

```
/*  
* File: calculator.c  
* Author: Phil Glazzard  
*  
* Created on 22 November 2016, 11:03  
* ver 1.01  
*/
```

```
// PIC16F690 Configuration Bit Settings
```

```
// microcontroller keypad pin lcd
```

```
// RA0 out 19 RS
```

```
// RA1 out 18 E
```

```
// RB4 out 13 D4
```

```
// RB5 out 12 D5
```

```
// RB6 out 11 D6
```

```
// RB7 out 10 D7
```

```
//
```

```
// RC7 out ROW1 9
```

```
// RC6 out ROW2 8
```

```
// RC5 out ROW3 5
```

```
// RC4 out ROW4 6
```

```
// RC3 in COLUMN 1 7
```

```
// RC2 in COLUMN 2 14
```

```
// RC1 in COLUMN 3 15
```

```
// RC0 in COLUMN 4 16
```

```
// 'C' source line config statements
```

```

#include <xc.h>

// CONFIG

#pragma config FOSC = INTRCIO // Oscillator Selection bits (INTOSCIO oscillator: I/O function on
RA4/OSC2/CLKOUT pin, I/O function on RA5/OSC1/CLKIN)

#pragma config WDTE = OFF // Watchdog Timer Enable bit (WDT disabled and can be enabled by
SWDTEN bit of the WDTCON register)

#pragma config PWRTE = OFF // Power-up Timer Enable bit (PWRT disabled)

#pragma config MCLRE = ON // MCLR Pin Function Select bit (MCLR pin function is MCLR)

#pragma config CP = OFF // Code Protection bit (Program memory code protection is disabled)

#pragma config CPD = OFF // Data Code Protection bit (Data memory code protection is
disabled)

#pragma config BOREN = OFF // Brown-out Reset Selection bits (BOR disabled)

#pragma config IESO = OFF // Internal External Switchover bit (Internal External Switchover mode
is disabled)

#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is
disabled)

#define _XTAL_FREQ 4000000 // internal clock frequency 4MHz

#define RS PORTAbits.RA0 // RS = 1 = data register, 0 = Instruction register pin 19

#define E PORTAbits.RA1 // enable write pin 18

#define D4 PORTBbits.RB4 // D7 MSB ...D4 LSB of 4 bit LCD data bus

#define D5 PORTBbits.RB5

#define D6 PORTBbits.RB6

#define D7 PORTBbits.RB7

#define ROW1 PORTCbits.RC7

#define ROW2 PORTCbits.RC6

#define ROW3 PORTCbits.RC5

```

```

#define ROW4 PORTCbits.RC4

#define COL1 PORTCbits.RC3

#define COL2 PORTCbits.RC2

#define COL3 PORTCbits.RC1

#define COL4 PORTCbits.RC0

char key, num1, op, num2 = 0;

char num = 0;

int result = 0;

void init_ports()
{
    OSCCONbits.IRCF2 = 1;    // 4MHz clock

    OSCCONbits.IRCF1 = 1;

    OSCCONbits.IRCF0 = 0;

    TRISCbits.TRISC7 = 0;    // keypad outputs - row 1

    TRISCbits.TRISC6 = 0;    // keypad outputs - row 2

    TRISCbits.TRISC5 = 0;    // keypad outputs - row 3

    TRISCbits.TRISC4 = 0;    // keypad outputs - row 4

    TRISCbits.TRISC3 = 1;    // keypad inputs - col 1

    TRISCbits.TRISC2 = 1;    // keypad inputs - col 2

    TRISCbits.TRISC1 = 1;    // keypad inputs - col 3

    TRISCbits.TRISC0 = 1;    // keypad inputs - col 4

    TRISAbits.TRISA0 = 0;    // RS output to LCD

    TRISAbits.TRISA1 = 0;    // E output to LCD

```

```

TRISBbits.TRISB4 = 0;    // D4 LCD
TRISBbits.TRISB5 = 0;    // D5 LCD
TRISBbits.TRISB6 = 0;    // D6 LCD
TRISBbits.TRISB7 = 0;    // D7 LCD

ANSEL = 0x00;           // ADC module off
ANSELH = 0x00;          //
CM1CON0 = 0x00;         // comparators off
CM2CON0 = 0x00;
PORTB = 0x00;
PORTC = 0x00;
}

void pulseE()
{
    E = 1;    // E = RA1 pin 18
    __delay_ms(1); // E latch pulse to get commands into LCD
    E = 0;
}

/* function void display_on_lcd (char data)
* This function receives a single 8-bit argument from the calling function
* and splits this into two 4-bit nibbles (hi_nib and lo_nib) which are then sent by
* PORTB (RB7 RB6 RB5 RB4) to the 4-bit data interface on the LCD module
*/
void display_on_lcd (char data) // char data is an 8-bit value which will be split

```

```

{
    // into two 4-bit nibbles (hi_nib and lo_nib)
    char hi_nib, lo_nib;    // hi_nib is most significant nibble
    hi_nib = data & 0xf0;  // split data into high nibble
    lo_nib = (data & 0x0f)<<4; // split data into low nibble, and move four bits left
    RS = 1;                // lcd data mode
    PORTB = hi_nib;
    pulseE();
    PORTB = lo_nib;
    pulseE();
}

```

```

position_cursor (char xy)

```

```

{
    char hi_nib, lo_nib;    // hi_nib is most significant nibble
    hi_nib = xy & 0xf0;    // split data into high nibble
    lo_nib = (xy & 0x0f)<<4; // split data into low nibble, and move four bits left
    RS = 0;                // lcd command mode
    PORTB = hi_nib;
    pulseE();
    PORTB = lo_nib;
    pulseE();
}

```

```

void splash_screen()

```

```

{
    char splash[] = "phils calculator"; // splash screen message
    char *p;                            // define pointer to point to first character of string
    p = &splash[0];                      // point to first character
}

```

```

while (*p != '\0')    // loop until null character is found (end of string)
{
display_on_lcd(*p);    // send data found at *p to display_on_lcd function
p = p + 1;            // point to the next character
}
}

```

```

void init_lcd()
{
__delay_ms(100);    // power on, wait for 100ms
RS = 0;            // LCD is in command mode
PORTB = 0b00110000; // RS=0, D7 = 0, D6 = 0, D5 = 1, D4 = 1 - 8 bit mode (1)
                    // 8 bit function set (lower 4 bits are irrelevant)
__delay_ms(5);
pulseE();

PORTB = 0b00110000; // RS=0, D7 = 0, D6 = 0, D5 = 1, D4 = 1 - 8 bit mode (2)
                    // 8 bit function set (lower 4 bits are irrelevant)
__delay_ms(1);
pulseE();

PORTB = 0b00110000; // RS=0, D7 = 0, D6 = 0, D5 = 1, D4 = 1 - 8 bit mode (3)
                    // 8 bit function set (lower 4 bits are irrelevant)
__delay_ms(1);
pulseE();

PORTB = 0b00100000; // RS=0, D7 = 0, D6 = 0, D5 = 1, D4 = 0 - 4 bit mode (4)

```

```

        //initial function set to change from 8 bit interface to 4 bit
        // lower 4 bits are irrelevant

__delay_ms(1);
pulseE();

PORTB = 0b00100000; // RS=0, D7 = 0, D6 = 0, D5 = 1, D4 = 0 - 4 bit mode (5a)
__delay_ms(1); // two nibble 4 bit functions set, send 0010 first then 1100
pulseE(); // (1100 = N F **) N = number of LCD lines (0 = 1, 1 = 2 lines)
        // F = font size (0 = small, 1 = big)

PORTB = 0b11000000; // RS=0, D7 = 1, D6 = 1, D5 = 0, D4 = 0 - 4 bit mode (5b)
__delay_ms(1); // see 5a above for description
pulseE();

PORTB = 0b11000000; // RS=0, D7 = 1, D6 = 1, D5 = 0, D4 = 0 - 4 bit mode (6a)
__delay_ms(1); // display ON/ OFF control
pulseE();

PORTB = 0b10000000; // RS=0, D7 = 1, D6 = 0, D5 = 0, D4 = 0 - 4 bit mode (6b)
__delay_ms(1);
pulseE();

PORTB = 0b00000000; // RS=0, D7 = 0, D6 = 0, D5 = 0, D4 = 0 - 4 bit mode (7a)
__delay_ms(1); // clear display control
pulseE();

PORTB = 0b00010000; // RS=0, D7 = 1, D6 = 1, D5 = 0, D4 = 1 - 4 bit mode (7b)

```

```

__delay_ms(1);
pulseE();

PORTB = 0b00000000; // RS=0, D7 = 1, D6 = 1, D5 = 1, D4 = 0 - 4 bit mode (8a)
__delay_ms(1); //entry mode set first nibble
pulseE();

PORTB = 0b01100000; // RS=0, D7 = 1, D6 = 1, D5 = 1, D4 = 0 - 4 bit mode (8b)
__delay_ms(1); // entry mode set second nibble - I/D S as required
pulseE();

// END OF INITIALISATION

//*****
****

PORTB = 0b00000000; // RS=0, D7 = 0, D6 = 0, D5 = 0, D4 = 0 - 4 bit mode
__delay_ms(1); // display ON/OFF control (9a)
pulseE();

PORTB = 0b11110000; // RS=0, D7 = 1, D6 = 1, D5 = 0, D4 = 0 - 4 bit mode (9b)
__delay_ms(1); // D (display) = 1(ON), C (cursor) and B (blinking cursor) as required
pulseE();
}

/* function char read_keypad()
* This function scans the keypad for a number or character, and then returns that
* number or character to the calling function
*/
char read_keypad()

```



```
{
```

```
switch (PORTC)
```

```
{
```

```
case 0b10001000: // RC7 = 1, RC3 = 1
```

```
key = '1';
```

```
display_on_lcd(key);
```

```
num = key & 0x0f; // number pressed = 1
```

```
while(RC3 == 1); // wait for button to be released
```

```
break; // exit the switch statement
```

```
case 0b10000100: // RC7 = 1, RC2 = 1
```

```
key = '2';
```

```
display_on_lcd(key); // latch the 4-bit data in the LCD
```

```
num = key & 0x0f; // number pressed = 2
```

```
while(RC2 == 1); // wait for button to be released
```

```
break;
```

```
case 0b10000010: // RC7 = 1, RC1 = 1
```

```
key = '3';
```

```
display_on_lcd(key);
```

```
num = key & 0x0f; // number pressed = 3
```

```
while(RC1 == 1); // wait for button to be released
```

```
break;
```

```
case 0b10000001: // RC7 = 1, RC0 = 1
```

```
key = '+';
```

```
display_on_lcd(key);
```

```

num = key; // number pressed = +
//num = key-0x30; // number pressed = +
while(RC0 == 1); // wait for button to be released
break;

case 0b01001000: // RC6 = 1, RC3 = 1
    key = '4';
    display_on_lcd(key);
    num = key & 0x0f; // number pressed = 4
    while(RC3 == 1); // wait for button to be released
    break;

case 0b01000100: // RC6 = 1, RC2 = 1
    key = '5';
    display_on_lcd(key);
    num = key & 0x0f; // number pressed = 5
    while(RC2 == 1); // wait for button to be released
    break;

case 0b01000010: // RC6 = 1, RC1 = 1
    key = '6';
    display_on_lcd(key);
    num = key & 0x0f; // number pressed = 6
    while(RC1 == 1); // wait for button to be released
    break;

case 0b01000001: // RC6 = 1, RC0 = 1

```

```
key = '-';  
display_on_lcd(key);  
num = key;  
while(RC0 == 1); // wait for button to be released  
break;  
  
case 0b00101000: // RC5 = 1, RC3 = 1  
key = '7';  
display_on_lcd(key);  
num = key & 0x0f; // number pressed = 7  
while(RC3 == 1); // wait for button to be released  
break;  
  
case 0b00100100: // RC5 = 1, RC2 = 1  
key = '8';  
display_on_lcd(key);  
num = key & 0x0f; // number pressed = 8  
while(RC2 == 1); // wait for button to be released  
break;  
  
case 0b00100010: // RC5 = 1, RC1 = 1  
key = '9';  
display_on_lcd(key);  
num = key & 0x0f; // number pressed = 9  
while(RC1 == 1); // wait for button to be released  
break;
```

```
case 0b00100001: // RC5 = 1, RC0 = 1

    key = '*';

    display_on_lcd(key);

    num = key;

    while(RC0 == 1); // wait for button to be released

    break;
```

```
case 0b00011000: // RC5 = 1, RC3 = 1

    key = 'c';

    display_on_lcd(key);

    num = key;

    while(RC3 == 1); // wait for button to be released

    break;
```

```
case 0b00010100: // RC5 = 1, RC2 = 1

    key = '0';

    display_on_lcd(key);

    num = key & 0x0f; // number pressed = 0

    while(RC2 == 1); // wait for button to be released

    break;
```

```
case 0b00010010: // RC5 = 1, RC1 = 1

    key = '=';

    display_on_lcd(key);

    num = key;

    while(RC1 == 1); // wait for button to be released

    break;
```

```

    case 0b00010001:    // RC5 = 1, RC0 = 1

        key = '/';

        display_on_lcd(key);

        num = key;

        while(RC0 == 1); // wait for button to be released

        break;

    default:

        ;

    }

    return num;
}

void main(void)
{
    char i;

    int symbol[4];

    init_ports();

    init_lcd();

    splash_screen();

    position_cursor (0xc0); // cursor starts at first position of second line of LCD

    for (i = 0; i < 4; i++)
    {
        while(num == 0)
        {

```

```
PORTC = 0x80;      // check row1 of keypad for button press
read_keypad();    // store key pressed in num1
```

```
PORTC = 0x40;      // check row1 of keypad for button press
read_keypad();    // store key pressed in num1
```

```
PORTC = 0x20;      // check row1 of keypad for button press
read_keypad();    // store key pressed in num1
```

```
PORTC = 0x10;      // check row1 of keypad for button press
read_keypad();    // store key pressed in num1
```

```
}
```

```
symbol[i] = num;
```

```
num = 0;
```

```
}
```

```
if(symbol[1] == '+' && symbol[3] == '=') // add two numbers,
```

```
{ // acsii characters in quotes
```

```
    result = symbol[0] + symbol[2];
```

```
    display_on_lcd(result+0x30);
```

```
    __delay_ms(5000);
```

```
}
```

```
if(symbol[1] == '-' && symbol[3] == '=') // subtract two numbers
```

```
{
```

```
    result = symbol[0] - symbol[2];
```

```
display_on_lcd(result+0x30);
__delay_ms(5000);
}

if(symbol[1] == '*' && symbol[3] == '=') // multiply two numbers
{
    result = symbol[0] * symbol[2];
    display_on_lcd(result+0x30);
    __delay_ms(5000);
}

if(symbol[1] == '/' && symbol[3] == '=') // divide two numbers
{
    result = (symbol[0] / (symbol[2]));
    display_on_lcd (result+0x30);
    __delay_ms(5000);
}
}
```