Lab 1
Pengzhao Zhu
Section: 112D

B) Prelab Questions

N/A. There are no prelab questions in this lab.

C) Problems Encountered

The first problem I encountered was that I didn't realize how to create a 10ms delay subroutine that does nothing other than delay 10ms. Finally, I realized that the simplest thing I could do to delay a program is to load a register with a certain value. I will then decrement that value to take up processor running time.

The second problem I encountered was that I didn't know how to mask away the other bits on the tactile switch port. I had to look closely at Schwartz's hint in the lab document before deciding to use the T-flag like I did in Lab 1.

D) Future Application

By completing this lab, I have gained the skills to control the GPIO pins on a microcontroller board. In the future, I could use the skills I have gained in this lab to connect and communicate with other hardware extensions. For example, I could possibly use GPIO pins to communicate with a speaker or robot the next time around.

This lab also allowed me to learn the use of delay functions. This is a concept that I could possibly use in high-frequency circuits later in my career.

E) Schematics

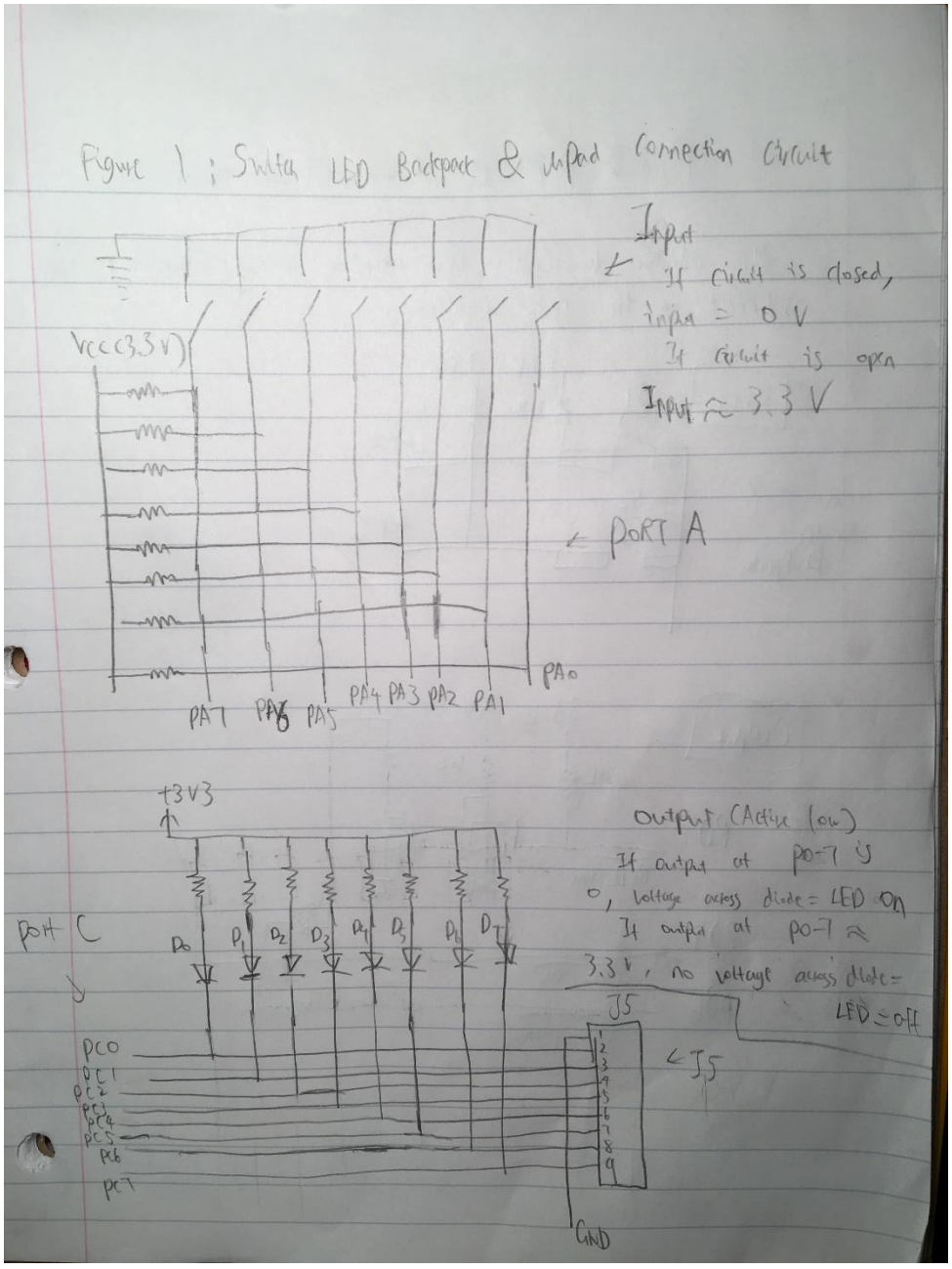

Figure 1: Switch LED Backpack & uPad Connection Circuit (Used in Part A,B,C,D)

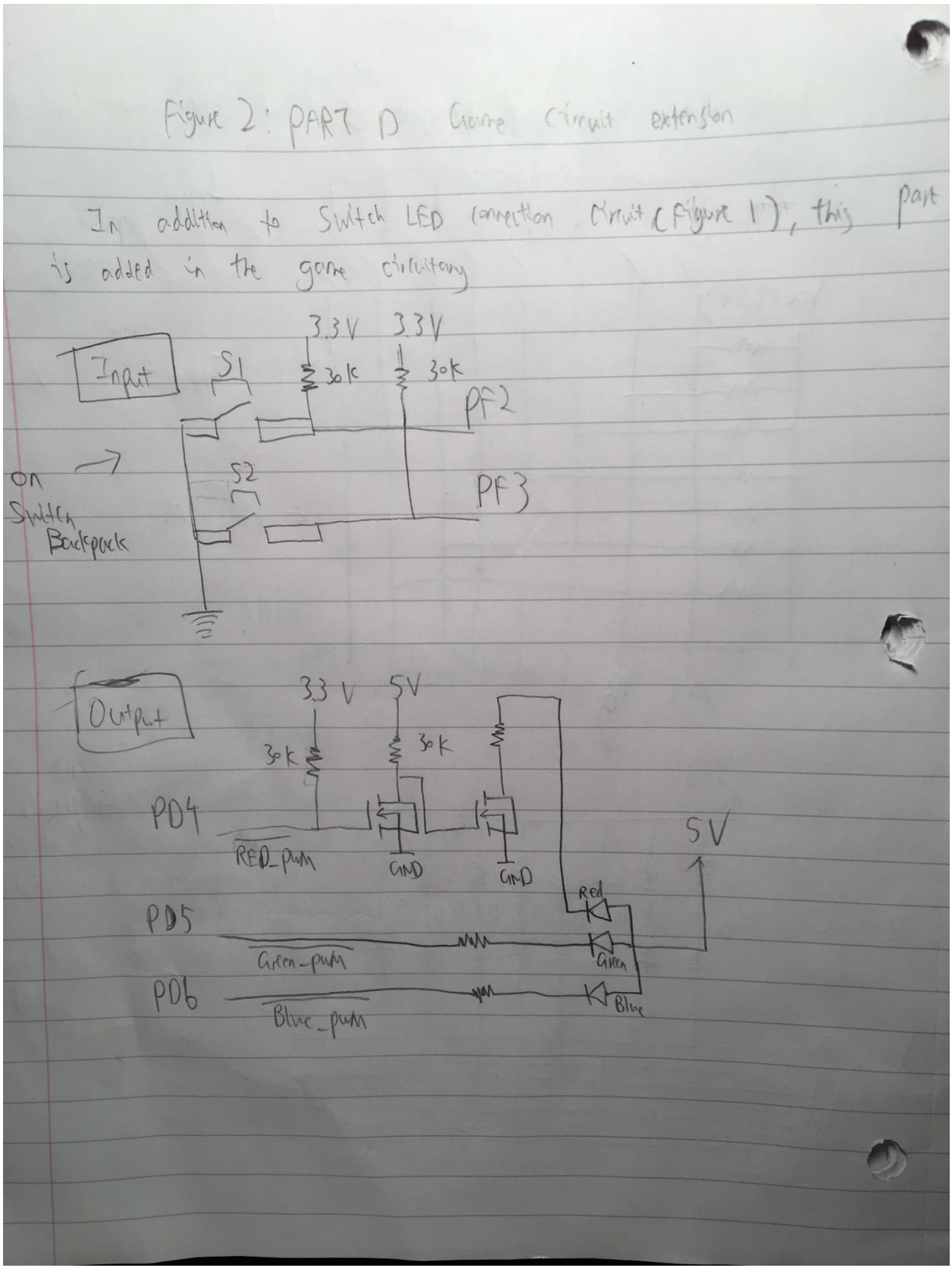

Figure 2: Part D Game Circuitry Extension

F) Pseudocode/Flowcharts

Part B Pseudocode:

Initialize PORTA switches to be input

Initialize PORTC LED to be output

LOOP:

Load value at PORTA_IN to Register 16

Transfer data at Register 16 to PORTC_OUT

Jump back to LOOP for infinite loop

Part C Pseudocode

Initialize stack pointer address at 0x3FFF

Load stack pointer address at Y register

Output Y register value to Stack Pointer register

Initialize the last LED at PORTC as output

LOOP ;to do the 10ms delay or multiples of 10ms delay

OUTTGL last LED at PORT C

Call Delay_10ms or Delay_100ms subroutine

Back to LOOP (Delay_10ms or Delay_100ms) ;the main code basically stop here.

; the rest are just subroutine

Delay_100ms ;this subroutine multiplies delay_10ms by 10

Push r16

Load register 16 with 10

Call delay10ms subroutine

Decrement r16

Check if r16=0

If not, back to Delay_100ms

Else {

pop r16

ret ;return to main routine

}

Delay_10ms ;delay 10 subroutine

push r16

;code start

1. use two for loops to take up running time.
2. Basically just load one register with a value
3. Decrement that register a Set the values to put in the register so it is exactly 10ms delay
4. and keep running to take up running time

;code ends

pop r16

ret ;return to main routine

Part D Pseudocode

Initialize Stack Pointer

Load Stack Pointer address into Y register

Output Y register value to Stack Pointer register

Load a set of 8 data values that will determine which LED to output in program memory

Initialize PORTC LED as output

Initialize S2 and S1 as input

Initialize Green and Red LED(Port D) as output

MAIN

while (S2 is not pressed) { ;while loop to continue game is green or red is not on

Shift LED towards middle and then outward with delay of 100 ms. Check after every delay to see if S2 is pressed. One LED goes 0,1,2,3,4,5,6,7,1,2.etc…… Another one goes 7,6,5,4,3,2,1,0,1,2,3,4,5.etc

}

; after the while loop ends (if somebody have pressed S2), then:……..

if ((S2 = true) AND (LED 3,4 not on)) {

Turn on Red LED

rjmp to RESET

}

else If ((S2=true) AND (LED 3,4 on))

Turn on Green LED

rjmp RESET

}

RESET

Read input from S1 to reset game

Back to RESET if red and green LED is still on. ;to make sure user reset game.

If S1 is press. Next line of code

Back to MAIN

G) Program Code

Part B Code

```
/* Lab 1 Part B
   Name: Pengzhao Zhu
   Section#: 112D
   TA Name: Chris Crary
   Description: This Program turns the 8 LED on/off by reading the data at the switch
*/

.include "ATxmega128A1Udef.inc"        ;include the file
.list                                  ;list it

.org 0x0000                            ;start our program here
rjmp MAIN                              ;jump to main

.dseg                                   ;data segment. not really needed

.equ set1=0xFF                  ;set all for output.used later

.cseg                           ;code segment

.org 0x200           ;where we will start the program

MAIN:
ldi r16, set1         ;load inputs(0xFF) to r16
sts PORTA_DIRCLR, r16    ;set Port A to be input

ldi r17, set1       ;load outputs (0xFF) to r17
sts PORTC_DIRSET, r17    ;set Port C to be output

sts PORTC_OUTSET, r16       ;turn off all LED (active low LED)

LOOP:
lds r16, PORTA_IN        ;load value at input to r16. switch=0n, closed circuit. Port A
grounded.
sts PORTC_OUT, r16        ;input to output.  0 to output. active low output. LED on
rjmp LOOP                 ;infinite loop
```

Part C Code (Delay 10ms)

```
/* Lab 1 Part C
   Name: Pengzhao Zhu
   Section#: 112D
   TA Name: Chris Crary
   Description: This program toggles a LED with a 10ms delay between toggles.
*/



.include "ATxmega128A1Udef.inc"      ;include the file
.list                                ;list it
```

```
.org 0x00                           ;start the program here
rjmp MAIN                           ; jump to main

.equ stack_init=0x3FFF   ;initialize stack pointer

MAIN:
ldi YL, low(stack_init)    ;Load 0xFF to YL
out CPU_SPL, YL                     ;transfer to CPU_SPL
ldi YL, high(stack_init)   ;Load 0x3F to YH
out CPU_SPH, YL                     ;transfer to CPU_SPH

ldi r16, 0x80  ;set last LED as output
sts PORTC_DIRSET, r16    ;set last LED as output using DIRSET

LOOP:
ldi r17, 0x80            ;load r17 with 0x80
sts PORTC_OUTTGL, r17    ;toggle last LED of PORTC
rcall Delay_10ms         ;call delay 10ms subroutine
rjmp LOOP                ;infinite loop

Delay_10ms:      ;delay 10ms subroutine
push r16         ;push r16
push r17         ;push r17
ldi r17, 15      ;do this loop 15 times. Just need a large number to make sure the delay
is long enough

START:
ldi r16, 0xFF    ;some value to take up running time

HI:
cpi r16,0        ;compare to 0
breq SECOND      ;go to second loop if loop one is done
dec r16          ;dec 16
rjmp HI          ;jump to HI if first loop is not done

SECOND:
cpi r17, 0       ;compare r17 too 0
breq rdone       ;if r17=0, we are finished with the subroutine and ready to return to
main code
dec r17          ;dec r17
rjmp START       ;start loop one

rdone:
pop r17           ;pop r17. restore it
pop r16           ;pop r16. restore it
ret               ; return to main routine
```

Part C Code(Delay 100ms, X=10)

```
/* Lab 1 Part C
   Name: Pengzhao Zhu
   Section#: 112D
   TA Name: Chris Crary
   Description: This program toggles a LED with a 100ms delay between toggles.
*/
```

```
.include "ATxmega128A1Udef.inc"     ;include the file
.list                               ;list it

.org 0x00                           ;start the program here
rjmp MAIN                           ;jump to MAIN

.equ stack_init=0x3FFF   ;initialize stack pointer

MAIN:
ldi YL, low(stack_init)    ;Load 0xFF to YL
out CPU_SPL, YL                     ;transfer to CPU_SPL
ldi YL, high(stack_init)   ;Load 0x3F to YH
out CPU_SPH, YL                     ;transfer to CPU_SPH

ldi r16, 0x80  ;set last LED as output
sts PORTC_DIRSET, r16    ;set last LED as output using DIRSET

ldi r21, 10   ;initialize how many times i want to mulitply delay_10ms for

LOOP:
ldi r17, 0x80            ;load r17 with 0x80
sts PORTC_OUTTGL, r17    ;toggle last LED of PORTC

rcall Delay_mult        ;call delay 100ms subroutine
rjmp LOOP                ;infinite loop



Delay_mult:
push r20        ;push r20
mov r20, r21      ;load 9 in r20. it runs it 10 times and the delay will be 10ms.

REPEAT:
rcall Delay_10ms    ;call delay_10ms
dec r20             ;decrement r20. keep the code running
cpi r20, 0          ;load 0 to r20
breq DONE           ;when it is done, prepare to get back into main routine
rjmp REPEAT         ;going back to the place to call delay_10ms again

DONE:
pop r20             ;pop r20
ret                 ;return to main routine

Delay_10ms:      ;delay 10ms subroutine
push r16         ;push r16
push r17         ;push r17
ldi r17, 15      ;do this loop 15 times. Just need a large number to make sure the delay
is long enough

START:
ldi r16, 0xFF    ;some value to take up running time

HI:
cpi r16,0        ;compare to 0
breq SECOND      ;go to second loop if loop one is done
dec r16          ;dec 16
rjmp HI          ;jump to HI if first loop is not done
```

```
SECOND:
cpi r17, 0       ;compare r17 too 0
breq rdone       ;if r17=0, we are finished with the subroutine and ready to return to
main code
dec r17          ;dec r17
rjmp START       ;start loop one

rdone:            ;by the time the code gets back here.
pop r17           ;should be a 10ms delay
pop r16
ret               ; return to main routine
```

Part D Code

```
/* Lab 1 Part D
   Name: Pengzhao Zhu
   Section#: 112D
   TA Name: Chris Crary
   Description: This program is game that will turn on the green LED if user wins, or
turn on the red LED if user loses.
                          LEDs will move inward and outward before game the concludes
(win or lose). There is also an option to reset
                          the game after the game concludes
*/

.include "ATxmega128A1Udef.inc"       ;include the file
.list                                 ;list it

.org 0x00                             ;start the program here
rjmp MAIN                             ;jump to MAIN

.equ stack_init=0x3FFF   ;initialize stack pointer

.org 0x200    ;put table here
Table: .db 0b10000001, 0b01000010, 0b00100100, 0b00011000,0b00100100,
0b01000010,0b10000001,0b00000000  ;load table to turn on LED

.org 0x300    ;code start here

MAIN:
ldi YL, low(stack_init)    ;Load 0xFF to YL
out CPU_SPL, YL                       ;transfer to CPU_SPL
ldi YL, high(stack_init)   ;Load 0x3F to YH
out CPU_SPH, YL                       ;transfer to CPU_SPH

ldi r21, 10   ;initialize how many times i want to mulitply delay_10ms for

ldi r17, 0xFF   ;to turn off all active low LED later
ldi r18, 0b00110000 ;to turn off the red and green

ldi r22, 0b00001100     ;set S1 and S2 as input
sts PORTF_DIRCLR, r22   ;transfer to PORTF_DIRCLR
ldi r22, 0b00110000      ;bit 5 is red, bit 6 is green
sts PORTD_DIRSET, r22    ;set as input
sts PORTD_OUTSET, r18  ;to turn off the LED for now. set them as HIGH so they will be off
```

```
ldi r16, 0xFF            ;load r26 with 0xFF
sts PORTC_DIRSET, r16    ;set Port C to be output

LEDLOOP:

ldi ZL, low(Table << 1)        ;load low byte of table to ZL
ldi ZH, high(Table << 1)       ;load high byte of table to ZH
ldi r20, 8                     ;table counter=8

LEDSWITCH:
lpm r16, Z+                  ;load the first data at the table to r16. post increment
sts PORTC_OUTSET, r17        ;turn off all LED (active low LED)
sts PORTC_OUTCLR, r16        ;turn the LED I want on
rcall Delay_mult             ;call my delay

lds r23, PORTF_IN                        ;if PORTF_IN is pressed
bst r23, 3                               ;if pressed. store bit 3 of r23 into T-flag
brtc PRESS                               ;if pressed. voltage is 0, so that is why it
breaks if cleared


dec r20                                 ;decrement r20
cpi r20, 0                              ;if r20=0, start the table over
breq LEDLOOP                            ;start table over
rjmp LEDSWITCH                          ;load the next value in table


PRESS:

bst r16, 4    ;store bit 4 of r16 into T-flag
brts GREEN    ;if bit 4 is set, then I win the game
brtc RED      ;if bit 4 is not set, then I lose the game


GREEN:
ldi r23, 0b00100000    ;low true. load onto r23
sts PORTD_OUTCLR, r23  ;turn on GREEN
rjmp RESET             ;jump to where I will reset the game


RED:
ldi r23, 0b00010000       ;low true. load onto r23
sts PORTD_OUTCLR, r23     ;turn on RED
rjmp RESET               ;jump to where I will reset the game

RESET:
lds r23, PORTF_IN            ;check if user have pressed S1 to reset
bst r23, 2                   ;read where S1 switch is at. i.e. bit 2
brtc OFF                     ;if pressed. reset game
brts RESET                   ;if not pressed. infinite loop

OFF:
sts PORTD_OUTSET, r18  ;to turn off all active low LED
rjmp LEDLOOP           ;jump to LEDLOOP and start everything over

Delay_mult:
push r20        ;push r20
```

```
mov r20, r21      ;load 9 (r21) in r20. it runs it 10 times and the delay will be 100ms.

REPEAT:
rcall Delay_10ms    ;call delay_10ms
dec r20             ;decrement r20. keep the code running
cpi r20, 0          ;load 0 to r20
breq DONE           ;when it is done, prepare to get back into main routine
rjmp REPEAT         ;going back to the place to call delay_10ms again

DONE:
pop r20             ;pop r20
ret                 ;return to main routine

Delay_10ms:      ;delay 10ms subroutine
push r16         ;push r16
push r17         ;push r17
ldi r17, 15      ;do this loop 15 times. Just need a large number to make sure the delay
is long enough

START:
ldi r16, 0xFF    ;some value to take up running time

HI:
cpi r16,0        ;compare to 0
breq SECOND      ;go to second loop if loop one is done
dec r16          ;dec 16
rjmp HI          ;jump to HI if first loop is not done

SECOND:
cpi r17, 0       ;compare r17 too 0
breq rdone       ;if r17=0, we are finished with the subroutine and ready to return to
main code
dec r17          ;dec r17
rjmp START       ;start loop one

rdone:            ;by the time the code gets back here.
pop r17           ;should be a 10ms delay
pop r16
ret               ; return to main routine
```

H) Appendix

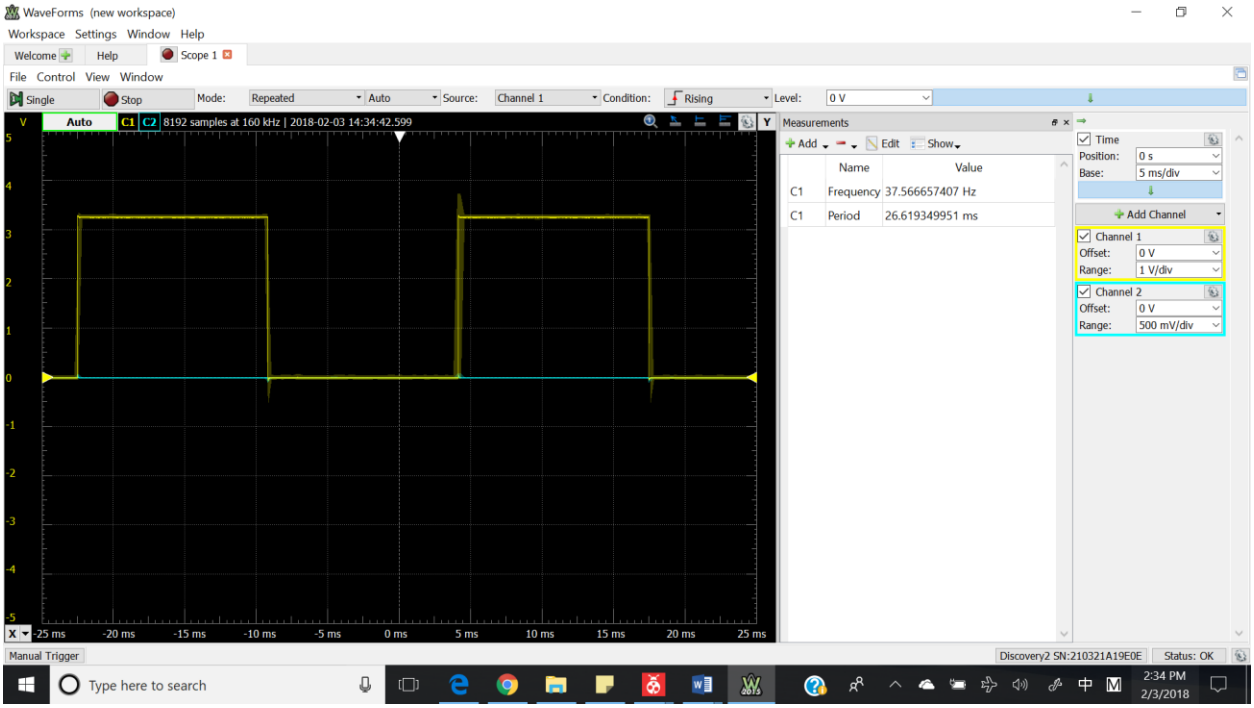

Figure 3: Delay_10ms First Attempt

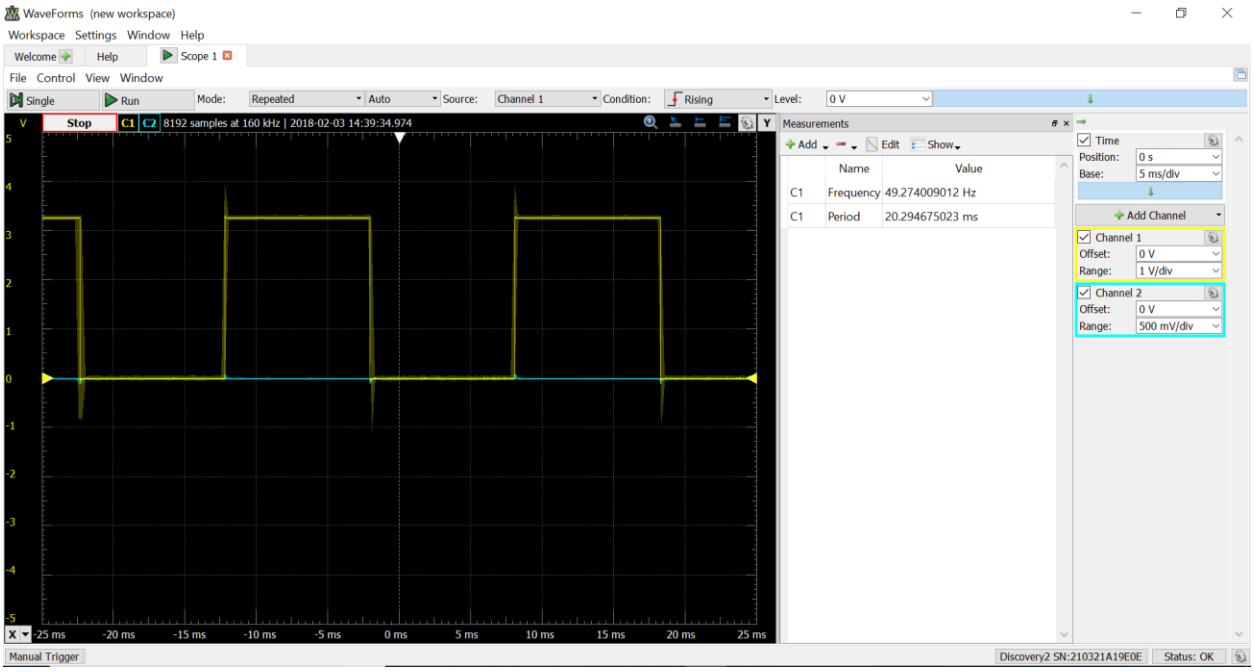

Figure 4: Delay_10ms Successful Attempt

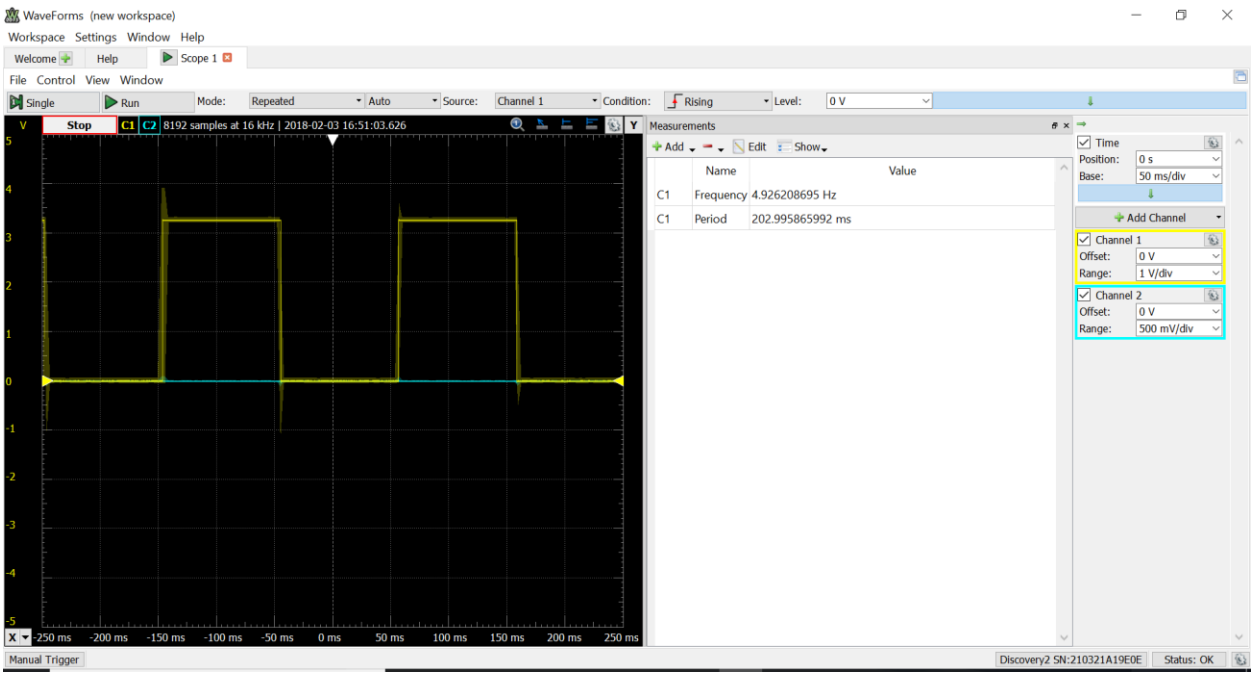

Figure 5: Delay 100ms (X=10 on the Delay10ms Subroutine) Successful Attempt (Part C and D)

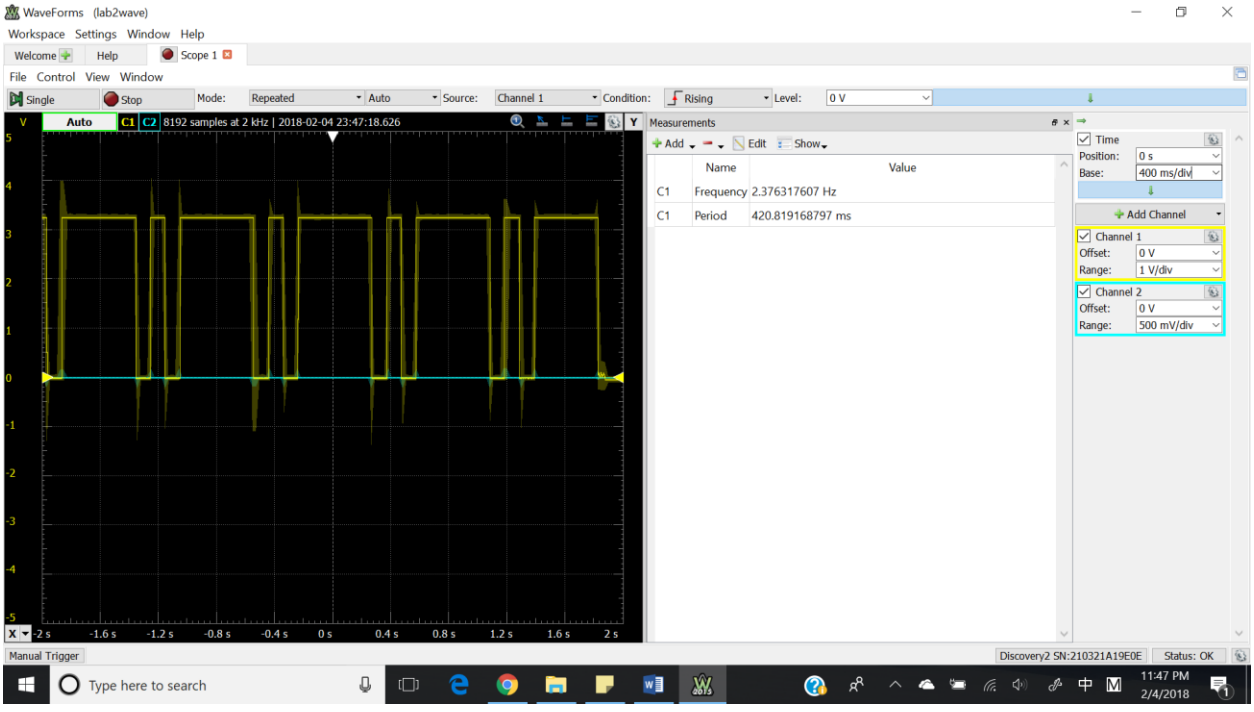

Figure 6: Part D Game Animation Pattern (Waveform of One Pin Only)

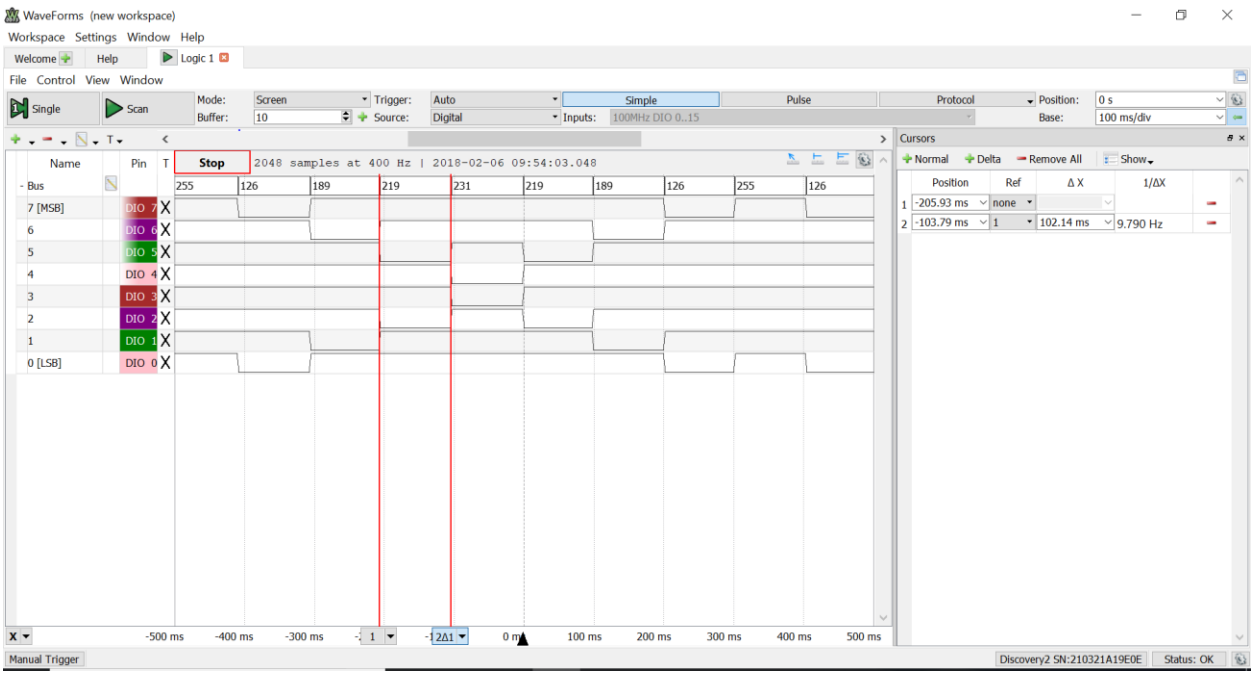

Figure 7: Game D Logic Analyzer Pattern (100ms delay)