

Embedded C Lab3 Report

In this lab we will simulate and debug code on TevaC kit that has TM4C123GH6PM SOC and ARM-CortexM4 processor.

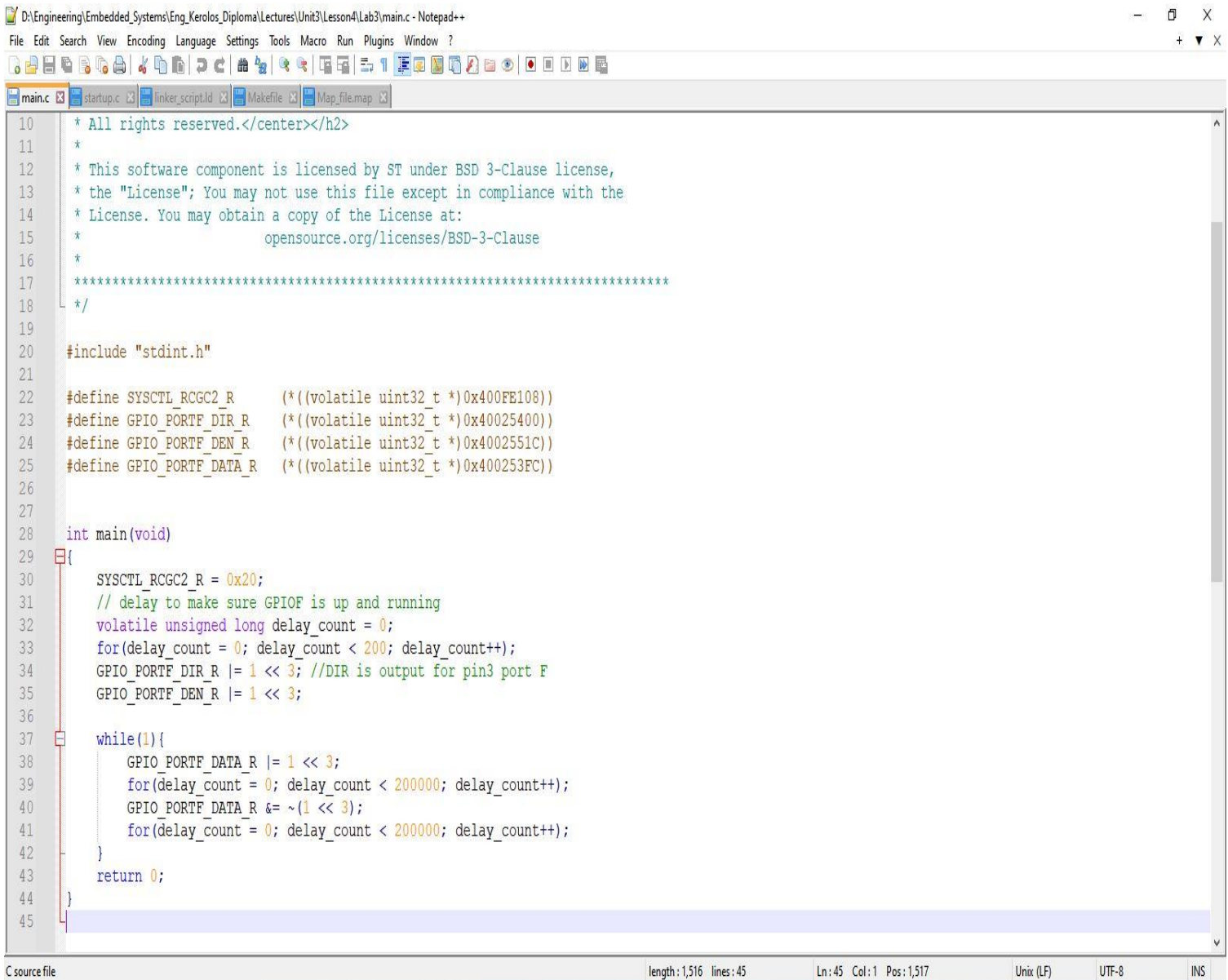
The aim of this lab is to toggle a LED connected to pin3 of PORTF.

We will write Main.c , Startup.c, linker script and make file from scratch.

According to the data sheet:

- Flash memory starts with address 0x00000000 and has size of 512M.
 - Sram memory starts 0x20000000 and has size of 512M.
 - SYSCTL is system control module that we will use to enable clock for PORTF has base address of 0x400FE000
 - SYSCTL_RCGC2_R has offset address of 0x108 under SYSCTL we will assign this register with value of 0x00000020 to enable clock for PORTF
 - GPIO module has base address of 0x40025000, and we will use three registers inside
- First GPIO_PORTF_DIR_R has offset of 0x400, and we will assign value of 1 in pin3 to define this pin as an output
- First GPIO_PORTF_DEN_R has offset of 0x51C, and we will assign value of 1 in pin3 to enable this pin
- GPIO_PORTF_DR_R has offset of 0x400, and we will assign value of 1 in pin3 and 0 to toggle the output.

Main.c



```
10  * All rights reserved.</center></h2>
11  *
12  * This software component is licensed by ST under BSD 3-Clause license,
13  * the "License"; You may not use this file except in compliance with the
14  * License. You may obtain a copy of the License at:
15  *      opensource.org/licenses/BSD-3-Clause
16  *
17  *****
18  */
19
20  #include "stdint.h"
21
22  #define SYSTCL_RCGC2_R      (*((volatile uint32_t *)0x400FE108))
23  #define GPIO_PORTF_DIR_R   (*((volatile uint32_t *)0x40025400))
24  #define GPIO_PORTF_DEN_R   (*((volatile uint32_t *)0x4002551C))
25  #define GPIO_PORTF_DATA_R  (*((volatile uint32_t *)0x400253FC))
26
27
28  int main(void)
29  {
30      SYSTCL_RCGC2_R = 0x20;
31      // delay to make sure GPIOF is up and running
32      volatile unsigned long delay_count = 0;
33      for(delay_count = 0; delay_count < 200; delay_count++);
34      GPIO_PORTF_DIR_R |= 1 << 3; //DIR is output for pin3 port F
35      GPIO_PORTF_DEN_R |= 1 << 3;
36
37      while(1){
38          GPIO_PORTF_DATA_R |= 1 << 3;
39          for(delay_count = 0; delay_count < 200000; delay_count++);
40          GPIO_PORTF_DATA_R &= ~(1 << 3);
41          for(delay_count = 0; delay_count < 200000; delay_count++);
42      }
43      return 0;
44  }
45
```

C source file length: 1,516 lines: 45 Ln: 45 Col: 1 Pos: 1,517 Unix (LF) UTF-8 INS

Make file :

-we will make some changes on make file : project name and we will copy a .axf file to run on kiel microvision simulator and change the processor name to cortex-M4.

```
D:\Engineering\Embedded_Systems\Eng_Kerolos_Diploma\Lectures\Unit3\Lesson4\Lab3\Makefile - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
main.c startup.c linker_script.ld Makefile Map_file.map
1  #@Copyright : Abdo Halem
2
3  CC=arm-none-eabi
4  CFLAGS= -gdwarf-2 -mcpu=cortex-m4 -g
5  INCS=-I .
6  LIBS=
7  SRC = $(wildcard *.c)
8  OBJ = $(SRC:.c=.o)
9  #As = $(wildcard *.s)
10 #AsOBJ = $(As:.s=.o)
11 Project_name=Unit3_Lesson4_Lab3_cortex_m4
12
13 all: $(Project_name).bin
14     @echo "=====Build is Done=====
15
16 #startup.o: startup.s
17 # $(CC)-as $(CFLAGS) $< -o $@
18
19 %.o: %.c
20     $(CC)-gcc -c $(CFLAGS) -mthumb $< -o $@
21
22 $(Project_name).elf: $(OBJ) $(AsOBJ)
23     $(CC)-ld -T linker_script.ld $(LIBS) $(OBJ) $(AsOBJ) -o $@ -Map=Map_file.map
24     cp $(Project_name).elf $(Project_name).axf
25
26 $(Project_name).bin: $(Project_name).elf
27     $(CC)-objcopy -O binary $< $@
28
29 clean_all:
30     rm *.o *.elf *.bin
31
32 clean:
33     rm *.elf *.bin
34
length: 726 lines: 34 Ln: 22 Col: 37 Pos: 461 Windows (CR LF) UTF-8 INS
Type here to search 30°C غائم غاليًا 11:25 PM 6/24/2024
```

Startup.c :

In this lab we will use a new approach by initialize SP in Startup.c
Instead of creating its symbol in linker script. Our scope here to fix SP
after 1024 byte of .bss section.

We will use an uninitialized array of integers with 256 elements.

The total size of array will be 1024 byte and this is where SP will be
at the end of the array.

Then we will make an array of pointers to functions take nothing and

return void these pointers will points to each function that will handle its relative interrupt according to interrupt vector table.

```
D:\Engineering\Embedded_Systems\Eng_Kerolos_Diploma\Lectures\Unit3\Lesson4\Lab3\startup.c - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
main.c startup.c linker_script.ld Makefile Map_file.map
1  /* startup.c: Eng Abdo Halem */
2
3  #include <stdint.h>
4  extern int main();
5  void Reset_Handler();
6
7  void Default_Handler(){
8      Reset_Handler();
9  }
10
11  void NMI_Handler() __attribute__((weak, alias("Default_Handler")));
12  void H_Fault_Handler() __attribute__((weak, alias("Default_Handler")));
13
14  // Booking 1024 byte located by .bss through uninitialized array of 256 int (256*4 = 1024)
15  static unsigned int stack_top[256];
16
17  void (* const g_p_fn_Vectors[])() __attribute__((section(".vectors"))) = {
18      (void (*)())((unsigned long)stack_top + sizeof(stack_top)),
19      &Reset_Handler,
20      &NMI_Handler,
21      &H_Fault_Handler
22  };
23
24  extern unsigned int _E_text;
25  extern unsigned int _S_DATA;
26  extern unsigned int _E_DATA;
27  extern unsigned int _S_bss;
28  extern unsigned int _E_bss;
29
30  void Reset_Handler(){
31      // Copy data section from flash to ram
32      unsigned int DATA_size = (unsigned char*)&_E_DATA - (unsigned char*)&_S_DATA;
33      unsigned char* P_src = (unsigned char*)&_E_text;
34      unsigned char* P_dst = (unsigned char*)&_S_DATA;
35      unsigned int i = 0;
36      for(i = 0; i < DATA_size; i++){
37          *((unsigned char*)P_dst++) = *((unsigned char*)P_src++);
38      }
39  }
```

C source file length: 1,462 lines: 47 Ln: 19 Col: 20 Pos: 583 Windows (CR LF) UTF-8 INS

Type here to search 30°C غائم غائما 11:26 PM 6/24/2024

```
D:\Engineering\Embedded_Systems\Eng_Kerolos_Diploma\Lectures\Unit3\Lesson4\Lab3\startup.c - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
main.c startup.c linker_script.ld Makefile Map_file.map
13
14 // Booking 1024 byte located by .bss through uninitialized array of 256 int (256*4 = 1024)
15 static unsigned int stack_top[256];
16
17 void (* const g_p_fn_Vectors[])() __attribute__((section(".vectors"))) = {
18     (void (*)())((unsigned long)stack_top + sizeof(stack_top)),
19     &Reset_Handler,
20     &NMI_Handler,
21     &H_Fault_Handler
22 };
23
24 extern unsigned int _E_text;
25 extern unsigned int _S_DATA;
26 extern unsigned int _E_DATA;
27 extern unsigned int _S_bss;
28 extern unsigned int _E_bss;
29
30 void Reset_Handler(){
31     // Copy data section from flash to ram
32     unsigned int DATA_size = (unsigned char*)&_E_DATA - (unsigned char*)&_S_DATA;
33     unsigned char* P_src = (unsigned char*)&_E_text;
34     unsigned char* P_dst = (unsigned char*)&_S_DATA;
35     unsigned int i = 0;
36     for(i = 0; i < DATA_size; i++){
37         *((unsigned char*)P_dst++) = *((unsigned char*)P_src++);
38     }
39     // Initialize .bss section in SRAM
40     unsigned int bss_size = (unsigned char*)&_E_bss - (unsigned char*)&_S_bss;
41     P_dst = (unsigned char*)&_S_bss;
42     for(i = 0; i < bss_size; i++){
43         *((unsigned char*)P_dst++) = (unsigned char)0;
44     }
45     // Jump to main()
46     main();
47 }
```

C source file length: 1,462 lines: 47 Ln: 19 Col: 20 Pos: 583 Windows (CR LF) UTF-8 INS 30°C غائم 11:26 PM 6/24/2024

Linker script:

We will just edit the sizes of memory sections and delete stack top symbol.


```
D:\Engineering\Embedded_Systems\Eng_Kerolos_Diploma\Lectures\Unit3\Lesson4\Lab3\linker_script.ld - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
main.c startup.c linker_script.ld Makefile Map_file.map
1 /* linker script cortexM4
2 Eng Abdo Halem */
3
4 MEMORY
5 {
6 flash(RX) : ORIGIN = 0x00000000 , LENGTH = 512M
7 sram(RWX) : ORIGIN = 0x20000000 , LENGTH = 512M
8 }
9
10 SECTIONS
11 {
12     .text : {
13         *(.vectors*)
14         *(.text*)
15         *(.rodata)
16         _E_text = . ;
17     }> flash
18
19     .data : {
20         _S_DATA = . ;
21         *(.data)
22         . = ALIGN(4);
23         _E_DATA = . ;
24     }> sram AT> flash
25
26     .bss : {
27         _S_bss = . ;
28         *(.bss*)
29         _E_bss = . ;
30     }> sram
31 }
32 }
```

Map file:

.bss section starts with address of 0x20000000 and ends with 0x20000400 that has been incremented by 0x400 that equivalent to 1024 in decimal.

```

.data          0x20000000      0x0 load address 0x000001a4
               0x20000000      _S_DATA = .
*(.data)
.data          0x20000000      0x0 main.o
.data          0x20000000      0x0 startup.o
               0x20000000      . = ALIGN (0x4)
               0x20000000      _E_DATA = .

.igot.plt      0x20000000      0x0 load address 0x000001a4
.igot.plt      0x00000000      0x0 main.o

.bss           0x20000000      0x400 load address 0x000001a4
               0x20000000      _S_bss = .
*(.bss*)
.bss           0x20000000      0x0 main.o
.bss           0x20000000      0x400 startup.o
               0x20000400      _E_bss = .
LOAD main.o
LOAD startup.o
OUTPUT(Unit3_Lesson4_Lab3_cortex_m4.elf elf32-littlearm)

```

- Flash starts with 0x00000000 and the first section is .vectors Section.

```

2  Memory Configuration
3
4  Name          Origin          Length          Attributes
5  flash         0x00000000      0x20000000      xr
6  sram          0x20000000      0x20000000      xrw
7  *default*     0x00000000      0xffffffff
8
9  Linker script and memory map
10
11
12 .text          0x00000000      0x1a4
13 *(.vectors*)
14 .vectors       0x00000000      0x10 startup.o
15               0x00000000      g_p_fn_Vectors
16 *(.text*)
17 .text          0x00000010      0xd0 main.o
18               0x00000010      main
19 .text          0x000000e0      0xc4 startup.o
20               0x000000e0      H_Fault_Handler
21               0x000000e0      Default_Handler
22               0x000000e0      NMI_Handler
23               0x000000ec      Reset_Handler
24 *(.rodata)
25               0x000001a4      _E_text = .
26

```

Here we can see the LED blinking using logic analyzer at kiel simulator.

The screenshot displays the Keil uVision IDE interface during a simulation. The main window is divided into several panes:

- Registers:** Shows the state of various registers. R15 (PC) is highlighted at address 0x00000088, and xPSR is at 0x21000000.
- Logic Analyzer:** Displays a waveform for PORTF, showing a square wave between 0 and 1. The time scale is 0.796 s, d. 0.796 s. The signal is labeled 'PORTF'.
- GPIOF:** Shows the configuration for the GPIOF peripheral. The DATA register is 0x08080819, and the DIR register is 0x08080808.
- Code Editor:** Contains the C code for the main function. The code defines GPIO_PORTF_DATA_R and implements a while loop that toggles the LED state.
- Hardware Diagram:** A schematic diagram of the TM4C123 microcontroller. It shows the connection of the LED to the PF0 pin, which is configured as an output. The LED is labeled 'LED Green'.
- Port F Registers:** A table showing the configuration for the Port F registers. The DATA register is 0x19, DIR is 0x08, DEN is 0x08, and RCGC2 is 0x00000020.
- Call Stack + Locals:** Shows the current function 'main' at location 0x00000010, with a local variable 'delay_count' at 0x003D040.

The bottom status bar indicates the simulation is running at 11:39:40.0000 sec, with a clock speed of 141 C1. The system temperature is 31°C.