UART - A Hardware Communication Protocol :- [IC No. 8250, 16450, 16550]

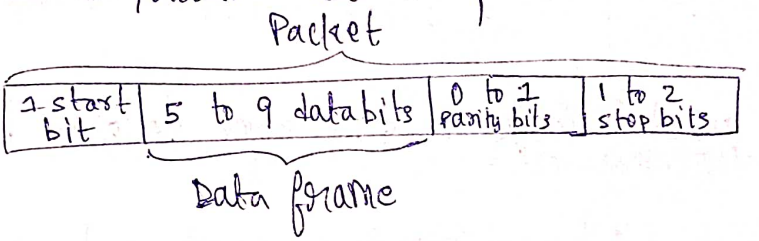
→ UART is not a communication protocol like SPI and I2C, but a physical circuit in a micro controller, or a stand-alone IC.

→ UART uses only 2 wires to transmit and receive data b/w 2 devices.

→ Its 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 Txing → 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 atleast 2 bit durations.

*** Embedded Systems, micro controllers, 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.

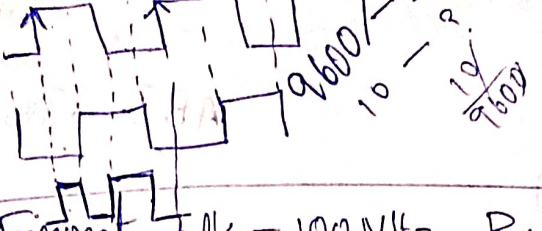
*** All Arduino boards and microcontrollers (Ex:- PIC) have atleast 1 serial port, also known as a UART or USART.

** 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.

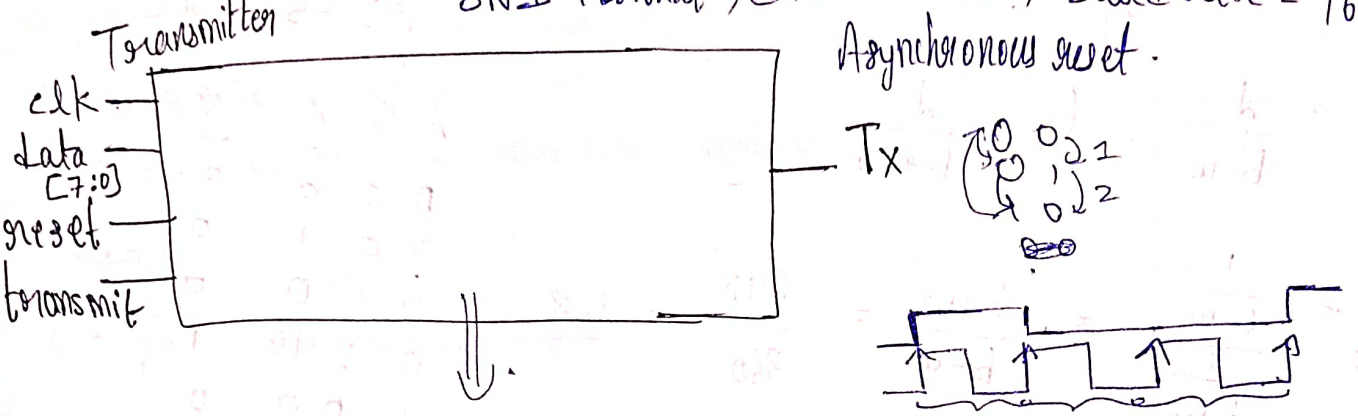
*** Disadvantage is the size of the data frame is limited to a maximum of 9 bits.

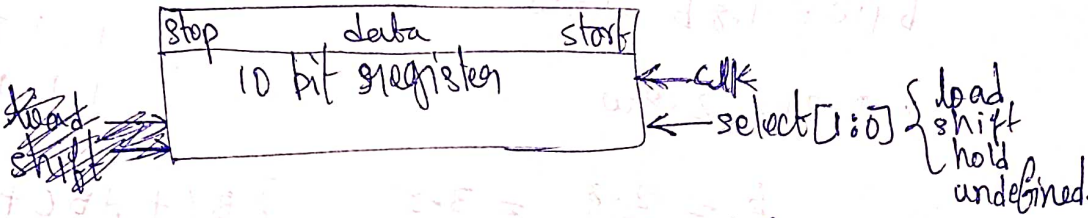
** Frame Format → Ex → 8N1
- No parity bit
- 1 stop bit
- 8 data bits

→ Debouncing
→ JTAG Debug.

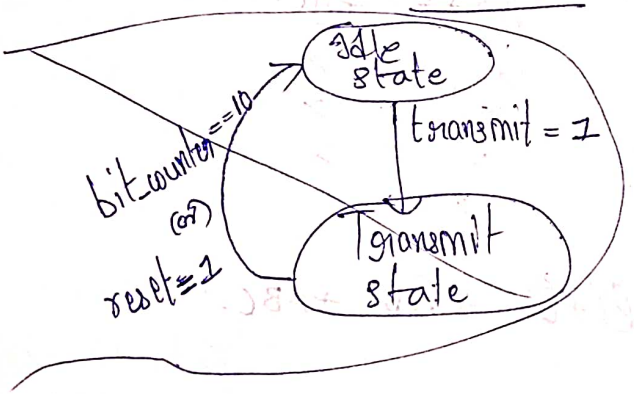


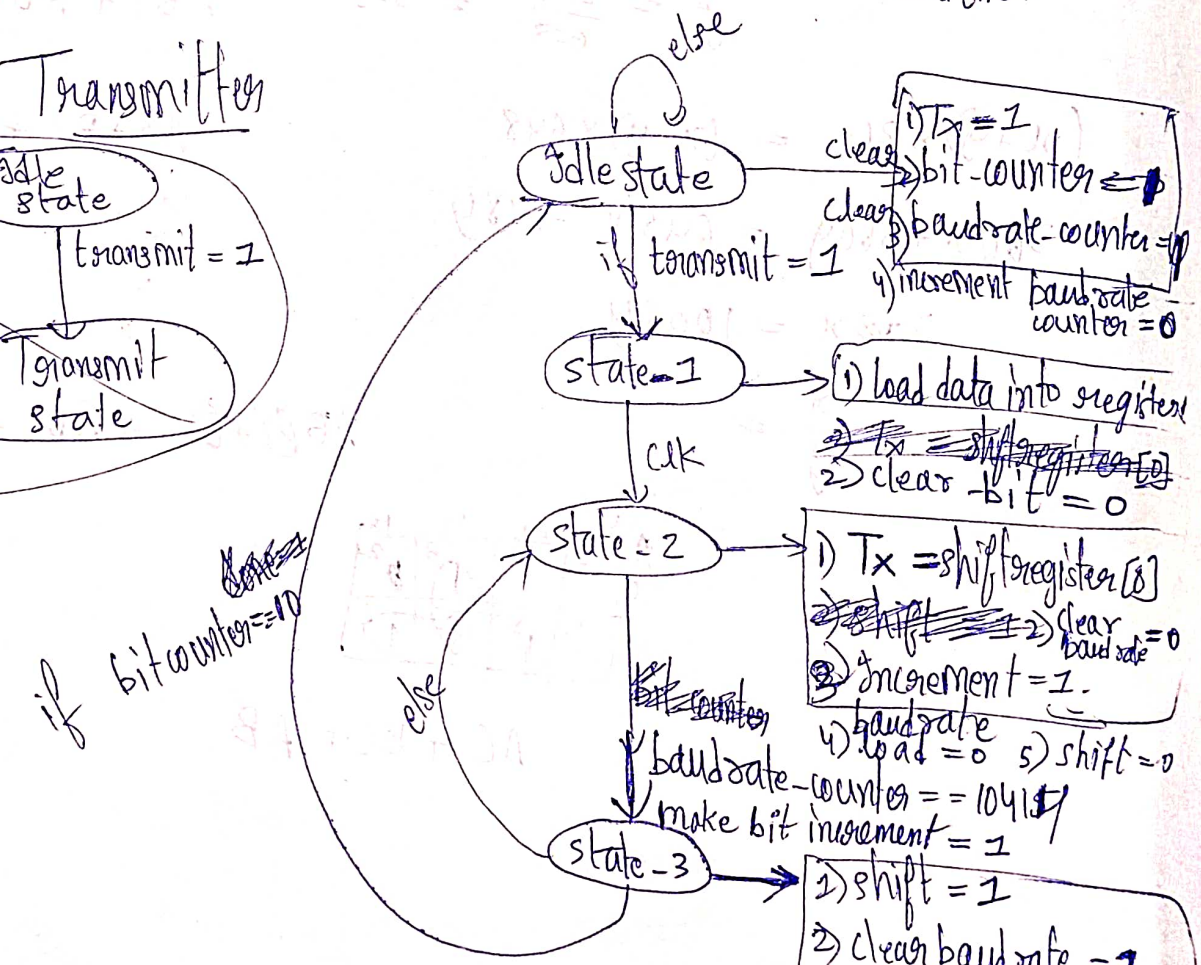
8N1 Format, Clk = 100 MHz, Baud rate = 9600 bps - Asynchronous sent.





Moore machine: Transmitter





10,41,625
25
10416

10416 x 10 ns — 1 bit

1 sec — ?

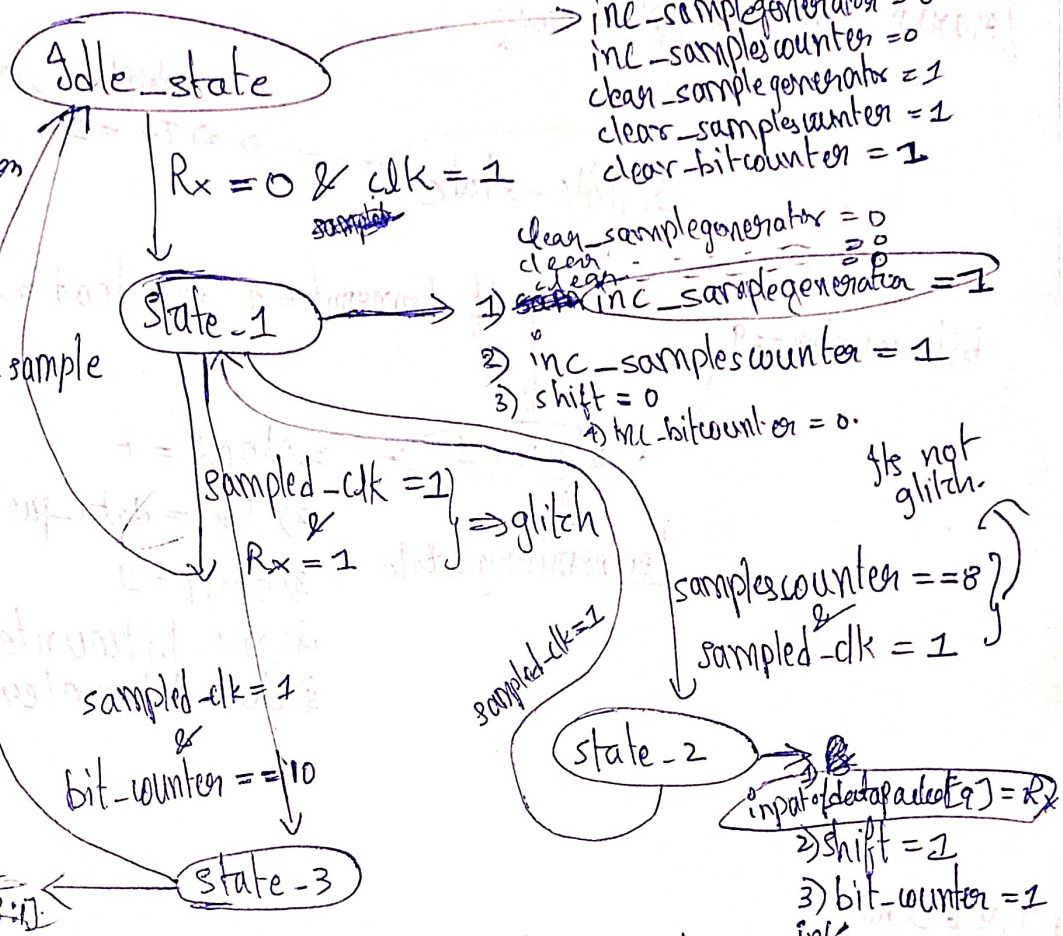
$\frac{1}{10416 \times 10} \times 10^9 : 10^8$

2 3 4 5
1 1 1 1

0 1 2 3 ④

OSR = 16

Recievers :



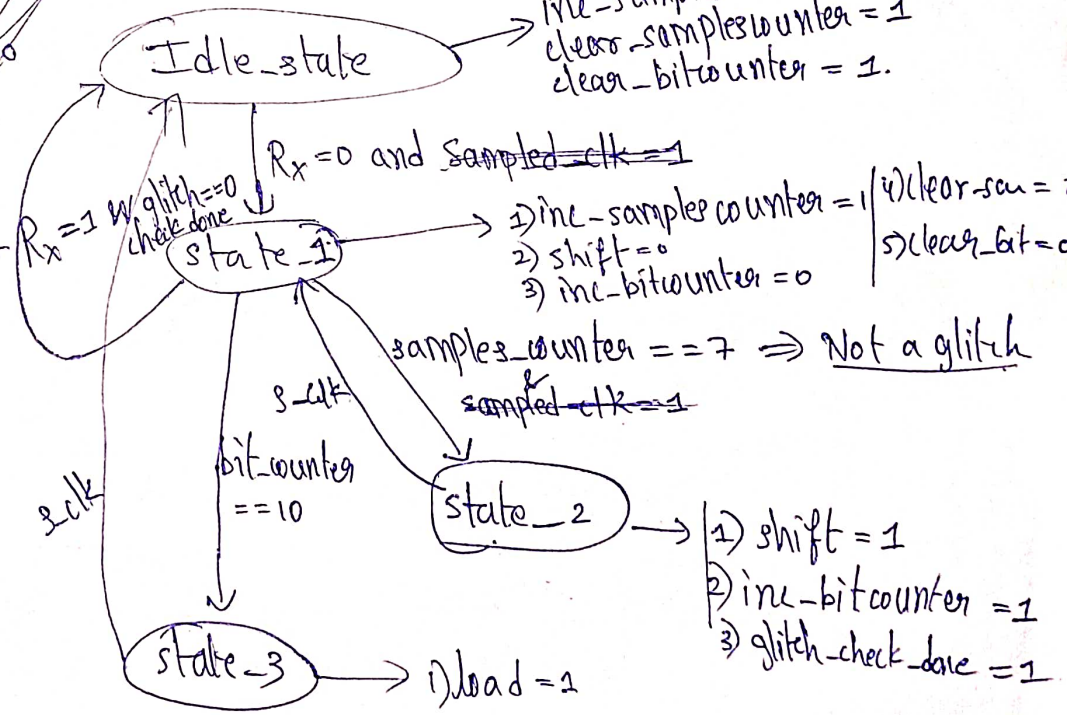
10416 clock cycles → 1 bit duration

OSR = 16

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

load = 0, done = 0
glitch_check_done = 0
inc_samplescounter = 0
clear_samplescounter = 1
clear_bitcounter = 1.

Glitch ∈



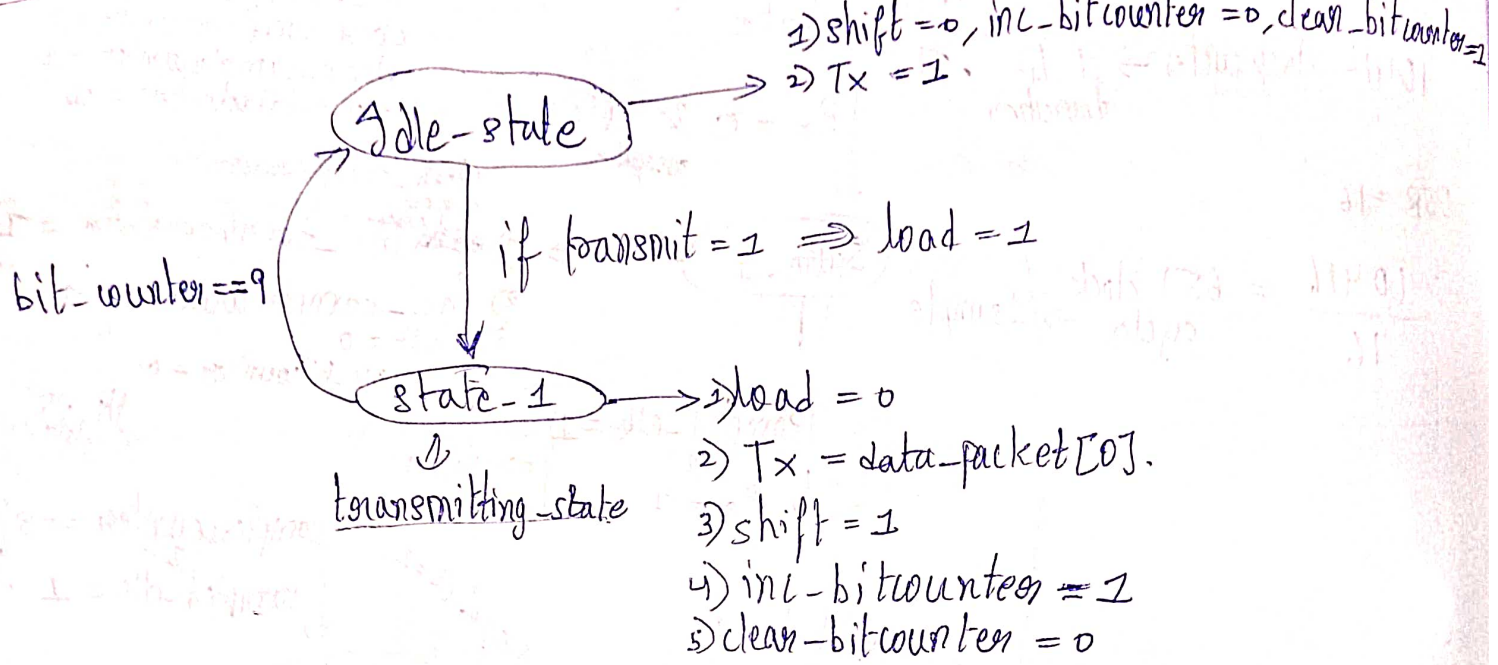
10416 xtons

104160 ns
+ 12
104172 ns

01010101.

85

Transmitter (Sampled clock)



10416 Cycles

52,080