Lab 4

In this lab we will simulate and debug code on Teva C kit that has tm4c123 SOC and arm-cortexM4 processor .

The scope is toggling a LED connected to pin3 of PORTF,

We will write Main.c , Startup.c, linker script and make file from scratch

According to specs we found out these information:

- Flash memory starts with address 0x00000000 and has size of 512M.
- Sram memory starts 0x20000000 and has size of 512M.

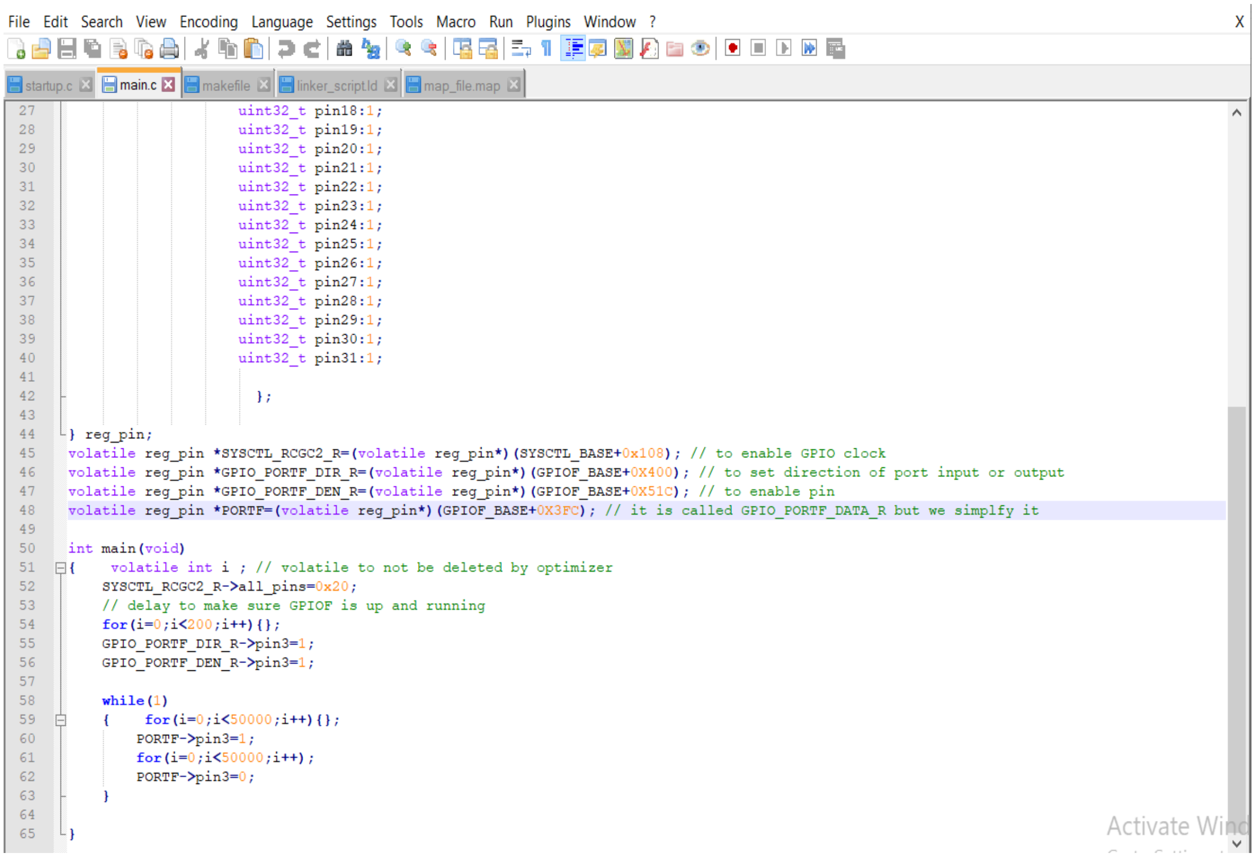
- SYSCTL is system control module that we will use to enable clock for PORTF has base address of 0x400FE000

- 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.

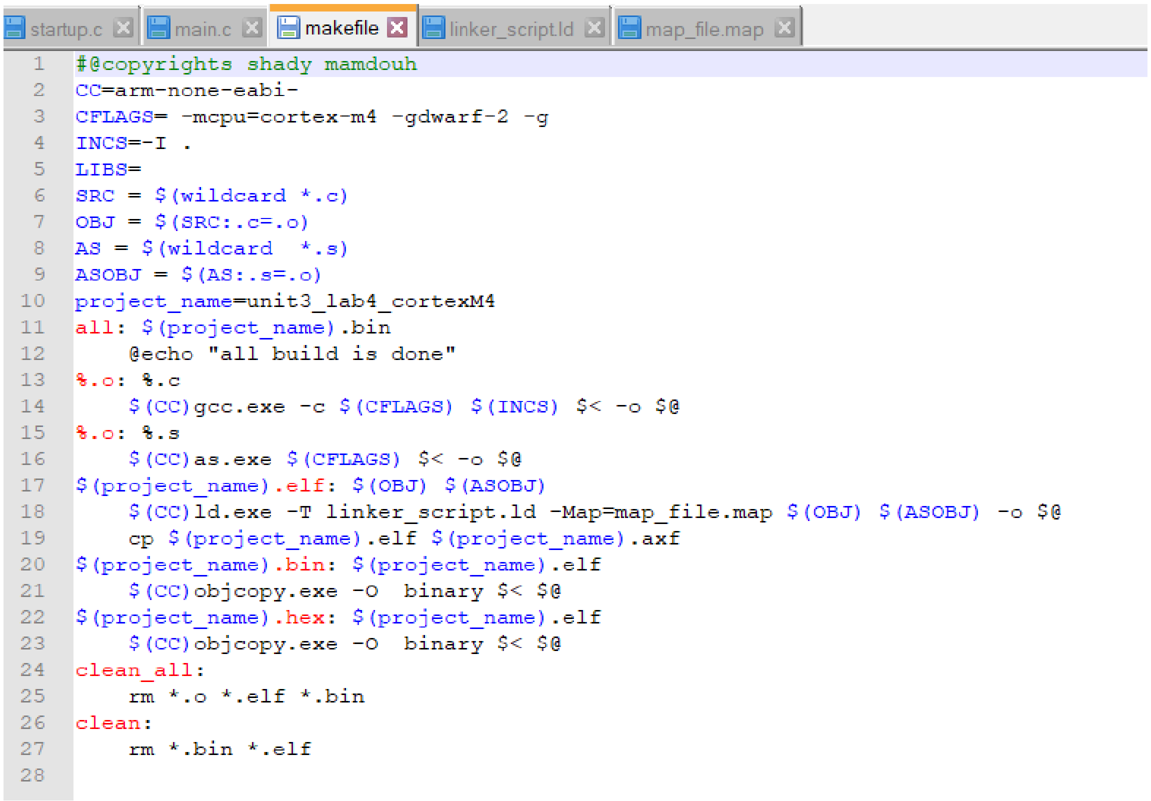
Main.c


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
X
startup.c
main.c
makefile
linker_script.ld
map_file.map
Activate Wind

```c
                        uint32_t pin18:1;
                        uint32_t pin19:1;
                        uint32_t pin20:1;
                        uint32_t pin21:1;
                        uint32_t pin22:1;
                        uint32_t pin23:1;
                        uint32_t pin24:1;
                        uint32_t pin25:1;
                        uint32_t pin26:1;
                        uint32_t pin27:1;
                        uint32_t pin28:1;
                        uint32_t pin29:1;
                        uint32_t pin30:1;
                        uint32_t pin31:1;

                          };

} reg_pin;
volatile reg_pin *SYSCTL_RCGC2_R=(volatile reg_pin*)(SYSCTL_BASE+0x108); // to enable GPIO clock
volatile reg_pin *GPIO_PORTF_DIR_R=(volatile reg_pin*)(GPIOF_BASE+0X400); // to set direction of port input or output
volatile reg_pin *GPIO_PORTF_DEN_R=(volatile reg_pin*)(GPIOF_BASE+0X51C); // to enable pin
volatile reg_pin *PORTF=(volatile reg_pin*)(GPIOF_BASE+0X3FC); // it is called GPIO_PORTF_DATA_R but we simplfy it

int main(void)
{    volatile int i ; // volatile to not be deleted by optimizer
    SYSCTL_RCGC2_R->all_pins=0x20;
    // delay to make sure GPIOF is up and running
    for(i=0;i<200;i++){};
    GPIO_PORTF_DIR_R->pin3=1;
    GPIO_PORTF_DEN_R->pin3=1;

    while(1)
    {    for(i=0;i<50000;i++){};
        PORTF->pin3=1;
        for(i=0;i<50000;i++);
        PORTF->pin3=0;
    }

}
```

Make file :

-we will make some changes on make file : project name and we will copy a .axf file to run on kiel micro vision tool and processor name



```
#@copyrights shady mamdouh
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=unit3_lab4_cortexM4
all: $(project_name).bin
	@echo "all build is done"
%.o: %.c
	$(CC)gcc.exe -c $(CFLAGS) $(INCS) $< -o $@
%.o: %.s
	$(CC)as.exe $(CFLAGS) $< -o $@
$(project_name).elf: $(OBJ) $(ASOBJ)
	$(CC)ld.exe -T linker_script.ld -Map=map_file.map $(OBJ) $(ASOBJ) -o $@
	cp $(project_name).elf $(project_name).axf
$(project_name).bin: $(project_name).elf
	$(CC)objcopy.exe -O  binary $< $@
$(project_name).hex: $(project_name).elf
	$(CC)objcopy.exe -O  binary $< $@
clean_all:
	rm *.o *.elf *.bin
clean:
	rm *.bin *.elf

```

Startup.c :

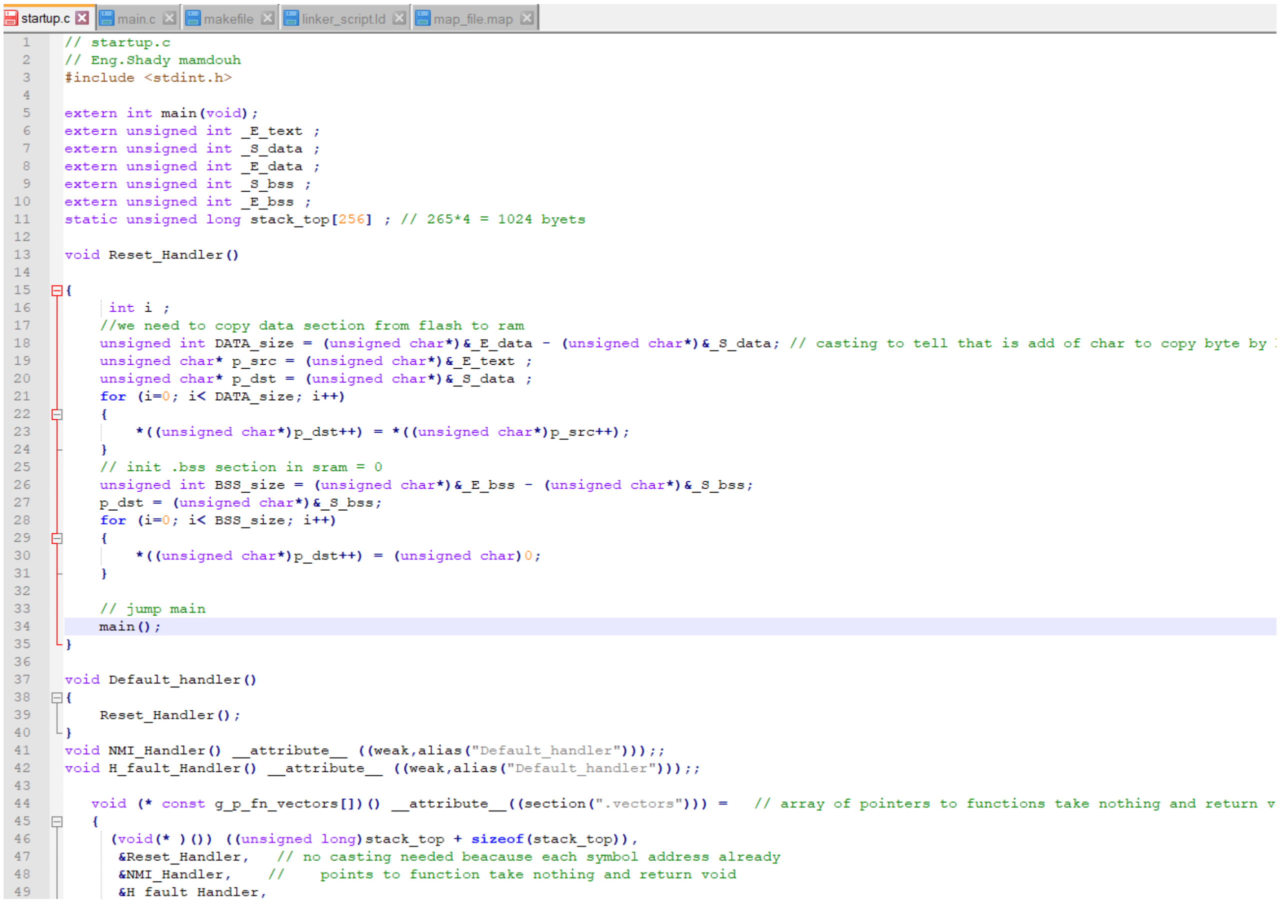
In this lab we will use a new approach by initialize SP in Startup.c

Instead of create it's 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

That the total size of array will be 1024 byte and this is where SP will be at the end of the array.

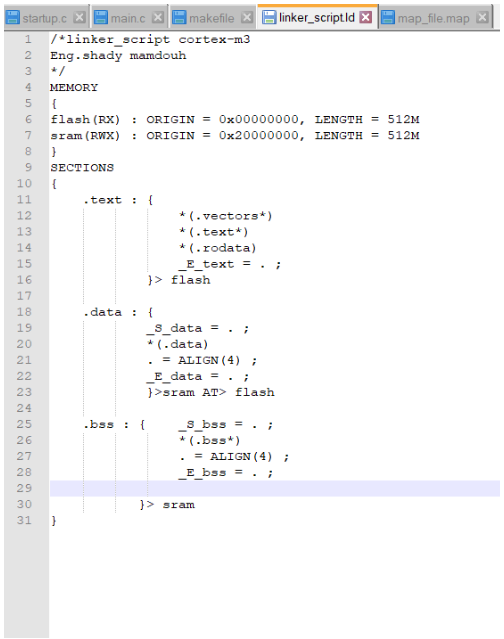
Then we will make an array of pointers to functions take nothing and return void these pointers will points to each function that will handle it's relative interrupt according to interrupt vector table .



```c
// startup.c
// Eng.Shady mamdouh
#include <stdint.h>

extern int main(void);
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 ;
static unsigned long stack_top[256] ; // 265*4 = 1024 byets

void Reset_Handler()

{
    int i ;
    //we need to copy data section from flash to ram
    unsigned int DATA_size = (unsigned char*)&_E_data - (unsigned char*)&_S_data; // casting to tell that is add of char to copy byte by 
    unsigned char* p_src = (unsigned char*)&_E_text ;
    unsigned char* p_dst = (unsigned char*)&_S_data ;
    for (i=0; i< DATA_size; i++)
    {
        *((unsigned char*)p_dst++) = *((unsigned char*)p_src++);
    }
    // init .bss section in sram = 0
    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 main
    main();
}

void Default_handler()
{
    Reset_Handler();
}
void NMI_Handler() __attribute__ ((weak,alias("Default_handler")));;
void H_fault_Handler() __attribute__ ((weak,alias("Default_handler")));;

   void (* const g_p_fn_vectors[])() __attribute__((section(".vectors"))) =   // array of pointers to functions take nothing and return v
   {
     (void(* )()) ((unsigned long)stack_top + sizeof(stack_top)),
      &Reset_Handler,    // no casting needed beacause each symbol address already
      &NMI_Handler,     //    points to function take nothing and return void
      &H_fault_Handler,
```

Linker script :

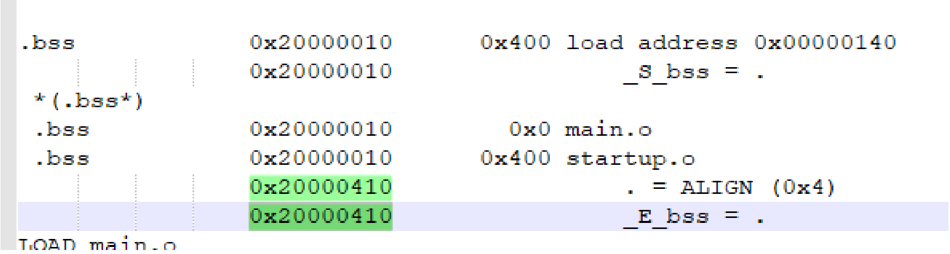
We will just edit sizes and delete stack top symbol



```
/*linker_script cortex-m3
Eng.shady mamdouh
*/
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*)
                . = ALIGN(4) ;
                _E_bss = . ;

           }> sram
}
```

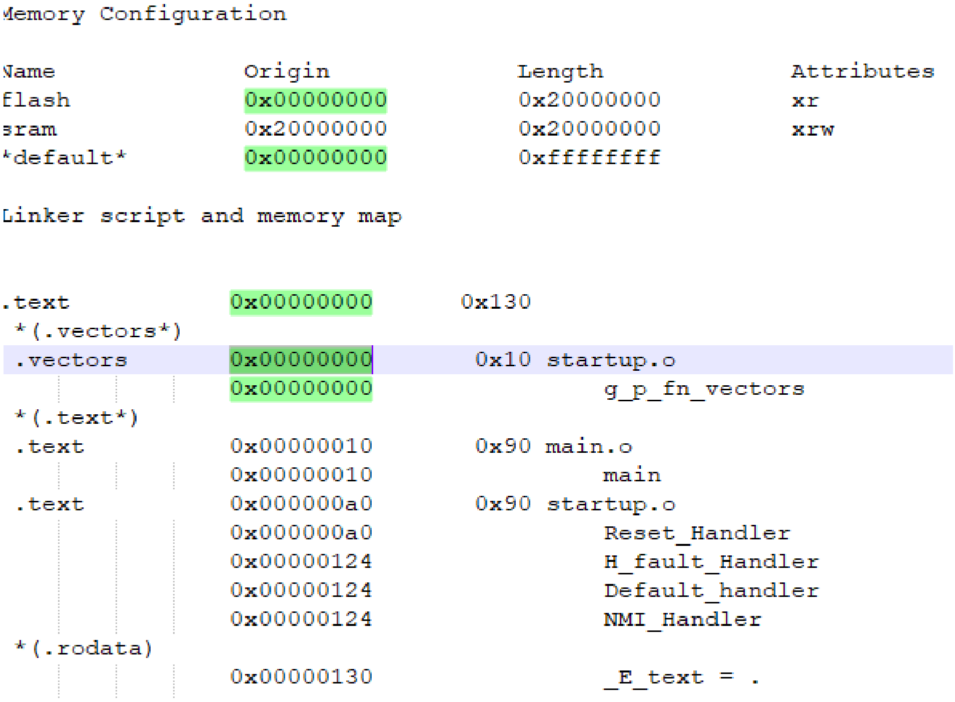
Map file :

.bss section starts with address of 0x20000010 and ends with 0x20000410 that has been incremented by 0x400 that equivalent to 1024 in decimal



```
.bss            0x20000010      0x400 load address 0x00000140
                0x20000010                _S_bss = .
 *(.bss*)
 .bss           0x20000010        0x0 main.o
 .bss           0x20000010      0x400 startup.o
                0x20000410                . = ALIGN (0x4)
                0x20000410                _E_bss = .
LOAD main.o
```

- Flash starts with 0x0000000 and the first section is .vectors section

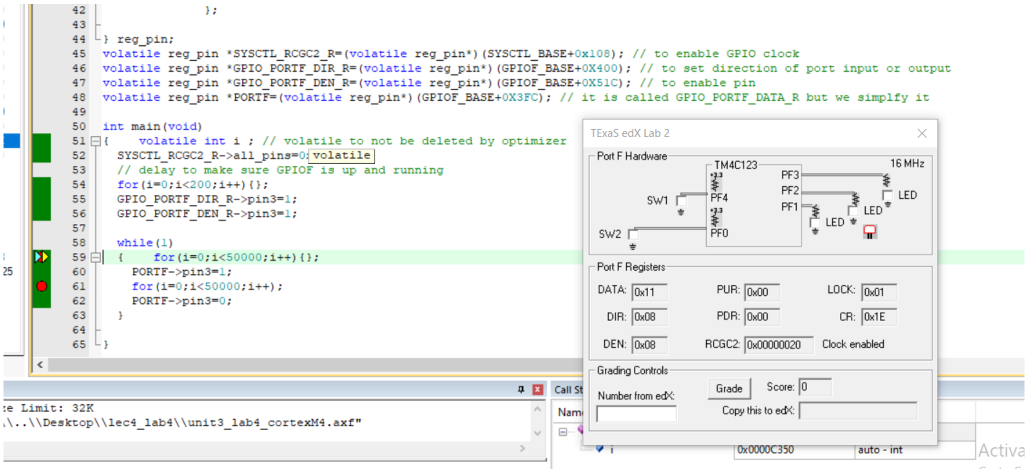


```
Memory Configuration

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

Linker script and memory map


.text           0x00000000      0x130
 *(.vectors*)
 .vectors       0x00000000       0x10 startup.o
                0x00000000                g_p_fn_vectors
 *(.text*)
 .text          0x00000010       0x90 main.o
                0x00000010                main
 .text          0x000000a0       0x90 startup.o
                0x000000a0                Reset_Handler
                0x00000124                H_fault_Handler
                0x00000124                Default_handler
                0x00000124                NMI_Handler
 *(.rodata)
                0x00000130                _E_text = .
```

Debugging using kiel Microvision :

Here we show led blinking and the values of register using Texas virtual board

At low level :



```
                };

} reg_pin;
volatile reg_pin *SYSCTL_RCGC2_R=(volatile reg_pin*)(SYSCTL_BASE+0x108); // to enable GPIO clock
volatile reg_pin *GPIO_PORTF_DIR_R=(volatile reg_pin*)(GPIOF_BASE+0X400); // to set direction of port input or output
volatile reg_pin *GPIO_PORTF_DEN_R=(volatile reg_pin*)(GPIOF_BASE+0X51C); // to enable pin
volatile reg_pin *PORTF=(volatile reg_pin*)(GPIOF_BASE+0X3FC); // it is called GPIO_PORTF_DATA_R but we simplfy it

int main(void)
{     volatile int i ; // volatile to not be deleted by optimizer
  SYSCTL_RCGC2_R->all_pins=0; volatile
  // delay to make sure GPIOF is up and running
  for(i=0;i<200;i++){};
  GPIO_PORTF_DIR_R->pin3=1;
  GPIO_PORTF_DEN_R->pin3=1;

  while(1)
  {     for(i=0;i<50000;i++){};
    PORTF->pin3=1;
    for(i=0;i<50000;i++);
    PORTF->pin3=0;
  }

}
```

```
e Limit: 32K
\..\\Desktop\\lec4_lab4\\unit3_lab4_cortexM4.axf"
```

TExaS edX Lab 2

Port F Registers

DATA: 0x11 PUR: 0x00 LOCK: 0x01

DIR: 0x08 PDR: 0x00 CR: 0x1E

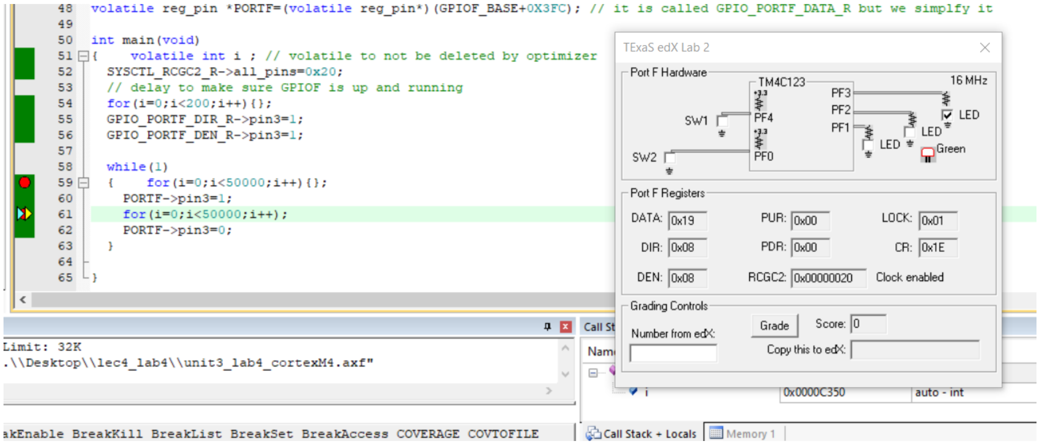
DEN: 0x08 RCGC2: 0x00000020 Clock enabled

Grading Controls

Score: 0

i 0x0000C350 auto - int

At high level :



```
volatile reg_pin *PORTF=(volatile reg_pin*)(GPIOF_BASE+0X3FC); // it is called GPIO_PORTF_DATA_R but we simplfy it

int main(void)
{     volatile int i ; // volatile to not be deleted by optimizer
  SYSCTL_RCGC2_R->all_pins=0x20;
  // delay to make sure GPIOF is up and running
  for(i=0;i<200;i++){};
  GPIO_PORTF_DIR_R->pin3=1;
  GPIO_PORTF_DEN_R->pin3=1;

  while(1)
  {     for(i=0;i<50000;i++){};
    PORTF->pin3=1;
    for(i=0;i<50000;i++);
    PORTF->pin3=0;
  }

}
```

```
Limit: 32K
.\\Desktop\\lec4_lab4\\unit3_lab4_cortexM4.axf"

akEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE
```

TExaS edX Lab 2

Port F Registers

DATA: 0x19 PUR: 0x00 LOCK: 0x01

DIR: 0x08 PDR: 0x00 CR: 0x1E

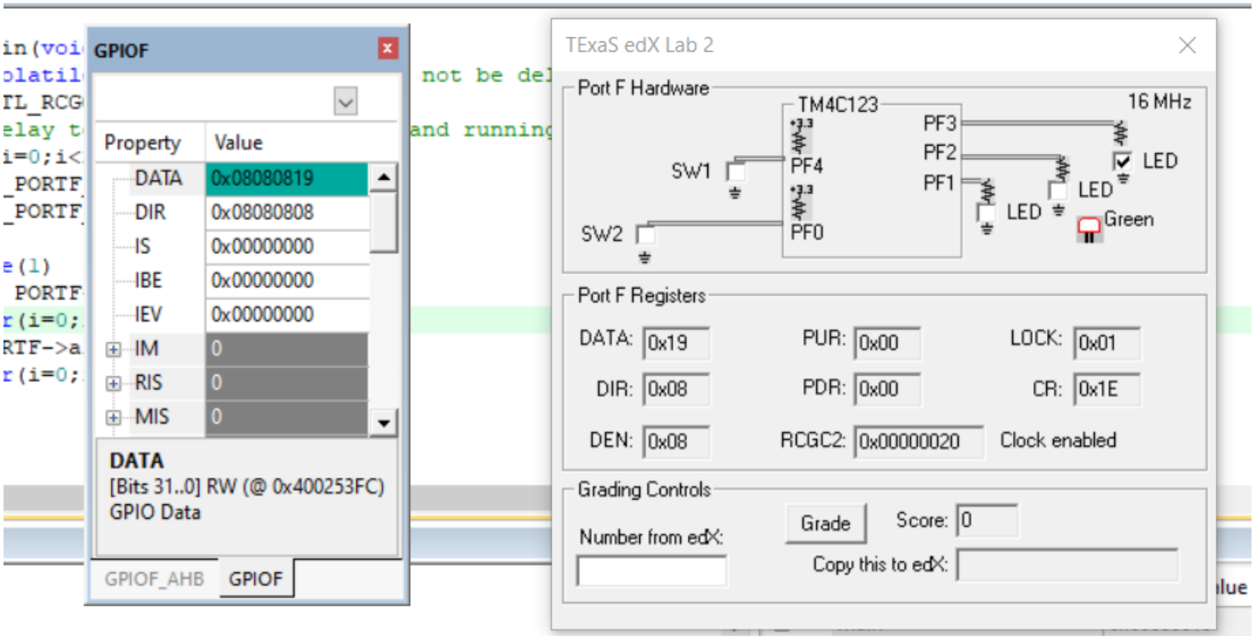
DEN: 0x08 RCGC2: 0x00000020 Clock enabled

Grading Controls

Score: 0

i 0x0000C350 auto - int

The value of PORTF data register that changes frequently :



```
in(voi
olatil
TL_RCG
elay t
i=0;i<
_PORTF
_PORTF

e(1)
 PORTF
r(i=0;
RTF->a
r(i=0;
```

```
not be del
and running
```

GPIOF

Property	Value
DATA	0x08080819
DIR	0x08080808
IS	0x00000000
IBE	0x00000000
IEV	0x00000000
IM	0
RIS	0
MIS	0

DATA
[Bits 31..0] RW (@ 0x400253FC)
GPIO Data

TExaS edX Lab 2

Port F Registers

DATA:	0x19	PUR:	0x00	LOCK:	0x01
DIR:	0x08	PDR:	0x00	CR:	0x1E
DEN:	0x08	RCGC2:	0x00000020	Clock enabled	

Grading Controls

Number from edX:
Score: 0
Copy this to edX: