Jordan Murti
ECE 474 UW
Lab 2

**Procedure**

Before starting my lab, I decided to do a little bit of organizing. I separated each lab demonstration into it's on function and had the main run different functions. For the first section the code for a rainbow light was modified to use the timer, and for use of the timer, the right initialization was done with help from the datasheet. The while loop does nothing whilst checking for the timer to reach 0, then does the act and resets the timer. The traffic lights portion and the lab was more of reworking of the original with the challenge of using while loops instead of for loops that checked for change conditions. The change conditions included that if neither of the buttons were pressed, the timer for the change would reset, and to exit the loop would require the secondary timers to meet zero, as well as the buttons being pressed corresponding to their action. For the interrupt initial portion the cstartup_M.c file was modified appropriately and the initialization to allow for interrupts was done as well. The rainbow sequence was done through a while loop doing nothing, and a separate function that would cause iteration through the lights and resetting the timer. The second interrupt function had one interrupt function for the timer working similar to the previous, and a new interrupt function that would stop the timer if one button was pressed, and continue the timer if the other value was pressed. For the last portion I did not get far, and only attempted to write code to iterate to new states for lights.

**Results**

The entirety of the of lab worked as to be expected except for the last part of traffic lights. The use of timers worked as well as the code to introduce interrupts. The last traffic light portion failed to work correctly even with the task of iterating through different colored lights more time spent on the project could have fixed this.

**0x63 Problems**

The first part to demonstrate wasn't terribly brutal or unfamiliar. I used the data sheet and instructions from the lab. The second part had a bit of a hiccup because I did it on a table with a table cloth and realized that me hands being near it were changing the functions so I concluded there was a lot of static. I moved to the glass table and it worked fine. The next portion (interrupts) although very challenging did feel pretty cool when it worked, especially in knowing that i going BEYOND the level of machine code. For one section I spent a super long time and even one of the TA's helping me couldn't figure it out until we realized that En0 was set to 0xE000E000 when it should be 0xE000E100.

## Appendix

### Lab 2 Part 1

```
lab2p1 - Printed on 7/21/2019 10:15:26 PM
Page 1
#include "lab1p1header.h" #include <stdint.h> //unsigned long COLOR = 0x2; int j;
void initializelab2p1(){ RCGCTIMER = 0x1;//set timer0 to 1. Enableing the use of the time GPTMCTL_Timer_0 &= ~0x1;//pauses the timer GPTMCFG_Timer_0 = 0x0;//sets the 16bit it to 32 bit. GPTMCAMR_Timer_0 = 0x2;//pet to periodic timer mode - 0x1 = one shot, 0x3 = capture GPTMCAMR_Timer_0 &= ~0x10;//set timer to count down GPTMTAILR_Timer_0 = Seconds;//countdown to begin at 160000000 GPTMCTL_Timer_0 |= 0x1;//starts the timer
RCGCGPIO = PORT_F; // enable Port F GPIO PortF_DIR = 0xE; // set ALL OF PF as output PortF_DEN = 0xE; // enable digital pin ALL PF GPIO_PORTF_DATA_R = COLOR; // set PF1 to 1 and the other port F pins to 0
}
void lab2p1() { initializelab2p1();
while(1) { if(COLOR<0xE)//I figured the pins next to it were the other lights, turns out my lucky guess of left shifting was right, COLOR += 0x2;//counting from 0010 to 1110 should cover all the values else COLOR=0x2;//then reset after 111
GPIO_PORTF_DATA_R = COLOR;
//GPTMCTL_Timer_0 |= 0x1;//starts the timer///no i do not need 2 do dis while((GPTMRIS_Timer_0 & 0x1) == 0x0){ //do nothing; } //GPTMCTL_Timer_0 &= ~0x1;//stops the timer///no i do not need 2 do dis GPTMICR_Timer_0 |= 0x1;
//GPTMTAILR_Timer_0 = 0xF42400;//reloads countdown to begin at 160000000
```

### Lab 2 Part 2

```
#include "lab1p1header.h"
#include <stdint.h>
//unsigned long COLOR = 0x2;
int j;
//int stay;


void LED_initALLLL2(void)
{
 RCGCGPIO |= 0x1; // enable Port A GPIO clock
volatile unsigned long delay;
SYSCTL_RCGC2_R |= 0x01; // activate clock for Port A//Do I really need this?Ill assume so
delay = SYSCTL_RCGC2_R; // allow time for clock to start//Do I really need this?
 GPIO_PORTA_PCTL_R &= ~0x00000F00; // regular GPIO//no idea what this does but i moved the F to the port A spot
 GPIO_PORTA_AMSEL_R &= ~0xF4; // disable analog function of ALL PA Used - 11101100
 GPIO_PORTA_DIR_R |= 0xE0; // set PA7-5 to output 1110000
 GPIO_PORTA_AFSEL_R &= ~0xF4; // regular port function
 GPIO_PORTA_DEN_R |= 0xF4; // enable digital on ALL PA USED
}

```

```
void initializelab2p2(){
RCGCTIMER = 0x1;//set timer0 to 1. Enableing the use of the time


GPTMCTL_Timer_0 &= ~0x101;//pauses the timer (both a and b)
GPTMCFG_Timer_0 = 0x0;//sets the 16bit it to 32 bit.

GPTMCAMR_Timer_0 = 0x2;//pet to periodic timer mode - 0x1 = one shot, 0x3 = capture
GPTMCAMR_Timer_0 &= ~0x10;//set timer to count down

GPTMCBMR_Timer_0 = 0x2;//set B to periodic timer mode
GPTMCBMR_Timer_0 &= ~0x10;//set B timer to count down

GPTMTAILR_Timer_0 = Seconds*5;//countdown to begin at 160000000*5
GPTMTBILR_Timer_0 = Seconds*2;//countdown to begin at 160000000*2

GPTMCTL_Timer_0 |= 0x101;//starts both the timers


}


void lab2p2(){////////////////////////////////////MAIN FOR THE 3RD LAB PART(THE NUMBER is swapped out for use )
  LED_initALLLL2();
  initializelab2p2();
  state = GO;

RCGCGPIO |= PORT_F; // enable Port F GPIO//program doesnt work without it, dont ask me why...
PortF_DIR = 0xE; // set ALL OF PF as output
PortF_DEN = 0xE; // enable digital pin ALL PF
GPIO_PORTF_DATA_R = COLOR; // set PF1 to 1 and the other port F pins to 0



while(1){

  if (state == GO){
      GoLED();
      state = STOP;
      while(((GPTMRIS_Timer_0 & 0x1) == 0x0)){

         if ((switch_inputPED() == 0x0) && (switch_inputSS() == 0x0)){////////////////interupt if pedestrian light pressed
           GPTMICR_Timer_0 |= 0x100;//resets timer B
        }


```

```
        if ((switch_inputPED() == 0x1)  && (GPTMRIS_Timer_0 & 0x100 == 0x100))
{///////////////interupt if pedestrian light pressed
          //stay = 0;
          state = WAIT;
          GPTMICR_Timer_0 |= 0x1;//resets A
          GPTMICR_Timer_0 |= 0x100;//resets B
          break;
        }

        if ((switch_inputSS() == 0x1)  && (GPTMRIS_Timer_0 & 0x100 == 0x100))
{///////////////interupt if pedestrian light pressed
          state = OFF;
          GPTMICR_Timer_0 |= 0x1;//resets A
          GPTMICR_Timer_0 |= 0x100;//resets timer B
          break;
        }


      }
  GPTMICR_Timer_0 |= 0x1;//resetsA
  }



  if (state == STOP){

    StopLED();
    state = GO;

  while((GPTMRIS_Timer_0 & 0x1) == 0x0){

         if ((switch_inputPED() == 0x0) && (switch_inputSS() == 0x0))
{///////////////interupt if pedestrian light pressed
          GPTMICR_Timer_0 |= 0x100;//resets timer B
        }



        if ((switch_inputSS() == 0x1)  && (GPTMRIS_Timer_0 & 0x100 == 0x100))
{///////////////interupt if pedestrian light pressed
          state = OFF;
          GPTMICR_Timer_0 |= 0x1;//resets A
          GPTMICR_Timer_0 |= 0x100;//resets timer B
          break;
        }
  }
  GPTMICR_Timer_0 |= 0x1;//resets
  }


```

```
   if (state == WAIT){

            WaitLED();
            state = STOP;
          while((GPTMRIS_Timer_0 & 0x1) == 0x0){
                        if ((switch_inputPED() == 0x0) && (switch_inputSS() == 0x0))
{///////////////interupt if pedestrian light pressed
                  GPTMICR_Timer_0 |= 0x100;//resets timer B
                }



                if ((switch_inputSS() == 0x1)  && (GPTMRIS_Timer_0 & 0x100 == 0x100))
{///////////////interupt if pedestrian light pressed
                  state = OFF;
                  GPTMICR_Timer_0 |= 0x1;//resets A
                  GPTMICR_Timer_0 |= 0x100;//resets timer B
                  break;
                }
          }

          GPTMICR_Timer_0 |= 0x1;//resets
    }

    if(state == OFF){

        OffLED();
        if ( switch_inputSS() == 0) {///////////////interupt if pedestrian light pressed
                  GPTMICR_Timer_0 |= 0x1;//resets A
                  GPTMICR_Timer_0 |= 0x100;//resets timer B
        }


        if ( (switch_inputSS() == 1) && (GPTMRIS_Timer_0 & 0x100 == 0x100))
{///////////////interupt if pedestrian light pressed
            state = STOP;
            GPTMICR_Timer_0 |= 0x1;//resetsA
            GPTMICR_Timer_0 |= 0x100;//resets timer B... yah i guesss i could do both
          }

    }









}
```

```
}
```

Lan 2 part 3

```
#include "lab1p1header.h"
//#include "lab2p1.c"

#include <stdint.h>

//unsigned long COLOR = 0x2;
//int j;
//int stay;



void initializelab2p3(){
RCGCTIMER = 0x1;//set timer0 to 1. Enableing the use of the time
GPTMCTL_Timer_0 &= ~0x1;//pauses the timer
GPTMCFG_Timer_0 = 0x0;//sets the 16bit it to 32 bit.
GPTMCAMR_Timer_0 = 0x2;//pet to periodic timer mode - 0x1 = one shot, 0x3 = capture
GPTMCAMR_Timer_0 &= ~0x10;//set timer to count down
GPTMTAILR_Timer_0 = Seconds;//countdown to begin at 160000000


GPTMIMR_Timer_0 |= 0x1;//enable interupt A
En0 |= 0x80000;


GPTMCTL_Timer_0 |= 0x1;//starts the timer

RCGCGPIO = PORT_F; // enable Port F GPIO
PortF_DIR = 0xE; // set ALL OF PF as output
PortF_DEN = 0xE; // enable digital pin ALL PF
GPIO_PORTF_DATA_R = COLOR; // set PF1 to 1 and the other port F pins to 0

}


//////////////////////////////////////////////this needs to be commented out for the
other labs to workd
 void Timer_Handler( void ){


//   int i;


```

```
  if(COLOR<0xE)//I figured the pins next to it were the other lights, turns out my lucky guess of left shifting was right,
      COLOR += 0x2;//counting from 0010 to 1110 should cover all the values
    else
      COLOR=0x2;//then reset after 111

    GPIO_PORTF_DATA_R = COLOR;
    GPIO_PORTF_DATA_R |= 0x1;///reset the timer

    }


  void lab2p3(){

    initializelab2p3();
    COLOR=0x2;

    GPIO_PORTF_DATA_R = COLOR;

    while(1)
    {
      ///do nothing,
      // dont worry you still have value
    }









  }
```

Lab 2 Part 4

```
  void lab2p4();

  void initializelab2p4(){
  RCGCTIMER = 0x1;//set timer0 to 1. Enableing the use of the time
  GPTMCTL_Timer_0 &= ~0x1;//pauses the timer
  GPTMCFG_Timer_0 = 0x0;//sets the 16bit it to 32 bit.
  GPTMCAMR_Timer_0 = 0x2;//pet to periodic timer mode - 0x1 = one shot, 0x3 = capture
  GPTMCAMR_Timer_0 &= ~0x10;//set timer to count down
  GPTMTAILR_Timer_0 = Seconds;//countdown to begin at 160000000


  RCGCGPIO = PORT_F; // enable Port F GPIO
  PortF_DIR = 0xE; // set PF1,2,3 as output
  PortF_DEN = 0xFF; // enable digital pin PF1-F5
  //GPIO_PORTF_DATA_R = 0x0; // set PF1 to 1 and the other port F pins to 0
  GPIO_LOCK = 0x4C4F434B;
```

```
GPIO_CMT=0xFF;
GPIO_PUR=0x11;



GPTMIMR_Timer_0 |= 0x1;//enable interupt A
En0 |= 0x80000;

En0 |= 0x40000000;///new enable PortF Interupt
GPIOM_Port_F = 0x11;  //enable SWITCH 0,4 interup

//GPIOM_Port_F = hello;
GPTMCTL_Timer_0 |= 0x1;//starts the timer



}




////////////////////////////////////////////this needs to be commented out for the
other labs to workd
 void Timer_Handler( void ){

     GPIO_PORTF_DATA_R ^= 0x4;
  while((GPTMRIS_Timer_0 & 0x1) == 0x0){
   //do nothing but wait...
  }

  GPTMICR_Timer_0 |= 0x1;

  //GPTMTAILR_Timer_0 = 0xF42400;//reloads countdown to begin at 160000000


  }



void PortF_Handler (void ){

  switch(GPIO_PORTF_DATA_R & 0x11){//////Look at the last bits
  case 0x01:
      GPIO_PORTF_DATA_R = 0x4;///if button1 go green...or red... whichever it is
       GPTMICR_Timer_0 |= 0x1;
      GPTMCTL_Timer_0 |= 0x1;//starts the timer

      break;
   case 0x10:
      GPIO_PORTF_DATA_R = 0x2;///other button do other color
```

```
        GPTMCTL_Timer_0 &= ~0x1;//stops the timer
        //lab2p4();
        break;
     default:
        break;
    }
    //return;
}



void lab2p4(){

  initializelab2p4();
  //GPIO_PORTF_DATA_R = 0x4;


  while(1){
    ///do nothing,
    // dont worry you still have value
  }




}
```

Lab 2 Part 5

```
void initlab2p5(){
RCGCTIMER = 0x1;//set timer0 to 1. Enableing the use of the time
GPTMCTL_Timer_0 &= ~0x1;//pauses the timer
GPTMCFG_Timer_0 = 0x0;//sets the 16bit it to 32 bit.
GPTMCAMR_Timer_0 = 0x2;//pet to periodic timer mode - 0x1 = one shot, 0x3 = capture
GPTMCAMR_Timer_0 &= ~0x10;//set timer to count down
GPTMTAILR_Timer_0 = Seconds;//countdown to begin at 160000000



// RCGCGPIO = 0x1; // enable Port A GPIO clock
 volatile unsigned long delay;
 SYSCTL_RCGC2_R |= 0x01; // activate clock for Port A//Do I really need this?Ill assume so
 delay = SYSCTL_RCGC2_R; // allow time for clock to start//Do I really need this?
 GPIO_PORTA_PCTL_R &= ~0x00000F00; // regular GPIO//no idea what this does but i moved the F to the port A spot
 GPIO_PORTA_AMSEL_R &= ~0xF4; // disable analog function of ALL PA Used - 11101100
 GPIO_PORTA_DIR_R |= 0xE0; // set PA7-5 to output 1110000
```

```
   GPIO_PORTA_AFSEL_R &= ~0xF4; // regular port function
   GPIO_PORTA_DEN_R |= 0xF4; // enable digital on ALL PA USED


  En0 |= 0x80000;//enable Timer0A interupt
  //En0 |= 0x100000;//enable Timber0B interupt
  En0 |= 0x1;//enable PortA interupt

  GPTMIMR_Timer_0 |= 0x1;//enable interupt TimerA AND TimerB
  GPIOM_Port_A |= 0x14;









  //En0 |= 0x40000000;///new enable PortF Interupt
  //GPIOM_Port_F = 0x11;  //enable SWITCH 0,4 interup

  //GPIOM_Port_F = hello;
  GPTMCTL_Timer_0 |= 0x1;//starts the timer









  }


  ////////////////////////////////////////////this needs to be commented out for the
  other labs to workd
   void Timer_Handler( void ){
    GPTMICR_Timer_0 |= 0x1;//resetsA
    //GPTMIMR_Timer_0 |= 0x101;//enable interupt TimerA AND TimerB


    switch(state){//////Look at the last bits
    case GO:
      ////////////////////////////////////////////////////////////////////ON
      state=STOP;
      GoLED();
      ////////////////////////////////////////////////////////////////////ON
        break;
     case STOP:
      ////////////////////////////////////////////////////////////////////stop
      state=GO;
      StopLED();
      ////////////////////////////////////////////////////////////////////STOP
        break;
     default:
```

```
        break;
    }


}









void lab2p5(){

  initlab2p5();
  state = OFF;

  while (1) {}
  }
```

Cstartup.m

```
/**************************************************
 *
 * This file contains an interrupt vector for Cortex-M written in C.
 * The actual interrupt functions must be provided by the application developer.
 *
 * Copyright 2007-2017 IAR Systems AB.
 *
 * $Revision: 112610 $
 *
 **************************************************/

#pragma language=extended
#pragma segment="CSTACK"

extern void __iar_program_start( void );

extern void PortA_Handler( void );///me I jordan Added this
extern void Timer_Handler( void );///me I jordan Added this
extern void PortF_Handler( void );///me I jordan Added this
extern void NMI_Handler( void );
extern void HardFault_Handler( void );
extern void MemManage_Handler( void );
extern void BusFault_Handler( void );
extern void UsageFault_Handler( void );
extern void SVC_Handler( void );
```

```
extern void DebugMon_Handler( void );
extern void PendSV_Handler( void );
extern void SysTick_Handler( void );

typedef void( *intfunc )( void );
typedef union { intfunc __fun; void * __ptr; } intvec_elem;

// The vector table is normally located at address 0.
// When debugging in RAM, it can be located in RAM, aligned to at least 2^6.
// If you need to define interrupt service routines,
// make a copy of this file and include it in your project.
// The name "__vector_table" has special meaning for C-SPY, which
// is where to find the SP start value.
// If vector table is not located at address 0, the user has to initialize
// the  NVIC vector table register (VTOR) before using interrupts.


#pragma location = ".intvec"
const intvec_elem __vector_table[] =
{
  { .__ptr = __sfe( "CSTACK" ) },
  __iar_program_start,

  NMI_Handler,
  HardFault_Handler,
  MemManage_Handler,
  BusFault_Handler,
  UsageFault_Handler,
  0,
  0,
  0,
  0,
  SVC_Handler,
  DebugMon_Handler,
  0,
  PendSV_Handler,
  SysTick_Handler,
  PortA_Handler,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
  0,
```

```
    0,
    0,
    0,
    0,
    0,
    0,
    Timer_Handler,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    PortF_Handler


};

#pragma call_graph_root = "interrupt"
__weak void NMI_Handler( void ) { while (1) {} }//you need more protien
#pragma call_graph_root = "interrupt"
__weak void HardFault_Handler( void ) { while (1) {} }
#pragma call_graph_root = "interrupt"
__weak void MemManage_Handler( void ) { while (1) {} }
#pragma call_graph_root = "interrupt"
__weak void BusFault_Handler( void ) { while (1) {} }
#pragma call_graph_root = "interrupt"
__weak void UsageFault_Handler( void ) { while (1) {} }
#pragma call_graph_root = "interrupt"
__weak void SVC_Handler( void ) { while (1) {} }
#pragma call_graph_root = "interrupt"
__weak void DebugMon_Handler( void ) { while (1) {} }
#pragma call_graph_root = "interrupt"
__weak void PendSV_Handler( void ) { while (1) {} }
#pragma call_graph_root = "interrupt"
__weak void SysTick_Handler( void ) { while (1) {} }//up until this point i did not add
it so it's lame
#pragma call_graph_root = "interrupt"
__weak void Timer_Handler (void ) { while(1) {} }//me me me I added this i am so cool
#pragma call_graph_root = "interrupt"
__weak void PortF_Handler (void ) { while(1) {} }//me me me I added this i am so cool
#pragma call_graph_root = "interrupt"
__weak void PortA_Handler (void ) { while(1) {} }////me me me I added this i am so cool


void __cmain( void );
```

```
__weak void __iar_init_core( void );
__weak void __iar_init_vfp( void );

#pragma required=__vector_table
void __iar_program_start( void )
{
  __iar_init_core();
  __iar_init_vfp();
  __cmain();
}
```

Header

```
void initializelab2p1();
//void lab2p1();
//void Timer_Handler( void );///me I jordan Added this


#define GPIO_PORTF_DATA_R (*((volatile uint32_t *)0x400253FC))//from the lab handout
#define RCGCGPIO (*((unsigned long *)0x400fE608))
#define PORT_F ((unsigned long)0x20)
#define PortF_DIR (*((unsigned long *)0x40025400))
#define PortF_DEN (*((unsigned long *)0x4002551C))
#define PortF_DATA (*((unsigned long *)0x40025000))
#define GPIO_LOCK (*((unsigned long *)0x40025520))
#define UNLOCKED ((unsigned long)0x4C4F434B))
#define GPIO_CMT (*((unsigned long *)0x40025524))
#define GPIO_PUR (*((unsigned long *)0x40025510))
#define GPIO_PORTA_AMSEL_R (*((volatile uint32_t *)0x40004528))/////FOR port A, I decided to stick to convention since I used the code copied from the lab... but then i unused it... so... well the convention is probably good.
#define SYSCTL_RCGC2_R (*((volatile uint32_t *)0x400FE108))///whats volitile unit 32? why not unsigned log... well the first thing i used was volitle so VOLITLE it is...
#define GPIO_PORTA_PCTL_R (*((volatile uint32_t *)0x4000452C))
#define GPIO_PORTA_DIR_R (*((volatile uint32_t *)0x40004400))
#define GPIO_PORTA_AFSEL_R (*((volatile uint32_t *)0x40004420))
#define GPIO_PORTA_DEN_R (*((volatile uint32_t *)0x4000451C))///that was easy, I already know the bias so.....
#define GPIO_PORTA_DATA_R (*((volatile uint32_t *)0x400043FC))
#define RCGCTIMER (*((volatile uint32_t *)0x400FE604))

#define GPTMCTL_Timer_0 (*((volatile uint32_t *)0x4003000C))//controlling pausing, ect
#define GPTMCFG_Timer_0 (*((volatile uint32_t *)0x40030000))//bittype set

#define GPTMCAMR_Timer_0 (*((volatile uint32_t *)0x40030004))//modeset
#define GPTMCBMR_Timer_0 (*((volatile uint32_t *)0x40030008))//modeset

#define GPTMTAILR_Timer_0 (*((volatile uint32_t *)0x40030028))//load time value
#define GPTMTBILR_Timer_0 (*((volatile uint32_t *)0x4003002C))//load time value

```

```

#define GPTMTAV_Timer_0 (*((volatile uint32_t *)0x40030050))// time value


#define GPTMRIS_Timer_0 (*((volatile uint32_t *)0x4003001C))// time value Poll
#define GPTMICR_Timer_0 (*((volatile uint32_t *)0x40030024))// time value CLEAR

#define GPTMIMR_Timer_0 (*((volatile uint32_t *)0x40030018))// time value CLEAR


#define Seconds 16000000// Seconds unit for CPU


#define En0 (*((volatile uint32_t *)0xE000E100))//

#define GPIOM_Port_F (*((volatile uint32_t *)0x40025410))//GPIOInterupt MASK,
#define GPIOM_Port_A (*((volatile uint32_t *)0x40004410))//GPIOInterupt MASK,


//
//GPIO Port F (APB) base: 0x4002.5000 offset 51C [Digital enable]
//GPIO Port F (APB) base: 0x4002.5000 offset 400 [Control direction - input/output]
//General-Purpose Input/Output Run Mode Clock Gating Control (RCGCGPIO) Base
0x400F.E000 Offset 0x608
```