UART

Introduction:

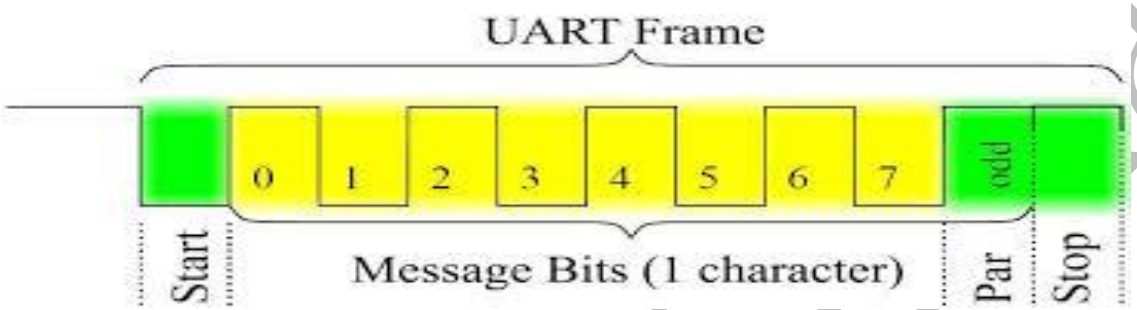
UART is a protocol that can be applied to any transmitter and receiver for data communication.

Universal - The protocol can work with any compatible transmitter and receiver.

Asynchronous - No clock signal is used for data communication. Instead, the devices agree on a data rate in advance and synchronize only during the start and stop bits of each data frame.

When two devices communicate using UART, they need at least two connections between them: one for transmission (TX) and one for reception (RX).

UART is a character-oriented protocol, meaning data is sent byte by byte. Each communication uses a specific bit frame structure, including start bits, data bits, optional parity bits, and stop bits. This bit frame structure allows consistent data exchange between devices.



- In UART, data length and speed can be configured based on the required operation, allowing for 5, 6, 7, or 8 data bits.
- The start bit is an active low signal used to reset and initiate data transmission.
- The parity bit, which is optional, can be set to even, odd, or none for error checking.
- This Parity bit is optional for some devices need this some don't it is used only if the devices need to check the error present in the data stream.
- The stop bit is used to signify the end of the current data transmission, ensuring there is a brief pause before the next data byte is received.

Baud Rate Generator:

Both Transmitter and Receiver Configure same baud-rate for proper hand shaking

According to baud rate we are configure the clock or speed of UART

Baud Rate generator determines transmission speed in asynchronous communication

It is number of Symbols per second transferred

Each bit is 1/(baud rate)wide

Baud rate =clockfrq/(16xdivisior).

Some standard baud rate.

- 2400, 9600, 19200, 38400,115200.

Program Approach:

There are two ways to program in UART

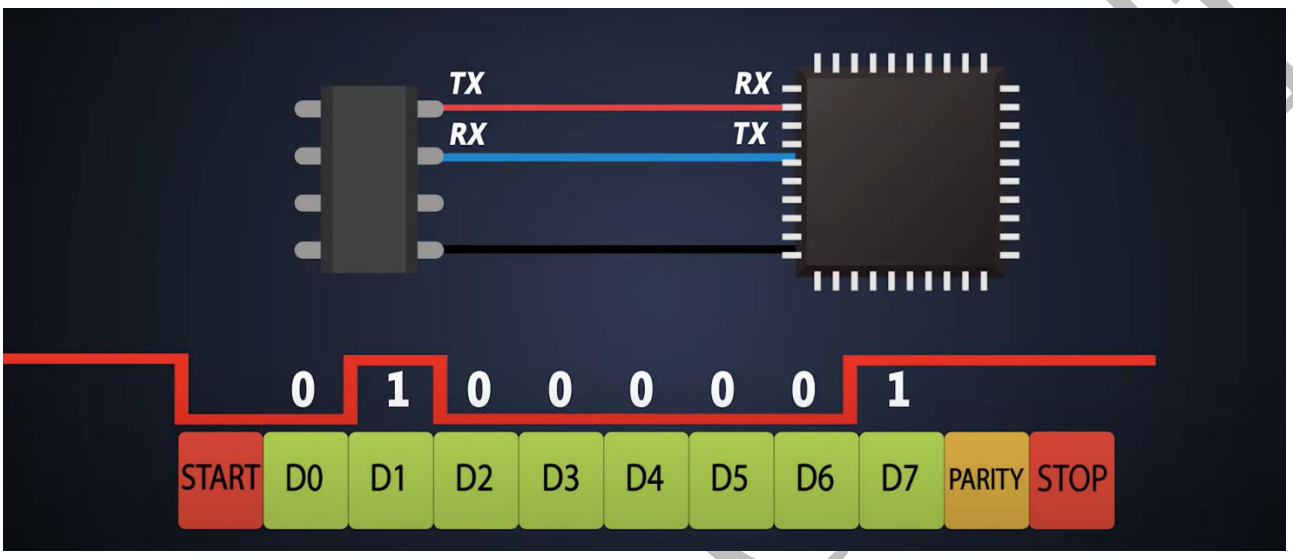
1.Bit Banging

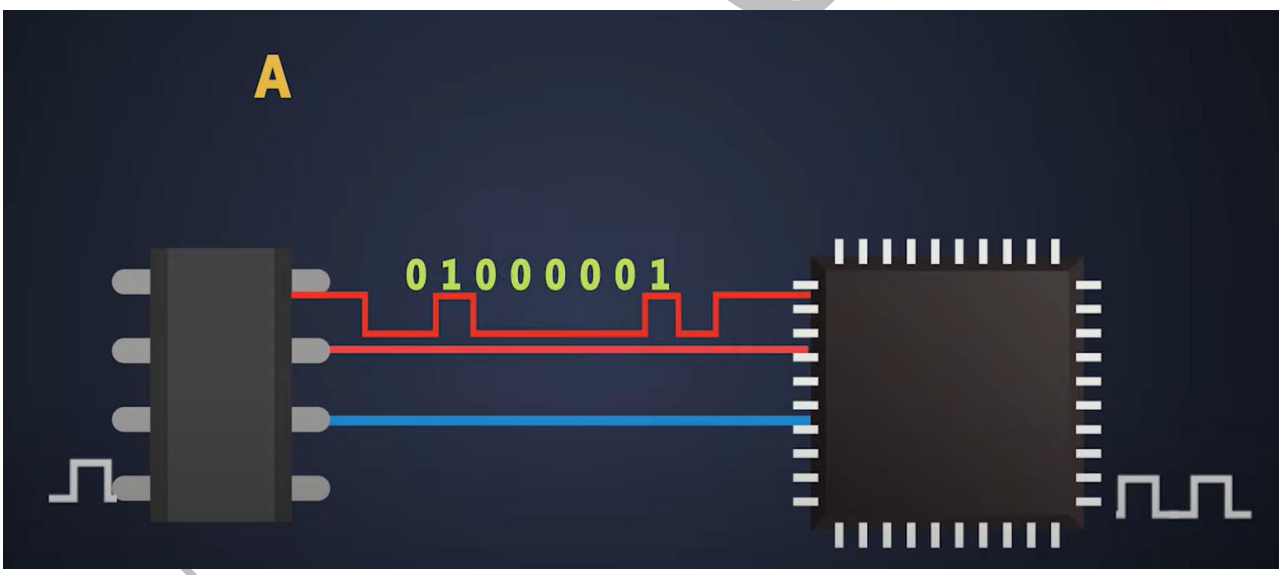
2. UART peripherals

Bit Banging:

Suppose we need to transfer the character **"A"** with an ASCII value of 65, which is represented in binary as 01000001. We will transmit this data using the bit-banging method at a baud rate of 1 byte per second, meaning each bit will be sent in 1-second intervals.
For this communication, we use a standard GPIO pin, starting the transmission with the least significant bit (LSB) first. Initially, the data line is held high. When the transmission begins, a start bit, which is an active low signal, is sent to indicate the start of data transmission.

After the start bit, the actual data bits (01000001) are transmitted sequentially. Finally, a stop bit, which is a high signal, is sent to end the transmission and signify the end of communication between the transmitter and receiver.



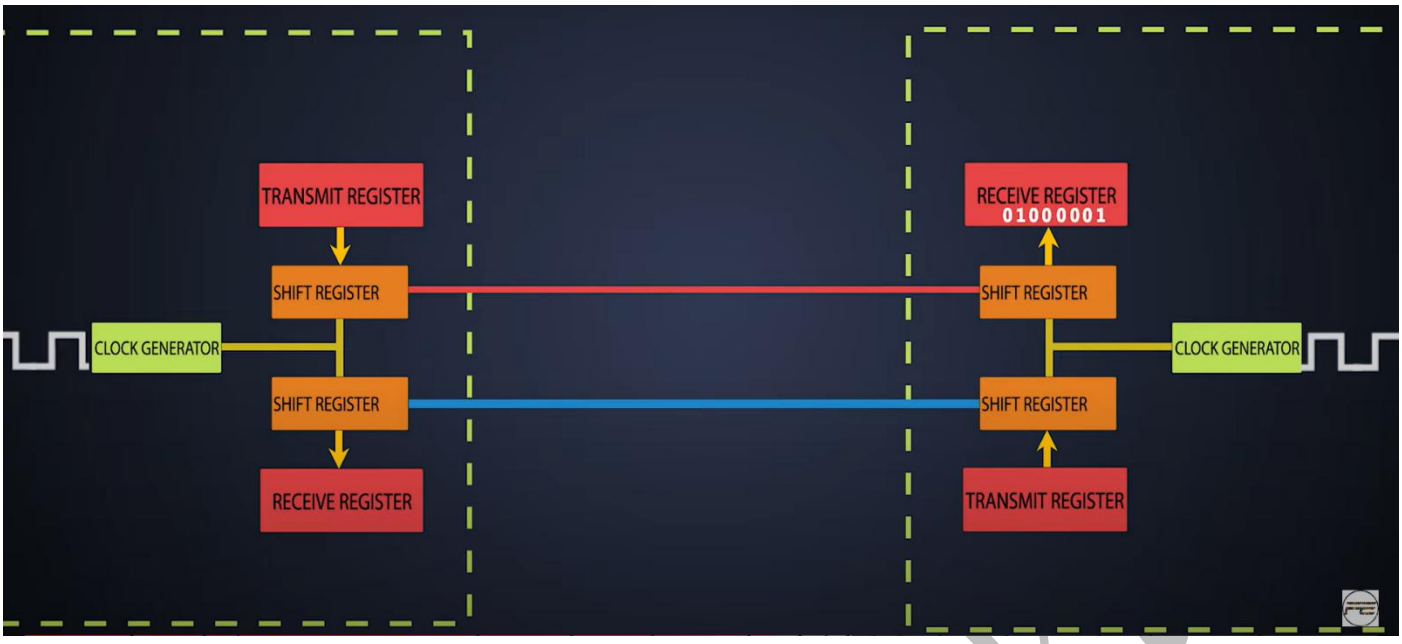


In the bit-banging method, the receiver receives the data only after the complete data transfer is finished.

UART Peripherals:

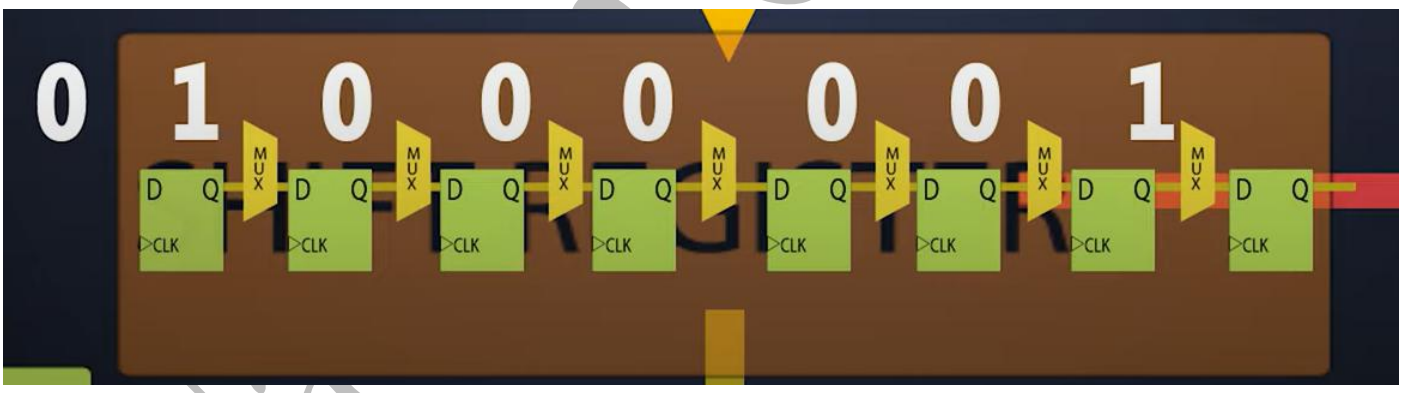
So, most of the controllers provide peripherals for UART, so programmers configure peripherals and make it easy. The two devices that need to talk with each other have particular blocks in each device, which are transmit registers connected to the shift register, and there is one receiver register that is also connected to another shift register. Both of these shift registers are connected

to the clock signal of the device. The same blocks are arranged in the second device as well.



Device-1(Transmitter) **Device-2(Receiver)**

The transmit register sends the entire data stream to a shift register in parallel, which is known as **parallel-in, serial-out (PISO)**. The shift register then transmits this data bit by bit to the next device (receiver), synchronizing with the clock pulse. Meanwhile, at the receiver side, the reverse process, known as **serial-in, parallel-out (SIPO)**, occurs. The shift register at the receiver end collects the data serially, one bit at a time, and then sends the entire data packet in parallel to the receive register.



For example, to send the character “A,” we first load its binary representation into the transmitter register. This data packet is then transferred to the shift register, where it is organized across the bits (D0, D1, D2, ...). The shift register transmits each bit sequentially, in sync with each clock pulse, to the receiving device. At the receiver end, the shift register collects the incoming bits one by one, then transfers the entire data in parallel to the receiver register.

The clock plays a crucial role in this communication, ensuring that the baud rate is properly configured for both devices. With each clock pulse, data is transmitted and received simultaneously, allowing synchronized data transfer.

Configuring for UART:

Step-1: Clock Configuration

The first of the communication is to configure the clock of the UART peripheral according to the baud rate

Step-2: Data Loading

After that the programmer needs to load the data in transmit register

Step-3: Data Transmission

Next step is to turn on the timer do it will start putting the data from the shift register to the transmission line and the other device will get the data according to the sampling

Step-4: Monitor Data

1.Looping Method

In a looping method, when data needs to be sent, it's essential to monitor the structure of the bits to verify whether all 8 bits have been transmitted. Similarly, on the receiver end, it should continuously monitor to ensure that all data has been received completely

2.Interrupt Method

We do not need to monitor the status continuously, as a specific interrupt is generated when data is successfully received or transmitted. This interrupt triggers an interrupt service routine (ISR) to respond accordingly. The interrupt method has advantages over the looping method because it prevents the code from getting stuck in one place, allowing other tasks in the system to be completed in parallel.

Advantages:

1. Easy to interface it needs less hardware
2. Less Hardware only two wires make a job easy
3. Less software complications like adjusting the slaves because only two devices talking to each other

Disadvantages:

1.For synchronization, the baud rate of both devices needs to be common, and that's where limitations come. If one device has a low-capacity baud rate and the other device can support higher baud rates, we still need to go for the low baud rate that the first device supports.

2.UART needs Ad Hoc communication topology, so here there is no third device in this communication. Even after sending the data, we don't get any acknowledgment from the receiver, so it becomes very hard to check whether the data is received by the receiver successfully or not.

Applications:

Simple applications of UART include devices like printers, GPS modules, Bluetooth modules, modems, and embedded systems for serial communication.

C. Naveen