**Note** - In Report2 submission we were working on the project and didn't have enough Idea to comment on each and every section with confidence. Thus our report2 was very short. We apologize for that and after working very much on the project, Here is the report 3.

Description:

What is UART?

A Universal asynchronous receiver-transmitter is a hardware device for asynchronous serial communication in which the data-format and transmission speeds are configurable. A UART is usually an individual or part of an integrated circuit (IC) used for serial communications.

How is transmission and receiving done?

The UART takes bytes of data and transmits the individual bits in a sequential fashion. At the destination(i.e, at receiver), a second UART re-assembles the bits into complete bytes. Each UART contains a shift register, which is the fundamental method of conversion between serial and parallel forms.

UART that is going to transmit data receives the data from a data bus. This bus is used to send data to UART by another device, which we will also use. Data is transferred from the bus in parallel form. The transmitting UART gets the parallel data from the data bus, it adds a start bit, a parity bit(Sometimes, we haven't implemented it yet), and a stop bit, creating the data packet. Next, the data packet is output serially, bit by bit at the TX pin. The receiving UART reads the data packet bit by bit at its RX pin. The receiving UART then converts the data back into parallel form and removes the start bit, parity bit(sometimes), and stop bits. Finally, the receiving UART transfers the data packet in parallel to the data bus on the receiving end:

**'START BIT'** - UART transmission line is held at high voltage level when it’s not transmitting data. To start the transfer of data, the transmitting UART pulls the transmission line from high to low for one clock cycle(we have to see whether this change from high to low is for more than half of the pulse). When the receiving UART detects the high to low voltage transition, it begins reading the bits in the data frame at the frequency of the baud rate.

**'DATA FRAME' or 'MIDDLE BITS'** - The data frame contains the actual data being transferred. It can be 5 bits up to 9 bits long. In most cases, the data is sent with the least significant bit first.

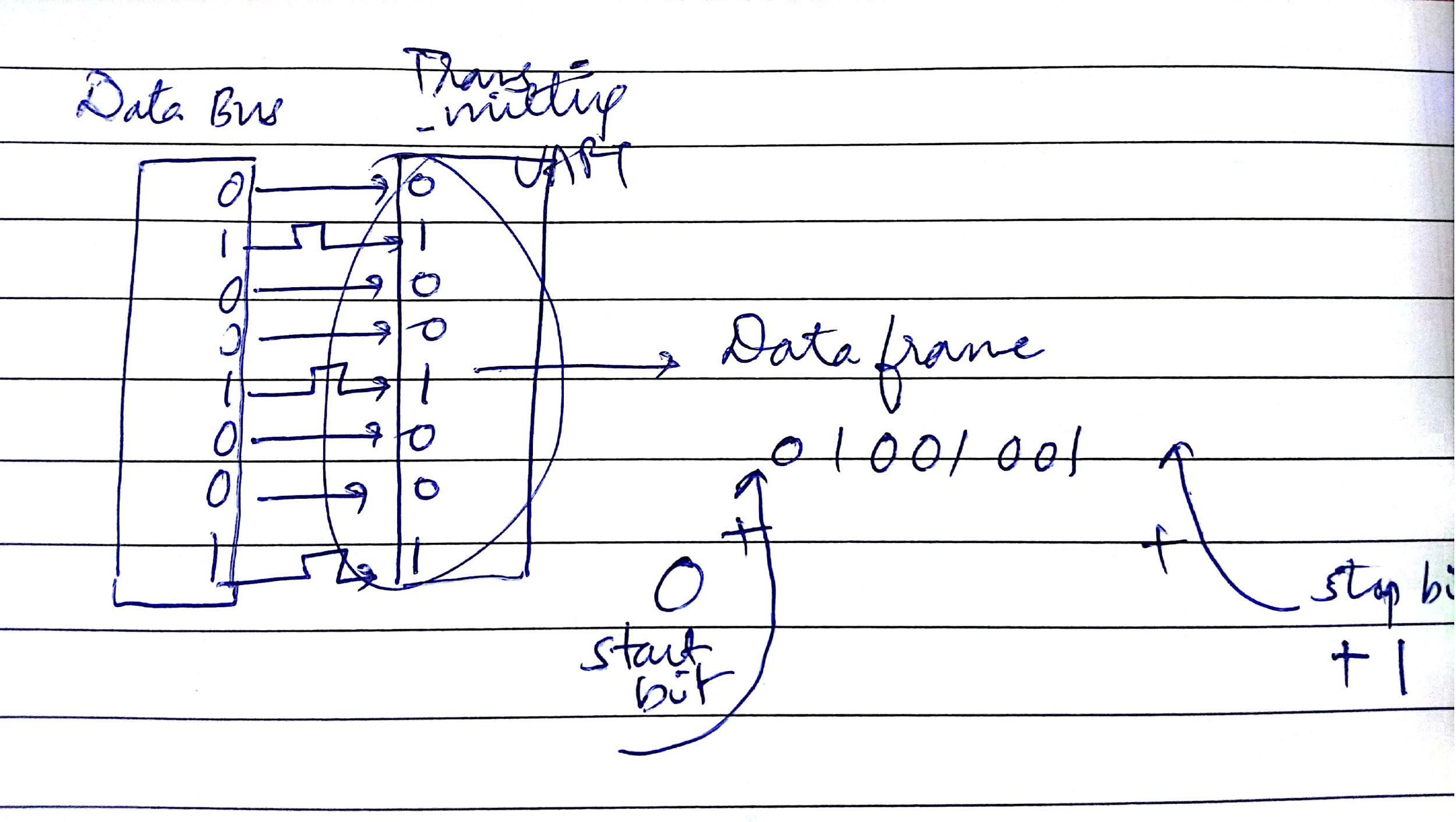
**'STOP BITS'**

To signal the end of the data packet, the sending UART drives the data transmission line from a low voltage to a high voltage for two bit or one bit durations. It shows that this is where it should stop reading the character bits.

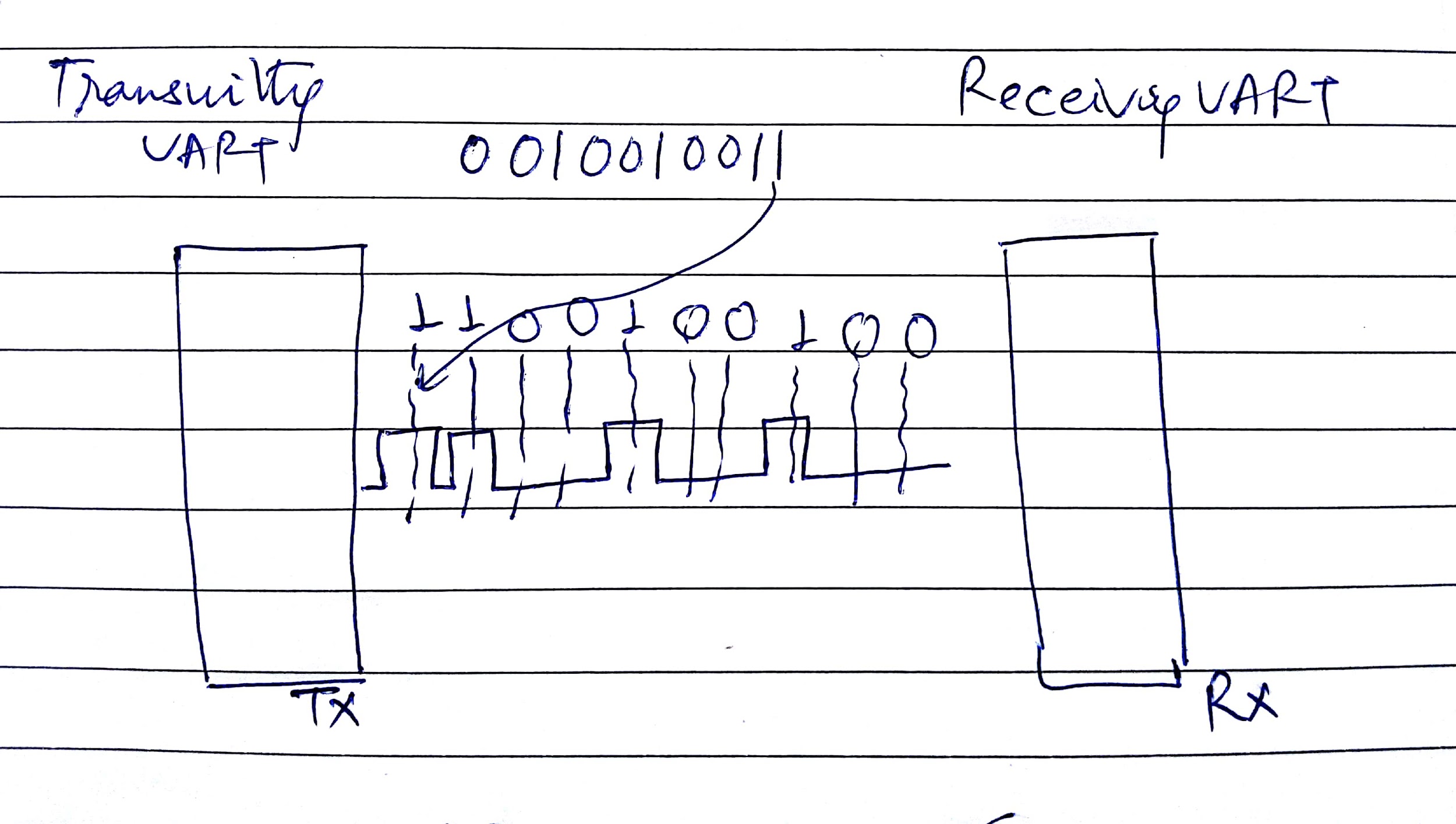
Overall approach:

**STEPS OF UART TRANSMISSION**

1. The transmitting UART receives data in parallel from the data bus:

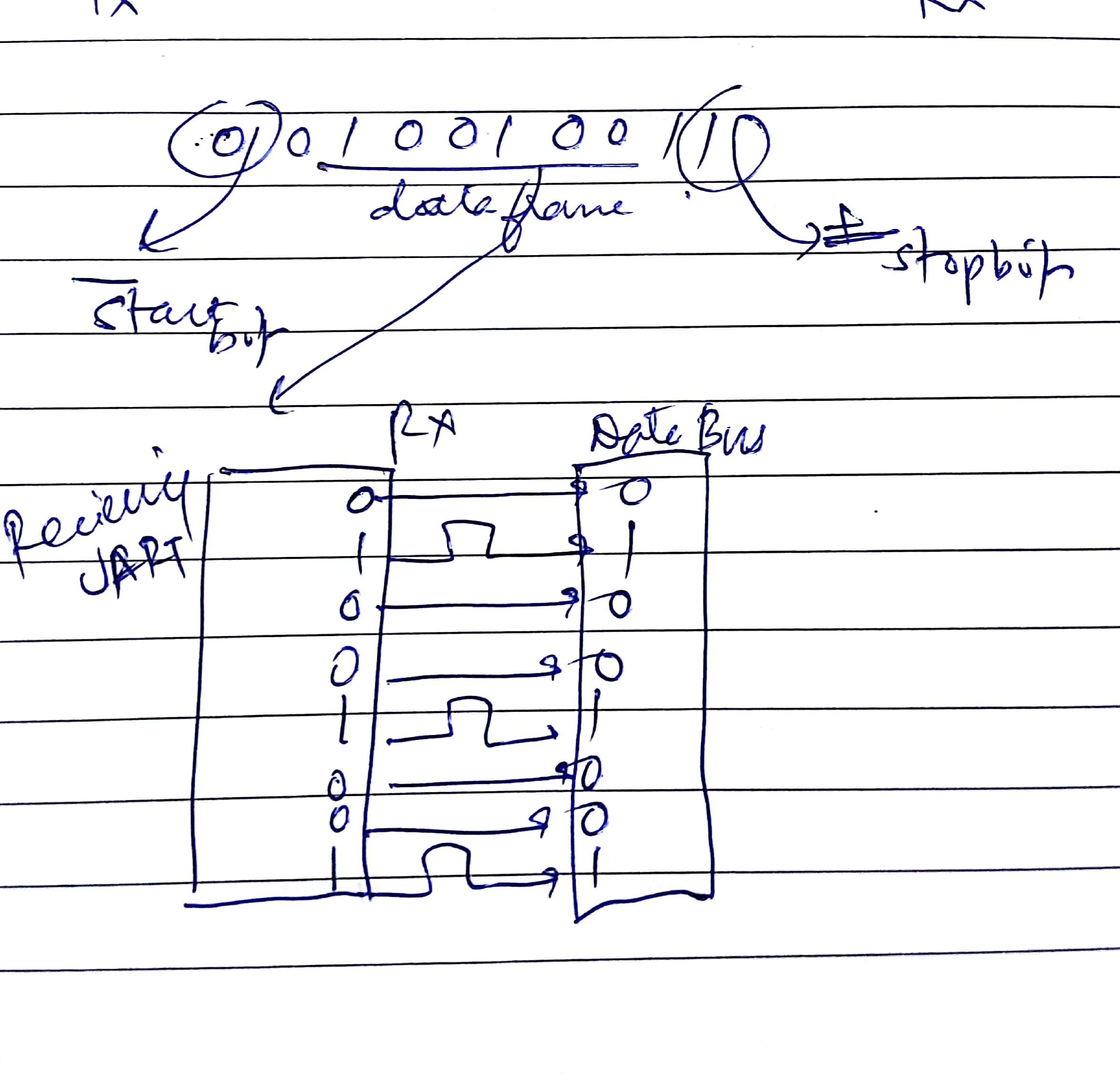


2. The transmitting UART adds the start bit and the stop bit(s) to the data frame: --parity bit is not mentioned for now



3. The entire packet is sent serially from the transmitting UART to the receiving UART. The receiving UART samples the data line at the pre-configured baud rate:

4. The receiving UART discards the start bit and stop bit from the data frame:



5. The receiving UART converts the serial data back into parallel and transfers it to the data bus on the receiving end:

Status of Coding:

We are almost finished with both the UART Receiver(RX) and UART Transmitter(TX) part.

As stated in the description the we defined the entities of UART\_RX and UART\_TX as

**entity uart\_RX** is

generic (

g\_clock\_per\_bit : integer := x -- Needs to be set correctly and as needed

);

port (

in\_clk : in std\_logic; --input clock

in\_RX\_serial : in std\_logic; --input RX serial communication

out\_RX\_DV : out std\_logic; -- output RX DV

out\_RX\_byte : out std\_logic\_vector(7 downto 0) --output RX in byte vector

);

**end uart\_RX;**

and

**entity uart\_TX** is

generic (

g\_clock\_per\_bit : integer := x -- Needs to be set correctly and as needed

);

port (

in\_clk : in std\_logic; --input clock

in\_TX\_serial : in std\_logic; --input RX serial communication

in\_TX\_byte : in std\_logic\_vector(7 downto 0) --8 bit vector is input to transmitter

out\_TX\_active : out std\_logic;

out\_TX\_serial : out std\_logic;

out\_TX\_done : out std\_logic

);

**end uart\_TX;**

the comments in the entities are neatly explaining the way the entities take input and output.

Testing Status:

We are still trying to test it. Hopefully, we will do it before final submission. :)