

A dark blue vertical bar on the left side of the page. A blue arrow points to the right from this bar, containing the date.

8/22/2023

Lab3 Report

Lab3 (Toggling green led on TM4C123GH6PZ)

Learn-In-Depth Diploma

Several thin, curved, light blue lines that sweep upwards from the bottom left towards the center of the page.

Name: Andrew Adel Hosny Goued

Position: Student at Faculty of Engineering

Ain Shams University

Computer and System Engineering

Contents

Introduction.....	2
TM4C123GH6PZ information	2
Main file	3
Main.c.....	3
Symbols.....	3
Startup file	4
Startup.c	4
Symbols.....	5
Linker_script.ld.....	6
Makefile	7
symbols	8
Map_file.map	8

Introduction

In this lab we will simulate and debug application code on TivaC kit with tm4c123gh6pz and arm-cortexM4 processor family.

The application is toggling green Led (pin_3 in PortF).

We will write from scratch: main.c, startup.c, makefile.

TM4C123GH6PZ information

Flash memory occupies addresses from 0x00000000 to 0x20000000.

SRAM memory occupies addresses from 0x20000000 to 0x40000000.

SYSCTL register is register control enabling and disabling clock on each register in system, it has address 0x400FE000

To enable portf we need to assign 0x20 to address away from 0x400FE000 by offset 0x108.

Then we need to define the direction of pin3 as output, we define direction by putting 1 on bit3 on register GPIO_PORTF_DIR_R, which has address 0x40025000 and offset 0x400.

Then enable the pin3 by putting 1 on bit3 of register GPIO_PORTF_DEN_R which has address 0x40025000 and offset 0x51C.

Finally to turn on and off of led we put 1 and 0 respectively on register GPIO_PORTF_DR_R that has address 0x40025000 and offset 0x3FC.

Main file

Main.c

```
/**
*****
* @file           : main.c
* @author          : Andrew Adel
* @brief           : Main program body
**/
#include "platform_types.h"

#define SYSCTL_RCGC2_R (*(volatile uint32*) 0x400FE108)
#define GPIO_PORTF_DIR_R (*(volatile uint32*) 0x40025400)
#define GPIO_PORTF_DEN_R (*(volatile uint32*) 0x4002551C)
#define GPIO_PORTF_DATA_R (*(volatile uint32*) 0x400253FC)

int main(){
    SYSCTL_RCGC2_R = 0x20;
    //delay to ensure gpiof is up and running
    volatile uint32 delay_counter;
    for (delay_counter = 0; delay_counter < 200; ++delay_counter);
    GPIO_PORTF_DIR_R |= 1<<3;
    GPIO_PORTF_DEN_R |= 1<<3;

    while(1){
        GPIO_PORTF_DATA_R |= 1<<3;
        for (delay_counter = 0; delay_counter < 200000;
++delay_counter);
        GPIO_PORTF_DATA_R &= ~(1<<3);
        for (delay_counter = 0; delay_counter < 200000;
++delay_counter);
    }

    return 0;
}

// parameter to use texas edx lab2:
// -dedXLab2
```

Symbols

```
00000000 T main
```

Startup file

Startup.c

```

/*startup_cortexM3.c
Eng. Andrew Adel
*/

/*SRAM @ 0x20000000*/
#include "platform_types.h"

extern uint32 _stack_top;
extern uint32 _S_DATA;
extern uint32 _E_DATA;
extern uint32 _S_BSS;
extern uint32 _E_BSS;
extern uint32 _E_TEXT;

void Rest_Handler(void);

extern int main(void);

void Default_Handler(void) {
    Rest_Handler();
}

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

static volatile uint32 stack[256];

void (* const g_p_fn_Vectors[])() __attribute__((section(".vectors"))) =
{
    ( void(* const)() ) ((uint32)&stack[255] +4) ,
    &Rest_Handler,
    &NMI_Handler,
    &H_Fault_Handler,
    &MM_Fault_Handler,
    &Bus_Fault_Handler,

```

```

        &Usage_Fault_Handler
    };

void Rest_Handler(void) {
    uint32 DATA_SIZE = (uint8*)&_E_DATA - (uint8*)&_S_DATA;
    uint8* P_src = (uint8*)&_E_TEXT;
    uint8* P_dst = (uint8*)&_S_DATA;
    int i;
    for (i = 0; i < DATA_SIZE; ++i)
    {
        *(P_dst++) = *(P_src++);
    }

    uint32 BSS_SIZE = (uint8*)&_E_BSS - (uint8*)&_S_BSS;
    P_dst = (uint8*)&_S_BSS;
    for (i = 0; i < BSS_SIZE; ++i)
    {
        *(P_dst++) = *(uint8*)0;
    }

    main();
}

```

Symbols

```

            U _E_BSS
            U _E_DATA
            U _E_TEXT
            U _S_BSS
            U _S_DATA
00000000 W Bus_Fault_Handler
00000000 T Default_Handler
00000000 R g_p_fn_Vectors
00000000 W H_Fault_Handler
            U main
00000000 W MM_Fault_Handler
00000000 W NMI_Handler
0000000c T Rest_Handler
00000000 b stack
00000000 W Usage_Fault_Handler

```

Linker_script.ld

```

/*Linker_script CortexM3
Eng. Andrew Adel
*/

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*)
        _E_DATA = .;
    }>sram AT> flash

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

```

Makefile

```

#@copyright : Andrew Adel
#toolchain
CC=arm-none-eabi-
#repeated options
CFLAGS =-mcpu=cortex-m4 -mthumb -gdwarf-2 -g
INCS =-I .
LIBS =
#source files .c
SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)      #source files after compilation
#source files .s
As = $(wildcard *.s)
AsOBJ = $(As:.s=.o)     #source files after compilation
#project name
Project_Name=unit3_lab4_cortexM4

# default make
all: $(Project_Name).bin
    @echo "=====Build is Done===== "

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

#linking all objects files to .elf file and generate map file
$(Project_Name).elf: $(OBJ)
    $(CC)ld.exe -T linker_script.ld $(LIBS) $(OBJ) -o $@ -
Map=Map_file.map
    cp $(Project_Name).elf $(Project_Name).axf

#generate binary file which will be executed
$(Project_Name).bin: $(Project_Name).elf
    $(CC)objcopy -O binary $< $@

#remove all .o .elf .bin .map files
clean_all:
    rm *.o *.elf *.bin *.map *.axf
#remove only final files
clean:
    rm *.elf *.bin *.map

```


unit3 lab4 cortexM4.elf

symbols

```

20000400 B _E_BSS
20000000 T _E_DATA
000001a4 T _E_TEXT
20000000 B _S_BSS
20000000 T _S_DATA
000000e4 W Bus_Fault_Handler
000000e4 T Default_Handler
00000000 T g_p_fn_Vectors
000000e4 W H_Fault_Handler
0000001c T main
000000e4 W MM_Fault_Handler
000000e4 W NMI_Handler
000000f0 T Rest_Handler
20000000 b stack
000000e4 W Usage_Fault_Handler

```

Map_file.map

Memory Configuration

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

Linker script **and** memory map

.text	0x00000000	0x1a4	
(.vectors)			
.vectors	0x00000000	0x1c	startup.o
	0x00000000		g_p_fn_Vectors
(.text)			
.text	0x0000001c	0xc8	main.o
	0x0000001c		main
.text	0x000000e4	0xc0	startup.o
	0x000000e4		Bus_Fault_Handler
	0x000000e4		H_Fault_Handler
	0x000000e4		MM_Fault_Handler
	0x000000e4		Default_Handler
	0x000000e4		Usage_Fault_Handler
	0x000000e4		NMI_Handler
	0x000000f0		Rest_Handler

LAB3 REPORT

```

*(.rodata)
                                0x000001a4                _E_TEXT = .

.glue_7                        0x000001a4                0x0
.glue_7                        0x00000000                0x0 linker stubs

.glue_7t                       0x000001a4                0x0
.glue_7t                       0x00000000                0x0 linker stubs

.vfp11_veneer                  0x000001a4                0x0
.vfp11_veneer                  0x00000000                0x0 linker stubs

.v4_bx                         0x000001a4                0x0
.v4_bx                         0x00000000                0x0 linker stubs

.iplt                          0x000001a4                0x0
.iplt                          0x00000000                0x0 main.o

.rel.dyn                       0x000001a4                0x0
.rel.iplt                      0x00000000                0x0 main.o

.data                          0x20000000                0x0 load address 0x000001a4
                                0x20000000                _S_DATA = .

*(.data*)
.data                          0x20000000                0x0 main.o
.data                          0x20000000                0x0 startup.o
                                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                . = ALIGN (0x4)
                                0x20000400                _E_BSS = .

LOAD main.o
LOAD startup.o
OUTPUT(unit3_lab4_cortexM4.elf elf32-littlearm)

.debug_info                    0x00000000                0x263
.debug_info                    0x00000000                0xb6 main.o
.debug_info                    0x000000b6                0x1ad startup.o

.debug_abbrev                   0x00000000                0x145
.debug_abbrev                   0x00000000                0x67 main.o
.debug_abbrev                   0x00000067                0xde startup.o

```

LAB3 REPORT

.debug_loc	0x00000000	0x9c
.debug_loc	0x00000000	0x38 main.o
.debug_loc	0x00000038	0x64 startup.o
.debug_aranges	0x00000000	0x40
.debug_aranges	0x00000000	0x20 main.o
.debug_aranges	0x00000020	0x20 startup.o
.debug_line	0x00000000	0xf4
.debug_line	0x00000000	0x77 main.o
.debug_line	0x00000077	0x7d startup.o
.debug_str	0x00000000	0x18a
.debug_str	0x00000000	0xfa main.o
		0x12e (size before relaxing)
.debug_str	0x000000fa	0x90 startup.o
		0x1a4 (size before relaxing)
.comment	0x00000000	0x11
.comment	0x00000000	0x11 main.o
		0x12 (size before relaxing)
.comment	0x00000000	0x12 startup.o
.ARM.attributes		
	0x00000000	0x33
.ARM.attributes		
	0x00000000	0x33 main.o
.ARM.attributes		
	0x00000033	0x33 startup.o
.debug_frame	0x00000000	0x78
.debug_frame	0x00000000	0x2c main.o
.debug_frame	0x0000002c	0x4c startup.o