Embedded C

Assignment lesson 4

-Toggle led on Arm_cortex m4 32-bit Tivac tm4c123 chip :

1-Write codes

-main.c

```c
/* main.c
   toggle led on coretx m4 microcontroller tm4c123
   Eng: Mostafa Besher
*/


#include "Header_Platform.h"

#define SYSCTL_base 0x400FE000
#define GPIOF_base  0x40025000

#define sysctl_RCGC2         *((vuint32_t *)(SYSCTL_base + 0x108))
#define GPIO_PORTF_DIR_R     *((vuint32_t *)(GPIOF_base + 0x400))
#define GPIO_PORTF_DEN_R     *((vuint32_t *)(GPIOF_base + 0x51c))

typedef union
{
    vuint32_t all_fields;
    struct
    {
        vuint32_t reserved : 3;
        vuint32_t pin3 : 1;
    }pin;

}GPIO_PORTF_DATA_R;

volatile GPIO_PORTF_DATA_R * portf = (volatile GPIO_PORTF_DATA_R *)(GPIOF_base + (0x3fc));

int main()
{
    vuint32_t delay_counter;
    sysctl_RCGC2 = 0x00000020;
    //delay to make sure GPIOF is up and running
    for(delay_counter = 0; delay_counter < 200; delay_counter++);

    GPIO_PORTF_DIR_R |= 1<<3;   //Dir is output for pin3 portf
    GPIO_PORTF_DEN_R |= 1<<3;

    while(1)
    {
        portf ->pin.pin3 = 1;
        for(delay_counter = 0; delay_counter < 200000; delay_counter++);    //delay
        portf -> pin.pin3 = 0;
        for(delay_counter = 0; delay_counter < 200000; delay_counter++);    //delay
    }
    return 0;
}
```

-Headers_platform.h

```c
/*
 * Header_Platform.h
 *
 *  Created on: Mar 30, 2021
 *      Author: mostafa
 */

#ifndef HEADER_PLATFORM_H_
#define HEADER_PLATFORM_H_


#define CPU_TYPE        CPU_TYPE_32
#define CPU_BIT_ORDER   MSB_FIRST
#define CPU_BYTE_ORDER  HIGH_BYTE_FIRST

typedef unsigned char       uint8_t;
typedef signed char         sint8_t;
typedef unsigned short      uint16_t;
typedef signed short        sint16_t;
typedef unsigned int        uint32_t;
typedef signed int          sint32_t;
typedef unsigned long long  uint64_t;
typedef signed long long    sint64_t;
typedef float               float32_t;
typedef double              float64_t;

typedef volatile unsigned char          vuint8_t;
typedef volatile signed char            vsint8_t;

typedef volatile unsigned short         vuint16_t;
typedef volatile signed short           vsint16_t;

typedef volatile unsigned int           vuint32_t;
typedef volatile signed int             vsint32_t;

typedef volatile unsigned long long     vuint64_t;
typedef volatile signed long long       vsint64_t;

typedef volatile float                  vfloat32_t;
typedef volatile double                 vfloat64_t;


#endif /* HEADER_PLATFORM_H_ */
```

-startup.c

```c
//startup.c
//Eng:Mostafa Besher

#include <stdint.h>
//prototypes
extern int main(void);
void Default_handler();
void Reset_handler();
void NMI_handler()__attribute__((weak,alias("Default_handler")));;
void HARD_FAULT_handler()__attribute__((weak,alias("Default_handler")));;
void MM_handler()__attribute__((weak,alias("Default_handler")));;
void BUS_handler()__attribute__((weak,alias("Default_handler")));;
void USAGE_FAULT_handler()__attribute__((weak,alias("Default_handler")));;

//declaration of symbols
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;


static unsigned long stack_top[256];
//.vectors section
void (* const vectors[])() __attribute__((section(".vectors"))) =
{
    (void (*) ())((unsigned long)&stack_top + sizeof(stack_top)),
    &Reset_handler,
    &NMI_handler,
    &HARD_FAULT_handler,
    &MM_handler,
    &BUS_handler,
    &USAGE_FAULT_handler
};


void Reset_handler()
```

```c

void Reset_handler()
{
    int i = 0;
    //copy .data from flash to sram
    uint32_t Data_size = (unsigned char *)&_E_data - (unsigned char *)&_S_data;
    unsigned char *p_src = (unsigned char *)&_E_text;   //starting address of .data in flash
    unsigned char *p_dst = (unsigned char *)&_S_data;   //starting address of .data in sram

    for(i = 0; i < Data_size; i++)
    {
        *((unsigned char *)p_dst++) = *((unsigned char *)p_src++);
    }

    //initialize .bss with zero
    uint32_t bss_size = (unsigned char *)&_E_bss - (unsigned char *)&_S_bss;
    p_dst = (unsigned char *)&_S_bss;

    for(i = 0; i < bss_size; i++)
    {
        *((unsigned char *)p_dst++) = (unsigned char)0;
    }

    //jump to main
    main();
}

void Default_handler()
{
    Reset_handler();
}
```

-linker_script.ld

```
/* linker_script cortexM4
Eng.Mostafa_Besher
*/


MEMORY
{
    FLASH(RX): ORIGIN = 0x00000000, LENGTH = 512M
    SRAM(RWX): ORIGIN = 0x20000000, LENGTH = 512M
}

SECTIONS
{
    .text :
    {
        *(.vectors*)
        *(.text*)
        *(.rodata)
        . = ALIGN(4);
        _E_text = .;
    }>FLASH
    .data :
    {
        _S_data = .;
        *(.data)
        . = ALIGN(4);
        _E_data = .;
    }>SRAM AT> FLASH
    .bss :
    {
        _S_bss = .;
        *(.bss)
        _E_bss = .;
    }>SRAM


}
```

-Makefile

```
#@copyright : Mostafa_Besher
CC=arm-none-eabi-
CFLAGS=-mcpu=cortex-m4 -gdwarf-2 -g
INCS= -I .
LIBS=
SRC= $(wildcard *.c)
OBJ= $(SRC:.c=.o)
As= $(wildcard *.s)
AsOBJ= $(As:.s=.o)
Project_name= toggle_led_lab3


all: $(Project_name).bin
    @echo "=======Build is complete======="

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

$(Project_name).elf: $(OBJ) $(AsOBJ)
    $(CC)ld.exe -T linker_script.ld $(LIBS) $(OBJ) $(AsOBJ) -o $(Project_name).elf -Map=map_file.map

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

clean_all:
    rm *.o *.elf *.bin
clean:
    rm *.elf *.bin
```

2-Get object_files using Makefile

```
mostafa@DESKTOP-6K5T62N MINGW32 /d/Embedded_Diploma/assignments/2_Embedded c/lesson 4/lab 3
$ make
arm-none-eabi-gcc.exe -c -mcpu=cortex-m4 -gdwarf-2 -g -I . main.c -o main.o
arm-none-eabi-gcc.exe -c -mcpu=cortex-m4 -gdwarf-2 -g -I . startup.c -o startup.o
arm-none-eabi-ld.exe -T linker_script.ld  main.o startup.o  -o toggle_led_lab3.elf -Map=map_file.map
cp toggle_led_lab3.elf toggle_led_lab3.axf
arm-none-eabi-objcopy.exe -O binary toggle_led_lab3.elf toggle_led_lab3.bin
```

3-Show Symbols For :

1-main.o

```
mostafa@DESKTOP-6K5T62N MINGW32 /d/Embedded_Diploma/assignments/2
$ arm-none-eabi-nm.exe main.o
00000000 T main
00000000 D portf
```

2- startup.o

```
mostafa@DESKTOP-6K5T62N MINGW32 /d/Embedded_Diploma/assignm
$ arm-none-eabi-nm.exe startup.o
         U _E_bss
         U _E_data
         U _E_text
         U _S_bss
         U _S_data
00000088 W BUS_handler
00000088 T Default_handler
00000088 W HARD_FAULT_handler
         U main
00000088 W MM_handler
00000088 W NMI_handler
00000000 T Reset_handler
00000000 b stack_top
00000088 W USAGE_FAULT_handler
00000000 R vectors
```

3-toggle_led_lab3.elf

```
mostafa@DESKTOP-6K5T62N MINGW32 /d/Embedded_Diploma
$ arm-none-eabi-nm.exe toggle_led_lab3.elf
20000404 B _E_bss
20000004 D _E_data
0000013c T _E_text
20000004 B _S_bss
20000000 D _S_data
00000130 W BUS_handler
00000130 T Default_handler
00000130 W HARD_FAULT_handler
0000001c T main
00000130 W MM_handler
00000130 W NMI_handler
20000000 D portf
000000a8 T Reset_handler
20000004 b stack_top
00000130 W USAGE_FAULT_handler
00000000 T vectors
```

4-Map File

```
Memory Configuration

Name             Origin             Length             Attributes
FLASH            0x0000000000000000 0x0000000020000000 xr
SRAM             0x0000000020000000 0x0000000020000000 xrw
*default*        0x0000000000000000 0xffffffffffffffff

Linker script and memory map


.text           0x0000000000000000      0x13c
 *(.vectors*)
 .vectors       0x0000000000000000       0x1c startup.o
                0x0000000000000000                vectors
 *(.text*)
 .text          0x000000000000001c       0x8c main.o
                0x000000000000001c                main
 .text          0x00000000000000a8       0x94 startup.o
                0x00000000000000a8                Reset_handler
                0x0000000000000130                USAGE_FAULT_handler
                0x0000000000000130                BUS_handler
                0x0000000000000130                HARD_FAULT_handler
                0x0000000000000130                MM_handler
                0x0000000000000130                Default_handler
                0x0000000000000130                NMI_handler
 *(.rodata)
                0x000000000000013c                . = ALIGN (0x4)
                0x000000000000013c                _E_text = .
```

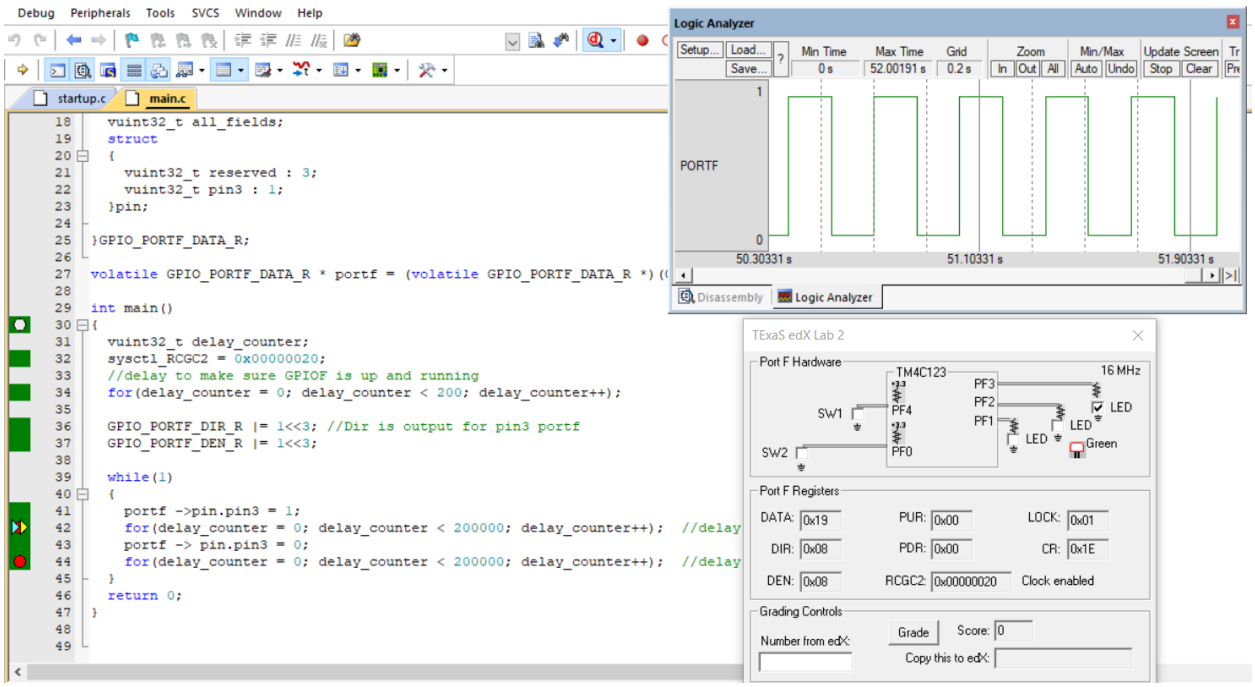
```
.data           0x0000000020000000        0x4 load address 0x000000000000013c
                0x0000000020000000                _S_data = .
 *(.data)
 .data          0x0000000020000000        0x4 main.o
                0x0000000020000000                portf
 .data          0x0000000020000004        0x0 startup.o
                0x0000000020000004                . = ALIGN (0x4)
                0x0000000020000004                _E_data = .

.igot.plt       0x0000000020000004        0x0 load address 0x0000000000000140
 .igot.plt      0x0000000020000004        0x0 main.o

.bss            0x0000000020000004      0x400 load address 0x0000000000000140
                0x0000000020000004                _S_bss = .
 *(.bss)
 .bss           0x0000000020000004        0x0 main.o
 .bss           0x0000000020000004      0x400 startup.o
                0x0000000020000404                _E_bss = .
```

5-KEIL Simulation

- When led is on



-when led is off