

## Assignment-4

### Lesson-4

*Name : Ayat Mohamed*



## Lab(4)

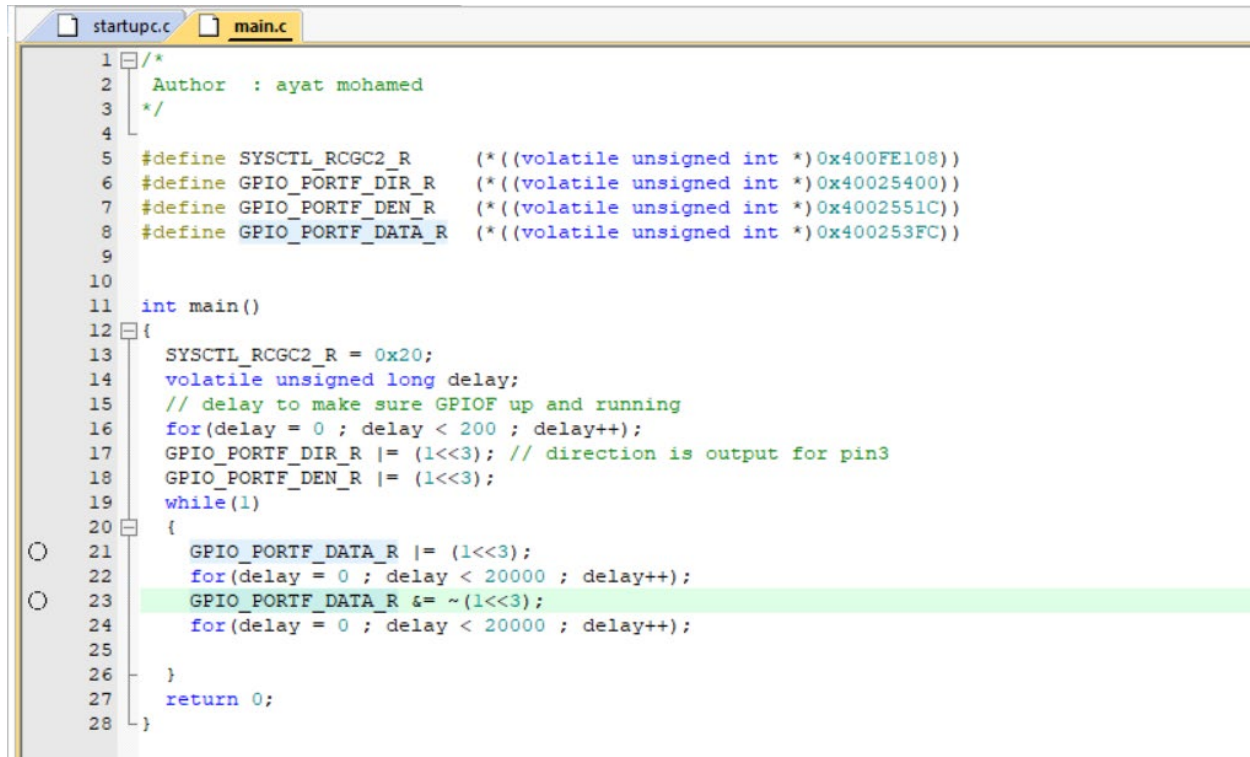
-board name :tm4c123 (arm-cortexM4) processor or called (tiva-c) kit

-in this lab we will toggle a led connected to pin3 in PORTF .

-According to specs information:

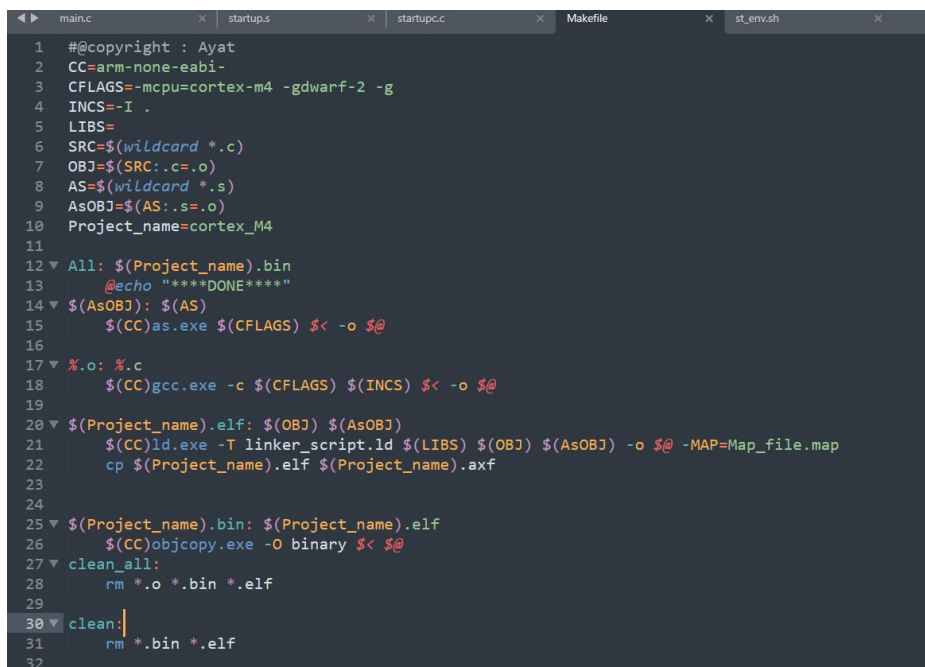
- Flash start with addres 0x00000000 and has a size 0.5GB -> 512MB.
- SRAM start with addres 0x2 0000000 and has a size 0.5GB -> 512MB.
- -SYSTCTL:is a system control ,it used to enable the clk for PORTF
- GPIO-Module has three regiters :
  - GPIO\_PORTF\_DIR : For direction(IN/OUT).
  - GPIO\_PORTF\_DEN : For enable gpio.
  - GPIO\_PORTF\_DR : For data regiter to toggel led(Assign 1 on pin3 to enable led ,and 0 on pin3 to disable led).

-main.c



```
1  /*
2  Author : ayat mohamed
3  */
4
5  #define SYSCTL_RCGC2_R      (*(volatile unsigned int *)0x400FE108)
6  #define GPIO_PORTF_DIR_R    (*(volatile unsigned int *)0x40025400)
7  #define GPIO_PORTF_DEN_R    (*(volatile unsigned int *)0x4002551C)
8  #define GPIO_PORTF_DATA_R   (*(volatile unsigned int *)0x400253FC)
9
10
11 int main()
12 {
13     SYSCTL_RCGC2_R = 0x20;
14     volatile unsigned long delay;
15     // delay to make sure GPIOF up and running
16     for(delay = 0 ; delay < 200 ; delay++);
17     GPIO_PORTF_DIR_R |= (1<<3); // direction is output for pin3
18     GPIO_PORTF_DEN_R |= (1<<3);
19     while(1)
20     {
21         GPIO_PORTF_DATA_R |= (1<<3);
22         for(delay = 0 ; delay < 20000 ; delay++);
23         GPIO_PORTF_DATA_R &= ~(1<<3);
24         for(delay = 0 ; delay < 20000 ; delay++);
25     }
26     return 0;
27 }
28 }
```

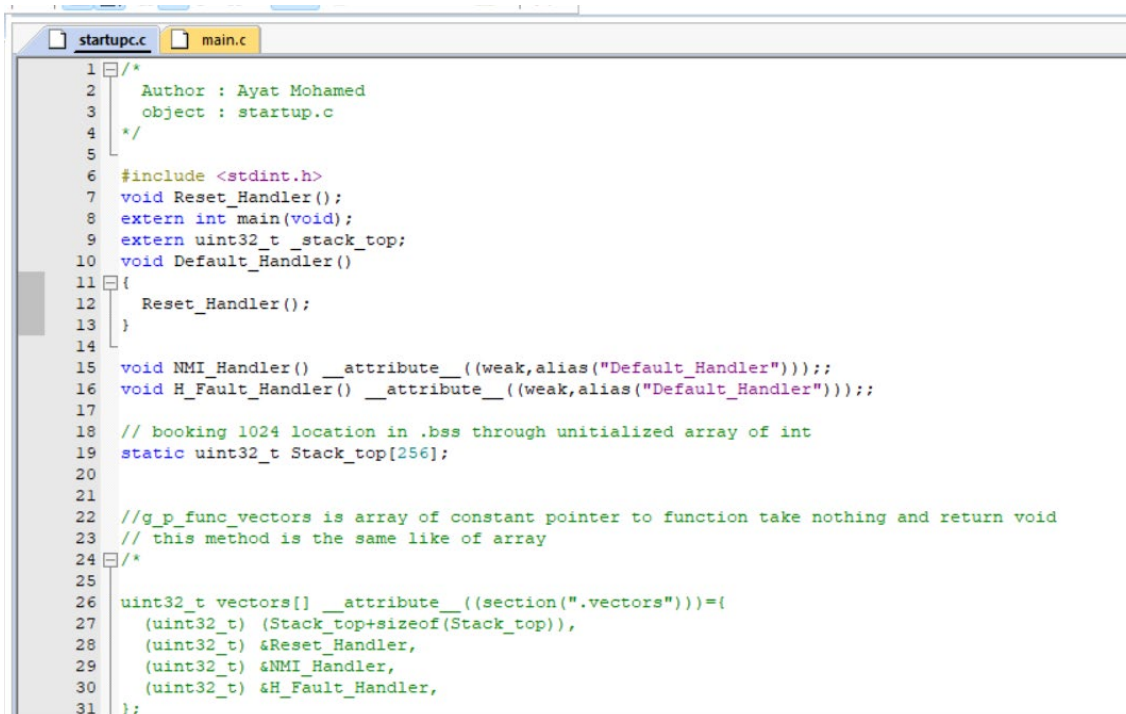
Makefile:



```
1  #@copyright : Ayat
2  CC=arm-none-eabi-
3  CFLAGS=-mcpu=cortex-m4 -gdwarf-2 -g
4  INCS=-I .
5  LIBS=
6  SRC=$(wildcard *.c)
7  OBJ=$(SRC:.c=.o)
8  AS=$(wildcard *.s)
9  ASOBJ=$(AS:.s=.o)
10 Project_name=cortex_M4
11
12 All: $(Project_name).bin
13     @echo "****DONE****"
14 $(AsOBJ): $(AS)
15     $(CC)as.exe $(CFLAGS) $< -o $@
16
17 %.o: %.c
18     $(CC)gcc.exe -c $(CFLAGS) $(INCS) $< -o $@
19
20 $(Project_name).elf: $(OBJ) $(AsOBJ)
21     $(CC)ld.exe -T linker_script.ld $(LIBS) $(OBJ) $(AsOBJ) -o $@ -MAP=Map_file.map
22     cp $(Project_name).elf $(Project_name).axf
23
24
25 $(Project_name).bin: $(Project_name).elf
26     $(CC)objcopy.exe -O binary $< $@
27 clean_all:
28     rm *.o *.bin *.elf
29
30 clean:
31     rm *.bin *.elf
32
```

## *-startup.c*

- In this lab we will edit on starup.c of cortexM3*
- Use anther method to initialize SP ,instead of using stack\_top in linker\_script*
- We will use an array to define uninitialized array of integers with 256 elements to define 1024 location in memory*
- The SP will be at the end of array, then define pointer to function take nothing and return void to handle all interrupt according to interrupt vector table.*



```
1  /*
2   * Author : Ayat Mohamed
3   * object : startup.c
4   */
5
6  #include <stdint.h>
7  void Reset_Handler();
8  extern int main(void);
9  extern uint32_t _stack_top;
10 void Default_Handler()
11 {
12     Reset_Handler();
13 }
14
15 void NMI_Handler() __attribute__((weak, alias("Default_Handler")));
16 void H_Fault_Handler() __attribute__((weak, alias("Default_Handler")));
17
18 // booking 1024 location in .bss through uninitialized array of int
19 static uint32_t Stack_top[256];
20
21
22 //g_p_func_vectors is array of constant pointer to function take nothing and return void
23 // this method is the same like of array
24 /*
25
26 uint32_t vectors[] __attribute__((section(".vectors")))={
27     (uint32_t) (Stack_top+sizeof(Stack_top)),
28     (uint32_t) &Reset_Handler,
29     (uint32_t) &NMI_Handler,
30     (uint32_t) &H_Fault_Handler,
31 };
```

```

33 void (*const g_p_func_vectors[])() __attribute__((section(".vectors"))) =
34 {
35     (void (*)()) (Stack_top+sizeof(Stack_top)),
36     &Reset_Handler,
37     &NMI_Handler,
38     &H_Fault_Handler,
39 };
40
41 extern uint32_t _E_text;
42 extern uint32_t _S_DATA;
43 extern uint32_t _E_DATA;
44 extern uint32_t _S_bss;
45 extern uint32_t _E_bss;
46
47
48 void Reset_Handler()
49 {
50     // copy data section from flash to SRAM
51     uint32_t DATA_Size = (unsigned char*)&_E_DATA - (unsigned char*)&_S_DATA ;
52     unsigned char * P_src = (unsigned char *)&_E_text;
53     unsigned char * P_dst= (uint8_t*)&_S_DATA;
54     for (int i = 0 ; i < DATA_Size ; i ++)
55     {
56         *((unsigned char*)P_dst++) = *((unsigned char*)P_src++);
57     }
58
59     //init .bss section in SRAM = 0;
60     uint32_t BSS_Size = (unsigned char*)&_E_bss - (unsigned char*)&_S_bss;
61     unsigned char* bss_dst= (unsigned char*)&_S_bss;
62     for (int i = 0 ; i < BSS_Size ; i ++)
63     {
64         *((unsigned char*)bss_dst++) = (unsigned char)0x00;
65     }
66     //jump on main
67     main();
68 }

```

*-linker\_script*

*- stack\_top is deleted*

```
1  /* Author : Ayat mohamed
2     Linker_script : cortex_M3
3  */
4
5  MEMORY
6  {
7     FLASH(RX) : ORIGIN = 0x00000000 , LENGTH = 512M
8     SRAM(RWX) : ORIGIN = 0x20000000 , LENGTH = 512M
9  }
10
11  SECTIONS
12  {
13     .text :
14     {
15         *(.vectors*)
16         *(.text*)
17         *(.rodata)
18         _E_text = .;
19     }>FLASH
20     .data :
21     {
22         _S_DATA = .;
23         *(.data)
24         . = ALIGN(4);
25         _E_DATA = .;
26     }>SRAM AT> FLASH
27     .bss :
28     {
29         _S_bss = .;
30         *(.bss)
31         _E_bss = .;
32         . = ALIGN(4);
33         . = . + 0X1000;
34     }
35     }>SRAM
36  }
```

## *-Map file*

- vector section start at 0x00000000
- .bss section start at 0x20000000 and end at 0x20000400 ,where 0x400 -> 1024 in decimal

```
Q@Ayat-Mohamed MINGW64 /e/KEROLoS_Diploma/embedded_repo/Embedded_system_online_diploma/C_programming/Unit_3/Lesson_4 (master)
$ make
arm-none-eabi-gcc.exe -c -mcpu=cortex-m4 -gdwarf-2 -g -I . main.c -o main.o
arm-none-eabi-ld.exe -T linker_script.ld main.o startupc.o -o cortex_M4.elf -M
AP=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	0x12c	
*(.vectors*)			
.vectors	0x00000000	0x10	startupc.o
	0x00000000		g_p_func_vectors
*(.text*)			
.text	0x00000010	0x8c	main.o
	0x00000010		main
.text	0x0000009c	0x90	startupc.o
	0x0000009c		H_Fault_Handler
	0x0000009c		Default_Handler
	0x0000009c		NMI_Handler
	0x000000a8		Reset_Handler
*(.rodata)			
	0x0000012c		_E_text = .

```

.data          0x20000000      0x0 load address 0x0000012c
               0x20000000      _S_DATA = .

*(.data)
.data          0x20000000      0x0 main.o
.data          0x20000000      0x0 startupc.o
               0x20000000      . = ALIGN (0x4)
               0x20000000      _E_DATA = .

.igot.plt      0x20000000      0x0 load address 0x0000012c
.igot.plt      0x20000000      0x0 main.o

.bss           0x20000000      0x1400 load address 0x0000012c
               0x20000000      _S_bss = .

*(.bss)
.bss           0x20000000      0x0 main.o
.bss           0x20000000      0x400 startupc.o
               0x20000400      _E_bss = .
               0x20000400      . = ALIGN (0x4)
               0x20001400      . = (. + 0x1000)
*fill*         0x20000400      0x1000

```

The screenshot displays the TExaS edX Lab 2 IDE interface. The main window shows the C source code for `main.c`, which includes a delay function and GPIO setup for a pin. The left sidebar shows the project structure with `main.c` selected. The right sidebar shows the hardware configuration for the TM4C123 microcontroller, including pin settings for PF4, PF2, and PF1, and the configuration of the internal pull-up resistor (PDR) and clock (RCCG2).

**Registers:**

Register	Value
R0	0x00000000
R1	0x00000000
R2	0x400251C
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x20000FC8
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000FC8
R14 (LR)	0x00000111
R15 (PC)	0x00000048
PCSR	0x21000000

**main.c Code:**

```

15 // delay to make sure GPIOF up and running
16 for(delay = 0 ; delay < 200 ; delay++);
17 GPIO_PORTF_DIR_R |= (1<<3); // direction is output for pin3
18 GPIO_PORTF_DEN_R |= (1<<3);
19 while(1)
20 {
21     GPIO_PORTF_DATA_R |= (1<<3);
22     for(delay = 0 ; delay < 20000 ; delay++);
23     GPIO_PORTF_DATA_R &= ~(1<<3);
24     for(delay = 0 ; delay < 20000 ; delay++);
25 }
26 return 0;
27
28

```

**Hardware Configuration (TM4C123):**

- Port F Hardware: SW1, SW2, PF4, PF2, PF1, LED, LED.
- Port F Registers: DATA: 0x11, DIR: 0x08, DEN: 0x08, PUR: 0x00, PDR: 0x00, RCGC2: 0x00000020, LOCK: 0x01, CR: 0x1E.
- Grading Controls: Number from edX, Grade, Score: 0, Copy this to edX.

**Call Stack - Locals:**

Name	Location/Value	Type
main	0x00000010	int f0
delay	0x000000C8	auto - uint



Registers

Register	Value
R0	0x00000000
R1	0x00000000
R2	0x4002551C
R3	0x00000008
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x20000FC8
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000FC8
R14 (LR)	0x00000111
R15 (PC)	0x00000446
SPSR	0x21000000

Logic Analyzer

Command Window

Save...

0s

1.487437ms

0.1s

Zoom

Min/Max

Update Screen

Transition

Jump to

Signal Info

Amplitude

Timestamps Enable

Disassembly

Logic Analyzer

System Analyzer

startup.c

main.c

```

15 // delay to make sure GPIOF up and running
16 for(delay = 0 ; delay < 200 ; delay++);
17 GPIO_PORTF_DIR_R |= (1<<3); // direction is output for pin3
18 GPIO_PORTF_DEN_R |= (1<<3);
19 while(1)
20 {
21     GPIO_PORTF_DATA_R |= (1<<3);
22     for(delay = 0 ; delay < 20000 ; delay++);
23     GPIO_PORTF_DATA_R &= ~(1<<3);
24     for(delay = 0 ; delay < 20000 ; delay++);
25 }
26 }
27 return 0;
28

```

Property

Value

DATA	0x00000111
DIR	0x08080808
IS	0x00000000
IBE	0x00000000
IEV	0x00000000
IM	0
RIS	0
MIS	0
ICR	0
AFSEL	0x00000000
DR2R	0xFFFFFFFF
DR4R	0x00000000
DR8R	0x00000000
ODR	0x00000000
PUR	0x00000000
PDR	0x00000000
SLR	0x00000000
DEN	0x08080808
LOCK	0x01010101
CR	0x1E1E1E1E
ARRSEL	0x00000000

Command

Call Stack - Locals

```

BS \\cortex_M4\\main.c\\21
BS \\cortex_M4\\main.c\\23
LA (PORTF & 0xF) >> 3

```

Name	Location/Value	Type
main	0x00000010	int f0
delay	0x000000C8	auto - uint

Port F Hardware

Port F Registers

DATA: 0x19

PUR: 0x00

LOCK: 0x01

DIR: 0x08

PDR: 0x00

CR: 0x1E

DEN: 0x08

RCGC2: 0x00000020

Clock enabled

Grading Controls

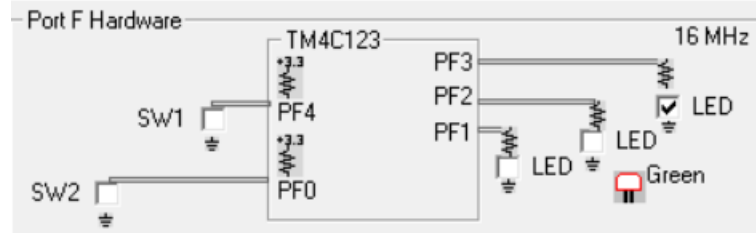
Number from edX:

Grade

Score: 0

Copy this to edX:

## TEXaS edX Lab 2



Port F Registers

DATA: <input type="text" value="0x19"/>	PUR: <input type="text" value="0x00"/>	LOCK: <input type="text" value="0x01"/>
DIR: <input type="text" value="0x08"/>	PDR: <input type="text" value="0x00"/>	CR: <input type="text" value="0x1E"/>
DEN: <input type="text" value="0x08"/>	RCGC2: <input type="text" value="0x00000020"/>	Clock enabled

Grading Controls

Number from edX:

Grade:  Score:

Copy this to edX:

Property	Value
DATA	0x08080819
DIR	0x08080808
IS	0x00000000
IBE	0x00000000
IEV	0x00000000
IM	0
RIS	0
MIS	0
ICR	0
AFSEL	0x00000000
DR2R	0xFFFFFFFF
DR4R	0x00000000
DR8R	0x00000000
ODR	0x00000000
PUR	0x00000000
PDR	0x00000000

Logic Analyzer

Setup... Load... Save... Min Time 0 s Max Time 5.884022 s Grid 50 ms Zoom In Out All Min/Max Auto Undo Update Screen Stop Clear Transition Prev Next Jump to Code Trace Signal Info Amplitude Timestamps Enable

PORTF

34 ms 0.15892 s 0.65892 s

Disassembly Logic Analyzer System Analyzer

startup.c main.c

```

15 // delay to make sure GPIOF up and running
16 for(delay = 0 ; delay < 200 ; delay++);
17 GPIO_PORTF_DIR_R |= (1<<3); // direction is output for pin3
18 GPIO_PORTF_DEN_R |= (1<<3);
19 while(1)
20 {
21     GPIO_PORTF_DATA_R |= (1<<3);
22     for(delay = 0 ; delay < 20000 ; delay++);
23     GPIO_PORTF_DATA_R ^= (1<<3);
24     for(delay = 0 ; delay < 20000 ; delay++);
25 }
26 }
27 return 0;
28 }
    
```

TEXaS edX Lab 2

Port F Hardware

Port F Registers

DATA: <input type="text" value="0x19"/>	PUR: <input type="text" value="0x00"/>	LOCK: <input type="text" value="0x01"/>
DIR: <input type="text" value="0x08"/>	PDR: <input type="text" value="0x00"/>	CR: <input type="text" value="0x1E"/>
DEN: <input type="text" value="0x08"/>	RCGC2: <input type="text" value="0x00000020"/>	Clock enabled

Grading Controls

Number from edX:

Grade:  Score:

Copy this to edX:

Property Value

DATA	0x08080819
DIR	0x08080808
IS	0x00000000
IBE	0x00000000
IEV	0x00000000
IM	0
RIS	0
MIS	0
ICR	0
AFSEL	0x00000000
DR2R	0xFFFFFFFF
DR4R	0x00000000
DR8R	0x00000000
ODR	0x00000000
PUR	0x00000000
PDR	0x00000000
SLR	0x00000000
DEN	0x08080808
LOCK	0x01010101
CR	0x1E1E1E1E
AMSEL	0x00000000

Logic Analyzer

Setup... Load... Save... ?

Min Time 0 s Max Time 38.85105 s Grid 10 ms

Zoom In Out All Auto Undo

Min/Max Update Screen Transition Prev Next Jump to Code Trace

Signal Info Show Cycles Amplitude Cursor

Timestamps Enable

1

PORTF

0

0.38452 s

0.48452 s

Disassembly Logic Analyzer System Analyzer

startup.c main.c

```
15 // delay to make sure GPIOF up and running
16 for(delay = 0 ; delay < 200 ; delay++);
17 GPIO_PORTF_DIR_R |= (1<<3); // direction is output for pin3
18 GPIO_PORTF_DEN_R |= (1<<3);
19 while(1)
20 {
21     GPIO_PORTF_DATA_R |= (1<<3);
22     for(delay = 0 ; delay < 20000 ; delay++);
23     GPIO_PORTF_DATA_R &= ~(1<<3);
24     for(delay = 0 ; delay < 20000 ; delay++);
25 }
26 }
27 return 0;
28 }
```

Texas edX Lab 2

Port F Hardware

Port F Registers

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

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

DEN: 0x08 RCGC2: 0x00000020 Clock enabled

Grading Controls

Number from edX:

Grade Score: 0

Copy this to edX:

GPIOF

Property	Value
DATA	0x00000011
DIR	0x08080808
IS	0x00000000
IBE	0x00000000
IEV	0x00000000
IM	0
MIS	0
RIS	0
ICR	0
AFSEL	0x00000000
DR2R	0xFFFFFFFF
DR4R	0x00000000
DR8R	0x00000000
ODR	0x00000000
PUR	0x00000000
PDR	0x00000000
SLR	0x00000000
DEN	0x08080808
LOCK	0x01010101
CR	0x1E1E1E1E
AFSEL	0x00000000