((unsigned Long*)0x4002551C))
#define GPIO_PORTF_DIR_R    (*((unsigned Long*)0x40025400))
#define GPIO_PORTF_DATA_R   (*((unsigned Long*)0x400253FC))



int main ()
{
    SYSCTL_RCGC2_R = (unsigned Long)0x20;
    volatile unsigned Long delay_count;
    // delay to makesure set and running
    for(delay_count=0;delay_count<200;delay_count++);
    GPIO_PORTF_DEN_R = 1<<3 ;
    GPIO_PORTF_DIR_R = 1<<3 ;

    while (1)
    {
        GPIO_PORTF_DATA_R = 1<<3;
        for(delay_count=0;delay_count<200000;delay_count++);
        GPIO_PORTF_DATA_R &= ~(1<<3);
        for(delay_count=0;delay_count<200000;delay_count++);
    }


    return 0;
}


```

3.2- startup with c :

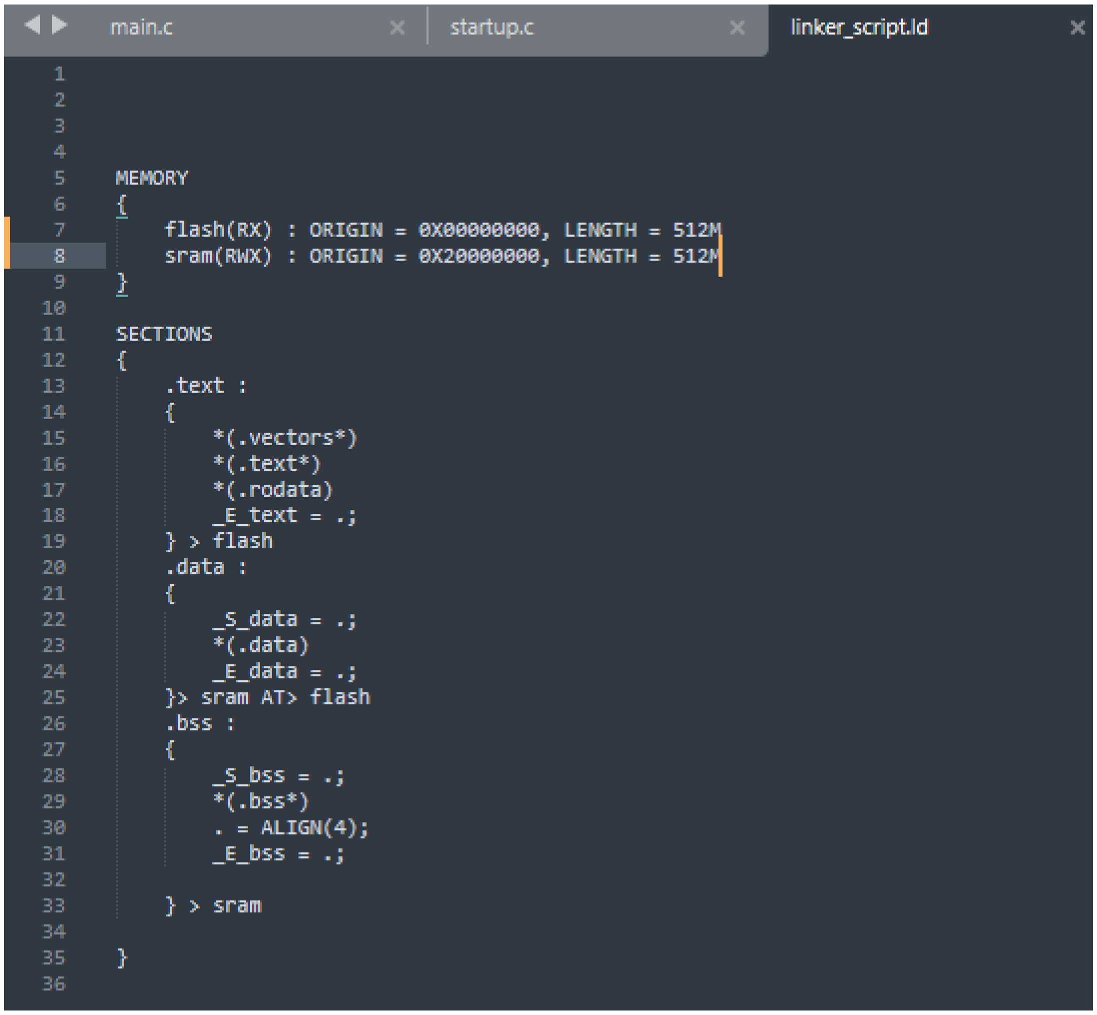

File Edit Selection Find View Goto Tools Project Preferences Help
main.c
startup.c
linker_script.ld
makefile

```c
//starup.c
//Eng.Ahmed MOhsen

#include<stdint.h>
extern int main();
void Reset_Handeler();

void Default_Handeler(){
    Reset_Handeler();
}
void NMI_Handeler() __attribute__((weak,alias("Default_Handeler")));;
void H_Fault_Handeler() __attribute__((weak,alias("Default_Handeler")));;

static unsigned long stack_top[256];//reserve stack size (256*4=1024B)

// making array of pointers to function
void (* g_p_fun_vectors[])() __attribute__((section(".vectors"))) ={

    (void (*)())(stack_top+sizeof(stack_top)),
    &NMI_Handeler,
    &H_Fault_Handeler
};
extern unsigned int _S_data;
extern unsigned int _E_data;
extern unsigned int _S_bss;
extern unsigned int _E_bss;
extern unsigned int _E_text;

void Reset_Handeler(){
    //copy data from rom to ram
    unsigned int data_size=(unsigned char*)&_E_data - (unsigned char*)&_S_data;
    unsigned char* p_src =(unsigned char*)&_E_text;
    unsigned char* p_dst =(unsigned char*)&_S_data;
    int i;
    for(i=0;i<data_size;i++){
        *((unsigned char*)p_dst++)=*((unsigned char*)p_src++);
    }
    //init bss with zero
    unsigned int bss_size=(unsigned char*)&_E_bss - (unsigned char*)&_S_bss;

    p_dst =(unsigned char*)&_S_bss;

    for(i=0;i<bss_size;i++){
        *((unsigned char*)p_dst++)=(unsigned char) 0;
    }
    // jump to main
    main();
}
```

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



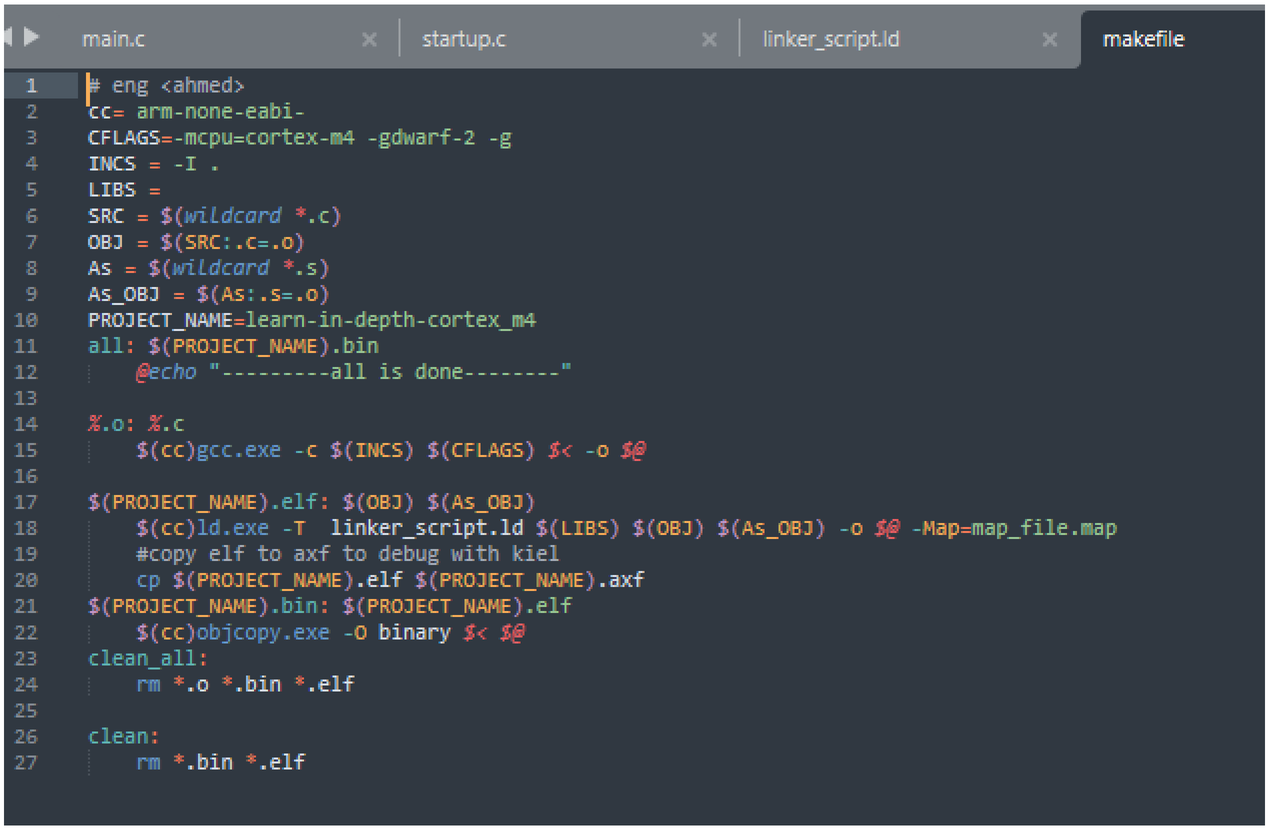
```



MEMORY
{
    flash(RX) : ORIGIN = 0X00000000, LENGTH = 512M
    sram(RWX) : ORIGIN = 0X20000000, LENGTH = 512M
}

SECTIONS
{
    .text :
    {
        *(.vectors*)
        *(.text*)
        *(.rodata)
        _E_text = .;
    } > flash
    .data :
    {
        _S_data = .;
        *(.data)
        _E_data = .;
    }> sram AT> flash
    .bss :
    {
        _S_bss = .;
        *(.bss*)
        . = ALIGN(4);
        _E_bss = .;

    } > sram

}

```

:

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 .

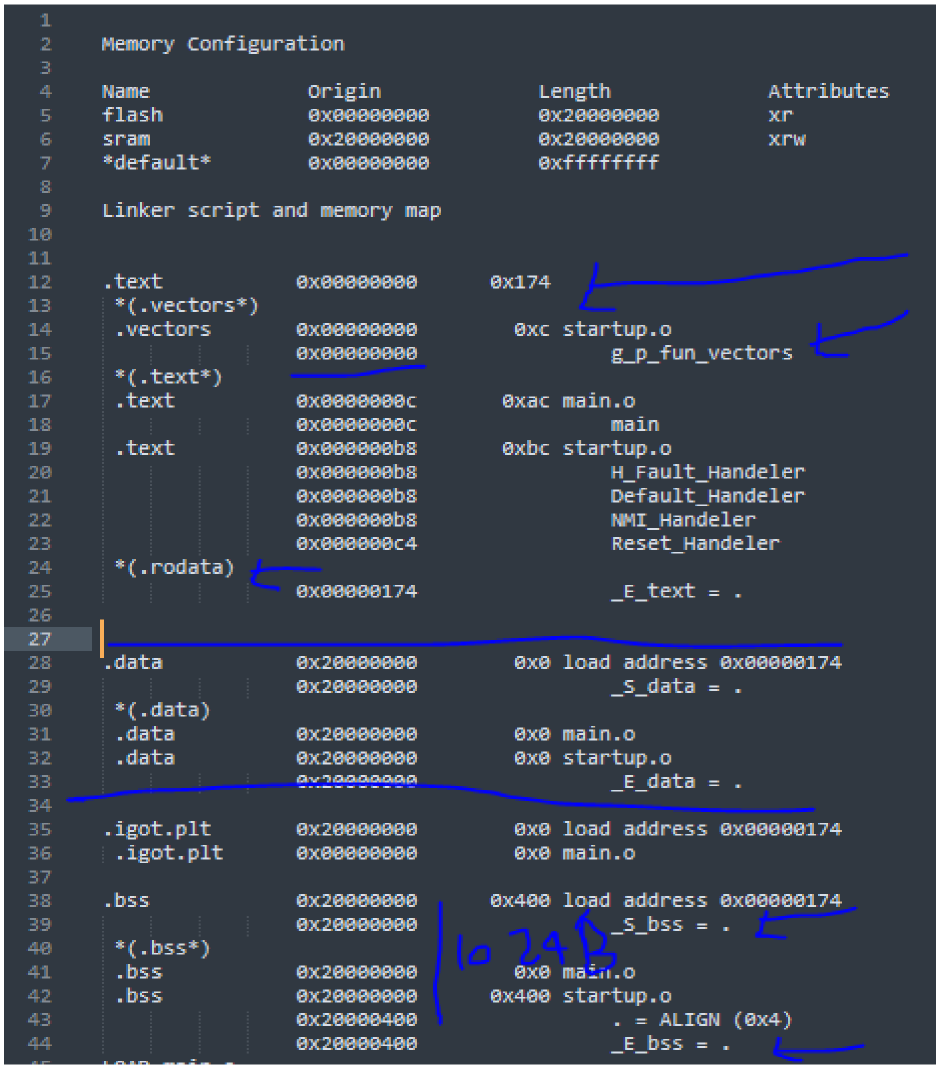
3.4 – make file :



```makefile
# eng <ahmed>
cc= arm-none-eabi-
CFLAGS=-mcpu=cortex-m4 -gdwarf-2 -g
INCS = -I .
LIBS =
SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)
As = $(wildcard *.s)
As_OBJ = $(As:.s=.o)
PROJECT_NAME=learn-in-depth-cortex_m4
all: $(PROJECT_NAME).bin
    @echo "---------all is done--------"

%.o: %.c
    $(cc)gcc.exe -c $(INCS) $(CFLAGS) $< -o $@

$(PROJECT_NAME).elf: $(OBJ) $(As_OBJ)
    $(cc)ld.exe -T  linker_script.ld $(LIBS) $(OBJ) $(As_OBJ) -o $@ -Map=map_file.map
    #copy elf to axf to debug with kiel
    cp $(PROJECT_NAME).elf $(PROJECT_NAME).axf
$(PROJECT_NAME).bin: $(PROJECT_NAME).elf
    $(cc)objcopy.exe -O binary $< $@
clean_all:
    rm *.o *.bin *.elf

clean:
    rm *.bin *.elf
```

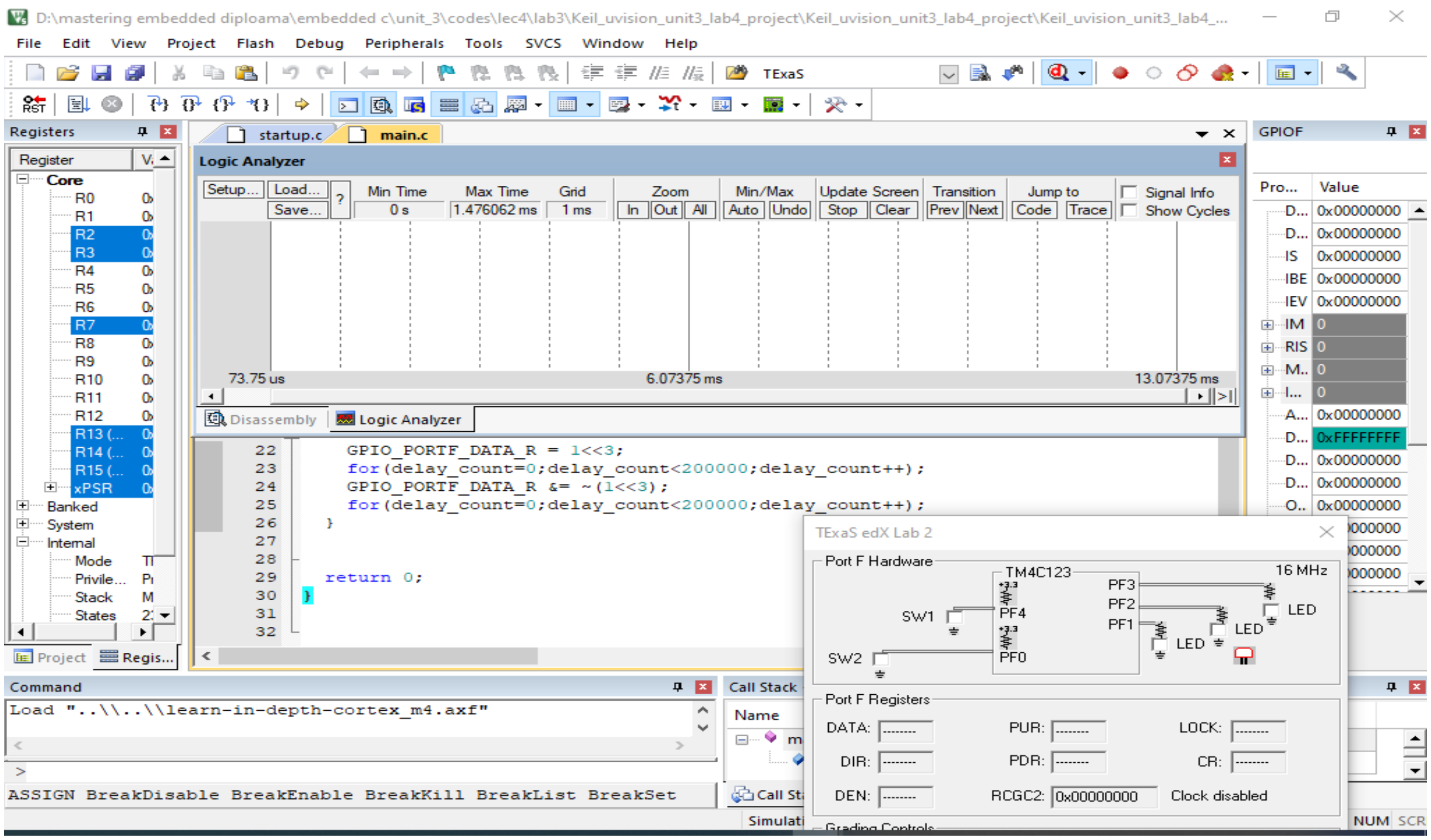
This make file is optimize compiling the program , so I used some make feature to do it like simplifaction dry and wildcards.

4-map file



```

Memory Configuration

Name             Origin             Length             Attributes
flash            0x00000000         0x20000000         xr
sram             0x20000000         0x20000000         xrw
*default*        0x00000000         0xffffffff

Linker script and memory map


.text           0x00000000      0x174
 *(.vectors*)
 .vectors       0x00000000        0xc startup.o
                0x00000000                g_p_fun_vectors
 *(.text*)
 .text          0x0000000c       0xac main.o
                0x0000000c                main
 .text          0x000000b8       0xbc startup.o
                0x000000b8                H_Fault_Handeler
                0x000000b8                Default_Handeler
                0x000000b8                NMI_Handeler
                0x000000c4                Reset_Handeler
 *(.rodata)
                0x00000174                _E_text = .


.data           0x20000000        0x0 load address 0x00000174
                0x20000000                _S_data = .
 *(.data)
 .data          0x20000000        0x0 main.o
 .data          0x20000000        0x0 startup.o
                0x20000000                _E_data = .

.igot.plt       0x20000000        0x0 load address 0x00000174
 .igot.plt      0x00000000        0x0 main.o

.bss            0x20000000      0x400 load address 0x00000174
                0x20000000                _S_bss = .
 *(.bss*)
 .bss           0x20000000        0x0 main.o
 .bss           0x20000000      0x400 startup.o
                0x20000400                . = ALIGN (0x4)
                0x20000400                _E_bss = .
```

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



D:\mastering embedded diploama\embedded c\unit_3\codes\lec4\lab3\Keil_uvision_unit3_lab4_project\Keil_uvision_unit3_lab4_project\Keil_uvision_unit3_lab4_...

```
	GPIO_PORTF_DATA_R = 1<<3;
	for(delay_count=0;delay_count<200000;delay_count++);
	GPIO_PORTF_DATA_R &= ~(1<<3);
	for(delay_count=0;delay_count<200000;delay_count++);
}

return 0;
}
```

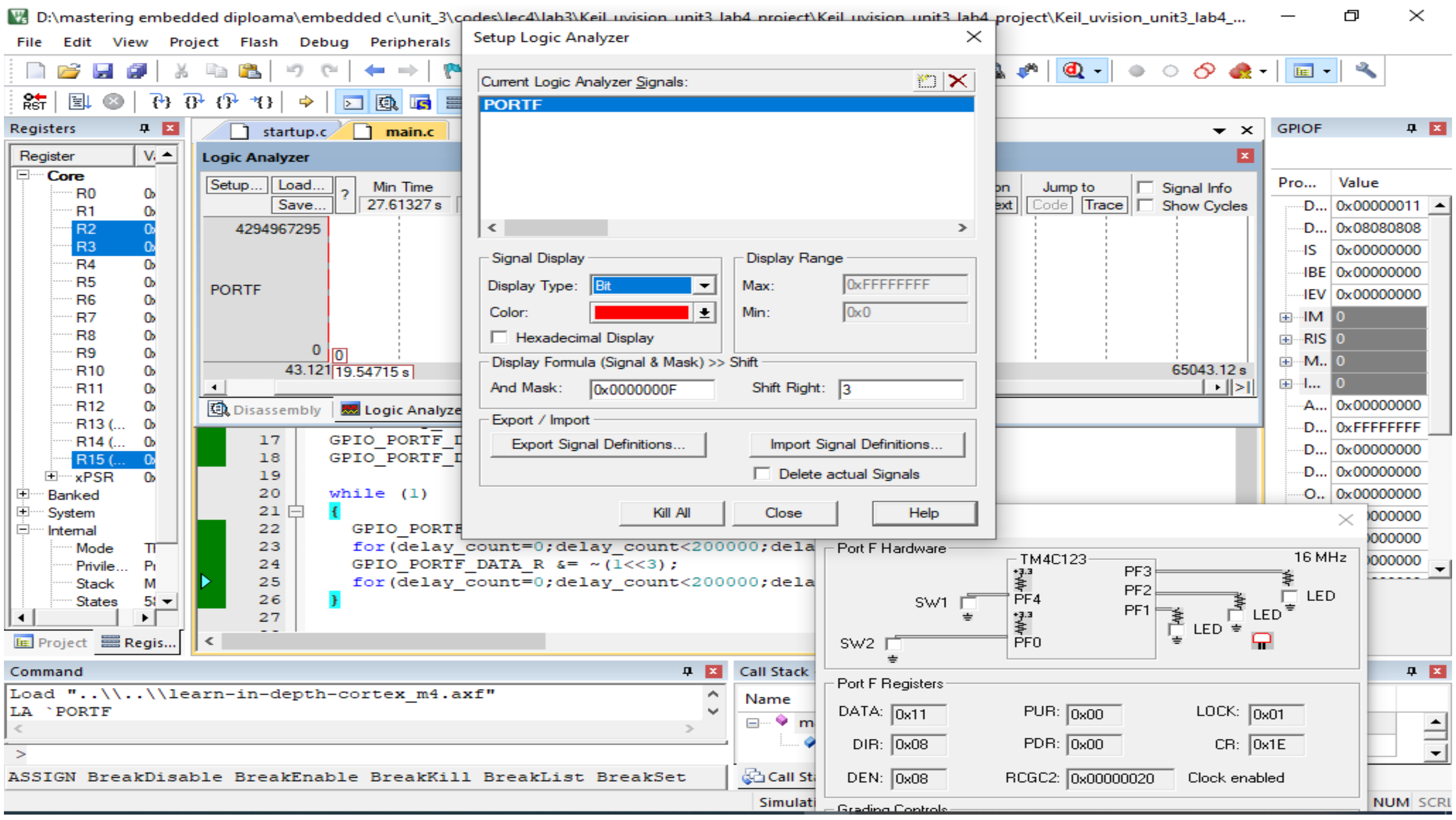
```
Load "..\\..\\learn-in-depth-cortex_m4.axf"
```

```
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
```

Logic Analyzer: Min Time 0 s, Max Time 1.476062 ms, Grid 1 ms; 73.75 us, 6.07375 ms, 13.07375 ms

TExaS edX Lab 2 – Port F Registers:

DATA:	--------	PUR:	--------	LOCK:	--------
DIR:	--------	PDR:	--------	CR:	--------
DEN:	--------	RCGC2:	0x00000000	Clock disabled	



D:\mastering embedded diploama\embedded c\unit_3\codes\lec4\lab3\Keil_uvision_unit3_lab4_project\Keil_uvision_unit3_lab4_project\Keil_uvision_unit3_lab4_...

Setup Logic Analyzer

Current Logic Analyzer Signals: PORTF

Signal Display: Display Type: Bit; Color: 🟥; Hexadecimal Display: ☐

Display Range: Max: 0xFFFFFFFF; Min: 0x0

Display Formula (Signal & Mask) >> Shift: And Mask: 0x0000000F; Shift Right: 3

```
	GPIO_PORTF_D
	GPIO_PORTF_D

	while (1)
	{
		GPIO_PORTE
		for(delay_count=0;delay_count<200000;dela
		GPIO_PORTF_DATA_R &= ~(1<<3);
		for(delay_count=0;delay_count<200000;dela
	}
```

```
Load "..\\..\\learn-in-depth-cortex_m4.axf"
LA `PORTF
```

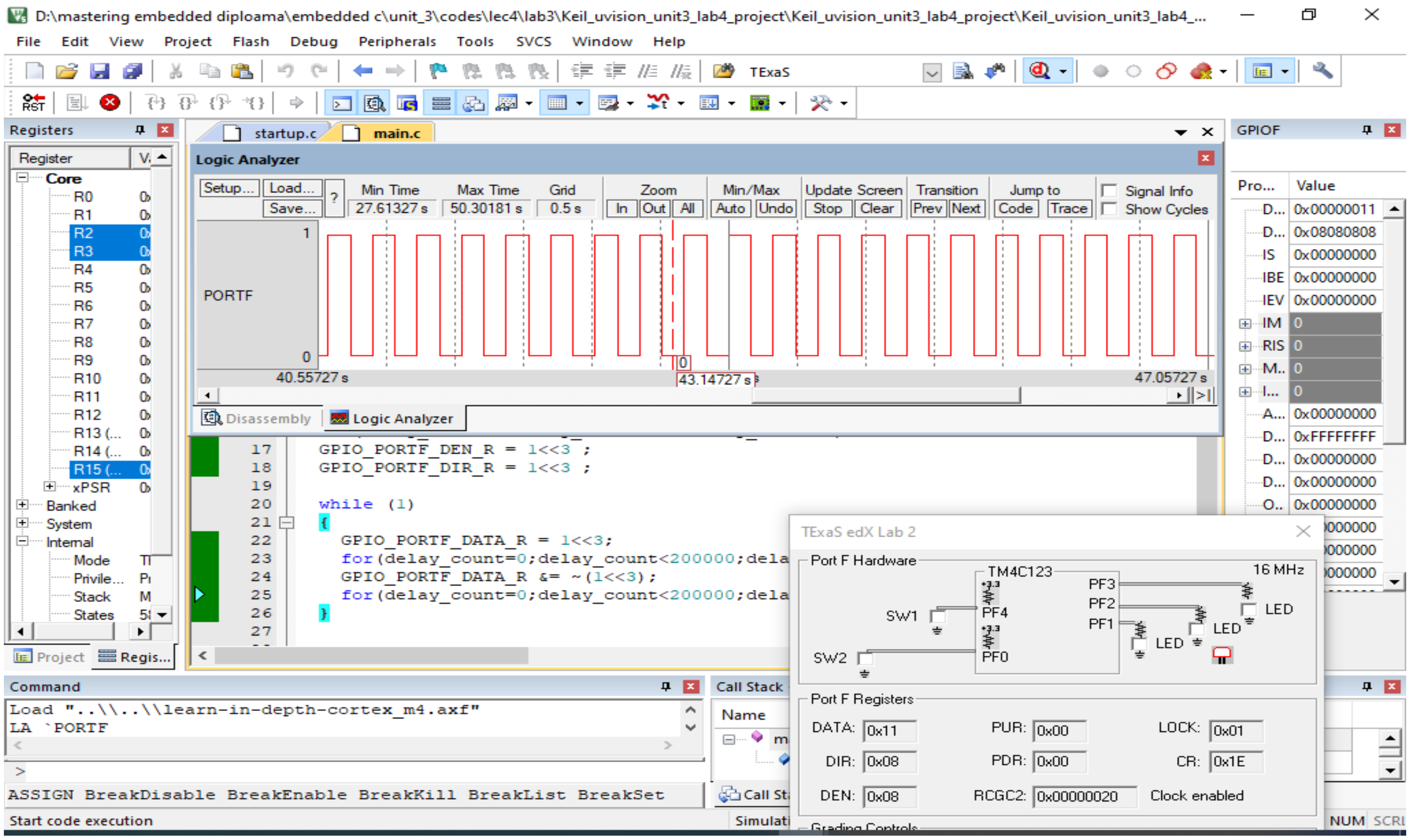
```
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
```

Logic Analyzer: Min Time 27.61327 s; 4294967295; PORTF; 0; 43.121; 19.54715 s; 65043.12 s

Port F Registers:

DATA:	0x11	PUR:	0x00	LOCK:	0x01
DIR:	0x08	PDR:	0x00	CR:	0x1E
DEN:	0x08	RCGC2:	0x00000020	Clock enabled	

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


D:\mastering embedded diploama\embedded c\unit_3\codes\lec4\lab3\Keil_uvision_unit3_lab4_project\Keil_uvision_unit3_lab4_project\Keil_uvision_unit3_lab4_...

Logic Analyzer — Min Time: 27.61327 s, Max Time: 50.30181 s, Grid: 0.5 s

```
GPIO_PORTF_DEN_R = 1<<3 ;
GPIO_PORTF_DIR_R = 1<<3 ;

while (1)
{
  GPIO_PORTF_DATA_R = 1<<3;
  for(delay_count=0;delay_count<200000;dela
  GPIO_PORTF_DATA_R &= ~(1<<3);
  for(delay_count=0;delay_count<200000;dela
}
```

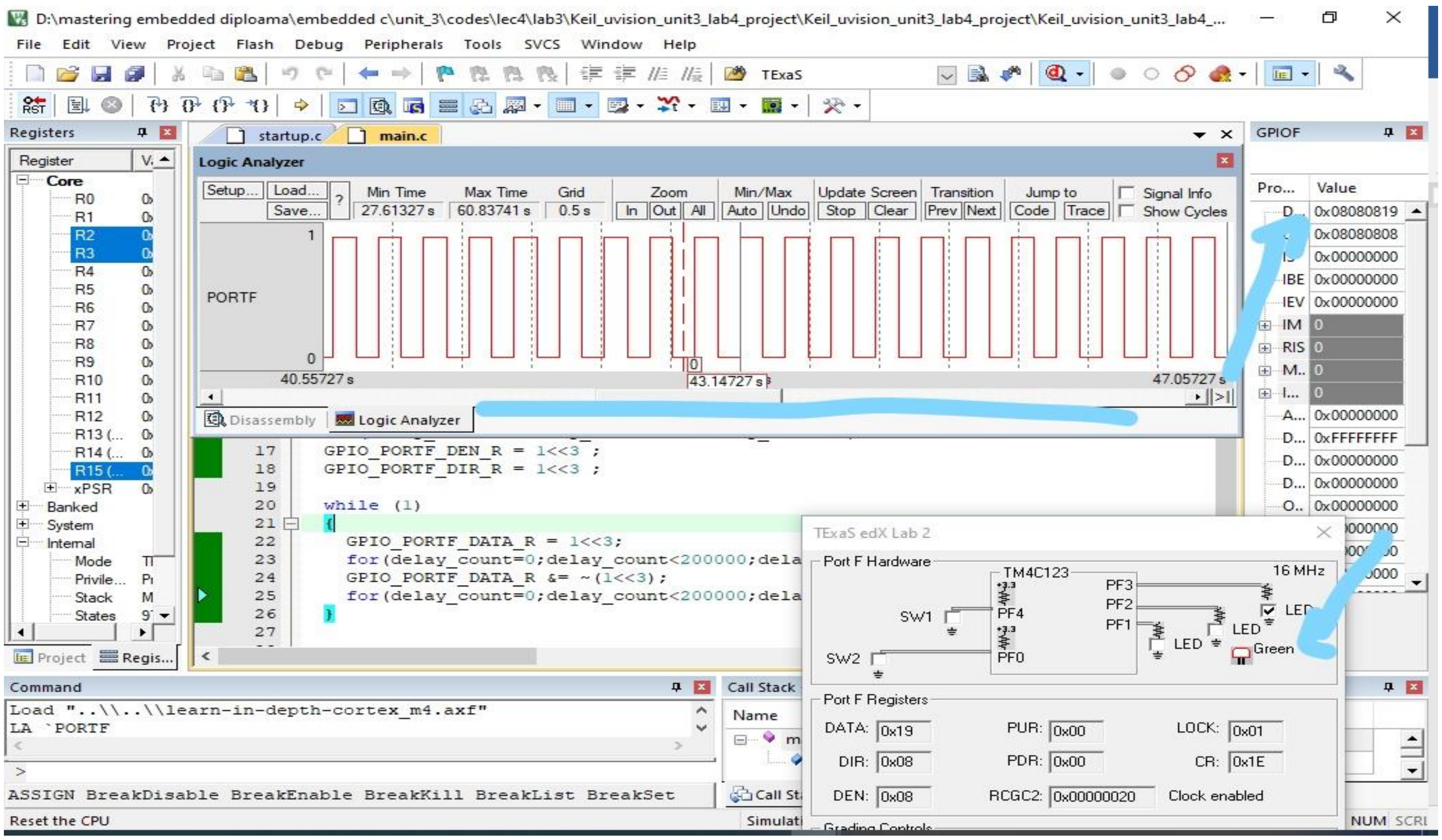
```
Load "..\\..\\learn-in-depth-cortex_m4.axf"
LA `PORTF
```

```
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
Start code execution
```

Pro...	Value
D...	0x00000011
D...	0x08080808
IS	0x00000000
IBE	0x00000000
IEV	0x00000000
IM	0
RIS	0
M...	0
I...	0
A...	0x00000000
D...	0xFFFFFFFF
D...	0x00000000
D...	0x00000000
O...	0x00000000

TExaS edX Lab 2 — Port F Registers

DATA:	0x11	PUR:	0x00	LOCK:	0x01
DIR:	0x08	PDR:	0x00	CR:	0x1E
DEN:	0x08	RCGC2:	0x00000020	Clock enabled	


D:\mastering embedded diploama\embedded c\unit_3\codes\lec4\lab3\Keil_uvision_unit3_lab4_project\Keil_uvision_unit3_lab4_project\Keil_uvision_unit3_lab4_...

Logic Analyzer — Min Time: 27.61327 s, Max Time: 60.83741 s, Grid: 0.5 s

```
GPIO_PORTF_DEN_R = 1<<3 ;
GPIO_PORTF_DIR_R = 1<<3 ;

while (1)
{
  GPIO_PORTF_DATA_R = 1<<3;
  for(delay_count=0;delay_count<200000;dela
  GPIO_PORTF_DATA_R &= ~(1<<3);
  for(delay_count=0;delay_count<200000;dela
}
```

```
Load "..\\..\\learn-in-depth-cortex_m4.axf"
LA `PORTF
```

```
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
Reset the CPU
```

Pro...	Value
D...	0x08080819
	0x08080808
I...	0x00000000
IBE	0x00000000
IEV	0x00000000
IM	0
RIS	0
M...	0
I...	0
A...	0x00000000
D...	0xFFFFFFFF
D...	0x00000000
D...	0x00000000
O...	0x00000000

TExaS edX Lab 2 — Port F Registers

DATA:	0x19	PUR:	0x00	LOCK:	0x01
DIR:	0x08	PDR:	0x00	CR:	0x1E
DEN:	0x08	RCGC2:	0x00000020	Clock enabled	