Lab 2
Jineidy Mak
Secton 7F31

Pre-Lab Questions:

None.

Problems Encountered:

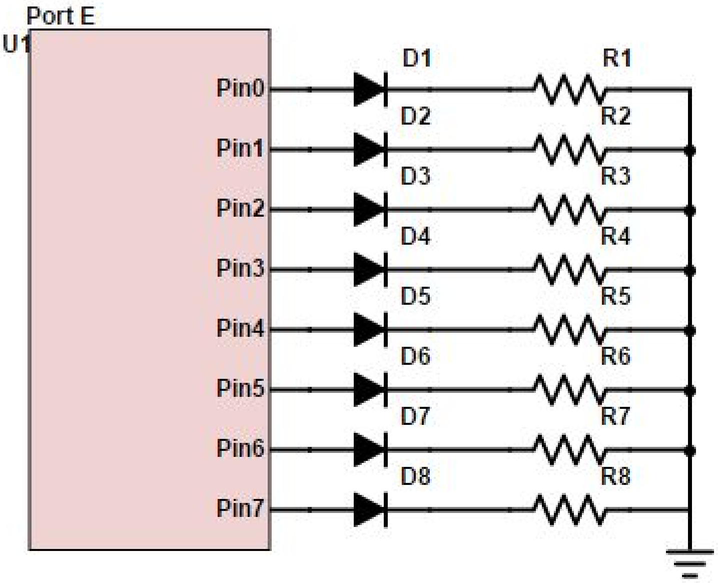
My Keypad breakout connector broke so I had to manually connect it. Could of used the lecture that goes over Portn_CTRLxPIN before the lab.

Future Work/Applications:

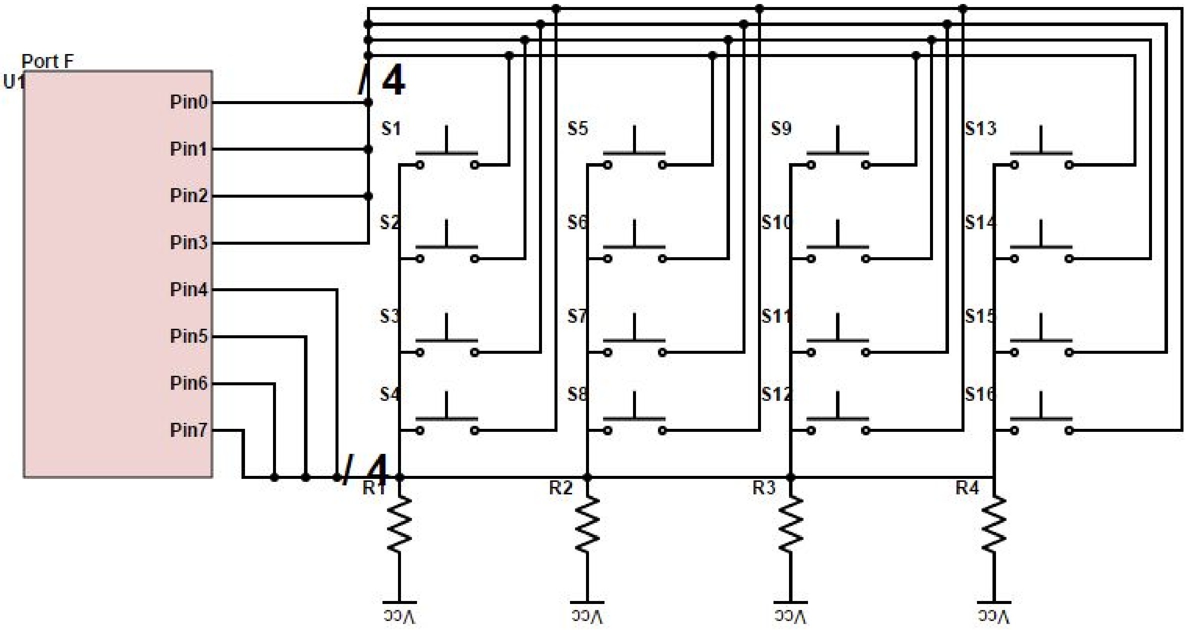
We could possibly output the inputs to a LCD. Also we could input signals from many different peripherals. We could expand a lab, perhaps this is the basis of how keypad door entries work for some rooms in NEB?

Schematics:

Part A: LEDs from PortE to Ground.



Part B: Keypad Wiring to Port F, with Pull-Up Resistors.



Decoding Logic:

Pseudocode/Flowcharts:

Part A:

```
Set PortE Pins to Output.
Output 0xFF to PortE Pins.
```

Part B:

```
Set PortE Pins to Output.
Output 0x55
Delay for 250uS
Output 0xAA
Delay for 250uS
```

Part C:

```
Set PortF Pins 3-0 to Output.
Set PortF Pins 7-4 to Input.
Col1 = 1000
Col2 = 0100
Col3 = 0010
Col4 = 0001.

Check for Input, Input is store as 8 bit number, Row is bits 3-0 and Col is bits 7-0. Store in X.
For(Row 1 to 4)

        if(X = “ColRow”)
                Output appropriate value that corresponds to the row and col value.

Repeat.
```

Part D:

```
Set PortE Pins 7-0 to Output.
Set PortF Pins 3-0 to Output.
Set PortF Pins 7-4 to Input.
Col1 = 1000
Col2 = 0100
Col3 = 0010
Col4 = 0001.

Check for Input, Input is store as 8 bit number, Row is bits 3-0 and Col is bits 7-0. Store in X.
for (Row1 to 4)
        if(X = “ColRow”)
                Output appropriate value that corresponds to PortE

Repeat.
```

Program Code:

Part A:

```
;
; lab2a.asm
;
; Created: 5/28/2016 10:17:41 PM
; Author : James Mak
;

.include "ATxmega128A1Udef.inc"

.org 0x0000

                    rjmp main

.org 0x0100

main:

            ldi R16, 0xFF                       ;Load R16 with FF.
            sts PORTE_DIRSET, R16          ;Set GPIO's in four bit PORTE as outputs.
            ldi R16, 0xFF                       ;Load R16 with 01.
            sts PORTE_OUT, R16                  ;Set the output of PortD pin 1.

end:
            rjmp end                            ;Endless loop.
```

Part B

```
;
; lab2b.asm
;
; Created: 5/29/2016 7:32:19 PM
; Author : James Mak
;

.include "ATxmega128A1Udef.inc"

.equ stack_init = 0x2FFF                 ;Initialize stack pointer.
.equ even = 0x55                              ;Even numbered bits on.
.equ odd = 0xAA                                   ;Odd numbered bits on.
.equ delay_counter = 125     ;Because the delay takes 3 instructions 250/3 is 166.
.equ zero = 0x00           ;Zero constant.
.equ ones = 0xFF                              ;FF for dir_set.

.org 0x0000
```

```
        rjmp main

.org 0x200

main:
                                ldi YL, low(stack_init)             ;Initialize low byte of stack pointer.
                                out CPU_SPL, YL
                                ldi YL, high(stack_init)    ;Initialize high byte of stack pointer.
                                out CPU_SPH, YL         ;We can use the same register YL for both high and
low.

                                ldi R16, even           ;Store even in R16.
                                ldi R17, odd            ;Store odd in R17.
                                ldi R18, delay_counter      ;Store delay_counter into  R18.
                                ldi R19, ones           ;Store FF into R19.
                                ldi R20, zero           ;Store zero into R20.
                                sts PORTE_DIRSET, R19       ;Set the directon bits of PortE to output.




loop:

                                sts PORTE_OUT, R16          ;Turn on Even LEDs.
                                rcall DELAY_250us           ;Delay for 2Khz. Call the delay subroutine.
                  sts PORTE_OUT, R17            ;Turn on Odd LEDs.
                                rcall DELAY_250us           ;Delay for 2Khz. Call the delay subroutine.
                                rjmp loop               ;Endless repetitive loop.


DELAY_250us:                                                                ;500 us delay divided by two for
50% duty cycle.

                                push R18                ;Push R18 onto stack for future use.
delay_loop:             dec R18             ;Decrement R18.
                                cpi R18, zero                               ;Compare R18 with zero.
                                brne delay_loop         ;If not equal to zero loop.
                                pop R18             ;Pop R18 so we can use it again.
                                ret
```

Part C

```
;
; lab2c.asm
;
; Created: 5/30/2016 3:59:22 PM
; Author : James Mak
;


.include "ATxmega128A1Udef.inc"

;My keypad will have the rows as low and the columns as high.

.equ row1 = 0b0111              ;This turns on the first row for scanning.
.equ row2 = 0b1011                      ;This turns on the second row for scanning.
.equ row3 = 0b1101                      ;This turns on the third row for scanning.
.equ row4 = 0b1110                      ;This turns on the fourth row for scanning.
.equ config = 0x0F      ;This is the configuration used for DIR_SET.
.equ stack_init = 0x2FFF    ;This is where we'll place our stack.
.equ pull_up = 0x18     ;The configuration bits for pull-up resistor.
.equ table_size = 200     ;Reserve 200 bytes of memory for data.
.equ no_press = 0x07            ;Zero.

.org 0x100          ;We want to place a table here.

table: .db 0x78, 0x74, 0x72, 0x71, 0xB8, 0xB4, 0xB2, 0xB1, 0xD8, 0xD4, 0xD2, 0xD1, 0xE8, 0xE4, 0xE2, 0xE1 ;This
table represents the combinations of a keypad button.
key:   .db 0x1, 0x2, 0x3, 0xA, 0x4, 0x5, 0x6, 0xB, 0x7, 0x8, 0x9, 0xC, 0xE, 0x0, 0xF, 0xD ;This is the key that the
above table corresponds to.

.dseg
.org 0x2000                             ;This is where our outputs will go.

out_table: .byte table_size       ;Reserve some space for data.


.cseg
.org 0x0000
                    rjmp MAIN

.org 0x0200

MAIN:
                            ldi XL, low(out_table)      ;Load the low byte of the new table location (2FFF).
                            ldi XH, high(out_table)     ;Load the high byte of the new table location (2FFF).
                            ldi R16, config                         ;Load the DIR_SET
configuration 0x0F.
                            sts PORTF_DIRSET, R16       ;Configure the I/O pins of Port F.
```

```
                ldi R16, pull_up                        ;Load the pull-up resistor
configuration.
                sts PORTF_PIN7CTRL, R16                 ;Set pull-up resistor to pin 7.
                sts PORTF_PIN6CTRL, R16                         ;Set pull-up resistor to pin
6.
                sts PORTF_PIN5CTRL, R16                         ;Set pull-up resistor to pin
5.
                sts PORTF_PIN4CTRL, R16                 ;Set pull-up resistor to pin 4.
                ldi YL, low(stack_init)         ;Load the low byte of the stack pointer.
                out CPU_SPL, YL                 ;Initialize the low byte of the stack pointer.
                ldi YL, high(stack_init)        ;Load the high byte of the stack pointer.
                out CPU_SPH, YL                                         ;Initialize the high
byte of the stack pointer.


LOOP:
                rcall SCAN              ;Call the SCAN subroutine.
                rjmp LOOP               ;Repeat Infinitely.

SCAN:

ROW:            ldi R16, row1           ;Load row 1 into R16.
                sts PORTF_OUT, R16              ;Turn row 1 on.
                lds R18, PORTF_IN               ;Load Input values from Port F into R18.
                nop
                cpi R18, 0xF7                   ;Check to see if a key was pressed.
                brne COL                        ;If a key is pressed branch to see what the column is.
                ldi R16, row2                   ;Load row 2 into R16.
                sts PORTF_OUT, R16              ;Turn row 2 on.
                lds R18, PORTF_IN               ;Load Input values from Port F into R18.
                nop
                cpi R18, 0xFB                   ;Check to see if a key was pressed.
                brne COL                        ;If a key is pressed branch to see what the column is.
                ldi R16, row3                   ;Load row 3 into R16.
                sts PORTF_OUT, R16              ;Turn row 3 on.
                lds R18, PORTF_IN               ;Load Input values from Port F into R18.
                nop
                cpi R18, 0xFD                   ;Check to see if a key was pressed.
                brne COL                        ;If a key is pressed branch to see what the column is.
                ldi R16, row4                   ;Load row 4 into R16.
                sts PORTF_OUT, R16              ;Turn row 4 on.
                lds R18, PORTF_IN               ;Load Input values from Port F into R18.
                nop
                cpi R18, 0xFE                 ;Check to see if a key was pressed.
                brne COL                        ;If a key is pressed branch to see what the column is.
                rjmp ROW                        ;Repeat if no key is pressed.

COL:
                ldi ZL, low(key << 1)                   ;Load the low byte of table location.
                ldi ZH, high(key << 1)          ;Load the high byte of the table location.
```

```
OUTPUT:
                    cpi R18, 0x77           ;Check to see if 1 is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0xB7                                   ;Check to see if 2 is
pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0xD7            ;Check to see if 3 i pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0xE7            ;Check to see if A is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0x7B            ;Check to see if 4 is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0xBB            ;Check to see if 5 is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0xDB            ;Check to see if 6 is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0xEB            ;Check to see if B is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0x7D            ;Check to see if 7 is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0xBD            ;Check to see if 8 is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0xDD            ;Check to see if 9 is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0xED            ;Check to see if C is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0x7E            ;Check to see if E is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0xBE            ;Check to see if 0 is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0xDE            ;Check to see if F is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0xEE            ;Check to see if D is pressed.
                    lpm R19, Z+             ;Access Key.
```

```
                                    breq WRITE
WRITE:
                                    st X+, R19              ;Store E if it is equal.
                                    ret                     ;Return to start of subroutine.
```

Part D

```
;
; lab2c.asm
;
; Created: 5/30/2016 3:59:22 PM
; Author : James Mak
;


.include "ATxmega128A1Udef.inc"

;My keypad will have the rows as low and the columns as high.

.equ row1 = 0b0111                  ;This turns on the first row for scanning.
.equ row2 = 0b1011                           ;This turns on the second row for scanning.
.equ row3 = 0b1101                           ;This turns on the third row for scanning.
.equ row4 = 0b1110                           ;This turns on the fourth row for scanning.
.equ config_f = 0x0F         ;This is the configuration used for PORTF DIR_SET.
.equ config_e = 0xFF        ;This is the configuration used for PORTE DIR_SET.
.equ stack_init = 0x2FFF    ;This is where we'll place our stack.
.equ pull_up = 0x18         ;The configuration bits for pull-up resistor.
.equ table_size = 200       ;Reserve 200 bytes of memory for data.
.equ no_press = 0x07                ;Zero.

.org 0x100              ;We want to place a table here.

table: .db 0x78, 0x74, 0x72, 0x71, 0xB8, 0xB4, 0xB2, 0xB1, 0xD8, 0xD4, 0xD2, 0xD1, 0xE8, 0xE4, 0xE2, 0xE1 ;This table represents the combinations of a keypad button.
key:   .db 0x1, 0x2, 0x3, 0xA, 0x4, 0x5, 0x6, 0xB, 0x7, 0x8, 0x9, 0xC, 0xE, 0x0, 0xF, 0xD ;This is the key that the above table corresponds to.

.dseg
.org 0x2000                                  ;This is where our outputs will go.

out_table: .byte table_size         ;Reserve some space for data.


.cseg
.org 0x0000
                           rjmp MAIN
```

```
.org 0x0200

MAIN:
                                    ldi XL, low(out_table)      ;Load the low byte of the new table location (2FFF).
                                    ldi XH, high(out_table)     ;Load the high byte of the new table location (2FFF).
                                    ldi R16, config_f                            ;Load the DIR_SET configuration
0x0F.
                                    sts PORTF_DIRSET, R16         ;Configure the I/O pins of Port F.
                                    ldi R16, config_e           ;Load the DIR_SET configuration 0xFF.
                                    sts PORTE_DIRSET, R16      ;Configure the I/O pins of Port E, all are outputs.
                                    ldi R16, pull_up                             ;Load the pull-up resistor
configuration.
                                    sts PORTF_PIN7CTRL, R16                  ;Set pull-up resistor to pin 7.
                                    sts PORTF_PIN6CTRL, R16                        ;Set pull-up resistor to pin
6.
                                    sts PORTF_PIN5CTRL, R16                        ;Set pull-up resistor to pin
5.
                                    sts PORTF_PIN4CTRL, R16                  ;Set pull-up resistor to pin 4.
                                    ldi YL, low(stack_init)     ;Load the low byte of the stack pointer.
                                    out CPU_SPL, YL              ;Initialize the low byte of the stack pointer.
                                    ldi YL, high(stack_init)    ;Load the high byte of the stack pointer.
                                    out CPU_SPH, YL                                          ;Initialize the high
byte of the stack pointer.


LOOP:
                                    rcall SCAN             ;Call the SCAN subroutine.
                                    rjmp LOOP               ;Repeat Infinitely.

SCAN:

ROW:                       ldi R16, row1            ;Load row 1 into R16.
                                    sts PORTF_OUT, R16        ;Turn row 1 on.
                                    nop
                                    nop
                                    lds R18, PORTF_IN         ;Load Input values from Port F into R18.
                                    nop
                                    nop
                                    cpi R18, 0xF7            ;Check to see if a key was pressed.
                                    brne COL               ;If a key is pressed branch to see what the column is.
                                    ldi R16, row2           ;Load row 2 into R16.
                                    sts PORTF_OUT, R16        ;Turn row 2 on.
                                    nop
                                    nop
                                    lds R18, PORTF_IN         ;Load Input values from Port F into R18.
                                    nop
                                    nop
                                    cpi R18, 0xFB            ;Check to see if a key was pressed.
                                    brne COL               ;If a key is pressed branch to see what the column is.
```

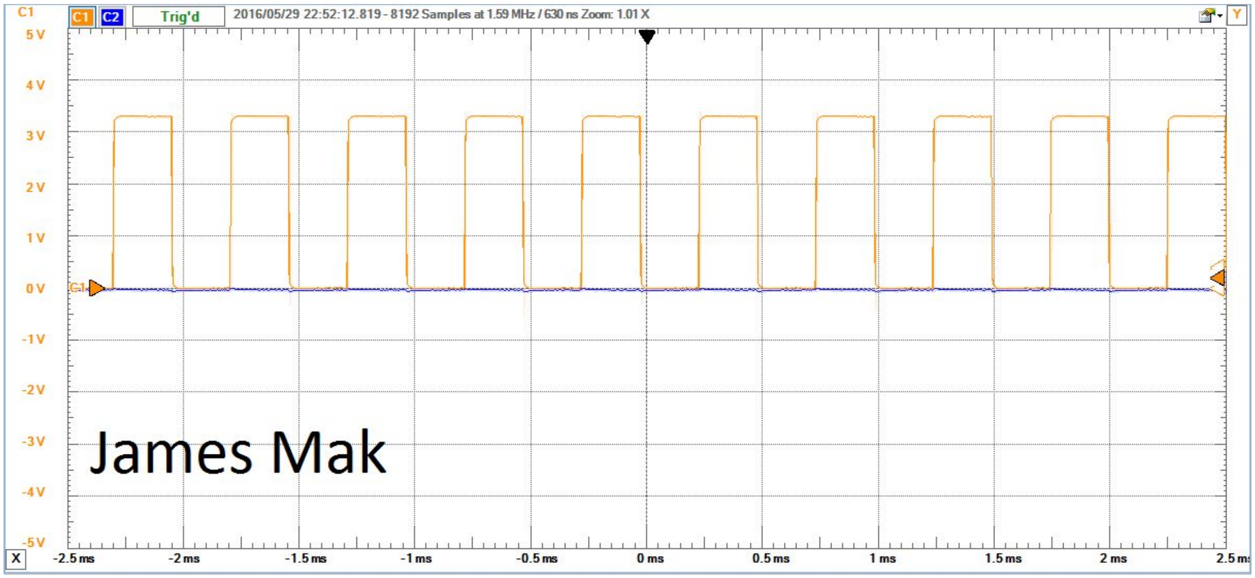
```
                    ldi R16, row3              ;Load row 3 into R16.
                    sts PORTF_OUT, R16             ;Turn row 3 on.
                    nop
                    nop
                    lds R18, PORTF_IN              ;Load Input values from Port F into R18.
                    nop
                    nop
                    cpi R18, 0xFD               ;Check to see if a key was pressed.
                    brne COL                   ;If a key is pressed branch to see what the column is.
                    ldi R16, row4              ;Load row 4 into R16.
                    sts PORTF_OUT, R16             ;Turn row 4 on.
                    nop
                    nop
                    lds R18, PORTF_IN              ;Load Input values from Port F into R18.
                    nop
                    nop
                    cpi R18, 0xFE               ;Check to see if a key was pressed.
                    brne COL                   ;If a key is pressed branch to see what the column is.
DEFAULT:            ldi R16, config_e          ;Load all ones to R16.
                    sts PORTE_OUT, R16                                  ;Output all ones if no
button is pressed.
                    nop
                    rjmp ROW                    ;Repeat if no key is pressed.

COL:
                    ldi ZL, low(key << 1)                           ;Load the low byte of table location.
                    ldi ZH, high(key << 1)         ;Load the high byte of the table location.

OUTPUT:
                    cpi R18, 0x77               ;Check to see if 1 is pressed.
                    lpm R19, Z+                 ;Access Key.
                    breq WRITE
                    cpi R18, 0xB7                                       ;Check to see if 2 is
pressed.
                    lpm R19, Z+                 ;Access Key.
                    breq WRITE
                    cpi R18, 0xD7               ;Check to see if 3 i pressed.
                    lpm R19, Z+                 ;Access Key.
                    breq WRITE
                    cpi R18, 0xE7               ;Check to see if A is pressed.
                    lpm R19, Z+                 ;Access Key.
                    breq WRITE
                    cpi R18, 0x7B               ;Check to see if 4 is pressed.
                    lpm R19, Z+                 ;Access Key.
                    breq WRITE
                    cpi R18, 0xBB               ;Check to see if 5 is pressed.
                    lpm R19, Z+                 ;Access Key.
                    breq WRITE
                    cpi R18, 0xDB               ;Check to see if 6 is pressed.
                    lpm R19, Z+                 ;Access Key.
                    breq WRITE
                    cpi R18, 0xEB           ;Check to see if B is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0x7D           ;Check to see if 7 is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0xBD           ;Check to see if 8 is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0xDD           ;Check to see if 9 is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0xED           ;Check to see if C is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0x7E           ;Check to see if E is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0xBE           ;Check to see if 0 is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0xDE           ;Check to see if F is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    cpi R18, 0xEE           ;Check to see if D is pressed.
                    lpm R19, Z+             ;Access Key.
                    breq WRITE
                    rjmp DEFAULT                ;Back to row if nothing matches.
WRITE:
                    sts PORTE_OUT, R19          ;Store PORT E.
                    ret                     ;Return to start of subroutine.
```

Appendix:

Part B: 2KHz Waveform PORTE output.


C1
C1
C2
Trig'd
2016/05/29 22:52:12.819 - 8192 Samples at 1.59 MHz / 630 ns Zoom: 1.01 X
Y
5 V
4 V
3 V
2 V
1 V
0 V
-1 V
-2 V
-3 V
-4 V
-5 V
C1
James Mak
X
-2.5 ms
-2 ms
-1.5 ms
-1 ms
-0.5 ms
0 ms
0.5 ms
1 ms
1.5 ms
2 ms
2.5 ms