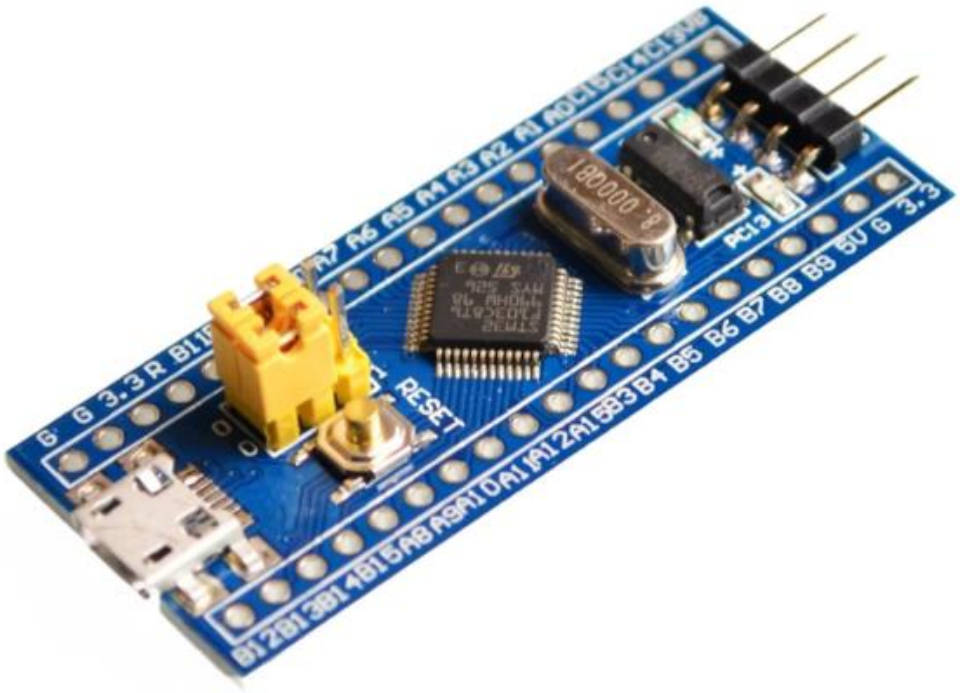
Embedded C Lab 2 Report : write bare-metal software to toggle a led connecting to GPIO portA13 on STM32F103C8T6 board.

ARM CORTEX-M3 STM32F103C8T6



By\ Eng . Belal Hani Abu Sabha

Required Modules After Reading Datasheet:

1-RCC (Reset and Clock Control) Module :
is necessary because GPIO has disabled clock by default .
Base address : 0x40021000
Registers : APB2ENR Register with offset 0x18
Enable bit : bit number 2

2-GPIO(general purpose input/output) Module :
I am working with GPIO portA
Base address : 0x40010800
Registers : A- CHR Register with offset 0x04 and we need to write 2 on byte number 5 to configure PIN13 Mode
B- ODR Register with offset 0x0C to write 0/1 on pin 13 .

Startup Process:

when power is on to the MCU the PC value will be 0 which mapped to 0x08000000 address automatically and in this address contains a vector table and the first four bytes at address 0x08000000 contain address of Stack Pointer then address 0x08000004 contains Reset Handler Address then next address contains next handler from the vector table.

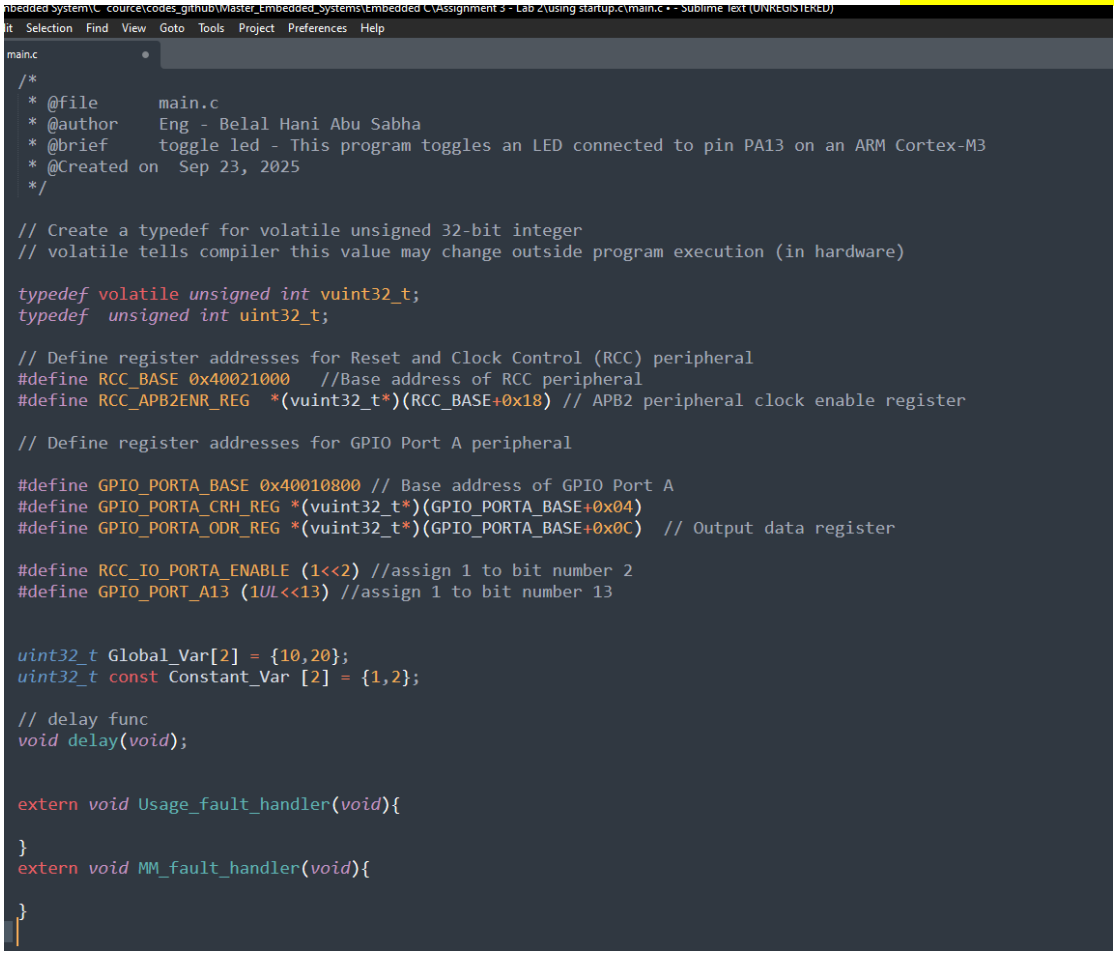
Notes :-

1- SP is loaded automatically : this feature only for Cortex-M3 a
2- We can write a startup in C language (startup.c) not only in Assembly language (startup.s) .

3- in linker file , The ALIGN(4) command ensure memory alignment for more efficiency .

4- use –gdwarf-2 debug to simulate code on proteus

Main.c:

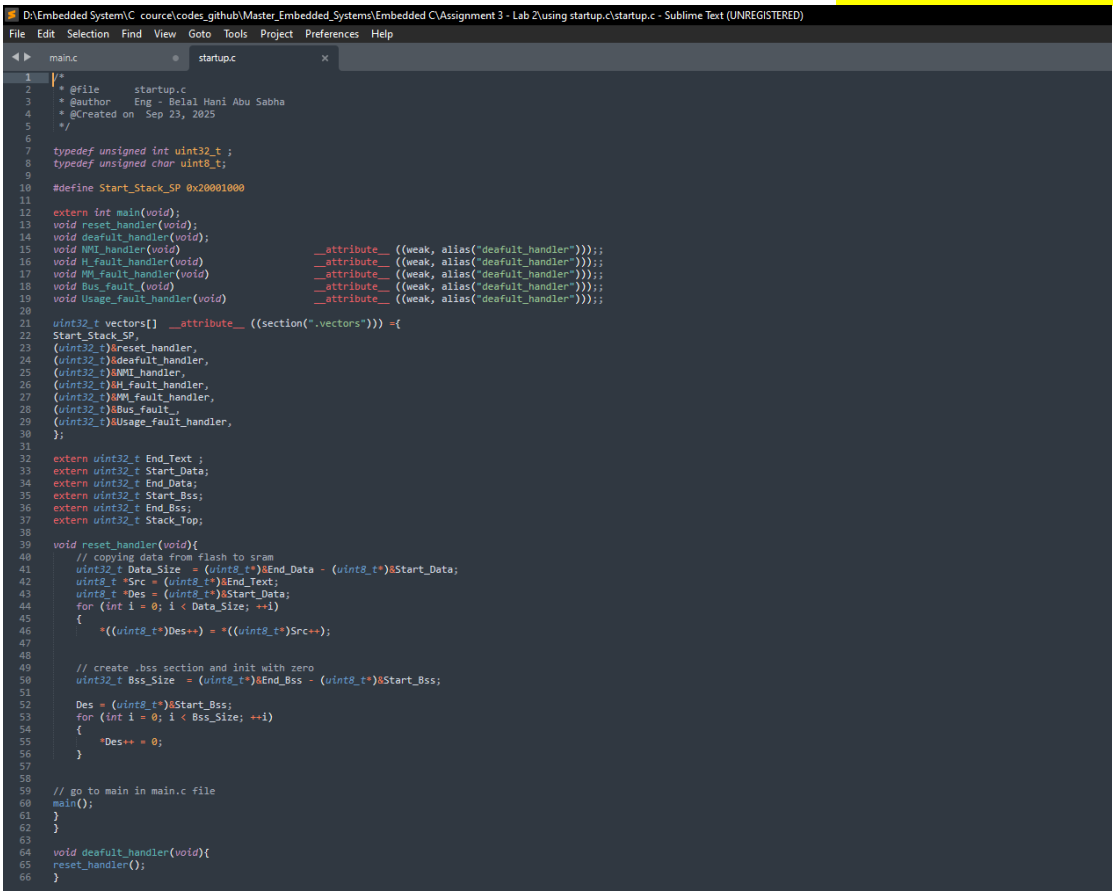

Selection Find View Goto Tools Project Preferences Help
main.c
/*
* @file main.c
* @author Eng - Belal Hani Abu Sabha
* @brief toggle led - This program toggles an LED connected to pin PA13 on an ARM Cortex-M3
* @Created on Sep 23, 2025
*/
// Create a typedef for volatile unsigned 32-bit integer
// volatile tells compiler this value may change outside program execution (in hardware)
typedef volatile unsigned int vuint32_t;
typedef unsigned int uint32_t;
// Define register addresses for Reset and Clock Control (RCC) peripheral
#define RCC_BASE 0x40021000 //Base address of RCC peripheral
#define RCC_APB2ENR_REG *(vuint32_t*)(RCC_BASE+0x18) // APB2 peripheral clock enable register
// Define register addresses for GPIO Port A peripheral
#define GPIO_PORTA_BASE 0x40010800 // Base address of GPIO Port A
#define GPIO_PORTA_CRH_REG *(vuint32_t*)(GPIO_PORTA_BASE+0x04)
#define GPIO_PORTA_ODR_REG *(vuint32_t*)(GPIO_PORTA_BASE+0x0C) // Output data register
#define RCC_IO_PORTA_ENABLE (1<<2) //assign 1 to bit number 2
#define GPIO_PORT_A13 (1UL<<13) //assign 1 to bit number 13
uint32_t Global_Var[2] = {10,20};
uint32_t const Constant_Var [2] = {1,2};
// delay func
void delay(void);
extern void Usage_fault_handler(void){
}
extern void MM_fault_handler(void){
}

main.c

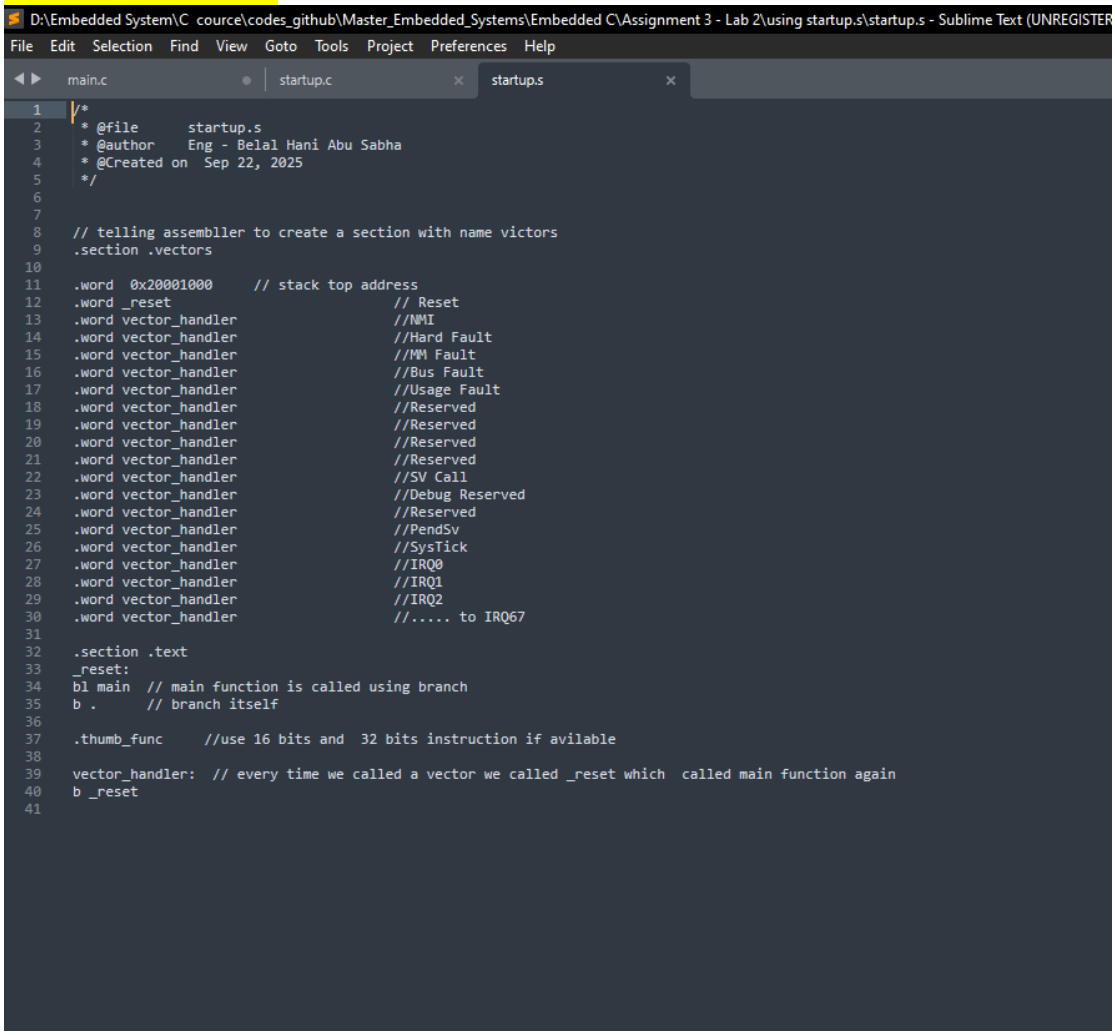
```c

/*    ----------------------------another solution to W/R on PortA13 using union and structure bit field-----------------------------------------------------------------

// Create a union typedef for accessing each bits of  ODR register

typedef union{
    vuint32_t _32_bits; // ensure we can access all 32 bits
    struct  // struct bit filed to access every bit
    {
        vuint32_t reserved:13; //reserved bit 0 to bit 12
        vuint32_t PIN_no_13:1;  //  bit 13 control output  state
    }Spin;
}U_ODR;


// Create a pointer to the ODR register

volatile U_ODR* Ptr_ODR = (volatile U_ODR*)(GPIO_PORTA_BASE+0x0C);

//------------------------------------------------------------------------------------------------------------------------------------------------------------ */

int main(void)
{
    RCC_APB2ENR_REG |=RCC_IO_PORTA_ENABLE; // XOR with 0000 0000 0000 0010 (assign 1 to bit 2) to enable RCC Clock

    // Configure PortA13 as outpu
    GPIO_PORTA_CRH_REG &=0xFF0FFFFF; // Clear bits 20-23
    GPIO_PORTA_CRH_REG |=0x00200000 ;// assign 1 to  bits to bit 21

    // infinite loop
    while(1){
    //   Ptr_ODR->Spin.PIN_no_13=1; // OUTPUT =1 , led is on until end of loop

      GPIO_PORTA_ODR_REG |= GPIO_PORT_A13; // OUTPUT =1 , led is on until end of loop
        delay(); // call delay func
      //  Ptr_ODR->Spin.PIN_no_13=0; // OUTPUT =0 , led is off until end of loop

  GPIO_PORTA_ODR_REG &= ~GPIO_PORT_A13; // OUTPUT =0 , led is off until end of loop
        delay(); // call delay func




    }
    return 0;
}

//  delay function
void delay(void) {
    for(vuint32_t i = 0; i < 500; i++);
}

```

Line 41, Column 1

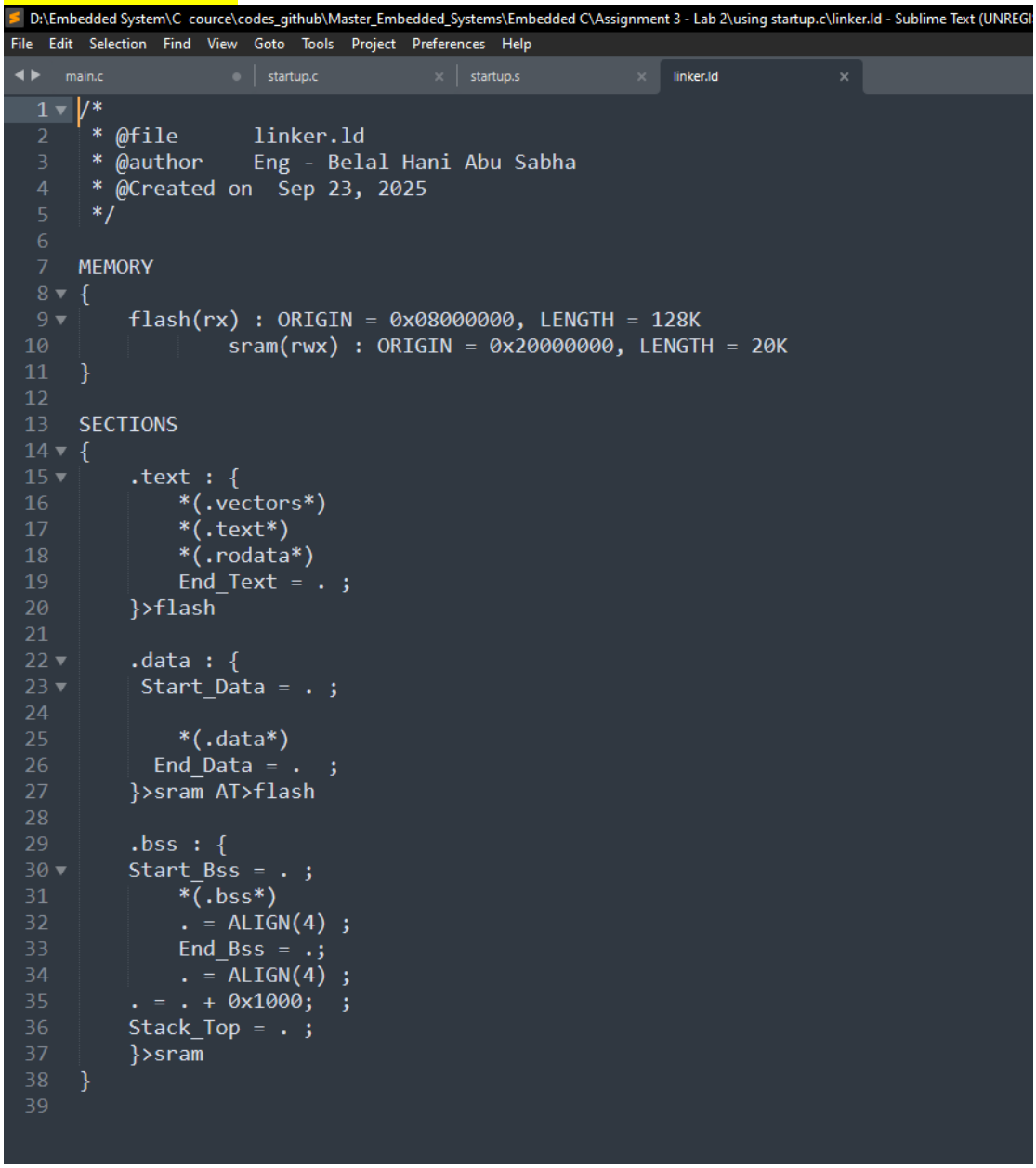
Startup.c:


D:\Embedded System\C cource\codes_github\Master_Embedded_Systems\Embedded C\Assignment 3 - Lab 2\using startup.c\startup.c - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
main.c
startup.c
/*
* @file startup.c
* @author Eng - Belal Hani Abu Sabha
* @Created on Sep 23, 2025
*/
typedef unsigned int uint32_t ;
typedef unsigned char uint8_t;
#define Start_Stack_SP 0x20001000
extern int main(void);
void reset_handler(void);
void deafult_handler(void);
void NMI_handler(void) __attribute__ ((weak, alias("deafult_handler")));;
void H_fault_handler(void) __attribute__ ((weak, alias("deafult_handler")));;
void MM_fault_handler(void) __attribute__ ((weak, alias("deafult_handler")));;
void Bus_fault_(void) __attribute__ ((weak, alias("deafult_handler")));;
void Usage_fault_handler(void) __attribute__ ((weak, alias("deafult_handler")));;
uint32_t vectors[] __attribute__ ((section(".vectors"))) ={
Start_Stack_SP,
(uint32_t)&reset_handler,
(uint32_t)&deafult_handler,
(uint32_t)&NMI_handler,
(uint32_t)&H_fault_handler,
(uint32_t)&MM_fault_handler,
(uint32_t)&Bus_fault_,
(uint32_t)&Usage_fault_handler,
};
extern uint32_t End_Text ;
extern uint32_t Start_Data;
extern uint32_t End_Data;
extern uint32_t Start_Bss;
extern uint32_t End_Bss;
extern uint32_t Stack_Top;
void reset_handler(void){
// copying data from flash to sram
uint32_t Data_Size = (uint8_t*)&End_Data - (uint8_t*)&Start_Data;
uint8_t *Src = (uint8_t*)&End_Text;
uint8_t *Des = (uint8_t*)&Start_Data;
for (int i = 0; i < Data_Size; ++i)
{
((uint8_t)Des++) = *((uint8_t*)Src++);
}
// create .bss section and init with zero
uint32_t Bss_Size = (uint8_t*)&End_Bss - (uint8_t*)&Start_Bss;
Des = (uint8_t*)&Start_Bss;
for (int i = 0; i < Bss_Size; ++i)
{
*Des++ = 0;
}
// go to main in main.c file
main();
}
}
void deafult_handler(void){
reset_handler();
}

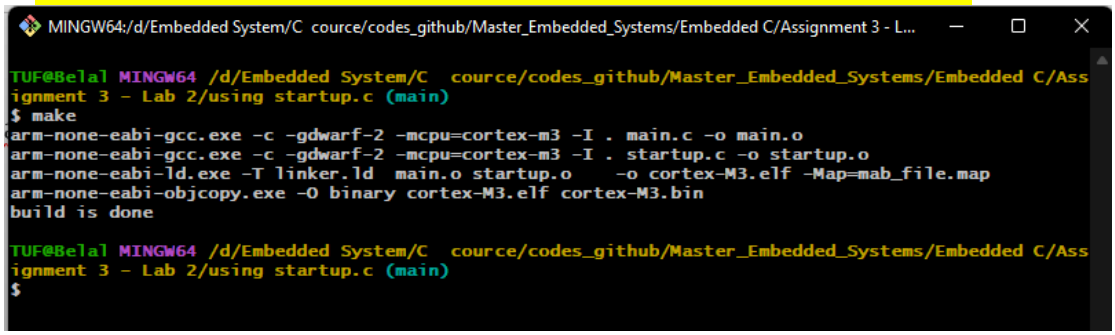
Startup.s:


D:\Embedded System\C cource\codes_github\Master_Embedded_Systems\Embedded C\Assignment 3 - Lab 2\using startup.s\startup.s - Sublime Text (UNREGISTER
File Edit Selection Find View Goto Tools Project Preferences Help
main.c
startup.c
startup.s
/*
* @file startup.s
* @author Eng - Belal Hani Abu Sabha
* @Created on Sep 22, 2025
*/
// telling assembller to create a section with name victors
.section .vectors
.word 0x20001000 // stack top address
.word _reset // Reset
.word vector_handler //NMI
.word vector_handler //Hard Fault
.word vector_handler //MM Fault
.word vector_handler //Bus Fault
.word vector_handler //Usage Fault
.word vector_handler //Reserved
.word vector_handler //Reserved
.word vector_handler //Reserved
.word vector_handler //Reserved
.word vector_handler //SV Call
.word vector_handler //Debug Reserved
.word vector_handler //Reserved
.word vector_handler //PendSv
.word vector_handler //SysTick
.word vector_handler //IRQ0
.word vector_handler //IRQ1
.word vector_handler //IRQ2
.word vector_handler //..... to IRQ67
.section .text
_reset:
bl main // main function is called using branch
b . // branch itself
.thumb_func //use 16 bits and 32 bits instruction if avilable
vector_handler: // every time we called a vector we called _reset which called main function again
b _reset

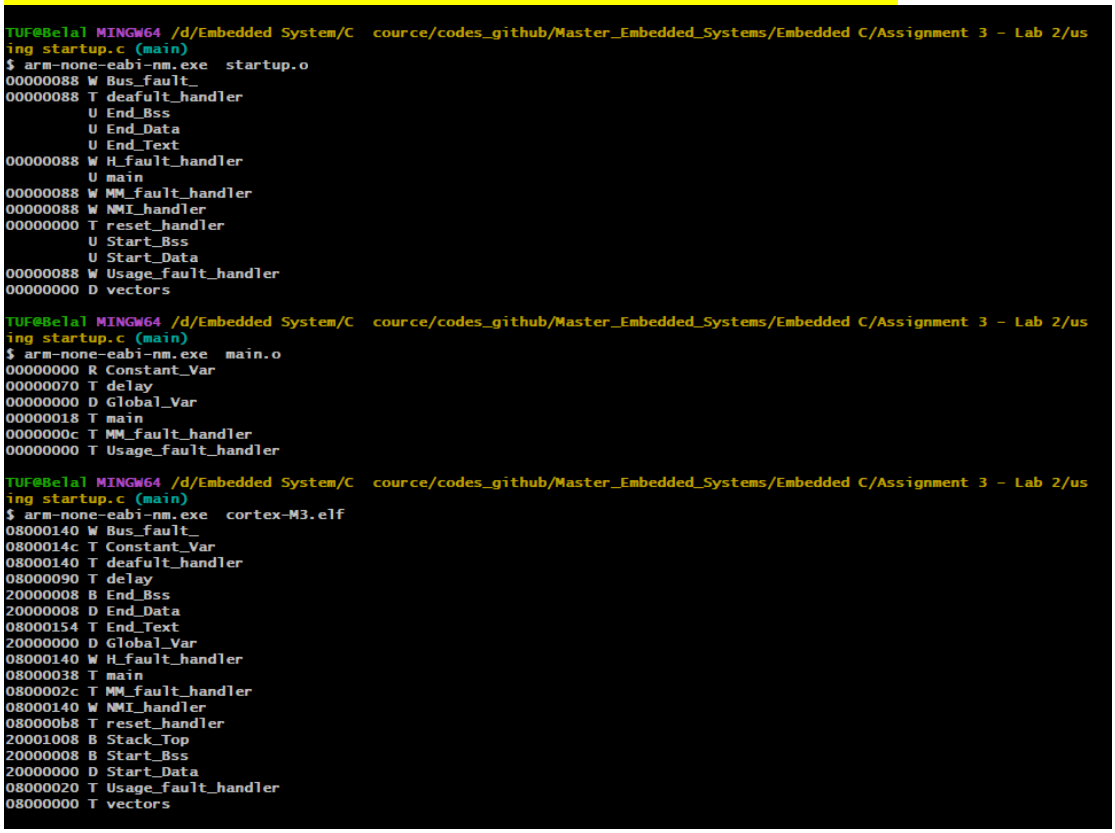
Linker.ld:


D:\Embedded System\C cource\codes_github\Master_Embedded_Systems\Embedded C\Assignment 3 - Lab 2\using startup.c\linker.ld - Sublime Text (UNREG
File Edit Selection Find View Goto Tools Project Preferences Help
main.c
startup.c
startup.s
linker.ld
/*
* @file linker.ld
* @author Eng - Belal Hani Abu Sabha
* @Created on Sep 23, 2025
*/
MEMORY
{
flash(rx) : ORIGIN = 0x08000000, LENGTH = 128K
sram(rwx) : ORIGIN = 0x20000000, LENGTH = 20K
}
SECTIONS
{
.text : {
(.vectors)
(.text)
(.rodata)
End_Text = . ;
}>flash
.data : {
Start_Data = . ;
(.data)
End_Data = . ;
}>sram AT>flash
.bss : {
Start_Bss = . ;
(.bss)
. = ALIGN(4) ;
End_Bss = .;
. = ALIGN(4) ;
. = . + 0x1000; ;
Stack_Top = . ;
}>sram
}

Compile and link using MakeFile:



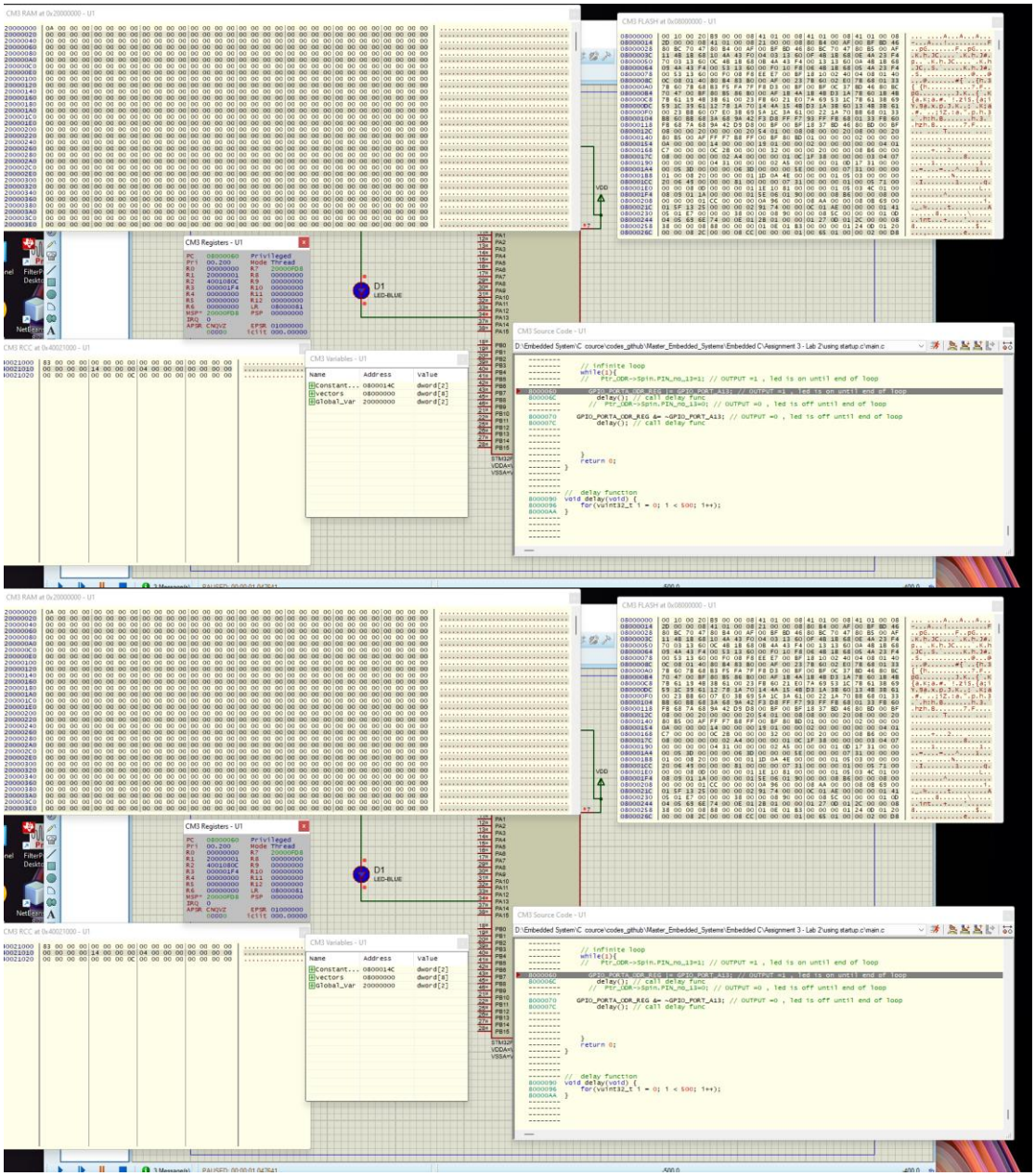
Symbols for *.o file and .elf file:



Mabfile.mab:

```
Memory Configuration

Name             Origin             Length             Attributes
flash            0x08000000         0x00020000         xr
sram             0x20000000         0x00005000         xrw
*default*        0x00000000         0xffffffff

Linker script and memory map


.text           0x08000000      0x154
 *(.vectors*)
 .vectors       0x08000000       0x20 startup.o
                0x08000000                vectors
 *(.text*)
 .text          0x08000020       0x98 main.o
                0x08000020                Usage_fault_handler
                0x0800002c                MM_fault_handler
                0x08000038                main
                0x08000090                delay
 .text          0x080000b8       0x94 startup.o
                0x080000b8                reset_handler
                0x08000140                Bus_fault_
                0x08000140                deafult_handler
                0x08000140                H_fault_handler
                0x08000140                NMI_handler
 *(.rodata*)
 .rodata        0x0800014c        0x8 main.o
                0x0800014c                Constant_Var
                0x08000154                End_Text = .

.glue_7         0x08000154        0x0
 .glue_7        0x08000154        0x0 linker stubs

.glue_7t        0x08000154        0x0
 .glue_7t       0x08000154        0x0 linker stubs

.vfp11_veneer   0x08000154        0x0
 .vfp11_veneer  0x08000154        0x0 linker stubs

.v4_bx          0x08000154        0x0
 .v4_bx         0x08000154        0x0 linker stubs

.iplt           0x08000154        0x0
 .iplt          0x08000154        0x0 main.o

.rel.dyn        0x08000154        0x0
 .rel.iplt      0x08000154        0x0 main.o
```

Run And Debug :



Thank You