Microcontrollers & GPIO

Student 1, Student 2, Student 3

Syracuse University

Microcontrollers & GPIO

Introduction

In the following experiments, we learn about two different microcontrollers, the ATMEGA328P and the ATSAMD21E18A. For our experiments, we are tasked with understanding how to work with GPIO on the microcontrollers and write programs that rely on GPIO input and output. We wrote our code in c using MPLAB and used a SNAP programmer in order for the microcontroller to understand our code and produce the desired results. We used the Waveforms program with the Analog Discovery 2 to examine our inputs and outputs as well as provide power to our circuit.

Experiment 1 ATMega328P

Introduction

In experiment 1, the goal is to build a machine based on ATMega328P which will flip a GPIO and show the square waveforms. Also, we need to measure the current and resistance of pull-up and pull-down resistors. Finally, we need to connect to the pushbutton which can control the start and stop the output pin from flipping.

Materials

- ATMega328P
- Snap programmer
- Analog Discovery 2
- 680 Ohm resistor
- Pushbutton.
- Wires

Experimental Setup

According to the 28 SPDIP chart in the datasheet of ATMega328P, we connected 2 (VTG) from Snap programmer to pin7 (ATMega328P), 3 (GND) to pin8, 4 (MISO) to pin18, 5 (SCK) to pin19, 6 (NRESET) to pin1, and 7 (MOSI) to pin17. According to the Port C charts in the datasheet of ATMega328P, We chose pin C5 so that the three main registers that will control this pin are PORTC, DDRC, and PINC. And for the pushbutton, we connected it to the GND, VTG, and input pins on ATMega328P. (check Figure 1.1, 1.2, 1.3)

Experimental Procedure

The code DDRC = 16; will make PIN 5 as the output pin. Then we write a while loop to create the square waveform. The if statement “if ((PINC & 00000001) == 00000001)” is to test

whether the pushbutton was pushed or not. If yes, the program will set PORTC = 0; so that it will stop the output pin from flipping. If not, it will run “PORTC = 16; PORTC = 0;” and create square waveform. “PORTC = 16” will make pin5 a 1, and “PORTC = 0” will make pin5 a 0. (check Figure 1.4, 1.5)

Results

We used a 680 Ohm resistor. The low voltage change (the pull-down) circuit is 0.3V which lowers from 4.2V to 3.9V.
The current = 3.9V / 680 Ohm = 0.00574 Amps
The Req = 0.3V / 0.00574A = 52.26 Ohms

The high voltage change (the pull-up) circuit is 0.2V which raises from 4.2V to 4.4V.
The current = 4.4V / 680 Ohms = 0.0065 Amps
The Req = 0.2V / 0.0065A = 30.77 Ohms

Experiment 2 ATSAMD21E18A

Introduction

In experiment 2, the goal is to build a machine based on ATSAMD21E18A which will flip a GPIO and show the square waveforms. Also, we need to measure the current and resistance of NMOS and PMOS resistors. Finally, we need to connect to the pushbutton which can control the start and stop the output pin from flipping.

Materials

- ATSAMD21E18A
- Snap programmer
- Analog Discovery 2
- 680 Ohm resistor
- Pushbutton.
- Wires

Experimental Setup

According to the QFN32/TQFP32 chart in the datasheet of ATSAMD21E18A, we connected 2 (VTG) from Snap programmer to pin30 (ATSAMD21E18A), 3 (GND) to pin10 or 28, 5 (SWCLK) to pin31, 6 (NREST) to pin26, 8 (SWDIO) to pin32. And for the pushbutton, we connected it to the GND, VTG, and input pins on ATSAMD21E18A. (check Figure 2.1, 2.2)

Experimental Procedure

The DIRSET sets the pin to input or output, and writing 1 to a port with DIRSET will make it an output. We wrote “PORT_REGS -> GROUP[0].PORT_DIRSET = 0b00000000000000000000000000100000” to set pin 6 to an output 00000000000000000000000000100000. Writing 1 to a port with DIRCLR with configuring it as an input.
Then we write a while loop to create the square waveform. The if statement “if ((PORT_REGS -> GROUP[0].PORT_IN & 0b00000000000000000000000000010000) ==

0b00000000000000000000000000010000)” is to test whether the pushbutton was pushed or not. If yes, the program will set OUTCLR to 0b00000000000000000000000000100000; (Writing 1 to a port with OUTCLR will drive it low) so that it will stop the output pin from flipping. If not, it will set OUTSET to 0b00000000000000000000000000100000; (Writing 1 to a port with OUTSET will drive it high) and create a square waveform. (check Figure 2.3)

Results

We used a 680 Ohm resistor. When it connects to the Ground (NMOS), the voltage changes from 5V to 4.619V. (check Figure 2.4)

The Current = 4.691 V / 680 Ohm = 0.00690 Amps
The Req = (5V – 4.691V) / 0.00690 Amps = 44.78 Ohms

When it connects to the voltage (PMOS), the voltage changes from 5V to 5.147V. (check Figure 2.5)
The Current = 5.147 V / 680 Ohm = 0.0076Amps
The Req = (5V - 5.147V) / 0.00076Amps = 19.34 Ohms

Experiment 3 Applying GPIOs, LEDs, and Multiplexing

Introduction

In experiment 3, the aim was to continue learning more about GPIO and microcontrollers by creating a 0 to 99 counter with three pushbuttons to control three different functions on the board. The first pushbutton will reset the value of the counter to 00. The second pushbutton will change the direction of the counter. The third pushbutton will increment or decrement the count (depending on the value of the direction). This experiment is more GPIO heavy than the two previous experiments and requires careful thinking about organizing the circuit, assigning pins, and writing the C code for the counter.

Materials

- ATSAMD21E18A
- MPLAB Snap Programmer
- Analog Discovery 2
- 10 680 Ohm resistors
- 3 pushbuttons
- 2 Seven segment LED Displays
- Wires

Experimental Setup

In our experiment we chose to stick with the ATSAMD21E18A (ATSAMD) chip and we chose to display our counter on two seven segment displays. The alternative choices were the ATMega328P chip and/or using an LED matrix display. Constraints on the project are that the project must be programmed in C using MPLAB and that only 9 wires may be used to drive the display.

The circuit will go through several stages during this experiment and look different at certain stages. When first learning how to turn on an LED from a seven segment display you will need the Snap Programmer and ATSAMD set up as they were in Experiment 2, with the addition

of a GPIO output pin wired to one of the letter terminals on the seven segment display. You will also need to connect a wire from a “com” terminal to GND. Refer to **Figure 3.1** for the seven segment terminal layout.

Figure 3.3 shows an example of how the first circuit can be configured to turn on an LED. Note how there is a resistor connected in series between the port and seven segment letter “A” terminal. All connections to a seven segment letter terminal will need a resistor connected in series. Use a resistor that would limit the current to about 1mA if the LED and resistor were connected between power and ground.

Figure 3.6 shows how a circuit may be wired to display a single number on the seven segment display without multiplexing. Again, series resistors are used for each connection to a letter terminal on the seven segment display. The experimental procedure describes how to complete the connections for using both displays while multiplexing.

Figure 3.12 shows how pushbuttons should be introduced into the circuit.

Experimental Procedure

The first task in experiment 3 is to create a software delay for flipping a GPIO. This task requires a call to a method that has a for loop run for a variable amount of time. The parameter of that method call will determine how many times the for loop iterates and thus, how long the delay will be. This method can be seen in use in **Figure 3.4**.

After successfully flipping a GPIO, the next task is to turn on an LED. Once you have the circuit set up, you need to set a GPIO as an output. The PORT_REGS->GROUP[0].PORT_DIRSET line in **Figure 3.4** will accomplish this. Set the pin(s) that you want as outputs to 1 in this 32 bit binary number. The code in **Figure 3.4** will flip

the LED on for 1ms and off for 4ms, run for 200 times, and then run another 200 cycles of turning the LED on for 4ms and off for 1ms. **Figure 3.5** show the signal to the LED.

Once an LED has been turned on successfully, the next step in the project is to display an image on the seven segment displays without yet multiplexing. The task is to show the number "2" on a seven segment display. The layouts of the different numbers on a seven segment display can be seen in **Figure 3.2**.Once the circuit has been set up, you will need to write 7 GPIO pins as outputs and include their bits in the OUTCLR and OUTSET lines in your loops. The structure of the code does not change and can be seen in **Figure 3.7**.

Multiplexing is a technique that combines multiple signals into one signal over the same medium. In the context of our project we will be using multiplexing to combine the signals from the ports, into one pushbutton, and through another, rather than using 14 resistors, and many wires. Notice how in **Figure 3.8** we use the same signal coming from a GPIO pin in both seven segment displays.

To display an image with multiplexing you will need to turn off the second display, show the number you want in the first display, clear+turn off the first display, and then turn on the second display, show the number, and clear+turn off the second display. At very high frequencies this process will make the seven segment displays appear to both show numbers at the same time, when really they are taking turns. This process can be seen in **Figure 3.9**.

The next task is to add a pushbutton to count up. In order to do this three new variables have to be added. The variables oneCount and tenCount refer to the ones and tens digit of the number that we are trying to display. A flag variable is needed to ensure that count is properly incremented, as without it one click of the pushbutton may be interpreted as several. In the code for incrementing the count, oneCount and tenCount have their own switch cases. The switch

cases work almost identically and turn off the other display when one is being used. When the button is pressed, the flag variable is used in a while loop. Flag is initialized as 1 and when it is one the program goes through the logic of finding the next proper count values and then sets flag to 0. When the flag is 0 in this while loop, it is set to 1 again. All of this can be seen in the **Figure 3.10**.

Now that a simple counter has been implemented. The final step is to expand upon this and create the complete counter mentioned earlier. The code is the exact same, except there are new flags for the reset button, flagReset, and for the direction button flagDirection. There is also a new variable named direction, that dictates whether the count will be increased or decreased when the increment button is pressed. Both variables are initialized at 1. When the reset button is pressed, oneCount and tenCount are set to 0 and its flag is set to 0. When the change direction button is pressed, the direction variable, which is initialized at 0 is changed to either a 1 or a 0 depending on what it was prior to the button being pressed, and the flag is set to 0. The direction variable is checked in the increment button if statement to decide if it should use our increment or decrement logic. Look to **Figure 3.11** to see how these new changes are implemented.

Results

After following the setup and procedure of this experiment you should have a fully functioning 2 digit counter with 3 different functions. This experiment highlighted some of the strength of microcontrollers. One of which was its ability to time things, which we saw used when implementing the software delay. We also saw how microcontrollers can easily add new I/O ports to their function. This experiment went from using one GPIO pin to turn on an LED, to using twelve GPIO pins for the full implementation. After learning multiplexing the objective of

the constraints on the project become clear. The clever programming here gave us a cleaner circuit, with less materials used.

Experiment 4 Timers and Interrupts

Introduction

Our goal for this experiment was to use the same circuit we built for experiment 3 and implement a hardware delay and interrupt. The purpose for the new delay was to become familiar with activating and using the delay built into our ATSAMD21E18A microcontroller and use it as a replacement for our software delay function that we used in experiment 3 which, was used to hold our output on the display instead of flashing it. Our interrupt is meant to act as a replacement for the flag we used in our logic to determine if the increment button is pressed. The flag was used in order to make sure one button press was only interpreted as one button press. The interrupt would do the same thing, just use the hardware built into our microcontroller instead of using our software version.

Materials

- ATSAMD21E18A
- snap programmer
- analog discovery 2
- 10 680 Ohm resistors
- 3 pushbuttons
- 2 seven-segment LED displays
- Wires

Experimental Setup

The setup for this experiment is the exact same as it was for experiment 3. We use the same circuit with the ATSAMD21E18A microcontroller, three pushbuttons, ten 680 ohm resistors and two seven-segment LED displays, as well as the same connections.

Experimental Procedure

The first step for our experiment was to write and test a hardware delay function using the ATSAMD21E18A. To do so, we needed to create a new function called setUP(), that would configure our microcontroller to use the built-in delay. Our setup for the hardware delay turns on the synchronous clock and asynchronous clock to TC5, sets wavegen operation to use CC0 as top, sets the value in the 16-bit output compare register and finally, enables the TC5. In order to set up our timer in the ATSAMD21E18A, we use control register A to set the counter to 16-bit mode, instead of 32-bit and set the prescaler to 1. For our testing purposes, we just wanted to see if the delay worked, so we used a new function micros(x) that resets the timer to 0 and uses a number x, a variable given in the parameter of the function, as our time to delay. Our setUP() and micros(x) functions are shown in Figure 4.1. We created a new main() that flipped the output of a pin, from 0 to 1, every time the delay passed to test that our function worked as expected. A snip of our output while testing the micros(x) function can be seen in Figure 4.2.

After we confirmed our delay worked, we added our setUP() function to our code from experiment 3, replaced the delay(m) function with micros(x) and called micros(x) everywhere we had previously called delay(m).

Now that we had successfully implemented our hardware delay, we needed to test and implement our interrupt. The procedure was the same as it was for the delay, we need to

configure the hardware to use the interrupt in setUP() and instead of a new function, we call the EIC_Handler() that acts as a function that is only called when the GPIO detects input, our button press. The setUP() function turns on the input enable, pull enable, and mux enable for function A for PA16, our input, turns on the synchronous clock and asynchronous clock to EIC, enables the external interrupt for the GPIO input pin, configures the interrupt to 0 on a filtered, rising edge, enables the external interrupt controller and finally, enables the global interrupt. The EIC_Handler() for our testing, sets the output to high when the button is pressed and clears the interrupt flag. Our new setUP() that includes the configuration for interrupts and EIC_Handler() can be seen in Figure 4.3. Our output for our working interrupt can be seen in Figure 4.4.

With our interrupt working, we implemented it into our code from before, added the interrupt configuration to setUP(), added the EIC_Handler(), removed our flag for incrementing, changed oneCount, tenCount, and direction to volatile variables, and moved our increment button logic, into the EIC_Handler().

Results

The results for this experiment, is a working circuit that functions the same as the one in experiment 3. The circuit increments the count on the display when one button is pressed, changes direction so that the count can be decremented by one or vice versa when another is pressed, and resets the count when a third is pressed. This time, our code utilizes the hardware delay and interrupt built into our microcontroller as a replacement to the delay function and one of the flags we used previously, and includes one new function, setUP(), in order to configure the microcontroller. Our completed code with the hardware delay and interrupt implemented can be seen in Figure 4.5.

Conclusion

In Experiment 1 and 2, we used different microcontrollers (ATMega328P and ATSAMD21E18A) to do the similar things. In Experiment 1, we set up the machine based on the 28 SPDIP chart in the datasheet of ATMega328P, and create the square waveform that shows the flipping GPIO. We used a 680 Ohm resistor and measured the current and the Req for the pull-down and pull-up circuits. Then, we set a push button that stops the output pin from flipping. Similar to Experiment 1, Experiment 2 just changed the microcontroller and datasheet to ATSAMD21E18A and the QFN32/TQFP32 chart in the datasheet of ATSAMD21E18A. In Experiment 3, we created a 0 to 99 counter, and we connected it to three pushbuttons. The first pushbutton will reset the value of the counter to 00. The second pushbutton will change the direction of the counter. And the third pushbutton will increment or decrement the count (depending on the value of the direction). Experiment 4 was similar to Experiment 3 except for the delay and interrupt. We implemented a hardware delay, and it interrupted and used the interrupt instead of the flag we used for the increment button being pressed.

Appendix

Figure 1.1: Machine with ATMega328P

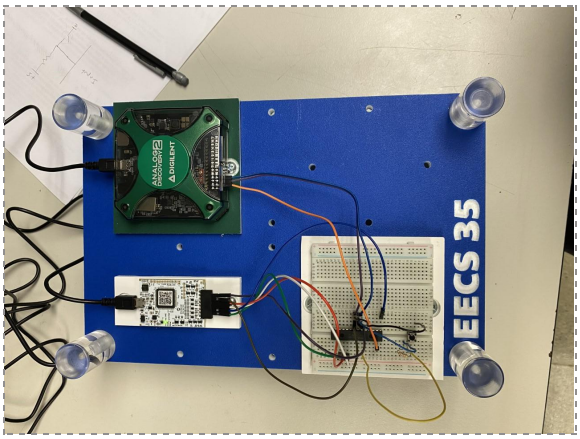


Figure 1.2: the 28 SPDIP chart in the datasheet of ATMega328P

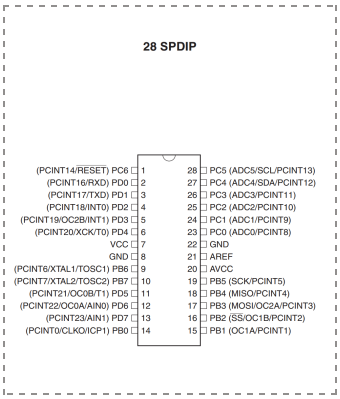
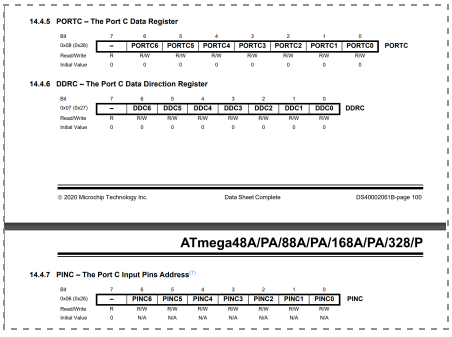


Figure 1.3: the Port C charts in the datasheet of ATMega328P


14.4.5 PORTC – The Port C Data Register

Bit	7	6	5	4	3	2	1	0	
0x08 (0x28)	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

14.4.6 DDRC – The Port C Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x07 (0x27)	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

14.4.7 PINC – The Port C Input Pins Address[1]

Bit	7	6	5	4	3	2	1	0	
0x06 (0x26)	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Figure 1.4: Code of Experiment 1

```
#include <xc.h>

void main(void) {

    DDRC = 16; //will make pin 5 an output 0B 00010000

    while(1)
    {
        if((PINC & 00000001) == 00000001)
        {
            //PORTC = 16;
            PORTC = 0;
        }
        else
        {
            PORTC = 16;
            PORTC = 0;
        }

    }
    return;
}
```

Figure 1.5: Square waveform of ATMega328P

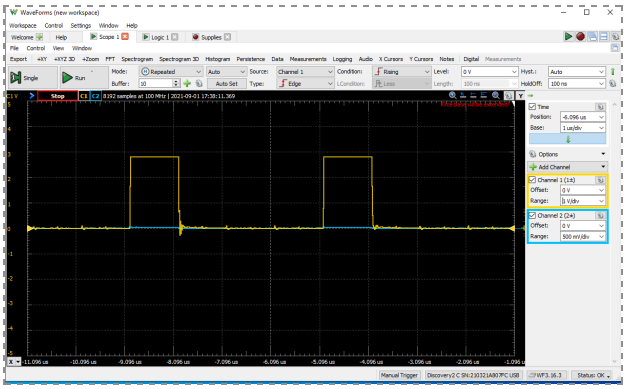


Figure 2.1: Machine with ATSAMD21E18A

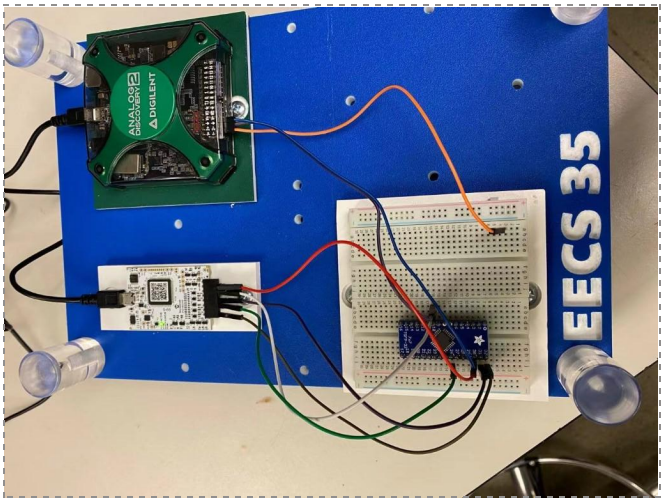


Figure 2.2: the QFN32/TQFP32 chart in the datasheet of ATSAMD21E18A

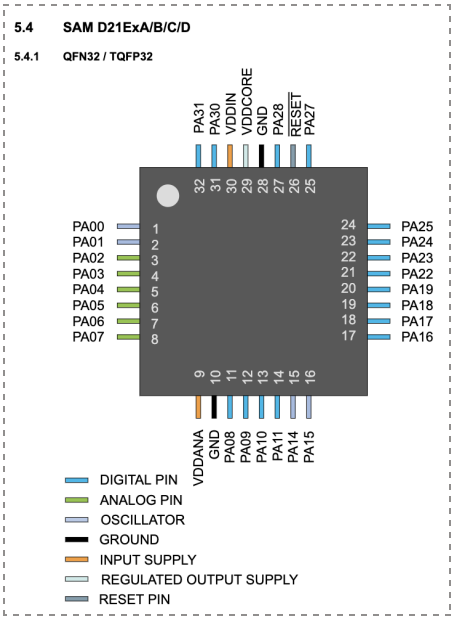


Figure 2.3: Code of Experiment 2

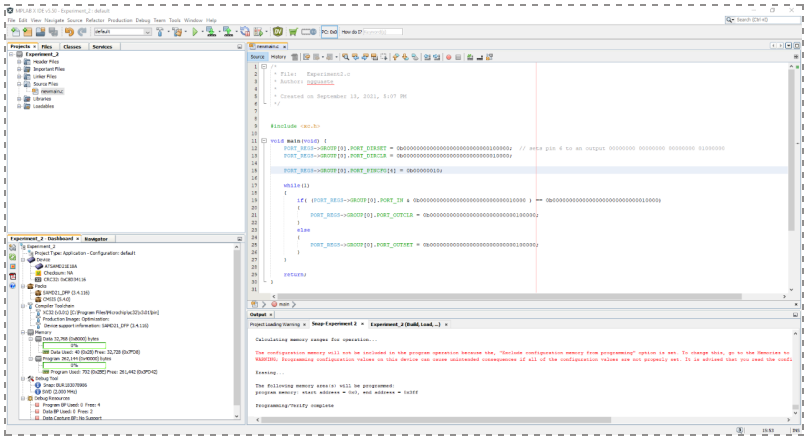


Figure 2.4: Square waveform of NMOS

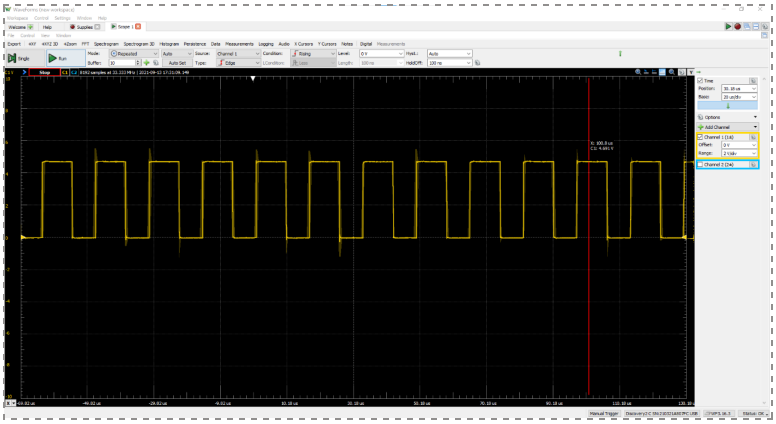


Figure 2.5: Square waveform of PMOS

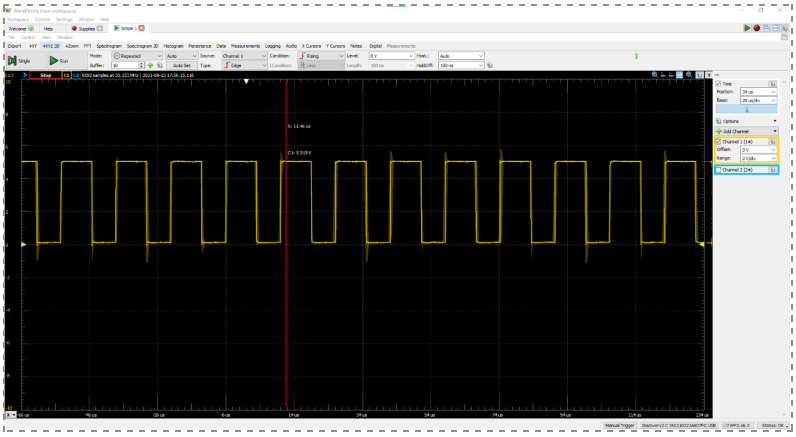


Figure 3.1: Layout of Seven Segment Display Pins

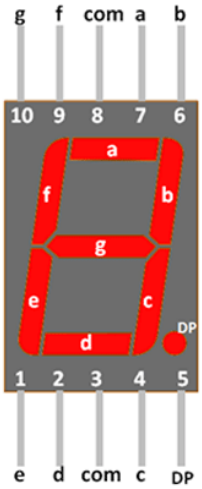


Figure 3.2: Different Number Layouts on a Seven Segment Display

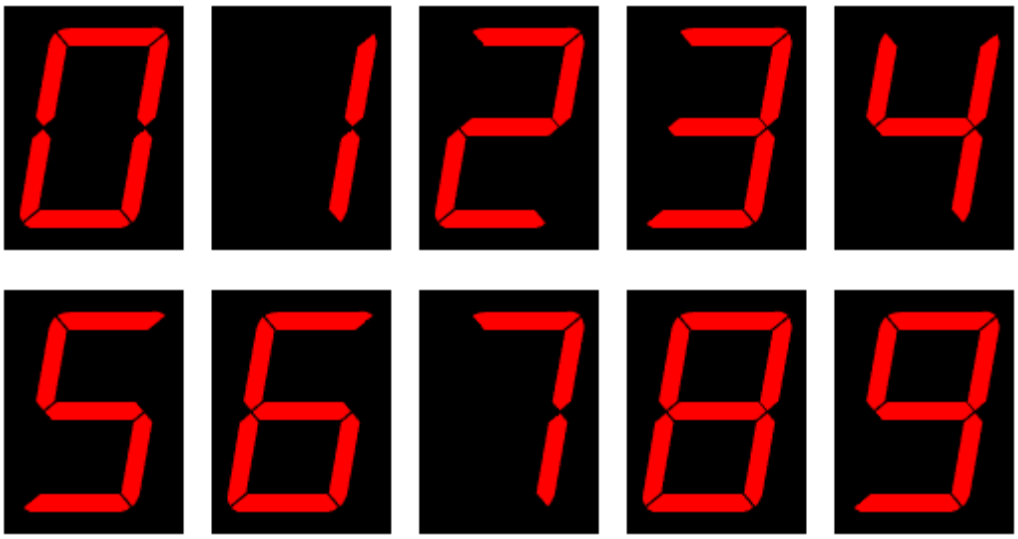


Figure 3.3: Circuit for Turning on an LED

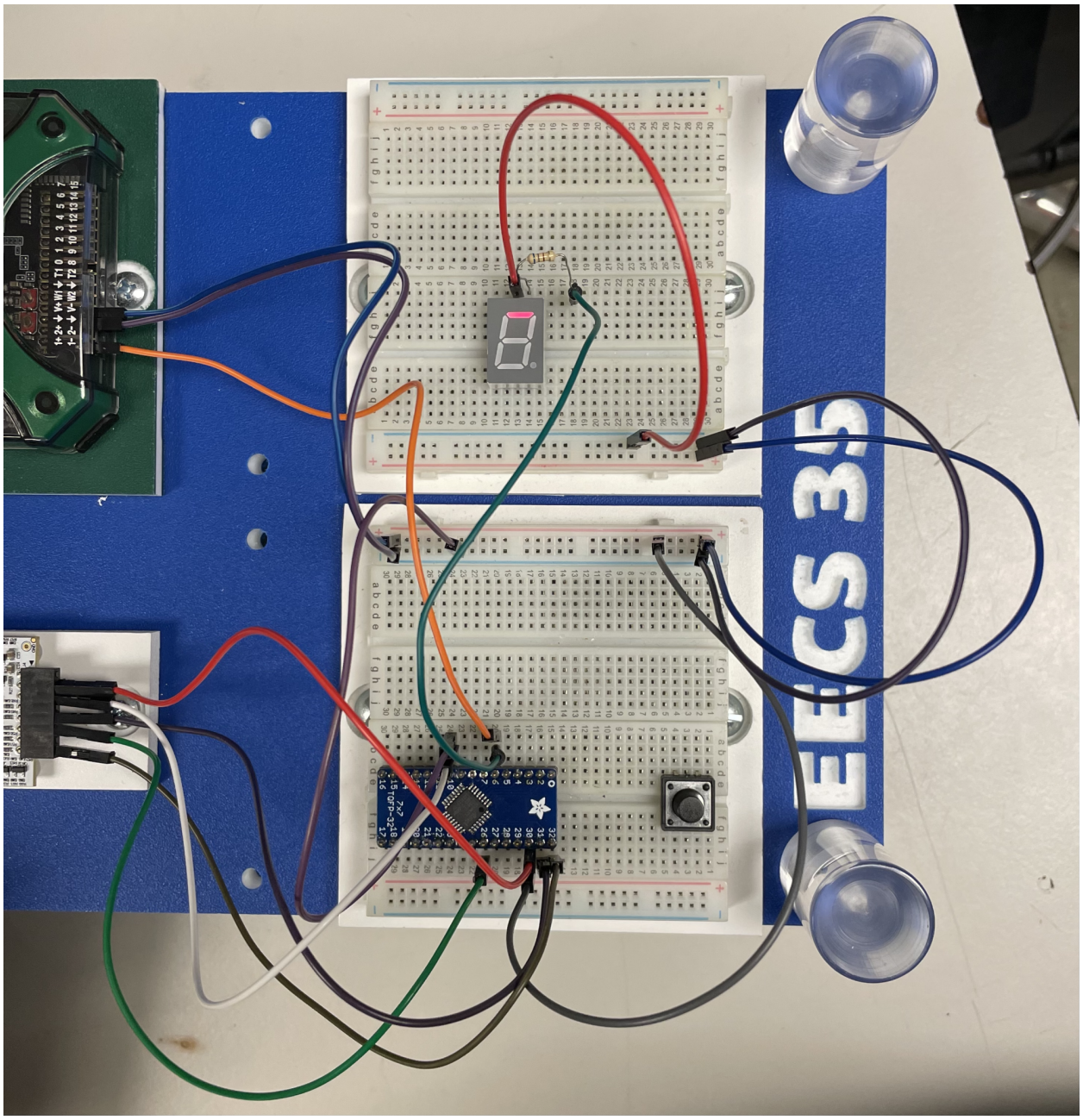


Figure 3.4: Code for Turning on an LED

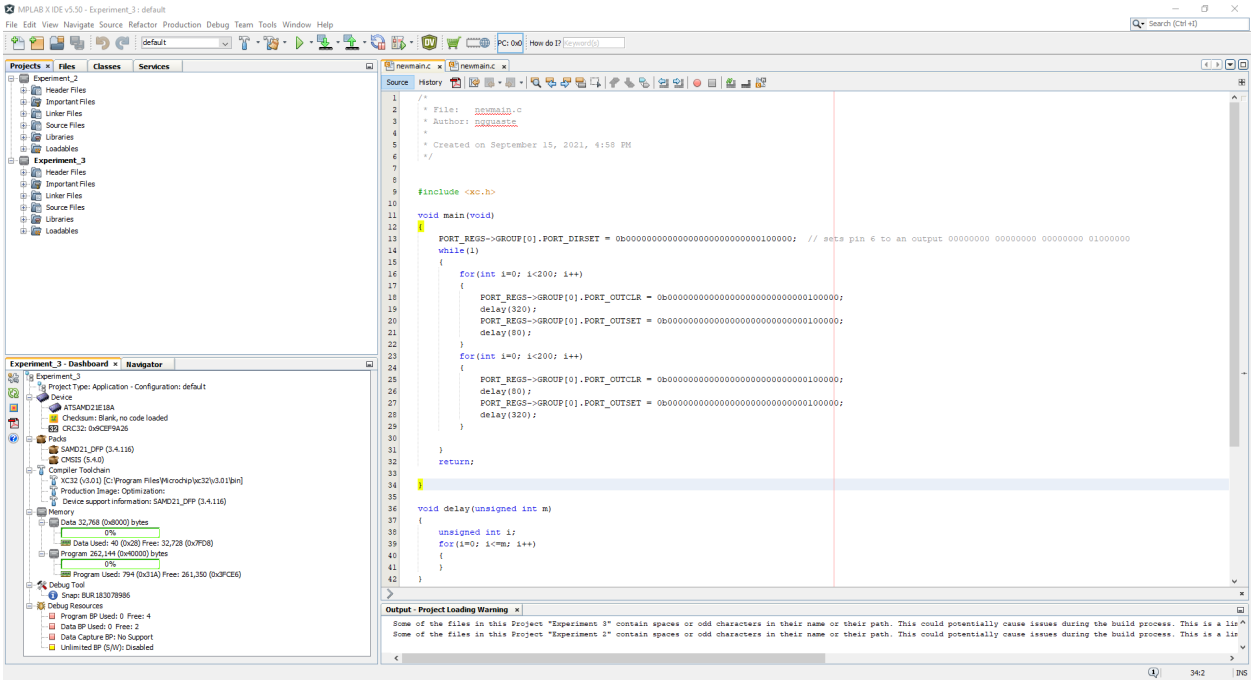
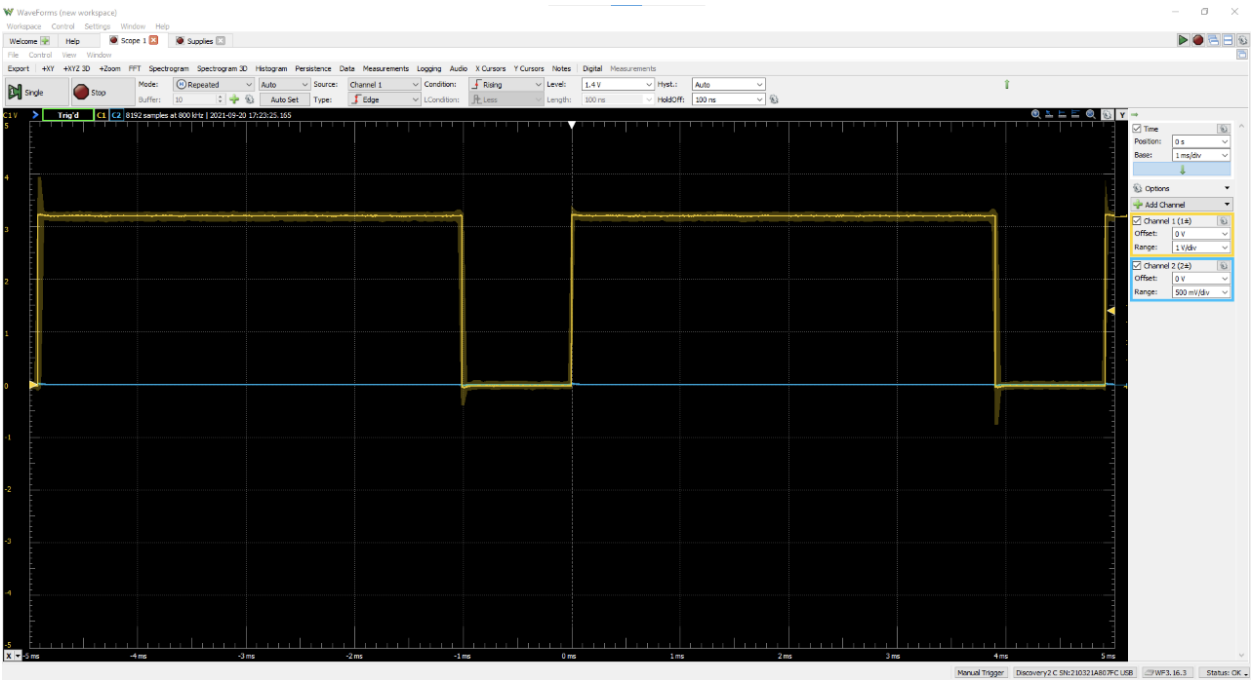


Figure 3.5: Signal when Turning on an LED



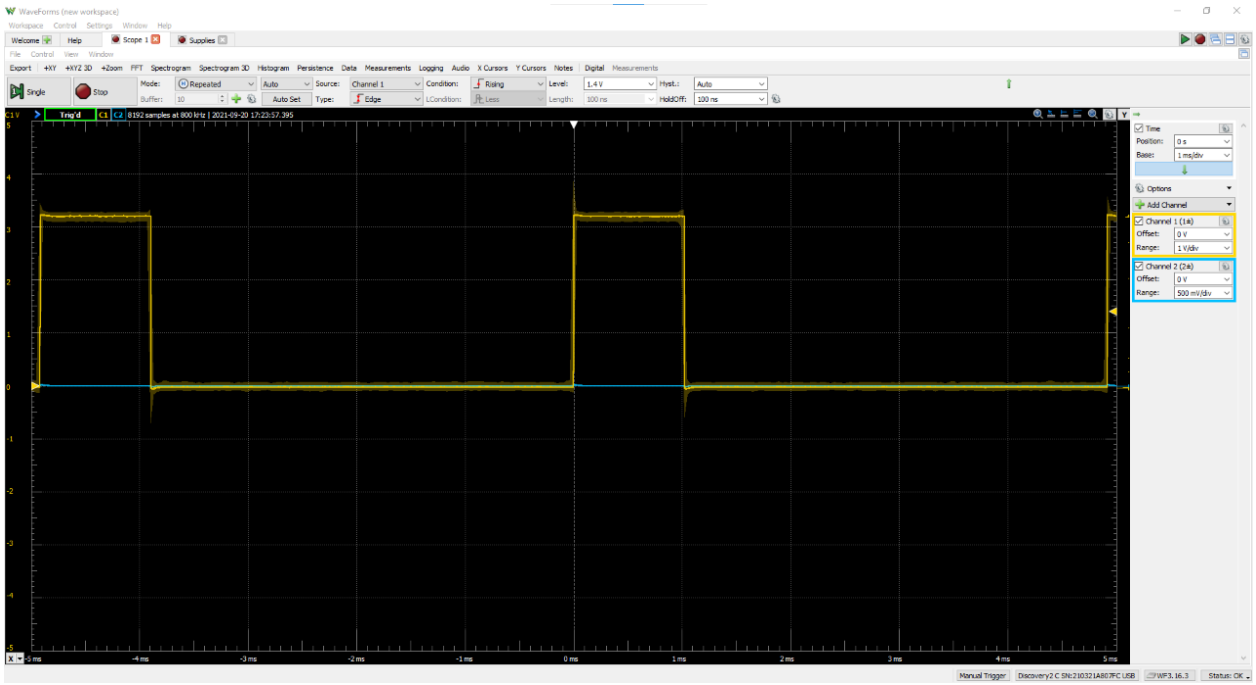


Figure 3.6: Circuit for Displaying an Image without Multiplexing

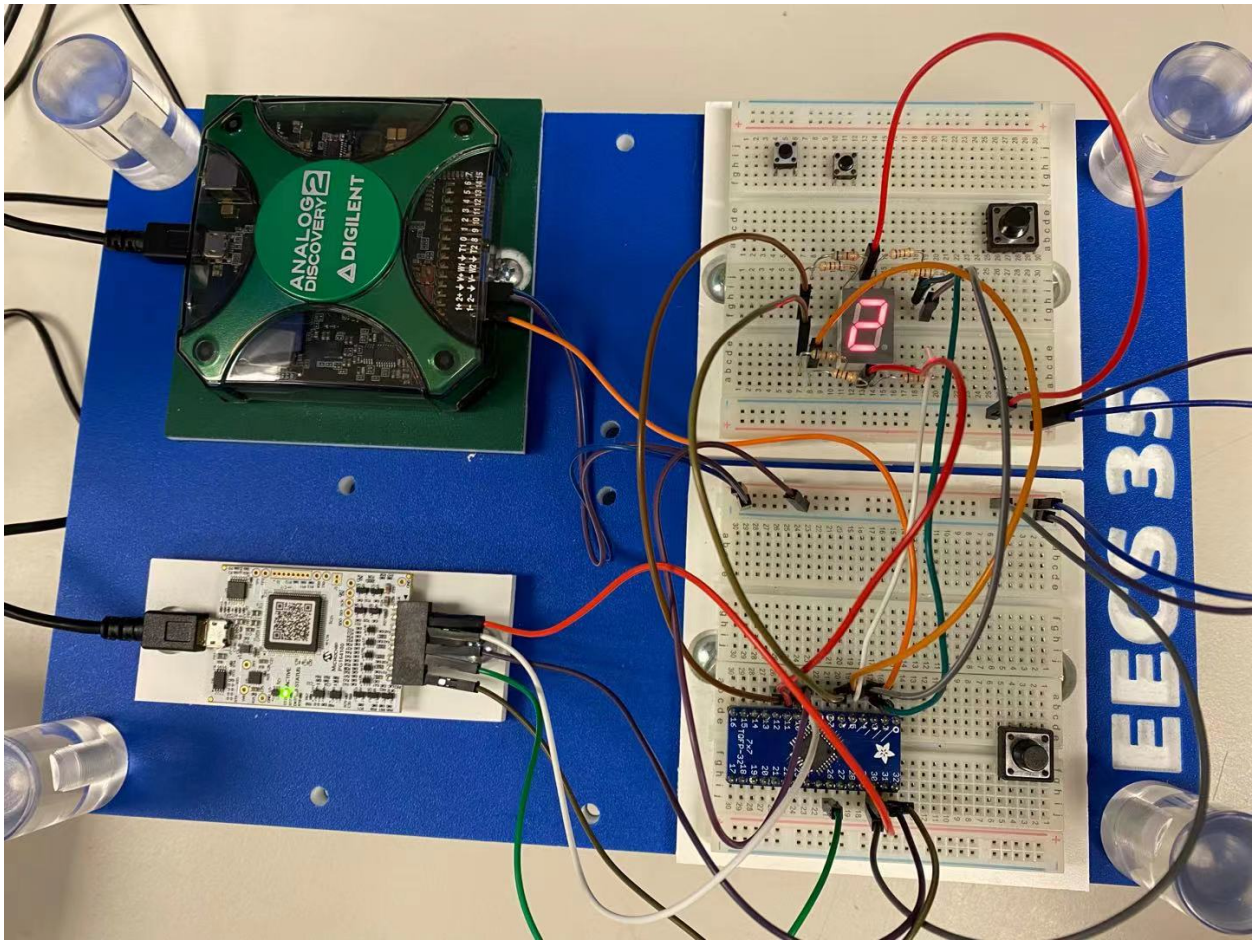


Figure 3.7: Code for Displaying an Image without Multiplexing

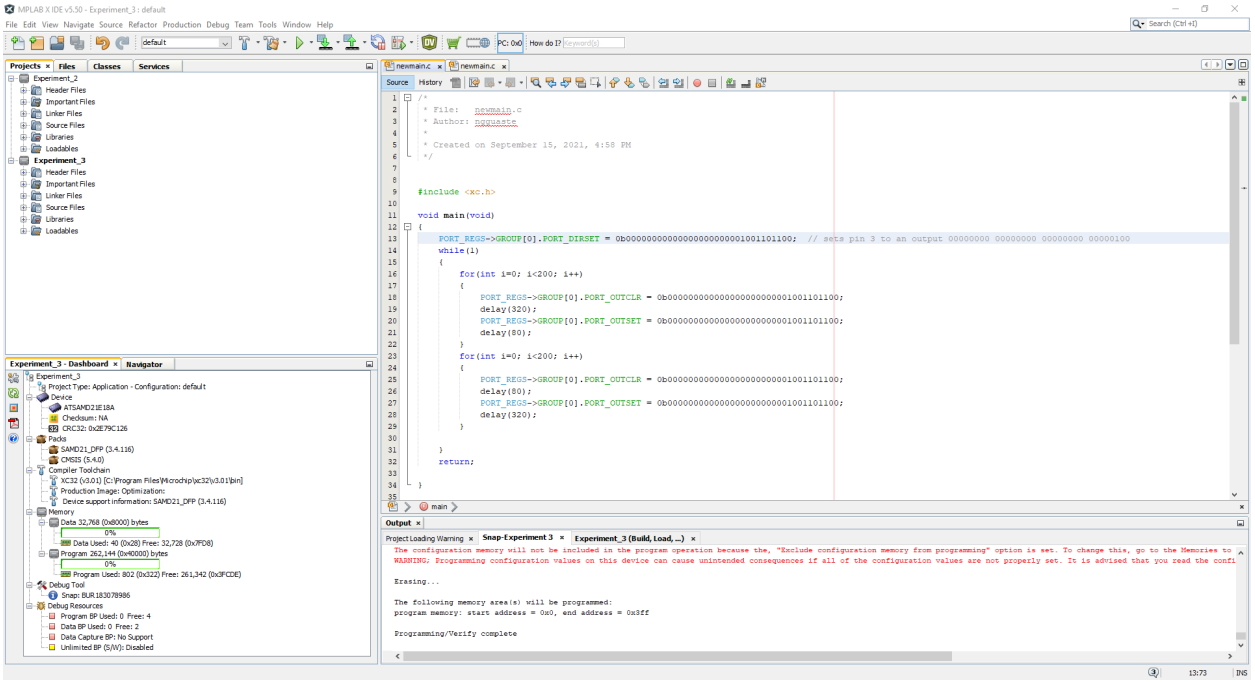


Figure 3.8: Circuit for Displaying an Image with Multiplexing

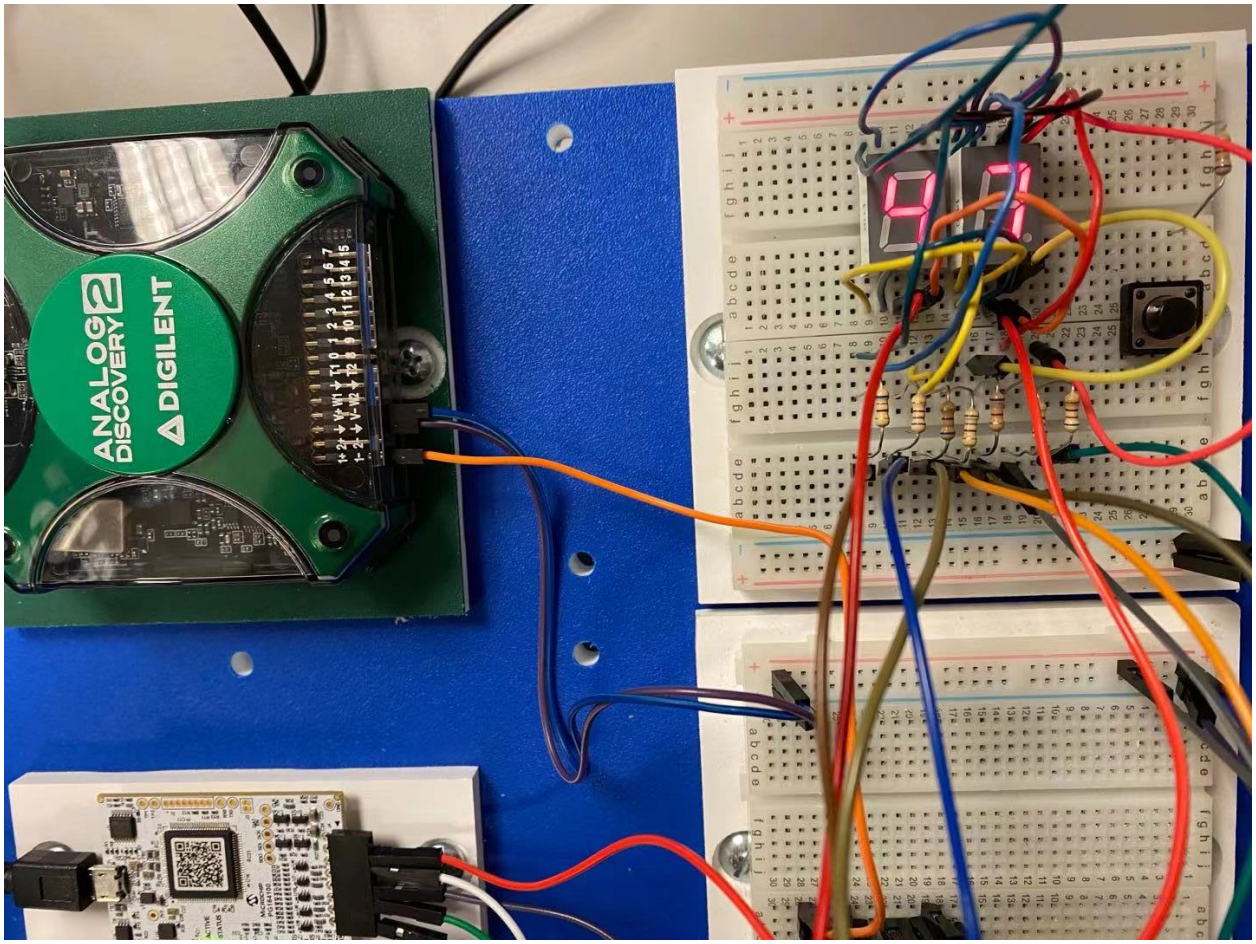


Figure 3.9: Code for Displaying an Image with Multiplexing

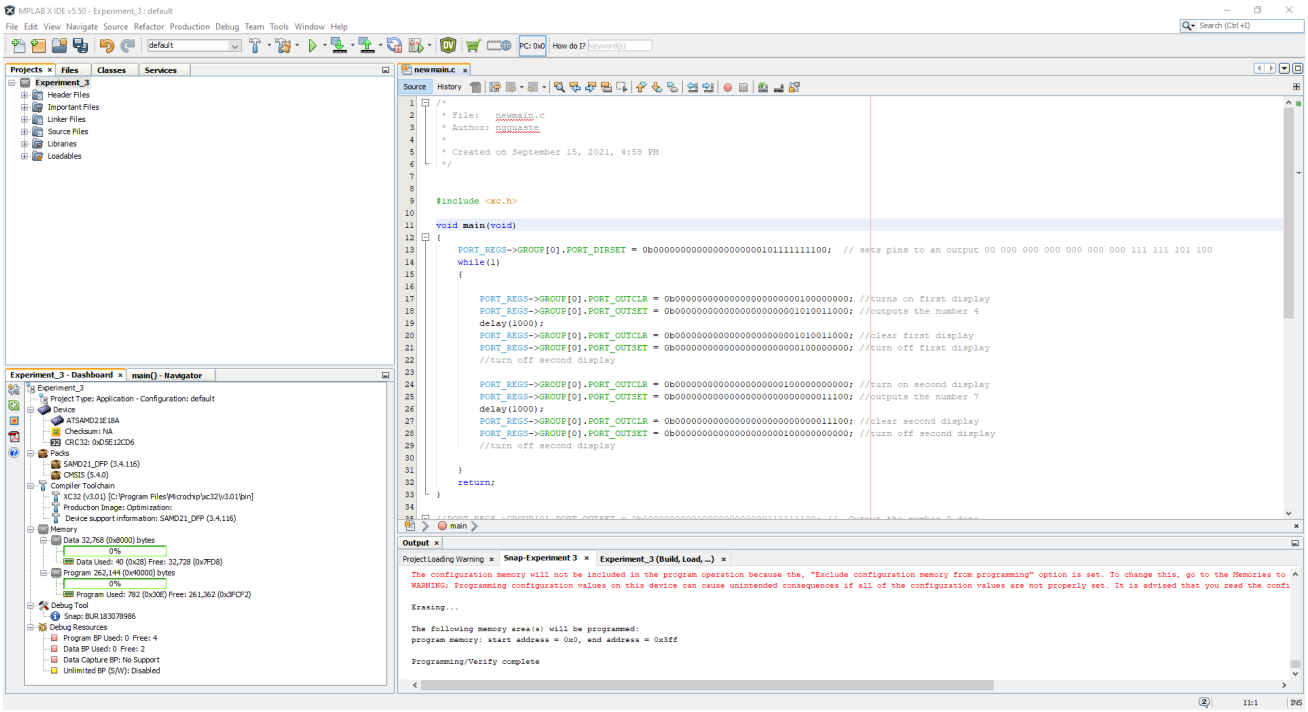


Figure 3.10: Code for Adding One Push Button to Count Up

```
/*
 * File:   newmain.c
 * Author: ngguaste
 *
 * Created on September 15, 2021, 4:58 PM
 */


#include <xc.h>

void main(void)
{
        PORT_REGS->GROUP[0].PORT_DIRSET =
0b00000000000000000000101111111100;   // sets pins to an output
        PORT_REGS->GROUP[0].PORT_DIRCLR =
0b00000000000000001000000000000000;  // Pushbutton for counting up set as input on
PA16, physical pin 17
        PORT_REGS->GROUP[0].PORT_PINCFG[16] = 0b00000010;
// Input enable for PA16, physical pin 7

        int tenCount = 0;
```

```
        int oneCount = 0;
        int flag = 1;

        while(1)
        {
        if ((PORT_REGS->GROUP[0].PORT_IN &
0b00000000000000010000000000000000) =0b00000000000000010000000000000000)
        {
        if (flag == 1)
        {
        if (oneCount < 9)
        {
                oneCount = oneCount + 1;
                flag = 0;
        }
        else if ((tenCount == 9)&&(oneCount == 9))
        {
                tenCount = 0;
                oneCount = 0;
                flag = 0;
        }
        else if (oneCount == 9)
        {
                oneCount = 0;
                tenCount = tenCount + 1;
                flag = 0;
        }
        }
        }
        else
        {
        flag = 1;
        }

        //Switch case for the ones digit (second display)
        switch(oneCount)
        {
        case 0:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000011111100; //  Output the number 0
```

```
                delay(1000);
                break;

        case 1:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000000011000; //  Output the number 1
                delay(1000);
                break;

        case 2:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001001101100; //  Output the number 2
                delay(1000);
                break;

        case 3:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001000111100; //  Output the number 3
                delay(1000);
                break;

        case 4:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
```

```
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001010011000; //  Output the number 4
                    delay(1000);
                    break;

          case 5:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001010110100; //  Output the number 5
                    delay(1000);
                    break;

          case 6:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001011110100; //  Output the number 6
                    delay(1000);
                    break;

          case 7:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000000011100; //  Output the number 7
                    delay(1000);
                    break;

          case 8:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
```

```
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001011111100; //  Output the number 8
                  delay(1000);
         break;

         case 9:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001010111100; //  Output the number 9
                  delay(1000);
                  break;

         }

         //Switch case for the tens digit (first display)
         switch(tenCount)
         {
         case 0:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000011111100; //  Output the number 0
                  delay(1000);
                  break;

         case 1:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000000011000; //  Output the number 1
                  delay(1000);
```

```
                break;

        case 2:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001001101100; //  Output the number 2
                delay(1000);
                break;

        case 3:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001000111100; //  Output the number 3
                delay(1000);
                break;

        case 4:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001010011000; //  Output the number 4
                delay(1000);
                break;

        case 5:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
           PORT_REGS->GROUP[0].PORT_OUTSET =
```

```
0b00000000000000000000001010110100; //  Output the number 5
                delay(1000);
                break;

        case 6:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001011110100; //  Output the number 6
                delay(1000);
                break;

        case 7:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000000011100; //  Output the number 7
                delay(1000);
                break;

        case 8:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001011111100; //  Output the number 8
                delay(1000);
                break;

        case 9:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
```

```
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001010111100; //  Output the number 9
                  delay(1000);
                  break;

         default:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000011111100; //  Output the number 0
                  delay(1000);
         }
         }
         return;
}

//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000000011111100; //
Output the number 0
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000000000011000; //
Output the number 1
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001001101100; //
Output the number 2
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001000111100; //
Output the number 3
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001010011000; //
Output the number 4
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001010110100; //
Output the number 5
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001011110100; //
Output the number 6
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000000000011100; //
Output the number 7
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001011111100; //
Output the number 8
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001010111100; //
Output the number 9
//PORT_REGS->GROUP[0].PORT_OUTCLR = 0b00000000000000000000001011111100;
//clear first display
//PORT_REGS->GROUP[0].PORT_OUTCLR = 0b00000000000000000000001011111100;
//clear second display
```

```
void delay(unsigned int m)
{
        unsigned int i;
        for(i=0; i<=m; i++)
           {
        }
}
```

Figure 3.11: Code for the Complete Counter Project

```
/*
 * File:   newmain.c
 * Author: ngguaste
 *
 * Created on September 15, 2021, 4:58 PM
 */


#include <xc.h>

void main(void)
{
        PORT_REGS->GROUP[0].PORT_DIRSET =
0b00000000000000000000101111111100;  // sets pins to an output 00 000 000 000 000 000
000 111 111 101 100
        PORT_REGS->GROUP[0].PORT_DIRCLR =
0b00000000000000001000000000000000;  // Pushbutton  for increment set as input on PA16,
physical pin 17
        PORT_REGS->GROUP[0].PORT_DIRCLR =
0b00000000000001000000000000000000;  // Pushbutton  for direction set as input on PA18,
physical pin 19
        PORT_REGS->GROUP[0].PORT_DIRCLR =
0b00000000010000000000000000000000;  // Pushbutton  for reset          set as input on
PA22, physical pin 21
        PORT_REGS->GROUP[0].PORT_PINCFG[16] = 0b00000010;                    //
Input enable for PA16, physical pin 17
        PORT_REGS->GROUP[0].PORT_PINCFG[18] = 0b00000010;                    //
Input enable for PA18, physical pin 19
        PORT_REGS->GROUP[0].PORT_PINCFG[22] = 0b00000010;                    //
Input enable for PA22, physical pin 21
```

```
	int tenCount = 0;
	int oneCount = 0;
	int flagInc = 1;
	int flagDirection = 1;
	int flagReset = 1;
	int direction = 0; //0 for forward, 1 for reverse

	while(1)
	{
	//Code for changing direction
	if((PORT_REGS->GROUP[0].PORT_IN &
0b00000000000001000000000000000000) == 0b00000000000001000000000000000000)
	{
	if(flagDirection == 1)
	{
		if(direction == 1)
		{
		direction = 0;
		flagDirection = 0;
		}
		else if(direction == 0)
		{
		direction = 1;
		flagDirection = 0;
		}
	}
	}
	else
	{
	flagDirection = 1;
	}

	//Code for reset button
	if((PORT_REGS->GROUP[0].PORT_IN &
0b00000000010000000000000000000000) == 0b00000000010000000000000000000000)
	{
	if(flagReset == 1)
	{
		tenCount = 0;
		oneCount = 0;
		flagReset = 0;
	}
	}
	else
	{
	flagReset = 1;
```

```
        }

        //Code for increment or decrementing the count
        if ((PORT_REGS->GROUP[0].PORT_IN &
0b00000000000000010000000000000000) == 0b00000000000000010000000000000000)
        {
        if (flagInc == 1)
        {
        if(direction == 0) // forward/counting up
        {
                if (oneCount < 9)
                {
                oneCount = oneCount + 1;
                flagInc = 0;
                }
                else if ((tenCount == 9)&&(oneCount == 9))
                {
                tenCount = 0;
                oneCount = 0;
                flagInc = 0;
                }
                else if (oneCount == 9)
                {
                oneCount = 0;
                tenCount = tenCount + 1;
                flagInc = 0;
                }
        }
        else if(direction == 1) // backwards/counting down
        {
                if (oneCount > 0)
                {
                oneCount = oneCount - 1;
                flagInc = 0;
                }
                else if ((tenCount == 0)&&(oneCount == 0))
                {
                tenCount = 9;
                oneCount = 9;
                flagInc = 0;
                }
                else if (oneCount == 0)
                {
                oneCount = 9;
                tenCount = tenCount - 1;
                flagInc = 0;
```

```
                }
        }
        }
        }
        else
        {
        flagInc = 1;
        }

        //Switch case for the ones digit (second display)
        switch(oneCount)
        {
        case 0:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000011111100; //  Output the number 0
                delay(1000);
                break;

        case 1:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000000011000; //  Output the number 1
                delay(1000);
                break;

        case 2:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001001101100; //  Output the number 2
                delay(1000);
```

```
                break;

        case 3:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001000111100; //  Output the number 3
                delay(1000);
                break;

        case 4:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001010011000; //  Output the number 4
                delay(1000);
                break;

        case 5:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001010110100; //  Output the number 5
                delay(1000);
                break;

        case 6:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
           PORT_REGS->GROUP[0].PORT_OUTSET =
```

```
0b00000000000000000000001011110100; //  Output the number 6
                delay(1000);
                break;

        case 7:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000000011100; //  Output the number 7
                delay(1000);
                break;

        case 8:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001011111100; //  Output the number 8
                delay(1000);
                break;

        case 9:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001010111100; //  Output the number 9
                delay(1000);
                break;
        }

        //Switch case for the tens digit (first display)
        switch(tenCount)
        {
        case 0:
           PORT_REGS->GROUP[0].PORT_OUTSET =
```

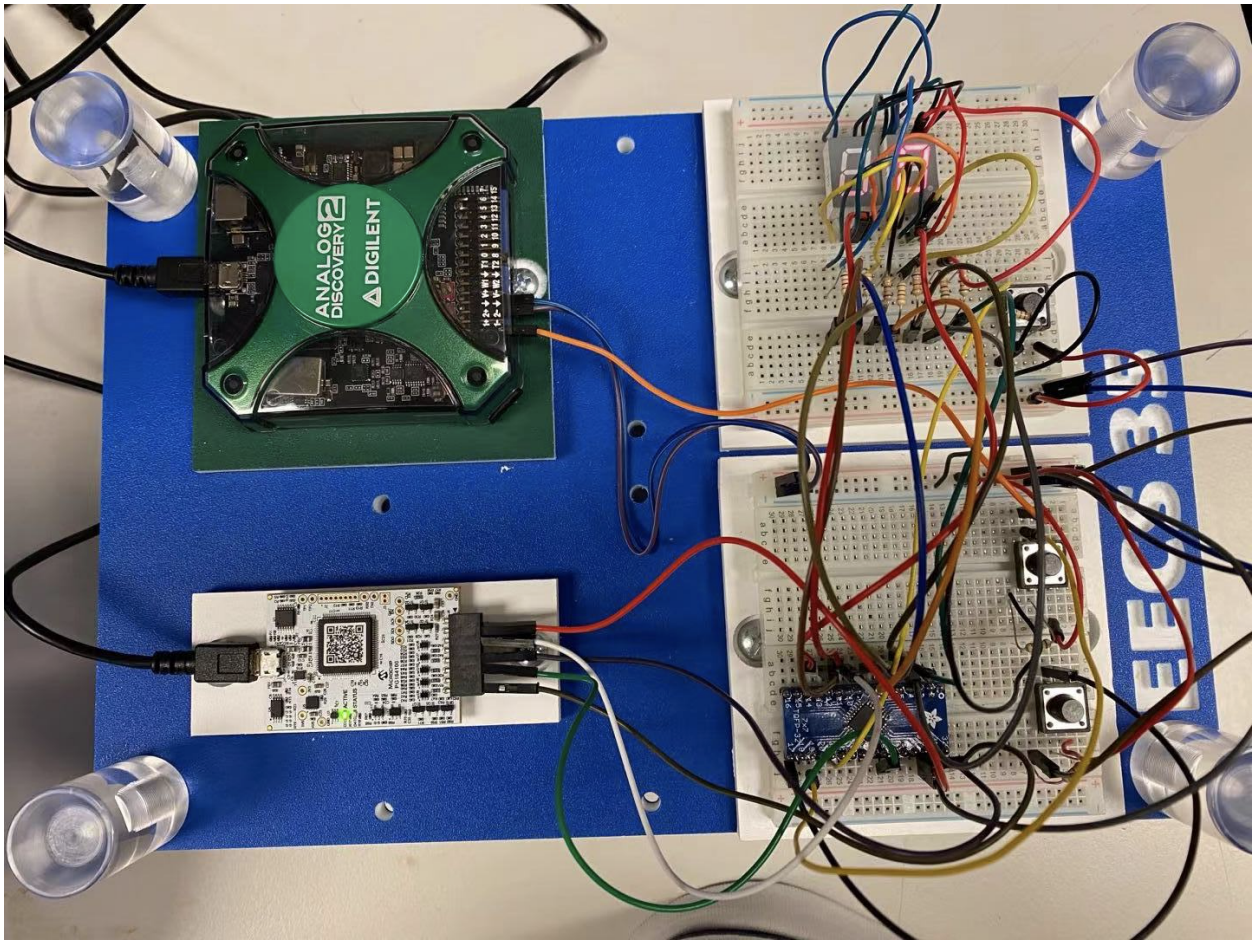
```
0b00000000000000000000100000000000; //turn off second display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000011111100; //  Output the number 0
                    delay(1000);
                    break;

          case 1:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000000011000; //  Output the number 1
                    delay(1000);
                    break;

          case 2:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001001101100; //  Output the number 2
                    delay(1000);
                    break;

          case 3:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001000111100; //  Output the number 3
                    delay(1000);
                    break;
```

```
        case 4:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001010011000; //  Output the number 4
                delay(1000);
                break;

        case 5:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001010110100; //  Output the number 5
                delay(1000);
                break;

        case 6:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001011110100; //  Output the number 6
                delay(1000);
                break;

        case 7:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000000011100; //  Output the number 7
                delay(1000);
```

```
                break;

        case 8:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001011111100; //  Output the number 8
                delay(1000);
                break;

        case 9:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001010111100; //  Output the number 9
                delay(1000);
                break;

        default:
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
           PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
           PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000011111100; //  Output the number 0
                delay(1000);
        }
        }
        return;
}

//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000000011111100; //
Output the number 0
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000000000011000; //
Output the number 1
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001001101100; //
```

```
Output the number 2
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001000111100; //
Output the number 3
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001010011000; //
Output the number 4
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001010110100; //
Output the number 5
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001011110100; //
Output the number 6
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000000000011100; //
Output the number 7
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001011111100; //
Output the number 8
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001010111100; //
Output the number 9
//PORT_REGS->GROUP[0].PORT_OUTCLR = 0b00000000000000000000001011111100;
//clear first display
//PORT_REGS->GROUP[0].PORT_OUTCLR = 0b00000000000000000000001011111100;
//clear second display

void delay(unsigned int m)
{
        unsigned int i;
        for(i=0; i<=m; i++)
          {
        }
}
```

Figure 3.12: Circuit for the Complete Counter Project


ANALOG DISCOVERY 2
DIGILENT

```
304   //PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001011110100; //  Output
305   //PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000000000011100; //  Output
306   //PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001011111100; //  Output
307   //PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001010111100; //  Output
308   //PORT_REGS->GROUP[0].PORT_OUTCLR = 0b00000000000000000000001011111100; //clear firs
309   //PORT_REGS->GROUP[0].PORT_OUTCLR = 0b00000000000000000000001011111100; //clear seco
310
311   void micros(unsigned int x)
312   {
313       TC5_REGS -> COUNT16.TC_COUNT = 0;
314       TC5_REGS -> COUNT16.TC_INTFLAG = 0b00010000;
315       TC5_REGS -> COUNT16.TC_CC[0] = x;
316       while ((TC5_REGS ->COUNT16.TC_INTFLAG & 0b00010000) == 0b00000000);
317   }
318   void setUP (void)
319   {
320       PM_REGS -> PM_APBCMASK |= 0b00000000000000000010000000000000;
321       GCLK_REGS -> GCLK_CLKCTRL = 0b0100000000011100;
322       TC5_REGS -> COUNT16.TC_CTRLA = 0b0000000000100000;
323       TC5_REGS -> COUNT16.TC_CC[0] = 1000;
324       TC5_REGS -> COUNT16.TC_CTRLA |= 0b0000000000000010;
325   }

Output × Notifications
Project Loading Warning × Snap-Experiment 3 × Experiment_3 (Build, Load, ...) ×
WARNING; Programming configuration values on this device can cause unintended consequences if all of the conf
Erasing...
```

Figure 4.1: Our micros(x) and setUP() functions

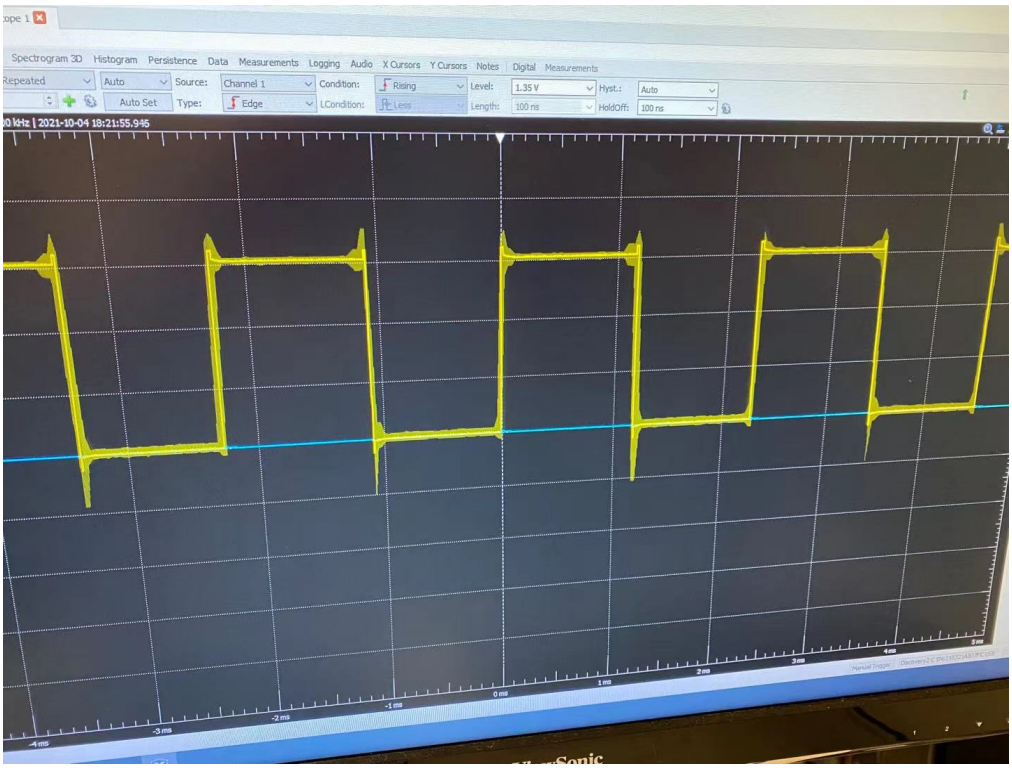
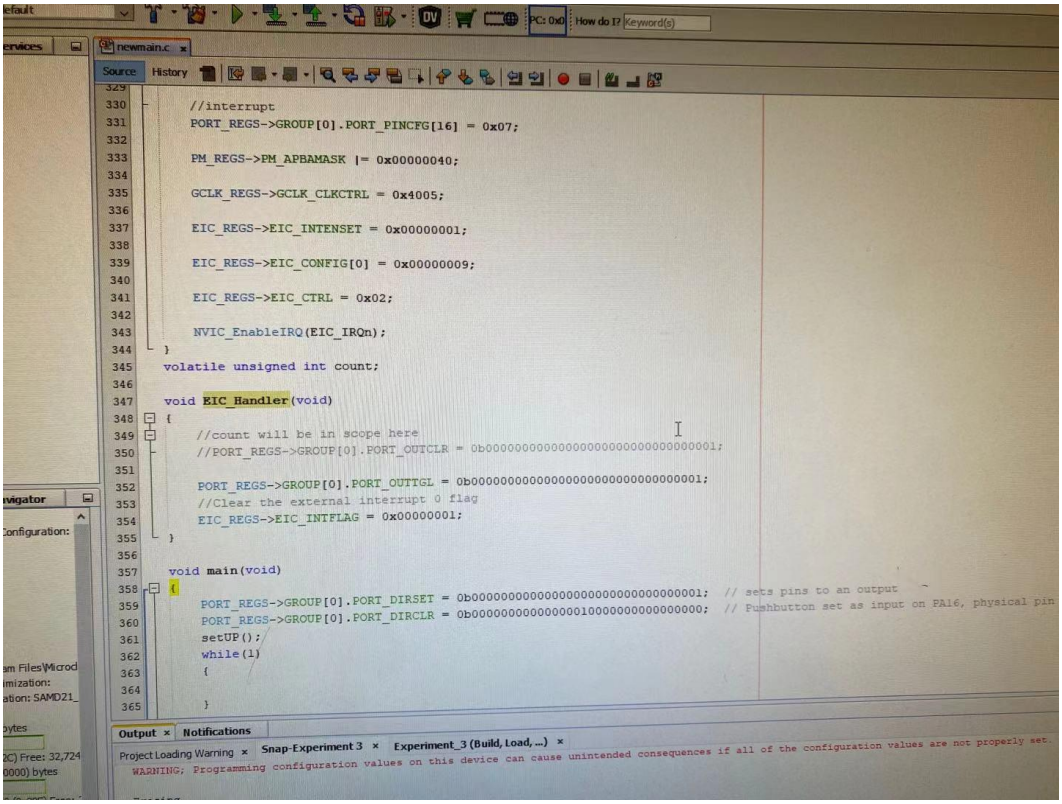


Figure 4.2: Our GPIO output while testing the hardware delay



```
//interrupt
PORT_REGS->GROUP[0].PORT_PINCFG[16] = 0x07;

PM_REGS->PM_APBAMASK |= 0x00000040;

GCLK_REGS->GCLK_CLKCTRL = 0x4005;

EIC_REGS->EIC_INTENSET = 0x00000001;

EIC_REGS->EIC_CONFIG[0] = 0x00000009;

EIC_REGS->EIC_CTRL = 0x02;

NVIC_EnableIRQ(EIC_IRQn);
}
volatile unsigned int count;

void EIC_Handler(void)
{
    //count will be in scope here
    //PORT_REGS->GROUP[0].PORT_OUTCLR = 0b00000000000000000000000000000001;

    PORT_REGS->GROUP[0].PORT_OUTTGL = 0b00000000000000000000000000000001;
    //Clear the external interrupt 0 flag
    EIC_REGS->EIC_INTFLAG = 0x00000001;
}

void main(void)
{
    PORT_REGS->GROUP[0].PORT_DIRSET = 0b00000000000000000000000000000001;  // sets pins to an output
    PORT_REGS->GROUP[0].PORT_DIRCLR = 0b00000000000000001000000000000000;  // Pushbutton set as input on PA16, physical pin
    setUP();
    while(1)
    {

    }
```

Figure 4.3: Our setUP() and EIC_Handler() for our interrupt

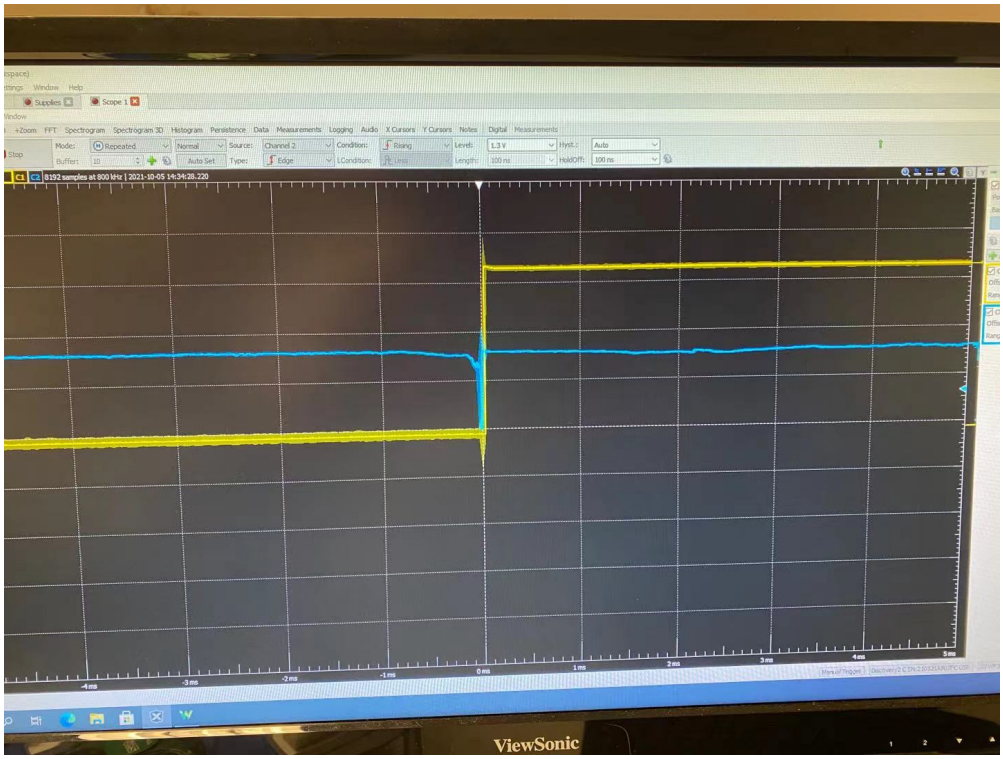


Figure 4.4 Our interrupt triggering on a rising edge, our button press

```
#include <xc.h>

volatile int oneCount = 0;
volatile int tenCount = 0;
volatile int direction = 0;
void main(void)
{
   PORT_REGS->GROUP[0].PORT_DIRSET = 0b00000000000000000000101111111100;  // sets pins to an output 00 000 000 000 000 000 000 111 111 101 100
   //PORT_REGS->GROUP[0].PORT_DIRCLR = 0b00000000000000001000000000000000; // Pushbutton  for increment set as input on PA16, physical pin 17
   PORT_REGS->GROUP[0].PORT_DIRCLR = 0b00000000000001000000000000000000; // Pushbutton  for direction set as input on PA18, physical pin 19
   PORT_REGS->GROUP[0].PORT_DIRCLR = 0b00000000010000000000000000000000; // Pushbutton  for reset     set as input on PA22, physical pin 21
   //PORT_REGS->GROUP[0].PORT_PINCFG[16] = 0b00000010;                    // Input enable for PA16, physical pin 17
   PORT_REGS->GROUP[0].PORT_PINCFG[18] = 0b00000010;                    // Input enable for PA18, physical pin 19
   PORT_REGS->GROUP[0].PORT_PINCFG[22] = 0b00000010;                    // Input enable for PA22, physical pin 21

   setUP();
   //int tenCount = 0;
   //int oneCount = 0;
   //int flagInc = 1;
   int flagDirection = 1;
   int flagReset = 1;
   //int direction = 0; //0 for forward, 1 for reverse

   while(1)
   {

      //Code for changing direction
      if((PORT_REGS->GROUP[0].PORT_IN & 0b00000000000001000000000000000000) == 0b00000000000001000000000000000000)
      {
         if(flagDirection == 1)
         {
            if(direction == 1)
            {
               direction = 0;
               flagDirection = 0;
            }
```

```
            else if(direction == 0)
            {
               direction = 1;
               flagDirection = 0;
            }
         }
      }
      else
      {
         flagDirection = 1;
      }

      //Code for reset button
      if((PORT_REGS->GROUP[0].PORT_IN & 0b00000000010000000000000000000000)
== 0b00000000010000000000000000000000)
      {
         if(flagReset == 1)
         {
               tenCount = 0;
               oneCount = 0;
               flagReset = 0;
         }
      }
      else
      {
         flagReset = 1;
      }

      /*
      //Code for increment or decrementing the count
      if ((PORT_REGS->GROUP[0].PORT_IN & 0b00000000000000010000000000000000)
== 0b00000000000000010000000000000000)
      {
        if (flagInc == 1)
        {
           if(direction == 0) // forward/counting up
           {
              if (oneCount < 9)
              {
                 oneCount = oneCount + 1;
                 flagInc = 0;
              }
              else if ((tenCount == 9)&&(oneCount == 9))
              {
                 tenCount = 0;
                 oneCount = 0;
```

```
                flagInc = 0;
            }
            else if (oneCount == 9)
            {
                oneCount = 0;
                tenCount = tenCount + 1;
                flagInc = 0;
            }
        }
        else if(direction == 1) // backwards/counting down
        {
            if (oneCount > 0)
            {
                oneCount = oneCount - 1;
                flagInc = 0;
            }
            else if ((tenCount == 0)&&(oneCount == 0))
            {
                tenCount = 9;
                oneCount = 9;
                flagInc = 0;
            }
            else if (oneCount == 0)
            {
                oneCount = 9;
                tenCount = tenCount - 1;
                flagInc = 0;
            }
        }
      }
    }
    else
    {
       flagInc = 1;
    }
    */

    //Switch case for the ones digit (second display)
    switch(oneCount)
    {
       case 0:
          PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
          PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
          PORT_REGS->GROUP[0].PORT_OUTCLR =
```

```
0b00000000000000000000100000000000; //turn on second display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000011111100; //  Output the number 0
            micros(1000);
            break;

         case 1:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000000011000; //  Output the number 1
            micros(1000);
            break;

         case 2:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001001101100; //  Output the number 2
            micros(1000);
            break;

         case 3:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001000111100; //  Output the number 3
            micros(1000);
            break;

         case 4:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
```

```
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001010011000; //  Output the number 4
            micros(1000);
            break;

         case 5:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001010110100; //  Output the number 5
            micros(1000);
            break;

         case 6:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001011110100; //  Output the number 6
            micros(1000);
            break;

         case 7:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000000011100; //  Output the number 7
            micros(1000);
            break;

         case 8:
            PORT_REGS->GROUP[0].PORT_OUTSET =
```

```
0b00000000000000000000000100000000; //turn off first display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001011111100; //  Output the number 8
            micros(1000);
            break;

         case 9:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000100000000; //turn off first display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000100000000000; //turn on second display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001010111100; //  Output the number 9
            micros(1000);
            break;
      }

      //Switch case for the tens digit (first display)
      switch(tenCount)
      {
         case 0:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000011111100; //  Output the number 0
            micros(1000);
            break;

         case 1:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
```

```
0b00000000000000000000000000011000; //  Output the number 1
            micros(1000);
            break;

        case 2:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001001101100; //  Output the number 2
            micros(1000);
            break;

        case 3:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001000111100; //  Output the number 3
            micros(1000);
            break;

        case 4:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001010011000; //  Output the number 4
            micros(1000);
            break;

        case 5:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
```

```
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001010110100; //  Output the number 5
            micros(1000);
            break;

         case 6:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001011110100; //  Output the number 6
            micros(1000);
            break;

         case 7:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000000011100; //  Output the number 7
            micros(1000);
            break;

         case 8:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001011111100; //  Output the number 8
            micros(1000);
            break;

         case 9:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
```

```
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000001010111100; //  Output the number 9
            micros(1000);
            break;

          default:
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000100000000000; //turn off second display
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000001011111100; //global clear
            PORT_REGS->GROUP[0].PORT_OUTCLR =
0b00000000000000000000000100000000; //turns on first display
            PORT_REGS->GROUP[0].PORT_OUTSET =
0b00000000000000000000000011111100; //  Output the number 0
            micros(1000);
        }
    }
    return;
}

//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000000011111100; //
Output the number 0
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000000000011000; //
Output the number 1
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001001101100; //
Output the number 2
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001000111100; //
Output the number 3
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001010011000; //
Output the number 4
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001010110100; //
Output the number 5
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001011110100; //
Output the number 6
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000000000011100; //
Output the number 7
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001011111100; //
Output the number 8
//PORT_REGS->GROUP[0].PORT_OUTSET = 0b00000000000000000000001010111100; //
Output the number 9
//PORT_REGS->GROUP[0].PORT_OUTCLR = 0b00000000000000000000001011111100;
//clear first display
//PORT_REGS->GROUP[0].PORT_OUTCLR = 0b00000000000000000000001011111100;
```

```
//clear second display

void micros(unsigned int x)
{
   TC5_REGS -> COUNT16.TC_COUNT = 0;
   TC5_REGS -> COUNT16.TC_INTFLAG = 0b00010000;
   TC5_REGS -> COUNT16.TC_CC[0] = x;
   while ((TC5_REGS ->COUNT16.TC_INTFLAG & 0b00010000) == 0b00000000);
}

void setUP (void)
{
   //hardware delay
   PM_REGS->PM_APBCMASK |= 0b00000000000000000010000000000000;
   GCLK_REGS->GCLK_CLKCTRL = 0b0100000000011100;
   TC5_REGS->COUNT16.TC_CTRLA = 0b0000000000100000;
   TC5_REGS->COUNT16.TC_CC[0] = 1000;
   TC5_REGS->COUNT16.TC_CTRLA |= 0b0000000000000010;

   //interrupt
   PORT_REGS->GROUP[0].PORT_PINCFG[16] = 0x07;
   PM_REGS->PM_APBAMASK |= 0x00000040;
   GCLK_REGS->GCLK_CLKCTRL = 0x4005;
   EIC_REGS->EIC_INTENSET = 0x00000001;
   EIC_REGS->EIC_CONFIG[0] = 0x00000009;
   EIC_REGS->EIC_CTRL = 0x02;
   NVIC_EnableIRQ(EIC_IRQn);
}

void EIC_Handler(void)
{
  //PORT_REGS->GROUP[0].PORT_OUTTGL = 0b00000000000000000000000000000001;

  if(direction == 0) // forward/counting up
  {
     if (oneCount < 9)
     {
        oneCount = oneCount + 1;
     }
     else if ((tenCount == 9)&&(oneCount == 9))
     {
        tenCount = 0;
        oneCount = 0;
     }
     else if (oneCount == 9)
     {
```

```
        oneCount = 0;
        tenCount = tenCount + 1;
      }
    }
    else if(direction == 1) // backwards/counting down
    {
      if (oneCount > 0)
      {
        oneCount = oneCount - 1;
      }
      else if ((tenCount == 0)&&(oneCount == 0))
      {
        tenCount = 9;
        oneCount = 9;
      }
      else if (oneCount == 0)
      {
        oneCount = 9;
        tenCount = tenCount - 1;
      }
    }
     //Clear the external interrupt 0 flag
     EIC_REGS->EIC_INTFLAG = 0x00000001;
}
```

Figure 4.5: Our final code with