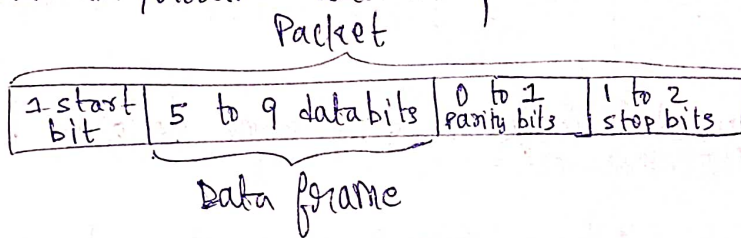


UART - A Hardware Communication Protocol :- $\left[\begin{array}{c} \text{IC No.} \\ 8250, 16450, 16550 \end{array} \right]$

- UART is not a communication protocol like SPI and I2C, but a physical circuit in a microcontroller, or a stand-alone IC.
- UART uses only 2 wires to transmit and receive data b/w 2 devices.
- It's asynchronous, which means there is no clock signal to synchronize the output of bits from the transmitting UART to the sampling of bits by the receiving UART.
- Baud rate → symbols (bits) per second.
- UART transmits/receives data as packets.



- * → The UART data Tx line → usually held logic high voltage when not Tx'ing → start bit = 0 for 1 clock cycle ⇒ Rx starts reading.
- Data Frame → LSB sent first
- We use parity bit for error detection.
- Bits can be changed by electromagnetic radiation, mismatched baud rates, or long distance data transfers.
- Even parity → parity bit = 0
Odd parity → parity bit = 1
- * → UART drives the Tx line from a low voltage to a high voltage for at least 2 bit durations.
- *** Embedded Systems, microcontrollers, and computers mostly use UART as a form of device-to-device hardware communication protocol.

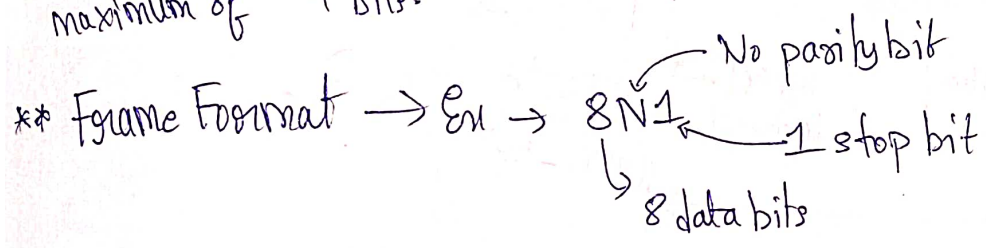
*** UART is universal because, the parameters like transfer speed, data speed, etc. are configurable.

*** It uses only 2 wires.

*** All Arduino boards and microcontrollers (Ex: PIC) have atleast 1 serial port, also known as a UART or USART.

** The no. of bits b/w the start and stop bits in a UART are 5 to 9, only because the ~~error~~ sampling error accumulates as the no. of bits increase and the possibility of sampling a wrong bit.

*** Disadvantage is the size of the data frame is limited to a maximum of 9 bits.



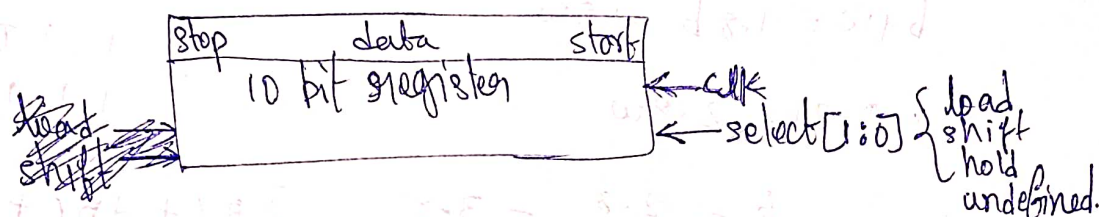
→ Debounce
→ JTAG debug



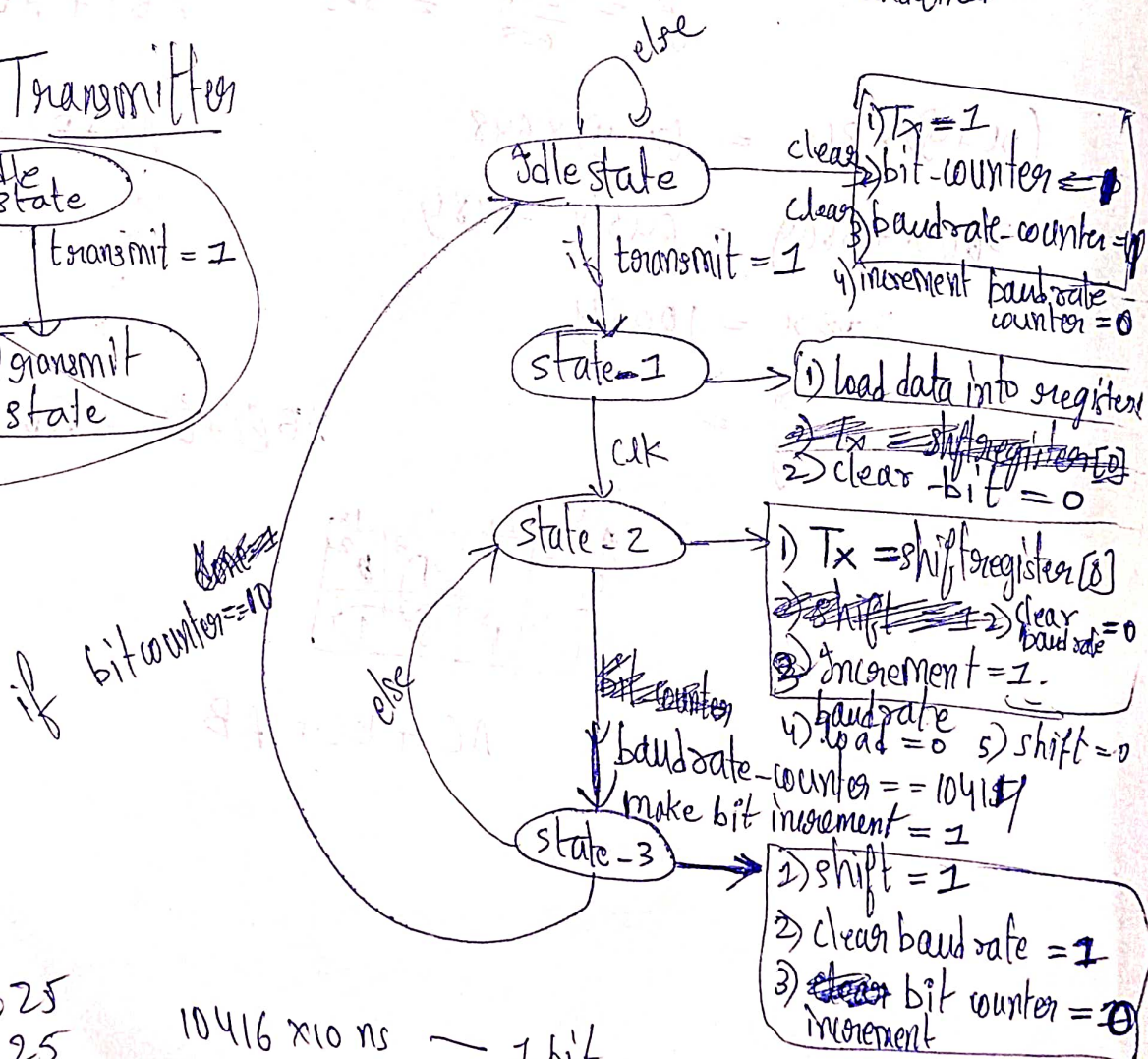
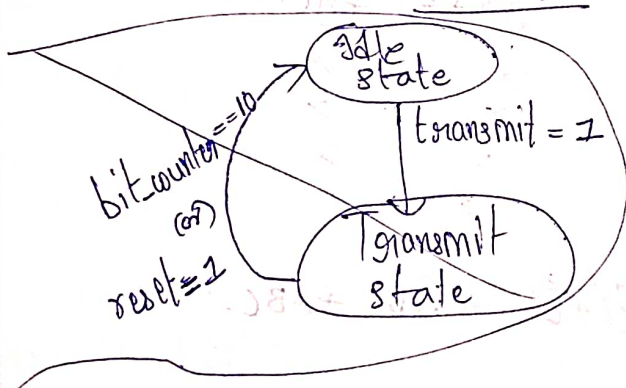
8N1 Format, $\text{clk} = 100 \text{ MHz}$, Baud rate = 9600 bps-
Asynchronous serial.

Transmitter
clk
data [7:0]
reset
transmit

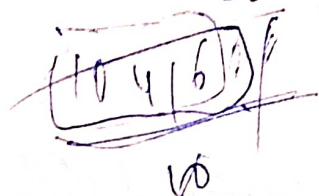
Tx



Moore machine: Transmitter



10, 41, 625
25



10416 x 10 ns — 1 bit

1 sec

$\frac{1}{10416 \times 10^6} \times 10^9 = 10^8$

2 3 4 5
1 1 1 1

Receiver:

0 1 2 3 4

OSR = 16

10416 clock cycles \rightarrow 1 bit duration

OSR = 16

$\Rightarrow \frac{10416}{16} = 651$ clock cycles \rightarrow 1 sample

Idle_state

$R_x = 0$ & $clk = 1$

State_1

inc-samplegenerator = 0
inc-samplescounter = 0
clear-samplegenerator = 1
clear-samplescounter = 1
clear-bitcounter = 1

clear-samplegenerator = 0
clear-samplescounter = 0
inc-samplegenerator = 1
inc-samplescounter = 1
shift = 0
inc-bitcounter = 0

sampled-clk = 1

$R_x = 1$

\Rightarrow glitch

samplescounter == 8
sampled-clk = 1

is not glitch

state_2

input of datapath [9] = R_x
shift = 1
bit-counter = 1
inc

sampled-clk = 1
&
bit-counter == 10

state_3

load = 1
1) Received-data = datapacket [8:0]

done = 1

inc-samplegenerator = 0
inc-samplescounter = 0

load = 0, done = 0
glitch-check-done = 0
inc-samplescounter = 0
clear-samplescounter = 1
clear-bitcounter = 1

Idle_state

$R_x = 0$ and sampled-clk = 1

state_1

inc-samplescounter = 1
shift = 0
inc-bitcounter = 0
clear-samplescounter = 1
clear-bitcounter = 0

Glitch

$R_x = 1$ w glitch check done

samplescounter == 7 \Rightarrow Not a glitch

sampled-clk = 1

state_2

shift = 1
inc-bitcounter = 1
glitch-check-done = 1

s-clk
bit-counter == 10

state_3

load = 1
done = 1

10416 xrons

104160 ns

+ 12

104172 ns

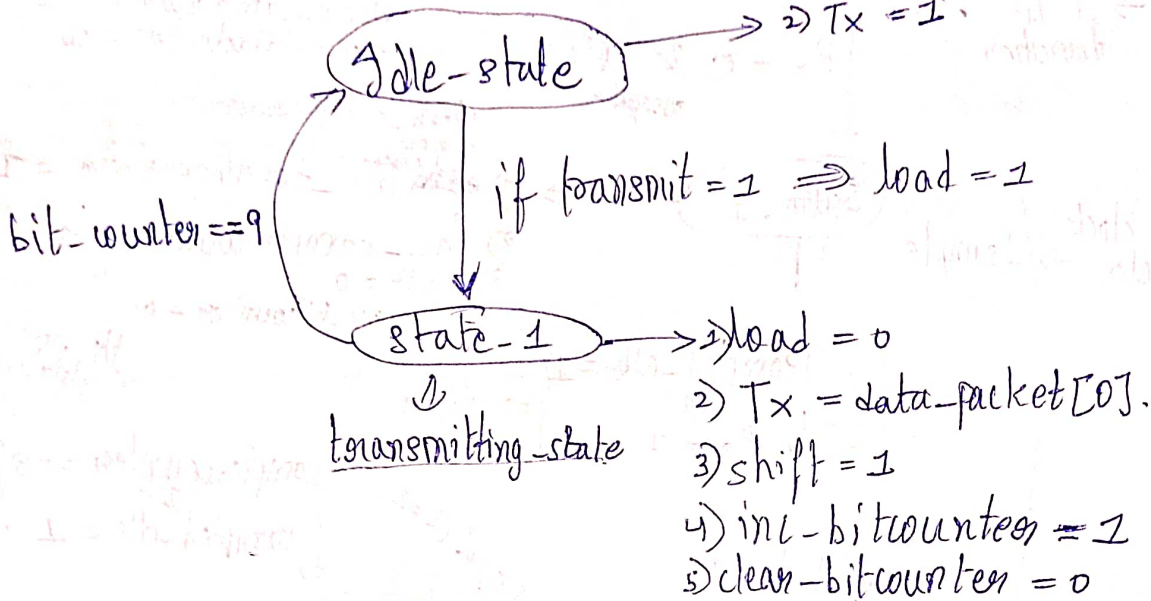
0101 0101
401

14
16
64
85

85
93

Transmitter (Sampled clock)

1) shift = 0, inc-bitcounter = 0, clear-bitcounter = 1
2) Tx = 1.



10416 Qns

52,080