




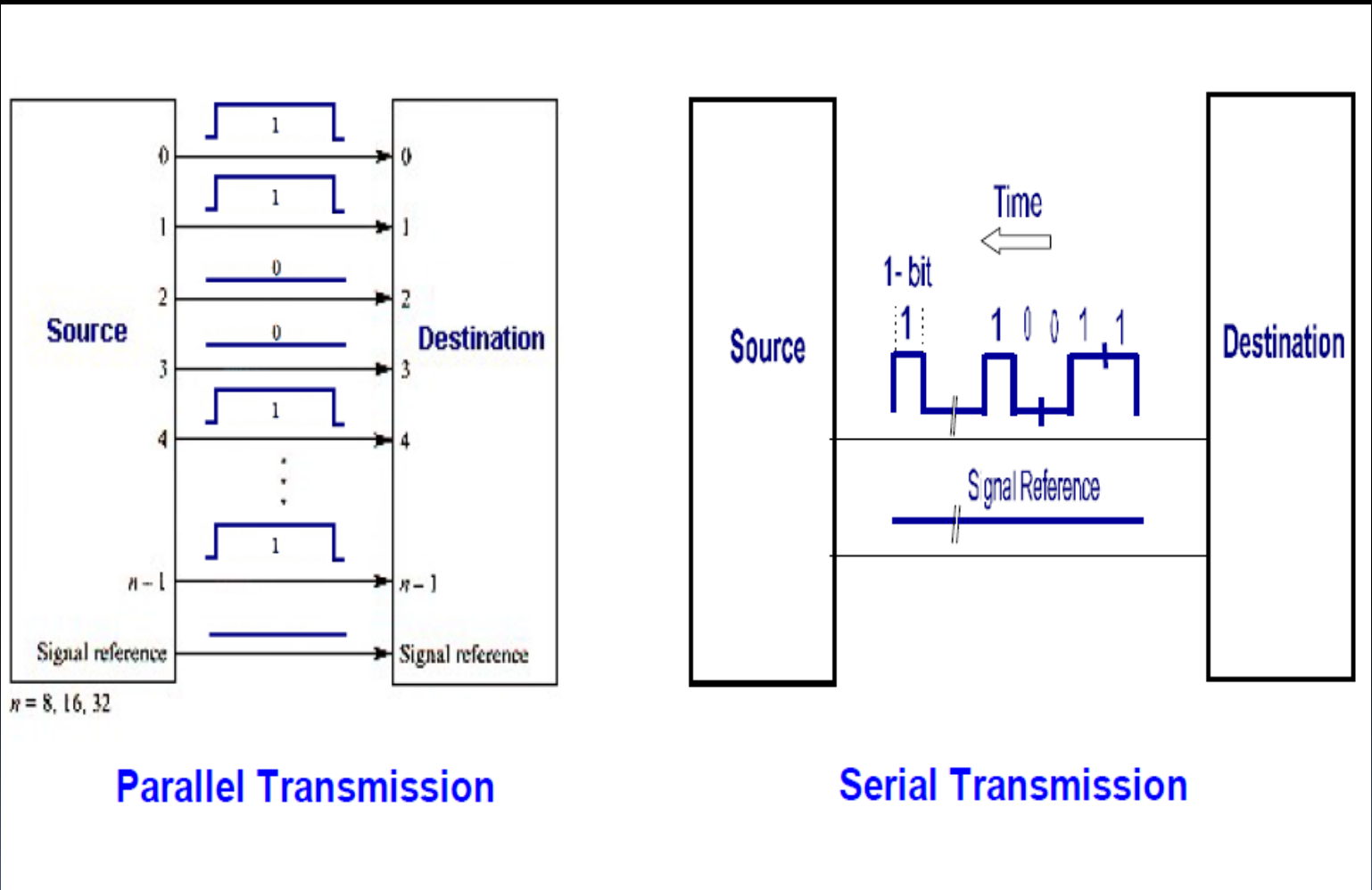
DATA COMMUNICATION



Data Communications

- Data communications refers to the ability of one computer to exchange data with another computer or a peripheral
 - Standard data communication interfaces and standards are needed
 - Centronic's parallel printer interface
 - RS-232 defines a serial communications standard
 - 8251 USART (Universal Synchronous/Asynchronous Receiver/Transmitter) is the key component for converting parallel data to serial form and vice versa
 - Two types of serial data communications are widely used
 - Asynchronous communications
 - Synchronous communications
- 

Parallel/Serial Transmissions



Communication Modes

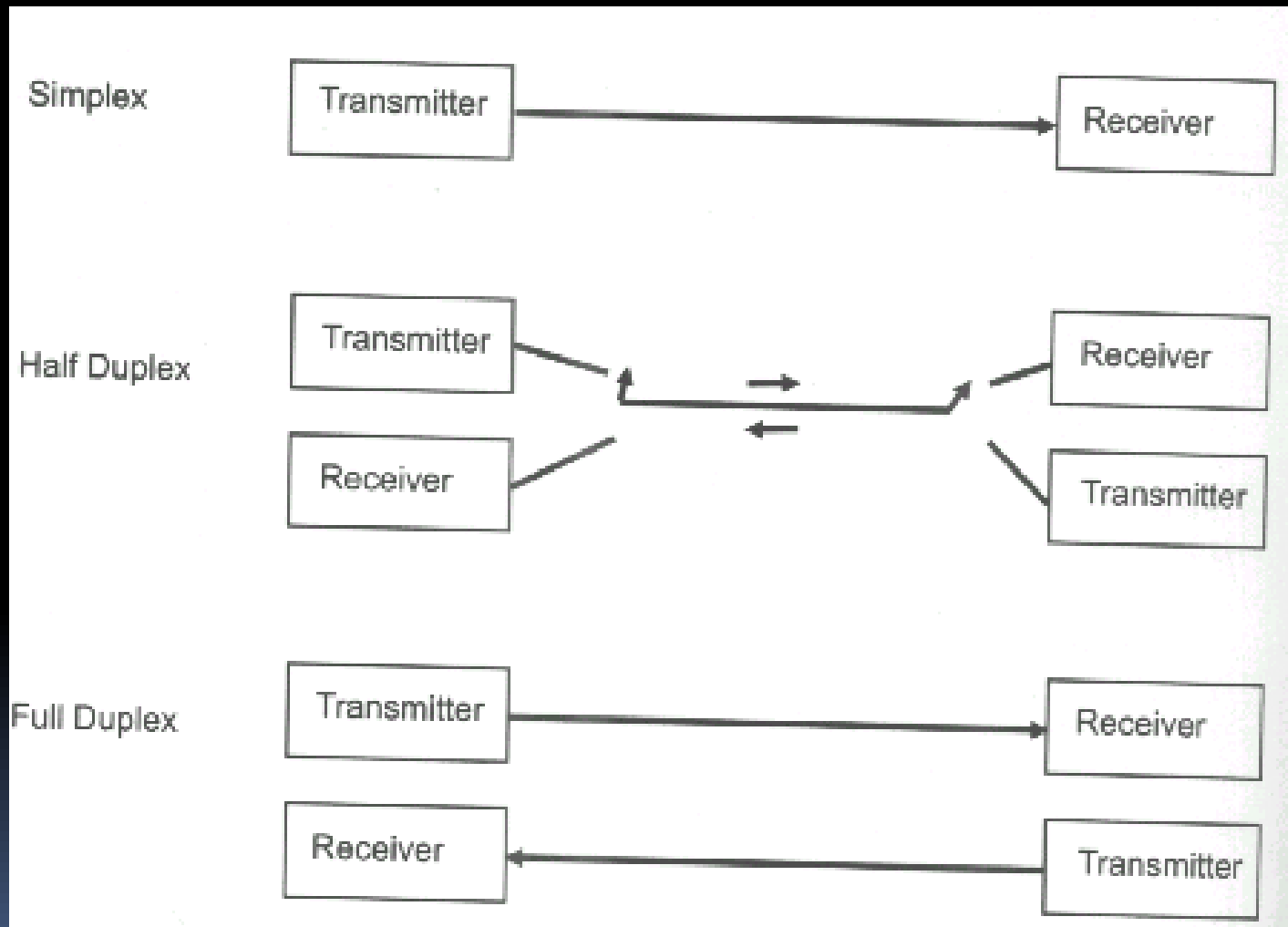
- When data is transmitted between two piece of equipment, 3 modes of communication are used
 - Simplex

Data is transmitted in one direction only
 - Half Duplex

This is used when to devices wants information alternatively, but one after another
 - Full Duplex

This is used when data is to be exchanged between two devices in both directions simultaneously

Communication Modes

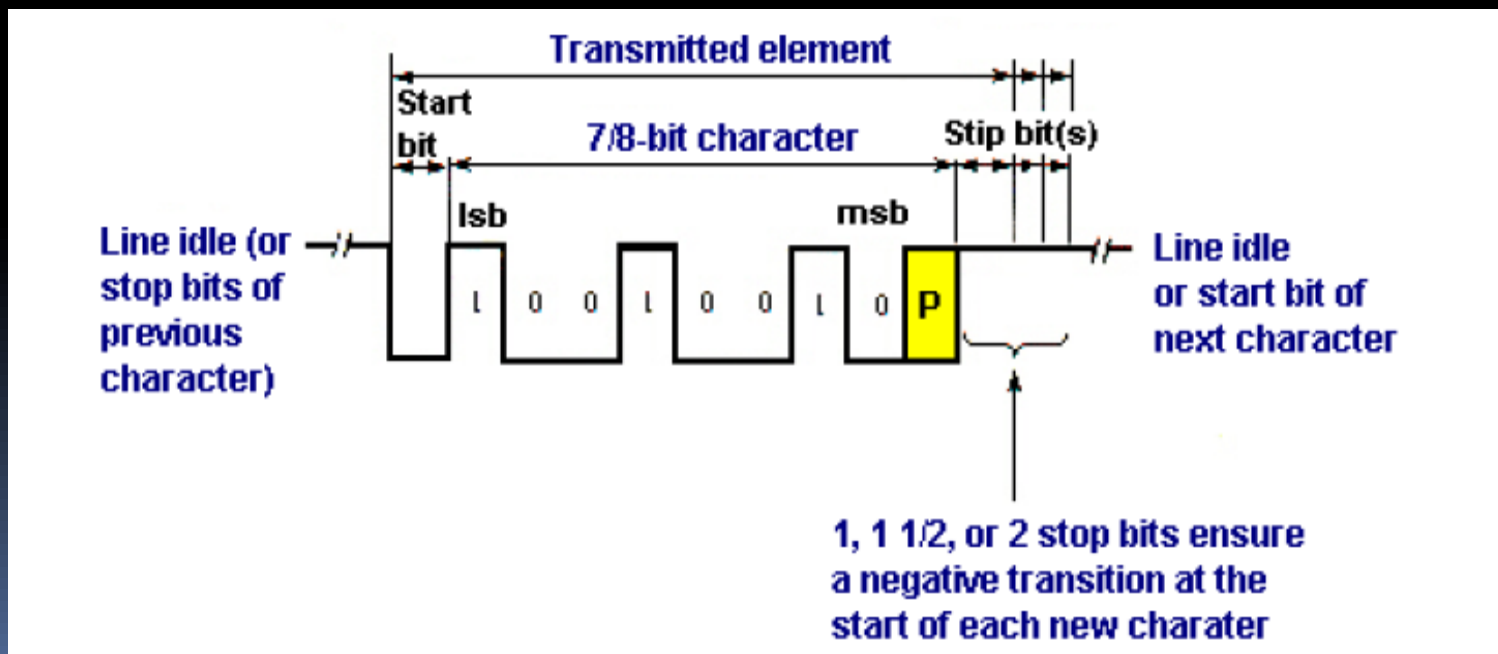


Transmission Modes

- For Receiving device to interpret bit pattern correctly, it must able to determine the following
 - Bit Synchronization
 - Start of each bit cell period
 - Character Synchronization
 - Start and end of each character or byte
 - Frame Synchronization
 - Start and end of each complete message block(frame)
- Types Synchronization
 - Asynchronous Transmission
 - Synchronous Transmission

Asynchronous Transmission

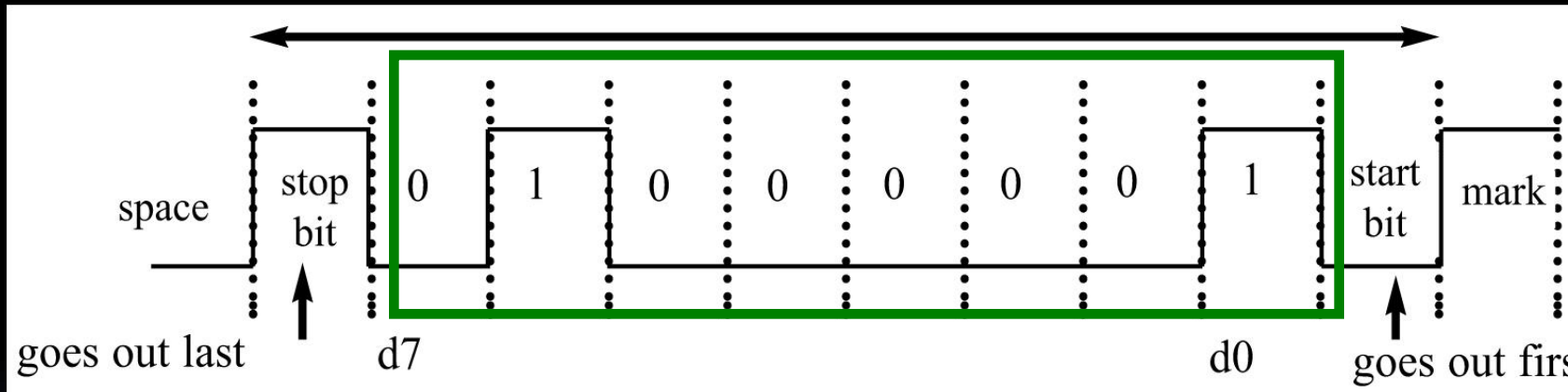
- In Asynchronous Transmission receiver clock runs(RxC) in unsynchronized with respect to the incoming signal(RxD)
- Additional start and stop bits are added in character(byte) data
- State of signal on transmission line between characters is idle



Asynchronous communications

- In asynchronous communications, the **data**, such as ASCII characters, are packed between a start bit and a stop bit, a process called *framing*.

ASCII character "A", binary **0100 0001**, framed between the **start** bit and 2 **stop** bits.

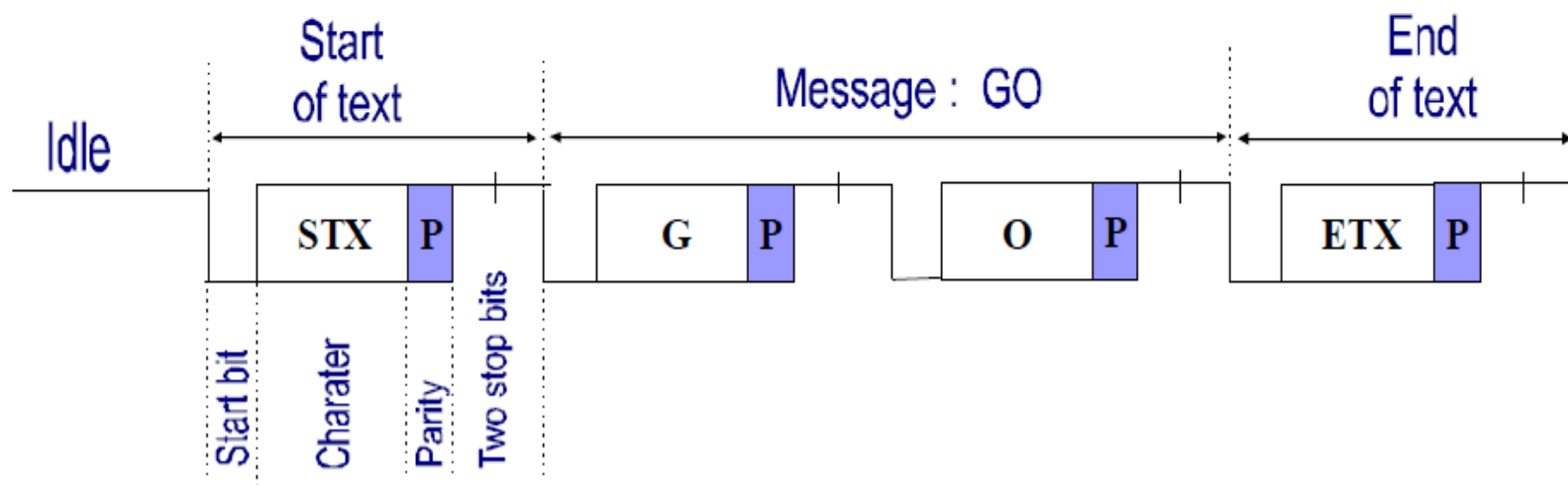


- The **start** bit is always one bit and always a 0. (*low*)
- The **stop** bit can be one or two bits, and is 1 (*high*).

Asynchronous Transmission

Example:

Construct the transmitted frame using *asynchronous transmission mode* which contains the following data: **GO**. Assume that the number of stop bits is 2 and parity bit is used.

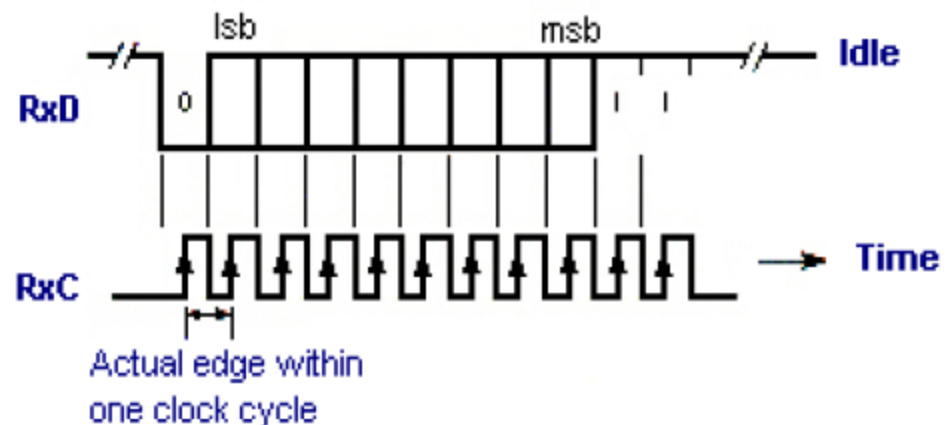
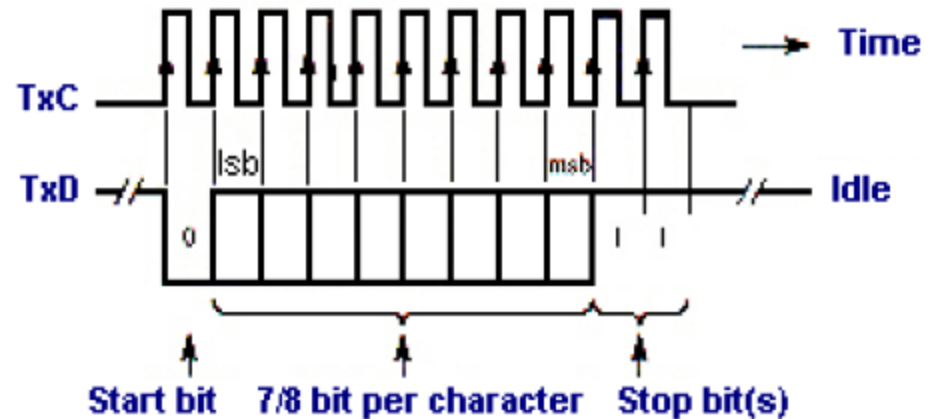


Asynchronous Transmission

- **Baud (signaling rate)** is used to define number of line signal transition per second.
- **Bit rate** is the number of bits transmitted per second.
- **Special case:** (Baud = bit rate) when a signal has only two levels: 0 or one.
- **Example:** A signaling rate of 300 baud with 4 bits per signaling element would yield a bit rate of 1200 bps.

Asynchronous Transmission

Principle of Operation and Timing:



Asynchronous Transmission

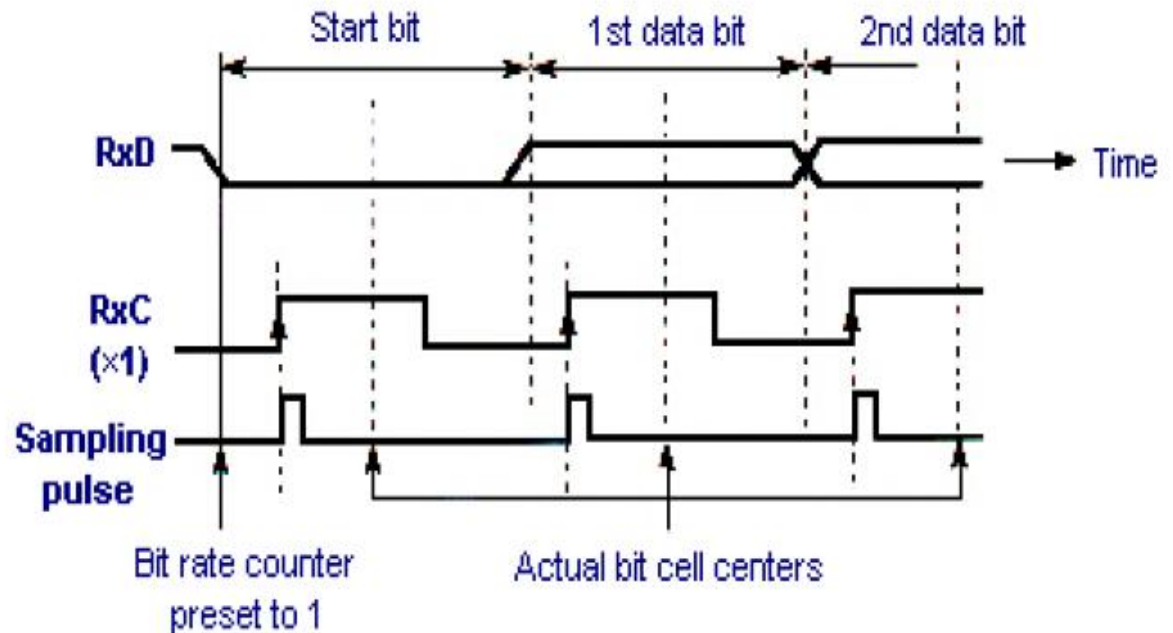
Bit Synchronization in Asynchronous Transmission:

- The local receiver clock is N times the transmitted bit rate ($N=16$ is common).
- The first 1→0 transition is associated with the start bit.
- Each bit is sampled at the center to avoid delay distortion problem.
- After the first transition is detected, the signal is sampled after $N/2$ clock cycles and then subsequently after N clock cycles for each bit in the character.

Asynchronous Transmission

Bit Synchronization in Asynchronous Transmission:

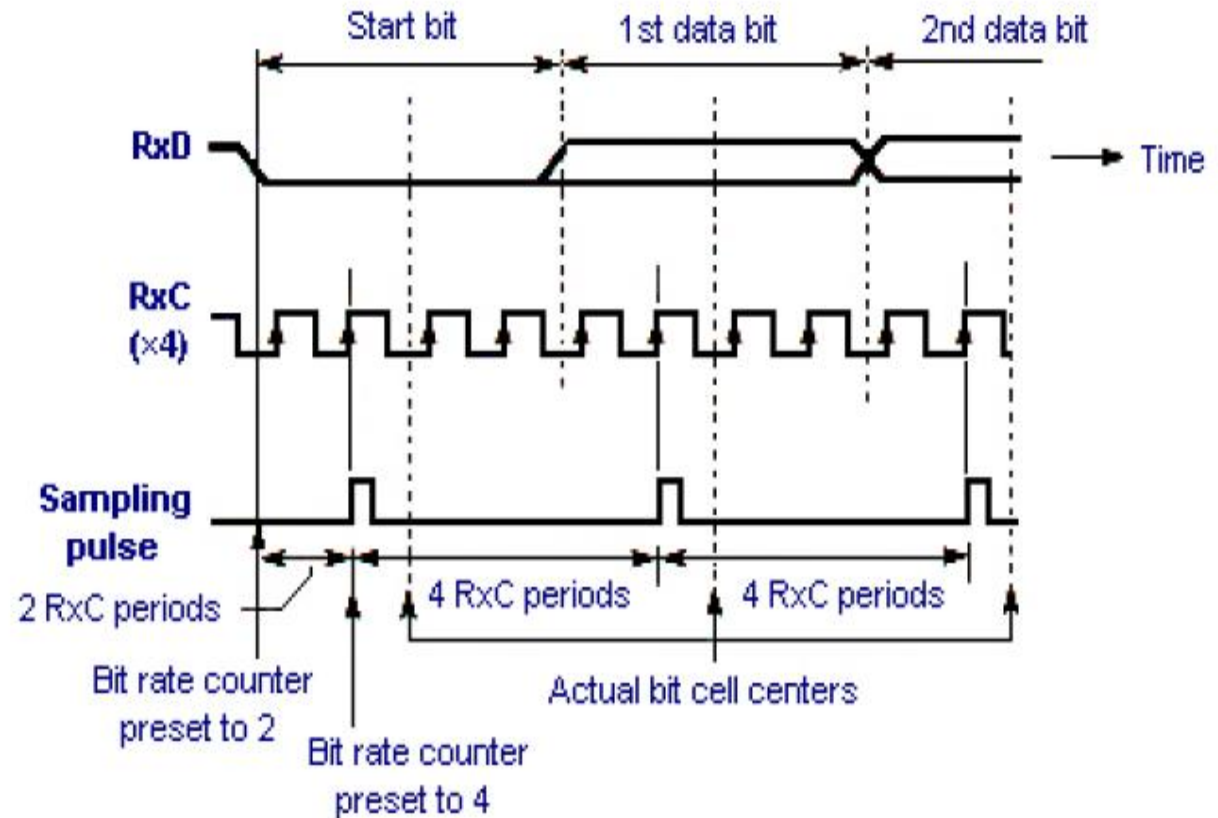
$N = 1$



Asynchronous Transmission

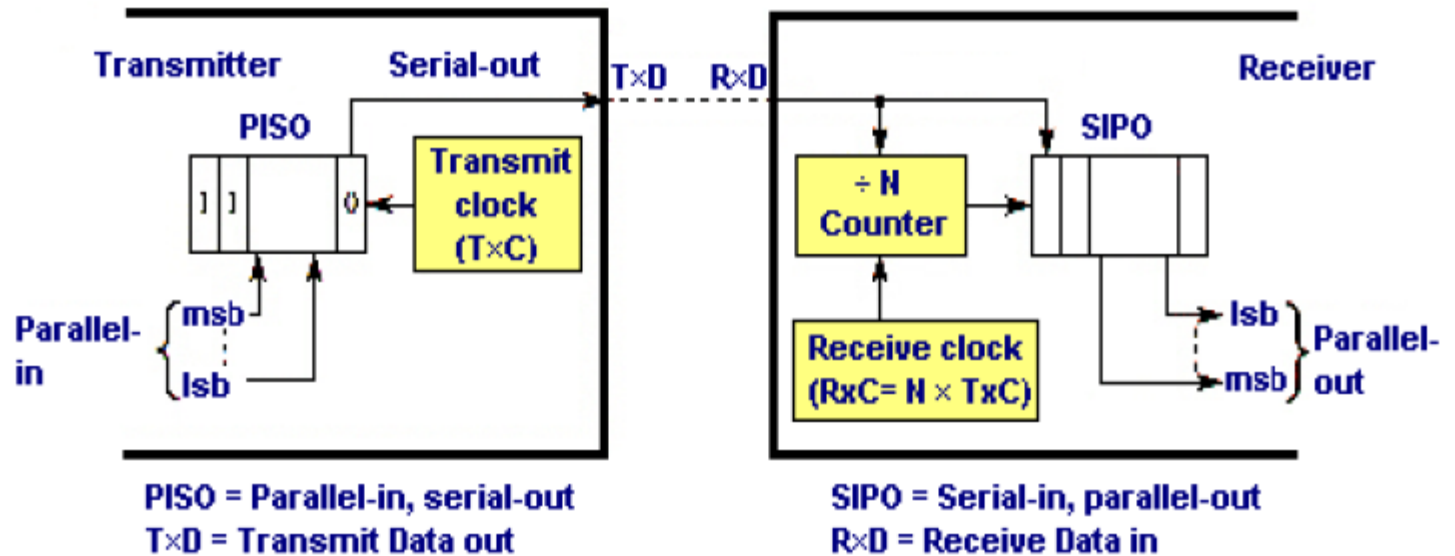
Bit Synchronization in Asynchronous Transmission:

$N = 4$



Asynchronous Transmission

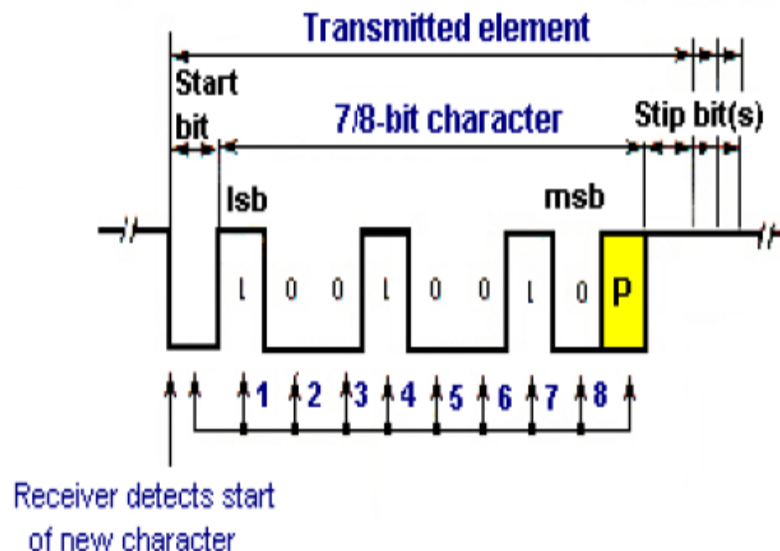
Principle of operation and Timing:



Asynchronous Transmission

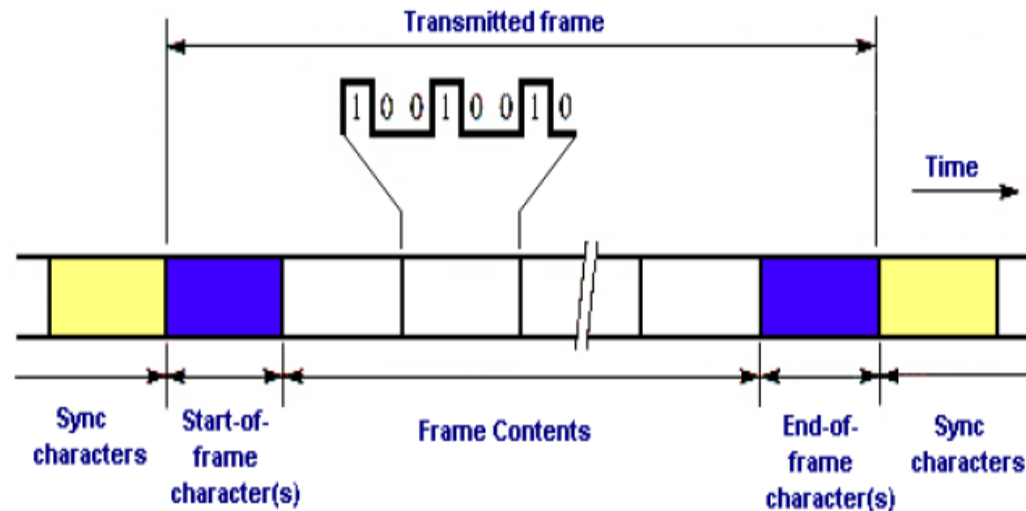
Character Synchronization in Asynchronous Transmission:

- After the start bit is detected, the receiver achieves character synchronization simply by counting the programmed number of bits.



Synchronous Transmission

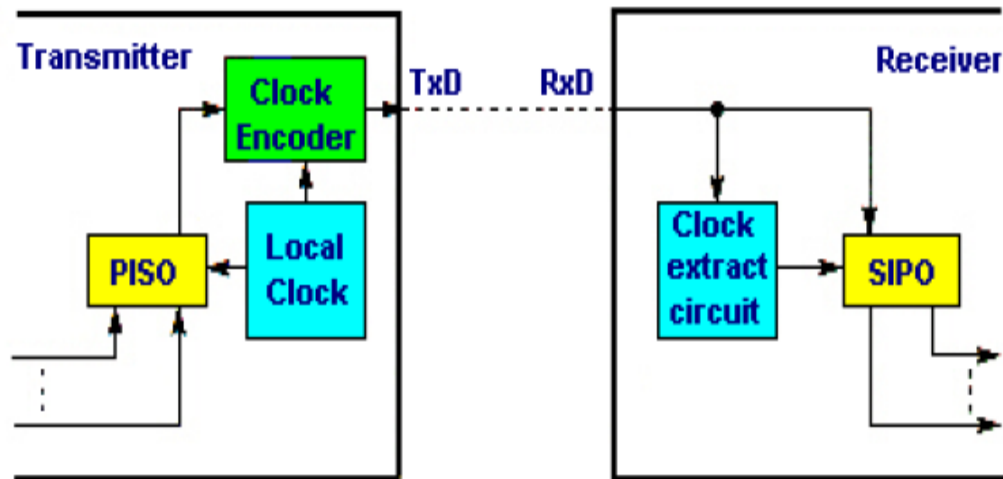
- The complete block or frame of data is transmitted as a **contiguous stream** with no delay between each 8-bit element.



Synchronous Transmission

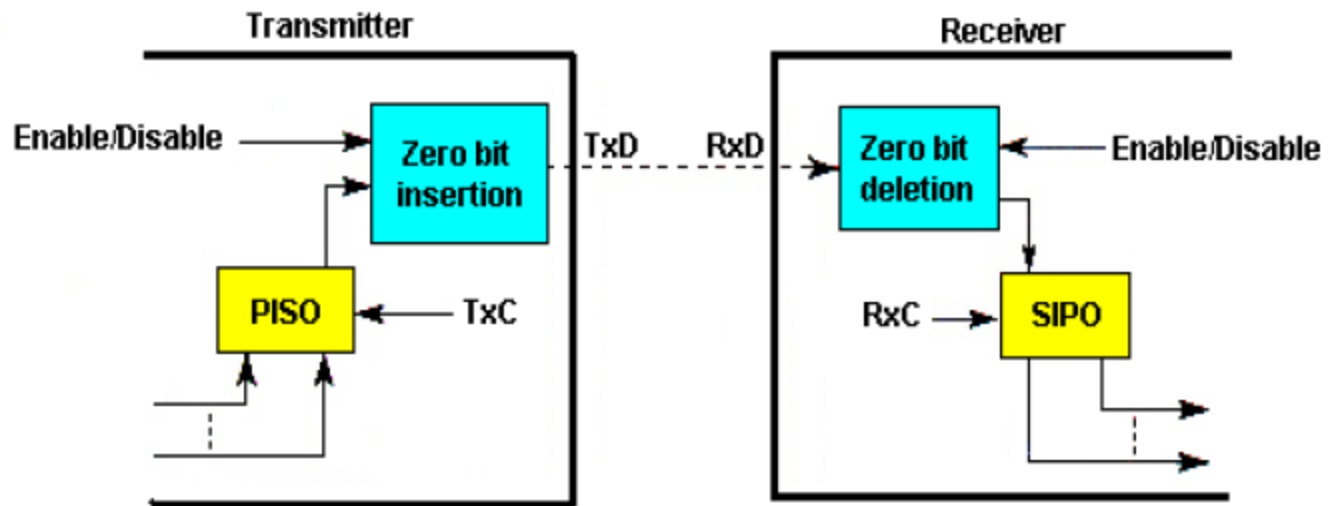
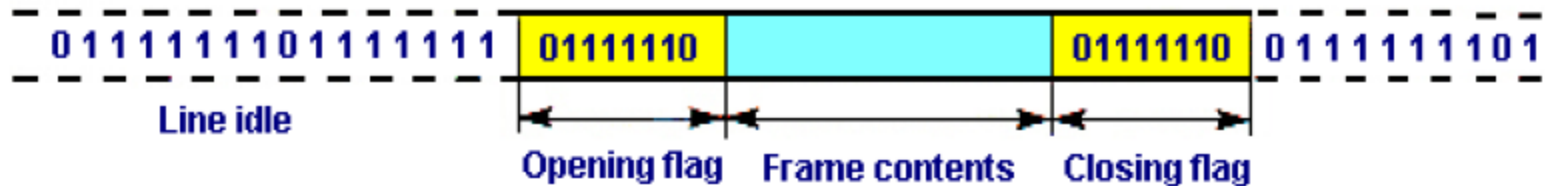
Bit Synchronization using Synchronous Transmission:

- With synchronous transmission, the receiver clock ($R \times C$) operates in synchronism with the received data signal ($R \times D$).
- **Clock Encoding and Extraction:** The clock information is embedded into the transmitted signal and subsequently extracted by the receiver.



Synchronous Transmission

Bit-Oriented Synchronous Transmission:



RS232

- For compatibility in data communication equipment, an interfacing standard called RS232 was set by the Electronics Industries Association (EIA) in 1960.
- Today's most widely used serial I/O interface standard

Pins and their labels for the RS232 cable, which is commonly referred to as the DB-9 connector.

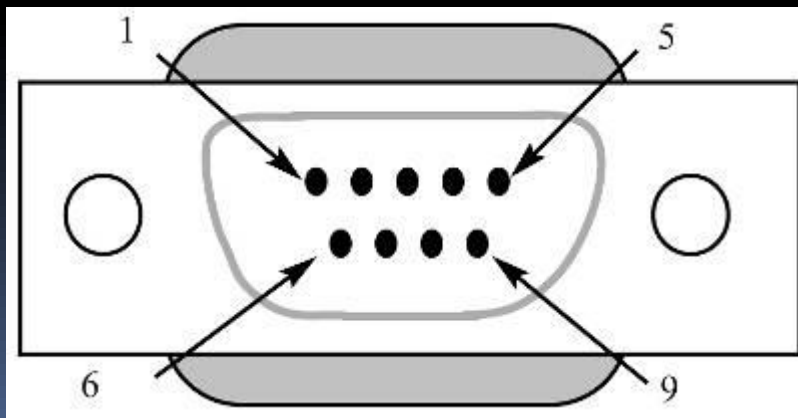
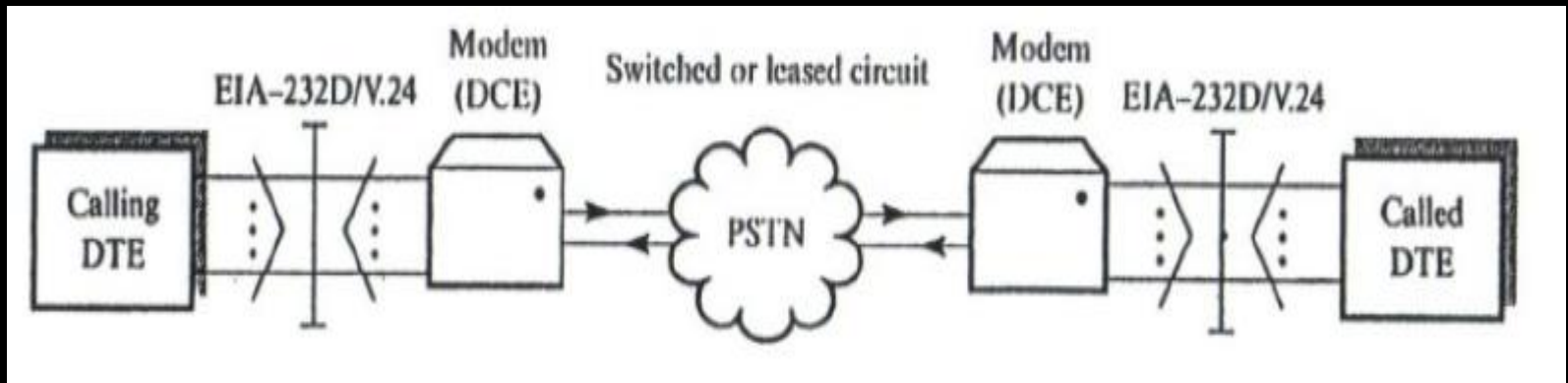


Fig DB9 9-Pin Connector

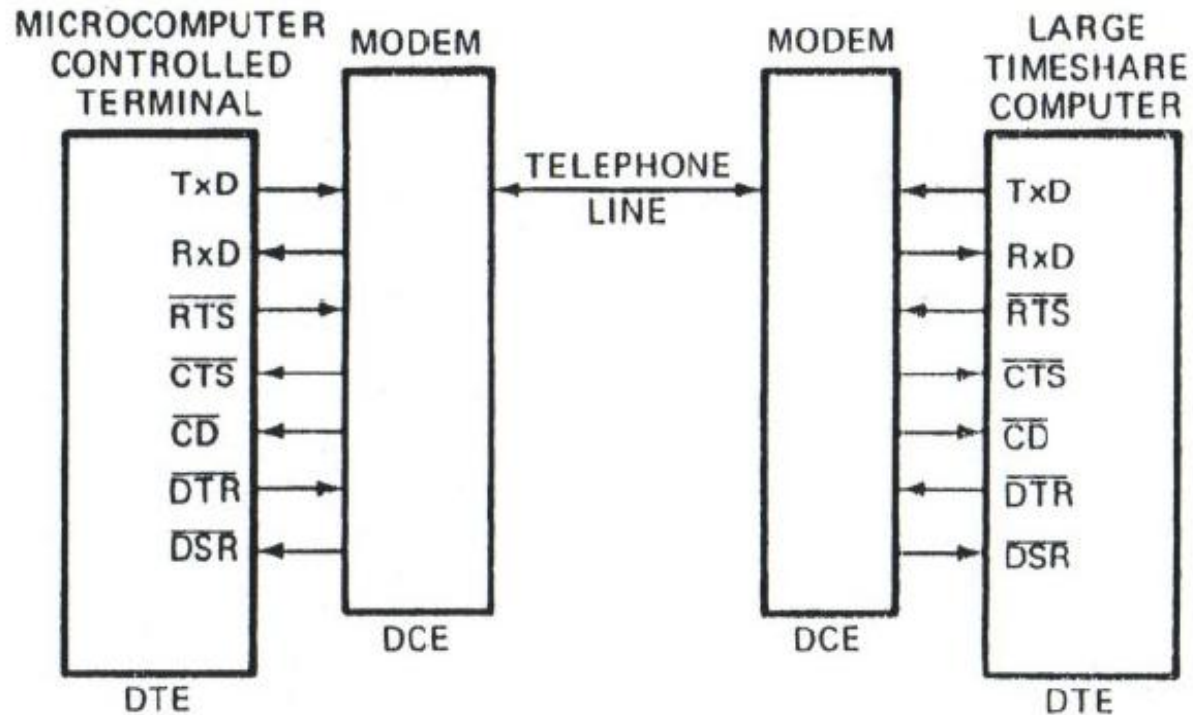
Table 17-1: IBM PC DB-9 Signals

Pin	Description
1	Data carrier detect ($\overline{\text{DCD}}$)
2	Received data (RxD)
3	Transmitted data (TxD)
4	Data terminal ready (DTR)
5	Signal ground (GND)
6	Data set ready ($\overline{\text{DSR}}$)
7	Request to send ($\overline{\text{RTS}}$)
8	Clear to send ($\overline{\text{CTS}}$)
9	Ring indicator (RI)

Digital Data Transmission using MODEM



Digital Data Transmission using MODEM



DTE = DATA TERMINAL EQUIPMENT
DCE = DATA COMMUNICATION EQUIPMENT

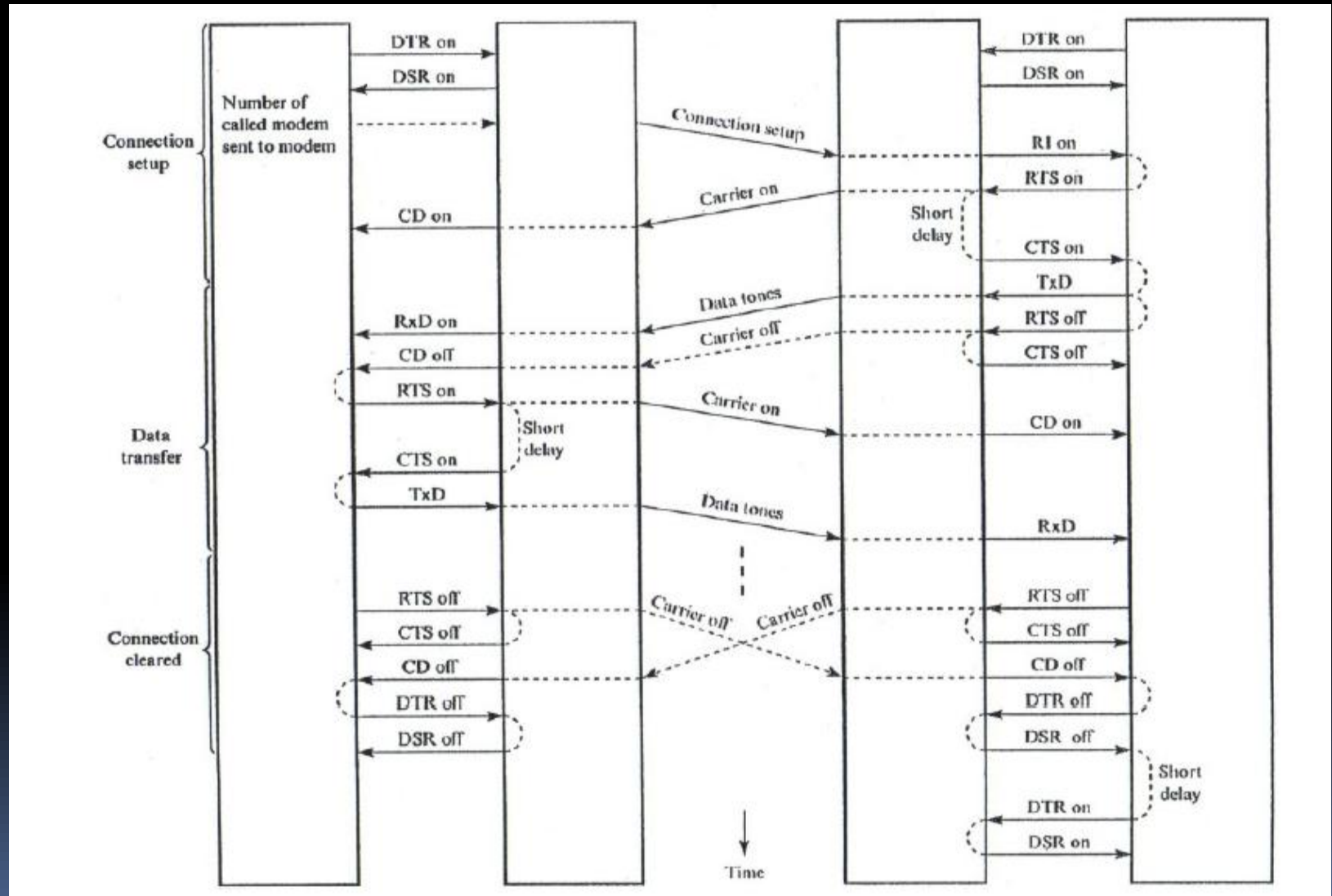
Sequence of Modem Control Signals

- After the PC power is turned on and the PC runs any self-checks, it asserts the **data-terminal-ready (DTR/)** signal to tell the modem it is ready.
- When the modem is powered up and ready to transmit and receive data, the modem will assert the **data-set-ready (DSR/)**.
- The calling PC sends the telephone number of the modem associated with the called computer. The modem then dials up the called computer.
- When the called modem receives ring tones, it will set the **ring indicator (RI)** line to **on** and the called computer responds by setting the **request-to-send (RTS/)** line **on**.
- In response, the called modem sends a carrier signal - the data tone for a binary 1- to the calling modem to indicate that the call has been accepted by the called computer.

Sequence of Modem Control Signals

- After a short delay to allow the calling modem to prepare to receive data, the called modem sets the **clear-to-send (CTS/)** line **on** to inform the called computer that it can start sending data.
- On detecting the carrier signal, the calling modem sets the **carrier-detect (CD/)** line **on**.
- The called computer starts by sending a short message over the set-up connection.
- When this message has been sent, it prepares itself to receive the response from the calling terminal by setting the **RTS/** line **off**, and on detecting this, the called modem stops sending the carrier signal and sets the **CTS/** line **off**.
- At the calling side, the removal of carrier signal is detected by the calling modem and, in response, it set the **CD/** line **off**.
- The calling PC sets **RTS/** line **on** in order to send the response message and, on receipt of the **CTS/** signal from the modem, starts to send the message.

Digital Data Transmission using MODEM





8251 PROGRAMMABLE COMMUNICATION INTERFACE

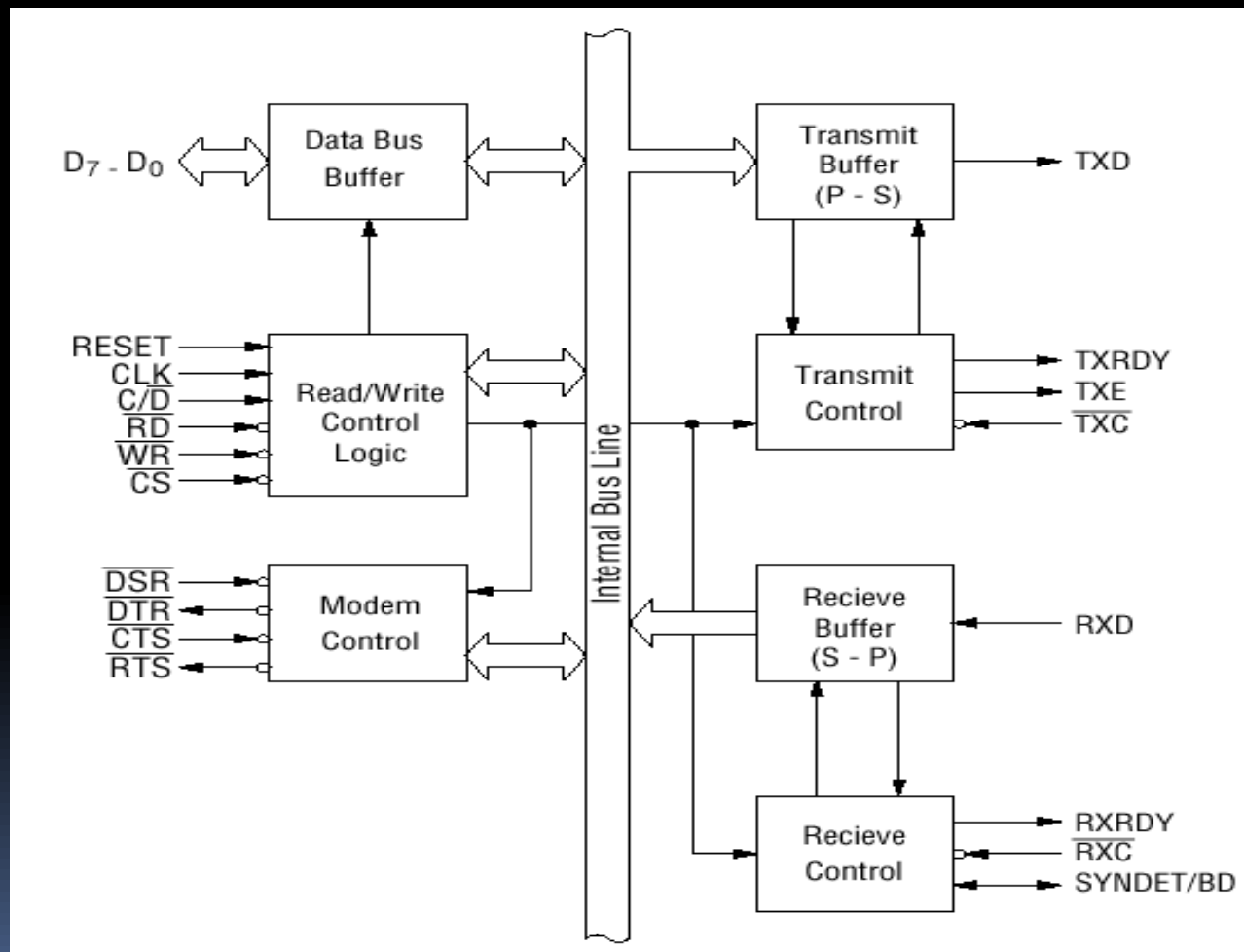




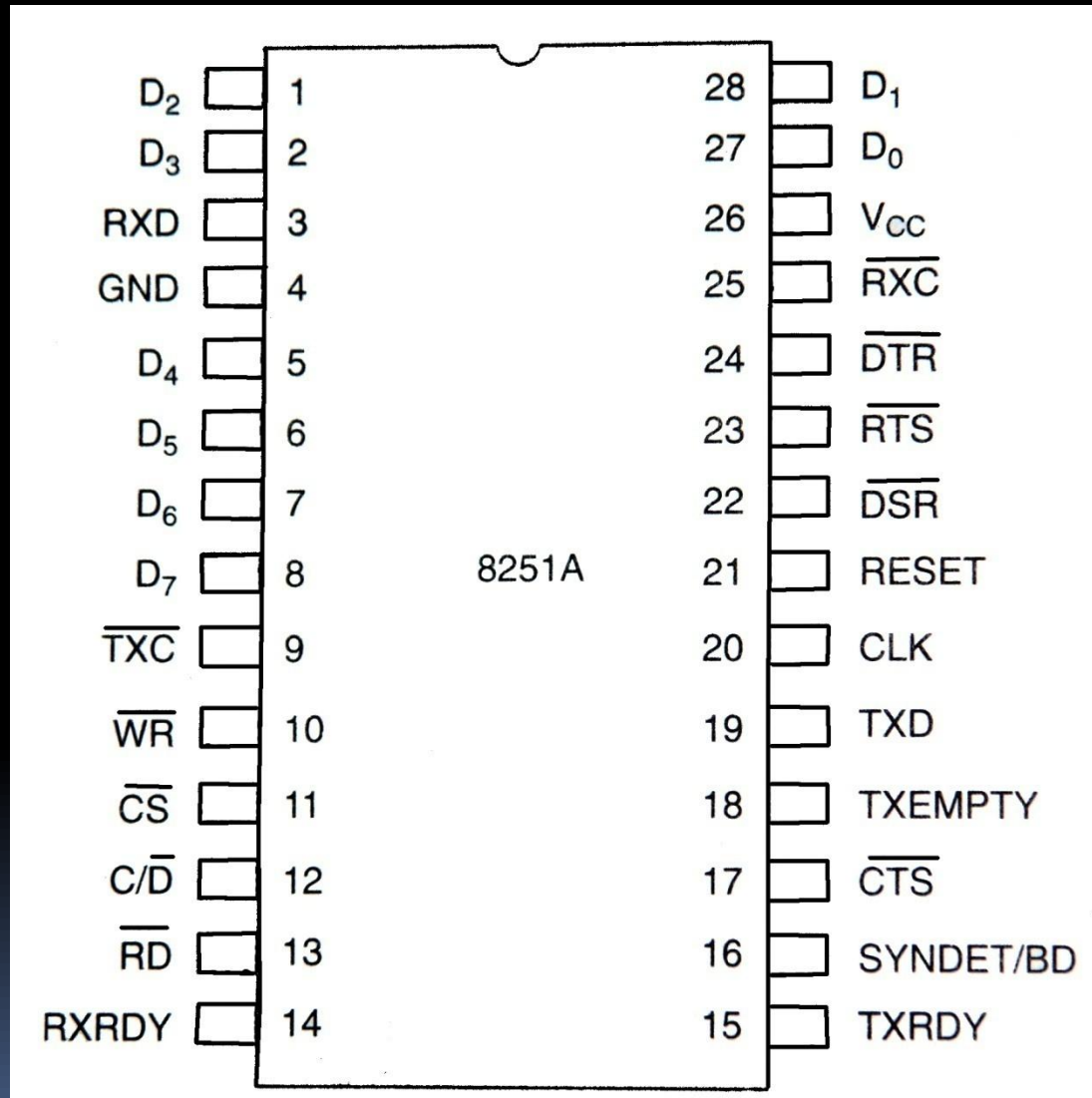
Introduction

- 8251 is a USART (Universal Synchronous Asynchronous Receiver Transmitter) for serial data communication.
- Programmable peripheral designed for synchronous /asynchronous serial data communication, packaged in a 28-pin DIP.
- Receives parallel data from the CPU & transmits serial data after conversion.
- Also receives serial data from the outside & transmits parallel data to the CPU after conversion.

Block diagram of the 8251 USART



Pin diagram



Signals of 8251

- $\overline{\text{CS}}$ – Chip Select : When signal goes low, the 8251A is selected by the MPU for communication.
- $\text{C}/\overline{\text{D}}$ – Control/Data : When signal is high, the control or status register is addressed; when it is low, data buffer is addressed. (Control register & status register are differentiated by WR and RD signals)
- $\overline{\text{WR}}$: When signal is low, the MPU either writes in the control register or sends output to the data buffer.
- $\overline{\text{RD}}$: When signal goes low, the MPU either reads a status from the status register or accepts data from data buffer.
- RESET : A high on this signal reset 8252A & forces it into the idle mode.
- CLK : Clock input, usually connected to the system clock for communication with the microprocessor.

Signals of 8251

- The 8251A is **doubled-buffered**. This means that one character can be loaded into a **data-out buffer register** while another character is being shifted out of the actual **transmit shift register**.
- The **TxRDY** output of the 8251A will go high when:
 - The **data-out buffer register** is Empty for another character from the CPU.
 - The **CTS/** input has been asserted low.
 - The **transmit-enable (TxEN)** bit of the 8251A's command word is set.
- The **TxEMPTY** output of the 8251A will go high when both the **data-out buffer register** and the **transmit shift register** are empty.
- The **RxRDY** output of the 8251A will go high when:
 - The **data-in buffer register** is full and is ready to be read by the CPU.
 - The **receive-enable (RxE)** bit of the 8251A's command word is set.
- If the CPU does not read a character from the **data-in buffer register** before another character is shifted in, the first character will be overwritten and lost.

Signals of 8251

- The **sync-detect/break-detect (SYNDET/BD)** pin has two uses:
 1. When the device is operating in **asynchronous mode**, the pin will go high if the serial data input line (**RxD**) stays low for more than 2 character times (i.e., the **RxD** remains low through two consecutive stop bit sequences including the start bits, data bits, and parity bits). This signal then indicates an intentional break in data transmission. It is reset only upon chip **RESET** or **RxD** returning to a "one" state.
 2. When the device is operating in **synchronous mode**, the **SYNDET** pin can be programmed as either input or output. When used as an output, then **SYNDET** pin will go high to indicate that the 8251A has located the **SYN** character in the receiver mode. If the 8251 A is programmed to use double **SYN** characters, then the **SYNDET** will go high in the middle of the last bit of the second **SYN** character. **SYNDET** is automatically reset upon a status read operation. If the search for **SYN** characters is conducted by an external device, then **SYNDET** can be used to input a signal, indicating that a match has been found by the external device.



Sections of 8251

- Data Bus buffer
- Read/Write Control Logic
- Modem Control
- Transmitter
- Receiver



1. Data Bus Buffer

- D0-D7 : 8-bit data bus used to read or write status, command word or data from or to the 8251A

2. Read/Write Control logic

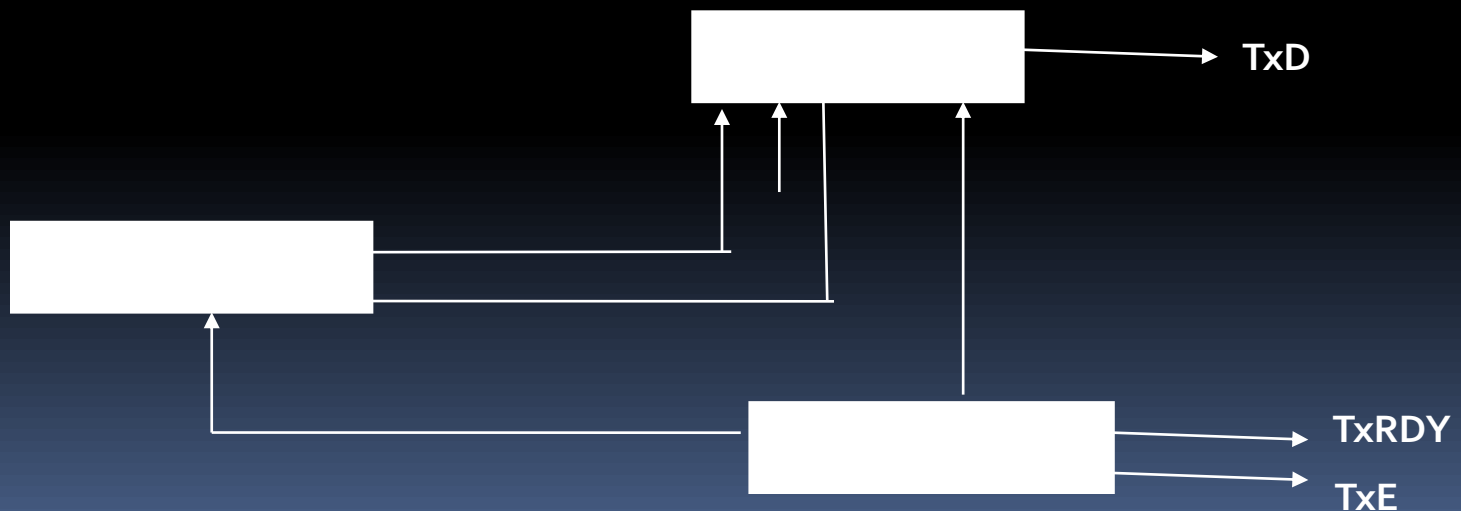
- Includes a control logic, six input signals & three buffer registers: Data register, control register & status register.
- Control logic : Interfaces the chip with MPU, determines the functions of the chip according to the control word in the control register & monitors the data flow.

3. Modem Control

- $\overline{\text{DSR}}$ - Data Set Ready : Checks if the Data Set is ready when communicating with a modem.
- $\overline{\text{DTR}}$ - Data Terminal Ready : Indicates that the device is ready to accept data when the 8251 is communicating with a modem.
- $\overline{\text{CTS}}$ - Clear to Send : If its low, the 8251A is enabled to transmit the serial data provided the enable bit in the command byte is set to '1'.
- RTS - Request to Send Data : Low signal indicates the modem that the receiver is ready to receive a data byte from the modem.

4. Transmitter section

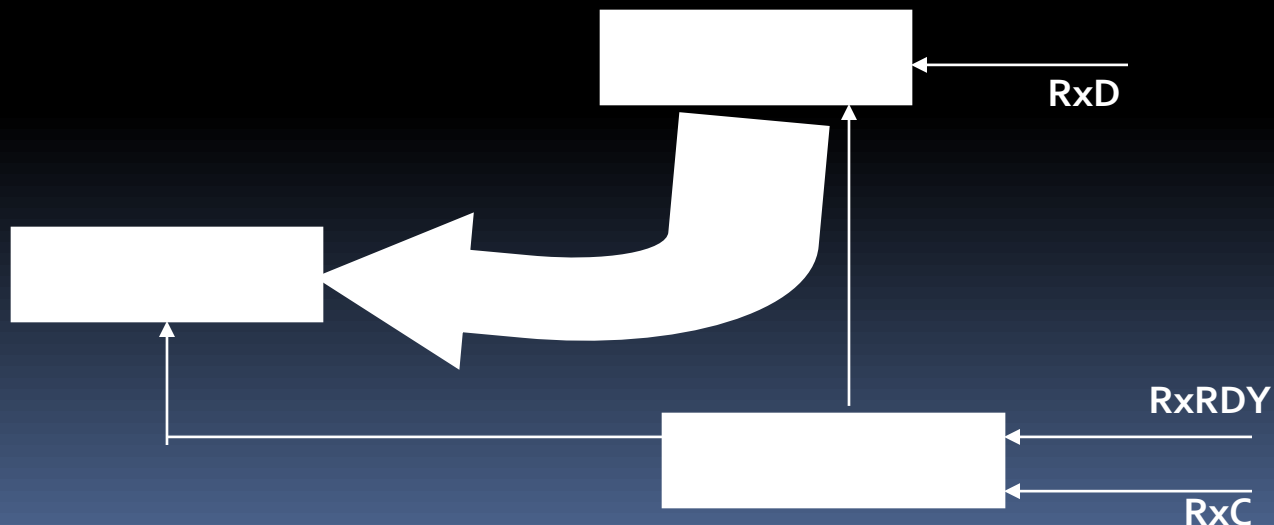
- Accepts parallel data from MPU & converts them into serial data.
- Has two registers:
 - Buffer register : To hold eight bits
 - Output register : To convert eight bits into a stream of serial bits.



- The MPU writes a byte in the buffer register.
- Whenever the output register is empty; the contents of buffer register are transferred to output register.
- Transmitter section consists of three output & one input signals
 - TxD - Transmitted Data Output : Output signal to transmit the data to peripherals
 - $\overline{\text{TxC}}$ - Transmitter Clock Input : Input signal, controls the rate of transmission.
 - TxRDY - Transmitter Ready : Output signal, indicates the buffer register is empty and the USART is ready to accept the next data byte.
 - TxE - Transmitter Empty : Output signal to indicate the output register is empty and the USART is ready to accept the next data byte.

5. Receiver Section


- Accepts serial data on the RxD pin and converts them to parallel data.
- Has two registers :
 - Receiver input register
 - Buffer register



- When RxD goes low, the control logic assumes it is a start bit, waits for half bit time, and samples the line again. If the line is still low, the input register accepts the following data, and loads it into buffer register at the rate determined by the receiver clock.
- RxRDY - Receiver Ready Output: Output signal, goes high when the USART has a character in the buffer register & is ready to transfer it to the MPU.
- RxD - Receive Data Input : Bits are received serially on this line & converted into a parallel byte in the receiver input register.
- RxC - Receiver Clock Input : Clock signal that controls the rate at which bits are received by the USART.



Control Register

- 16-bit register for a control word consist of two independent bytes namely mode word & command word.
 - Mode word : Specifies the general characteristics of operation such as baud, parity, number of bits etc.
 - Command word : Enables the data transmission and reception.
 - Register can be accessed as an output port when the Control/Data pin is high.
- 

Status register

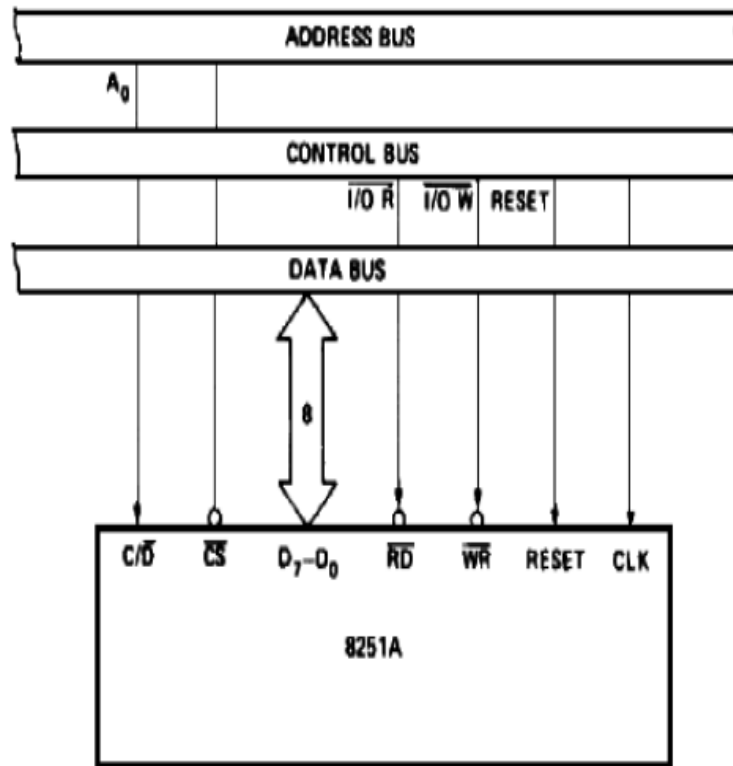
- Checks the ready status of the peripheral.
- Status word in the status register provides the information concerning register status and transmission errors.

Data register

- Used as an input and output port when the C/D is low

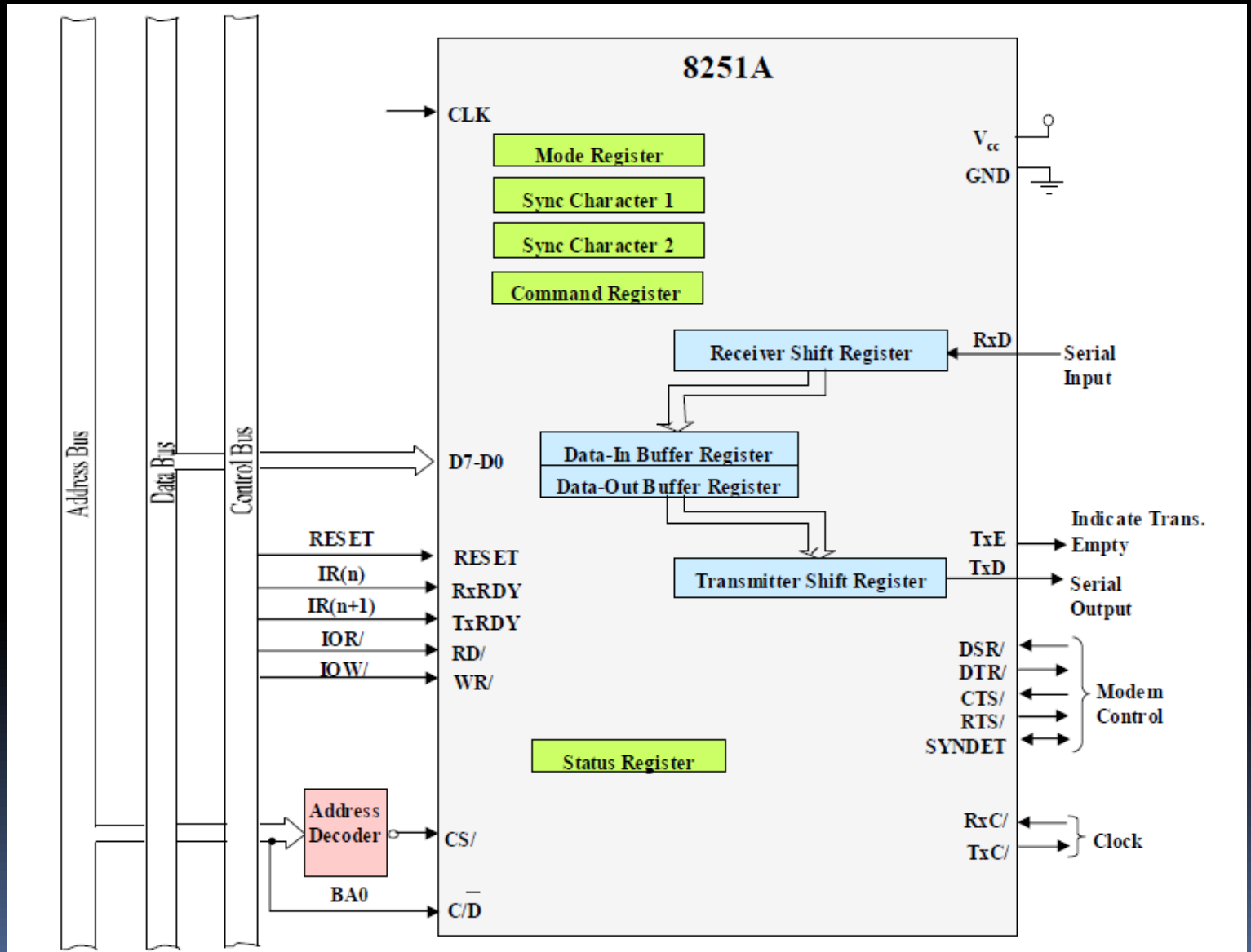
<u>CS</u>	<u>C/D</u>	<u>WR</u>	<u>RD</u>	<u>Operation</u>
0	0	1	0	MPU reads data from data buffer
0	0	0	1	MPU writes data from data buffer
0	1	0	1	MPU writes a word to control register
0	1	1	0	MPU reads a word from status register
1				Chip is not selected for any operation

Interfacing 8251 to 8088

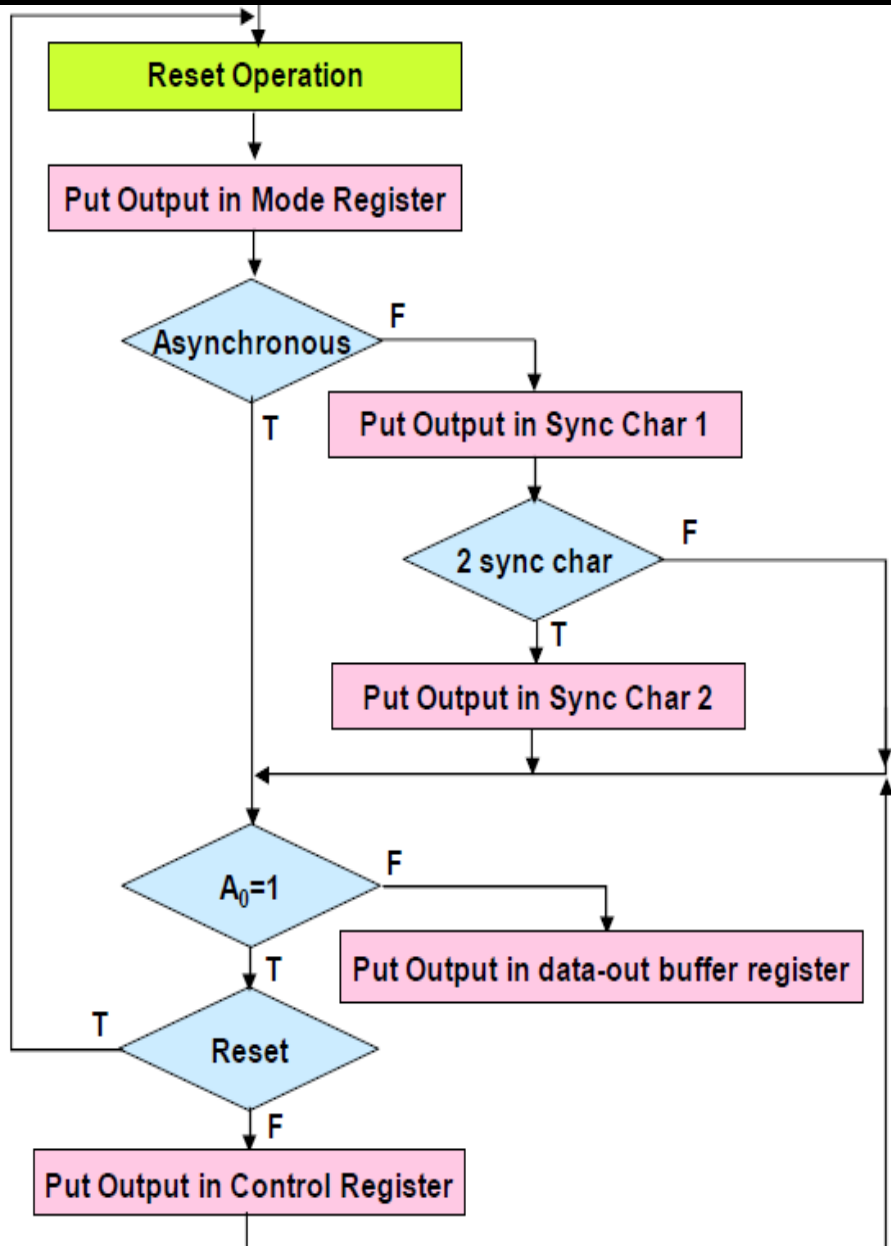
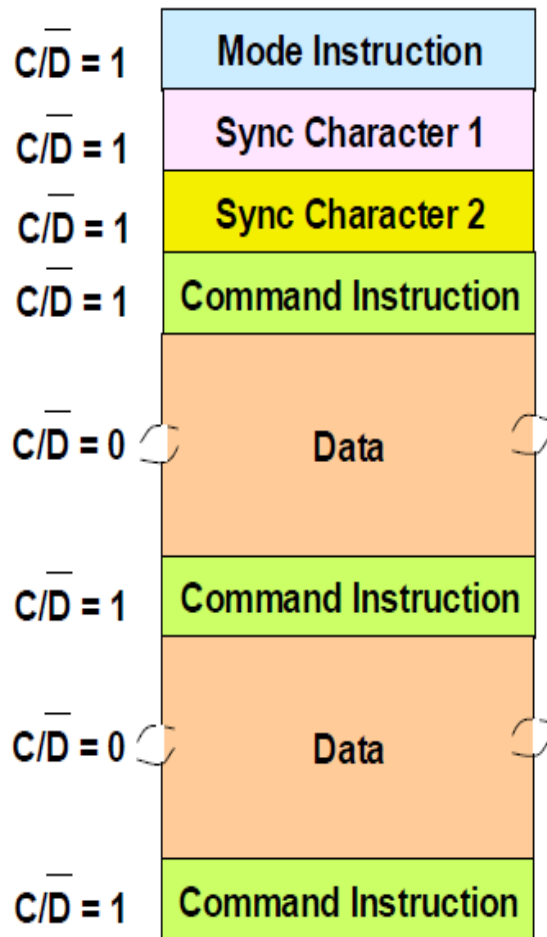


$\overline{C/D}$	RD/	WR/	CS/	Function
0	0	1	0	8251A DATA \rightarrow DATA BUS
0	1	0	0	DATA BUS \rightarrow 8251A DATA
1	0	1	0	STATUS \rightarrow DATA BUS
1	1	0	0	DATA BUS \rightarrow Control
X	1	1	0	DATA BUS \rightarrow 3-STATE
X	X	X	1	DATA BUS \rightarrow 3-STATE

8251 Communication Interface



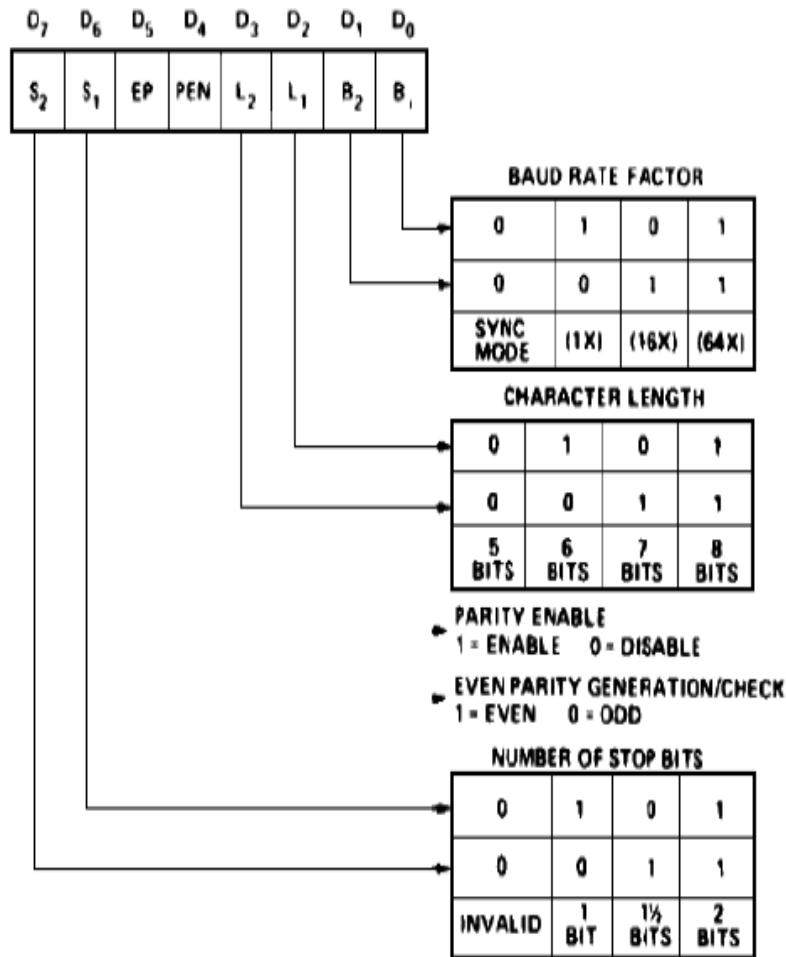
Initializing 8251



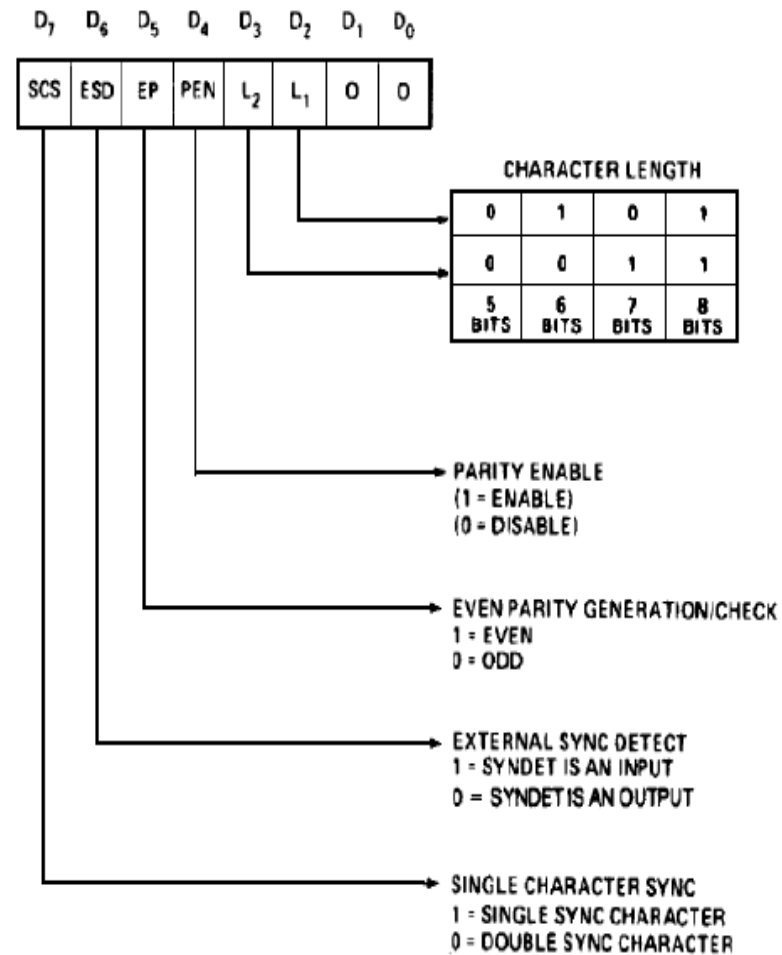
Initializing 8251

- To implement serial communication the MPU must inform the 8251 about the mode, baud, stop bits, parity etc. A set of control words must be loaded.
 - Mode Words
 - Specifies general characteristics of the operation.
 - Command Words
 - Enables the data transmission and/or reception
 - Status Word provides the information concerning register status and transmission errors.
- Any control word written into the control register after a mode word is interpreted as a command word; that means a command word can be changed anytime, however 8251 should be reset prior to writing a Mode word.
- 8251 can be reset internally by using the Internal Reset Bit D6.

8251 Mode Word

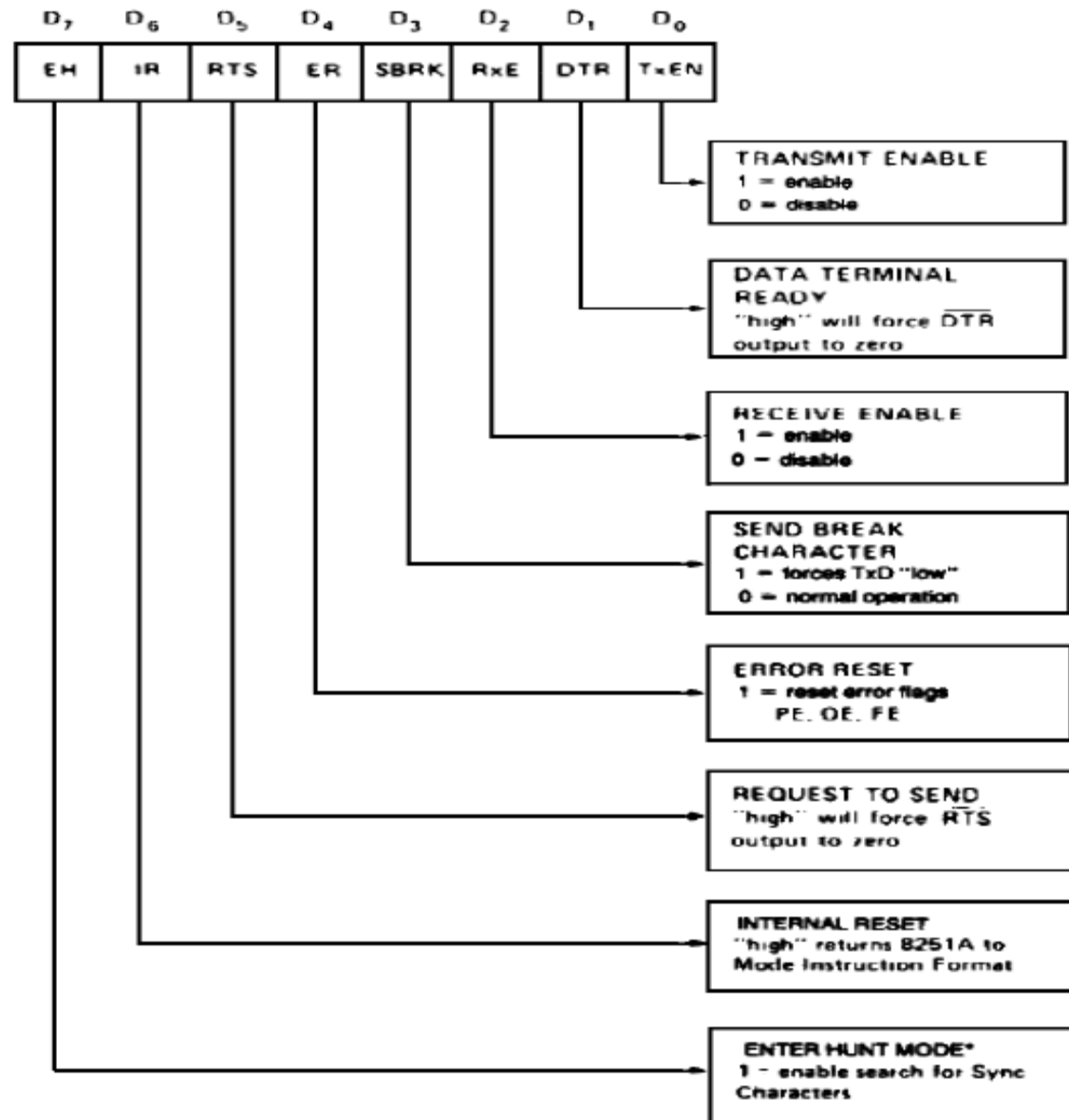


Asynchronous



Synchronous

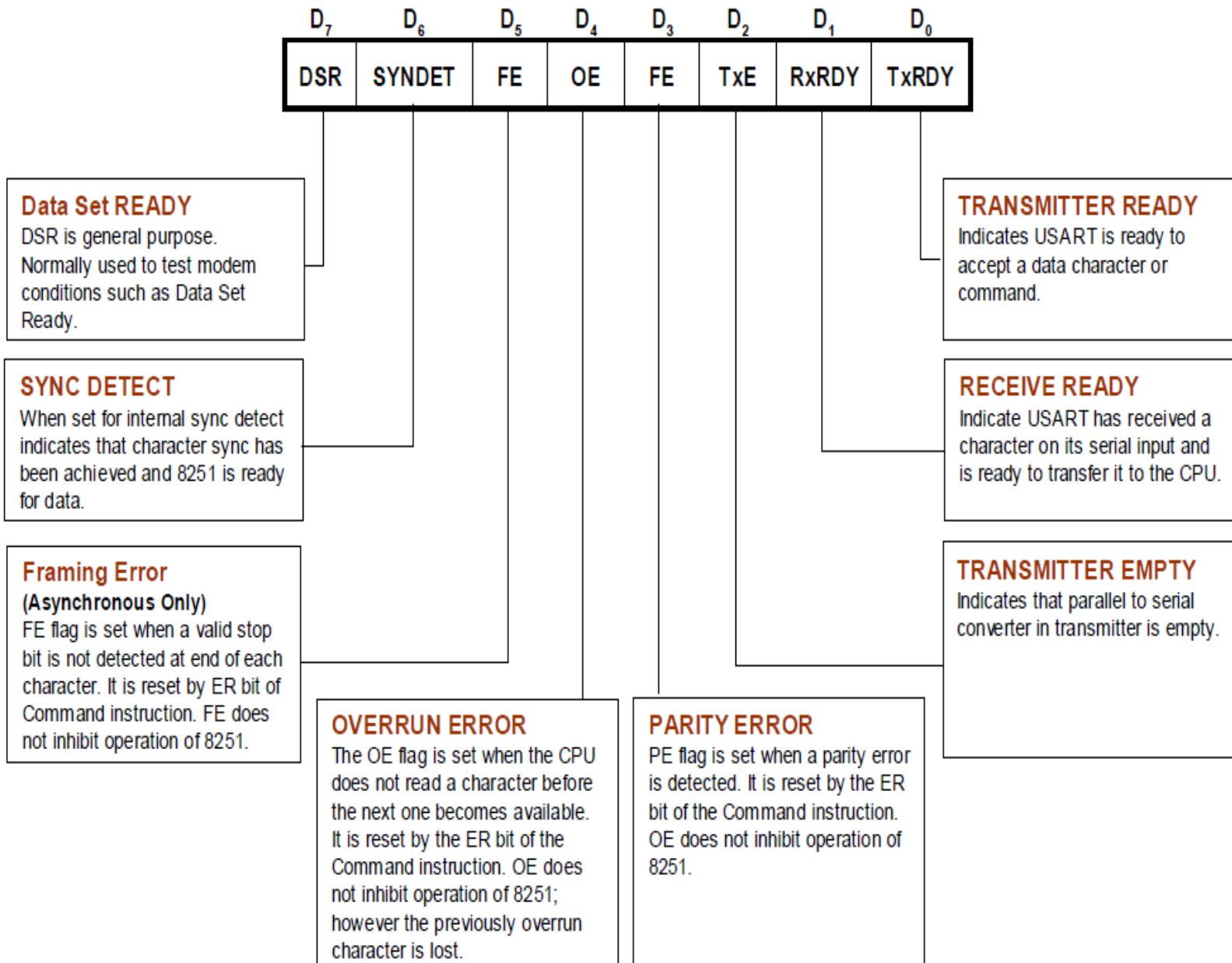
8251 Command Word



8251 Command Word

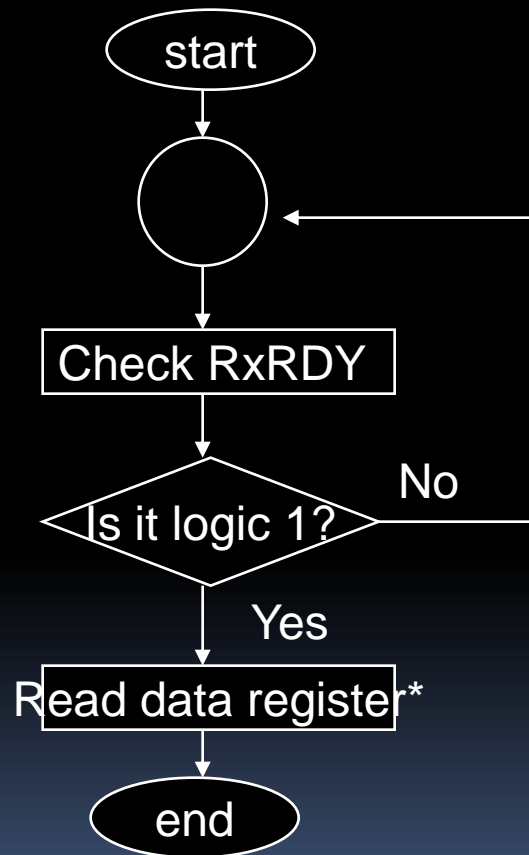
- Initializing the **TxEN** bit to 1 will enable the transmitter section of the 8251A and the **TxRDY** output.
- Initializing **DTR/** bit to 1 will cause the DTR/ output of the 8251A to be asserted low. This signal is used to tell a modem that a PC or terminal is operational.
- Initializing **RxE** bit to 1 will enable the RxRDY output of the 8251A.
- Initializing **SBRK** bit to 1 will cause the 8251A to output characters of 0's including start bits, data bits, and parity bits (**break character**). A break character is used to indicate the end of block of transmitted data.
- Initializing **ER** bit to 1 will cause the 8251A to reset the **parity**, **overrun**, and **framing** error flags in the 8251A status register.
- Initializing **RTS** bit to 1 will cause the 8251A to assert its **request-to-send** (**RTS/**) output low. This signal is sent to a modem to ask whether a modem and the receiving system are ready for a data character to be sent.
- Initializing **IR** bit to 1 will cause 8251A to be internally reset. After the software- reset command, a new mode word must be sent.
- Initializing **EH** bit to 1 will cause 8251A to enter hunt mode (search for **SYN** characters, and is used only in synchronous mode.

8251 Status Word



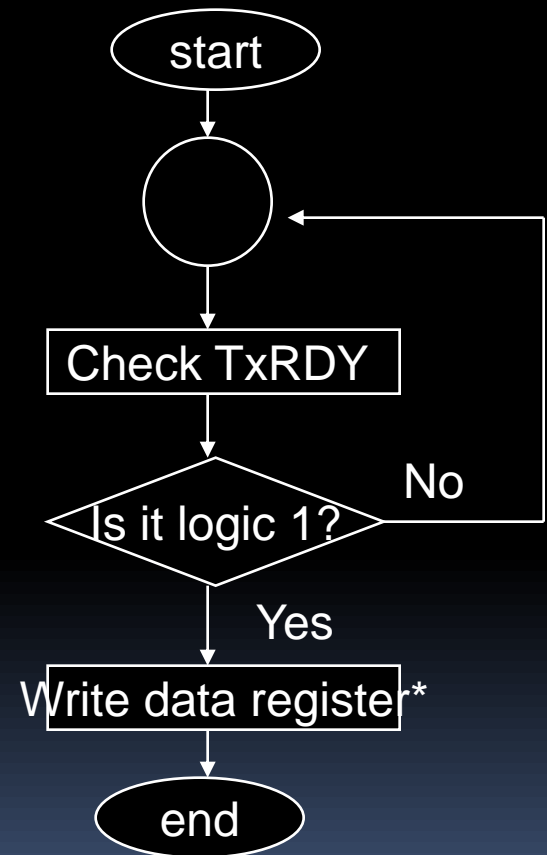
Simple Serial I/O Procedures

□ Read



* This clears RxRDY

□ Write



* This clears TxRDY

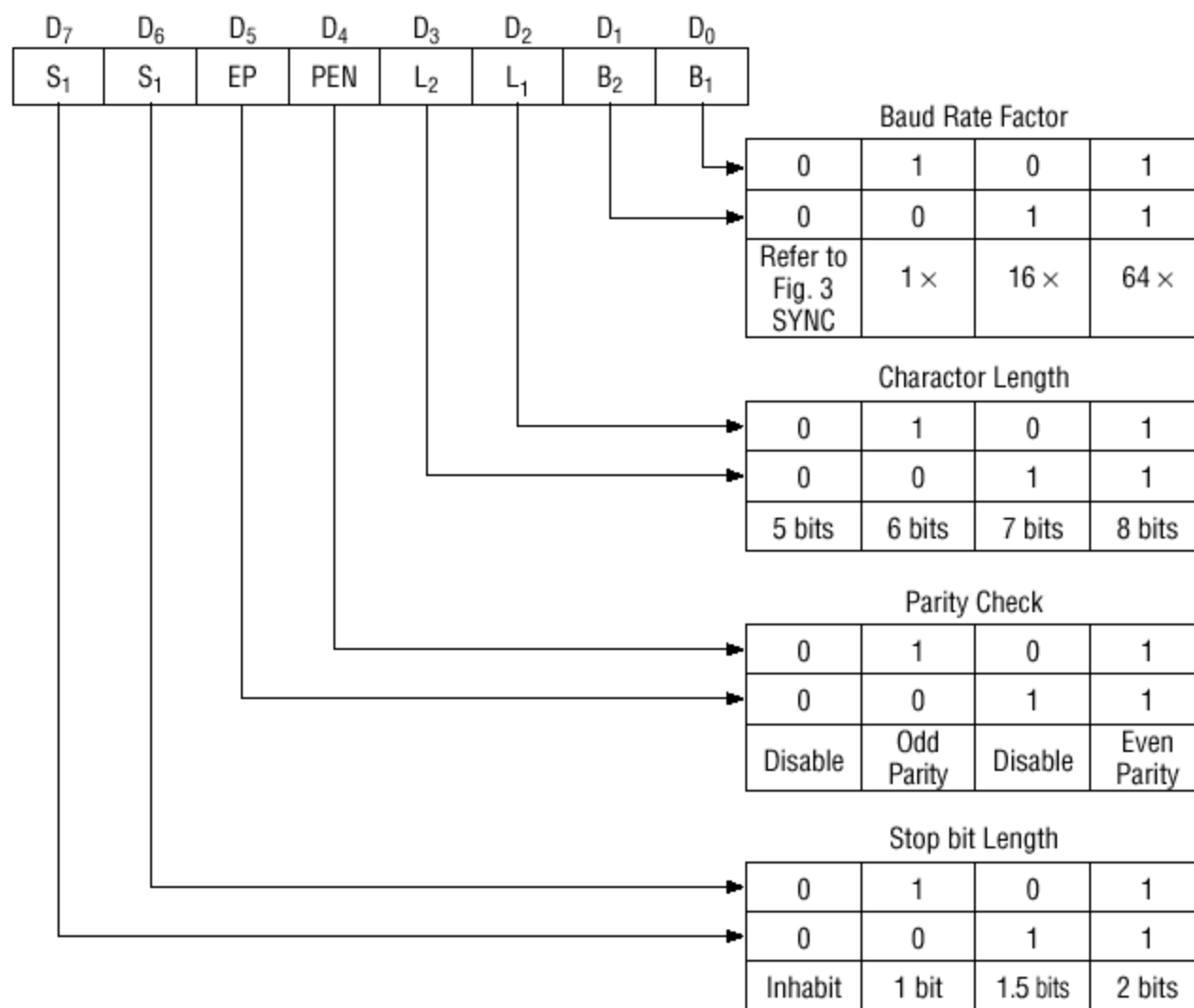


Fig. 2 Bit Configuration of Mode Instruction (Asynchronous)

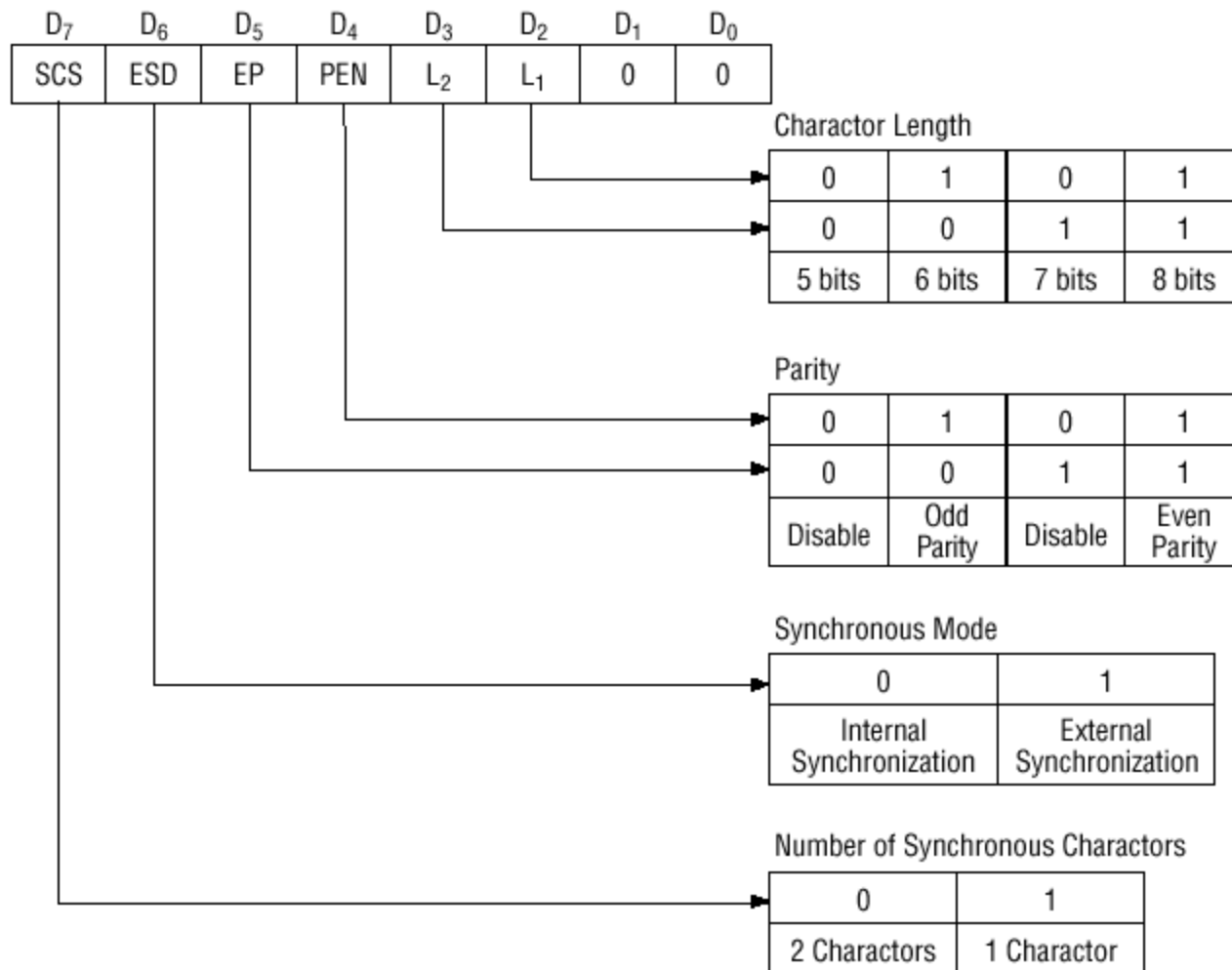
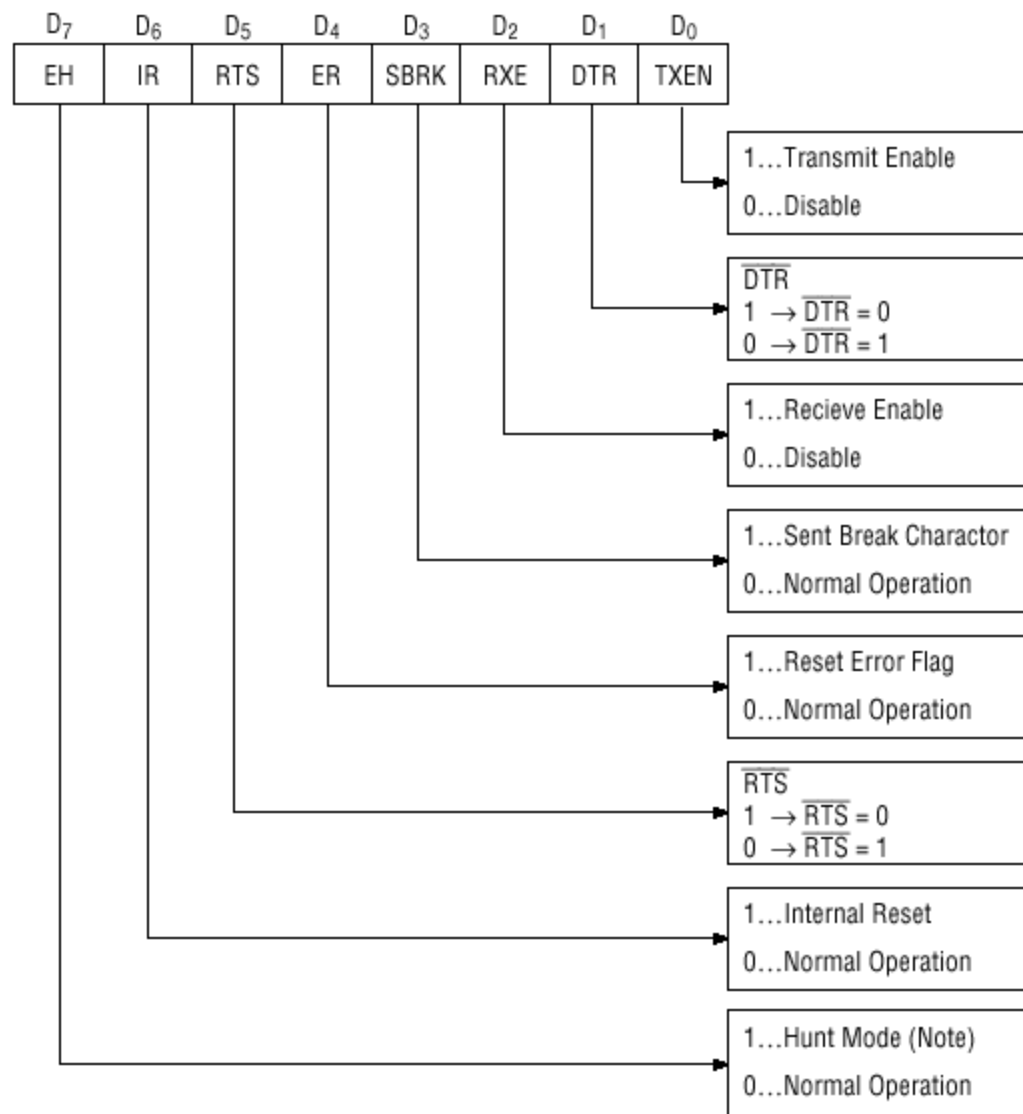


Fig. 3 Bit Configuration of Mode Instruction (Synchronous)



Note: Seach mode for synchronous charactors in synchronous mode.

Fig. 4 Bit Configuration of Command

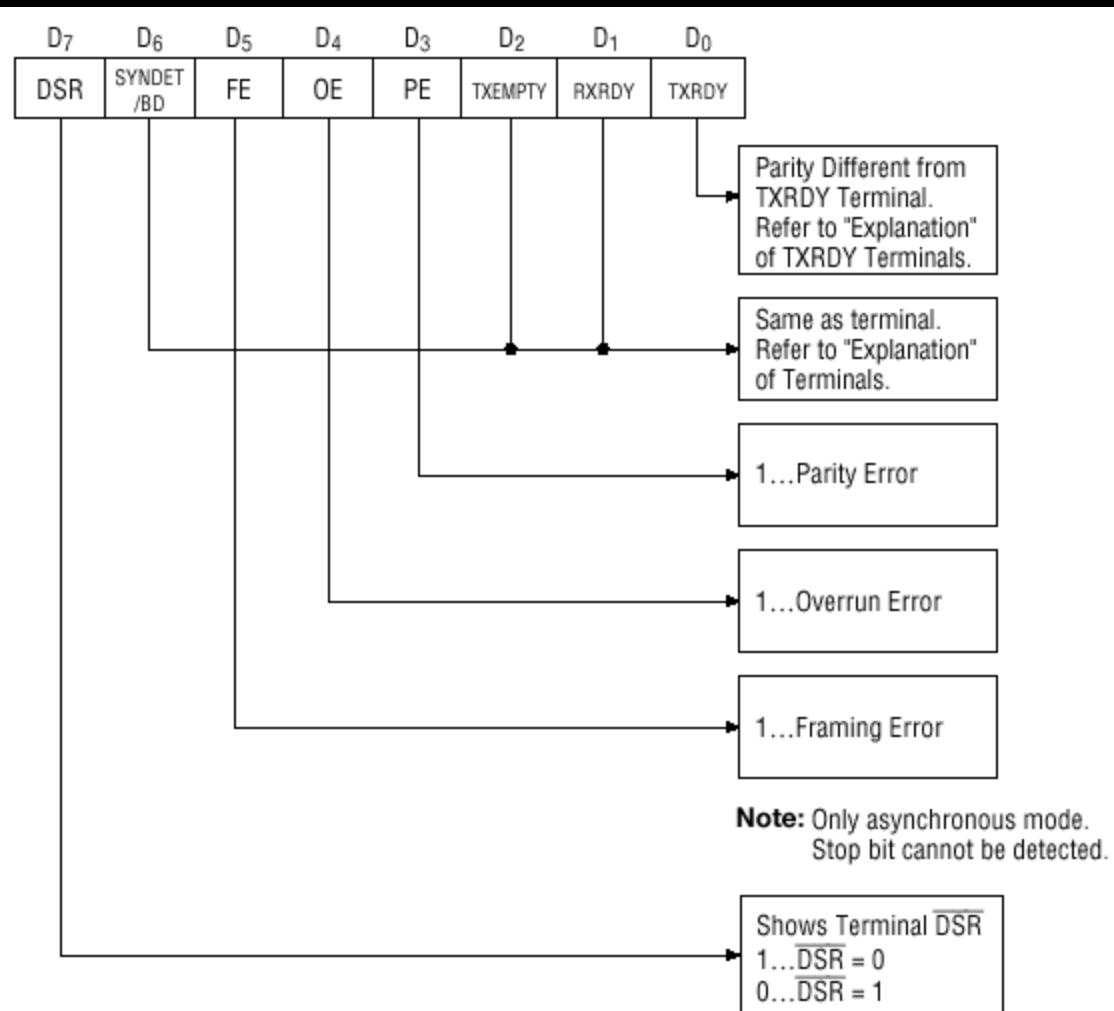


Fig. 5 Bit Configuration of Status Word

8251 A Serial Communication Interface

- The 8251A internally interprets the C/D, RD and WR signals as follow:

C/ \overline{D} (=A ₀)	\overline{RD}	\overline{WR}	
0	0	1	Data input from the data-in buffer
0	1	0	Data output to the data-out buffer
1	0	1	Status register is put on data bus
1	1	0	Data bus is put in mode, control or sync character register

- Whether the mode, control or sync character register is selected depends on the accessing sequence.

Example 1

- A program sequence which initializes the mode register and gives a command to enable the transmitter and begin an asynchronous transmission of 7-bit characters followed by an even-parity bit and 2 stop bits is:

MOV AL,1111010B

OUT 51H,AL

MOV AL,00110011B

OUT 51H,AL

Example 2

- This sequence assumes that the mode and control registers are at address 51H and the clock frequencies are to be 16 times the corresponding baud rates.

The sequence:

MOV AL,00111000B

OUT 51H,AL

MOV AL,16H

OUT 51H,AL

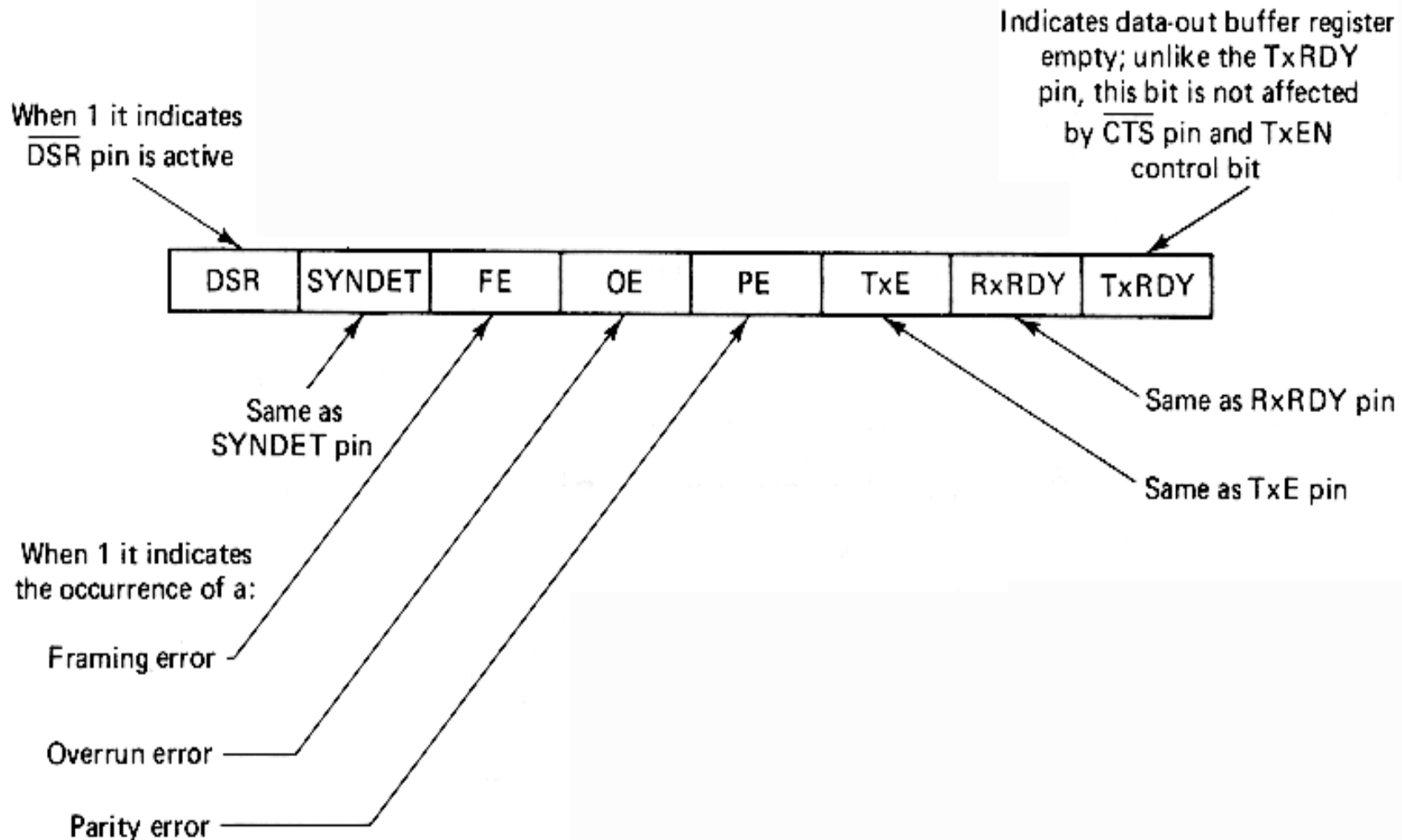
OUT 51H,AL

MOV AL,10010100B

OUT 51H,AL


would cause the same 8251A to be put in synchronous mode and to begin searching for two successive ASCII sync characters

Format of the status register





Example 3

- A typical program sequence which uses programmed I/O to input 80 characters from the 8251A, whose data buffer register's address is 0050, and put them in the memory buffer beginning at LINE.
- 

```

        MOV     AL,00110101B           ;ENABLE TRANSMITTER AND RECEIVER
        OUT     51H,AL                 ;AND CLEAR ERROR BITS
        MOV     DI,0                   ;INITIALIZE INDEX
        MOV     CX,80                  ;PUT COUNT IN CX
BEGIN:  IN      AL,51H
        TEST    AL,02H
        JZ      BEGIN
        IN      AL,50H                 ;INPUT CHARACTER AND
        MOV     LINE[DI],AL           ;PUT IN LINE BUFFER
        INC     DI
        IN      AL,51H                 ;CHECK ERROR
        TEST    AL,00111000B          ;BITS AND
        JNZ     ERROR                 ;IF NO ERROR IS FOUND
        LOOP    BEGIN                 ;CONTINUE INPUTTING
        JMP     SHORT EXIT
ERROR:  CALL    NEAR PTR ER_ROUT       ;ELSE CALL ERR_ROUT
EXIT:   .
        .
        .

```

Example 4

- 1000 000 0 : data register address: xx80h
- 1000 000 1: control or status register address: xx81h
- Mode word:
 - 2 stop bits. no parity, 8 bit characters. Baud rate factor of 16 (1200 Kbps)
 - 1110 1110 =EEh
- Command Word:
- 0001 0101 = 15h ; enable TxRDY and RxRDY and reset all flags first

INIT8251:MOV AL,0EEh

OUT 81h, AL

MOV AL, 15h

OUT 81h, AL

Initialize the Mode Word and Command Word

CHKRX:IN AL,81h

ROR AL,1

ROR AL,1

JNC CHKRX

Receive Ready?

IN AL,80h

NOT AL

MOV BL,AL

If Ready get data

CHKTX:IN AL,81h

ROR AL,1

JNC CHKTX

OUT 80h,AL

JMP CHKRX

Send data if the T buffer register is available



Thank You