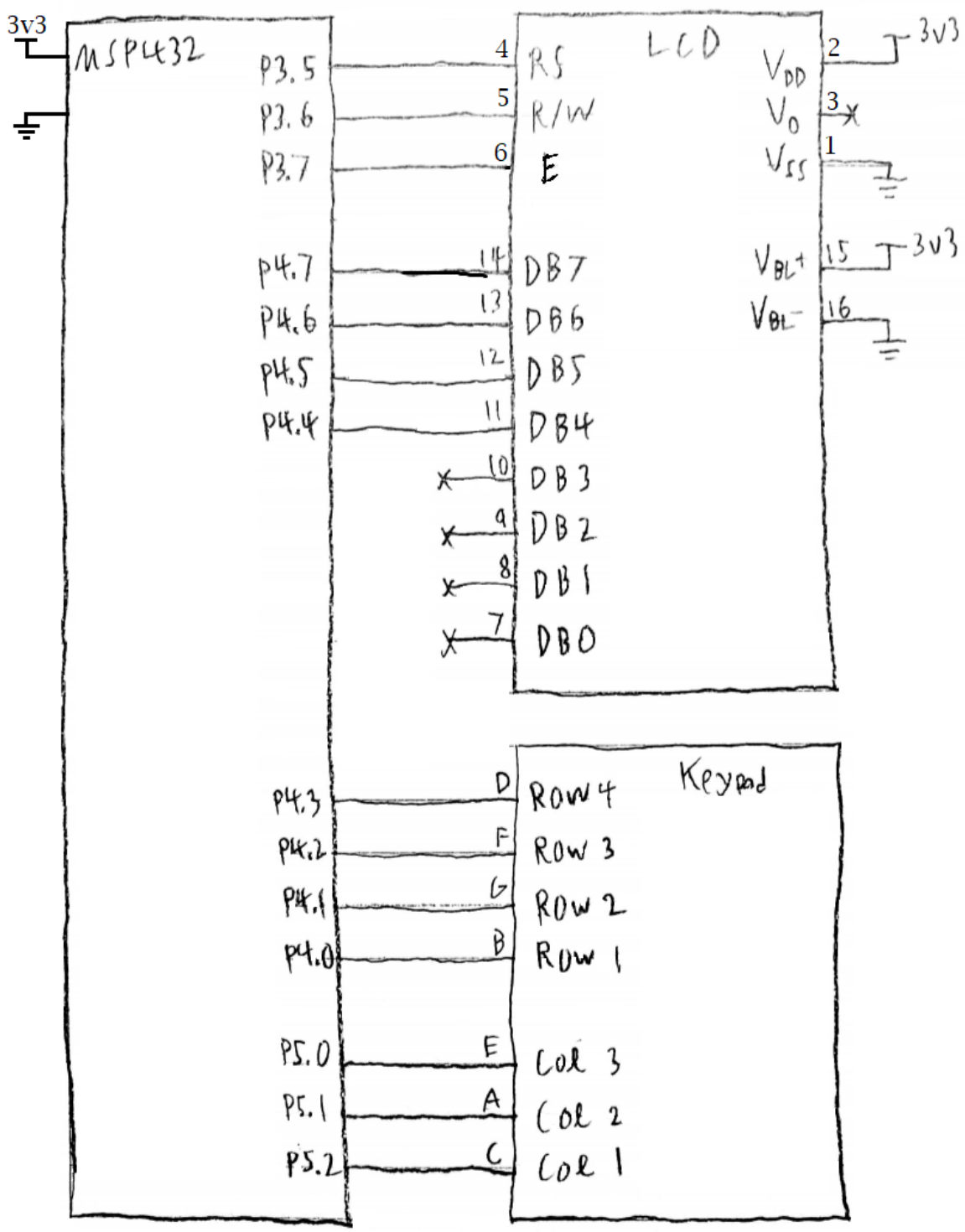
A4 - Keypad Integration

Students: Zach Bunce & Garrett Maxon
Class: CPE329-03
Professor: Gerfen, Jeffrey

Youtube Video Demonstration
https://youtu.be/LCjJVc3zTTk

Schematic Diagram



Main Code Used in Demo

```
/**
 * main.c
 *
 * Date: April 13 2018
 * Authors: Zach Bunce, Garrett Maxon
 *
 * Links the keypad to the LCD, displaying key pressed
 * Consecutive key presses overwrite previous entry
 */

#include "msp.h"
#include "set_DCO.h"
#include "delays.h"
#include "LCD.h"
#include "keypad.h"

void LCD_INIT(int);
void write_char_LCD(uint8_t, uint8_t, int);
void clear_LCD(int);

void delay_ms(int, int);
void set_DCO(int);

void KEYPAD_INIT();
uint8_t chk_Keypad();

void main(void)
{
    WDT_A->CTL = WDT_A_CTL_PW | WDT_A_CTL_HOLD;           // stop watchdog timer

    int CLK = 480;
    set_DCO(CLK);
    LCD_INIT(CLK);
    KEYPAD_INIT();

    uint8_t key = 0x10;
    uint8_t addr = 0x00;
    while(1) {
        //Polls the keypad
        while(key == 0x10) {
            key = chk_Keypad();
        }

        write_char_LCD(key, addr, CLK); //Writes the key pressed to the top left pixel
        home_LCD(CLK); //Returns the cursor to the top left position
        delay_ms(500,CLK);
        key = 0x10; //Clears the keypress
    }
}
```

Keypad Library

```
/*
 * keypad.h
 *
 * Date: April 13, 2018
 * Author: Zach Bunce, Garrett Maxon
 */

#ifndef KEYPAD_H_
#define KEYPAD_H_
void KEYPAD_INIT();
uint8_t chk_Keypad();
uint8_t KEY_LOCATE();
#endif /* KEYPAD_H_ */

/*
 * keypad.c
 *
 * Date: Apr 13, 2018
 * Author: Zach Bunce, Garrett Maxon
 *
 * Call KEYPAD_INIT to enable use of keypad
 * chk_Keypad meant for external use
 * Returns ASCII code of symbol pressed (Including numbers)
 */

#include "msp.h"
#define ROWA    BIT0
#define ROWB    BIT1
#define ROWC    BIT2
#define ROWD    BIT3
#define COL1    BIT0
#define COL2    BIT1
#define COL3    BIT2

//Rows: A-D; Columns 1-3
//First row
#define A1      0x13
#define A2      0x15
#define A3      0x16

//Second row
#define B1      0x23
#define B2      0x25
#define B3      0x26

//Third row
#define C1      0x43
#define C2      0x45
#define C3      0x46
```

```
//Fourth row
#define D1      0x83
#define D2      0x85
#define D3      0x86

//ASCII hex values for keypad characters
#define K_1     0x31
#define K_2     0x32
#define K_3     0x33
#define K_4     0x34
#define K_5     0x35
#define K_6     0x36
#define K_7     0x37
#define K_8     0x38
#define K_9     0x39
#define K_Ast   0x2A
#define K_0     0x30
#define K_Pnd   0x23
#define K_NP    0x10

uint8_t KEY_LOCATE();

//Sets P4.0-4.3 as rows and P5.0-5.2 as columns
void KEYPAD_INIT() {
    P4 -> DIR |= ROWA | ROWB | ROWC | ROWD; //Sets DIR reg of the outputs
    P5 -> DIR &= ~(COL1 | COL2 | COL3);     //Clears DIR reg of the inputs
    P5 -> REN |=  COL1 | COL2 | COL3;       //Enables PU or PD resistor
    P5 -> OUT |=  COL1 | COL2 | COL3; //Confusing syntax, but PU or PD is set through PxOUT reg
}

//Decodes row & column of key pressed into ASCII value
uint8_t chk_Keypad() {
    uint8_t RC_Info = KEY_LOCATE();
    switch(RC_Info) {
    case A1:
        return K_1;
    case A2:
        return K_2;
    case A3:
        return K_3;
    case B1:
        return K_4;
    case B2:
        return K_5;
    case B3:
        return K_6;
    case C1:
        return K_7;
    case C2:
        return K_8;
```

```
    case C3:
        return K_9;
    case D1:
        return K_Ast;
    case D2:
        return K_0;
    case D3:
        return K_Pnd;
    default:
        return K_NP; //Returns no press; empty value in LCD table
    }
}

//Finds and returns row and column of uppermost key pressed
uint8_t KEY_LOCATE()
{
    uint8_t i;
    uint8_t col = 0;
    for (i = 1; i <= 0x08; i = i << 1) {
        P4 -> OUT |= (ROWA | ROWB | ROWC | ROWD); //Sets all rows high
        P4 -> OUT &= ~i & (ROWA | ROWB | ROWC | ROWD); //Sets selected row low
        col = P5->IN & 0x07;   //Reads column states into lower nibble
        if (col == 0x07) {
            //Do nothing
        }
        else {
            i = i << 4; //Shift row info into upper nibble
            col |= i;   //Add row info to column info
            return col; //Return column and row info
        }
    }
    return 0;
}
```

Delays Library

```
/*
 * delays.h
 * Header file for both the ms and us delay functions.
 *
 * Created on: Apr 9, 2018
 * Date: Zach Bunce, Garrett Maxon
 */

#ifndef DELAYS_H_
#define DELAYS_H_

void delay_ms(int, int);
void delay_us(int, int);

#endif /* DELAYS_H_ */
```

```
/*
 * delays.c
 * Code file for both the ms and us delay functions.
 * Utilizes __delay__cycles
 * Divide us by 2 cause ?
 *
 * Date: April 9, 2018
 * Authors: Zach Bunce, Garret Maxon
 */

#include "msp.h"

#define F_1p5_MeHz  15      //Defines various frequency values in almost MHz (10^5)
#define F_3_MeHz    30      //MeHz labels are used to indicate this
#define F_6_MeHz    60      //Blame data type truncation
#define F_12_MeHz   120
#define F_24_MeHz   240
#define F_48_MeHz   480

//Takes in desired time delay in ms and clock frequency in MeHz
//Accurate to less than 1%
void delay_ms(int time_ms, int freq_MeHz)
{
    int i;
    //Unit Conversion: MeHz * ms = 10^5 * 10^-3 = 10^2 = 100
    switch (freq_MeHz) {
    case F_1p5_MeHz:
        //Accurate to 0.1%
        for (i = time_ms; i > 0; i--) {
            __delay_cycles(1500);
        }
        break;
    case F_3_MeHz:
        //Accurate to 0.4%
        for (i = time_ms; i > 0; i--) {
            __delay_cycles(3000);
        }
        break;
    case F_6_MeHz:
        //Accurate to 0.6%
        for (i = time_ms; i > 0; i--) {
            __delay_cycles(6000);
        }
        break;
    case F_12_MeHz:
        //Accurate to 0.7%
        for (i = time_ms; i > 0; i--) {
            __delay_cycles(12000);
        }
        break;
```

```
    case F_24_MeHz:
        //Accurate to 0.6%
        for (i = time_ms; i > 0; i--) {
            __delay_cycles(24000);
        }
        break;
    case F_48_MeHz:
        //Accurate to 0.4%
        //Div by 2 cause ?
        for (i = time_ms; i > 0; i--) {
            __delay_cycles(24000);
        }
        break;
    default:
        break;
    }
}

//Takes in desired time delay in us and clock frequency in MeHz
void delay_us(int time_us, int freq_MeHz)
{
    int i;
    int j;
    int z;
    int time_fix;

    //Unit Conversion: MeHz * us = 10^5 * 10^-6 = 10^-1 = 0.1; Accounted for in decrement
    switch (freq_MeHz)
    {
    case F_1p5_MeHz:
        time_fix = time_us / 2;
        for (i = time_fix; i > 0; i--) {
            for (j = 10; j > 0; j--) {
                __delay_cycles(15);
            }
        }
        break;
    case F_3_MeHz:
        time_fix = time_us / 2;
        for (i = time_fix; i > 0; i--) {
            __delay_cycles(3);
        }
        break;
    case F_6_MeHz:
        time_fix = time_us / 2;
        for (i = time_fix; i > 0; i--) {
            __delay_cycles(6);
        }
        break;
```

```
    case F_12_MeHz:
        time_fix = time_us / 2;
        for (i = time_fix; i > 0; i--) {
            __delay_cycles(12);
        }
        break;
    case F_24_MeHz:
        //Within +1us 26 < t < 100; Accurate within 1% up to 1000 us
        time_fix = (time_us - 1) / 2;
        for (i = time_fix; i > 0; i--) {
            __delay_cycles(24);
            z++;
            z++;
            z++;
        }
        break;
    case F_48_MeHz:
        //Within +1us 25 < t < 100; Accurate within 2% up to 1000 us
        time_fix = time_us / 2;
        for (i = time_fix; i > 0; i--) {
            __delay_cycles(48);
            z++;
            z++;
            z++;
            z++;
        }
        break;
    default:
        break;
    }
}
```

LCD Library

```
/*
 * LCD.h
 * Header file for LCD control.
 *
 * Date: April 10, 2018
 * Author: Zach Bunce, Garrett Maxon
 */

#ifndef LCD_H_
#define LCD_H_
void LCD_INIT(int);
void LCD_CMD(uint8_t, uint8_t, int);
void write_char_LCD(uint8_t, uint8_t, int);
void clear_LCD(int);
void home_LCD(int);
#endif /* LCD_H_ */
```

```
/*
 * LCD.c
 * Code file for LCD control.
 * Functions designed for external use:
 * write_char_LCD, clear_LCD, & home_LCD
 *
 * Date: April 10, 2018
 * Authors: Zach Bunce, Garret Maxon
 */

#include "msp.h"
#include "delays.h"

#define RS          BIT5
#define RW          BIT6
#define EN          BIT7

#define DB0         BIT0
#define DB1         BIT1
#define DB2         BIT2
#define DB3         BIT3
#define DB4         BIT4
#define DB5         BIT5
#define DB6         BIT6
#define DB7         BIT7

#define DISP_CLR    0x01
#define HOME_RET    0x02
//Change the definitions below to change initialization
#define DISP_SET    0x0F //0x0F Disp ON, Cursor ON, Blink ON
#define FXN_SET     0x28 //0x28 4-Bit, 2 line, 5x8 font
#define SHIFT_SET   0x10 //0x10 shifts cursor or disp
#define ENTRY_SET   0x06 //0x06 -> cursor++

void LCD_CMD(uint8_t, uint8_t, int);
void LCD_CTRL(uint8_t, int);

//Takes in DDRAM address pixel and ASCII character sym
//0x00...0x0F DDRAM Addresses
//0x40...0x4F
void write_char_LCD(uint8_t sym, uint8_t pixel, int CLK)
{
    pixel |= DB7;
    uint8_t CTRL = EN;
    LCD_CMD(pixel, CTRL, CLK);
    delay_us(37, CLK);
    CTRL = EN | RS;
    LCD_CMD(sym, CTRL, CLK);
    delay_us(37, CLK);
}
```

```
//1.52 ms delay required after operation
void home_LCD(int CLK)
{
    LCD_CMD(HOME_RET, EN, CLK);
    delay_ms(2, CLK);
}

//1.52 ms delay required after operation
void clear_LCD(int CLK)
{
    LCD_CMD(DISP_CLR, EN, CLK);
    delay_ms(2, CLK);
}

//Sets up I/O register direction
//Runs through LCD setup procedure
//USE LCD_3.3V_LCD_NHD DATASHEET PROCEDURE
//Currently only works for 4-bit mode; scared of timings
void LCD_INIT(int CLK)
{
    P3 -> DIR |= RS | RW | EN;  //Sets reg directions
    P4 -> DIR |= DB7 | DB6 | DB5 | DB4; //Not using all 4 so don't be lazy
    P3 -> OUT &= ~(RS | RW | EN);
    P4 -> OUT &= ~(DB7 | DB6 | DB5 | DB4); //Sets output low
    delay_ms(40, CLK);  //Waits for safe power up
    P4 -> OUT |= 0x30;  //Sets wake up command
    delay_ms(5, CLK);
    LCD_CTRL(EN, CLK);  //Wake up #1
    P3 -> OUT &= ~EN;
    delay_us(160, CLK);
    LCD_CTRL(EN, CLK);  //Wake up #2
    P3 -> OUT &= ~EN;
    delay_us(160, CLK);
    LCD_CTRL(EN, CLK);  //Wake up #3
    P3 -> OUT &= ~EN;
    delay_us(160, CLK);

    P4 -> OUT &= ~(DB7 | DB6 | DB5 | DB4); //Sets output low
    P4 -> OUT |= 0x20;   //Guess we're awake
    LCD_CTRL(EN, CLK);
    P3 -> OUT &= ~EN;

    //The rest sets up LCD settings
    LCD_CMD(FXN_SET,    EN, CLK);
    LCD_CMD(SHIFT_SET,  EN, CLK);
    LCD_CMD(DISP_SET,   EN, CLK);
    LCD_CMD(ENTRY_SET,  EN, CLK);

    clear_LCD(CLK); //Bill Murray
}
```

```
//Sets control bits to states given in CTRL
//CTRL: Top three bits EN, RW, RS respectively
void LCD_CTRL(uint8_t CTRL, int CLK)
{
    P3 -> OUT &= ~(RS | RW | EN); //Clears RS, RW, and EN
    P3 -> OUT |= CTRL; //Sets RS, RW, and EN to desired state
    delay_us(30, CLK); //Nominal 460 ns W / 480 ns R
}

//Writes to the 8 data bits of the LCD
//Can access both data and instruction registers
//30 us delays to ensure known timings
void LCD_CMD(uint8_t CMD, uint8_t CTRL, int CLK)
{
    P4 -> OUT &= ~(DB7 | DB6 | DB5 | DB4); //Sets output low
    P4 -> OUT |= CMD & 0xF0;
    LCD_CTRL(CTRL, CLK);
    P3 -> OUT &= ~EN;   //Nibble 1
    delay_us(30, CLK);  //Nominal 10 ns
    if ((FXN_SET & DB4) == 0x00) {
        CMD = CMD << 4;
        P4 -> OUT &= ~(DB7 | DB6 | DB5 | DB4); //Sets output low
        P4 -> OUT |= CMD & 0xF0;
        LCD_CTRL(CTRL, CLK);
        P3 -> OUT &= ~EN;   //Nibble 2
        delay_us(30, CLK);  //Nominal 10 ns
    }
    P4 -> OUT &= ~(DB7 | DB6 | DB5 | DB4);
    delay_us(80, CLK); //Nominal 730 ns
}
```

Set DCO Library

```
/*
 *  set_DCO.h
 *  Header file for the DCO frequency change function.
 *
 *  Date: April 6, 2018
 *  Author: Zach Bunce, Garret Maxon
 */

#ifndef SET_DCO_H_
#define SET_DCO_H_

void set_DCO(int);

#endif /* SET_DCO_H_ */
```

```
/*
 *  set_DCO.c
 *  Code file for the DCO frequency change function.
 *
 *  Date: April 6, 2018
 *  Authors: Zach Bunce, Garret Maxon
 */

#include "msp.h"
#define F_1p5_MeHz  15      //Defines various frequency values in almost MHz (10^5)
#define F_3_MeHz    30      //MeHz labels are used to indicate this
#define F_6_MeHz    60      //Blame data type truncation
#define F_12_MeHz   120
#define F_24_MeHz   240
#define F_48_MeHz   480

void set_DCO(int freq)
{
    CS->KEY = CS_KEY_VAL;                                       //Unlocks CS registers
    CS->CTL0 = 0;                                               //Clears CTL0 register

    switch (freq)
    {
    case F_1p5_MeHz:
        CS->CTL0 = CS_CTL0_DCORSEL_0;                           //Sets DC0 to 1.5 MHz
        CS->CTL1 = CS_CTL1_SELA_2 | CS_CTL1_SELS_3 | CS_CTL1_SELM_3; //Sets the clock refs
        break;
    case F_3_MeHz:
        CS->CTL0 = CS_CTL0_DCORSEL_1;                           //Sets DC0 to 3 MHz
        CS->CTL1 = CS_CTL1_SELA_2 | CS_CTL1_SELS_3 | CS_CTL1_SELM_3; //Sets the clock refs
        break;
    case F_6_MeHz:
        CS->CTL0 = CS_CTL0_DCORSEL_2;                           //Sets DC0 to 6 MHz
        CS->CTL1 = CS_CTL1_SELA_2 | CS_CTL1_SELS_3 | CS_CTL1_SELM_3; //Sets the clock refs
        break;
    case F_12_MeHz:
        CS->CTL0 = CS_CTL0_DCORSEL_3;                           //Sets DC0 to 12 MHz
        CS->CTL1 = CS_CTL1_SELA_2 | CS_CTL1_SELS_3 | CS_CTL1_SELM_3; //Sets the clock refs
        break;
    case F_24_MeHz:
        CS->CTL0 = CS_CTL0_DCORSEL_4;                           //Sets DC0 to 24 MHz
        CS->CTL1 = CS_CTL1_SELA_2 | CS_CTL1_SELS_3 | CS_CTL1_SELM_3; //Sets the clock refs
        break;
    case F_48_MeHz:
        // Transition to VCORE Level 1: AM0_LDO --> AM1_LDO
        while ((PCM->CTL1 & PCM_CTL1_PMR_BUSY));
        PCM->CTL0 = PCM_CTL0_KEY_VAL | PCM_CTL0_AMR_1;
        while ((PCM->CTL1 & PCM_CTL1_PMR_BUSY));
```

```
        // Configure Flash wait-state to 1 for both banks 0 & 1
        FLCTL->BANK0_RDCTL = (FLCTL->BANK0_RDCTL
                & ~(FLCTL_BANK0_RDCTL_WAIT_MASK)) | FLCTL_BANK0_RDCTL_WAIT_1;
        FLCTL->BANK1_RDCTL = (FLCTL->BANK0_RDCTL
                & ~(FLCTL_BANK1_RDCTL_WAIT_MASK)) | FLCTL_BANK1_RDCTL_WAIT_1;

        CS->CTL0 = CS_CTL0_DCORSEL_5;                       //Sets DC0 to 48 MHz
        CS->CTL1 = CS->CTL1
                & ~(CS_CTL1_SELM_MASK | CS_CTL1_DIVM_MASK)| CS_CTL1_SELM_3; //Sets MCLK to DCO
        break;
    default:
        break;
    }
    CS->KEY = 0;                                        //Locks CS registers
}
```