UNIT – 3
INTERFACING MICROCONTROLLER

PROGRAMMING 8051 TIMERS

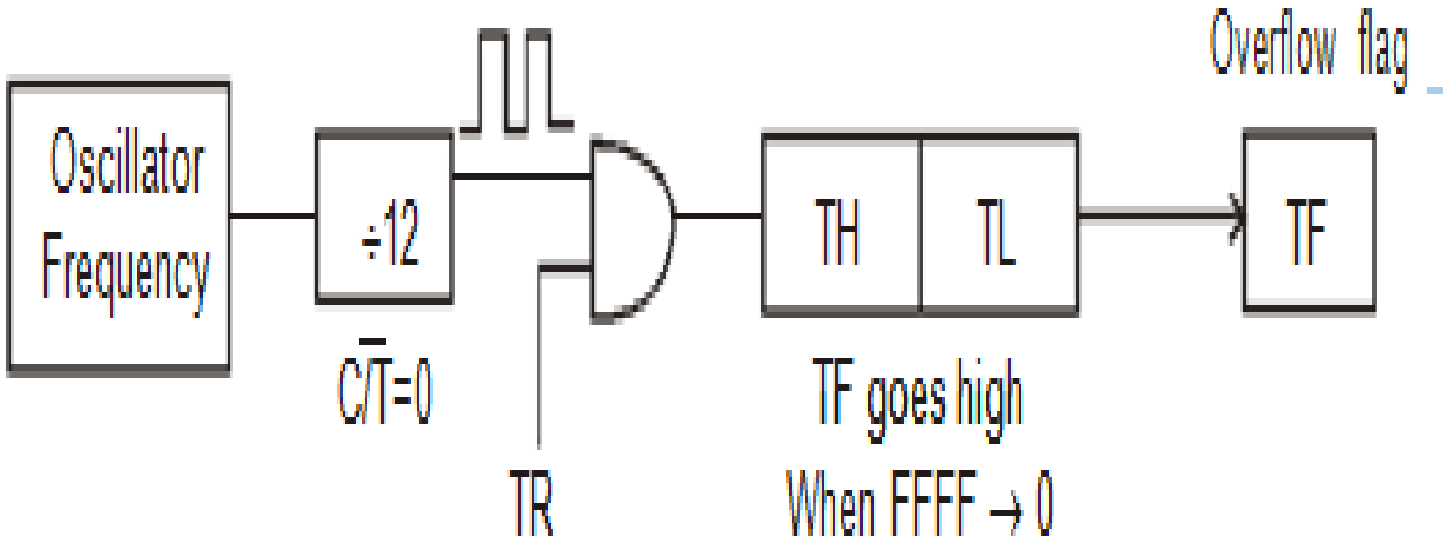
Mode 1 Programming

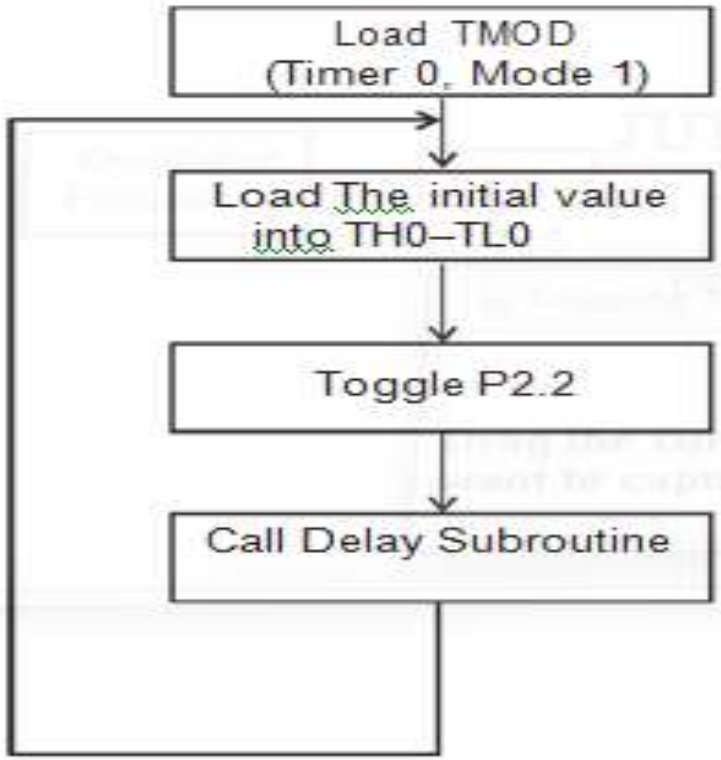
Operations of mode 1:

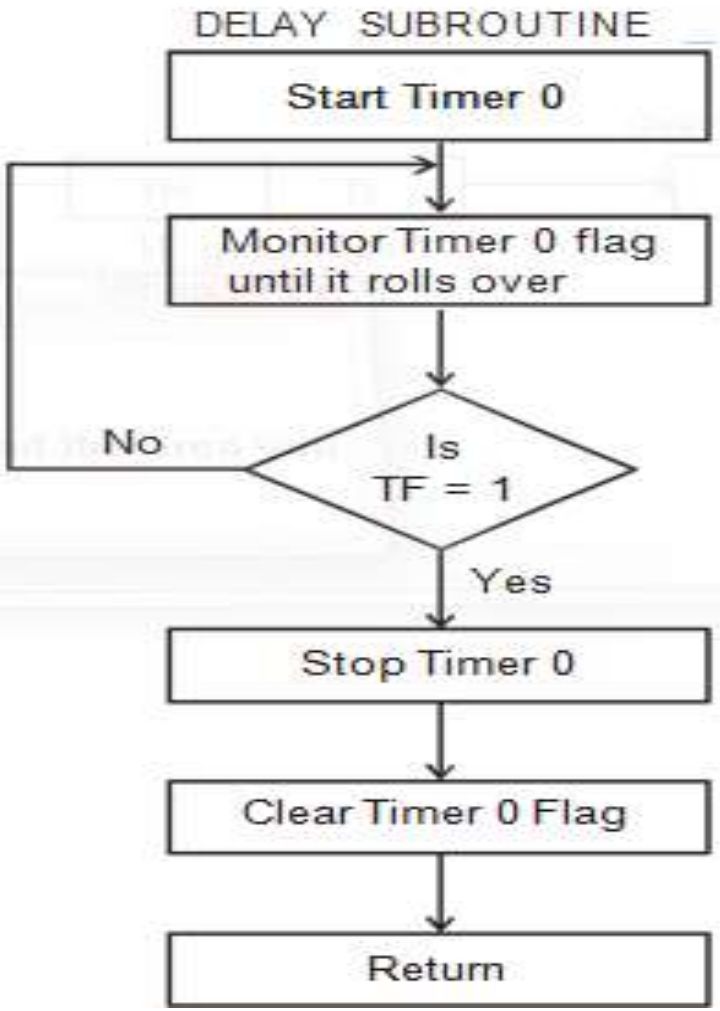
- It allows values of **0000 H to FFFF H to be loaded** into the timer's registers TL and TH.
- After TH and TL are loaded with a 16 - bit initial value, the timer must be started.
- This is done by "SET B TR0" for Timer 0 and "SET B TR1" for Timer 1.

- After the timer is started, it starts to count up. It counts up until it reaches its limit of FFFF H. When it rolls over from FFFF H to 0000H, it sets high a flag bit called TF (Timer Flag). This timer flag can be monitored. When this timer flag is raised, one option would be to stop the timer with the instructions “CLR TR0” or “CLR TR1” for Timer 0 and Timer 1 respectively.
- After the timer reaches its limit and rolls over to repeat the process the registers TH and TL must be reloaded with the original value and TF must be reset to 0

TIMER FOR MODE 1



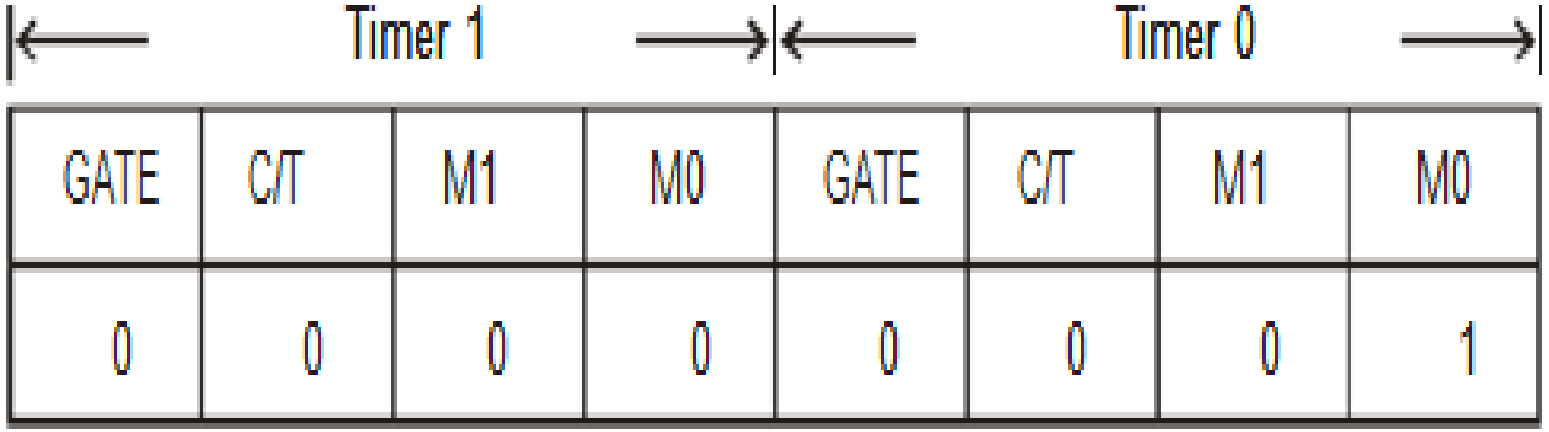

Load TMOD
(Timer 0, Mode 1)
Load The initial value
into TH0–TL0
Toggle P2.2
Call Delay Subroutine


DELAY SUBROUTINE
Start Timer 0
Monitor Timer 0 flag
until it rolls over
Is
TF = 1
No
Yes
Stop Timer 0
Clear Timer 0 Flag
Return

PROGRAM

```
            MOV     TMOD, # 01
            MOV     TL0, # 0F2 H
LOOP 1:     MOV     TH0, # 0FF H
            CPL     P2.2
            ACALL   DELAY
            SJMP    LOOP 1
DELAY:      SET     TR0
LOOP 2:     JNB     TF0, LOOP2
            CLR     TR0
            CLR     TF0
            RET
```

TMOD register :



GATE	C/T	M1	M0	GATE	C/T	M1	M0	
0	0	0	0	0	0	0	1	=01 H

Timer 0, Mode 1 –16 bit Timer mode.

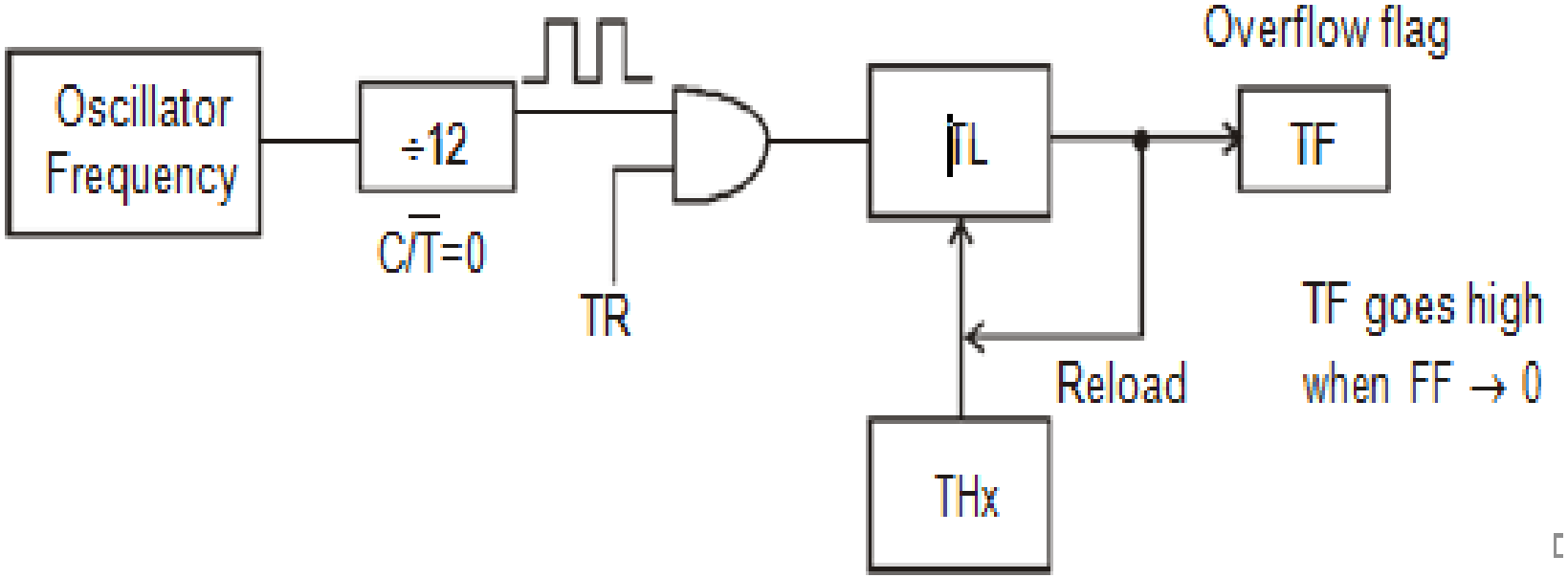
Program :

```
        CLR     P1.4
        MOV     TMOD,   #01 H
LOOP 1: MOV     TL0,    #0B0 H
        MOV     TH0,    #3C H
        SET B   P1.4
        SET B   TR0
LOOP2:  JNB     TF0, LOOP2
        CLR     TR0
        CLR     TF0
        CLR     P1.4
        LJMP    LOOP1
```

Mode 2 Programming

Operations of Mode 2:

- Mode 2 allows only values of 00 H to FF H to be loaded into the timer's register TH.
- After TH is loaded with the 8 bit value, the 8051 gives a copy of it to TL. Then the timer must be started. This is done by "SET B TR0" for Timer 0 and "SET B TR 1" for Timer 1.
- After the timer is started, it started it starts to count up by incrementing the TL register. It counts up until it reaches its limit of FFH. When it rolls over from FFH to 00H, it sets high the timer flag (TF) TF0 is raised for Timer 0 and TF 1 is raised for Timer 1.
- When the TL register rolls from FF H to 00 H and TF is set to 1, TL is reloaded automatically with the original value kept by the TH register. To repeat the process clear TF (anti - reloading).


Oscillator Frequency
÷12
C/T=0
TR
TL
Overflow flag
TF
TF goes high when FF → 0
Reload
THx

PROCEDURE

- Load the TMOD value register indicating which timer (Timer 0 or 1) is to be used and select the timer mode 2.
- Load the TH registers with the initial count value.
- Start the timer.
- Keep monitoring the timer flag (TF) with “JNB TFx” instruction. When TF becomes high get out of the loop.
- Clear the TF flag
- Go back to step 4, since Mode 2 is auto - reload.

TMOD register :

← Timer 1 →				← Timer 0 →				
GATE	C/T	M1	M0	GATE	C/T	M1	M0	
0	0	1	0	0	0	0	0	=20 H

Timer 1, Mode 2 – Auto reload

Program :

```
        MOV     TMOD, # 20H
        MOV     TH1, #6
        SETB    TR1
LOOP:   JNB     JF1, LOOP
        CPL     P1.3
        CLR     TF 1
        SJMP    LOOP
```

COUNTER PROGRAMMING

- When C/T = 1, the timer is used as a counter and gets its pulses from outside the 8051. The counter counts up as pulses are fed from pins T0 (Timer 0 input) and T1 (Timer 1 input). These two pins belong to port 3. For Timer 0, when C/T = 1 pin 3.4 provides the clock pulse and counter counts up for each clock pulse coming from that pin.
- Similarly for Timer 1, when C/T = 1 each clock pulse coming in from pin 3.5 makes the counter countup.

  P3.4 - T0 - Timer/Counter 0 external input

  P3.5 - T1 - Timer/Counter 1 external input
- In counter mode, the TMOD, TH and TL registers are the same as for the timer. Counter programming also same as timer programming.

```
        MOV     TMOD, # 0110 0000 B   ;  Counter 1, Mode 2, C/T =1
        MOV     TH 1, # 00 H          ;  Clear TH 1
        SET B   P3.5                  ;  Make T1 input
LOOP 1: SET B   TR 1                  ;  Start the counter
LOOP 2: MOV     A, TL 1               ; Get copy of count TL 1
        MOV     P2, A                 ;  Display it on Port 2
        JNB     TF 1, LOOP 2          ; Goto Loop 2 if TF = 0
        CLR     TR1                   ;  Stop the counter 1
        CLR     TF 1                  ;  Make TF = 0
        SJM P   LOOP 1                ;  Jump to Loop 1.
```

SERIAL PORT PROGRAMMING

Programming the 8051 to transfer serially

- The TMOD register is loaded with the value 20H, indicating the use of Timer 1 in mode 2

TMOD Register

GATE	C/T	M1	M0	GATE	C/T	M1	M0
0	0	1	0	0	0	0	0

If M_1 M_0 = 10, 8 bit Auto - reload counter

- The TH 1 is loaded with one of the values in Table to set the baud rate for serial data transfer.

Baud rate	TH 1 (Decimal)	TH 1 (Hex)
9600	–3	FD
4800	–6	FA
2400	–12	F4
1200	–24	E8

- The SCON register is loaded with the value 50 H, indicating serial mode 1, where 8-bit data is framed with start and stop bits.

SCON register

SM 0	SM 1	SM 2	REN	TB 8	RB 8	TI	RI
0	1	0	1	0	0	0	0

If SM 0, SM 1 = 01, Serial Mode 1, 8 bit data, 1 stop bit 1 start bit

- TR 1 is set to start Timer 1.
- TI is cleared by the “CLR TI” instruction.
- The character byte to be transferred serially is written into the SBUF registers.
- The TI flag bit is monitored with the use of the instruction “JNB TI, XX ” to see if the character has been transferred completely.
- To transfer next character, go to step 5.

Program

Write an ALP to transfer letter 'E' serially at 4800 baud continuously.

Solution:

```
        MOV     TMOD, # 20 H    ;   Timer 1, Mode 2 (Auto-reload)
        MOV     TH 1 #-6        ;   4800 baud rate
        MOV     SCON, # 50 H    ; 8-bit, 1 stop, 1 start, REN enabled
        SET B   TR 1            ;   Start Timer 1
LOOP 1: MOV     SBUF, # 'E'     ;   Letter 'E' to be transferred
LOOP 2: JNB     TI, LOOP2       ; Wait for the last bit
        CLR     TI              ;   Clear TI for next character
        SJMP    LOOP 1          ;   Go to Loop 1 for sending 'E'
```

Programming the 8051 to receive data serially

- The first 4 steps are as same in programming to transfer data serially.
- RI is cleared with “CLR RI “ instruction.
- The RI flag bit is monitored with the use of the instruction “JNB RI, XX” to see if the character has been received yet.
- When RI is raised, SBUF has the byte. Its contents are moved into a safe place.
- To receive the next character, go to step 5.

Program

Write an ALP to receive bytes of data serially and put them in Port 2. Set the baud rate at 2400, 8 bit data and 1 stop bit.

Solution:

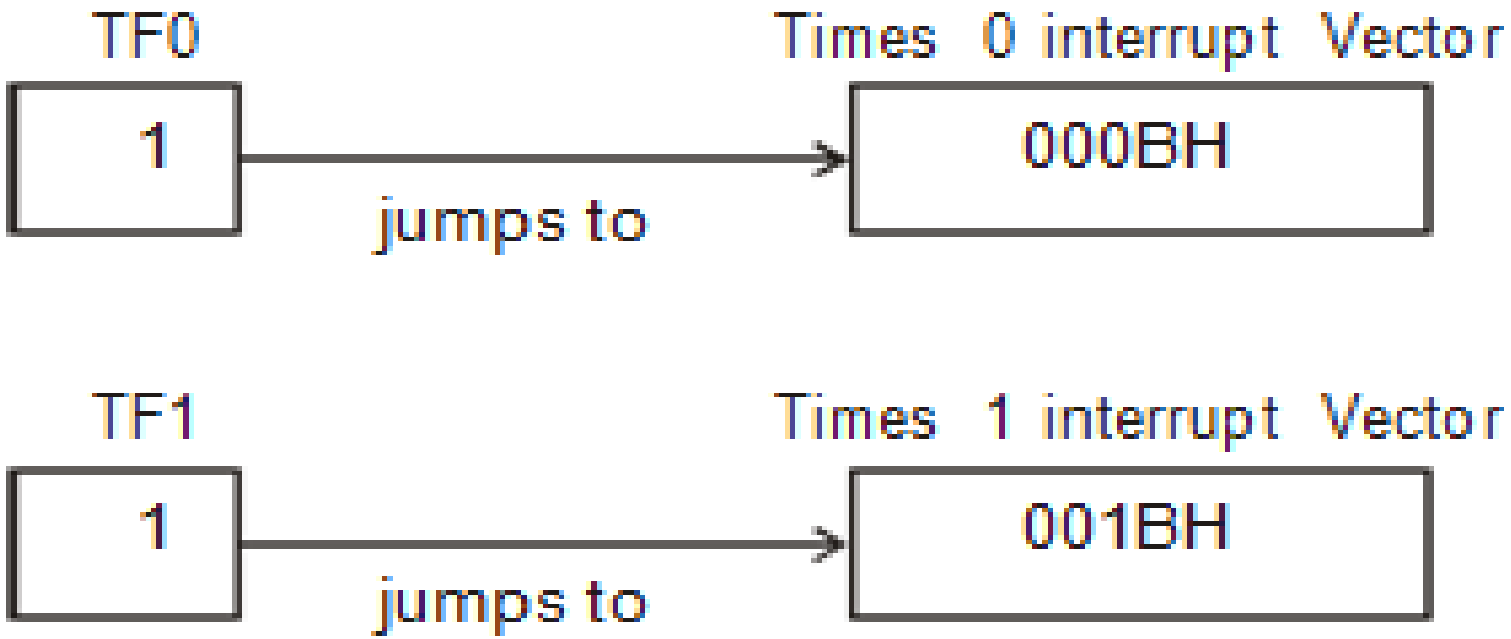
```
        MOV     TMOD, # 20 H    ;   Timer 1, mode 2
        MOV     TH 1, # F4 H    ;  For 2400 baud TH1=–12 (F4 H)
        MOV     SCON, # 50 H    ;  8-bit, 1 stop, REN enabled
        SET B   TR 1            ;   Start Timer 1
LOOP 1; JNB     RI, LOOP 1      ;   Wait for character to come in
        MOV     A, SBUF         ;   Save incoming byte in A
        MOV     P2, A           ;   Send to Port 2
        CLR     RI              ;  Get ready to receive next byte
        SJMP    LOOP 1          ;   Go to Loop 1, to keep getting data.
```

INTERRUPT PROGRAMMING

- An interrupt is an internal or external event that interrupts the microcontroller to inform it that a device needs its service. Every interrupt has a program associated with it called the interrupt service routine (ISR).
- The 8051 has 6 interrupts:
- Reset
- Timer interrupts :Timer 0 interrupt and Timer 1 interrupt
- External hardware interrupts : INT 0 INT 1
- Serial communication interrupt
- The 8051 can be programmed to enable or disable an interrupt and the interrupt priority can be altered. Register IE is responsible for enabling and disabling the interrupts.

Programming Timer Interrupts

- The timer flag (TF) is raised when the timer rolls over. In polling TF, we have to wait until the TF is raised.
- In problem with polling method is that the microcontroller is tied down while waiting for TF to be raised and cannot do anything else.
- Using interrupts solves this problem and avoids tying down the microcontroller.
- If the timer interrupt in the IE register is enabled, whenever the timer rolls over, TF is raised and the microcontroller is interrupted in whatever it is doing and jumps to the interrupt vector table to service the ISR.
- In this way the microcontroller can do other things until it is notified that the timer has rolled over.

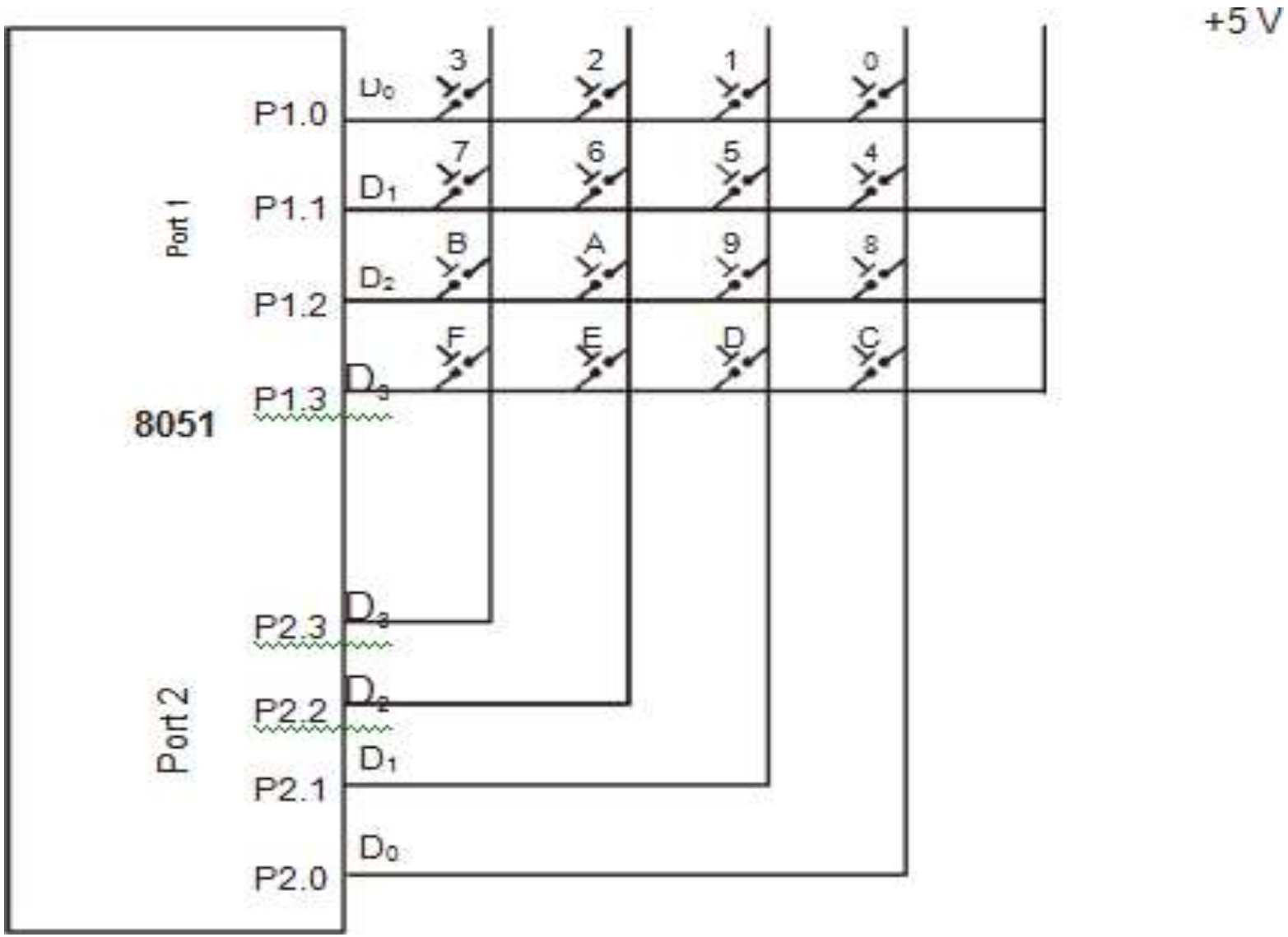

TF0
1
jumps to
Times 0 interrupt Vector
000BH
TF1
1
jumps to
Times 1 interrupt Vector
001BH

Programming External Hardware Interrupts

- The 8051 has two external hardware interrupts INT 0 and INT 1.
- Upon activation of these interrupts through Port pins P3.2 and P3.3, the 8051 gets interrupted in whatever it is doing and jumps to the interrupt vector table to perform the interrupt service routine (ISR).
- There are two types of activation for the external hardware interrupts: Level triggered and Edge triggered.

KEYBOARD INTERFACING

- The rows are connected to an output port and the columns are connected to an input port.
- When a key is pressed, a row and a column make a contact, otherwise there is no connection between rows and columns.
- 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.
- If no key has been pressed, reading the input port will yield 1s for all columns since they are connected to Vcc.


+5 V
8051
Port 1
Port 2
P1.0
P1.1
P1.2
P1.3
P2.3
P2.2
P2.1
P2.0
D0
D1
D2
D3
3
2
1
0
7
6
5
4
B
A
9
8
F
E
D
C

- If any key is pressed, the columns are scanned again and again until one of them has a 0 on it.
- After the key press detection, it waits 20 milli seconds for the bounce and then scans the columns again.
- After 20 ms delay, the key is still pressed, it goes to detect which row it belongs to. To detect the row it grounds one row at a time, reading the columns each time.
- If all columns are high, the pressed key cannot belong to that row. Therefore it grounds the next row and continues until it finds the row the key press belongs to.

- After finding the row, it sets up the starting address for the look-up table holding the ASCII codes for that row and goes to the next stage to identify the key.
- Now it rotates the column bits, one bit at a time into the carry flag and checks if it is low.
- When carry flag is zero, it pulls out the ASCII code for that key from look-up table; otherwise it increments the pointer to point to the next element of the look-up table.

Program

Write 8051 ALP to interface 4x4 matrix keyboard.

Solution :

```
ROW_1 :   MOV DPTR, #KEY1
          SJMP FIND
ROW_2 :   MOV DPTR, #KEY2
          SJMP FIND
ROW_3 :   MOV DPTR, #KEY3
FIND :    RRC A
          JNC MATCH
          INC DPTR
          SJMP FIND
MATCH :   CLR A
          MOV CA, @A +DPTR
          MOV P0, A
          MOV P1, #00H
L3 :      MOV A, P2
          ANL A, #0F H
          CJNE A, #0FH, L3
          CALL DELAY
          SJMP L2
```

ASCII-Look up table for each row

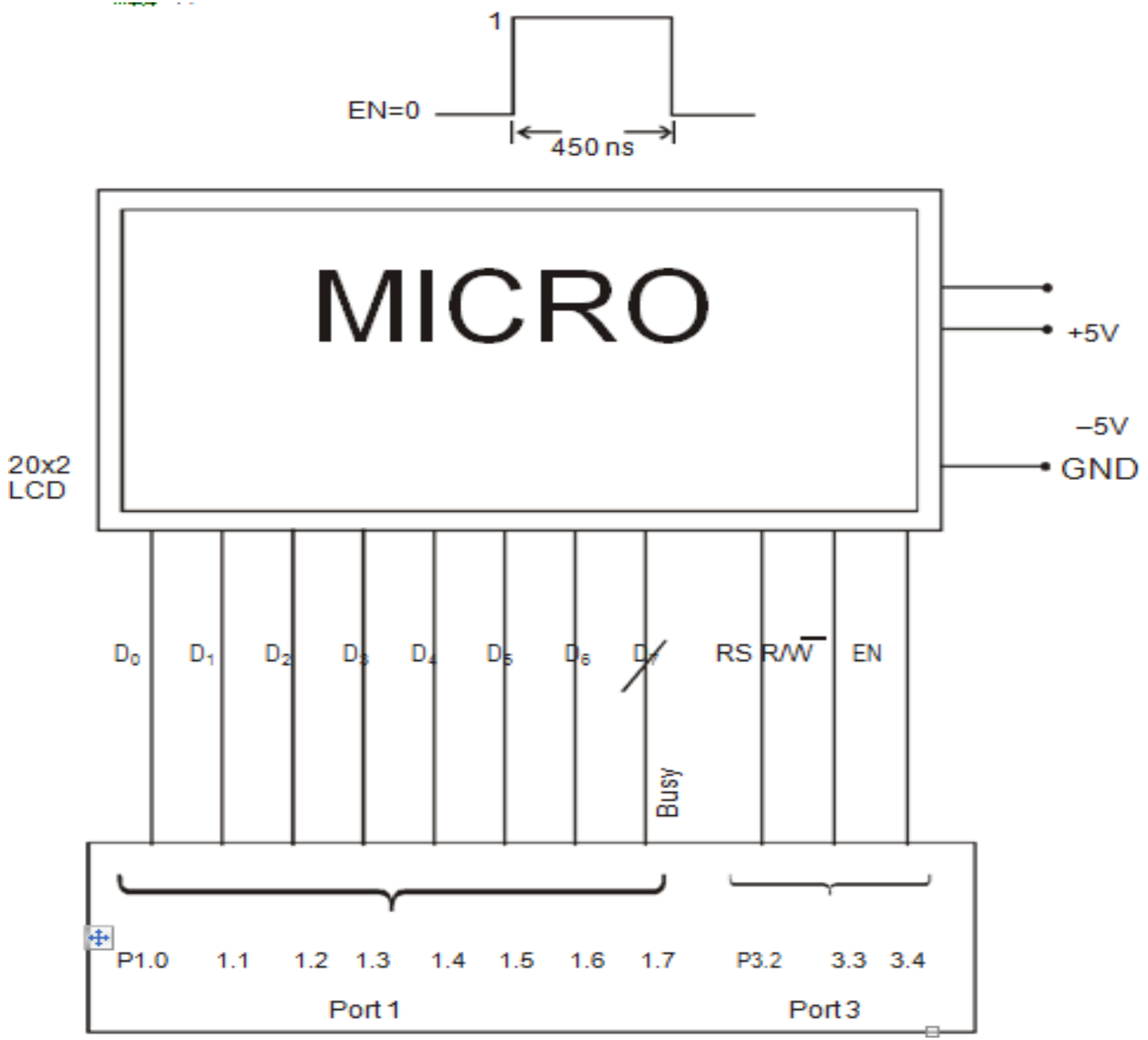
```
        ORG         3000H
KEY 0 : DB '0' : '1' : '2' : '3'
KEY 1 : DB '4' : '5' : '6' : '7'
KEY 2 : DB '8' : '9' : 'A' : 'B'
KEY3 :  DB 'C' : 'D' : 'E' : 'F'
        END
```

Program for Keyboard

```
                MOV P2, #0FF H
                MOV P1, #00H
L2:             MOV A, P2
                ANL A, #0F H
                CJNE A, #0F H, OVER
                SJMP L2
OVER:           ACALL DELAY
                MOV A, P2
                ANL A, #0F H
                CJNE A, #0FH, OVER1
                SJMP L2
OVER1:          MOV P1, #0FEH
                MOV A, P2
                ANL A, #0FH
                CJNE A, #0FH, ROW_0
                MOV P1, #0FD H
                MOV A, P2
                ANL A, #0F H
                CJNE A, #0FH, ROW_1
                MOV P1, # 0FB H
                MOV A, P2
                ANL A, #0F H
                CJNE A, #0F H, ROW_2
                MOV P1, # 0F7 H
                MOV A, P2
                ANL A, #0F H
                CJNE A, # 0FH, ROW_3
```

LCD INTERFACING

- The various types of LCD displays are, 16x2, 20x1, 20x2, 20x4, 40x2 and 40x4 LCDs. 16x2 LCD means that it having two lines, 16 characters per line.
- The 8 bit data pins (D_0–D_7) are used to send information tot he LCD or read the contents of the LCD's internal registers.
- The data lines are connected to Port 1. Register Select (RS),
- Read/Write ($_{R/W}$) and Enable (EN) plans are connected to Port 3.


1
EN=0
450 ns
MICRO
20x2
LCD
+5V
–5V
GND
D0
D1
D2
D3
D4
D5
D6
D7
RS
R/W
EN
Busy
P1.0
1.1
1.2
1.3
1.4
1.5
1.6
1.7
P3.2
3.3
3.4
Port 1
Port 3

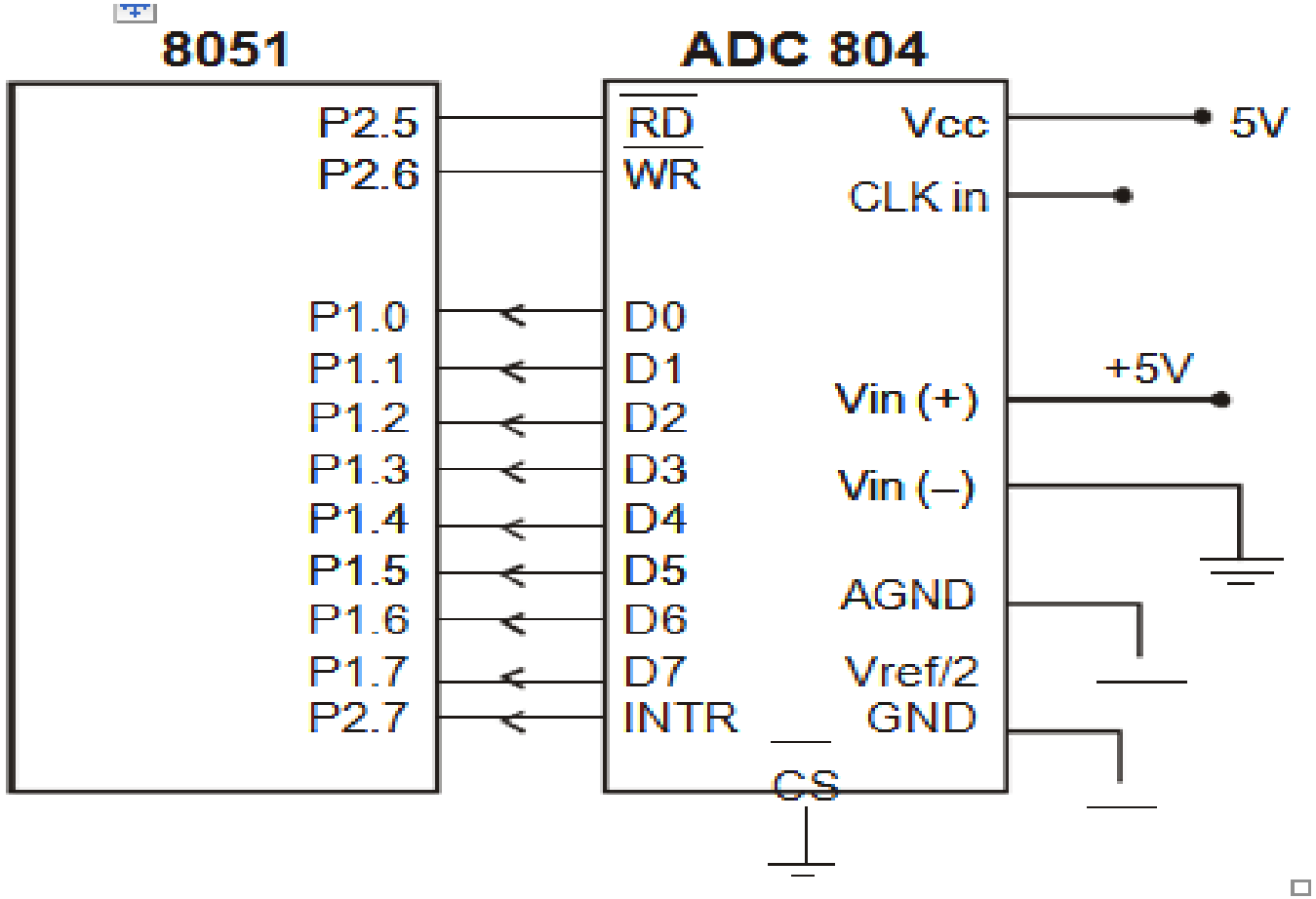
- There are two important registers are available inside the LCD. They are (i) instruction command register, (ii) data register.
- The RS pin is used to select the register. If RS=0, the instruction command code register is selected, allowing the user to send a command. If RS=1, the data register is selected, allowing the user to send a command. If RS=1, the data register is selected, allowing the user to send data to be displayed on the LCD.
- $_{R/W}$ pin is used to write information to the LCD or read information from it. EN (enable) pin is used to latch information presented to its data pins.
- When data is supplied to data pins, a high-to-low pulse must be applied to EN pin in order for the LCD to latch in the data present at the data pins.
- This pulse must be a minimum of 450 ns.

- If RS=0 and $_{R/W}$ = 0
  When busy flag (D_7)=1, the LCD is busy and will not accept any new information.
- When busy flag (D_7) = 0, the LCD is ready to receive new information.

ADC interfacing

- ADCs are used to convert the analog signals to digital numbers so that the microcontroller can read them.
- ADC [like ADC 0804 IC] works with +5 volts and has a resolution of 8 bits.
- Conversion time is defined as the time taken to convert the analog input to digital (binary) number. The conversion time varies depending upon the clock signals; it cannot be faster than 110 μs .
- Analog input is given to the pins V_{in} (+) and V_{in} (-).
- V_{in} (-) is connected to ground.
- Digital output pins are D_0 - D_7. D_7 is the MSB and D_0 is the LSB.
- There are two pins for ground, analog ground and digital ground. Analog ground is connected to the ground of the analog V_{in} and digital ground is connected to the ground of the V_{CC} pin.

- The following steps are followed for data conversion :
- Make chip select (CS) = 0 and send a low - to - high pulse to pin WR to start the conversion.
- Keep monitoring the INTR pin. If INTR is low, the conversion is finished and go to the next step. If INTR is high, keep polling until it goes low.
- After the INTR has become low, we make CS = 0 and send a high- to-low pulse to the RD pin to get the data out.

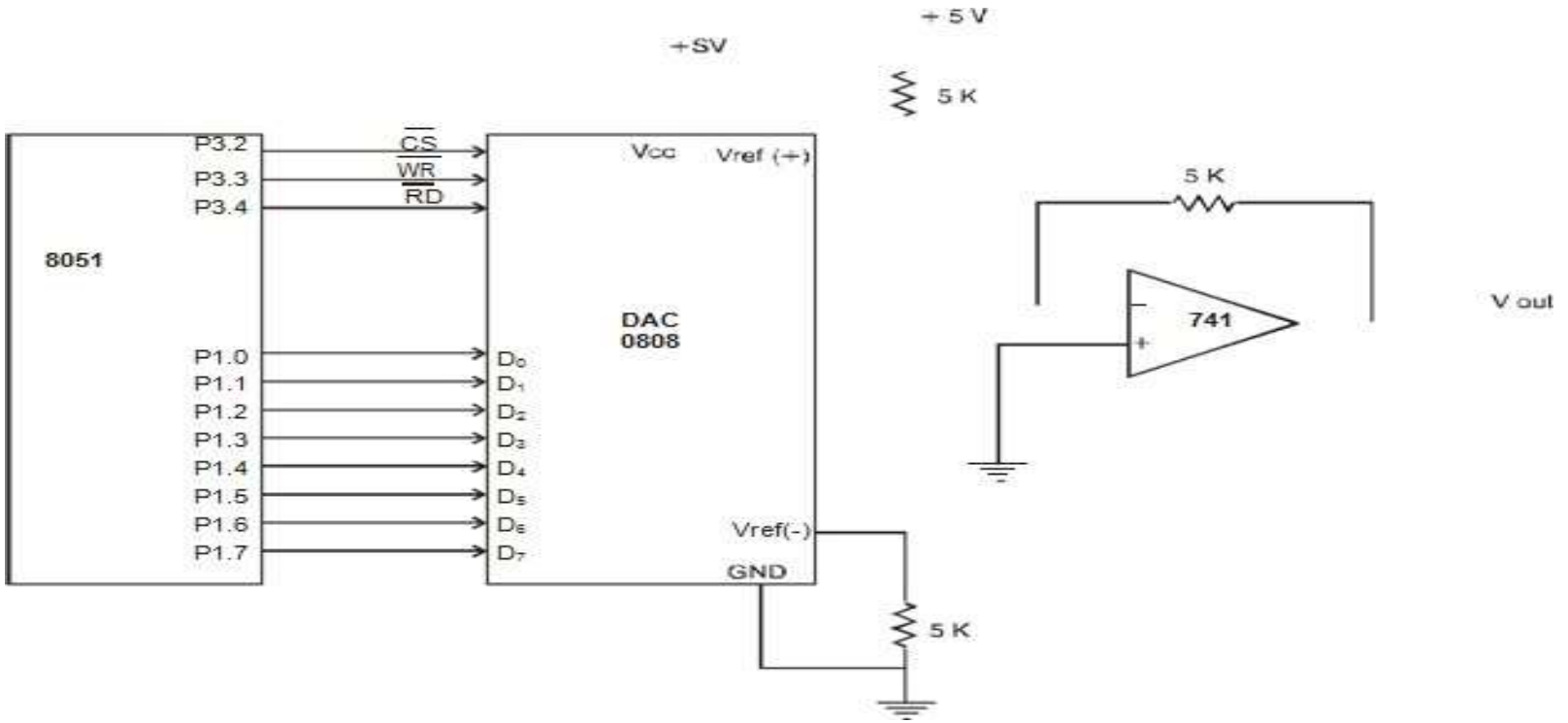

8051
ADC 804
P2.5
P2.6
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P2.7
RD
WR
D0
D1
D2
D3
D4
D5
D6
D7
INTR
CS
Vcc
CLK in
Vin (+)
Vin (–)
AGND
Vref/2
GND
5V
+5V

The program presents the concept to monitor the INTR pins and bring an analog input into register A. Then call a hex - to - ASCII conversion and data display subroutines continuously.

- P2.6 = WR (start conversion needs to low - to - high pulse)
- P2.7 = INTR, when low, end - of - conversion
- P2.5 = RD (a high-to-low will read the data from ADC chip)
- P1.0 - P1.7 = D_0 - D_7 of ADC 804

```
        MOV     P1, # 0FF H         ; make P1 = input
BACK :  CLR     P2.6                ;WR = 0
        SET B   P2.6                ;WR = 1 Low - to - high 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
        A CALL  CONVERSION          ;  hex - to - ASCII conversion
        A CALL  DATA_DISPLAY        ;  display the data
        SET B   P2.5                ;  make RD = 1 for next round
        SJMP    BACK
```

DAC INTERFACING



- The digital - to - analog converter (DAC) is used to convert digital pulses to analog signals.

The methods of creating a DAC are:

- Binary weighted
- R/2R ladder.
- Mostly R/2R method with DAC 0808 (MC 1408) is used since it can achieve a much higher degree of precision. Port 1 furnishes the digital byte to be converted to an analog Voltage and port 3 controls the conversion process.
- In DAC 0808, the digital inputs are converted to current. The total courrent provided by the I_{out} pin is a function of the binary numbers at the $D_0 - D_7$ inputs of DAC and the reference current I_{ref}.

$$I_{out} = I_{ref}\left(\frac{D_7}{2} + \frac{D_6}{4} + \frac{D_5}{8} + \frac{D_4}{16} + \frac{D_3}{32} + \frac{D_2}{64} + \frac{D_1}{128} + \frac{D_0}{256}\right)$$

Where $I_{ref} = 2$ mA.

SENSOR INTERFACING

Sensor :

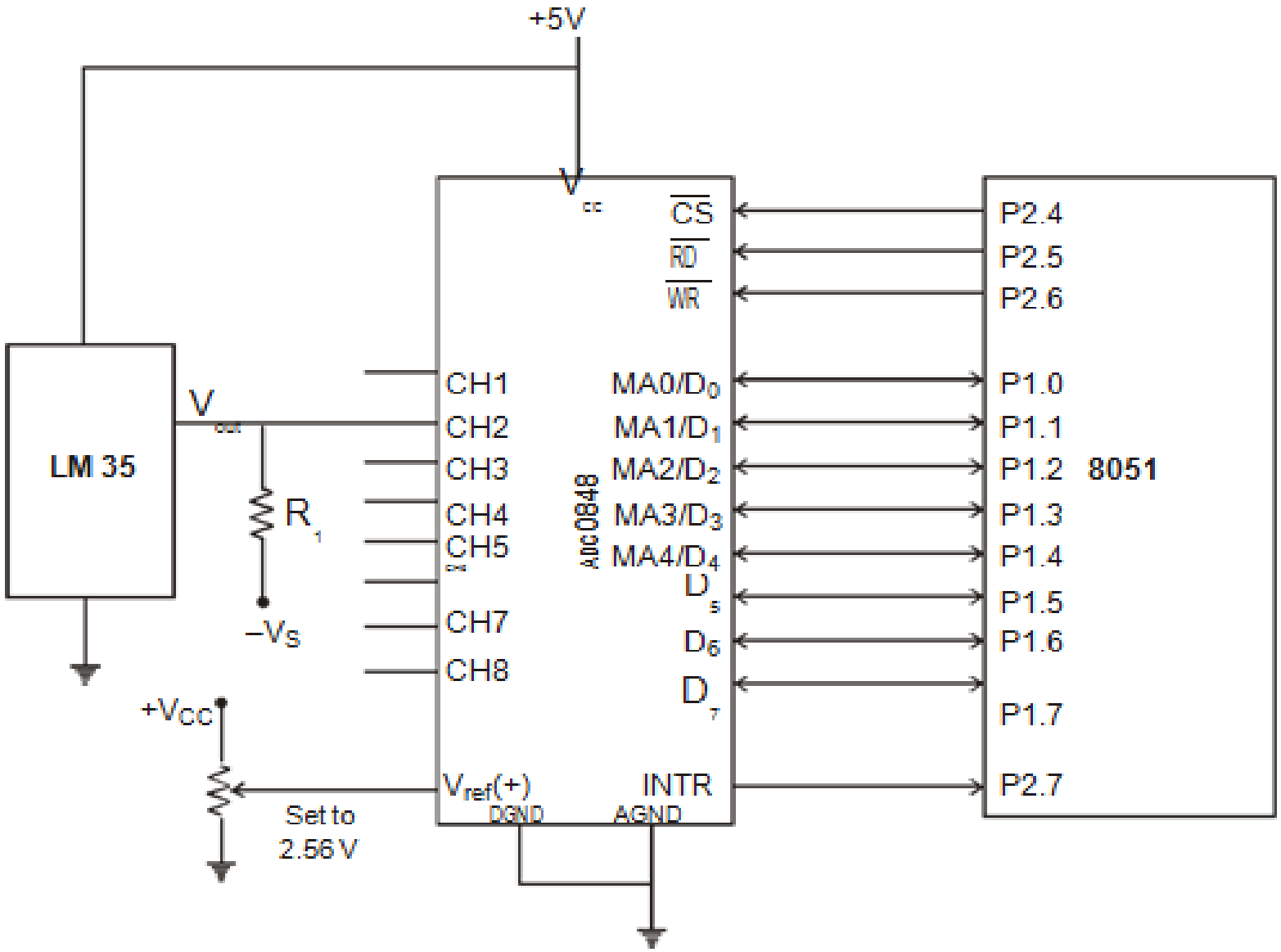
- Sensor converts the physical Pressure, Temperature or other variable to a proportional voltage or current.

Types of Sensors :

- Light Sensor
- Temperature Sensor
- Pressure Sensor
- Force Sensor
- Flow Sensor

Temperature Sensor

- There are many types of temperature sensors. Now we discuss about Semiconductor Temperature Sensor (LM 35). The LM35 series sensors are precision integrated circuit temperature sensor whose output voltage is proportional to the Celsius (centigrade) temperature.
- It outputs 10 mV for each degree of centigrade temperature. If the output is connected to a negative reference voltage V_S, the sensor will give a meaningful output for a temperature range of -55^0C to $+150^0$C. The output voltage can be amplified or filtered for a particular application.


+5V
V
cc
CS
RD
WR
P2.4
P2.5
P2.6
LM 35
V
out
CH1
CH2
CH3
CH4
CH5
CH7
CH8
MA0/D0
MA1/D1
MA2/D2
MA3/D3
MA4/D4
D5
D6
D7
ADC0848
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
8051
R1
–Vs
+Vcc
Set to
2.56 V
Vref(+)
DGND
AGND
INTR
P2.7

EXTERNAL MEMORY INTERFACING

- When the data is located in the code space of 8051, MOVC instruction is used to get the data, where ‘C’ stands for code.
- When the data memory space must be implemented externally, MOVX instruction is used, where ‘X’ stands for external.

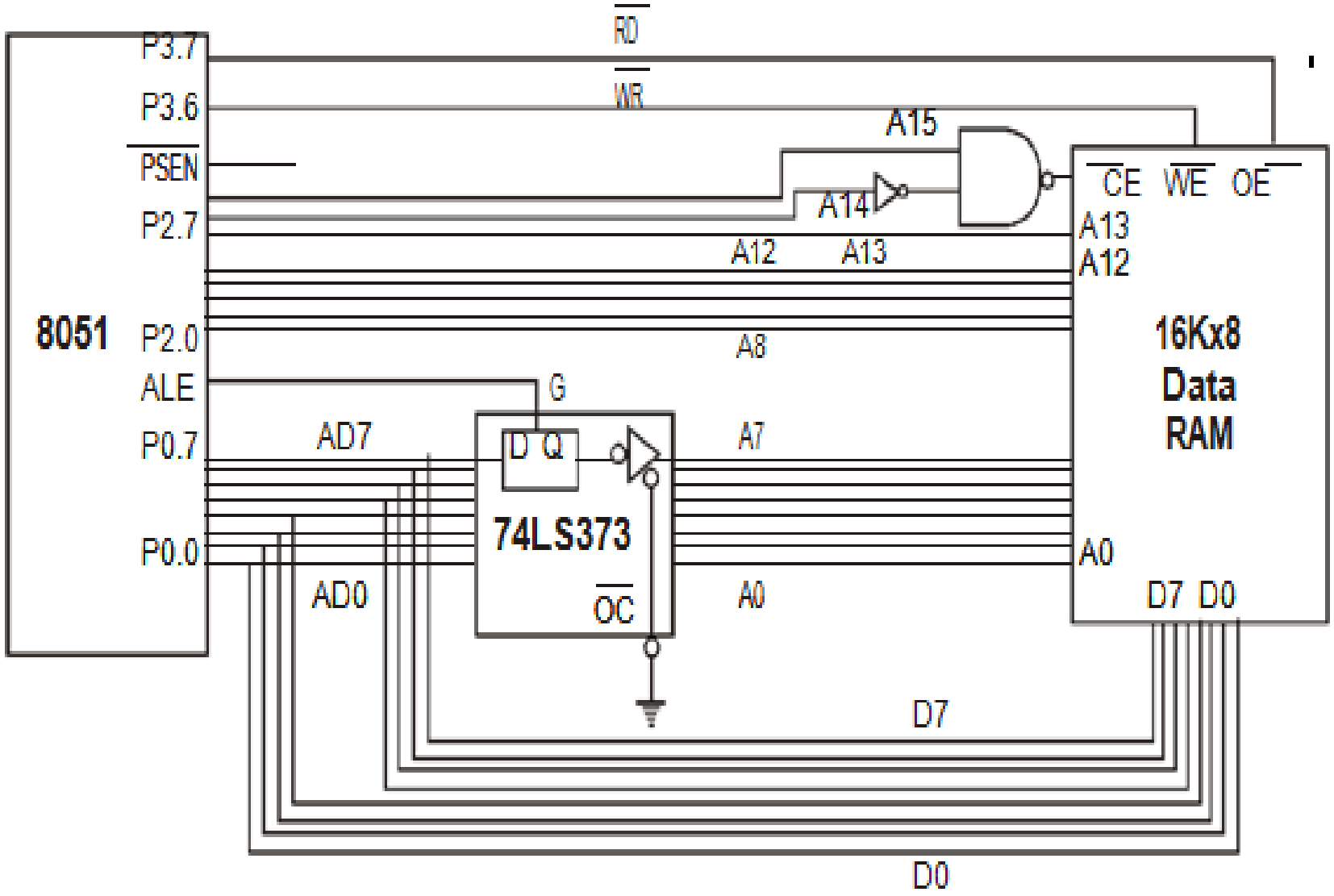
External data RAM interfacing

- To connect the 8051 to an external SRAM, we must use both RD (P3.7) and WR (P3.6).
- In writing data to external data RAM, the instruction “MOVX @DPTR, A” is used, where the contents of register A are written to external RAM whose address is pointed to by the DPTR register.

Program:

Write a program to read 200 bytes of data from Port 1 and save the data in external RAM starting at RAM location 5000H.

```
RAMDATA  EQU   5000H
COUNT    EQU   200
         MOV   DPTR, # RAMDATA ; pointer to external NV-RAM
         MOV   R3, #COUNT      ; counter
AGAIN :  MOV   A, P1           ; read data from P1
         MOVX  @DPTR, A        ; save it external NV-RAM
         ACALL DELAY           ; wait before next sample
         INC   DPTR            ; next data location
         DJNZ  R3, AGAIN       ; until all are read
HERE :   SJMP  HERE            ; stay here when finished
```


8051
P3.7
P3.6
PSEN
P2.7
P2.0
ALE
P0.7
P0.0
RD
WR
A15
A14
A12
A13
A8
G
AD7
A7
D Q
74LS373
OC
AD0
A0
CE WE OE
A13
A12
16Kx8
Data
RAM
A0
D7 D0
D7
D0

STEPPER MOTOR INTERFACING

- A stepper motor is a widely used device that translates electrical pulses into mechanical movement. In applications such as disk drives, dot matrix printers and robotics the stepper motor is used for position control. Every stepper motor has a permanent magnet rotor surrounded by four stator windings, that are paired with a center-tapped common.
- The center tap allows a change of current direction in each of two coils when a winding is grounded, thereby resulting in a polarity change of the stator. The stepper motor shaft runs in a fixed repeatable increment which allows one to move it to a precise position.
- This repeatable fixed movement is possible as a result of basic magnetic theory where poles of the same polarity repel and opposite poles 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, the polarity is also changed causing the reverse motion of the rotor As the sequence of power is applied to each stator winding, the rotor will rotate. There are several used sequences where each has a different degree of precision.

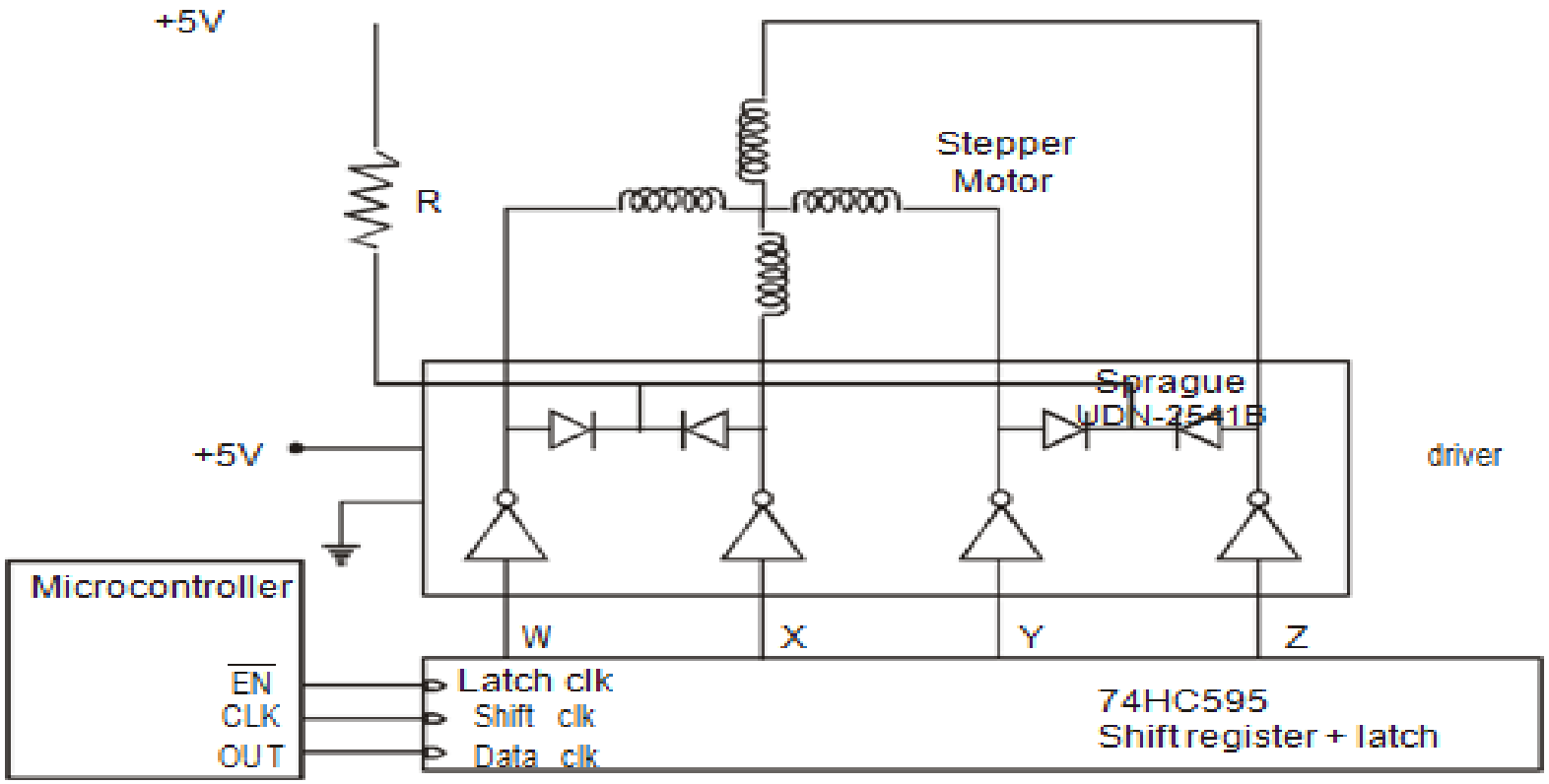


Fig 5.20 Drive circuitry for a stepper motor

The movement of the stepper motor with a single step is depends on the internal construction of the motor, in particular the number of teeth on the stator and the rotor. The step angle is the minimum degree of rotation associated with a single step. Various motors have different step angles. Table 5.3 shows some step angles for various motors.

Steps per revolution = Total number of steps needed to rotate one complete rotation or 360 degrees.

$$\text{Steps per second} = \frac{\text{RPM} \times \text{Steps per revolution}}{60}$$

WAVEFORM GENERATION:

- Steps to generate sine wave on 8051 microcontroller.
- Generate digital values of sine wave on a port that is 8 bit binary value.
- Convert that digital value into analog value to take that 8 bit output on 1 pin.
- Generated sine wave is in steps hence to obtain a pure sine wave, pass it through low pass filter. Thus by remove high frequency part, obtain smoother sine wave.
- First, generate digital values for sine wave. For this example take 16 points in 1 cycle. Thus 1 value will hold for 1/16th of 360 degree. Hence use sine(360 * (i/16)) where i runs from 0 to 15.
- This will cover 16 equally spaced points in one cycle. Place this cycle in while (1) loop so that will get continuous sine wave.
- In a cycle of sine wave, half cycle is positive and remaining half cycle is negative. Since microcontroller cannot have negative voltage, will shift sine wave to half of maximum value.
- As maximum value is 255 for 8 bits, half of it is 127.5.Thus digital value to be assigned to port is 127.5 + 127.5 * sine(360*(i/16)) where i runs from 0 to 15. Here minimum value is 127.5 - 127.5 = 0 and maximum value is 127.5 + 127.5 = 255
- Hence sine wave will be between 0 and 255 and which can be assigned to port. Since most of the values will come in fraction, have to round figure to assign integer value.

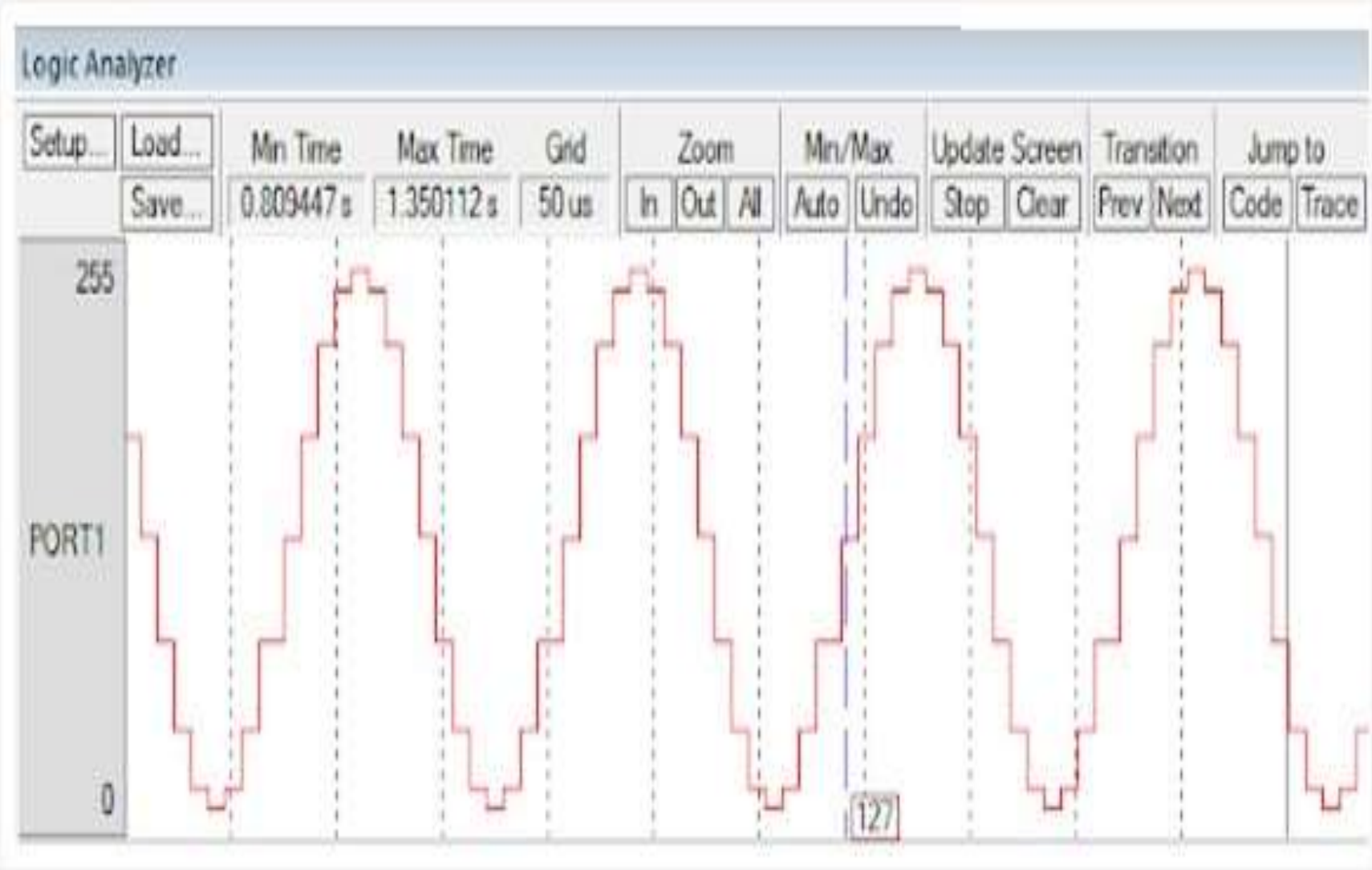
Program:

```
#include<reg51.h>

int main(void)
{
  //Digital values of sine wave
  unsigned char x[16]={127,176,218,245,255,245,218,176,128,79,37,10,0,10,37,79};

  unsigned char i;


  while(1)
  {
    for(i=0;i<16;i++)
    {
      P1=x[i];
    }
  }
}
```


Logic Analyzer
Setup...
Load...
Save...
Min Time
0.809447 s
Max Time
1.350112 s
Grid
50 us
Zoom
In
Out
All
Min/Max
Auto
Undo
Update Screen
Stop
Clear
Transition
Prev
Next
Jump to
Code
Trace
255
PORT1
0
127

COMPARISON OF MICROPROCESSOR, MICROCONTROLLER, PIC AND ARM PROCESSORS

Microprocessor

- Microprocessor has only a CPU inside them in one or few Integrated Circuits. Like microcontrollers it does not have RAM, ROM and other peripherals. They are dependent on external circuits of peripherals to work. But microprocessors are not made for specific task but they are required where tasks are complex and tricky like development of software's, games and other applications that require high memory and where input and output are not defined. It may be called heart of a computer system. Some examples of microprocessor are Pentium, I3, and I5 etc.

Microcontroller

- A micro-controller can be comparable to a little stand alone computer; it is an extremely powerful device, which is able of executing a series of pre-programmed tasks and interacting with extra hardware devices. Being packed in a tiny integrated circuit (IC) whose size and weight is regularly negligible, it is becoming the perfect controller for as robots or any machines required some type of intelligent automation.
- A single microcontroller can be enough to manage a small mobile robot, an automatic washer machine or a security system. Several microcontrollers contains a memory to store the program to be executed, and a lot of input/output lines that can be a used to act jointly with other devices, like reading the state of a sensor or controlling a motor.8051 microcontroller is an 8-bit family of microcontroller is developed by the Intel in the year 1981.

PIC Microcontroller

- Peripheral Interface Controller (PIC) is microcontroller developed by a Microchip, PIC microcontroller is fast and simple to implement program when we contrast other microcontrollers like 8051. The ease of programming and simple to interfacing with other peripherals PIC become successful microcontroller. Microcontroller is an integrated chip which is consists of RAM, ROM, CPU, TIMER and COUNTERS.
- The PIC is a microcontroller which as well consists of RAM, ROM, CPU, timer, counter, ADC (analog to digital converters), DAC (digital to analog converter). PIC Microcontroller also support the protocols like CAN, SPI, UART for an interfacing with additional peripherals. PIC mostly used to modify Harvard architecture and also supports RISC (Reduced Instruction Set Computer) by the above requirement RISC and Harvard we can simply that PIC is faster than the 8051 based controllers which is prepared up of Von-Newman architecture.

ARM Processor

- An ARM processor is also one of a family of CPUs based on the RISC (Reduced Instruction Set Computer) architecture developed by Advanced RISC Machines (ARM). An ARM makes at 32-bit and 64-bit RISC multi-core processors. RISC processors are designed to perform a smaller number of types of computer instructions so that they can operate at a higher speed, performing extra millions of instructions per second (MIPS).
- By stripping out unnecessary instructions and optimizing pathways, RISC processors give outstanding performance at a part of the power demand of CISC (complex instruction set computing) procedure. ARM processors are widely used in customer electronic devices such as smart phones, tablets, multimedia players and other mobile devices, such as wearables. Because of their reduced to instruction set, they need fewer transistors, which enable a smaller die size of the integrated circuitry(IC).