2.8 INTERFACING 4X4 KEYPAD WITH LPC1768

2.8.1 INTERFACING DIAGRAM

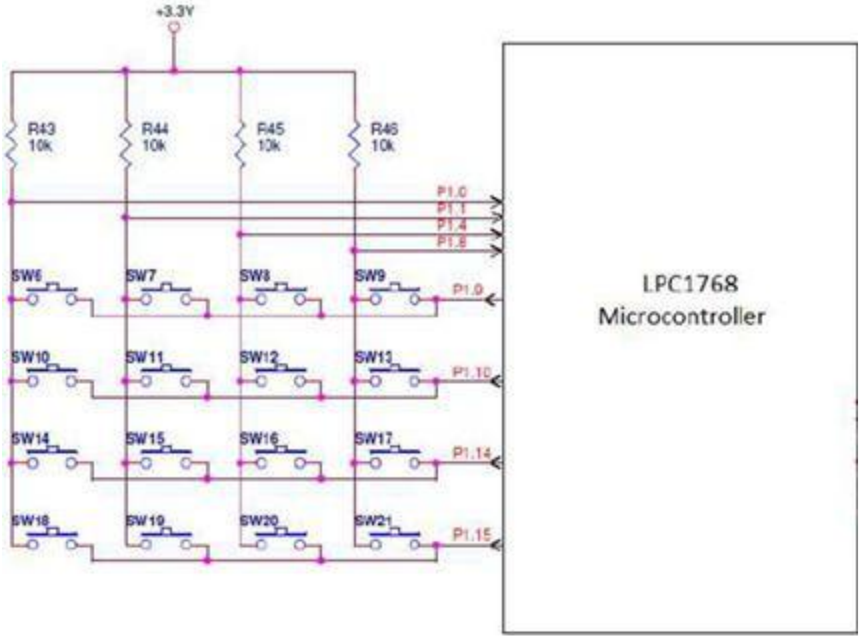

Fig.: 4x4 Keypad interfacing with LPC1768

The image shows a 4x4 keypad matrix connected to the LPC1768 microcontroller. The 4x4 keypad consists of 16 keys arranged in a matrix format of 4 rows and 4 columns.

The rows and columns are connected to GPIO pins of the microcontroller.

<u>Working Principle of 4x4 Keypad</u>

The working principle of the 4x4 keypad involves **scanning the rows and detecting the columns**.

1. **Rows (P1.6 to P1.9):** The rows are connected to **GPIO pins** (P1.6 to P1.9). These rows are initially **configured as output**.
2. **Columns (P1.10 to P1.15):** The columns are connected to **GPIO pins** (P1.10 to P1.15). These columns are initially **configured as input with pull-up resistors**.
3. **Pull-up Resistors (10kΩ):** The pull-up resistors ensure that the column pins stay **HIGH (logic 1)** when no key is pressed.
4. **Key Press Detection:**
    - The microcontroller **sets one row LOW at a time** (by making it output LOW).
    - Then it reads the **input column pins**. If any column pin is LOW, it indicates that a key in that row is pressed.
    - This process is repeated for all four rows to identify the pressed key.

Example:

Suppose SW6 is pressed (Row 1, Column 1):

- The microcontroller drives **Row 1 LOW** and keeps all other rows HIGH.
- It scans the column inputs. If **Column 1** reads **LOW**, it confirms that **SW6** is pressed.

This is repeated for all keys by scanning rows one by one.

Resistor Value Calculation:

The resistors (R43, R44, R45, R46) are **pull-up resistors**, which help to keep the GPIO pins at a **logic HIGH** when no key is pressed.

1. **Pull-up Voltage (Vcc):** +3.3V
2. **Input current for GPIO Pin:** Typically, in microamperes (~0.1mA)
3. **Desired Logic HIGH Voltage:** Close to +3.3V
4. **Resistor Value (R) :** Assuming I = 0.33 mA

$$R = \frac{Vcc}{I} = \frac{3.3}{0.33 \text{ x } 10^{-3}} \approx 10\ K\Omega$$

Advantages of Using Pull-up Resistors:

Ensures the input pins remain HIGH when no key is pressed.

Prevents floating input which can cause unpredictable behavior.

Consumes very low current when idle.

2.8.2 EXAMPLE CODE

Write a C program for the LPC17xx microcontroller that reads the status of a 4x4 key matrix. When any switch is pressed, display the hex value of key pressed on LEDs.

```
include <LPC17xx.h>
 void SetRowToZero(unsigned char);
unsigned char ReadColumnNumber(void);
void KeyPadInitialize(void); unsigned
char GetKeyPressed(void); void
delay(int count);
 char keyCodes[4][4]
=
{
     {0x1,0x2,0x3,0x4},
     {0x5,0x6,0x7,0x8},
     {0x9,0x0,0xA,0xB},
     {0xC,0xD,0xE,0xF}
};
int main(void)
```

```
{    unsigned char key;

     LPC_GPIO1->FIODIR = 0x07F80000;

     KeyPadInitialize();

      while(1) //forever loop
     {
          key = GetKeyPressed();
delay(20);     // Short delay

          if (key != 0xFF) // Valid key press
LPC_GPIO1->FIOPIN = (key << 19);           else
            LPC_GPIO1->FIOCLR = 0x07F80000;
     }
}  void
KeyPadInitialize(void)
{
  //Make Rows output    LPC_GPIO1->FIODIR |= (1 << 9)|(1 <<
10)|(1 << 14)|(1 << 15);

  //Make Columns as input
  LPC_GPIO1->FIODIR &= ~((1<<0)|(1<<1)|(1<<4)|(1<<8));
}  void SetRowToZero(unsigned char
rowNumber)
{
  LPC_GPIO1->FIOSET |= (1 << 9)|(1 << 10)|(1 << 14)|(1 << 15);
     switch(rowNumber)
     {
          case 0: LPC_GPIO1->FIOCLR |= (1 << 9);
                    break;
          case 1: LPC_GPIO1->FIOCLR |= (1 << 10);
                    break;
          case 2: LPC_GPIO1->FIOCLR |= (1 << 14);
                    break;
          case 3: LPC_GPIO1->FIOCLR |= (1 << 15);
                    break;
     }
}
unsigned char ReadColumnNumber(void)
{  unsigned long int temp = LPC_GPIO1->FIOPIN;
```

```
     if((temp & (1 << 0)) == 0)
return 0;
     else if((temp & (1 << 1)) == 0)
          return 1;
     else if((temp & (1 << 4)) == 0)
          return 2;
     else if((temp & (1 << 8)) == 0)
          return 3;
else
return 4;
}  unsigned char
GetKeyPressed(void)
{    unsigned char rowNumber, colNumber;

     while(1)
     {
          for(rowNumber = 0; rowNumber < 4; rowNumber++)
          {
               SetRowToZero(rowNumber);
     colNumber = ReadColumnNumber();

               if(colNumber < 4)                  return
keyCodes[rowNumber][colNumber];
          }
     }
}  void delay(int
count)
{    unsigned int i,j;

     for(i=0; i<count; i++)
for(j=0; j<1275; j++);
}
```

Further exploration:

Display the key pressed on LCD