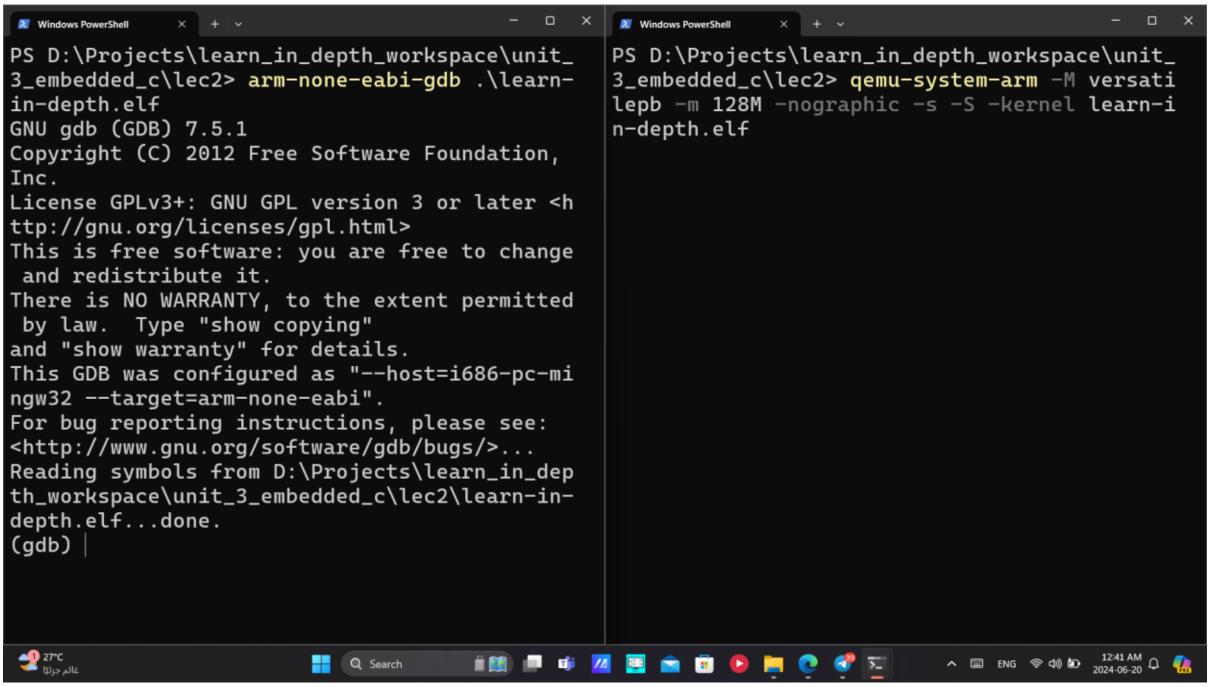
Youssef Samy Youssef

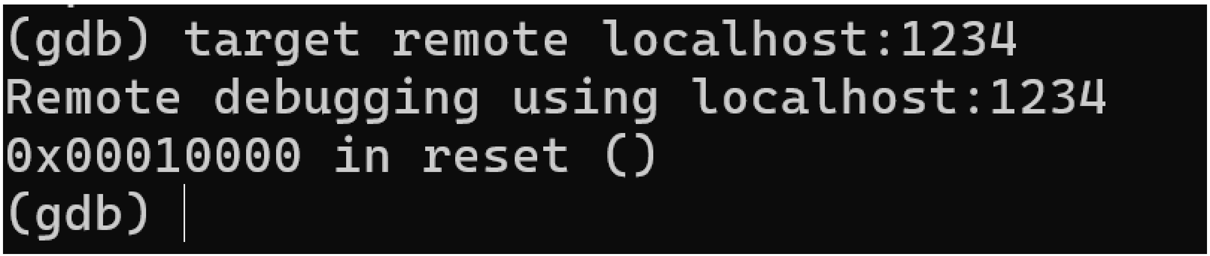
yosefsamy019@gmail.com

GDB

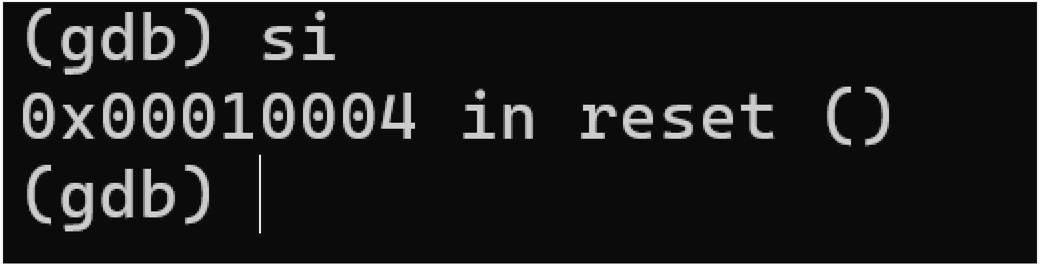


```
PS D:\Projects\learn_in_depth_workspace\unit_3_embedded_c\lec2> arm-none-eabi-gdb .\learn-in-depth.elf
GNU gdb (GDB) 7.5.1
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-mingw32 --target=arm-none-eabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from D:\Projects\learn_in_depth_workspace\unit_3_embedded_c\lec2\learn-in-depth.elf...done.
(gdb)
```

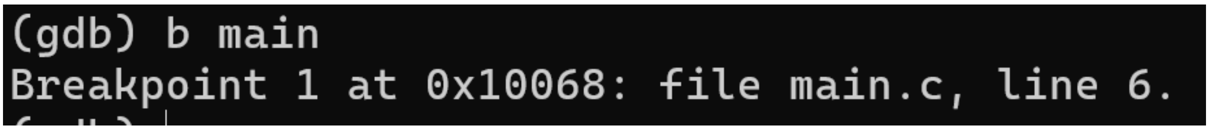
```
PS D:\Projects\learn_in_depth_workspace\unit_3_embedded_c\lec2> qemu-system-arm -M versatilepb -m 128M -nographic -s -S -kernel learn-in-depth.elf
```



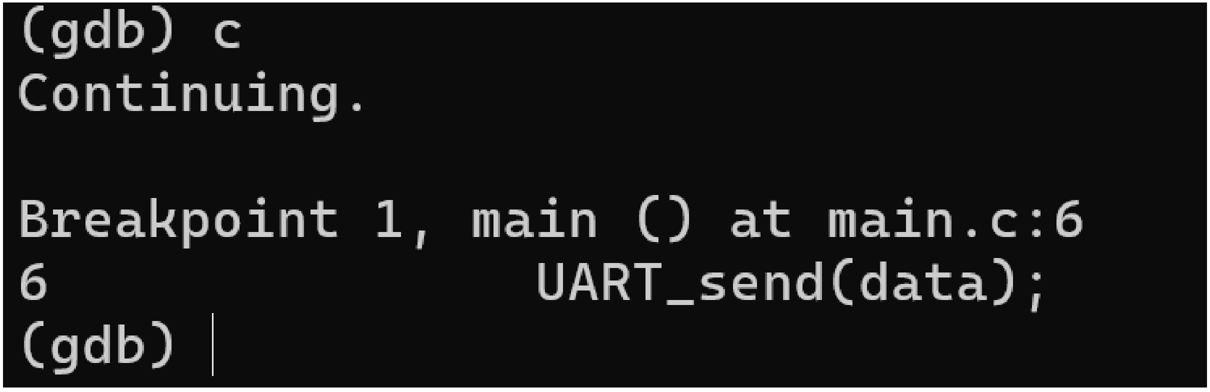
```
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
0x00010000 in reset ()
(gdb)
```



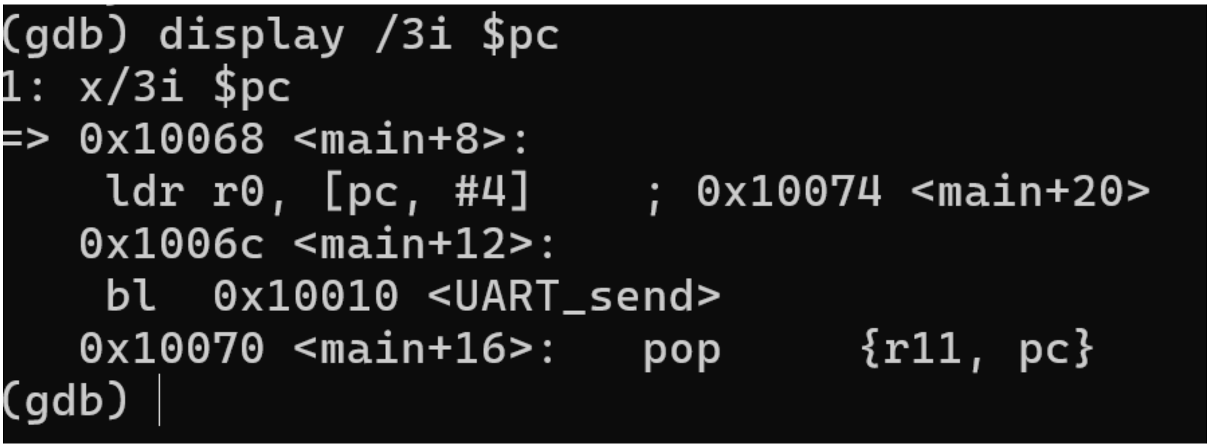
```
(gdb) si
0x00010004 in reset ()
(gdb)
```



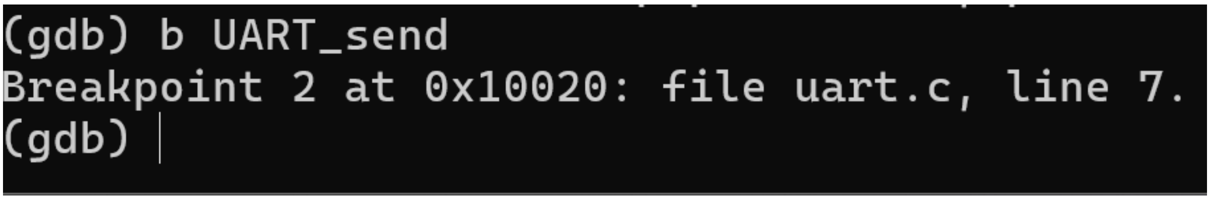
```
(gdb) b main
Breakpoint 1 at 0x10068: file main.c, line 6.
```



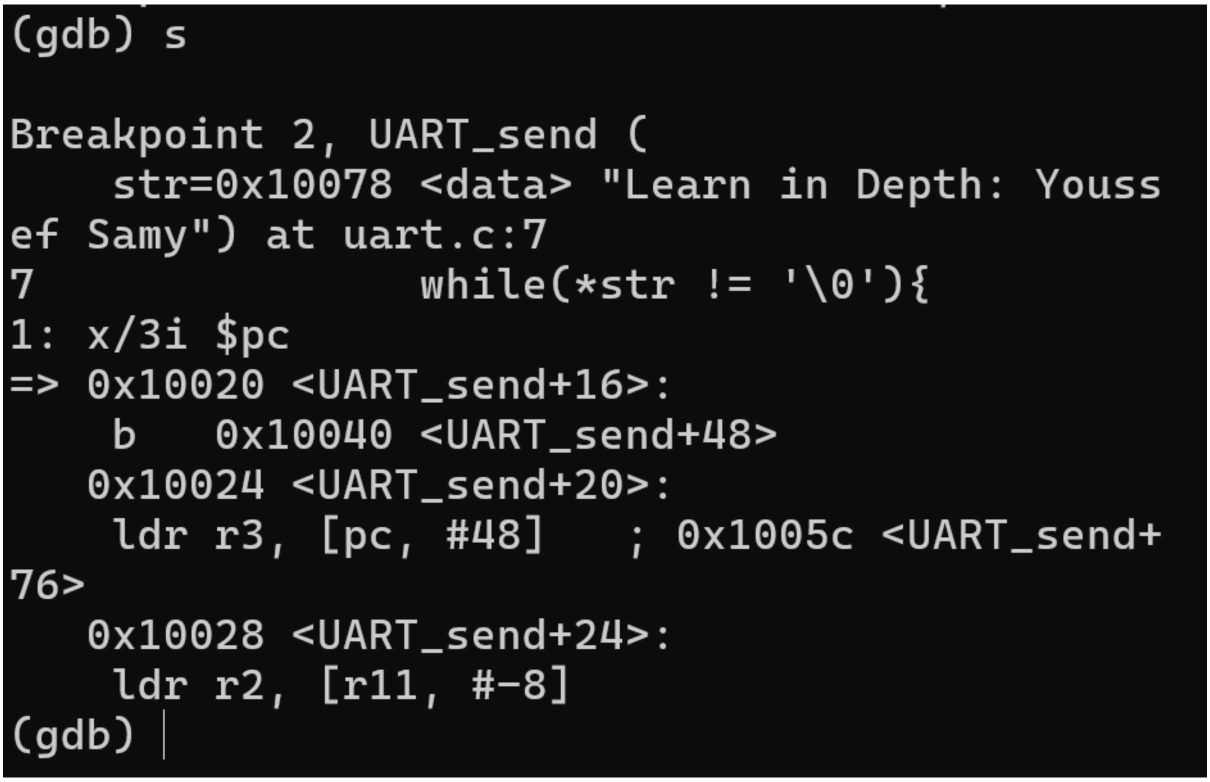
```
(gdb) c
Continuing.

Breakpoint 1, main () at main.c:6
6                 UART_send(data);
(gdb) 
```



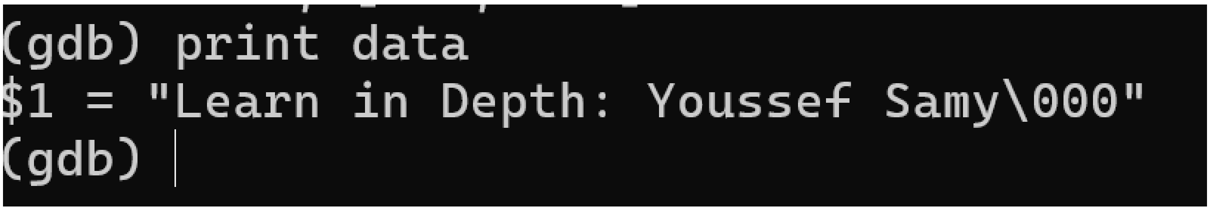
```
(gdb) display /3i $pc
1: x/3i $pc
=> 0x10068 <main+8>:
    ldr r0, [pc, #4]     ; 0x10074 <main+20>
   0x1006c <main+12>:
    bl  0x10010 <UART_send>
   0x10070 <main+16>:   pop     {r11, pc}
(gdb) 
```



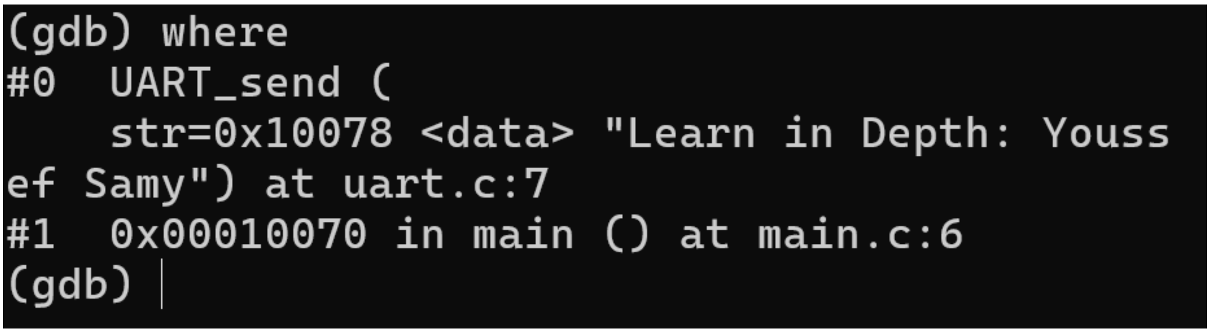
```
(gdb) b UART_send
Breakpoint 2 at 0x10020: file uart.c, line 7.
(gdb) 
```



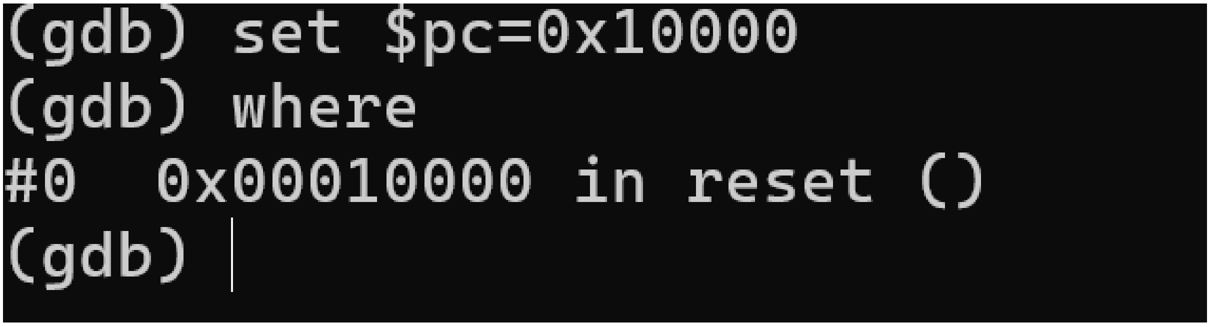
```
(gdb) s

Breakpoint 2, UART_send (
    str=0x10078 <data> "Learn in Depth: Youss
ef Samy") at uart.c:7
7               while(*str != '\0'){
1: x/3i $pc
=> 0x10020 <UART_send+16>:
    b   0x10040 <UART_send+48>
   0x10024 <UART_send+20>:
    ldr r3, [pc, #48]   ; 0x1005c <UART_send+
76>
   0x10028 <UART_send+24>:
    ldr r2, [r11, #-8]
(gdb)
```



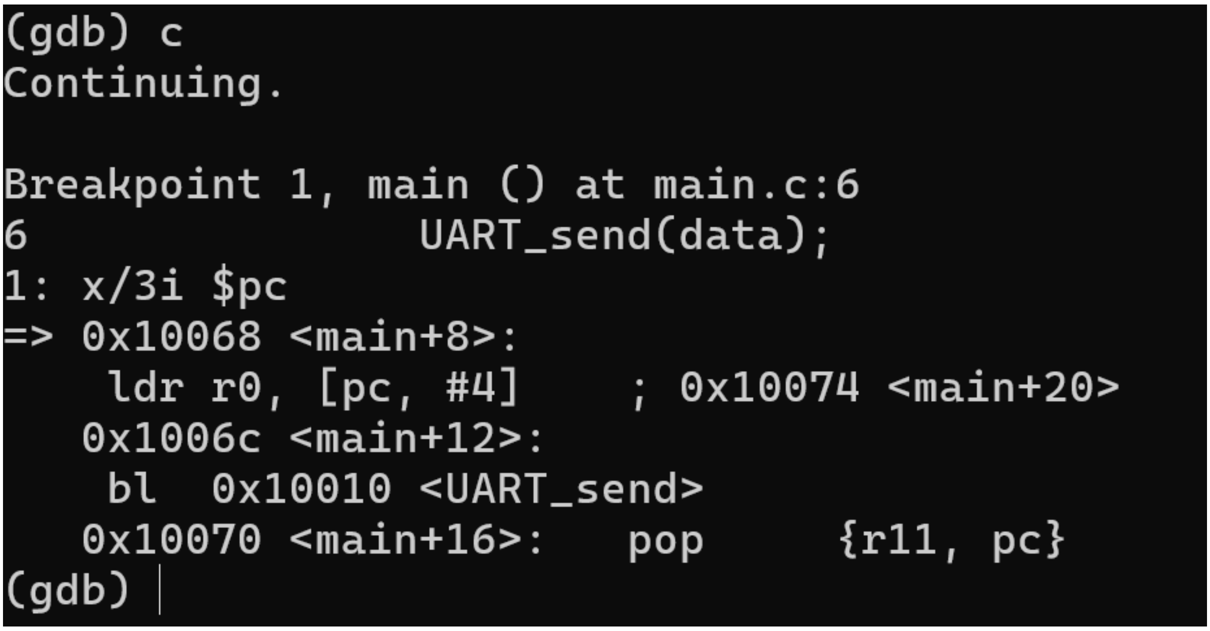
```
(gdb) print data
$1 = "Learn in Depth: Youssef Samy\000"
(gdb)
```



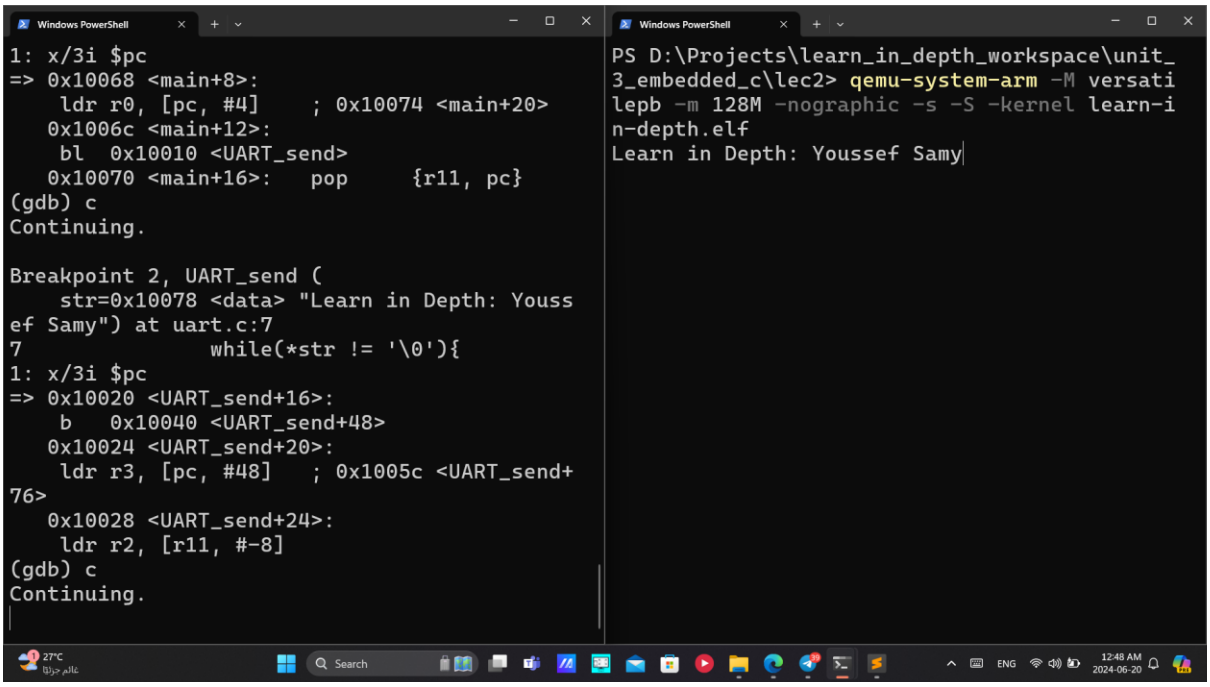
```
(gdb) where
#0  UART_send (
    str=0x10078 <data> "Learn in Depth: Youss
ef Samy") at uart.c:7
#1  0x00010070 in main () at main.c:6
(gdb)
```



```
(gdb) set $pc=0x10000
(gdb) where
#0  0x00010000 in reset ()
(gdb)
```



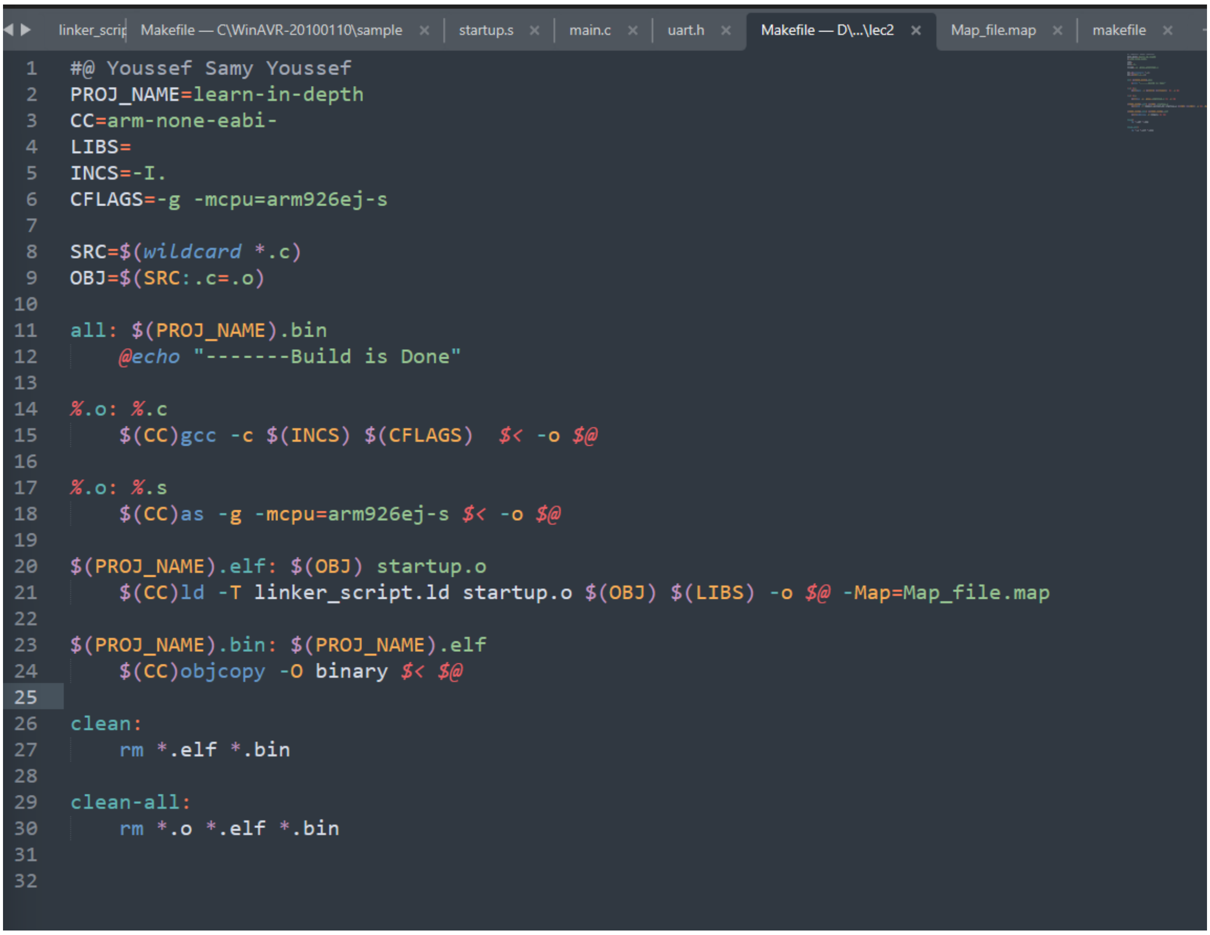
```
(gdb) c
Continuing.

Breakpoint 1, main () at main.c:6
6               UART_send(data);
1: x/3i $pc
=> 0x10068 <main+8>:
    ldr r0, [pc, #4]     ; 0x10074 <main+20>
   0x1006c <main+12>:
    bl  0x10010 <UART_send>
   0x10070 <main+16>:    pop      {r11, pc}
(gdb)
```



```
1: x/3i $pc
=> 0x10068 <main+8>:
    ldr r0, [pc, #4]     ; 0x10074 <main+20>
   0x1006c <main+12>:
    bl  0x10010 <UART_send>
   0x10070 <main+16>:    pop      {r11, pc}
(gdb) c
Continuing.

Breakpoint 2, UART_send (
    str=0x10078 <data> "Learn in Depth: Youss
ef Samy") at uart.c:7
7               while(*str != '\0'){
1: x/3i $pc
=> 0x10020 <UART_send+16>:
    b   0x10040 <UART_send+48>
   0x10024 <UART_send+20>:
    ldr r3, [pc, #48]    ; 0x1005c <UART_send+
76>
   0x10028 <UART_send+24>:
    ldr r2, [r11, #-8]
(gdb) c
Continuing.
```

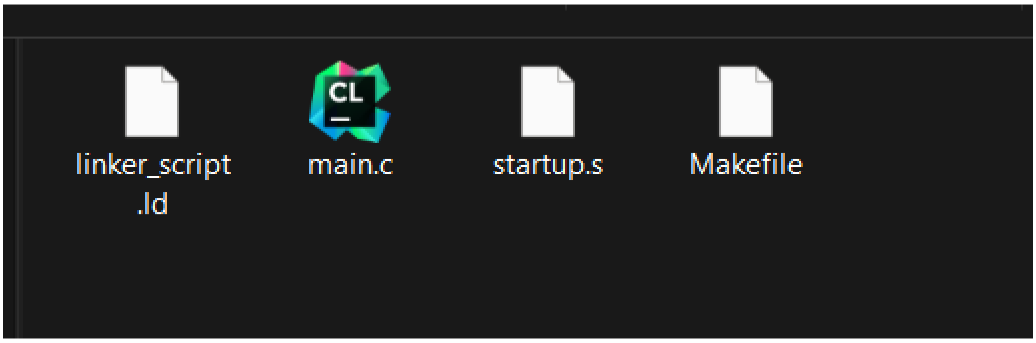
```
PS D:\Projects\learn_in_depth_workspace\unit_
3_embedded_c\lec2> qemu-system-arm -M versati
lepb -m 128M -nographic -s -S -kernel learn-i
n-depth.elf
Learn in Depth: Youssef Samy
```

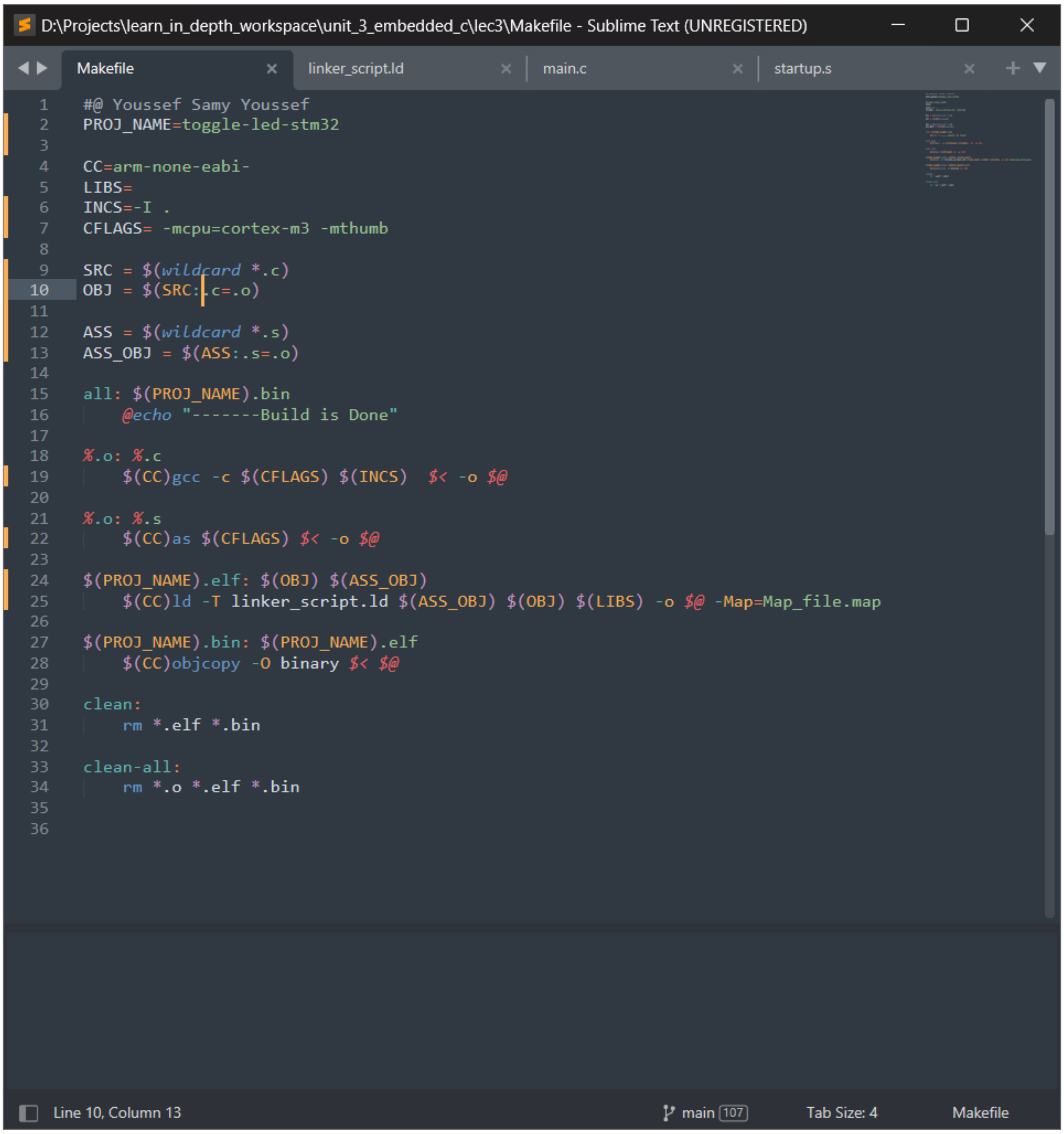
Make File



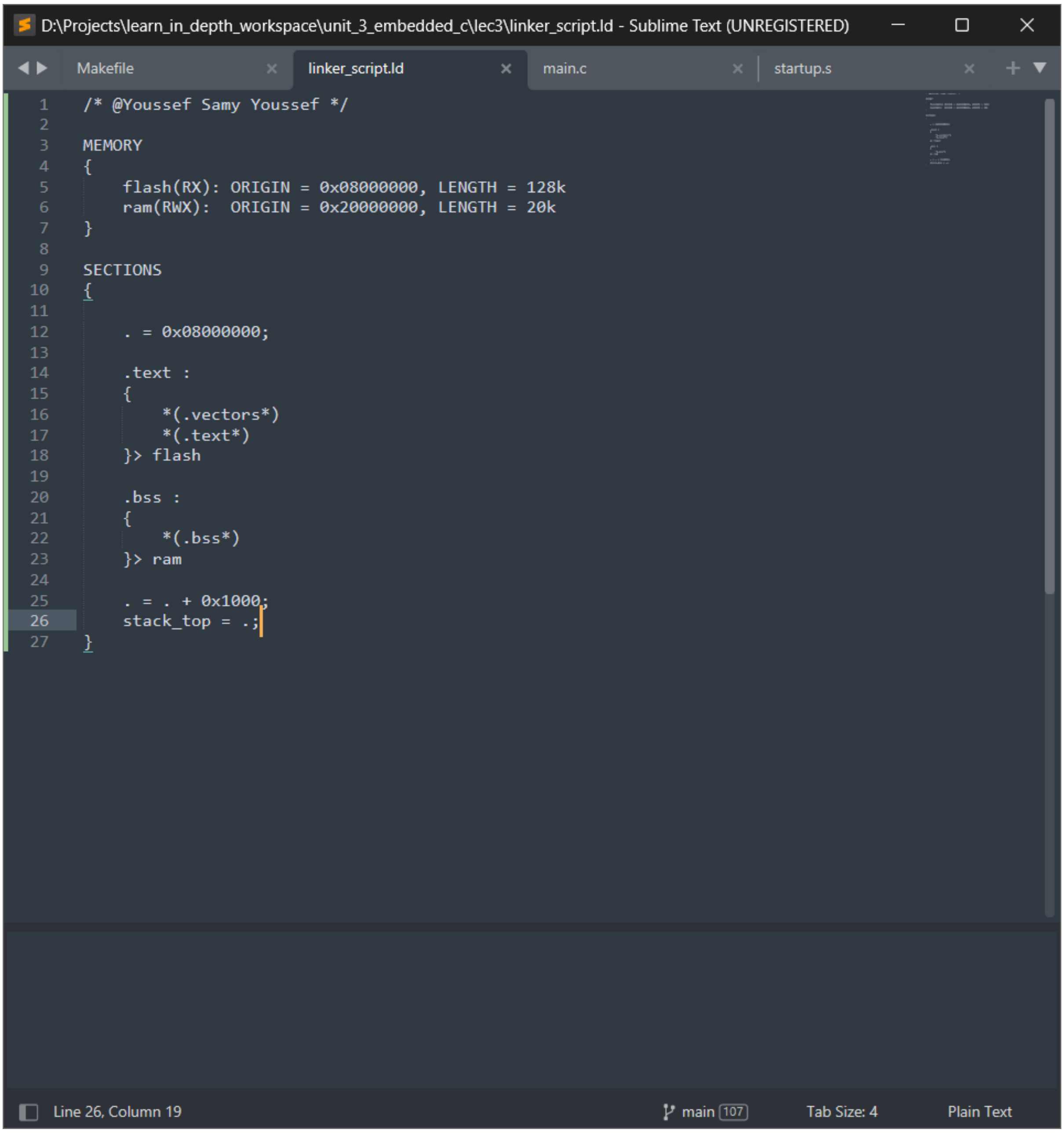
```
#@ Youssef Samy Youssef
PROJ_NAME=learn-in-depth
CC=arm-none-eabi-
LIBS=
INCS=-I.
CFLAGS=-g -mcpu=arm926ej-s

SRC=$(wildcard *.c)
OBJ=$(SRC:.c=.o)

all: $(PROJ_NAME).bin
    @echo "-------Build is Done"

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

%.o: %.s
    $(CC)as -g -mcpu=arm926ej-s $< -o $@

$(PROJ_NAME).elf: $(OBJ) startup.o
    $(CC)ld -T linker_script.ld startup.o $(OBJ) $(LIBS) -o $@ -Map=Map_file.map

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

clean:
    rm *.elf *.bin

clean-all:
    rm *.o *.elf *.bin
```

STM32 Toggle led Task using startup assembly file

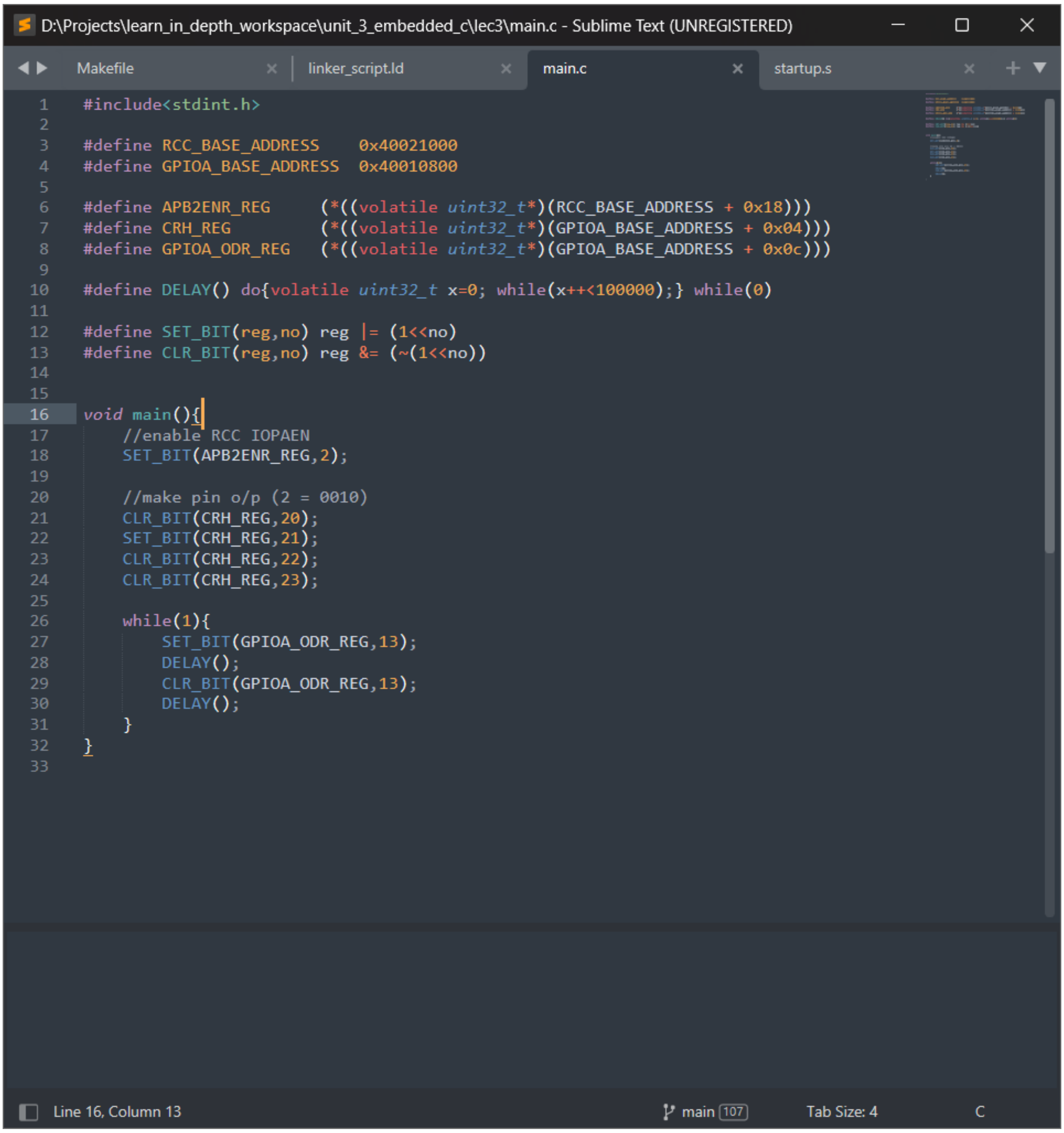



D:\Projects\learn_in_depth_workspace\unit_3_embedded_c\lec3\Makefile - Sublime Text (UNREGISTERED)
Makefile
linker_script.ld
main.c
startup.s
Line 10, Column 13
main 107
Tab Size: 4
Makefile

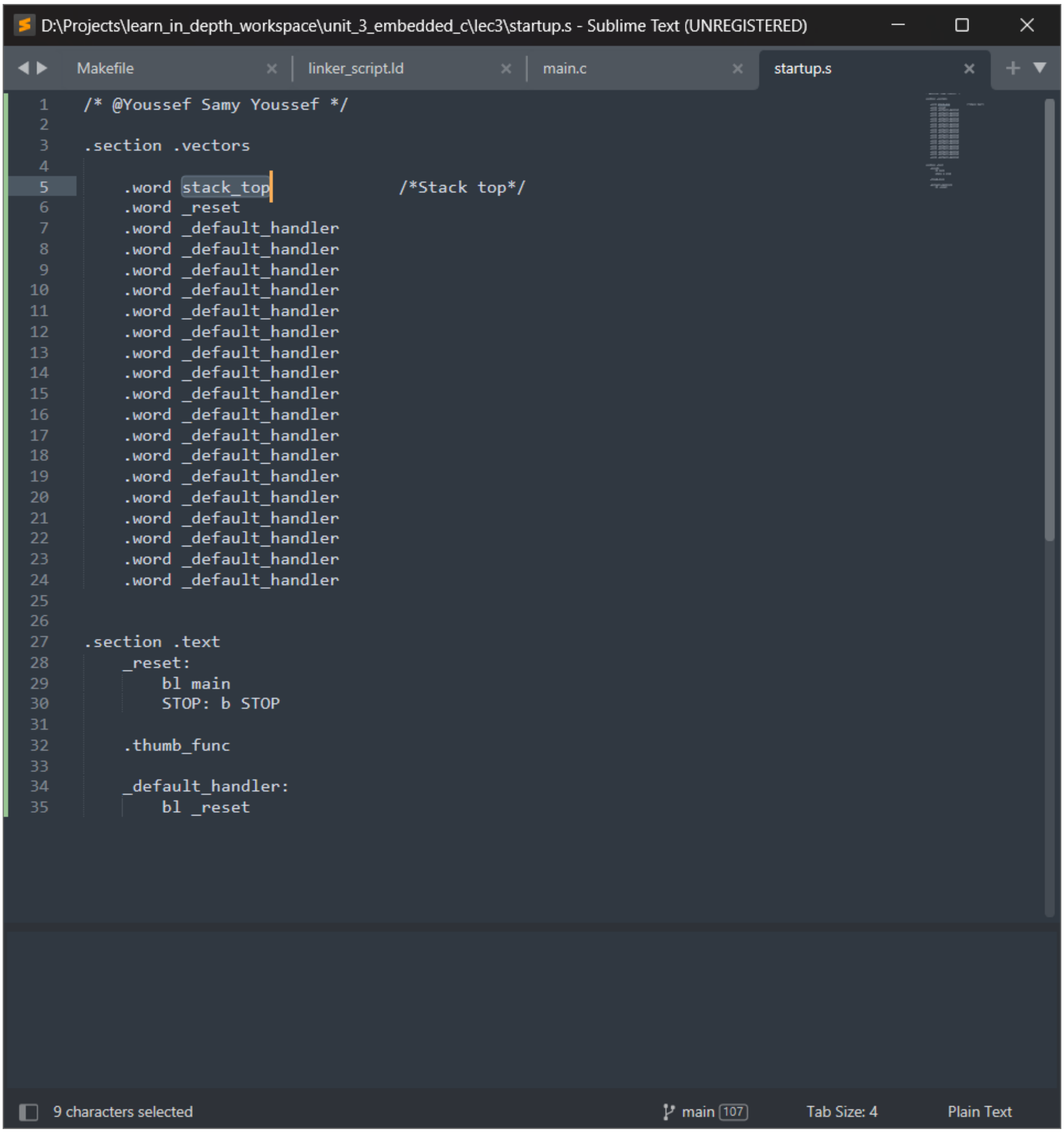
```makefile
#@ Youssef Samy Youssef
PROJ_NAME=toggle-led-stm32

CC=arm-none-eabi-
LIBS=
INCS=-I .
CFLAGS= -mcpu=cortex-m3 -mthumb

SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)

ASS = $(wildcard *.s)
ASS_OBJ = $(ASS:.s=.o)

all: $(PROJ_NAME).bin
	@echo "-------Build is Done"

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

%.o: %.s
	$(CC)as $(CFLAGS) $< -o $@

$(PROJ_NAME).elf: $(OBJ) $(ASS_OBJ)
	$(CC)ld -T linker_script.ld $(ASS_OBJ) $(OBJ) $(LIBS) -o $@ -Map=Map_file.map

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

clean:
	rm *.elf *.bin

clean-all:
	rm *.o *.elf *.bin
```


D:\Projects\learn_in_depth_workspace\unit_3_embedded_c\lec3\linker_script.ld - Sublime Text (UNREGISTERED)
Makefile
linker_script.ld
main.c
startup.s
Line 26, Column 19
main 107
Tab Size: 4
Plain Text

```
/* @Youssef Samy Youssef */

MEMORY
{
    flash(RX): ORIGIN = 0x08000000, LENGTH = 128k
    ram(RWX):  ORIGIN = 0x20000000, LENGTH = 20k
}

SECTIONS
{

    . = 0x08000000;

    .text :
    {
        *(.vectors*)
        *(.text*)
    }> flash

    .bss :
    {
        *(.bss*)
    }> ram

    . = . + 0x1000;
    stack_top = .;
}
```

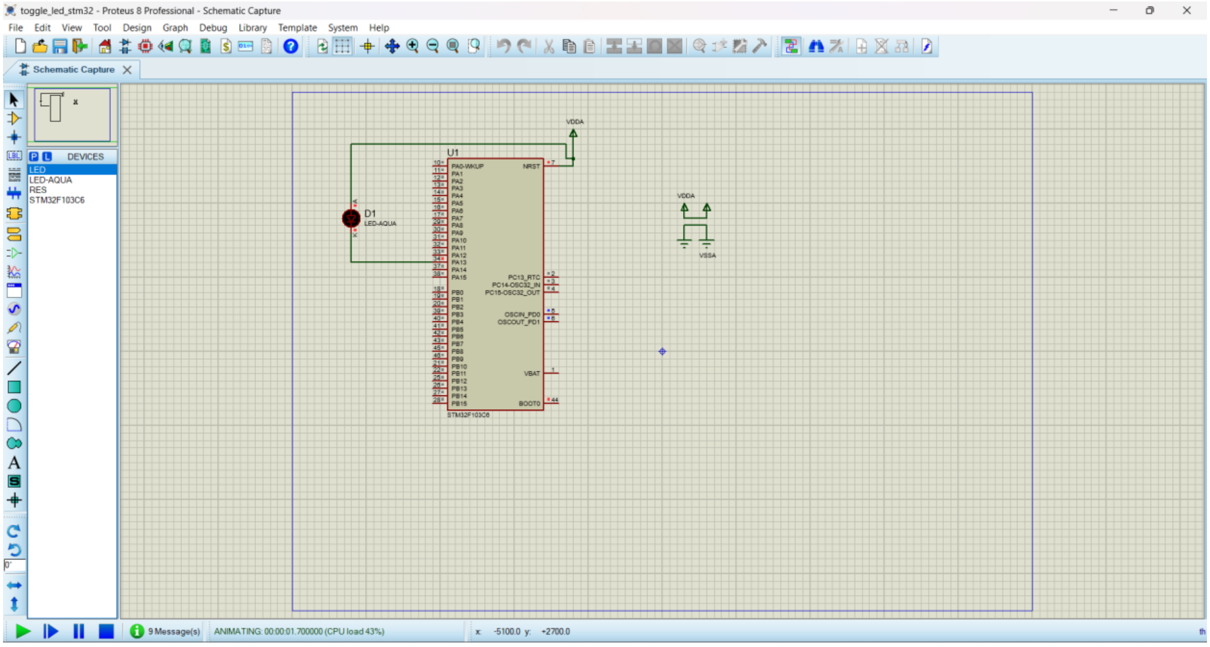

D:\Projects\learn_in_depth_workspace\unit_3_embedded_c\lec3\main.c - Sublime Text (UNREGISTERED)
Makefile
linker_script.ld
main.c
startup.s
Line 16, Column 13
main 107
Tab Size: 4
C

```c
#include<stdint.h>

#define RCC_BASE_ADDRESS    0x40021000
#define GPIOA_BASE_ADDRESS  0x40010800

#define APB2ENR_REG     (*((volatile uint32_t*)(RCC_BASE_ADDRESS + 0x18)))
#define CRH_REG         (*((volatile uint32_t*)(GPIOA_BASE_ADDRESS + 0x04)))
#define GPIOA_ODR_REG   (*((volatile uint32_t*)(GPIOA_BASE_ADDRESS + 0x0c)))

#define DELAY() do{volatile uint32_t x=0; while(x++<100000);} while(0)

#define SET_BIT(reg,no) reg |= (1<<no)
#define CLR_BIT(reg,no) reg &= (~(1<<no))


void main(){
    //enable RCC IOPAEN
    SET_BIT(APB2ENR_REG,2);

    //make pin o/p (2 = 0010)
    CLR_BIT(CRH_REG,20);
    SET_BIT(CRH_REG,21);
    CLR_BIT(CRH_REG,22);
    CLR_BIT(CRH_REG,23);

    while(1){
        SET_BIT(GPIOA_ODR_REG,13);
        DELAY();
        CLR_BIT(GPIOA_ODR_REG,13);
        DELAY();
    }
}

```



D:\Projects\learn_in_depth_workspace\unit_3_embedded_c\lec3\startup.s - Sublime Text (UNREGISTERED)
Makefile
linker_script.ld
main.c
startup.s
9 characters selected
main 107
Tab Size: 4
Plain Text

```
/* @Youssef Samy Youssef */

.section .vectors

    .word stack_top            /*Stack top*/
    .word _reset
    .word _default_handler
    .word _default_handler
    .word _default_handler
    .word _default_handler
    .word _default_handler
    .word _default_handler
    .word _default_handler
    .word _default_handler
    .word _default_handler
    .word _default_handler
    .word _default_handler
    .word _default_handler
    .word _default_handler
    .word _default_handler
    .word _default_handler
    .word _default_handler
    .word _default_handler
    .word _default_handler


.section .text
    _reset:
        bl main
        STOP: b STOP

    .thumb_func

    _default_handler:
        bl _reset
```

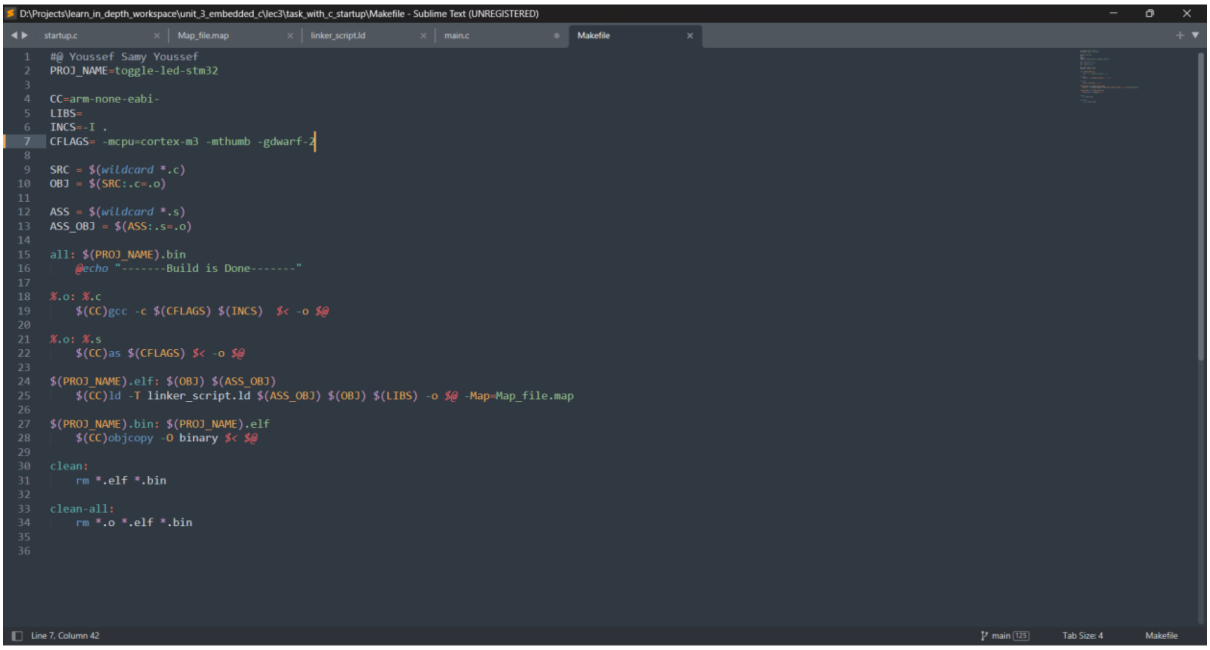


9 Message(s)

ANIMATING: 00:00:01.700000 (CPU load 43%)

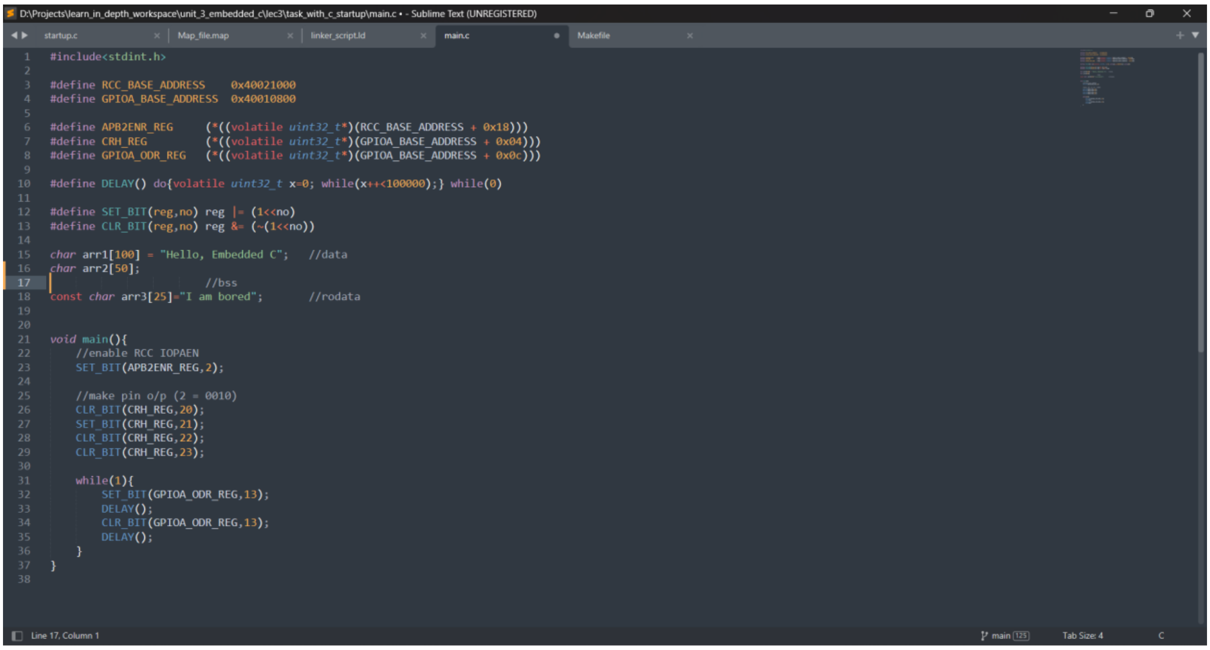
x: -5100.0 y: +2700.0

STM32 Toggle led using startup c code



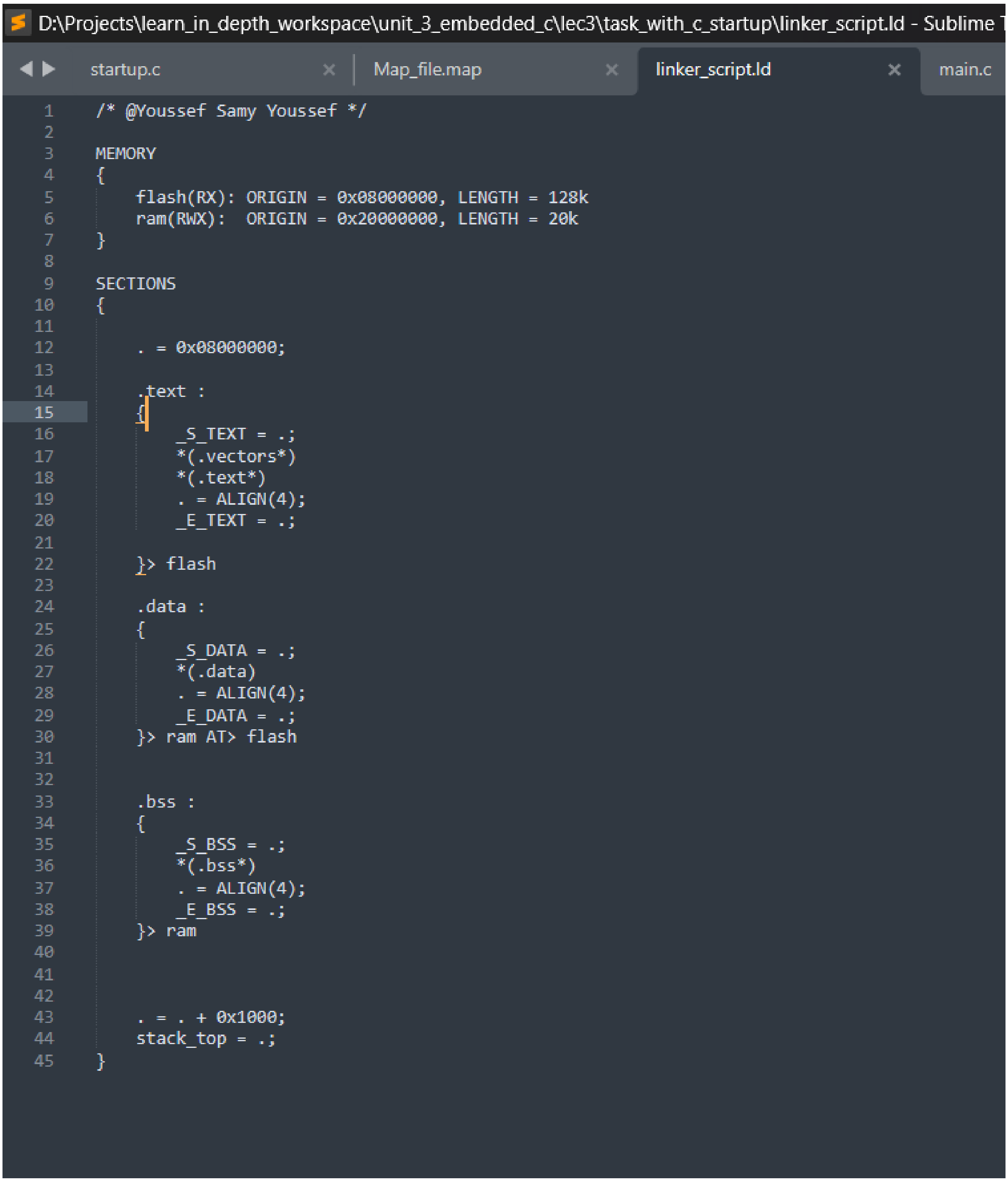
D:\Projects\learn_in_depth_workspace\unit_3_embedded_c\lec3\task_with_c_startup\Makefile - Sublime Text (UNREGISTERED)
startup.c
Map_file.map
linker_script.ld
main.c
Makefile
Line 7, Column 42
main 125
Tab Size: 4
Makefile

```makefile
#@ Youssef Samy Youssef
PROJ_NAME=toggle-led-stm32

CC=arm-none-eabi-
LIBS=
INCS=-I .
CFLAGS= -mcpu=cortex-m3 -mthumb -gdwarf-2

SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)

ASS = $(wildcard *.s)
ASS_OBJ = $(ASS:.s=.o)

all: $(PROJ_NAME).bin
	@echo "-------Build is Done-------"

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

%.o: %.s
	$(CC)as $(CFLAGS) $< -o $@

$(PROJ_NAME).elf: $(OBJ) $(ASS_OBJ)
	$(CC)ld -T linker_script.ld $(ASS_OBJ) $(OBJ) $(LIBS) -o $@ -Map=Map_file.map

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

clean:
	rm *.elf *.bin

clean-all:
	rm *.o *.elf *.bin
```

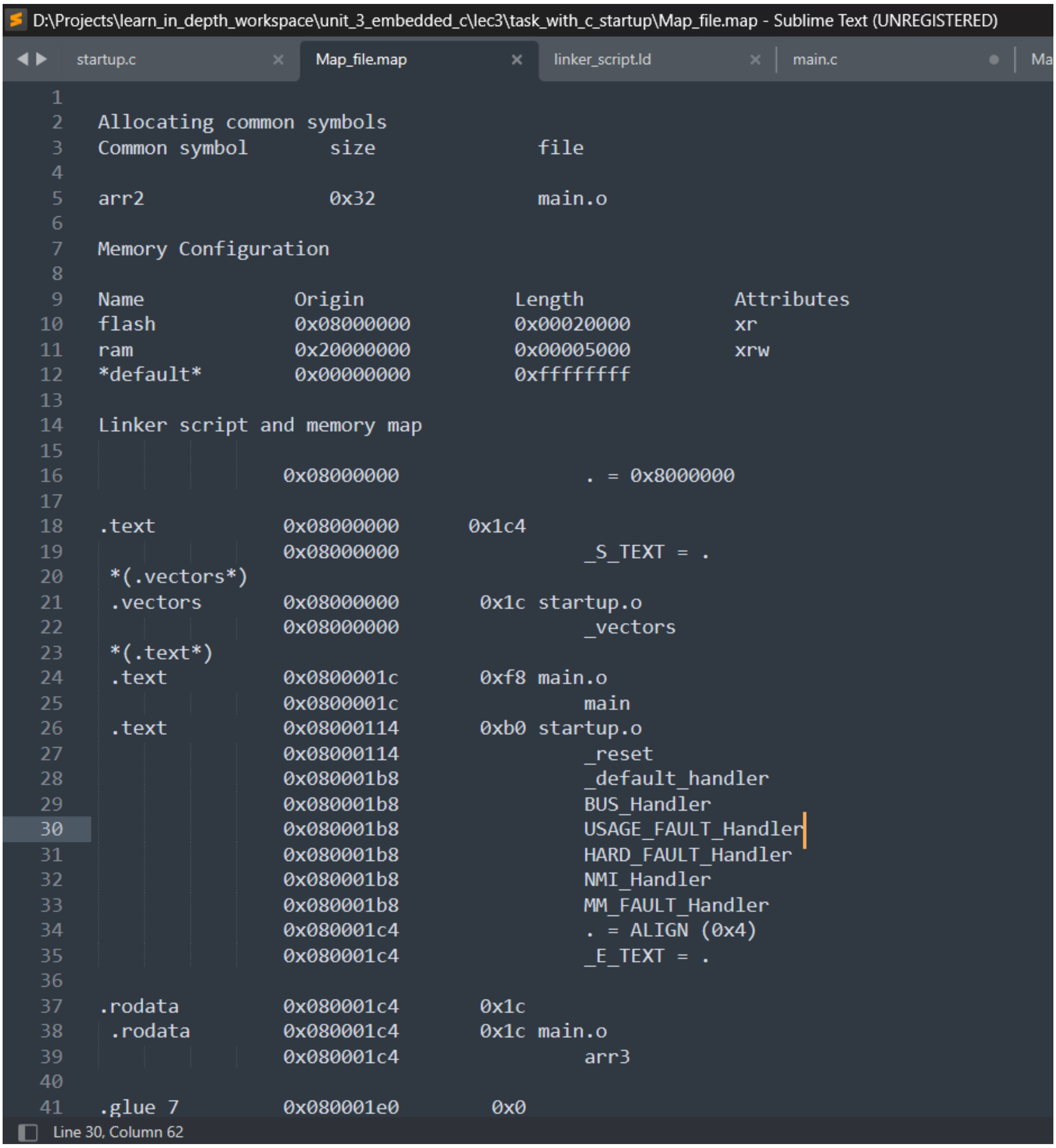


D:\Projects\learn_in_depth_workspace\unit_3_embedded_c\lec3\task_with_c_startup\main.c • - Sublime Text (UNREGISTERED)
startup.c
Map_file.map
linker_script.ld
main.c
Makefile
Line 17, Column 1
main 125
Tab Size: 4
C

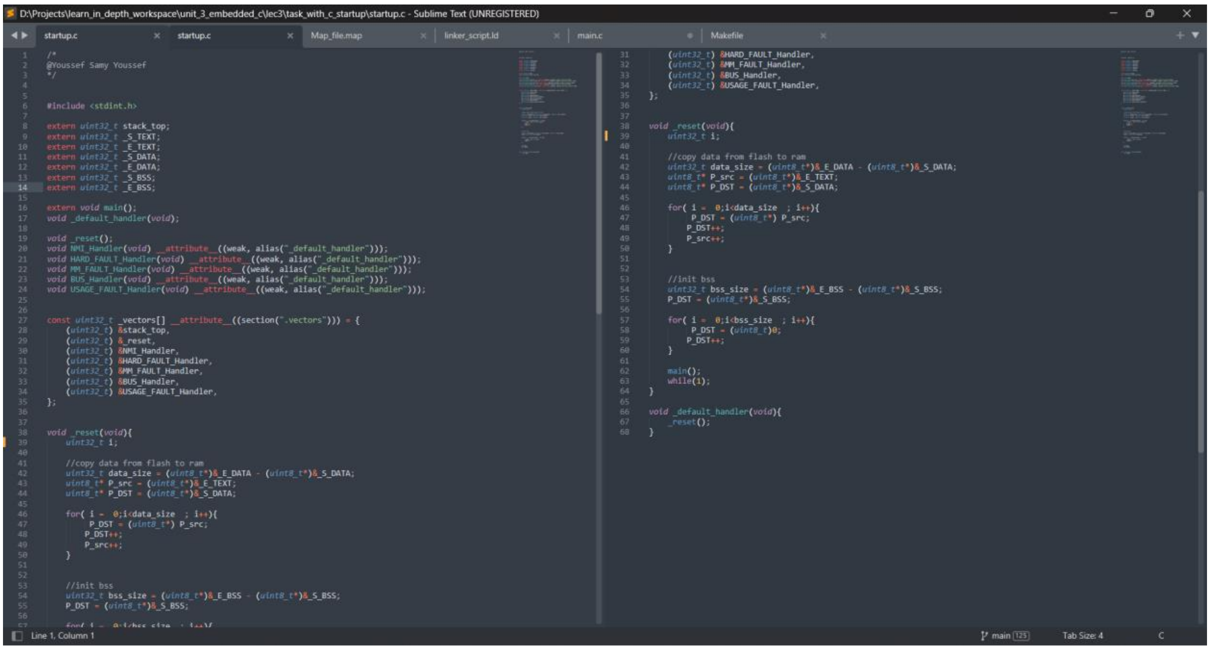
```c
#include<stdint.h>

#define RCC_BASE_ADDRESS    0x40021000
#define GPIOA_BASE_ADDRESS  0x40010800

#define APB2ENR_REG     (*((volatile uint32_t*)(RCC_BASE_ADDRESS + 0x18)))
#define CRH_REG         (*((volatile uint32_t*)(GPIOA_BASE_ADDRESS + 0x04)))
#define GPIOA_ODR_REG   (*((volatile uint32_t*)(GPIOA_BASE_ADDRESS + 0x0c)))

#define DELAY() do{volatile uint32_t x=0; while(x++<100000);} while(0)

#define SET_BIT(reg,no) reg |= (1<<no)
#define CLR_BIT(reg,no) reg &= (~(1<<no))

char arr1[100] = "Hello, Embedded C";   //data
char arr2[50];
                        //bss
const char arr3[25]="I am bored";       //rodata


void main(){
    //enable RCC IOPAEN
    SET_BIT(APB2ENR_REG,2);

    //make pin o/p (2 = 0010)
    CLR_BIT(CRH_REG,20);
    SET_BIT(CRH_REG,21);
    CLR_BIT(CRH_REG,22);
    CLR_BIT(CRH_REG,23);

    while(1){
        SET_BIT(GPIOA_ODR_REG,13);
        DELAY();
        CLR_BIT(GPIOA_ODR_REG,13);
        DELAY();
    }
}
```


D:\Projects\learn_in_depth_workspace\unit_3_embedded_c\lec3\task_with_c_startup\linker_script.ld - Sublime T
startup.c
Map_file.map
linker_script.ld
main.c

```
/* @Youssef Samy Youssef */

MEMORY
{
    flash(RX): ORIGIN = 0x08000000, LENGTH = 128k
    ram(RWX):  ORIGIN = 0x20000000, LENGTH = 20k
}

SECTIONS
{

    . = 0x08000000;

    .text :
    {
        _S_TEXT = .;
        *(.vectors*)
        *(.text*)
        . = ALIGN(4);
        _E_TEXT = .;

    }> flash

    .data :
    {
        _S_DATA = .;
        *(.data)
        . = ALIGN(4);
        _E_DATA = .;
    }> ram AT> flash


    .bss :
    {
        _S_BSS = .;
        *(.bss*)
        . = ALIGN(4);
        _E_BSS = .;
    }> ram



    . = . + 0x1000;
    stack_top = .;
}
```


D:\Projects\learn_in_depth_workspace\unit_3_embedded_c\lec3\task_with_c_startup\Map_file.map - Sublime Text (UNREGISTERED)
startup.c
Map_file.map
linker_script.ld
main.c
Ma
Line 30, Column 62

```

Allocating common symbols
Common symbol       size                file

arr2                0x32                main.o

Memory Configuration

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

Linker script and memory map

                0x08000000                . = 0x8000000

.text           0x08000000      0x1c4
                0x08000000                _S_TEXT = .
 *(.vectors*)
 .vectors       0x08000000       0x1c startup.o
                0x08000000                _vectors
 *(.text*)
 .text          0x0800001c       0xf8 main.o
                0x0800001c                main
 .text          0x08000114       0xb0 startup.o
                0x08000114                _reset
                0x080001b8                _default_handler
                0x080001b8                BUS_Handler
                0x080001b8                USAGE_FAULT_Handler
                0x080001b8                HARD_FAULT_Handler
                0x080001b8                NMI_Handler
                0x080001b8                MM_FAULT_Handler
                0x080001c4                . = ALIGN (0x4)
                0x080001c4                _E_TEXT = .

.rodata         0x080001c4       0x1c
 .rodata        0x080001c4       0x1c main.o
                0x080001c4                arr3

.glue_7         0x080001e0        0x0
```



D:\Projects\learn_in_depth_workspace\unit_3_embedded_c\lec3\task_with_c_startup\startup.c - Sublime Text (UNREGISTERED)
startup.c
startup.c
Map_file.map
linker_script.ld
main.c
Makefile
Line 1, Column 1
main 125
Tab Size: 4
C

```c
/*
@Youssef Samy Youssef
*/


#include <stdint.h>

extern uint32_t stack_top;
extern uint32_t _S_TEXT;
extern uint32_t _E_TEXT;
extern uint32_t _S_DATA;
extern uint32_t _E_DATA;
extern uint32_t _S_BSS;
extern uint32_t _E_BSS;

extern void main();
void _default_handler(void);

void _reset();
void NMI_Handler(void) __attribute__((weak, alias("_default_handler")));
void HARD_FAULT_Handler(void) __attribute__((weak, alias("_default_handler")));
void MM_FAULT_Handler(void) __attribute__((weak, alias("_default_handler")));
void BUS_Handler(void) __attribute__((weak, alias("_default_handler")));
void USAGE_FAULT_Handler(void) __attribute__((weak, alias("_default_handler")));


const uint32_t _vectors[] __attribute__((section(".vectors"))) = {
    (uint32_t) &stack_top,
    (uint32_t) &_reset,
    (uint32_t) &NMI_Handler,
    (uint32_t) &HARD_FAULT_Handler,
    (uint32_t) &MM_FAULT_Handler,
    (uint32_t) &BUS_Handler,
    (uint32_t) &USAGE_FAULT_Handler,
};


void _reset(void){
    uint32_t i;

    //copy data from flash to ram
    uint32_t data_size = (uint8_t*)&_E_DATA - (uint8_t*)&_S_DATA;
    uint8_t* P_src = (uint8_t*)&_E_TEXT;
    uint8_t* P_DST = (uint8_t*)&_S_DATA;

    for( i =  0;i<data_size  ; i++){
         P_DST = (uint8_t*) P_src;
        P_DST++;
        P_src++;
    }


    //init bss
    uint32_t bss_size = (uint8_t*)&_E_BSS - (uint8_t*)&_S_BSS;
    P_DST = (uint8_t*)&_S_BSS;

```

    (uint32_t) &HARD_FAULT_Handler,
    (uint32_t) &MM_FAULT_Handler,
    (uint32_t) &BUS_Handler,
    (uint32_t) &USAGE_FAULT_Handler,
};

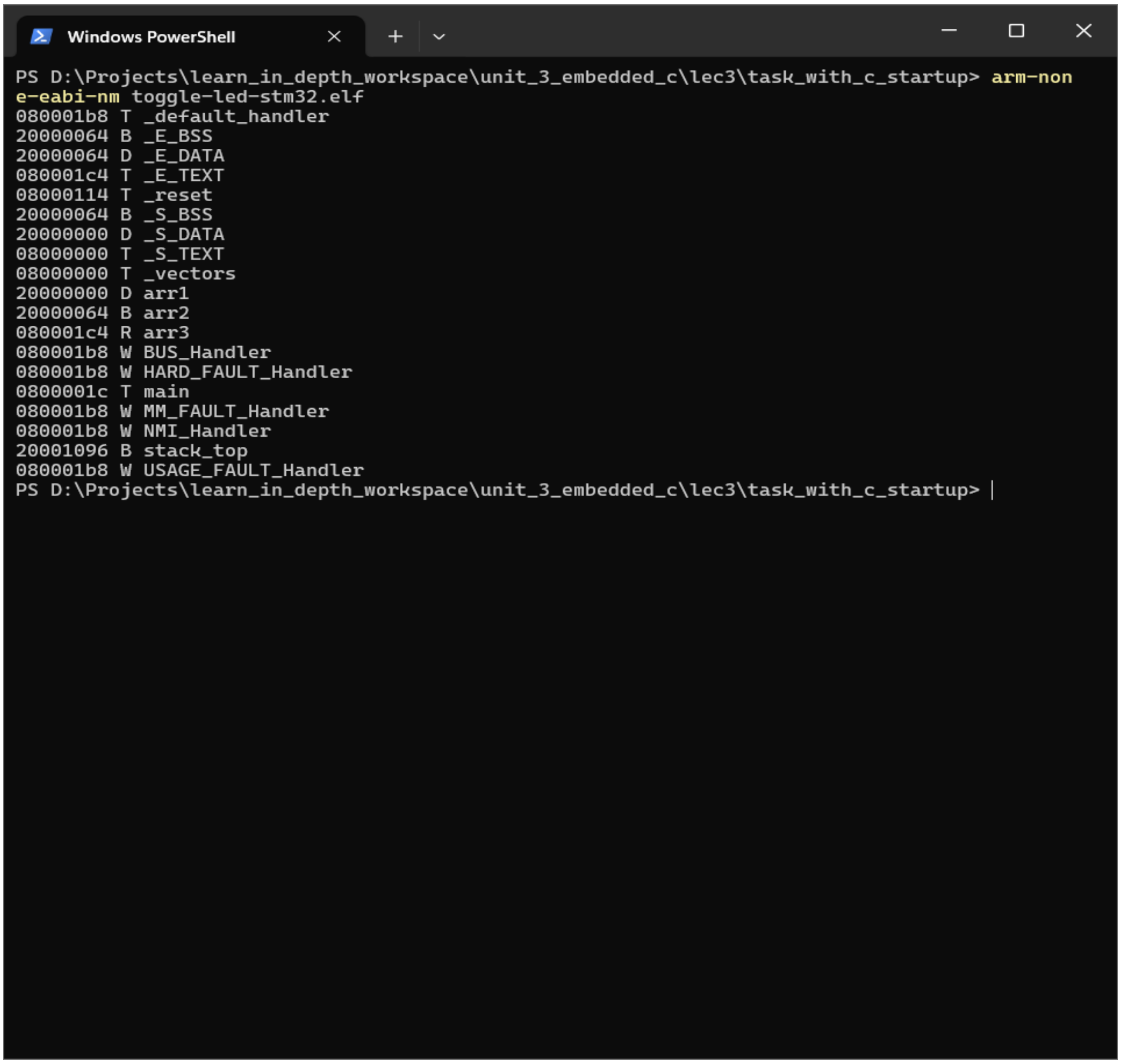
void _reset(void){
    uint32_t i;

    //copy data from flash to ram
    uint32_t data_size = (uint8_t*)&_E_DATA - (uint8_t*)&_S_DATA;
    uint8_t* P_src = (uint8_t*)&_E_TEXT;
    uint8_t* P_DST = (uint8_t*)&_S_DATA;

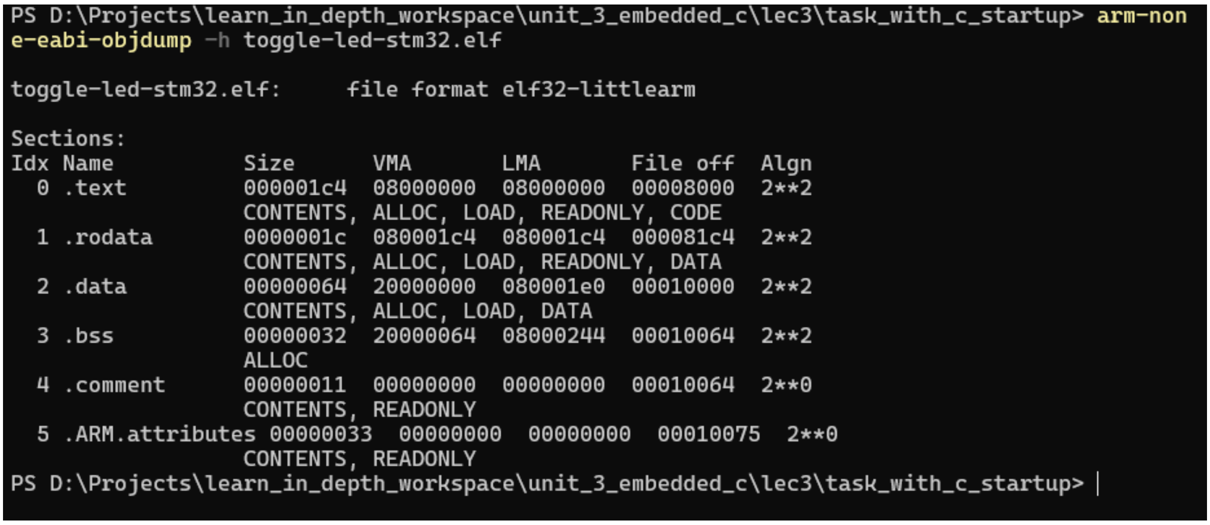
    for(i = 0;i<data_size ; i++){
         P_DST = (uint8_t*) P_src;
        P_DST++;
        P_src++;
    }

    //init bss
    uint32_t bss_size = (uint8_t*)&_E_BSS - (uint8_t*)&_S_BSS;
    P_DST = (uint8_t*)&_S_BSS;

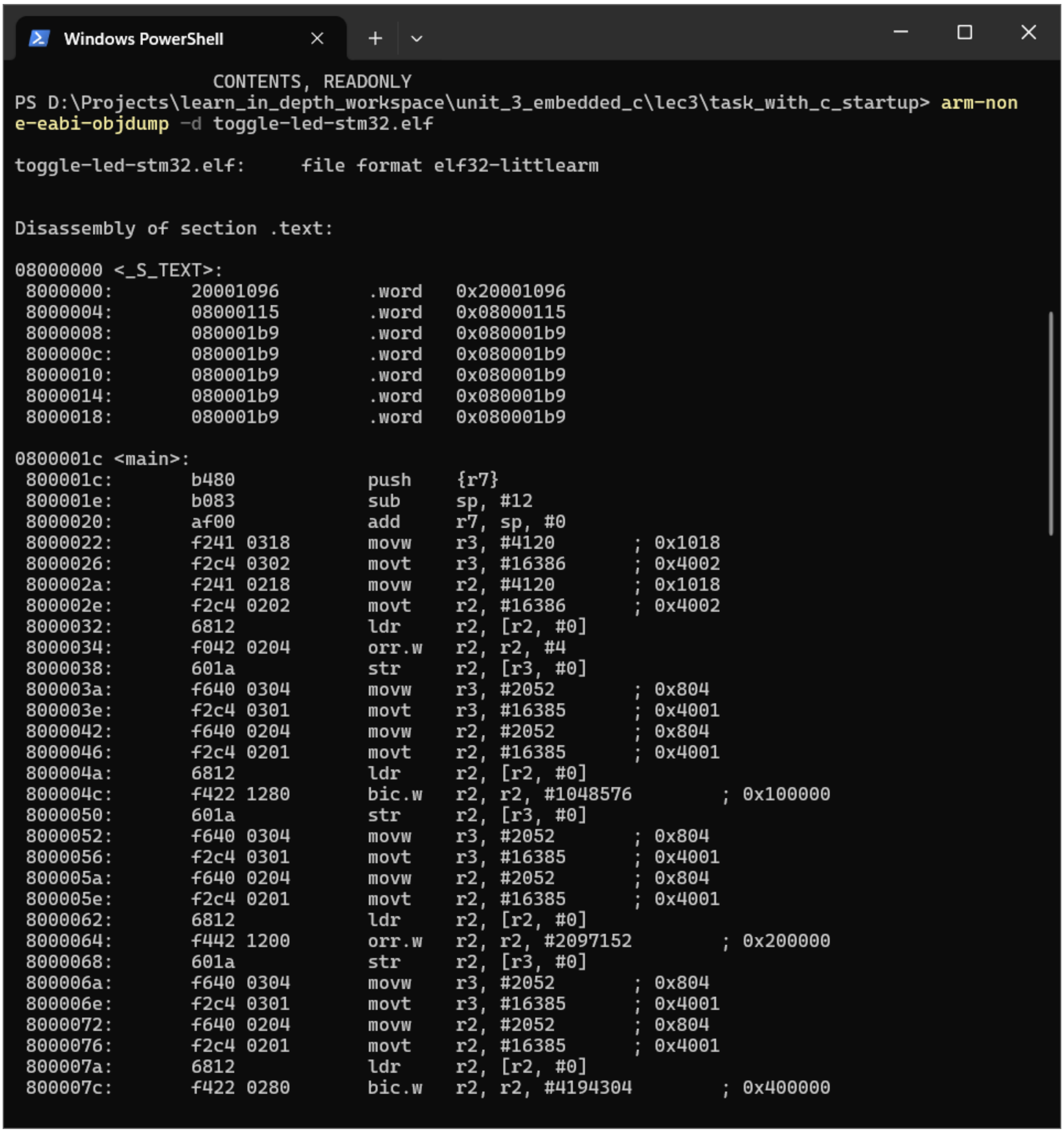
```c
    for( i =  0;i<bss_size  ; i++){
         P_DST = (uint8_t)0;
        P_DST++;
    }

    main();
    while(1);
}

void _default_handler(void){
    _reset();
}
```



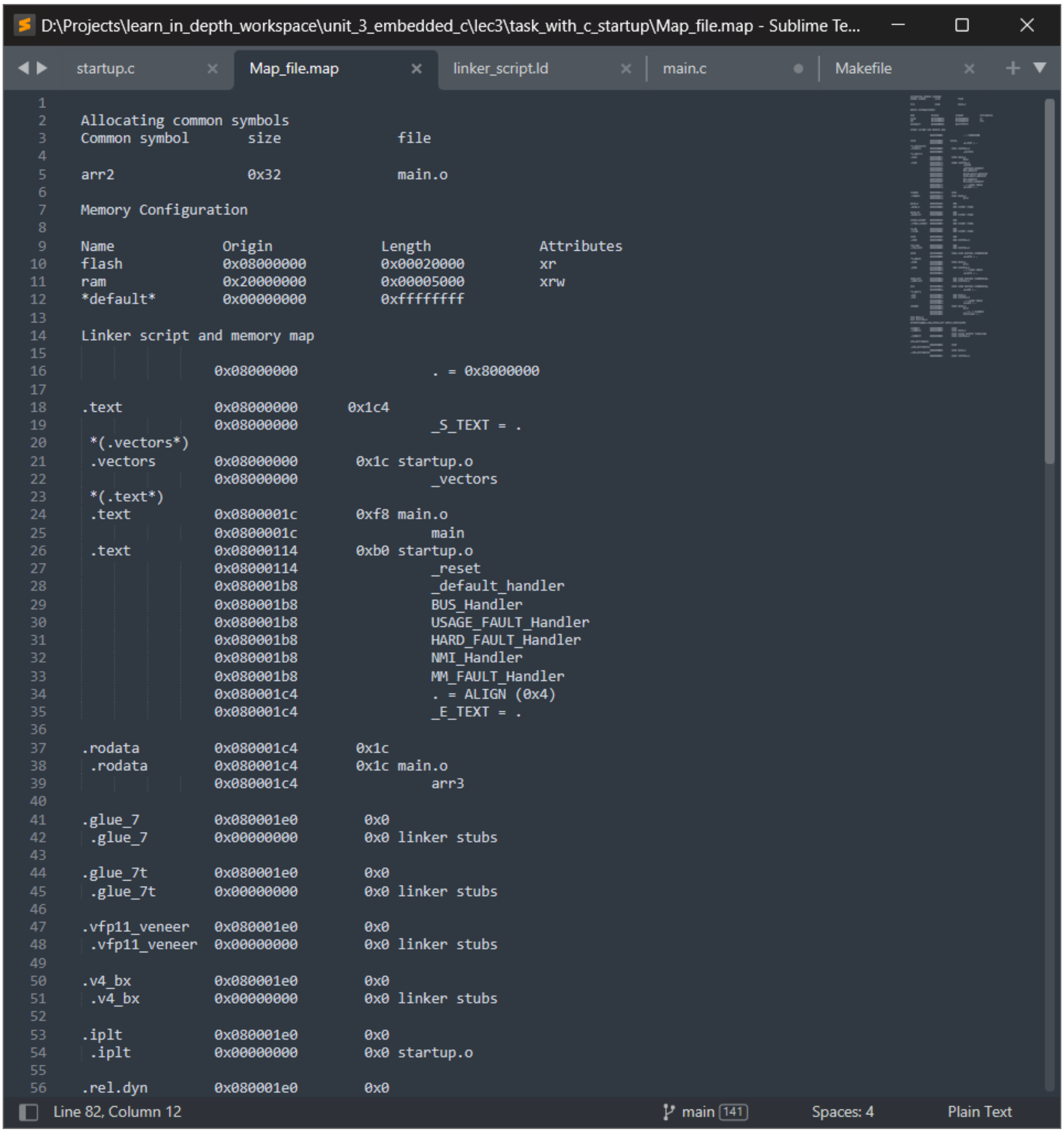
```
PS D:\Projects\learn_in_depth_workspace\unit_3_embedded_c\lec3\task_with_c_startup> arm-none-eabi-nm toggle-led-stm32.elf
080001b8 T _default_handler
20000064 B _E_BSS
20000064 D _E_DATA
080001c4 T _E_TEXT
08000114 T _reset
20000064 B _S_BSS
20000000 D _S_DATA
08000000 T _S_TEXT
08000000 T _vectors
20000000 D arr1
20000064 B arr2
080001c4 R arr3
080001b8 W BUS_Handler
080001b8 W HARD_FAULT_Handler
0800001c T main
080001b8 W MM_FAULT_Handler
080001b8 W NMI_Handler
20001096 B stack_top
080001b8 W USAGE_FAULT_Handler
PS D:\Projects\learn_in_depth_workspace\unit_3_embedded_c\lec3\task_with_c_startup>
```



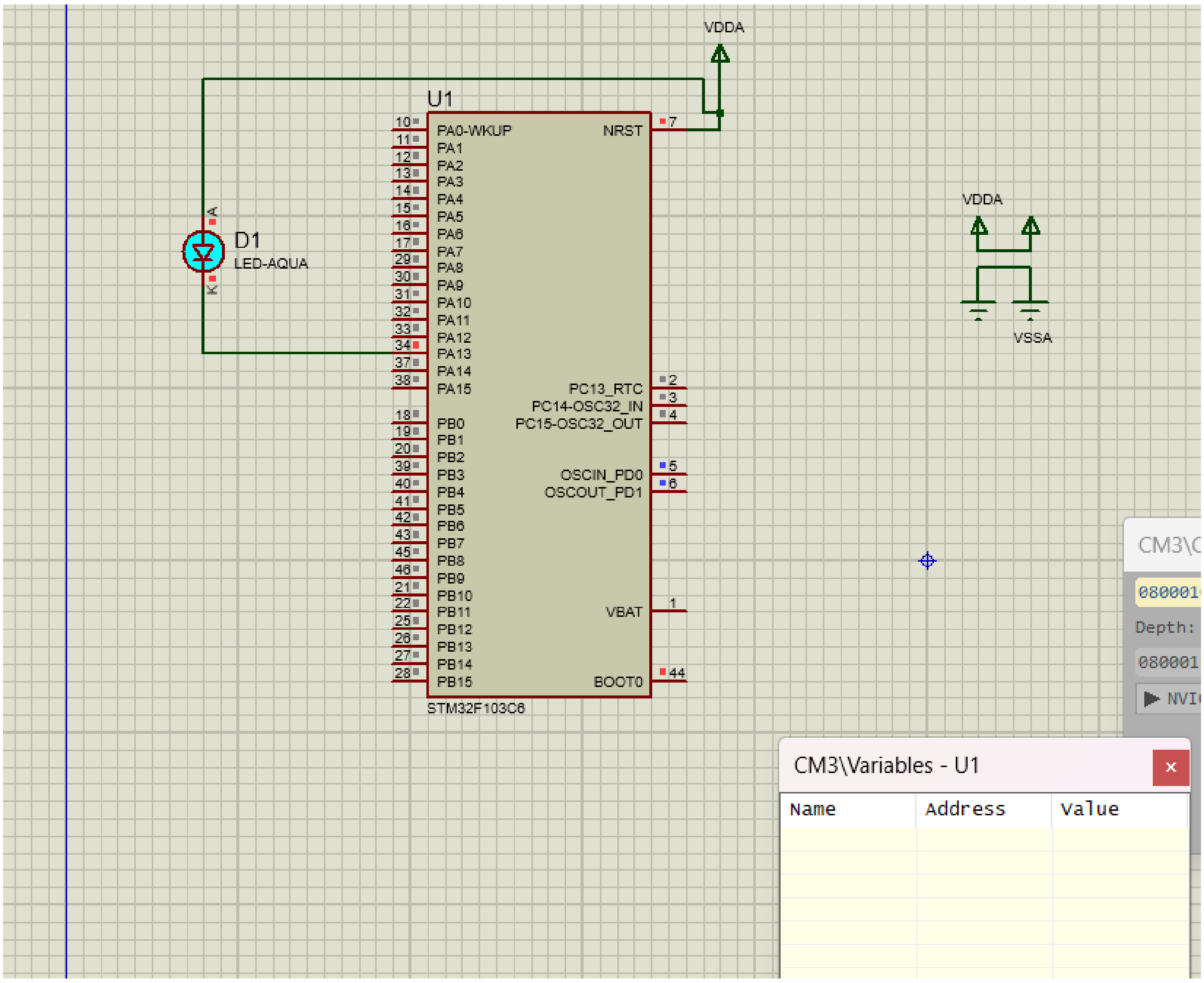
```
PS D:\Projects\learn_in_depth_workspace\unit_3_embedded_c\lec3\task_with_c_startup> arm-none-eabi-objdump -h toggle-led-stm32.elf

toggle-led-stm32.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000001c4  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .rodata       0000001c  080001c4  080001c4  000081c4  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  2 .data         00000064  20000000  080001e0  00010000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  3 .bss          00000032  20000064  08000244  00010064  2**2
                  ALLOC
  4 .comment      00000011  00000000  00000000  00010064  2**0
                  CONTENTS, READONLY
  5 .ARM.attributes 00000033  00000000  00000000  00010075  2**0
                  CONTENTS, READONLY
PS D:\Projects\learn_in_depth_workspace\unit_3_embedded_c\lec3\task_with_c_startup>
```



```
                  CONTENTS, READONLY
PS D:\Projects\learn_in_depth_workspace\unit_3_embedded_c\lec3\task_with_c_startup> arm-non
e-eabi-objdump -d toggle-led-stm32.elf

toggle-led-stm32.elf:     file format elf32-littlearm


Disassembly of section .text:

08000000 <_S_TEXT>:
 8000000:       20001096        .word   0x20001096
 8000004:       08000115        .word   0x08000115
 8000008:       080001b9        .word   0x080001b9
 800000c:       080001b9        .word   0x080001b9
 8000010:       080001b9        .word   0x080001b9
 8000014:       080001b9        .word   0x080001b9
 8000018:       080001b9        .word   0x080001b9

0800001c <main>:
 800001c:       b480            push    {r7}
 800001e:       b083            sub     sp, #12
 8000020:       af00            add     r7, sp, #0
 8000022:       f241 0318       movw    r3, #4120       ; 0x1018
 8000026:       f2c4 0302       movt    r3, #16386      ; 0x4002
 800002a:       f241 0218       movw    r2, #4120       ; 0x1018
 800002e:       f2c4 0202       movt    r2, #16386      ; 0x4002
 8000032:       6812            ldr     r2, [r2, #0]
 8000034:       f042 0204       orr.w   r2, r2, #4
 8000038:       601a            str     r2, [r3, #0]
 800003a:       f640 0304       movw    r3, #2052       ; 0x804
 800003e:       f2c4 0301       movt    r3, #16385      ; 0x4001
 8000042:       f640 0204       movw    r2, #2052       ; 0x804
 8000046:       f2c4 0201       movt    r2, #16385      ; 0x4001
 800004a:       6812            ldr     r2, [r2, #0]
 800004c:       f422 1280       bic.w   r2, r2, #1048576        ; 0x100000
 8000050:       601a            str     r2, [r3, #0]
 8000052:       f640 0304       movw    r3, #2052       ; 0x804
 8000056:       f2c4 0301       movt    r3, #16385      ; 0x4001
 800005a:       f640 0204       movw    r2, #2052       ; 0x804
 800005e:       f2c4 0201       movt    r2, #16385      ; 0x4001
 8000062:       6812            ldr     r2, [r2, #0]
 8000064:       f442 1200       orr.w   r2, r2, #2097152        ; 0x200000
 8000068:       601a            str     r2, [r3, #0]
 800006a:       f640 0304       movw    r3, #2052       ; 0x804
 800006e:       f2c4 0301       movt    r3, #16385      ; 0x4001
 8000072:       f640 0204       movw    r2, #2052       ; 0x804
 8000076:       f2c4 0201       movt    r2, #16385      ; 0x4001
 800007a:       6812            ldr     r2, [r2, #0]
 800007c:       f422 0280       bic.w   r2, r2, #4194304        ; 0x400000
```

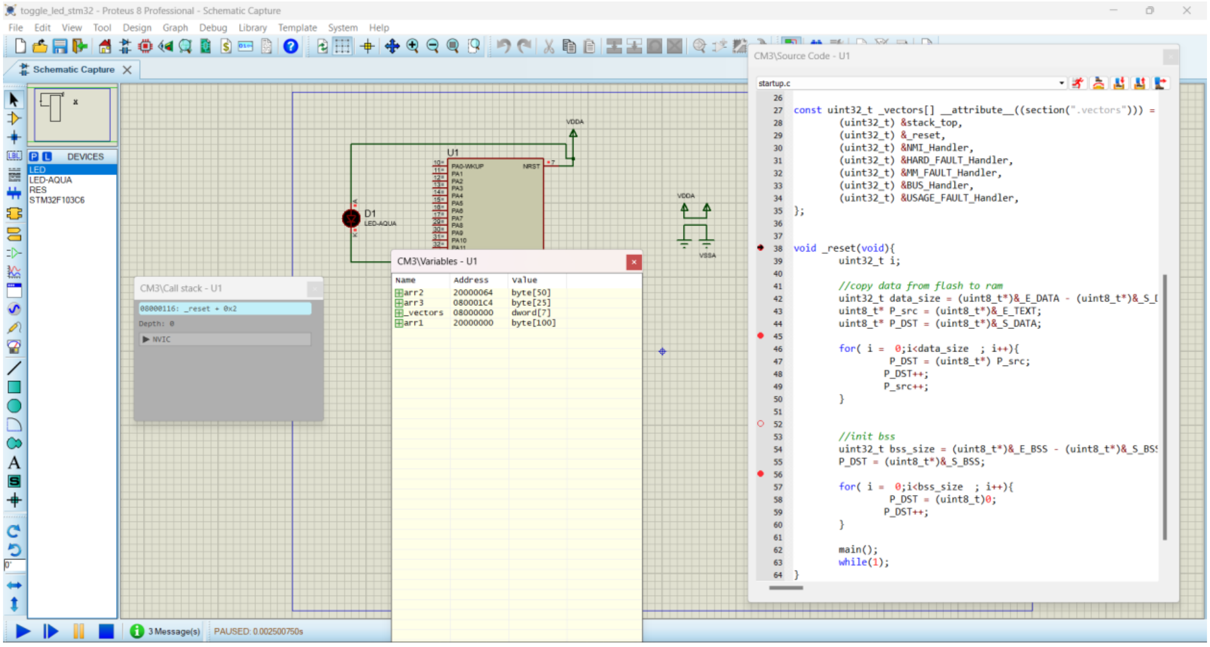

D:\Projects\learn_in_depth_workspace\unit_3_embedded_c\lec3\task_with_c_startup\Map_file.map - Sublime Te...
startup.c
Map_file.map
linker_script.ld
main.c
Makefile
Line 82, Column 12
main 141
Spaces: 4
Plain Text

```

Allocating common symbols
Common symbol       size              file

arr2                0x32              main.o

Memory Configuration

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

Linker script and memory map

                0x08000000                . = 0x8000000

.text           0x08000000      0x1c4
                0x08000000                _S_TEXT = .
 *(.vectors*)
 .vectors       0x08000000       0x1c startup.o
                0x08000000                _vectors
 *(.text*)
 .text          0x0800001c       0xf8 main.o
                0x0800001c                main
 .text          0x08000114       0xb0 startup.o
                0x08000114                _reset
                0x080001b8                _default_handler
                0x080001b8                BUS_Handler
                0x080001b8                USAGE_FAULT_Handler
                0x080001b8                HARD_FAULT_Handler
                0x080001b8                NMI_Handler
                0x080001b8                MM_FAULT_Handler
                0x080001c4                . = ALIGN (0x4)
                0x080001c4                _E_TEXT = .

.rodata         0x080001c4       0x1c
 .rodata        0x080001c4       0x1c main.o
                0x080001c4                arr3

.glue_7         0x080001e0        0x0
 .glue_7        0x00000000        0x0 linker stubs

.glue_7t        0x080001e0        0x0
 .glue_7t       0x00000000        0x0 linker stubs

.vfp11_veneer   0x080001e0        0x0
 .vfp11_veneer  0x00000000        0x0 linker stubs

.v4_bx          0x080001e0        0x0
 .v4_bx         0x00000000        0x0 linker stubs

.iplt           0x080001e0        0x0
 .iplt          0x00000000        0x0 startup.o

.rel.dyn        0x080001e0        0x0
```



CM3\Variables - U1

Name	Address	Value



CM3\Call stack - U1

```
08000116: _reset + 0x2
Depth: 0
▶ NVIC
```

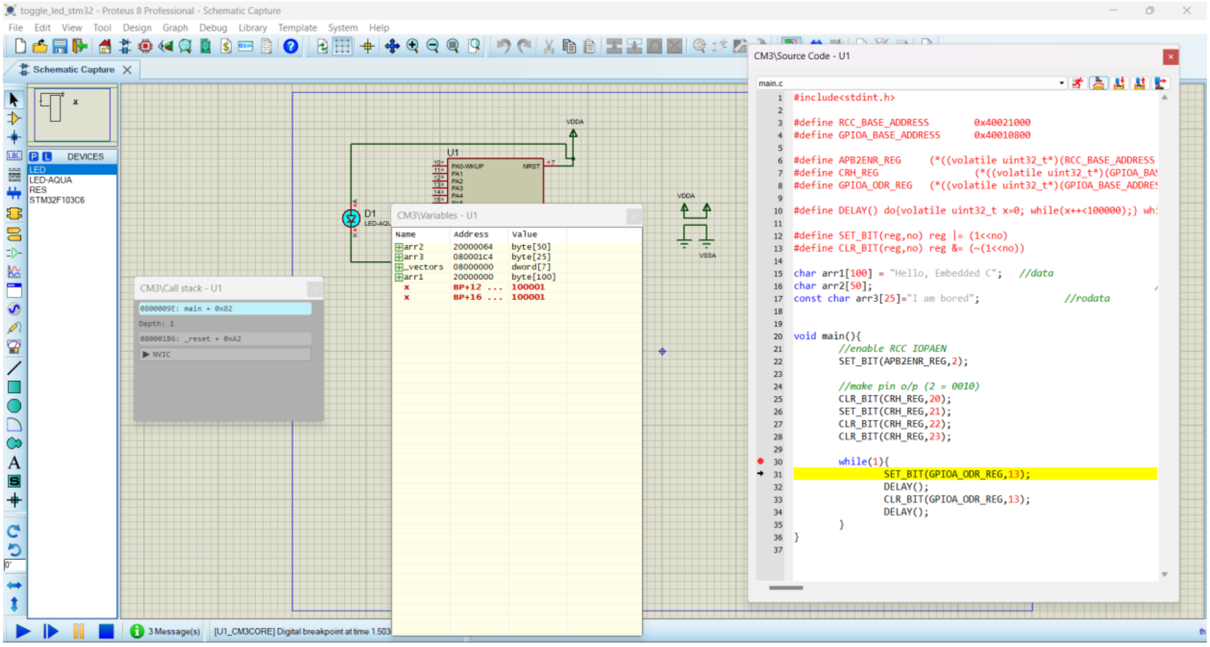
CM3\Variables - U1

Name	Address	Value
⊞arr2	20000064	byte[50]
⊞arr3	080001C4	byte[25]
⊞_vectors	08000000	dword[7]
⊞arr1	20000000	byte[100]

CM3\Source Code - U1

startup.c

```c

const uint32_t _vectors[] __attribute__((section(".vectors"))) =
        (uint32_t) &stack_top,
        (uint32_t) &_reset,
        (uint32_t) &NMI_Handler,
        (uint32_t) &HARD_FAULT_Handler,
        (uint32_t) &MM_FAULT_Handler,
        (uint32_t) &BUS_Handler,
        (uint32_t) &USAGE_FAULT_Handler,
};


void _reset(void){
        uint32_t i;

        //copy data from flash to ram
        uint32_t data_size = (uint8_t*)&_E_DATA - (uint8_t*)&_S_[
        uint8_t* P_src = (uint8_t*)&_E_TEXT;
        uint8_t* P_DST = (uint8_t*)&_S_DATA;

        for( i =  0;i<data_size  ; i++){
                 P_DST = (uint8_t*) P_src;
                P_DST++;
                P_src++;
        }


        //init bss
        uint32_t bss_size = (uint8_t*)&_E_BSS - (uint8_t*)&_S_BS
        P_DST = (uint8_t*)&_S_BSS;

        for( i =  0;i<bss_size  ; i++){
                 P_DST = (uint8_t)0;
                P_DST++;
        }

        main();
        while(1);
}
```



[U1_CM3CORE] Digital breakpoint at time 1.503

CM3\Call stack - U1

```
0800009E: main + 0x82
Depth: 1
080001B6: _reset + 0xA2
▶ NVIC
```

CM3\Variables - U1

Name	Address	Value
⊞arr2	20000064	byte[50]
⊞arr3	080001c4	byte[25]
⊞_vectors	08000000	dword[7]
⊞arr1	20000000	byte[100]
x	BP+12 ...	100001
x	BP+16 ...	100001

CM3\Source Code - U1

main.c

```c
#include<stdint.h>

#define RCC_BASE_ADDRESS        0x40021000
#define GPIOA_BASE_ADDRESS      0x40010800

#define APB2ENR_REG     (*((volatile uint32_t*)(RCC_BASE_ADDRESS
#define CRH_REG                 (*((volatile uint32_t*)(GPIOA_BA
#define GPIOA_ODR_REG   (*((volatile uint32_t*)(GPIOA_BASE_ADDRE

#define DELAY() do{volatile uint32_t x=0; while(x++<100000);} wh

#define SET_BIT(reg,no) reg |= (1<<no)
#define CLR_BIT(reg,no) reg &= (~(1<<no))

char arr1[100] = "Hello, Embedded C";   //data
char arr2[50];                                              /
const char arr3[25]="I am bored";               //rodata


void main(){
        //enable RCC IOPAEN
        SET_BIT(APB2ENR_REG,2);

        //make pin o/p (2 = 0010)
        CLR_BIT(CRH_REG,20);
        SET_BIT(CRH_REG,21);
        CLR_BIT(CRH_REG,22);
        CLR_BIT(CRH_REG,23);

        while(1){
                SET_BIT(GPIOA_ODR_REG,13);
                DELAY();
                CLR_BIT(GPIOA_ODR_REG,13);
                DELAY();
        }
}
```