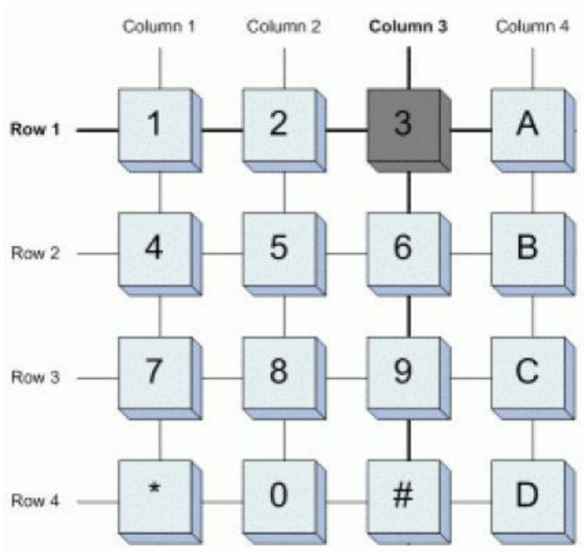
PROJECT 17

INTERFACE WITH 4x4keypad and 2x16 LCD

This 16-button keypad provides a useful human interface component for micro controller projects. Convenient adhesive backing provides a simple way to mount the keypad in a variety of applications. Matrix keypads use a combination of four rows and four columns to provide button states to the host device, typically a micro-controller. Underneath each key is a push button, with one end connected to one row, and the other end connected to one column.



In order for the micro-controller to determine which button is pressed, it first needs to pull each of the four columns either low or high one at a time, and then poll the states of the four rows. Depending on the states of the rows, the micro controller can tell which button is pressed. You can download the Keypad 4x4 user manual from here.

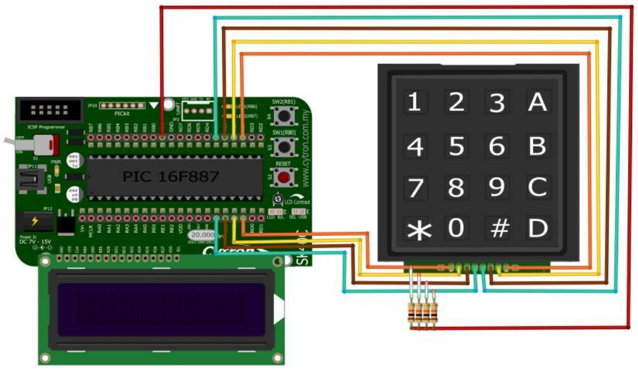
In this tutorial, you will learn on how to display the character on LCD by pressing the 4x4 keypad button.

COMPONENT NEEDED

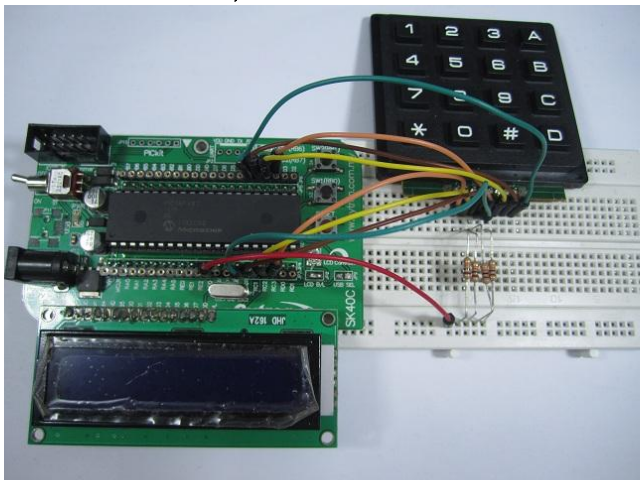
	1 x LCD (2x16)- Blue Backlight
	1 x Keypad 4x4
	4 x Resistor 5W 5% 10K
	Male to Male Jumper Wires

CONNECTION

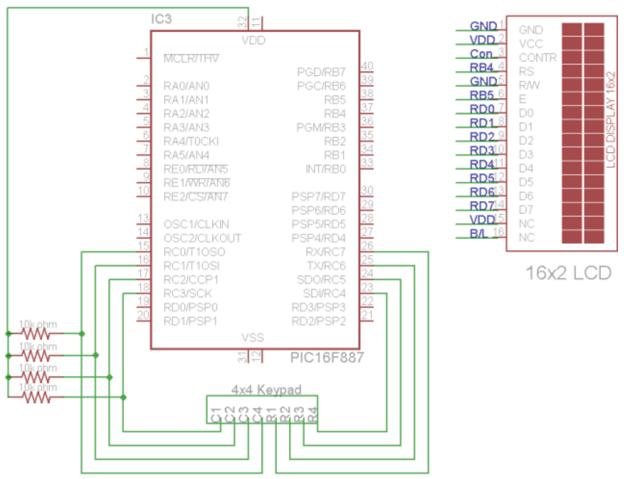
Connection in between SK40C and 4x4 keypad are only required 9 wires which are 5V (VDD) and Port C0-C7. Additional 4 of 10k ohm resistors were used to pull out the 4 columns of the keypad…



a) Circuit connection



b) Actual circuit connection



c) Schematic

ADDITIONAL INFORMATION

4x4 Keypad pin can directly connect to microcontroller or keypad decoder IC for decodes purpose. However, DIY scanning of keypad consumes a lot of understanding in programming and more program space. The better way to interface a keypad to the PIC is to use a keypad encoder in between the keypad and the microcontroller. One of the Keypad Encoder is the MMC74C922. Simply hook up the entire row and the column pins directly to the encoder and it will output a HIGH pulse on the Data Available pin whenever a key is pressed. After that, the encoder will output 4 bits of data to the PIC.

Features

- *Ultra-thin design*
- *Adhesive backing*
- *Excellent price/performance ratio*
- *Easy interface to any microcontroller*
- *Example programs provided for the BASIC*

Key Specifications

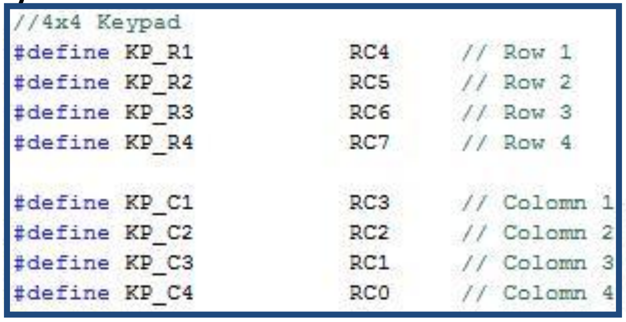
- *Maximum Rating: 24 VDC, 30 mA*
- *Interface: 8-pin access to 4x4 matrix*
- *Operating temperature: 32 to 122 °F(0 to 50°C)*
- *Dimensions:64x62x8 mm*

Application Ideas

- *Security systems*
- *Menu selection*
- *Data entry for embedded systems*

CODE OVERVIEW

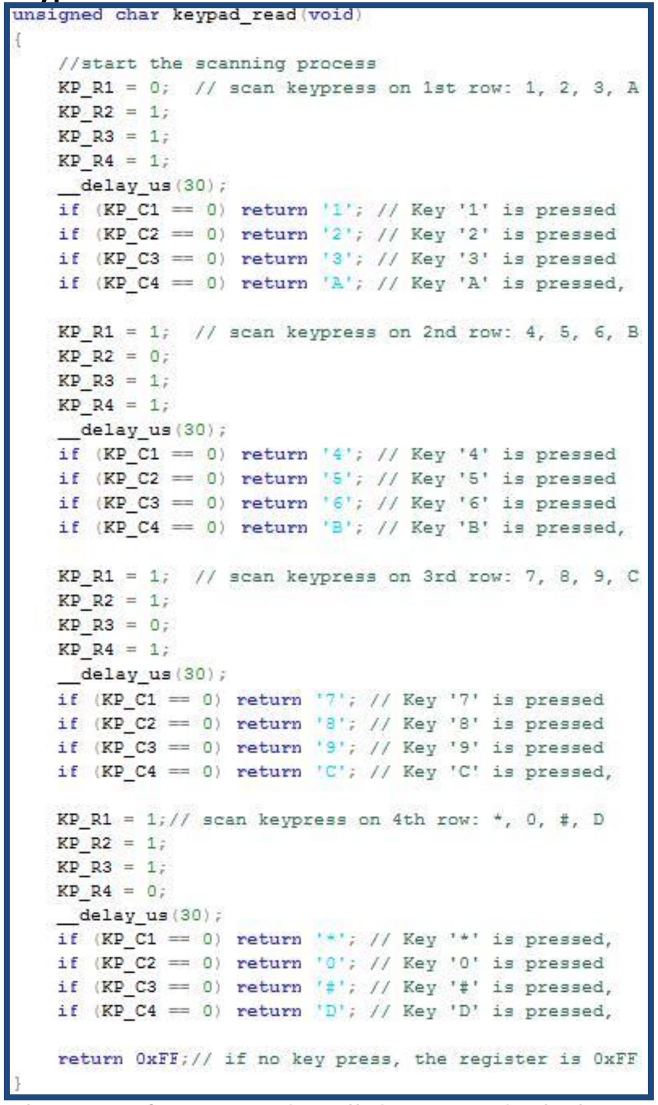
system.h



```
//4x4 Keypad
#define KP_R1          RC4      // Row 1
#define KP_R2          RC5      // Row 2
#define KP_R3          RC6      // Row 3
#define KP_R4          RC7      // Row 4

#define KP_C1          RC3      // Colomn 1
#define KP_C2          RC2      // Colomn 2
#define KP_C3          RC1      // Colomn 3
#define KP_C4          RC0      // Colomn 4
```

In this part of the source code, the pins which were connected to the columns and the rows of the 4x4 keypad were initialized at here.

keypad.c



```
unsigned char keypad_read(void)
{
    //start the scanning process
    KP_R1 = 0;  // scan keypress on 1st row: 1, 2, 3, A
    KP_R2 = 1;
    KP_R3 = 1;
    KP_R4 = 1;
    __delay_us(30);
    if (KP_C1 == 0) return '1'; // Key '1' is pressed
    if (KP_C2 == 0) return '2'; // Key '2' is pressed
    if (KP_C3 == 0) return '3'; // Key '3' is pressed
    if (KP_C4 == 0) return 'A'; // Key 'A' is pressed,

    KP_R1 = 1;  // scan keypress on 2nd row: 4, 5, 6, B
    KP_R2 = 0;
    KP_R3 = 1;
    KP_R4 = 1;
    __delay_us(30);
    if (KP_C1 == 0) return '4'; // Key '4' is pressed
    if (KP_C2 == 0) return '5'; // Key '5' is pressed
    if (KP_C3 == 0) return '6'; // Key '6' is pressed
    if (KP_C4 == 0) return 'B'; // Key 'B' is pressed,

    KP_R1 = 1;  // scan keypress on 3rd row: 7, 8, 9, C
    KP_R2 = 1;
    KP_R3 = 0;
    KP_R4 = 1;
    __delay_us(30);
    if (KP_C1 == 0) return '7'; // Key '7' is pressed
    if (KP_C2 == 0) return '8'; // Key '8' is pressed
    if (KP_C3 == 0) return '9'; // Key '9' is pressed
    if (KP_C4 == 0) return 'C'; // Key 'C' is pressed,

    KP_R1 = 1;// scan keypress on 4th row: *, 0, #, D
    KP_R2 = 1;
    KP_R3 = 1;
    KP_R4 = 0;
    __delay_us(30);
    if (KP_C1 == 0) return '*'; // Key '*' is pressed,
    if (KP_C2 == 0) return '0'; // Key '0' is pressed
    if (KP_C3 == 0) return '#'; // Key '#' is pressed,
    if (KP_C4 == 0) return 'D'; // Key 'D' is pressed,

    return 0xFF;// if no key press, the register is 0xFF
}
```

This part of source code will determined which 4x4 keypad button being pressed.

e.g:

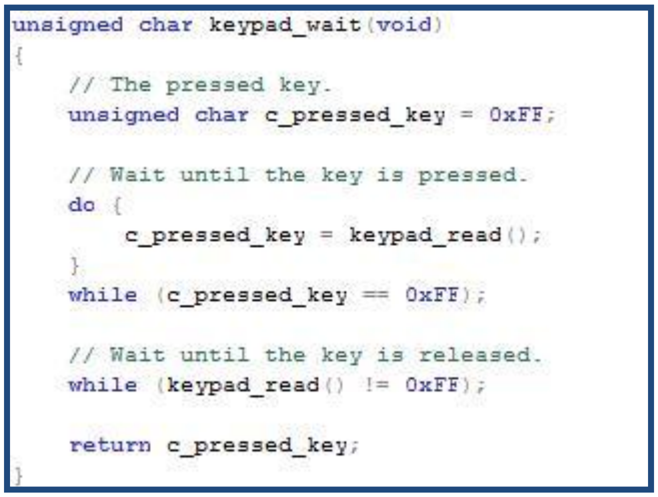
KP_R1 = 0;
KP_R2 = 1;
KP_R3 = 1;
KP_R4 = 1;
__delay_us(30);

It will test whether the 1st row of the keypad being pressed or not. If not, it will goto to 2nd If the button of the 1st row was being pressed, it will test which location is the column.

if (KP_C1 == 0) return '1';
if (KP_C2 == 0) return '2';
if (KP_C3 == 0) return '3';
if (KP_C4 == 0) return 'A';

If the 1st column, it will return 1 in ASCII form. If it was 2nd column, 2 in ASCII form being return. Same go to column 3 and 4.

return 0xFF;
If the keypad does not being pressed, this function will return 0xFF to the function which will used it.



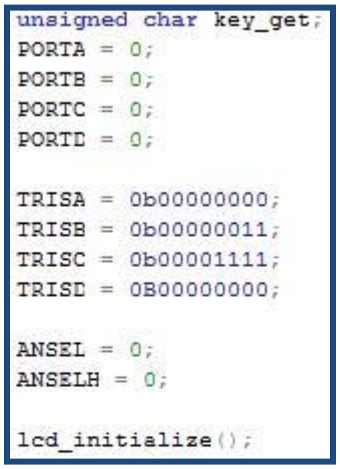
```
unsigned char keypad_wait(void)
{
    // The pressed key.
    unsigned char c_pressed_key = 0xFF;

    // Wait until the key is pressed.
    do {
        c_pressed_key = keypad_read();
    }
    while (c_pressed_key == 0xFF);

    // Wait until the key is released.
    while (keypad_read() != 0xFF);

    return c_pressed_key;
}
```

unsigned char keypad_wait();
This function will test whether the keypad button being pressed or not.

do{c_pressed_key = keypad_read();}while (c_pressed_key == 0xFF);
If the keypad_read() function does not return anything(return 0xFF only) to the c_pressed_key, it will wait at here until it accept one of the 16 characters on the keypad from keypad_read().

return c_pressed_key;
Return the accepted data(character in ASCII form) back to main function.

MAIN PROGRAM

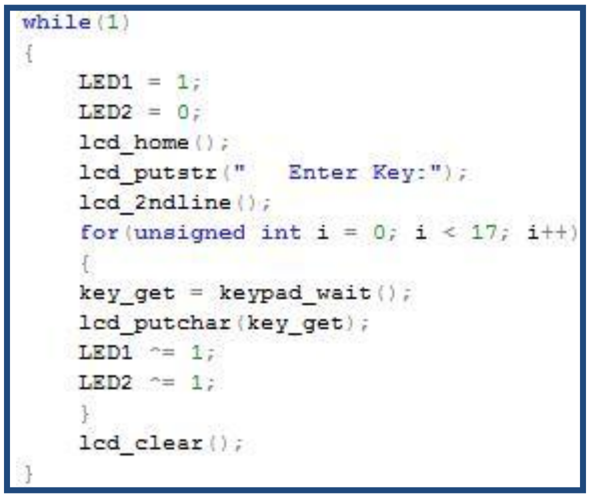


```
unsigned char key_get;
PORTA = 0;
PORTB = 0;
PORTC = 0;
PORTD = 0;

TRISA = 0b00000000;
TRISB = 0b00000011;
TRISC = 0b00001111;
TRISD = 0B00000000;

ANSEL = 0;
ANSELH = 0;

lcd_initialize();
```

unsigned char key_get;
It is an unsigned char variable used to store the data get from the 4x4 keypad.

TRISC = 0b00001111;
Initialize the PORTC<7:4> as output, PORTC<3:0> as input

lcd_initialize();
*Please refer here for the **SK40C** with **lcd** tutorial.*



```
while(1)
{
    LED1 = 1;
    LED2 = 0;
    lcd_home();
    lcd_putstr("   Enter Key:");
    lcd_2ndline();
    for(unsigned int i = 0; i < 17; i++)
    {
    key_get = keypad_wait();
    lcd_putchar(key_get);
    LED1 ^= 1;
    LED2 ^= 1;
    }
    lcd_clear();
}
```

while(1)
While loop used in here in order the micro-controller can always read the data from the keypad

LED1 = 1;
LED2 = 0;
Set the LED1 and LED2 light up and off at first respectively.

lcd_putstr("   Enter Key:");
Display Enter Key on the first row of the 16x2 lcd

lcd_2ndline();
Go to 2nd row of the 16x2 lcd

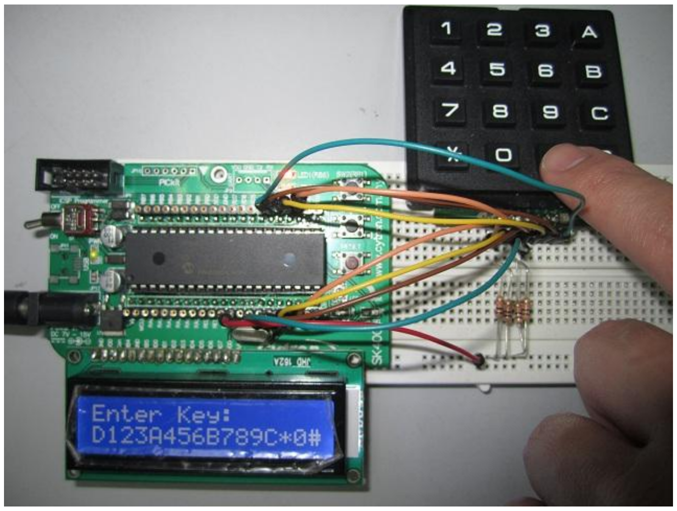
for(unsigned int i = 0; i < 17; i++)
This loop used in order for the LCD to displayed 16 characters

key_get = keypad_wait();
The variable key_get will store the data from keypad_wait() function.

LED1 ^= 1;
LED2 ^= 1;
The led will be toggle each time for the button of the keypad being pressed. Either LED1 on and LED2 off or vice versa.

lcd_clear();
Clear the LCD

RESULT



The LCD will display the character of the keypad as you pressed the button. It will showed 16 characters and if you pressed the keypad button furthermore, the 16 characters will be clear and started again from the first column of the 2nd row on the16x2 LCD.