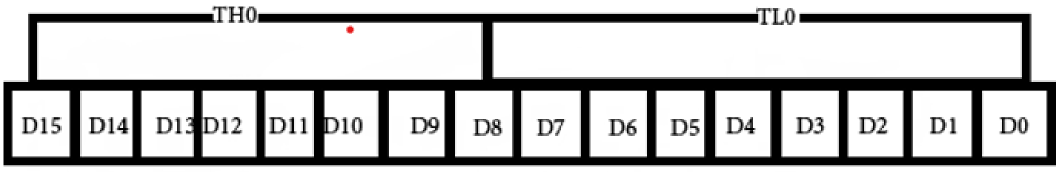
MODULE 4

PROGRAMMING ON CHIP RESOURCES AT ASSEMBLY LEVEL

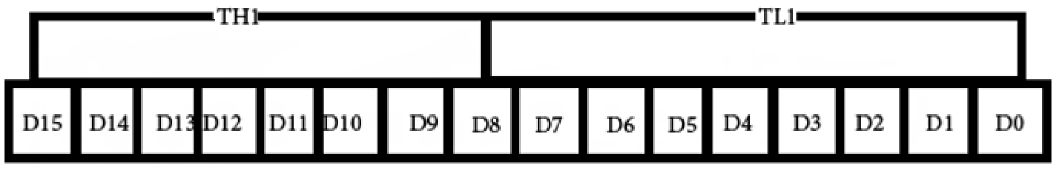
4.1 TIMERS AND COUNT ERS

- ❖ The 8051 has two counters/timers which can be used either as timer to generate a time delay or as counter to count events happening outside the microcontroller.
  - ❖ The 8051 has two timers: timer0 and timer1.
  - ❖ They can be used either as timers or as counters.
  - ❖ Both timers are 16 bits wide.
- ❖ Since the 8051 has an 8-bit architecture, each 16-bit is accessed as two separate registers of low byte and high byte.
  - ❖ First we shall discuss about Timer0 registers.
  - ❖ Timer0 registers is a 16 bits register and accessed as low byte and high byte.
  - ❖ The low byte is referred as a TL0 and the high byte is referred as TH0.
  - ❖ These registers can be accessed like any other registers.



Timer 0

- ❖ Timer1 registers is also a 16 bits register and is split into two bytes, referred to as TL1 and TH1.



Timer 1

❖ **TMOD (timer mode) Register:**

This is an 8-bit register which is used by both timers 0 and 1 to set the various timer modes. o In this TMOD register, lower 4 bits are set aside for timer0 and the upper 4 bits are set aside for timer1. o In each case, the lower 2 bits are used to set the timer mode and upper 2 bits to specify the operation.

GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
Timer1				Timer0			

TMOD Register

- In upper or lower 4 bits, first bit is a **GATE bit**.
- Every timer has a means of starting and stopping.
- Some timers do this by software, some by hardware, and some have both software and hardware controls.
- The hardware way of starting and stopping the timer by an external source is achieved by making **GATE=1** in the TMOD register.
- And if **GATE=0** then no need of external hardware to start and stop the timers.
- The second bit is **C/T bit** and is used to decide whether a timer is used as a **time delay generator or an event counter**.
- If this bit is **0 then it is used as a timer** and if it is **1 then it is used as a counter**.
- In upper or lower 4 bits, the last bits third and fourth are known as M1 and M0 respectively.
- These are used to select the timer mode.

Table 4.1.1 Timer Mode Operations

M	M	Mod	Operating Mode
0	0	0	3 bit timer mode 8 bit timer / counter THx with TLx as 5-bit prescalar
0	1	1	16 bit timer mode 16 bit timer/counters THx and TLx are cascaded; there is no prescalar
1	0	2	8 bit auto reload 8 bit auto reload timer/counter; THx holds a value that is to be reloaded into TLx each time it overflows
1	1	3	Split timer mode

❖ <u>**Mode 1**</u>

- It is a 16-bit timer; therefore it allows values from 0000 to FFFFH 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 "SETB TR0" for timer 0 and "SETB TR1" for timer 1.

- After the timer is started.
- It starts count up until it reaches its limit of FFFFH.
- When it rolls over from FFFF 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 stop the timer with the instructions "CLR TR0" or CLR TR1 for timer 0 and timer 1 respectively.
- Again, it must be noted that each timer flag TF0 for timer 0 and TF1 for timer1.
- After the timer reaches its limit and rolls over, in order to repeat the process the registers TH and TL must be reloaded with the original value and TF must be reset to 0.
- The figure 4.1.1 explains the Mode 1 operation of timer

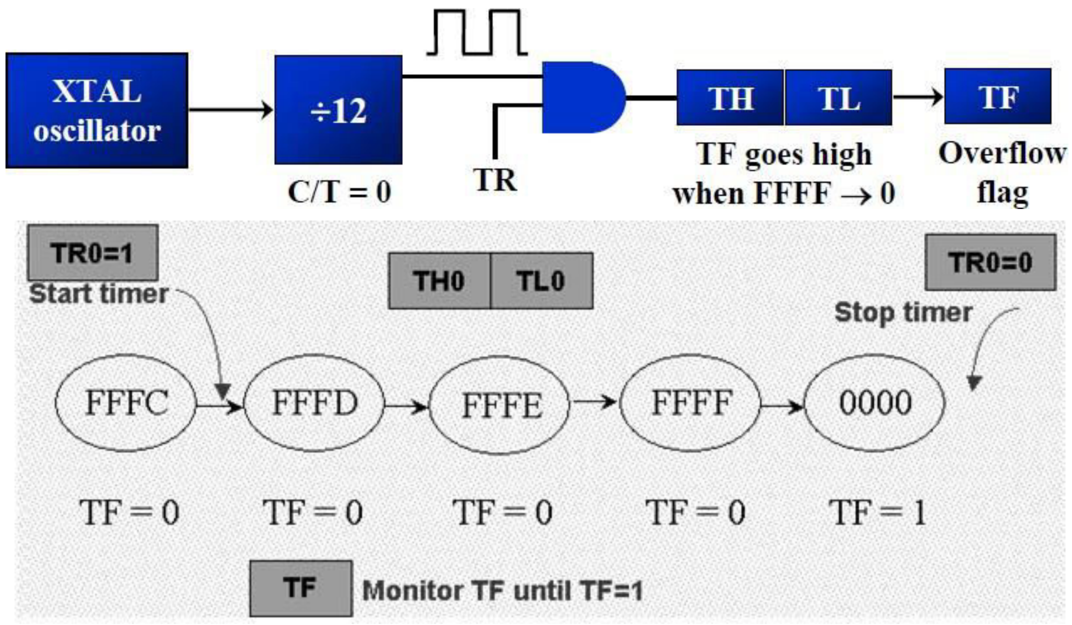


Fig 4.1.1 Mode 1 Operation of Timer

❖ **<u>Mode 2</u>**

❖ o The timer is an 8-bit device that only accepts values in the range of 00 to FFH to be entered into its register TH.
o The 8051 provides TL with a copy of the 8-bit value that was loaded into TH.
o Next, you need to set the timer.
o The commands "SETB TR0" for timer 0 and "SETB TR1" for timer 1 do this.
This resembles mode 1.
o The timer increments the TL register to begin counting up after it is started.

❖ The number counts up until the FFH limit is reached. when it shifts to 00 from FFH. The TF (timer flag) is raised.

o When utilizing timer 0, TF0 rises; when utilizing TF1, TF1 rises.
· The original value stored by the TH register is immediately reloaded into TL when the Tl register rolls from FFH to 00 and TF is set to 1.
o All that is required to repeat the operation is to clear TF and let it go, eliminating the need for the coder to reload the initial value.
o As a result, mode 2 reloads automatically, but mode 1 requires the programmer to reload TH and TL.
o The Mode 2 is explained in figure 4.1.2.

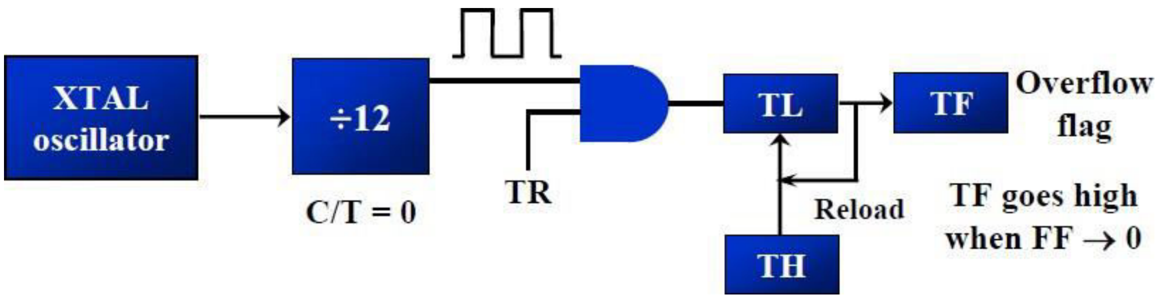
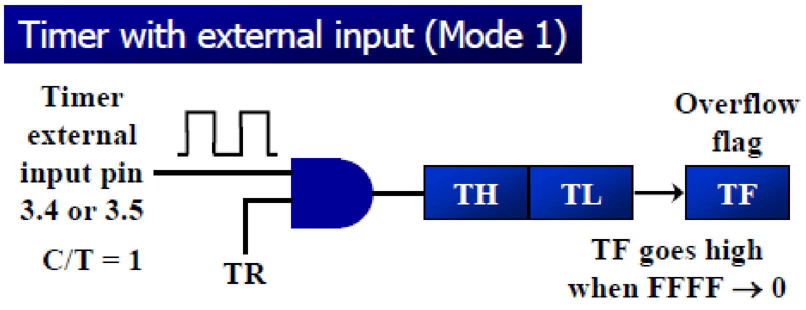


Fig 4.1.2 Mode 1 Operation of Timer


Timer with external input (Mode 1)
Timer external input pin 3.4 or 3.5
C/T = 1
TR
TH
TL
Overflow flag
TF
TF goes high when FFFF → 0

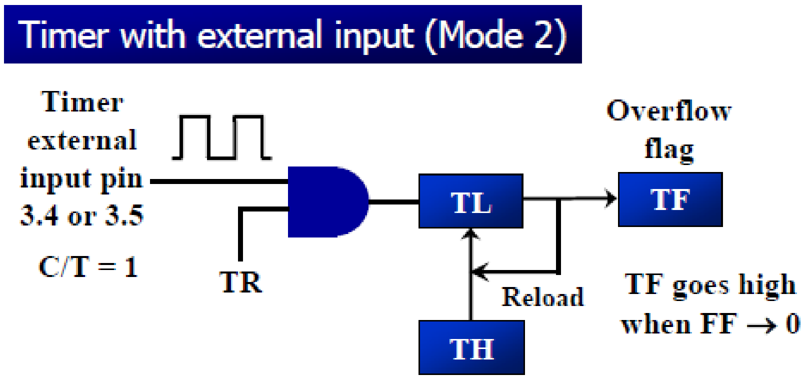

Timer with external input (Mode 2)
Timer external input pin 3.4 or 3.5
C/T = 1
TR
TL
Overflow flag
TF
Reload
TH
TF goes high when FF → 0

Figure 4.1.3 Mode 1 and Mode 2 Operation of Timer With External Input

❖ **Mode0**

o Mode 0 is identical to mode 1, with the exception that it uses a 13-bit timer rather than a 16-bit one. In TH-TL, the 13-bit counter may store values from 0000 to 1FFFH. As a result, the timer rolls over to 0000 and TF is raised when it reaches its maximum of 1FFH.

❖

❖ **Mode3**

- Mode 3 is also known as a split timer mode.
- Timer 0 and 1 may be programmed to be in mode 0, 1 and 2 independently of similar mode for other timer.
- This is not true for mode 3; timers do not operate independently if mode 3 is chosen for timer 0.
- Placing timer 1 in mode 3 causes it to stop counting; the control bit TR1 and the timer 1 flag TF1 are then used by timer0.

❖ **TCON register**

- Bits and symbol and functions of every bits of TCON are as follows:

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TCON Register

Table 4.1.2 : TCON Register Description

BIT	Symbol	Functions
7	TF1	Timer1 over flow flag. Set when timer rolls from all 1s to 0. Cleared When the processor vectors to execute interrupt service routine Located at program address 001Bh.
6	TR1	Timer 1 run control bit. Set to 1 by programmer to enable timer to count; Cleared to 0 by program to halt timer.
5	TF0	Timer 0 over flow flag. Same as TF1.
4	TR0	Timer 0 run control bit. Same as TR1.
3	IE1	External interrupt 1 Edge flag. Not related to timer operations
2	IT1	External interrupt1 signal type control bit. Set to 1 by program to Enable external interrupt 1 to be triggered by a falling edge signal. Set To 0 by program to enable a low level signal on external interrupt1 to generate an interrupt.
1	IE0	External interrupt 0 Edge flag. Not related to timer operations.
0	IT0	External interrupt 0 signal type control bit. Same as IT0

Calculating Time Delay: (By Assuming XTAL Frequency as 11.0592MHz)

In Hex	**In Decimal**
(FFFF – YYXX + 1) * 1.085us **Where YYXX are TH, TL initial values respectively.**	**Convert YYXX values of the TH, TL registers to decimal to get a NNNNN decimal number, then** **(65536 – NNNNN) * 1.085 us**

Finding the values to be loaded into the timer:

❖ Assuming XTAL = 11.0592MHz

1. Divide the desired time delay by 1.085us
2. Perform 65536 – n, where n is the decimal value we got from step 1
3. Convert the result of step 2 to hex, where YYXX is the initial hex value

Example

- **Indicate which mode and which timer are selected for each of the following.**
  (a) MOV TMOD, #01H (b) MOV TMOD, #20H (c) MOV TMOD, #12H

  Solution:

  We convert the value from hex to binary. From Figure we have:
  (a) TMOD = 00000001, mode 1 of timer 0 is selected.
  (b) TMOD = 00100000, mode 2 of timer 1 is selected.
  (c) TMOD = 00010010, mode 2 of timer 0, and mode 1 of timer 1 are

- **Find the timer's clock frequency and its period for various 8051-based system, with the crystal frequency 11.0592 MHz when C/T bit of TMOD is 0.**
  Solution:

  (1/12)× 11.0529 MHz = 921.6 MHz;
  T = 1/921.6 kHz = 1.085 us selected.

- **Write a program to create square wave of 50% duty cycle (with equal portions high and low) on the P1.5 bit use Timer 0 to generate the time delay.**
  Solution:

```
        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
        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
```

- **Write a program to continuously generate a square wave of 2 kHz frequency on pin P1.5 using timer 1. Assume the crystal oscillator frequency to be 12 MHz.**
  The period of the square wave is T = 1/(2 kHz)
  = 500 us. Each half pulse = 250 us.
  The value n for 250 us is: 250 us /1 us = 250
  65536 - 250 = FF06H.

TL = 06H and TH = 0FFH.

```
        MOV TMOD,#10            ;Timer
1,  mode  1  AGAIN:  MOV  TL1,#06H
                                ;TL0    =
06H
        MOV TH1,#0FFH           ;TH0 = FFH
        SETB TR1                ;Start timer 1
BACK: JNB TF1,BACK              ;Stay until timer
        rolls over CLR TR1      ;Stop timer 1
        CPL P1.5                ;Complement P1.5 to get Hi, Lo
        CLR TF1                 ;Clear timer flag 1
        SJMP AGAIN              ;Reload timer
```

❖ **Write a program segment that uses timer 1 in mode 2 to toggle P1.0 once whenever the counter reaches a count of 100. Assume the timer clock is taken from external source P3.5 (T1).**

The TMOD value is 60H
The initialization value to be loaded into TH1 is 256 - 100 = 156 = 9CH

```
        MOV TMOD,#60h           ;Counter1, mode 2, C/T'= 1
        MOV TH1,#9Ch            ;Counting 100 pulses
        SETB P3.5               ;Make T1 input
        SETB TR1                ;Start timer 1
BACK: JNB TF1,BACK              ;Keep doing it
        if TF = 0 CPL P1.0      ;Toggle   port
        bit
        CLR TF1                 ;Clear timer overflow flag
        SJMP BACK               ;Keep doing it
```

4.2 SERIAL COMMUNICATION:

Baud Rate:

- ❖ The 8051 transfers and receives data serially at many different baud rates.
- ❖ The baud rate in the 8051 is programmable.
- ❖ This is done with the help of Timer 1.
- ❖ The relationship between the crystal frequency and the baud rate is
    - o For XTAL = 11.0592 MHz, the machine cycle frequency is 921.6kHz
    - o The UART circuitry divides the machine cycle frequency of 921.6kHz by 32 once more before it is used by Timer 1 to set the baud rate. Therefore it gives 28,800Hz.
    - o When Timer 1 is used to set the baud rate it must be programmed in mode 2, i.e., 8-bit, auto reload.
    - o To get baud rates compatible with the PC, TH1 should be loaded with the values shown in the following table by assuming XTAL=11.0592MHz.

Table 4.2.1: Baud Rate Value For TH1

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

SBUF Register

- ❖ SBUF is an 8-bit register used solely for serial communication in the 8051.
- ❖ For a byte of data to be transferred via the TxD line, it must be placed in the SBUF register.
- ❖ Similarly, SBUF holds the byte of data when it is received by the RxD line.
- ❖ SBUF can be accesed like any other registers in the 8051

```
MOV SBUF, #'D'
        MOV   SBUF,   A
        MOV A, SBUF
```

- ❖ The moment a byte is written into SBUF, it is framed with the start and stop bits and transferred serially via the TxD pin
- ❖ Similarly when the bits are received serially via RxD, the 8051 deframes it by eliminating the start and stop bits, making a byte out of the data received, and then placing it in the SBUF

SCON Register:

- ❖ The SCON register is an 8-bit register used to program the start bit, stop bit and data bits of data framing, among other things.

- ❖ **SM0,SM1:**
  - These two bits determines the framing of data by the specifying the number of bits per character, and the start and stop bits.
  - They take the following combinations

Table 4.2.2: Serial Mode Operation

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

  - Serial Mode 0, 2 and 3 are not commonly used nowadays.
  - Serial Mode 1 is compatible with the COM port of IBM/PC's
  - Serial Mode 1, allows the baud rate to be variable and is set by Timer 1 of the 8051
  - In Serial Mode 1, for each character a total of 10 bits are transferred
- ❖ **SM2**
- ❖ Baud Rate Calculation for SMOD (Serial Mode) = 1

Program to Transfer Data Serially:

1. The TMOD register is loaded with the value 20H, indicating the use of Timer 1 in mode 2 to set baud rate
2. The TH1 is loaded with the value to set the baud rate for serial data transfer
3. The SCON register is loaded with the value 50H, indicating Serial Mode 1
4. TR1 is set to 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 instruction 'JNB TI, XX'
8. To Transfer next character go to

Program to Receive Data Serially:

1. The TMOD register is loaded with the value 20H, indicating the use of Timer 1 in mode 2 to set baud rate
2. The TH1 is loaded with the value to set the baud rate for serial data transfer
3. The SCON register is loaded with the value 50H, indicating Serial Mode 1 and receive enable is turned ON
4. TR1 is set to start Timer 1
5. RI is cleared by the 'CLR RI' instruction
6. The RI flag bit is monitored with the instruction 'JNB RI, XX'
7. When RI is raised, SBUF has the byte
8. To Receive next character goto Step 5

Doubling the Baud Rate:

- ❖ Use a higher frequency crystal
- ❖ Change bit in the PCON register, shown below

PCON Register

7	6	5	4	3	2	1	0
SMOD	**-**	**-**	**-**	**GF1**	**GF0**	**PD**	**IDL**

❖ Baud Rate Calculation for SMOD (Serial Mode) = 0

Machine Cycle Frequency $= \frac{Crystal\ \ Frequency}{12*32}$

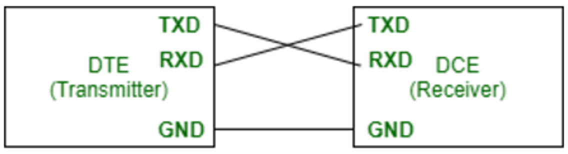
❖ Machine Cycle Frequency $= \frac{Crystal\ \ Frequency}{12*16}$

Table 4.2.3: Baud Rate Comparison For SMOD=0 Vs SMOD=1

TH1 (Decimal)	**TH1 (Hex)**	**SMOD = 0**	**SMOD = 1**
-3	FD	9600	19200
-6	FA	4800	9600
-12	F4	2400	4800
-24	E8	1200	2400

4.3 RS232 STANDARD:

RS232 is an Interface and the protocol between DTE(data terminal equipment) and DCE(data communication equipment) using serial binary data exchange. Here C is used for the current version. *Universal Asynchronous Data Receiver & Transmitter (UART)*, attached in a motherboard, used in connection with RS232 for transmitting data to any serial device like modem or printer from its DTE interface.



RS232 PROTOCOL

Electrical Specifications :
1.Voltages:
There can be two states in the signal level of RS232C pins.

- **Mark state –** It is the high bit which is represented by binary 1 and have negative voltages. Its voltage limits for transmitting signal ranges from -5 to -15V. Its voltage limits for receiving signals ranges from -3 to -25V.
- **Space state –** It is the low bit which is represented by binary 0 and have positive voltages. Its voltage limits for transmitting signal ranges from +5 to +15V. Its voltage limits for receiving signals ranges from +3 to +25V.

2.Cable sand Wires :

The maximum cable length for RS232C is equals to 15.24 meters or equal to the capacitance of 2500pF. Limits for the impedance of wires ranges from 3 ohms to 7 ohms.

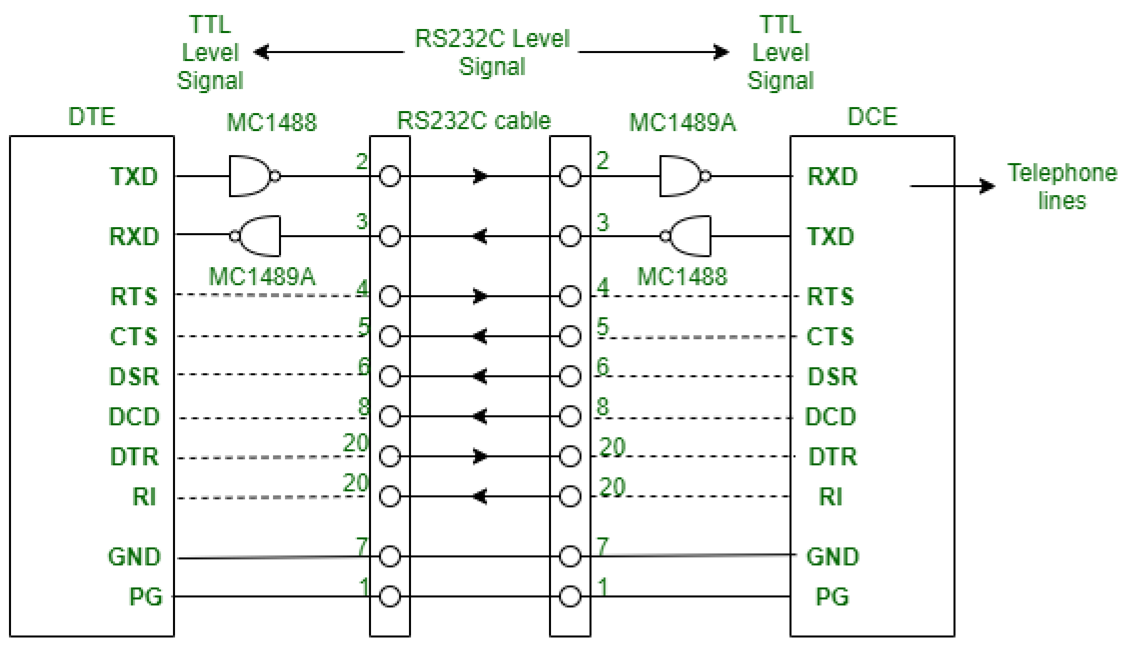
3.Data and Slew rates :

Rate of data transmission through RS232C is up to 20Kbps. The rate of change in signal levels ie. slew rate is up to 30V/microsecond.

4.Current :

Maximum current rating is 3Amps at the maximum operating voltage of 250V AC.

Pins and Working :



The RS232C Pins and Signals

RS232C requires 25 pins connector for connecting DTE and DCE. Here is the list of pins and signals of RS232C and the connection between DTE and DCE using drivers and receivers.

1. **TXD&RXD**

   Transmit Data and Receive Data on the DTE are the serial data lines. These lines have opposite functions on a DCE. TXT sends outgoing data to DCE. RXD receives incoming data from DTE.

2. **RTS & CTS**

   Transmitter activates the Request to Send when it requires to transmit data over the line. The line itself gets deactivated when the communication stops. Receiver activates the Clear To Send to tell the transmitter whether it is ready or not to receive the data. It remains active during the transmission.

3. **DTR & DSR**

   Through the Data Terminal Ready line, DTE informs the DCE that it is in online mode and the process of communication can occur. The main task of Data Set Ready signal is to inform that DCE is ready for communication.

4. **DCD–**
   DCE activates the Data Carrier Detect in order to show that it has been connectedtoDTE.

1. **RI** –
   When an incoming call on the telephone line is detected by DCE, then the Ring Indicator gets activates.

Handshaking:

Before the actual data transfer, signals are transmitted from DTE to DCE in order to make connections by a process known as handshaking. Following is the sequence of signal handshaking:

- Initially, the computer activates RTS signal to modem when a data is transferred from computer to modem.
- Modem in turn activates the DCD and then the CTS gets activated.
- Computer then sends data on TXD. After the data transmission is completed, the computer deactivates the RTS which causes the modem to deactivate CTS.

Applications :

However, most of functions performed by RS232C has been taken by the USB, but they are still successful in performing following applications.

1. It is used in establishing communication between the computer and embedded systems.
2. Due to its lower costs, It plays a vital role in CNC machines and servo controllers
3. Some microcontroller boards and PLC machines use RS232C.
4. RS232C ports are used to communicate in headless systems in the absence of any network connection.
5. Many Computerized Numerical Control Systems are contains RS232C port.

Limitations :

1. It cannot be used for chip to chip or chip to sensor device communication
2. It degrades the performance of the system in the presence of noise and requires shorter cables due to having common grounds between DTE and DCE
3. The cost of system increases as RS232C interface needs separate transceiver chips.
4. Its performance degrades to short distances only when transfer speed is high.