8251 USART

Dr. Manju Khurana
Assistant Professor, CSED
TIET, Patiala
manju.khurana@thapar.edu

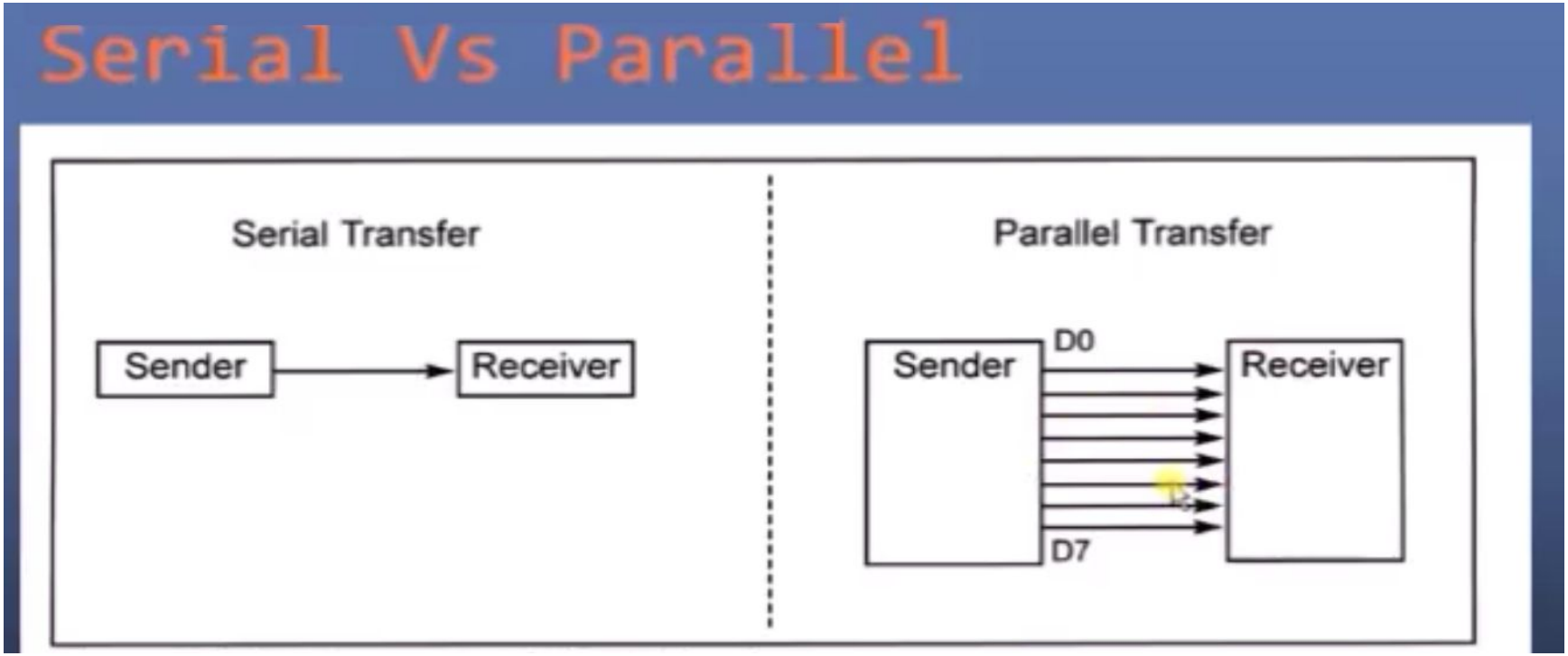
Methods of Communication

Parallel Data Transfer

Parallel data transfer is used to transfer data in the block of 8 or 16-bit at a time. This is normally used for the fast data transferring.

Serial Data Transfer

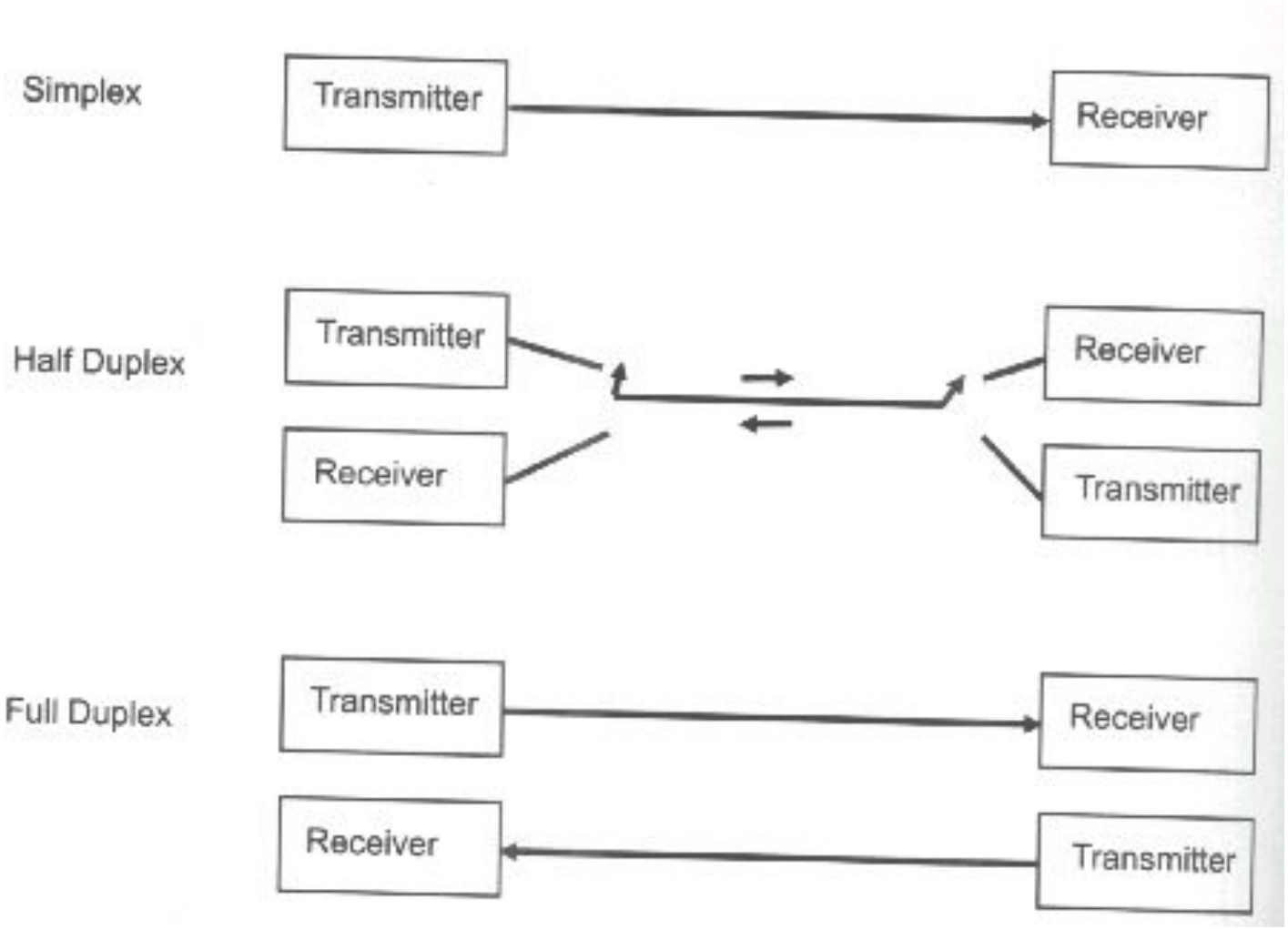
Serial data transfer is used to transfer single data (1-bit) at a time. This type of data transfer is used for the slow data transferring.



Serial/Parallel

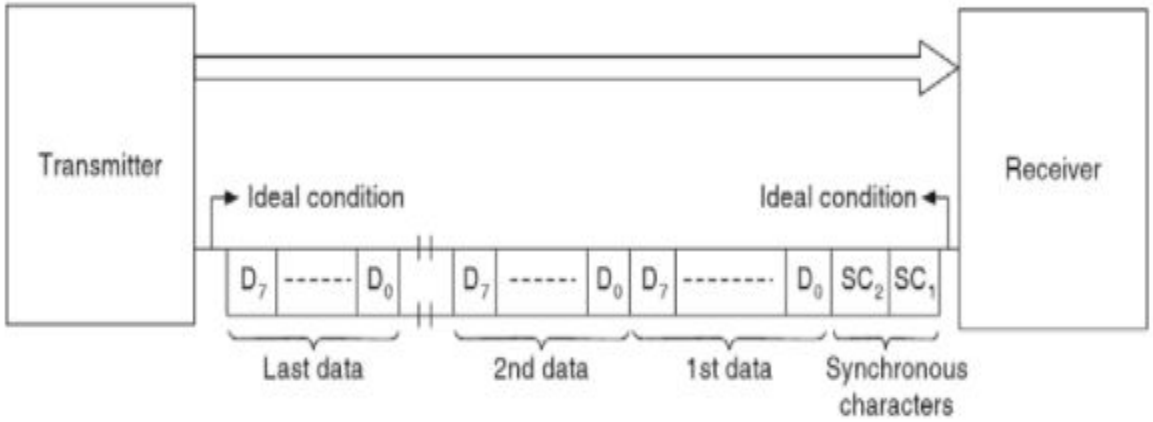
Serial data transfer	Parallel data transfer
1. 1 bit of data is transferred at a time.	1. 8/16 bits of data is transferred at a time.
2. Two lines are required to be connected. (one for data transfer and one to provide ground).	2. 9/17 lines are required to be connected. (8/16 bits for data transfer and one for ground).
3. Slow data transfer.	3. Fast data transfer.
4. Used for long distance communication.	4. Used for Short distance communication.

Types of Transmission



Formats of Serial Data Transfer

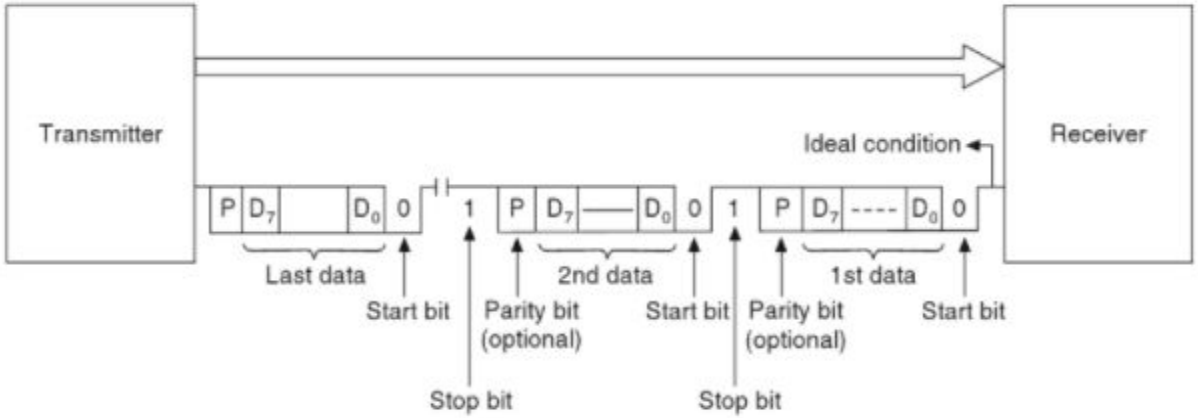
- Synchronous Serial Data Transfer



In this, block of data bytes are transferred serially at a time. The numbers of data bytes are not limited. The data bytes are transferred one after the other and so on up to the end. In this method in the starting of data transfer, first transfer is of synchronous characters in which one synchronous character is compulsory for the data transfer and the other is optional. Synchronous characters are informed to the receiver which will start the transferring of data, i.e. it will provide synchronization between transmitter and receiver. In one synchronous character more than 256 receivers are connected, i.e. from 00 to FFh

Formats of Serial Data Transfer

- Asynchronous Serial Data Transfer



In this, one byte of data is transferred serially at a time. Normally, the output line of transmitter is maintained high, and before transferring the data, the output line is made low (Logic 0) for one clock pulse to indicate start of data byte, therefore, it is called the start bit. After the data bits are transferred serially D_0 to D_7 (D_7 last), the output is maintained high for specific period to indicate end of data. These bits are called stop bits. These stop bits used are logical high signal, so after completion of stop bits, the same logical level is maintained on output line. At the end of every data bytes parity bit will also be sent. This is optional and only used to maintain specific parity of the data bits sent

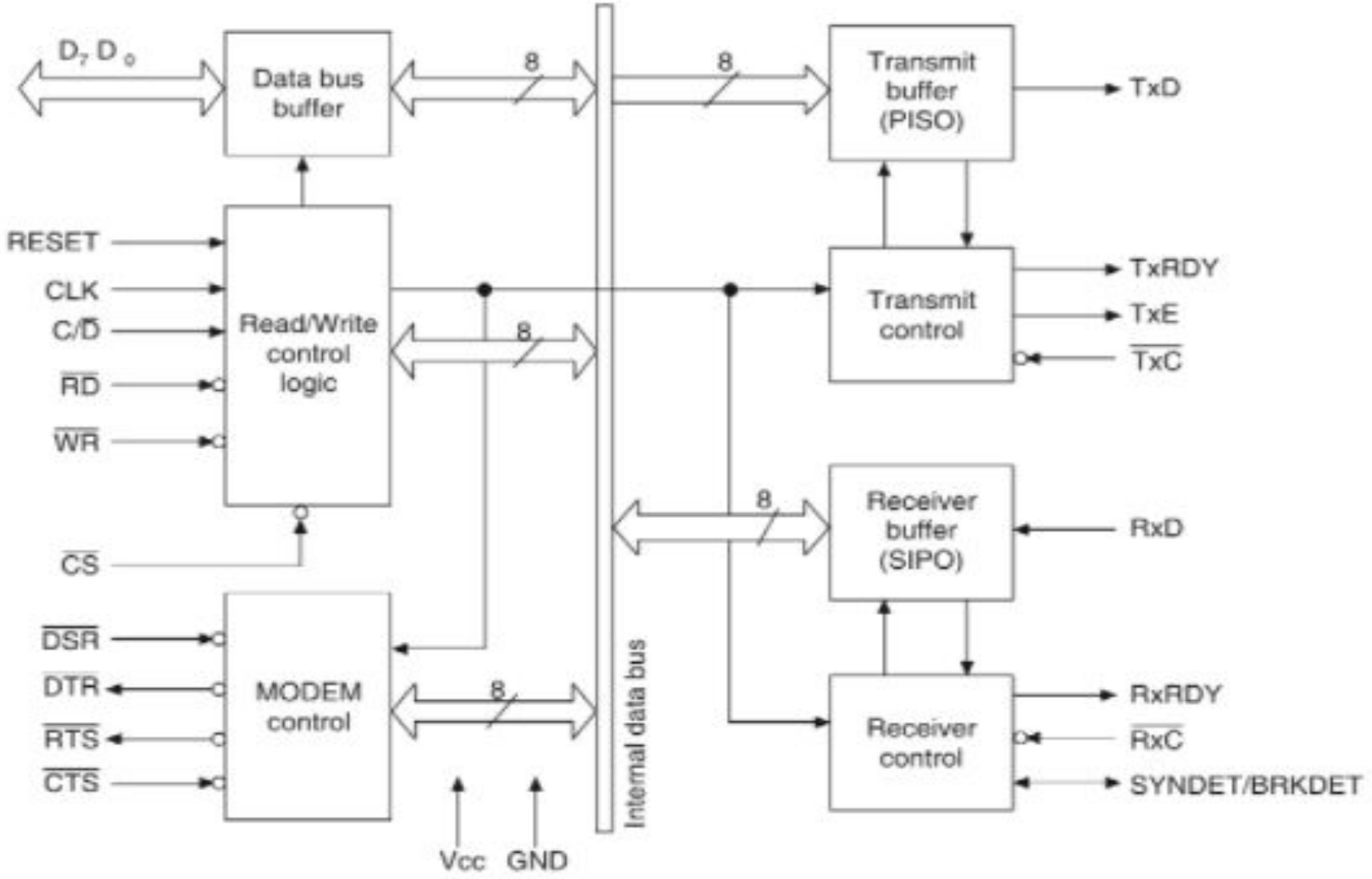
Differences between Synchronous and Asynchronous Serial Data Transfer

Synchronous data transfer	*Asynchronous data transfer*
1. It is used to transfer a group of characters at a time.	It is used to transfer one character at a time
2. Used for high data transfer rates ≥ 20 Kbps	Used for data transfer rates ≤ 20 Kbps.
3. Synchronous characters are transmitted along with the group of characters	Synchronous characters are not transmitted along with the characters
4. No start and stop bits for each character is used	Start and stop bit for each character is present which forms a frame
5. One clock is used for both transmitter and receiver	Two separate clock inputs can be used for transmitter and receiver
6. Since synchronization is involved, this can be implemented by using hardware only	No synchronization is required hence hardware and software implementation is possible.

Features of 8251

1. It supports both synchronous and asynchronous modes of operation.
2. In synchronous mode, it supports 5–8 bits characters.
3. In asynchronous mode, it supports 5–8 bits characters, clock rate selectable 1, 16 and 64 times baud rate, break character generation, 1, 1 ½ or 2 stop bits, false start bit detection, odd, even or no parity generation and detection, and automatic break detect circuitry is available.
4. Asynchronous baud rate DC to 19.2 Kbaud.
5. Synchronous baud rate DC to 64 Kbaud.
6. Transmitter and receiver contain full duplex, double buffered system.
7. Separate transmitter and receiver clock inputs for transmitter and receiver, so transmitter and receiver can be operated in different baud rates.
8. Single +5 V supply
9. Available in 28 pin DIP package, all inputs and outputs are TTL-compatible.

Block diagram of the 8251 USART



- Data Bus Buffer

A three-state, bidirectional 8-bit data bus is used to interface the 8251 USART data bus to the system data bus. It is internally connected to the data bus and its outer pins D_7–D_0 are connected to the system data bus directly. The directions of data buffer are decided by read and write control signals. When signal read is activated, then it transmits data to the system. When write signal is activated, then it receives data from the system data bus. This reading and writing operation is achieved by IN and OUT microprocessor instructions. The data bus buffer is also used to transfer control word, status word, data for transmitter, and data from receiver, depending on the signal given by read/write control logic.

- Read/Write Control Logic

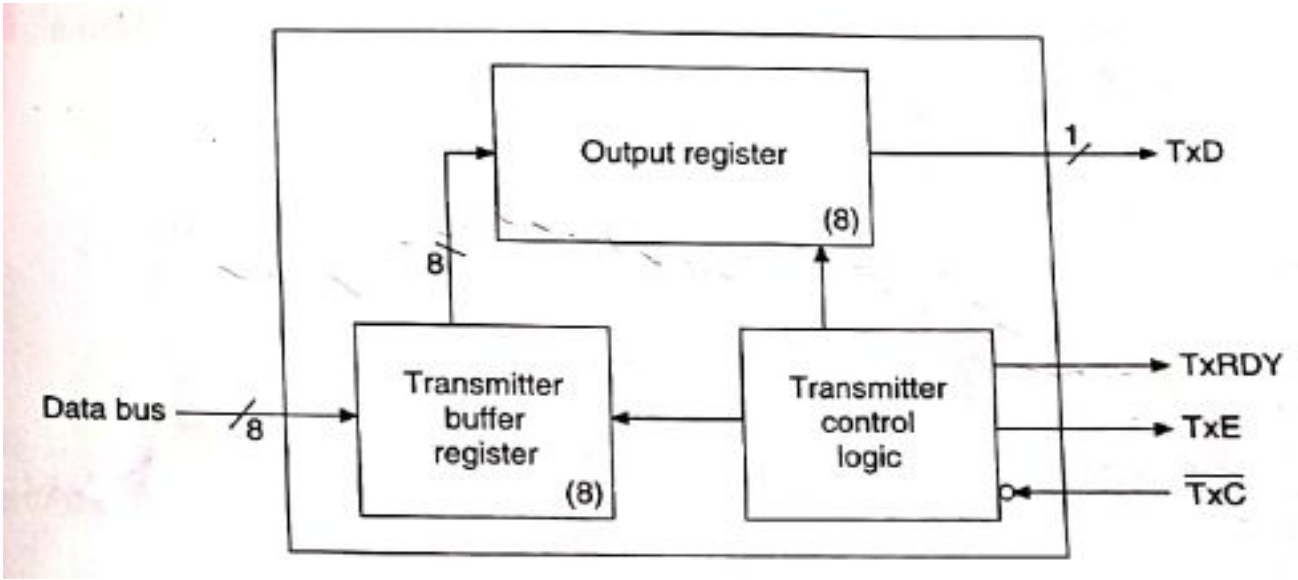
This is a control block for the overall device. This block accepts different control signals from the control bus and generates control signals for device operation. The 8251 USART itself will get selected when chip select ($\overline{CS}$) is active, i.e. low. When $\overline{CS}$ = low, then 8251 USART accepts $C/\overline{D}$ pin and decides which part to activate $C/\overline{D} = 1$ selects control part, and $C/\overline{D} = 0$ selects data part. These are shown in the following table:

$\overline{CS}$	$C/\overline{D}$	$\overline{RD}$	$\overline{WR}$	*Data transfer*
0	0	0	1	8251 data register of receiver to the data bus
0	0	1	0	Data bus to 8251 data register of transmitter
0	1	0	1	Status word to data bus
0	1	1	0	Data bus to control word
1	X	X	X	Data bus tristated

- Transmitter Section

TxD —Transmit data
TxRDY —Transmitter ready
TxE —Transmitter empty
TxC —Transmitter clock

This section consists of transmit buffer, transmit control block, and output register. The transmitter buffer register accepts data from the data bus buffer through internal data bus, if $\overline{CS} = 0$, $C/\overline{D} = 0$, $\overline{RD} = 1$, and $\overline{WR} = 0$. The contents of the transmitter buffer are automatically transferred to the output register, if the output register does not contain any data, i.e. output register is empty. The data is shifted out serially on the TxD pin, along with the appropriate bits are also added depending on mode selected i.e. synchronous (synchronous characters are sent) or asynchronous mode (start and stop bits are sent)



- Transmitter Section

If transmit buffer register does not contain any data, then TxRDY (TxRDY =1) is generated by transmit control to indicate microprocessor to send next data for transmission. If output register does not contain data, then TxE is generated by transmit control to indicate peripheral about inavailability of data for transmission.

If TxRDY = 1, this indicates that the transmit buffer register is empty and microprocessor is ready for data transfer to transmit buffer register. The data is first come to the transmit buffer register and then it transfers to the output register, and finally from output register it transfers one bit at a time on TxD pin.

If TxRDY = 0, this indicates that the transmit buffer register is not empty, i.e. it contains the data. After transferring last data from microprocessor to the transmit buffer register, this last data is transfered from output register to the TxD pin, that means transmit buffer register and output register are both empty, then it gives TxE = 1.

$\overline{\text{TxC}}$ is used to apply negative edge clock pulses.

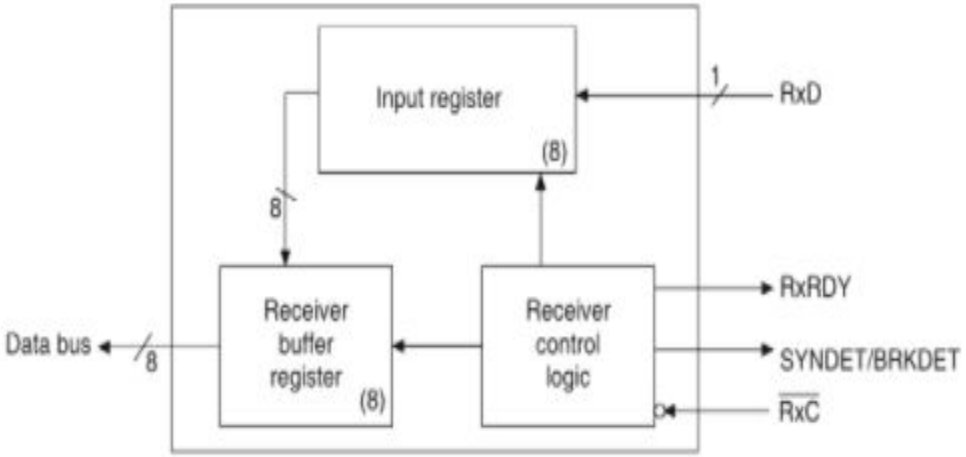
Under the following conditions TxD =1

1. Transmit buffer register is empty
2. TxE =1
3. CTS = 0
4. Upon master reset signal.

- Receiver Section

RxD	—	Receive data
RxRDY	—	Receiver ready
SYNDET/BRKDET	—	Synchronous detect / break detect
$\overline{\text{RxC}}$	—	Receiver clock

This section consists of receive buffer register, receive control block, and input register. The receiver section accepts serial data on RxD pin. The input register converts the serial data into parallel form. The working of conversion depends on the mode selected. In asynchronous mode, it will check for start bit, and if the start bit is detected, then the bits after start bit are converted to parallel form and are transferred to receiver buffer register. In synchronous mode after SYNDET signal input register will go on accepting data bits, convert to parallel form, and load into receiver buffer register



In asynchronous mode after data bits are accepted receiver checks programmed parity bit, if it is not same, then an error bit in status register is set.

When the data byte is transferred from input register to receiver buffer register, the control logic generates a signal RXRDY to signal microprocessor about availability of data byte to be read by microprocessor.

- MODEM Control

For sending data over long distances the telephone lines are used. The telephone lines are analog in nature, so MODEMs (Modulator–Demodulator) are used to convert digital data to analog data. To control or communicate MODEMs with 8251, the 8251 USART provides a block called MODEM control. The MODEM control block uses different lines such as $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, $\overline{\text{DTR}}$, and $\overline{\text{DSR}}$ for MODEM.

(a) **$\overline{\text{DTR}}$ (Data terminal ready):** When the terminal is made ON, it will perform different operations. When it is ready for data transmission/reception, it generates a signal $\overline{\text{DTR}}$ to indicate its readiness for data transfer.

(b) **$\overline{\text{DSR}}$ (Data set ready):** This input may be used as a general purpose, one bit inverting input port. Its status can be checked by the microprocessor using a status read operation. This is normally used to check, if the data set is ready when communicating with a MODEM.

(c) **$\overline{\text{CTS}}$ (Clear to send):** When the MODEM is ready to transmit data, it assets $\overline{\text{CTS}}$ signal to terminal. The terminal upon receiving $\overline{\text{CTS}}$ sends serial data character to the MODEM.

(d) **$\overline{\text{RTS}}$ (Request to send):** When the terminal is ready to transmit data and has a data character to be transmitted, the terminal assets a signal $\overline{\text{RTS}}$ to the MODEM.


Thank you!