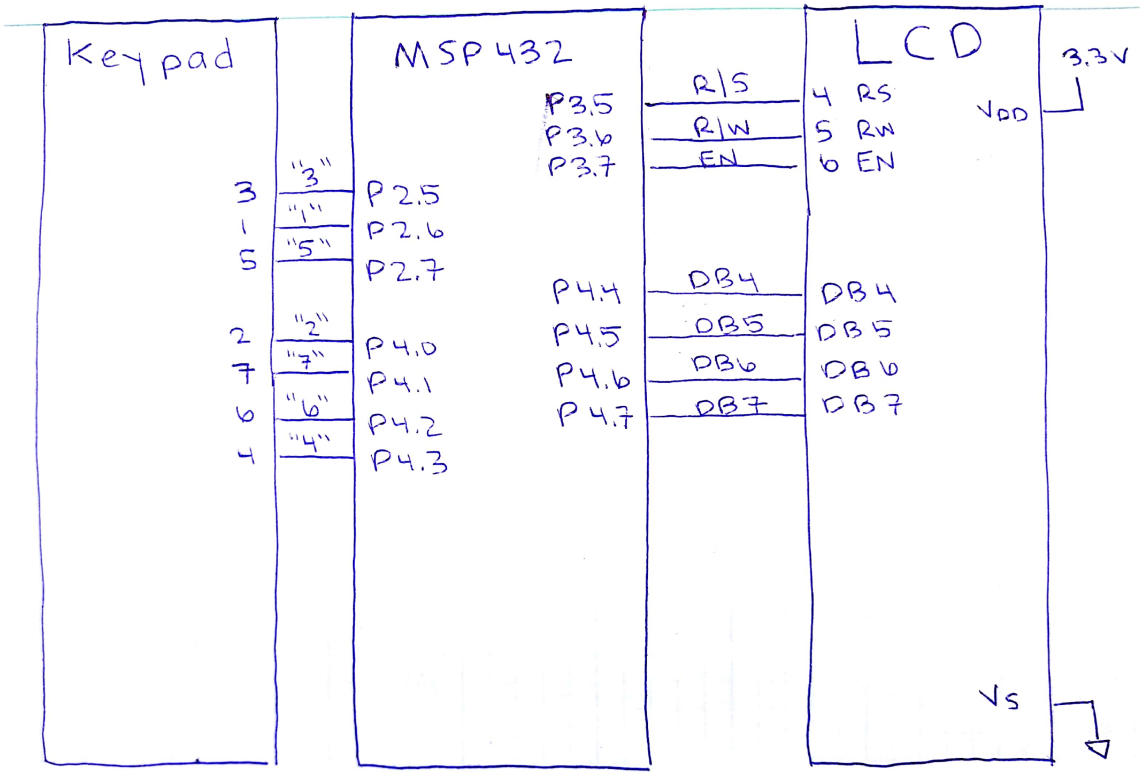
CPE329-03
A4 - Keypad

John Contovasilis
Andes Le

April 17, 2018

1. **https://youtu.be/nYQmfk50dAk**
2. Schematic:



```c
#include "msp.h"
#include "gpio.h"
#include "delay.h"
#include "lcd.h"
#include "keypad.h"


/**
 * main.c
 */

void updateDisplay();

void main(void)
{
    set_DCO(FREQ_24_MHZ);           //set DCO to 24MHZ
    lcdInit();                      //initialize display
    initKeypad();                   //initialize keypad

    while(1){
        delay_ms(50, FREQ_24_MHZ);
        updateDisplay();            //update display with most recent key press every 50
    ms

    }
}

void updateDisplay(){
    sendCommand(0x00);              //clear display
    sendCommand(0x10);
    sendCharacter(key);             //send key character
    sendCharacter(key<<4);
    sendCommand(0x00);              //home display
    sendCommand(0x20);
}
```

Listing 1: main.c source code

```
/*
 * keypad.h
 *
 *  Created on: Apr 16, 2018
 *      Author: Contovasilis
 */

#ifndef KEYPAD_H_
#define KEYPAD_H_

#include "msp.h"
#include "gpio.h"
#include <stdint.h>

#define R1_P            PORT4
#define R2_P            PORT4
#define R3_P            PORT4
#define R4_P            PORT4

#define R1              PIN0
#define R2              PIN1
#define R3              PIN2
#define R4              PIN3

#define C1_P            PORT2
#define C2_P            PORT2
#define C3_P            PORT2

#define C1              PIN5
#define C2              PIN6
#define C3              PIN7

#define RWPORT           R1_P | R2_P | R3_P | R4_P
#define CLPORT           C1_P | C2_P | C3_P
#define RWPINS           R1 | R2 | R3 | R4
#define CLPINS           C1 | C2 | C3

uint8_t initKeypad();
void scanKeypad(uint8_t iflg);

volatile uint8_t key;

#endif /* KEYPAD_H_ */
```

Listing 2: keypad.h source code

```c
/*
 * keypad.c
 *
 *  Created on: Apr 16, 2018
 *      Author: Contovasilis
 */
#include "keypad.h"
#include "interrupt.h"
#include <stdlib.h>

uint8_t initKeypad(){
    gpioStruct gpio;                        //define struct
    gpio.port = CLPORT;                     //select column ports
    gpio.pins = CLPINS;                     //select column pins
    gpio.type = INPUT_PD;                   //initialize GPIOs as inputs with pull-ups
    gpio_init(&gpio);                       //configure struct

    gpio.port = RWPORT;                     //select row ports
    gpio.pins = RWPINS;                     //select row pins
    gpio.type = OUTPUT;                     //initialize GPIOs as outputs
    gpio_init(&gpio);                       //configure struct

    GPIO_set(RWPORT, RWPINS);               //set all row pins high

    initExtInt(CLPORT, CLPINS, RISING);     //initialize external interrupts on
                                            //column pins with rising edge trigger

    enableExtInt(CLPORT, CLPINS);           //enable external interrupts
    initNVIC(PORT2_IRQn);                   //enable NVIC for column port

    __enable_irq();                         //enable global interrupts

}

void scanKeypad(uint8_t iflg){
    uint8_t chrArray[12] = {'1', '4','7','*','2','5','8','0','3','6', '9', '#'};
                                                //array with keypad characters
    uint8_t col = 0;                            //initialize column iterator
    while(col<4){                               //begin iteration loop
        GPIO_clear(RWPORT, (RWPINS) & ~(0x01<<col));    //set all except one row pin low
        if(GPIO_read(CLPORT, CLPINS) & iflg){           //read column that produced the
                                                        //interrupt
            key = chrArray[col + ((iflg>>4) & ~(0x02))];//set key to character pressed
            break;                                      //break if pressed key was found
        }
        col++;                                      //repeat with next row set low
    if
                                                    //pressed key was not found
        GPIO_set(RWPORT, RWPINS);                   //reset all rows high
    }
    GPIO_set(RWPORT, RWPINS);                       //reset all rows high for next
    key
                                                    //interrupt
}

void PORT2_IRQHandler(void)             //ISR handler for port 2
{
    disableExtInt(CLPORT, CLPINS);      //disable column interrupts to avoid multiple
    interrupts
    scanKeypad(readIFG(CLPORT,CLPINS));//find which key in the row was pressed
    enableExtInt(CLPORT, CLPINS);       //re-enable column interrupts
}
```

Listing 3: keypad.c source code

```
/*
 * interrupt.h
 *
 *  Created on: Apr 16, 2018
 *      Author: Contovasilis
 */

#ifndef INTERRUPT_H_
#define INTERRUPT_H_

#include <stdint.h>

#define HAL_IES          (uint32_t)&P1->IES -   (uint32_t)P1
#define HAL_IE           (uint32_t)&P1->IE -  (uint32_t)P1
#define HAL_IFG          (uint32_t)&P1->IFG -  (uint32_t)P1
#define HAL_IV           (uint32_t)&P1->IV -  (uint32_t)P1
#define RISING             0
#define FALLING            1
#define HWREG32(x)        (*((volatile uint32_t *)(x)))

void initExtInt(uint8_t selectedPort, uint32_t pins, uint8_t ies);
void enableExtInt(uint8_t selectedPort, uint32_t pins);
void disableExtInt(uint8_t selectedPort, uint32_t pins);
void writeIFG(uint8_t selectedPort, uint32_t pins);
uint8_t readIFG(uint8_t selectedPort, uint8_t mask);
void initNVIC(uint32_t irqNum);

#endif /* INTERRUPT_H_ */
```

Listing 4: interrupt.h source code

```c
/*
 * interrupt.c
 *
 *  Created on: Apr 16, 2018
 *      Author: Contovasilis
 */

#include "interrupt.h"
#include "msp.h"

static uint32_t GPIO_PORT_TO_BASE[] =
{   0x00,
    (uint32_t)P1,
    (uint32_t)P1+1,
    (uint32_t)P3,
    (uint32_t)P3+1,
    (uint32_t)P5,
    (uint32_t)P5+1,
    (uint32_t)P7,
    (uint32_t)P7+1,
    (uint32_t)P9,
    (uint32_t)P9+1
};

void initExtInt(uint8_t selectedPort, uint32_t pins, uint8_t ies){
    uint32_t baseAddress = GPIO_PORT_TO_BASE[selectedPort];
    if(ies){
        HWREG32(baseAddress + HAL_IES) |= pins;
    }
    else{
        HWREG32(baseAddress + HAL_IES) &= ~pins;
    }
}

void enableExtInt(uint8_t selectedPort, uint32_t pins){
    uint32_t baseAddress = GPIO_PORT_TO_BASE[selectedPort];
    writeIFG(selectedPort, pins);
    HWREG32(baseAddress + HAL_IE) |= pins;
}

void disableExtInt(uint8_t selectedPort, uint32_t pins){
    uint32_t baseAddress = GPIO_PORT_TO_BASE[selectedPort];
    HWREG32(baseAddress + HAL_IE) &= ~pins;
}
void writeIFG(uint8_t selectedPort, uint32_t pins){
    uint32_t baseAddress = GPIO_PORT_TO_BASE[selectedPort];
    HWREG32(baseAddress + HAL_IFG) &= ~pins;
}
uint8_t readIFG(uint8_t selectedPort, uint8_t mask){
    uint32_t baseAddress = GPIO_PORT_TO_BASE[selectedPort];
    return HWREG32(baseAddress + HAL_IFG) & mask;
}

void initNVIC(uint32_t irqNum){
    NVIC->ISER[1] = 1<<((irqNum)&31);
}
```

Listing 5: interrupt.c source code

```
/*
 * gpio.h
 *
 *  Created on: Apr 9, 2018
 *      Author: Contovasilis
 */

#ifndef GPIO_H_
#define GPIO_H_

#include <stdint.h>
#include "msp.h"
#include "lcd.h"

#define PIN0            0x0001
#define PIN1            0x0002
#define PIN2            0x0004
#define PIN3            0x0008
#define PIN4            0x0010
#define PIN5            0x0020
#define PIN6            0x0040
#define PIN7            0x0080

#define HAL_IN          (uint32_t)&P1->IN -   (uint32_t)P1
#define HAL_OUT         (uint32_t)&P1->OUT -  (uint32_t)P1
#define HAL_DIR         (uint32_t)&P1->DIR -  (uint32_t)P1
#define HAL_REN         (uint32_t)&P1->REN -  (uint32_t)P1
#define HAL_DS          (uint32_t)&P1->DS -   (uint32_t)P1
#define HAL_SEL0        (uint32_t)&P1->SEL0 - (uint32_t)P1
#define HAL_SEL1        (uint32_t)&P1->SEL1 - (uint32_t)P1


#define HWREG16(x)      (*((volatile uint16_t *)(x)))


typedef enum GPIO_AF{
    AF1 = 0,
    AF2,
    AF3
}GPIO_AF;

typedef enum GPIO_TYPE{
    OUTPUT=0,
    INPUT,
    INPUT_PU,
    INPUT_PD,
    AF
}GPIO_TYPE;

typedef enum GPIO_PORT{
    PORT1=1,
    PORT2,
    PORT3,
    PORT4,
    PORT5,
    PORT6
}GPIO_PORT;

typedef struct gpioStruct{
    GPIO_PORT port;
    uint32_t pins;
    GPIO_TYPE type;
    GPIO_AF af;
}gpioStruct;

void gpio_init(gpioStruct* gpio);
void GPIO_set(uint_fast8_t selectedPort, uint_fast16_t pins);
void GPIO_clear(uint_fast8_t selectedPort, uint_fast16_t pins);
void GPIO_toggle(uint_fast8_t selectedPort, uint_fast16_t pins);
uint8_t GPIO_read(uint_fast8_t selectedPort, uint_fast16_t pins);

#endif /* GPIO_H_ */
```

Listing 6: gpio.h source code

```c
/*
 * gpio.c
 *
 *  Created on: Apr 9, 2018
 *      Author: Contovasilis
 */

#include "gpio.h"

static uint32_t GPIO_PORT_TO_BASE[] =
{   0x00,
    (uint32_t)P1,
    (uint32_t)P1+1,
    (uint32_t)P3,
    (uint32_t)P3+1,
    (uint32_t)P5,
    (uint32_t)P5+1,
    (uint32_t)P7,
    (uint32_t)P7+1,
    (uint32_t)P9,
    (uint32_t)P9+1
};


void GPIO_set(uint_fast8_t selectedPort, uint_fast16_t pins){
    uint32_t baseAddress = GPIO_PORT_TO_BASE[selectedPort];

    HWREG16(baseAddress + HAL_OUT) |= pins;
}

void GPIO_clear(uint_fast8_t selectedPort, uint_fast16_t pins){
    uint32_t baseAddress = GPIO_PORT_TO_BASE[selectedPort];

    HWREG16(baseAddress + HAL_OUT)  &= ~pins;
}

void GPIO_toggle(uint_fast8_t selectedPort, uint_fast16_t pins){
    uint32_t baseAddress = GPIO_PORT_TO_BASE[selectedPort];

    HWREG16(baseAddress + HAL_OUT)  ^= pins;
}

uint8_t GPIO_read(uint_fast8_t selectedPort, uint_fast16_t pins){
    uint32_t baseAddress = GPIO_PORT_TO_BASE[selectedPort];

    return HWREG16(baseAddress + HAL_IN) & pins;
}

void gpio_init(gpioStruct* gpio){
    uint32_t baseAddress = GPIO_PORT_TO_BASE[gpio -> port];
    switch(gpio -> type){
        case(OUTPUT):
            HWREG16(baseAddress + HAL_SEL0) &= ~(gpio -> pins);
            HWREG16(baseAddress + HAL_SEL1) &= ~(gpio -> pins);
            HWREG16(baseAddress + HAL_DIR) |= gpio -> pins;
            HWREG16(baseAddress + HAL_OUT) &= ~(gpio -> pins);
            break;
        case(INPUT):
            HWREG16(baseAddress + HAL_SEL0) &= ~(gpio -> pins);
            HWREG16(baseAddress + HAL_SEL1) &= ~(gpio -> pins);
            HWREG16(baseAddress + HAL_DIR) &= ~(gpio -> pins);
            HWREG16(baseAddress + HAL_REN) &= ~(gpio -> pins);
            break;
        case(INPUT_PU):
            HWREG16(baseAddress + HAL_SEL0) &= ~(gpio -> pins);
            HWREG16(baseAddress + HAL_SEL1) &= ~(gpio -> pins);
            HWREG16(baseAddress + HAL_DIR) &= ~(gpio -> pins);
            HWREG16(baseAddress + HAL_REN) |= gpio -> pins;
            HWREG16(baseAddress + HAL_OUT) |= gpio-> pins;
            break;
        case(INPUT_PD):
            HWREG16(baseAddress + HAL_SEL0) &= ~(gpio -> pins);
            HWREG16(baseAddress + HAL_SEL1) &= ~(gpio -> pins);
            HWREG16(baseAddress + HAL_DIR) &= ~(gpio -> pins);
            HWREG16(baseAddress + HAL_REN) |= gpio -> pins;
            HWREG16(baseAddress + HAL_OUT) &= ~(gpio -> pins);
```

```
            break;
        case(AF):
            switch(gpio -> af){
                case(AF1):
                    HWREG16(baseAddress + HAL_SEL0) &= gpio -> pins;
                    HWREG16(baseAddress + HAL_SEL1) &= ~(gpio -> pins);
                    break;
                case(AF2):
                    HWREG16(baseAddress + HAL_SEL0) &= ~(gpio -> pins);
                    HWREG16(baseAddress + HAL_SEL1) &= gpio -> pins;
                    break;
                case(AF3):
                    HWREG16(baseAddress + HAL_SEL0) &= gpio -> pins;
                    HWREG16(baseAddress + HAL_SEL1) &= gpio -> pins;
                    break;
            }
    }
}
```

Listing 7: gpio.c source code

```
/*
 * lcd.h
 *
 *  Created on: Apr 9, 2018
 *      Author: Contovasilis
 */

#ifndef LCD_H_
#define LCD_H_

#include <stdint.h>
#include <stdio.h>


#define CLEAR       0x01
#define HOME        0x02
#define ENTRY       0x04
#define DISPLAY     0x08
#define CURSOR      0x10
#define FUNCTION    0x20
#define CGRAM       0x40
#define DDRAM       0x80

#define DPORT   PORT4

#define DB0     PIN0
#define DB1     PIN1
#define DB2     PIN2
#define DB3     PIN3
#define DB4     PIN4
#define DB5     PIN5
#define DB6     PIN6
#define DB7     PIN7

#define CPORT   PORT3

#define ENA     PIN5
#define RW      PIN6
#define RS      PIN7

void sendCommand(uint8_t cmd);
void sendCharacter(uint8_t chr);
void sendString(char *str);
void sendInteger(int IntegerToDisplay);
void lcdInit();

#endif /* LCD_H_ */
```

Listing 8: lcd.h source code

```c
/*
 * lcd.c
 *
 *  Created on: Apr 9, 2018
 *      Author: Contovasilis
 */

#include "lcd.h"
#include "delay.h"
#include "gpio.h"


void lcdInit(){
    delay_ms(50, FREQ_24_MHZ);      //startup delay

    gpioStruct gpio;

    gpio.port = CPORT;
    gpio.pins = RS | RW | ENA;
    gpio.type = OUTPUT;
    gpio_init(&gpio);


    gpio.port = DPORT;
    gpio.pins = DB4 | DB5 | DB6 | DB7;
    gpio.type = OUTPUT;
    gpio_init(&gpio);

    sendCommand(0x20);              //set LDC in 4 bit mode

    sendCommand(0x20);              //5*8 Font, 1 line display
    sendCommand(0x00);

    sendCommand(0x00);              //Display ON, no cursor, no blink
    sendCommand(0xC0);

    sendCommand(0x00);              //Clear LCD
    sendCommand(0x10);

    sendCommand(0x00);              //Increment cursor, no shift
    sendCommand(0x60);
}

void sendCommand(uint8_t cmd){
    GPIO_clear(CPORT, RS);           //clear RS
    GPIO_clear(CPORT, RW);           //clear RW
    GPIO_set(CPORT, ENA);           //set ENA

    GPIO_set(DPORT, cmd);           //send command byte or nibble
    delay_ms(5, FREQ_24_MHZ);       //wait
    GPIO_clear(CPORT, ENA);         //clear ENA
    GPIO_clear(DPORT, 0xF0);        //clear data port
}


void sendCharacter(uint8_t chr){
    GPIO_set(CPORT, RS);             //set RS
    GPIO_clear(CPORT, RW);           //clear RW
    GPIO_set(CPORT, ENA);           //set ENA

    GPIO_set(DPORT, chr);           //send character byte or nibble
    delay_ms(5, FREQ_24_MHZ);       //wait
    GPIO_clear(CPORT, ENA);         //clear ENA
    GPIO_clear(CPORT, RS);           //clear RS
    GPIO_clear(DPORT, 0xF0);        //clear data port
}

void sendString(char *str)
{
    while(*str > 0)
    {
        sendCharacter(*str);        //send first character
        sendCharacter(*str<<4);
        *str++;
    }
}
```

```
77
void sendInteger(int IntegerToDisplay)
79 {
    char StringToDisplay[16];
81  sprintf(StringToDisplay, "%d", IntegerToDisplay);
    sendString(StringToDisplay);
83 }
```

Listing 9: lcd.c source code

```
1  /*
 * delay.h
3  *
 *  Created on: Apr 4, 2018
5  *      Author: Contovasilis
 */
7
#ifndef DELAY_H_
9 #define DELAY_H_

11 #include <stdint.h>

13 #define FREQ_1_5_MHZ    (uint16_t)15
#define FREQ_3_MHZ      (uint16_t)30
15 #define FREQ_6_MHZ      (uint16_t)60
#define FREQ_12_MHZ     (uint16_t)120
17 #define FREQ_24_MHZ     (uint16_t)240
#define FREQ_48_MHZ     (uint16_t)480
19


21 inline delay_ms(uint32_t time, uint16_t freq);
inline delay_us(uint32_t time, uint16_t freq);
23 void set_DCO(uint16_t freq);

25 #endif /* DELAY_H_ */
```

Listing 10: delay.h source code

```
/*
 * delay.c
 *
 *  Created on: Apr 4, 2018
 *      Author: Contovasilis
 */
#include "delay.h"
#include "msp.h"

inline delay_ms(uint32_t time, uint16_t freq){
    uint32_t ticks = freq * time * 10;

    while(ticks){
        ticks--;
    }
}

inline delay_us(uint32_t time, uint16_t freq){

    uint32_t ticks = ((freq * time) / 100) - 2;

    while(ticks){
        ticks--;
    }

}

void set_DCO(uint16_t freq){

    CS->KEY = CS_KEY_VAL;

    switch(freq)
            {
            case FREQ_1_5_MHZ:
                CS->CTL0 = CS_CTL0_DCORSEL_0;
                break;
            case FREQ_3_MHZ:
                CS->CTL0 = CS_CTL0_DCORSEL_1;
                break;
            case FREQ_6_MHZ:
                CS->CTL0 = CS_CTL0_DCORSEL_2;
                break;
            case FREQ_12_MHZ:
                CS->CTL0 = CS_CTL0_DCORSEL_3;
                break;
            case FREQ_24_MHZ:
                CS->CTL0 = CS_CTL0_DCORSEL_4;
                break;
            case FREQ_48_MHZ:
                while ((PCM->CTL1 & PCM_CTL1_PMR_BUSY));
                 PCM->CTL0 = PCM_CTL0_KEY_VAL | PCM_CTL0_AMR_1;
                while ((PCM->CTL1 & PCM_CTL1_PMR_BUSY));
                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;
                break;
            }

    CS->CTL1 = CS_CTL1_SELM__DCOCLK;
    CS->KEY = 0;
}
```

Listing 11: delay.c source code