LAB 3

TM4C123 cortexM4

BY: ABDELRAHMAN MATARAWY

❖ Main.c:

```c
/* By Abdelrahman Matarawy */

/* Register Address */
#define SYSCTL_RCGC2_R      (*((volatile unsigned long*)0x400FE108)) /* Enable GPIO Port */
#define GPIO_PORTF_DIR_R    (*((volatile unsigned long*)0x40025400)) /* Set Direction As OutPut */
#define GPIO_PORTF_DEN_R    (*((volatile unsigned long*)0x4002551C)) /* Enable GPIO Pin */
#define GPIO_PORTF_DATA_R   (*((volatile unsigned long*)0x400253FC)) /* To Write Data on it */

int main(void)
{
    volatile unsigned long count;
    SYSCTL_RCGC2_R = 0x20 ; /* GPIO Enable */
    /* Delay to be ensure that GPIO is up and running */
    for(count = 0 ; count < 200 ; count++);
    GPIO_PORTF_DIR_R |= 1 << 3; /* Write on Bit 3 to be output */
    GPIO_PORTF_DEN_R |= 1 << 3; /* Enable bit 3 to take data */
    while(1)
    {
        /* Write logic high on bit 3 */
        GPIO_PORTF_DATA_R |= (1 << 3);
        for(count = 0 ; count < 200000 ; count++);
        /* Write logic low on bit 3 */
        GPIO_PORTF_DATA_R &= ~(1 << 3);
        for(count = 0 ; count < 200000 ; count++);
    }
    return 0;
}
```

❖ LinkerScript:

```
/* Linker_Script for CortexM4 */
/* By Abdelrahman Matarawy */

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*)
        . = ALIGN(4);
        _E_bss = . ;
    } > sram
}
```

❖ Startup:

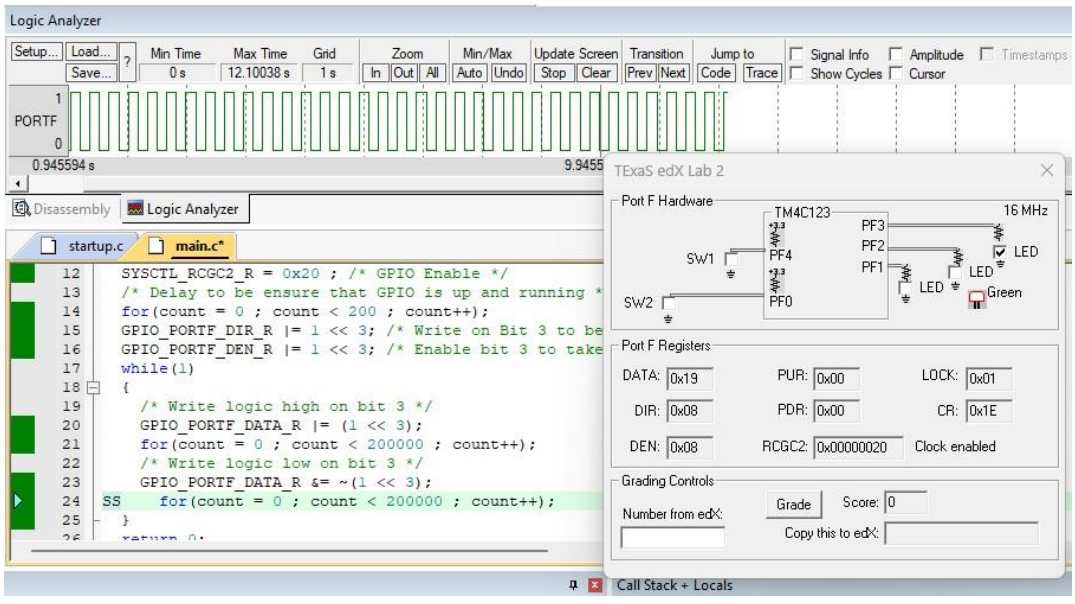
```c
/* StartUp.c For CortexM4
By Eng:Abdelrahman Matarawy
*/

#include "Platform_Types.h"

int i;
extern int main(void);

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")));

/* Reserve 1024 byte (256*4) from bss to added stack pointer after it */
static unsigned long Stack_Top[256];

/* Array for IVT */
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 uint32_t _E_text;
extern uint32_t _S_data;
extern uint32_t _E_data;
extern uint32_t _S_bss;
extern uint32_t _E_bss;

void Reset_Handler()
{
    /* Copy data section from flash to ram */
    uint32_t  Data_Size = (uint8_t*)&_E_data - (uint8_t*)&_S_data;
    uint8_t* source = (uint8_t*)&_E_text;
    uint8_t* destination = (uint8_t*)&_S_data;

    for(i = 0 ; i < Data_Size ; i++ )
    {
        *((uint8_t*)destination++) = *((uint8_t*)source++);
    }

    /* Init bss section with zeros in ram */
    uint32_t  Bss_Size = (uint8_t*)&_E_bss - (uint8_t*)&_S_bss;
    destination = (uint8_t*)&_S_bss;
    for(i = 0 ; i < Data_Size ; i++ )
    {
        *((uint8_t*)destination++) = (uint8_t)0;
    }

    /* jump to main */
    main();
}
```

❖ MakeFile:

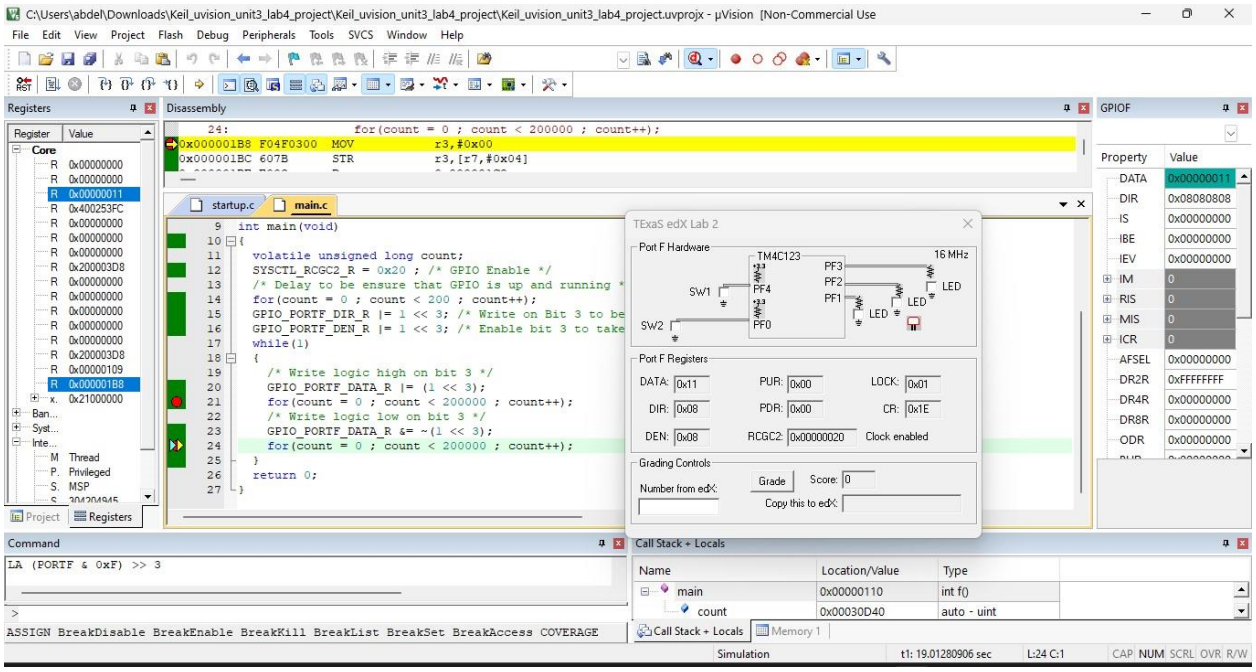
```
#@CopyWriter: Matarawy

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= learn-in-depth-cortex-m4


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

%.o: %.s
    $(CC)as.exe $(CFLAGS) -mthumb $< -o $@

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

$(Project_Name).elf: $(OBJ) $(AsOBJ)
    $(CC)ld.exe -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.exe -O binary $< $@

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

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

❖ OutPut:

- Logic Analayzer:



➢ At LED OFF:



➢ At LED ON: