Microprocessors and Microcontrollers (EE-231)

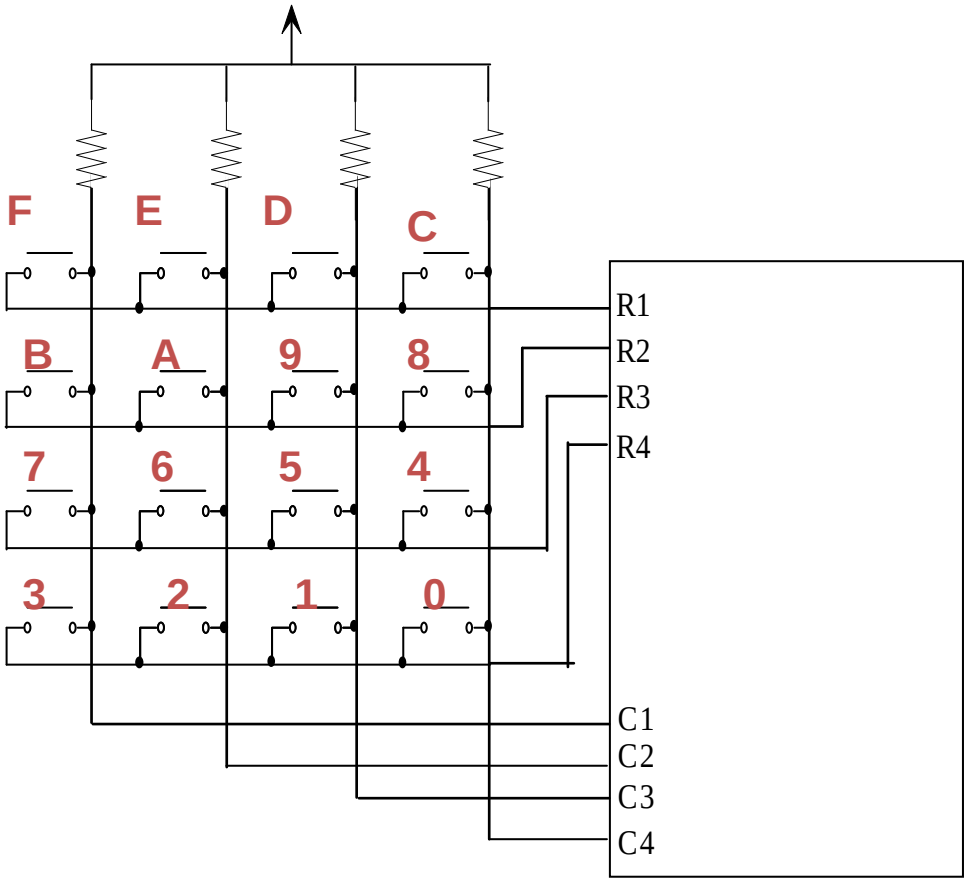
Lab-14

Objective

- Keypad interfacing and Programming in C
  - Implementation on Proteus
  - Implementation on Easy 8051 Kit

Interfacing a Keypad

- A 16-key keypad is built as shown in the figure below.
  - 16 keys arranged as a 4X4 matrix.
  - Must “activate” each row by placing a 0 on its R output.
    - Then the column output is read.
    - If there is a 0 on one of the column bits, then the button at the column/row intersection has been pressed.
    - Otherwise, try next row.
  - Repeat constantly



Bouncing Contacts

- Push-button switches, toggle switches, and electromechanical relays all have one thing in common: contacts.
- Metal contacts make and break the circuit and carry the current in switches and relays. Because they are metal, contacts have mass.
- Since at least one of the contacts is movable, it has springiness.
- Since contacts are designed to open and close quickly, there is little resistance to their movement

Bouncing

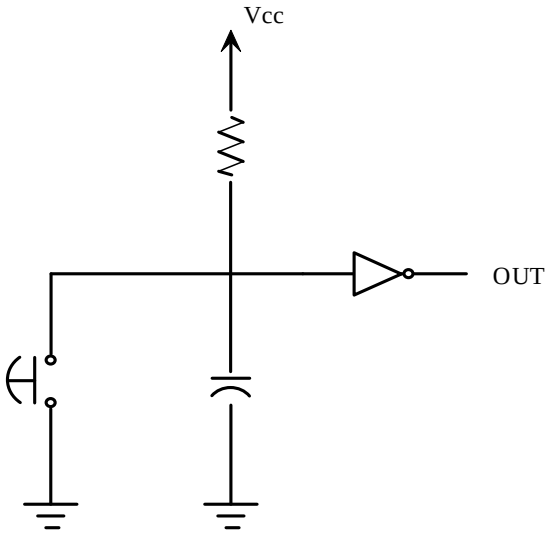
- Because the moving contacts have mass and springiness with low damping they will be "bouncy" as they make and break.
- That is, when a normally open (N.O.) pair of contacts is closed, the contacts will come together and bounce off each other several times before finally coming to rest in a closed position.
- The effect is called "contact bounce" or, in a switch, "switch bounce”.

Why is it a problem?

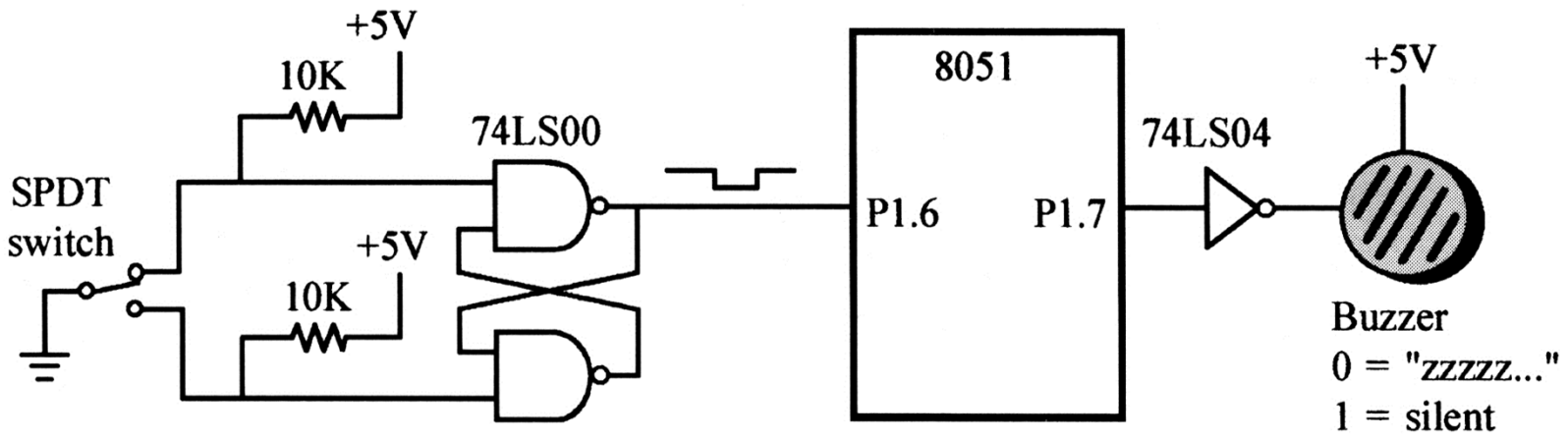
- If such a switch is used as a source to an edge-triggered input such as INT0, then the 8051 will think that there were several “events” and respond several times.
- The bouncing of the switch may last for several milliseconds.
  - Given that the 8051 operates at microsecond speed, a short ISR may execute several times in response to the above described bounciness

Hardware Solution

- The simplest hardware solution uses an RC time constant to suppress the bounce. The time constant has to be larger than the switch bounce and is typically 0.1 seconds.
- As long as capacitor voltage does not exceed a threshold value, the output signal will be continued to be recognized as a logic 1.
- The buffer after the switch produces a sharp high-to-low transition.



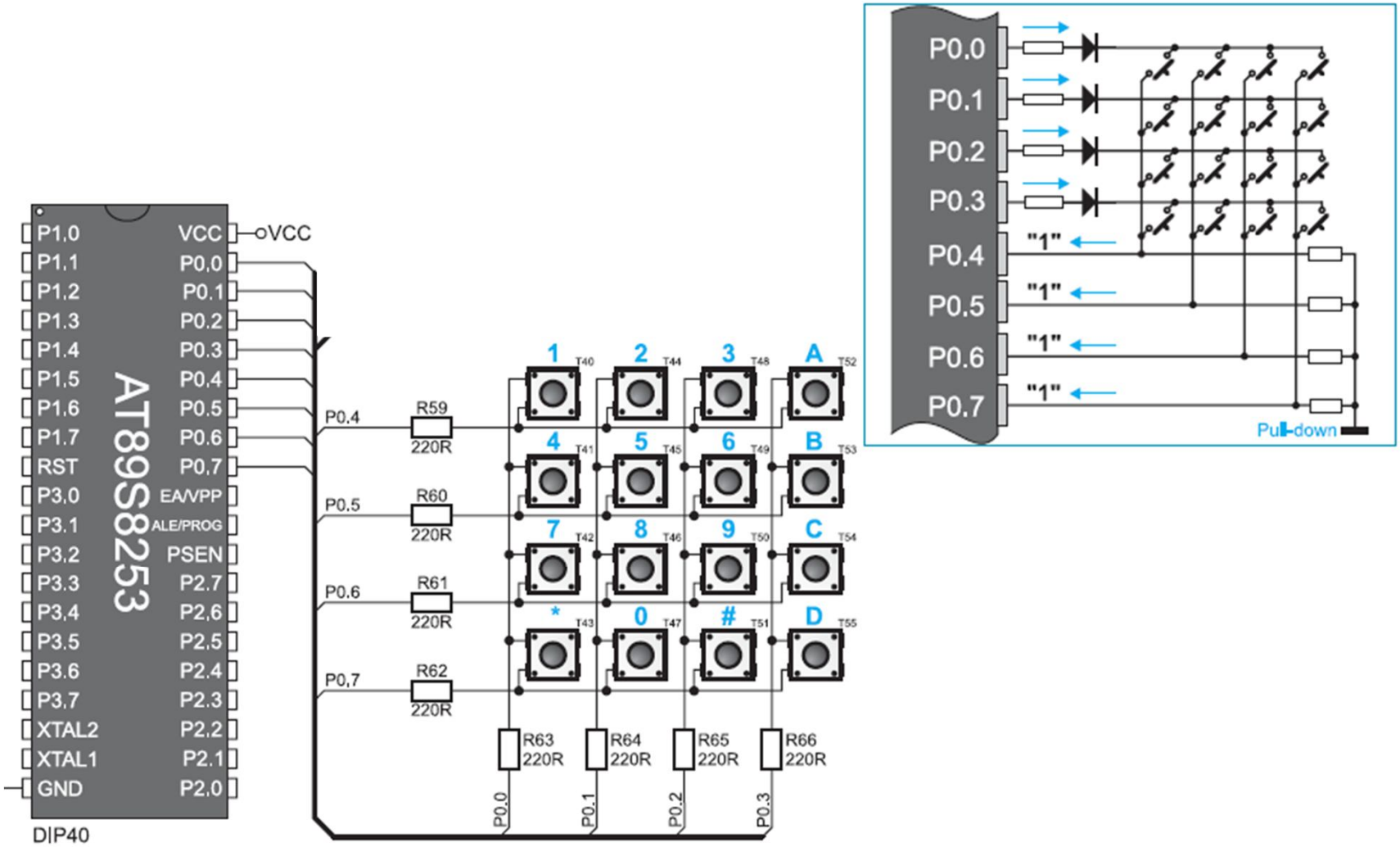
Hardware Solution



Software Solution

- It is also possible to counter the bouncing problem using software.
- The easiest way is the wait-and-see technique
  - When the input drops, an “appropriate” delay is executed (10 ms), then the value of the line is checked again to make sure the line has stopped bouncing

Keypad on Easy8051 Kit



Programming Method

```
#include<reg51.h>

sbit col1=P0^4;
sbit col2=P0^5;
sbit col3=P0^6;
sbit col4=P0^7;

sbit row1=P0^0;
sbit row2=P0^1;
sbit row3=P0^2;
sbit row4=P0^3;


void check_col1() //Function for checking column one
{
  row1=row2=row3=row4=1;
row1=0;
if(col1==0)
//Do whatever you want to do [1]
row1=1; row2=0;
if(col1==0)
//Do whatever you want to do [4]
row2=1; row3=0;
if(col1==0)
//Do whatever you want to do [7]
row3=1; row4=0;
if(col1==0)
//Do whatever you want to do [*]
row4=1;
}
```

```
void main()
{

col1=col2=col3=col4=1;  //Input Port
while(1)
{
  row1=row2=row3=row4=0;
  if(col1==0)
  check_col1();
  else
   if(col2==0)
   check_col2();
   else
    if(col3==0)
    check_col3();
   else
    if(col4==0)
    check_col4();
}
}
```

Todays Task 1

- Implement this on easy 8051 Kit and Proteus
- Write a code for keypad so that whatever key you press, it is displayed on the LCD.

Task Code

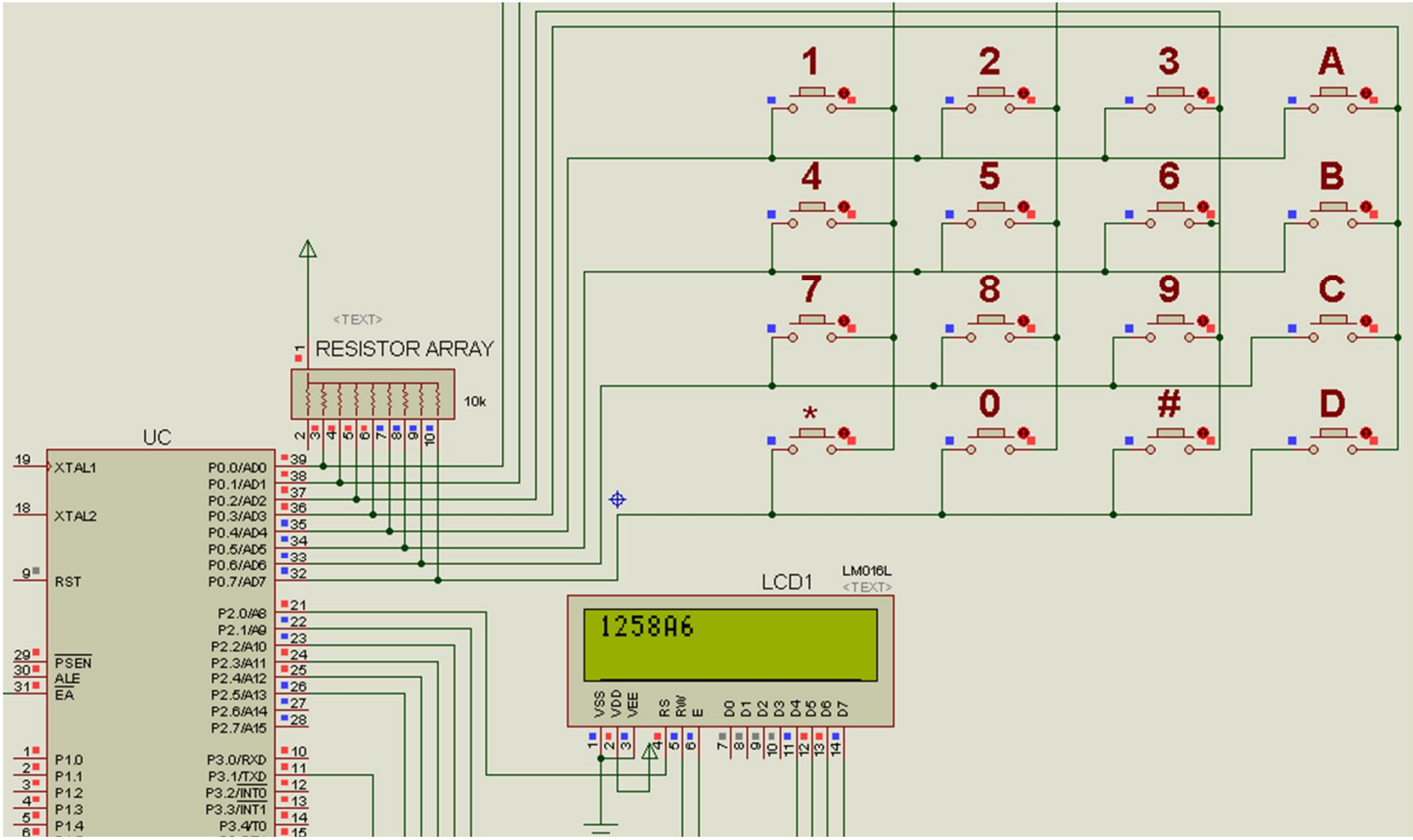
- Only some portions of code are shown here.

```
#include<reg51.h>
#define lcd P2
sbit RS=P2^0;
sbit E =P2^1;

sbit col1=P0^0;
sbit col2=P0^1;
sbit col3=P0^2;
sbit col4=P0^3;
sbit row1=P0^4;
sbit row2=P0^5;
sbit row3=P0^6;
sbit row4=P0^7;

void LCD_CMD(unsigned char);
void LCD_Data(unsigned char);
void delay_ms(unsigned int);
void Display_String(unsigned char*);

void check_col1()
{
```

```
void check_col1()
{
row1=row2=row3=row4=1;
row1=0;
if(col1==0)
LCD_Data('1');
row1=1;  row2=0;
if (col1==0)
LCD_Data('4');
row2=1;  row3=0;
if(col1==0)
LCD_Data('7');
row3=1;  row4=0;
if(col1==0)
LCD_Data('*');
row4=1;
row1=row2=row3=row4=0;
while(col1==0);
}
```

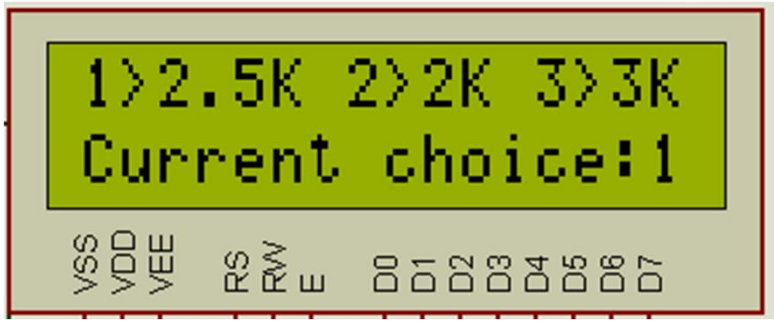
```
void main(void)
{
  lcd=0;
  lcd=lcd|0x08;
  E=1;
  E=0;
  delay_ms(1);
  LCD_CMD(0x28);// Function Set Command
  LCD_CMD(0x06);// Entry Mode Set
  LCD_CMD(0x0C);// Display on/off Control
  LCD_CMD(0x01);// Clear Display
  delay_ms(1);
  col1=col2=col3=col4=1;
  while(1)
  {
  row1=row2=row3=row4=0;
  if(col1==0)
  check_col1();
  else
  if(col2==0)
  check_col2();
  else
  if (col3==0)
  check_col3();
  else
  if(col4==0)
  check_col4();
  }
```

Proteus Simulation



Todays Task 2

- Implement this on easy 8051 Kit and proteus.
- Depending upon the key pressed on the keypad, generate 3 square waves. LCD should show the current status of the clock generator.



Task Code

```
#include<reg51.h>
#define lcd P2
sbit RS=P2^0;
sbit E =P2^1;

sbit col1=P0^0;
sbit col2=P0^1;
sbit col3=P0^2;
sbit col4=P0^3;
sbit row1=P0^4;
sbit row2=P0^5;
sbit row3=P0^6;
sbit row4=P0^7;

sbit wave=P3^1;

void LCD_CMD(unsigned char);
void LCD_Data(unsigned char);
void delay_ms(unsigned int);
void Display_String(unsigned char*);

void timer0(void) interrupt 1
{
  wave=~wave;
}
```

```
void check_col1()
{
row1=row2=row3=row4=1;
row1=0;
if(col1==0)
{
LCD_CMD(0xCF);//Display at character posiotion no 16 of line 2
LCD_Data('1');
TH0=-184;
}
row1=1;  row2=0;
if (col1==0)
LCD_Data('4');
row2=1;  row3=0;
if(col1==0)
LCD_Data('7');
row3=1;  row4=0;
if(col1==0)
LCD_Data('*');
row4=1;
row1=row2=row3=row4=0;
while(col1==0);
}
```

```
void check_col2()
{
row1=row2=row3=row4=1;
row1=0;
if(col2==0)
{
LCD_CMD(0xCF);//Display
LCD_Data('2');
TH0=-230;
```

```
void check_col3()
{
row1=row2=row3=row4=1;
row1=0;
if(col3==0)
{
LCD_CMD(0xCF);//Display at character pc
LCD_Data('3');
TH0=-153;
```

Task Code

```
void main(void)
{
  lcd=0;
  lcd=lcd|0x08;
  E=1;
  E=0;
  delay_ms(1);
  LCD_CMD(0x28);// Function Set Command
  LCD_CMD(0x06);// Entry Mode Set
  LCD_CMD(0x0C);// Display on/off Control
  LCD_CMD(0x01);// Clear Display
  delay_ms(1);

  TMOD=0x02;
  EA=1;
  ET0=1;
  TR0=1;

  Display_String("1>2.5K 2>2K 3>3K");
  LCD_CMD(0xC0);
  Display_String("Current choice:");
```

```
  col1=col2=col3=col4=1;
  while(1)
  {
  row1=row2=row3=row4=0;
  if(col1==0)
  check_col1();
  else
  if(col2==0)
  check_col2();
  else
  if (col3==0)
  check_col3();
  else
  if(col4==0)
  check_col4();
  }
```

Proteus Simulation