e-PG Pathshala

Subject: Computer Science

Paper: Embedded System
Module: Interfacing External Devices using Embedded C
Module No: CS/ES/22

Quadrant 1 – e-text

In this lecture interfacing of external devices with 8051 will be discussed. The Interfacing of external memory with 8051 will be discussed. The Embedded C program for the interfacing of 8051 with external devices will be discussed in detail with examples.

1. LCD

A liquid crystal display (LCD) is a flat panel display that uses the light modulating properties of liquid crystals (LCs). LCD Modules can present textual information to user. LCD is used to provide user interface for normal output as well as for debugging purpose.

1.1 LCD Interfacing

LCD modules are available in different configurations. Here, we will consider the 2x16 character module available from Optrex to illustrate the interfacing with the microcontroller. The 2x16 character LCD module supports both 4-bit and 8-bit modes. These modes differ in how data is sent to LCD. In 8-bit mode, to write a character, 8-bit ASCII data is sent through data lines D0-D7 and data strobe is given through pin E of the LCD. But 4-bit mode uses only 4 data lines D4-D7. In this mode, 8-bit ASCII data and command data are divided into two parts and sent sequentially through the data lines. It is used in communication because, less pins will be used. Therefore 4-bit mode is preferred than 8-bit even with speed difference. Figure 1 shown below gives the details of interfacing 4-bit LCD to Microcontroller.

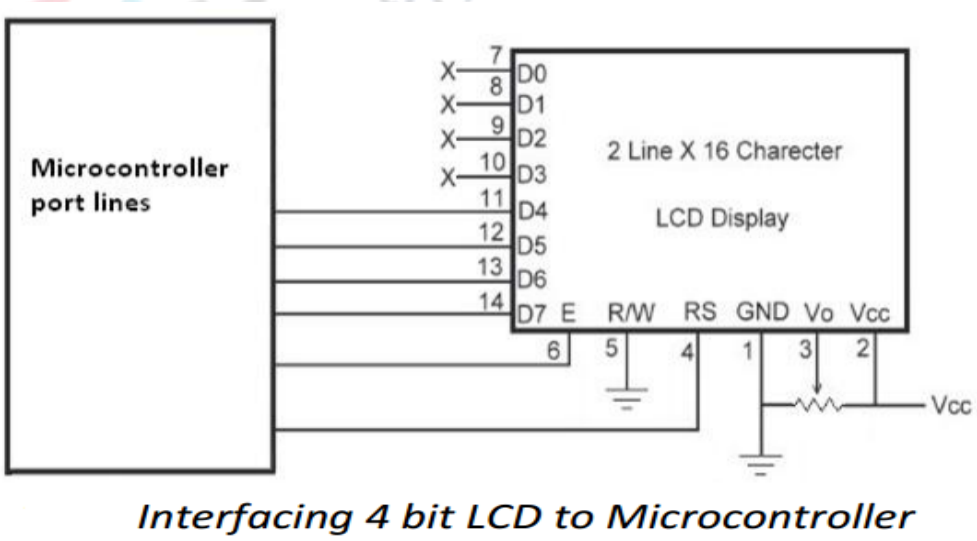


Figure 1 Interfacing of 4-bit LCD to Microcontroller

Pin Descriptions of LCD are described in Table 1 and Pin Positions for various LCDs are shown in Figure 2.

Pin no.	Symbol	External connection	Function
1	Vss	Power supply	Signal ground for LCM
2	V_{DD}		Power supply for logic for LCM
3	V_0		Contrast adjust
4	RS	MPU	Register select signal
5	R/W	MPU	Read/write select signal
6	E	MPU	Operation (data read/write) enable signal
7~10	DB0~DB3	MPU	Four low order bi-directional three-state data bus lines. Used for data transfer between the MPU and the LCM. These four are not used during 4-bit operation.
11~14	DB4~DB7	MPU	Four high order bi-directional three-state data bus lines. Used for data transfer between the MPU

Table 1. Pin Descriptions of LCD

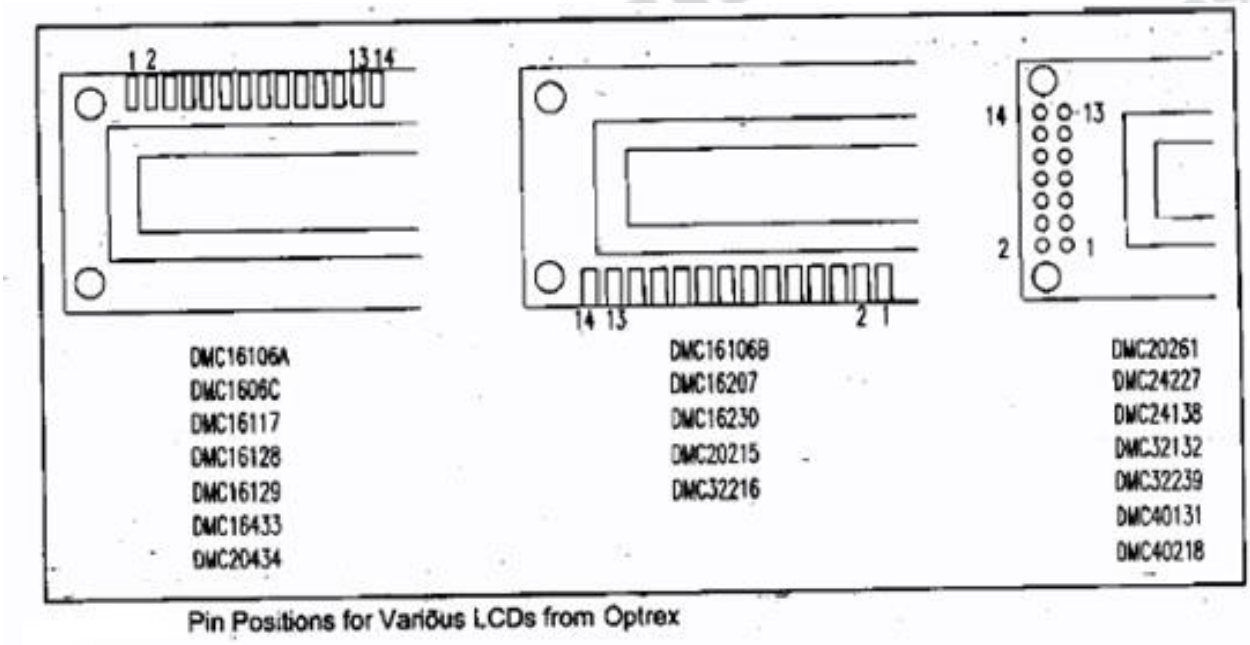


Figure 2. Pin Positions for various LCDs from optrex.

1.2 LCD command codes

The LCD's internal controller can accept several commands and modify the display accordingly. These commands would be things like:

- Clear screen
- Return home
- Decrement/Increment cursor

After writing to the LCD, it takes some time for it to complete its internal operations. During this time, it will not accept any new commands or data. We need to insert time delay between any two commands or data sent to LCD. The LCD command codes are described in Table 2 shown below.

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 off, cursor on
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 to 1st line
C0	Force cursor to beginning to 2nd line
38	2 lines and 5x7 matrix

Table 2. LCD command codes

1.3 Circuit diagram to Interface 4-bit LCD with 8051

The circuit diagram for interfacing 4-bit LCD with 8051 is shown in Figure 3 below.

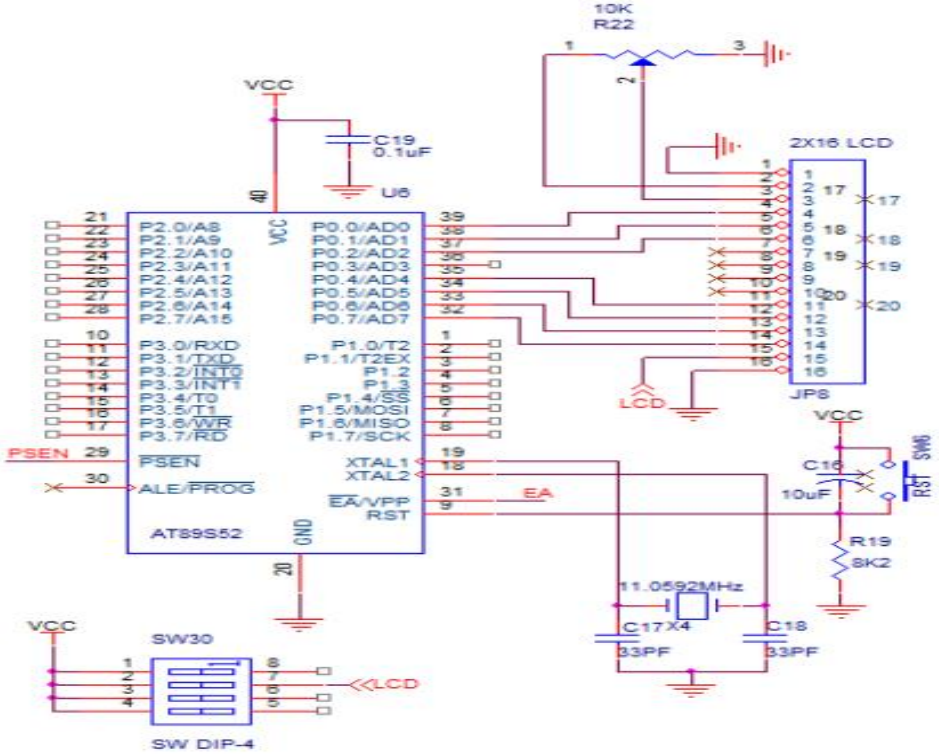


Figure 3 Interfacing 4-bit LCD with 8051

Example 1 described below gives the details about how to write embedded C program for interfacing of LCD to 8051.

Example 1

Write an 8051 C program to send letters ‘M’, ‘D’, and ‘E’ to the LCD using the busy flag method. Port1 connected to LCD data pins D0-D7.
P2.0=RS , P2.1=R/W, P2.2=E pins.

Solution:

```
#include <reg51.h>
sfr ldata = 0x90; //P1=LCD data pins
sbit rs = P2^0;
sbit rw = P2^1;
sbit en = P2^2;
sbit busy = P1^7;

Void main()

{
        lcdcmd(0x38);// 2 lines 5x7 matrix
        lcdcmd(0x0E);//display on cursor blinking
        lcdcmd(0x01);//clear display screen
        lcdcmd(0x06);//increment cursor
        lcdcmd(0x86); //line 1, position 6
        lcddata(‘M’);
        lcddata(‘D’);
        lcddata(‘E’);
}

void lcdcmd(unsigned char value)

{
        lcdready(); //check the LCD busy flag
        ldata = value; //put the value on the pins port1
        rs = 0; // for command rs=0
        rw = 0;// r/w=0 for write
        en = 1; //strobe the enable pin
        MSDelay(1);
        en = 0; // high low pulse to E
        return;
}

void lcddata(unsigned char value)

{
        lcdready(); //check the LCD busy flag
        ldata = value; //put the value on the pins
        rs = 1;  // rs=1 for data
        rw = 0; // for write
        en = 1; //strobe the enable pin
```

```
        MSDelay(1);
        en = 0;  //high low pulse
        return;
}

void lcdready()

{
        busy = 1; //make the busy pin at input
        rs = 0;
        rw = 1;
        while(busy==1){ //wait here for busy flag
        en = 0; //strobe the enable pin
        MSDelay(1);
        en = 1;
}
void msdelay (unsigned int itime)

{
        unsigned int i, j;
        for(i=0;i<itime;i++)
        for(j=0;j<1275;j++);
}
```

The above program sends the letters 'M', 'D', and 'E' to the LCD using the busy flag method.

2. Keyboard Interfacing

Keys in keyboards are arranged in a matrix of rows and columns. Controller accesses both rows and columns through ports. Using two ports we can connect 8x8 matrix key board. When a key is pressed, a row and column make contact; otherwise there is no contact.

4 x 4 Keyboard:

A 4x4 matrix is connected to two ports.The rows are connected to an output port and the columns are connected to an input port. Port1 of 8051 is connected to the rows of the key matrix, hence it acts as an output port. Port 2 of 8051 is connected to the columns of the key matrix, hence it acts as an input port. The matrix keyboard connection to ports is shown in Figure 4.

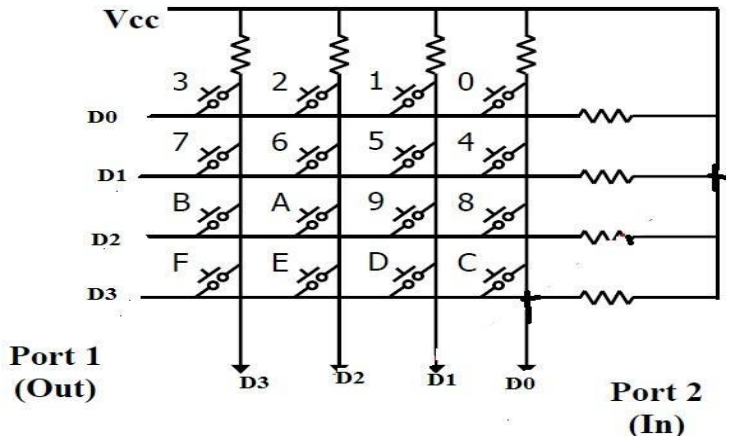


Figure 4. Matrix keyboard connection to ports

Key Scan:

To find out the key pressed, the controller grounds all rows by sending '0' on the port, then it reads the column data. If the data from column is D3-D0=1111, then no key is pressed. If any bit of the column is '0', it indicates key is pressed.

Example: 1110 – key pressed in column 0
1101 – key pressed in column 1
1011 – key pressed in column 2
0111 – key pressed in column 3

2.1 Steps to find out key pressed

The following are the steps to find out how the key is pressed in the keyboard

- Beginning with the row 0, the microcontroller grounds it by providing a low to row D0 only.
- It reads the columns(port2). If the data read is all 1s, no key in that row is activated and the process is moved to the next row.
- It grounds the next row, reads the columns, and checks for any zero.
- This process continues until the row is identified.
- After identification of the row in which the key has been pressed, find out which column the pressed key belongs to by looking for a zero on it.

Example:

(a) If D3 – D0 = 1101 for the row, D3 – D0 = 1011 for the column, then the key located at row 1 and column 3, (key 6) is pressed.
(b) If D3 – D0 = 1011 for the row, D3 – D0 = 0111 for the column, then the key located at row 2 and column 3, (key 'B') is pressed.

The Figure 5 shown below gives the interfacing of keyboard with 8051.

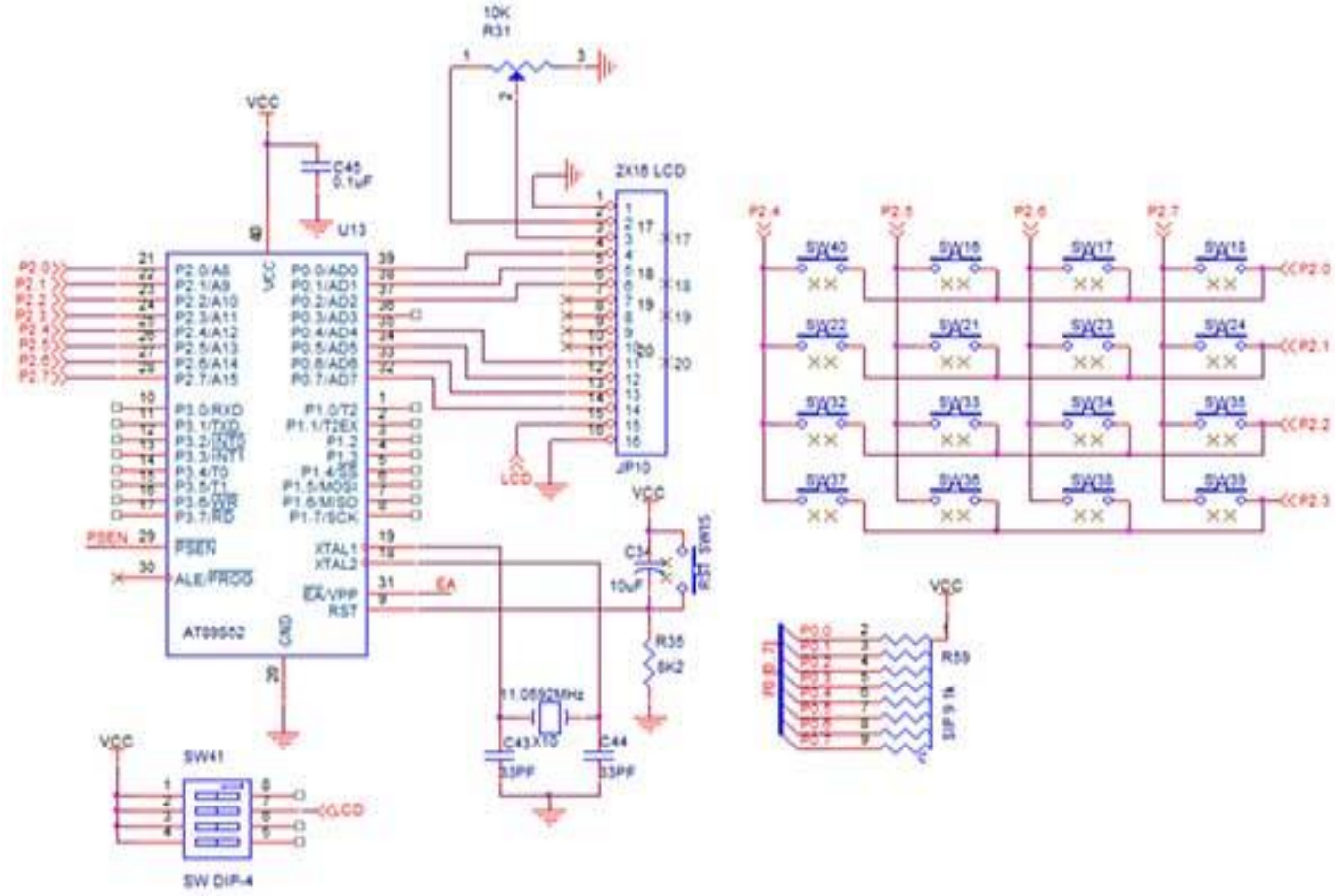


Figure 5 Interfacing of Keyboard with 8051.

2.2 Keyboard interfacing program in C

The example 2 given below describes how to read the input from keyboard and send the result to serial port. For this 9600 baud rate is used.

Example 2

Write a C program to read the keypad and send result to first serial port P1.0 to P1.3 connected to rows, and P2.0 to P2.3 connected to columns. Configure serial port for 9600 baud, 8 data bits and one stop bit.

```
#include <reg51.h>

#define COL P2                          //define ports for easier reading
#define ROW P1

void MSDelay(unsigned int value);
void SerTX(unsigned char);

unsigned char keypad[4][4] =      {'0','1','2','3',
                                   '4','5','6','7',
                                   '8','9','A','B',
                                   'C','D','E','F'};

void main()
  {
    unsigned char colloc, rowloc;
    TMOD = 0x20;                        //timer 1, mode 2
    TH1 = -3;                           //9600 baud
    SCON = 0x50;                        //8-bit, 1 stop bit
    TR1 = 1;                            //start timer 1
  //keyboard routine. This sends the ASCII
  //code for pressed key to the serial port
  COL = 0xFF;                           //make P2 an input port
  while(1)                              //repeat forever
    {
    do
      {
      ROW = 0x00;                       //ground all rows at once
      colloc = COL;                     //read the columns
      colloc &= 0x0F;                   //mask used bits
      }while(colloc != 0x0F);           //check until all keys released
```

```
do
  {
  do
    {
    MSDelay(20);                    //call delay
    colloc = COL;                   //see if any key is pressed
    colloc &= 0x0F;                 //mask unused bits
    } while(colloc == 0x0F);        //keep checking for keypress
  MSDelay(20);                      //call delay for debounce
  colloc = COL;                     //read columns
  colloc &= 0x0F;                   //mask unused bits
  } while(colloc == 0x0F);          //wait for keypress
  while(1)
    {
      ROW = 0xFE;                   //ground row 0
      colloc = COL;                 //read columns
      colloc &= 0x0F;               //mask unused bits
      if(colloc != 0x0F)            //column detected
        {
          rowloc = 0;               //save row location
          break;                    //exit while loop
        }
ROW = 0xFD;                         //ground row 1
colloc = COL;                       //read columns
colloc &= 0x0F;                     //mask unused bits
if(colloc != 0x0F)                  //column detected
  {
    rowloc = 1;                     //save row location
    break;                          //exit while loop
  }

ROW = 0xFB;                         //ground row 2
colloc = COL;                       //read columns
colloc &= 0x0F;                     //mask unused bits
if(colloc != 0x0F)                  //column detected
  {
    rowloc = 2;                     //save row location
    break;                          //exit while loop
  }

ROW = 0xF7;                         //ground row 3
colloc = COL;                       //read columns
colloc &= 0x0F;                     //mask unused bits
rowloc = 3;                         //save row location
break;                              //exit while loop
}
```

```
    //check column and send result to the serial port
    if(colloc == 0x0E)
      SerTX(keypad[rowloc][0]);
    else if(colloc == 0x0D)
      SerTX(keypad[rowloc][1]);
    else if(colloc == 0x0B)
      SerTX(keypad[rowloc][2]);
    else
      SerTX(keypad[rowloc][3]);
    }
  }
void SerTX(unsigned char x)
  {
    SBUF = x;                    //place value in buffer
    while(TI==0);                //wait until transmitted
    TI = 0;                      //clear flag
  }

void MSDelay(unsigned int value)
  {
    unsigned int x, y;
    for(x=0;x<1275;x++)
      for(y=0;y<value;y++);
  }
```

This section described about LCD and keyboard interfacing with 8051.The following section describes interfacing of 8051 with external ROM.

3.8051 Interfacing with External ROM

The reason for interfacing 8051 with external ROM is that it has a limited amount of on-chip ROM. Hence for sufficient memory allocation 8051 is interfaced with external ROM. CPU gives the address of the data required. Decoding circuit has to locate the selected memory block. Memory chips have CS (chip select) pin, which must be activated for the memory's contents to be accessed.

EA pin:

Connect the EA pin to Vcc to indicate that the program code is stored in the microcontroller on-chip ROM. To indicate that the program code is stored in external ROM, this pin must be connected to GND. The figure 6 shown below gives the details of 8051 pin diagram.

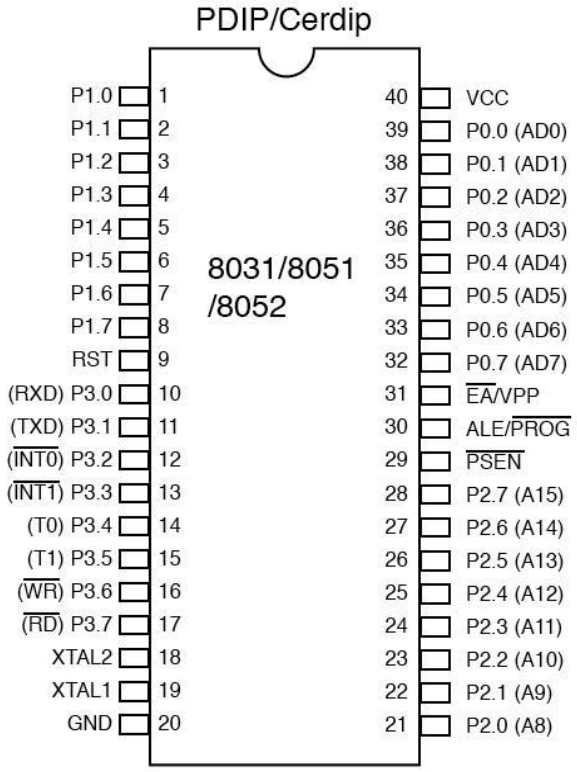


Figure 6. 8051 Pin Diagram

3.1 Steps to be followed when connecting external memory

The following are the steps to follow when connecting to external memory:

1. CPU data bus is connected directly to the data pins of the memory chip.
2. Control signals RD (read) and WR (memory write) from the CPU are connected to the OE (output enable) and WE (write enable) pins of the memory chip.
3. The lower bits of the address from the CPU is connected directly to the memory chip of the address pins.
4. The upper ones are used to activate the CS pin of the memory chip.

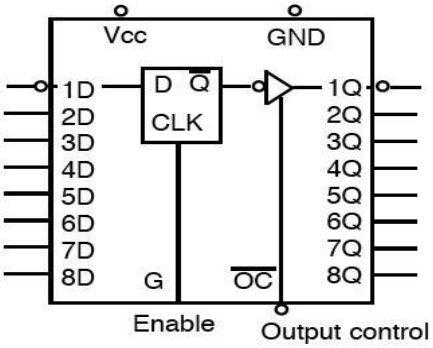
P0 and P2 role in providing addresses:

The PC (program counter) of the 8031/51 is 16-bit, it is capable of accessing up to 64K bytes of program code. 16-bit address is provided by port 0 and port 2 to access external memory. P0 provides the lower 8-bit address A0 – A7 whereas P2 provides the upper 8-bit address A8 – A15. P0 is also used to provide the 8-bit data bus D0–D7. P0.0 – P0.7 are used for both the address and data paths (address/data multiplexing), to extract the addresses from the P0 pins. It will connect to 74LS373 D Latch.

ALE (Address Latch Enable):

ALE (address latch enable) pin is an output pin for 8031/51. If ALE = 0, P0 is used for data path, whereas if ALE = 1, P0 is used for address path. Separation of address and data can be obtained by using 74LS373. The Figure 7 shown below gives the specifics about 74LS373 D

latch.



Funtion Table

Output control	Enable G	D	Output
L	H	H	H
L	H	L	L
L	L	X	Q0
H	X	X	Z

Figure 7. 74LS373 D Latch

The Data, Address, Control Buses for the 8031/8051 are shown in Figure 8 below.

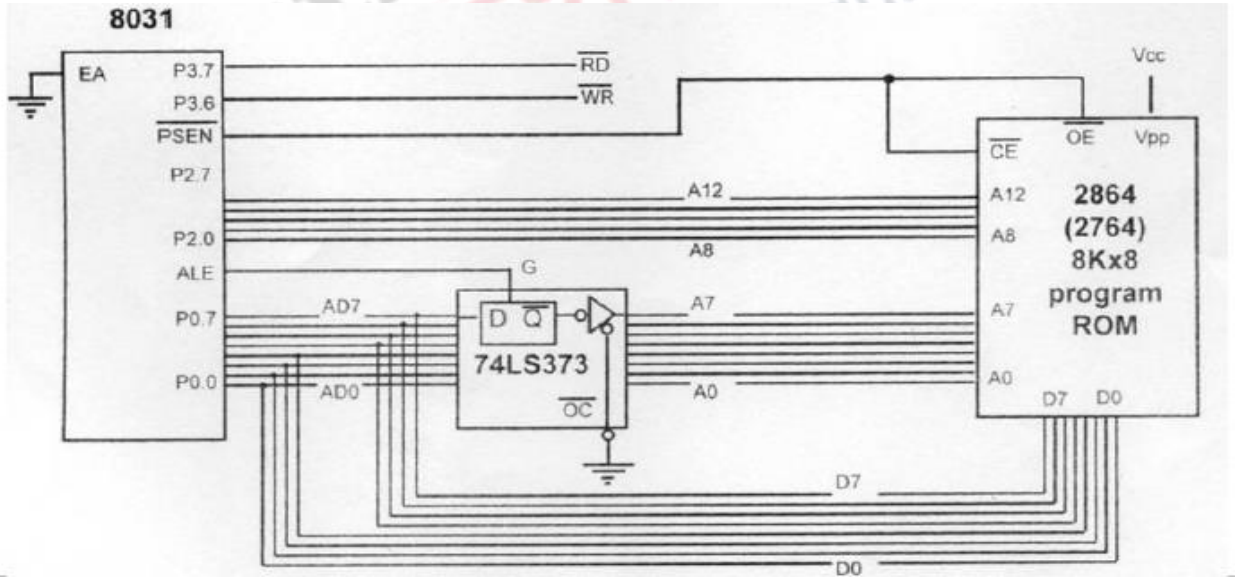


Figure 8. 8051 connections to External program ROM

8051 connection to external data RAM:

8051 connection to external data RAM is shown in Figure 9 below.

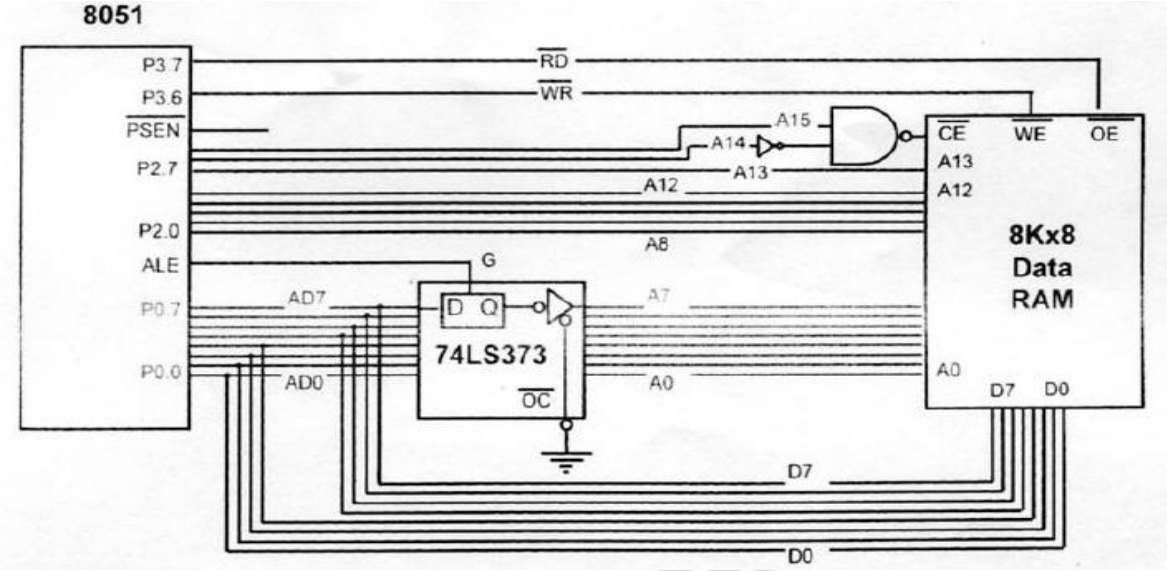


Figure 9. 8051 connections to external data RAM

A single ROM for both program and data:

For certain applications we need program ROM, Data ROM and Data RAM in a system. To allow a single ROM chip to provide both program code space and data code space, we use an AND gate to signal the OE pin of the ROM chip. The figure 10 shown below gives the details about connection of 8051 with single ROM for both program and data.

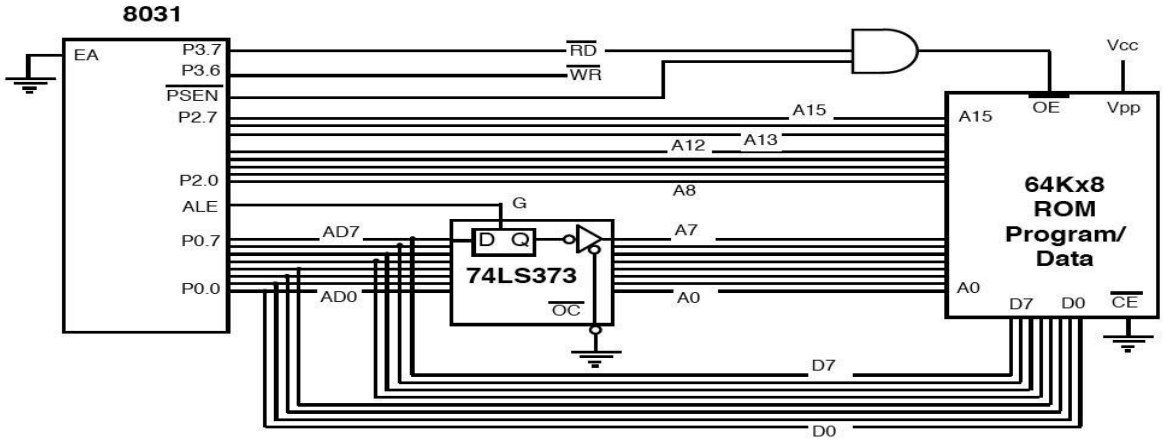


Figure 10. A single ROM for both program and data

3.2 External memory interfacing program in C

In some applications, we need a large amount of memory to store data. However, the 8051 supports only 64K bytes of external data memory,since DPTR is 16-bit. To solve this problem, we connect A_0 - A_{15} of the 8051 directly to the external memory's A_0 - A_{15} pins and use some of the P1 pin to access this 64K bytes block inside the single 256K X 8 memory chip.

Example 3 shown below describes how to store ASCII letters to external RAM address starting at 0. It then gets data from external RAM and sends it to PI, one byte at a time.

Example 3:

Write a C program (a) to store ASCII letters 'A' to 'E' in external RAM addresses starting at 0, then (b) get the same data from the external RAM and send it to PI one byte at a time.

Solution:

```
#include <reg51.h>
#include <absacc.h> //notice the header file for XBYTE

void main(void)
  {
    unsigned char x;
    XBYTE[0]='A';  //write ASCII 'A' to External RAM location 0
    XBYTE[1]='B';  //write ASCII 'B' to External RAM location 1
    XBYTE[2]='C';  //write ASCII 'C' to External RAM location 2
    XBYTE[3]='D';
    XBYTE[4]='E';
    for(x=0;x<5;x++)
      P1=XBYTE[x];  //read external RAM data and send it to P1
  }
```

Example 4 given below describes how to write the C program to read 30 bytes of data from an external ROM and send it to port P1. An external ROM uses 8051 data space to store the value in lookup table for the DAC data.

Example 4

An external ROM uses the 8051 data space to store the look-up table (starting at 100H) for DAC data. Write a C program to read 30 bytes of table data and send it to PI.

Solution:

```
#include <reg51.h>
#include <absacc.h>  //notice the header file for XBYTE

void main(void)
  {
    unsigned char count;
    for(count=0;count<30;count++)
      P1=XBYTE[0x100+count];
  }
```

Example 5 given below describes how to write the C program to move the message into

external RAM and read the same in RAM and send it to serial port. For this SFR register is used to declare the new addresses and timer 1 is used for baud rate generation.

Example 5

Assume that we have an external RAM with addresses 0000 – 2FFFH for a given 8051-based system, (a) Write a C program to move the message "Hello" into external RAM, and (b) read the same data in external RAM, and send it to the serial port.

Solution:

```
#include <reg51.h>
#include <absacc.h>  //notice the header file for XBYTE

unsigned char msg[5]="Hello";




void main(void)
  {
    unsigned char x;
    TMOD = 0x20;  //USE TIMER 1,8-BIT AUTO-RELOAD
    TH1  = 0xFD;  //9600
    SCON = 0x50;
    TR1  = 1;

    for(x=0;x<5;x++)
      XBYTE[0x000+x] = msg[x];

    for(x=0;x<5;x++)
      {
        SBUF = XBYTE[0x000+x];
        while(TI==0);
        TI=0;
      }
  }
```

Example 6 given below describes how to write the C program to access 1KB SRAM of the DS89C4XO chip. But the ASCII letters are stored before itself in SRAM, for which SBUF register is used. For this SFR register is used to declare the new addresses and timer 1 is used for baud rate generation.

Example 6

Write a C program (a) to enable the access to the 1KB SRAM of the DS89C4xO, (b) put the ASCII letters 'A', 'B' and 'C' in SRAM, and (c) read the same data from SRAM and send each one to ports PO, PI, and P2. This is the C version of an earlier example.

Solution:

```
#include <reg51.h>
#include <absacc.h>  //notice the header file for XBYTE
sfr PMRREG = 0xC4;
void main(void)
  {
    unsigned char x;
    PMRREG = 0x81;
    XBYTE[0]='A';  //write ASCII 'A' to External RAM location 0
    XBYTE[1]='B';  //write ASCII 'B' to External RAM location 1
    XBYTE[2]='C';  //write ASCII 'C' to External RAM location 2

    for(x=0;x<3;x++)
      {
        P0=XBYTE[x];  //read ext RAM data and send it to P0
        P1=XBYTE[x];  //read ext RAM data and send it to P1
        P2=XBYTE[x];  //read ext RAM data and send it to P2
      }
  }
```

4. Summary

In this lecture, basics of interfacing of 8051 to external devices are discussed. The Interfacing of 8051 to external memory has also been discussed. Interfacing of 8051 with external devices in Embedded C is discussed in detail with examples.

5. References

1. Muhammad Ali Mazidi, Janice Gillispie Mazidi, Rolin D. McKinlay, "The 8051 Microcontroller and Embedded Systems Using Assembly and C -Second Edition", New Delhi(2000).