Lab 3
Jacob Hillebrand
CEE-345 Microprocessor System Design
Keypad Interfacing Lab

Part 1

This lab's focus was exploring interfacing with a Digilent keypad with the Atmel 8515 microprocessor. The idea was that the keypad would be connected to the PORTD pins on the processor and when a number was pressed, the PORTB LEDs on the microprocessor would blink the corresponding number of times (ie. if "2" was pressed, all 8 PORTB LEDs would flash 2 times). Part 1 of this assignment was written in C.
The code loaded the board's registers, set up PORTB and PORTD, performed a rapid column vs. row scan of the keypad, accurately recognized and debounced a keypad press, took in the value of the keypress, then looped through the blinking process for the LEDs. At this point, the process would start over, and the system would wait for another keypress. The code for this program is shown on the next pages.

Part 2

The second part of this lab had the exact same result as the first part, but this time, the code was written in Assembly.
Again, this code loaded the board's registers and polled the keypad for a key press. When one was detected, the value was loaded into special "Z" pointers, and passed to 2 distinct functions (f1 and f2) to iterate through the keypress number and flash the LEDs the given number of times. The code for this program is also given on the next few pages.

Part 1 Screenshot

```c
/*
 * Part_1.c
 *  C version of Lab_3 - Keypad input
 * Created: 2/11 3:03:28
 * Author : Jacob Hillebrand
 */
//This exercise is a Keypad-LED blinker program. The LEDs on the STK-600
//blink number of times according to the number pressed on the
//16-key Digilent keypad. Keys 1,2,3 are implemented in C

//PORTD is connected to Digilent PmodKeypad
//PORTB is connected to LEDs


#include <avr/io.h>
#define F_CPU 4000000UL     //Set clock frequency to 4 MHz
#include <util/delay.h>     //Header file allowing a delay function
#include <stdlib.h>         //imports the C standard library

char key_code[] = {0xE7,0xEE,0xDE,0xBE,0xED,0xDD,0xBD,0xEB,0xDB,0xBB,0x7E,0x7D,0x7B,0x77,0xB7,0xD7};

//a delay function for switch debounce
void delay(long value){
    while(--value);
}

//Scan function to scan keypad to read key-press
char scan(){
    //Declare necessary variables
    char temp,key,i;

    //Begin the scan loop
    PORTD = 0x0f;

    //Make temp == PIND
    do {
        temp = PIND;
    } while (temp != 0x0f);

    while (1) {
        for (i = 0; i < 4; i++){
            temp =~(0x80 >> i);     //~(0b1000 0000) = 0b0111 1111
            PORTD = temp;
            key = PIND;
            if (key != temp){
                delay(20);      //delay for the switch debounces
                key = PIND;     //a pressed key is detected and return return its value
                if (key != temp) return (key);
            }
        }
    }

}

//Test if a key was pressed
char gotkey(){
    char temp;
    int i;
    temp = scan();
    for (i = 0; i <= 15; i++){
        if (temp == key_code[i]) return(i);
    }
    return (16);
}

int main(void)
{

    char keys;

    //Initialize Port B
    PORTB = 0x00;
    DDRB = 0xFF;

    //Initialize Port D
    PORTD = 0x00;
    DDRD = 0xF0;

    while (1) {
        keys = gotkey();

        while(keys != 0){
            //Cycle the LEDs on
            PORTB = 0xFF;
            _delay_ms(500);
            //Cycle the LEDs off
            PORTB = 0x00;
            _delay_ms(500);
            //Decrement keys
            keys = keys-1;
        }

    }
}
```

Figure 1: Code from Part 1

Part 2 Screenshots

```
;***************************************************************************
;INTERFACEING OF A 4 BY 4 KEYPAD AND 8 LEDS WITH THE ATMEGA8515 MICROCONTROLLERS
;
; This assembly program is to illustrate the interfacing of a 4x4 keypad and 8
; 8 LEDs to the ATMEGA8515 Microcontrollers using the STK-600 board
;
; The keypad has keys from numeric hex numbers (0 to 9) and (A to F) with a total
; of 16 keys. This program is to flash LEDs with a number of times depending on
; what number on the keypad was pressed. ie. if 2 is pressed, the LEDs will flash
; 2 times.
;
; 4x4 keypad is connected to PORTD
; LEDs are connected to PORTB
;***************************************************************************
;
; Part_2.asm
;
; Created: 2/13 2:18:36
; Author : Jacob Hillebrand

.def        temp = r16

.include "M8515DEF.INC"

.def        key = r17
.def        check = r24
.def        xtime = r25
.cseg
.org        0

rjmp        start

;THE "start" Routine is to create the stack and set output port with PORTB

start:
    ldi temp, low(RAMEND)       ;load SPL (the lowest byte of the stack)
    out SPL, temp               ;load low byte address to SPL pointer register
    ldi temp, high(RAMEND)      ;load SPH (the high byte of the stack)
    out SPH, temp               ;load high byte address to SPH pointer address

    ldi temp, 0xff              ;write all 1s to temp
    out DDRB, temp              ;set up DDRB as output
    out PORTB, temp             ;set up PORTB as output


main:
    rcall   get_key     ;the get_key function call is to get the pressed keys from a 4x4 keypad
    mov     xtime, key  ;store the pressed key r17 to register r25
    rcall   f1          ;function call to flash LEDs xtime on the PORTB connected to LEDs
    rjmp    main        ;jump back to the main routine


scan:
    ldi     temp, 0xF0      ; load 0xF0 to the temp register;
    out     DDRD, temp      ; -the higher byte of (0xF0) is connected to keypad outputs
    ldi     check, 0x0F     ; load 0x0F to a register "check"
    ldi     temp, 0x0F      ; load 0x0F to a temp; this assigns a high signal to the keypad inputs
    out     PORTD, temp     ; load 0x0F to PORTD

;THE "not_ready" routine checks if any pressed key on the keypad; if not key is pressed, return
; to not_ready routine

not_ready:

    in temp, PIND           ; poll the status of PORTD pins and detect if any key is pressed
    eor temp, check         ; use xor to determine which key is pressed
    brne not_ready          ; exit loop (not_ready) if the values in the temp and check are NOT equal

; The scan_loop routine is to load two variables (temp & xtime) for scan keypad scan
scan_loop:

    ldi     temp, 0xFF      ; load 0xFF to the temp register
    ldi     xtime, 0x04     ; load 0x04 to xtime register to scan rows and columns
    clc                     ; clear carry flag


scan_next_row:

    ror temp                ; rotate to the right for the temp register
    mov check, temp         ; move the temp register to check
    out PORTD, temp         ; output temp to PORTD
    in key, PIND            ; input PIND to key
    eor key, temp           ; compare the values in the temp register and the key register
    breq next_key           ; when equal, branch to the next_key function
    rcall sdelay            ; call the delay function sdelay for keypad
    in key, PIND            ; input PIND to ky
    eor temp, key           ; xor temp with key
    brne gotkey             ; branch to gotkey if a pressed, exit if the values in temp and check
                        ;   are NOT equal (brne)

```

Figure 2: First Snippet of Part 2

```
next_key:

    mov temp, check         ; move check register to temp register
    dec xtime               ; decrement xtime by 1 (xtime-1)
                            ; compare temp and xtime
    brne scan_next_row      ; if temp is not zero, jump to scan_next_row
    rjmp scan_loop          ; otherwise jump to scan_loop routine

gotkey:
    ret         ;return to the main routine when a key match occurs


get_key:

    rcall   scan                        ; call the function scan to scan for any pressed keys
    ldi     xtime, 0x00                 ; load 0x00 to xtime register
    ldi     ZL, LOW(2*key_code)         ; load the key code to the lower byte of Z register
    ldi     ZH, HIGH(2*key_code)        ; load the key code to the higher byter of Z register

;check_next is used to check the key pressed key with the pre-loaded keycodes to identify
; which key is pressed on the keypad

check_next:

    lpm                             ; use the instruction LPM to load the Z register pointer to the address
                                    ; the key codes stored in memory; use the default register r0
    mov     temp, r0                ; move the r0 to the temp register
    cp      temp, key               ; compare (cp) temp with the pressed key
    breq    found                   ; branch to found routine if temp is zero
    adiw    ZH:ZL,1                 ; increment the Z register pointer to the next position
    inc     xtime                   ; increment the xtime by 1
    cpi     xtime, 0x10             ; compare (cpi) with 0x10
    brne    check_next              ; branch to check_next routine if xtime is not equal to zero
    ldi     key,0x20                ; load 0x20 to key if temp is equal to zero
    ret                             ; return to the main routine

found:
    mov     key, xtime              ; if the pressed key is found, move xtime to key register
    ret                             ; return to the main routine


; f1 routine is the routine to flash leds the number of times according to the number in (0-9)
; and (A-F) key is pressed on a keypad. If the key "A" in hex is pressed, the LEDs flash 10 times

f1:
    ldi     temp, 0x00
    out     PORTB, temp
    rcall   one_sec_delay
    ldi     temp, 0xff
    out     PORTB, temp
    rcall   one_sec_delay

    cpi     xtime, 0
    brne    f2
    inc     xtime

f2:
    dec     xtime
    brne    f1
    ret

;**** Time delay for a keypad between each press ************************
sdelay:
    ldi     r21, 64
    ldi     r22, 255

idelay:
    dec     r22
    brne    idelay
    dec     r21
    brne    idelay
    ret

;*****Time delay for LEDs flash ***********************************

one_sec_delay:

    ldi     r20, 2
    ldi     r21, 255
    ldi     r22, 255

delay:  dec     r22
        brne    delay
        dec     r21
        brne    delay
        dec     r20
        brne    delay
        ret
```

Figure 3: Second Snippet from Part 2

```
180
181
182  ;**********key_code for each key on the 4x4 keypad ********************************
183  key_code:
184
185      .db     0xE7, 0xEE, 0xDE, 0xBE, 0xED, 0xDD, 0xBD, 0xEB
186      .db     0xDB, 0xBB, 0x7E, 0x7D, 0x7B, 0x77, 0xB7, 0xD7
187
188  ; END
```

Figure 4: Third Snippet from Part 2