BIRZEIT UNIVERSITY

Electrical and Computer Engineering

Computer Design Lab – ENCS4110

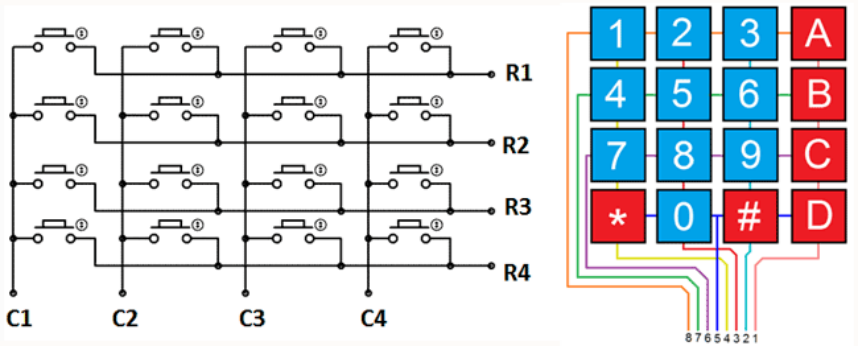
Exp10: Keypad

Objectives:

To know how to interface keypads.

Keypads

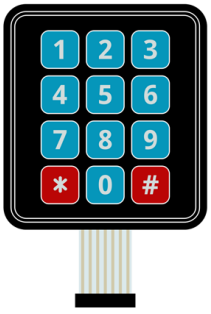
Matrix keypads are simply an extension to the simple tact switch inputs. They consist of keys interconnected in the shape of a matrix. Each key is a simple mechanical switch located at the crossing between the matrix rows and columns. When a key is pressed, its row and column form an electrical contact. The rows and columns can be connected to the pins of microcontroller ports. The big advantage of using a matrix keypad is that it allows interfacing a large number of keys with a relatively small number of microcontroller pins. For example, a 16-key keypad requires only 8 (instead of 16, if interfaced individually) I/O pins of the microcontroller if organized into a 4 rows and 4 columns matrix.



Let's consider the case of a 12 keys matrix as shown in the figure and we want to interface this keypad to LPC2138 microcontroller. In order to detect that one key is pressed we will use the following strategy:

1. The 12 keys are arranged in 3 columns and 4 rows. Each column and row is connected by a separated wire.

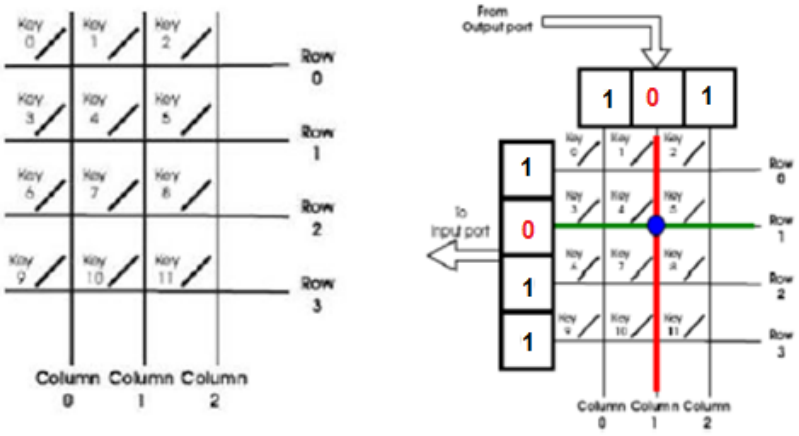
2. Pushing a key merely connects the corresponding row to the corresponding column.



3. We connect the columns to 3 output pin of the LPC and the rows to other 4 input pins.

4. For each of the columns we output logic '0' holding other columns to logic '1' and scan the values of the rows, If one of the rows is logic '0', this means that the key located at the intersection of the current column and the current row is pressed. If no row is zero, the logic '0' should be moved to the next column and the rows then rescanned.

5. The LPC should keep repeating this process every time it's required to detect a key press.



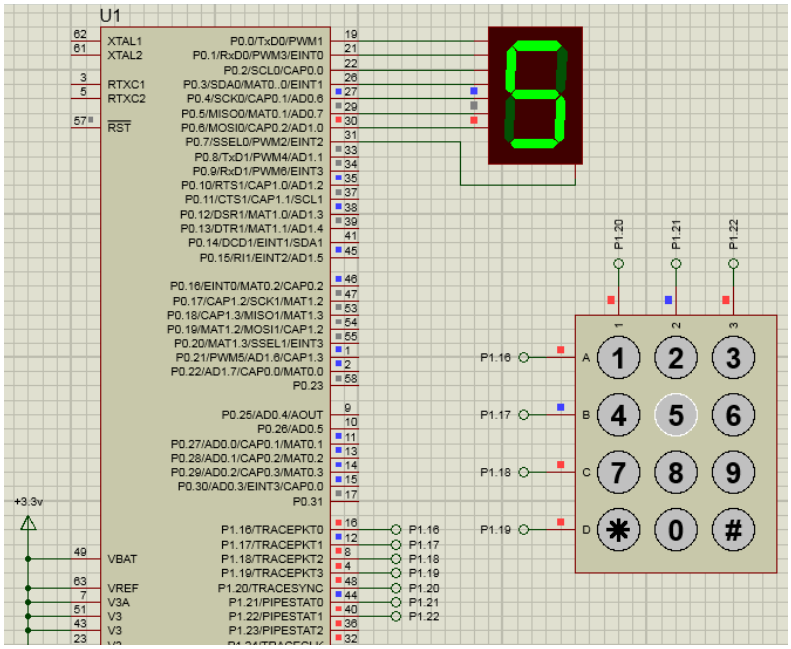
Lab Work 1

You are going to use this keywords when you search for parts in proteus:

Part	Keyword
Microcontroller	LPC2138
Resistor	res
Keypad	keypad-p

Write and simulate a program that displays the value of the pressed key on a 7-seg.

Proteus:



Keil:

```
#include<lpc213x.h>
#define bit(x) 1<<x

void msDelay(int d) {
      int i,j;
      long c=0;
      d=d*2;
      for(i=0;i<d;i++){
            for(j=0;j<1000;j++){
                  c++;
            }
      }
}


int dec_to_7seg(int number)
            {
            switch(number){
                  case 0 : return 0x3F;
                  case 1 : return 0x06;
                  case 2 : return 0x5B;
                  case 3 : return 0x4F;
                  case 4 : return 0x66;
                  case 5 : return 0x6D;
                  case 6 : return 0x7D;
                  case 7 : return 0x07;
                  case 8 : return 0x7F;
                  case 9 : return 0x6F;
                  default : return 0x00;
            }
      }

      int main(){
            IO0DIR |= 0x000000FF; //P0.0 ...P0.7 output
            IO1DIR &= ~(0x000F0000); //P1.16 ...P1.19  input
            IO1DIR |= 0x00700000; // P1.20 ...P1.22 output

            while(1){
                  //test first column
                  IO1CLR = bit(20);
                  IO1SET = bit(21)| bit(22);

                  if(!(IO1PIN & bit(16))){
                        msDelay(50);
                        IO0CLR = 0x000000FF;
                        IO0PIN |= dec_to_7seg(1);
                  }
                  else if (!(IO1PIN & bit(17))){
                         msDelay(50);
                         IO0CLR = 0x000000FF;
                         IO0PIN |= dec_to_7seg(4);
                  }
                  else if (!(IO1PIN & bit(18))){
                         msDelay(50);
                         IO0CLR = 0x000000FF;
```

```
                    IO0PIN |= dec_to_7seg(7);
                }

                //test second column

            IO1CLR = bit(21);
                IO1SET = bit(20)| bit(22);

                if(!(IO1PIN & bit(16))){
                     msDelay(50);
                     IO0CLR = 0x000000FF;
                     IO0PIN |= dec_to_7seg(2);
                }
                else if (!(IO1PIN & bit(17))){
                      msDelay(50);
                      IO0CLR = 0x000000FF;
                      IO0PIN |= dec_to_7seg(5);
                }
                else if (!(IO1PIN & bit(18))){
                      msDelay(50);
                      IO0CLR = 0x000000FF;
                      IO0PIN |= dec_to_7seg(8);
                }
                else if (!(IO1PIN & bit(19))){
                      msDelay(50);
                      IO0CLR = 0x000000FF;
                      IO0PIN |= dec_to_7seg(0);
                }

            // test third column

               IO1CLR = bit(22);
               IO1SET = bit(20)| bit(21);

                if(!(IO1PIN & bit(16))){
                     msDelay(50);
                     IO0CLR = 0x000000FF;
                     IO0PIN |= dec_to_7seg(3);
                }
                else if (!(IO1PIN & bit(17))){
                      msDelay(50);
                      IO0CLR = 0x000000FF;
                      IO0PIN |= dec_to_7seg(6);
                }
                else if (!(IO1PIN & bit(18))){
                      msDelay(50);
                      IO0CLR = 0x000000FF;
                      IO0PIN |= dec_to_7seg(9);
                }

        }
    }
```

ToDo Task:

1. Edit your program and simulation in order to display the value of the pressed key on a BCD 7-segment. (Binary coded decimal 7-segment display takes the numbers from 0-9 and displays them normally. Its internal work converts the entered decimal to binary by representing the number with four digits then views it as normal 7-seg. At Proteus: bcd display . It has four pins 1-4 from right to left connected to the pins of your port.)

2. Simulate a program that enable the user to enter his password by a keypad then press "check" button, if the password matches the stored one, the LED will shine.