

LAB 4:

- In this lab the code is going to run on TivaC board of the following properties:
 - Processor ARM CortexM4.
 - SRAM 32KB.
 - Flash Memory 256KB.
- First editing (main.c and startup.c):

main.c: defined GPIO Port F registers' addresses and initialized GPIO Port F.

```
1  #include "Platform_Types.h"
2
3  #define SYSCTL_RCG2_R      *((volatile uint32*)0x400FE108)
4  #define GPIO_PORTF_DIR_R  0x40025400
5  #define GPIO_PORTF_DEN_R  0x4002551C
6  #define GPIO_PORTF_DATA_R 0x400253FC
7
8
9  typedef union{
10     volatile uint32 ALL_FIELDS;
11     struct {
12         volatile uint32 :3;
13         volatile uint32 pin3:1;
14     }pins;
15 }GPIO_PF_Rs;
16
17
18
19 volatile GPIO_PF_Rs * dir_R= (volatile GPIO_PF_Rs *) (GPIO_PORTF_DIR_R);
20 volatile GPIO_PF_Rs * den_R= (volatile GPIO_PF_Rs *) (GPIO_PORTF_DEN_R);
21 volatile GPIO_PF_Rs * data_R= (volatile GPIO_PF_Rs *) (GPIO_PORTF_DATA_R);
22
23 static uint32 i;
24
25 int main(void)
26 {
27     SYSCTL_RCG2_R=0x20;
28
29     for(i=0;i<200;i++);
30     dir_R->pins.pin3=1;
31     den_R->pins.pin3=1;
32
33     while(1){
34         data_R->pins.pin3=1;
35         for(i=0;i<200000;i++);
36         data_R->pins.pin3=0;
37         for(i=0;i<200000;i++);
38     }
39 }
40
```

startup.c: defined stack_top without using the linker script and edited the vectors to be array of pointers to functions instead of array of integers.

```
1  /*startup.c Mostafa Mahmoud*/
2  #include "Platform_Types.h"
3
4  extern int main(void);
5  void Reset_Handler();
6  void Default_Handler(){
7      Reset_Handler();
8  }
9  void NMI_Handler() __attribute__((weak,alias("Default_Handler")));;
10 void H_fault_Handler() __attribute__((weak,alias("Default_Handler")));;
11
12 extern uint32 _E_text;
13 extern uint32 _S_data;
14 extern uint32 _E_data;
15 extern uint32 _S_bss;
16 extern uint32 _E_bss;
17
18 static uint32 stack_top[256];
19
20 void (* const vectors[])() __attribute__((section(".vectors"))) = {
21
22     (void(*)())((uint32) stack_top+sizeof(stack_top)),
23     Reset_Handler,
24     NMI_Handler,
25     H_fault_Handler,
26 };
27 void Reset_Handler(){
28     /*copy data section*/
29     uint32 i;
30     uint32 DATA_size= (uint8*)&_E_data-(uint8*)&_S_data;
31     uint8 *P_src =(uint8*)&_E_text;
32     uint8 *P_dst =(uint8*)&_S_data;
33     for(i=0;i<DATA_size;i++){
34         *((uint8*)P_dst++)=*((uint8*)P_src++);
35     }
36     /*init bss section*/
37     uint32 BSS_size=(uint8*)&_E_bss-(uint8*)&_S_bss;
38     P_dst=(uint8*)&_S_bss;
39     for(i=0;i<BSS_size;i++){
40         *((uint8*)P_dst++)=(uint8)0;
41     }
42     /*call main()*/
43     main();
44 }
```

- Editing linker_script.ld:

linker_script.ld: just removed the stack_top symbol and edited the addresses of SRAM and Flash.

```
1  /*linker script CortexM4
2  Mostafa Mahmoud */
3
4  MEMORY
5  {
6  flash (rx) : ORIGIN = 0x00000000, LENGTH = 256k
7  sram (rwx) : ORIGIN = 0x20000000, LENGTH = 32k
8  }
9
10 SECTIONS{
11
12     .text :
13     {
14         *(.vectors*)
15         *(.text*)
16         *(.rodata*)
17         _E_text = . ;
18     }>flash
19
20     .data :
21     {
22         _S_data = . ;
23         *(.data)
24         . = ALIGN(4) ;
25         _E_data = . ;
26     }> sram AT> flash
27
28     .bss :
29     {
30         _S_bss = . ;
31         *(.bss*)
32         . = ALIGN(4);
33         _E_bss = . ;
34     }>sram
35
36 }
```

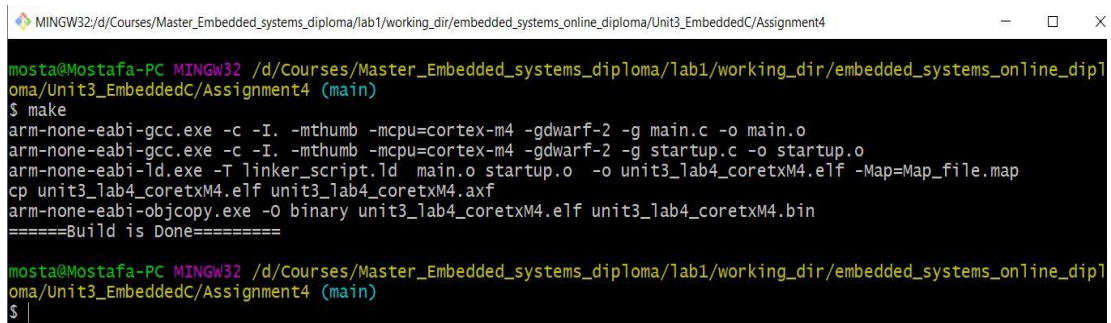
- Now editing Makefile:

```

1  #@copyright : Mostafa Mahmoud
2  CC=arm-none-eabi-
3  CFLAGS=-mthumb -mcpu=cortex-m4 -gdwarf-2 -g
4  INCS=-I.
5  LIBS=
6  SRC= $(wildcard *.c)
7  OBJ= $(SRC:.c=.o)
8  As= $(wildcard *.s)
9  AsOBJ= $(As:.s=.o)
10 Project_name=unit3_lab4_coretxM4
11
12
13
14 all: $(Project_name).bin
15     @echo "=====Build is Done======"
16
17
18 %.o: %.c
19     $(CC)gcc.exe -c $(INCS) $(CFLAGS) $< -o $@
20
21 $(Project_name).elf: $(OBJ) $(AsOBJ)
22     $(CC)ld.exe -T linker_script.ld $(LIBS) $(OBJ) $(AsOBJ) -o $@ -Map=Map_file.map
23     cp $(Project_name).elf $(Project_name).axf
24
25 $(Project_name).bin: $(Project_name).elf
26     $(CC)objcopy.exe -O binary $< $@
27
28
29 clean_all:
30     rm *.o *.elf *.bin
31
32 clean:
33     rm *.elf *.bin

```

- Building files:



```

MINGW32; d/Courses/Master_Embedded_systems_diploma/lab1/working_dir/embedded_systems_online_diploma/Unit3_EmbeddedC/Assignment4
mosta@Mostafa-PC MINGW32 /d/Courses/Master_Embedded_systems_diploma/lab1/working_dir/embedded_systems_online_diploma/Unit3_EmbeddedC/Assignment4 (main)
$ make
arm-none-eabi-gcc.exe -c -I. -mthumb -mcpu=cortex-m4 -gdwarf-2 -g main.c -o main.o
arm-none-eabi-gcc.exe -c -I. -mthumb -mcpu=cortex-m4 -gdwarf-2 -g startup.c -o startup.o
arm-none-eabi-ld.exe -T linker_script.ld main.o startup.o -o unit3_lab4_coretxM4.elf -Map=Map_file.map
cp unit3_lab4_coretxM4.elf unit3_lab4_coretxM4.axf
arm-none-eabi-objcopy.exe -O binary unit3_lab4_coretxM4.elf unit3_lab4_coretxM4.bin
=====Build is Done=====

mosta@Mostafa-PC MINGW32 /d/Courses/Master_Embedded_systems_diploma/lab1/working_dir/embedded_systems_online_diploma/Unit3_EmbeddedC/Assignment4 (main)
$ make clean
rm *.o *.elf *.bin

```

- Checking symbols in main.o, startup.o and final executable .elf:

MINGW32:/d/Courses/Master_Embedded_systems_diploma/lab1/working_dir/embedded_systems_online_diploma/Unit3_EmbeddedC/Assignment4

```
mosta@Mostafa-PC MINGW32 /d/Courses/Master_Embedded_systems_diploma/lab1/working_dir/embedded_systems_online_diploma/Unit3_EmbeddedC/Assignment4 (main)
$ arm-none-eabi-nm.exe main.o
00000008 D data_R
00000004 D den_R
00000000 D dir_R
00000000 b i
00000000 T main

mosta@Mostafa-PC MINGW32 /d/Courses/Master_Embedded_systems_diploma/lab1/working_dir/embedded_systems_online_diploma/Unit3_EmbeddedC/Assignment4 (main)
$ arm-none-eabi-nm.exe startup.o
                 U _E_bss
                 U _E_data
                 U _E_text
                 U _S_bss
                 U _S_data
00000000 T Default_Handler
00000000 W H_fault_Handler
                 U main
00000000 W NMI_Handler
0000000c T Reset_Handler
00000000 b stack_top
00000000 R vectors

mosta@Mostafa-PC MINGW32 /d/Courses/Master_Embedded_systems_diploma/lab1/working_dir/embedded_systems_online_diploma/Unit3_EmbeddedC/Assignment4 (main)
$ arm-none-eabi-nm.exe unit3_lab4_coretxM4.elf
20000410 B _E_bss
2000000c D _E_data
000001e0 T _E_text
2000000c B _S_bss
20000000 D _S_data
20000008 D data_R
00000120 T Default_Handler
20000004 D den_R
20000000 D dir_R
00000120 W H_fault_Handler
2000000c b i
00000010 T main
00000120 W NMI_Handler
0000012c T Reset_Handler
20000010 b stack_top
00000000 T vectors
```


- Simulating and debugging on KEIL:

The screenshot shows the KEIL uVision IDE interface during a simulation. The main window displays the Logic Analyzer, which shows a square wave signal on the PORTF pin. The signal is high for most of the time and low for a short duration. The Disassembly window shows the main function code, which includes a loop that toggles the PORTF pin. The Property window shows the values of various registers and peripherals, including the DATA register (0x00000019), DIR register (0x00000008), and the LOCK register (0x01010101).

This screenshot is similar to the one above, showing the KEIL uVision IDE interface during a simulation. The Logic Analyzer window displays a square wave signal on the PORTF pin. The Disassembly window shows the main function code, which includes a loop that toggles the PORTF pin. The Property window shows the values of various registers and peripherals, including the DATA register (0x00000019), DIR register (0x00000008), and the LOCK register (0x01010101). The simulation time is slightly different from the first screenshot, indicating a continuation of the simulation.