Module No: 05

Interfacing 8051 to LCD

LCD is finding widespread use replacing LEDs for the following reasons:

- The declining prices of LCD
- The ability to display numbers, characters, and graphics
- Incorporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD
- Ease of programming for characters and graphics

Interfacing LCD with 8051

The LCD requires 3 control lines (RS, R/W & EN) & 8 (or 4) data lines. The number on data lines depends on the mode of operation. If operated in 8-bit mode then 8 data lines + 3 control lines i.e. total 11 lines are required. And if operated in 4-bit mode then 4 data lines + 3 control lines i.e. 7 lines are required.

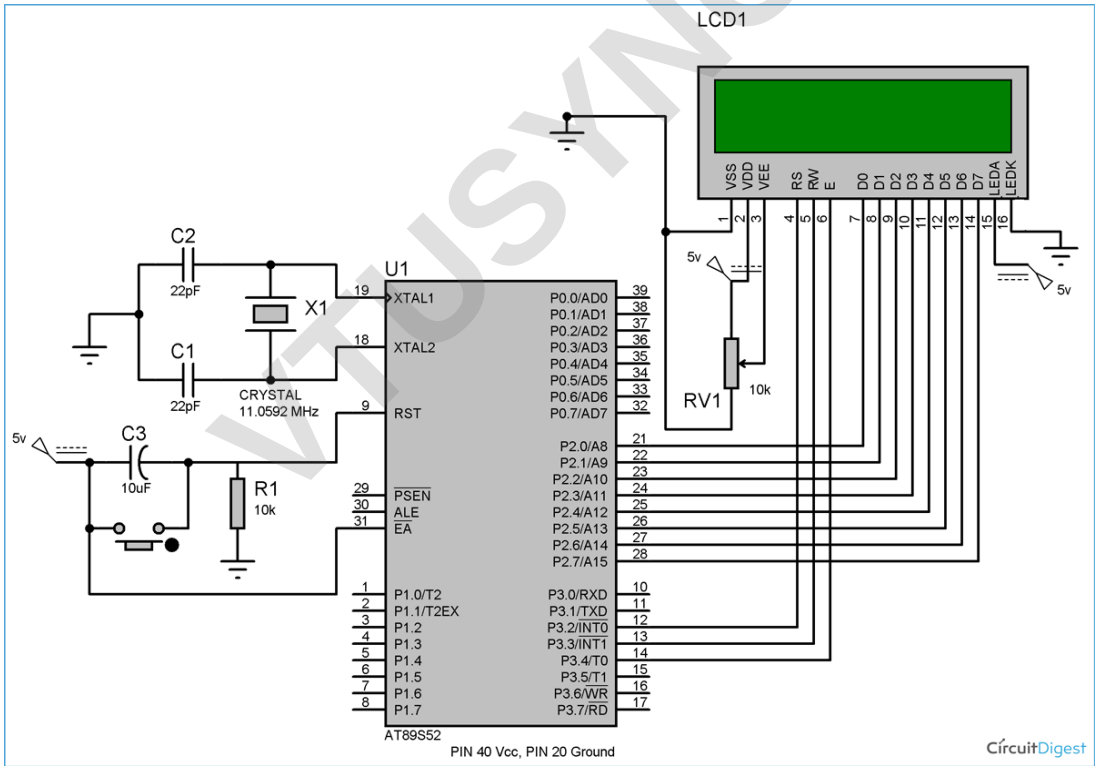


Figure: Interfacing 8051 with LCD

Pin Description:

Pin	Symbol	I/O	Description
1	V_{SS}	--	Ground
2	V_{CC}	--	+5V power supply
3	V_{EE}	--	Power supply to control contrast
4	RS	I	RS=0 to select command register, RS=1 to select data register
5	R/W	I	R/W=0 for write, R/W=1 for read
6	E	I/O	Enable
7	DB0	I/O	The 8-bit data bus
8	DB1	I/O	The 8-bit data bus
9	DB2	I/O	The 8-bit data bus
10	DB3	I/O	The 8-bit data bus
11	DB4	I/O	The 8-bit data bus
12	DB5	I/O	The 8-bit data bus
13	DB6	I/O	The 8-bit data bus
14	DB7	I/O	The 8-bit data bus

- Send displayed information or instruction command codes to the LCD
- Read the contents of the LCD's internal registers

LCD timing diagram for reading and writing is as shown in figure

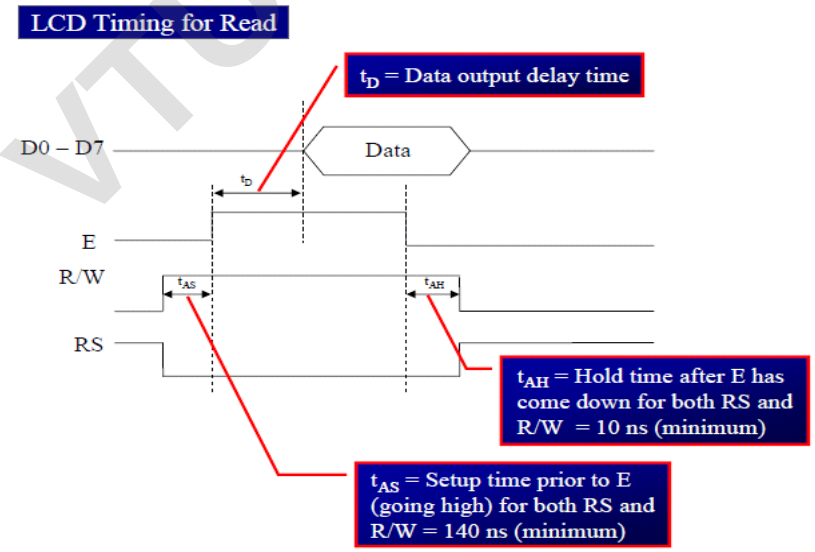


Figure : LCD timing for read

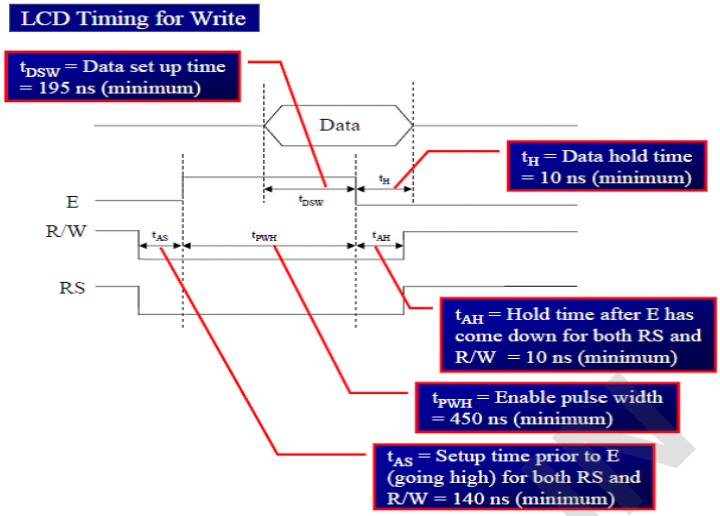


Figure : LCD timing for write

Sending Data/ Commands to LCDs with Time Delay:

To send any of the commands to the LCD, make pin RS=0. For data, make RS=1. Then send a high-to-low pulse to the E pin to enable the internal latch of the LCD. This is shown in the code below. The interfacing diagram of LCD to 8051 is as shown in the figure .

Example 11: Write an ALP to initialize the LCD and display message "YES". Say the command to be given is :38H (2 lines ,5x7 matrix), 0EH (LCD on, cursor on), 01H (clear LCD), 06H (shift cursor right), 86H (cursor: line 1, pos. 6)
Program: Calls a time delay before sending next data/command ;P1.0-P1.7 are connected to LCD data pins D0-D7 ;P2.0 is connected to RS pin of LCD ;P2.1 is connected to R/W pin of LCD ;P2.2 is connected to E pin of LCD

```
        ORG 0H
        MOV A,#38H          ;INIT. LCD 2 LINES, 5X7 MATRIX
        ACALL COMNWRT       ;call command subroutine
        ACALL DELAY         ;give LCD some time
        MOV A,#0EH          ;display on, cursor on
        ACALL COMNWRT       ;call command subroutine
        ACALL DELAY          ;give LCD some time
        MOV A,#01           ;clear LCD
        ACALL COMNWRT       ;call command subroutine
        ACALL DELAY         ;give LCD some time
        MOV A,#06H          ;shift cursor right
        ACALL COMNWRT        ;call command subroutine
        ACALL DELAY         ;give LCD some time
        MOV A,#86H          ;cursor at line 1, pos. 6
        ACALL COMNWRT       ;call command subroutine
        ACALL DELAY         ;give LCD some time
        MOV A,#'Y'          ;display letter Y
```

```
        ACALL DATAWRT       ;call display subroutine
        ACALL DELAY         ;give LCD some time
        MOV A,#'E'          ;display letter E
        ACALL DATAWRT       ;call display subroutine
        ACALL DELAY         ;give LCD some time
        MOV A,#'S'          ;display letter S
        ACALL DATAWRT       ;call display subroutine
AGAIN: SJMP AGAIN           ;stay here
COMNWRT:                    ;send command to LCD
        MOV P1,A            ;copy reg A to port 1
        CLR P2.0            ;RS=0 for command
        CLR P2.1            ;R/W=0 for write
        SETB P2.2           ;E=1 for high pulse
        ACALL DELAY         ;give LCD some time
        CLR P2.2            ;E=0 for H-to-L pulse
RET
DATAWRT:                    ;write data to LCD
        MOV P1,A            ;copy reg A to port 1
        SETB P2.0           ;RS=1 for data
        CLR P2.1            ;R/W=0 for write
        SETB P2.2           ;E=1 for high pulse
        ACALL DELAY         ;give LCD some time
        CLR P2.2            ;E=0 for H-to-L pulse
RET

DELAY:
        MOV R3,#50          ;50 or higher for fast CPUs
HERE2: MOV R4,#255          ;R4 = 255
HERE: DJNZ R4,HERE          ;stay until R4 becomes 0
        DJNZ R3,HERE2
        RET

        END
```

Example 12: Modify example 11, to check for the busy flag (D7=>P1.7), then send the command and hence display message "NO".

```
;Check busy flag before sending data, command to LCD;p1=data pin ;P2.0 connected to RS pin
;P2.1 connected to R/W pin ;P2.2 connected to E pin
ORG 0H
        MOV A,#38H          ;init. LCD 2 lines ,5x7 matrix
        ACALL COMMAND       ;issue command
        MOV A,#0EH          ;LCD on, cursor on
        ACALL COMMAND       ;issue command
        MOV A,#01H          ;clear LCD command
        ACALL COMMAND       ;issue command
        MOV A,#06H          ;shift cursor right
```

```
        ACALL COMMAND          issue command
        MOV A,#86H             ;cursor: line 1, pos. 6
        ACALL COMMAND          ;command subroutine
        MOV A,#'N'             ;display letter N
        ACALL DATA_DISPLAY
        MOV A,#'O'             ;display letter O
        ACALL DATA_DISPLAY
HERE:SJMP HERE          ;STAY HERE
COMMAND:
        ACALL READY            ;is LCD ready?
        MOV P1,A               ;issue command code
        CLR P2.0               ;RS=0 for command
        CLR P2.1               ;R/W=0 to write to LCD
        SETB P2.2              ;E=1 for H-to-L pulse
        CLR P2.2               ;E=0,latch in
RET
DATA_DISPLAY:
        ACALL READY            ;is LCD ready?
        MOV P1,A               ;issue data
        SETB P2.0              ;RS=1 for data
        CLR P2.1               ;R/W =0 to write to LCD
        SETB P2.2              ;E=1 for H-to-L pulse
        CLR P2.2               ;E=0,latch in
RET
READY:
        SETB P1.7              ;make P1.7 input port
        CLR P2.0               ;RS=0 access command reg
        SETB P2.1              ;R/W=1 read command reg ;
 BACK:SETB P2.2                ;E=1 for H-to-L pulse
        CLR P2.2               ;E=0 H-to-L pulse
        JB P1.7,BACK           ;stay until busy flag=0
        RET

        END
```

Sending Data/ Commands to LCDs checking the Busy Flag

Example 13: Write an 8051 C program to send letters 'P', 'I', and 'C' to the LCD using the busy flag method.

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);
        lcdcmd(0x0E);
        lcdcmd(0x01);
        lcdcmd(0x06);
        lcdcmd(0x86);                   //line 1, position 6
        lcddata('P');
        lcddata('I');
        lcddata('C');
}
void lcdcmd(unsigned char value){
        lcdready();                     //check the LCD busy flag
        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){
        lcdready();                     //check the LCD busy flag
        ldata = value;                  //put the value on the pins
        rs = 1;
        rw = 0;
        en = 1;                         //strobe the enable pin
        MSDelay(1);
        en = 0;
        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++);}
```

Keyboard Interfacing:

Keyboards are organized in a matrix of rows and columns. The CPU accesses both rows and columns through ports. Therefore, with two 8-bit ports, an 8 x 8 matrix of keys can be connected to a microprocessor. When a key is pressed, a row and a column make a contact. Otherwise, there is no connection between rows and columns. A 4x4 matrix connected to two ports. The rows are connected to an output port and the columns are connected to an input port.

Scanning and Identifying the Key:

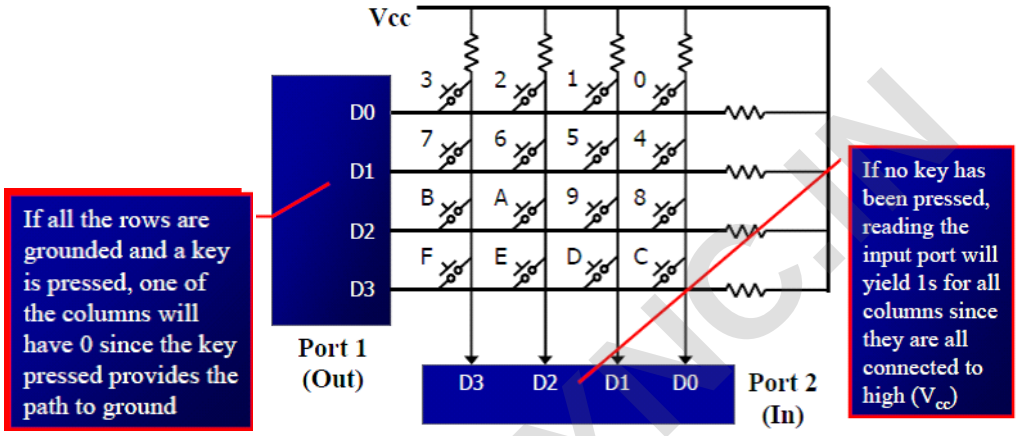


Figure : A 4X4 matrix keyboard

It is the function of the microcontroller to scan the keyboard continuously to detect and identify the key pressed

- To detect a pressed key, the microcontroller grounds all rows by providing 0 to the output latch, then it reads the columns
- If the data read from columns is D3 – D0 =1111, no key has been pressed and the process continues till key press is detected
- If one of the column bits has a zero, this means that a key press has occurred For example, if D3 – D0 = 1101, this means that a key in the D1 column has been pressed After detecting a key press, microcontroller will go through the process of identifying the key
- Starting with the top row, the microcontroller grounds it by providing a low to row D0 only. It reads the columns, 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.

Algorithm for detection and identification of key activation goes through the following stages:

1. To make sure that the preceding key has been released, 0s are output to all rows at once, and the columns are read and checked repeatedly until all the columns are high

- When all columns are found to be high, the program waits for a short amount of time before it goes to the next stage of waiting for a key to be pressed

2. To see if any key is pressed, the columns are scanned over and over in an infinite loop until one of them has a 0 on it

- Remember that the output latches connected to rows still have their initial zeros (provided in stage 1), making them grounded
- After the key press detection, it waits 20 ms for the bounce and then scans the columns again
  (a) It ensures that the first key press detection was not an erroneous one due a spike noise
  (b) The key press. If after the 20-ms delay the key is still pressed, it goes back into the loop to detect a real key press

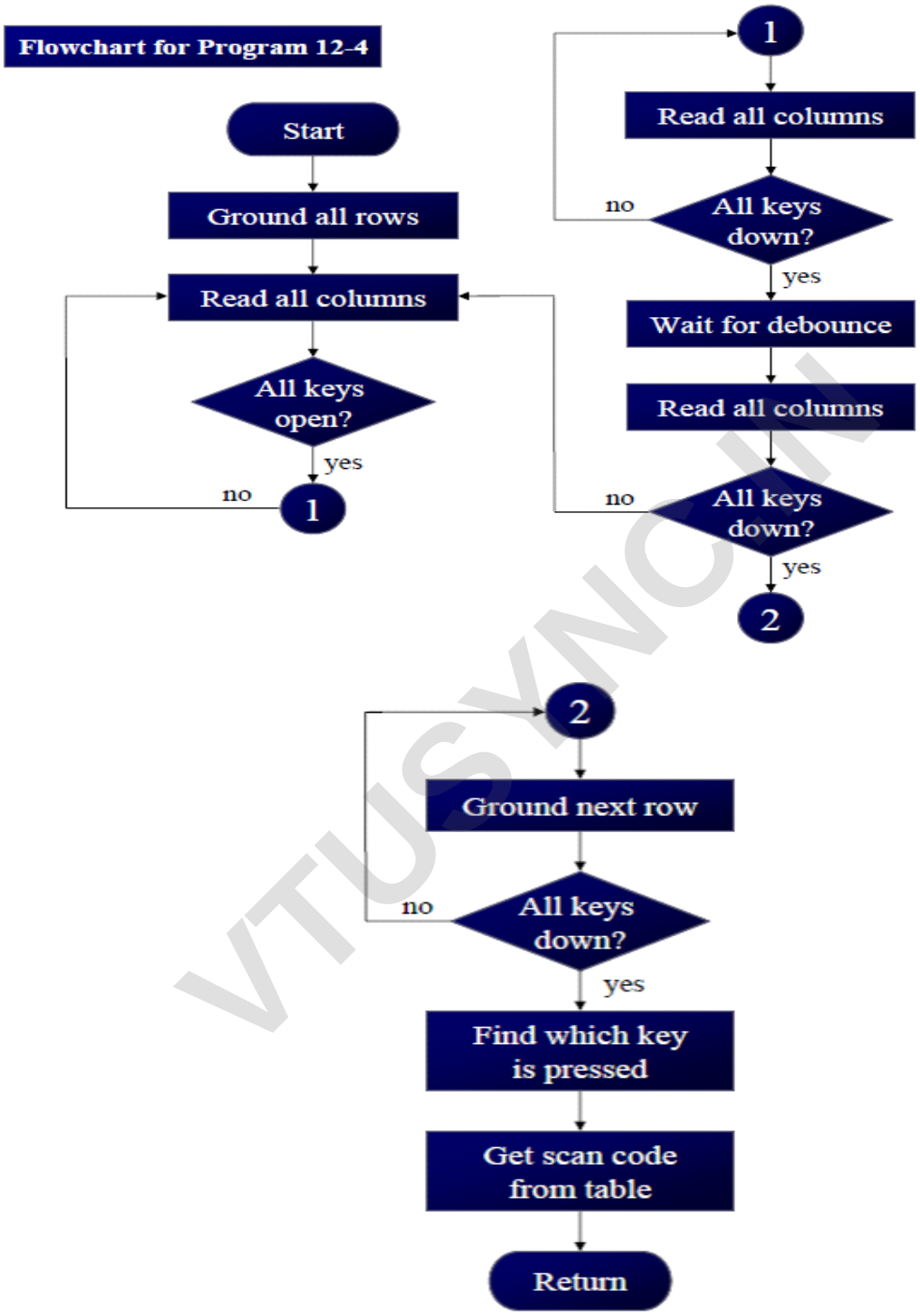
3. To detect which row key press belongs to, it grounds one row at a time, reading the columns each time

- If it finds that all columns are high, this means that the key press cannot belong to that row. Therefore, it grounds the next row and continues until it finds the row the key press belongs to
- Upon finding the row that the key press belongs to, it sets up the starting address for the look-up table holding the scan codes (or ASCII) for that row

4. To identify the key press, it rotates the column bits, one bit at a time, into the carry flag and checks to see if it is low

- Upon finding the zero, it pulls out the ASCII code for that key from the look-up table
- otherwise, it increments the pointer to point to the next element of the look-up table

The flowchart for the above algorithm is as shown below:



Note: The assembly as well as the C program can be written in accordance to the algorithm of the flowchart shown.

Interfacing 8051 to parallel and Serial ADC

Analog-to-digital converter (ADC) interfacing

ADCs (analog-to-digital converters) are among the most widely used devices for data acquisition. A physical quantity, like temperature, pressure, humidity, and velocity, etc., is converted to electrical (voltage, current) signals using a device called a transducer, or sensor
We need an analog-to-digital converter to translate the analog signals to digital numbers, so microcontroller can read them.

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 110μs.

Pin Description of ADC804:

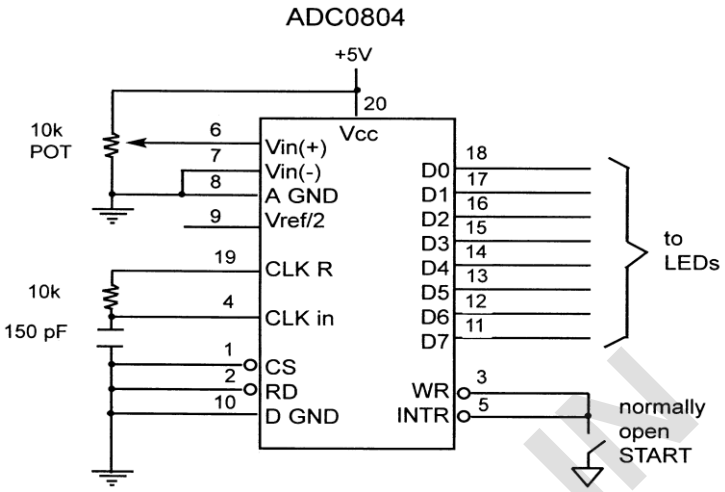


Figure: Pin out of ADC0804

- **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:

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

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

- **Vref/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, Vref/2 is connected to 2 volts. Step size is the smallest change can be discerned by an ADC Vref/2 Relation to Vin 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

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

Analog ground and digital ground: Analog ground is connected to the ground of the analog Vin and digital ground is connected to the ground of the Vcc pin. To isolate the analog Vin signal from transient voltages caused by digital switching of the output D0 – D7. This contributes to the accuracy of the digital data output.

Vin(+) & Vin(-): Differential analog inputs where Vin = Vin (+) – Vin (-). Vin (-) is connected to ground and Vin (+) is used as the analog input to be converted.

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

INTR: It is "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.

WR: It is "start conversion" When WR makes a low-to-high transition, ADC804 starts converting the analog input value of Vin to an 8- bit digital number.

CS: It is an active low input used to activate ADC804.

The following steps must be followed for data conversion by the ADC804 chip:

- Make CS= 0 and send a L-to-H pulse to pin WR to start conversion.
- Monitor the INTR pin, if high keep polling but if low, conversion is complete, go to next step.
- Make CS= 0 and send a H-to-L pulse to pin RD to get the data out

Figure shows the read and write timing for ADC804. Figure 9 and 10 shows the self clocking with the RC component for frequency and the external frequency connected to XTAL2 of 8051.

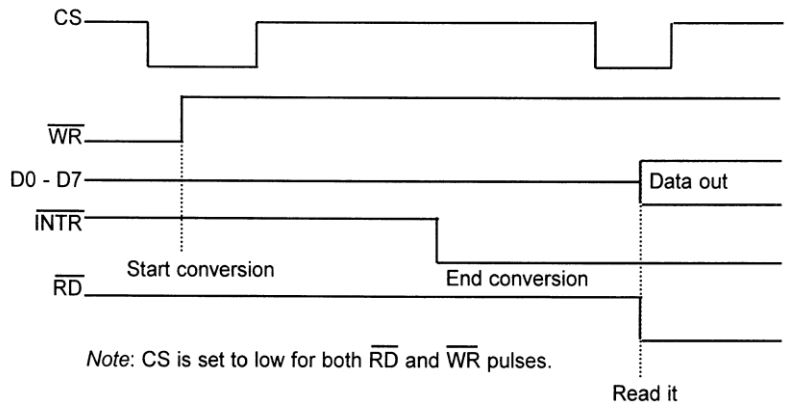


Figure : Read and Write timing for ADC0804

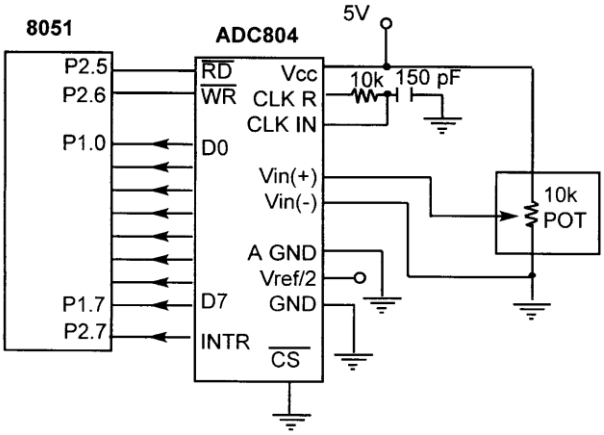


Figure : 8051 Connection to ADC0804 with Self-clocking

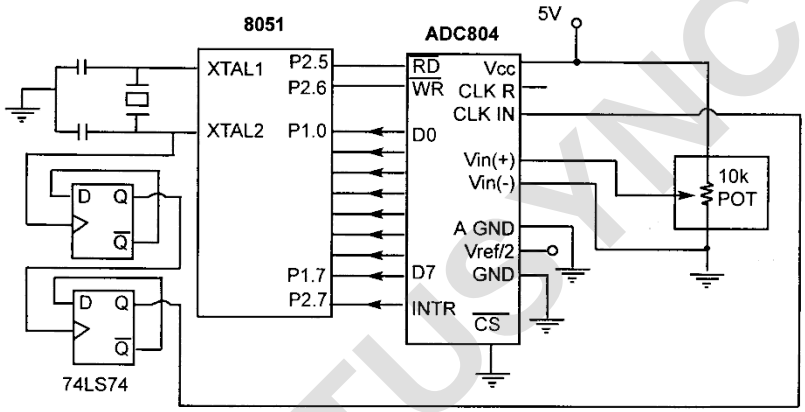


Figure : 8051 Connection to ADC0804 with Clock from XTAL2 of 8051

Now let us see how we write assembly as well as C program for the interfacing diagram shown in figure .

Programming ADC0804 in assembly

```
        MYDATA EQU P1
        MOV P1, #0FFH
        SETB P2.7
BACK:   CLR P2.6
        SETB P2.6
HERE:   JB P2.7, HERE
        CLR P2.5
        MOV A, MYDATA
        SETB P2.5
        SJMP BACK
```

Programming ADC0804 in 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;
        INTR=1;
        RD=1;
        WR=1;
        While (1)
        {
          WR=0;
           WR=1;
           While (INTR == 1);
          RD=0;
          Value =Mydata;
          RD=1;
        }
}
```

ADC0808/0809 chip:

ADC808 has 8 analog inputs. It allows us to monitor up to 8 different transducers using only single chip. The chip has 8-bit data output just like the ADC804. The 8 analog input channels are multiplexed and selected according to the values given to the three address pins, A, B, and C. that is; if CBA=000, CH0 is selected; CBA=011, CH3 is selected and so on. The pin details of ADC0808 are as shown in the figure below. (Explanation can be done as is with ADC0804).

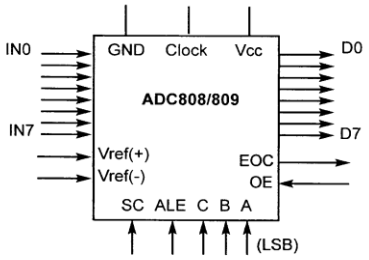


Figure : Pin out of ADC0808

Steps to Program ADC0808/0809

1. Select an analog channel by providing bits to A, B, and C addresses.
2. Activate the ALE pin. It needs an L-to-H pulse to latch in the address.
3. Activate SC (start conversion) by an H-to-L pulse to initiate conversion.
4. Monitor EOC (end of conversion) to see whether conversion is finished.

5. Activate OE (output enable) to read data out of the ADC chip. An H-to-L pulse to the OE pin will bring digital data out of the chip.

Let us write an assembly and C program for the interfacing of 8051 to ADC0808

Programming ADC0808/0809 in assembly

```
        MYDATA EQU P1
        ORG 0000H
        MOV MYDATA, #0FFH
        SETB P2.7
        CLR P2.4
        CLR P2.6
        CLR P2.5
BACK:   CLR P2.0
        CLR P2.1
        SETB P2.2
        ACALL DELAY
        SETB P2.4
        ACALL DELAY
        SETB P2.6
        ACALL DELAY
        CLR P2.4
        CLR P2.6
HERE:   JB P2.7, HERE
HERE1:  JNB P2.7, HERE1
        SETB P2.5
        ACALL DELAY
        MOV A, MYDATA
        CLR P2.5
        SJMP BACK
```

Interfacing 8051 to DAC

Digital-to-Analog (DAC) converter:

The DAC is a device widely used to convert digital pulses to analog signals. In this section we will discuss the basics of interfacing a DAC to 8051.

The two method of creating a DAC is binary weighted and R/2R ladder.
The Binary Weighted DAC, which contains one resistor or current source for each bit of the DAC connected to a summing point. These precise voltages or currents sum to the correct output value. This is one of the fastest conversion methods but suffers from poor accuracy because of the high precision required for each individual voltage or current. Such high-precision resistors and current-sources are expensive, so this type of converter is usually limited to 8-bit resolution or less.

The R-2R ladder DAC, which is a binary weighted DAC that uses a repeating cascaded structure of resistor values R and 2R. This improves the precision due to the relative ease of producing

equal valued matched resistors (or current sources). However, wide converters perform slowly due to increasingly large RC-constants for each added R-2R link.

The first criterion for judging a DAC is its resolution, which is a function of the number of binary inputs. The common ones are 8, 10, and 12 bits. The number of data bit inputs decides the resolution of the DAC since the number of analog output levels is equal to 2^n, where n is the number of data bit inputs.

DAC0808:

The digital inputs are converter to current $I_{out,}$ and by connecting a resistor to the I_{out} pin, we can convert the result to voltage. The total current I_{out} is a function of the binary numbers at the D0-D7 inputs of the DAC0808 and the reference current I_{ref}, and is as follows:

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

Usually reference current is 2mA. Ideally we connect the output pin to a resistor, convert this current to voltage, and monitor the output on the scope. But this can cause inaccuracy; hence an opamp is used to convert the output current to voltage. The 8051 connection to DAC0808 is as shown in the figure below.

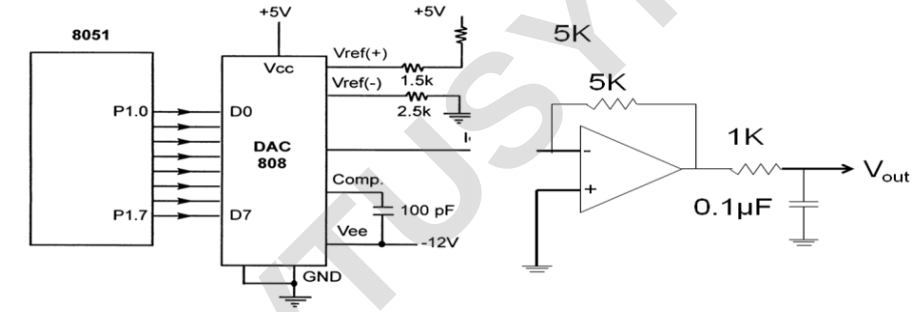


Figure : 8051 connection to DAC0808

Example 9: Write an ALP to generate a triangular waveform.
Program:

```
        MOV A, #00H
INCR:   MOV P1, A
        INC A
        CJNE A, #255, INCR
DECR:   MOV P1, A
        DEC A
        CJNE A, #00, DECR
        SJMP INCR
        END
```

Example 10: Write an ALP to generate a sine waveform.

$V_{out} = 5V(1+\sin\theta)$

Solution: Calculate the decimal values for every 10 degree of the sine wave. These values can be maintained in a table and simply the values can be sent to port P1. The sinewave can be observed on the CRO.

Program:

```
        ORG 0000H
AGAIN:MOV DPTR, #SINETABLE
        MOV R3, #COUNT
  UP:CLR A
        MOVC A, @A+DPTR
        MOV P1, A
        INC DPTR
        DJNZ R3, UP
        SJMP AGAIN
        ORG 0300H
SINETABLE DB 128, 192, 238, 255, 238, 192, 128, 64, 17, 0, 17, 64, 128
        END
```

Note: to get a better wave regenerate the values of the table per 2 degree.

Sensor interfacing and signal conditioning

The sensors of the LM34/LM35 series are precision integrated-circuit temperature sensors whose output voltage is linearly proportional to the Fahrenheit/Celsius temperature. The LM34/LM35 requires no external calibration since it is inherently calibrated. It outputs 10 mV for each degree of Fahrenheit/Celsius temperature.

Temperature sensors

Transducers convert physical data such as temperature, light intensity, flow, and speed to electrical signals. Depending on the transducer, the output produced is in the form of voltage, current, resistance, or capacitance. For example, temperature is converted to electrical signals using a transducer called a *thermistor.* A thermistor responds to temperature change by changing resistance, but its response is not linear.

LM34 and LM35 temperature sensors

The sensors of the LM34 series are precision integrated-circuit temperature sensors whose output voltage is linearly proportional to the Fahrenheit temperature. See Table 13-9. The LM34 requires no external calibration since it is internally calibrated. It outputs 10 mV for each degree of Fahrenheit temperature. Table 13-9 is a selection guide for the LM34.

Table 13-9: LM34 Temperature Sensor Series Selection Guide

Part	Temperature Range	Accuracy	Output Scale
LM34A	–50 F to +300 F	+2.0 F	10 mV/F
LM34	–50 F to +300 F	+3.0 F	10 mV/F
LM34CA	–40 F to +230 F	+2.0 F	10 mV/F
LM34C	–40 F to +230 F	+3.0 F	10 mV/F
LM34D	–32 F to +212 F	+4.0 F	10 mV/F

Note: Temperature range is in degrees Fahrenheit.

The LM35 series sensors are precision integrated-circuit temperature sensors whose output voltage is linearly proportional to the Celsius (centigrade) temperature. The LM35 requires no external calibration since it is internally calibrated. It outputs 10 mV for each degree of centigrade temperature.

Signal Conditioning

Signal conditioning is widely used in the world of data acquisition. The most common transducers produce an output in the form of voltage, current, charge, capacitance, and resistance. However, we need to convert these signals to voltage in order to send input to an A-to-D converter. This conversion (modification) is commonly called *signal conditioning.* Signal conditioning can be a current-to-voltage conversion or a signal amplification. For example, the thermistor changes resistance with temperature. The change of resistance must be translated into voltages in order to be of any use to an ADC. Look at the case of connecting an LM35 to an ADC0848. Since the ADC0848 has 8-bit resolution with a maximum of 256 (2^8) steps and the LM35 (or LM34) produces 10 mV for every degree of temperature change, we can condition V_{in} of the ADC0848 to produce a V_{out} of 2560 mV (2.56 V) for full-scale output. Therefore, in order to produce the full-scale V_{out} of 2.56 V for the ADC0848, we need to set V_{ref} = 2.56. This makes V_{out} of the ADC0848 correspond directly to the temperature as monitored by the LM35.
Figure 13-21 shows the connection of a temperature sensor to the ADC0848.

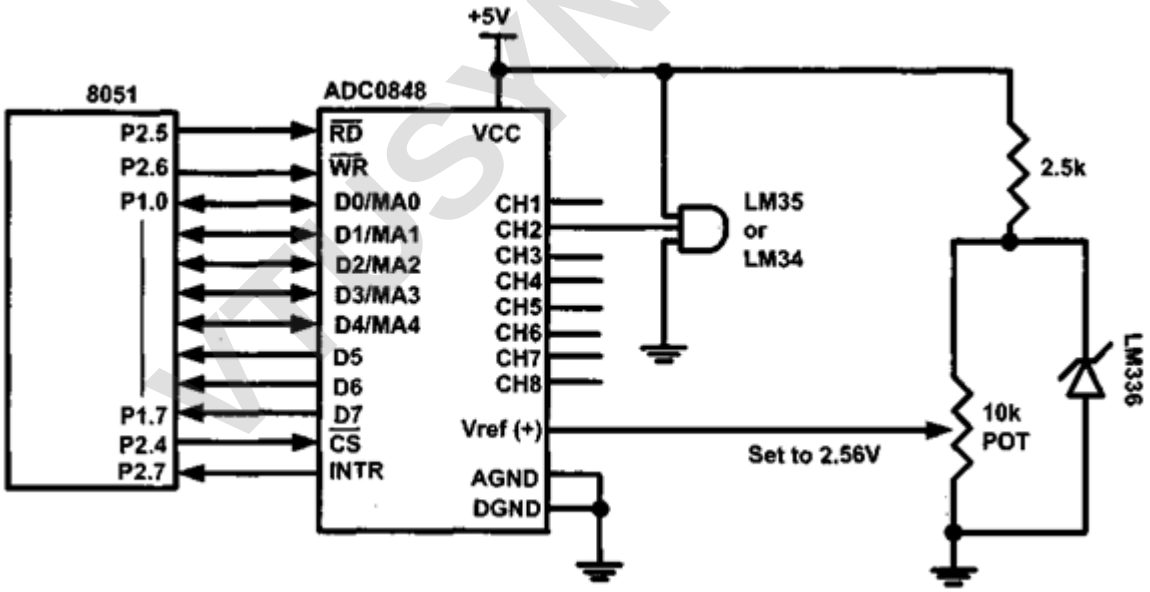


Figure. 8051 Connection to ADC0848 and Temperature Sensor

Stepper Motor Interfacing

Stepper motor is a widely used device that translates electrical pulses into mechanical movement. Stepper motor is used in applications such as; disk drives, dot matrix printer, robotics etc,. The construction of the motor is as shown in figure 1 below.

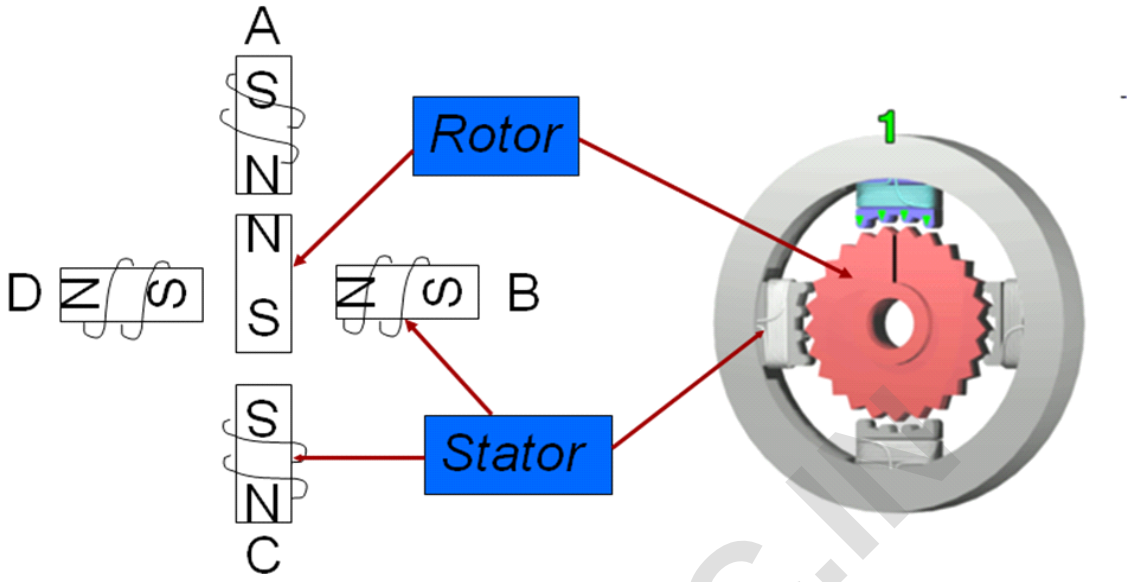


Figure: Structure of stepper motor

It has a permanent magnet rotor called the shaft which is surrounded by a stator. Commonly used stepper motors have four stator windings that are paired with a center – tapped common. Such motors are called as four-phase or unipolar stepper motor.

The stator is a magnet over which the electric coil is wound. One end of the coil are connected commonly either to ground or +5V. The other end is provided with a fixed sequence such that the motor rotates in a particular direction. Stepper motor shaft moves in a fixed repeatable increment, which allows one to move it to a precise position. Direction of the rotation is dictated by the stator poles. Stator poles are determined by the current sent through the wire coils.

Step angle:

Step angle is defined as the minimum degree of rotation with a single step.

No of steps per revolution = 360° / step angle

Steps per second = (rpm x steps per revolution) / 60

Example: step angle = 2°

No of steps per revolution = 180

Switching Sequence of Motor:

As discussed earlier the coils need to be energized for the rotation. This can be done by sending a bits sequence to one end of the coil while the other end is commonly connected. The bit sequence sent can make either one phase ON or two phase ON for a full step sequence or it can be a combination of one and two phase ON for half step sequence. Both are tabulated below.

Full Step:

Two Phase ON

Clockwise

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

Counter-clockwise

One Phase ON

Clockwise

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

Counter-clockwise

Half Step (8 – sequence):

The sequence is tabulated as below:

Clk

Step #	A	B	C	D
1	1	0	0	1
2	1	0	0	0
3	1	1	0	0
4	0	1	0	0
5	0	1	1	0
6	0	0	1	0
7	0	0	1	1
8	0	0	0	1

Anti-Clk

8051 Connection to Stepper Motor: (explanation of the diagram can be done)

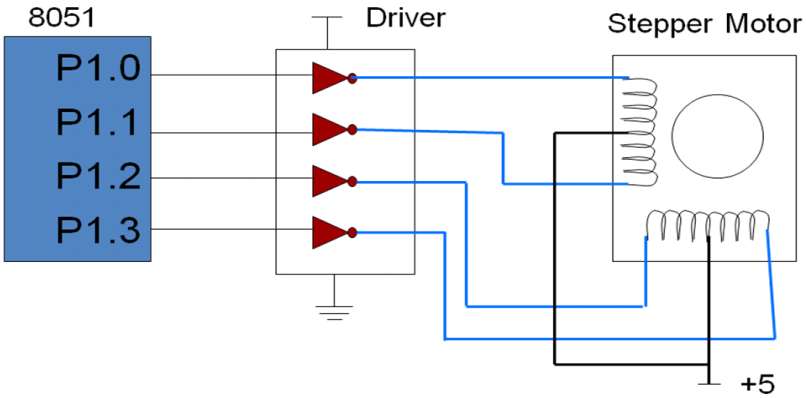


Figure : 8051 interface to stepper motor

Example 1: Write an ALP to rotate the stepper motor clockwise / anticlockwise continuously with full step sequence.

Program:

```
        MOV A,#66H
BACK: MOV P1,A
        RR A
        ACALL DELAY
        SJMP BACK
DELAY: MOV R1,#100
 UP1:    MOV R2,#50
 UP:     DJNZ R2,UP
         DJNZ R1,UP1
         RET
```

Note: motor to rotate in anticlockwise use instruction RL A instead of RR A

Example 2: A switch is connected to pin P2.7. Write an ALP to monitor the status of the SW. If SW = 0, motor moves clockwise and if SW = 1, motor moves anticlockwise.

Program:

```
        ORG 0000H
        SETB P2.7
        MOV A, #66H
        MOV P1,A
TURN: JNB P2.7, CW
        RL A
        ACALL DELAY
        MOV P1,A
        SJMP TURN
CW:    RR A
        ACALL DELAY
        MOV P1,A
        SJMP TURN
DELAY: as previous example
```

Example 4: Rotate the stepper motor continuously clockwise using half-step 8-step sequence. Say the sequence is in ROM locations.

Program:

```
        ORG 0000H
START: MOV R0, #08
         MOV DPTR, #HALFSTEP
RPT:    CLR A
         MOVC A, @A+DPTR
         MOV P1, A
         ACALL DELAY
         INC DPTR
         DJNZ R0, RPT
         SJMP START
```

```
        ORG 0200H
HALFSTEP DB 09, 08, 0CH, 04, 06, 02, 03, 01
        END
```

Programming Stepper Motor with 8051 C

The following examples 5 and 6 will show the programming of stepper motor using 8051 C.

Example 5: Problem definition is same as example 1.

Program:

```
#include <reg51.h>
void main ()
{
        while (1)
         {
            P1=0x66;
            MSDELAY (200);
            P1=0x33;
            MSDELAY (200);
                    P1=0x99;
            MSDELAY (200);
                 P1=0xCC;
            MSDELAY (200);
         }
}
void MSDELAY (unsigned char value)
{
         unsigned int x,y;
         for(x=0;x<1275;x++)
         for(y=0;y<value;y++);
}
```

Example 6: Problem definition is same as example 2.

Program:

```
#include <reg51.h>
sbit SW=P2^7;
void main ()
{
        SW=1;
         while (1)
         {
           if(SW==0){
            P1=0x66;
            MSDELAY (100);
            P1=0x33;
            MSDELAY (100);
```

```
            P1=0x99;
            MSDELAY (100);
            P1=0xCC;
            MSDELAY (100);
        }
        else {
            P1=0x66;
            MSDELAY (100);
            P1=0xCC;
            MSDELAY (100);
            P1=0x99;
            MSDELAY (100);
            P1=0x33;
            MSDELAY (100);
}
void MSDELAY (unsigned char value)
{
        unsigned int x,y;
        for(x=0;x<1275;x++)
        for(y=0;y<value;y++);
}
```

DC Motor Interfacing with 8051:

The DC motor is another widely used device that translates electrical pulses into mechanical movement. Motor has 2 leads +ve and – ve , connecting them to a DC voltage supply moves the motor in one direction. On reversing the polarity rotates the motor in the reverse direction. Basic difference between Stepper and DC motor is stepper motor moves in steps while DC motor moves continuously. Another difference is with stepper motor the number of steps can be counted while it is not possible in DC motor. Maximum speed of a DC motor is indicated in rpm. The rpm is either with no load it is few thousands to tens of thousands or with load rpm decreases with increase in load.

Voltage and current rating : Nominal voltage is the voltage for a motor under normal condition. It ranges from 1V to 150V. As voltage increases, rpm goes up. Current rating is the current consumption when the nominal voltage is applied with no load that is 25mA to a few amperes. As load increases, rpm increases, unless voltage or current increases implies torque increases. With fixed voltage, as load increases, power consumption of a DC motor is increased.

Unidirectional Control:

Figure 3 shows the rotation of the DC motor in clockwise and anticlockwise direction.

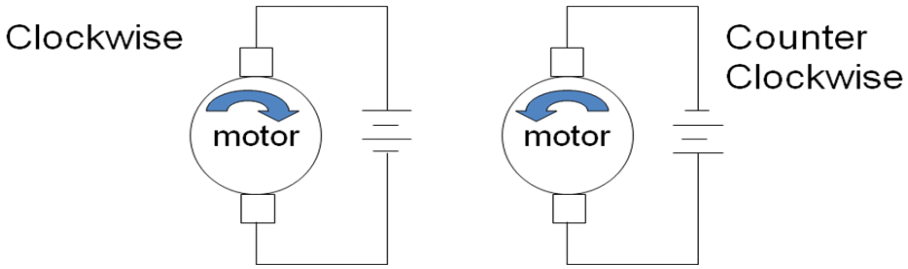
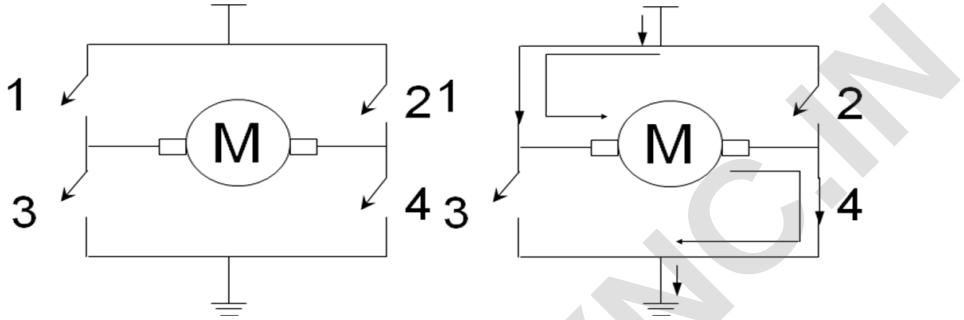
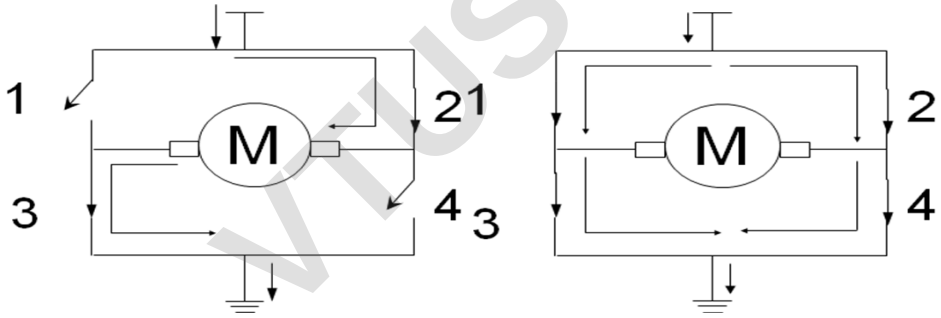


Figure 3: DC motor rotation

Bidirectional Control:



- Motor not running (b) Clockwise direction



(c) Counter clockwise direction (d) Invalid state (short circuit)

Figure : H-Bridge Motor Configuration

Figure shows the H-Bridge motor configuration. It consists of four switches and based on the closing and opening of these switches the motor either rotates in clockwise or anti-clockwise direction.

As seen in figure 4a, all the switches are open hence the motor is not running. In b, turning of the motor is in one direction when the switches 1 and 4 are closed that is clockwise direction.

Similarly, in c the switches 2 and 3 are closed so the motor rotates in anticlockwise direction, while in figure 4d all the switches are closed which indicates a invalid state or a short circuit.

The interfacing diagram of 8051 to bidirectional motor control can be referred to fig 17-18 from text prescribed.

Example 6: A switch is connected to pin P2.7. Write an ALP to monitor the status of the SW. If SW = 0, DC motor moves clockwise and if SW = 1, DC motor moves anticlockwise.

Program:

```
        ORG 0000H
        CLR P1.0
        CLR P1.1
        CLR P1.2
        CLR P1.3
        SETB P2.7
MONITOR: JNB P2.7, CLOCK
        SETB P1.0
        CLR P1.1
        CLR P1.2
        SETB P1.3
        SJMP MONITOR
CLOCK:  CLR P1.0
        SETB P1.1
        SETB P1.2
        CLR P1.3
        SJMP MONITOR
        END
```

Pulse Width Modulation (PWM)

The speed of the motor depends on 3 parameters: load, voltage and current. For a given load, we can maintain a steady speed by using PWM. By changing the width of the pulse applied to DC motor, power provided can either be increased or decreased. Though voltage has fixed amplitude, has a variable duty cycle. The wider the pulse, higher the speed obtained. One of the reasons as to why dc motor are referred over ac is, the ability to control the speed of the DC motor using PWM. The speed of the ac motor is dictated by the ac frequency of voltage applied to the motor and is generally fixed. Hence, speed of the AC motors cannot be controlled when load is increased.

Figure below shows the pulse width modulation comparison.

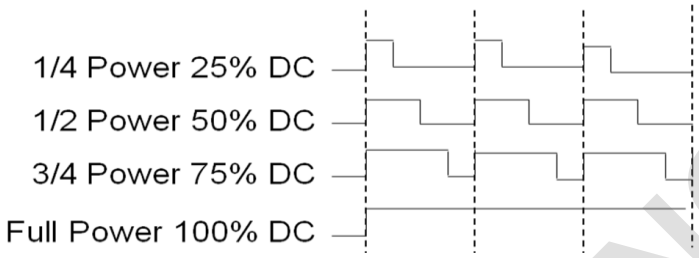


Figure : PWM comparison

Example 7 and 8 are the 8051 C version of the programs written earlier.

Example 7: A switch is connected to pin P2.7. Write a C to monitor the status of the SW. If SW = 0, DC motor moves clockwise and if SW = 1, DC motor moves anticlockwise.

Program:

```
# include <reg51.h>
sbit SW =P2^7;
sbit Enable = P1^0;
sbit MTR_1 = P1^1;
sbit MTR_2 = P1^2;
void main ( )
{
        SW=1;
        Enable = 0;
        MTR_1=0;
        MTR_2=0;
while( )
{
        Enable =1;
        if( SW==1)
        {       MTR_1=1;
                MTR_2=0;
        }
        else
        {       MTR_1=0;
```

```
            MTR_2=1;
        }
        }
        }
```

Example 8: A switch is connected to pin P2.7. Write an C to monitor the status of the SW. If SW = 0, DC motor moves 50% duty cycle pulse and if SW = 1, DC motor moves with 25% duty cycle pulse.

Program:

```
# include <reg51.h>
sbit SW =P2^7;
sbit MTR = P1^0;
void main ( )
{
        SW=1;
        MTR=0;
while( )
{       if( SW==1)
        {       MTR=1;
                Msdelay(25);
                MTR=0;
                Msdelay(75);
        }
        else
        {       MTR=1;
                Msdelay(50);
                MTR=0;
                Msdelay(50);
        }
        }
        }
```

8051 interfacing with the 8255

The Intel 8255 Programmable Peripheral Interface (PPI) chip, developed and manufactured by Intel in the first half of the 1970s for the Intel 8080 microprocessor. The 8255 provides 24 parallel input/output lines with a variety of programmable operating modes. The 8255 gives a CPU or digital system access to programmable parallel I/O. The 8255 has 24 input/output pins. These are divided into three 8-bit ports (A, B, C). Port A and port B can be used as 8-bit input/output ports. Port C can be used as an 8-bit input/output port or as two 4-bit input/output ports or to produce handshake signals for ports A and B.

The three ports are further grouped as follows:

1. Group A consisting of port A and upper part of port C.
2. Group B consisting of port B and lower part of port C.

Eight data lines (D0–D7) are available (with an 8-bit data buffer) to read/write data into the ports or control register under the status of the **RD** (pin 5) and **WR** (pin 36), which are active-low

signals for read and write operations respectively. Address lines A_1 and A_0 allow to access a data register for each port or a control register.

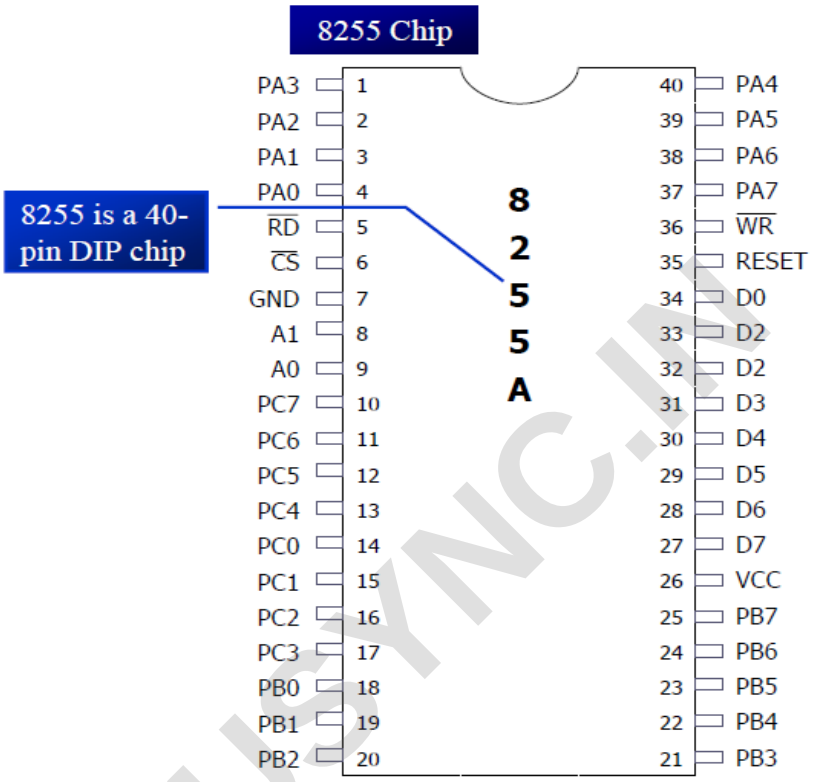


Figure: Pin diagram of 8255

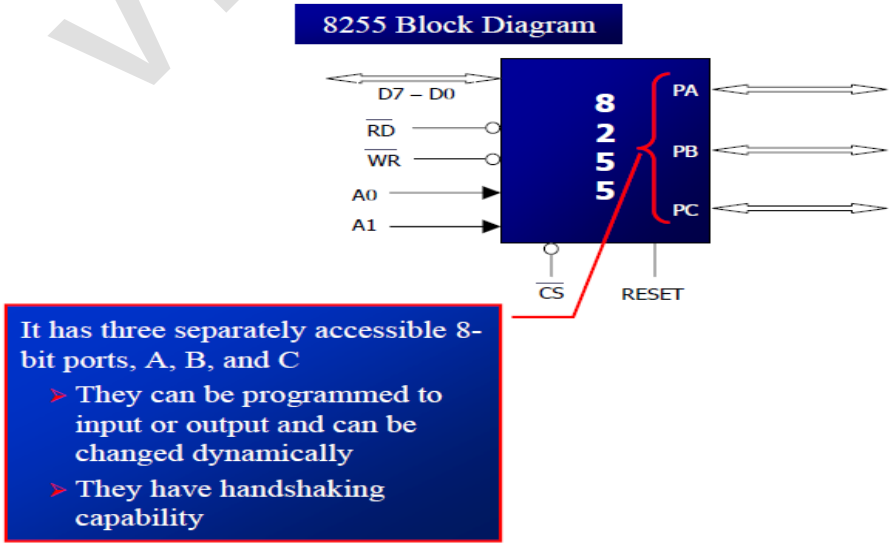


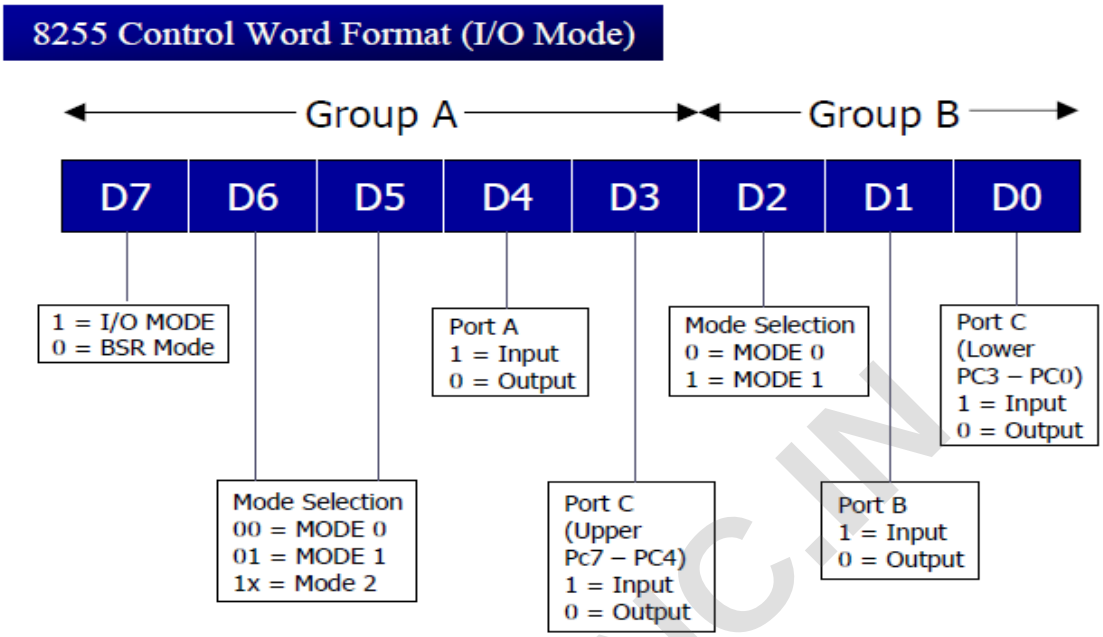
Figure: Block diagram of 8255

PA0 - PA7 (8-bit port A): Can be programmed as all input or output, or all bits as bidirectional input/output.
PB0 - PB7 (8-bit port B): Can be programmed as all input or output, but cannot be used as a bidirectional port.
PC0 – PC7 (8-bit port C): Can be all input or output, can also be split into two parts.
CU (upper bits PC4 - PC7), CL (lower bits PC0 – PC3) each can be used for input or output
Any of bits PC0 to PC7 can be programmed individually.
RD and WR: These two active-low control signals are inputs to the 8255. The RD and WR signals from the 8031/51 are connected to these inputs.
D0 – D7: are connected to the data pins of the microcontroller allowing it to send data back and forth between the controller and the 8255 chip.
RESET: An active-high signal input. Used to clear the control register. When RESET is activated, all ports are initialized as input ports.
A0, A1, and CS (chip select): CS is active-low. While CS selects the entire chip, it is A0 and A1 that select specific ports. These 3 pins are used to access port A, B, C or the control register.

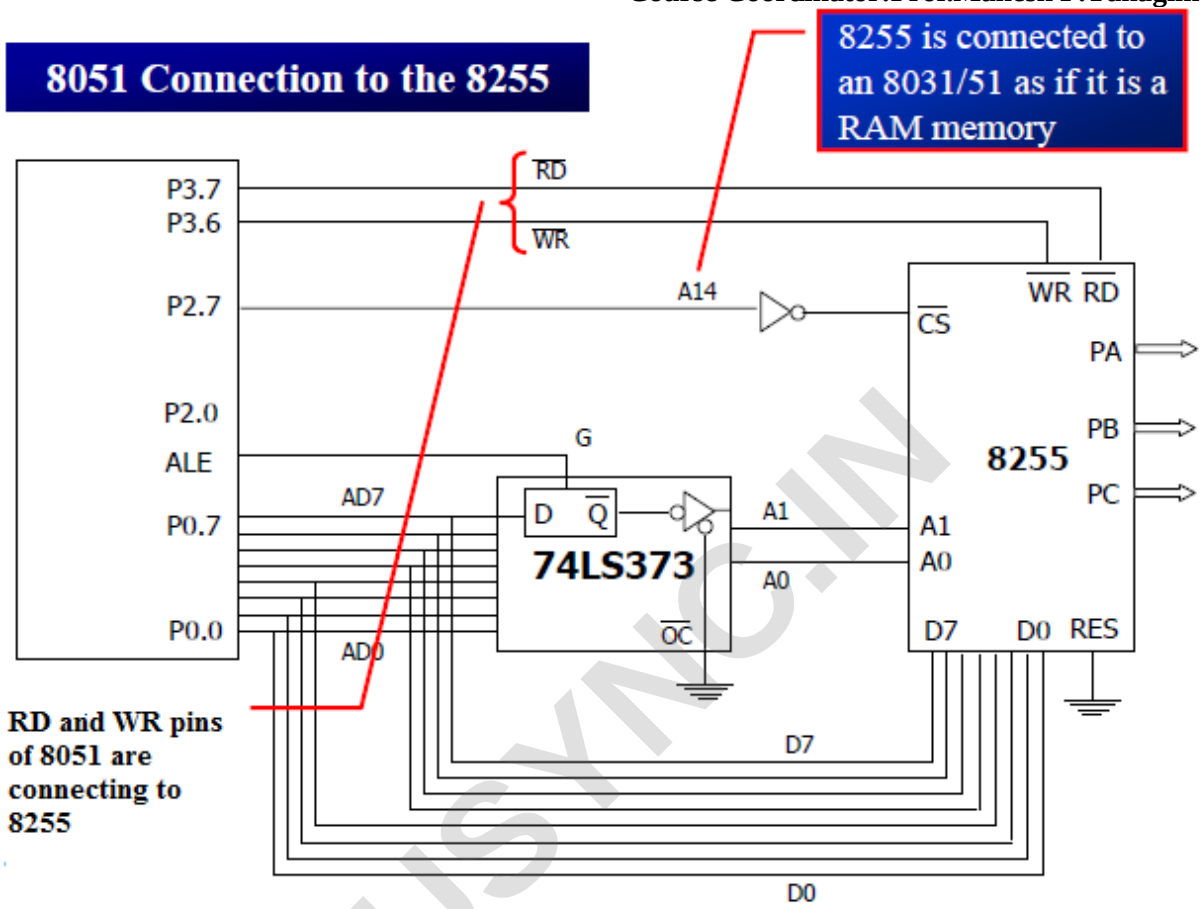
8255 Port Selection

CS	A1	A0	Selection
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Control register
1	X	X	8255 is not selected

While ports A, B and C are used to input or output data, the control register must be programmed to select operation mode of three ports. The ports of the 8255 can be programmed in any of the following modes:
1. Mode 0, simple I/O: Any of the ports A, B, CL, and CU can be programmed as input out output. All bits are out or all are in. There is no signal-bit control as in P0-P3 of 8051
2. Mode 1: Port A and B can be used as input or output ports with handshaking capabilities. Handshaking signals are provided by the bits of port C.
3. Mode 2: Port A can be used as a bidirectional I/O port with handshaking capabilities provided by port C. Port B can be used either in mode 0 or mode 1.
4. BSR (bit set/reset) mode
Only the individual bits of port C can be Programmed.



The more commonly used term is I/O.
Mode 0: Intel calls it the basic input/output mode. In this mode, any ports of A, B, or C can be programmed as input or output . A given port cannot be both input and output at the same time
The 8255 chip is programmed in any of the 4 modes mentioned earlier by sending a byte (Intel calls it a control word) to the control register of 8255. We must first find the port address assigned to each of ports A, B ,C and the control register called mapping the I/O port.



Example

Find the control word of the 8255 for the following configurations:

(a) All the ports of A, B and C are output ports (mode 0)

(b) PA = in, PB = out, PCL = out, and PCH = out

Solution:

From Figure 15-3 we have:

(a) 1000 0000 = 80H (b)1001 0000 = 90H.

Example

For Figure shown below

(a) Find the I/O port addresses assigned to ports A, B, C, and the control register.

(b) Program the 8255 for ports A, B, and C to be output ports.

(c) Write a program to send 55H and AAH to all ports continuously.

Solution

(a) The base address for the 8255 is as follows:

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	
X	1	X	x	X	x	x	x	x	x	x	x	x	x	0	0	= 4000H PA
X	1	X	X	x	x	x	x	x	x	x	x	x	X	0	1	= 4001H PB
X	1	X	X	x	x	x	x	x	x	x	x	X	X	1	0	= 4002H PC
x	1	x	X	x	x	x	x	x	x	x	x	x	x	1	1	= 4003H CR

(b) The control byte (word) for all ports as output is 80H.

(c)

```
        MOV A,#80H ;control word;(ports output)
        MOV DPTR,#4003H ;load control reg; port address
        MOVX @DPTR, A ;issue control word
        MOV A,#55H        ;A = 55H
AGAIN: MOV DPTR,#4000H ;PA address
        MOVX @DPTR,A ;toggle PA bits
        INC DPTR ;PB address
        MOVX @DPTR,A ;toggle PB bits
        INC DPTR ;PC address
        MOVX @DPTR,A ;toggle PC bits
        CPL A ;toggle bit in reg A
ACALL DELAY ;wait
         SJMP AGAIN ;continue.
```