COURSE NAME WITH CODE: **MICROCONTROLLERS(10EC42)** **Faculty Name: M R , V T G , RGS, AAK** **DATE:** **LECTURE HOUR:**	**UNIT-V**

OBJECTIVES OF THE LECTURE:

To define interfacing

How to interface various peripherals with 8051

Understand base and driver board working

Embedded C programming concepts

REFERENCES:

1. " The 8051 Microcontroller Architecture Programming & Applications", 2e Kenneth J. Ayala
2. " The 8051 Microcontroller and Embedded Systems-using Assembly and C", Muhammad

LIST OF ACTIVITY TO FILL THE PROGRAMME OUTCOME GAP-IF ANY(WITH EVALUATION PROCEDURE):

Interfacing is a process of cohesively connecting peripheral hardware to microcontroller with proper compatibility.

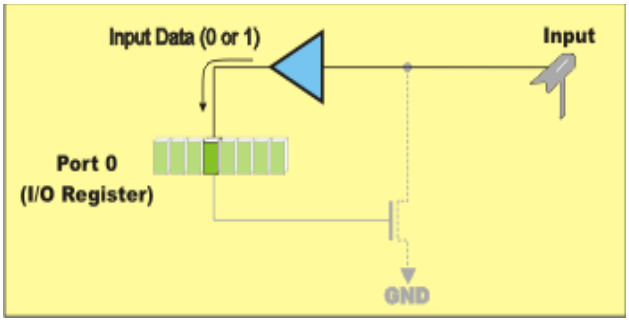
Advantages:

- Provides flexibility in application design
- Modular design and development
- Elegant user interface
- Easy maintenance and upgradability

Complexity issues involved

- Compatibility issues
- Availability of required hardware

I/O Concepts of 8051:



Port0:

Can be used as general IO port Used as multiplexed port: As lower byte address port and data port for memory interface. Only Port without internal pull up resistor

Port1:

Used exclusively as IO port No special functions are associated with P1 Internal pull up resistor is provided.

Port2:

Port2 can be used as I/O port Also can be used as higher byte address port for memory interface Internal pull up resistor is provided.

Port3:

Port3 can be used as I/O port Also each pin is defined for some special function .

HARDWARE INTERFACE

Hardware interface is a process of connecting peripheral devices to microcontroller with appropriate compatibility.

Interfacing considerations are:

- Availability of IO port at both microcontroller and peripheral devices
- Matching protocol between two hardware
- Identifying required control signals
- Processing time and number of bits for communication

Note:8051 can be interfaces with Programmable Peripheral Interface(PPI) chip for additional port if required.

Dual DAC Interface to generate different types of waveforms

Implementation of DAC 0808 interface to 8051 to generate square, triangular, ramp waveforms. Dual DAC Interface to generate …..

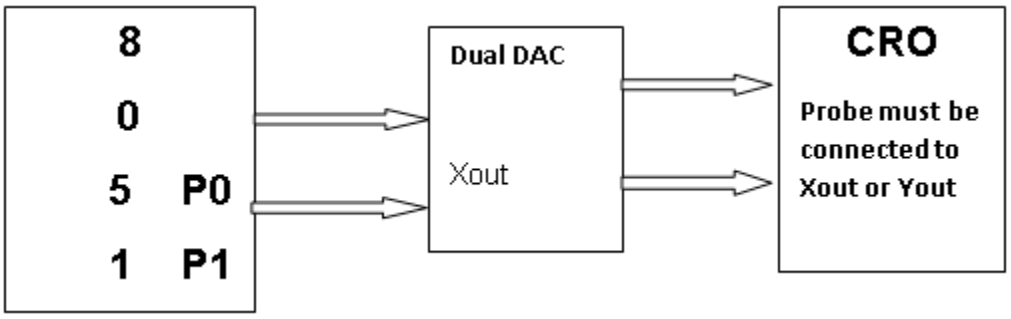
a. Square waveform

b. Triangular Waveform

c. Ramp waveform

d. Sine waveform

e. Sin wave



Theory:

Majority of the integrated circuits of DAC use the R/2R method since it can achieve higher degree of precision. The basic criterion for judging a DAC is its resolution, which is a function of the 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 levels is equal to 2^n, where n is number of data nit inputs. Therefore the 8-input DAC such as DAC0800 provides 256 discrete voltage (or current) levels of output.

The digital inputs are converted to current (I_{out}) and by connecting resistor or op-amp to the I_{out} pin, we convert the result into voltage. The total current provided by the Iout pin is a function of the binary numbers at the D0-D7 inputs of the DAC and the reference current (I_{ref}), and is as follows:
$I_{out}= I_{ref}$ (D7/2 + D6/3 + D5/8 + D4/16 + D3/32 + D2/64 + D1/128 + D0/256)Where D0 is the LSB and D7 is the MSB of the inputs, and Iref is the input current that must be applied to pin 14. The Iref current is generally 2mA. Some DAC also use zener diode (LM336) which overcomes any flu8ctuations associated with the power supply. If I_{ref} is 2mA then when all inputs are high the maximum current is 1.99mA.

Driver Circuit Description:

The Dual DAC interface can be used to generate different waveforms using microcontroller. There are two 8-bit analog to digital converters provided based on DAC0800. The digital inputs to these DACs are provided through the Port 0 and Port 1. The analog output from the DAC is given to operational amplifier which acts as current to voltage converter and isolator between CRO circuit and DAC chip. The output of the op-amp is connected to Xout and Yout points on board from which the waveforms can be observed on CRO. Two 10k Ohm pots are provided for the offset balancing of op-amps. The reference voltage required for the DAC is obtained from onboard voltage regulator uA723. The voltage generated by this regulator is about 8V. The output of the DAC vary from 0 V to 5V corresponding to values between 00 to FF respectively.

Installation:

- The interface module has a 26-pin connector at one edge of the card which is connected to Microcontroller board through FRC (Flat Ribbon Cable) connector.
- External power supply of +12, -12 and GND are connected to points marked through 4-pin connector provided.

Applications: Sound card, CD players, Digital music players etc…

Program for square wave:

```
#include <REG51xD2.H>
sbit Amp = P3^3;                    /* Port line to change amplitude */
sbit Fre = P3^2;                /* Port line to change frequency */
void delay (unsigned int x)         /* delay routine */
{
    for (;x>0;x--);
}
main()
{
    unsigned char on = 0x7f,off=0x00;
    unsigned int fre = 100;
    while(1)
```

```
{
    if(!Amp)                    /* if user choice is to change amplitude */
    {
        while(!Amp);            /* wait for key release */
        on+=0x08;       /* Increase the amplitude */
  }
if(!Fre)    /* if user choice is to change frequency */
    {
    if(fre > 1000)              /* if frequency exceeds 1000 reset to default */
    fre = 100;
    while(!Fre);                /* wait for key release */
    fre += 50;
    }                           /* Increase the frequency */
P0=on;                          /* write amplitude to port */
P1=on;
delay(fre);
P0 = off;                       /* clear port */
P1 = off;
delay(fre);
}
    }
```

b) <u>Algorithm for Triangular wave generation:</u>

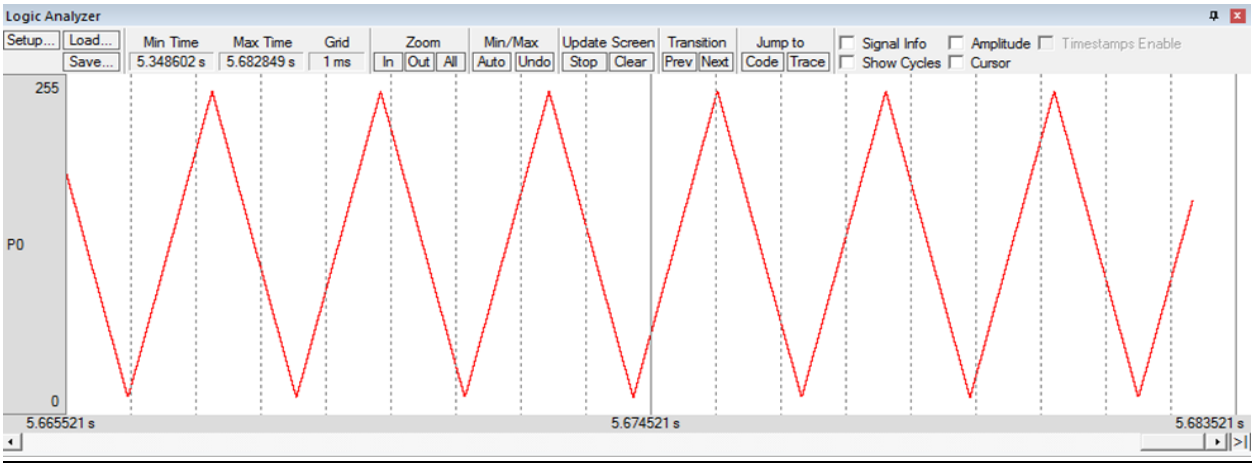
- Output the initial value 00 through P0.
- Increment it in steps of 1 until a count value of FFh (5V) is reached. Every time repeat step 1.
- Decrement it in steps of 1 until a zero value is reached and repeat step 1.

<u>Program for triangular wave:</u>

```
#include <REG51xD2.H>
main()
{
    unsigned char i=0,slope=1;
   P0 = 0x00;                   /* P0 as Output port */
    while(1)
    {
        for(i=0;i<0xfe;)                /* Generate ON pulse */
```

```
        {
                P1 = i;
                P0 = i;
                i=i+slope;
        }
        for(i=0xfe;i>0x00;)             /* Generate OFF pulse */
        {
                P0 = i;
                P1 = i;
                i=i-slope;
        }
   }
}
```

Expected output:



c) Algorithm for Ramp wave generation

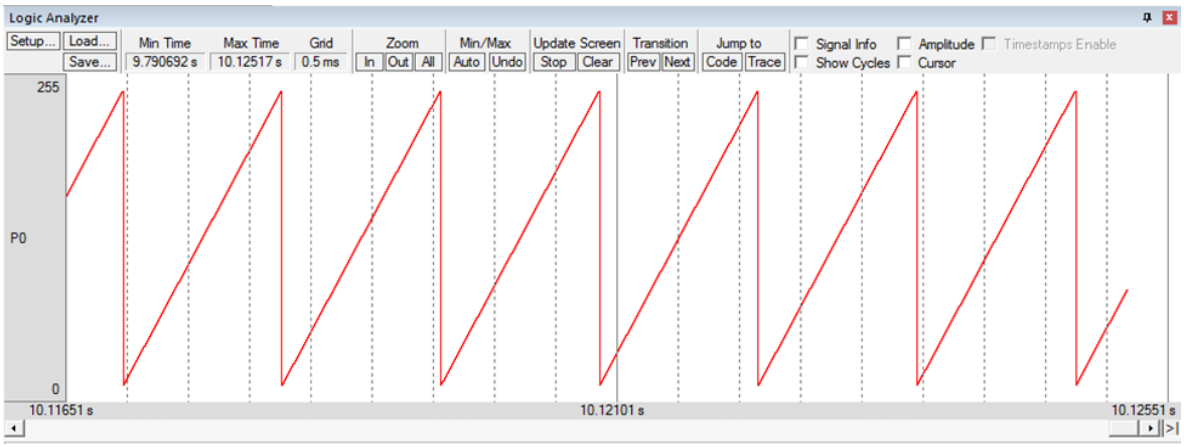
- Output the initial value 00 through P0.
- Increment it in steps of 1 until a count value of FFh (5V) is reached. Every time repeat step 1.
- Repeat step 1 & 2 continuously.

Program for Ramp waveform

```
#include <REG51xD2.H>
main ()
{
    Unsigned char i=0,slope=1,rising=1;
    P0 = 0x00;                          /* P0 as Output port */
    while (1)
    {
```

```
If(rising==1)
    {
        for(i=0;i<0xfe;)            /* Generate ON pulse */
        {
            P1 = i;
            P0 = i;
            i=i+slope;
        }
    }
else
    {
        for(i=0xfe;i>0x00;)         /* Generate OFF pulse */
        {
            P0 = i;
            P1 = i;
            i=i-slope;
        }
      }
    }
 }
```

Simulation output:



D) Sine wave generation

To generate a sine wave, we need a table whose values represent the magnitude of the sine of angle between 0 to 360 degrees. The values for the sine function vary from -1.0 to +1.0.Ensure that only integer numbers are output to the DAC by 8051 microcontroller. Assuming the full scale voltage is 10V for DAC which is achieved when all the data inputs of the DAC are high. Therefore, to achieve the full scale 10V output, we use the following equation.

V_{out}= 5V + (5 x Sin θ)

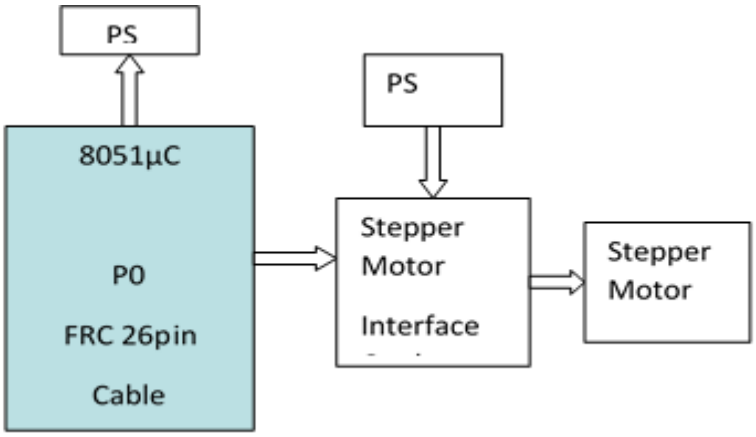
Compute different step values (θ = 20^o,15^o…) of sine using the equation V= 5V +5Vsinθ. . Output the values through P0.More the steps smoother will be sine wave.

E.g.: θ = 0^o

Volts= 5V +5sinθ = 5V (5V is DC shift & 5sinθ is AC variation)

The value sent to DAC is **25.6XVolts = value**. (8 bit DAC has 256 states, assuming peak value of sine wave is 10V. Each volt is represented in the step of 25.6 states)

STEPPERMOTORINTERFACE



Description:

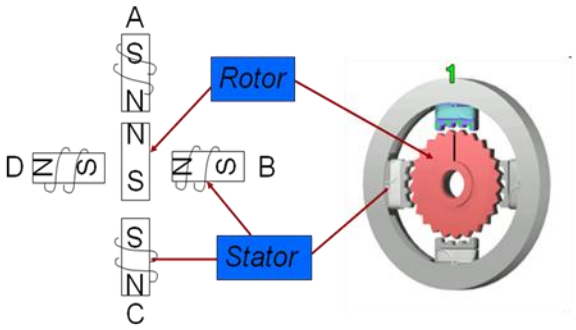
- Unlike DC motor Stepper motor rotates in steps.
- Programmatically following parameters can be controlled
  - Angle of rotation
  - Direction of rotation
  - Speed of rotation (RPM)
- Stepper motor has 4 coils which forms the stator and a central rotor.
- Rotation depends on excitation of stator coils.

step	coil A	coil B	coil C	coil D
1	0	0	0	1
2	1	0	0	0
3	0	1	0	0
4	0	0	0	1

Anyone of these values forms the initial value. To get 360o revolution 200 steps are required. Step angle= 360^o /200 = 1.8^o (difference between 2 teeth).

Algorithm for Stepper Motor

- Configure P0 as output.
- Apply the initial excitation of 11 to motor coils through P0.
- For clockwise motion -Rotate right once the excitation and repeat step 2.
- For anticlockwise motion -Rotate left once the excitation and repeat step 2.



Theory:

A stepper motor is a device that translates electrical pulses into mechanical movement. The stepper motor shaft moves in a fixed repeatable increment, which allows precise angle control. This repeatable fixed movement is possible as a result of basic magnetic theory where poles of the same polarity repel and opposite polarity attract. The direction of the rotation is dictated by the stator poles. The stator poles are determined by the current sent through the wire coils. As the direction of the current is changed, polarity is also changed causing the reverse motion of the rotor.

Step angle : This depends on the number of teeth on the stator and the rotor. Step angle is the minimum degree of rotation associated with the single step.

We are using a stepper motor with 50 teeth on rotor and 4 on stator, hence the step angle is calculated as

Step Angle = 360/ (No. of teeth on rotor × No of teeth on Stator)
= 360/ (50 × 4)
= 1.8

Therefore steps per revolution are 200.

Steps per second and rpm relation.

Steps/sec = (rpm x Steps per revolution) / 60

Drive sequences:

4 step sequence: 1001,1100,0110,0011
8 Step sequence: 1001, 1000,1100,0100,0110,0010,0011,0001
Wave drive 4 step sequence: 1000,0100,0010,0001
Types of stepper motor:

- Permanent magnet (PM)
- Variable reluctance (VR)

Comparison of different types (based on phase) of stepper motor are :

Parameter	Universal	Unipolar	Bipolar
Number of connections	8	6	4
Modes	All 3	2 (Uni / Bi)	Only Bi
Extra circuitry	-	-	H-bridge
Operational current	Low	Low	High
Holding torque	Low	Low	High

Construction:

Stepper motors commonly have permanent magnet rotor (also referred as shaft) surrounded by a stator. Stepper motor have four stator windings that are paired with the centre tapped common, this type is commonly referred as four phase or Unipolar stepper motor. The centre tap allows change of the current direction in each of two coils when winding is grounded thereby resulting in polarity change of stator. The stepper motor used has total of 6 leads, 4 leads represent 4 stator winding and 2 common for the centre tapped leads.

Driver Circuit Description:

The stepper motor interface uses 4 transistor pairs (SL100 & 2N3055) in a Darlington pair configuration. Each Darlington pair is used to excite the particular winding of the motor connected to 4 pin connector on the interface. The inputs to these transistors are from the Microcontroller board. Lower nibble of Port 0 i.e. P0.0, P0.1, P0.2, p0.3 are the four lines brought out of the 26 pin FRC male connector (J7) on the interface module. The freewheeling diodes across each winding protect transistor from switching transients.

Installation:

- The interface has two 3-pin and one 4pin connectors.
- Plug in 4-pin polarized connector of the motor to interface and the 3-pin connector of the motor to 3-pin connector of the interface marked as “WHT BLK”.
- Connect 3-pin female connector of the stepper motor power supply to the connector of the interface marked as “GND +5/12V”.
- Connect the 26-pin FRC on the interface module to J7 of controller kit.

Applications: Card reader, dot matrix printers, Hard disk drive (HDD), Floppy disk drive (FDD), CD/ DVD drive, Clocks to rotate hands etc…

//Program for stepper motor interface //

```
#include <REG51xD2.H>
void delay (unsigned int x)                    /* Delay Routine */
    {
        for(;x>0;x--);
        return;
    }
Main ( )
{
    unsigned char Val, i;
    P0=0x00;
    Val = 0x11;
    for (i=0;i<4;i++)
    {
        P0 = Val;
        Val = Val<<1;   /* Val= Val>>1; for clockwise direction*/
        delay (500);
    }
}
```

DC MOTOR INTERFACE TO 8051

Algorithm for DC motor interface:

- Configure P0, P1 as output port and P3 as input port.
- Let initially the motor rotate with half speed count 7fh.
- If "INR" button is pressed reduce the count because the speed is inversely proportional to count.
- If "DEC" button is pressed increase the count.

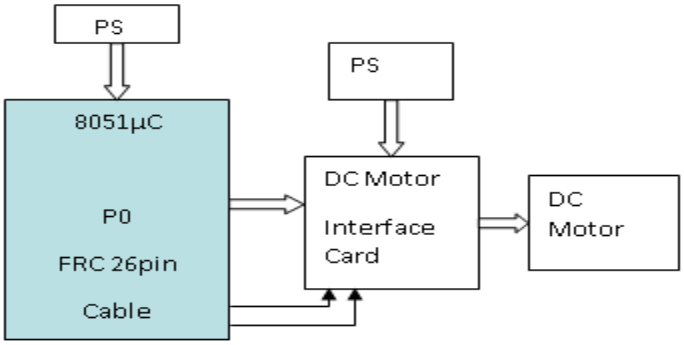


Fig. : Block-diagram of the interfacing of a DC motor to a microcontroller

Theory:

Direct current (DC) motor is another widely used device that translates electrical to mechanical movement. In DC motor we have only + and – leads. Connecting from DC voltage source moves the motor in one direction and by reversing the polarity, the motor will move in opposite direction. DC motors have two rpms : no load and loaded which will be indicated on data sheet specifications. The normal voltage rating varies from 1V to 150V and current rating varies from 25mA to few amperes. The DC motor follows three important relations shown below.

Rpm = k / load; at constant current
Rpm = k × current; at constant load
Current = k × load; at constant Rpm

where, k is constant of proportionality.

DC motor speed can be varied using PWM (Pulse Width Modulation) technique. By changing the width of the pulse applied to the DC motor the speed of the motor is varied. Even though the amplitude of the voltage is same as the width of the ON time of pulse increases speed also increases.

Installation:

- Connect DC motor to Microcontroller board through FRC cable to J7 (26-Pin connector).
- AC main Power supply is independently connected to DC motor.

Application:

1. CPU fan, Processor cooling fan etc…

Program for DC motor :

```
#include <REG51xD2.H>
sbit  inr= P3^2;    //speed increment switch
sbit  dcr= P3^3;   //speed decrement switch
main()
{
    unsigned char i=0x80;
    P0 = 0x7f;  /*Run the motor at half speed.*/
    while (1)
    {
        if (!inr)
        {
            while (!inr);
            if(i>10)
            i=i-10;        //increase the DC motor speed
        }
        if(!dcr)
        {
            while(!dcr);
            if(i<0xf0)
            i=i+10;        //decrease the DC motor speed
        }
        P0=i;
    }
}
```

ALPHANUMERIC LCD PANEL INTERFACE TO 8051

Theory:

A liquid crystal display (LCD) is a thin, flat electronic visual display that uses the light modulating properties of liquid crystals (LCs).LCD panels have in built refreshing controller relieving CPU from the task. LCDs are more energy efficient, and offer safer disposal, than CRTs. Its low electrical power consumption enables it to be used in battery-powered electronic equipment. It is an electronically-modulated optical device made up of any number of pixels filled with liquid crystals and arrayed in front of a light source (backlight) or reflector to produce images in color or monochrome.

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	'0' to select command register ; '1' to select data register
5	R/W	I	'0' for write ; '1' for Read
6	E	I/O	Enable
7-14	DB0 – DB 7	I/O	The 8-bit data bus

LCD Command codes:

Hex Code	Instruction description	Hex Code	Instruction description
01	Clear display screen	0E	Display ON, cursor blinking
02	Return home	0F	Display ON, cursor blinking
04	Decrement cursor (Shift cursor to left)	10	Shift cursor position to left
06	Increment cursor (Shift cursor to right)	14	Shift cursor position to right
05	Shift display right	18	Shift the entire display to left
07	Shift display left	1C	Shift the entire display to right
08	Display OFF, cursor OFF	80	Force cursor to beginning of 1st line
0A	Display OFF, cursor ON	C0	Force cursor to beginning of 2nd line
0C	Display ON, cursor OFF	38	2 lines and 5x7 matrix

Driver Circuit Description:

LCD accepts characters in ASCII format. Character display font in LCD module is dot matrix i.e. each character in LCD module can be represented by 7x5 matrix. This module is built over 16x1 LCD in which the display data RAM address for the first line is from 00H to 14H and for second line it is 29H to 3CH

LCD module has got an automatic reset which is critically dependent upon power supply voltage. Voltage has to rise from 0.2V to 5V within 10 to 15 ms for LCD to reset. Since this is not accurate; it can also be reset during initialization. To reset 30H has to be sent 3 times with some delay, busy flag of LCD module is set while LCD is resetting, during this time data can't be written on to the LCD.

- Port 2 of microcontroller is connected to D0 to D7 pins of LCD module.
- Control signals RS, R/W and E are connected to P3.7, P3.6 and P3.5 respectively.

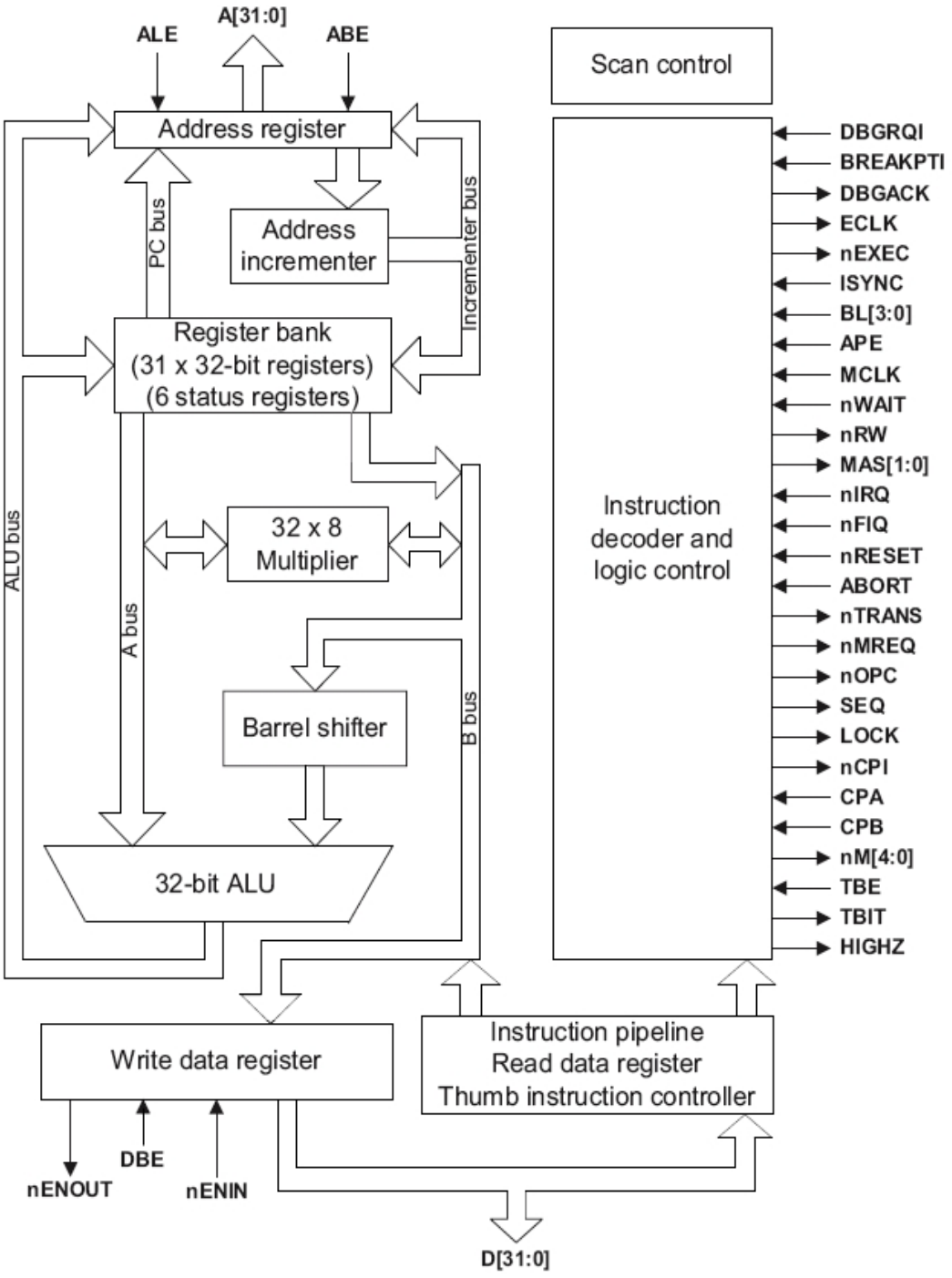
PROGRAM

```
#include<reg51xd2.h>
//Function prototype declaration
void lcdcmd (unsigned char value);
void msdelay (unsigned int itime);
void lcddata (unsigned char value);
unsigned int i;
sfr ldata=0XA0;         //0xA0 is address of Port 2
sbit rs = P3^7; //rs -> Register Select, 0 – Command Register: 1-Data Register
sbit rw = P3^6;         //rw -> Read / Write, 0 – Write : 1- Read
sbit en = P3^5;         //en -> Enable
void main()
{
        lcdcmd(0x38);//Defines character matrix i.e 7x5
        msdelay(250); //Delay is introduced as LCD need time to respond
        lcdcmd(0x0E);//Display on cursor blinking
        msdelay(250);
        lcdcmd(0x01);//Clear display
        msdelay(250);
        lcdcmd(0x06);//Increment cursor
        msdelay(250);
        lcdcmd(0x86);
        msdelay(250);
        lcddata('E');
        msdelay(250);
        lcddata('N');
        msdelay(250);
        lcddata('C');
}
void lcdcmd(unsigned char value)
{
        ldata=value;            //Information bits are copied to Port 2
        rs=0;                   //Selecting Command register
        rw=0;                   //Opted for Write operation
        en=1;
```

```
        msdelay(1);
        en=0;
        return;
}
void lcddata(unsigned char value)
{
        ldata=value;
        rs=1;                   //Selecting Data Register
        rw=0;
        en=1;
        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++);
}
```

ARM7

ARM-Advanced RISC Machine is a 32-bit RISC (Reduced Instruction Set Computer) processor architecture developed by ARM Holdings. Many beginners sometimes misunderstood that the ARM is microcontroller or processor but in reality, ARM is an architecture which is used in many processors and microcontrollers. The ARM architecture licensed to companies that want to manufacture ARM-based CPUs or System-on-Chip products. This enables the companies to develop their own processors compliant with the ARM instruction set architecture. For example, the device we are using LPC2148 is ARM architecture based SOC product developed by NXP Semiconductor. Similarly, all major semiconductor manufacturers like Atmel, Samsung, TI etc. they all make ARM based SOCs.



ARM7-BLOCK DIAGRAM

In 2005, about 98% of all mobile phones sold used at least one ARM processor. The low power consumption of ARM processors has made them very popular: 37 billion ARM processors have been produced as of 2013, up from 10 billion in 2008. The ARM architecture (32-bit) is the most widely used architecture in mobile devices, and most popular 32-bit one in embedded systems.

According to ARM Holdings, in 2010 alone, producers of chips based on ARM architectures reported shipments of 6.1 billion ARM Based processors, representing 95% of smartphones, 35% of digital televisions and set-top boxes and 10% of mobile computers. It is the most widely used 32-bit instruction set architecture in terms of quantity produced.