

**Microprocessors and Microcontrollers**  
**Prof. Santanu Chattopadhyay**  
**Department of E & EC Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 20**  
**8085 Microprocessors (Contd.)**

When we are talking about this data transfer, so it can be classified into two categories, one category is synchronous data transfer.

(Refer Slide Time: 00:23)

**Synchronous Data Transmission**

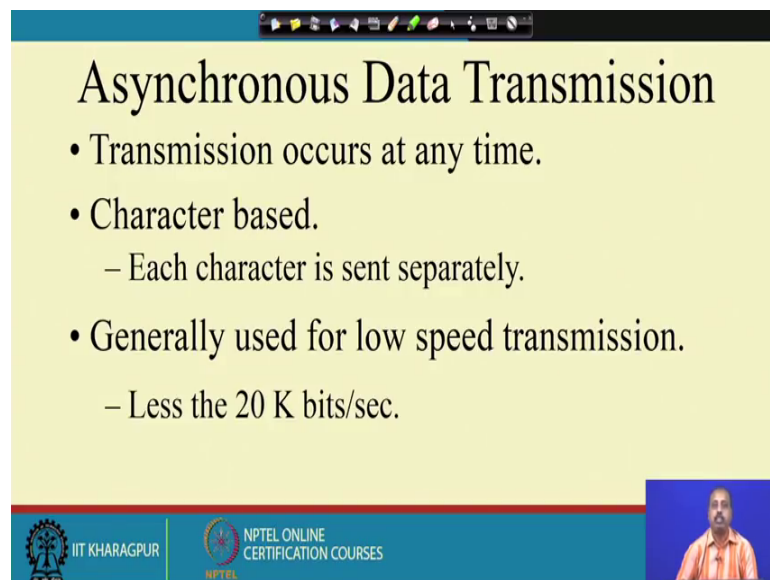
- The transmitter and receiver are synchronized.
  - A sequence of synchronization signals is sent before the communication begins.
- Usually used for high speed transmission.
  - More than 20 K bits/sec.
- Message based.
  - Synchronization occurs at the beginning of a long message.

When the transmission between the two devices they are with respect to a common clock, if we have a common clock like if this is the transmitter and this is the receiver. So, this data transfer takes place and they are driven by a common clock signal. So, this type of transfer they are known as synchronous data transfer. So, sequence of synchronizing signal can be sent before the communication take place.

So, like for the telling that I want to simple, I want to the same data now, so that way this clock is acting as the synchronization signal. So, that way or it may be that there are some signals that are sent from transmitter to receiver telling that it is going to start the communication and data if the bit rate and everything can be received and this is usually high speed data transmission. So, we have got no transmission of the range, range of 2 kilo bits per second and things like that and it is a message based communication.

So, synchronization occurs at the beginning of a long message. So, if you are sending a long message at the beginnings of synchronization bits are sent so that the transmitter and receiver they get synchronized and then the data transfer starts. So, this way we can have this synchronous data transfer.

(Refer Slide Time: 01:52)



The slide features a yellow background with a blue header and footer. At the top, there is a navigation bar with various icons. The main content area contains the title 'Asynchronous Data Transmission' in a large, black, serif font. Below the title, there are three bullet points in a black serif font. The first bullet point is 'Transmission occurs at any time.' The second is 'Character based.' with a sub-bullet '– Each character is sent separately.' The third is 'Generally used for low speed transmission.' with a sub-bullet '– Less the 20 K bits/sec.' The footer contains the IIT KHARAGPUR logo on the left, the NPTEL ONLINE CERTIFICATION COURSES logo in the center, and a small video inset of a man in an orange shirt on the right.

## Asynchronous Data Transmission

- Transmission occurs at any time.
- Character based.
  - Each character is sent separately.
- Generally used for low speed transmission.
  - Less the 20 K bits/sec.

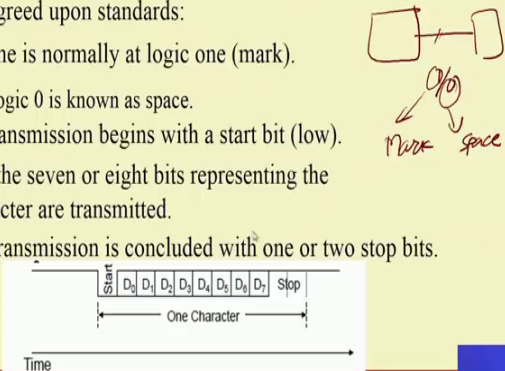
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

On the other hand we can have a synchronous data transfer where the transmission can occur at any point of time. So, and it is character based transmission every character will be sent separately and it is less low speed transmission less than 20 kilo bits per second. However, the advantage is that we do not need any synchronization sequence between the transmitter and receiver so that way the communication overhead is less you can say.

(Refer Slide Time: 02:21)

## Asynchronous Data Transmission

- Follows agreed upon standards:
  - The line is normally at logic one (mark).
    - Logic 0 is known as space.
  - The transmission begins with a start bit (low).
  - Then the seven or eight bits representing the character are transmitted.
  - The transmission is concluded with one or two stop bits.



The diagram illustrates the timing of asynchronous data transmission. It shows a horizontal timeline labeled 'Time'. The signal starts at a high level (logic one, 'mark'). A handwritten diagram to the right shows a high level labeled 'Mark' and a low level labeled 'Space'. The transmission begins with a start bit (low). This is followed by data bits labeled D<sub>0</sub>, D<sub>1</sub>, D<sub>2</sub>, D<sub>3</sub>, D<sub>4</sub>, D<sub>5</sub>, D<sub>6</sub>, and D<sub>7</sub>. The transmission concludes with one or two stop bits (high). The entire sequence is labeled 'One Character'.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, the generally the standard that is followed is like this. The line is normally at logic one. So, when nothing no, no transmission is taking place between the devices. So, line is at logic high. So, it is like this. So, it is at logic high then. So, this is the one is called mark and logic 0 is called a space. So, when you are transmitting a bit, so the bit sorry, when you are transmitting a transmitting a bit. So, this bit may be logic high or logic low. So, we can have these two devices. So, this bit that is transmitting. So, it may be 1 or it may be 0. So, in the transmission terminology, this 1 will be called a mark this is a technical term for mark and for 0 is called a space some sort. So, this is a very old nomenclature, but that is followed in the literature. So, we have got this mark and space.

And transmission begins with a start bit which is low. So, you can understand from this diagram that we have got this thing initially it is high and then this low it means that the this is the start bit fine. Then after that 7 or 8 bits are transmitted representing the character that we want to transmit. So, this D<sub>0</sub> to D<sub>7</sub> are the bits that are transmitted. So, normally we are transmitting in ASCII code. So, ASCII is a 7 bit code or if you go to extended ASCII says 8 bit code or then there is a EBCDIC coding which is 8 bit code. So, that way whatever be the coding, those 8 bits will be transmitted and the transmission is concluded by 2 stop bits.

So, these are the 2 stop bits which are high for 2 successive clock cycles and then successive clock 2 successive bit times and then they will be taken as the end of the

transmission. So, again for if you are going to transmit the next characters again here to send a start bit and send the next character. So, this way this asynchronous data transmission will take place.

Now, between the episode, in this transmission, we can have this type of classification simplex, duplex and half-duplex; simplex, half-duplex and full-duplex.

(Refer Slide Time: 04:49)

**Simplex and Duplex Transmission**

- Simplex.
  - One-way transmission.
  - Only one wire is needed to connect the two devices
  - Like communication from computer to a printer.
- Half-Duplex.
  - Two-way transmission but one way at a time.
  - One wire is sufficient.
- Full-Duplex.
  - Data flows both ways at the same time.
  - Two wires are needed.
  - Like transmission between two computers.

The slide includes three diagrams: 1. Simplex: A box labeled 'TX' with an arrow pointing to a box labeled 'RX'. 2. Half-Duplex: Two boxes labeled 'D1' and 'D2' with arrows pointing in opposite directions between them. 3. Full-Duplex: Two boxes labeled 'D1' and 'D2' with arrows pointing in opposite directions between them, representing simultaneous two-way communication.

Logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES are visible at the bottom of the slide.

So, simplex is one way transmission. So, it probably there is a there are two devices and out of them one of them is a master other one is a slave and data transfer is always either data transfer is always from the master to the slave or it is always from slave to master only in one direction. So, it cannot be in both the directions.

Half-duplex means that transmission can take place in both the directions, but any at any point of time. So, it is in a single direction only. So, that is half-duplex type of communication and we have got full-duplex where the data flows both ways at the same time. So, we can have this, so half of this simplex transmission means, so it is, that the direction is fixed. So, if this is the transmitter and this is the receiver the directional transmission is always from the transmitter to the receiver.

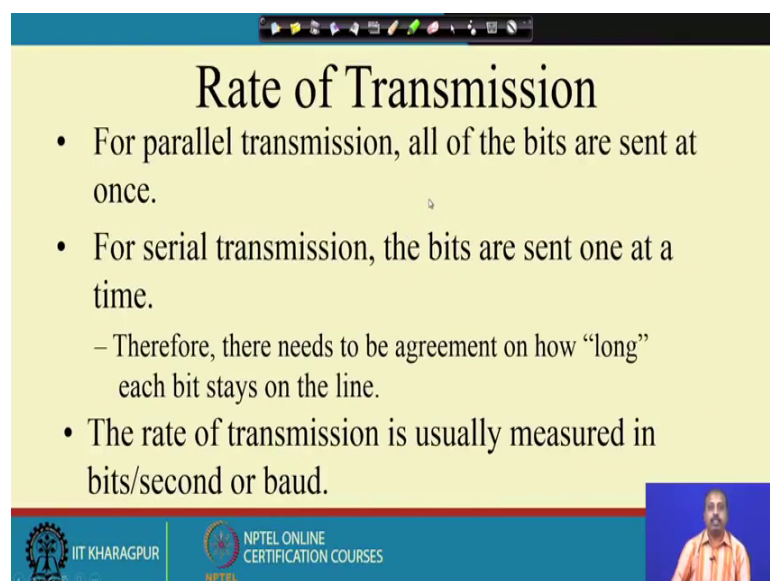
Then in case of, in case, it is very simple. So, only one wire is needed to connect between the devices. So, typical example may be a communication from computer to printer. So, of course, there are some status checks and all that. So, ignoring those

statistics actual the data transfer is from the computer to the printer only. So, that way it is a simplex type of connection. Then half-duplex connection, here this data transfer can be both in both the directions. So, at some point of time it is from device one to device two sometimes it is from device 2 to device 1. So, typical example is the telephone line where two people are talking, at one point of time at one point of time the line is in one direction only, and it switches continually and. So, that it at some point of for some time it is in from person 1 to person 2 and again in the next slot it is from person 2 to person 1, but this happens so fast that we may not be able to understand that switching.

So, in this case also one wire is sufficient and the best connection is the full-duplex connection. So, you have two wires will be needed. So, we have transmitting and the transmission and receiving where so they are separate. So, we can have the transmitter and the receiver. So, this, this is a device 1, this is device 2. So, this is transmit data which is the known as TX D and this is the received data. Similarly for this side we have got the transmit data, we have got transmit data here and the receive data here. So, this way we can have this we can have this transmission and receiving taking place simultaneously.

So, in this case two wires are necessary one for transmission and one for receiving and. So, it is like communication between two computers. So, we have got this full-duplex communication.

(Refer Slide Time: 08:13)



## Rate of Transmission

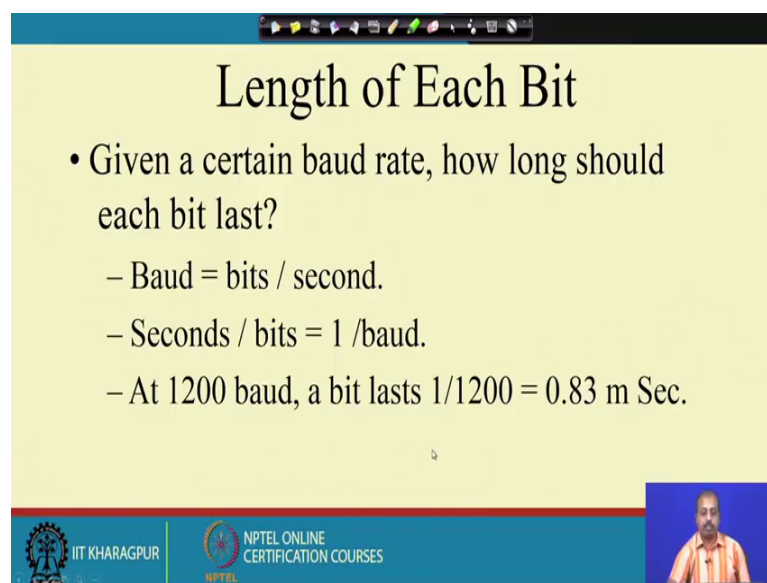
- For parallel transmission, all of the bits are sent at once.
- For serial transmission, the bits are sent one at a time.
  - Therefore, there needs to be agreement on how “long” each bit stays on the line.
- The rate of transmission is usually measured in bits/second or baud.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Next important thing is the rate of transmission. So, parallel transmission. So, all 8 bits are sent simultaneously. So, if you are sending one character at a time. So, even for parallel communication all the 8 bits will go together. For serial transmission the bits will be sent one at a time. So, 1 bit will go at a time and so it will take longer time. So, we have to endure because the bits will go serially and since that transmission and receiving devices. So, they may not be operating on the same clock. So, we have to have an agreement like how long is one bit time. So, as you if you look into this diagram, if you look into this diagram you see that I have said that this is D 0 this is D 1. So, apparently it seems that this is the this D 0 D 1 say demarcation point like when D 0 ends and D 1 starts. So, there must be an agreement about what is the duration of this bit D 0. So, what is the individual bit durations. So, this transmitter should activate the value of D 0 for that much time and receiver should expect that value of D 0 within that time, so that way that that agreement should be there.

So, there needs to be an agreement on how long each bit stays on the line. So, rate of transmission is measured at bits per second or baud. So, how many bits you are transmitting per second. So, once this ball bit rate is fixed bit rate is agreed upon the transmitter and receiver can find out what is the bit time. So, for that bit time the individual bits will be sent.

(Refer Slide Time: 09:53)



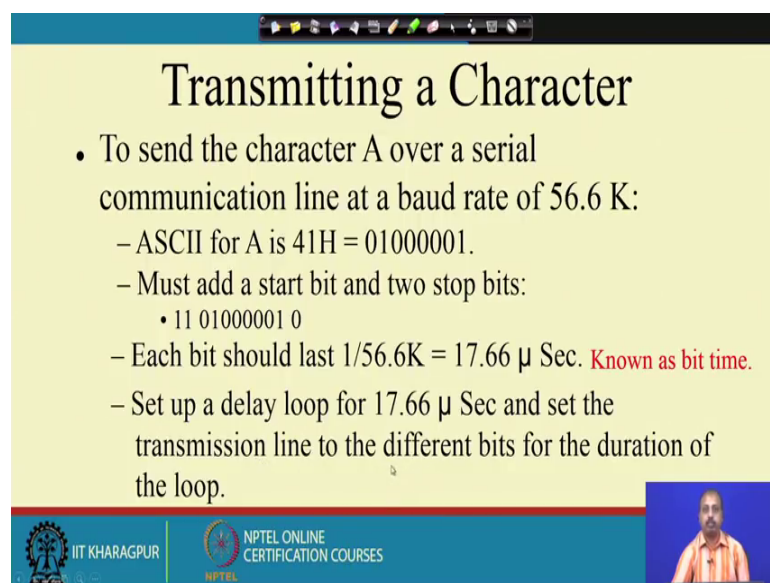
**Length of Each Bit**

- Given a certain baud rate, how long should each bit last?
  - Baud = bits / second.
  - Seconds / bits = 1 /baud.
  - At 1200 baud, a bit lasts  $1/1200 = 0.83$  m Sec.

The slide features a yellow background with a blue header and footer. The footer contains the logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES. A small video inset in the bottom right corner shows a man in an orange shirt speaking.

So, we have got for a certain baud rate how long should it each bit last. So, you can find out like baud bits per second. So, second per seconds divided by bits is 1 baud. So, at 1200 baud a bit will last  $1 \text{ by } 1200 = 0.83 \text{ millisecond}$ . Now, this term baud is slightly misnomer in here because in communication technology. So, this baud is the symbols per second, but here we are taking each bit as a symbol so we will be simply talking them as bits per second keeping in mind that baud is actually symbols per second. So, this 1200 baud rate, we can say that individual bits will last or will be therefore,  $1 \text{ by } 1200 = 0.83 \text{ millisecond}$ . So, this bit timing is fixed.

(Refer Slide Time: 10:44)



**Transmitting a Character**

- To send the character A over a serial communication line at a baud rate of 56.6 K:
  - ASCII for A is 41H = 01000001.
  - Must add a start bit and two stop bits:
    - 11 01000001 0
  - Each bit should last  $1/56.6\text{K} = 17.66 \mu \text{ Sec}$ . **Known as bit time.**
  - Set up a delay loop for 17.66  $\mu \text{ Sec}$  and set the transmission line to the different bits for the duration of the loop.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

And then to transmit a character shape, we want to send this character a over serial communication line at a board rate of 56.6 kilo. Now, first of all we find out the ASCII of a ASCII of a is forty one hicks that is 65 decimal. So, this is the corresponding 8 bit pattern. So, now, we have to add one start bit and two stop bits. So, first this start bit will go, then this bit pattern will go and then two stop bits will go total 8 plus 1, 9 plus 2, 11 bits are to be transmitted.

Now, at 56.6 kilo baud rate, the each bit timing is 17.66 microsecond and so we have to set up a delay loop for 17.66 microsecond after sending one character serially. So, we have to wait for this much of time and after that only we should change the change to the next bit. So, we have to set up a loop for this way at 17.66 micro second and then this

and say the transmission line two different bits for the duration of the loop so that way we have to do the serial transmission.

(Refer Slide Time: 12:02)

The slide is titled "Error Checking" and contains the following text:

- Various types of errors may occur during transmission.
  - To allow checking for these errors, additional information is transmitted with the data.
- Error checking techniques:
  - Parity Checking.
  - Checksum.
- These techniques are for error checking not correction.
  - They only indicate that an error has occurred.
  - They do not indicate where or what the correct information is

Handwritten on the slide is a diagram showing a bit stream "1101000" with a box around it. Above the box is the word "Parity" and below it is a checkmark. To the left of the box is a small 'x' and to the right is a small 'y'.

The slide footer includes the IIT KHARAGPUR logo, the NPTEL ONLINE CERTIFICATION COURSES logo, and a small video inset of a man in an orange shirt.

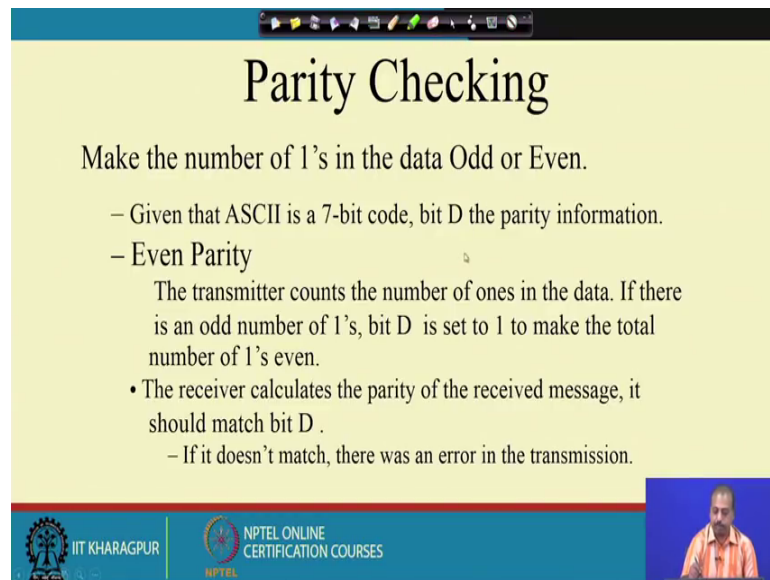
Another important issue with the serial transmission is the error checking. So, there can be various types of errors that can come while transmission is taking place. So, some of the bits may be modified some messages may not be received at all. So, like that, there are different types of errors depending upon the transmission policy that we are following.

So, we have to have some additional information sent to check at the receiver whether this transmission had some error or not. So, there are two very popular techniques for doing it, one is called parity checking and another is the checksum. So, parity checking is actually checking whether the total number of 1's in the bit stream that has been transmitted is even or odd.

So, two systems that, the two devices before starting the communication they can agree upon the policy like they can say that we will accept odd parity means the number of 1's transmitted must be odd always odd or they may agree upon an even parity they are telling that number 1's transmitted is always even.






(Refer Slide Time: 13:10)




## Parity Checking

Make the number of 1's in the data Odd or Even.

- Given that ASCII is a 7-bit code, bit D the parity information.
- Even Parity
  - The transmitter counts the number of ones in the data. If there is an odd number of 1's, bit D is set to 1 to make the total number of 1's even.
  - The receiver calculates the parity of the received message, it should match bit D.
    - If it doesn't match, there was an error in the transmission.



So, if this is that bit stream that is transmitted. So, if this is the bit stream so, if some bit stream is transmitted in that bit stream. So, we can say that if this is the bit stream to be transmitted and this bit stream has got say even the odd number even number of 1s. So, bit stream may be 1 1 0 1 1 0 0 0 this may be the 8 bits transmitted. So, we have a parity bit at the end of it. So, this bit is called parity bit and this parity bit, if we say that we will follow odd parity then this bit will be set to 1 and if we say that we will follow even parity then this bit will be set to 0.

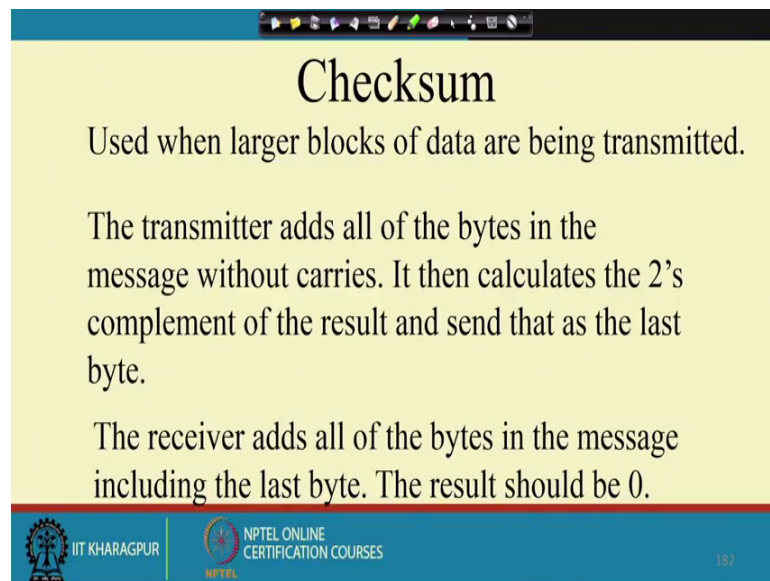
So, at the receiving end if any of these bits get corrupted say this bit has changed from 0 to 1. So, when you compute the parity at the receiving end. So, it will not match. So, we can check that there is a problem with the bit that is the bit stream that has been transmitted. So, though we cannot detect like what exactly has gone wrong, but we know that something has gone wrong and we can go we can try out retransmission of the bit stream.

So, another possibility is to use of checksum. So, this parity is very rudimentary type of checks the checksum means it computes some polynomial based on this bit stream that has been transmitted and the remainder of the polynomial. So, that is added at the end of the bit stream to as a pattern and at the receiving end which again check it. So, the multi errors may be detected by using this checksum facility. So, they are for checking, but not

correction that they will indicate that a error has occurred and they do not say like where and where is the error that cannot be told by these schemes.

So, next we will see the parity checking. So, the number of 1's in the as I was telling. So, to make the number of 1's in the data odd or even, ASCII is 7 bit code. So, we have got 1 bit extra. So, bit D of the parity into. So, we can 7 bit ASCII code is there. So, we can add another bit for parity information. So, making a total 8 bit now for even parity the transmitter will count number of 1's in the data if the number of once is an odd number then this bit D will be set to 1 and make the total number of 1's even and if the receiver will calculate the parity at the received message and it will again calculate the value of D and if the calculated value matches with the received D value; that means, it will assume that there is no error. Of course it does not guarantee because. So, if multiple bits got corrupted then there may be problem, but in general, for most of the cases. So, this may be correct.

(Refer Slide Time: 16:15)



**Checksum**

Used when larger blocks of data are being transmitted.

The transmitter adds all of the bytes in the message without carries. It then calculates the 2's complement of the result and send that as the last byte.

The receiver adds all of the bytes in the message including the last byte. The result should be 0.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | 182

So, when you are got a when you have got larger blocks of data transmission. So, for example, you are transmitting data from disk to the computer or to the memory then there can be even the more number of, if you are relying on this parity and all that then after transmitting every bit every per character you have to do this parity bit. So, that may be over it may be high. So, instead of that, after transmitting one block of data. So, we add the check bits that is the checksum. So, transmitter adds all the bytes in the

message without carries and then calculate the 2's complement of the result and same that as the last byte. So, this is one possible way of calculating checksum.

The other one I was telling is by means of some polynomial. So, the entire message is represented as a and then it is computing some remainder that polynomial is divided by a fixed polynomial and it will be the remainder is put as the checksum. So, that is other policy. So, this way whatever be the policy, ultimately it comes up with a signature and that signature or the remainder whatever we call it so that will be put at the end of the message and the receiver will.

So, that way and this signature will be checked at the receiving end. So, the for the policy where all the bytes were added like the policy that is detailed here all the bytes are added and the result we complete compute the 2's complement of the result and same as the last byte. So, what happens is that that we are after taking the sum we are negating the sum and sending it to the receiver.

The receiver after adding all the bytes, it will get the plus of that sum signal whose sign sig sign will be positive and when the last byte will be added the result should become 0 the result does not become 0; that means, there is some error. So, that is why it is called checksum type of things. So, it is putting the checksum value at the end of the bytes that are transmitted.

(Refer Slide Time: 18:24)

**RS 232**

A communication standard for connecting computers to printers, modems, etc.

- The most common communication standard.
- Defined in the 1950's.
- It uses voltages between +15 and -15 V.
- Restricted to speeds less than 20 K baud.
- Restricted to distances of less than 50 feet (15 m).

The original standard uses 25 wires to connect the two devices. **However, in reality only three of these wires are needed.**

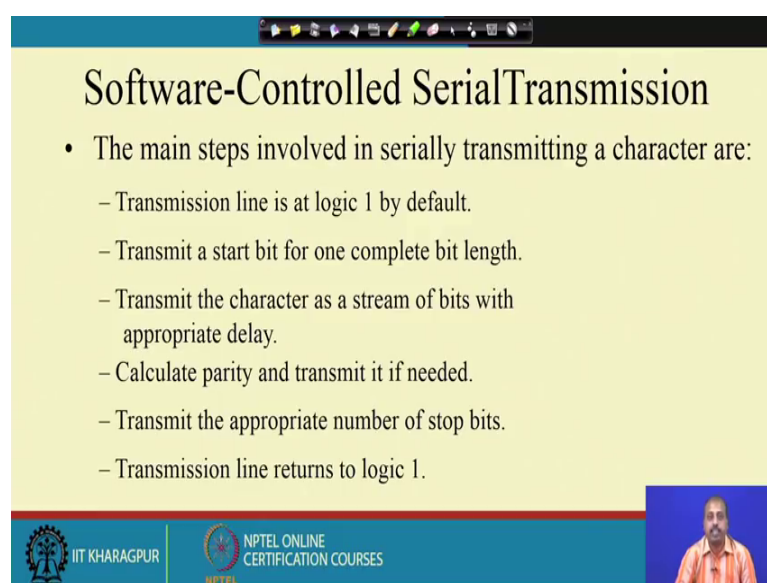
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

A very standard communication protocol that is used for quite a long time since 1960s onwards is the RS 232 type of protocol and there are many variants of it say we have got 422, 489 like that.

So, this is a communication standard for connecting computers to printers modems etcetera and it is defined in 1950s. So, it uses to the voltage levels plus 15 volt and minus 15 volts, so minus 15 volt is taken as logic high and plus 15 volt is taken as logic low. So, you see that when we are talking about 8085, it cannot have minus 15 and plus 15 voltage output. So, we need some other power supplies for giving these some values. So, that is a different issue, but it is the standard takes it like that. In fact, this is a range, so this minus 15 is actually minus 4 to minus 15 and plus 15 is plus 4 to plus 15.

So, this speed is less than 20 kilo baud rate and restricted to the distance also less than 50 feet of 15 meter. The original standard uses 25 wires to connect to devices. So, in, but many of those wires are not connected in the communication because. So, these 25 wires, this interface was very would say very robust type of connection and many times we do not need that much rugged connection. So, with their devices that two computers are located close to each other then we do not need soap rigorous connection between them. So, only fuel of wires may be sufficient. So, it may it may be brought down to 3 wires in the best case.

(Refer Slide Time: 20:17)



### Software-Controlled Serial Transmission

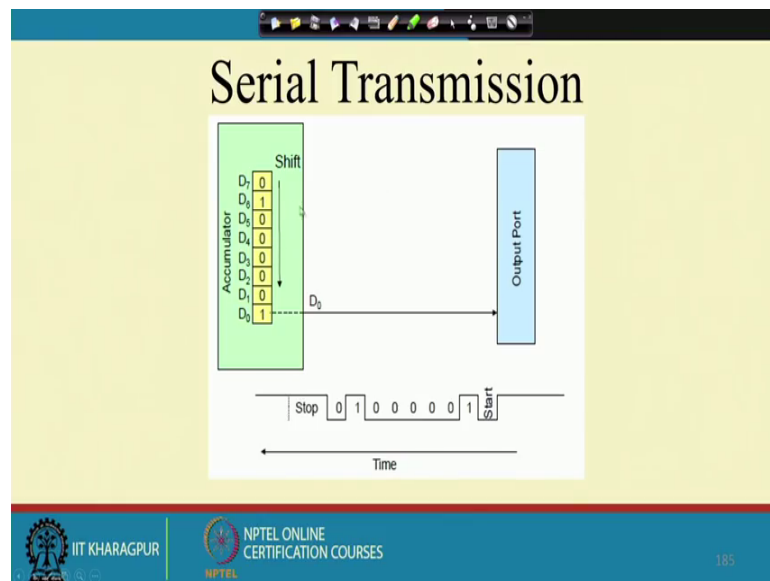
- The main steps involved in serially transmitting a character are:
  - Transmission line is at logic 1 by default.
  - Transmit a start bit for one complete bit length.
  - Transmit the character as a stream of bits with appropriate delay.
  - Calculate parity and transmit it if needed.
  - Transmit the appropriate number of stop bits.
  - Transmission line returns to logic 1.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So what about the software side of this serial transmission? So, software for it is doing it like this. So, if the transmission line is at logic 1 by default. So, as I said that the line is logic high then it has to transmit a start bit for one complete bit length. So, bit length is decided by determining that bit timing. So, from that we can send a start a bit for that time and then we can transmit the character as a stream of bits with appropriate delay.

So, after transmitting every bit we have to wait for the bit time before changing it to the next bit. After that we should calculate the parity and transmit it if needed and transmit the appropriate number of stop bits. So, this again, these are all programmable like how many stop bits then what is the baud rate. So, all these are programmable. So, if these things are agreed upon to between the device and the processor. So, based on that it can do the transmission and then the transmission line will return to logic high. So, this is the, if you are developing the software for doing transmission. So, these are the steps to be followed.

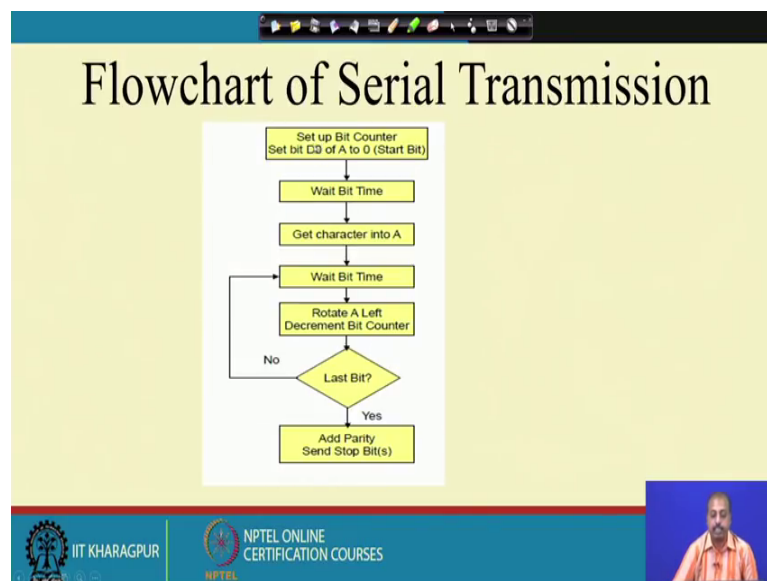
(Refer Slide Time: 21:29)



So, this is the protocol this is the diagram the schematic diagram that you can see that in case of 8085 in the accumulator registers. So, we can put the value on the D 0 line the bit that we are trying to try to transmit. So, this is the start bit that we are transmitting suppose. So, start bit should be 1, sorry start bit should be 0. So, we are and this accumulator has got the 8 bit pattern that we want to transmit.

So, after sending the start bit, we will be sending the first bit. So, this D 0 is put onto the line and that is going to the after that there is a shift. So, this shift will take this 0 to this and this one will go out. So, this 0 will be transmitted next then this way all these bits are transmitted and when this 1 comes. So, this one comes then it will be coming like this then after that this again the 7 bits or 7 bits is D 0 to D 6. So, that is done and then the again another start bit may be sent and this communication will take place. So, that way it will be taking place.

(Refer Slide Time: 22:40)



So, the flowchart is like this. That set up the bit counter set bit D 0 of A to 0. So, for the start bit part then we wait for the bit time. So, this is the, that will help us in transmitting the start bit, then we get the character into the accumulator then wait for the bit into a wait for a bit time. So, then we wrote. So, this is in a loop now. So of course, you can say that you are waiting for two bit times here, but this may be brought down also.

Rotate a left and decrement the bit counter. So, we rotate left. So, this next bit will be coming to the D 0 pin and then will be transmitting that bit there and, it way this way it will go on then that. So, this way it will go on and till all the bits are transmitted, when you come to the last bit. So, it is also last bit has been transmitted. So, you add the parity bit and the parity bit will be transmitted and then we will send the stop bits for ending the transmission of the character.

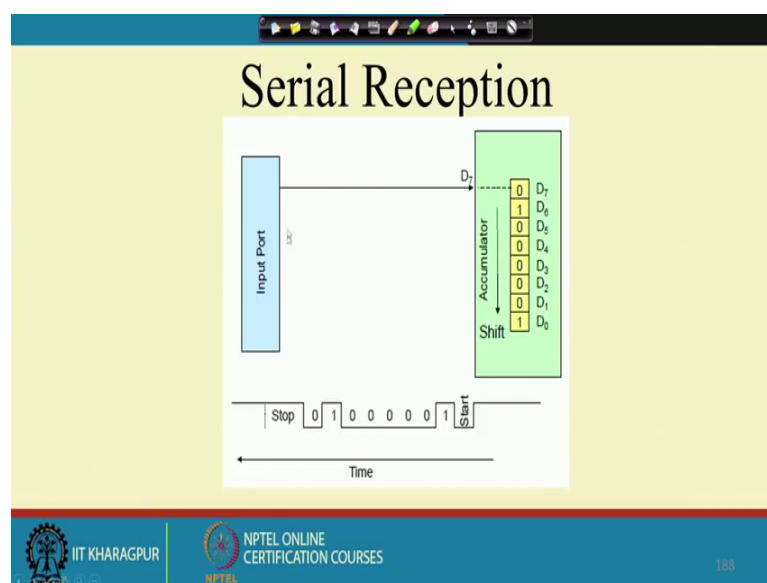
So, for the reception part, for the receiver part, it will wait for a low to appear on the transmission line because receiver it will see that line to be active to be high when there is no communication and when a transmission is going to start then this line is brought low. So, that is that will indicate a start bit. Then it will read the value of the line over next a 8 bit lengths. So, next 8 bit, it will get over the every bit time it will sample the line. So, it normally samples the line at the middle of the bit time to get the value that is available on the serial on the data line.

So, that all the 8 bits are received then it will calculate the parity and compared it to a bit 8 of the characters about to bit 7 is the ASCII code and bit 8 is the parity. So, it will compute the parity and check with the bit 8. So, if they are matching then it is fine and if it is not so, if it is not. So, then it will check that it will tell that the character received is not correct that is error. So, some higher level activity has to take place.

And then it will wait for the appropriate number of stop bits. So, we will check that two stop bits if it is agreed upon then to our if it is agreed upon that only one stop bit will be sent. So, it will wait for the appropriate number of stop bits and then that will end the transmission of one character.

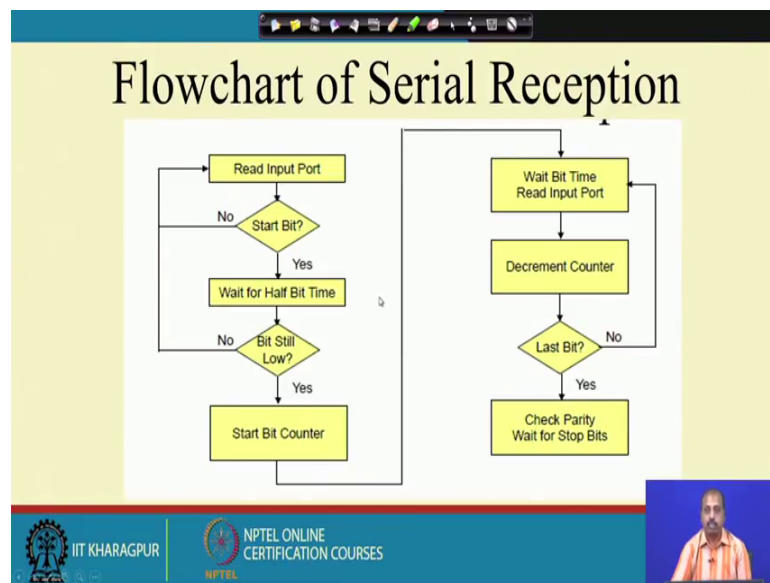
Again for the next character another start bit it will wait for the start bit to come and it will go like this.

(Refer Slide Time: 25:19)



So, serial reception will be like this that first this bit on this bit D 7. So, it will be getting this 0. So, this 0 will be shifted and then the accumulator content will get shifted this way. So, ultimately the pattern, the bit stream that we have sent, so 0 1 0 0 0 1, so this bit pattern will be appearing on to the accumulator. So, first it gets the start bit, then it will get this, this is the first bit D 1 D 0 that is received it comes here finally. So, initially the 1 came and then ultimately that 1 got shifted. So, this 1 is coming here.

(Refer Slide Time: 26:04)



So, this is the serial reception is like this. So, it should read the input port for start bit the start bit is not there then it will wait in the loop, if it is yes then it will wait for half bit time and the bit is still low then it will wait for it will go it will check when the start bit goes high. When the start bit under style, just if it is not low yet that means that was a flicker, so that was a flicker. So, that is not accepted. And if the bit is still low that means, it is really a start symbol start bit so that start bit. So, start bit counter is set and then this it will go to the wait for bit time, read the input port, decrement counter and till this process will go on till the last bit has been received, it check for check parity for the bits that are received and then that will be the end of the transmission.