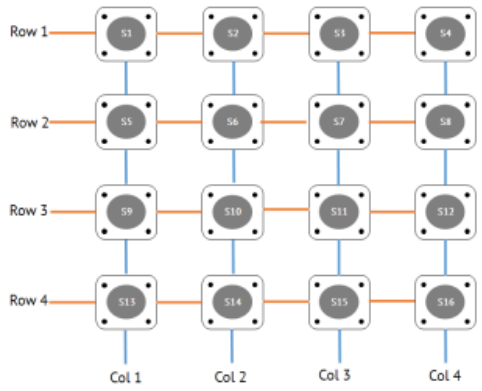
Keypad Interfacing with PIC16F877A

Keypad is an input device and it is a vital part of the user interface in many consumer devices and instruments such as telephones, remotes calculators, etc. A 4*4 matrix keypad consists of 16 keys which are arranged into rows and columns such that it leaves only 8 pins for interfacing with the devices. The Matrix keypad interfaces 16 keys to the microcontroller port using only 8 pins and a suitable firmware scans the pins for detecting the keypress. In this tutorial, we are discussing the keypad interfacing with PIC16F877A microcontroller and the firmware implementation to identify the keypress in the 4*4 keypad matrix.

Keypad Hardware

The hardware part of the keypad before writing the firmware as the firmware logic depends on the hardware connections. Similar to the General convention we will consider the horizontal connections as rows and the vertical connections as columns. The 4*4 matrix keypad is interfaced with one of the 8-bit ports of the microcontroller. These types of keypad modules consist of 4 ROWS and 4 COLUMNS. The 16 keys can be arranged into rows and columns such that we can identify each key as a combination of row and column address similar to matrix element.

The LSB of the microcontroller port will be considered as rows and MSB as columns. The MSB pins will be taken as inputs and LSB pins as output. The input pins can be connected in a pull-up and pull-down mode. The default state of the input pins connected in pull-up mode will be high and similarly, in pull-down mode, it will be LOW. The programmer can set these conventions according to their convenience but it is important to match these conventions both in hardware and firmware. Matrix keypad pinout will have a slight difference according to the manufacturer and please refer the datasheet for the pinout details of the keypad module you are using. You can also build the matrix keypad by arranging the switches in the matrix format as shown below.



Working Principle

Let us consider the input pins connected in pull-up mode. The default state of input pins will be high in the pull-up mode. When a switch has pressed the column and row corresponding to the pressed key will be shorted and thus the value in the output line will be reflected in the input. we will write values to row pins in order to override the default state of the input pins when the key is pressed. The logic low is written to one of the rows and other row pins will be kept high as in the case of pull up mode. we will scan the column pins and the column pins corresponding to the pressed key will reflect a logic low. We will continue this procedure and write a logic low to other row pins and check the columns pins to detect the keypress.

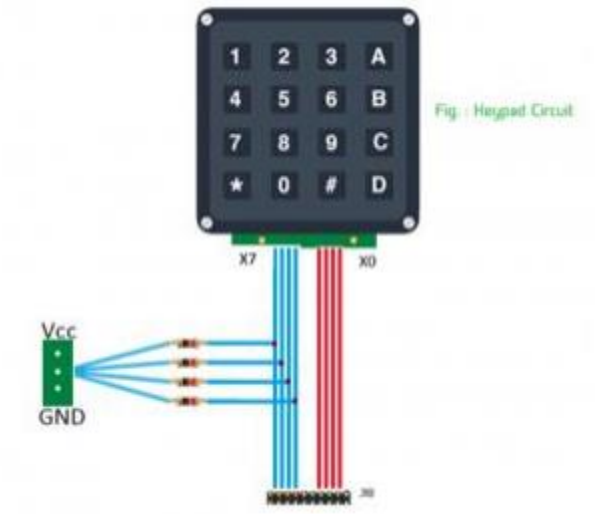


Fig : Keypad Circuit

FIRMWARE

We will have to connect the keypad to one of the controller ports. The row pins are connected to the LSB of PORTD AND Column pin to the MSB of PORTD.

Pin name	Col 4	Col 3	Col 2	Col 1	Row 4	Row 3	Row 2	Row 1
Port pin	7	6	5	4	3	2	1	0

We will set the direction of row pins as output and the column pins as inputs.

TRISD=0xF0; /*The direction of(0-3)row pins are set as output and

(4-7)coloumn pins are set as inputs*/

We have to create a function to detect the keypress. we will update the row values and do a column scan to detect the pressed key. The row pins are connected to LSB of PORT B and column pins in the MSB. You have to connect the inputs in either pullup or pull down mode. We will consider the pull-up mode here and the default input states will be high. When a key has pressed the row and column of that key will be shorted and the row value will be reflected in the column.
when a keypress occurs and a logic zero arrives at that row the column read will change the default high to low and we can confirm the keypress. we will write the logic zero to the row pins one by one and a column pin will be scanned and checked the values against a switch case to identify the matched column and the pressed key.

we will write a logic low to the R0 pin initially and all the other row pins and column pins are kept high. The hexadecimal number corresponding to the above condition is OXFE and it is written to PORTD. We will mask the column pins and it is read to a variable. The column value is checked against a switch case to identify the pressed key. if the 1st key is pressed the RO and C0 will be shorted and the value 0 appears in the column. The switch case value 0XE0 matches in that case. A similar procedure is followed for the rest of the rows. The While statement waits till the key pressed state changes and avoids denouncing and also we can add a small delay to ensure a proper reading.

char key()

```
{
int e;
    while(1)
    {

        PORTD = 0XFE;                /*First Row made low and scanning the columns*/
        __delay_ms(10);
        e = PORTD & 0xF0;
        switch(e)
        {
            case 0xE0:
                 while(!(PORTDbits.RD4));
            return('1');
            case 0xD0:
                while(!(PORTDbits.RD5));
            return('2');
```

```
        case 0xB0:
            while(!(PORTDbits.RD6));
        return('3');

        case 0x70:
            while(!(PORTDbits.RD7));
            return('A');
    }
    PORTD = 0XFD;                /*Second Row made low and scanning the columns*/
  __delay_ms(10);
    e = PORTD & 0xF0;
        switch(e)
{
        case 0xE0:
            while(!(PORTDbits.RD4));
        return('4');
```

```
    case 0xD0:
        while(!(PORTDbits.RD5));
    return('5');

    case 0xB0:
        while(!(PORTDbits.RD6));
    return('6');
            case 0x70:
         while(!(PORTDbits.RD7));
  return('B');

 }
 PORTD = 0XFB;                /*Third row made low and scanning the columns*/
__delay_ms(10);
 e = PORTD & 0xF0;
 switch(e)
 {
```

```
    case 0xE0:
        while(!(PORTDbits.RD4));
    return('7');

    case 0xD0:
        while(!(PORTDbits.RD5));
    return('8');
            case 0xB0:
        while(!(PORTDbits.RD6));
    return('9');
            case 0x70:
        while(!(PORTDbits.RD7));
    return('C');
        }
 PORTD = 0XF7;                  /*Fourth row made low and scanning the columns*/
__delay_ms(10);
 e = PORTD & 0xF0;
```

```
switch(e)
{
   case 0xE0:
      while(!(PORTDbits.RD4));
   return('*');
   case 0xD0:
      while(!(PORTDbits.RD5));
   return('0');
            case 0xB0:
      while(!(PORTDbits.RD6));
   return('#');
             case 0x70:
      while(!(PORTDbits.RD7));
   return('D');

}
```

```
    }

}
```

we will call the key function continuously in the main function to detect the keypress. We can show the detected keypress in any of the display modules according to your convenience.

```
  while(1)

    {

        a=key();

      }
```

we can develop a similar code for pull-down mode and in that case, the default column state will be LOW. We have to write a logic high to row pins one by one and we will scan the column pins to detect the keypress.