To: John Hung, Victor Nelson

From: Jake Neal and Cameron Shea

Section: Wednesday, 3:00 p.m.

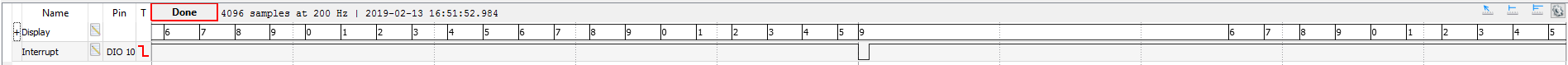
Date: 22 February 2019

Subject: Memo 3

Labs 5 and 6: Keyboard and Stopwatch

The past two weeks were focused mainly on interrupts. Last week, we focused on the concept of how to setup and use an interrupt in order for a keypad to interface with our program. This week we focused on another interrupt meant specifically for the timer.

Last week, we were to write a code that initially increments from 0 to 9 and then rolls over to 0. This cycle runs indefinitely. However, when a key is pressed, the program is to interrupt the incrementing counter to display the value of the pressed key. With the hardware connecions checked ran and the syntax debugged, we were ready to test our program. The operation of our code seemed fine at first, but the interrupt failed to occur at all. After help from Graham, we realized that our pin setup contained the error: we failed to activate clock B and we adjusted PUPDR before we changed the MODER. With the clock started and the registers ordered properly, we could achieve interrupts by pressing keys. We then ran into the next problem, but we were stumped about the cause. When “A” was pressed, the screen would display “3.” This was the only numbering error for the display. After 15 minutes of various troubleshooting ideas, we finally found the typo. In the statement defining which key was “3,” we assigned the improper row (we assigned the same row as A). Once this typo was fixed, our program was checked and approved by the TA. Below is a screenshot of the logic analyzer as proof of proper function of the program. I apologize for the picture taking two lines, but it the text was too small to keep as one continuous photo.



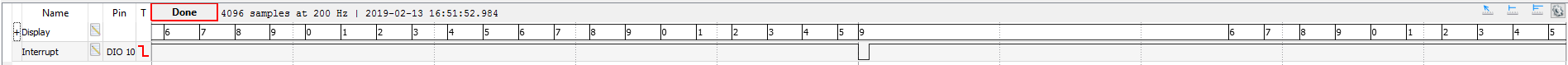


Figure 1. The logic analyzer for our code in Lab 5

The code for Lab 5 can be seen at the end of this memo. It can also be found at:

<https://github.com/WhatIsYourQuest/3040-Lab>

For Lab 6, our task was to create a digital stopwatch with a seconds counter and a tenth of a second counter. In contrast to the previous lab where we used a delay between incrementing, this lab required us to use a built is timer to interrupt the main program and increment the stopwatch. The keypad and its code from last week were kept as a controller for the stopwatch.

After compiling the code, we ran the program. Immediately we found that the timer interrupt was not working. As with last week, our first error was in the pin setup. We placed the timer initializations out of order. With this fixed, our code ran well. The only other issue was that the buttons were inconsistent. For example, pressing “0” only stopped the code approximately 70% of the time. At first we considered bouncing to be the culprit. After the usual troubleshooting, bouncing was not the issue. On a whim, Cameron unplugged the power wire and plugged it back in. With this simple toggling of the power, our program ran perfectly. Graham checked our code and scopes and confirmed proper function. Below, the oscilloscope and logic analyzer outputs can be seen. The code for Lab 6 can be seen at the end of this memo. It can also be found at:

<https://github.com/WhatIsYourQuest/3040-Lab>

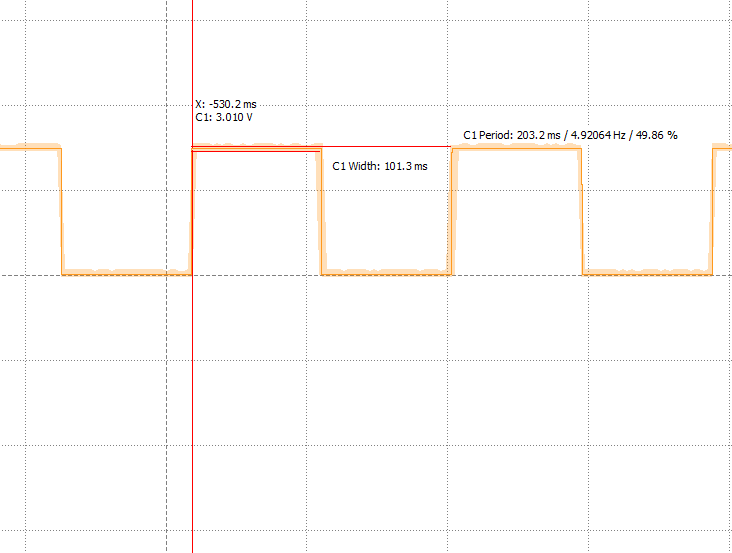


Figure 2. The oscilloscope reading of the LSB to prove proper time duration.

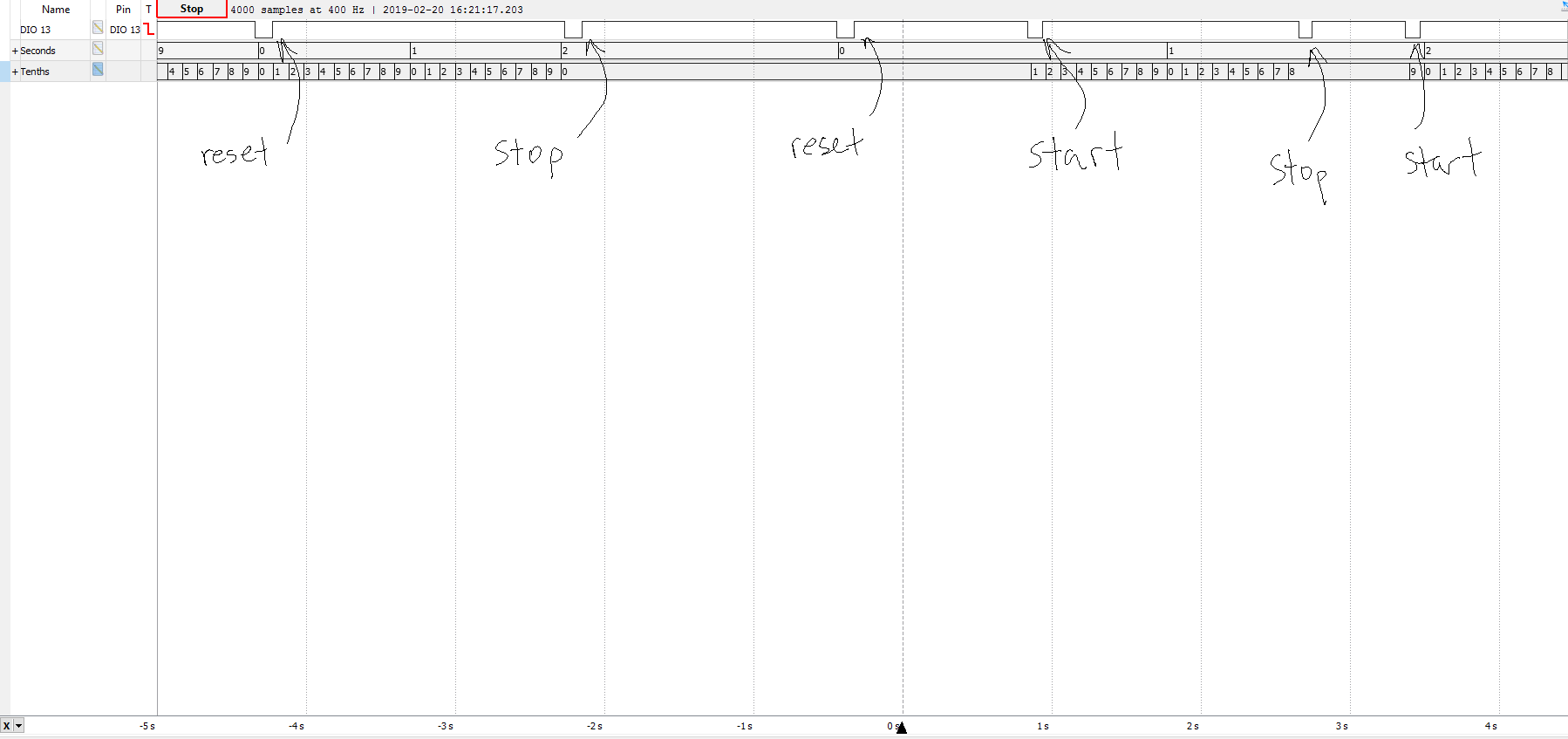


Figure . Logic analyzer showing every function of the program

/\*====================================================\*/

/\* Jake Neal and Cameron Shea \*/

/\* ELEC 3040/3050 - Lab 5 \*/

/\*====================================================\*/

#include "STM32L1xx.h" /\* Microcontroller information \*/

/\* Define global variables \*/

int state=0; //current state of the LEDs in counter1

int key=0; //key that was pressed

int key\_var=10; //counter for deciding to display key or counter (set to 10 so default is display counter)

unsigned char led1=0; //state of LED1

unsigned char led2=0; //state of LED2

unsigned char led3=0; //state of LED3

unsigned char led4=0; //state of LED4

unsigned char led5=0; //state of LED1 (for displaying key that was pressed)

unsigned char led6=0; //state of LED2 (for displaying key that was pressed)

unsigned char led7=0; //state of LED3 (for displaying key that was pressed)

unsigned char led8=0; //state of LED4 (for displaying key that was pressed)

/\*---------------------------------------------------\*/

/\* Initialize GPIO pins used in the program \*/

// PA1 input IRQ

// PB0-PB3 input/output keypad rows

// PB4-PB7 input/output keypad columns

// PC0-PC3 output counter LEDs

/\*---------------------------------------------------\*/

void PinSetup () {

/\* Configure PA1 as input for IRQ \*/

RCC->AHBENR |= 0x01; // Enable GPIOA clock (bit 0)

GPIOA->MODER &= ~(0x00000000); // General purpose input mode

/\* Configure PC0-PC3 as output pins to drive LEDs \*/

RCC->AHBENR |= 0x04; // Enable GPIOC clock (bit 2)

GPIOC->MODER &= ~(0x000000FF); // Clear PC0-PC3 mode bits

GPIOC->MODER |= (0x00000055); // General purpose output mode for PC0-PC3\*/

RCC->AHBENR |= 0x02; // Enable GPIOB clock (bit 0)

GPIOB->MODER &= ~(0x0000FF00); // PB4-PB7 output keypad rows

GPIOB->MODER |= (0x00005500); // ^^^^

GPIOB->PUPDR &= ~0x0000FFFF; //clear bits 0-15 for PB0-PB7 \*HERE I AM MAKING SURE THE AND GATE READS LOW\*

GPIOB->PUPDR |= 0x00000055; //set bits 0-7 to 01 for PB0-PB3 pull-up resistors, \*CHECK THIS STEP\*

GPIOB->BSRR = 0x0010 << 16; // send 0 to pin 4

GPIOB->BSRR = 0x0020 << 16; // send 0 to pin 5

GPIOB->BSRR = 0x0040 << 16; // send 0 to pin 6

GPIOB->BSRR = 0x0080 << 16; // send 0 to pin 7

//EXTI SECTION

SYSCFG->EXTICR[0] &= 0xFF0F; //clears EXTI1 bit

SYSCFG->EXTICR[0] |= 0x0000; //set EXTI1 = 0 to select PA1

EXTI->FTSR |= 0x0002; //Bit0=1 to make EXTI1 falling-edge trig.

EXTI->IMR |= 0x0002; //Bit0=1 to enable EXTI1

EXTI->PR |= 0x0002; //Bit0=1 to clear EXTI1 pending status

//NVIC SECTION

NVIC\_EnableIRQ(7); //set bit n to enable IRQ7

NVIC\_ClearPendingIRQ (7); // clears pending status

//CPU SECTION

\_\_enable\_irq(); //enable interrupts

}

/\*----------------------------------------------------------\*/

/\* EXTI1 Interrupt Function (signals the pressing of a keyboard button

/\*----------------------------------------------------------\*/

void EXTI1\_IRQHandler ()

{

int pb0=1; //reading from PB0

int pb1=1; //reading from PB1

int pb2=1; //reading from PB2

int pb3=1; //reading from PB3

int pb4=1; //reading from PB4

int pb5=1; //reading from PB5

int pb6=1; //reading from PB6

int pb7=1; //reading from PB7

int i,j,n;

// wait 1 ms

for (i=0; i<40; i++) //outer loop

{

for (j=0; j<18; j++)

{ //inner loop

n = j; //dummy operation for single-step test

} //do nothing

}

key\_var=0;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//reading columns \*CHECK THIS STEP\*

GPIOB->MODER &= ~(0x000000FF); // PB0-PB3 input keypad columns

GPIOB->MODER |= (0x00000000); // ^^^^

GPIOB->MODER &= ~(0x0000FF00); // PB4-PB7 output keypad rows

GPIOB->MODER |= (0x00005500); // ^^^^

//Pull Up Pull Down Section \*CHECK THIS STEP\*

GPIOB->PUPDR &= ~0x0000FFFF; //clear bits 0-15 for PB0-PB7

GPIOB->PUPDR |= 0x00000055; //set bits 0-7 to 01 for PB0-PB3 pull-up resistors, \*CHECK THIS STEP\*

GPIOB->BSRR = 0x0010 << 16; // send 0 to pin 4

GPIOB->BSRR = 0x0020 << 16; // send 0 to pin 5

GPIOB->BSRR = 0x0040 << 16; // send 0 to pin 6

GPIOB->BSRR = 0x0080 << 16; // send 0 to pin 7

for (int x=0;x<4;x++); // just making sure it has time

pb0 = GPIOB->IDR & 0x0001; //reading PB0

pb1 = GPIOB->IDR & 0x0002; //reading PB1

pb2 = GPIOB->IDR & 0x0004; //reading PB2

pb3 = GPIOB->IDR & 0x0008; //reading PB3

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//reading rows

GPIOB->MODER &= ~(0x0000FF00); // PB4-PB7 input keypad rows

GPIOB->MODER |= (0x00000000); // ^^^^

GPIOB->MODER &= ~(0x000000FF); // PB0-PB3 output keypad columns

GPIOB->MODER |= (0x00000055); // ^^^^

GPIOB->PUPDR &= ~0x0000FFFF; //clear bits 0-15 for PB0-PB7

GPIOB->PUPDR |= 0x00005500; //set bits 08-15 to 01 for PB4-PB7 pull-up resistors, \*CHECK THIS STEP\*

GPIOB->BSRR = 0x0001 << 16; // send 0 to pin 0

GPIOB->BSRR = 0x0002 << 16; // send 0 to pin 1

GPIOB->BSRR = 0x0004 << 16; // send 0 to pin 2

GPIOB->BSRR = 0x0008 << 16; // send 0 to pin 3

for (int x=0;x<4;x++); // just making sure it has time

pb4 = GPIOB->IDR & 0x0010; //reading PB4

pb5 = GPIOB->IDR & 0x0020; //reading PB5

pb6 = GPIOB->IDR & 0x0040; //reading PB6

pb7 = GPIOB->IDR & 0x0080; //reading PB7

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//decide key

if((pb4==0) && (pb0==0)) // button 1

{

key=1;

}

else if((pb4==0) && (pb1==0)) // button 2

{

key=2;

}

else if((pb4==0) && (pb2==0)) // button 3

{

key=3;

}

else if((pb5==0) && (pb0==0)) // button 4

{

key=4;

}

else if((pb5==0) && (pb1==0)) // button 5

{

key=5;

}

else if((pb5==0) && (pb2==0)) // button 6

{

key=6;

}

else if((pb6==0) && (pb0==0)) // button 7

{

key=7;

}

else if((pb6==0) && (pb1==0)) // button 8

{

key=8;

}

else if((pb6==0) && (pb2==0)) // button 9

{

key=9;

}

else if((pb4==0) && (pb3==0)) // button A

{

key=10;

}

else if((pb5==0) && (pb3==0)) // button B

{

key=11;

}

else if((pb6==0) && (pb3==0)) // button C

{

key=12;

}

else if((pb7==0) && (pb3==0)) // button D

{

key=13;

}

else if((pb7==0) && (pb0==0)) // button \* (treat like E)

{

key=14;

}

else if((pb7==0) && (pb2==0)) // button # (treat like F)

{

key=15;

}

else if((pb7==0) && (pb1==0)) // button 0

{

key=0;

}

else{}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//display key on LEDs

switch(key)

{

case 0:

led5=0;

led6=0;

led7=0;

led8=0;

break;

case 1:

led5=1;

led6=0;

led7=0;

led8=0;

break;

case 2:

led5=0;

led6=1;

led7=0;

led8=0;

break;

case 3:

led5=1;

led6=1;

led7=0;

led8=0;

break;

case 4:

led5=0;

led6=0;

led7=1;

led8=0;

break;

case 5:

led5=1;

led6=0;

led7=1;

led8=0;

break;

case 6:

led5=0;

led6=1;

led7=1;

led8=0;

break;

case 7:

led5=1;

led6=1;

led7=1;

led8=0;

break;

case 8:

led5=0;

led6=0;

led7=0;

led8=1;

break;

case 9:

led5=1;

led6=0;

led7=0;

led8=1;

break;

case 10: //A

led5=0;

led6=1;

led7=0;

led8=1;

break;

case 11: //B

led5=1;

led6=1;

led7=0;

led8=1;

break;

case 12: //C

led5=0;

led6=0;

led7=1;

led8=1;

break;

case 13: //D

led5=1;

led6=0;

led7=1;

led8=1;

break;

case 14: //\*

led5=0;

led6=1;

led7=1;

led8=1;

break;

case 15: //#

led5=1;

led6=1;

led7=1;

led8=1;

break;

}

if (led5 == 0)

GPIOC->BSRR = 0x0001 << 16;

else

GPIOC->BSRR = 0x0001;

if (led6 == 0)

GPIOC->BSRR = 0x0002 << 16;

else

GPIOC->BSRR = 0x0002;

if (led7 == 0)

GPIOC->BSRR = 0x0004 << 16;

else

GPIOC->BSRR = 0x0004;

if (led8 == 0)

GPIOC->BSRR = 0x0008 << 16;

else

GPIOC->BSRR = 0x0008;

// wait 1 ms

for (i=0; i<40; i++) //outer loop

{

for (j=0; j<18; j++)

{ //inner loop

n = j; //dummy operation for single-step test

} //do nothing

}

GPIOB->MODER &= ~(0x000000FF); // PB0-PB3 input keypad columns

GPIOB->MODER |= (0x00000000); // ^^^^

GPIOB->MODER &= ~(0x0000FF00); // PB4-PB7 output keypad rows

GPIOB->MODER |= (0x00005500); // ^^^^

GPIOB->PUPDR &= ~0x0000FFFF; //clear bits 0-15 for PB0-PB7 \*HERE I AM MAKING SURE THE AND GATE READS LOW\*

GPIOB->PUPDR |= 0x00000055; //set bits 0-7 to 01 for PB0-PB3 pull-up resistors, \*CHECK THIS STEP\*

GPIOB->BSRR = 0x0010 << 16; // send 0 to pin 4

GPIOB->BSRR = 0x0020 << 16; // send 0 to pin 5

GPIOB->BSRR = 0x0040 << 16; // send 0 to pin 6

GPIOB->BSRR = 0x0080 << 16; // send 0 to pin 7 resistors, \*CHECK THIS STEP\*

NVIC\_ClearPendingIRQ (7); // clears pending status

EXTI->PR |= 0x0002; //Bit0=1 to clear EXTI1 pending status

}

/\*----------------------------------------------------------\*/

/\* Delay function - do nothing for about .5 seconds \*/

/\*----------------------------------------------------------\*/

void delay ()

{

int i,j,n;

for (i=0; i<20; i++) { //outer loop

for (j=0; j<8772; j++) { //inner loop

n = j; //dummy operation for single-step test

} //do nothing

}

}

/\*---------------------------------------------------------------\*/

/\* Count - counts up or down based on value passed into function \*/

/\*---------------------------------------------------------------\*/

void count (a)

{

state=led1\*1+led2\*2+led3\*4+led4\*8;

if(a==0) //incrementing

{

if(state==9)

{

state=0;

}

else

{

state++;

}

}

switch(state)

{

case 0:

led1=0;

led2=0;

led3=0;

led4=0;

break;

case 1:

led1=1;

led2=0;

led3=0;

led4=0;

break;

case 2:

led1=0;

led2=1;

led3=0;

led4=0;

break;

case 3:

led1=1;

led2=1;

led3=0;

led4=0;

break;

case 4:

led1=0;

led2=0;

led3=1;

led4=0;

break;

case 5:

led1=1;

led2=0;

led3=1;

led4=0;

break;

case 6:

led1=0;

led2=1;

led3=1;

led4=0;

break;

case 7:

led1=1;

led2=1;

led3=1;

led4=0;

break;

case 8:

led1=0;

led2=0;

led3=0;

led4=1;

break;

case 9:

led1=1;

led2=0;

led3=0;

led4=1;

break;

}

if(key\_var<10)

{

key\_var++;

}

else

{

if (led1 == 0)

GPIOC->BSRR = 0x0001 << 16;

else

GPIOC->BSRR = 0x0001;

if (led2 == 0)

GPIOC->BSRR = 0x0002 << 16;

else

GPIOC->BSRR = 0x0002;

if (led3 == 0)

GPIOC->BSRR = 0x0004 << 16;

else

GPIOC->BSRR = 0x0004;

if (led4 == 0)

GPIOC->BSRR = 0x0008 << 16;

else

GPIOC->BSRR = 0x0008;

if (led5 == 0)

GPIOC->BSRR = 0x0010 << 16;

else

GPIOC->BSRR = 0x0010;

if (led6 == 0)

GPIOC->BSRR = 0x0020 << 16;

else

GPIOC->BSRR = 0x0020;

if (led7 == 0)

GPIOC->BSRR = 0x0040 << 16;

else

GPIOC->BSRR = 0x0040;

if (led8 == 0)

GPIOC->BSRR = 0x0080 << 16;

else

GPIOC->BSRR = 0x0080;

}

}

/\*------------------------------------------------\*/

/\* Main program \*/

/\*------------------------------------------------\*/

int main(void)

{

PinSetup(); //Configure GPIO pins

//the infinite loop will begin by counting up from zero once SW1

while(1)

{

count(0);

delay();

}

}

/\*====================================================\*/

/\* Jake Neal and Cameron Shea \*/

/\* ELEC 3040/3050 - Lab 6 \*/

/\*====================================================\*/

#include "STM32L1xx.h" /\* Microcontroller information \*/

/\* Define global variables \*/

int state=0; //current state of the LEDs in tenths

int state2=0; //current state of the LEDs in seconds

int key=0; //key that was pressed

int startstop=0; //0 is not running, and 1 is running (button 0)

unsigned char led1=0; //state of LED1

unsigned char led2=0; //state of LED2

unsigned char led3=0; //state of LED3

unsigned char led4=0; //state of LED4

unsigned char led5=0; //state of LED5

unsigned char led6=0; //state of LED6

unsigned char led7=0; //state of LED7

unsigned char led8=0; //state of LED8

/\*---------------------------------------------------\*/

/\* Initialize GPIO pins used in the program \*/

// PA1 input IRQ

// PB0-PB3 input/output keypad rows

// PB4-PB7 input/output keypad columns

// PC0-PC3 output counter LEDs

/\*---------------------------------------------------\*/

void PinSetup () {

/\* Configure PA1 as input for IRQ \*/

RCC->AHBENR |= 0x01; // Enable GPIOA clock (bit 0)

GPIOA->MODER &= ~(0x00000000); // General purpose input mode

/\* Configure PC0-PC7 as output pins to drive LEDs \*/

RCC->AHBENR |= 0x04; // Enable GPIOC clock (bit 2)

GPIOC->MODER &= ~(0x0000FFFF); // Clear PC0-PC7 mode bits

GPIOC->MODER |= (0x00005555); // General purpose output mode for PC0-PC7

RCC->AHBENR |= 0x02; // Enable GPIOB clock (bit 0)

GPIOB->MODER &= ~(0x0000FF00); // PB4-PB7 output keypad rows

GPIOB->MODER |= (0x00005500); // ^^^^

GPIOB->PUPDR &= ~0x0000FFFF; //clear bits 0-15 for PB0-PB7 \*HERE I AM MAKING SURE THE AND GATE READS LOW\*

GPIOB->PUPDR |= 0x00000055; //set bits 0-7 to 01 for PB0-PB3 pull-up resistors, \*CHECK THIS STEP\*

GPIOB->BSRR = 0x0010 << 16; // send 0 to pin 4

GPIOB->BSRR = 0x0020 << 16; // send 0 to pin 5

GPIOB->BSRR = 0x0040 << 16; // send 0 to pin 6

GPIOB->BSRR = 0x0080 << 16; // send 0 to pin 7

//counter setup

RCC->APB2ENR |= 0x00000008; //TIM10EN is enabled

TIM10->PSC = 0x19; //enable prescale register

TIM10->ARR = 0x1FFF; //enable auto reload register

TIM10->DIER |= 0x01; //enable interrupt from counter

TIM10->CNT; //enable counter

TIM10->SR &= ~0x01;

//EXTI SECTION

SYSCFG->EXTICR[0] &= 0xFF0F; //clears EXTI1 bit

SYSCFG->EXTICR[0] |= 0x0000; //set EXTI1 = 0 to select PA1

EXTI->FTSR |= 0x0002; //Bit0=1 to make EXTI1 falling-edge trig.

EXTI->IMR |= 0x0002; //Bit0=1 to enable EXTI1

EXTI->PR |= 0x0002; //Bit0=1 to clear EXTI1 pending status

//NVIC SECTION

NVIC\_EnableIRQ(7); //set bit n to enable IRQ7

NVIC\_ClearPendingIRQ (7); // clears pending status

NVIC\_EnableIRQ(TIM10\_IRQn); //set bit n to enable TIM10 IRQ

NVIC\_ClearPendingIRQ (TIM10\_IRQn); // clears pending status

//CPU SECTION

\_\_enable\_irq(); //enable interrupts

TIM10->CR1 |=0x01; //enable counting

}

/\*----------------------------------------------------------\*/

/\* TIM10\_IRQ26 Interrupt Function (signals the pressing of a keyboard button

/\*----------------------------------------------------------\*/

void TIM10\_IRQHandler ()

{

state = led1\*1+led2\*2+led3\*4+led4\*8;

state2 = led5\*1+led6\*2+led7\*4+led8\*8;

if(startstop==1) //running

{

if(state==9)

{

state=0;

if(state2==9)

{

state2=0;

}

else

{

state2++;

}

}

else

{

state++;

}

}

else

{

//wasting time like the writers of monty python

}

TIM10->SR &= ~0x01;

NVIC\_ClearPendingIRQ (TIM10\_IRQn); // clears pending status

}

/\*----------------------------------------------------------\*/

/\* EXTI1 Interrupt Function (signals the pressing of a keyboard button

/\*----------------------------------------------------------\*/

void EXTI1\_IRQHandler ()

{

int pb0=1; //reading from PB0

int pb1=1; //reading from PB1

int pb2=1; //reading from PB2

int pb3=1; //reading from PB3

int pb4=1; //reading from PB4

int pb5=1; //reading from PB5

int pb6=1; //reading from PB6

int pb7=1; //reading from PB7

int i,j,n;

// wait 1 ms

for (i=0; i<40; i++) //outer loop

{

for (j=0; j<18; j++)

{ //inner loop

n = j; //dummy operation for single-step test

} //do nothing

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//reading columns \*CHECK THIS STEP\*

GPIOB->MODER &= ~(0x000000FF); // PB0-PB3 input keypad columns

GPIOB->MODER |= (0x00000000); // ^^^^

GPIOB->MODER &= ~(0x0000FF00); // PB4-PB7 output keypad rows

GPIOB->MODER |= (0x00005500); // ^^^^

//Pull Up Pull Down Section \*CHECK THIS STEP\*

GPIOB->PUPDR &= ~0x0000FFFF; //clear bits 0-15 for PB0-PB7

GPIOB->PUPDR |= 0x00000055; //set bits 0-7 to 01 for PB0-PB3 pull-up resistors, \*CHECK THIS STEP\*

GPIOB->BSRR = 0x0010 << 16; // send 0 to pin 4

GPIOB->BSRR = 0x0020 << 16; // send 0 to pin 5

GPIOB->BSRR = 0x0040 << 16; // send 0 to pin 6

GPIOB->BSRR = 0x0080 << 16; // send 0 to pin 7

for (int x=0;x<4;x++); // just making sure it has time

pb0 = GPIOB->IDR & 0x0001; //reading PB0

pb1 = GPIOB->IDR & 0x0002; //reading PB1

pb2 = GPIOB->IDR & 0x0004; //reading PB2

pb3 = GPIOB->IDR & 0x0008; //reading PB3

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//reading rows

GPIOB->MODER &= ~(0x0000FF00); // PB4-PB7 input keypad rows

GPIOB->MODER |= (0x00000000); // ^^^^

GPIOB->MODER &= ~(0x000000FF); // PB0-PB3 output keypad columns

GPIOB->MODER |= (0x00000055); // ^^^^

GPIOB->PUPDR &= ~0x0000FFFF; //clear bits 0-15 for PB0-PB7

GPIOB->PUPDR |= 0x00005500; //set bits 08-15 to 01 for PB4-PB7 pull-up resistors, \*CHECK THIS STEP\*

GPIOB->BSRR = 0x0001 << 16; // send 0 to pin 0

GPIOB->BSRR = 0x0002 << 16; // send 0 to pin 1

GPIOB->BSRR = 0x0004 << 16; // send 0 to pin 2

GPIOB->BSRR = 0x0008 << 16; // send 0 to pin 3

for (int x=0;x<4;x++); // just making sure it has time

pb4 = GPIOB->IDR & 0x0010; //reading PB4

pb5 = GPIOB->IDR & 0x0020; //reading PB5

pb6 = GPIOB->IDR & 0x0040; //reading PB6

pb7 = GPIOB->IDR & 0x0080; //reading PB7

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//decide key

if((pb4==0) && (pb0==0)) // button 1

{

key=1;

}

else if((pb4==0) && (pb1==0)) // button 2

{

key=2;

}

else if((pb4==0) && (pb2==0)) // button 3

{

key=3;

}

else if((pb5==0) && (pb0==0)) // button 4

{

key=4;

}

else if((pb5==0) && (pb1==0)) // button 5

{

key=5;

}

else if((pb5==0) && (pb2==0)) // button 6

{

key=6;

}

else if((pb6==0) && (pb0==0)) // button 7

{

key=7;

}

else if((pb6==0) && (pb1==0)) // button 8

{

key=8;

}

else if((pb6==0) && (pb2==0)) // button 9

{

key=9;

}

else if((pb4==0) && (pb3==0)) // button A

{

key=10;

}

else if((pb5==0) && (pb3==0)) // button B

{

key=11;

}

else if((pb6==0) && (pb3==0)) // button C

{

key=12;

}

else if((pb7==0) && (pb3==0)) // button D

{

key=13;

}

else if((pb7==0) && (pb0==0)) // button \* (treat like E)

{

key=14;

}

else if((pb7==0) && (pb2==0)) // button # (treat like F)

{

key=15;

}

else if((pb7==0) && (pb1==0)) // button 0

{

key=0;

}

else{}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//handling key that was pressed

if(key==0) //toggling startstop

{

if(startstop==0)

{

startstop=1;

}

else

{

startstop=0;

}

}

else if (key==1) //reset was pressed

{

if(startstop==0)

{

state = 0;

state2 = 0;

}

else

{

//I could make another wasting time joke, but that would waste time

}

}

else

{

//lol we're doing nothing

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//display key on LEDs

// wait 1 ms

for (i=0; i<200; i++) //outer loop

{

for (j=0; j<18; j++)

{ //inner loop

n = j; //dummy operation for single-step test

} //do nothing

}

GPIOB->MODER &= ~(0x000000FF); // PB0-PB3 input keypad columns

GPIOB->MODER |= (0x00000000); // ^^^^

GPIOB->MODER &= ~(0x0000FF00); // PB4-PB7 output keypad rows

GPIOB->MODER |= (0x00005500); // ^^^^

GPIOB->PUPDR &= ~0x0000FFFF; //clear bits 0-15 for PB0-PB7 \*HERE I AM MAKING SURE THE AND GATE READS LOW\*

GPIOB->PUPDR |= 0x00000055; //set bits 0-7 to 01 for PB0-PB3 pull-up resistors, \*CHECK THIS STEP\*

GPIOB->BSRR = 0x0010 << 16; // send 0 to pin 4

GPIOB->BSRR = 0x0020 << 16; // send 0 to pin 5

GPIOB->BSRR = 0x0040 << 16; // send 0 to pin 6

GPIOB->BSRR = 0x0080 << 16; // send 0 to pin 7 resistors, \*CHECK THIS STEP\*

NVIC\_ClearPendingIRQ (7); // clears pending status

EXTI->PR |= 0x0002; //Bit0=1 to clear EXTI1 pending status

}

/\*---------------------------------------------------------------\*/

/\* Count - counts up or down based on value passed into function \*/

/\*---------------------------------------------------------------\*/

void count (a)

{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//updating tenths LEDs

switch(state)

{

case 0:

led1=0;

led2=0;

led3=0;

led4=0;

break;

case 1:

led1=1;

led2=0;

led3=0;

led4=0;

break;

case 2:

led1=0;

led2=1;

led3=0;

led4=0;

break;

case 3:

led1=1;

led2=1;

led3=0;

led4=0;

break;

case 4:

led1=0;

led2=0;

led3=1;

led4=0;

break;

case 5:

led1=1;

led2=0;

led3=1;

led4=0;

break;

case 6:

led1=0;

led2=1;

led3=1;

led4=0;

break;

case 7:

led1=1;

led2=1;

led3=1;

led4=0;

break;

case 8:

led1=0;

led2=0;

led3=0;

led4=1;

break;

case 9:

led1=1;

led2=0;

led3=0;

led4=1;

break;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//updating seconds LEDs

switch(state2)

{

case 0:

led5=0;

led6=0;

led7=0;

led8=0;

break;

case 1:

led5=1;

led6=0;

led7=0;

led8=0;

break;

case 2:

led5=0;

led6=1;

led7=0;

led8=0;

break;

case 3:

led5=1;

led6=1;

led7=0;

led8=0;

break;

case 4:

led5=0;

led6=0;

led7=1;

led8=0;

break;

case 5:

led5=1;

led6=0;

led7=1;

led8=0;

break;

case 6:

led5=0;

led6=1;

led7=1;

led8=0;

break;

case 7:

led5=1;

led6=1;

led7=1;

led8=0;

break;

case 8:

led5=0;

led6=0;

led7=0;

led8=1;

break;

case 9:

led5=1;

led6=0;

led7=0;

led8=1;

break;

}

if (led1 == 0)

GPIOC->BSRR = 0x0001 << 16;

else

GPIOC->BSRR = 0x0001;

if (led2 == 0)

GPIOC->BSRR = 0x0002 << 16;

else

GPIOC->BSRR = 0x0002;

if (led3 == 0)

GPIOC->BSRR = 0x0004 << 16;

else

GPIOC->BSRR = 0x0004;

if (led4 == 0)

GPIOC->BSRR = 0x0008 << 16;

else

GPIOC->BSRR = 0x0008;

if (led5 == 0)

GPIOC->BSRR = 0x0010 << 16;

else

GPIOC->BSRR = 0x0010;

if (led6 == 0)

GPIOC->BSRR = 0x0020 << 16;

else

GPIOC->BSRR = 0x0020;

if (led7 == 0)

GPIOC->BSRR = 0x0040 << 16;

else

GPIOC->BSRR = 0x0040;

if (led8 == 0)

GPIOC->BSRR = 0x0080 << 16;

else

GPIOC->BSRR = 0x0080;

}

/\*------------------------------------------------\*/

/\* Main program \*/

/\*------------------------------------------------\*/

int main(void)

{

PinSetup(); //Configure GPIO pins

//the infinite loop will begin by counting up from zero once SW1

while(1)

{

count(0);

}

}