DEPARTMENT

OF

ELECTRONICS & COMMUNICATION ENGINEERING

MICROCONTROLLERS

(Theory Notes)

Autonomous Course

Prepared by

PROF Vibha T G

PROF Ramgopal Segu

Module – 4 Contents

8051 Timers and serial port: Timer introduction, Different modes of timer operations, Assembly and C programming on Timers (Mode1 and Mode 2) . Basics of 8051 Serial Communication, RS 232 connections, C programming on serial communication
Text Book-1

Dayananda Sagar College of Engineering
Shavige Malleshwara Hills, Kumaraswamy Layout,
Banashankari, Bangalore-560078, Karnataka
Tel : +91 80 26662226 26661104 Extn : 2731 Fax : +90 80 2666 0789
Web - http://www.dayanandasagar.edu Email : hod-ece@dayanandasagar.edu
(An Autonomous Institute Affiliated to VTU, Approved by AICTE & ISO 9001:2008 Certified)
(Accredited by NBA, National Assessment & Accreditation Council (NAAC) with 'A' grade)

4.1 Introduction to 8051 timers

- 8051 microcontroller have two timers timer 0 and timer 1.These are 16 bit timers, but 8051 is a 8 bit controller so they are referred are lower byte and higher byte shown below TL0, TH0, Tl1,TH1.
- They can be used as either timers to generate time delay or as counters to count events.
- The required delay is obtained by loading the number machine cycles into those registers. Normally delay count value will be decremented in software delay program, but count value will be incremented in timers. i.e.(means delay count need to be subtracted from max count value and then timer has to be activated.)
- Timer 0 and Timer 1 register formats are as shown below in fig 4.1

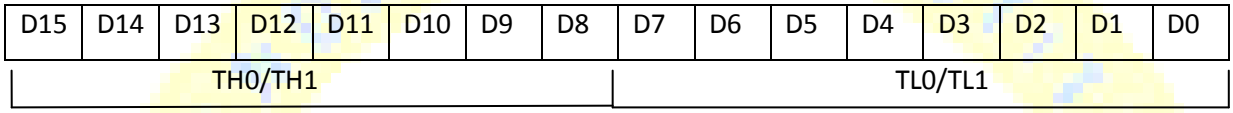


Fig 4.1 bit format for Timer 0 and Timer 1 registers.

4.2 TMOD register(Timer mode):

- Both the timers,timer0 and timer 1 use TMOD register to set various timer operation modes.
- TMOD is an 8 bit register with lower 4 bits set aside for timer 0 and higher 4 bits for timer 1.
- Lower two bits are used are to set the timer mode and higher 2 bits specify the operation.
- TMOD register format is shown in fig 2 below.

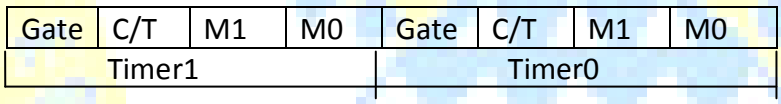


Figure 4.2:TMOD register format

Gate Bit:

- Timer control will be decided through this bit value
- Gate-=0 : indicates that timer can be controlled through program by TR and TF bit's in TCON register.Bit TR is used to start the timer by making TR=1 and stop the timer by making TR=0.This is called software control.
- Gate=1: indicates that timer control is done by external source. This is called hardware control.

C/T (Counter/ Timer) Bit:

This bit in the TMOD register is used to decide whether the timer is used

As a delay generator or an event counter.

- If C/T =0, Then timer is working as delay generator
- If C/T=1, Then that timer is working as event counter

MODE bit 1 and Mode bit 0(M1 and M0):

These two bits are used to select different timer modes. The below table 1 shows the different modes.

M1	M0	Mode	Operation
0	0	0	13 bit timer mode 8 bit timer/counter THx with TLx as 5 bit prescalar..
0	1	1	16 bit timer mode.16 bit timer/counter with THx and TLx cascaded: no prescalar
1	0	2	8 bit auto reload mode.8 bit auto reload timer/counter THx holds a value that is to be reloaded to TLx each time as it overflows.
1	1	3	Split timer mode.

Table 1:Timer operating modes

Example 1: Indicate which mode and which timer/counter are selected for each of the following?

a) MOV TMOD,#01h
TMOD=00000001H,mode 1 timer 0 is selected.
b) MOV TMOD,#20h
TMOD=00100000H,mode 2 ,timer 1 is selected.
c) MOV TMOD,#69h
TMOD=01101001,counter1 is selected with mode2 and software gating control.
Timer 0 is selected with mode1 and gating control is through external

Clock source for timer :

The frequency of timer is always $1/12^{th}$ the frequency of the crystal attached to 8051.
Example: If Master CLK=12 MHz,
Timer Clock frequency = Master CLK/12 = 1 MHz
Timer Clock Period = 1micro second
This indicates that one increment in count will take 1 micro second.
Similarly for clock frequencies for **12MHZ** and **16MHZ.**
1/12X12M=1MHz= and time =1 us
1/12X16M= 1.333MHz and time=.75 us.

4.3 Timer MODE 1 Programming :(16 bit)

- Load the TMOD register value indicating which timer (timer 0 or timer 1) is to be used and which timer mode (0 or 1) is selected.
- Load registers TL and TH with initial count value. Since it's a 16 bit timer values from 0000h to FFFFh can be loaded into timer register.
- Start the timer using "SETB TR0" and "SETB TR1" for timer0 and timer1 respectively.
- Keep monitoring the timer overflow flag (TF) with the "JNB TFx, target "instruction to check if it is raised.
- Stop the timer using "CLR TR0" and "CLR TR1" instructions for timer0 and timer1 respectively.
- Clear the TF flag for the next round.
- Go back to Step 2 to load TH and TL again.

The block diagram for mode1 is shown in fig 4.3 below.

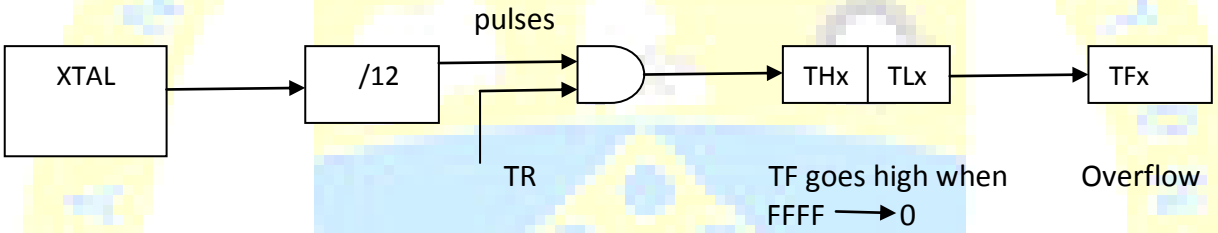


Fig 4.3:Block diagram for mode 1

Example 1: In the following program, we create a square wave of 50% duty cycle (with equal portions high and low) on the P1.5 bit. Timer 0 is used to generate the time delay. Analyze the program.

```
LABEL    Code                Comments
         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 P1.5           toggle P1.5 to create a square wave.
          ACALL DELAY        call a delay subroutine
         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                 Return from subroutine.
```

The program is analyzed as follows.

1. TMOD is loaded.
2. FFF2H is loaded into TH0-TL0.
3. P1.5 is toggled for the high and low portions of the pulse.
4. The DELAY subroutine using the timer is called.
5. In the DELAY subroutine, timer 0 is started by the SETB TR0 instruction.
6. Timer 0 counts up with the passing of each clock, which is provided by the crystal oscillator. As the timer counts up, it goes through the states of FFF3,FFF4, FFF5, FFF6, FFF7, FFF8, FFF9, FFFA, FFFB, and so on until it reaches FFFFH. One more clock rolls it to 0, raising the timer flag (TF0=1).At that point, the JNB instruction falls through.
7. Timer 0 is stopped by the instruction CLR TR0. The DELAY subroutine ends, and the process is repeated. Notice that to repeat the process, we must reload the TL and TH registers, and start the process is repeated.

Delay calculation:

- The timer works with a clock frequency of 1/12 of the XTAL frequency;
- therefore, we get 11.0592 MHz / 12 = 921.6 kHz as the timer frequency.
- So each clock has a period of T = 1/921.6kHz = 1.085us.
- Delay = number of counts × 1.085us.
- The number of counts for the roll over is FFFFH – FFF2H = 0DH (13 decimal).
- However, we add one to 13 because of the extra clock needed when it rolls over from FFFF to 0 and raise the TF flag.
- This gives 14 × 1.085us = 15.19us for half the pulse.
- For the entire period it is T = 2 × 15.19us = 30.38us as the time delay generated by the timer.
- The count to be loaded can be calculated using the formula
  Initial count value= [Maximum value – Required delay *crystal freq/12]

2. Calculate the frequency of the square wave generated on pin P1.5.

Solution:

In the above delay calculation overhead due to instruction was not included. To get a more accurate timing, add clock cycles due to these instructions are included. This is as shown below
.

Label	Code	cycles
HERE:	MOV TL0,#0F2H	2
	MOV TH0,#0FFH	2
	CPL P1.5	1
	ACALL DELAY	2
	SJMP HERE	2
DELAY:	SETB TR0	1
AGAIN:	JNB TF0,AGAIN	2*14
	CLR TF0	1
	CLR TR0	1
	RET	2

Total 42

T = 2 × 42 × 1.085 us = 91.94 us and Freq = 10.9 KHz

3. Assume that XTAL = 11.0592 MHz What value to be loaded to the timer's register to create a time delay of 5 ms (milliseconds)? Show the program for timer 0 MODE 1 to create a pulse width of 5 ms on P2.3.

Calculation:

- Since XTAL = 11.0592 MHz, the counter counts up every 1.085 us.
- This means that out of many 1.085 us intervals 5 ms pulse to be created.
  So 5 ms / 1.085 us = 4608 clocks.
- The value to be loaded into TL and TH is
  65536 – 4608 = EE00H, where TH = EE and TL = 00.

Program to create pulse width of 5ms

```
        ORG 00H
        CLR P2.3          Clear P2.3
        MOV TMOD,#01H     Timer 0, 16-bitmode
HERE:   MOV TL0,#00H      TL0=0, the low byte
        MOV TH0,#0EEH     TH0=EE, the high byte
        SETB P2.3         SET high P2.3

        SETB TR0          Start timer 0
AGAIN:  JNB TF0,AGAIN     Monitor timer flag 0
        CLR  TR0          Stop the timer 0
        CLR TF0           Clear timer 0 flag
        END
```

4. Assume that XTAL = 11.0592 MHz, Write an 8051 ALP to generate a square wave of 2 kHz frequency on pin P1.5.

Solution:

Calculation:

(a) T = 1 / f = 1 / 2 kHz = 500 us the period of square wave.
(b) 1 / 2 of it for the high and low portion of the pulse is 250 us.
(c) 250 us / 1.085 us = 230 and 65536 – 230 = 65306 which in hex is FF1AH.
(d) TL = 1Ah and TH = FFh.

Program

```
        ORG 00H
        MOV TMOD,#10H
AGAIN:  MOV TL1,#1AH
        MOV TH1,#0FFH
        SETB TR1
BACK:   JNB TF1,BACK
        CLR TR1
        CPL P1.5
        CLR TF1
        SJMP AGAIN
        END
```

Timers are programmed not only in assembly language; they can also programmed in C.

1. Write an 8051 C program to toggle all the bits of port P1 continuously with some delay in between. Use Timer 0, 16-bit mode to generate the delay.

Solution:

```
#include <reg51.h>
Void T0Delay (void);                //Delay function
Void main (void){
While (1) {
P1=0x55;
T0Delay ();                         To toggle all pins of P1 by loading 55h and AAh into it and calling a delay
P1=0xAA;                            function in between.
T0Delay ();
}
}
Void T0Delay(){
TMOD=0x01;                  //timer 0 mode 1
TL0=0x00;
TH0=0x35;
TR0=1;                      //start timer
While (TF0==0);             //wait for timer flag to get raised by monitoring it.
TR0=0;                      //stop timer
TF0=0;                      //clear timer flag.
}
```

2.Write an 8051 C program to toggle only bit P1.5 continuously every 50 ms. Use Timer 0, mode 1 (16-bit) to create the delay.

Calculation:

Count value=65536-(50ms/1.085µs)
=4BFEH

Solution:

```
#include <reg51.h>
Void T0M1Delay (void);
sbit mybit=P1^5;
Void main (void){
While (1) {
mybit=~mybit;
T0M1Delay ();
}
}
Void T0M1Delay(void){
TMOD=0x01;
TL0=0xFE;
TH0=0x4B;
TR0=1;
```

```
While (TF0==0);
TR0=0;
TF0=0;
}
```

3. A switch is connected to pin P1.2.Write an 8051 C program to monitor SW and create the following frequencies on pin P1.7:
SW=0:500Hz
SW=1:750Hz.Use timer 0, mode1.
For 500Hz-count value to be loaded-FC67h
For 750 Hz-count value to be loaded-FD9Ah

Program:

```
#include<reg51.h>
sbit mybit=P1^7;
sbit SW=P1^2;
Void delay (unsigned char);
Void main ()
{
SW=1;
While (1)
    {
      mybit=~mybit;
      if(SW==0)
          delay(0);
      if(SW==1)
          delay(1);
      }
}
void delay(unsigned char c)
    {
        TMOD=0x01;
         if(c==0)
           {
             TL0 =0x67;
              TH0=0xFC;
           }
        else
            {
              TL0 =0x9A;
              TH0=0xFD;
             }
        TR0=1;
        While (TF==0);
        TR0=0;
         TF0=0;
}
```

4.4 Timer Mode 2 Programming :(8 bit auto reload):

1. Load the TMOD value register indicating which timer (timer 0 or timer 1) is to be used, and the timer mode (mode 2) is selected.
2. Load the TH registers with the initial count value.
3. Start timer.
4. Keep monitoring the timer flag (TF) with the JNB TFx, target instruction to see whether it is raised .Get out of the loop when TF goes high
5. Clear the TF flag.
6. Go back to Step4; since mode 2 is auto reload.

The block diagram for mode 2 is shown in below figure 4.4

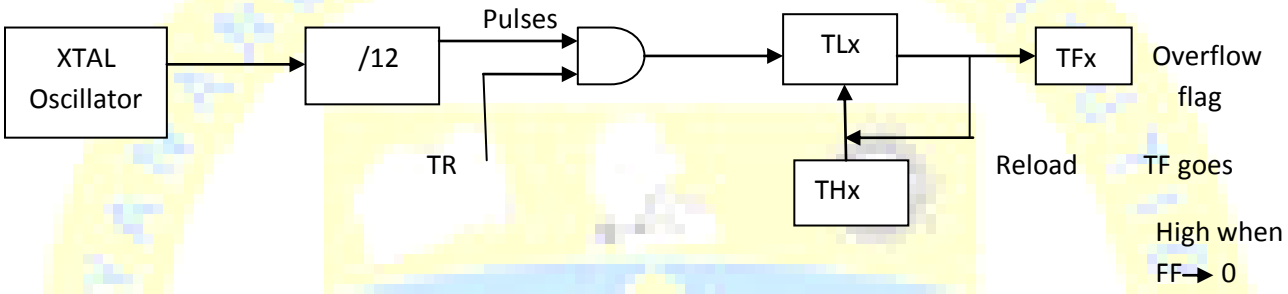


Fig4.4: Mode 2 programming block diagram.

1. Assume XTAL = 11.0592 MHz, find the frequency of the square wave generated on pin P1.0 in the following program. Also find the smallest achievable frequency in this program and TH value to do that.

```
        ORG 00H
        MOV TMOD,#20H       T1/8-bit/auto reload
        MOV TH1,#5          TH1 = 5
        SETB TR1            start the timer 1
BACK:   JNB TF1,BACK        till timer rolls over
        CPL P1.0            Create square wave
        CLR TF1             clear Timer 1 flag
        SJMP BACK           Since mode 2 is auto-reload jump to label back
                            to monitor the overflow flag .
        END
```

First notice the target address of SJMP. In mode 2 so no need to reload TH since it is auto-reload.
Now (256 - 05) × 1.085 us = 251 × 1.085 us = 272.33 us is the high portion of the pulse.
Since it is a 50% duty cycle square wave, the period T is twice that; as a result
T = 2 × 272.33 us = 544.67 us
Frequency (F) = 1.83597 kHz.

2. Write an 8051 ALP to generate a square wave of frequency 10Khz on pin P1.4.Use timer 0 mode2 with XTAL Frequency=22MHz.

Time period of square wave =1/f=1/10K=0.1ms
Ton=Toff=0.05ms

Count value=256-(0.05m/0.54µ)
=256-94
=A4h

```
        ORG 00H
        MOV TMOD,#02H
        MOV TH0,#0A4H
        SETB TR0
BACK:   JNB TF0,BACK
        CPL P1.4
        CLR TF0
        SJMP BACK
        END
```

3. Write an 8051 ALP to toggle all bits of port 2 continuously every 50ms.Use timer1 mode2 with XTAL Freq=11.0592MHz.

Calculation:

Maximum delay possible in mode 2 is 0.277ms but given is 50ms .So we have to go for multiple delay count.
Take 0.25ms,now to get 50ms ,50ms/0.25ms=200.
Count value to be loaded to get 0.25ms delay i.e.,
=256-0.25m/1.085µ
=20h or 26 in decimal.

Program:

```
        ORG 00H
        MOV TMOD,#20H
        MOV TH1,#20H
        SETB TR1
AGAIN:  MOV P2,#00H
        MOV R0,#200
        ACALL DELAY
        MOV P2,#0FFH
        MOV R0,#200
        ACALL DELAY
        SJMP AGAIN
DELAY:  JNB TF1,DELAY
        CLR TF1
        DJNZ R0,DELAY
        CLR TR1
        RET
```

4. Write an 8051 C program to toggle only pin P1.5 continuously every 250ms.Use timer 0 mode 2 to create delay.

Calculation:
Maximum delay possible in mode 2 -0.277ms
Consider 0.25ms.
To create 250ms multiple count required=250/0.25=1000(250*4)
0.25ms count to be loaded=233(256-23)

Program:

```
#include<reg51.h>
void ToM2Delay(void);
sbit mybit=P1^5;
void main(void)
{
unsigned char x,y;
while(1)
{
mybit= ~mybit;
for(x=0;x<250;x++)
    for(y=0;y<4;y++)
        ToM2Delay();
}
}
void ToM2Delay(void)
{
TMOD=0x02;
TH0=-23;
TR0=1;
while(TF0==0);
TR0=0;
TF0=0;
}
```

4.5 Programming Timer Interrupts:

1. Write a program that continuously gets 8 bit data from P0 and sends it to P1 while simultaneously creating a square wave of 200µs on pin P2.1.Use timer 0 .

Mode2 used, TH0=100/1.085=92.Interrupt mode is used.

```
        ORG 00H
        LJMP MAIN
        ORG 0BH
        CPL P2.1
        RETI
        ORG 30H
MAIN: MOV TMOD,#02H
```

```
        MOV TH0,#-92
        MOV IE,#82H
        SETB TR0
BACK:   MOV A,P0
        MOV P1,A
        SJMP BACK
        END
```

2. Write a program to generate a square wave of 50hz frequency on pin P1.2 in interrupt mode using timer 0 mode 1.Assume XTAL=11.0592MHz.

Given F =50hz,so T=1/50=0.02s
Ton =Toff=0.01s
Count value =65536-0.01/1.085µs
=65536-9216
=56320=DC00H

```
        ORG 00h
        LJMP Main
        ORG 000Bh
        CPL P1.2
        MOV  TL0,#00h
        MOV TH0,#0DCh
        RETI
        ORG 30H
MAIN:   MOV TMOD,#01H
        MOV TL0,#00H
        MOV TH0,#0DCH
        MOV IE,#82H
        SETB TR0
        SJMP $
        END
```

3.Write an 8051 ALP continuously gets 8 bit data from P0 and sends it to 91 which simultaneously generates square wave of 200 usec period on pin P2.1. use timer 0 and assume XTAL=11.0592MHz

```
LABEL       MNEMONIC AND OPERANDS
            ORG 00h
            LJMP MAIN
            ORG 0BH
            CPL P2.1
            RETI
            ORG 030H
MAIN:       MOV TMOD,#02H
            MOV P0,#0FFH
            MOV TH0, #-92
            MOV IE,#82H
            SETB TR0
```

```
BACK:   MOV A,P0
        MOV P1,A
        SJMP BACK
        END
```

4.Write an ALP to generate a square wave on P2 with xtal=11.0592MHz. and also send a value of P0 and display it at port 1.

```
LABEL   NEMONIC AND OPERANDS
        ORG 00H
        SJMP START
        ORG 010H
        SJMP SERVE
        ORG 30H
START:  MOV TMOD,#10H
        MOV IE,#88H
        MOV TH1,#0FCH
        MOV TL1,#67H
        SETB P2.4
        SETB TR1
AGAIN:  MOV A,P0
        MOV P1,A
        SJMP AGAIN
        ORG 060H
SERVE:  CLR TR1
        JNB P2.4,HIGH
        MOV TH1,#0F8H
        MOV TL1,#0CEH
        CLR P2.4
        SETB TR1
        RETI
HIGH:   MOV TH1,#0FCH
        MOV TL1,#67H
        SETB P2.4
        SETB TR1
        RETI
        END
```

4. Write an ALP to generate two square wave of 5KHz and 25KHz at pin P1.3 and P2.3 respectively. With XTAL=22MHz in interrupt mode.

```
LABEL   MNEMONIC AND OPERANDS
        ORG 00H
        LJMP START
```

```
            ORG 0BH
            CPL  P1.3
            RETI
            ORG 1BH
            CPL P2.3
            RETI
            ORG 30H
START:      MOV TMOD,#22H
            MOV IE,#8AH
            MOV TH0,#49H
            MOV TH1,#0DBH
            SETB TR0
            SETB TR1
HERE:       SJMP HERE
            END
```

4.6 Serial Port Programming:

INTRODUCTION:

- Computer can transfer data in two ways [Parallel and serial].
- 8 bits of data can transfer at a time in parallel and bit by bit can transfer in serial.
- Parallel transfer can be used for short distances like printer and scanner, and each one uses cables with many wire strips, even there is chance of signal distortion and it is to expensive.
- Serial data transfer can be used between two computers which for away, like connected through internet.
- Here we can discuss basics of serial communication, 8051 interfacing to RS232 via MA232 line drivers, and serial port programming.
- It is shown diagrammatically in fig 4.6.1

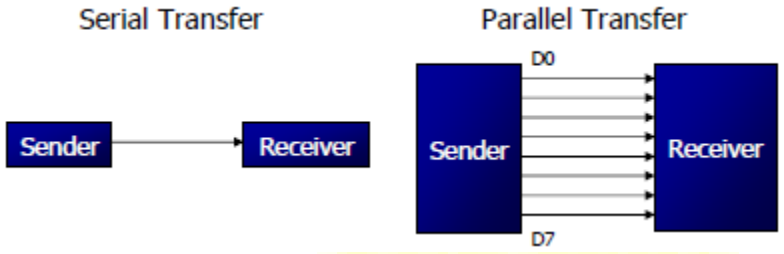


Fig4.6.1: Serial and Parallel data transfer

Basics of Serial Communication:

- For serial data communication to work, the byte of data must be converted to serial bits using PISO shift register, then it can transmitted over a single data line.
- Receiver end should have SIPO shift register, then only serial data can packed in to a byte of data.

- If the data is to be transferred on the telephone line, it must be converted from 0s and 1s to audio tones, which are sinusoidal shaped signal. This conversion is performed by a peripheral device called a modem, which modulator/demodulator.
- Serial data communication uses two methods, asynchronous and synchronous.
- The synchronous method transfers a block of data at a time, where as asynchronous method transfers single byte at a time, but programs can be tedious and long.
- For this reason, there are special IC chips made refereed as UART [universal asynchronous receiver transmitter] and USART [universal synchronous asynchronous receiver transmitter]
- The 8051 has inbuilt UART.

Half and full duplex transmission:

Simplex mode is to only send data, whereas duplex transmission is the one where both transmission and reception of data can happen. Duplex can be either half or full duplex.In half duplex data transfer not simultaneous whereas in full duplex data transfer is simultaneous. The concept is as shown in below figure 4.6.2

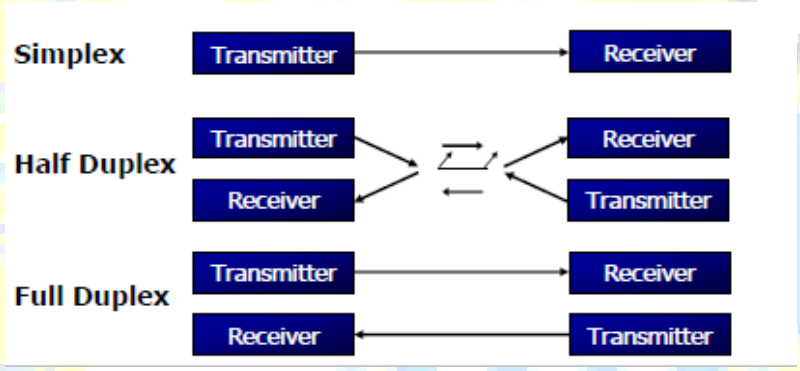


Fig:4.6.2 Half and Full duplex transmission.

Asynchronous serial Communication and data framing:

- A protocol is a set of rules agreed by both the sender and receiver on.
- How the data is packed
- How many bits constitute a character
- When the data begins and ends
- Asynchronous serial data communication is widely used for character-oriented transmissions
- Each character is placed in between start and stop bits, this is called framing.
- Block-oriented data transfers use the synchronous method
- 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).
- The data asynchronous data framing format is as shown in below figure 4.6.3

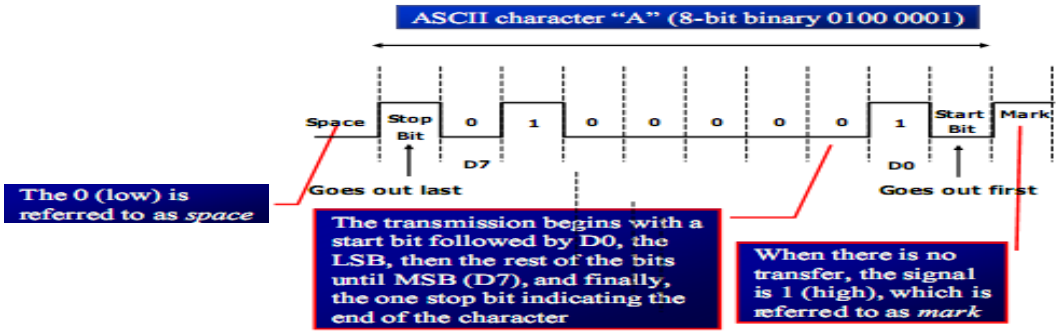


Fig:4.6.3 Asynchronous data format.

- Assuming that we are transferring a text file of ASCII characters using 1 stop bit, we have a total of 10 bits for each character. This gives 25% overhead, i.e. each 8-bit character with an extra 2 bits.
- UART chips allow programming of the parity bit for odd-, even-, and no-parity options

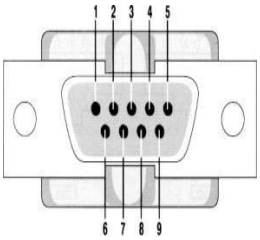
The rate of data transfer in serial data communication is stated in bps (bits per second) Another widely used terminology for bps is baud rate.The data transfer rate of given computer system depends on communication ports incorporated into that system

- IBM PC/XT could transfer data at the rate of 100 to 9600 bps.
- Pentium-based PCs transfer data at rates as high as 56K bps.
- In asynchronous serial data communication, the baud rate is limited to 100K bps.

CONNECTIONS TO RS-232

RS-232 standards:

- To allow compatibility among data communication equipment made by various manufactures, an interfacing standard called RS232 was set by the Electronics Industries Association (EIA) in 1960.
- The standard was set long before the advent of logic family, so its input and output voltage levels are not TTL compatible.
- In RS232, a logic one (1) is represented by -3 to -25V and referred as MARK while logic zero (0) is represented by +3 to +25V and referred as SPACE.
- To connect any RS232 to a microcontroller system voltage converters such as MAX232 to convert the TTL logic level to RS232 voltage levels and vice-versa are used. MAX232 IC chips are commonly referred as line drivers.
- In RS232 standard there are two types of connectors. DB9 connector or DB25 connector which are as shown in below fig.4.6.5 and 4.6.6



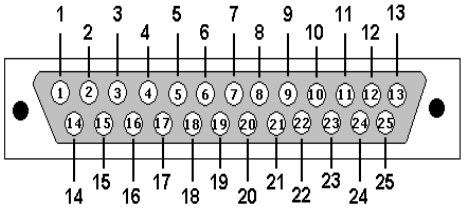


Fig 4.6.5:DB9 Male Connector Fig 4.6.6: DB25 Male Connector

- The pin description of DB9 and DB25 Connectors are as follows:

DB-25 Pin No.	DB-9 Pin No.	Abbreviation	Full Name
Pin 2	Pin 3	TD	Transmit Data
Pin 3	Pin 2	RD	Receive Data
Pin 4	Pin 7	RTS	Request To Send
Pin 5	Pin 8	CTS	Clear To Send
Pin 6	Pin 6	DSR	Data Set Ready
Pin 7	Pin 5	SG	Signal Ground
Pin 8	Pin 1	CD	Carrier Detect
Pin 20	Pin 4	DTR	Data Terminal Ready
Pin 22	Pin 9	RI	Ring Indicator

Current terminology classifies data communication equipment as

- DTE (data terminal equipment) refers to terminal and computers that send and receive data.
- DCE (data communication equipment) refers to communication equipment, such
- as modems
- The simplest connection between a PC and microcontroller requires a minimum of three pins, TxD, RxD, and ground.

DB-9 RS232 PIN explanation :

- DTR (data terminal ready): When terminal is turned on, it sends out signal DTR to indicate that it is ready for communication.
- DSR (data set ready): When DCE is turned on and has gone through the self-test, it assert DSR to indicate that it is ready to communicate.
- RTS (request to send) When the DTE device has byte to transmit, it assert RTS to signal the modem that it has a byte of data to transmit.
- CTS (clear to send): When the modem has room for storing the data it is to receive, it sends out signal CTS to DTE to indicate that it can receive the data now .
- DCD (data carrier detect): The modem asserts signal DCD to inform the DTE that a valid carrier has been detected and that contact between it and the other modem is established.
- RI (ring indicator): An output from the modem and an input to a PC indicates that the telephone is ringing.It goes on and off in synchronous with the ringing sound.

The 8051 connection to MAX232 is as follows.

- The 8051 has two pins that are used specifically for transferring and receiving data serially. These two pins are called TXD, RXD. Pin 11 of the 8051 (P3.1) assigned to TXD and pin 10 (P3.0) is designated as RXD.
- These pins TTL compatible; therefore they require line driver (MAX 232) to make them RS232 compatible.
- MAX 232 converts RS232 voltage levels to TTL voltage levels and vice versa. One advantage of the MAX232 is that it uses a +5V power source which is the same as the source voltage for the 8051.
- The typical connection diagram between MAX 232 and 8051 is shown below in fig 4.6.7

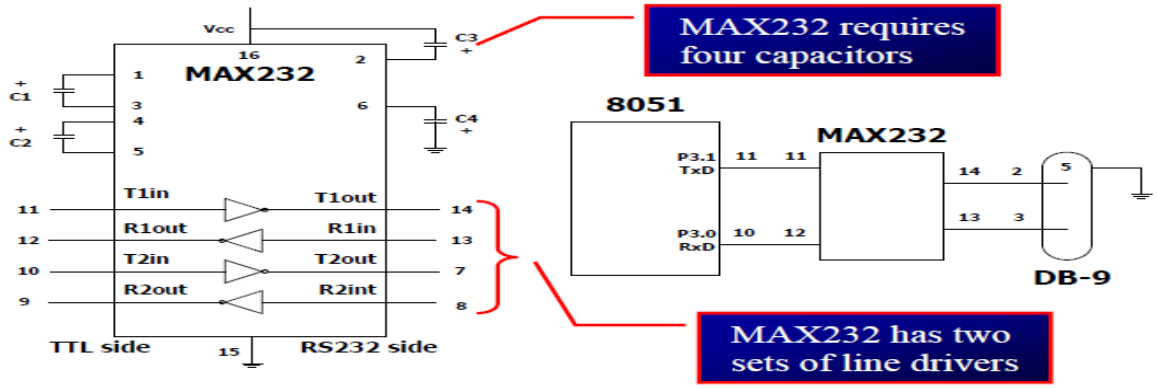


Fig 4.6.7:Connection diagram between MAX232,8051 and RS232

Baud rate calculation:

With XTAL = 11.0592 MHz, find the TH1 value needed to have the following baud rates. (a) 9600 (b) 2400 (c) 1200

Solution: The machine cycle frequency of 8051 = 11.0592 / 12 = 921.6 kHz,
and 921.6 kHz / 32 = 28,800 Hz is frequency by UART to timer 1 to set baud rate.
(a) 28,800 / 3 = 9600 where -3 = FD (hex) is loaded into TH1.
(b) 28,800 / 12 = 2400 where -12 = F4 (hex) is loaded into TH1.
(c) 28,800 / 24 = 1200 where -24 = E8 (hex) is loaded into

Baud Rate	TH1 (Decimal)	TH1 (Hex)
9600	-3	FD
4800	-6	FA
2400	-12	F4
1200	-24	E8

SCON register format(8 bit):

The SCON register format is as shown in below

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0	SCON.7	Serial port mode specifier
SM1	SCON.6	Serial port mode specifier
SM2	SCON.5	Used for multiprocessor communication
REN	SCON.4	Set/cleared by software to enable/disable reception
TB8	SCON.3	Not widely used
RB8	SCON.2	Not widely used
TI	SCON.1	Transmit interrupt flag. Set by HW at the begin of the stop bit mode 1. And cleared by SW
RI	SCON.0	Receive interrupt flag. Set by HW at the begin of the stop bit mode 1. And cleared by SW

- **SM0, SM1** They determine the framing of data by specifying the number of bits per character and the start and stop bits.

SM0	SM1	
0	0	Serial Mode 0
0	**1**	**Serial Mode 1, 8-bit data, 1 stop bit, 1 start bit**
1	0	Serial Mode 2
1	1	Serial Mode 3

Only mode 1 is of interest to us

M2

- SM2: This enables the multiprocessing capability of the 8051
- REN (receive enable): It is a bit-addressable register When it is high, it allows 8051 to receive data on RxD pin .If low, the receiver is disable.
- TI (transmit interrupt): When 8051 finishes the transfer of 8-bit character.It raises TI flag to indicate that it is ready to transfer another byte.TI bit is raised at the beginning of the stop bit.
- RI (receive interrupt): When 8051 receives data serially via RxD, it gets rid of the start and stop bits and places the byte in SBUF register. It raises the RI flag bit to indicate that a byte has been received and should be picked up before it is lost.RI is raised halfway through the stop bit.

SERIAL COMMUNICATION PROGRAMMING IN ASSEMBLY AND C:

Steps to programming the 8051 to transfer data serially

1. The TMOD register is loaded with the value 20H, indicating the use of the Timer 1 imode 2 (8-bit auto reload) to set the baud rate.

2. The TH1 is loaded with one of the values in table 5.1 to set the baud rate for serialdata transfer.
3. The SCON register is loaded with the value 50H, indicating serial mode 1, where an8-bit data is framed with start and stop bits.
4. TR1 is set to 1 start timer 1.
5. TI is cleared by the "CLR TI" instruction.
6. The character byte to be transferred serially is written into the SBUF register.
7. The TI flag bit is monitored with the use of the instruction JNB TI, target to see if the Character has been transferred completely.
8. To transfer the next character, go to step 5.

1. Write an 8051 ALP to receive data serially and put them in P1. Set the baud rate as 4800, 8bit data, 1stop and start bit.

   Algorithm:

   i. TMOD =20H, Timer1 in mode2
   ii. TH1 is loaded with one of values to set baud rate
   iii. SCON =50H
   iv. Start TR1
   v. RI is cleared with 'CLR RI' instruction
   vi. RI flag bit is monitored with use of instruction 'JNB RI, XX' to see if an entire character has been received yet.
   vii. When TI is raised, SBUF has the byte contents are moved into safe place
   viii. To receive next character.

```
LABEL      MNEMONIC AND OPERANDS
           MOV TMOD,#20H
           MOV TH1,#-6
           MOV SCON,#50H
           SETB TR1
HERE:      JNB RI,HERE
           MOV A, SBUF
           MOV P1,A
           CLR RI
           SJMP HERE
```

2. Write an ALP to implement following operation do a once, and b & c continuously 4800 baud rate.
   a. Send to the PC message "we are ready'
   b. Receive any data sent by PC and put it on LED's connected to P1
   c. Get data on swiitches connected to P2 and send to PC

```
LABEL       MNEMONIC AND OPERANDS
            MOV P2,#0FFH
            MOV TMOD,#20H
            MOV TH1,#0FAH
            MOV SCON,#50H
            SERTB TR1
            MOV DPTR,#MYDATA
H1:         CLR A
            MOVC A, @A+DPTR
            JZ B1
            ACALL SEND
            INC DPTR
            SJMP H1
B1:         MOV A,P2
            ACALL SEND
            ACALL RECV
            MOV P1,A
            SJMP B1
SEND:       MOV SBUF,A
H2:         JNB TI,H2
            CLR TI
            RET
RECV:       JNB RI,RECV
            MOV A,SBUF
            CLR RI
            RET
MYDATA:     BD " WE ARE READY", 0
            END
```

3. Write an 8051 ALP to read data from P1 and write it to P2 continually while giving a copy of it to the serial port to be transferred serially.

```
LABEL       MNEMONIC AND OPERANDS
            ORG 00
            LJMP MAIN
            ORG 23H
            LJMP SERIAL
            ORG 30H
MAIN:       MOV P1,#0FFH
            MOV TMOD,#20H
            MOV TH1,#-3
            MOV SCON,#50H
            MOV IE,#10010000B
            SETB TR1
BACK:       MOV A,P1
```

```
        MOV SBUF,A
        MOV P2,A
        SJMP BACK
        ORG 100H
        JB TI,TRANS
        MOV A,SBUF
        CLR RI
        RETI
TRANS:  CLR TI
        RETI
        END
```

4. Write a program using interrupts to do the following
   a. Receive data serially and send it to P0
   b. Have port P1 read and txed serially and a copy given to P2
   c. Make Timer 0 generate a square wave of 5 KHz on P0.1, when XTAL=11.059MHzand baud rate is 4800.

```
LABEL   MNEMONIC AND OPERANDS
        ORG 00
        LJMP MAIN
        ORG OOOBH
        CPL P0.1
        RETI
        ORG 23H
        LJMP SERIAL
        ORG 30H
        MOV P1,#OFFH
        MOV TMOD,#22H
        MOV TH1,#0F6HMOV SCON,#50H
        MOV TH0,#-92
        MOV IE,#10010010b
        SETB TR1
        SETB TR0
BACK:   MOV A,P1
        MOV SBUF, A
        MOV P2,A
        SKMP BACK
        ORG 100H
SERIAL: JB TI,TRANS
        MOV A,SBUF
        MOV P0,A
        CLR RI
        RETI
TRANS:  CLR TI
        RETI
        END
```