Lab 7
Jacob Hillebrand
CEE-345 Microprocessor System Design
Keypad Exploration Lab

In this lab, we did some extensive exploration of using the Digilent keypad with the FreedomBoard. For part 1, we connected the keypad to the board and programmed it so that when any key was pressed and held on the keypad, the blue LED on the FreedomBoard would turn on and stay on. As soon as the key was released, the light turned off. To do this, we set up the columns of the keypad to outputs and the rows to inputs. We then powered the columns of the keypad, and searched for a signal on the rows. If a signal was found, the LED was set to be on. If not, the LED was set to be off. For part 2 of the lab, we switched up the functionality of the keypad a bit by programming the board such that when "1" was pressed on the keypad, the red onboard LED turned on, and when "2" was pressed, the red LED was turned off. Similarly, pressing the "E" key would turn on the green LED, and pressing the "D" key would turn off the green LED. To achieve this, we had a similar setup as part 1. However, in this part, when input *was* detected from the keypad, we then performed a further scanning action. We powered 1 column at a time, then scanned all rows. If no input was detected, the column was powered off, the next one powered on, and all rows scanned again. This process was repeated until the row-and-column values were found for the key, which could then be indexed and returned. Then, it was just a simply matter of running a check to see if certain keys were pressed, and if so, turning on or off certain LEDs. The code for both part 1 and part 2 can be found below.

Part 1

```
 * PortC 7-4 are connected to the columns and PortC 3-0 are connected
 * to the rows.
 */

#include <MKL25Z4.H>
#include "gpio_defs.h"

void delayUs(int n);

void keypad_init(void);
char keypad_kbhit(void);

int main(void)
{
    keypad_init();
    SIM->SCGC5 |= 0x1000;       /* enable clock to Port D */
    PORTD->PCR[1] = 0x100;      /* make PTD1 pin as GPIO */
    PTD->PDDR |= 0x02;          /* make PTD1 as output pin */

    while(1)
    {
        if (keypad_kbhit() != 0)    /* if a key is pressed? */
          //  PTD->PCOR |= 0x02;          /* turn on blue LED */
                PTD->PDOR =  ~MASK(BLUE_LED_POS); /* turn on blue LED */
        else
           // PTD->PSOR |= 0x02;          /* turn off blue LED */
                PTD->PDOR =  MASK(BLUE_LED_POS); /* turn off blue LED */
    }
}


/* this function initializes PortC that is connected to the keypad.
 * All pins are configured as GPIO input pin with pullup enabled.
 */
void keypad_init(void)
{
    SIM->SCGC5 |= 0x0800;       /* enable clock to Port C */
    PORTC->PCR[0] = 0x103;      /* make PTD0 pin as GPIO and enable pullup*/
    PORTC->PCR[1] = 0x103;      /* make PTD1 pin as GPIO and enable pullup*/
    PORTC->PCR[2] = 0x103;      /* make PTD2 pin as GPIO and enable pullup*/
    PORTC->PCR[3] = 0x103;      /* make PTD3 pin as GPIO and enable pullup*/
    PORTC->PCR[4] = 0x103;      /* make PTD4 pin as GPIO and enable pullup*/
    PORTC->PCR[5] = 0x103;      /* make PTD5 pin as GPIO and enable pullup*/
    PORTC->PCR[6] = 0x103;      /* make PTD6 pin as GPIO and enable pullup*/
    PORTC->PCR[7] = 0x103;      /* make PTD7 pin as GPIO and enable pullup*/
    PTD->PDDR = 0x0F;          /* make PTD7-0 as input pins */
}
```

Figure 1: Snippet 1/2

```
/* This is a non-blocking function.
 * If a key is pressed, it returns 1.
 * Otherwise, it returns a 0 (not ASCII '0'). */
char keypad_kbhit(void)
{
    int col;

    PTC->PDDR |= 0x0F;          /* enable column (PTC4-PTC7) as input */
                                /* enable row (PTC0-PTC3) as output */
    PTC->PCOR = 0x0F;           /* enable all rows driven low */
    delayUs(2);                 /* wait for signal return */
    col = PTC->PDIR & 0xF0;     /* read all columns */
                                /* PDOR has a vaue of 0xF0 when no key pressed */

    PTC->PDDR = 0;              /* disable all rows */
    if (col == 0xF0)
        return 0;               /* no key pressed */
    else
        return 1;               /* a key is pressed */
}

void delayUs(int n)
{
    int i; int j;
    for(i = 0 ; i < n; i++) {
        for(j = 0; j < 8; j++) ;
    }
}
```

Figure 2: Snippet 2/2

Part 2

```c
/* Matrix keypad scanning

 * This program scans a 4x4 matrix keypad and returns a unique number
 * for each key pressed.  The number will turn on/off the red and green
 * LEDs.
 *
 *                  FRDM Pin
 * row 0    PTC10   J1-13
 * row 1    PTC11   J1-15
 * row 2    PTC12   J2-1
 * row 3    PTC13   J2-3
 * col 0    PTE20   J10-1
 * col 1    PTE21   J10-3
 * col 2    PTE22   J10-5
 * col 3    PTE23   J10-7
 */

#include <MKL25Z4.H>
#include "gpio_defs.h"


void delayMs(int n);
void delayUs(int n);
void keypad_init(void);
char keypad_getkey(void);
void LED_init(void);
void LED_set(char value);

int main(void)
{
    unsigned char key;

    keypad_init();
    LED_init();

    while(1)
    {
        key = keypad_getkey();
        LED_set(key);

                        /* if key == '1' */
                        /* turn on red LED*/
                        /* if key == '2' */
                        /* turn off red LED*/
                        /* if key == 'E' */
                        /* turn on green LED*/
                        /* if key == 'D' */
                        /* turn off green LED*/
    }
}
```

Figure 3: Snippet 1/4

```c
/* this function initializes PortC that is connected to the keypad.
 * All pins are configured as GPIO input pin with pull-up enabled.
 */
void keypad_init(void)
{
    SIM->SCGC5 |= 0x0800;       /* enable clock to Port C */
    SIM->SCGC5 |= 0x2000;       /* enable clock to Port E */

    PORTC->PCR[10] = 0x100;     /* make PTC10 pin as GPIO */
    PORTC->PCR[11] = 0x100;     /* make PTC11 pin as GPIO */
    PORTC->PCR[12] = 0x100;     /* make PTC12 pin as GPIO */
    PORTC->PCR[13] = 0x100;     /* make PTC13 pin as GPIO */

    PORTE->PCR[20] = 0x100;     /* make PTE20 pin as GPIO */
    PORTE->PCR[21] = 0x100;     /* make PTE21 pin as GPIO */
    PORTE->PCR[22] = 0x100;     /* make PTE22 pin as GPIO */
    PORTE->PCR[23] = 0x100;     /* make PTE23 pin as GPIO */

  //  PTC->PDDR &= ~(0xF << 10);  /* make PTC13-10 as inputs connected to rows */
  //  PTE->PDDR &= ~(0xF << 20);  /* make PTE23-20 as inputs connected to columns */
}
```

Figure 4: Snippet 2/4

```c
 * This is a non-blocking function to read the keypad.
 * If a key is pressed, it returns a key code. Otherwise, a zero
 * is returned.
 * when the keys are not pressed, these pins are pull down LOW.
 * The Port E (PTE23-20) is used as output that drives the keypad cloumns.
 * First all rows are driven low and the input pins are read. If no
 * key is pressed, it will read as low at all rows.
 * If any key is pressed, the program drives one column high at a time and
 * leave the rest of the rows inactive (low) then read the output pins.
 * Knowing which row is active and which column is active, the program
 * can decide which key is pressed.
 */
char keypad_getkey(void)
{
    int row, col;
    const char col_select[] = {0x01, 0x02, 0x04, 0x08}; /* one row is active */
    const char keymap[] = " 123A456B789C0FED";

    /* check to see any key pressed first */
    PTE->PDDR |= 0xF << 20;             /* enable columns as outputs (PTE20-23)*/
    PTE->PSOR = 0xF << 20;              /* set column data pins all high in PDOR for PTE */
    delayUs(2);                         /* wait for signal to settle */
    row = (PTC->PDIR >> 10) & 0xF;      /* read all rows at PORT C (PTC10-13) */
    PTE->PCOR = 0xF << 20;              /* set column data pins all low */
    PTE->PDDR &= ~(0xF << 20);          /* disable all columns */

    if (row == 0) return 0;             /* no key pressed, return a zero */

    /* If a key is pressed, it gets here to find out which key.
     * It activates one row at a time and read the input to see
     * which column is active. */

    for (col = 0; col <= 4; col++)
    {
        PTE->PDDR &= ~(0xF << 20);          /* disable all cols */
        PTE->PDDR |= col_select[col] << 20; /* enable one col */
        PTE->PSOR = 0x0F << 20;             /* turn the active col high */
        delayUs(2);                         /* wait for signal to settle */
        row = (PTC->PDIR >> 10) & 0xF;      /* read all rows */
        PTE->PCOR = 0xF << 20;              /* set column data pins all low */
        if (row != 0) break;                /* if one of the input is low, some key is pressed. */
    }

    PTE->PDDR &= ~(0xF << 20);          /* disable all cols */
    if (col == 4)
        return 0;                       /* if we get here, no key is pressed */

    /* gets here when one of the rows has key pressed, check which column it is */
        /* col goes from 0 to 3 as shown in the col scan above */
   // if (row == 0x01) return col * 4 + 1;    /* key in column 0 */
   // if (row == 0x02) return col * 4 + 2;    /* key in column 1 */
   // if (row == 0x04) return col * 4 + 3;    /* key in column 2 */
   // if (row == 0x08) return col * 4 + 3;    /* key in column 3 */

        if (row == 0x01) return keymap[col + 0 * 4 + 1];    // key in column 0
    if (row == 0x02) return keymap[col + 1 * 4 + 1];    // key in column 1
    if (row == 0x04) return keymap[col + 2 * 4 + 1];    // key in column 2
    if (row == 0x08) return keymap[col + 3 * 4 + 1];    // key in column 3

    return 0;   /* just to be safe */
}
```

Figure 5: Snippet 3/4

```c
/* initialize all three LEDs on the FRDM board */
void LED_init(void)
{
    SIM->SCGC5 |= 0x400;         /* enable clock to Port B */
    SIM->SCGC5 |= 0x1000;        /* enable clock to Port D */
    PORTB->PCR[18] = 0x100;      /* make PTB18 pin as GPIO */
    PTB->PDDR |= 0x40000;        /* make PTB18 as output pin */
    PTB->PSOR |= 0x40000;        /* turn off red LED */
    PORTB->PCR[19] = 0x100;      /* make PTB19 pin as GPIO */
    PTB->PDDR |= 0x80000;        /* make PTB19 as output pin */
    PTB->PSOR |= 0x80000;        /* turn off green LED */
    PORTD->PCR[1] = 0x100;       /* make PTD1 pin as GPIO */
    PTD->PDDR |= 0x02;           /* make PTD1 as output pin */
    PTD->PSOR |= 0x02;           /* turn off blue LED */
}

/* turn on or off the LEDs according to bit 3-0 of the value */
void LED_set(char value)
{
    if (value == '1')     /* use bit 0 of value to control red LED */
        PTB->PCOR = 0x40000;    /* turn on red LED */
    else if (value == '2')
        PTB->PSOR = 0x40000;    /* turn off red LED */
          else if (value == 'E')
                        PTB->PCOR = 0x80000;    /* turn on green LED */
                      else if (value == 'D')
        PTB->PSOR = 0x80000;    /* turn off green LED */

                      /*
                      Dealing with the blue LED
                      PTD->PDOR = ~MASK(BLUE_LED_POS); //ON
                      PTD->PDOR = MASK(BLUE_LED_POS); //OFF
                      */
}

/* delay n microseconds
 * The CPU core clock is set to MCGFLLCLK at 41.94 MHz in SystemInit().
 */
void delayUs(int n)
{
    int i; int j;
    for(i = 0 ; i < n; i++) {
        for(j = 0; j < 5; j++) ;
    }
}

```

Figure 6: Snippet 4/4