**UART**(**Universal Asynchronous Transmitter Receiver**), this is the most common protocol used for full-duplex [serial communication](https://www.codrey.com/embedded-systems/serial-communication-basics/). It is a single [LSI](https://www.techopedia.com/definition/2368/large-scale-integration-lsi) (large scale integration) chip designed to perform asynchronous communication. This device sends and receives data from one system to another system.

In this tutorial, you will learn the basics of UART communication, and the working of the UART.

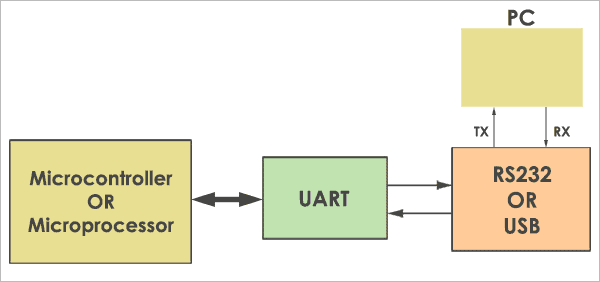
**Contents**[[show](https://www.codrey.com/embedded-systems/uart-serial-communication-rs232/)]

**What is UART?**

“UART” stands for Universal Asynchronous receiver-transmitter. It is a hardware peripheral that is present inside a microcontroller. The **function of UART** is to convert the incoming and outgoing data into the serial binary stream. An 8-bit serial data received from the peripheral device is converted into the parallel form using serial to parallel conversion and parallel data received from the CPU is converted using serial to parallel conversion. This data is present in modulating form and transmits at a defined baud rate.

**Why UART is used?**

Protocols like SPI (serial peripheral interface) and USB (Universal Serial Bus) are used for fast communication. When the high-speed data transfer is not required UART is used. It is a cheap communication device with a single transmitter/receiver. It requires a single wire for transmitting the data and another wire for receiving.

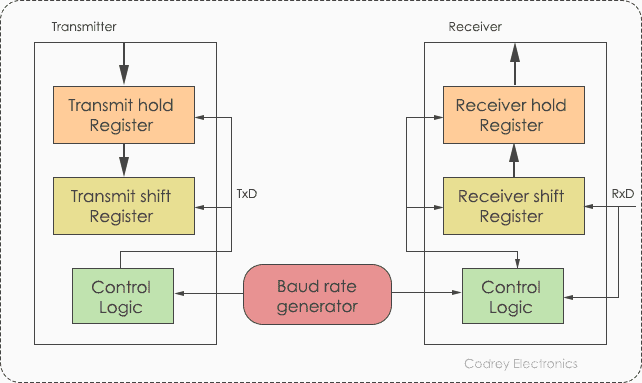


*UART Interface*

It can be interfaced with a PC (personal computer) using an RS232-TTL converter or USB-TTL converter. The common thing between [RS232](https://www.codrey.com/embedded-systems/rs232-serial-communication/) and UART is they both don’t require a clock to transmit and receive data. The Uart frame consists of 1 start bit, 1 or 2 stop bits and a parity bit for serial data transfer.

**Block Diagram**

The UART consists of the following core components. They are the transmitter and receiver. The transmitter consists of the Transmit hold register, Transmit shift register, and control logic. Similarly, the receiver consists of a Receive hold register, Receiver shift register, and control logic. In common, both the transmitter and receiver are provided with a baud rate generator.

[](https://www.codrey.com/wp-content/uploads/2017/10/UART-Block-Diagram.png)

*UART Block Diagram*

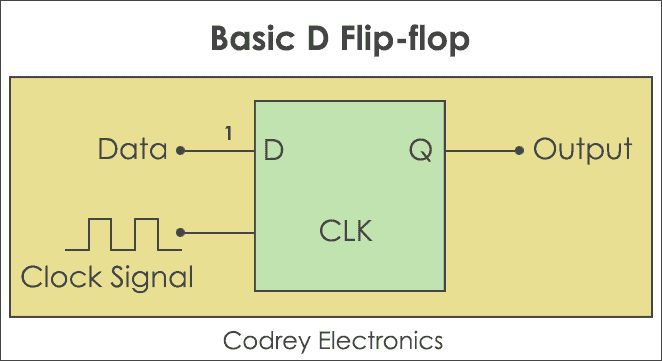
The baud rate generator generates the speed at which the transmitter and receiver have to send/receive the data. The Transmit hold register contains the data byte to be transmitted. The transmit shift register and receiver shift register shift the bits to the left or right until a byte of data is sent/received.

In addition to these, a read or write control logic is provided to tell when to read/write. The baud rate generator generates speeds ranging from 110 bps (bits per second) to 230400. Mostly, microcontrollers come up with higher baud rates such as 115200 and 57600 for faster data transfer. Devices like [GPS](https://www.codrey.com/electronics/gps-working-applications/) and GSM use slower baud rates in 4800 and 9600.

**How UART works?**

To know the working of UART, you need to understand the basic functionality of [serial communication](https://www.codrey.com/embedded-systems/serial-communication-basics). In short, transmitter and receiver use start bit, stop bit, and timing parameters to synchronize with each other. The original data is in parallel form. For example, we have 4-bit data, to convert it into the serial form, we need a parallel to serial converter. Generally, D flip-flops or latches are used to design the converters.

**Working of D – Flip-flop**

[](https://www.codrey.com/wp-content/uploads/2017/10/Basic-D-Flip-Flop.png)

*Basic D Flip-Flop*

**D flip-flop** also known as Data flip-flop shifts one bit from the input side to the output side if and only, when the clock changes the transition from a high state to a low state or low state to a high state. Likewise, if you want to transfer four bits of data you need 4 flip-flops.

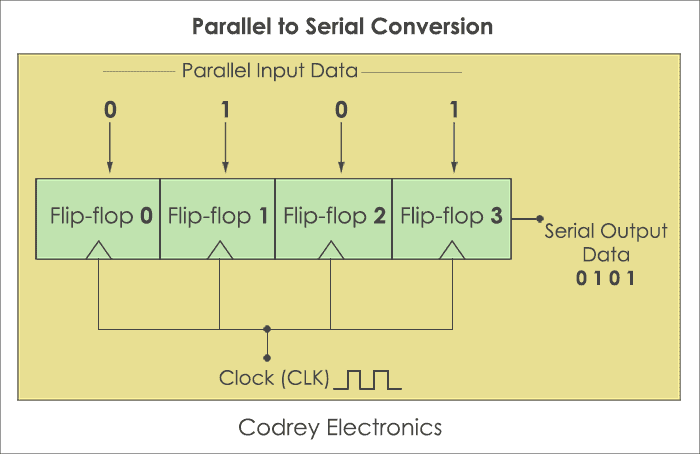
**Note:**Here,

‘**D**‘ represents input data.

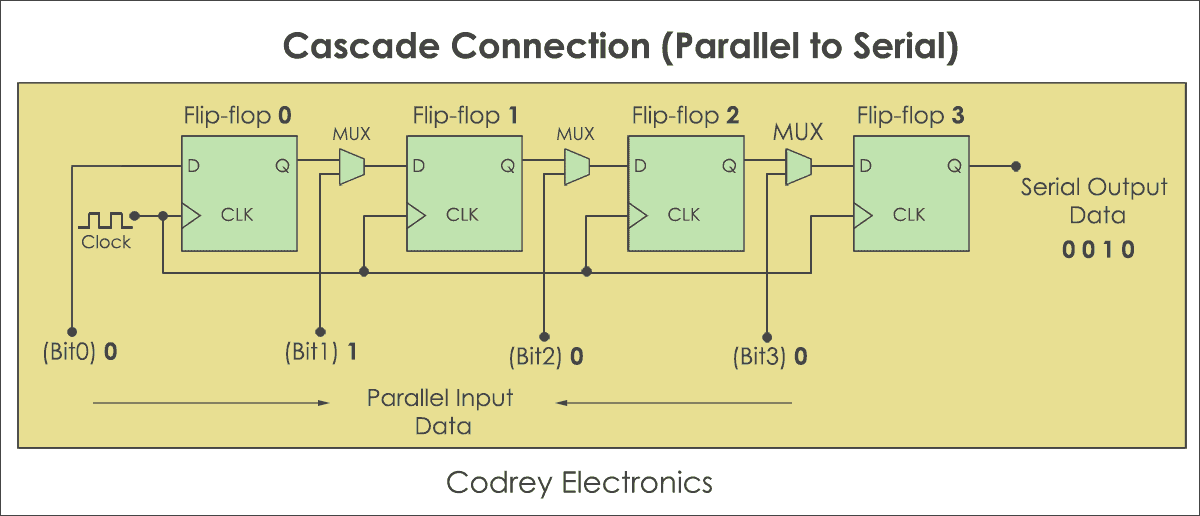
‘**CLK**‘ indicates clock pulses.

‘**Q**‘ denotes output data. Now, let’s design a parallel to serial and serial to parallel converter.

**Parallel to Serial Conversion**

[](https://www.codrey.com/wp-content/uploads/2017/10/Parallel-to-Serial-Conversion.png)

*Parallel to Serial Conversion*

[](https://www.codrey.com/wp-content/uploads/2017/10/Cascade-Connection-Parallel-to-Serial.png)

*Cascade Connection – Parallel to Serial*

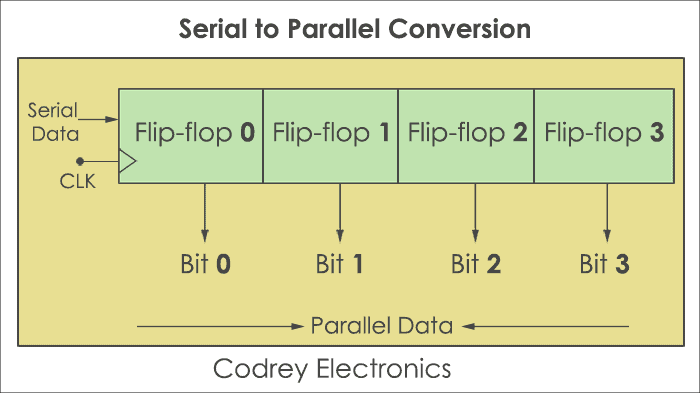
**Step#1:**

Take 4 Flip-flops. The number of flip-flops is equivalent to the number of bits to be transmitted. Similarly, put Multiplexers in front of each flip-flop, but excluding the first one. A multiplexer is placed to combine the data and convert it to serial bits. It has two inputs, one parallel bit data and another from the previous flip-flop.

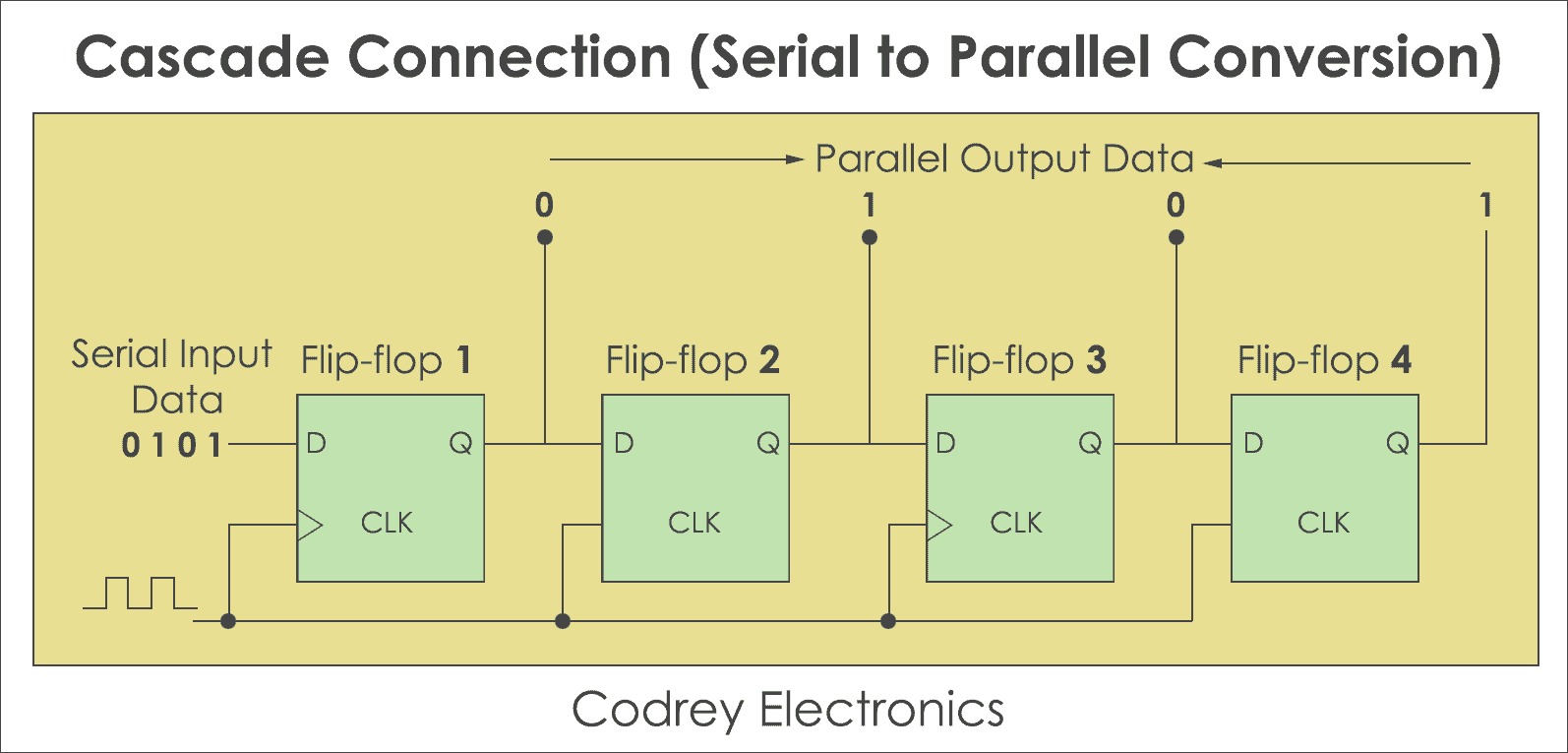
**Step#2:**

Now, Load the data at a time in the D flip-flops. It will pull the parallel data and moves the last bit of last flip-flop (four), and then the third bit, second bit, and finally the first bit. Now, to reconvert the parallel data into serial form serial to parallel converter is used.

**Serial to Parallel Conversion**

[](https://www.codrey.com/wp-content/uploads/2017/10/Serial-to-Parallel-Conversion.png)

*Serial to Parallel Conversion*

[](https://www.codrey.com/wp-content/uploads/2017/10/Cascade-Connection-Serial-to-Parallel.png)

*Cascade Connection – Serial to Parallel Conversion*

**Step#1:**

Take 4 Flip-flops. The number of flip-flops is the same as the number of bits to be transmitted.

**Step#2:**

Initially, disable the parallel bus. Don’t enable until all bits are loaded. Store the data at the input of the first flip-flop. Now make clock high, this will shift the least significant bit to the input of the second flip-flop and the output of the first one. Similarly, shift all the bits one by one by making the clock pulse high. The converter is in the hold state until all bits are transferred to the output.

**Step#3:**

Now each flip-flop contains one bit of serial data. Erstwhile all bits are transferred to the flip-flop output, enable the bus. This will make the converter to send all the bits at a time.

**Protocol Format**

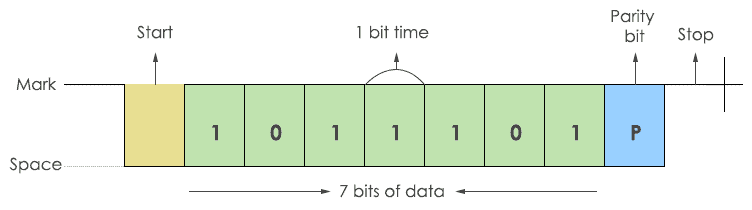
The UART starts the communication with a start bit ‘0’. The start bit initiates the transfer of serial data and stop bit ends the data transaction.

*Protocol format*

It is also provided with a parity bit (even or odd). Even parity bit is represented by ‘0’ (even number of 1’s) and the odd parity bit is represented by ‘1’ (odd number of 1’s).

**Transmission**

The transmission of data is done using a single transmission line (TxD). Here ‘0’ is considered as space and ‘1’ is known as mark state.

[](https://www.codrey.com/wp-content/uploads/2017/10/UART-Transmission-frame.png)

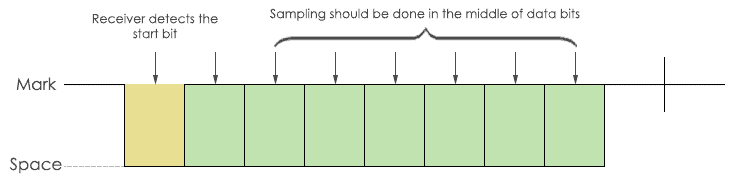
*Transmission Frame*

The transmitter sends a single bit at a time. After sending one bit, the next bit is sent. In this way, all the data bits are sent to the receiver with a predefined baud rate. There will be a certain delay in transmitting each bit. For example, to send one byte of data at 9600 baud rate, each bit is sent at 108 µsec delay. The data is added with a parity bit. So, 10 bits of data are required to send 7 bits of data.

**Note**: In transmission, always LSB (Least Significant Bit) is sent first.

**Reception**

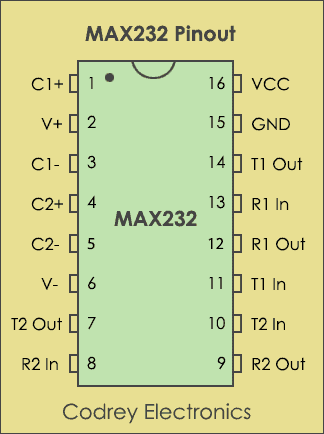
During the reception, the RxD line (Receiver) is used for receiving the data.

[](https://www.codrey.com/wp-content/uploads/2017/10/UART-Receive-Frame.png)

*Receive Frame*

**Example of UART interfacing**

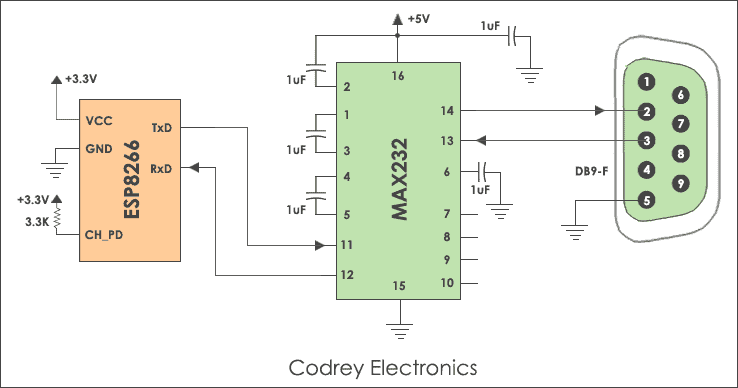
This example demonstrates the interfacing of ESP8266 UART with MAX232. But, before I jump into the details of interfacing, let me share the pin details of the Max232 driver.

[](https://www.codrey.com/wp-content/uploads/2017/10/MAX232-Pinout.png)

*MAX232 Pinout*

MAX232 IC is powered up by 5V supply which includes a capacitive voltage generator to drive 232 level voltages. It comes with dual transmitters also called the driver (TIN, TOUT) and receivers (RIN and ROUT).

Here, I have used ESP8266 (32-bit microcontroller) which has inbuilt UART. The communication can be done with ESP8266 using AT commands via [RS232](https://www.codrey.com/embedded-systems/rs232-serial-communication/) to TTL level converter (MAX232). The below figure shows the connection of ESP8266 with PC (personal computer).

[](https://www.codrey.com/wp-content/uploads/2017/10/ESP8266-interfacing-with-UART.png)

*ESP8266 interfacing with UART*

By requesting valid AT commands through the PC the Wi-Fi chip will respond with an acknowledgement. I don’t want to go in-depth about ESP8266 and it will be explained in the future tutorials.

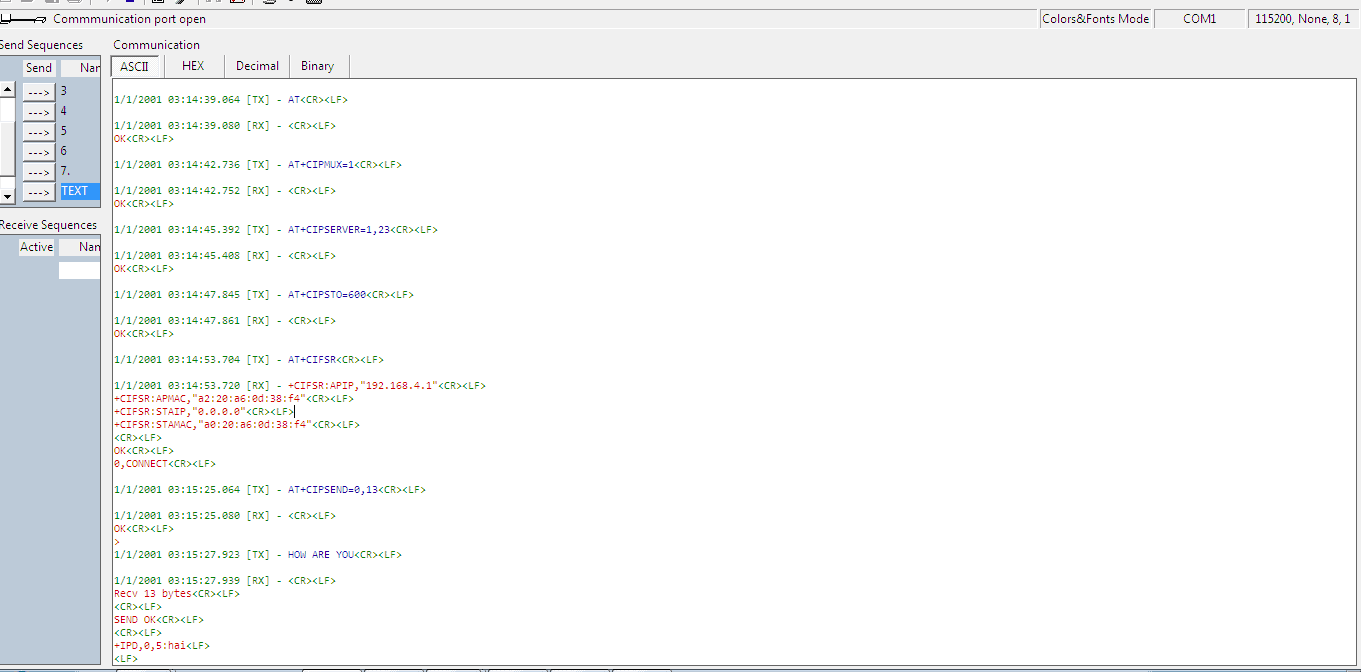
Here are the steps to implement [serial communication](https://www.codrey.com/embedded-systems/serial-communication-basics/) with PC.

1. Connect Transmitter (TX) of ESP8266 to Receiver (TX) of RS232 to TTL level converter (MAX232) and RX of PC.
2. Connect Receiver (RX) of ESP8266 to TX of PC and RX of TTL converter.

**ESP8266 Commands**

| **AT command (Sent from PC)** | **ESP8266 Response (received by PC)** |
| --- | --- |
| AT<CR><LF> | OK<CR><LF> |
| AT+CIPMUX=1 | OK<CR><LF> |
| AT+CIPSERVER=1,23<CR><LF> | OK<CR><LF> |

The below screenshot shows the response given by the ESP8266 module.

[](https://www.codrey.com/wp-content/uploads/2017/10/ESP8266-Commands-Docklight.png)

*ESP8266 Commands – Docklight Program*

**UART vs USART**

**USART** is the basic form of UART. Technically, they are not the same. But, the definition is the same for both of them. These are [microcontroller](https://www.codrey.com/microcontroller/microcontroller-basics/) peripherals that convert parallel data into serial bits and vice versa.

The main difference between UART and USART is, UART supports only asynchronous communication, whereas USART support synchronous as well as asynchronous communication. For easy understanding, here is the comparison between USART and UART.

| **UART** | **USART** |
| --- | --- |
| The Clock is generated internally by the microcontroller. | The sending device will generate the clock. |
| The data rate is slow. | The data rate is higher due to external clock. |
| Standalone protocol | Supports multiple protocols like LIN, RS-485, IrDA, Smart Card etc. |
| The baud rate should be known before transmission. | No need to know the baud rate earlier. |
| Suitable for low speed communications | Suitable for high speed communications. |
| Reduced energy footprint. | Handles serial communication during high energy consumption |

**RS232 and UART**

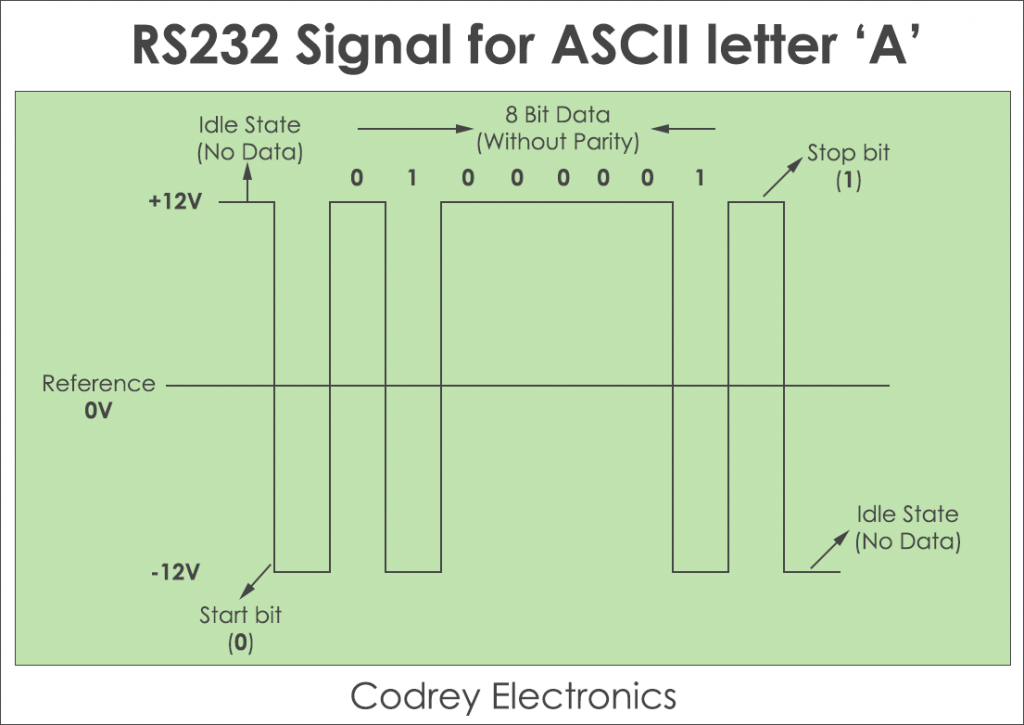
**Logic levels** represent the operating voltage levels that a device can withstand to operate in a safe zone. Here are the voltage levels for [RS232](https://www.codrey.com/embedded-systems/rs232-serial-communication/) and TTL.

**RS232 Logic:**

**RS232 Voltage Levels**

| **Logic Level** | **Voltage Range** |
| --- | --- |
| Logic High or OFF output | -5V to -15V |
| Logic Low or ON output | +5V to +15V |
| Logic High or OFF input | -3V to -15V |
| Logic Low or ON input | +3V to +15V |

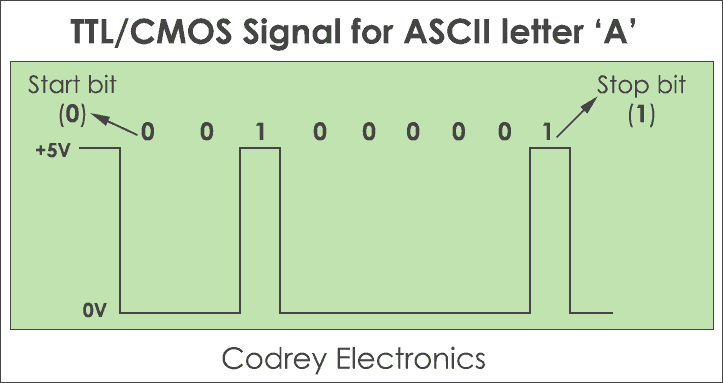
In most cases [RS232](https://www.codrey.com/embedded-systems/rs232-serial-communication/) levels range from -12V to +12V. For instance, an ASCII value for a character ‘**A**‘ in RS232 is 65 and 41 in Hexadecimal. So in an 8-bit binary format, it is 0100 0001. Here shows the representation of RS232 logic levels for ASCII ‘**A**‘.



*RS232 Signal for ASCII Letter A*

**TTL/CMOS Logic:**

The UART works on TTL logic.



*TTL CMOS Signal for ASCI Letter A*

* Initially, the serial line is in the idle state commonly called a **Mark state** (Logic 1). Now, the data transmission begins with a start bit (logic 0).
* Further, eight data bits are sent over the serial line one after another with LSB (Least significant bit) first.
* After the entire transmission is over, a stop bit (logic 1) is encountered.

**Advantages**

The advantage of UART is, it supports full-duplex communication using two wires. Also, it requires no external clock for data communication. It supports error checking using a parity bit and the length of the data can be changed easily.

**Disadvantages**

The major disadvantage of UART is, it doesn’t support multi-slave or multi-master configuration. And, the size of the data packet is limited to 9 bits. The UART is not suitable for heavy lifting serial communication during high energy consumption.

**Applications**

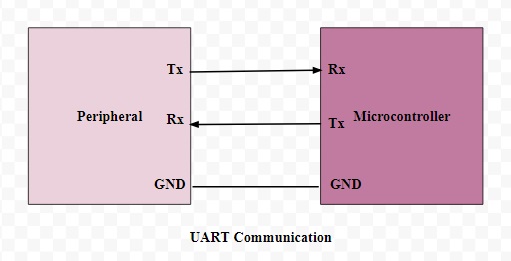
1. **Serial debug port** uses the UART driver to print out the data coming from the external world.
2. We can use it to send and receive commands to and from the embedded devices.
3. Communication in [GPS](https://www.codrey.com/electronics/gps-working-applications/), GSM/GPRS modem, Wi-Fi chips, etc operates with UART.
4. Used in Mainframe access to connect different computers.

# UART Communication: Block Diagram and Its Applications

If we remember the old computer parts like printer, mouse, the keyboard is associated with the help of connectors. The communication process between the computer and these parts could be done using the UART. Universal Serial Bus (USB) has changed all kinds of communication principles on computers. But, UART is still used in the above-declared applications. Approximately all [types of microcontroller](https://www.elprocus.com/different-microcontrollers-used-in-automobiles/) architectures have built-in UART hardware due to serial communication and use only two cables for communication. This article discusses what UART,**How UART Works,** the difference between serial and parallel communication, **UART block diagram**, UART communication, UART interfacing, Applications, Advantages, and Disadvantages.

## ****What is UART?****

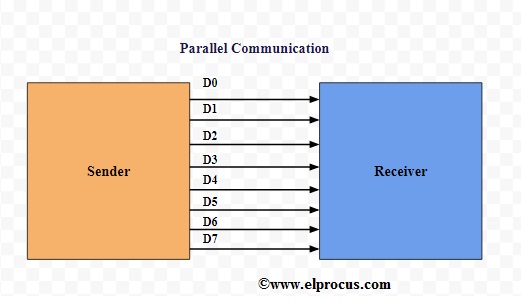
The**UART full form** is “Universal Asynchronous Receiver/Transmitter”, and it is an inbuilt IC within a microcontroller but not like a communication protocol (I2C & SPI). The main function of UART is to serial data communication. In UART, the communication between two devices can be done in two ways namely serial data communication and parallel data communication.

**UART**

## ****Serial and Parallel Communication****

In serial data communication, the data can be transferred through a single cable or line in a bit- by- bit form and it requires just two cables. Serial data communication is not expensive when we compared with parallel communication. It requires very less circuitry as well as wires. Thus, this communication is very useful in compound circuits compared with parallel communication.

In parallel data communication, the data can be transferred through multiple cables at once. Parallel data communication is expensive as well as very fast, as its requires additional hardware and cables. The best examples for this communication are old printers, PCI, RAM, etc.

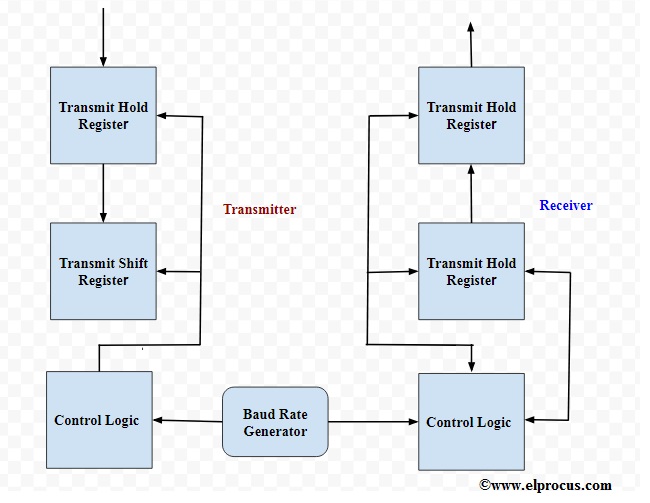
**Parallel Communication**

### **UART Block Diagram**

The UART block diagram consists of two components namely the transmitter & receiver that is shown below.  The transmitter section includes three blocks namely transmit hold register, shift register and also control logic. Likewise, the receiver section includes a receive hold register, shift register, and control logic. These two sections are commonly provided by a baud-rate-generator. This generator is used for generating the speed when the transmitter section & receiver section has to transmit or receive the data.

The hold register in the transmitter comprises the data-byte to be transmitted. The shift registers in transmitter and receiver move the bits to the right or left till a byte of data is transmitted or received. A read (or) write control logic is used for telling when to read or write.

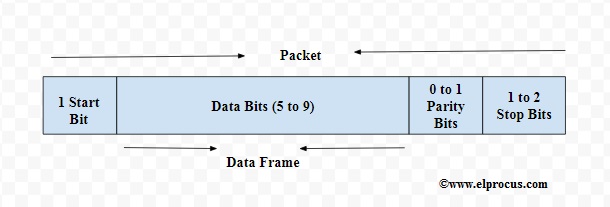
The baud-rate-generator among the transmitter and the receiver generates the speed that ranges from 110 bps to 230400 bps. Typically, the baud rates of microcontrollers are 9600 to 115200.

**UART Block Diagram**

## ****UART Communication****

In this communication, there are two types of UARTs available namely transmitting UART and receiving UART, and the communication between these two can be done directly by each other.  For this, simply two cables are required to communicate between two UARTs. The flow of data will be from both the transmitting (Tx) & receiving (Rx) pins of the UARTs. In UART, the data transmission from Tx UART to Rx UART can be done asynchronously (there is no CLK signal for synchronizing the o/p bits).

The data transmission of a UART can be done by using a data bus in the form of parallel by other devices like a microcontroller, memory, CPU, etc. After receiving the parallel data from the bus, it forms a data packet by adding three bits like start, stop and parity. It reads the data packet bit by bit and converts the received data into the parallel form to eliminate the three bits of the data packet. In conclusion, the data packet received by the UART transfers in parallel toward the data bus at the receiving end.

UART Communication

**Start Bit**

Start-bit is also known as a synchronization bit that is placed before the actual data. Generally, an inactive data transmission line is controlled at a high-voltage level. In order to begin the data transmission, the UART transmission drags the data-line from a high voltage level (1) to a low voltage level (0). The obtaining UART notices this transform from the high level to low level over the data line as well as starts understanding the real data. Generally, there is just a single start bit.

**Stop Bit**

The Stop Bit is placed at the ending of the data packet. Usually, this bit is 2-bits lengthy but frequently on bit only utilized. In order to stop the broadcast, the[UART](https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter) keeps the data-line on high voltage.

**Parity Bit**

Parity bit lets the receiver to ensure whether the collected data is right or not. It is a low-level fault checking system & parity bit is available in two ranges such as Even Parity as well as Odd Parity.  Actually, this bit is not widely used so it is not compulsory.

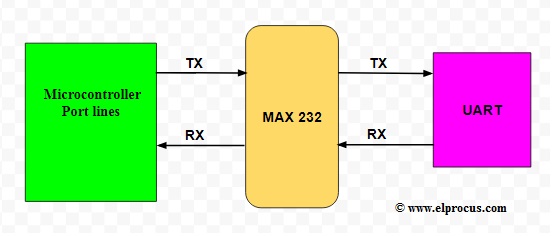
**Data Bits or Data Frame**

The data bits include the real data being conveyed from the sender to receiver. The data frame length could be between 5 & 8. If the parity bit is not used when the data frame length could be 9-bit long. Generally, the LSB of the data to be transmitted first then it is very useful for transmitting.

### **UART Interfacing**

The following figure shows UART interfacing with [a microcontroller](https://www.elprocus.com/microcontrollers-types-and-applications/). The UART communication can be done using three signals like TXD, RXD, and GND.

By using this, we can exhibit a text in personal computer from 8051 microcontroller board as well as the UART module. In 8051 board, there are two serial interfaces such as UART0 and UART1. Here, UART0 interfacing is used. The Tx pin transmits the information to PC & Rx pin receives the information from PC. Baud rate can be used to denote the speeds of both the microcontroller and PC. The data transmission and reception can be done properly when the baud rates of both microcontroller & PC are similar.

**UART Interfacing**

### **Applications of UART**

UART is normally used in microcontrollers for exact requirements, and these are also available in various communication devices like [wireless communication](https://www.elprocus.com/?s=wireless+communication),  GPS units, [Bluetooth module](https://www.elprocus.com/how-does-bluetooth-work/), and many other applications.

The communication standards like RS422 & TIA are used in UART except for RS232. Usually, a UART is a separate IC used in **UART serial communications.**

### **Advantages and Disadvantages of UART**

The pros and cons of UART include the following

* It requires only two wires for data communication
* CLK signal is not required.
* It includes a parity bit for allowing to check the errors
* The data packet arrangement can be modified because both surfaces are arranged for it
* The data frame size is a max of 9 bits
* It doesn’t hold several slave (or) master systems
* The every UART baud rate should be in 10% of each other

Thus, this is all about an overview of **Universal Asynchronous Receiver Transmitter** (UART) is one of the fundamental interfaces which gives a simple, cost-effective & consistent communication among microcontroller as well as PC. Here is a question for you what are **UART pins**?