ISC 502
Applications of Microcontroller

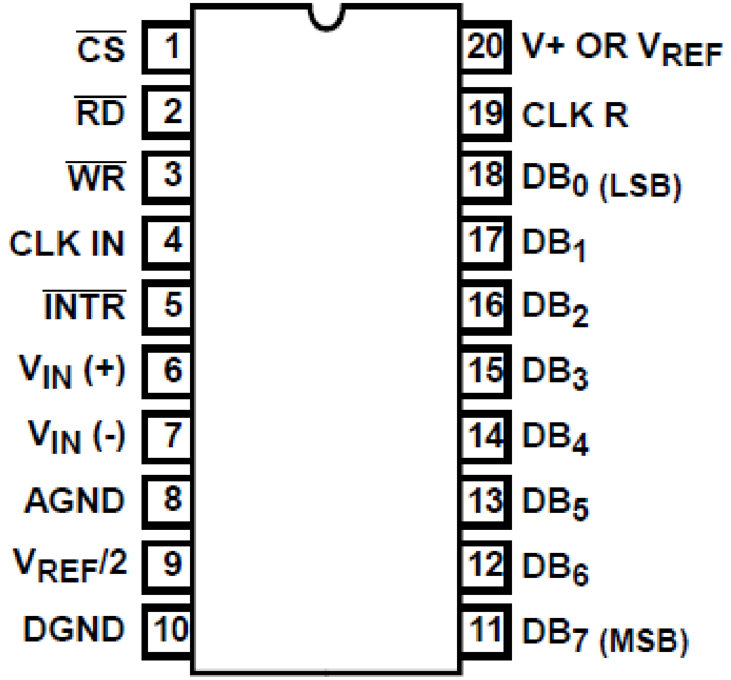
Mrs. Kanchan Chavan

INTERFACING TO ADC AND SENSORS

ADC804 Chip

- ADC804 IC is an analog-to-digital converter
  - It works with +5 volts and has a resolution of 8 bits
  - *Conversion time* is another major factor in judging an ADC
    - Conversion time is defined as the time it takes the ADC to convert the analog input to a digital (binary) number
    - In ADC804 conversion time varies depending on the clocking signals applied to CLK R and CLK IN pins, but it cannot be faster than 110 µs

ADC0802, ADC0803, ADC0804
(PDIP, CERDIP)
TOP VIEW



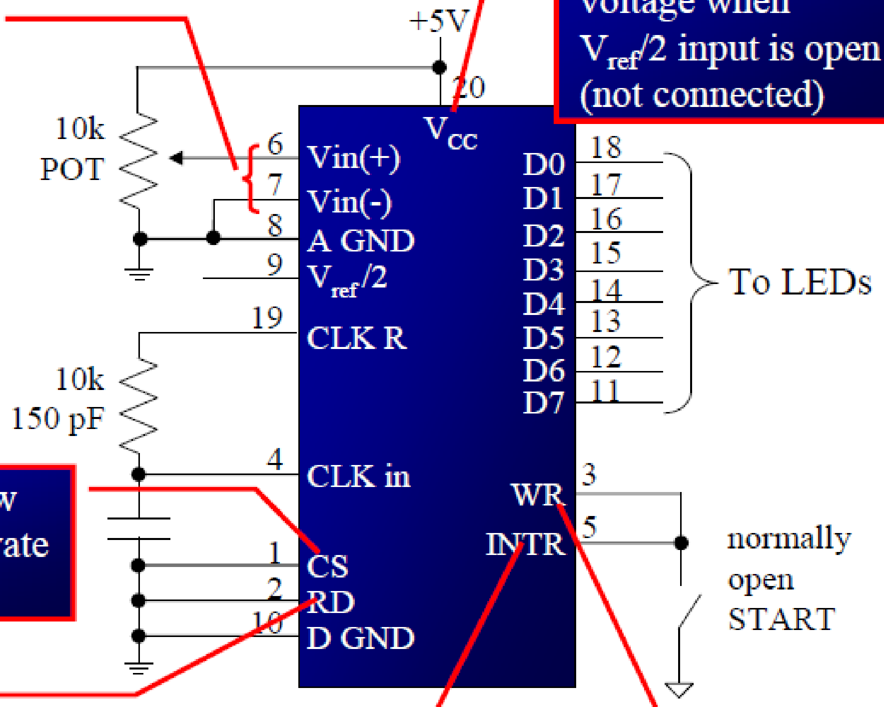
INTERFACING TO ADC AND SENSORS

ADC804 Chip (cont')

+5V power supply or a reference voltage when $V_{ref}/2$ input is open (not connected)

Differential analog inputs where $V_{in} = V_{in}(+) - V_{in}(-)$

Vin (-) is connected to ground and Vin (+) is used as the analog input to be converted



CS is an active low input used to activate ADC804

"output enable"
a high-to-low RD pulse is used to get the 8-bit converted data out of ADC804

"end of conversion"
When the conversion is finished, it goes low to signal the CPU that the converted data is ready to be picked up

"start conversion"
When WR makes a low-to-high transition, ADC804 starts converting the analog input value of V_{in} to an 8-bit digital number

- CLK IN and CLK R
  - CLK IN is an input pin connected to an external clock source
  - To use the internal clock generator (also called self-clocking), CLK IN and CLK R pins are connected to a capacitor and a resistor, and the clock frequency is determined by

$$f = \frac{1}{1.1\,RC}$$

    - Typical values are R = 10K ohms and C = 150 pF
    - We get f = 606 kHz and the conversion time is 110 μs

- $V_{ref}/2$
  - It is used for the reference voltage
    - If this pin is open (not connected), the analog input voltage is in the range of 0 to 5 volts (the same as the Vcc pin)
    - If the analog input range needs to be 0 to 4 volts, $V_{ref}/2$ is connected to 2 volts

$V_{ref}/2$ Relation to V_{in} Range

$V_{ref}/2$(v)	V_{in}(V)	Step Size (mV)
Not connected*	0 to 5	5/256=19.53
2.0	0 to 4	4/255=15.62
1.5	0 to 3	3/256=11.71
1.28	0 to 2.56	2.56/256=10
1.0	0 to 2	2/256=7.81
0.5	0 to 1	1/256=3.90

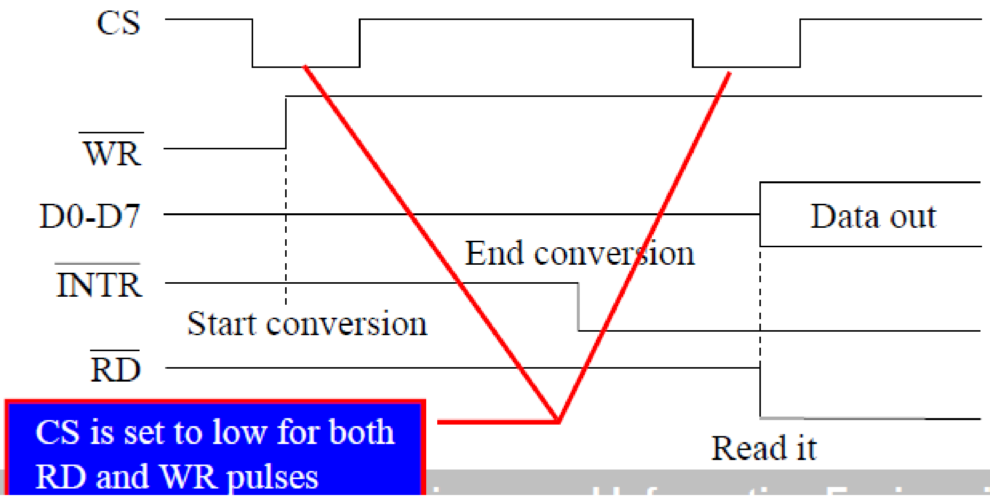
Step size is the smallest change can be discerned by an ADC

- D0-D7
  - The digital data output pins
  - These are tri-state buffered
    - The converted data is accessed only when CS = 0 and RD is forced low
  - To calculate the output voltage, use the following formula

$$D_{out} = \frac{V_{in}}{step\ size}$$

    - *Dout* = digital data output (in decimal),
    - *Vin* = analog voltage, and
    - *step size* (resolution) is the smallest change

- The following steps must be followed for data conversion by the ADC804 chip
  - Make CS = 0 and send a low-to-high pulse to pin WR to start conversion
  - Keep monitoring the INTR pin
    - If INTR is low, the conversion is finished
    - If the INTR is high, keep polling until it goes low
  - After the INTR has become low, we make CS = 0 and send a high-to-low pulse to the RD pin to get the data out of the ADC804

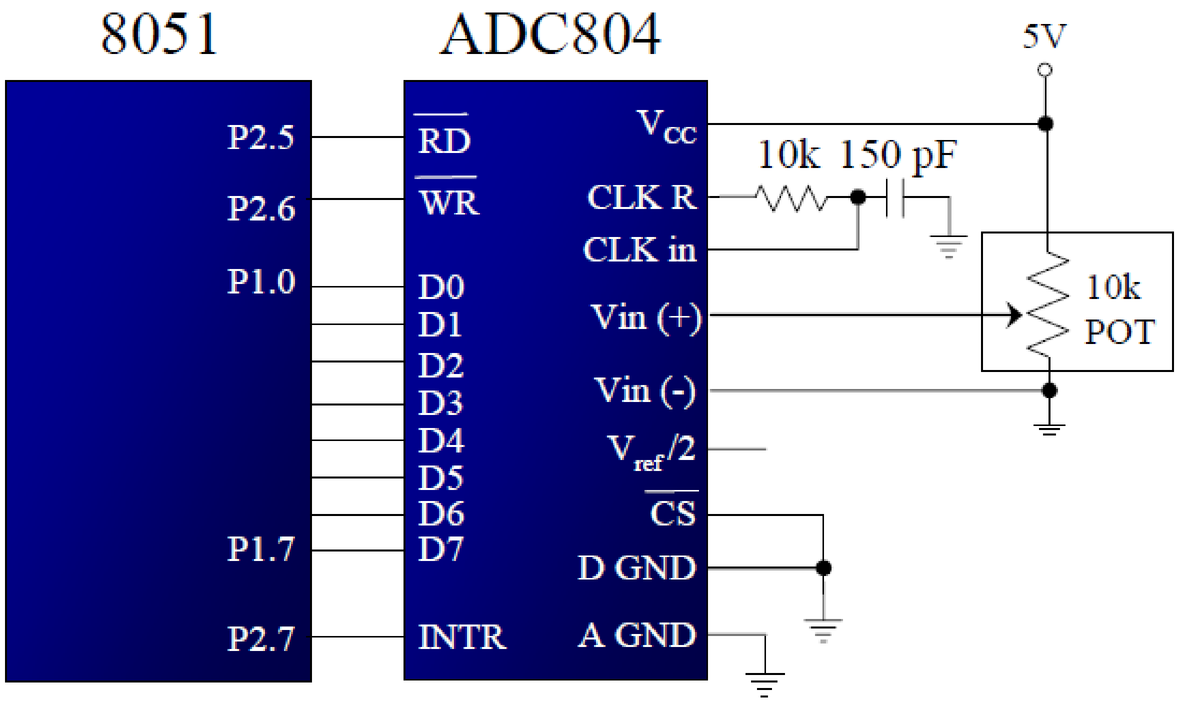


Examine the ADC804 connection to the 8051 in Figure 12-7. Write a program to monitor the INTR pin and bring an analog input into register A. Then call a hex-to ACSII conversion and data display subroutines. Do this continuously.

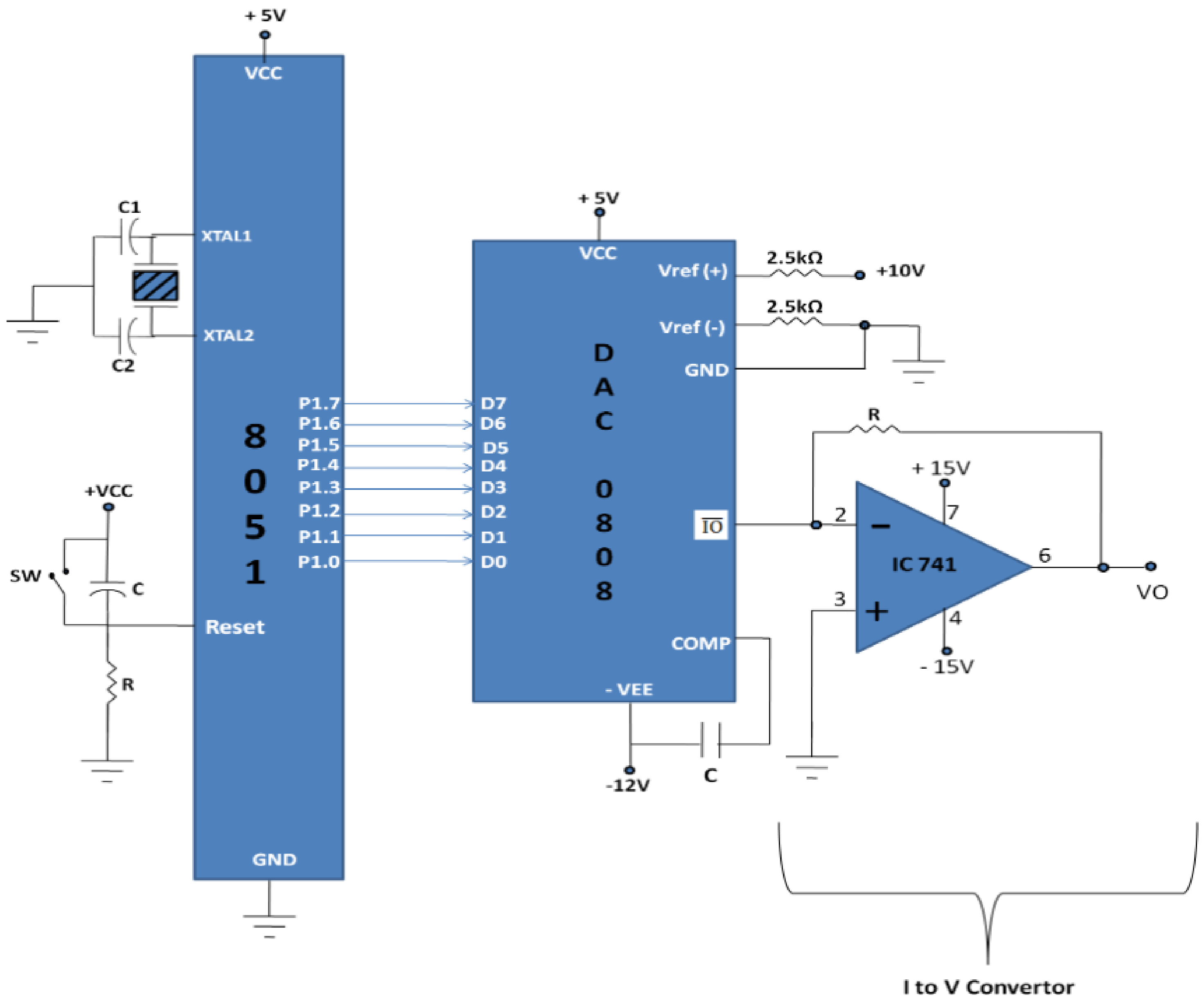
```
;p2.6=WR (start conversion needs to L-to-H pulse)
;p2.7 When low, end-of-conversion)
;p2.5=RD (a H-to-L will read the data from ADC chip)
;p1.0 - P1.7= D0 - D7 of the ADC804
;
      MOV   P1,#0FFH      ;make P1 = input
BACK: CLR   P2.6          ;WR = 0
      SETB  P2.6          ;WR = 1 L-to-H to start conversion
HERE: JB    P2.7,HERE     ;wait for end of conversion
      CLR   P2.5          ;conversion finished, enable RD
      MOV   A,P1          ;read the data
      ACALL CONVERSION    ;hex-to-ASCII conversion
      ACALL DATA_DISPLAY;display the data
      SETB  p2.5          ;make RD=1 for next round
      SJMP  BACK
```

```c
#include <reg51.h>
sbit RD = P2^5;
sbit WR = P2^6;
sbit INTR = P2^7;
sfr MYDATA = P1;
void main()
  {
    unsigned char value;
    MYDATA = 0xFF;            //make P1 and input
    INTR = 1;                 //make INTR and input
    RD = 1;                   //set RD high
    WR = 1;                   //set WR high
    while(1)
     {
       WR = 0;                //send WR pulse
       WR = 1;                //L-to-H(Start Conversion)
       while(INTR == 1);      //wait for EOC
       RD = 0;                //send RD pulse
       value = MYDATA;        //read value
       ConvertAndDisplay(value);  //(Chap 7 and 12)
       RD = 1;
     }
  }
```

8051 Connection to ADC804 with Self-Clocking



DAC Interfacing with 8051



+ 5V
VCC
C1
XTAL1
XTAL2
C2
P1.7
P1.6
P1.5
P1.4
P1.3
P1.2
P1.1
P1.0
8051
+VCC
SW
C
Reset
R
GND
+ 5V
VCC
Vref (+)
Vref (-)
GND
2.5kΩ
+10V
2.5kΩ
D7
D6
D5
D4
D3
D2
D1
D0
DAC 0808
IO
COMP
- VEE
-12V
C
R
+ 15V
7
2
IC 741
6
3
4
- 15V
VO
I to V Convertor

write a program to send data to the DAC to generate a stair-step ramp.

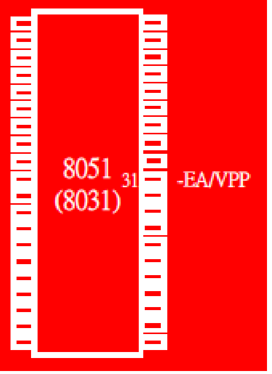
```
        CLR A
AGAIN:  MOV P1,A      ; SEND DATA TO DAC
        INC A         ; COUNT FROM 0 TO FFH
        ACALL DELAY   ; LET DAC RECOVER
        SJMP AGAIN
```

$$I_{out} = I_{ref}\left(\frac{D7}{2}+\frac{D6}{4}+\frac{D5}{8}+\frac{D4}{16}+\frac{D3}{32}+\frac{D2}{64}+\frac{D1}{128}+\frac{D0}{256}\right)$$

8031/51 INTERFACING TO EXTERNAL MEMORY

INTERFACING EXTERNAL ROM

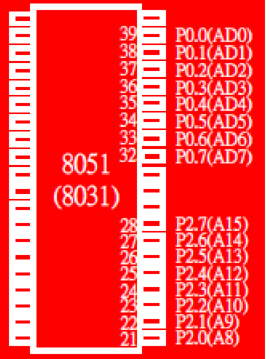
EA Pin



- For 8751/89C51/DS5000-based system, we connected the EA pin to V_{cc} to indicate that the program code is stored in the microcontroller's on-chip ROM
  - To indicate that the program code is stored in external ROM, this pin must be connected to GND

INTERFACING EXTERNAL ROM

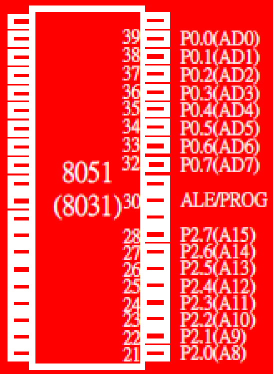
P0 and P2 in Providing Address



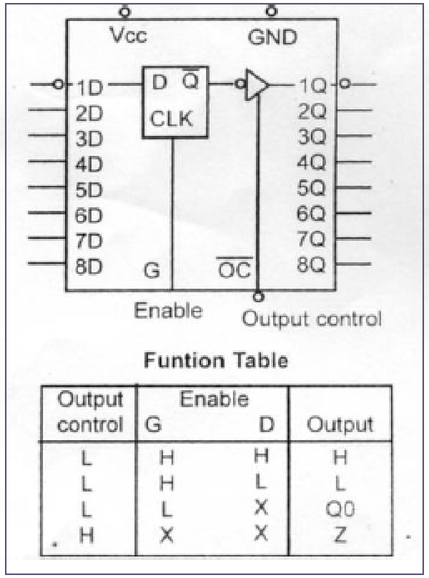
- Since the PC (program counter) of the 8031/51 is 16-bit, it is capable of accessing up to 64K bytes of program code
  - In the 8031/51, port 0 and port 2 provide the 16-bit address to access external memory
    - P0 provides the lower 8 bit address A0 – A7, and P2 provides the upper 8 bit address A8 – 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

INTERFACING EXTERNAL ROM

P0 and P2 in Providing Address (cont')



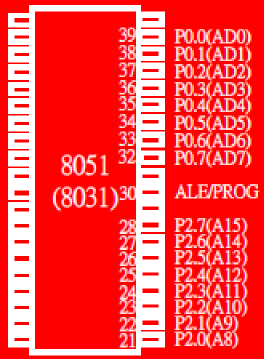
- ALE (address latch enable) pin is an output pin for 8031/51
  - ALE = 0, P0 is used for data path
  - ALE = 1, P0 is used for address path
- To extract the address from the P0 pins we connect P0 to a 74LS373 and use the ALE pin to latch the address



74LS373 D Latch

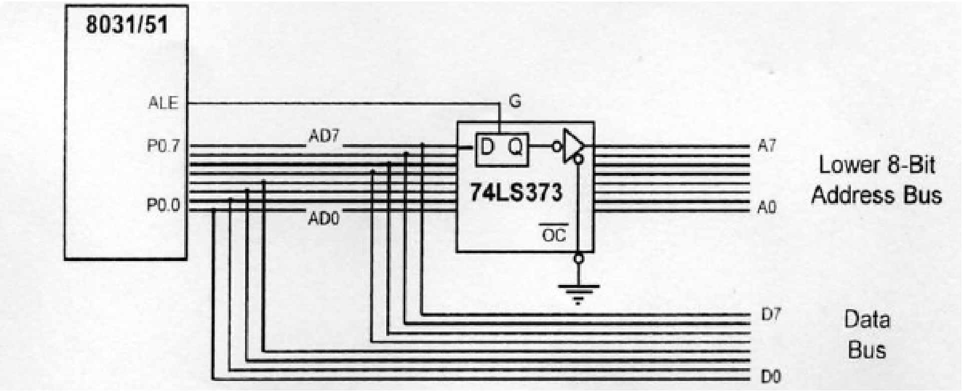
INTERFACING EXTERNAL ROM

P0 and P2 in Providing Address (cont')



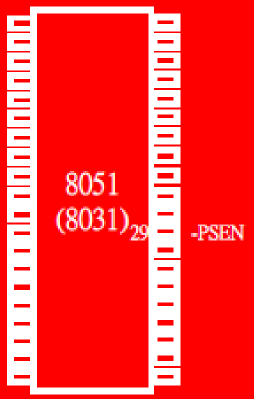
- Normally ALE = 0, and P0 is used as a data bus, sending data out or bringing data in
- Whenever the 8031/51 wants to use P0 as an address bus, it puts the addresses A0 – A7 on the P0 pins and activates ALE = 1

Address/Data Multiplexing



INTERFACING EXTERNAL ROM

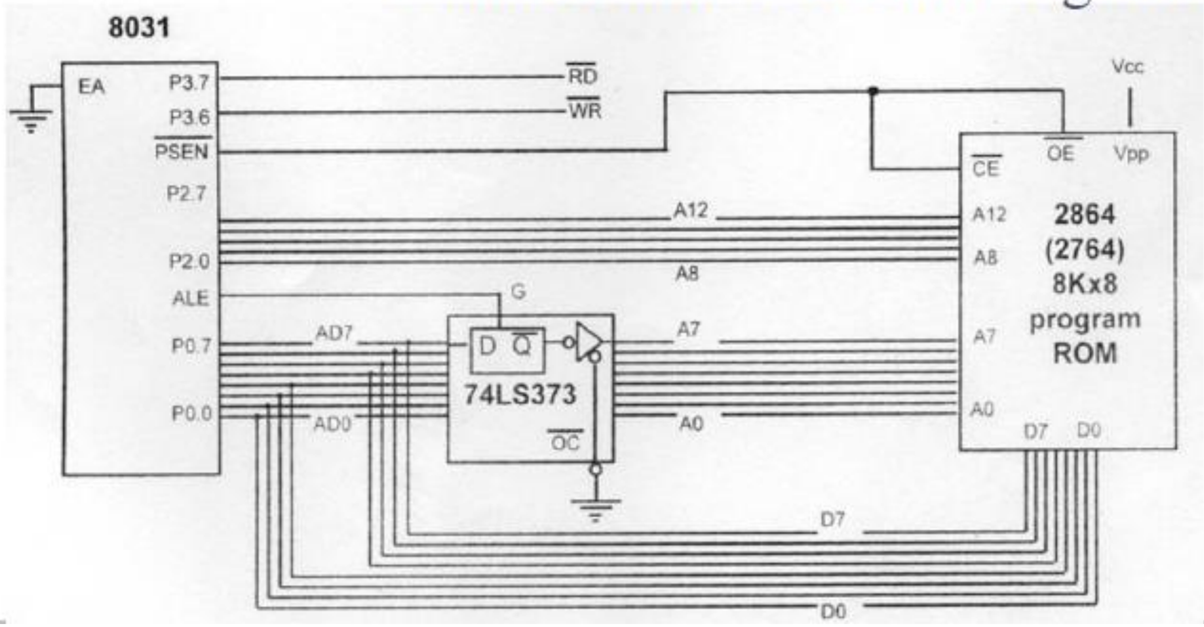
PSEN



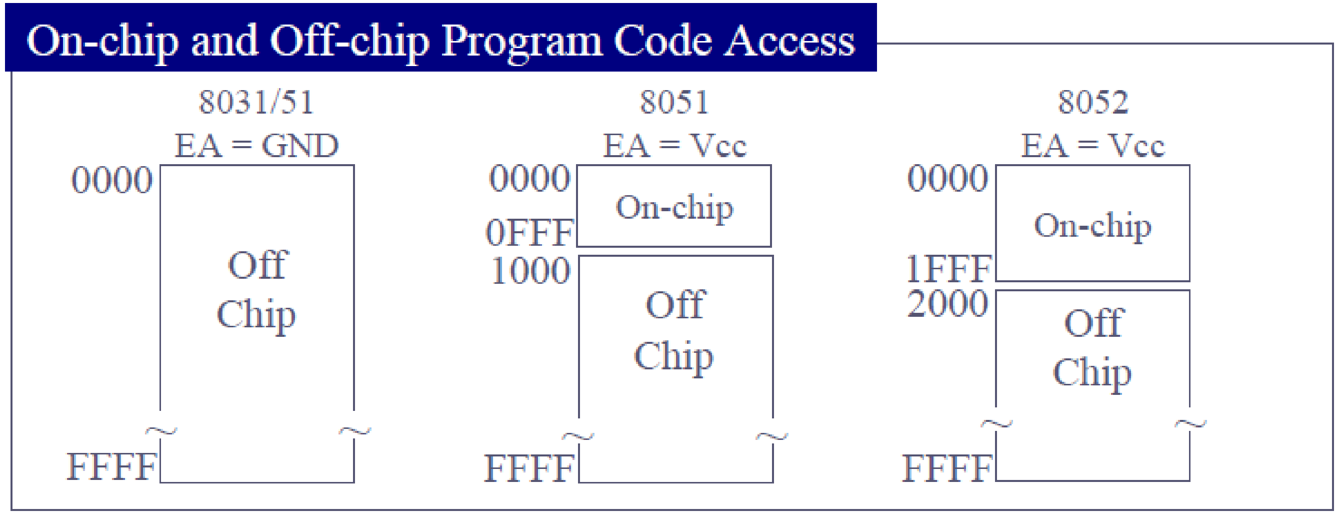
- PSEN (program store enable) signal is an output signal for the 8031/51 microcontroller and must be connected to the OE pin of a ROM containing the program code
- It is important to emphasize the role of EA and PSEN when connecting the 8031/51 to external ROM
  - When the EA pin is connected to GND, the 8031/51 fetches opcode from external ROM by using PSEN

- The connection of the PSEN pin to the OE pin of ROM
  - In systems based on the 8751/89C51/DS5000 where EA is connected to V_{cc}, these chips do not activate the PSEN pin
    - This indicates that the on-chip ROM contains program code

Connection to External Program ROM



- In an 8751 system we could use on-chip ROM for boot code and an external ROM will contain the user's program
  - We still have EA = V_{cc},
    - Upon reset 8051 executes the on-chip program first, then
    - When it reaches the end of the on-chip ROM, it switches to external ROM for rest of program

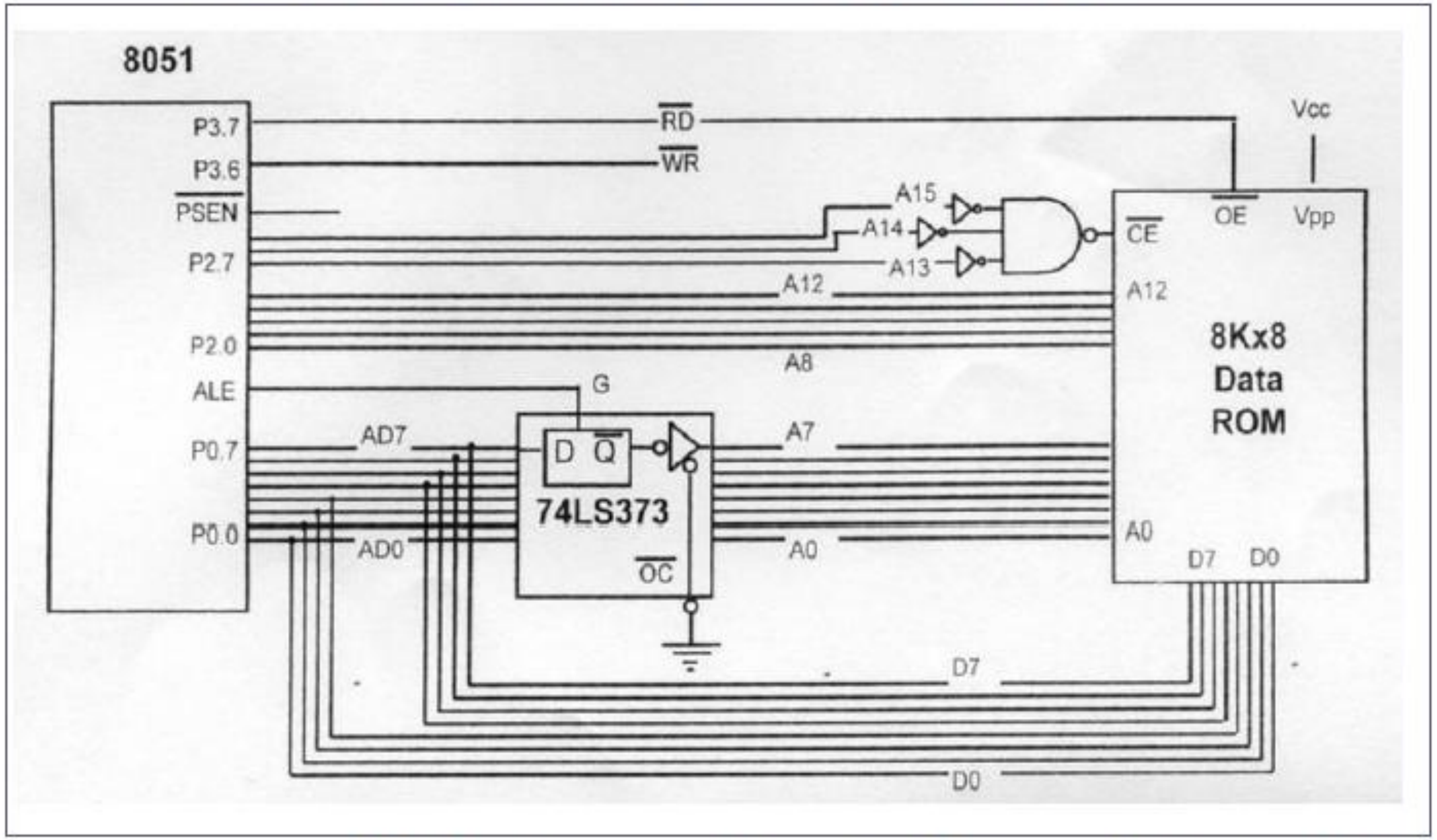


8051 DATA MEMORY SPACE

Data Memory Space

- The 8051 has 128K bytes of address space
  - 64K bytes are set aside for program code
    - Program space is accessed using the program counter (PC) to locate and fetch instructions
    - In some example we placed data in the code space and used the instruction
      `MOVC A,@A+DPTR` to get data, where C stands for code
  - The other 64K bytes are set aside for data
    - The data memory space is accessed using the DPTR register and an instruction called `MOVX`, where X stands for external
      - The data memory space must be implemented externally

- We use RD to connect the 8031/51 to external ROM containing data
  - For the ROM containing the program code, PSEN is used to fetch the code



8051 Connection to External Data ROM

- `MOVX` is a widely used instruction allowing access to external data memory space
  - To bring externally stored data into the CPU, we use the instruction `MOVX A,@DPTR`

An external ROM uses the 8051 data space to store the look-up table (starting at 1000H) for DAC data. Write a program to read 30 Bytes of these data and send it to P1.

Solution:

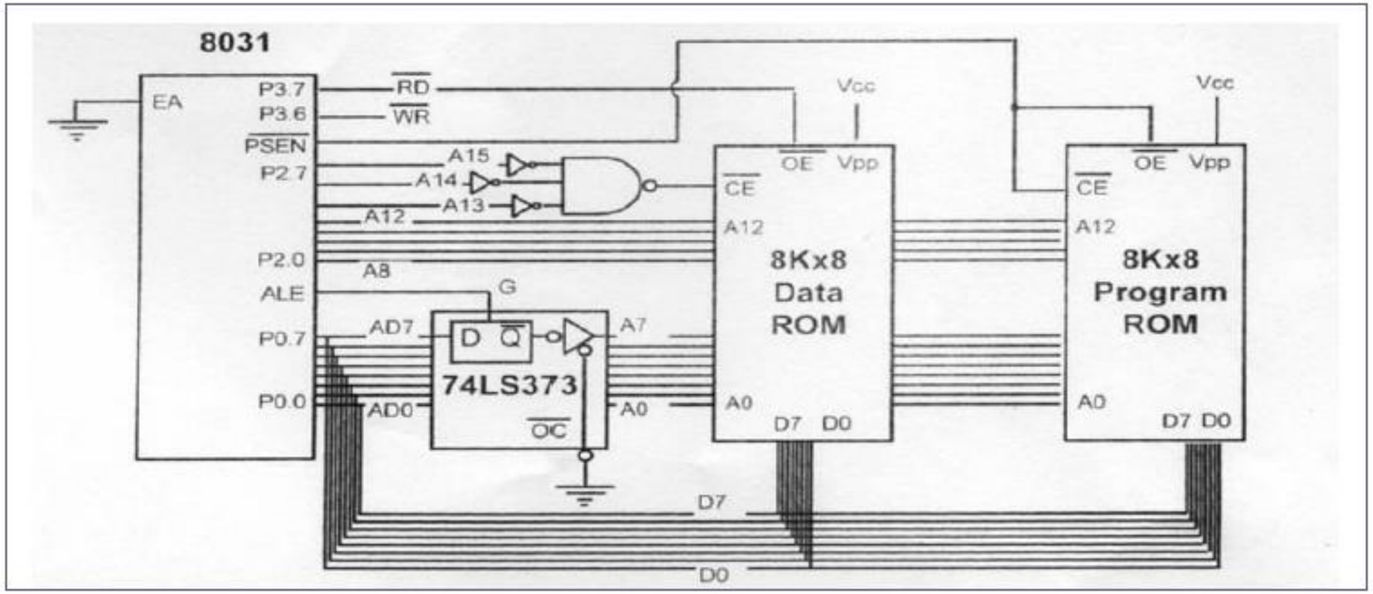
```
MYXDATA EQU   1000H
COUNT   EQU   30
         ...
        MOV   DPTR,#MYXDATA
        MOV   R2,#COUNT
AGAIN:  MOVX  A,@DPTR
        MOV   P1,A
        INC   DPTR
        DJNZ  R2,AGAIN
```

Although both `MOVC A,@A+DPTR` and `MOVX A,@DPTR` look very similar, one is used to get data in the code space and the other is used to get data in the data space of the microcontroller

Show the design of an 8031-based system with 8K bytes of program ROM and 8K bytes of data ROM.

Solution:

Figure 14-14 shows the design. Notice the role of PSEN and RD in each ROM. For program ROM, PSEN is used to activate both OE and CE. For data ROM, we use RD to active OE, while CE is activated by a Simple decoder.

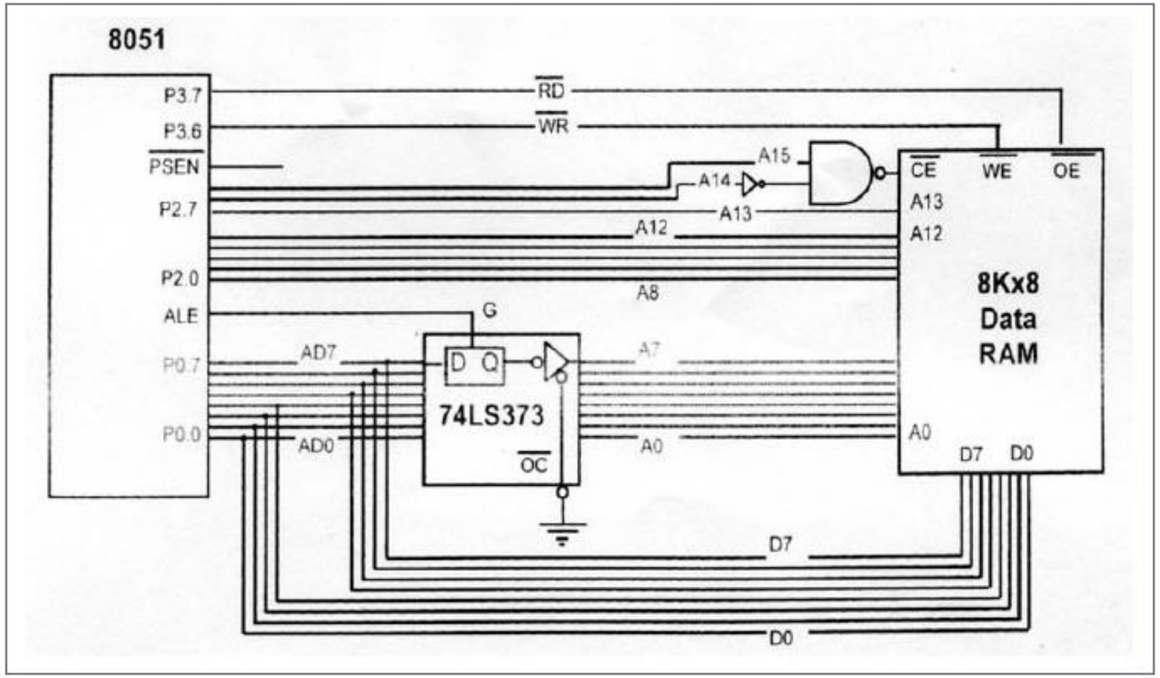


8031 Connection to External Data ROM and External Program ROM

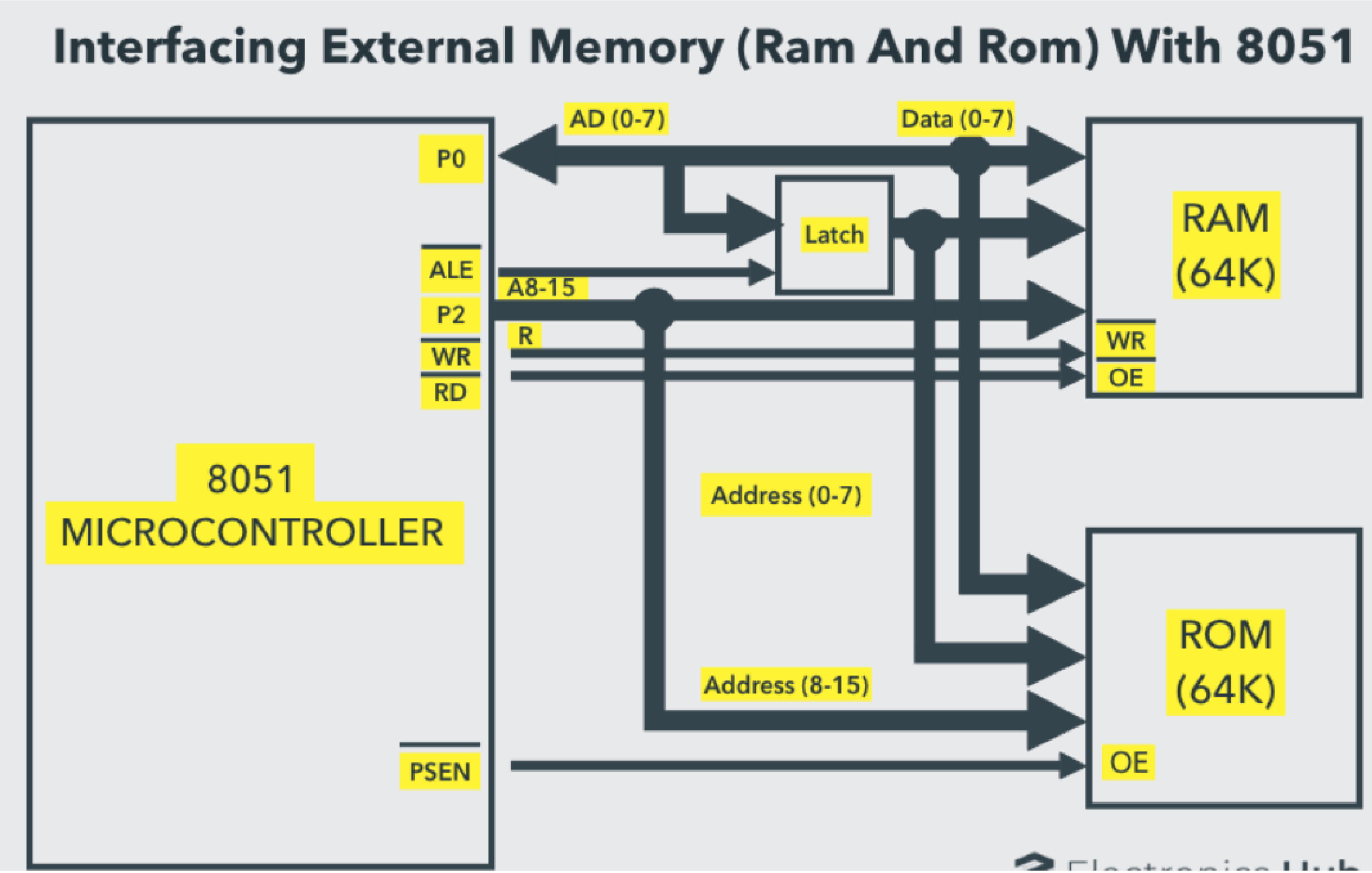
8051 DATA MEMORY SPACE

External Data RAM

- To connect the 8051 to an external SRAM, we must use both RD (P3.7) and WR (P3.6)



8051 Connection to External Data RAM


Interfacing External Memory (Ram And Rom) With 8051
AD (0-7)
Data (0-7)
P0
Latch
RAM
(64K)
ALE
A8-15
P2
R
WR
RD
WR
OE
8051
MICROCONTROLLER
Address (0-7)
ROM
(64K)
Address (8-15)
PSEN
OE

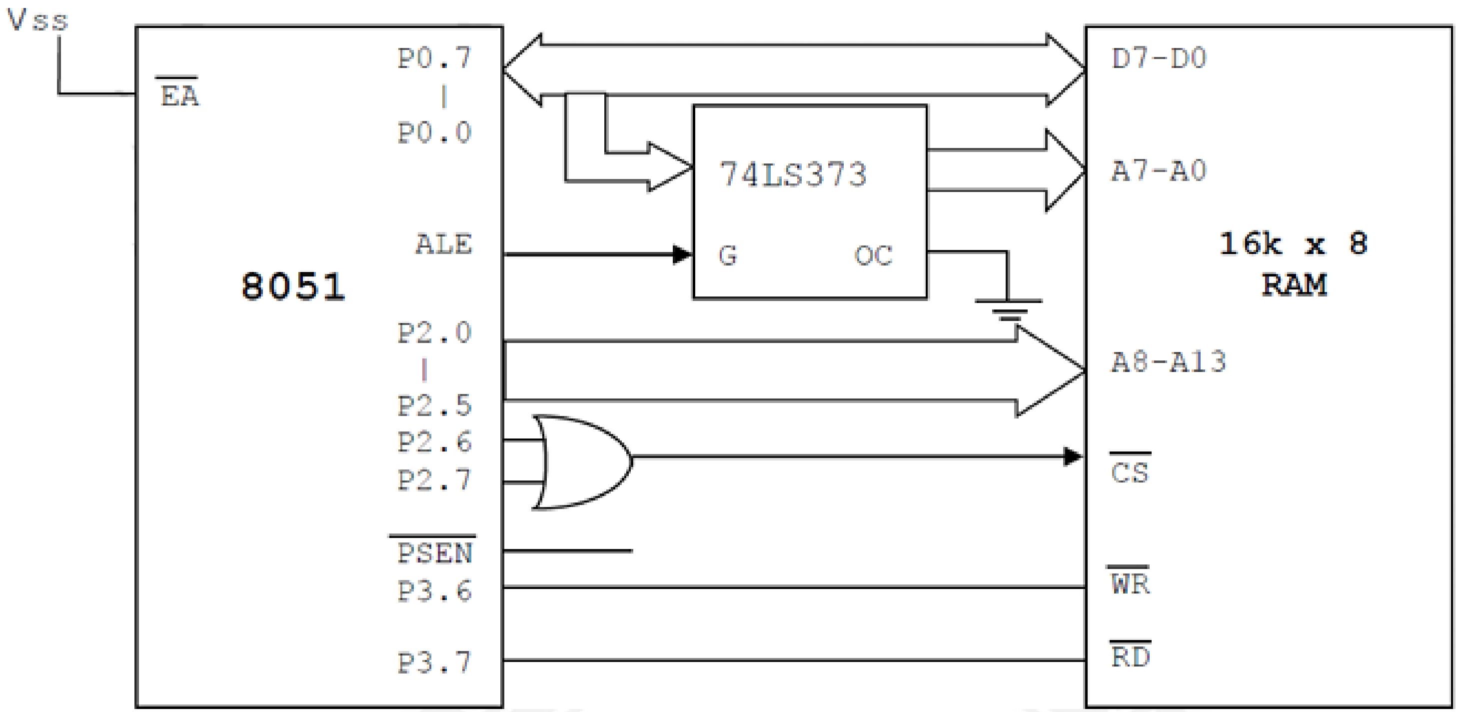


Figure 5.6.3 16Kx8 Memory (RAM) Interfacing with 8051

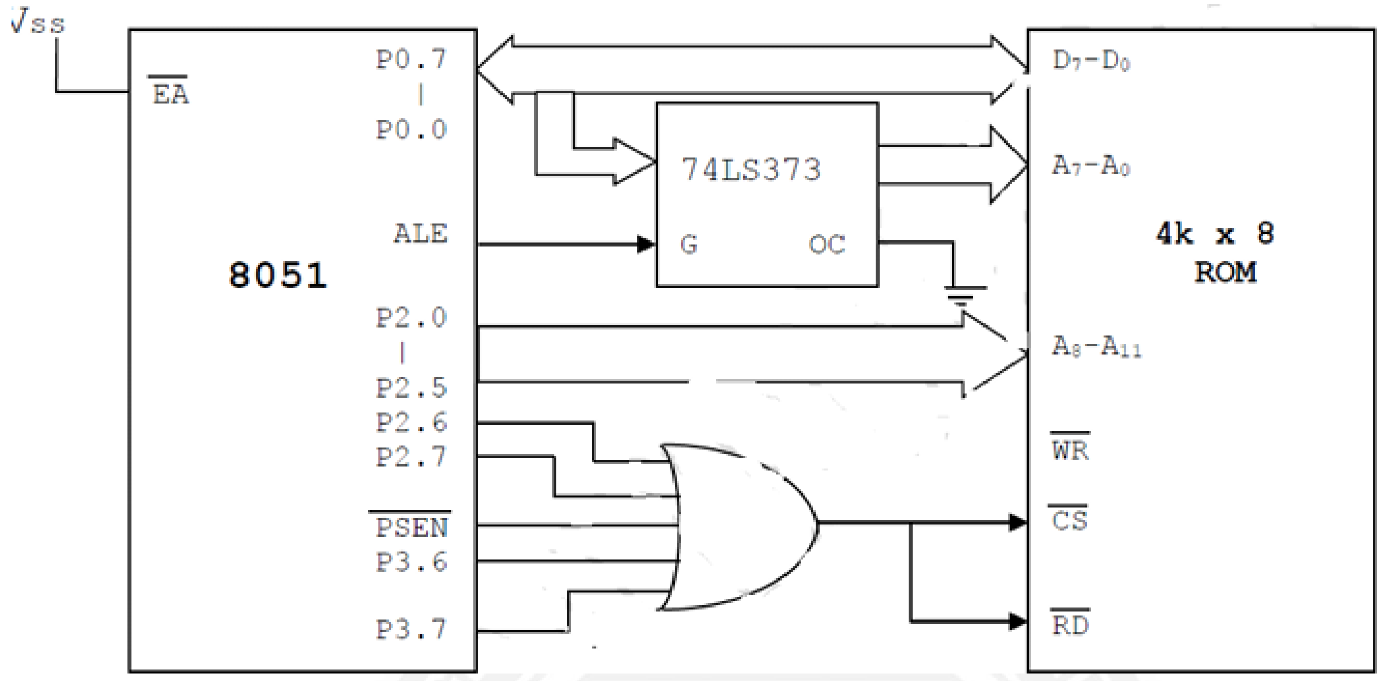


Figure 5.6.4 4Kx8 Memory (ROM) Interfacing with 8051

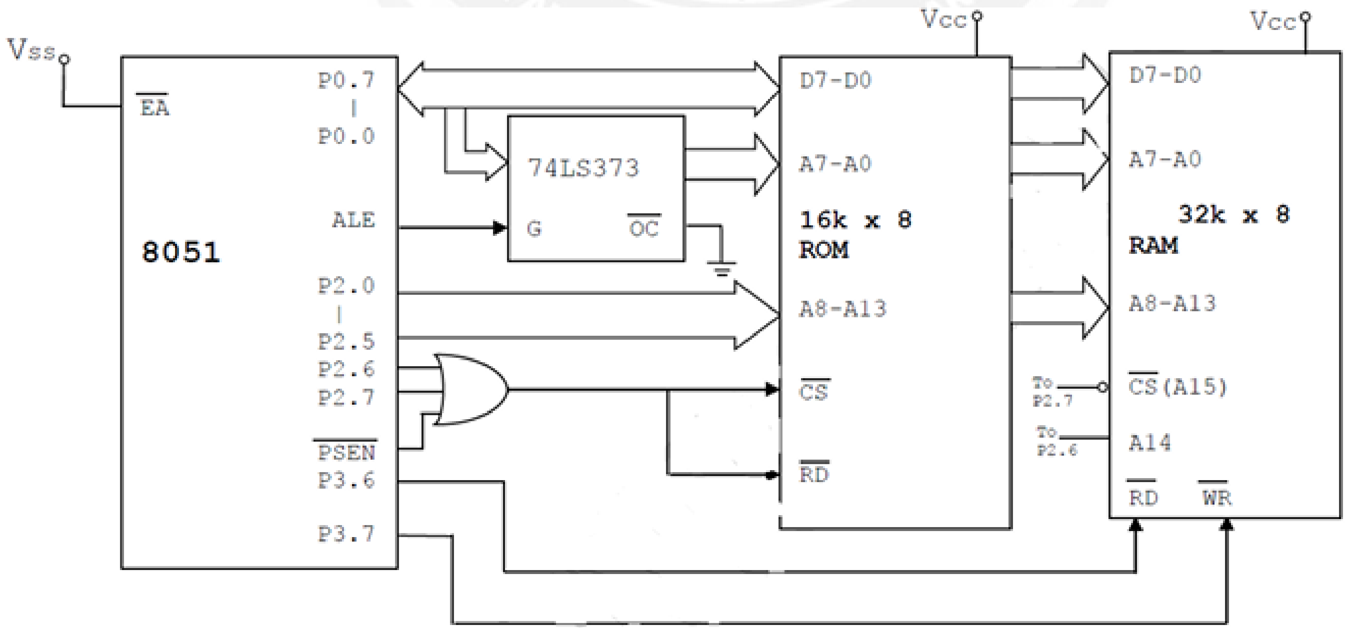
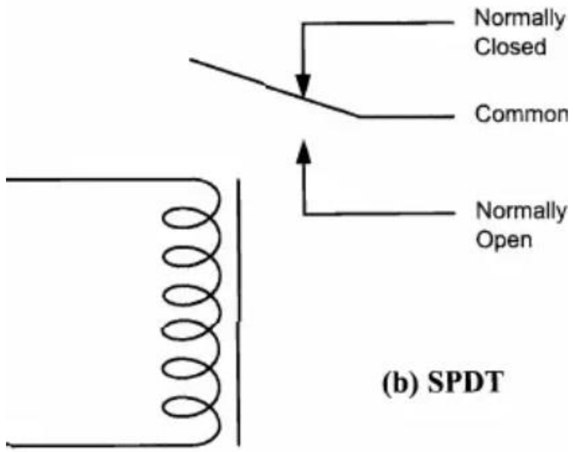


Figure 5.6.5 16Kx8 Memory (ROM) and 32Kx8 RAM Interfacing with 8051

How to Interface Relay with 8051 Advanced Development Board



(b) SPDT

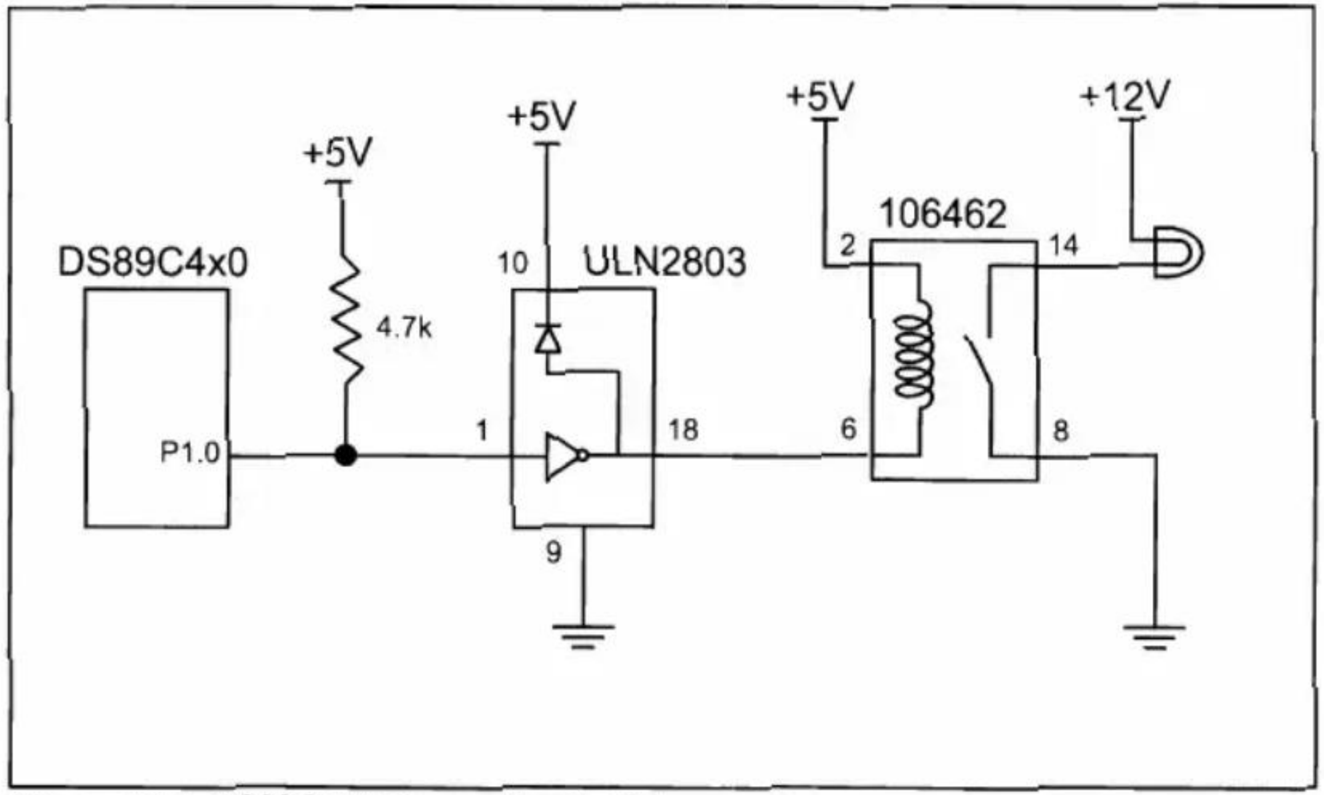


Figure 17-2. DS89C4x0 Connection to Relay

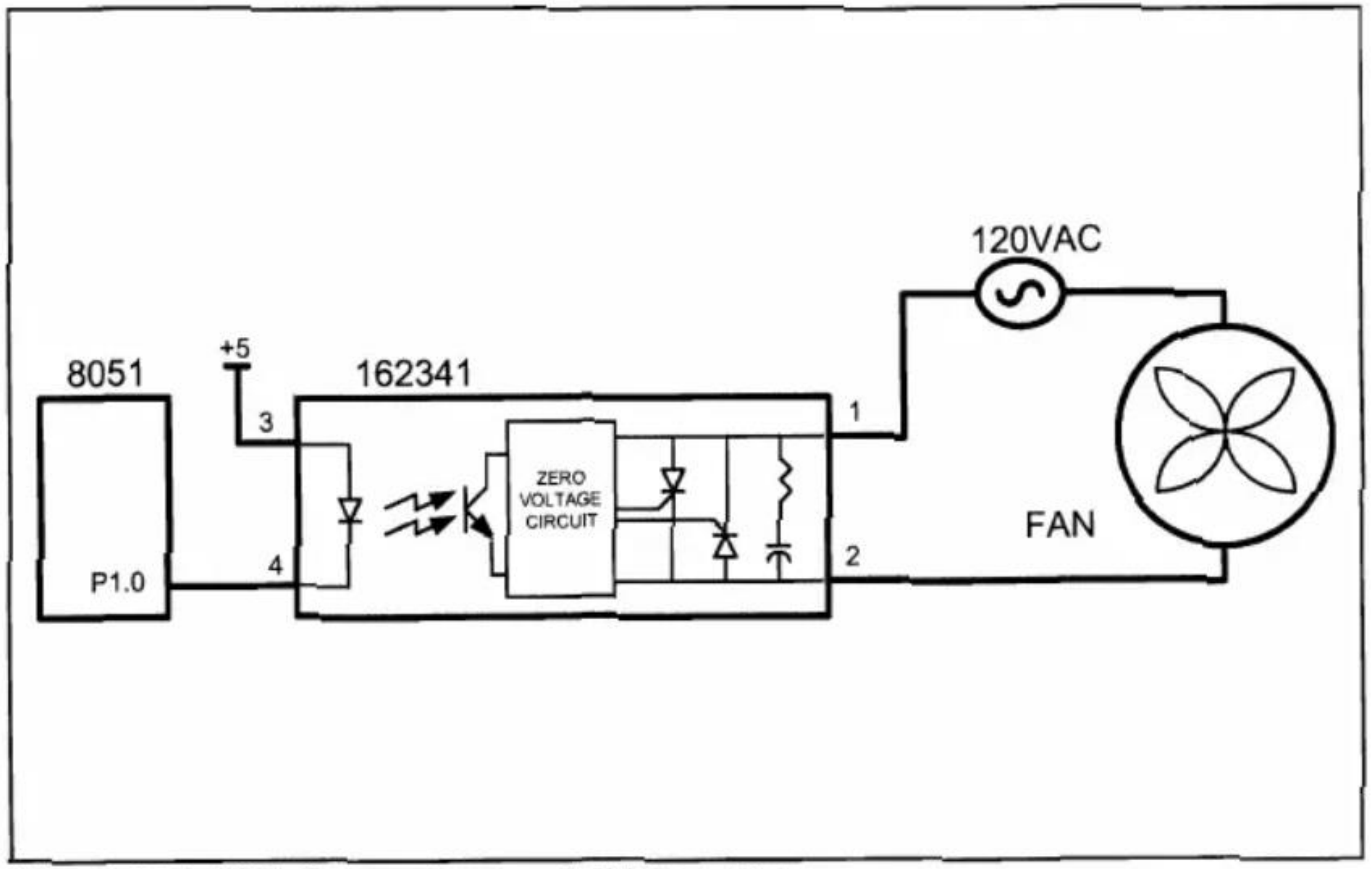


Figure 17-3. 8051 Connection to a Solid-State Relay

```
#include <reg51.h>
sbit relay1 = P2^0;
void DelayMs(unsigned int);
void main (void)
{
P2 = 0; //Initialize Port
while(1) //Loop Forever
{
		relay1 = 1;
		DelayMs(200);
		relay1 = 0;
		DelayMs(200);
		}
}
void DelayMs(unsigned int n)
{
unsigned int i,j;
		for(j=0;j<n;j++)
		{
				for(i=0;i<1000;i++);
		}
}
```

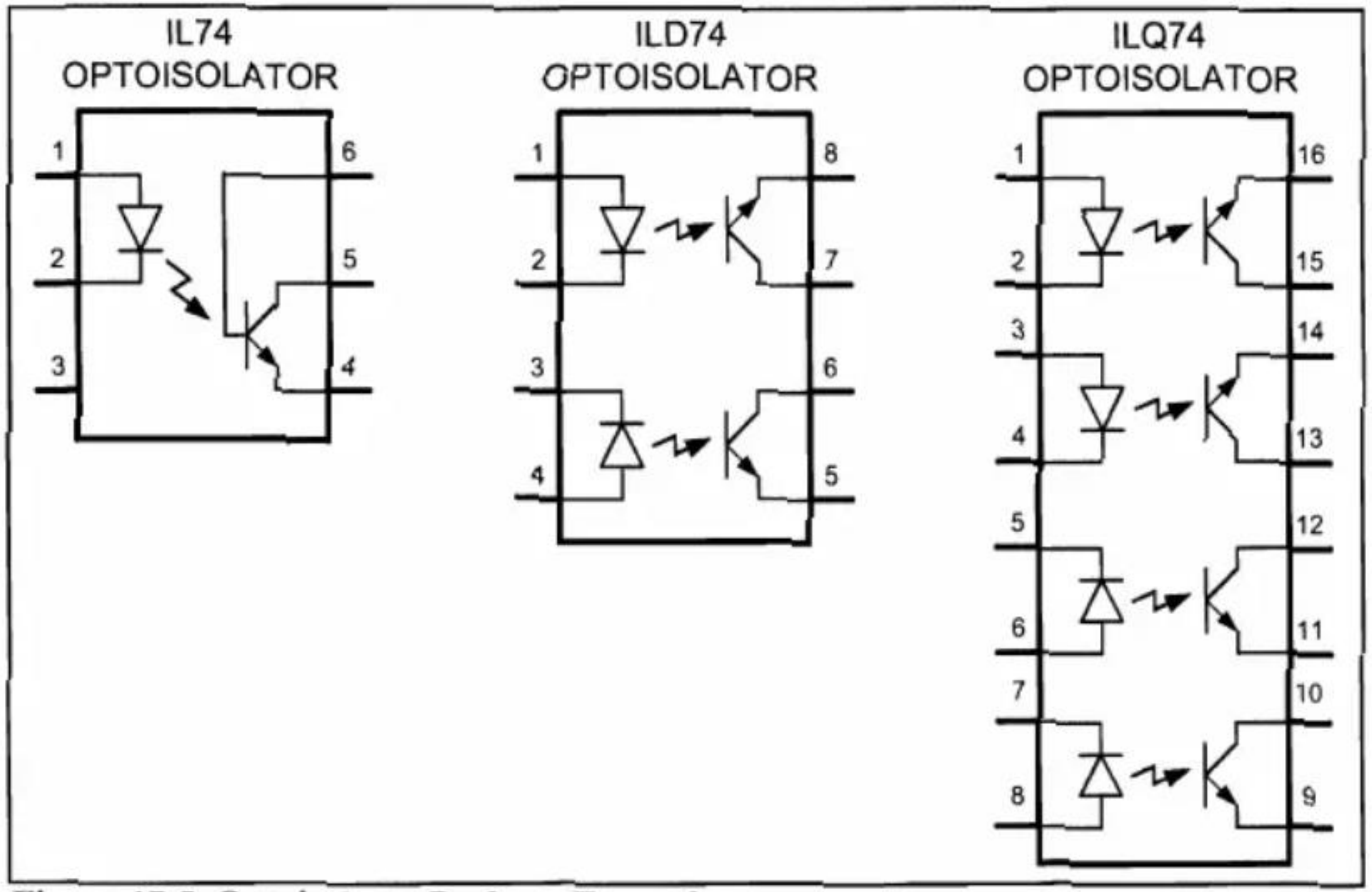


Figure 17-5. Optoisolator Package Examples

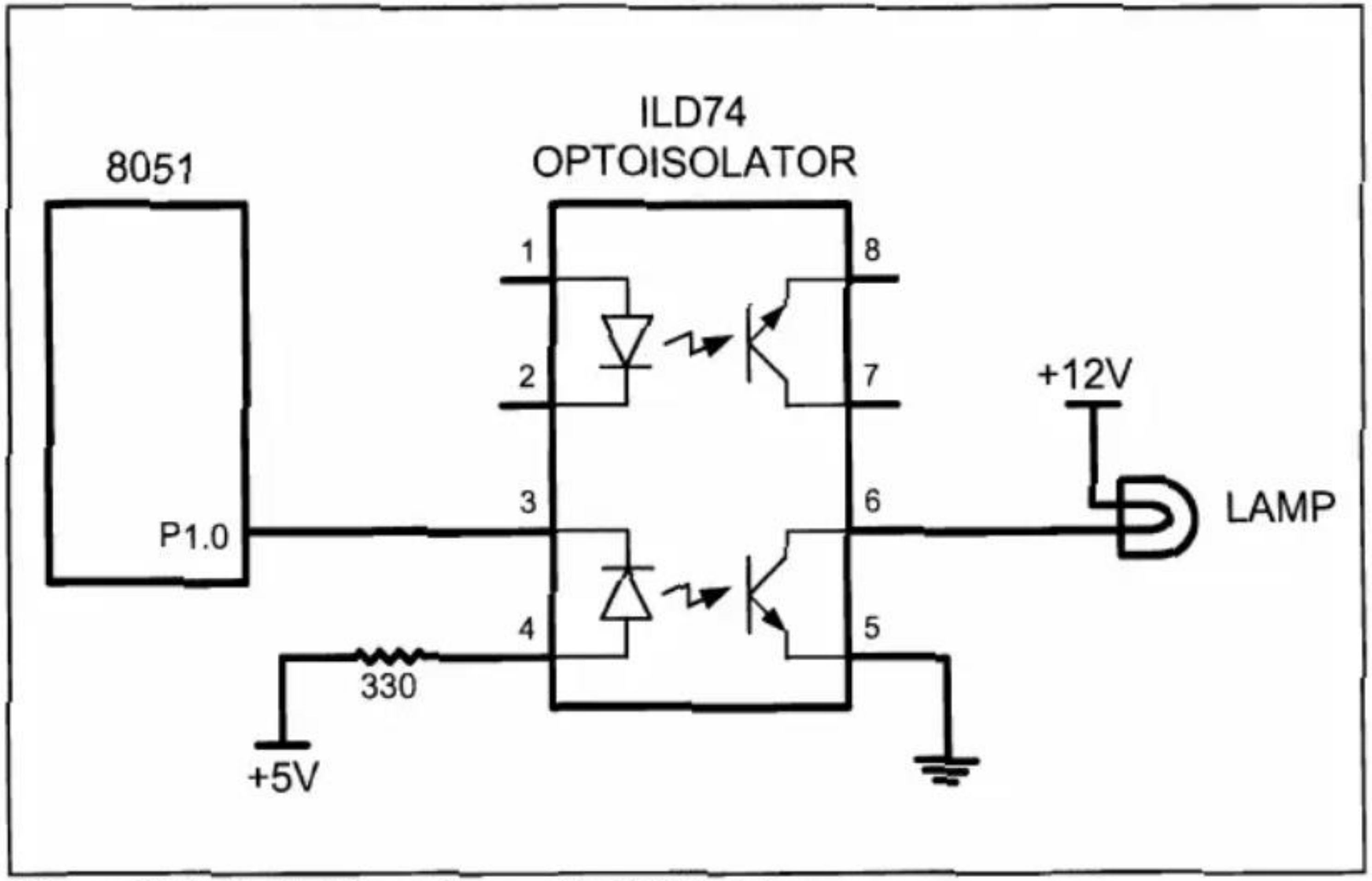


Figure 17-6. Controlling a Lamp via Optoisolator

Stepper Motor interfacing with 8051

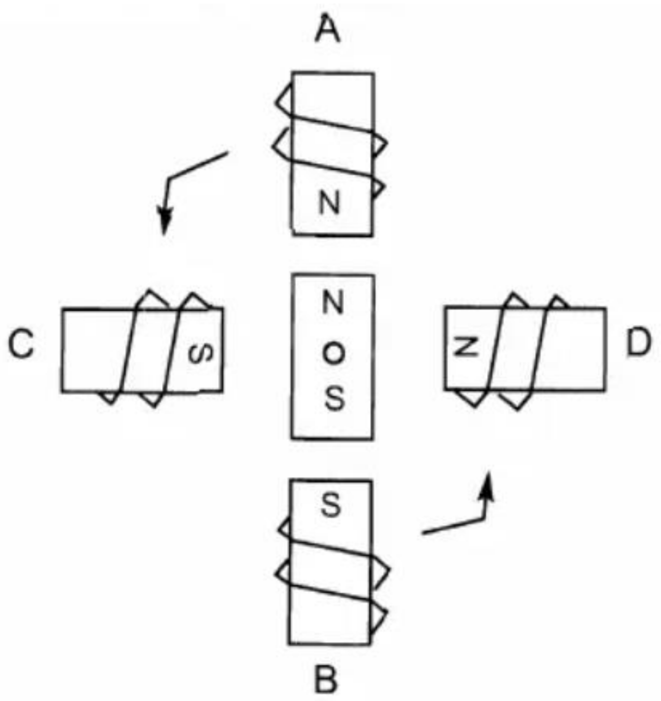


Table 17-4: Stepper Motor Step Angles

Step Angle	Steps per Revolution
0.72	500
1.8	200
2.0	180
2.5	144
5.0	72
7.5	48
15	24

Table 17-3: Normal 4-Step Sequence

Clockwise ↓

Step #	Winding A	Winding B	Winding C	Winding D
1	1	0	0	1
2	1	1	0	0
3	0	1	1	0
4	0	0	1	1

Counter-clockwise ↑

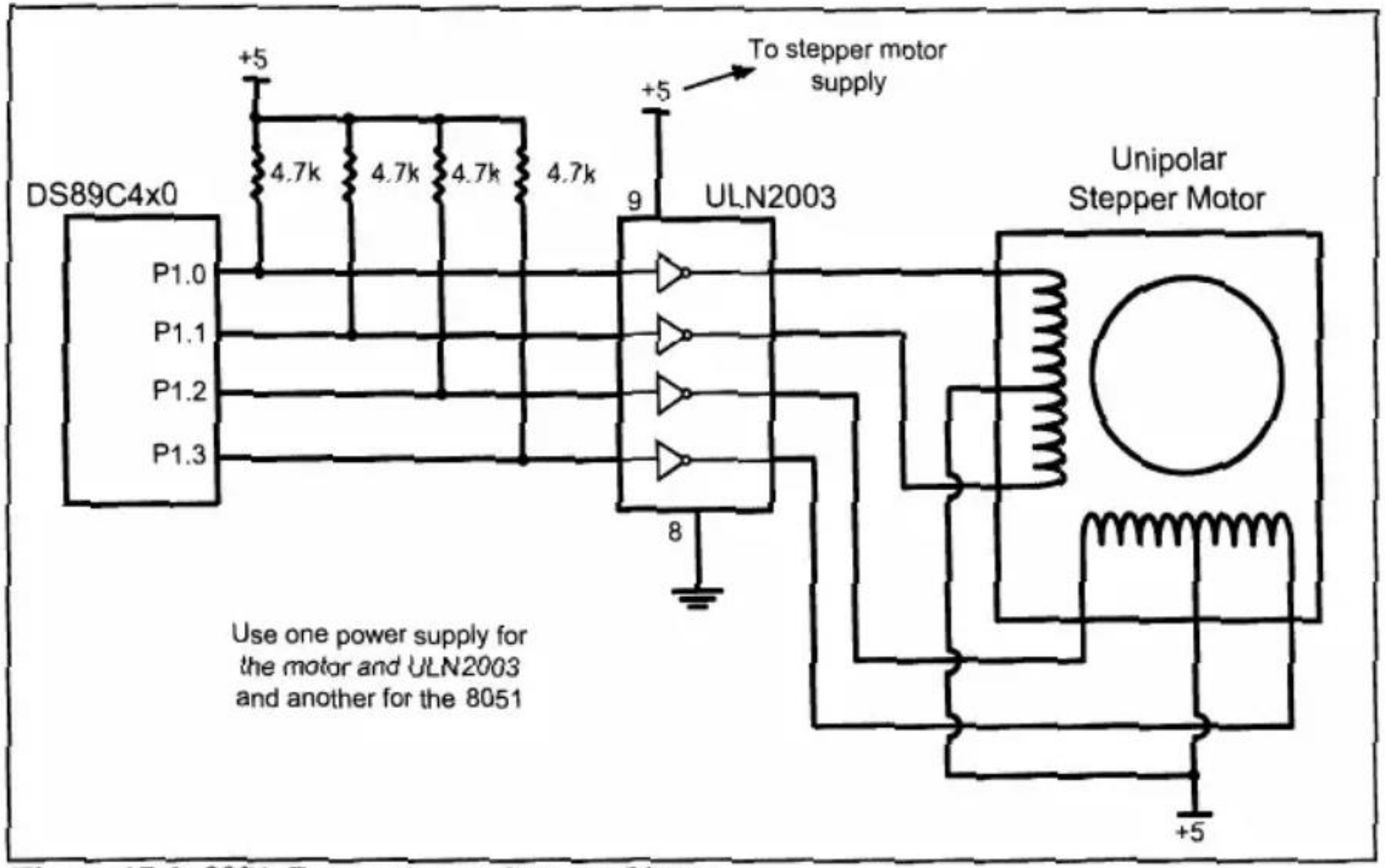
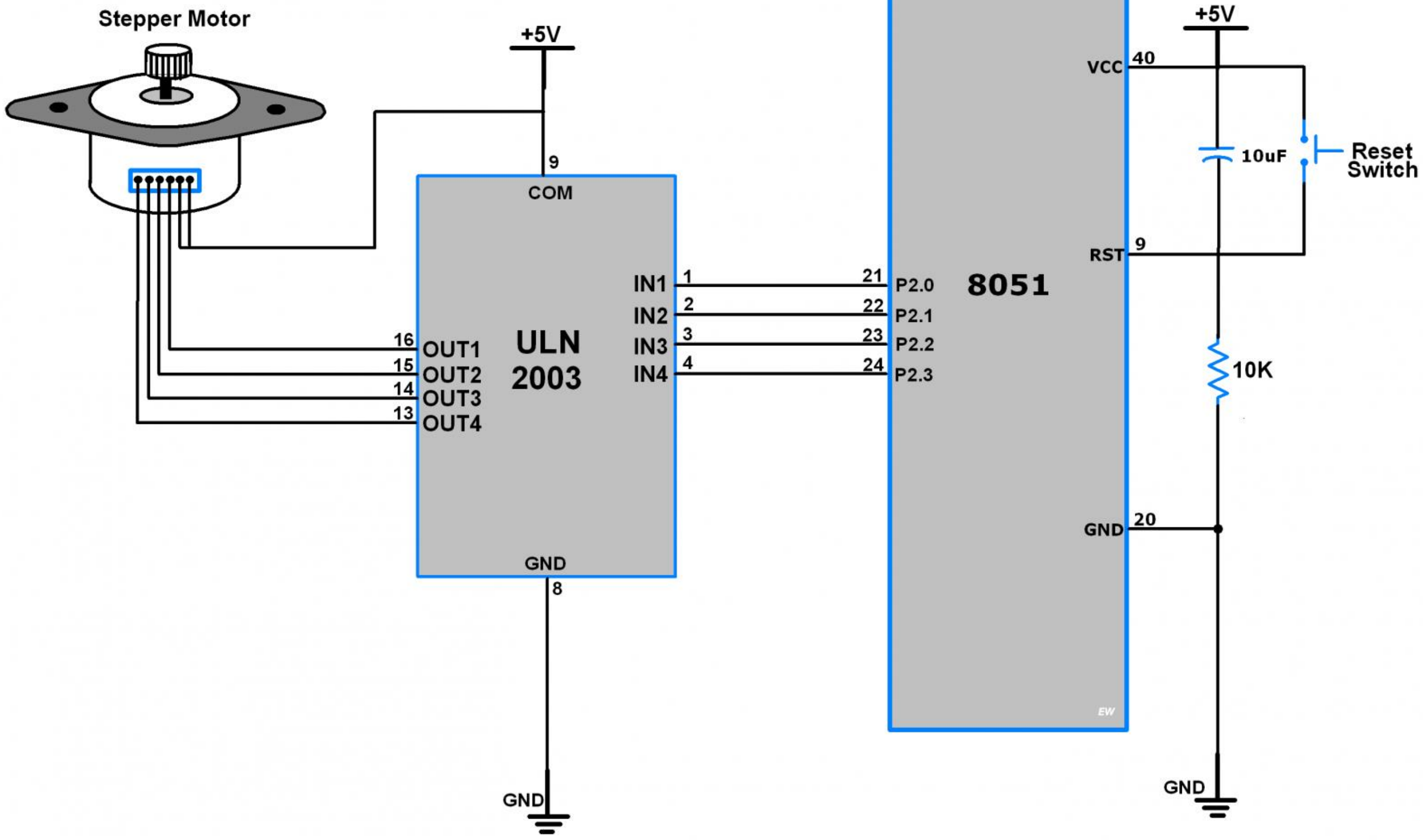


Figure 17-9. 8051 Connection to Stepper Motor

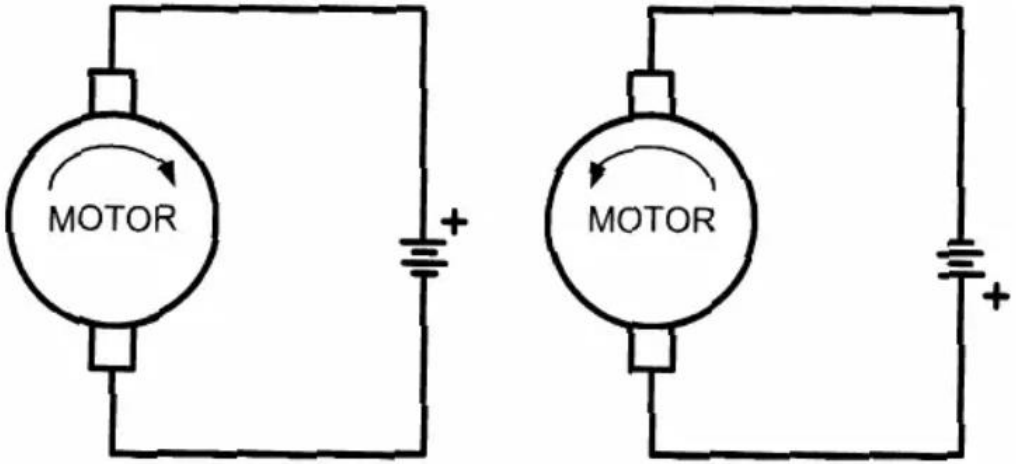


Stepper Motor
+5V
9
COM
ULN
2003
16 OUT1
15 OUT2
14 OUT3
13 OUT4
IN1 1
IN2 2
IN3 3
IN4 4
GND
8
GND
8051
21 P2.0
22 P2.1
23 P2.2
24 P2.3
VCC 40
RST 9
GND 20
+5V
10uF
Reset
Switch
10K
GND

```
        MOV  A,#66H     ;load step sequence
BACK:   MOV  P1,A       ;issue sequence to motor
        RR   A          ;rotate right clockwise
        ACALL DELAY     ;wait
        SJMP BACK       ;keep going

        ...
DELAY
        MOV  R2,#100
H1:     MOV  R3,#255
H2:     DJNZ R3,H2
        DJNZ R2,H1
        RET
```

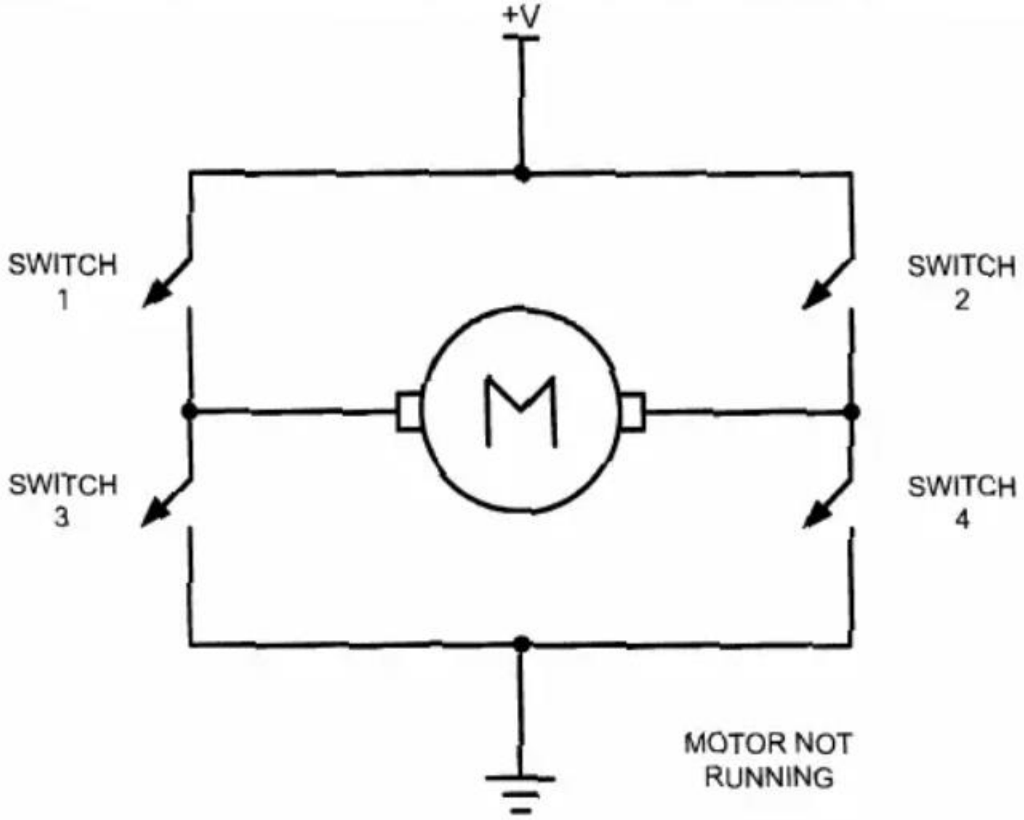
```
#include <reg51.h>
void main()
  {
    while(1)
      {
        P1 = 0x66;
        MSDelay(100);
        P1 = 0xCC;
        MSDelay(100);
        P1 = 0x99;
        MSDelay(100);
        P1 = 0x33;
        MSDelay(100);
      }
  }
```

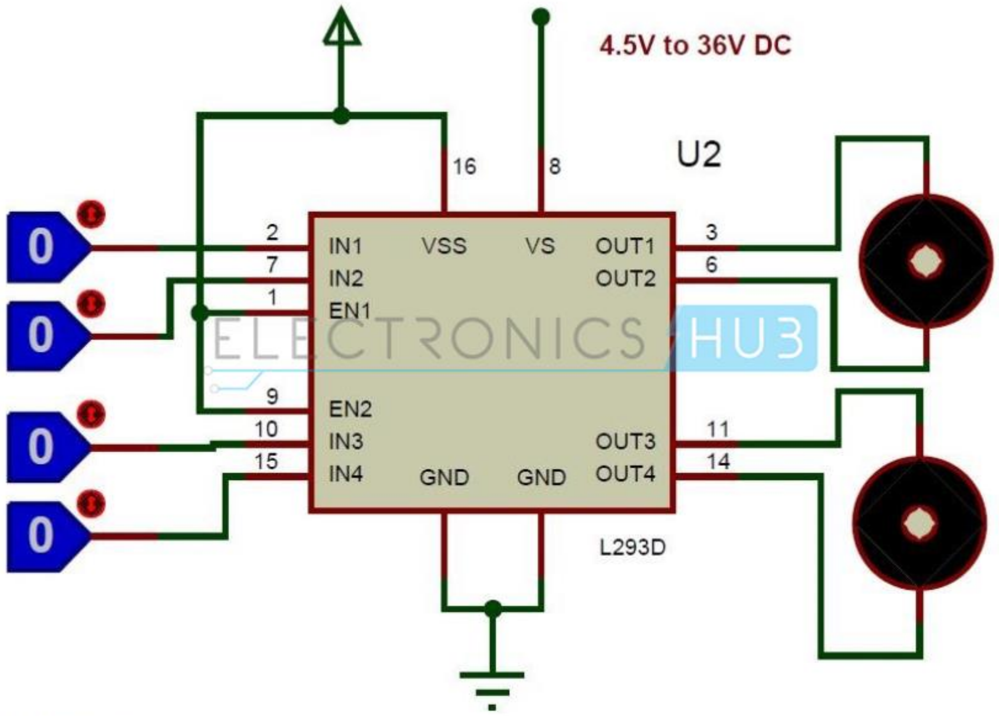
DC motor interfacing with 8051



Clockwise Rotation

Counter-Clockwise Rotation



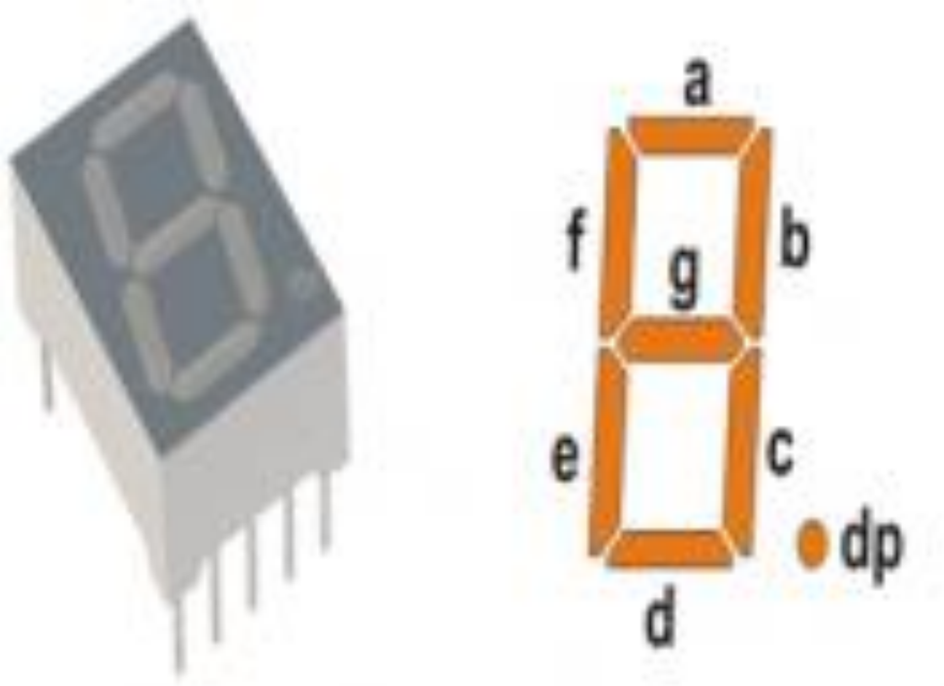


L293D Circuit

- IN1=0 and IN2=0 -> Motor1 idle
- IN1=0 and IN2=1 -> Motor1 Anti-clock wise direction
- IN1=1 and IN2=0 -> Motor1 Clock wise direction
- IN1=1 and IN2=1 -> Motor1 idle
- IN3=0 and IN4=0 -> Motor2 idle
- IN3=0 and IN4=1 -> Motor2 Anti-clock wise direction
- IN3=1 and IN4=0 -> Motor2 Clock wise direction
- IN3=1 and IN4=1 -> Motor2 idle

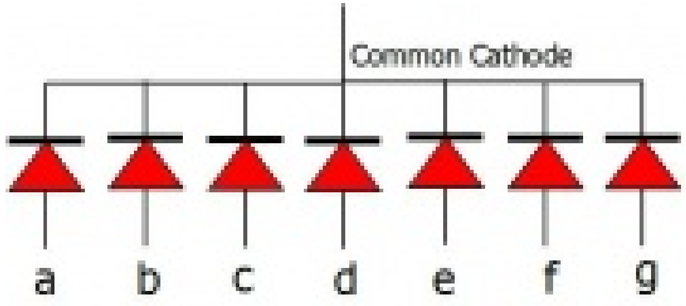
```c
#include<reg51.h>
sbit switch1=P2^0;
sbit switch2=P2^1;
sbit clk=P3^0;
sbit anticlk=P3^1;

void main()
{

        switch1=switch2=1;                              //making P2.0 and P2.1 as inputs
        switch1=switch2=0;
        clk=anticlk=0;
        while(1)
        {
                if((switch1))
                        clk=1;
                 else if((switch2))
                        anticlk=1;
                 else
                         P3=0x00;

        }
}
```

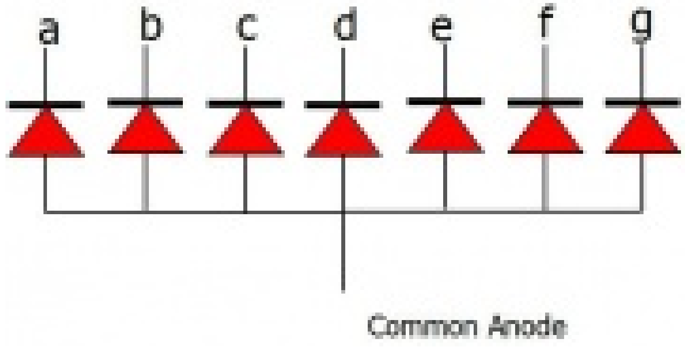
Interfacing Seven Segment with 8051



•**Common Cathode**: In this type of segments all the cathode terminals are made common and tied to GND. Thus the segments **a** to **g** needs a logic High signal(5v) in order to glow.

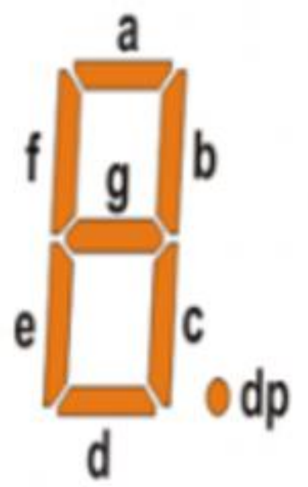
•**Common Anode**: In this type of segments all the anodes terminals are made common and tied to VCC(5v). Thus the segments **a** to **g** needs a logic LOW signal(GND) in order to glow.



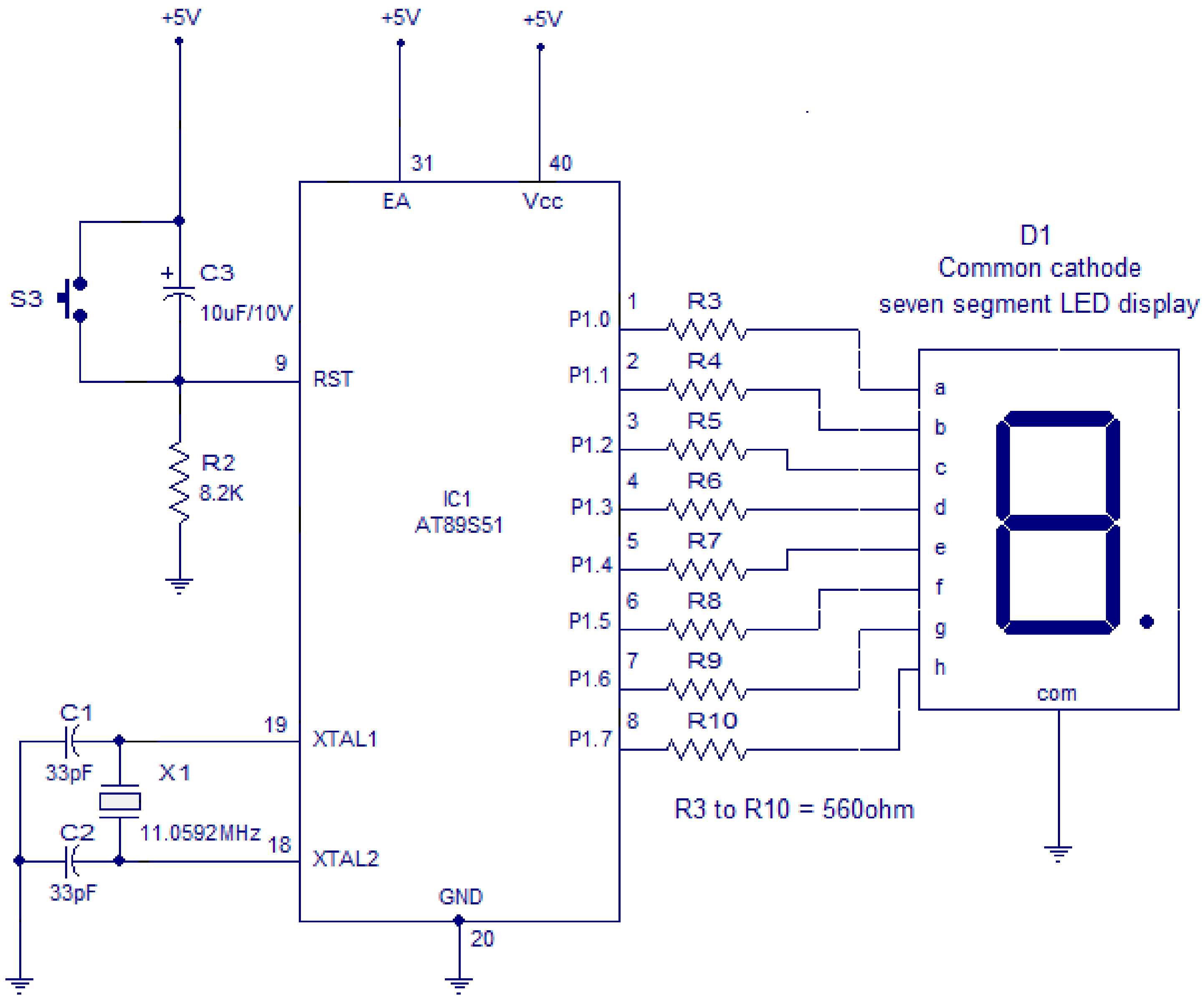


Common Anode seven segment display

Digit	h	g	f	e	d	c	b	a	Hex Value
0	1	1	0	0	0	0	0	0	0xC0
1	1	1	1	1	1	0	0	1	0xF9
2	1	0	1	0	0	1	0	0	0xA4
3	1	0	1	1	0	0	0	0	0xB0
4	1	0	0	1	1	0	0	1	0x99
5	1	0	0	1	0	0	1	0	0x92
6	1	0	0	0	0	0	1	0	0x82
7	1	1	1	1	1	0	0	0	0xF8
8	1	0	0	0	0	0	0	0	0x80
9	1	0	0	1	0	0	0	0	0x90



```c
#include <reg51.h>

void DELAY_ms(unsigned int ms_Count)
{
    unsigned int i,j;
    for(i=0;i<ms_Count;i++)
    {
        for(j=0;j<100;j++);
    }
}

int main() {
    char seg_code[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
    int i;

    while (1)
    {
        for (i = 0; i <= 9; i++) // loop to display 0-9
        {
            P2 = seg_code[i];
            DELAY_ms(1000);
        }
    }
}
```



Interfacing 7 segment display to 8051

Interfacing LCD Display with 8051

LCD INTERFACING:

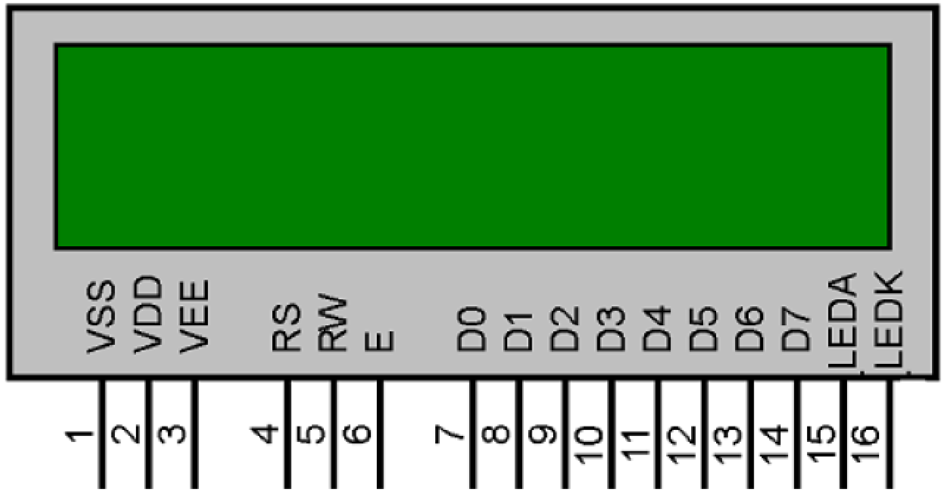


Figure 5.1 16X2 LCD Module

Pin No:	Name	Function
1	VSS	This pin must be connected to the ground
2	VCC	Positive supply voltage pin (5V DC)
3	VEE	Contrast adjustment
4	RS	Register selection
5	R/W	Read or write
6	E	Enable
7	DB0	Data
8	DB1	Data
9	DB2	Data
10	DB3	Data
11	DB4	Data
12	DB5	Data
13	DB6	Data
14	DB7	Data
15	LED+	Back light LED+
16	LED-	Back light LED

RS = 1 – data register, 0 – command register

R/W = 1 – Read, 0 - write

Table 12-2: LCD Command Codes

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

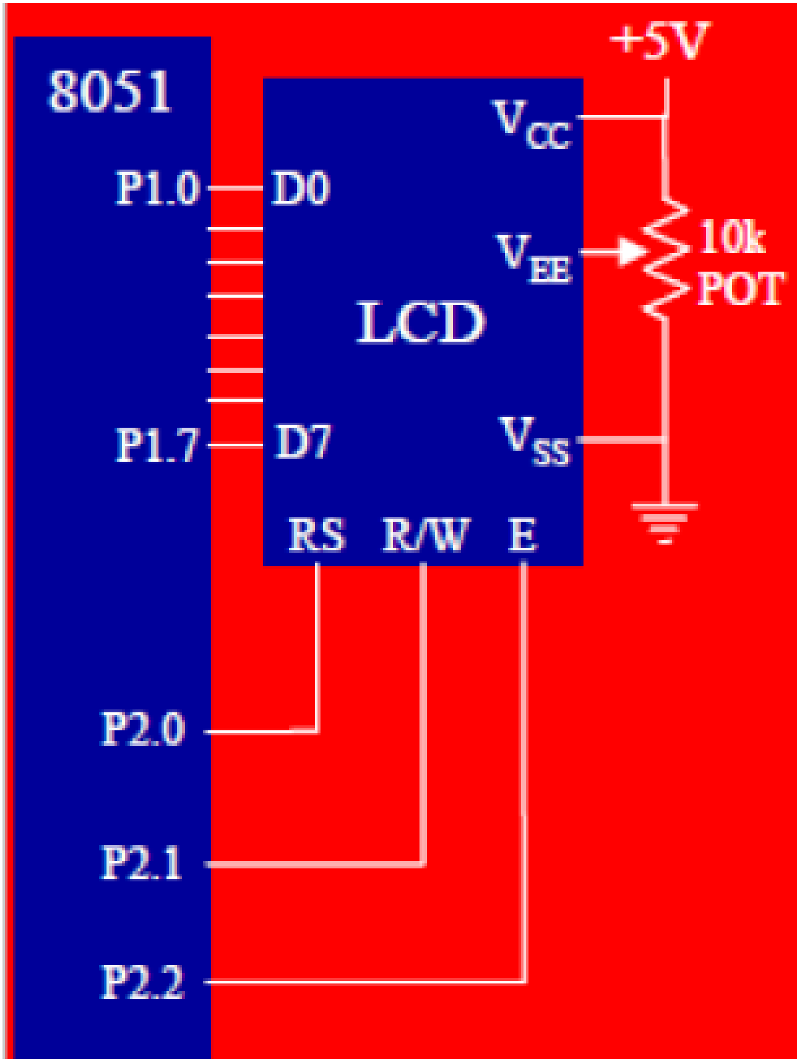


Figure 5.2 LCD Interfacing With 8051

Write an 8051 C program to send letters 'M', 'D', and 'E' to the LCD using delays.

Solution:

```
#include <reg51.h>
sfr ldata = 0x90;                //P1=LCD data pins (Fig. 12-2)
sbit rs = P2^0;
sbit rw = P2^1;
sbit en = P2^2;
void main()
  {
    lcdcmd(0x38);
    MSDelay(250);
    lcdcmd(0x0E);
    MSDelay(250);
    lcdcmd(0x01);
    MSDelay(250);
    lcdcmd(0x06);
    MSDelay(250);
    lcdcmd(0x86);                //line 1, position 6
    MSDelay(250);
    lcddata('M');
    MSDelay(250);
    lcddata('D');
    MSDelay(250);
    lcddata('E');
  }
```

Table 12-2: LCD Command Codes

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

```
void lcdcmd(unsigned char value)
  {
    ldata = value;                 // put the value on the pins
    rs = 0;
    rw = 0;
    en = 1;                        // strobe the enable pin
    MSDelay(1);
    en = 0;
    return;
  }

void lcddata(unsigned char value)
  {
    ldata = value;                 // put the value on the pins
    rs = 1;
    rw = 0;
    en = 1;                        // strobe the enable pin
    MSDelay(1);
    en = 0;
    return;
  }

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