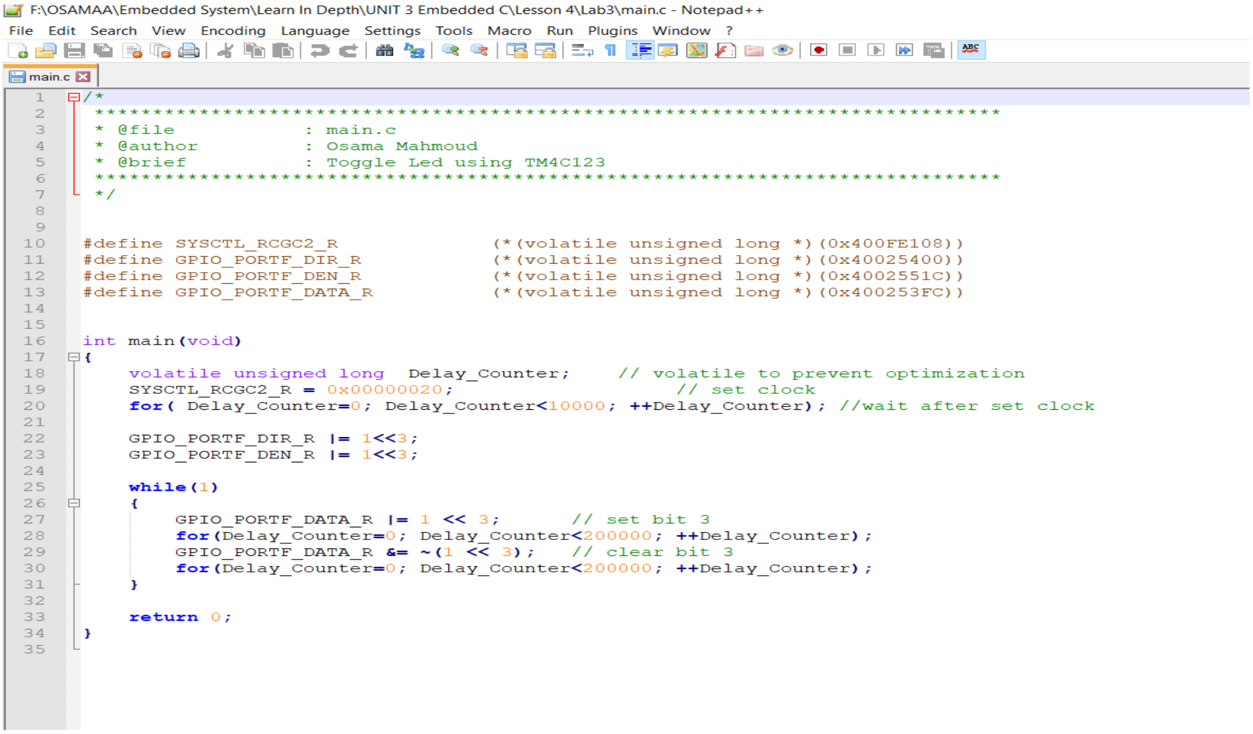
Lab3

In this lab, I will show how to make a startup file and make file for another microcontroller (TM4C123).

Main.c:

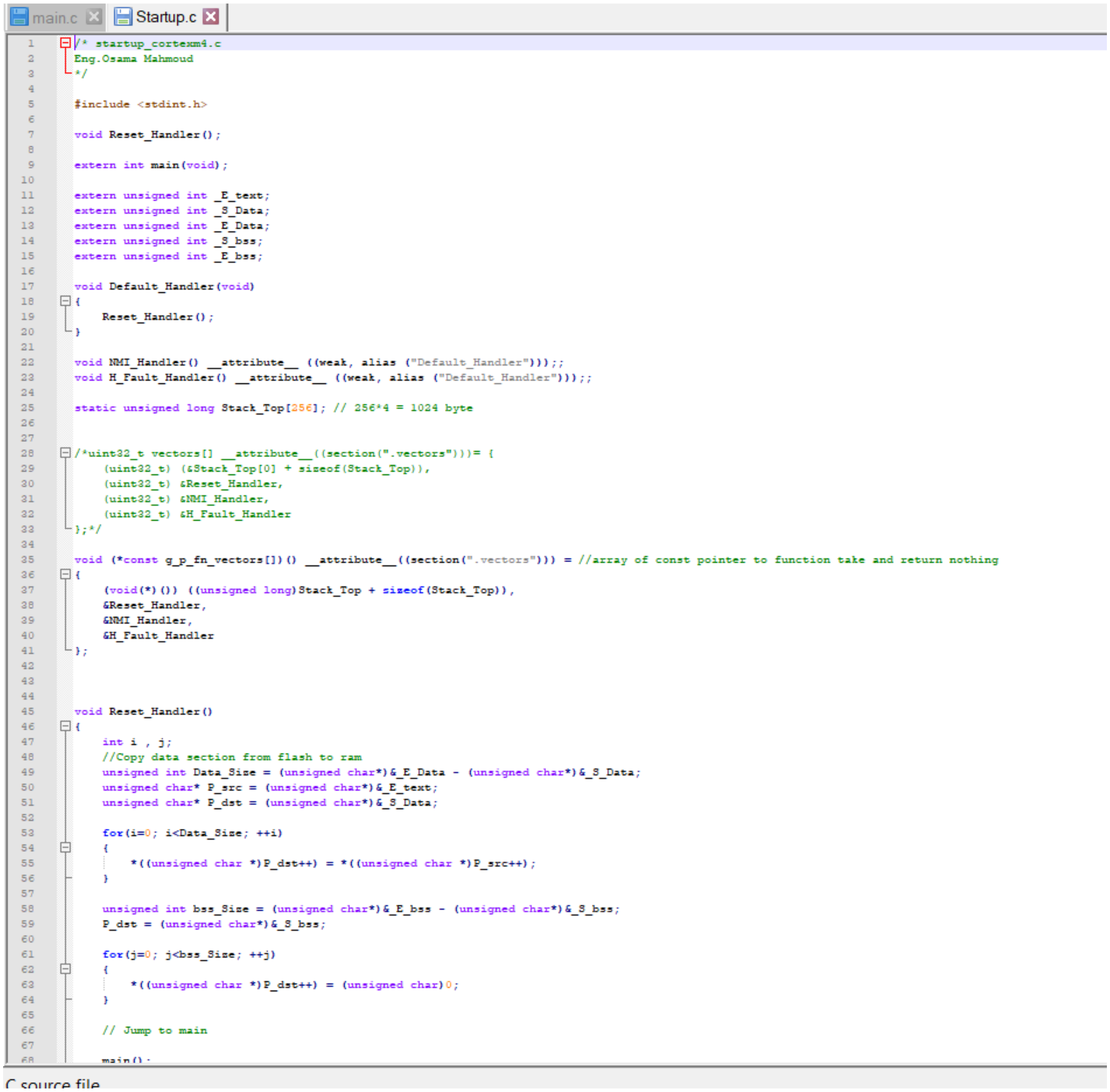
I use a TM4C123 cortexM4 to toggle the pin3 on PORTF


F:\OSAMAA\Embedded System\Learn In Depth\UNIT 3 Embedded C\Lesson 4\Lab3\main.c - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
main.c

```c
/*
 ******************************************************************************
 * @file            : main.c
 * @author          : Osama Mahmoud
 * @brief           : Toggle Led using TM4C123
 ******************************************************************************
 */


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


int main(void)
{
    volatile unsigned long  Delay_Counter;     // volatile to prevent optimization
    SYSCTL_RCGC2_R = 0x00000020;                       // set clock
    for( Delay_Counter=0; Delay_Counter<10000; ++Delay_Counter); //wait after set clock

    GPIO_PORTF_DIR_R |= 1<<3;
    GPIO_PORTF_DEN_R |= 1<<3;

    while(1)
    {
        GPIO_PORTF_DATA_R |= 1 << 3;       // set bit 3
        for(Delay_Counter=0; Delay_Counter<200000; ++Delay_Counter);
        GPIO_PORTF_DATA_R &= ~(1 << 3);    // clear bit 3
        for(Delay_Counter=0; Delay_Counter<200000; ++Delay_Counter);
    }

    return 0;
}

```

Startup

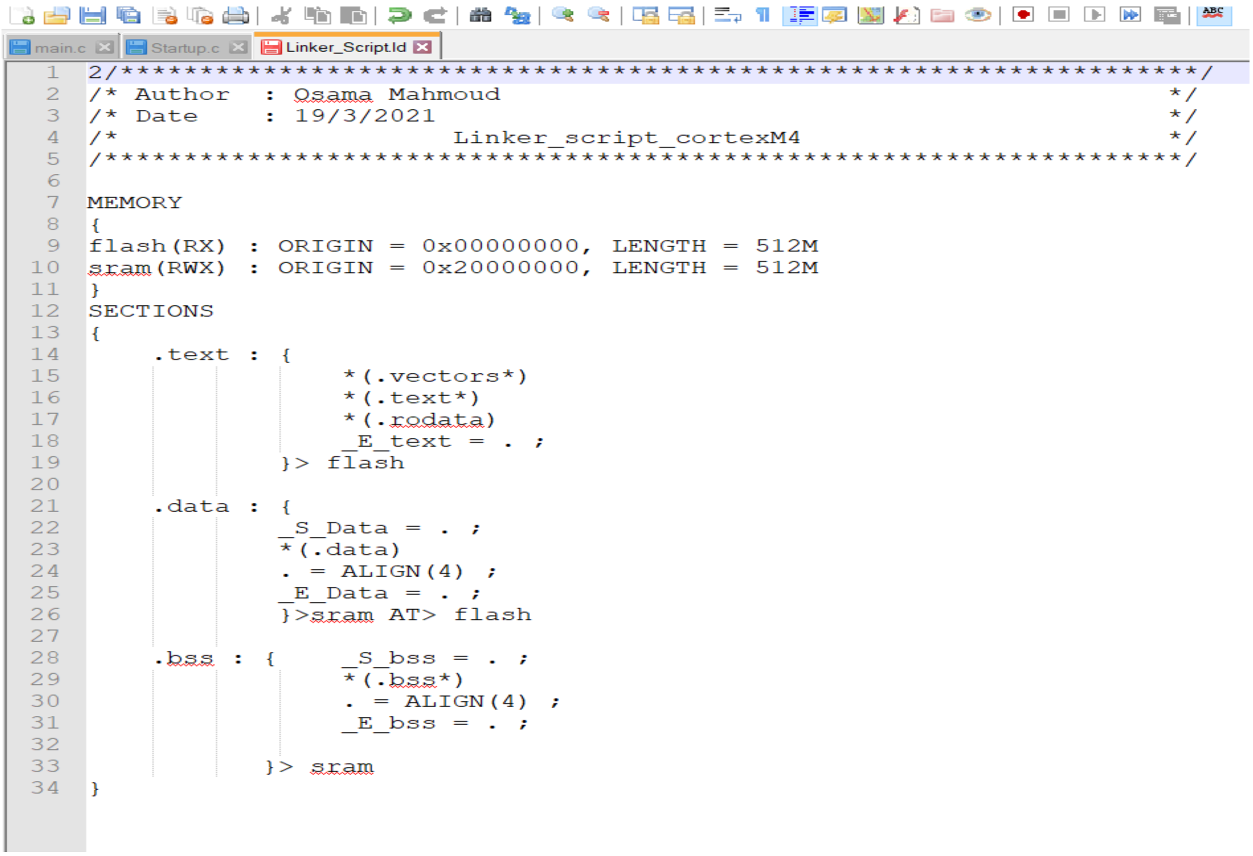
- Stack top: will defined it automatically on the bss section.
- The vector array will be defined by array of constant pointer to function takes no thing and return void



```c
/* startup_cortexm4.c
Eng.Osama Mahmoud
*/

#include <stdint.h>

void Reset_Handler();

extern int main(void);

extern unsigned int _E_text;
extern unsigned int _S_Data;
extern unsigned int _E_Data;
extern unsigned int _S_bss;
extern unsigned int _E_bss;

void Default_Handler(void)
{
    Reset_Handler();
}

void NMI_Handler() __attribute__ ((weak, alias ("Default_Handler")));;
void H_Fault_Handler() __attribute__ ((weak, alias ("Default_Handler")));;

static unsigned long Stack_Top[256]; // 256*4 = 1024 byte


/*uint32_t vectors[] __attribute__((section(".vectors")))= {
    (uint32_t) (&Stack_Top[0] + sizeof(Stack_Top)),
    (uint32_t) &Reset_Handler,
    (uint32_t) &NMI_Handler,
    (uint32_t) &H_Fault_Handler
};*/

void (*const g_p_fn_vectors[])() __attribute__((section(".vectors"))) = //array of const pointer to function take and return nothing
{
    (void(*)()) ((unsigned long)Stack_Top + sizeof(Stack_Top)),
    &Reset_Handler,
    &NMI_Handler,
    &H_Fault_Handler
};



void Reset_Handler()
{
    int i , j;
    //Copy data section from flash 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;

    for(i=0; i<Data_Size; ++i)
    {
        *((unsigned char *)P_dst++) = *((unsigned char *)P_src++);
    }

    unsigned int bss_Size = (unsigned char*)&_E_bss - (unsigned char*)&_S_bss;
    P_dst = (unsigned char*)&_S_bss;

    for(j=0; j<bss_Size; ++j)
    {
        *((unsigned char *)P_dst++) = (unsigned char)0;
    }

    // Jump to main

    main();
```

Linker script

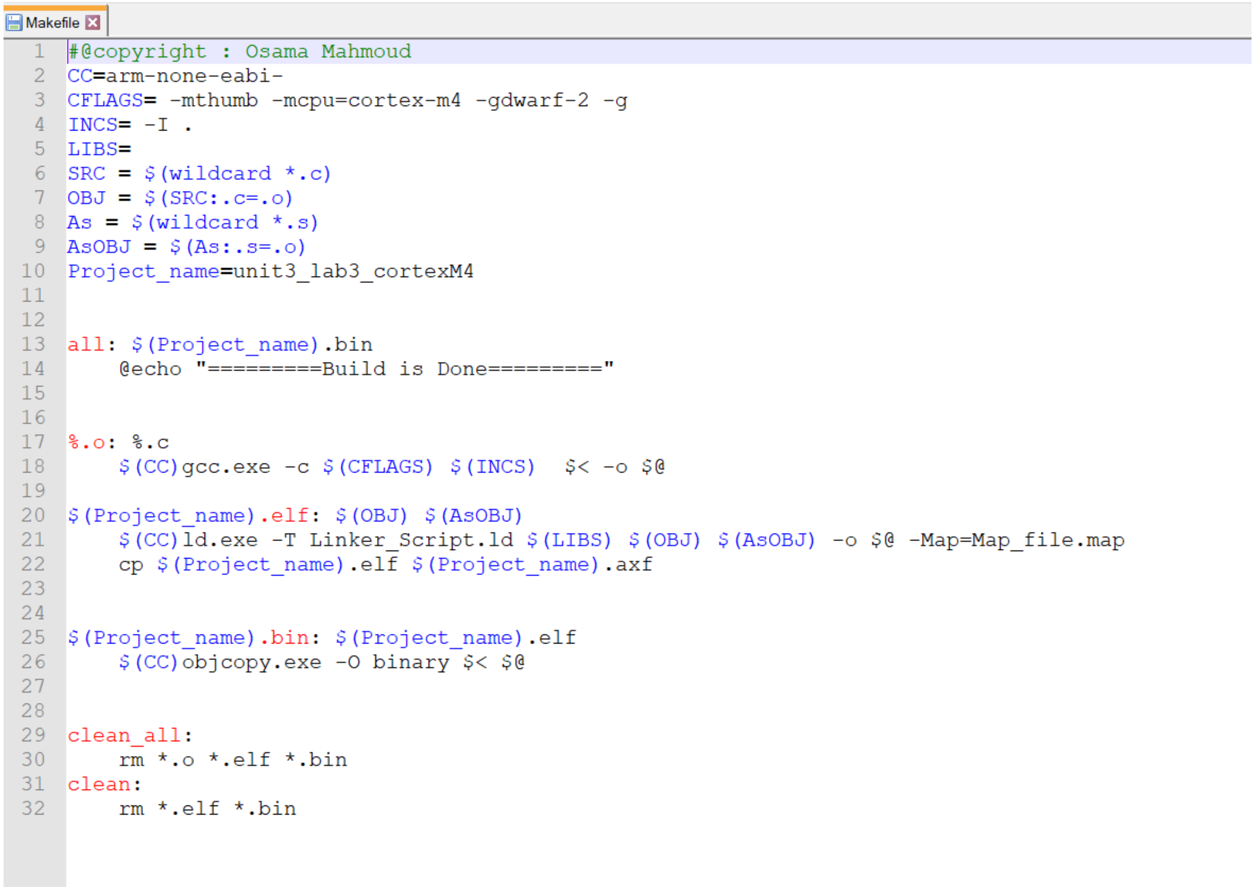
In this file, I show the memory layout and the size of the flash and ram and its addresses and that helps the linker to put the data in its right location through the running time.



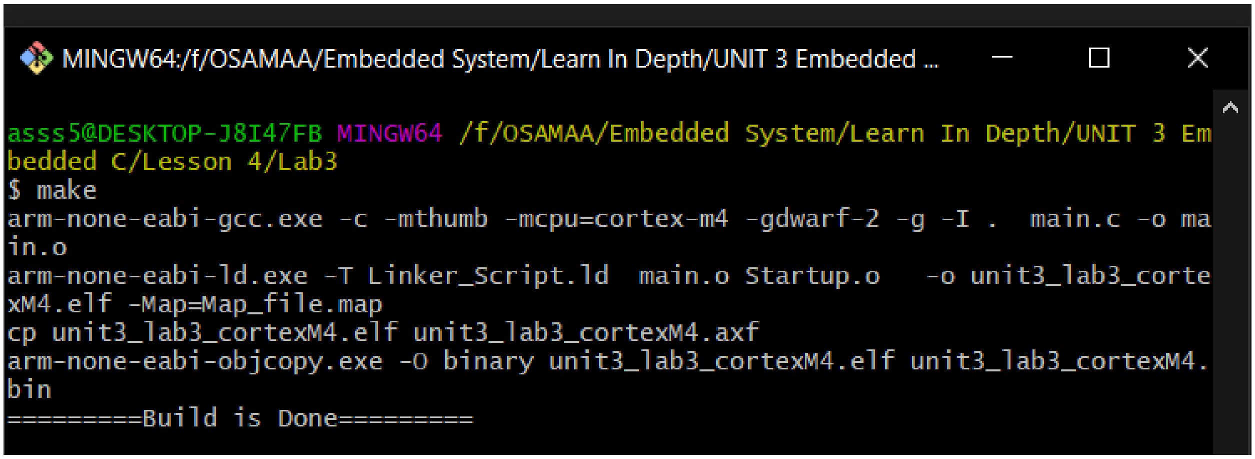
```
2/*********************************************************************/
/* Author   : Osama Mahmoud                                           */
/* Date     : 19/3/2021                                               */
/*                     Linker_script_cortexM4                         */
/*********************************************************************/

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)
            . = ALIGN(4) ;
            _E_Data = . ;
            }>sram AT> flash

    .bss : {    _S_bss = . ;
                *(.bss*)
                . = ALIGN(4) ;
                _E_bss = . ;

           }> sram
}
```

Makefile

To Automate the Building process. extract the object files from the source files then the executable file. In this lab we copy the elf extension to axf



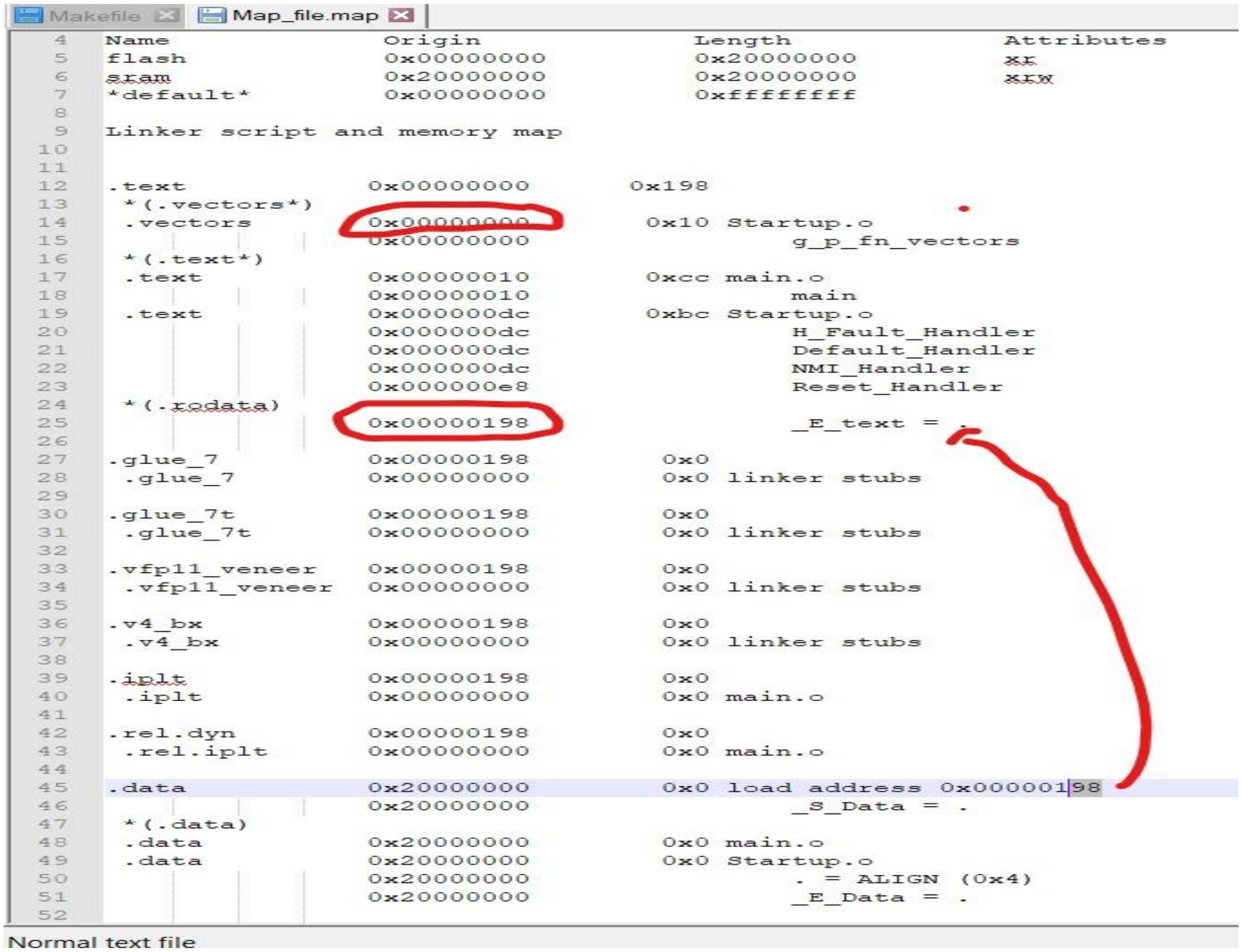
```
#@copyright : Osama Mahmoud
CC=arm-none-eabi-
CFLAGS= -mthumb -mcpu=cortex-m4 -gdwarf-2 -g
INCS= -I .
LIBS=
SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)
As = $(wildcard *.s)
AsOBJ = $(As:.s=.o)
Project_name=unit3_lab3_cortexM4


all: $(Project_name).bin
	@echo "=========Build is Done========="


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

$(Project_name).elf: $(OBJ) $(AsOBJ)
	$(CC)ld.exe -T Linker_Script.ld $(LIBS) $(OBJ) $(AsOBJ) -o $@ -Map=Map_file.map
	cp $(Project_name).elf $(Project_name).axf


$(Project_name).bin: $(Project_name).elf
	$(CC)objcopy.exe -O binary $< $@


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


MINGW64:/f/OSAMAA/Embedded System/Learn In Depth/UNIT 3 Embedded ...

```
asss5@DESKTOP-J8I47FB MINGW64 /f/OSAMAA/Embedded System/Learn In Depth/UNIT 3 Em
bedded C/Lesson 4/Lab3
$ make
arm-none-eabi-gcc.exe -c -mthumb -mcpu=cortex-m4 -gdwarf-2 -g -I .  main.c -o ma
in.o
arm-none-eabi-ld.exe -T Linker_Script.ld  main.o Startup.o   -o unit3_lab3_corte
xM4.elf -Map=Map_file.map
cp unit3_lab3_cortexM4.elf unit3_lab3_cortexM4.axf
arm-none-eabi-objcopy.exe -O binary unit3_lab3_cortexM4.elf unit3_lab3_cortexM4.
bin
=========Build is Done=========
```

Mapfile

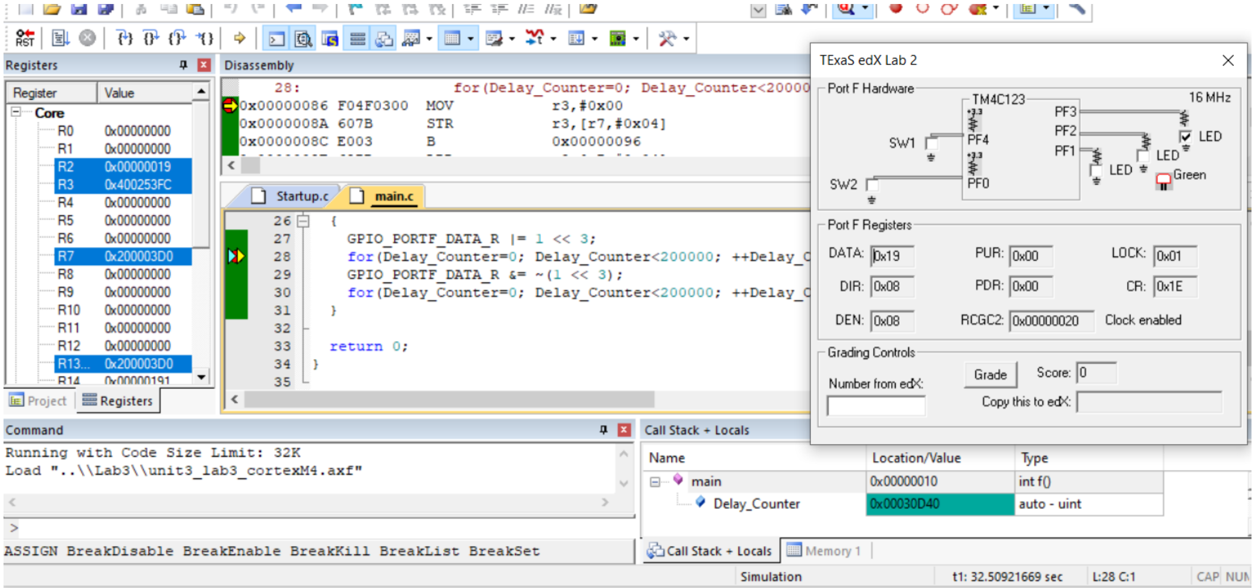
Review the locations and size of the data section and the main function and if there is an alignment or not



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

Linker script and memory map


.text           0x00000000      0x198
 *(.vectors*)
 .vectors       0x00000000       0x10 Startup.o
                0x00000000                g_p_fn_vectors
 *(.text*)
 .text          0x00000010       0xcc main.o
                0x00000010                main
 .text          0x000000dc       0xbc Startup.o
                0x000000dc                H_Fault_Handler
                0x000000dc                Default_Handler
                0x000000dc                NMI_Handler
                0x000000e8                Reset_Handler
 *(.rodata)
                0x00000198                _E_text = .

.glue_7         0x00000198        0x0
 .glue_7        0x00000000        0x0 linker stubs

.glue_7t        0x00000198        0x0
 .glue_7t       0x00000000        0x0 linker stubs

.vfp11_veneer   0x00000198        0x0
 .vfp11_veneer  0x00000000        0x0 linker stubs

.v4_bx          0x00000198        0x0
 .v4_bx         0x00000000        0x0 linker stubs

.iplt           0x00000198        0x0
 .iplt          0x00000000        0x0 main.o

.rel.dyn        0x00000198        0x0
 .rel.iplt      0x00000000        0x0 main.o

.data           0x20000000        0x0 load address 0x00000198
                0x20000000                _S_Data = .
 *(.data)
 .data          0x20000000        0x0 main.o
 .data          0x20000000        0x0 Startup.o
                0x20000000                . = ALIGN (0x4)
                0x20000000                _E_Data = .
```

Output

The green led is toggled



Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000019
R3	0x400253FC
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x200003D0
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13...	0x200003D0
R14	0x00000191

```
     28:                    for(Delay_Counter=0; Delay_Counter<20000
0x00000086 F04F0300  MOV          r3,#0x00
0x0000008A 607B      STR          r3,[r7,#0x04]
0x0000008C E003      B            0x00000096
```

```
{
  GPIO_PORTF_DATA_R |= 1 << 3;
  for(Delay_Counter=0; Delay_Counter<200000; ++Delay_C
  GPIO_PORTF_DATA_R &= ~(1 << 3);
  for(Delay_Counter=0; Delay_Counter<200000; ++Delay_C
}

return 0;
}
```

TExaS edX Lab 2

Port F Hardware: TM4C123, 16 MHz, ☑ LED Green

Port F Registers

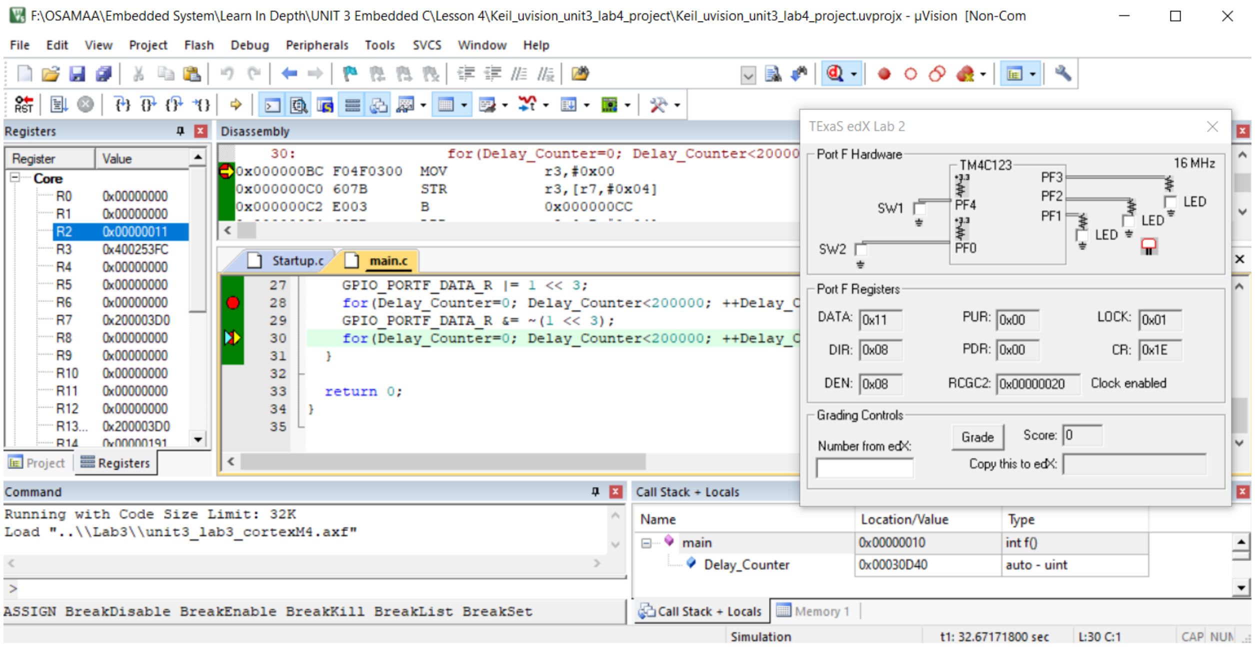
DATA:	0x19	PUR:	0x00	LOCK:	0x01
DIR:	0x08	PDR:	0x00	CR:	0x1E
DEN:	0x08	RCGC2:	0x00000020	Clock enabled	

Grading Controls: Score: 0

```
Running with Code Size Limit: 32K
Load "..\\Lab3\\unit3_lab3_cortexM4.axf"
```

Name	Location/Value	Type
main	0x00000010	int f()
Delay_Counter	0x00030D40	auto - uint

Simulation t1: 32.50921669 sec L:28 C:1



F:\OSAMAA\Embedded System\Learn In Depth\UNIT 3 Embedded C\Lesson 4\Keil_uvision_unit3_lab4_project\Keil_uvision_unit3_lab4_project.uvprojx - µVision [Non-Com

Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000011
R3	0x400253FC
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x200003D0
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13...	0x200003D0
R14	0x00000191

```
     30:                    for(Delay_Counter=0; Delay_Counter<20000
0x000000BC F04F0300  MOV          r3,#0x00
0x000000C0 607B      STR          r3,[r7,#0x04]
0x000000C2 E003      B            0x000000CC
```

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

return 0;
}
```

TExaS edX Lab 2

Port F Hardware: TM4C123, 16 MHz, □ LED

Port F Registers

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

Grading Controls: Score: 0

```
Running with Code Size Limit: 32K
Load "..\\Lab3\\unit3_lab3_cortexM4.axf"
```

Name	Location/Value	Type
main	0x00000010	int f()
Delay_Counter	0x00030D40	auto - uint

Simulation t1: 32.67171800 sec L:30 C:1