LECTURE NOTES (E- CONTENTS) for

B.Sc. II Electronics (2021-22)

Semester: IV Paper- V DSE 1005D

Advance Communication and Microcontroller 8051

Section II: Microcontroller 8051

Unit- 3: Timers, Serial port and Interrupts (Assembly) programming of 8051:

Prepared and Circulated for

B .Sc. II Electronics Students

BY

Dr. C. B. Patil

Associate Professor, Department of Electronics

Vivekanand College (Autonomous), Kolhapur

(For Private Circulation only)

UNIT 3

Timers, Serial port and Interrupts (Assembly) programming of 8051

3.1 Introduction Timers facility of 8051:

The microcontroller 8051 has two 16 - bit timer/counter registers namely Timer 0 and Timer 1. Both these registers can be configured independently to operate as a timer or as event counter. When used as a timer the register is programmed to count the internal clock pulses. The internal clock pulses are generated from a constant clock generator. The count is loaded in the register gives a constant time. The register is incremented every machine cycle. One machine cycle consist of 12 oscillator period and so the counting rate is 1/12 of the oscillator frequency.

When it is used as a counter, the microcontroller is programmed to count external pulses. The register is in oriented in a response to high to low transition of the corresponding external input pin T_0 and T_1. The external input does not have a constant frequency and hence it is not used for timing reference.

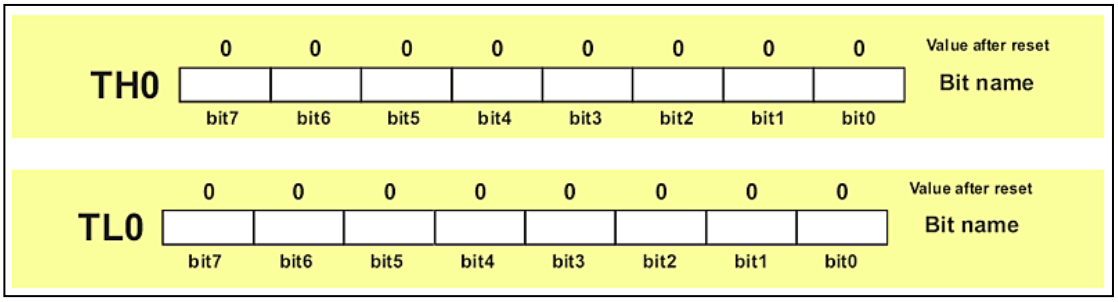


Figure 4.1 Timer Register

The timer/ counter registers are divided into two 8 bit registers called as timer low byte i.e. TL_0 and TL_1 and timer high i.e. TH_0 & TH_1. TL_0 and TH_0 together form the 16 - bit Timer 0. and TL_1 and TH_1 together forms the 16-bit Timer 1.

4.2 TMOD Register :-

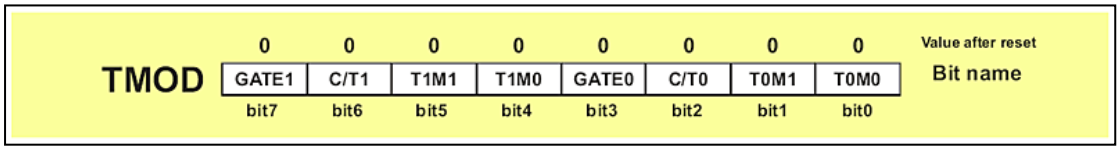


Figure 4.2 TMOD Register

The timer/ counter action is controlled by the bits in the timer mode control register i.e. TMOD the figure 4.2 shows TMOD Register.

i) GATE (Gating control) :- When gate is set to logic 1 then timer/counter is enabled only with INT_X is high and TR control pin is set. When cleared (set to logic 0) the timer is enable whenever TR control bit is set (logic 1). Gate control bit is used to measure pulse width of the pulse on INT pin.

ii) C/$\overline{T}$ (Timer or counter selector) :- When this bit is set to 1, timer act as a counter; when it is cleared, timer is used for timing applications.

iii) M_0 and M_1 :- These bits select the particular mode in which timer will operate.

Mode 0 :- 13 bit timer / counter mode. 8 bit are stored in the register & 5- bits are stored in a TL register.

Mode 1 :- It is 16-bit timer/counter is with TH and TL are cascaded.

Mode 2 :- It is 8-bit auto reload timer/counter. TH holds the value which is to be reloaded into TL register each time it overflows.

Mode 3 :- (Split timer mode) :- In this mode only timer can be configured. TL0 is as 8-bit timer/ counter controlled by standard Timer 0 Control bits. TH0 is as 8-bit timer controlled by Timer 1 control bits.

4.3TCON Register :-

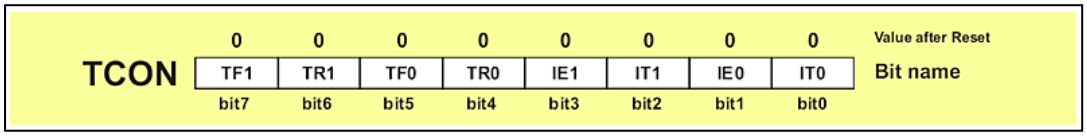


Figure 4.3 TCON Register

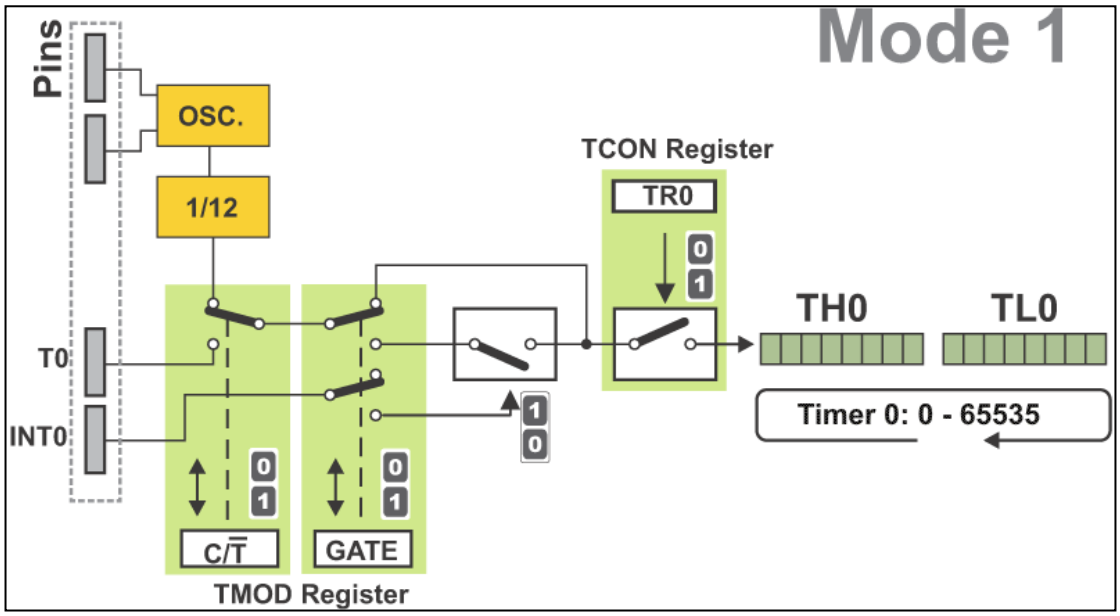
1) TF1 :- Timer 1 overflow flag set by hardware on timer/counter overflow. Cleared by Hardware when processor vectors to interrupt subroutine.
2) TR1 :- Timer 1 run control bit. Set/ Clear by software to turn timer/counter on or off.
3) TF0 :- Timer 0 overflow flag set by hardware on timer/counter overflow. Cleared by Hardware when processor vectors to interrupt subroutine.
4) TR0 :- Timer 0 run control bit. Set/clear by software to run timer/counter on of off.
5) IE1 :- Interrupt 1 edge flag. Set by hardware when which at interrupt edge is detected. Cleared when interrupt is processed.
6) IT1(Interrupt 1 Time control bit) set / clear by software to specify falling edge of level trigger interrupt.
7) IE0 (Interrupt 0 edge flag) set by hardware when external interrupt edge is detected cleared when interrupt is processed.
8) IT0 (Interrupt 0 time control bit) set/ clear by software to specify falling edge of level trigger interrupt.

4.4 Time Modes of Operation:

Depending on the mode control bits M1 and M0 in the timer/counter mode control register (TMOD) can be operate in any one of four modes namely mode 0, mode 1, mode 2 and mode 3.

4.4.2 Mode 1 :-

Mode 1 of the timer/ counter is same as the mode 0. The only difference is that in a mode 0 the timer/ counter was 13-bit but in a mode 1 is 16-bit timer/ counter. The timer is enabled when TR = 0, Gate = 0 and, INTx = 1. If the Gate = 1 then the timer is controlled by external input INTx to allow pulse width measurement. TR is a control bit in the TCON register.



igure 5.5 Timer mode 1

Algorithm for mode 1 programming

1.Load the TMOD register with a value indicating which timer (timer 0 or timer 1) is to be used and which timer mode (0 or 1) is selected

2. Load registers TL and TH with initial count value

3. Start the timer

4. Keep monitoring the timer flag (TF) with the *JNB TFx, target* instruction to see if it is raised. Get out of the loop when TF becomes high

5. Stop the timer

6. Clear the TF flag for the next round

7. Go back to Step 2 to load TH and TL again

❖ In the following program, we create a square wave of 50% duty cycle (with equal portions high and low) on the P1.6 bit. Timer 0 is used to generate the time delay.

The timer works with a clock frequency of 1/12 of the XTAL

11.0592 MHz / 12 = 921.6 kHz as the

T = **1/921.6kHz = 1.085us.**

Delay = number of counts × 1.085us.

The number of counts for the roll over is

FFFFH – FFF2H = 0DH (13decimal +1 clock extra).

14 × 1.085us = 15.19us for half the pulse.

For the entire period it T = 2 × 15.19us = 30.38 us

Program

```
        MOV TMOD,#01 ;Timer 0, mode 1(16-bit mode)
HERE:   MOV TL0,#0F2H ;TL0=F2H, the low byte
        MOV TH0,#0FFH ;TH0=FFH, the high byte
        CPL P1.6 ;toggle P1.6
        ACALL DELAY
        SJMP HERE
DELAY:  SETB TR0 ;start the timer 0
AGAIN:  JNB TF0, AGAIN ;monitor timer flag 0 until it rolls over
        CLR TR0 ;stop timer 0
        CLR TF0 ;clear timer 0 flag
        RET
```

iii) Mode 2 :-

4.4.3 Mode 2: In this mode the timer/ counter resister is configured as an 8-bit counter with auto reload facility. TL_x acts as a basic timer/ counter. In auto reload mode TH is loaded with the count. When the timer starts, it keeps on incrementing and when it overflows, this will

set the timer interrupt flag TF and it will also reloads the contest of TH_x to TL_X. The reloading operation will not alter the content value in TH_x register.

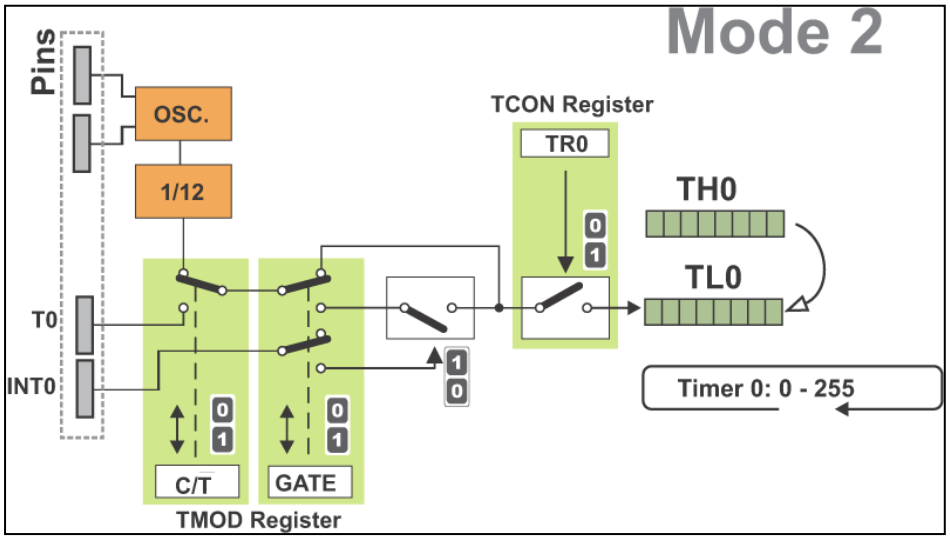


Figure 5.6 Timer mode 2

Algorithm for Mode 2 programming

1.Load the TMOD value register indicating which timer (timer 0 or timer 1) is to be used and which timer mode (0 or 1) is selected
2. Load registers TL and TH with initial count value
3. Start the timer
4. Keep monitoring the timer flag (TF) with the JNB TFx, target instruction to see if it is raised Get out of the loop when TF becomes high
5. Stop the timer
6. Clear the TF flag for the next round
7. Go back to Step 4

❖ In the following program, we create a square wave of 1.8 KHz with 50% duty cycle (with equal portions high and low) on the P1.2 bit. Timer 0 is used to generate the time delay.
Soln. F=1.8KHz T=1/F T=544 t=T/2 Since it is a 50% duty cycle square wave, the period T is twice that
t=544/2= 272
11.0592 MHz / 12 = 921.6 kHz as the
T = 1/921.6kHz = 1.085us.
count =272/1.085us=250 counts
so 255-250=05H TH should be loaded with 05H

Program:

```
        MOV TMOD,#20H ;T1/8-bit/auto reload
        MOV TH1,#05H   ;TH1 = 5
        SETB TR1       ;start the timer 1
BACK:   JNB TF1,BACK   ;till timer rolls over
        CPL P1.2       ; compliment  P1.0
        CLR TF1        ;clear Timer 1 flag
        SJMP BACK      ;mode 2 is auto-reload
```

register uses the timer 0 control bits i.e. C/T, Gate TR0, $\overline{INT}_0$ and TF0. TH0 counts the machine cycles. The TH0 register uses the timer 0 control bits i.e. C/T, Gate TR1, $\overline{INT}_1$ and TF1.

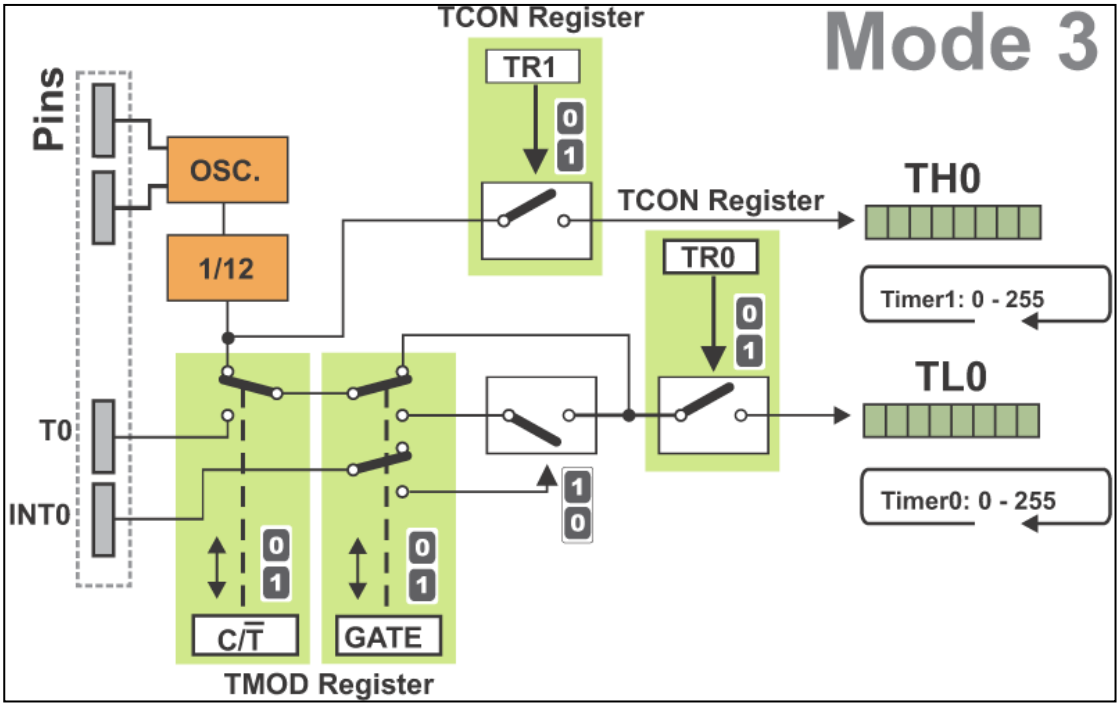


Figure 5.7 Timer mode 3

Introduction of Serial communication facility of 8051

In this section the serial communication facility of 8051 is discussed. The 8051 has an on chip UART (Universal Asynchronous Reception Transmission) port used for serial data transmission and reception with different speeds (baud rates). Two pins of the 8051 are available for serial communication. The RXD(P3.0) pin is used for serial data reception and the TXD (P3.1)pin is used for serial data reception.

Modes of Data Transmission/Reception

There are two basic modes of data transmission and reception: parallel and serial.

In parallel mode, more than one bits are transmitted/received at a time. Therefore, it speed of data transmission/reception is high. It uses more

number of lines. This mode is useful for short range data transmission/reception. Using P0, P1, P2 and P3 8051 transmit/receive parallel 8-bit data.

Serial communication on the other hand, uses only one or two data lines to transfer data. In serial communication the data is sent one bit at a time. It is generally used for long distance communication. Using pins TXD and RXD, 8051 transmit/receive serial data.

Serial communication uses two methods for data transmission/reception.

1) Asynchronous data transmission: used for character-oriented transmissions

2) Synchronous data transmission: used for block-oriented data transfers

Asynchronous serial communication and data framing

In the asynchronous method, each character is placed between start and stop bits. This process is called as data framing. The start bit is always one bit, but the stop bit can be one or two bits. The start bit is always a 0 (low) and the stop bit(s) is 1 (high). For example, look at Fig.1. in which the ASCII character "A" (8-bit binary 0100 0001) is framed between the start bit and a single stop bit. Here LSB is sent out first.

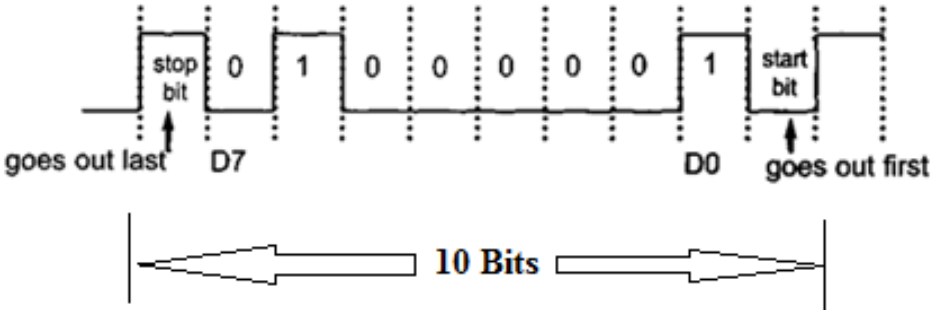


Fig. 1. Asynchronous data framing

Baud Rate

In serial communication, data is transmitted 1 bit at a time. The number of bits transmitted per second known as Baud Rate and its unit is bps (bit/sec). The standard baud rate are defined by the IBM PCXT and it is 110, 150, 300, 600, 1200, 2400, 4800, 9600 etc. The 8051 can be useful to transfer and receive serial data at different baud rates using software instructions. The timer1 mode2 (8-bit auto reload) mode of 8051 is used to set the baud rate in serial communication.

Steps for calculate baud rate for serial communication in 8051

- The 8051 microcontroller transfers and receives data serially at different baud rates. The baud rate in the 8051 is programmable and can be set using Timer 1 mode 2.
- To generate standard baud rate, external crystal of the frequency 11.0592 MHz is used. So it can generate a clock frequency of 11.0592 MHz.
- The machine cycle is calculated by dividing clock frequency by 12. Here machine cycle is 921.6 KHz.
- Before it is used by Timer 1 to set the baud rate, the 8051 UART circuitry divides the machine cycle frequency of 921.6 kHz by 32. Therefore, 921.6 kHz divided by 32 gives baud frequency of 28,800 Hz.
- When Timer 1 is used to set the baud rate it must be programmed in mode 2 (8-bit auto reload). To get baud rates compatible with PC, we must load TH1 with any one values for the given baud rate as shown in table 1.

Sr. No.	Baud Rate	TH1(Decimal)	TH1(Hex)	Baud Frequency
1	9600	-3	FDH	28,800/3
2	4800	-6	FAH	28,800/6
3	2400	-12	F4H	28,800/12
4	1200	-24	E8H	28,800/24

Table 1: TH1 values and Baud Rate

- These values of baud rate are compatible with PC's COM port.

SBUF register

It is an important 8-bit special function register used for serial communication in the 8051. During transmission of the serial data it must be placed in SBUF register. Similarly during reception of the data, SBUF holds received data. The SBUF register can be accessed like any other register in the 8051. E.g. 1) SBUF = 'A' ; Load SBUF register with ASCII of 'A' character

2) mybyte = SBUF ;save value from SBUF into variable mybyte

During transmitting data via TXD pin SBUF frames the data by adding start and stop bit. During reception of the data using RXD pin SBUF deframes the data by eliminating start and stop bit. The framing and deframing of the data is depends on the selected mode of serial communication.

SCON register

The SCON **(serial control) register** is an 8-bit bit as well as byte addressable register of 8051. It is used during programming of serial communication. Fig 2. shows the SCON register internal bit structure.

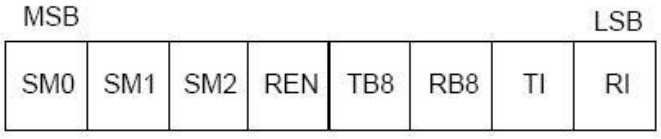


Fig. 2: SCON register

SM0, SM1

SM0 and SM1 are bit number D7 and D6 of the SCON register respectively. These two bits determine the modes of serial data transmission/reception. Table 1. 2 shows SM0, SM1 bits and selected mode name

SM0	SM1	Mode	Name
0	0	Mode 0	Shift Register
0	1	Mode1	8-bit Standard UART 1 start and one stop bit
1	0	Mode2	9-bit UART (Baud rate Fixed)
1	1	Mode3	9-bit UART (Baud rate Variable)

Table 2: bits of SCON register

The mode 1 is standard and it is mostly widely used mode than other.

SM2

SM2 is the D5 bit of the SCON register. It is multiprocessing communication bit. This bit enables/disables the multiprocessing capability of the 8051. It is mostly useful in mode 2 and mode 3 serial communication mode.

REN

REN (receive enable) is D4 bit of the SCON register. This bit is important in serial data reception. If it is '1', it allows to receive data through the RXD pin. If it is '0' the data reception disabled. If we want both transmission and reception REN must be '1'.

TB8*

The TB8 is bit D3 of SCON register. If we select mode 2 or mode 3 it is 8^{th} bit, because total 9 bits are transmitted in mode 2 and mode 3.

RB8*

The RB8 is bit D2 of SCON register. If we select mode 2 or mode 3 it stored 8^{th} bit ,because total 9 bits are received in mode 2 and mode 3.

TI

TI (transmit interrupt) is bit D1 of the SCON register. If this bit is set to '1'indicates data transmission is over and it is ready to transfer next byte of data. For next data transmission it must be cleared to '0'.

RI

RI (receive interrupt) is the D0 bit of the SCON register. If this bit is set to '1' indicates data reception is over. After removing start and stop data is placed in SBUF register and it is ready to receive next byte of data. For next data reception it must be clear to '0'.

*In mode 2 and mode 3 the total 11 bits (1 start bit +1 bit TB8/RB8+ 8 bit SBUF+ 1 stop bit) are transmitted/received.

Modes in serial communication

The mode of serial communications are set by using SM0 and SM1 bit of SCON register (Please see table 1.2). There are four modes in serial communication data transmission and reception in 8051 which are given below.

a) Mode 0: It is a Shift Register Mode. It is half duplex mode. In this RXD pin is used for data transmission and reception. The TXD pin is used as a clock out pin. The total 8 bits are transmitted/received. The baud frequency is fixed and it is F/12. Where F is the crystal frequency.

b) Mode 1: It is an 8-bit standard UART mode. It is full duplex mode. In this RXD pin is used for data reception. The TXD pin is used data transmission. The total 10 bits (1 start bit + 8 bit data + 1 stop bit) are transmitted/received. The baud frequency is variable. The timer 1 mode 2 (8-bit auto reload) is used to generate baud rate.

c) Mode 2: It is a 9-bit standard UART mode (baud frequency fixed). The total 11 bits [1 start bit+TB8/RB8+ 8 bit data(SBUF) +1 stop bit] are

transmitted/received. The baud frequency is fixed and it is F/32 or F/64. Where F is crystal frequency.

c) Mode 3: It is a 9-bit standard UART mode (baud frequency variable). The total 11 bits (1 start bit+TB8/RB8+ 8 bit data (SBUF)+1 stop bit) are transmitted/received in this mode. The baud frequency is variable. The timer 1 mode 2 (8-bit auto reload) is used to generate baud rate.

RS232 standards

A RS232 standard is an interfacing standard developed by the Electronics Industries Association (EIA) for the compatibility among all the serial data communication equipments. Now it is most widely used serial I/O interfacing standard. This standard is used in PCs and numerous types of equipment's. But this standard is not compatible with input and output voltage levels of TTL. The logic levels of the RS232 is as follow,

- '1' is represented by -3 to -25 V,
- '0' bit is +3 to +25 V,
- undefined -3 to +3V

For this reason, we cannot connect any RS232 standard equipment to the microcontroller system and vice versa. It is required to convert voltage levels TTL into RS232 and vice versa. A MAX232, converts the TTL logic levels into the RS232 voltage levels and vice versa. MAX232 chip is commonly referred as line drivers. It converts RS232 into TTL and vice versa.

MAX232

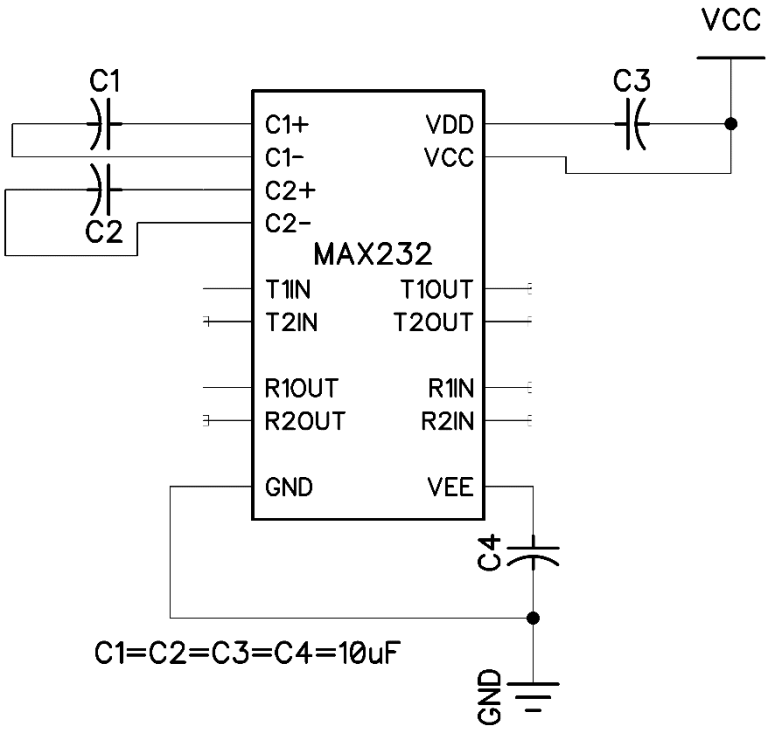


Fig. 3. Pin Connection MAX232

The RS232 standards are not compatible with TTL. So, a line driver required which can convert RS232 voltage levels and vice versa. The MAX232 is a line driver which can converts RS232 to TTL and vice versa. The pin connection of the MAX232 is shown in fig 1.2. It is 16 pin IC. It has two set of line driver for transmitting and receiving. T1 and T2 for the transmission and R1 and R2 for the reception. It converts RS232 into TTL and vice versa. The interfacing of MAX232 with 8051 microcontroller is shown in fig.4.

The serial port TXD pin of the 8051 is connected to the T1IN pin of the MAX232 and the serial port RXD pin of the 8051 is connected to the R1OUT pin of the MAX232. The MAX232 pins T1OUT and R1INis connected to DB9 connector pin 2 and 3 respectively. Through this connector we can connect a serial port of computer. It also required four capacitor C1, C2, C3 and C4 ranging from 1µF to 22 µF. MAX232 required only +5V supply.

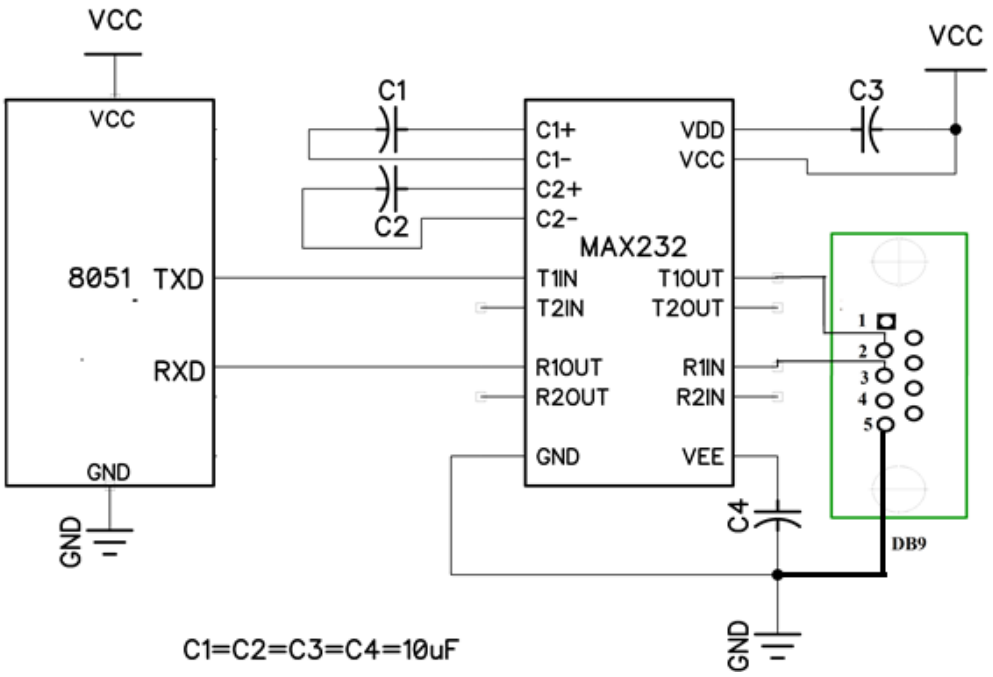


Fig 4: Interfacing MAX232 with 8051

Steps for programming the 8051 to transfer data serially

The following steps must be taken during serial data transmission,

1. Load the TMOD register with the value 20H. It uses Timer1 in mode 2 (8-bit auto-reload) to set the baud rate.
2. Load the TH1 register with the values shown in table 1 to set the baud rate for serial data transmission.
3. Load the SCON register with the value 50H, indicating serial mode 1 (8-bit standard UART mode) and reception is enabled.
4. Start the Timer 1 by setting TR1 bit to '1'.
5. Load the SBUF register with character byte to be transferred serially.
1. Monitor the TI flag bit using instruction ***while (TI==0);*** to see transmission is over. [TI flag is set to '1' during the transfer of the stop bit indicating that the last character is transmitted and it is ready to transfer the next character].
6. Clear the TI flag using instruction ***TI=0;*** for next data transmission.

7. Goto step number 5 for next data transmission.

Ex. 1 Write a C program for 8051 to transfer the letter "A" serially at 4800 baud continuously. Use 8-bit data and 1 stop bit.

Solution:

```
#include <reg51.h>
void main(void)
{
TMOD=0x20;    //use Timer 1, mode 2
TH1=0xFA;     //4800 baud rate
SCON=0x50;    // serial mode 1 and reception is enabled
TR1=1;        //start the timer
while (1)     //repeat forever
 {
SBUF='A';     //place value in buffer
while (TI==0); // monitor the TI flag bit
TI=0;         // clear the TI flag
}
}
```

Ex. 2 Write a program to transfer the message "YES" serially at 9600 baud, 8-bit data, with 1 stop bit.

```
#include <reg51.h>
void SerTx(unsigned char);      //user defined function named as SexTx
void main(void)
{
TMOD=0x20;      //use Timer 1, mode 2
TH1=0xFD;       //9600 baud rate
SCON=0x50;      // serial mode 1 and reception is enabled
TR1=1;          //start timer
while (1)       //repeat forever
```

```
{
SerTx('Y');          //call SexTx function
SerTx('E');          //call SexTx function
SerTx('S');          //call SexTx function
}
}
void SerTx(unsigned char x)
{
SBUF=x;              //place value in buffer
while (TI==0);       // monitor the TI flag bit (wait until transmitted)
TI=0;                // clear the TI flag
}
```

Steps for programming the 8051 to Receive data serially

The following steps must be taken during serial data reception,

1. Load the TMOD register with the value 20H. It uses Timer1 in mode 2 (8-bit auto-reload) to set the baud rate.
2. Load the TH1 register with the values shown in table 2 to set the baud rate for serial data transmission.
3. Load the SCON register with the value 50H, indicating serial mode 1 (8-bit standard UART mode) and reception is enable.
4. Start the Timer 1 by setting TR1 bit to '1'.
5. Monitor the RI flag bit using instruction ***while (RI==0);*** to check reception is over.
6. If RI arises means SBUF has entire character byte which is received serially. [When stop bit is received, the 8051 makes RI= '1' indicating that entire character or byte is received and it is in SBUF register. User can copy the contents of SBUF register into the safe place like memory or register]. Store the SBUF at safe place.
7. Clear the RI flag using instruction ***RI=0;*** for next data reception.
8. Goto step number 5 for next data transmission.

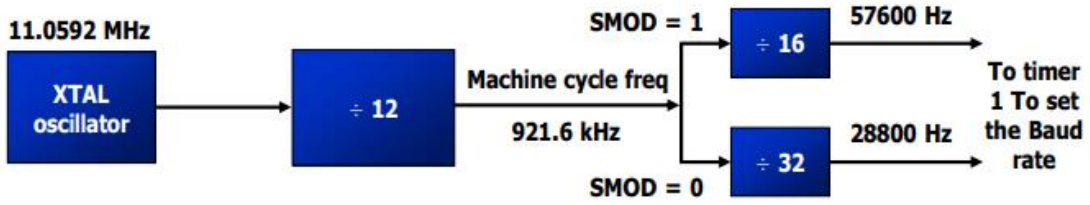
Ex. 3 Write a program to receive serially at 4800 baud, 8-bit data, with 1 stop bit. After receiving send it continuously to P0.

```
#include <reg51.h>
void main(void)
{
unsigned char mybyte;
TMOD=0x20;    //use Timer 1, mode 2
TH1=0xFA;     //4800 baud rate
SCON=0x50;    // serial mode 1 and reception is enabled
TR1=1;        //start timer
while (1)     //repeat forever
{
while (RI==0); //wait to receive
mybyte=SBUF;  //save value
P1=mybyte;    //send value to port
RI=0;         // clear the RI flag
}
}
```

Doubling the baud rate using 8051

There are two methods are used to increase the baud rate of serial communication.

1. First is by increasing frequency of the crystal but is not feasible because IBM PC baud rate can't exactly achieved.(e.g. 12 MHz/12=1 MHz and 1MHz/32=31250 Hz no correct value for baud rate 9800)
2. Another way to increase baud rate is use of PCON register.



PCON Register

It is an 8 bit special function register. It is byte addressable register and useful to doubling the baud rate as well power control in CHMOS controller. Fig 1.5 shows the internal structure of the PCON register.

PCON

0	x	x	x	0	0	0	0
SMOD	-	-	-	GF1	GF0	PD	IDL
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0

Fig.5 PCON register

SMOD

It is D7 bit of PCON register. As soon power is ON D7 bit is zero and we use usual baud rate set by Timer 1 mode 2. But if we make D7 bit=1 by software then baud rate of 8051 doubles as shown in Table.

Crystal frequency	Machine cycle frequency	SMOD	Serial communication frequency	If Value of TH1=-6 then baud rate=
11.0592MHz	11.0592/12=921.6Khz	=0	921.6/32=28800Hz	28800/6=4800
11.0592MHz	11.0592/12=921.6Khz	=1	921.6/16=57600Hz	57600/6=9800

Table 3: Doubling the Baud rate

If we set D7 =1, the internal circuitry for serial communication divides machine cycle frequency by 16 instead of 32. Hence baud rate will doubles for same value of TH1.

Following program for doubling baud rate for same crystal frequency.

```
MOV A,PCON        ;Copy of PCON into accumulator
```

```
SETB ACC.7          ;Make D7 bit of Accu.high
MOV  PCON,A         ;Make D7 of PCON =1
```

GF0 and GF1 are the general flag 0 and 1.

PD is a power down bit and **IDL** is an ideal mode bit. Both the bits are used in CHMOS processor only.

INTERRUPTS:

Introduction • In most of the real-time processes, to handle certain conditions properly, the actual task must be halt for some time – it takes required action – and then must return to the main task. •e.g. Microcontroller has to do different tasks simultaneously such as,

i) Receive data serially from external device and sent it to P0,

ii) read port P1 and transmit it serially to external device, and also send a copy to P2,

iii) generate a square wave of 5kHz frequency on P3.1. For executing such type of tasks, interrupt technique is to be used in programming. •An interrupt is an external or internal event/ signal that interrupts the microcontroller to inform it that a device needs its service •Interrupts are basically the events that temporarily suspend the main program, pass the control to the external sources and execute their task. It then passes the control to the main program where it had left off.

Interrupts Vs. Polling

Interrupts –

• Whenever any device needs its service, the device notifies the microcontroller by sending it an interrupt signal

• Upon receiving an interrupt signal, the microcontroller interrupts whatever it is doing and serves the device

• The program which is associated with the interrupt is called the interrupt service routine (ISR) or interrupt handler

• Each devices can get the attention of the microcontroller based on the assigned priority

• The microcontroller can also ignore (mask) a device request for service

Polling –

• The microcontroller continuously monitors the status of a given device • When the conditions met, it performs the service

• After that, it moves on to monitor the next device until every one is serviced

• The polling method is not efficient, since it wastes much of the microcontroller's time by polling devices that do not need service

• it is not possible to assign priority since it checks all devices in a round-robin fashion

• ignoring (mask) a device request for service is not possible

Interrupt Service Routine

¬ For every interrupt, there must be an interrupt service routine (ISR), or interrupt handler

¬ When an interrupt is invoked, the microcontroller runs the interrupt service routine

¬ For every interrupt, there is a fixed location in memory that holds the address of its ISR

¬ The group of memory locations set aside to hold the addresses of ISRs is called interrupt vector table

Steps in Executing an Interrupt

1. It finishes the instruction it is executing and saves the address of the next instruction (PC) on the stack

2. It also saves the current status of all the interrupts internally (i.e: not on the stack)

3. It jumps to a fixed location in memory, called the interrupt vector table, that holds the address of the ISR

4. The microcontroller gets the address of the ISR from the interrupt vector table and jumps to it

– It starts to execute the interrupt service subroutine until it reaches the last instruction of the subroutine which is RETI (return from interrupt)

5. Upon executing the RETI instruction, the microcontroller returns to the place where it was interrupted

– First, it gets the program counter (PC) address from the stack by popping the top two bytes of the stack into the PC

- Then it starts to execute from that address

Types of Interrupts

• The 8051 microcontroller can recognize five different events that cause the main program to interrupt from the normal execution. These five sources of interrupts in 8051 are:

1. Timer 0 overflow interrupt- TF0
2. Timer 1 overflow interrupt- TF1
3. External hardware interrupt 0- INT0
4. External hardware interrupt 1- INT1
5. Serial communication interrupt- RI/TI

Interrupt vector table

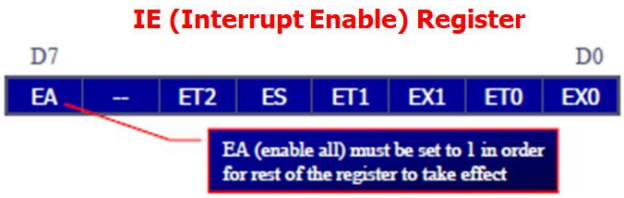
Interrupt vector table

Interrupt	ROM Location (hex)	Pin
Reset	0000	9
External HW (INT0)	0003	P3.2 (12)
Timer 0 (TF0)	000B	
External HW (INT1)	0013	P3.3 (13)
Timer 1 (TF1)	001B	
Serial COM (RI and TI)	0023	

¬Upon reset, all interrupts are disabled (masked), meaning that none will be responded to by the microcontroller if they are activated

¬The interrupts must be enabled by software in order for the microcontroller to respond to them ¬There is a register called IE (interrupt enable) that is responsible for enabling (unmasking) and disabling (masking) the interrupts

IE (Interrupt Enable) Register:



EA IE.7 Disables all interrupts

-- IE.6 Not implemented, reserved for future use

ET2 IE.5 Enables or disables timer 2 overflow or capture interrupt (8952)

ES IE.4 Enables or disables the serial port interrupt

ET1 IE.3 Enables or disables timer 1 overflow interrupt EX1

IE.2 Enables or disables external interrupt 1 ET0

IE.1 Enables or disables timer 0 overflow interrupt

EX0 IE.0 Enables or disables external interrupt 0

Enable Interrupt

Example1: Show the instructions to (a) enable the serial interrupt, timer 0 interrupt, and external hardware interrupt 1 (EX1),and (b) disable (mask) the timer 0 interrupt, then (c) show how to disable all the interrupts with a single instruction.

Solution: (a) MOV IE,#10010110B ;enable serial, ;timer 0, EX1 Another way to perform the same manipulation is SETB IE.7 ;EA=1, global enable SETB IE.4 ;enable serial interrupt SETB IE.1 ;enable Timer 0 interrupt SETB IE.2 ;enable EX1

(b) CLR IE.1 ;mask (disable) timer 0 ;interrupt only

(c) CLR IE.7 ;disable all interrupts

TIMER INTERRUPTS :

¬ 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 until it is notified that the timer has rolled over

Example -2 : Write a program that continuously get 8-bit data from P0 and sends it to P1 while simultaneously creating a square wave of 200 µs period on pin P2.1. Use timer 0 to create the square wave. Assume that XTAL = 11.0592 MHz.

• Solution: We will use timer 0 in mode 2 (auto reload). TH0 = 100 us /1.085 us = 92 ;

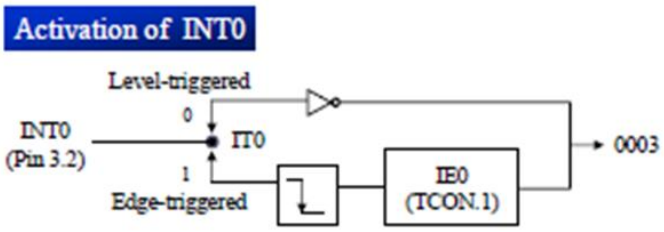
--upon wake-up go to main, avoid using memory allocated to Interrupt Vector Table

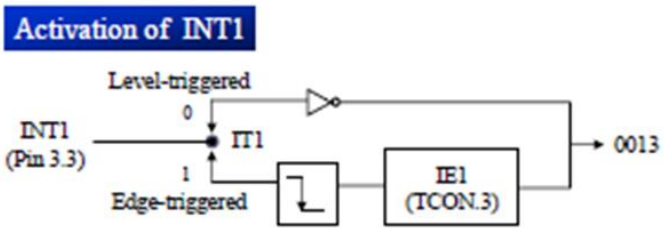
```
        ORG 0000H
        LJMP MAIN ;by-pass interrupt vector table ;
        ;--ISR for timer 0 to generate square wave
        ORG 000BH ;Timer 0 interrupt vector table
        CPL P2.1 ;toggle P2.1 pin
        RETI ;return from ISR
```

```
;--The main program for initialization
        ORG 0030H ;after vector table space
        MAIN: MOV TMOD,#02H ;Timer 0, mode 2
        MOV P0,#0FFH ;make P0 an input port
        MOV TH0,#-92 ;TH0=A4H for -92
        MOV IE,#82H ;IE=10000010 (bin) ; enable Timer 0
        SETB TR0 ;Start Timer 0
BACK: MOV A,P0 ;get data from P0
        MOV P1,A ;issue it to P1
        SJMP BACK ;keep doing it loop ;unless interrupted by TF0
        END
```

EXTERNAL HARDWARE INTERRUPTS

- The 8051 has two external hardware interrupts
  - ¬Pin 12 (P3.2) and pin 13 (P3.3) of the 8051, designated as INT0 and INT1, are used as external hardware interrupts
  - ¬The interrupt vector table locations 0003H and 0013H are set aside for INT0 and INT1
- There are two activation levels for the external hardware interrupts
  - ¬Level triggered
  - ¬Edge triggered





TCON (Timer/Counter) Register (Bit-addressable)

D7 D0

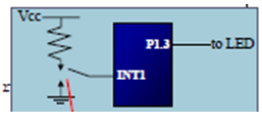
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1	TCON.7	Timer 1 overflow flag. Set by hardware when timer/counter 1 overflows. Cleared by hardware as the processor vectors to the interrupt service routine
TR1	TCON.6	Timer 1 run control bit. Set/cleared by software to turn timer/counter 1 on/off
TF0	TCON.5	Timer 0 overflow flag. Set by hardware when timer/counter 0 overflows. Cleared by hardware as the processor vectors to the interrupt service routine
TR0	TCON.4	Timer 0 run control bit. Set/cleared by software to turn timer/counter 0 on/off
IE1	TCON.3	External interrupt 1 edge flag. Set by CPU when the external interrupt edge (H-to-L transition) is detected. Cleared by CPU when the interrupt is processed
IT1	TCON.2	Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low-level triggered external interrupt
IE0	TCON.1	External interrupt 0 edge flag. Set by CPU when the external interrupt edge (H-to-L transition) is detected. Cleared by CPU when the interrupt is processed
IT0	TCON.0	Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low-level triggered external interrupt

Level-Triggered Interrupt

- In the level-triggered mode, INT0 and INT1 pins are normally high
- If a low-level signal is applied to them, it triggers the interrupt
- Then the microcontroller stops whatever it is doing and jumps to the interrupt vector table to service that interrupt
- The low-level signal at the INT pin must be removed before the execution of the last instruction of the ISR, RETI; otherwise, another interrupt will be generated
- This is called a level-triggered or level activated interrupt and is the default mode upon reset of the 8051

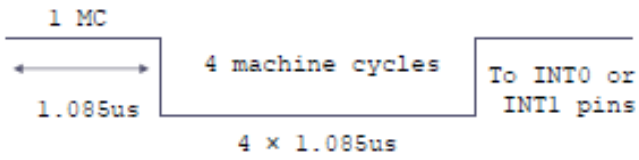
Example-5 : Assume that the INT1 pin is connected to a switch that is normally high. Whenever it goes low, it should turn on an LED. The LED is connected to P1.3 and is normally off. When it is turned on it should stay on for a fraction of a second. As long as the switch is pressed low, the LED should stay on.




```
Solution:   ORG 0000H
            LJMP MAIN ;by-pass interrupt ;vector table
    ;--ISR for INT1 to turn on LED
            ORG 0013H ;INT1 ISR
            SETB P1.3 ;turn on LED
            MOV R3,#255
    BACK: DJNZ R3,BACK ;keep LED on for a while
            CLR P1.3 ;turn off the LED
            RETI ;return from ISR
    ;--MAIN program for initialization
            ORG 30H
    MAIN: MOV IE,#10000100B ;enable external INT 1
    HERE: SJMP HERE ;stay here until get interrupted
            END
```

- Pins P3.2 and P3.3 are used for normal I/O unless the INT0 and INT1 bits in the IE register are enabled
    - After the hardware interrupts in the IE register are enabled, the controller keeps sampling the INTn pin for a low-level signal once each machine cycle
    - According to one manufacturer's data sheet,
        - The pin must be held in a low state until the start of the execution of ISR
        - If the INTn pin is brought back to a logic high before the start of the execution of ISR there will be no interrupt
        - If INTn pin is left at a logic low after the RETI instruction of the ISR, another interrupt will be activated after one instruction is executed

- To ensure the activation of the hardware interrupt at the INTn pin, make sure that the duration of the low-level signal is around 4 machine cycles, but no more
  - This is due to the fact that the leveltriggered interrupt is not latched
  - Thus the pin must be held in a low state until the start of the ISR execution



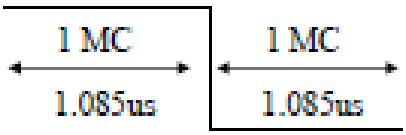
Edge-Triggered Interrupt:

- To make INT0 and INT1 edge triggered interrupts, we must program the bits of the TCON register
  - The TCON register holds, among other bits, the IT0 and IT1 flag bits that determine level- or edge-triggered mode of the hardware interrupt
    - IT0 and IT1 are bits D0 and D2 of the TCON register
    - They are also referred to as TCON.0 andTCON.2 since the TCON register is bit addressable

Example-6: Assume that pin 3.3 (INT1) is connected to a pulse generator, write a program in which the falling edge of the pulse will send a high to P1.3, which is connected to an LED (or buzzer). In other words, the LED is turned on and off at the same rate as the pulses are applied to the INT1 pin.

```
Solution:       ORG 0000H
                LJMP MAIN
;--ISR for hardware interrupt INT1 to turn on LED
                ORG 0013H ;INT1 ISR
                SETB P1.3 ;turn on LED
                MOV R3,#255
BACK:           DJNZ R3,BACK ;keep the buzzer on for a while
                CLR P1.3 ;turn off the buzzer
                RETI ;return from ISR
;------MAIN program for initialization
                ORG 30H
MAIN:           SETB TCON.2 ;make INT1 edge-triggered int.
                MOV IE,#10000100B ;enable External INT 1
HERE: SJMP HERE ;stay here until get interrupted END
```

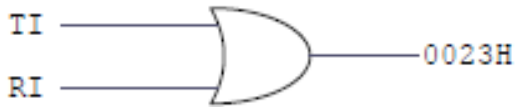
- The external source must be held high for at least one machine cycle, and then held low for at least one machine cycle
- The falling edge of pins INT0 and INT1 are latched by the 8051 and are held by the TCON.1 and TCON.3 bits of TCON register
    - Function as interrupt-in-service flags
    - It indicates that the interrupt is being serviced now and on this INTn pin, and no new interrupt will be responded to until this service is finished



- Regarding the IT0 and IT1 bits in the TCON register, the following two points must be emphasized
    - ¬ When the ISRs are finished (that is, upon execution of RETI), these bits (TCON.1 and TCON.3) are cleared, indicating that the interrupt is finished and the 8051 is ready to respond to another interrupt on that pin
    - ¬ During the time that the interrupt service routine is being executed, the INTn pin is ignored, no matter how many times it makes a high-to-low transition
        - RETI clears the corresponding bit in TCON register (TCON.1 or TCON.3)
        - There is no need for instruction CLR TCON.1 before RETI in the ISR associated with INT0

SERIAL COMMUNICATION INTERRUPT

- TI (transfer interrupt) is raised when the last bit of the framed data, the stop bit, is transferred, indicating that the SBUF register is ready to transfer the next byte
- RI (received interrupt) is raised when the entire frame of data, including the stop bit, is received
  - ¬ In other words, when the SBUF register has a byte, RI is raised to indicate that the received byte needs to be picked up before it is lost (overrun) by new incoming serial data
- In the 8051 there is only one interrupt set aside for serial communication
  - ¬ This interrupt is used to both send and receive data
  - ¬ If the interrupt bit in the IE register (IE.4) is enabled, when RI or TI is raised the 8051 gets interrupted and jumps to memory location 0023H to execute the ISR
  - ¬ In that ISR we must examine the TI and RI flags to see which one caused the interrupt and respond accordingly



Example-6: Write a program in which the 8051 reads data from P1 and writes it to P2 continuously while giving a copy of it to the serial COM port to be transferred serially. Assume that XTAL=11.0592. Set the baud rate at 9600.

```
Solution:       ORG 0000H
                LJMP MAIN
                ORG 23H
                LJMP SERIAL ;jump to serial int ISR
                ORG 30H
MAIN:           MOV P1,#0FFH ;make P1 an input port
                MOV TMOD,#20H ;timer 1, auto reload
                MOV TH1,#0FDH ;9600 baud rate
                MOV SCON,#50H ;8-bit,1 stop, ren enabled
                MOV IE,10010000B ;enable serial int.
                SETB TR1 ;start timer 1
```

```
BACK:   MOV A,P1 ;read data from port 1
        MOV SBUF,A ;give a copy to SBUF
        MOV P2,A ;send it to P2
        SJMP BACK ;stay in loop indefinitely
;----------------SERIAL PORT ISR
        ORG 100H
SERIAL: JB TI,TRANS     ;jump if TI is high
        MOV A, SBUF     ;otherwise due to receive
        CLR RI          ;clear RI since CPU doesn't
        RETI            ;return from ISR
TRANS:  CLR TI          ;clear TI since CPU doesn't
        RETI            ;return from ISR
        END
```

The moment a byte is written into SBUF it is framed and transferred serially. As a result, when the last bit (stop bit) is transferred the TI is raised, and that causes the serial interrupt to be invoked since the corresponding bit in the IE register is high. In the serial ISR, we check for both TI and RI since both could have invoked interrupt.

Example : Write a program in which the 8051 gets data from P1 and sends it to P2 continuously while incoming data from the serial port is sent to P0. Assume that XTAL=11.0592. Set the baud rata at 9600.

Solution:

```
        ORG 0000H
        LJMP MAIN
        ORG 23H
        LJMP SERIAL    ;jump to serial int ISR
        ORG 30H
MAIN:   MOV P1,#0FFH        ;make P1 an input port
        MOV TMOD,#20H       ;timer 1, auto reload
        MOV TH1,#0FDH       ;9600 baud rate
        MOV SCON,#50H       ;8-bit,1 stop, ren enabled
        MOV IE,10010000B ;enable serial int.
        SETB TR1            ;start timer 1
BACK:   MOV A,P1        ;read data from port 1
        MOV P2,A        ;send it to P2
SJMP    BACK            ;stay in loop indefinitely
;----------------SERIAL PORT ISR
        ORG 100H
SERIAL: JB TI,TRANS;jump if TI is high
        MOV A,SBUF ;otherwise due to receive
        MOV P0,A ;send incoming data to P0
        CLR RI ;clear RI since CPU doesn't
```

```
        RETI ;return from ISR
TRANS: CLR TI ;clear TI since CPU doesn't
        RETI ;return from ISR
        END
```

Example : Write a program using interrupts to do the following:
(a) Receive data serially and sent it to P0,
(b) Have P1 port read and transmitted serially, and a copy given to P2,
(c) Make timer 0 generate a square wave of 5kHz frequency on P0.1.
Assume that XTAL-11,0592. Set the baud rate at 4800.

```
Solution:       ORG 0
        LJMP MAIN
        ORG 000BH    ;ISR for timer 0
        CPL P0.1     ;toggle P0.1
        RETI         ;return from ISR
        ORG 23H ;
        LJMP SERIAL   ;jump to serial interrupt ISR
        ORG 30H
MAIN:  MOV P1,#0FFH        ;make P1 an input port
        MOV TMOD,#22H;timer 1,mode 2(auto reload)
        MOV TH1,#0F6H      ;4800 baud rate
        MOV SCON,#50H;8-bit, 1 stop, receive enabled
        MOV TH0,#-92       ;for 5kHZ wave
MOV IE,10010010B ;enable serial int.
        SETB TR1     ;start timer 1
        SETB TR0     ;start timer 0
BACK:  MOV A,P1      ;read data from port 1
        MOV SBUF,A   ;give a copy to SBUF
        MOV P2,A     ;send it to P2
SJMP   BACK          ;stay in loop indefinitely
;----------------SERIAL PORT ISR
        ORG 100H
SERIAL:JB TI,TRANS  ;jump if TI is high
        MOV A,SBUF   ;otherwise due to receive
        MOV P0,A     ;send serial data to P0
        CLR RI              ;clear RI since CPU doesn't
        RETI         ;return from ISR
TRANS: CLR TI               ;clear TI since CPU doesn't
        RETI         ;return from ISR
        END
```

INTERRUPT PRIORITY:

- When the 8051 is powered up, the priorities are assigned according to the following

- In reality, the priority scheme is nothing but an internal polling sequence in which the 8051 polls the interrupts in the sequence listed and responds accordingly

Interrupt Priority Upon Reset:

Interrupts	Priority
External Interrupt 0 (INT0)	1
Timer Interrupt 0 (TF0)	2
External Interrupt 1 (INT1)	3
Timer Interrupt 1 (TF1)	4
Serial Communication (RI + TI)	5

- We can alter the sequence of interrupt priority by assigning a higher priority to any one of the interrupts by programming a register called IP (interrupt priority)

- To give a higher priority to any of the interrupts, we make the corresponding bit in the IP register high
- When two or more interrupt bits in the IP register are set to high
- While these interrupts have a higher priority than others, they are serviced according to the sequence of Table 1

Interrupt Priority Register (Bit-addressable)

D7							D0
--	--	PT2	PS	PT1	PX1	PT0	PX0

- PX0 IP.0 External interrupt 0 priority bit
- PT0 IP.1 Timer 0 interrupt priority bit
- PX1 IP.2 External interrupt 1 priority bit
- PT1 IP.3 Timer 1 interrupt priority bit
- PS IP.4 Serial port interrupt priority bit
- PT2 IP.5 Timer 2 interrupt priority bit (8052 only)
- -- IP.6 Reserved
- --

- **Priority bit=1 assigns high priority**
- **Priority bit=0 assigns low priority**

Example 11-12

(a) Program the IP register to assign the highest priority to INT1(external interrupt 1), then

(b) discuss what happens if INT0, INT1, and TF0 are activated at the same time. Assume the interrupts are both edge-triggered.

Solution:

(a) MOV IP,#00000100B ;IP.2=1 assign INT1 higher priority. The instruction SETB IP.2 also will do the same thing as the above line since IP is bit-addressable.

(b) The instruction in Step (a) assigned a higher priority to INT1 than the others; therefore, when INT0, INT1, and TF0 interrupts are activated at the same time, the 8051 services INT1 first, then it services INT0, then TF0. This is due to the fact that INT1 has a higher priority than the other two because of the instruction in

Step (a). The instruction in Step (a) makes both the INT0 and

TF0 bits in the IP register 0. As a result, the sequence in Table 1 is followed which gives a higher priority to INT0 over TF0

Example:

Assume that after reset, the interrupt priority is set the instruction

MOV IP,#00001100B. Discuss the sequence in which the interrupts are serviced.

Solution:

The instruction “MOV IP #00001100B” (B is for binary) and timer 1 (TF1)to a higher priority level compared with the rest of the interrupts. However, since they are polled according to Table, they will have the following priority.

Highest Priority	External Interrupt 1 (INT1)
	Timer Interrupt 1 (TF1)
	External Interrupt 0 (INT0)
	Timer Interrupt 0 (TF0)
Lowest Priority	Serial Communication (RI+TI)