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

D:\Engineering\Embedded_Systems\Eng_Kerolos_Diploma\Lectures\Unit3\Lesson4\Lab3\main.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

```c
 * All rights reserved.</center></h2>
 *
 * This software component is licensed by ST under BSD 3-Clause license,
 * the "License"; You may not use this file except in compliance with the
 * License. You may obtain a copy of the License at:
 *                       opensource.org/licenses/BSD-3-Clause
 *
 ******************************************************************************
 */

#include "stdint.h"

#define SYSCTL_RCGC2_R      (*((volatile uint32_t *)0x400FE108))
#define GPIO_PORTF_DIR_R    (*((volatile uint32_t *)0x40025400))
#define GPIO_PORTF_DEN_R    (*((volatile uint32_t *)0x4002551C))
#define GPIO_PORTF_DATA_R   (*((volatile uint32_t *)0x400253FC))


int main(void)
{
    SYSCTL_RCGC2_R = 0x20;
    // delay to make sure GPIOF is up and running
    volatile unsigned long delay_count = 0;
    for(delay_count = 0; delay_count < 200; delay_count++);
    GPIO_PORTF_DIR_R |= 1 << 3; //DIR is output for pin3 port F
    GPIO_PORTF_DEN_R |= 1 << 3;

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

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++

```makefile
#@Copyright : Abdo Halem

CC=arm-none-eabi
CFLAGS= -gdwarf-2 -mcpu=cortex-m4 -g
INCS=-I .
LIBS=
SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)
#As = $(wildcard *.s)
#AsOBJ = $(As:.s=.o)
Project_name=Unit3_Lesson4_Lab3_cortex_m4

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

#startup.o: startup.s
#   $(CC)-as $(CFLAGS) $< -o $@

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

$(Project_name).elf: $(OBJ) $(AsOBJ)
    $(CC)-ld -T linker_script.ld  $(LIBS) $(OBJ) $(AsOBJ) -o $@ -Map=Map_file.map
    cp $(Project_name).elf $(Project_name).axf

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

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

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

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

```c
/* startup.c: Eng Abdo Halem */

#include <stdint.h>
extern int main();
void Reset_Handler();

void Default_Handler(){
    Reset_Handler();
}

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

// Booking 1024 byte located by .bss through uninitialized array of 256 int (256*4 = 1024)
static unsigned int stack_top[256];

void (* const g_p_fn_Vectors[])() __attribute__((section(".vectors"))) = {
    (void (*)())((unsigned long)stack_top + sizeof(stack_top)),
    &Reset_Handler,
    &NMI_Handler,
    &H_Fault_Handler
};

extern unsigned int _E_text;
extern unsigned int _S_DATA;
extern unsigned int _E_DATA;
extern unsigned int _S_bss;
extern unsigned int _E_bss;

void Reset_Handler(){
    // Copy data section from flash to ram
    unsigned int DATA_size = (unsigned char*)&_E_DATA - (unsigned char*)&_S_DATA;
    unsigned char* P_src = (unsigned char*)&_E_text;
    unsigned char* P_dst = (unsigned char*)&_S_DATA;
    unsigned int i = 0;
    for(i = 0; i < DATA_size; i++){
        *((unsigned char*)P_dst++) = *((unsigned char*)P_src++);
```

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 غالبًا | ENG | 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

```c

// Booking 1024 byte located by .bss through uninitialized array of 256 int (256*4 = 1024)
static unsigned int stack_top[256];

void (* const g_p_fn_Vectors[])() __attribute__((section(".vectors"))) = {
    (void (*)())((unsigned long)stack_top + sizeof(stack_top)),
    &Reset_Handler,
    &NMI_Handler,
    &H_Fault_Handler
};

extern unsigned int _E_text;
extern unsigned int _S_DATA;
extern unsigned int _E_DATA;
extern unsigned int _S_bss;
extern unsigned int _E_bss;

void Reset_Handler(){
    // Copy data section from flash to ram
    unsigned int DATA_size = (unsigned char*)&_E_DATA - (unsigned char*)&_S_DATA;
    unsigned char* P_src = (unsigned char*)&_E_text;
    unsigned char* P_dst = (unsigned char*)&_S_DATA;
    unsigned int i = 0;
    for(i = 0; i < DATA_size; i++){
        *((unsigned char*)P_dst++) = *((unsigned char*)P_src++);
    }
    // Initialize .bss section in SRAM
    unsigned int 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();
}
```

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 غالبًا | ENG | 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++

```
/* linker script cortexM4
Eng Abdo Halem */

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)
        . = ALIGN(4);
        _E_DATA = . ;
    }> sram AT> flash

    .bss : {
        _S_bss = . ;
        *(.bss*)
        _E_bss = . ;

    }> sram
}
```

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 0x0000000 and the first section is .vectors Section.

```
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       0x10 startup.o
                0x00000000                g_p_fn_Vectors
 *(.text*)
 .text          0x00000010       0xd0 main.o
                0x00000010                main
 .text          0x000000e0       0xc4 startup.o
                0x000000e0                H_Fault_Handler
                0x000000e0                Default_Handler
                0x000000e0                NMI_Handler
                0x000000ec                Reset_Handler
 *(.rodata)
                0x000001a4                _E_text = .

```

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