Notebook 2
ELEC 3040/3050 Spring 2021
By Jacob Howard, Undergraduate Computer Engineering

Objective

The objective of this notebook was to record lab 3 to lab 5 progress and data. In Lab 3, we were introduced to interrupts, Lab 4 had us programing functionality for a physical keypad, and Lab 5 was Pulse-width modulation using a programmable timer. *(Note that all code and screenshots will be provided at the end of the notebook)*

Pre-Lab

Each lab required a pre-lab and the code design to be finished before coming to the lab.

- For Lab 3, we were required to review the module 3 class slides, have the hardware design wired up, and the software to be written.
- For lab 4, we were required to review module 4 class slides, have the hardware design wired up, sketch a rough circuit schematic design for the lab, and have the software design finished.
- For lab 5, we were required to create a flowchart describing our code, have a draft program, and a plan for testing the PWM generator.

In-Lab

- In Lab 3, our code was required to use 2 buttons instead of switches. Button 1 (microcontroller I/O port pin PA1) turned on and off the entire code. It was the start and stop button. Button 2 (microcontroller I/O port pin PA2) would switch the direction of the second counter from counting up from 0 to 9 to counting down from 9 to 0. We coded interrupts for this process. We were then required to wire up the board to the 8 LED displays. After everything was set up, we were required to run and debug our code. For this lab, I had very little lab time to debug my code as uVision kept giving me an error. The error persisted until 15 minutes of class was left, giving me very little time to fix any problems with my code. One problem I noticed was that the delay was not working in the lab. The other issue I had was that the code would not work with a button push, but a button hold. I had to switch the switches to further test the program. Other than these two issues, the code seemed to work.
- In Lab 4, we were required to mount the keypad header and the AND gate chip onto the Studio breadboard and connect them to the microcontroller. Then we were required to wire everything up correctly and test and debug our program. Sadly, for this lab, my program did not seem to work at all on the board. I tried debugging the program, but did not have any success. For the remainder of the lab, I reworked some bugges from Lab 1 and demoed the program to the TA. After lab, I worked on the Lab 4 program, and came up with a new solution that I believe will work.

- In Lab 5, we were required to verify that system hardware is operating properly. We did this by using test programs from previous labs. Once we verified the hardware was working properly, we were asked to download , edit, compile, and test the designed PWM waveform generation program. Then we were asked to demonstrate a working keypad-selectable PWM generator to the TA and show that all design specifications are met, including the proper state of the waveform at 0% duty cycle. At the end of the lab we had an optional task to modify the design to reduce the PWM switching frequency to 100 Hz and also try to increase to a 10 kHz switching frequency. Sadly, I did not get to debug this code as I was sick for this lab. I am hoping for extra lab time to demonstrate my working programs to the TA whenever there is time available.

Summary

In conclusion, I felt that lab 4 and lab 5 were more difficult than the previous labs. I had a lot of trouble designing the code for these labs. Hopefully, I have come up with a solution that works for both of these labs when I am able to test and verify the code. I also feel like I am struggling to understand some of the class material, which makes it difficult to design code for each lab. I believe i am starting to understand the material more, but as each lab increases in difficulty, it is becoming hard to keep up with each lab.

```
/*=====================================================*/
/* Jacob Howard */
/* Lab 3 Code */
/*=====================================================*/

#include "stm32l4xx.h" /* Microcontroller information */

/* Define global variables */
//int toggles; /* number of times LED state toggled */
static int counter1, counter2;
unsigned char run, up, blueLED, greenLED;

static uint16_t sw1; //declare 16-bit variable that matches IDR size
(Switch 1 to start and stop counters)
static uint16_t sw2; //Switch 2 to reverse counter 1



/*-----------------------------------------------------*/
/* Initialize GPIO pins used in the program */
/* PA11 = push button */
/* PB4 = LDR, PB5 = green LED */
/*-----------------------------------------------------*/
void PinSetup () {
/* Configure PA0 as input pin to read push button */
  RCC->AHB2ENR |= 0x01; /* Enable GPIOA clock (bit 0) */
  GPIOA->MODER &= ~(0x03FFFC3C); // General purpose input mode */ //THIS
may be wrong. Might be ~0x0000003C.
 GPIOA->MODER |= 0x01555400; // Sets pins 1 and 2 as inputs and pins
5-12 as outputs, and all others as 00.

  RCC->AHB2ENR |= 0x03; //enable GPIOB clock (2bit)
  GPIOB->MODER &= ~(0x03C0); //setting pins 3 and 4 PB
  GPIOB->MODER |= 0x0180; //setting pins 3 and 4 as output pins
}

/*-------------------------------------------------------------*/
/* Interrupt settup
```

```
/*-------------------------------------------------------------*/
void InterruptSetup(){
  SYSCFG->EXTICR[0] &= ~(0x000F); //Clear EXTI0 (set for PA0)
  SYSCFG->EXTICR[0] &= ~(0x00F0); //Clear EXTI1 (set for PA1)

  EXTI->RTSR1 |= 0x0001; //Set rising trigger for PA0
  EXTI->RTSR1 |= 0x0002; //Set rising trigger for PA1

  EXTI->IMR1 |= 0x0001; //Enable interrupt PA0
  EXTI->IMR1 |= 0x0002; //Enable interrupt PA1

  EXTI->PR1 |= 0x001; //Clear pending
  EXTI->PR1 |= 0x002; //Clear pending

  NVIC_EnableIRQ(EXTI1_IRQn); //Enable interrupt for PA0
  NVIC_EnableIRQ(EXTI2_IRQn); //Enable interrupt for PA1

  NVIC_ClearPendingIRQ(EXTI1_IRQn); //Clear NVIC pending for PA0
  NVIC_ClearPendingIRQ(EXTI2_IRQn); //Clear NVIC pending for PA1
}

/*-------------------------------------------------------------*/
/* Interrupt Service Routine                                   */
/*-------------------------------------------------------------*/
void EXTI1_IRQHandler() {
   if (run == 1) {
       run = 0;
   }
   else {
       run = 1;
   }
   GPIOB->ODR = (GPIOB->ODR & ~(0x08)) | (run << 3);//Set to value of
run
   NVIC_ClearPendingIRQ(EXTI1_IRQn);
   EXTI->PR1 |= 0x0002;
}

void EXTI2_IRQHandler(){
  if (up == 1) {
```

```
      up = 0;
  }
    else {
      up = 1;
    }
    GPIOB->ODR = (GPIOB->ODR & ~(0x04)) | (up << 2);//Set to value of
run
   NVIC_ClearPendingIRQ(EXTI2_IRQn);
   EXTI->PR1 |= 0x0001;
  /*
  up = 1;
  if(greenLED == 0){
         GPIOB->BSRR = 0x0008;            //Toggle PB4
         greenLED = 1;
     }else{
         GPIOB->BSRR = 0x0008 << 16;      //Toggle PB4
         greenLED = 0;
     }
  EXTI->PR1 |= 0x002;           //Clear pending
  NVIC_ClearPendingIRQ(EXTI2_IRQn);   //Clear pending
  */
}


/*-------------------------------------------------------------*/
/* Function to Count up to 9 and then back to zero
/*-------------------------------------------------------------*/

void countUpandDown () {
 if (up == 1) {
    if (counter1 < 9) {
          counter1++;
      }
    else {
        counter1 = 0;
      }
    }
 else {
    if (counter1 > 0) {
```

```
        counter1--;
      }
    else {
      counter1 = 9;
      }
 }

}

/*-------------------------------------------------------------*/
/* Function to Count up to 9 indefinately
/*-------------------------------------------------------------*/

  void count () {
    if (counter2<9) {
      counter2++;

    }
   else {
   counter2 = 0;
   }
  }
/*-------------------------------------------------------------*/
/* Delay function - 0.5 second delay */
/*-------------------------------------------------------------*/
void delay (){
int a,b;
for (a=0; a<100; a++) { //outer loop
for (b=0; b<12000; b++) { //inner loop
}
}
}

/*-------------------------------------------------*/
/* Main program */
/*-------------------------------------------------*/
int main(void) {
  counter1 = 0; //initializing counter values
  counter2 = 0;
```

```
  run = 0; //setting initial run to 0
  up = 1; //setting direction as counting up


  //setting blue and green leds as off
  blueLED = 0;
 greenLED = 0;

  PinSetup(); //Configure GPIO pins
  InterruptSetup(); //Configure Interrupts
  __enable_irq(); //Enable CPU interrupts

  while (1) {

  run = (GPIOA->IDR & PWR_PUCRA_PA1) >> 1; //sets switch 1 to PA1 and
shifts right 1 bit
 up = (GPIOA->IDR & PWR_PUCRA_PA2) >> 2; //sets switch 2 to PA2 and
shifts right 2 bits

  if (run == 1) {
  countUpandDown();
 count();

  delay(); //delay

 // Writes counter1 and counter2 to the output pins
  GPIOA->ODR = (GPIOA->ODR & ~(0x1FE0)) | (((counter2 << 4) + counter1)
<< 5);
 }
}

}
```

Lab 3 Code

```
/*=====================================================*/
/* Jacob Howard */
/* ELEC 3040/3050 - Lab 4 */
/* Cuount up/down depending on specified direction */
/*=====================================================*/


#include "STM32l4xx.h"                        //Microcontroller
information

/* Define global variables */
int counter;            //number currently displayed
int something;
int key;
int reset;

struct {
    unsigned char row;
    unsigned char column;
    unsigned char event;
    const int row1[4];
    const int row2[4];
    const int row3[4];
    const int row4[4];
    const int* keys[];
} typedef matrix_keypad;

void PinSetup (void);
void delay (void);
void smallDelay (void);
void count (void);
void updateLEDs (int value);
int readColumn(void);
int readRow(void);

matrix_keypad keypad = {
```

```
    .row = ~0,
    .column = ~0,
    .event = 0,
    .row1 = {1, 2, 3, 10},
    .row2 = {4, 5, 6, 11},
    .row3 = {7, 8, 9, 12},
    .row4 = {14, 0, 15, 13},
    .keys = {keypad.row1, keypad.row2, keypad.row3, keypad.row4},
};

/*-------------------------------------------------*/
/* Main program */
/*-------------------------------------------------*/

int main(void) {
    PinSetup();                                 //Configure GPIO pins
    counter = 0;
    key = 0;
    reset = 0;
    updateLEDs(counter);

    /* Endless loop */
    while (1) {
        delay();
        count();
        if (keypad.event != 0) {
            updateLEDs(key);
        }
        else {
            updateLEDs(counter);
        }
}  /* repeat forever */
}

/*-------------------------------------------------*/
/* Initialize GPIO pins used in the program */
/* PA1 = start/stop PA2 = direction */
/* PC[0,3] = output */
/*-------------------------------------------------*/
```

```
void PinSetup () {
    /* Configure PA1 and PA2 as input pin to read push button */
    RCC->AHB2ENR |= 0x01;                       //Enable GPIOA clock
(bit 0)
    //GPIOA->MODER &= ~(0xFF0);     //clears bits 2-5
    //GPIOA->MODER &= ~(0xFF0000); //clears bits 8-11

    //GPIOA->MODER |= (0x550000); //setting 8-11 as output

    /*GPIOB*/
    RCC->AHB2ENR |= 0x02;                       //Enable clock
    GPIOB->MODER &= ~(0x3); //clear

    GPIOB->MODER &= ~(0x3FC0);
    GPIOB->MODER |= (0x1540);

    //GPIOA->PUPDR &= ~(0xFF0); //clear pu/pd
    //GPIOA->PUPDR |= (0x550);      //pull up/ pull down

    RCC->APB2ENR |= 0x01; //
    SYSCFG->EXTICR[0] &= 0xFFF0; //ext0
    SYSCFG->EXTICR[0] |= 0x1;
    EXTI->FTSR1 |= 0x0001;
    EXTI->IMR1 |= 0x0001;
    EXTI->PR1 |= 0x1;
    NVIC_EnableIRQ (EXTI1_IRQn);
    NVIC_ClearPendingIRQ (EXTI1_IRQn);

    __enable_irq();
}

/*-------------------------------------------------------------*/
/* interrupt handler*/
/*-------------------------------------------------------------*/
void EXTI1_IRQHandler () {
    EXTI->PR1 |= 0x0002;

    int row = readRow();
```

```
    int column = readColumn();
    keypad.event = 5;
    if (row != -1 & column != -1) {
        key = keypad.keys[row][column];
    }
    else {
        key = -1;
    }
    if (key != -1) {
        reset = 5;
    }
    else {
        reset = 0;
    }
}
/*-------------------------------------------------------------*/
/* Delay function - do nothing for about 1 second */
/*-------------------------------------------------------------*/

void delay () {
int i,j,n;
    i = j = n = 0;
    if (keypad.event > 0) {
        keypad.event--;
    }
for (i=0; i<20; i++) {          //outer loop
    for (j=0; j<20000; j++) { //inner loop
        n = j;                                          //dummy operation for
single-step test
    }                                                       //do nothing
}
}
/*-------------------------------------------------*/
/* Count function - increment or decrement according to direction*/
/*-------------------------------------------------*/

void count () {
    counter = (counter + 1) % 10;
}
```

```
/*-------------------------------------------------*/
/* Count function - increment or decrement according to direction*/
/*-------------------------------------------------*/

void updateLEDs (int value) {
    //display count information
    GPIOB->ODR &= ~(0x78); //sets 3-6 = 0
    GPIOB->ODR |= value << 3;
}

int readColumn() {
    //GPIOB->MODER &= ~(0x0000FFFF);
    //GPIOB->MODER |= (0x00000055);
    //GPIOB->PUPDR &= ~(0x0000FF00);
    //GPIOB->PUPDR |= (0x00005500);
    //rows are ouput
    //columns are input
    GPIOA->MODER &= ~(0xFF0);   //clears bits 2-5
    GPIOA->MODER &= ~(0xFF0000); //clears bits 8-11
    GPIOA->MODER |= (0x550); //setting 5-2 as output
    GPIOA->PUPDR &= ~(0xFF0000);    //clear pu/pd
    GPIOA->PUPDR |= (0x550000);     //pull up/ pull down
    GPIOA->ODR = 0;


    something = 4;

    while (something > 0) {
        something --;
    }
    int input = GPIOA ->IDR & ~(0xF00);
    input = input >> 8;
    switch(input) {
        case 0xE:
            return 1;
        case 0xD:
            return 2;
        case 0xB:
            return 3;
```

```
        case 0x7:
            return 4;
        default:
            return -1;
    }
}
int readRow() {
    /**
    GPIOB->MODER &= ~(0x0000FFFF);
    /GPIOB->MODER |= (0x00005500);
    GPIOB->ODR = 0;
    GPIOB->PUPDR &= ~(0x000000FF);
    GPIOB->PUPDR |= (0x00000055);
    **/
    //rows are input
    //columns are output
    GPIOA->MODER &= ~(0xFF0);   //clears bits 2-5
    GPIOA->MODER &= ~(0xFF0000); //clears bits 8-11
    GPIOA->MODER |= (0x550000); //setting 8-11 as output
    GPIOA->PUPDR &= ~(0xFF0);   //clear pu/pd
    GPIOA->PUPDR |= (0x550);        //pull up/ pull down
    GPIOA->ODR = 0;
    something = 4;

    while (something >4){
        something--;
    }
    int input = GPIOA->IDR & ~(0x3C);
    input = input >> 2;
    switch(input) {
        case 0xE:
            return 1;
        case 0xD:
            return 2;
        case 0xB:
            return 3;
        case 0x7:
            return 4;
        default:
```

```
            return -1;
    }
}
```

Lab 4 Code

```
/*==================================================*/
/* Lab 5 Code*/
/* Jacob Howard */
/*==================================================*/

#include "stm32l4xx.h" /* microcontroller information */

/* Define global variables */
static unsigned int counter;    //value of count (0-9)
static unsigned int button;             //value of button press
unsigned int col;               //what column has been pressed
unsigned int row;               //what row has been pressed
static unsigned int colNum;             //what column # has been pressed
static unsigned int rowNum;             //what row # has been pressed
unsigned int ccrNum;            //value for duty cycle
static unsigned int go;                 //handler variable
unsigned int i,j,n,k;           //delay variables
static unsigned int keypad_map [4][4] = {       //keypad matrix, no press = 0xFF
        {0x01,0x02,0x03,0x0A},          //0,0;1st row
```

```
        {0x04,0x05,0x06,0x0B},          //1,0;2nd row
        {0x07,0x08,0x09,0x0C},          //2,0;3rd row
        {0x0E,0x00,0x0F,0x0D}           //3,0;4th row
};//0,0; 0,1; 0,2; 0,3;
static unsigned int ccr_value [11] = {          //CCRy values according to button press
        0, 399, 799, 1199, 1599, 1999,
        2399, 2799, 3199, 3599, 3999
};

/*-------------------------------------------------*/

static void Setup() {

        /* enable clocks */
        RCC->AHB2ENR |= 0x03;       //enable GPIOA clock bit 0 and GPIOB clock bit 1

        /* configure GPIO pins */
        GPIOA->MODER &= (0xFFFFFFF0);                   //clearing |  PA0 = 00 and PA1 = 00
        GPIOA->MODER |=     (0x00000006);               // mapping | setting PA0 = 10 and
PA1 = 01

  GPIOB->MODER &= (0xFFFFC03C);                 //inputs  display and AND, PB[6:3,0] = 00
        GPIOB->MODER |= (0x00001540);           //outputs// display, PB[6:3] = 01

}
```

```
/*-----------------------------------------------*/

static void PinSetup1() {

        Setup();
        /* configure GPIOA pins */
        GPIOA->MODER &= (0xFF00F00F);     //inputs// column and row, PA[11:8,5:2] = 00
        GPIOA->MODER |= (0x00550000);     //outputs// column, PA[11:8] = 01

        /* configure push-pull pins */
        GPIOA->PUPDR &= (0xFFFFF00F);     //pull-reset// row, PA[5:2] = 00
        GPIOA->PUPDR |= (0x00000550);     //pull-up// row, PA[5:2] = 01


}

/*-----------------------------------------------*/
static void PinSetup2() {

        Setup();
        /* configure GPIOA pins */
        GPIOA->MODER &= (0xFF00F00F);     //inputs// column and row, PA[11:8,5:2] = 00
        GPIOA->MODER |= (0x0550);         //outputs// row, PA[5:2] = 01
```

```
	/* configure push-pull pins */
	GPIOA->PUPDR &= (0xFF00FFFF);      //pull-reset// row, PA[11:8] = 00
	GPIOA->PUPDR |= (0x00550000);      //pull-up// row, PA[11:8] = 01

}

/*-----------------------------------------------*/
/* enable PWM used in the program*/
/*-----------------------------------------------*/
static void PulseSetup() {

	RCC->AHB2ENR |= 0x01;        // Enable GPIOA clock bit 0

	GPIOA->MODER &= 0xFFFFFFFC; // PA0 = 00, clear
	GPIOA->MODER |= 0x0002;     // PA0 = 01, alternative function mode
	//select desired AF
	GPIOA->AFR[0] &= (0xFFFFFFF0);              //mask bit[3:0]=00
	GPIOA->AFR[0] |= (0x00000001);              //(0x0002);//configure bit[3:0]=0010, AF1
selected
	//timer
	RCC->APB1ENR1 |= RCC_APB1ENR1_TIM2EN;       //(0x01);//enable timer module
	TIM2->CR1 |= 0x01;                          //enable timer counter
	TIM2->CCMR1 &= (0xFFFFFF8C);       //mask channel one (bit[6:4]=00), output mode
	TIM2->CCMR1 |= (0x00000060);                //configure output mode for PWM mode 1
	TIM2->CCER &= (0xFFFC);            //bit[1:0]=00, clear timer channel 1 output
	TIM2->CCER |= (0x0001);            //bit[1:0]=01, enable timer channel 1 output (active
```

```
high)
        //pulse
        TIM2->PSC = 0;
        TIM2->ARR = 4000;
        TIM2->CCR1 = 0;
        NVIC_EnableIRQ(EXTI0_IRQn); // IRQ
}

/*----------------------------------------------*/

static void InterruptSetup() {

        RCC->APB2ENR |= 0x01;    // interrupt clock SYSCFG

        SYSCFG->EXTICR[0] &= 0xFFF0;         //clear EXTI1 bit in config reg ~(0xF)
  SYSCFG->EXTICR[0] |= 0x0001;         //PB configuration in EXTI0

        EXTI->FTSR1 |= 0x0001;                                  //falling edge trigger
enabled
        EXTI->IMR1 |= 0x0001;                                   //enable  EXTI0
        EXTI->PR1 |= 0x0001;                                    //clear EXTI0 pending bit
        NVIC_ClearPendingIRQ(EXTI0_IRQn);                       //Clear NVIC pending bit
        NVIC_EnableIRQ(EXTI0_IRQn);                            //enable IRQ with EXTI line 0 interrupt

}

/*-----------------------------------------------------*/
```

```
static void debounce() {                    //

        for (i=0; i<15; i++) {

        for (j=0; j<40; j++) {

        n = j;

        }

        }

}



/*-------------------------------------------------------*/


static void delay() {                       //

        for (i=0; i<15; i++) {
        for (j=0; j<10000; j++) {

        n = j;

        }

        }

}



/*-------------------------------------------------------*/


static void keypad() {

        button = 0;

        rowNum = 0;

        colNum = 0;

        ccrNum = 0;

        col = 0;
```

```
row = 0;


GPIOB->ODR &= 0xFF87;                   //mask PB[6:3]


row=0;

GPIOA->ODR &= (0xF0FF);

for(k=0; k<4; k++);                     //delay
row = (~GPIOA->IDR & 0x003C);           //get row inputs
row = row >> 2;

do {

row = row << 1;

rowNum++;                               //add to row count

} while(row < 0x10) ;                   //only shift 4 times


PinSetup2();

debounce();

col=0;

GPIOA->ODR &= (0xFFC3);

for(k=0; k<4; k++);                     //delay
col = (~GPIOA->IDR & 0xF00); //get column inputs

col = col >> 8;

do {

col = col << 1;

colNum++;                               //add to column count

} while(col < 0x10) ;                   //only shift 4 times


button = keypad_map[--rowNum][--colNum];
```

```
        ccrNum = ccr_value[button];
        TIM2->CCR1 = ccrNum;

        button = button << 3;
        GPIOB->ODR &= 0xFF87;                   //mask
        GPIOB->ODR |= button;



        ccrNum = 0;
        button = 0;
        delay ();
        delay ();
}

/*-------------------------------------------------------*/
void EXTI0_IRQHandler() {
        debounce();
        go=~go;
        keypad();
        PinSetup1();
        debounce();

        EXTI->PR1 |= 0x0001;            //clear EXTI0
        NVIC_ClearPendingIRQ(EXTI0_IRQn);           //clear NVIC pending bit
```

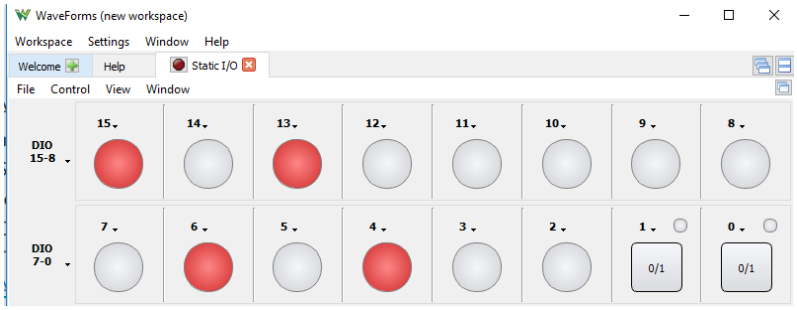
```
        __enable_irq();                         //enable interupts


/*-------------------------------------------*/
/* main program */
/*-------------------------------------------*/
int main(void) {


        Setup();
        PinSetup1();
        InterruptSetup();
        PulseSetup();
        go = 1;

        // Endless loop
        while(1){
        delay();
        if(go != 0x01){                 //check button press
        go=~go;
        }
        }
}
```

Lab 5 code


WaveForms (new workspace)
Workspace
Settings
Window
Help
Welcome
Help
Static I/O
File
Control
View
Window
DIO 15-8
15
14
13
12
11
10
9
8
DIO 7-0
7
6
5
4
3
2
1
0
0/1
0/1

Lab 3 LEDs