PART -B

SERIAL DATA TRANSFER SCHEMES

Contents at a glance:

- ✓ Asynchronous and synchronous data transfer schemes
- ✓ RS - 232C Serial data standard
- ✓ 8251 USART Architecture and Interfacing
- ✓ Sample program of serial data transfer

Introduction to Serial communication:

- ➢ Most of the microprocessors are designed for parallel communication.
- ➢ In parallel communication number of lines required to transfer data depend on the number of bits to be transmitted.
- ➢ For transmitting data over long distance, using parallel communication is impractical due to the increase in cost of cabling.
- ➢ In such cases serial communication is used.
- ➢ In serial communication one bit is transferred at a time over a single line.
- ➢ Serial communication can be classified on the basis how transmission occurs.

1. **Simplex:** In simplex, the hardware such that data transfer takes place in only one direction.
   Ex: Computer to Printer communication
2. **Half Duplex:** The half duplex transmission allows the data transfer in both direction but not simultaneously.
   Ex: Walkie talkie
3. **Full Duplex**: It allows the data transfer in both direction simultaneously.
   Ex: Telephone lines

Data transfer schemes:

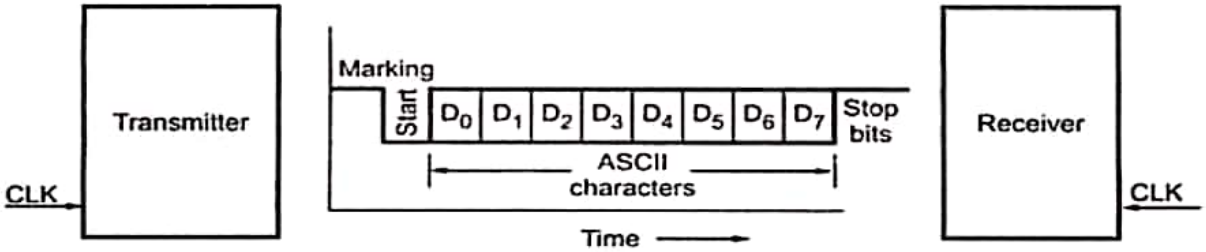
- ➢ The data in the serial communication may be sent in two formats:

1. Asynchronous
2. Synchronous

Asynchronous:

- ➢ Asynchronous formats are character oriented.
- ➢ In this type the bits or character or data word are sent at constant rate, but characters can come at any rate (asynchronously) as long as they do not overlap.
- ➢ When no characters are being sent a line stays high at logic1 called ***mark***, logic0 is called ***space***.
- ➢ The beginning of a character is indicated by start bit which is always low.
- ➢ This is used to synchronize the transmitter and receiver.
- ➢ After the start bit the data bits are sent with least significant bit first followed by one or more stop bits (active high).
- ➢ The stop bits indicate the end of character.
- ➢ The combination of start but, character and stop bits is known as frame.

- The start and stop bits carry no information, but are required because of asynchronous nature of data.



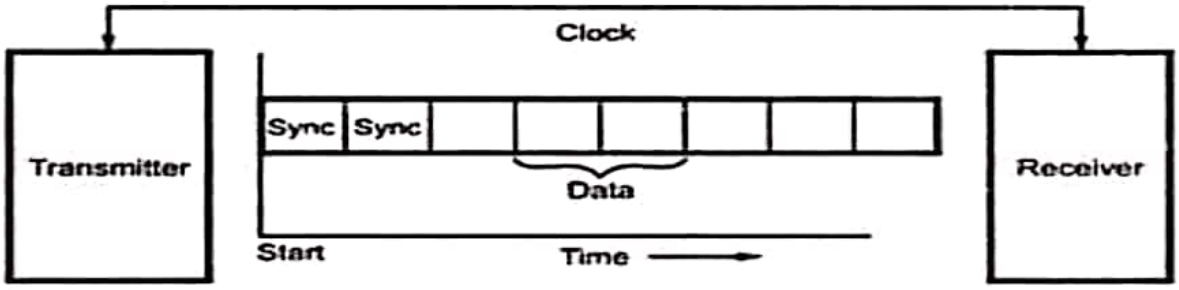
Transmission format for asynchronous transmission

Synchronous:

- The start and stop bits in each frame of asynchronous format represents wasted overhead bytes that reduce overall character rate.
- These start and stop bits can be eliminated by synchronizing receiver and transmitter.
- They can be synchronized by having a common clock signal.
- Such a communication is called **synchronous serial communication.**
- In this transmission synchronous bits are inserted instead of start and stop bits

S.No	Asynchronous	Synchronous
1.	Transmitters and receivers are not synchronized by clock.	Transmitters and receivers are synchronized by clock.
2.	Bits of data are transmitted at constant rate.	Data bits are transmitted with synchronization of clock.
3.	Character may arrive at any rate at receiver.	Character is received at constant rate.
4.	Data transfer is character oriented.	Data transfer takes place in blocks
5.	Start and stop bits are required to establish communication of each character.	Start and stop bits are not required to establish communication of each character. Synchronization bits are required to transfer the data block.
6.	Used in low-speed transmission at about speed less than 20 Kbits/sec.	Used in high speed transmissions.

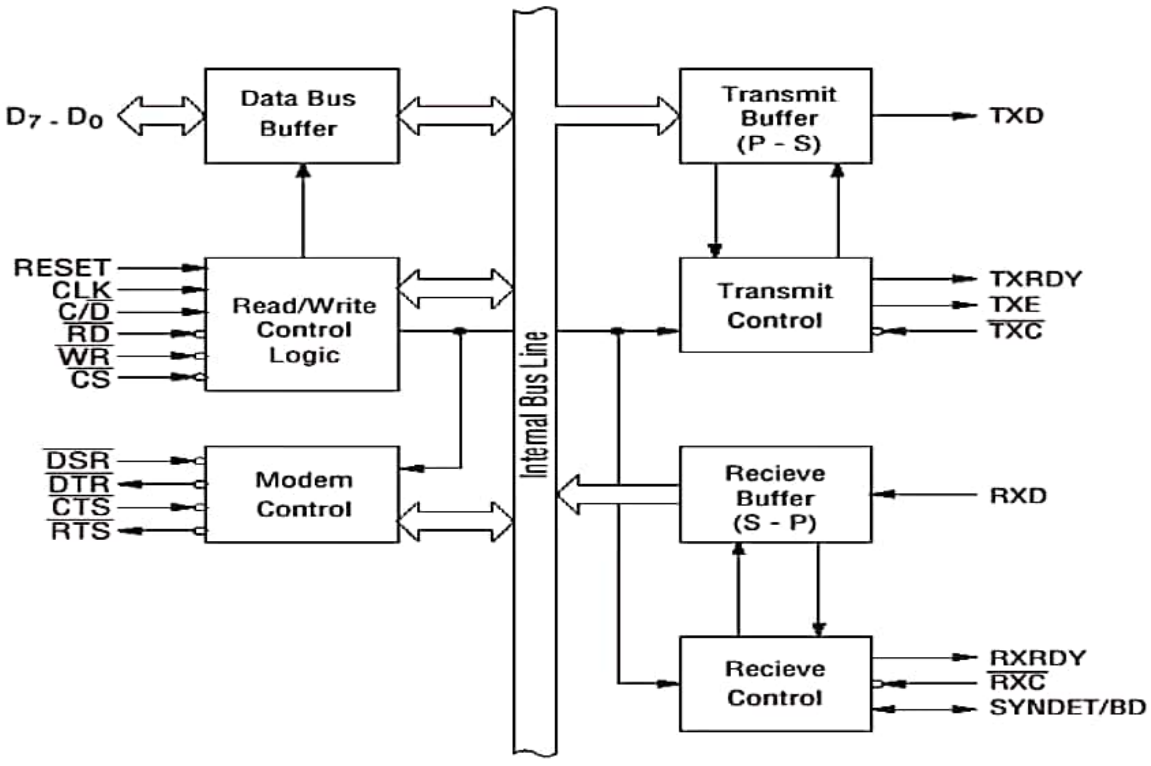
- The data rate can be expressed as bit/sec or character/sec.
- The term bit/sec is also called **baud rate**.



Synchronous transmission format

8251- Universal Synchronous Asynchronous Receiver-Transmitter (USART) Architecture:

- To implement serial communication in microprocessor system we need basically two devices:
  1. Parallel to Serial Converter
  2. Serial to Parallel Converter
- To transmit byte data it is necessary to convert byte into eight serial bits.
- This can be done by using the parallel to serial converter.
- Similarly at the reception these serial bits must be converted into parallel 8-bit data.
- The serial to parallel converter is used to convert serial data bits into parallel data.
- The devices are designed for this purpose are called universal synchronous asynchronous receiver-transmitter.
- These devices are software programmable for number of data bits, parity and number of stop bits.



Data Bus Buffer:

- This bidirectional 8-bit buffer is used to interface 8251 to the system data bus.
- Along with the data, control word, command words and status information are also transferred through the Data Bus Buffer.

Read/Write Control Logic:

- This functional block accepts inputs from the system control bus and generates control signals for over all operation.
- It contains the control word register and command word register that stores the various control formats for the device functional definitions.

Transmit Buffer:

- The transmit buffer accepts parallel data from CPU, adds the appropriate framing information, serializes it and transmits it on the TxD pin on the failing edge of TxC'

Transmit Control:

- It manages all activities associated with the transmission of serial data.
- It accepts and issues signals both externally and internally to accomplish this function.

Receiver Buffer:

- The receiver buffer accepts serial data on the RxD line, coverts this serial data to parallel formats, checks for bits or characters that are unique to the communication technique and sends an "assembled" character to CPU.

Receiver Control:

- It manages all activities associated with the receiving of serial data.
- Along with data reception, it does false start bit detection, parity error detection, framing error detection, sync detection and break detection.

Modem Control:

- The 8251 has set of control inputs and outputs that can be used to simplify the interface to almost any modem.
- **Data Bus (D_0-D_7):** This pin allows transfer of data bytes between CPU and the 8251A.
- **Read (RD')** : A low on this input allows CPU to read data or status byte from 8251A.
- **Write (WR')** : A low on this input allows CPU to write data or command word to 8251A.
- **Clock (CLK):** The CLK input is used to generate internal device timing.
- **RESET:** A high on this input force 8251A into an "IDLE" mode.
- **Control/Data (C/ D'):** This input in conjunction with the WR' and the RD' inputs informs the 8251A that the word on the DATA BUS is either a data character, control word or status information.

C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	Operation
0	0	1	CPU reads data from USART
0	1	0	CPU writes Data to USART
1	0	1	CPU reads Status from USART
1	1	0	CPU writes command to USART
X	1	1	USART Bus floating

- **Chip Select (CS'):** A low on this input allows communication between CPU and 8251A.

Modem Control Signals:

- The 8251A has a set of control inputs and outputs that can be used to simplify the interface to almost any modem.
- **Data Set ready (DSR'):** This input signal is used to test modem conditions such as Data Set Ready.
- **Data Terminal Ready (DTR'):** This output signal is used to tell modem that Data terminal is ready
- **Request to Send (RTS'):** This output signal is asserted to begin transmission.
- **Clear to Send (CTS'):** A low on this input enables the 8251A to transmit serial data if the TxE bit in the command byte is set to one.

Transmitter Signals:

- **Transmit Data (TxD):** This output signal outputs a composite serial stream of data on the falling edge of TxC.
- **Transmitter Ready (TxRDY):** This output signal indicates the CPU that the transmitter is ready to accept a data character.
- **Transmitter Empty (TxE):** This output signal indicates that the transmitter has no character to transmit.
- **Transmitter Clock (TxC):** This clock input controls the rate at which the character is to be transmitted.

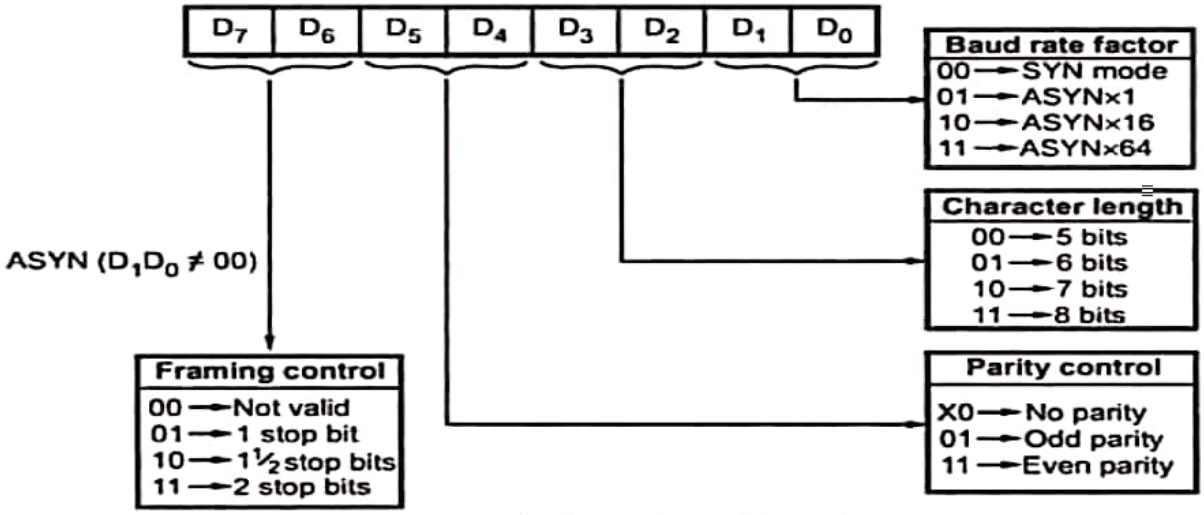
Receiver Signals

- **Receiver Data (RxD):** This input receives a composite serial stream of data on the raising edge of RxC.
- **Receiver Ready (RxRDY):** This output indicates that 8251A contains a character that is ready to be input to the CPU.
- **Receiver Clock (RxC):** This clock input controls the rate at which the character is to be received.
- **Sync/ Break Detect (SYNDET/BD):** This pin is used in synchronous mode for detection of synchronous characters and may be used as input or output.
- **In asynchronous mode this pin goes high if receiver line stays low for more than 2 character times. It then indicates a break in the data stream.**

- **8251A Control Words:** The control words defines the complete functional definition of 8251A and they must be loaded before any transmission or reception.
- The control words of 8251A are split into two formats:
  - Mode Instruction
  - Command Instruction

Mode Instruction:

- The instruction can be considered as four 2-bit fields
- The first 2-bit field (D_1-D_0) determines whether the USART is to operate in the synchronous (00) or asynchronous mode.
- In the asynchronous mode, this field determines the division factor for clock to decide baud rate.

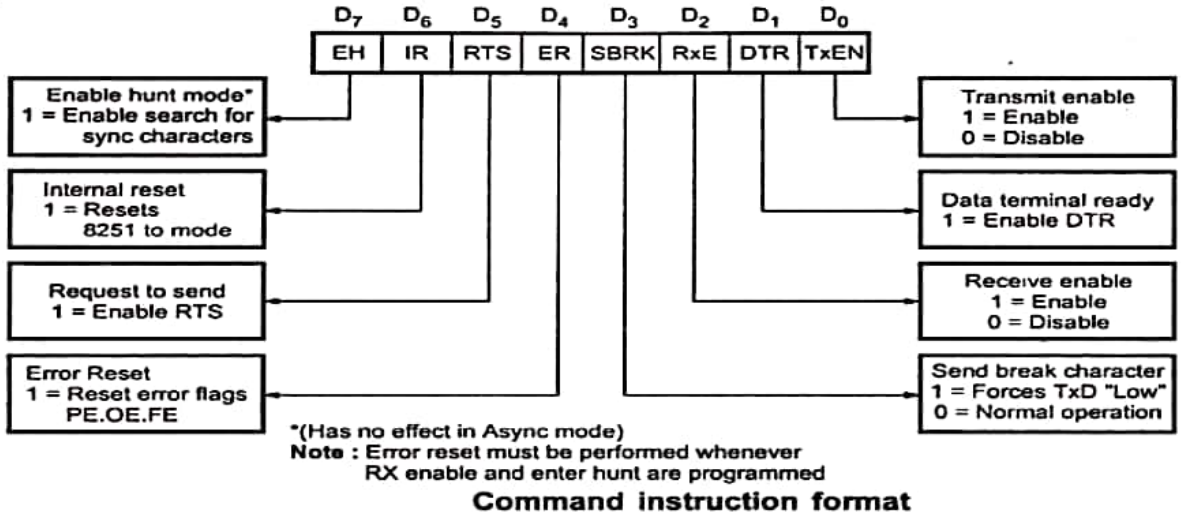


Mode instruction format

- The second 2-bit field (D_3-D_2) determines number of data bits in character. With this 2-bit field we can set character length from 5-bits to 8-bits.
- The third 2-bit field (D_5-D_4) controls the parity generation. The parity bit is added to the data pins only if parity is enabled.
- The last field (D_7-D_6) has two meanings depending on whether operation is to be in the synchronous or asynchronous mode.
- For asynchronous mode (D_1 D_0≠00) it controls the number of STOP bits to be transmitted with the character.
- In synchronous mode (D_1 D0=00) this field controls the synchronizing process.
- It decides whether to operate with external or internal synchronization and whether to transmit single synchronizing character or two synchronizing characters.
- Command Instruction: After the mode instruction, command character should be issued to the USART.
- It controls the operation of the USART within the basic frame work established by the mode instruction.
- It does function such as:

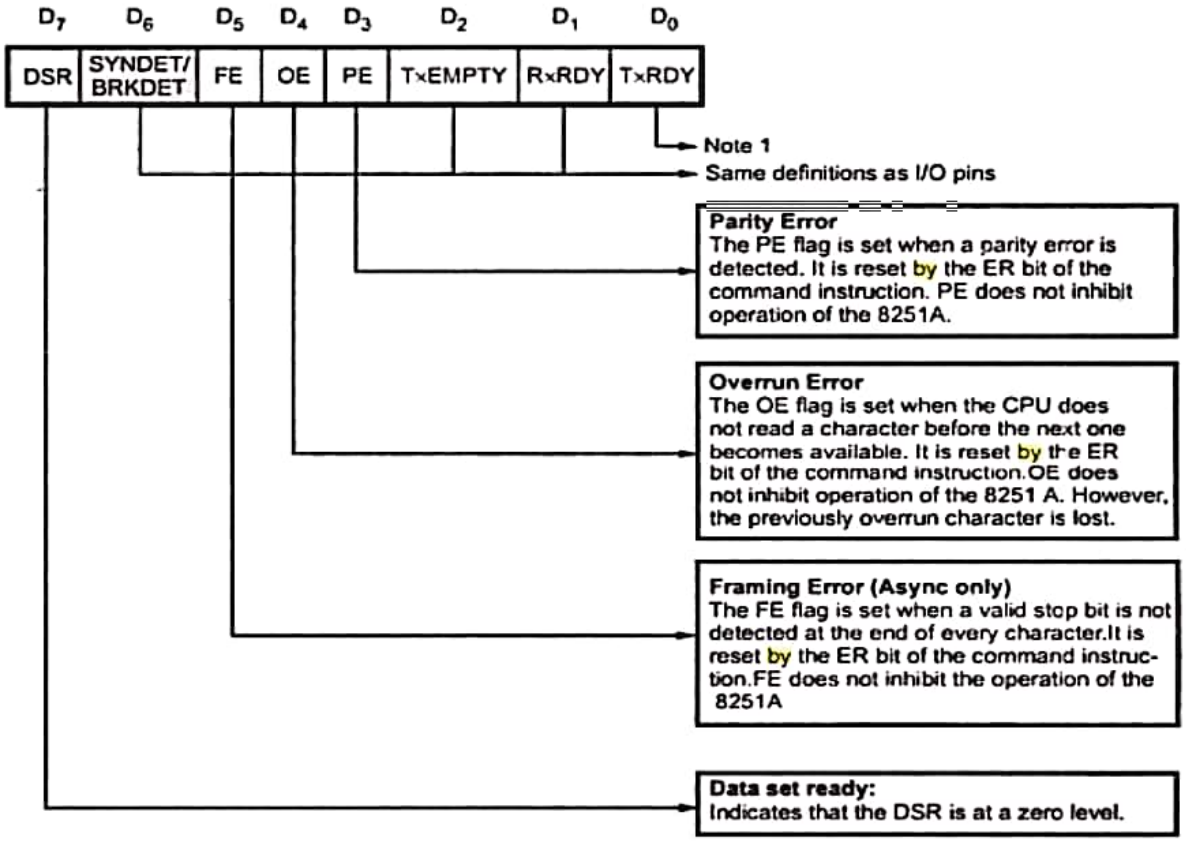
- Enable Transmit/ Receive
- Error Reset
- Modem Control

Command Instruction:



Command instruction format

- 8251A Status Word: In the data communication systems it is often necessary to examine the "status" of the transmitter and receiver.
- It is also necessary for CPU to know if any error has occurred during communication.
- The 8251A allow the programmer to read information from the status register any time during functional operation



Error Definitions:

- **Parity Error:** At the time of transmission of data an even or odd parity bit is inserted in the data stream. At the receiver end, if parity of the character does not match with the pre-defined parity, parity error occurs.
- **Overrun Error:** In the receiver section received character is stored in the receiver buffer. The CPU is supposed to read this character before reception of next character. But if CPU fails in reading the character loaded in the receiver buffer, the next the received character replaces the previous one and the OVERRUN Error occurs.
- **Framing Error:** If valid stop bit is not detected at the end of each character framing error occurs.
- All these errors when occur set the corresponding bits in the status register.
- These error bits are reset by setting ER bit in the command instruction.

Problem: Design the hardware interface circuit for interfacing 8251 with 8086. Set the 8251A in asynchronous mode as transmitter and receiver with even parity enable, 2 stop bits, 8-bit character length, frequency 160 kHz and baud rate 10 K.

a) Write an ALP to transmit 100 bytes of data string at location 2000:5000H.

b) Write an ALP to receive 100 bytes of data string and store it at location 3000:4000H.

Solution: Asynchronous Mode Control Word

D 7	D6	D5	D4	D3	D2	D1	D0	
1	1	1	1	1	1	1	0	=0FEH
2 Stop Bits		Even Parity Enabled		8- bit Format		CLK scaled by 16		

Command Word for Transmission

D7	D6	D5	D4	D3	D2	D1	D0	
0	0	0	1	0	0	0	1	=11H
EH	IR	RTS	ER	SBRK	RxE	DTR	TxEN	

ALP to initialize 8251 and transmit 100 bytes of data

```
ASSUME CS:CODE

CODE    SEGMENT

START: MOV AX,2000H

MOV DS,AX       ; DS points to byte string segment

MOV SI,5000H    ; SI points to byte string

MOV CL,64H      ; Length of string in CL (hex)
```

```
MOV AL,0FEH   ; Mode control word to
OUT 0FEH,AL   ; D0–D7
MOV AX,11H    ; Load command word
OUT 0FE,AL    ; to transmit enable and error reset
WAIT: IN AL,0FEH        ; Read status
AND AL,01H    ; Check transmitter enable
JZ WAIT                 ; bit, if zero wait for the transmitter to be ready
MOV AL,[SI]   ; If ready, first byte of string data
OUT 0FCH, AL  ; is transmitted
INC SI        ; Point to next byte
DEC CL        ; Decrement counter
JNZ WAIT                ; If CL is not zero, go for next byte
MOV AH, 4CH   : If CL is zero, return to DOS
INT 21H
CODE  ENDS
END  START
```

b. Write an ALP to receive 100 bytes of data string and store it at location 3000:4000H.

Command Word for Receiving

D7	D6	D5	D4	D3	D2	D1	D0	
0	0	0	1	0	1	0	0	=14H
EH	IR	RTS	ER	SBRK	RxE	DTR	TxEN	

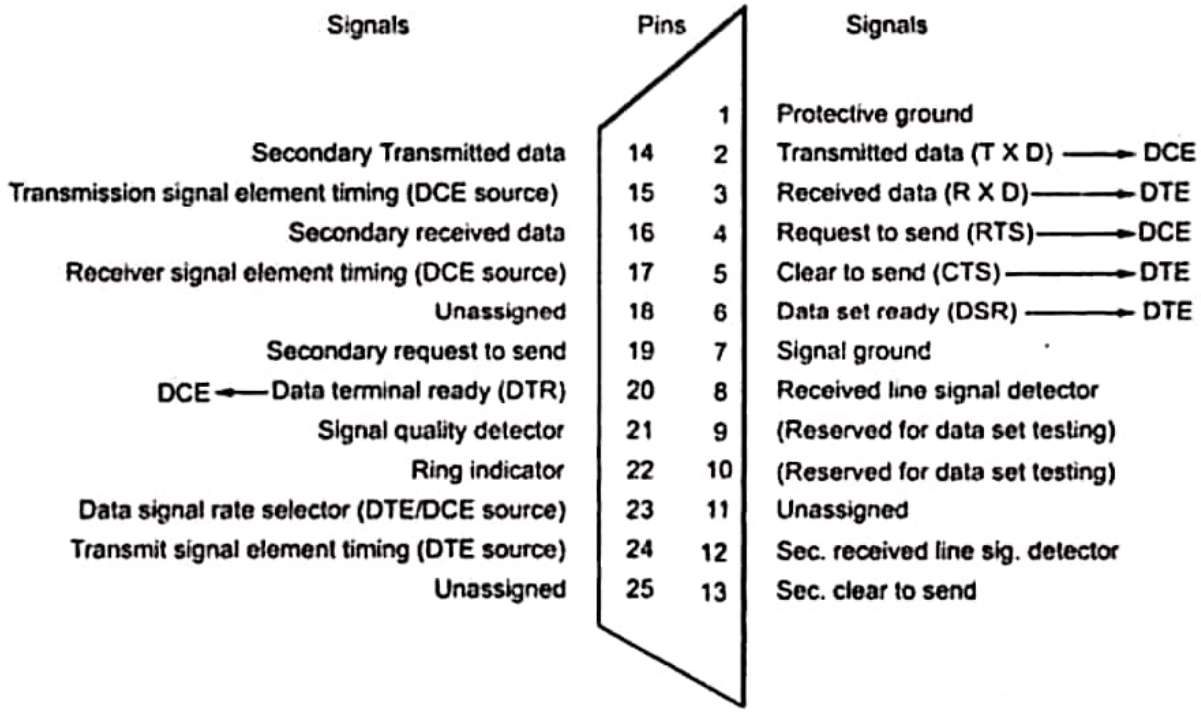
An ALP to initialize 8251 and receive 100 bytes of data

```
ASSUME CS:CODE
CODE  SEGMENT
START : MOV AX,3000H
MOV DS, AX              ; Data segment set to 3000H
MOV SI,4000H  ; Pointer to destination offset
MOV CL,64H    ; Byte count in CL
MOV AL,7EH    ; Only one stop bit for
```

```
OUT 0FEH, AL    ; receiver is set
MOV AL, 14H     ; Load command word to enable
OUT 0FEH, AL    ; the receiver and disable transmitter
NXTBT : IN AL,0FEH      ; Read status
AND AL,38H      ; Check FE, OE and PE
JZ READY        ; If zero, jump to READY
MOV AL,14H ; If not zero, clear them
OUT 0FEH,AL
READY: IN AL,0FEH       ; Check RXRDY.
AND AL,02H ; if receiver  is not ready
JZ READY        ; wait
IN AL,0FCH      ; If it is ready,
MOV [SI],AL     ; receive the character
INC SI    ; Increment pointer to next byte
DEC CL  ; Decrement counter
JNZ NXTBT       ; Repeat, if CL is not zero
MOV AH,4CH
INT 21H
CODE    ENDS
END             START
```

RS-232C Serial Data Standard:

- In response to the need for signals and handshake standards between DTE(Data Terminal Equipment) and DCE(Data Circuit-Terminating Equipment), the Electronic Industries Association (EIA) introduced EIA standard RS-232 in 1962.
- It is widely accepted for single ended data transmission over short distances with low data rates.
- This standard describes the function of 25 signals and handshake pins for serial data transfer

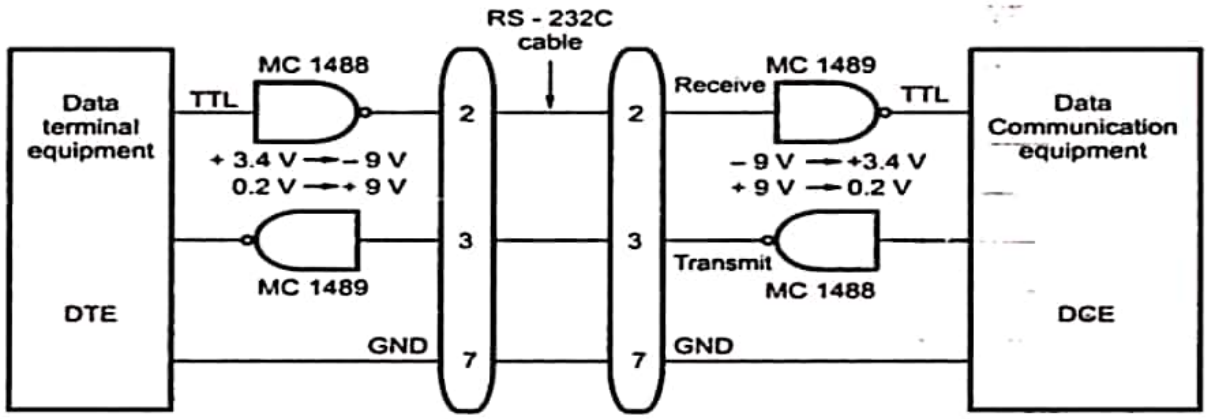


RS 232C 25 pin connector

Pin Number	Common Name	RS-232C Name	Description	Signal Direction On DEC
1		AA	Protective ground	
2	TxD	BA	Transmitted data	IN
3	RxD	BB	Received data	OUT
4	$\overline{\text{RTS}}$	CA	Request to send	IN
5	$\overline{\text{CTS}}$	CB	Clear to send	OUT
6	$\overline{\text{DSR}}$	CC	Data set ready	OUT
7	GND	AB	Signal ground (common return)	
8	$\overline{\text{CD}}$	CF	Received line signal detector	OUT
9			(Reserved for data set testing)	
10			(Reserved for data set testing)	

11			Unsigned	
12		SCF	Secondary recd. line sig. detector	OUT
13		SCB	Secondary clear to send	OUT
14		SBA	Secondary transmitted data	IN
15		DB	Transmission signal element timing (DCE source)	OUT
16		SBB	Secondary received data	OUT
17		DD	Receiver signal element timing (DCE source)	OUT
18			Unassigned	
19		SCA	Secondary request to send	IN
20	$\overline{\text{DTR}}$	CD	Data terminal ready	IN
21		CG	Signal quality detector	OUT
22		CE	Ring indicator	OUT
23		CH/CI	Data signal rate selector (DTE/DCE source)	IN/OUT
24		DA	Transmit signal element timing (DTE source)	IN
25			Unassigned	

- The voltage level +3V to +15V is defined as logic0; from -3V to -15V is defined as logic1.
- The timing and control signals are compatible with TTL level(Transistor-Transistor Logic).
- Because of the incompatibility of the data lines with the TTL level, voltage translators called LINE DRIVERS and LINE RECEIVERS are required to interface TTL level with RS-232 signals.
- The line driver, MC1488 converts logic 1 into approximately 9V.
- These levels at the receiving end are again converted by the line receiver MC1489 into TTL compatible level.



Line drivers and receivers