Learn-In-Depth

Be Professional in Embedded System

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

```c
/*
 Author  : ayat mohamed
*/

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


int main()
{
  SYSCTL_RCGC2_R = 0x20;
  volatile unsigned long delay;
  // delay to make sure GPIOF up and running
  for(delay = 0 ; delay < 200 ; delay++);
  GPIO_PORTF_DIR_R |= (1<<3); // direction is output for pin3
  GPIO_PORTF_DEN_R |= (1<<3);
  while(1)
  {
    GPIO_PORTF_DATA_R |= (1<<3);
    for(delay = 0 ; delay < 20000 ; delay++);
    GPIO_PORTF_DATA_R &= ~(1<<3);
    for(delay = 0 ; delay < 20000 ; delay++);

  }
  return 0;
}
```

Makefile:

```makefile
#@copyright : Ayat
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=cortex_M4

All: $(Project_name).bin
    @echo "****DONE****"
$(AsOBJ): $(AS)
    $(CC)as.exe $(CFLAGS) $< -o $@

%.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 $@ -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 *.bin *.elf

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

-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.

startupc.c main.c

```c
/*
  Author : Ayat Mohamed
  object : startup.c
*/

#include <stdint.h>
void Reset_Handler();
extern int main(void);
extern uint32_t _stack_top;
void Default_Handler()
{
  Reset_Handler();
}

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

// booking 1024 location in .bss through unitialized array of int
static uint32_t Stack_top[256];


//g_p_func_vectors is array of constant pointer to function take nothing and return void
// this method is the same like of array
/*

uint32_t vectors[] __attribute__((section(".vectors")))={
  (uint32_t) (Stack_top+sizeof(Stack_top)),
  (uint32_t) &Reset_Handler,
  (uint32_t) &NMI_Handler,
  (uint32_t) &H_Fault_Handler,
};
```

```c
void (*const g_p_func_vectors[])() __attribute__((section(".vectors"))) =
{
  (void (*)()) (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 SRAM
  uint32_t DATA_Size = (unsigned char*)&_E_DATA - (unsigned char*)&_S_DATA ;
  unsigned char * P_src = (unsigned char *)&_E_text;
  unsigned char  * P_dst= (uint8_t*)&_S_DATA;
  for (int i = 0 ;  i < DATA_Size ; i ++)
  {
    *((unsigned char*)P_dst++) = *((unsigned char*)P_src++);
  }
```

```c
  //init .bss section in SRAM = 0;
  uint32_t BSS_Size = (unsigned char*)&_E_bss - (unsigned char*)&_S_bss;
  unsigned char* bss_dst= (unsigned char*)&_S_bss;
  for (int i = 0 ;  i < BSS_Size ; i ++)
  {
    *((unsigned char*)bss_dst++) = (unsigned char)0x00;
  }
  //jump on main
  main();
}
```

-linker_script

- stack_top is deleted

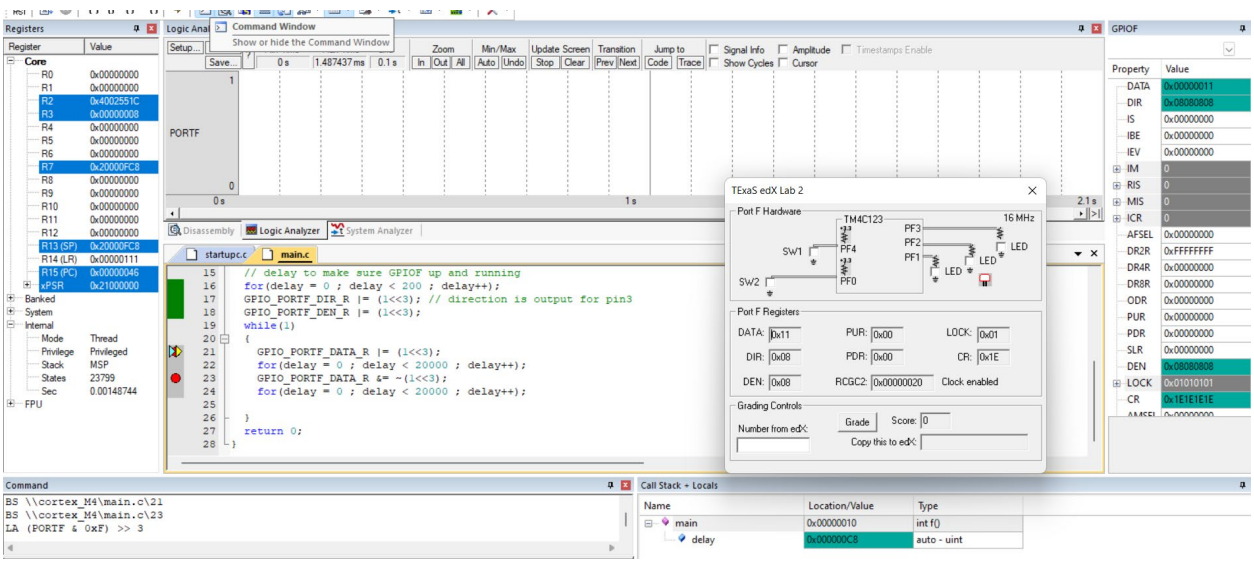
```
/* Author : Ayat mohamed
   Linker_script : cortex_M3
 */

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 = .;
      . = ALIGN(4);
      . = . + 0X1000;

   }>SRAM
}
```

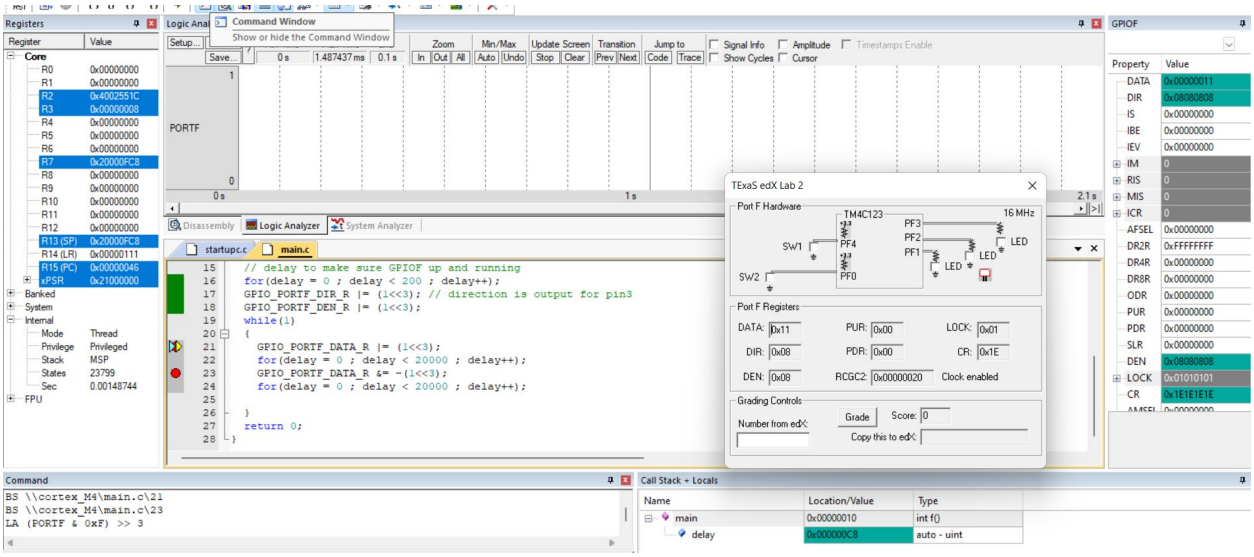
-Map file

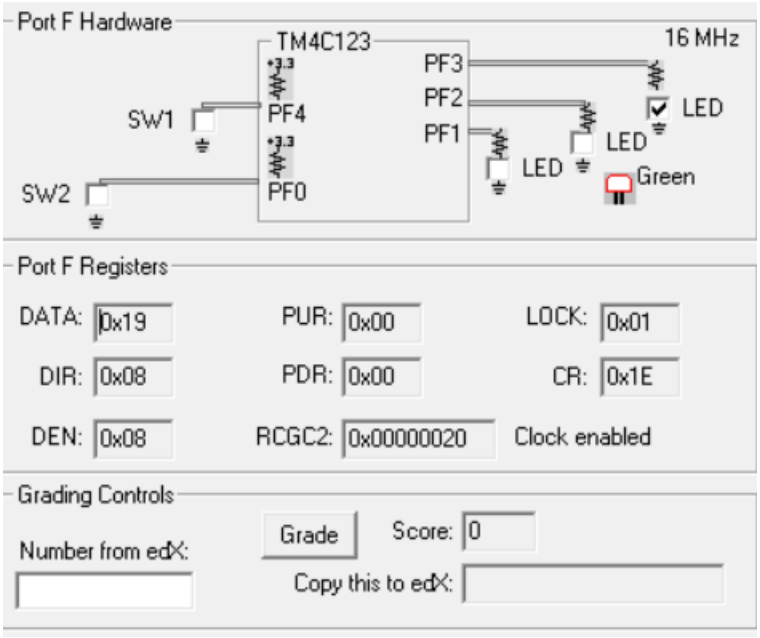
- vector section start at 0x00000000
- .bss section start at 0x20000000 and end at 0x20000400 ,where 0x400 -> 1024 in decimal

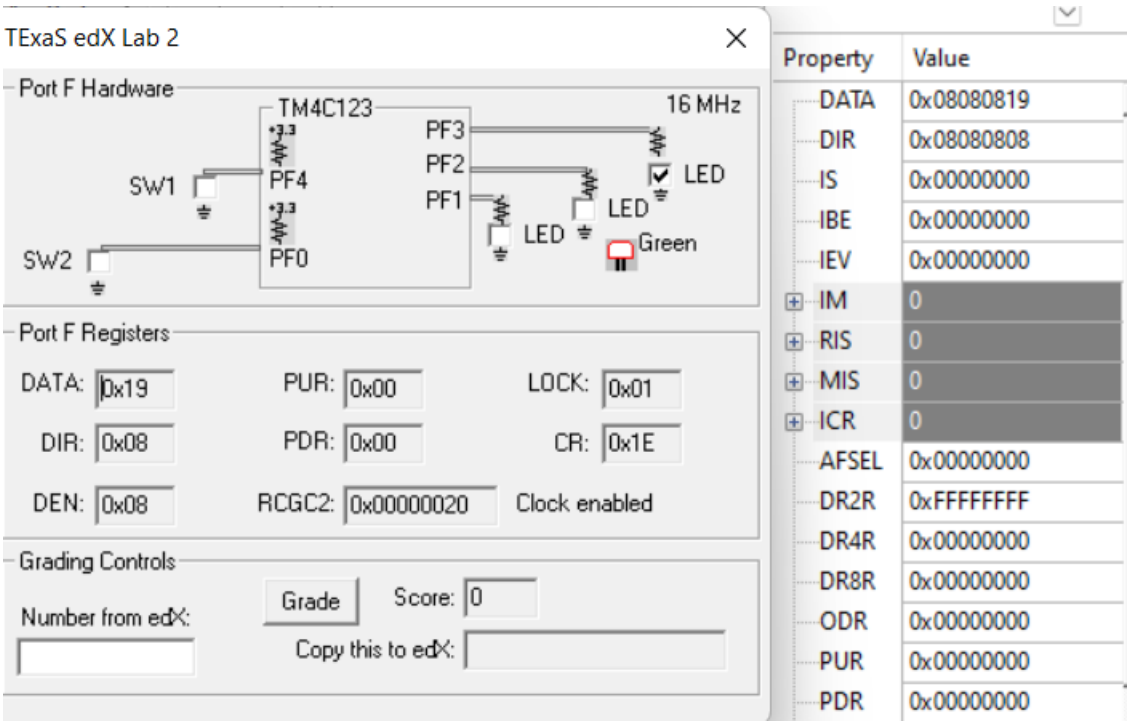
```
Q@Ayat-Mohamed MINGW64 /e/KEROLOS_Diploma/embedded_repo/Embedded_system_online_d
iploma/C_programming/Unit_3/Lesson_4 (master)
$ make
arm-none-eabi-gcc.exe -c -mcpu=cortex-m4 -gdwarf-2 -g -I . main.c -o main.o
arm-none-eabi-ld.exe -T linker_script.ld  main.o startupc.o  -o cortex_M4.elf -M
AP=Map_file.map

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 = .
```

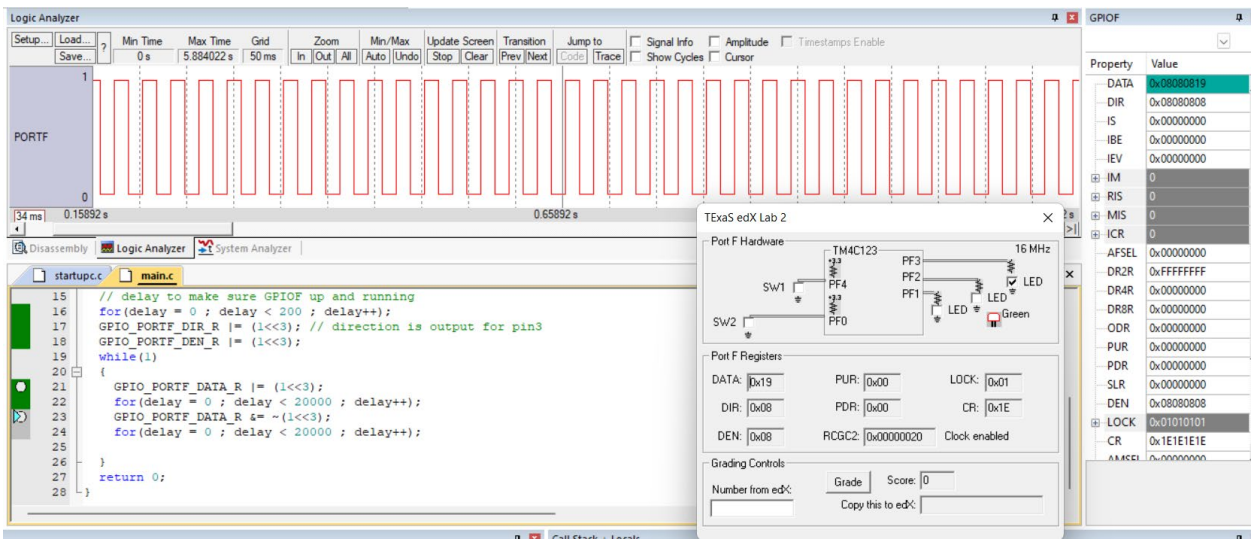
```
.data           0x20000000        0x0 load address 0x0000012c
                0x20000000                _S_DATA = .
 *(.data)
 .data          0x20000000        0x0 main.o
 .data          0x20000000        0x0 startupc.o
                0x20000000                . = ALIGN (0x4)
                0x20000000                _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
```

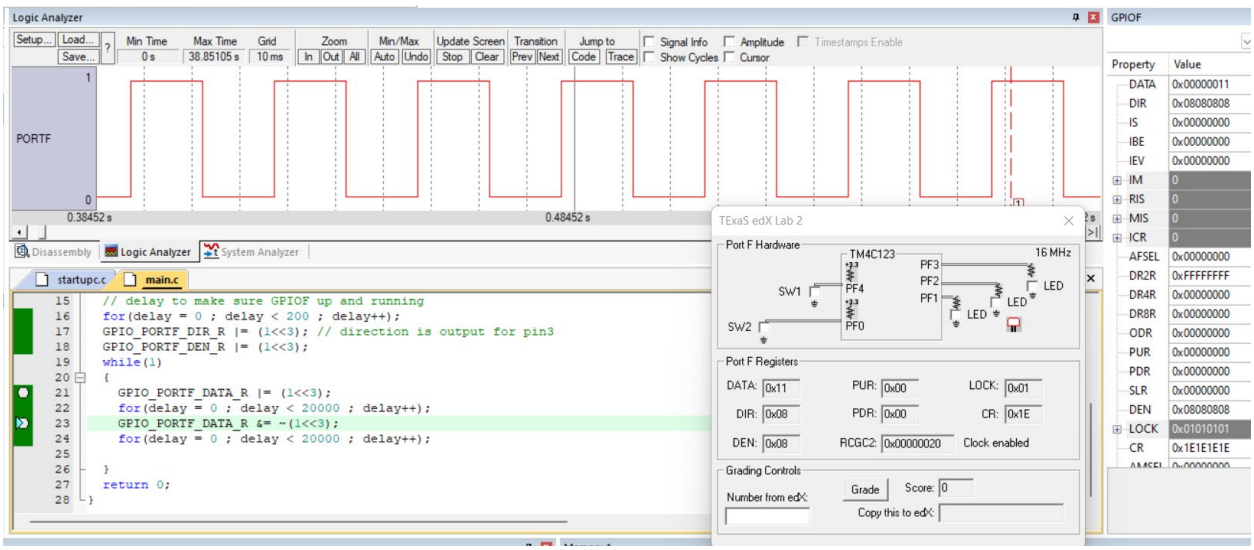



Registers
Logic Analyzer
Command Window
Show or hide the Command Window
GPIOF
PORTF
startup.c
main.c
// delay to make sure GPIOF up and running
for(delay = 0 ; delay < 200 ; delay++);
GPIO_PORTF_DIR_R |= (1<<3); // direction is output for pin3
GPIO_PORTF_DEN_R |= (1<<3);
while(1)
{
GPIO_PORTF_DATA_R |= (1<<3);
for(delay = 0 ; delay < 20000 ; delay++);
GPIO_PORTF_DATA_R &= ~(1<<3);
for(delay = 0 ; delay < 20000 ; delay++);
}
return 0;
}
TExaS edX Lab 2
Port F Hardware
TM4C123
16 MHz
SW1
SW2
PF4
PF0
PF3
PF2
PF1
LED
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:
Command
BS \\cortex_M4\main.c\21
BS \\cortex_M4\main.c\23
LA (PORTF & 0xF) >> 3
Call Stack + Locals
Name
Location/Value
Type
main
0x00000010
int f()
delay
0x000000C8
auto - uint


Port F Hardware
TM4C123
16 MHz
SW1
SW2
PF4
PF0
PF3
PF2
PF1
LED
LED
LED
Green
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 Hardware
TM4C123
16 MHz
PF3
PF2
PF1
PF4
PF0
SW1
SW2
LED
LED
LED
Green
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:
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
PORTF
startupc.c
main.c
// delay to make sure GPIOF up and running
for(delay = 0 ; delay < 200 ; delay++);
GPIO_PORTF_DIR_R |= (1<<3); // direction is output for pin3
GPIO_PORTF_DEN_R |= (1<<3);
while(1)
{
GPIO_PORTF_DATA_R |= (1<<3);
for(delay = 0 ; delay < 20000 ; delay++);
GPIO_PORTF_DATA_R &= ~(1<<3);
for(delay = 0 ; delay < 20000 ; delay++);
}
return 0;
}
GPIOF
TExaS edX Lab 2


Logic Analyzer
Setup... Load... Save...
Min Time 0 s
Max Time 38.85105 s
Grid 10 ms
Zoom In Out All
Min/Max Auto Undo
Update Screen Stop Clear
Transition Prev Next
Jump to Code Trace
Signal Info
Show Cycles
Amplitude
Cursor
Timestamps Enable
PORTF
1
0
0.38452 s
0.48452 s
Disassembly
Logic Analyzer
System Analyzer
startupc.c
main.c
// delay to make sure GPIOF up and running
for(delay = 0 ; delay < 200 ; delay++);
GPIO_PORTF_DIR_R |= (1<<3); // direction is output for pin3
GPIO_PORTF_DEN_R |= (1<<3);
while(1)
{
GPIO_PORTF_DATA_R |= (1<<3);
for(delay = 0 ; delay < 20000 ; delay++);
GPIO_PORTF_DATA_R &= ~(1<<3);
for(delay = 0 ; delay < 20000 ; delay++);
}
return 0;
}
TExaS edX Lab 2
Port F Hardware
TM4C123
16 MHz
SW1
SW2
PF4
PF0
PF3
PF2
PF1
LED
LED
LED
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
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