8051 Microcontroller: Interfacing

Lcd interfacing

LCD is finding the widespread use in the market because of 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

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 and R/W=1 for read
6	E	I/O	Enable
7	DB0	I/O	8-bit data bus
8	DB1	I/O	8-bit data bus
9	DB2	I/O	8-bit data bus
10	DB3	I/O	8-bit data bus
11	DB4	I/O	8-bit data bus
12	DB5	I/O	8-bit data bus
13	DB6	I/O	8-bit data bus
14	DB7	I/O	8-bit data bus

- V_{CC}, V_{SS}, V_{EE}

V_{CC} and V_{SS} provides +5V and ground while VEE is used for controlling LCD contrast.

- RS, register select

There are two important register inside the LCD. The RS pin is used for their selection. If RS=0, the instruction command code register is selected, allowing the user to send a command such as clear display, cursor position, etc. If RS=1 the data register is selected, allowing the user to send data to be displayed on the LCD.

- R/W read/write

Read/ Write allows the user to write information to the LCD or read information from it. R/W=1 while reading and R/W=0 while writing.

- E, Enable

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.

- D0-D7

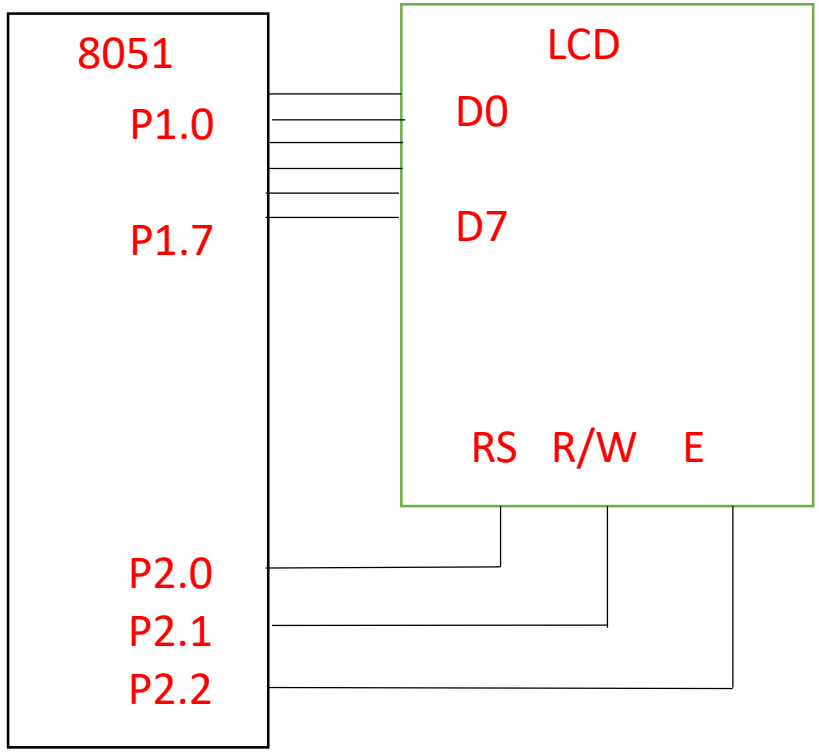
The 8-bit data pins, D0-D7 are used to send information to the LCD or read the contents of the LCD internal registers. To display letters and numbers, we send ASCII codes for the letters A-Z, a-z and numbers 0-9 to these pins while making RS=1. D7 is the busy flag, while D7=1, LCD is busy taking care of internal operations. When D7=0, the LCD is ready to receive new information. It is recommended to check the busy flag before writing any data to the LCD.

Table I: LCD Command Codes

Code (Hex)	Command to LCD Instruction Register
1	Clear display Screen
2	Return home
4	Decrement cursor (Shift cursor to left)
6	Increment cursor (Shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
IC	Shift the entire display to the right
80	Force cursor to beginning of 1st line
C0	Force cursor to beginning of 2nd line
38	2 lines and 5X7 matrix

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

```
 # include <reg.51.h>
Sfr ldata=0X90;                    // P1 = LCD data pins
sbit rs = P2^0;
sbit rw = P2^1;
sbit en = P2^2;
void main()
{
lcdcmd(0X38);   // init. LCD 2 lines, 5X7 matrix
MSDelay(250);
lcdcmd(0X0E);  // display on, cursor on
MSDelay(250);
```



```
  lcdcmd(0X01);      // clear LCD
  MSDelay(250);
  lcdcmd(0X06);     // shift cursor right
  MSDelay(250);
  lcdcmd(0X86);     // cursor at line 1, position 6
  MSDelay(250);
  lcddata('M');       // display letter M
  MSDelay(250);
  lcddata('D');        // display letter D
  MSDelay(250);
  lcddata('E');        // display letter E
}
```

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

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

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

- Write an 8051 C program to send letters ‘M’, ‘D’ and ‘E’ to the LCD using the busy flag method.

```
#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(‘M’);
 lcddata(‘D’);
lcddata(‘E’);
}
```

```
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;
}
```

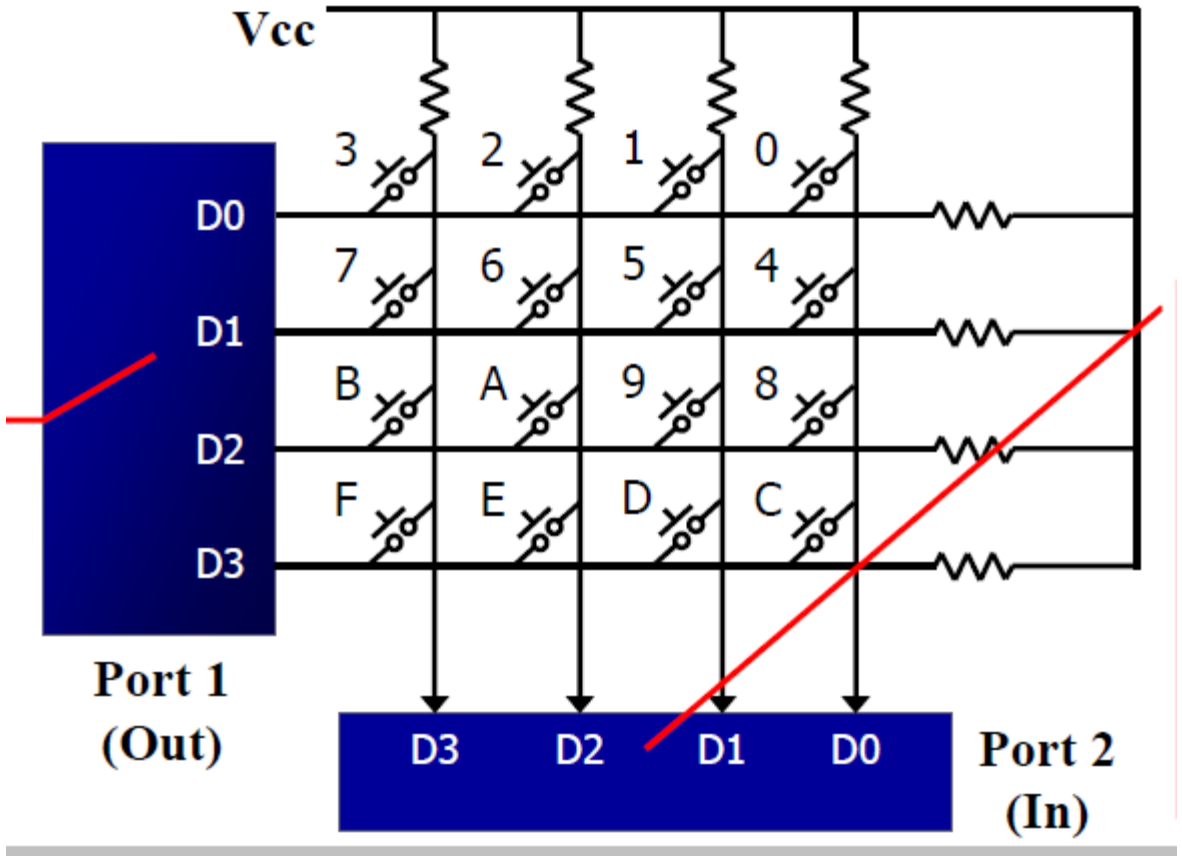
```
void lcdready()
 {
    busy = 1; //make the busy pin at input
    rs = 0;
    rw = 1;
    while(busy==1) //wait here for busy flag
     {
         en = 0; //strobe the enable pin
         MSDelay(1);
         en = 1;
     }
  void MSDelay(unsigned int itime)
{
   unsigned int i, j;
   for(i=0;i<itime; 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.
- In IBM PC keyboards, a single microcontroller takes care of hardware and software interfacing of the keyboard. A program stored in the EPROM scan the keys continuously, identify which one has been activated and present it to the motherboard.

Scanning and identifying the key

- Figure 1 shows a 4X4 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.
- 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

- Problem: Identify the row and column of the pressed key for each of the following.

  (a) D3-D0=1110 for the row, D3-D0=1011 for the column

  (b) D3-D0=1101 for the row, D3-D0=0111 for the column

Solution:

(a) The row belongs to D0 and the column belongs to D2, therefore key 2 is pressed.
(b) The row belongs to D1 and the column belongs to D3, therefore key 7 is pressed.

- Program: Write an 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 four major 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.

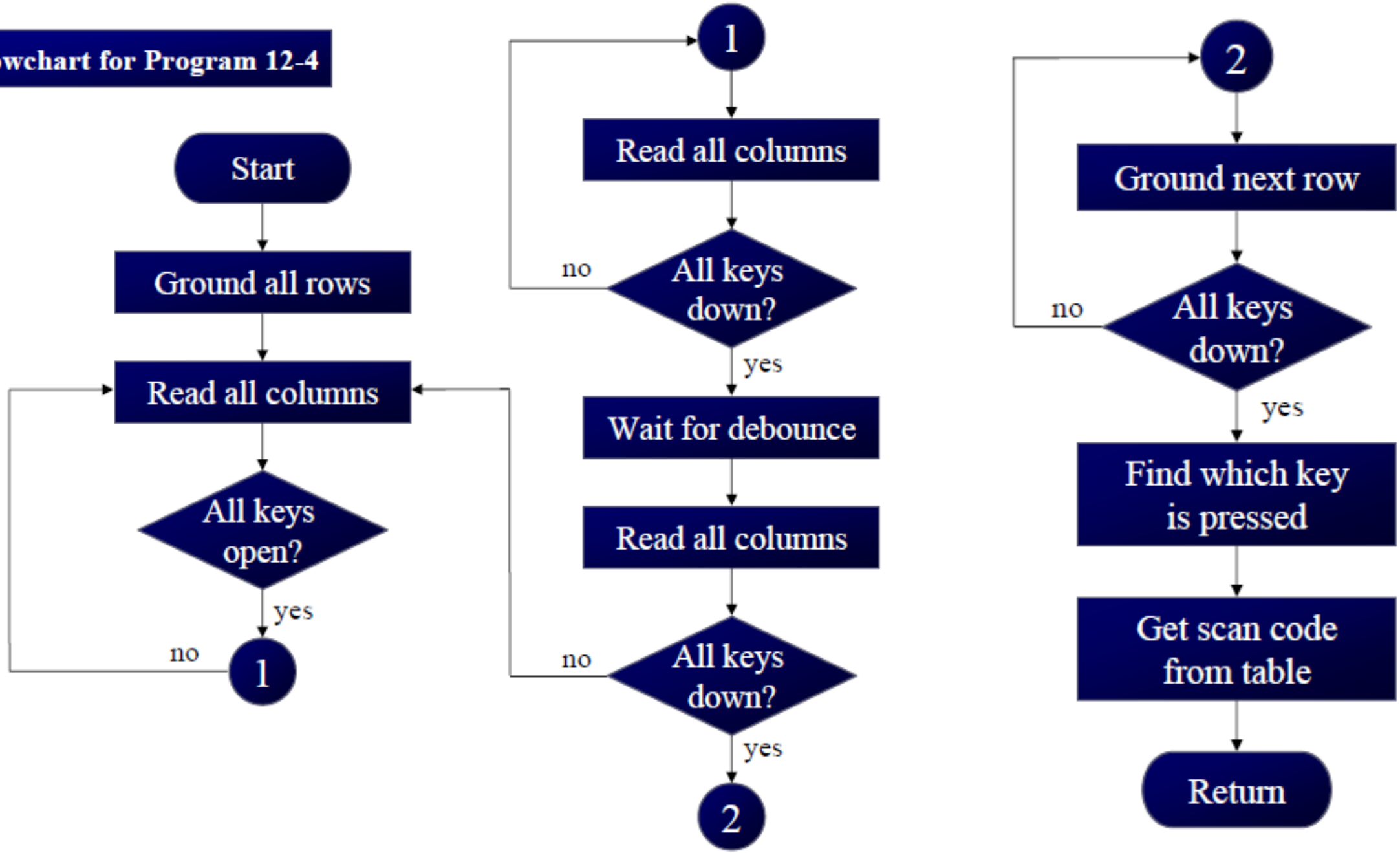
- Remember that the output latches connected to rows still have their initial zeros (provided in stage 1), making them grounded.
- After the key press detection, it waits 20 ms for the bounce and then scans the columns again

(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.

Flowchart for Program 12-4



- Assembly Language 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.

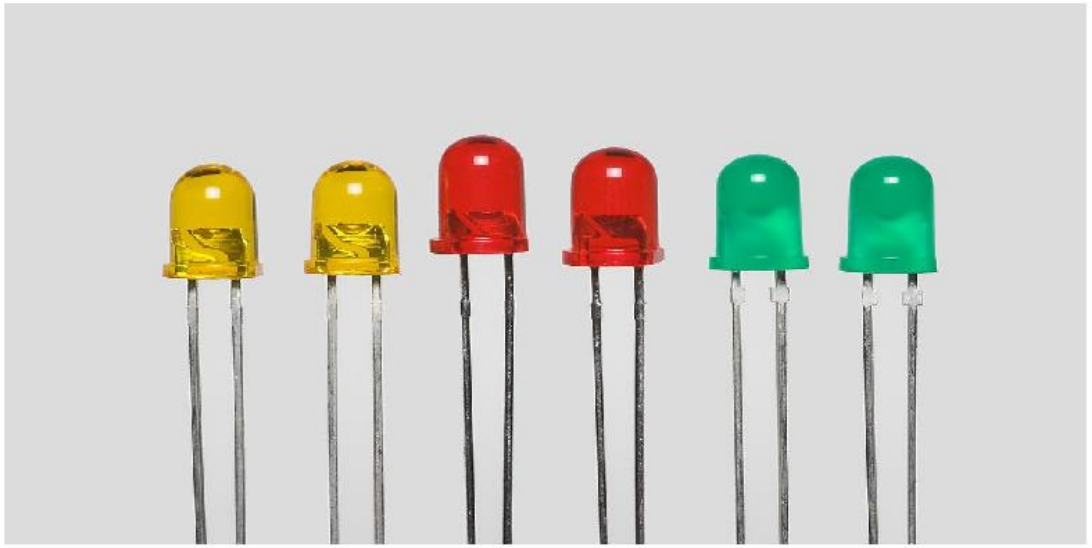
```
        MOV P2,#0FFH                    ; make P2 an input port
K1:     MOV P1,#0                       ; ground all rows at once
        MOV A,P2                        ; read all col ;(ensure keys open)
        ANL A,00001111B                 ; masked unused bits
        CJNE A,#00001111B,K1            ; 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
```

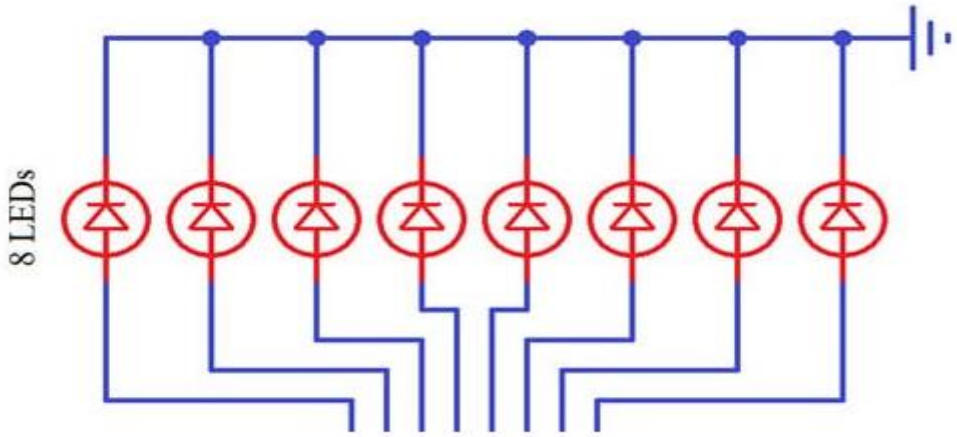
```
OVER1:    MOV P1, #11111110B          ;ground row 0
          MOV A,P2                    ;read all columns
          ANL A,#00001111B            ;mask unused bits
          CJNE A,#00001111B,ROW_0     ;key row 0, find col.
          MOV P1,#11111101B           ;ground row 1
          MOV A,P2                    ;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
```

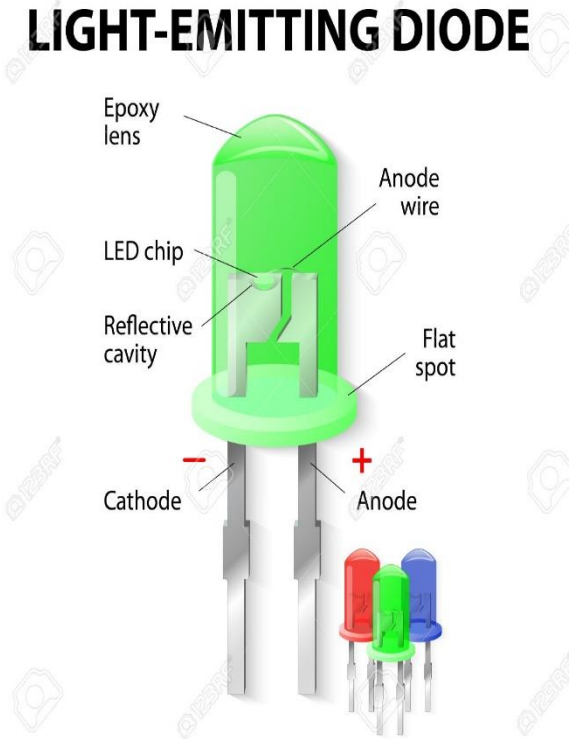
```
ROW_0:    MOV DPTR,#KCODE0      ;set DPTR=start of row 0
          SJMP FIND             ;find col. Key belongs to
ROW_1:    MOV DPTR,#KCODE1      ;set DPTR=start of row
          SJMP FIND             ;find col. Key belongs to
ROW_2:    MOV DPTR,#KCODE2      ;set DPTR=start of row 2
          SJMP FIND             ;find col. Key belongs to
ROW_3:    MOV DPTR,#KCODE3      ;set DPTR=start of row 3
          SJMP FIND
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
```

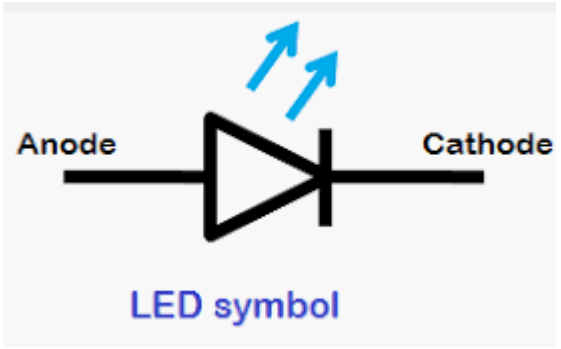
```
ASCII          LOOK-UP TABLE FOR EACH ROW
               ORG 300H
KCODE0:          DB ‘0’,’1’,’2’,’3’                ;ROW 0
KCODE1:          DB ‘4’,’5’,’6’,’7’                ;ROW 1
KCODE2:          DB ‘8’,’9’,’A’,’B’                ;ROW 2
KCODE3:          DB ‘C’,’D’,’E’,’F’                ;ROW 3
               END
```

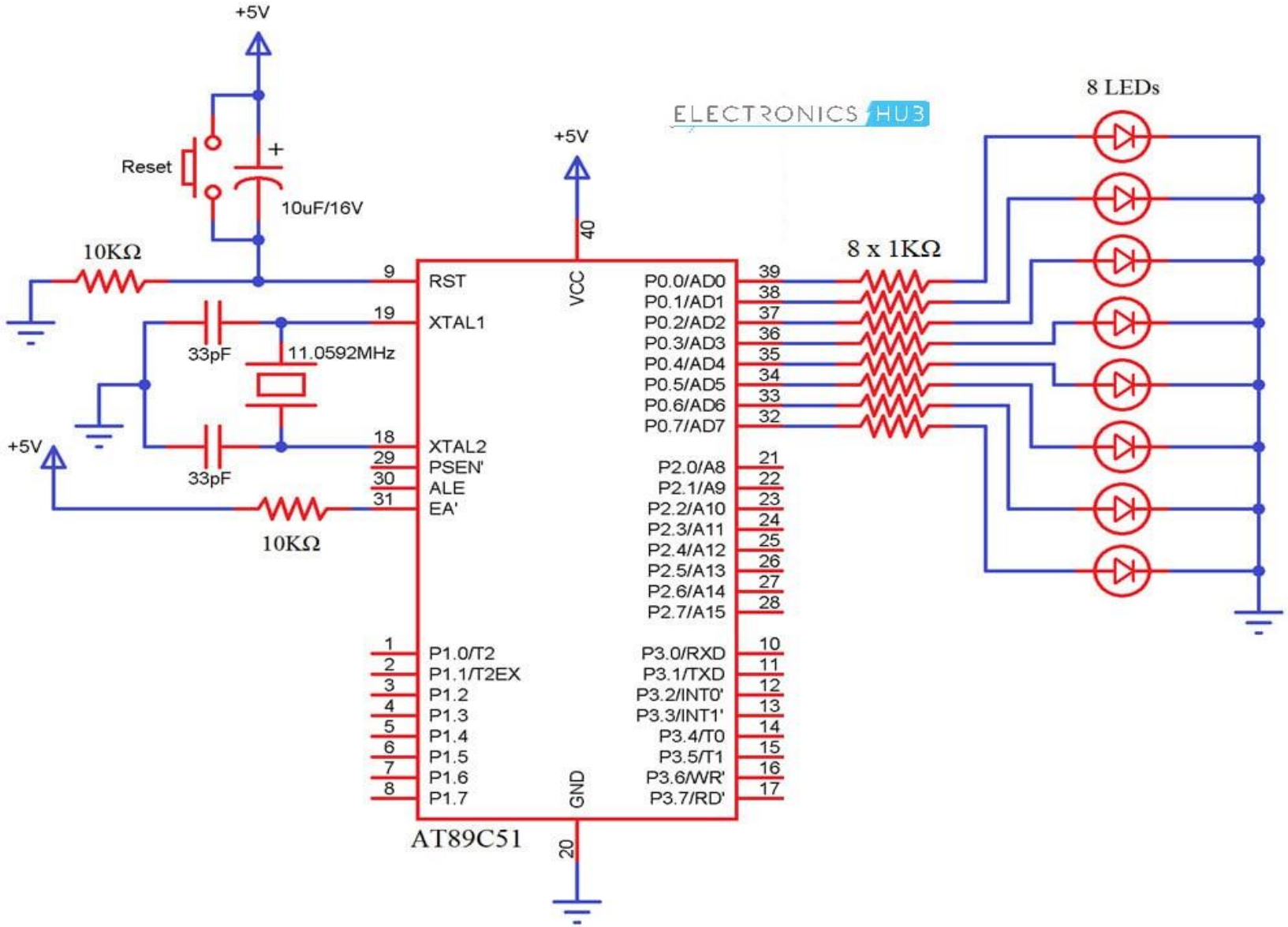
Interfacing Example (LED Interfacing)







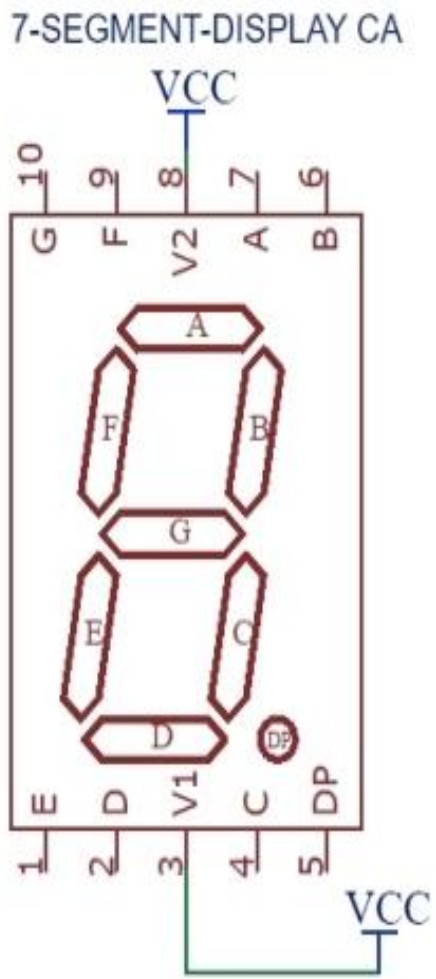


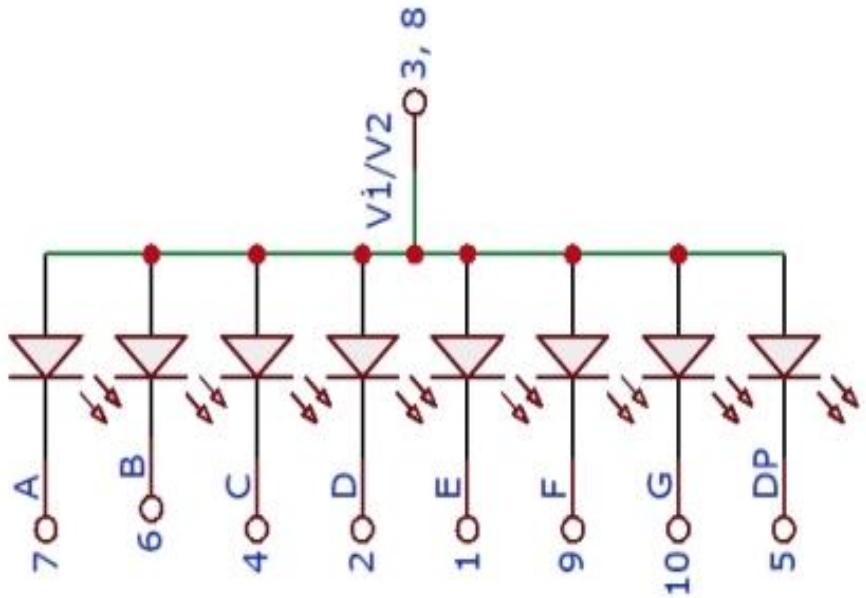

+5V
Reset
10uF/16V
10KΩ
9
RST
19
XTAL1
33pF
11.0592MHz
18
XTAL2
33pF
+5V
29
PSEN'
30
ALE
31
EA'
10KΩ
+5V
40
VCC
1
P1.0/T2
2
P1.1/T2EX
3
P1.2
4
P1.3
5
P1.4
6
P1.5
7
P1.6
8
P1.7
GND
20
AT89C51
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
39
38
37
36
35
34
33
32
8 x 1KΩ
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
21
22
23
24
25
26
27
28
P3.0/RXD
P3.1/TXD
P3.2/INT0'
P3.3/INT1'
P3.4/T0
P3.5/T1
P3.6/WR'
P3.7/RD'
10
11
12
13
14
15
16
17
8 LEDs
ELECTRONICS HUB

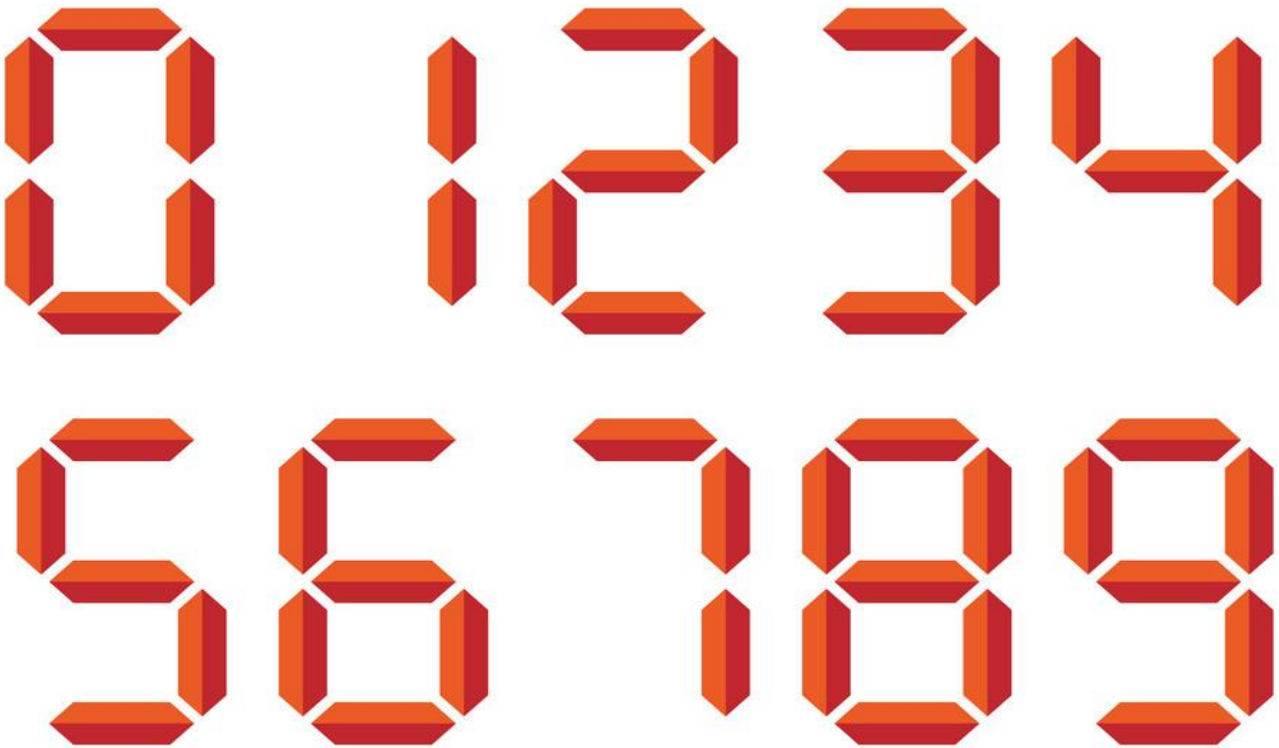
Program for LED Interfacing

```
# include<reg51.h>
sfr P0
void delay();
void main()
{
while(1)
{
P0=0X01;
delay();
P0=0X02;
delay();
P0=0X04;
delay();
P0=0X08;
delay();
P0=0X10;
delay();
P0=0X20;
delay();
P0=0X40;
delay();
P0=0X80;
delay();
}}
void delay()
{
int i;
for(i=0;i<=30000;i++);
}
```

7 segment display







digit	HEX Value	h g f e d c b a
0	BF	1 0 1 1 1 1 1 1
1	86	1 0 0 0 0 1 1 0
2	DB	1 1 0 1 1 0 1 1

Program

```
#include<reg51.h>
char arr [10] = {0xbf, 0x86, 0xdb, 0xcf, 0xe6, 0xed, 0xfd, 0x87, 0xff, 0xef};
void delay ();
void main(void)
{
   int a;
   for(a=0;a<10;a++)
{
```

```
        P0=arr [a];
        MSDelay(250)
      }
}
```

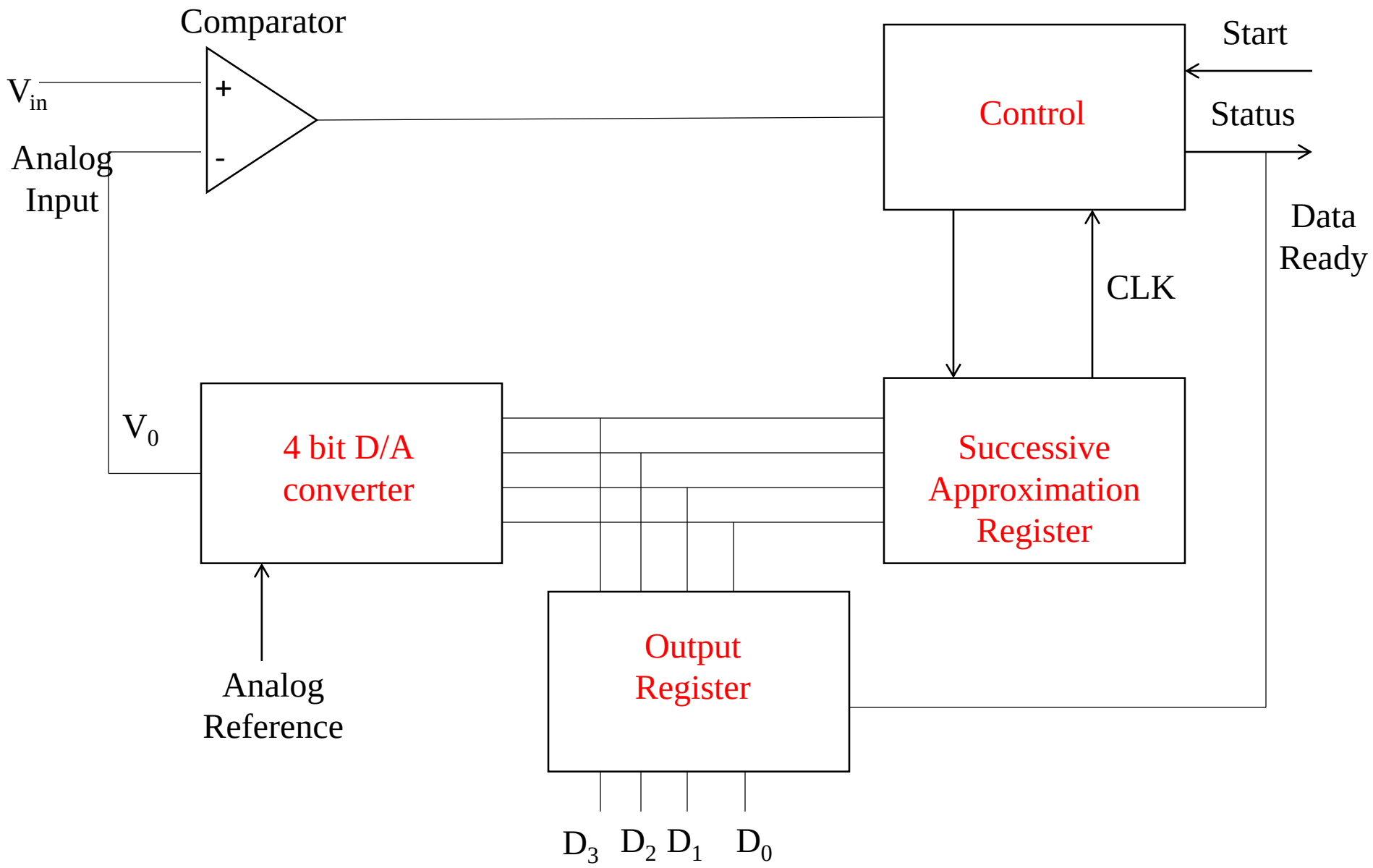
```
void MSDelay (unsigned int itime)
{
unsigned int i, j;
  for (i=0; i<itime; i++)
    for (j=0;j<1275; j++)
}
```

Analog to digital converter

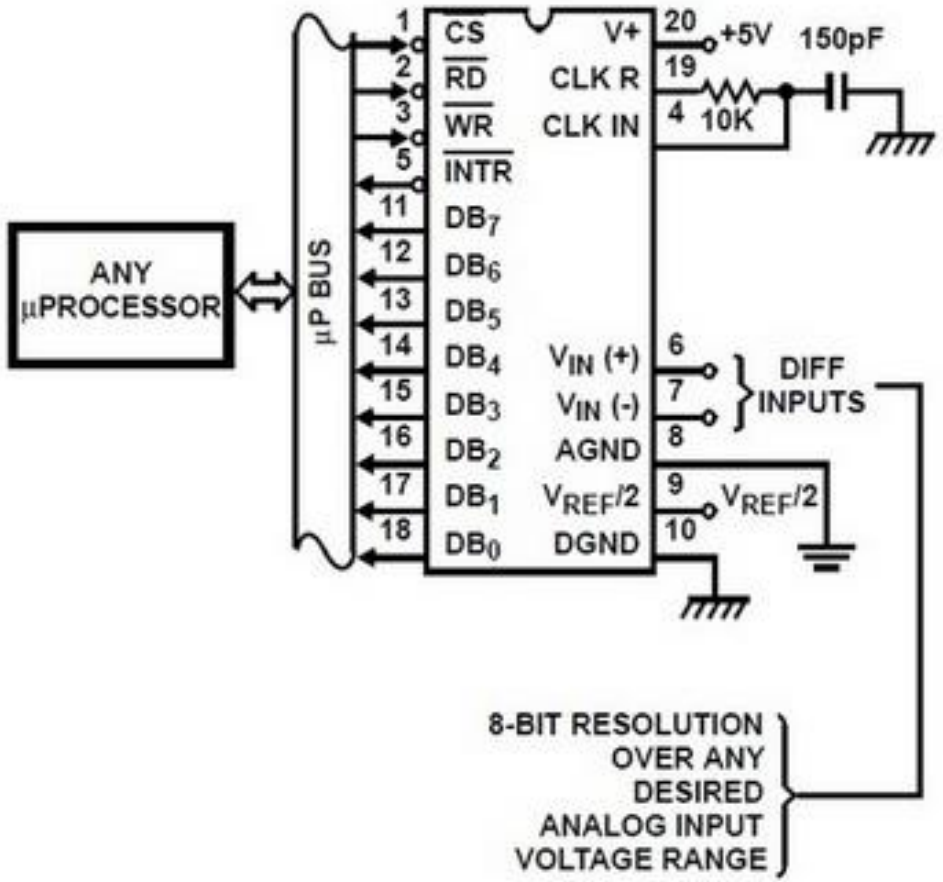
- ADC are most widely used devices for data acquisition.
- Digital computers use binary (discrete) values, but in the physical world everything is analog (continuous).
- Humidity, temperature, pressure are example of physical quantities that we deal in our day today life. These are measured using sensors and the output of sensor is in the form of voltage or current.
- Therefore we need an analog to digital converter to translate the analog signals to digital so that microcontroller read and process them.

Successive Approximation ADC

- ADC0804 IC is an 8 bit successive approximation analogue to digital converter from National semiconductors.
- It includes three major elements: D/A converter, successive approximation register and the comparator.
- The conversion process involves comparing the output of the D/A converter V_o with the analog input signal V_{in}.
- The digital input to the DAC is generated using the successive-approximation method.
- When the DAC output matches the analog signal, the input to the DAC is the equivalent digital signal.

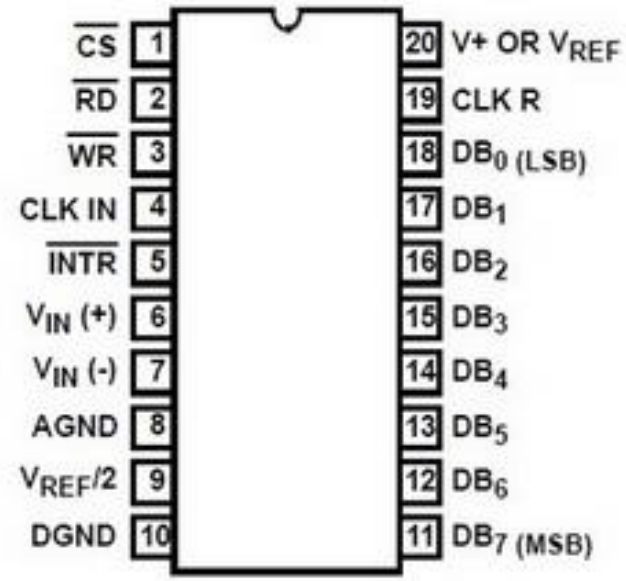

Comparator
V_{in}
Analog Input
+
-
Control
Start
Status
Data Ready
CLK
V_0
4 bit D/A converter
Successive Approximation Register
Analog Reference
Output Register
D_3 D_2 D_1 D_0

- **4-bit A/D converter**
- Initially Bit D3 is turned on first.
- The output of the DAC is compared with an analog signal.
- If the comparator changes the state, indicating that the output generated by D3 is larger than the analog signal, bit D3 is turned off and D2 is turned on.
- The process continues until the input reaches bit D_0.



Pinout

ADC0803, ADC0804
(PDIP)
TOP VIEW



ADC0804 Chip

ADC0804 pin out description

- CS- Chip select is an active low input used to activate the ADC0804 chip. To access the ADC0804, this pin must be low.
- RD (Read)- 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. When CS=0, if a high-to-low pulse is applied to the RD pin, the 8-bit digital output shows up at the D0-D7 data pins. The RD pin is also referred to as output enable.
- WR (Write)- This is an active low input used to inform the ADC0804 to start the conversion process. When the data conversion is complete the INTR pin is forced low by the ADC0804.

- 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. ADC0804 has a internal clock generator, to use the internal clock CLK IN and CLK R pins are connected to a capacitor and a resistor and the clock frequency is:

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

- INTR- This is an output pin and is active low. It is normally high pin and when the conversion is finished, it goes low to signal the CPU that the conversion that the converted data is ready to be picked up. After INTR goes low, we make CS=0 and send a high-to-low pulse to the RD pin to get the data out of the ADC0804 chip.

- Vin (+) and Vin(-)- These are the differential analog inputs where Vin=Vin(+)-Vin(-). Often Vin(-) pin is connected to ground and the Vin(+) pin is used as the analog input to be converted to digital.
- Vcc- This is a +5 volt power supply. It is also used as a reference voltage when the Vref./2 input is open.
- Vref./2- Pin 9 is an input voltage used for the reference voltage. If this pin is open, the analog input voltage for the ADC0804 is in the range of 0 to 5 volts.

Vin/2 (V)	Vin (V)	Step Size (mV)
not connected	0 to 5	5/256=19.53
2.0	0 to 4	4/256=15.62
1.5	0 to 3	3/256=11.71
1.28	0 to 2.56	2.56/256=10

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

- D0-D7- These are the digital data output pins since ADC0804 is a parallel ADC chip. Output voltage can be calculated by the following formula:

$$Dout = \frac{Vin}{Step\ Size}$$

- Analog ground and digital ground- These are the input pins providing the ground for both the analog signal and digital signal.

Steps for the data conversion by the ADC0804 chip

- Make CS=0 and send a low-to-high pulse to pin WR to start the conversion.
- Keep monitoring the INTR pin. If it is low, conversion is finished, otherwise keep polling until it goes low.
- After the INTR has become low, make CS=0 and send a high-to-low pulse to the RD pin to get the data out of the AD0804 IC chip.

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;
       ConvertAndDisplay (value);
       RD = 1;
}
```

```
Void ConvertAndDisplay (unsigned
char value)
 {
    Unsigned char x,d1,d2,d3;
     x=value/10;
     d1=value%10;
     d2=x%10;
    d3=x/10;
    P0=d1;
    MSDelay(250);
    P1=d2;
   MSDelay(250);
   P2=d3;
   MSDelay(250);
 }
```

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

- Sensor Interfacing and Signal Conditioning
- There are different types of sensors available like temperature, pressure, light intensity and flow sensor which converts physical quantity to electrical signals. These are transducers and the output is in the form of voltage, current, resistance and capacitance. For example, temperature is converted to electrical signal using a transducer called a thermistor. The response of thermistor is not linear.
- Linear temperature sensor include the LM34 and LM35 series from national Semiconductor Corp.

- LM-34 sensor- This series are precision integrated circuit temperature sensor whose output voltage is linearly proportional to the Fahrenheit temperature.
- LM-35 sensor- This series are precision integrated circuit temperature sensor whose output is linearly proportional to the Celsius temperature.

- Signal Conditioning- It 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 is commonly called signal conditioning. 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.

- Write an 8051 C program to read temperature from ADC0848, convert it to decimal and put it on P0 with some delay.

```
# include<reg.51.h>
sbit RD = P2^5;
sbit WR = P2^6;
sbit INTR = P2^7;
sfr MYDATA = P1;
void ConvertAndDisplay (unsigned char value)
void MSDelay(unsigned int value);
```

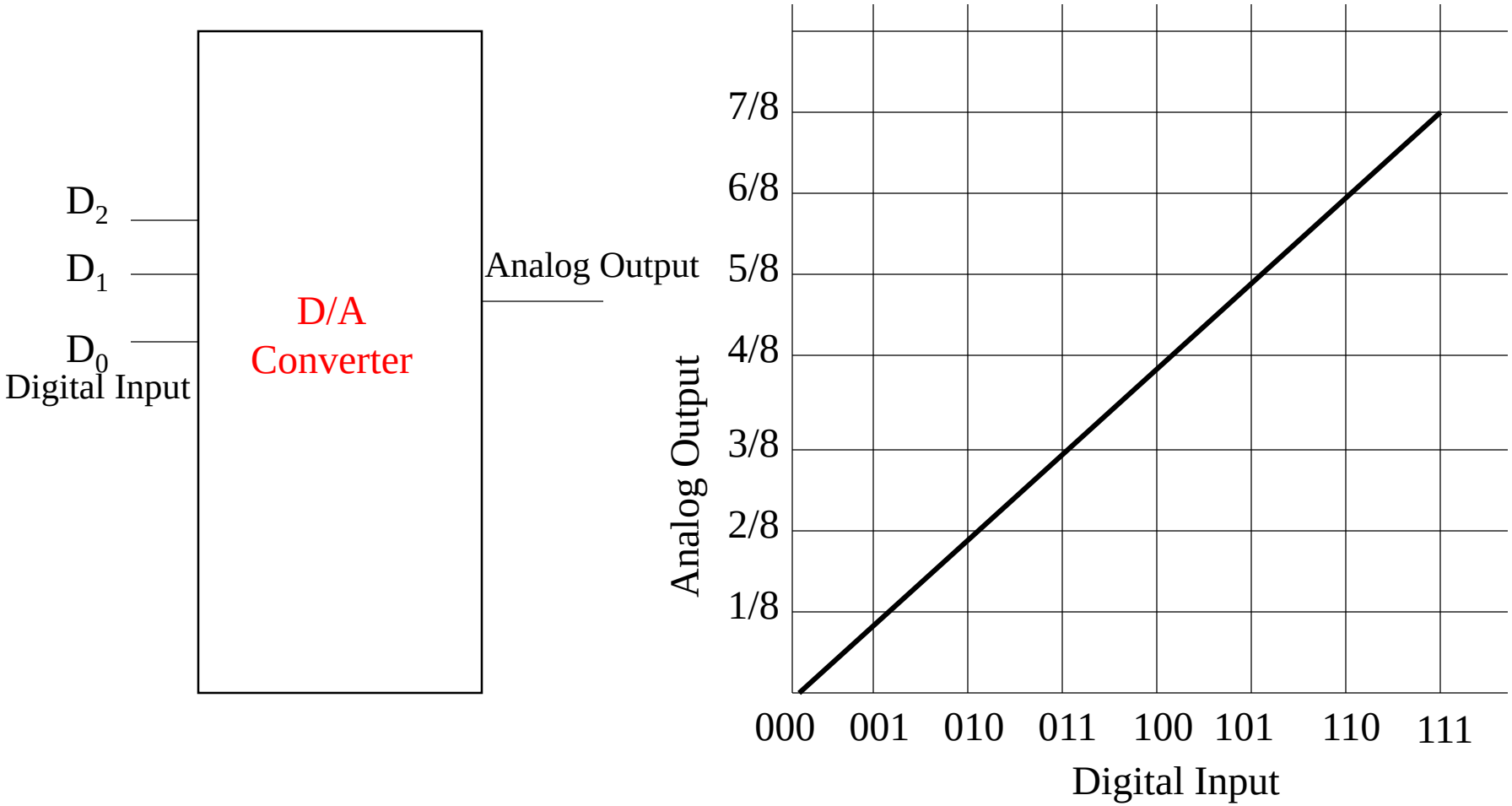
```
void main()
{
   MYDATA = 0XFF;
   INTR = 1;
   RD = 1;
   WR =  1;
while(1)
{
WR = 0;
WR = 1;
While(INTR = =1);
RD = 0;
Value = MYDATA;
ConvertAndDisplay(value);
RD = 1;
}
}
```

```
Void ConvertAndDisplay (unsigned
char value)
 {
    Unsigned char x,d1,d2,d3;
     x=value/10;
     d1=value%10;
     d2=x%10;
    d3=x/10;
    P0=d1;
    MSDelay(250);
    P0=d2;
   MSDelay(250);
   P0=d3;
   MSDelay(250);
 }
```

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

Digital to analog converter

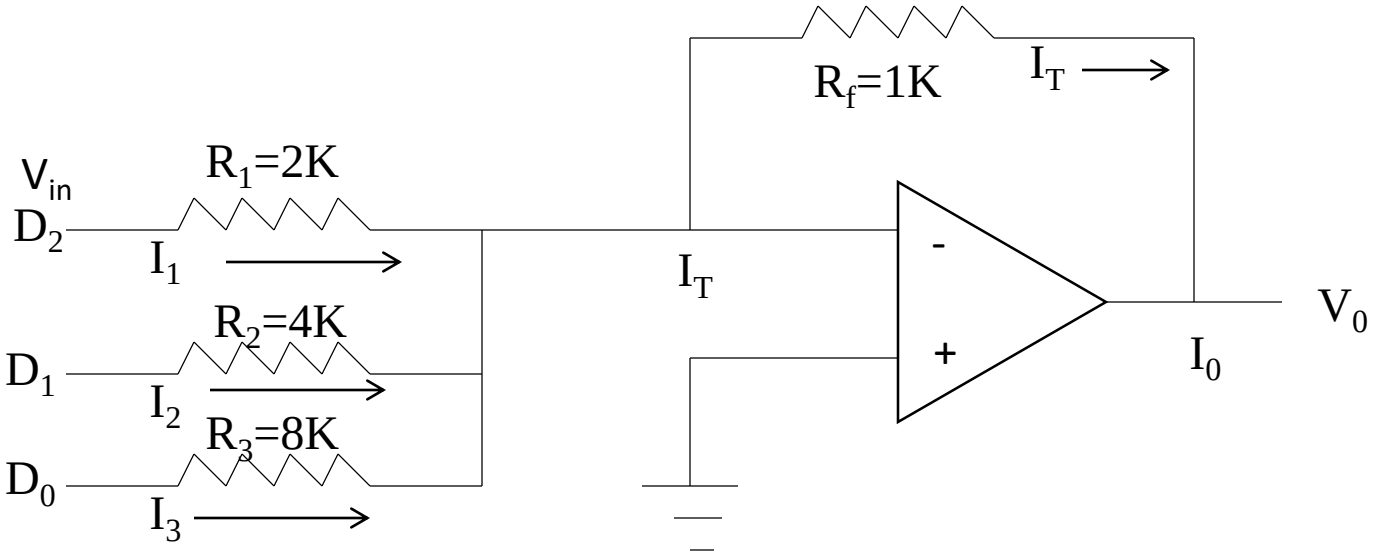
- The digital-to-analog converter (DAC) is a device widely used to convert digital pulsed to analog signals.
- There are two methods for DAC: binary weighted and R/2R ladder.
- The criteria for judging a DAC is its resolution, which is a function of the number of binary inputs.
- The number of data bit input 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 DAC0808 provides 256 discrete voltage levels of output.



A 3-Bit D/A Converter

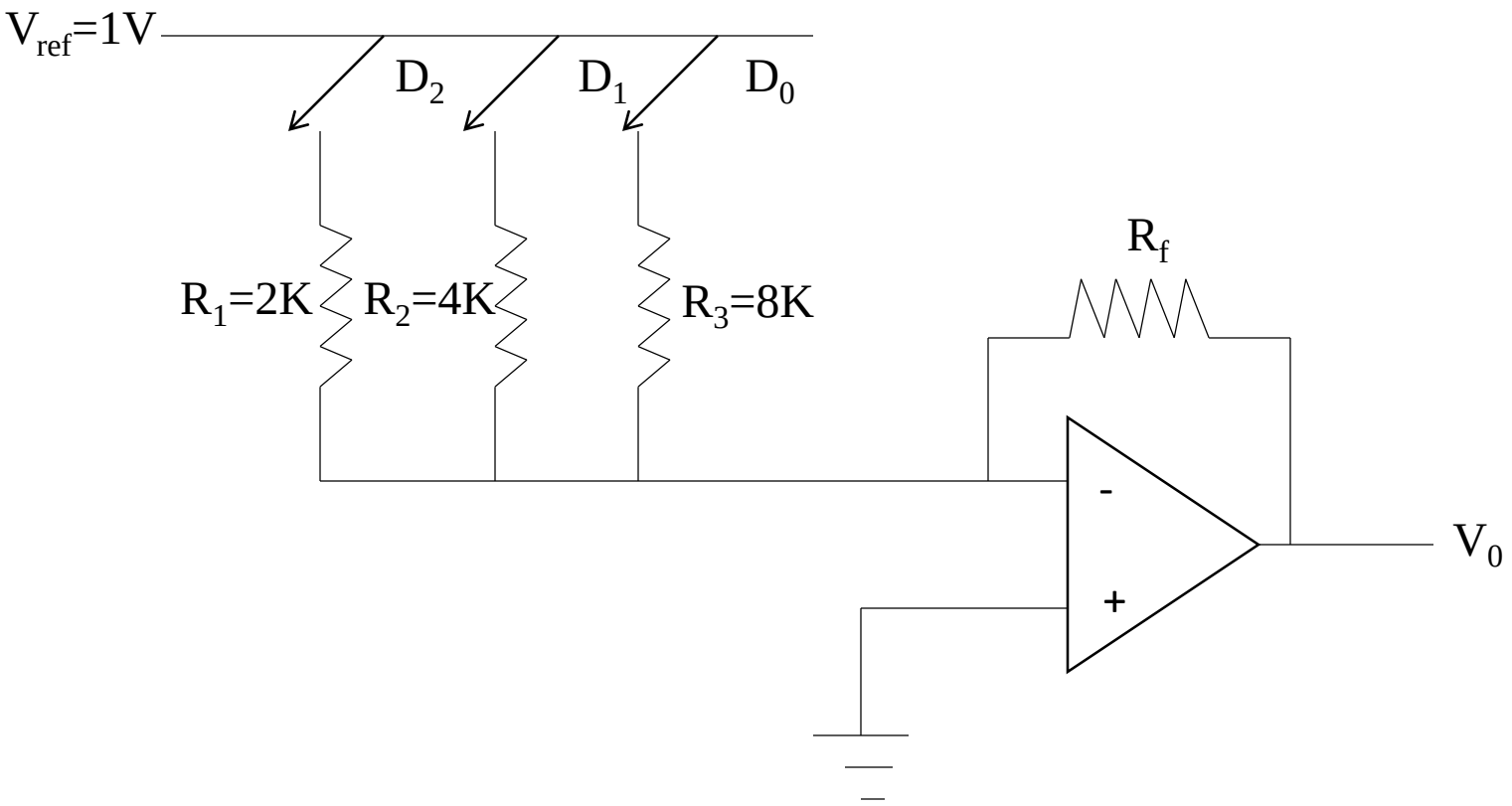
- N-bit D/A converter has n input lines, it can have 2^n input combinations.
- If the full scale analog voltage is 1V, the smallest unit or LSB is equivalent to $1/2^n$ of 1V. This is defined as resolution.
- The MSB represents the half of the full-scale value.
- For the maximum input signal, the output is equal to the value of the full scale input signal minus the value of the 1 LSB input signal.

Binary Weighted Input Resistors



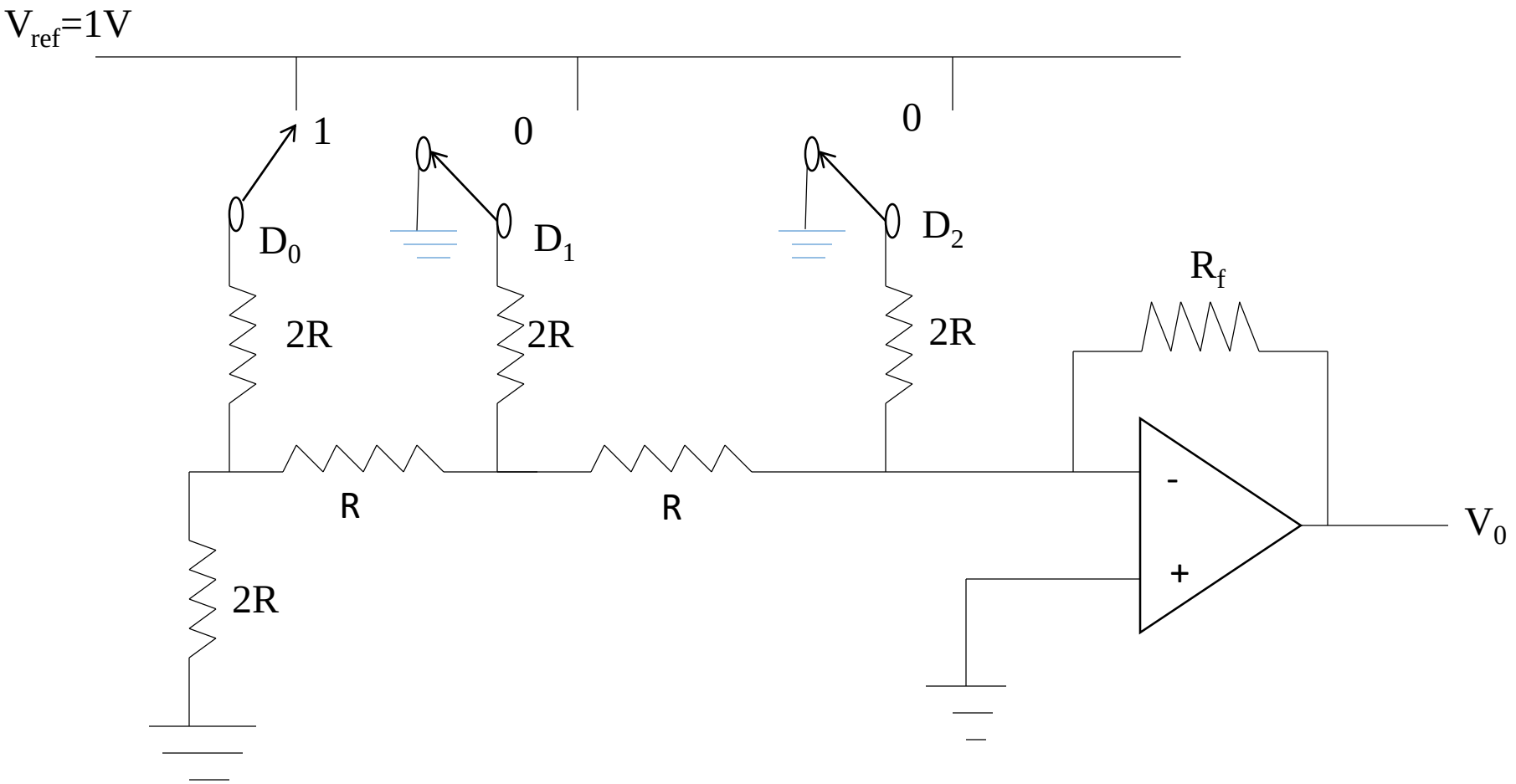
$$V0=-R_f I_T$$

$$Io= (\frac{V_{in}}{2} + \frac{V_{in}}{4} + \frac{V_{in}}{8})$$



Weighted Resistor D/A Converter

R/2R LADDDER NETWORK



R/2R Ladder Network

Dac 0808 chip

- DAC0808 use the R/2R method since it can achieve a much higher degree of precision
- 8 bit parallel digital data input
- Fast settling time (typical value): 150 ns
- Relative accuracy at ±0.19% maximum error
- Full scale current match: ±1 LSB
- Non-inverting digital inputs are TTL and CMOS compatible
- High speed multiplying input slew rate: 8 mA/µs
- Power supply voltage range: ±4.5V to ±18V
- Low power consumption: 33 mW@ ±5V
- Maximum Power dissipation: 1000 mW
- Operating temperature range: 0ºC to +75ºC

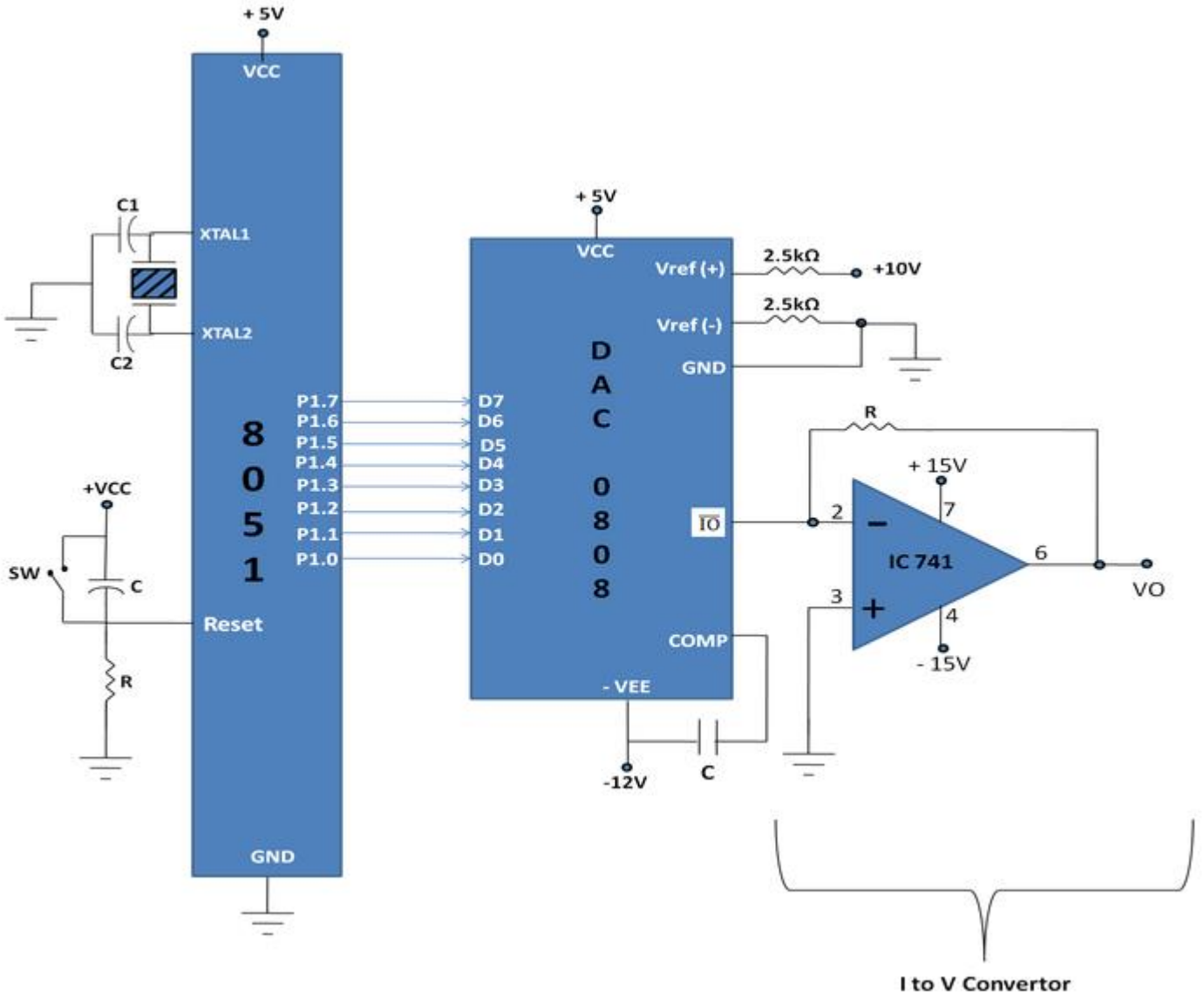
Working of DAC0808

- This DAC0808 IC converts digital data into equivalent analog Current.
- 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 given as:

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

- The I_{ref} current is generally set to 2.0mA.
- Hence we require an I to V converter to convert this current into equivalent voltage.

INTERFACING DAC WITH 8051



- Ideally we connect the output pin I_{out} to a resistor, convert 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.
- Thus, I_{ref} current output is isolated by connecting it to op-amp such as the 741 with R-5K ohms for the feedback resistor.

- Question: In the circuit of interfacing 8051 with DAC0808, write the program to generate a staircase waveform.
- Solution: Let us generate a staircase waveform for 5 steps. Now 255/5=51. So the increment of each step is 51. After the maximum value is reached, the output drops to zero and the next cycle starts.
- Program:

```
        ORG 0000H
CHECK:  MOV A, #00
        MOV P1, A
        ACALL DELAY
RPT:    ADD A, #51H
        MOV P1, A
        ACALL DELAY
        CJNE A, #255, RPT
        SJMP CHECK
```