LCD DISPLAY

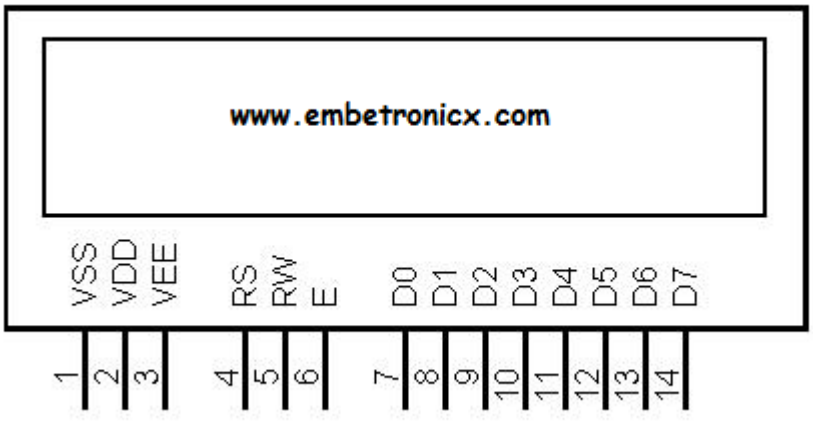
We always use devices made up of Liquid Crystal Displays (LCDs) like computers, digital watches, and also DVD and CD players. They have become very common and have taken a giant leap in the screen industry by clearly replacing the use of Cathode Ray Tubes (CRT). CRT draws more power than LCD and is also bigger and heavier. All of us have seen an LCD, but no one knows the exact working of it. Let us take a look at the working of an LCD.

Here we are using alphanumeric LCD 16×2. A 16×2 LCD display is a very basic module and is very commonly used in various devices and circuits. These modules are preferred over seven segments and other multi-segment LEDs.

The reasons are: LCDs are economical; easily programmable; have no limitation of displaying special & even custom characters (unlike in seven segments), animations, and so on.

A 16×2 LCD means it can display 16 characters per line and there are 2 such lines. In this LCD each character is displayed in a 5×7 pixel matrix. This LCD has two registers, namely, Command and Data. The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling the display, etc. The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD.

16×2 LCD Pin Diagram



Pin Description

Pin No	Function	Name
1	Ground (0V)	Ground
2	Supply voltage; 5V (4.7V – 5.3V)	Vcc

3	Contrast adjustment; through a variable resistor	V_{EE}
4	Selects command register when low; and data register when high	Register Select
5	Low to write to the register; High to read from the register	Read/write
6	Sends data to data pins when a high-to-low pulse is given	Enable
7	8-bit data pins	DB0
8		DB1
9		DB2
10		DB3
11		DB4
12		DB5
13		DB6
14		DB7
15	Backlight V_{CC} (5V)	Led+
16	Backlight Ground (0V)	Led-

The LCD display module requires 3 control lines as well as either 4 or 8 I/O lines for the data bus. The user may select whether the LCD is to operate with a 4-bit data bus or an 8-bit data bus. If a 4-bit data bus is used the LCD will require a total of 7 data lines (3 control lines plus the 4 lines for the data bus). If an 8-bit data bus is used the LCD will require a total of 11 data lines (3 control lines plus the 8 lines for the data bus).

The three control lines are referred to as EN, RS, and RW.

The EN line is called “Enable.” This control line is used to tell the LCD that you are sending it data. To send data to the LCD, your program should make sure this line is low (0) and then set the other two control lines and/or put data on the data bus. When the other lines are completely ready, bring EN high (1) and wait for the minimum amount of time required by the LCD datasheet (this varies from LCD to LCD), and end by bringing it low (0) again.

The RS line is the “Register Select” line. When RS is low (0), the data is to be treated as a command or special instruction (such as a clear screen, position cursor, etc.). When RS is high (1), the data being sent is text data which should be displayed on the screen. For example, to display the letter “T” on the screen you would set RS high.

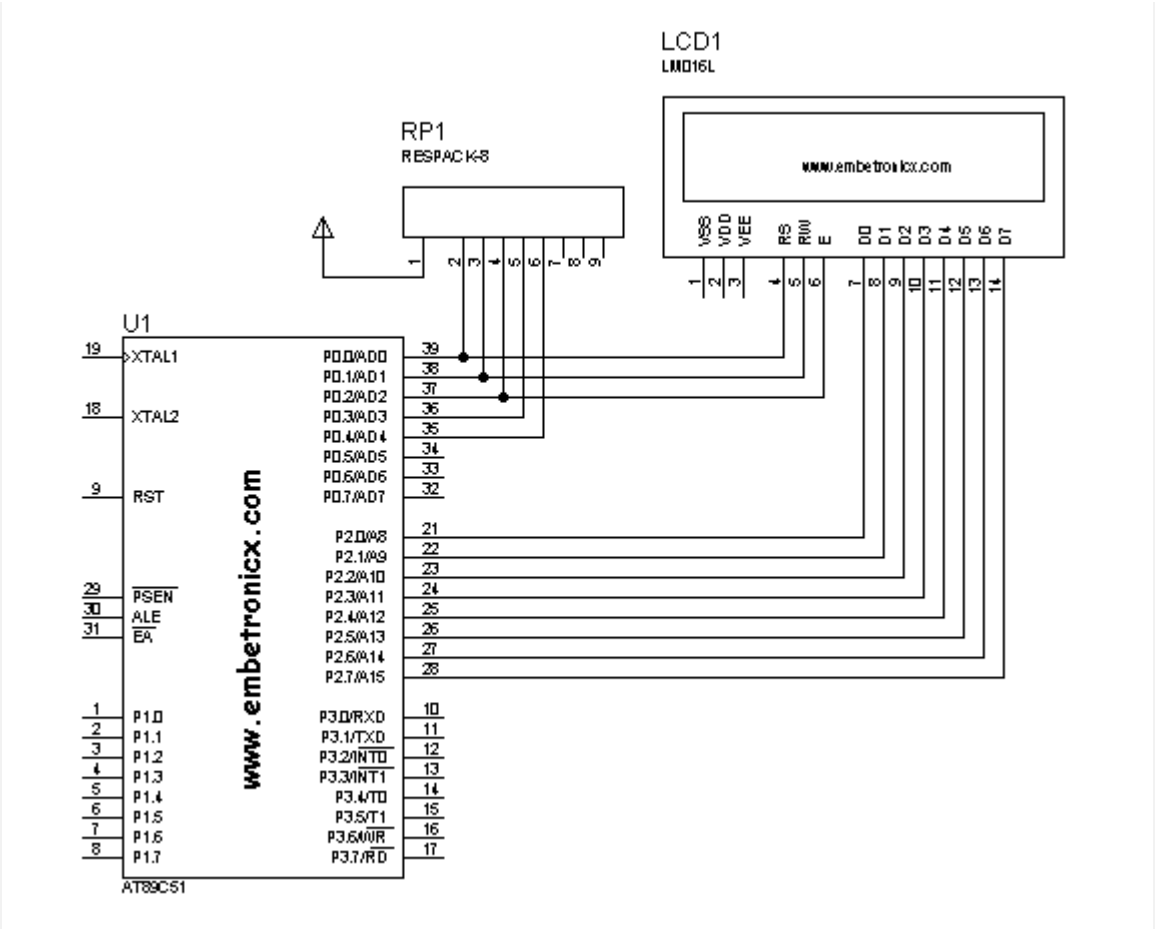
The RW line is the “Read/Write” control line. When RW is low (0), the information on the data bus is written to the LCD. When RW is high (1), the program is effectively querying (or reading) the LCD. Only one instruction (“Get LCD status”) is a read command. All others are write commands–so RW will almost always be LOW.

Finally, the data bus consists of 4 or 8 lines (depending on the mode of operation selected by the user). In the case of an 8-bit data bus, the lines are referred to as DB0, DB1, DB2, DB3,

DB4, DB5, DB6, and DB7.

Code (Hex)	Command to LCD Instruction Register
1	Clear Display screen
2	Return home
4	Decrement cursor (Shift cursor to left)
6	Increment cursor (Shift cursor to Right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display on, cursor off
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning of 1st line
0C0	Force cursor to beginning of 2nd line
38	2 lines and 5x7 Matrix www.embetronicx.com

Interfacing LCD with Microcontroller – CIRCUIT DIAGRAM



RS is connected to Port 0.0 (P0.0)

RW is connected to Port 0.1 (P0.1)

EN is connected to Port 0.2 (P0.2)

Data lines are connected to Port 2 (P2)

LCD Interfacing with 8051 Microcontroller – Programming

Send Data

To send data on the LCD, data is first written to the data pins with R/W = 0 (to specify the write operation) and RS = 1 (to select the data register). A high to low pulse is given at **EN** pin when data is sent. Each write operation is performed on the positive edge of the Enable signal.

```
void dat(unsigned char b)
{
lcd_data=b;
rs=1;

en=1;
lcd_delay();
en=0;
}
```

Send String

We cannot send more than 8 bits at the same time. Because data lines are only having 8 bits. So how we can send string? Any guess? Yeah, you are correct. We have to send the string by character. See this code.

```

{
while(*s) {
dat(*s++);
}
}
```

Send Command

To send a command on the LCD, a particular command is first specified to the data pins with R/W = 0 (to specify the write operation) and RS = 0 (to select the command register). A high to low pulse is given at the EN pin when data is sent.

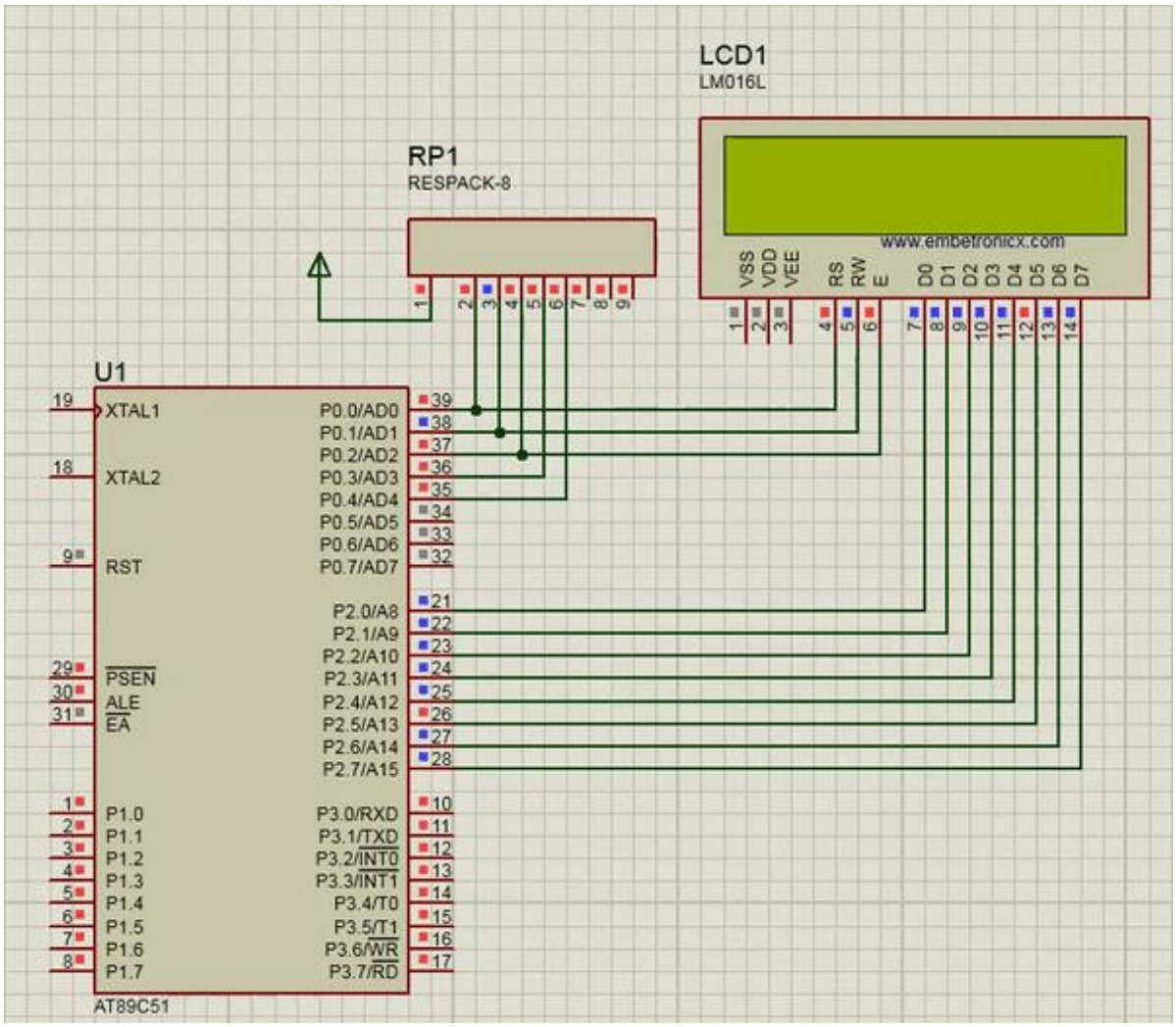
```

{
lcd_data=a;
rs=0;
rw=0;
en=1;
lcd_delay();
en=0;
}
```

LCD Initializing

To initialize the LCD we have to use certain commands.

```
{
cmd(0x38);
cmd(0x0e);
cmd(0x01);
cmd(0x06);
cmd(0x0c);
cmd(0x80);
}
```

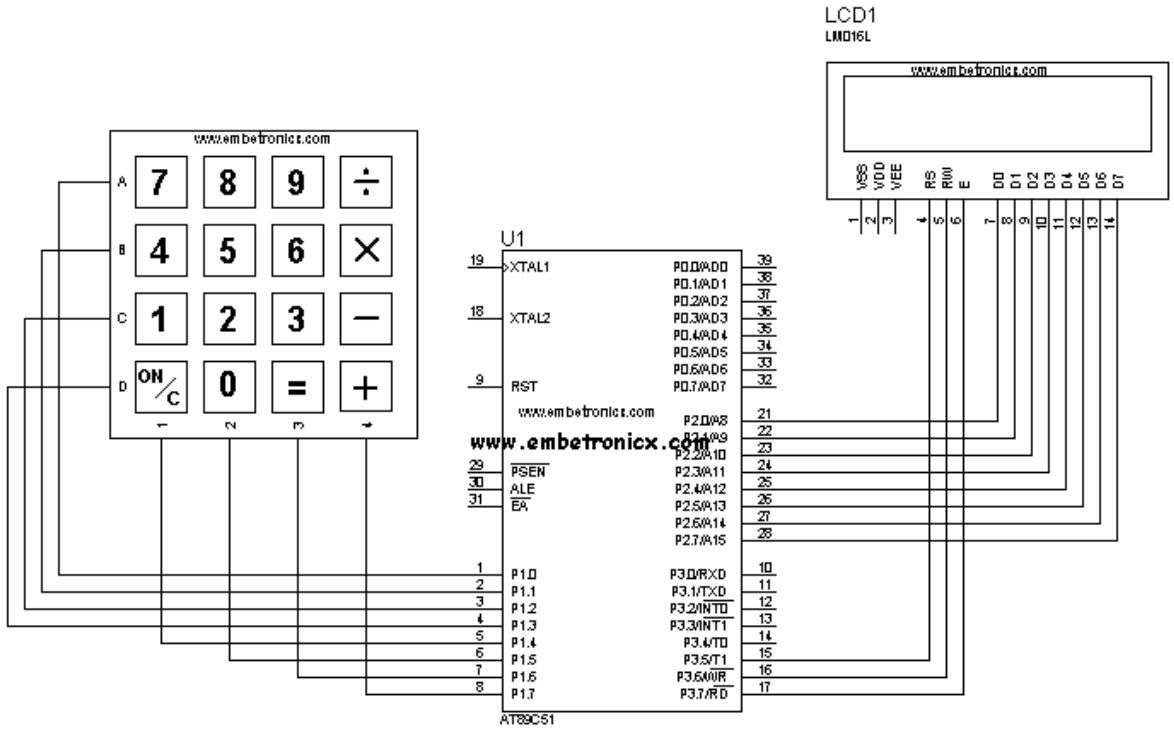


8051 – Keypad Interfacing

Components Required

- 4×4 Keypad or 3×4 Keypad (Here we will discuss both codes)
- LCD Module (To print the Keys pressed)
- 8051 Microcontroller

4×4 Matrix Keypad Interfacing



LCD

- RS – P3.5
- RW – P3.6
- EN – P3.7
- Data Lines – P2

Keypad

- R1 – P1.0
- R2 – P1.1
- R3 – P1.2
- R4 – P1.3
- C1 – P1.4
- C2 – P1.5
- C3 – P1.6
- C4 – P1.7

Code

This code might be looking big. But concept-wise it is very easy. Please go through this code.

```
#include<reg51.h>
#define lcd P2
sbit rs=P3^5;
sbit rw=P3^6;
sbit en=P3^7;
sbit r1=P1^0;
sbit r2=P1^1;
sbit r3=P1^2;
sbit r4=P1^3;
sbit c1=P1^4;
sbit c2=P1^5;
sbit c3=P1^6;
sbit c4=P1^7;
void lcd_init();
void cmd(unsigned char );
void dat(unsigned char );
lcd_string(unsigned char *);
void delay(unsigned int );
void keypad(void);
void main()
{
lcd_init();
while(1) {
cmd(0x80);
lcd_string("Enter the key:");
cmd(0xc7);
keypad();
}
}
void keypad()
{
c1=c2=c3=c4=1;
r1=0;r2=1;r3=1;r4=1;
```

```
if(c1==0){
while(c1==0);
dat('7');
} else if(c2==0) {
while(c2==0);
dat('8');
} else if(c3==0) {
while(c3==0);
dat('9');
} else if(c4==0) {
while(c4==0);
dat('/');
}
r1=1;r2=0;r3=1;r4=1;
if(c1==0){
while(c1==0);
dat('4');
} else if(c2==0) {
while(c2==0);
dat('5');
} else if(c3==0) {
while(c3==0);
dat('6');
} else if(c4==0) {
while(c4==0);
dat('*');
}
r1=1;r2=1;r3=0;r4=1;
if(c1==0){
while(c1==0);
dat('1');
} else if(c2==0) {
while(c2==0);
dat('2');
} else if(c3==0) {
while(c3==0);
```

```
dat('3');
} else if(c4==0) {
while(c4==0);
dat('-');
}
r1=1;r2=1;r3=1;r4=0;
if(c1==0){
while(c1==0);
cmd(0x01);
} else if(c2==0) {
while(c2==0);
dat('0');
} else if(c3==0) {
while(c3==0);
dat('=');
} else if(c4==0) {
while(c4==0);
dat('+');
}
}
void lcd_init()
{
cmd(0x38);
cmd(0x0e);
cmd(0x06);
cmd(0x01);
}
void cmd(unsigned char x)
{
lcd=x;
rs=0;
rw=0;
en=1;
delay(1000);
en=0;
}
```

```
void dat(unsigned char y)
{
lcd=y;
rs=1;
rw=0;
en=1;
delay(1000);
en=0;
}
lcd_string(unsigned char *s)
{
while(*s)
dat(*s++);
}
void delay(unsigned int z)
{
unsigned int i;
for(i=0;i<=z;i++);
}
```

Code Explanation

I assumed that you already know about LCD interfacing. Now look at these lines in keypad function,

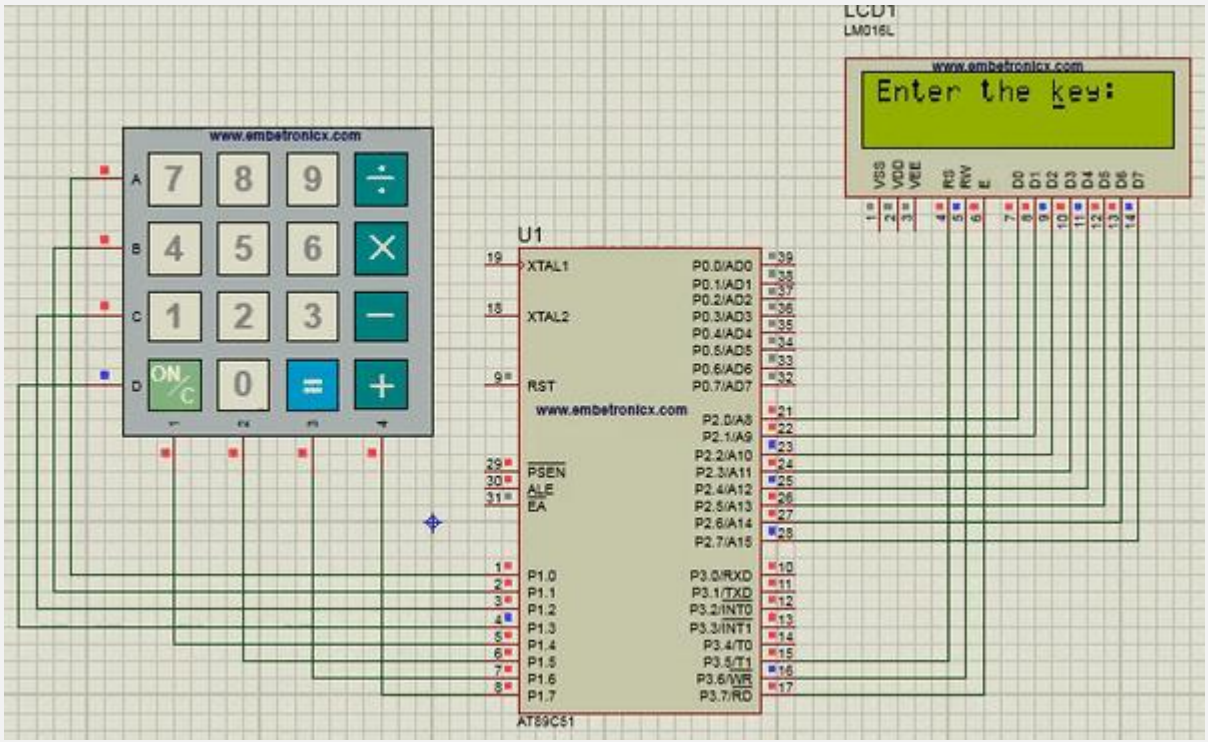
```
c1=c2=c3=c4=1;
r1=0;r2=1;r3=1;r4=1;
if(c1==0){
while(c1==0);
dat('7');
} else if(c2==0) {
while(c2==0);
dat('8');
} else if(c3==0) {
while(c3==0);
dat('9');
} else if(c4==0) {
```

```
while(c4==0);
dat('/');
}
```

In this code, I'm taking row as output and column as input.

1. In the first line, I'm assigning high to all columns. (c1=c2=c3=c4=1;)
2. Then I'm assigning the first row to zero and keeps the remaining row as high. (r1=0;r2=1;r3=1;r4=1;)
3. Then I'm checking the first column is zero or not. If it is zero then I should wait until that button depressed. Then I can know the pressed key.
4. If not I'm checking the next column. Like that, I'm checking all rows and columns.
5. If no keys pressed in row1, then I'm making row2 as zero. The remaining rows are high. Then follow the above steps.

Output

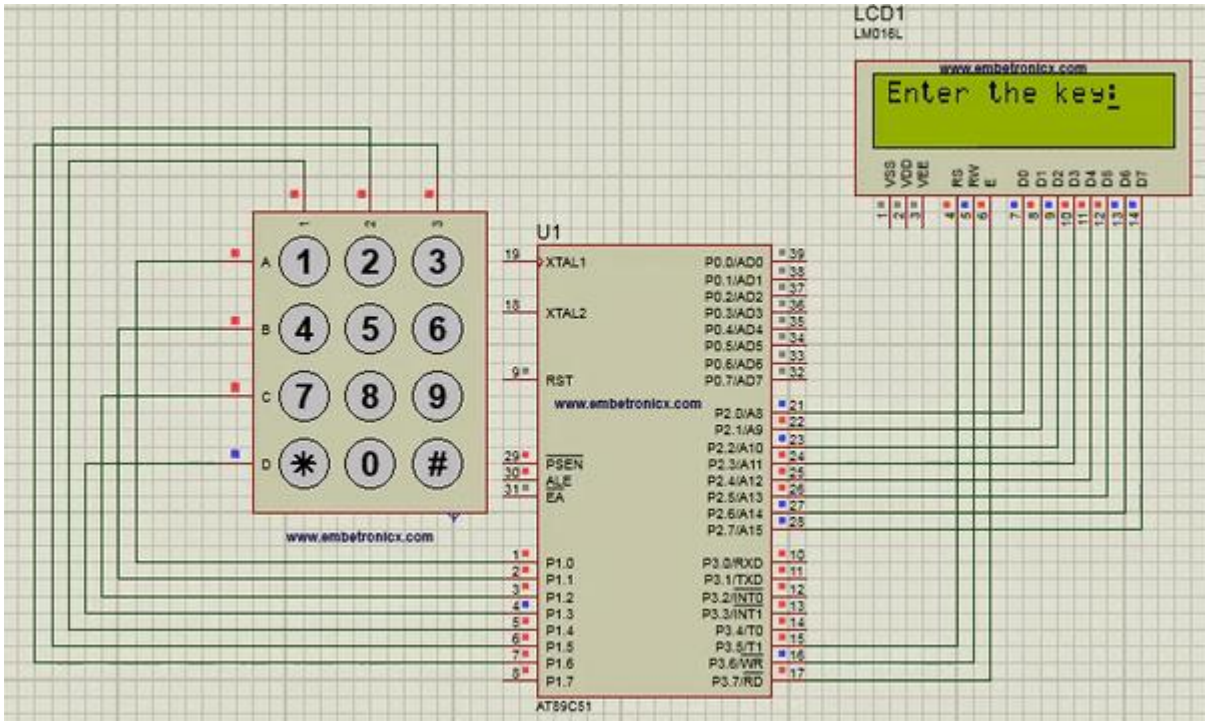



```
#include<reg51.h>
#define lcd P2
sbit rs=P3^5;
sbit rw=P3^6;
sbit en=P3^7;
sbit r1=P1^0;
```

```
sbit r2=P1^1;
sbit r3=P1^2;
sbit r4=P1^3;
sbit c1=P1^4;
sbit c2=P1^5;
sbit c3=P1^6;
void lcd_init();
void cmd(unsigned char );
void dat(unsigned char );
lcd_string(unsigned char *);
void delay(unsigned int );
void keypad(void);
void main()
{
lcd_init();
while(1) {
cmd(0x80);
lcd_string("Enter the key:");
cmd(0xc7);
keypad();
}
}
void keypad()
{
c1=c2=c3=1;
r1=0;r2=1;r3=1;r4=1;
if(c1==0){
while(c1==0);
dat('1');
} else if(c2==0) {
while(c2==0);
dat('2');
} else if(c3==0) {
while(c3==0);
dat('3');
}
```

```
r1=1;r2=0;r3=1;r4=1;
if(c1==0){
while(c1==0);
dat('4');
} else if(c2==0) {
while(c2==0);
dat('5');
} else if(c3==0) {
while(c3==0);
dat('6');
}
r1=1;r2=1;r3=0;r4=1;
if(c1==0){
while(c1==0);
dat('7');
} else if(c2==0) {
while(c2==0);
dat('8');
} else if(c3==0) {
while(c3==0);
dat('9');
}
r1=1;r2=1;r3=1;r4=0;
if(c1==0){
while(c1==0);
dat('*');
} else if(c2==0) {
while(c2==0);
dat('0');
} else if(c3==0) {
while(c3==0);
dat('#');
}
}
void lcd_init()
{
```

```
cmd(0x38);
cmd(0x0e);
cmd(0x06);
cmd(0x01);
}
void cmd(unsigned char x)
{
lcd=x;
rs=0;
rw=0;
en=1;
delay(1000);
en=0;
}
void dat(unsigned char y)
{
lcd=y;
rs=1;
rw=0;
en=1;
delay(1000);
en=0;
}
lcd_string(unsigned char *s)
{
while(*s)
dat(*s++);
}
void delay(unsigned int z)
{
unsigned int i;
for(i=0;i<=z;i++);
}
```


LCD1
LM016L
www.embetronicx.com
Enter the key:
VSS
VDD
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
U1
XTAL1
XTAL2
RST
PSEN
ALE
EA
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
AT89C51
www.embetronicx.com
1
2
3
4
5
6
7
8
9
*
0
#
A
B
C
D
www.embetronicx.com