UNIT - 5

8051 INTERFACING ANDAPPLICATIONS

Interfacing of 8051with: Analog Sensors, Keypad & LCD display, ADC, DAC, DC motor.

LCD INTERFACING:

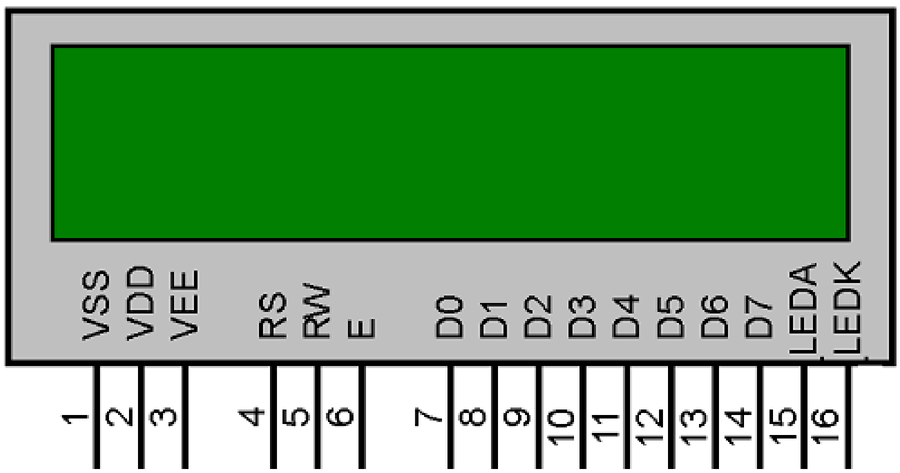


Figure 5.1 16X2 LCD Module

- 16×2 LCD module is a very common type of LCD module.
- It consists of 16 rows and 2 columns of 5×7 or 5×8 LCD dot matrices.
- It is available in a 16 pin package with back light, contrast adjustment function and each dot matrix has 5×8 dot resolution.
- The pin numbers, their name and corresponding functions are shown in the table 5.1.

Table 5.1 LCD Pin Description

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

V_{CC},V_{SS} & V_{EE} Pin:

- V_{CC} and V_{SS} provide +5V and Ground

- V_{EE} pin is meant for adjusting the contrast of the LCD display and the contrast can be adjusted by varying the voltage at this pin.
- This is done by connecting one end of a POT to the Vcc (5V), other end to the Ground and connecting the center terminal (wiper) of of the POT to the VEE pin. (Refer Figure 5.2)

RS:

- LCD has two built in registers namely data register and command register.
- Data register is for placing the data to be displayed, and the command register is to place the commands.
- High logic at the RS pin will select the data register and Low logic at the RS pin will select the command register.
- If we make the RS pin high and the put a data in the 8 bit data line (DB0 to DB7), the LCD module will recognize it as a data to be displayed.
- If we make RS pin low and put a data on the data line, the module will recognize it as a command.

R/W:

- R/W pin is meant for selecting between read and write modes.
- High level at this pin enables read mode and low level at this pin enables write mode.

Enable (E):

- E pin is for enabling the module.
- The enable pin is used by the LCD to latch information presented to its data pins.
- When data is supplied to data pins, a high to low pulse must be applied to this pin in order for the LCD to latch in the data present at the data pins.
- This pulse must be a minimum of 450ns wide.

Data Pin:

- The 8-bit data pins, DB0 to DB7 are used to send information to the LCD or read the contents of the LCD's internal register.
- To display letters and numbers, send ASCII codes for the letters A-Z; a-z and numbers 0-9 to these pins while making RS=1.
- There are also instruction command codes that can be sent to the LCD to clear the display or force the cursor to the home position or blink the cursor.
- Table 5.2 Lists the instructions command codes.

Table 5.2 LCD Command Codes

CODE(Hexa Decimal)	COMMAND
01	Clear display screen
02	Return Home
04	Decrement cursor (shift cursor to left)
05	Increment cursor (shift cursor to right)
06	shift display right
07	shift display left
08	Display off, cursor off
0A	Display off, cursor on

0C	Display on, cursor off
0E	Display on, cursor blinking
0F	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 the beginning of 1st line
C0	Force cursor to the beginning of 2nd line
38	2 lines and 5 x 7 matrix

- We also use RS=0 to check the busy flag bit to see if the LCD is ready to receive information's.
- The busy flag id D7 and can be read when R/W=1 and RS=0, as follows: if R/W=1, RS=0.
- When D7=1 (busy flag =1), the LCD is busy taking care of internal operations and will not accept any new information.

LED+ & LED-:

- LED+ is the anode of the back light LED and this pin must be connected to Vcc through a suitable series current limiting resistor.
- LED- is the cathode of the back light LED and this pin must be connected to ground.

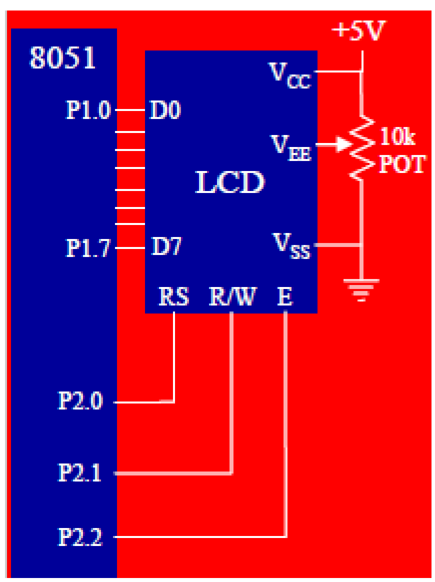


Figure 5.2 LCD Interfacing With 8051

LCD initialization

- The steps that has to be done for initializing the LCD display is given below and these steps are common for almost all applications.
  - Send 38H to the 8 bit data line for initialization
  - Send 0FH for making LCD ON, cursor ON and cursor blinking ON.

- Send 06H for incrementing cursor position.
- Send 80H for displaying the character from 1st row and 1st column in LCD
- Send 01H for clearing the display and return the cursor.

Sending data to the LCD.

- The steps for sending data to the LCD module is given below.
- It is the logic state of the pins (RS, R/W and E) that make the module to determine whether a given data input is a command or data to be displayed.
  - Make R/W low.
  - Make RS=0 if data byte is a command and make RS=1 if the data byte is a data to be displayed.
  - Place data byte on the data register.
  - Pulse E from high to low.
  - Repeat above steps for sending another data

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 0
        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,#84H          ;cursor at line 1, pos. 4
        ACALL COMNWRT       ;call command subroutine
        ACALL DELAY         ;give LCD some time
        MOV A,#'N'          ;display letter N
        ACALL DATAWRT       ;call display subroutine
        ACALL DELAY         ;give LCD some time
        MOV A,#'O'          ;display letter O
        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
            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
            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
            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
```

Check Busy Flag:

- The above code showed how to send command to the LCD without checking the busy flag.
- Notice that we put a long delay between issuing data or commands to the LCD.
- However a much better way is to monitor the busy flag before issuing a command or data to the LCD. This is shown in below program

```
            ORG 0
            MOV A,#38H        ;INIT. LCD 2 LINES, 5X7 MATRIX
            ACALL COMNWRT     ;call command subroutine
            MOV A,#0EH        ;display on, cursor on
            ACALL COMNWRT     ;call command subroutine
            MOV A,#01         ;clear LCD
            ACALL COMNWRT     ;call command subroutine
            MOV A,#06H        ;shift cursor right
            ACALL COMNWRT     ;call command subroutine
            MOV A,#84H        ;cursor at line 1, pos. 4
            ACALL COMNWRT     ;call command subroutine
            MOV A,#'N'        ;display letter N
            ACALL DATAWRT     ;call display subroutine
```

```
                MOV A,#'O'              ;display letter O
                ACALL DATAWRT           ;call display subroutine
AGAIN:          SJMP AGAIN              ;stay here

COMNWRT:        ACALL READY             ;send command to LCD if LCD is ready
                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
                CLR P2.2                ;E=0 for H-to-L pulse
                RET

DATAWRT:        ACALL READY             ;write data to LCD if LCD is ready
                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
                CLR P2.2                ;E=0 for H-to-L pulse
                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
                                        ;read command reg and check busy flag
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
```

LCD Interfacing Using MOVC Instruction:

```
                ORG 0
                MOV DPTR,#MYCOM
        C1:     CLR A
                MOVC A,@A+DPTR
                ACALL COMNWRT                   ;call command subroutine
                ACALL DELAY                     ;give LCD some time
                INC DPTR
                JZ SEND_DAT
                SJMP C1
SEND_DAT:
```

```
            MOV DPTR,#MYDATA
     D1:    CLR A
            MOVC A,@A+DPTR
            ACALL DATAWRT           ;call command subroutine
            ACALL DELAY             ;give LCD some time
            INC DPTR
            JZ AGAIN
            SJMP D1
AGAIN:      SJMP AGAIN

            ORG 300H
MYCOM:      DB 38H,0EH,01,06,84H,0  ; commands and null
MYDATA:     DB “HELLO”,0
            END
```

KEYBOARD INTERFACING:

- At the lowest level, 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

<u>Scanning and identifying the key</u>

- Figure 5.3 shows a 4 x4 matrix connected to two ports.
- The rows are connected to an output port and the columns are connected to an input port.
- If no key has been pressed, reading the input port will yield 1s for all columns since they are all connected to high (Vcc).
- If all the rows are grounded and a key is pressed, one of the columns will have 0 since the key pressed provides the path to ground.
- It is the function of the microcontroller to scan the keyboard continuously to detect and identify the key pressed, How it is done is explained next.

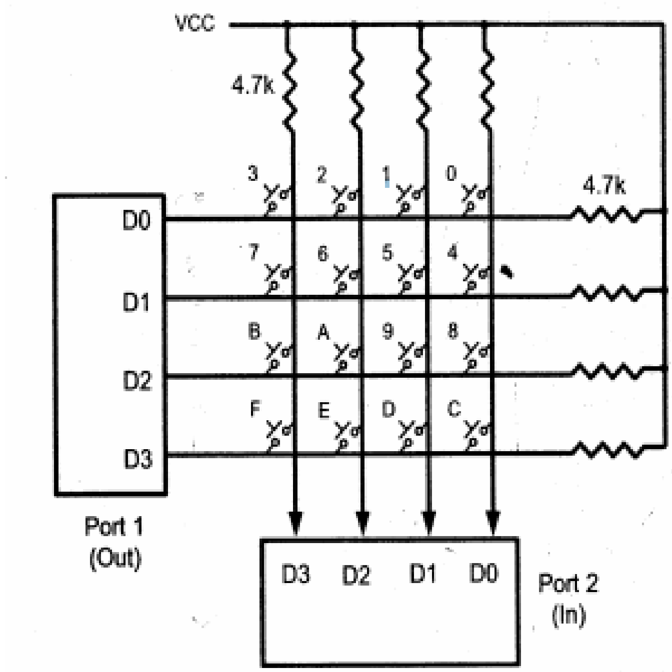


Figure 5.3 Matrix Keyboard Connection to Ports

Grounding rows and reading the columns

- 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 the columns is D3 - D0 =1111, no key has been pressed and the process continues until a key press is detected.
- However, 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 a key press is detected, the 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; then 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, the next task is to find out which column the pressed key belongs to.
- This should be easy since the microcontroller knows at any time which row and column are being accessed.
- Given keyboard program is the 8051 Assembly language program for detection and identification of key activation.
- In this program, it is assumed that P1 and P2 are initialized as output and input, respectively.
- Program goes through the following four major stages:

- 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.
- 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, the microcontroller waits 20 ms for the bounce and then scans the columns again. This serves two functions: (a) it ensures that the first key press detection was not an erroneous one due to a spike noise, and (b) the 20-ms delay prevents the same key press from being interpreted as a multiple key press. If after the 20-ms delay the key is still pressed, it goes to the next stage to detect which row it belongs to; otherwise, it goes back into the loop to detect a real key press.
- To detect which row the key press belongs to, the microcontroller 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 the ASCII value) for that row and goes to the next stage to identify the key.
- To identify the key press, the microcontroller 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. Figure 5.4 flowcharts this process.

➢ While the key press detection is standard for all keyboards, the process for determining which key is pressed varies.

➢ The look-up table method shown in keyboard Program can be modified to work with any matrix up t0 8 x 8.

➢ Figure 5.4 provides the flowchart for keyboard interfacing Program for scanning and identifying the pressed key.

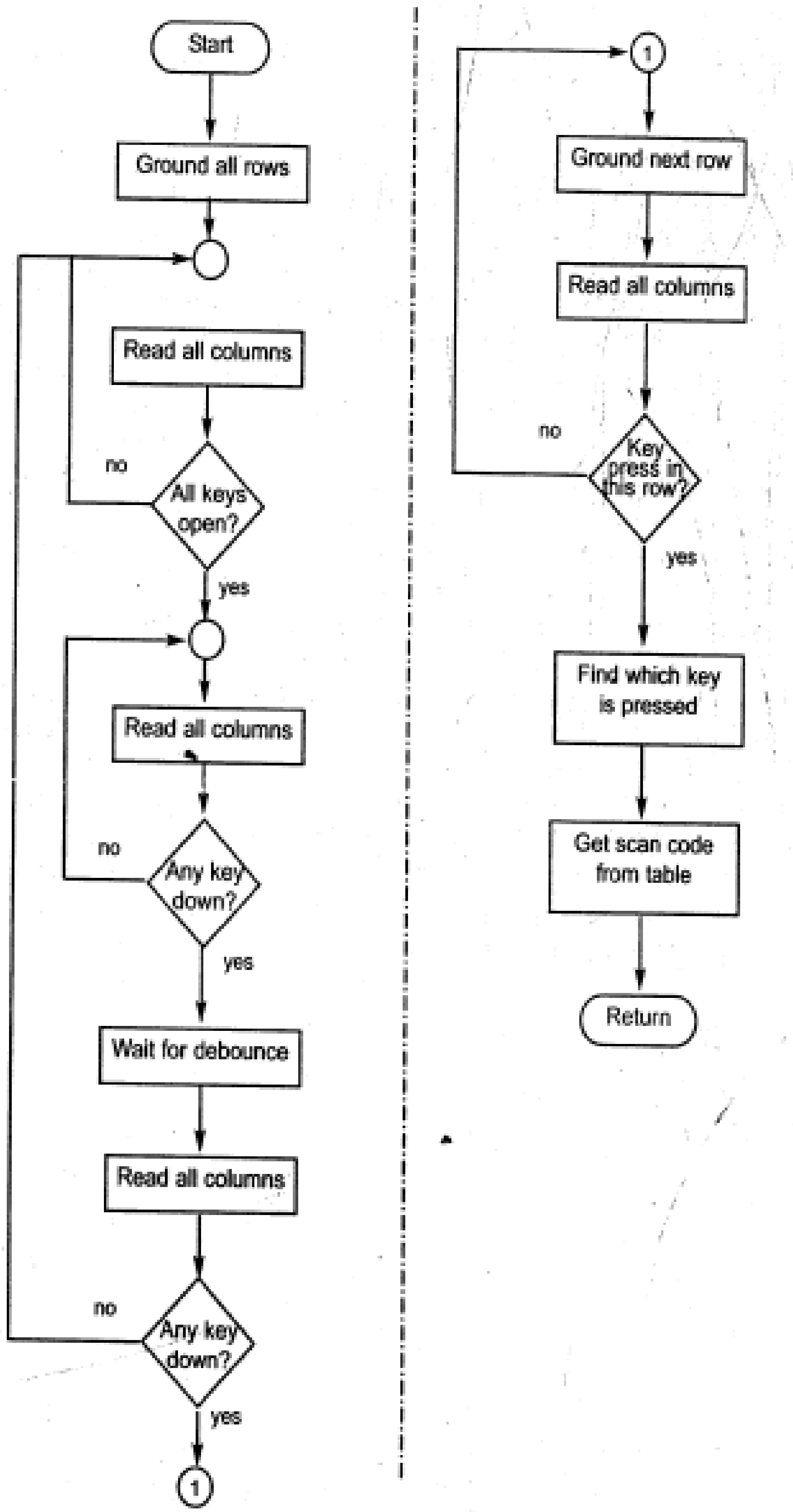


Figure 5.4 Flowchart for Programming Keyboard Interfacing

Program:

```
                    ;keyboard subroutine. This program sends the ASCII
                    ;code for pressed key to P0.1
                    ;P1.0-P1.3 connected to rows, P2.0-P2.3 to column
        MOV P2,#0FFH                ;make P2 an input port
K1:     MOV P1,#0                   ;ground all rows at once
        MOV A,P2                    ;read all col
                                    ;(ensure keys open)
        ANL A,00001111B             ;masked unused bits
        CJNE A,#00001111B,K1        ;till all keys release
K2:     ACALL DELAY                 ;call 20 msec delay
        MOV A,P2                    ;see if any key is pressed
        ANL A,00001111B             ;mask unused bits
        CJNE A,#00001111B,OVER      ;key pressed, find row
        SJMP K2                     ;check till key pressed
OVER: ACALL DELAY                   ;wait 20 msec debounce time
        MOV A,P2                    ;check key closure
        ANL A,00001111B             ;mask unused bits
        CJNE A,#00001111B,OVER1;key pressed, find row
        SJMP K2                     ;if none, keep polling
OVER1: MOV P1, #11111110B           ;ground row 0
        MOV A,P2                    ;read all columns
        ANL A,#00001111B            ;mask unused bits
        CJNE A,#00001111B,ROW_0 ;key row 0, find col.
        MOV P1,#11111101B           ;ground row 1
        MOV A,P2                    ;read all columns
        ANL A,#00001111B            ;mask unused bits
        CJNE A,#00001111B,ROW_1 ;key row 1, find col.
        MOV P1,#11111011B           ;ground row 2
        MOV A,P2                    ;read all columns
        ANL A,#00001111B            ;mask unused bits
        CJNE A,#00001111B,ROW_2 ;key row 2, find col.
        MOV P1,#11110111B           ;ground row 3
        MOV A,P2                    ;read all columns
        ANL A,#00001111B            ;mask unused bits
        CJNE A,#00001111B,ROW_3 ;key row 3, find col.
        LJMP K2                     ;if none, false input,
                                    ;repeat
ROW_0: MOV DPTR,#KCODE0             ;set DPTR=start of row 0
        SJMP FIND                   ;find col. Key belongs to
ROW_1: MOV DPTR,#KCODE1             ;set DPTR=start of row
        SJMP FIND                   ;find col. Key belongs to
ROW_2: MOV DPTR,#KCODE2             ;set DPTR=start of row 2
        SJMP FIND                   ;find col. Key belongs to
```

```
        ROW_3: MOV DPTR,#KCODE3 ;set DPTR=start of row 3
FIND:   RRC A                   ;see if any CY bit low
        JNC MATCH               ;if zero, get ASCII code
        INC DPTR                ;point to next col. addr
        SJMP FIND               ;keep searching
MATCH: CLR A                    ;set A=0 (match is found)
        MOVC A,@A+DPTR          ;get ASCII from table
        MOV P0,A                ;display pressed key
        LJMP K1
                                ;ASCII LOOK-UP TABLE FOR EACH ROW
        ORG 300H
KCODE0: DB '0','1','2','3' ;ROW 0
KCODE1: DB '4','5','6','7' ;ROW 1
KCODE2: DB '8','9','A','B' ;ROW 2
KCODE3: DB 'C','D','E','F' ;ROW 3
        END
```

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 and process them.
- An ADC has n-bit resolution where n can be 8, 10, 12, 16 or even 24 bits.
- The higher-resolution ADC provides a smaller step size, where step size is the smallest change that can be discerned by an ADC. This is shown in table 5.3

Table 5.3 Resolution Vs Step Size for ADC

n-bit	Number of steps	Step Size (mV)
8	256	5/256 = 19.53
10	1024	5/1024 = 4.88
12	4096	5/4096 = 1.2
16	65536	5/65536 = 0.076

Notes: $V_{CC} = 5$ V
Step size (resolution) is the smallest change that can be discerned by an ADC.

- In addition to resolution, 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.
- The ADC chips are either parallel or serial.
- In parallel ADC, we have 8 of more pins dedicated to bringing out the binary data, but in serial ADC we have only one pin for data out.

ADC804 chip:

- ADC804 IC is an 8-bit parallel analog-to-digital converter.
- It works with +5 volts and has a resolution of 8bits.
- In ADC804 conversion time varies depending on the clocking signals applied to the CLK R and CLK IN pins, but it cannot be faster than 110µs.
- Figure 5.5 Pin out of ADC0804 in free running mode.
- The following is the ADC0804 pin description.

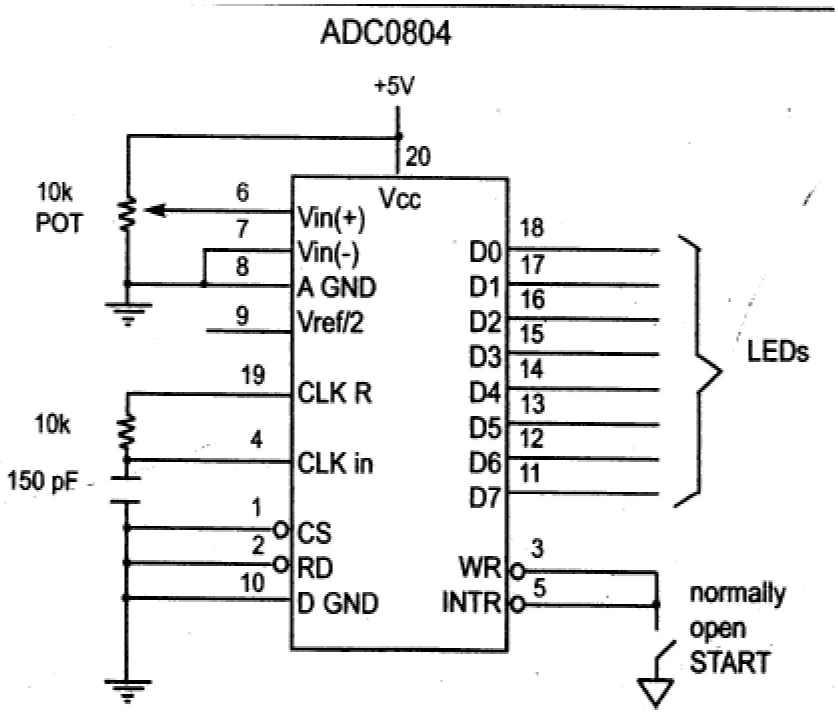


Figure 5.5 ADC0804 Chip (Testing ADC0804 in Free Running Mode)

- **CLK IN and CLK R:**
  - CLK IN is an input pin connected to an external clock source when an external clock is used for timing.
  - However, the 0804 has an internal clock generator.
  - 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 =150pF.
  - By substituting, we get f = 606 kHz and the conversion time is 110µs.
- **Vref/2: (Pin 9)**
  - 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.
  - Table 5.4 shows the Vin range for various Vref/2 inputs.

Table 5.4 Vref/2 Relation to Vin Range (ADC0804)

$V_{ref}/2$ (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

Notes: V_{CC} = 5 V

*When not connected (open), $V_{ref}/2$ is measured at 2.5 volts for V_{CC} = 5 V.

Step Size (resolution) is the smallest change that can be discerned by an ADC.

➢ **D0-D7:**

- D0-D7 are the digital data output pins.
- These are tri-state buffered and 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, which is (2 * Vref/2)/256 for ADC 0804

➢ **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.
- The reason that to have ground pin is 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:**

- This is an input signal and is active low.
- The ADC converts the analog input to its binary equivalent and holds it in an internal register.
- RD is, used to get the converted data out of the ADC0804 chip.
- Is “output enable” a high-to-low RD pulse is used to get the 8-bit converted data out of ADC804.

➢ **INTR:**

- This is an output pin and is active low.
- 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:**
  - This is an active low input
  - 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.
  - When the data conversion is complete, the INTR pin is forced low by the ADC0804.
- **CS:**
  - It is an active low input used to activate ADC804.
- **Steps to Be followed For Data Conversion:**
  - 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 5.6 shows the timing diagram for ADC process.

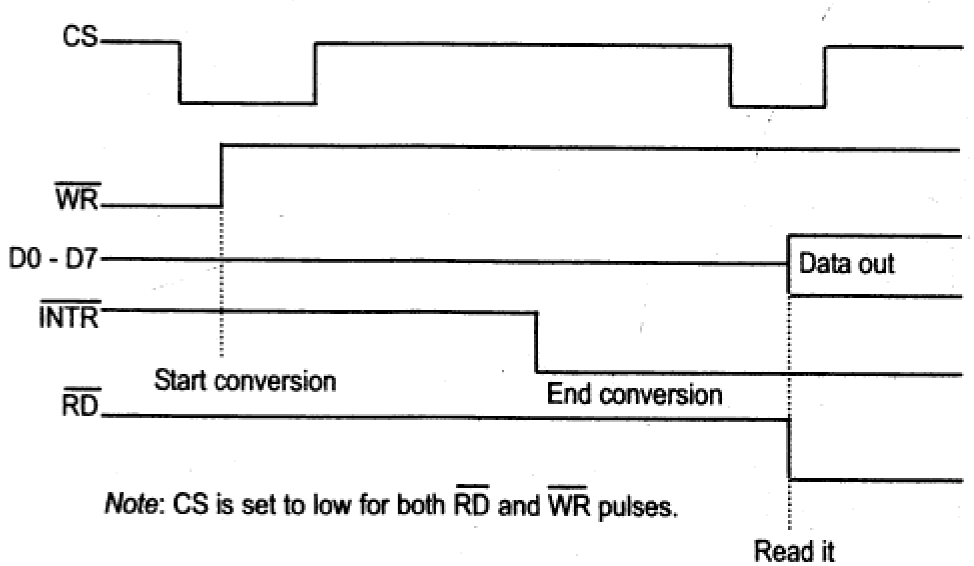


Figure 5.6 Read and Write Timing for ADC08804

<u>Clock source for ADC0804:</u>

- The speed at which an analog input is converted to the digital output depends on the speed of the CLK input.
- According to the ADC0804 datasheets, the typical operating frequency is approximately 640kHz at 5 volts.
- Figures 5.7 and 5.8 show two ways of providing clock to the ADC0804.
- In Figure 5.8, notice that the clock in for the ADC0804 is coming from the crystal of the microcontroller.
- Since this frequency is too high, we use D flip-flops (74LS74) to divide the frequency.
- A single D flip-flop divides the frequency by 2 if we connect its $\bar{Q}$ to the D input.
- For a higher-frequency crystal, you can use 4 flip-flops

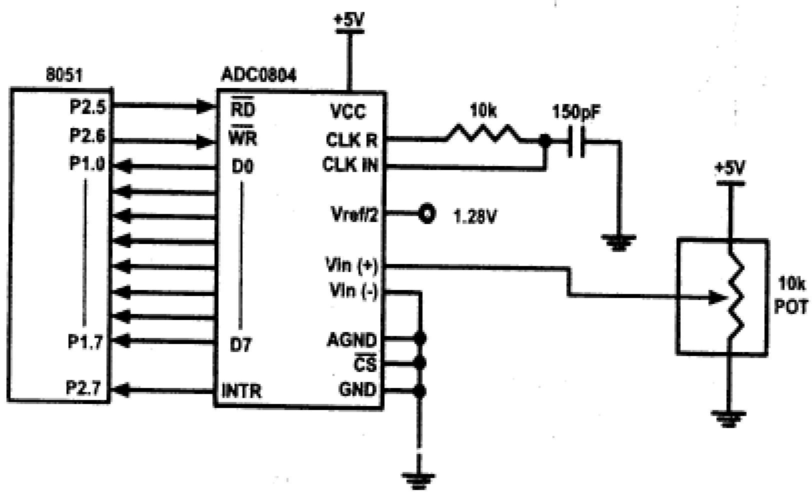


Figure 5.7 8051 Connection to ADC0804 with Self-Clocking

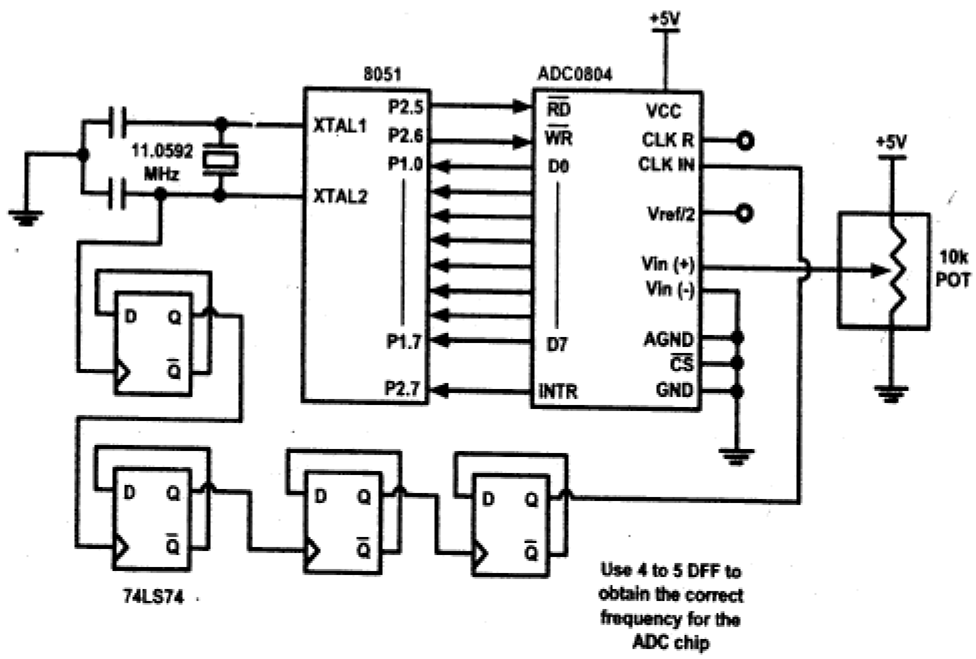


Figure 5.8 8051 Connection to ADC0804 with Clock from XTAL2 of the 8051

Example:

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
```

ADC0808:

- While the ADC0804 has only one analog input, this chip has 8 of them.
- The ADC0808/0809 chip allows us to monitor up to 8 different analog inputs using only a single chip.
- Notice that the ADC0808/0809 has an 8-bit data output just like the ADC804.
- The 8 analog input channels are multiplexed and selected according to Table 5.5 using three address pins, A, B, and C.

Table 5.5 Channel Selection in ADC0808

Selected Analog Channel	C	B	A
IN0	0	0	0
IN1	0	0	1
IN2	0	1	0
IN3	0	1	1
IN4	1	0	0
IN5	1	0	1
IN6	1	1	0
IN7	1	1	1

- In the ADC0808/0809, Vref (+) and Vref.(-) set the reference voltage.
- If Vref(-) = Gnd and Vref (+) = 5 V, the step size is 5 V/256 = 19.53 mV.
- Therefore, to get a l0 mV step size we need to set Vref (+) = 2.56 V and Vref.(-) = Gnd.
- From Figure 5.9, notice the ALE pin.
- We use A, B, and C addresses to select.IN0 - IN7, and activate ALE to latch in the address.
- SC is for start conversion.
- SC is the same as the WR pin in other ADC chips.
- EOC is for end-of-conversion, and OE is for output enable (READ).
- The EOC and OE are the same as the INTR and RD pins respectively.
- Table 5.6 shows the step size relation to the Vref voltage.
- Notice that there is no Vref/2 in the ADC0808/0809 chip.

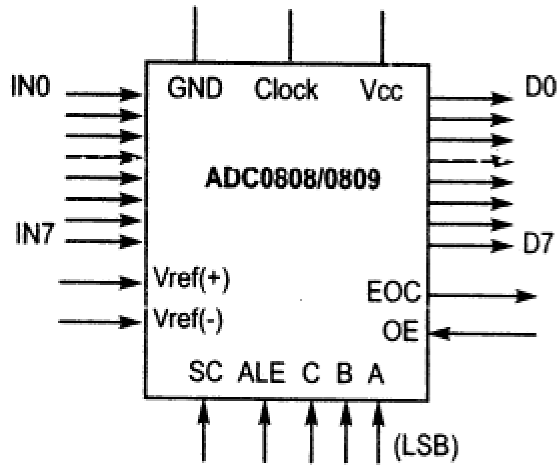


Figure 5.9 ADC0808/0809

Table 5.6 ADC0808/0809 Analog Channel Selection

V_{ref} (V)	V_{in} (V)	Step Size (mV)
not connected	0 to 5	5/256 = 19.53
4.0	0 to 4	4/255 = 15.62
3.0	0 to 3	3/256 = 11.71
2.56	0 to 2.56	2.56/256 = 10
2.0	0 to 2	2/256 = 7.81
1	0 to 1	1/256 = 3.90

Steps to program the ADC0808/0809

- The following are steps ro get data from an ADC0808/0809.
  - Select an analog channel by providing bits to A, B, and C addresses according to Table 5.6.
  - Activate the ALE (address latch enable) pin. It needs an L-to-H pulse to latch in the address.
  - Activate SC (start conversion) by an L-to-H pulse to initiate conversion.
  - Monitor EOC (end of conversion) to see whether conversion is finished. H-to- L output indicates that the data is converted and is ready to be picked up. If we do not use EOC, we can read the converted digital data after a brief time delay. The delay size depends on the speed of the external clock we connect to the CLK pin. Notice that the EOC is the same as the INTR pin in other ADC chips.
  - Activate OE (output enable) to read data out of the ADC chip. An L-to H pulse to the OE pin will bring digital data out of the chip. Also notice that the OE is "the same as the RD pin in other ADC chips.
- The speed of conversion depends on the frequency of the clock connected to the CLK pin, it cannot be faster than 100 microseconds

SENSOR INTERFACING:

LM35 Temperature sensors:

- 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 10mV for each degree of centigrade temperature.
- Table 5.7 is the selection guide for the LM35

Table 5.7 LM35 Temperature Sensor Series Selection Guide

Part	Temperature Range	Accuracy	Output Scale
LM35A	–55 C to +150 C	+1.0 C	10 mV/C
LM35	–55 C to +150 C	+1.5 C	10 mV/C
LM35CA	–40 C to +110 C	+1.0 C	10 mV/C
LM35C	–40 C to +110 C	+1.5 C	10 mV/C
LM35D	0 C to +100 C	+2.0 C	10 mV/C

Note: Temperature range is in degrees Celsius.

- The sensors of the LM34 series are precision integrated-circuit temperature sensors whose output voltage is linearly proportional to the Fahrenheit temperature.
- It also internally calibrated.
- It outputs 10mV for each degree Fahrenheit temperature.

Signal Conditioning and Interfacing the LM35 to the 8051

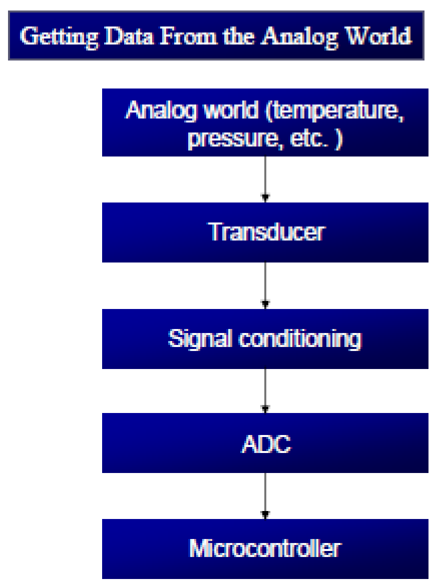


Figure 5.10 Getting Data from Analog World

- The above figure 5.10 shows the steps involved in acquiring data from analog world.
- 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 l0 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 Vout 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. Refer the table 5.8

Table 5.8 Temperature vs. Vout for ADC0848

Temp. (C)	V_{in} (mV)	V_{out} (D7 - D0)
0	0	0000 0000
1	10	0000 0001
2	20	0000 0010
3	30	0000 0011
10	100	0000 1010
30	300	0001 1110

- Figure 5.11 shows the connection of a temperature sensor to the ADC0848.
- The LM336-2.5 zener diode to fix the voltage across the 10K pot at 2.5V.
- The use of the LM336-2.5 should overcome any fluctuations in the power supply.

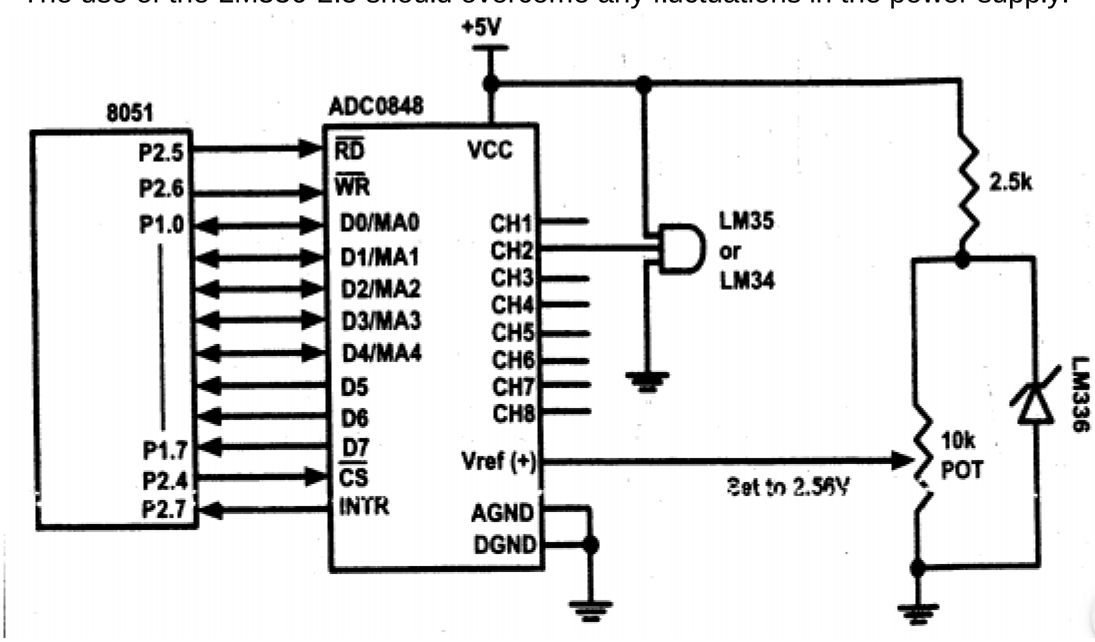


Figure 5.11 8051 Connection to ADC0848 and Temperature sensor

Program:

```
        RD BIT P2.5             ;RD
        WR BIT P2.6             ;WR
        INTR BIT P2.7           ; END OF CONVERSION
        MYDATA EQU P1           ; P1.0-P1.7 = D0-D7 OF THE ADC0848
        MOV P1,#0FFH            ;make P1 = input
        SETB INTR
BACK:   CLR WR                  ;WR = 0
        SETB WR                 ;WR = 1 L-to-H to start conversion
HERE:   JB INTR,HERE            ;wait for end of conversion
        CLR RD                  ;conversion finished, enable RD
        MOV A,MYDATA            ;read the data
        ACALL CONVERSION        ;hex-to-ASCII conversion
        ACALL DATA_DISPLAY      ;display the data
        SETB RD                 ;make RD=1 for next round
        SJMP BACK

CONVERSION:
        MOV B,#10
        DIV AB
        MOV R7,B
        MOV B,#10
        DIV AB
        MOV R6,B
        MOV R5,A
        RET

DATA_DISPLAY:
        MOV P0,R7
        ACALL DELAY
        MOV P0,R6
        ACALL DELAY
        MOV P0,R5
        ACALL DELAY
        RET
```

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.
- Therefore, an 8-input DAC such as the DAC0808 provides 256 discrete voltage (or current) levels of output.
- Similarly, the 12-bit DAC provides 4096 discrete voltage levels.
- There also 16-bit DACs, but they are more expensive.

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 provided by the I_{out} pin 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 DAC0808is as shown in the below figure 5.12.
- Now assuming that I_{ref} = 2mA, if all the inputs to the DAC are high, the maximum output current is 1.99mA.

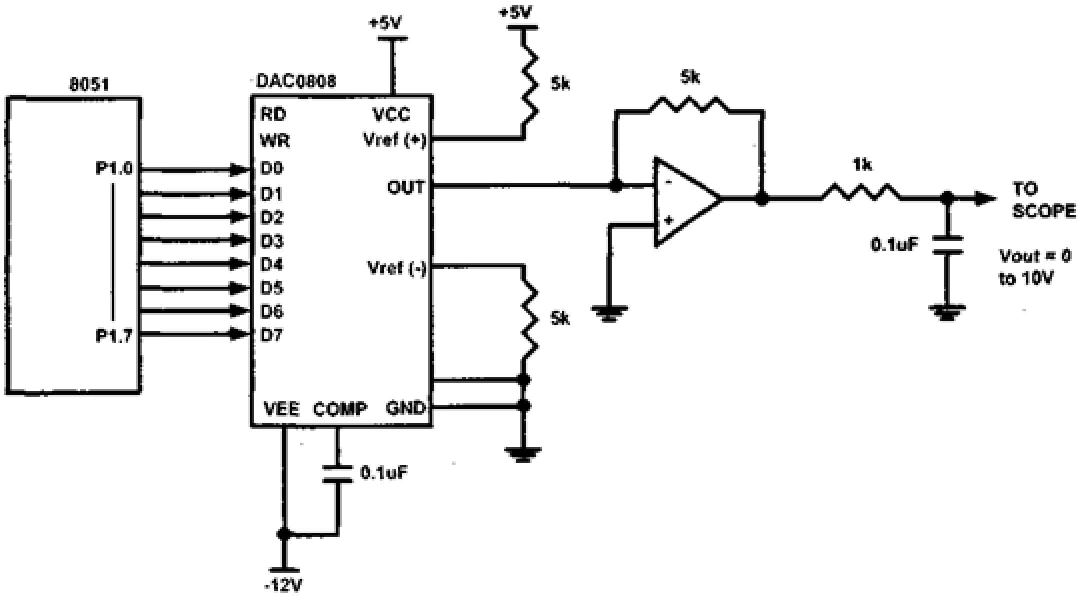


Figure 5.12 8051 Connection to DAC808

Example 1:

Assuming that R=5K and Iref=2mA, calculate Vout for the following binary inputs:

(a) 10011001B

(b) 11001000B

Solution:

(a) Iout = 2mA(153/256) = 1.195mA and Vout = 1.195mA * 5K =5.975V

(b) Iout = 2mA(200/256) = 1.562mA and Vout = 1.562mA * 5K =7.8125V

Converting I_{out} to voltage in DAC0808:

- Ideally we connect the output pin I_{out}, to a resistor, convert this current to voltage, and monitor the output on the scope.
- In real life, however, this can cause inaccuracy since the input resistance of the load where it is connected will also affect the output voltage.
- For this reason, the I_{ref} current output is isolated by connecting it to an op-amp such as the 741 with R_f = 5K ohms for the feedback resistor.
- Assuming that R= 5K ohms, by changing the binary input, the output voltage changes as shown in Example 2.

Example 2:

Inorder to generate a stair-step ramp, set up the circuit in figure 5.12 and connect the output to an oscilloscope. Then write a program to send data to the DAC to generate a stair-step ramp.

Solution:

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

Generating a sine wave

- To generate a sine wave, we first need a table whose values represent the magnitude of the sine of angles between 0 and 360 degrees.
- The values for the sine function vary from -1.0 to +1.0 for 0- to 360-degree angles.
- Therefore, the table values are integer numbers representing the voltage magnitude for the sine of theta.
- This method ensures that only integer numbers are output to the DAC by the 805l microcontroller.
- Table 5.9 shows the angles, the sine values, the voltage magnitudes, and the integer values representing the voltage magnitude for each angle (with 30-degree increments).
- To generate Table 5.9,we assumed the full-scale voltage of 10 V for DAC output (as designed in Example 4 Figure).
- Full-scale output of the DAC is achieved when all the data inputs of the DAc are high.
- Therefore, to achieve the full-scale 10 V output, we use the following equation

$$\mathbf{Vout = 5V(1+\sin\theta)}$$

- Vout of DAC for various angles is calculated and shown in Table 5.9. See Example 3 for verification of the calculations

Table 5.9 Angle Vs Voltage Magnitude for Sine Wave

Angle θ (degrees)	Sin θ	V_{out} (Voltage Magnitude) 5 V + (5 V × sin θ)	Values Sent to DAC (decimal) (Voltage Mag. × 25.6)
0	0	5	128
30	0.5	7.5	192
60	0.866	9.33	238
90	1.0	10	255
120	0.866	9.33	238
150	0.5	7.5	192
180	0	5	128
210	−0.5	2.5	64
240	−0.866	0.669	17
270	−1.0	0	0
300	−0.866	0.669	17
330	−0.5	2.5	64
360	0	5	128

Example 3:

Verify the values given for the following angles: (a) 30º (b) 60º

Solution:

(a) Vout = 5V+(5V * sin30) =7.5V
 DAC input values = 7.5V * 25.6 = 192 (Decimal)

(b) Vout = 5V+(5V * sin60) =9.33V
 DAC input values = 9.33V * 25.6 = 238 (Decimal)

➢ To find the values sent to the DAC for various angles, we simply multiply Vout voltage by 25.6 because there are 256 steps and full scale Vout is 10 volts.
  256 steps/10V = 25.6 steps per volt
➢ The following examples 9, 10 and 11 will show the generation of waveforms using DAC0808.

Example 4:

Write an ALP to generate a sine waveform.

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

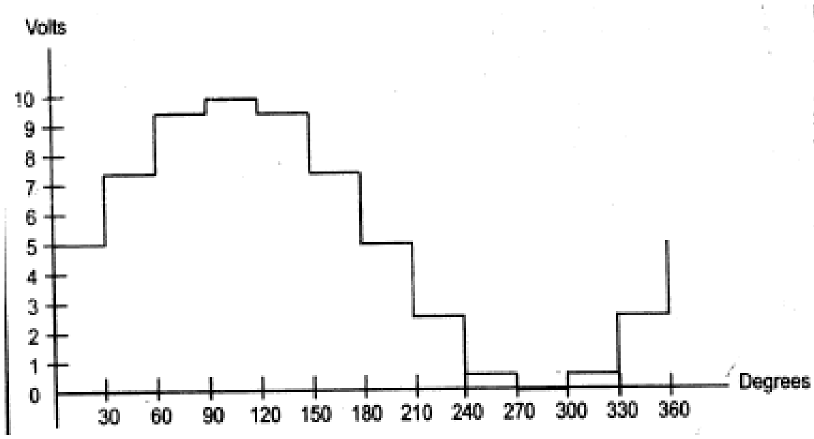
Solution:

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 sine wave 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.



Example 5:

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
```

DC MOTOR INTERFACING:

- DC motor is a device that translates electrical pulses into mechanical movement.
- The DC motor has + and – leads
- Connecting them to a DC voltage source moves the motor in one direction and by reversing the polarity, the DC motor will move in opposite direction.

Unidirectional Control:

- The following figure 5.13 shows the DC motor rotation for clockwise (CW) and counterclockwise (CCW) rotations.

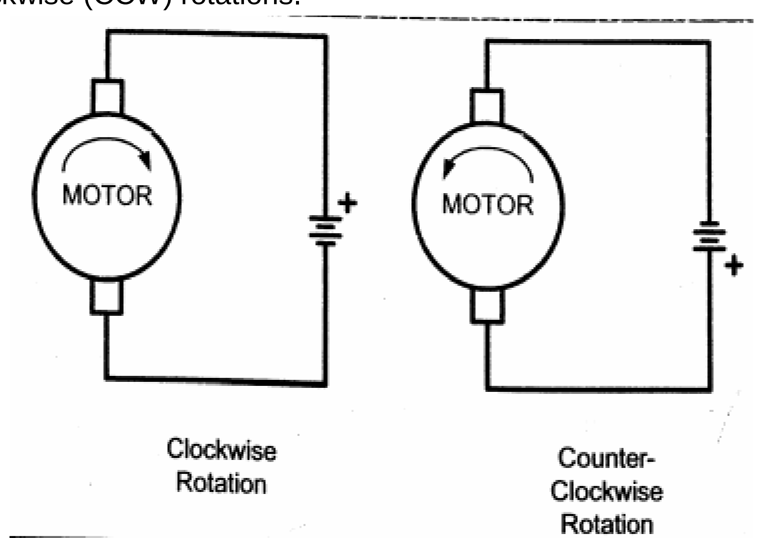


Figure 5.13 DC Motor Rotation (Permanent Magnet Field)

Bidirectional Control:

- With the help of relays or some specially designed chips we can change the direction of the DC motor rotation.
- Figure 5.14 through 5.17 shows the basic concepts of H-Bridge control of DC motors.

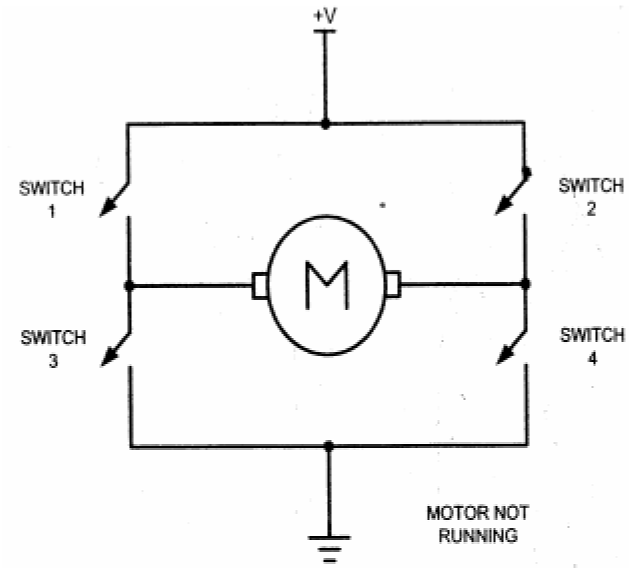


Figure 5.14 H-Bridge Motor Configuration

- Figure 5.2 shows the connection of an H-Bridge using simple switches.
- All the switches are open, which does not allow the motor to turn.

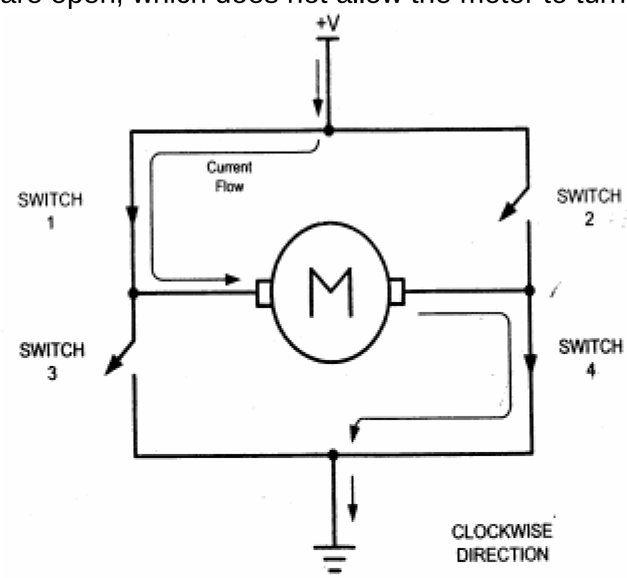


Figure 5.15 H-Bridge Motor Clockwise Configuration

- Figure 5.3 shows the switch configuration for turning the motor in one direction.
- When switches 1 and 4 are closed, current is allowed to pass through the motor.

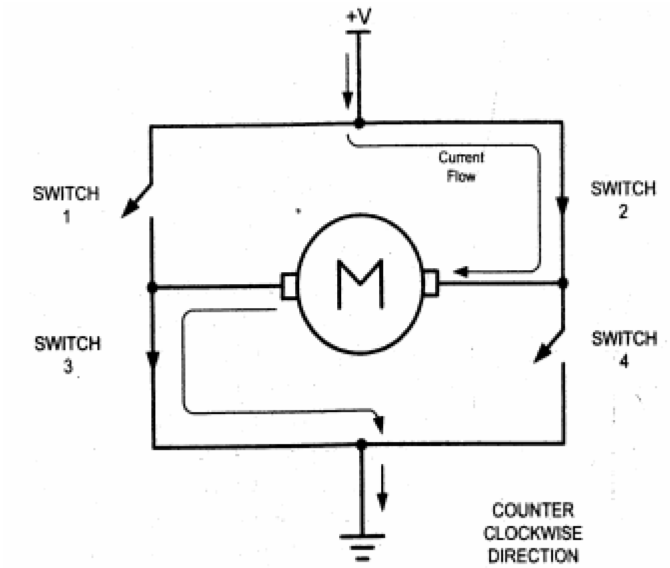


Figure 5.16 H-Bridge Motor Counterclockwise Configuration

- Figure 5.3 shows the switch configuration for turning the motor in the opposite direction from the configuration of Figure 5.3
- When switches 2 and 3 are closed, current is allowed to pass through the motor.

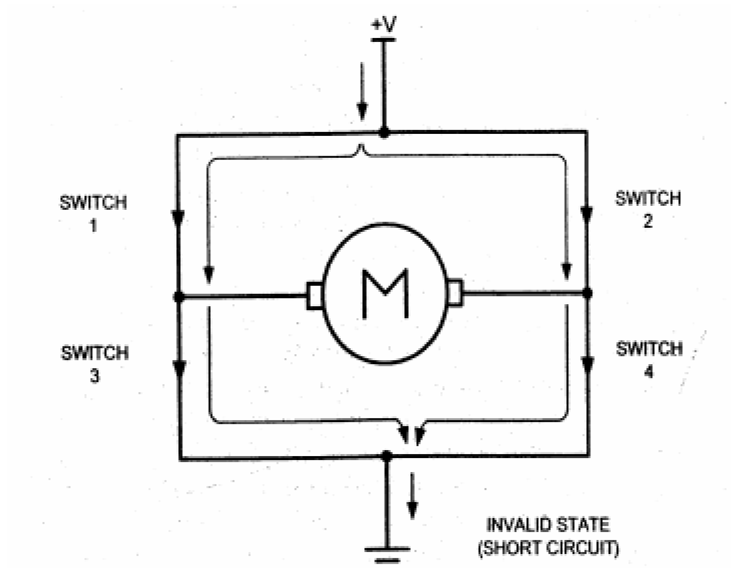


Figure 5.17 H-Bridge in an invalid configuration.

- Figure 5.4 shows an invalid configuration.
- Current flows directly to ground, creating a short circuit.
- The same effect occurs when switches 1 and 3 are closed or switches 2 and 4 are closed.
- Table 5.10 shows some of the logic configurations for the H-Bridge design.

Table 5.10 H-Bridge Logic Configurations

Motor Operation	**SW1**	**SW2**	**SW3**	**SW4**
OFF	Open	Open	Open	Open
Clockwise	Closed	Open	Open	Closed
Counter Clockwise	Open	Closed	Closed	Open
Invalid	Closed	Closed	Closed	Closed

- H-Bridge control can be created using relays, transistors, or a single IC Solution such as the L293.
- When using relays and transistors, must ensure that invalid configuration do not occur.
- **<u>Example:</u>**

  A switch is connected to pin P2.7. Write a program to monitor the status of SW and perform the following:

  (a) If SW=0, the DC motor moves clockwise

  (b) If SW=1, the DC motor moves counterclockwise

  Solution:

```
            ORG 0H
MAIN:       CLR P1.0              ; Switch 1
            CLR P1.1              ; Switch 2
            CLR P1.2              ; Switch 3
            CLR P1.3              ; Switch 4
            SETB P2.7
MONITOR:    JNB P2.7, CLOCKWISE
            SETB P1.0             ; Switch 1
            CLR P1.1              ; Switch 2
            CLR P1.2              ; Switch 3
            SETB P1.3             ; Switch 4
            SJMP MONITOR
CLOCKWISE:  CLR P1.0              ; Switch 1
            SETB P1.1             ; Switch 2
            SETB P1.2             ; Switch 3
            CLR P1.3              ; Switch 4
            SJMP MONITOR
            END
```

Motor Control Using L293

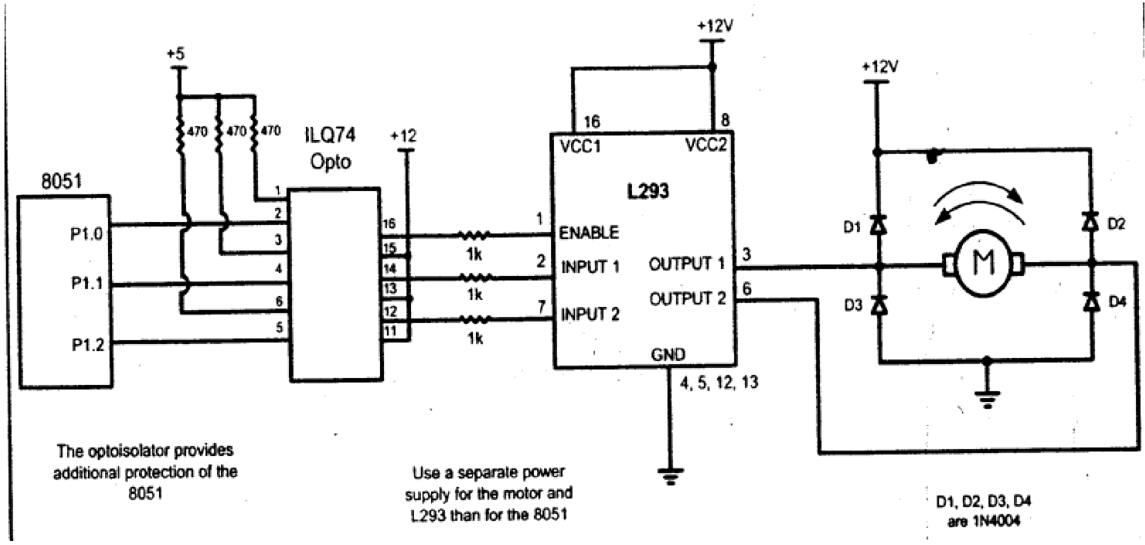


Figure 5.18 Bidirectional Motor Control Using L293 Chip

- Figure 5.18 shows the connection of L293 to an 8051.
- **Example:**

  A switch is connected to pin P2.7. Write a program to monitor the status of SW and perform the following:

  (a) If SW=0, the DC motor moves clockwise

  (b) If SW=1, the DC motor moves counterclockwise

  Solution:

```
ORG 0H
    MAIN:       CLR P1.0
                CLR P1.1
                CLR P1.2
                SETB P2.7
    MONITOR:    SETB P1.0               ; Enable the Chip
                JNB P2.7, CLOCKWISE
                CLR P1.1                ; Turn Motor counterclockwise
                SETB P1.2
                SJMP MONITOR
    CLOCKWISE:  SETB P1.1
                CLR P1.2                ; Turn Motor clockwise
                SJMP MONITOR
                END
```

PWM:

- The speed of the motor depends on three factors
  - Load
  - Voltage
  - Current
- For a given fixed load we can maintain a steady speed by using a method called Pulse Width Modulation(PWM)

- By changing (modulating) the width of the pulse applied to the DC motor we can increase or decrease the amount of power provided to the motor, thereby increasing or decreasing the motor speed.
- Notice that although the voltage has a fixed amplitude, it has a variable duty cycle
- That means the wider the pulse, the higher the speed.
- PWM is do widely used in DC motor control that some microcontrollers come with the PWM circuitry embedded in the chip.

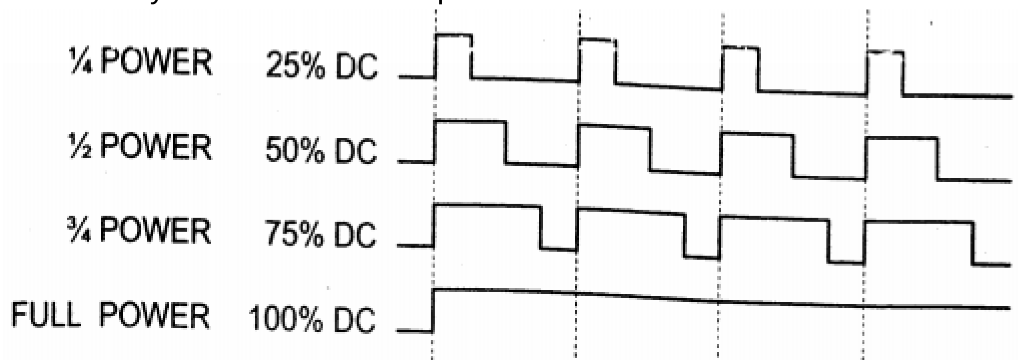


Figure 5.19 Pulse Width Modulation Comparison

<u>Optoisolator:</u>

- An **optoisolator** (also known as optical coupler, optocoupler and **opto-isolator**) is a semiconductor device that uses a short optical transmission path to transfer an electrical signal between circuits or elements of a circuit, while keeping them electrically isolated from each other.
- Advantage: Their high electrical isolation between the input and output terminals allowing relatively small digital signals to **control** much large AC voltages, currents and power.

Reference:

- Muhammed Ali Mazidi, Janice Gillispie Mazidi and Rolin D.McKinlay, “The 8051 Microcontroller and Embedded Systems: Using Assembly and C”