8/22/2023

Name: **Andrew Adel Hosny Goued**

Position: Student at Faculty of Engineering

Ain Shams University  
Computer and System Engineering

**Report**

**Lab3 (Toggling green led on TM4C123GH6PZ)**

**Learn-In-Depth Diploma**

**Contents**

[Introduction 2](#_Toc143632890)

[TM4C123GH6PZ information 2](#_Toc143632891)

[Main file 3](#_Toc143632892)

[Main.c 3](#_Toc143632893)

[Symbols 3](#_Toc143632894)

[Startup file 4](#_Toc143632895)

[Startup.c 4](#_Toc143632896)

[Symbols 5](#_Toc143632897)

[Linker\_script.ld 6](#_Toc143632898)

[Makefile 7](#_Toc143632899)

[symbols 8](#_Toc143632900)

[Map\_file.map 8](#_Toc143632901)

# 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 =  
#souce 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  
 \*(.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  
  
.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