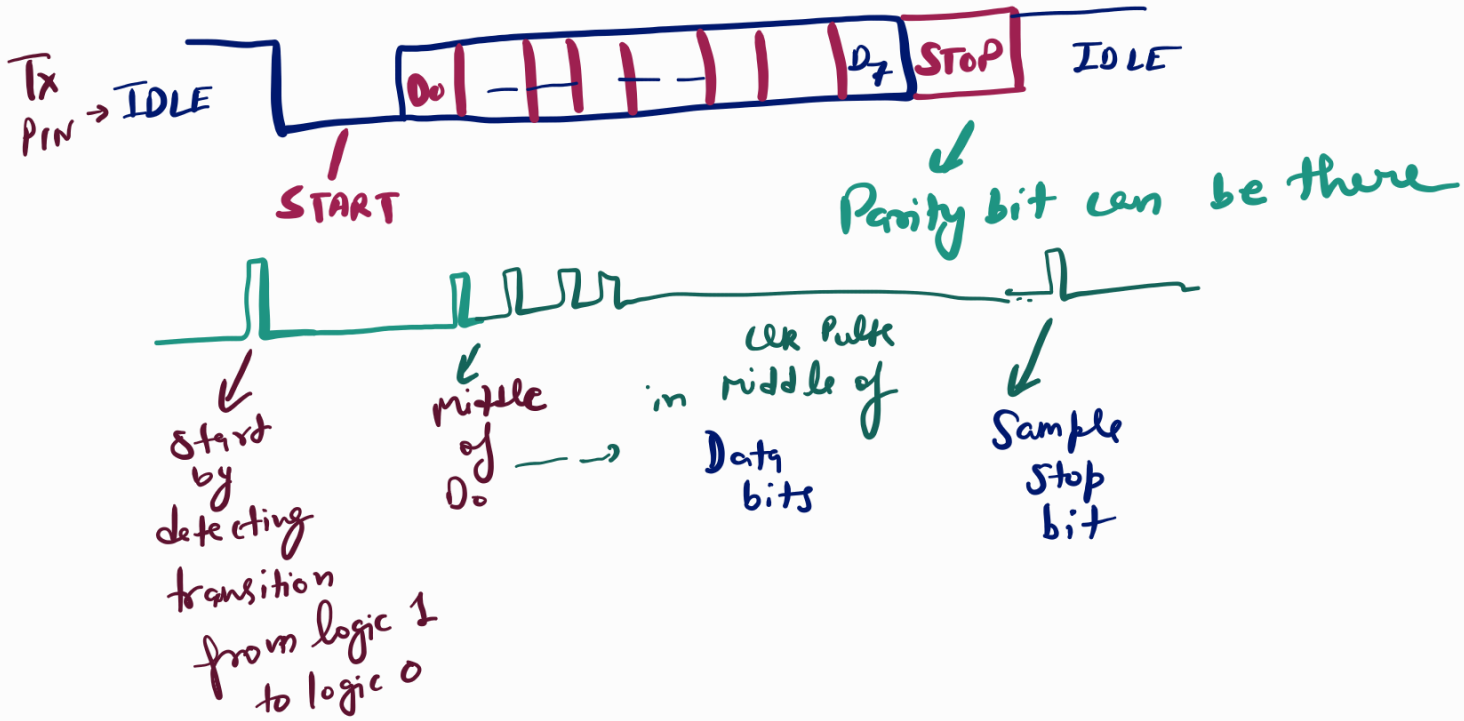
UART

① Baud Rate Generator : (BRG)



Baud rate

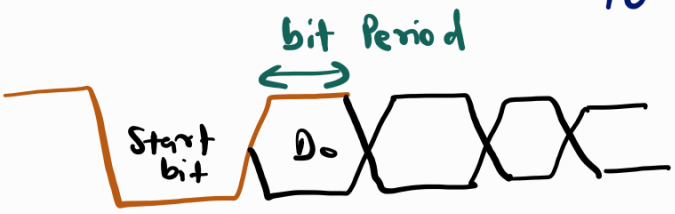
No. of bits transmitted per second are called Baud Rate.

Common B.R. = 9600 bits per sec.

Baud rate Generator Design Equation

$$\frac{1}{\text{Baud rate}} = \text{bit Period} \Rightarrow \frac{1}{b} = T_b$$

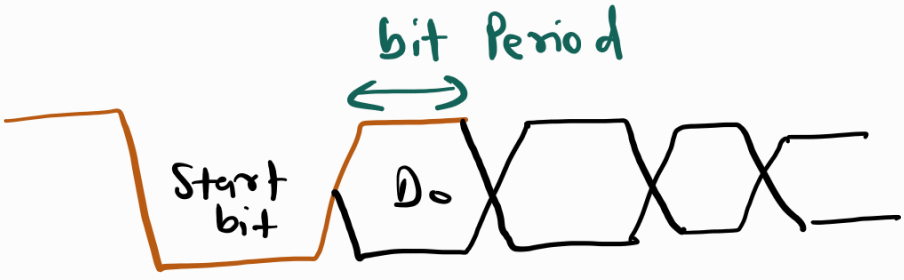
For 9600 ⟶ 104 μsec. time needed to transmit 1 bit.



How UART will know that T_b time has already passed?

#Oversampling Technique

with the help of Counter



T_b

Divide T_b into 16 equal parts → $\frac{T_b}{16}$

BRG: Generate (16) Active high pulse after every $\frac{T_b}{16}$ time (sec)

to transmit each bit

is working 16 times faster as compared to desired Baud Rate

Oversampling technique

Counter ckt → has Cmax → ? SOLVE

↓

Required: $\frac{T_b}{16}$ time $= (C_{max}+1)\,T_{CLK}$ Produced by: $\Rightarrow C_{max} = \frac{f_{CLK}}{16b} - 1$

Oversampling factor → It can be anything --

Baud rate

$C_{max} = \frac{f_{CLK}}{16b} - 1$ where $T_{CLK} = \frac{1}{f_{CLK}}$, $T_b = \frac{1}{b}$

Operating Clock freq. of FPGA board

Baud rate

EXAMPLE: FIND What should be the MAXIMUM VALUE of Counter to produce band rate of 9600 bps on FPGA board having clock signal frequency of 100 MHz.

$$C_{max} = \frac{f_{CLK}}{16b} - 1 = \frac{100 \times 10^6}{9600 \times 16} - 1 \simeq 650$$

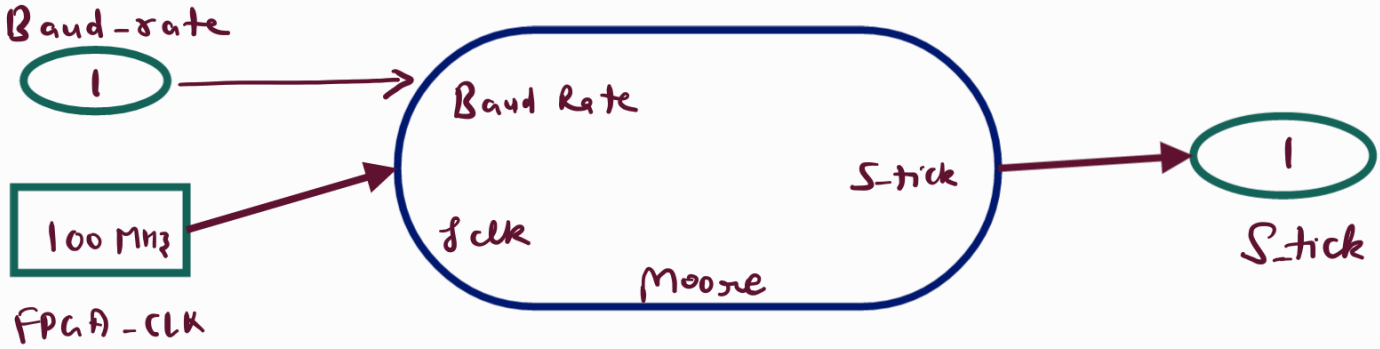
Counter counts from 0 to 650 & generates a HIGH PULSE each time it reaches to 650.

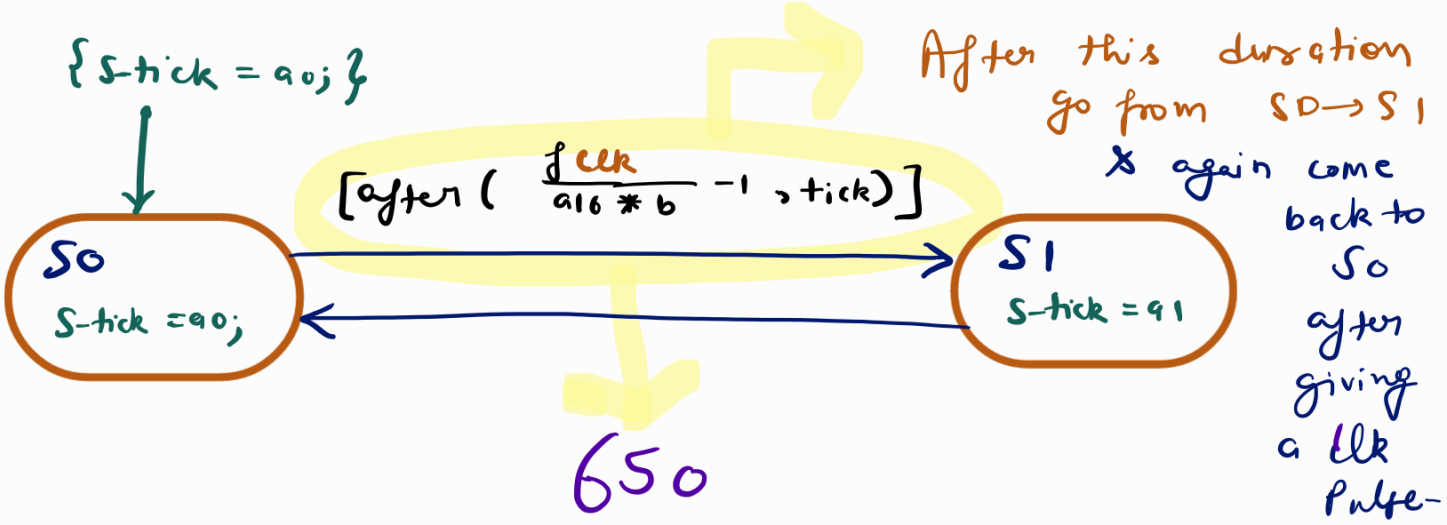
Time : 0 to $\frac{T_b}{16}$

Counter : 0 to 650

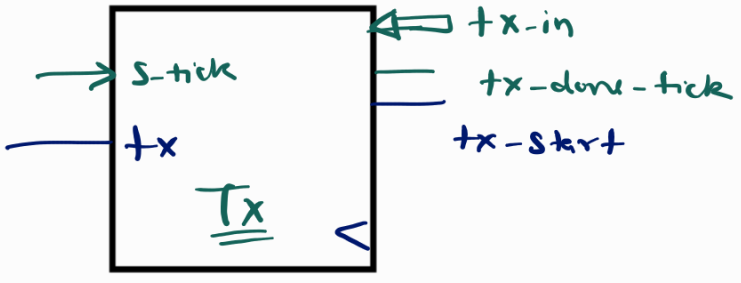
$\frac{T_b}{16} \rightarrow 650$

$$T_b = 16 \times 650$$

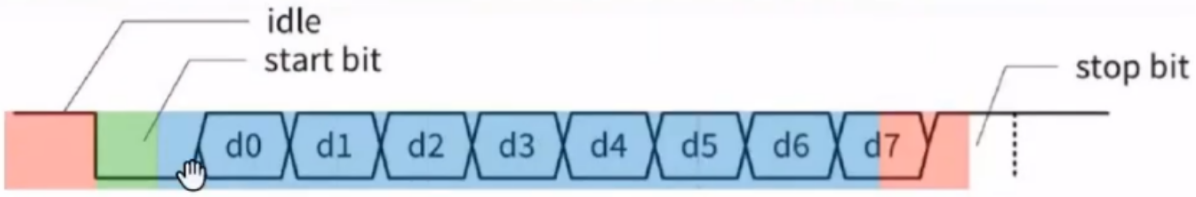




Transmitter

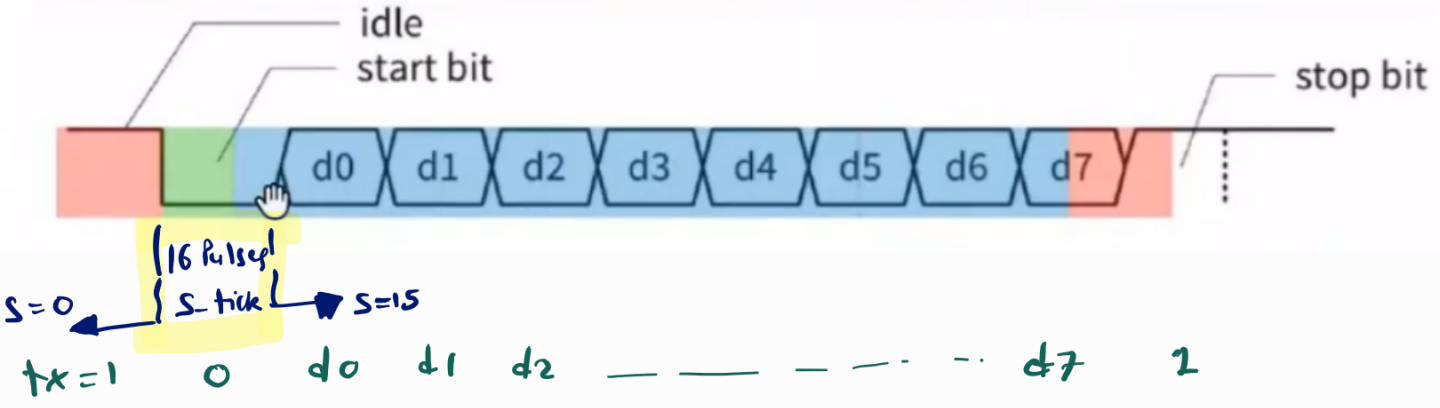


Oversampling Technique:

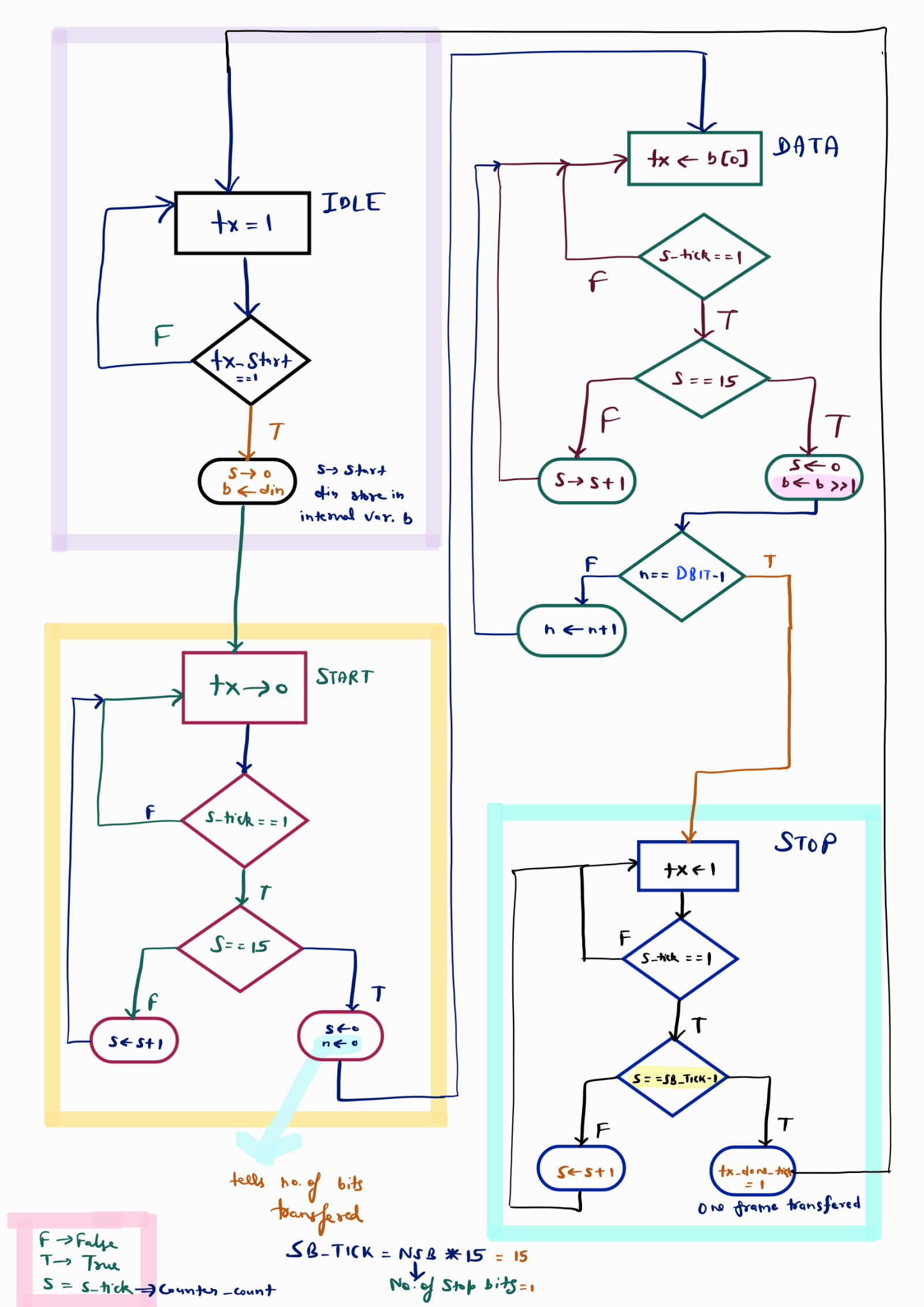


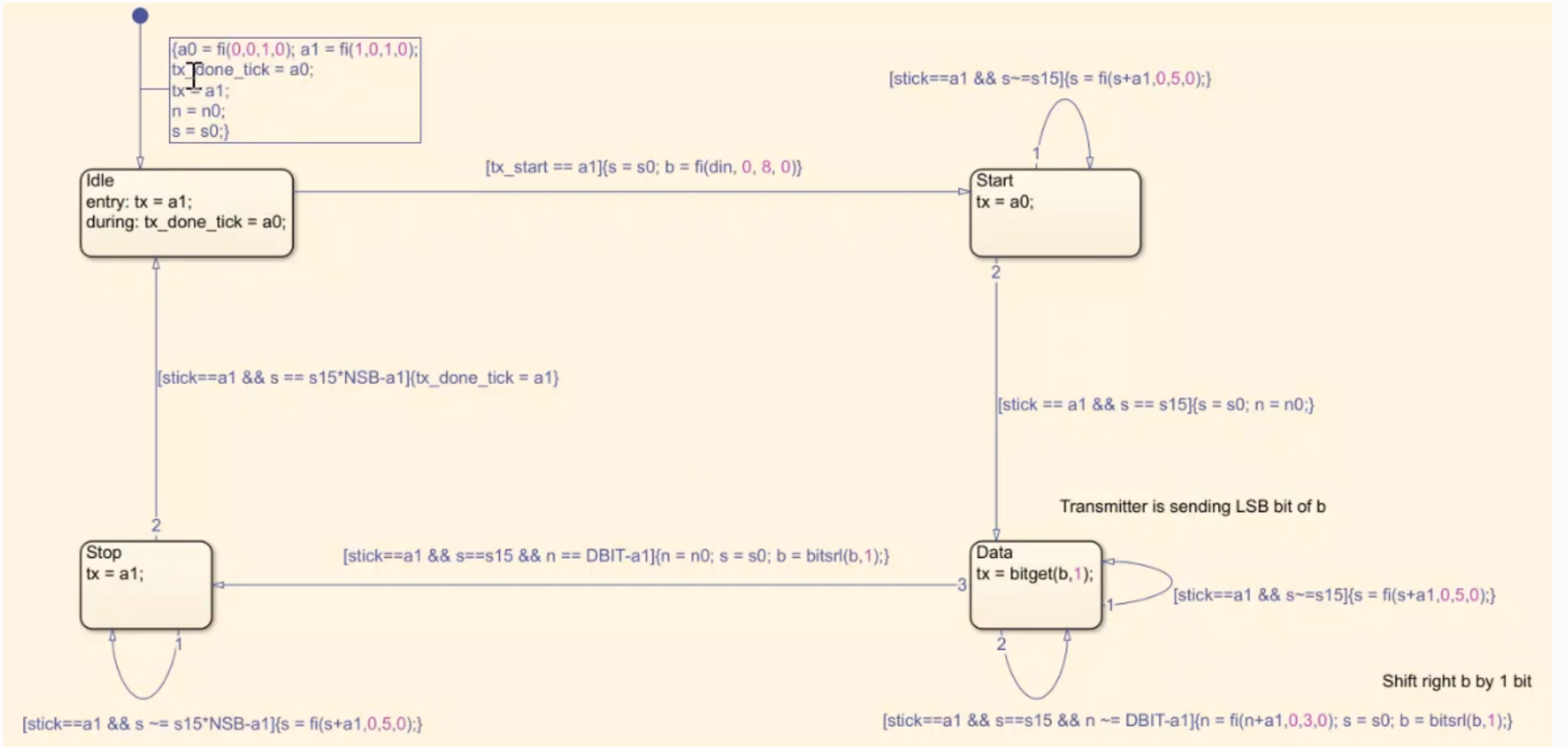
1. Wait until incoming signal becomes 0 (start bit), then start the sampling tick counter
2. When tick counter reaches 7 (middle of start bit), clear tick counter and restart
3. When counter reaches 15 (middle of first data bit), shift bit value into register & restart tick counter
4. Repeat step 3 (N - 1) more times to retrieve the remaining data bits
5. If optional parity bit is used, repeat step 3 one time
6. Repeat step 3 (M) more times to obtain stop bits

N data bits
M stop bits

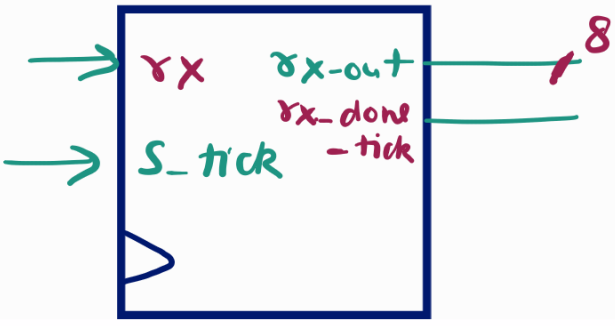


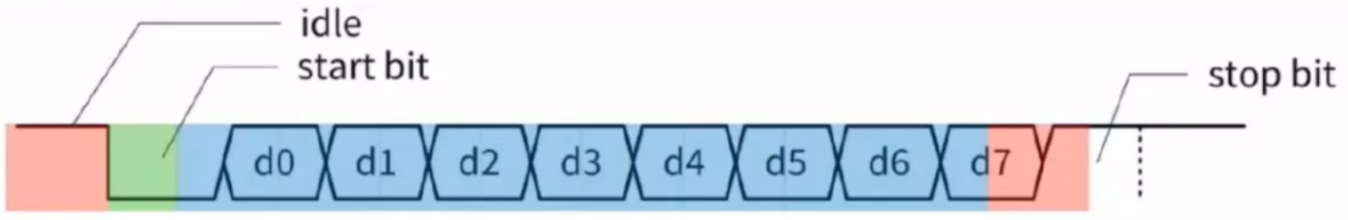
Right Shift all other Data already transfered


IDLE
tx = 1
F
tx_start ==1
T
S → 0
b ← din
S→ start
din store in
internal var. b
START
tx → 0
F
S_tick = = 1
T
S = = 15
F
S ← S+1
T
S ← 0
n ← 0
tells no. of bits
transfered
DATA
tx ← b[0]
F
S_tick = = 1
T
S = = 15
F
S → S+1
T
S ← 0
b ← b >>1
F
n = = DBIT-1
T
n ← n+1
STOP
tx ← 1
F
S_tick = = 1
T
S = = SB_TICK-1
F
S ← S+1
T
tx_done_tick = 1
One frame transfered
F → False
T → True
S = S_tick → Counter_count
SB_TICK = NSB * 15 = 15
No. of Stop bits = 1


{a0 = fi(0,0,1,0); a1 = fi(1,0,1,0);
tx_done_tick = a0;
tx = a1;
n = n0;
s = s0;}
Idle
entry: tx = a1;
during: tx_done_tick = a0;
[tx_start == a1]{s = s0; b = fi(din, 0, 8, 0)}
Start
tx = a0;
[stick==a1 && s~=s15]{s = fi(s+a1,0,5,0);}
1
2
[stick==a1 && s == s15*NSB-a1]{tx_done_tick = a1}
[stick == a1 && s == s15]{s = s0; n = n0;}
Transmitter is sending LSB bit of b
2
Stop
tx = a1;
[stick==a1 && s==s15 && n == DBIT-a1]{n = n0; s = s0; b = bitsrl(b,1);}
3
Data
tx = bitget(b,1);
[stick==a1 && s~=s15]{s = fi(s+a1,0,5,0);}
1
1
2
Shift right b by 1 bit
[stick==a1 && s ~= s15*NSB-a1]{s = fi(s+a1,0,5,0);}
[stick==a1 && s==s15 && n ~= DBIT-a1]{n = fi(n+a1,0,3,0); s = s0; b = bitsrl(b,1);}

Reciever





1.Wait until incoming signal becomes 0 (start bit), then start the sampling tick counter

2.When tick counter reaches 7 (middle of start bit), clear tick counter and restart

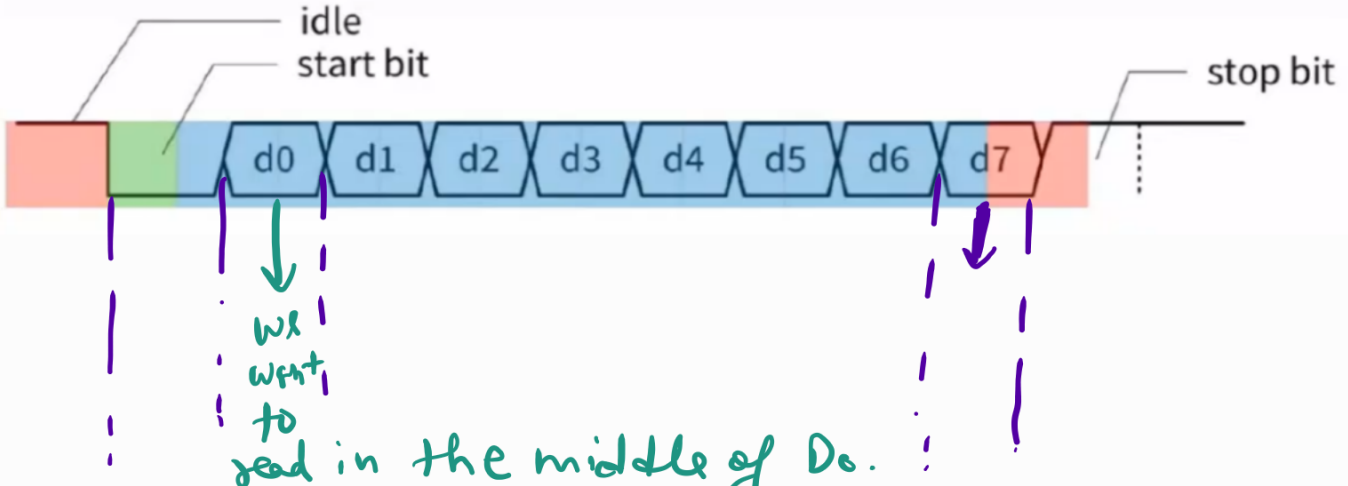
3.When counter reaches 15 (middle of first data bit), shift bit value into register & restart tick counter

4.Repeat step 3 (N - 1) more times to retrieve the remaining data bits

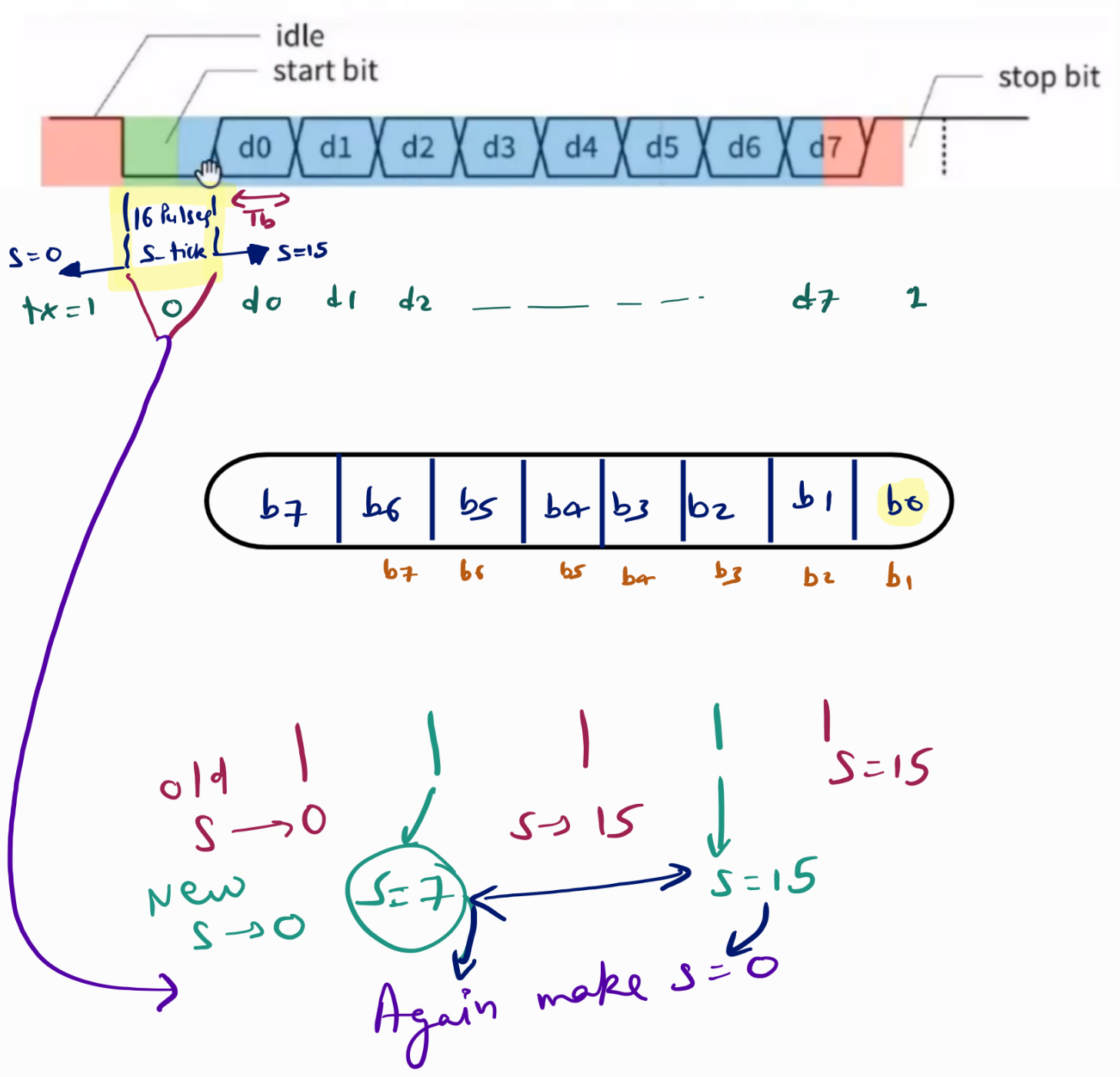
5.If optional parity bit is used, repeat step 3 one more time

6.Repeat step 3 (M) more times to obtain stop bits

N data bits
M stop bits

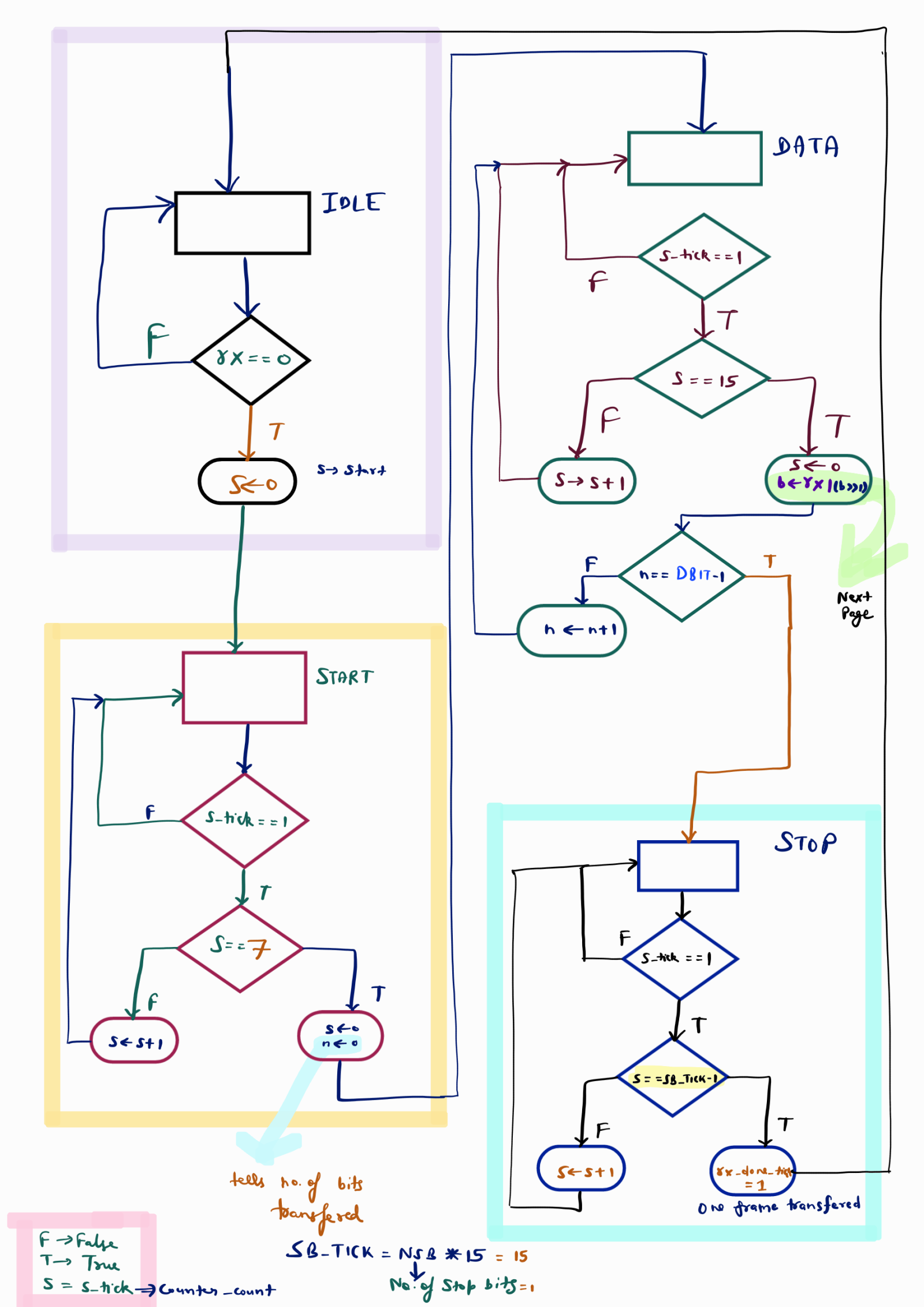


Read @ $\frac{T_b}{2}$

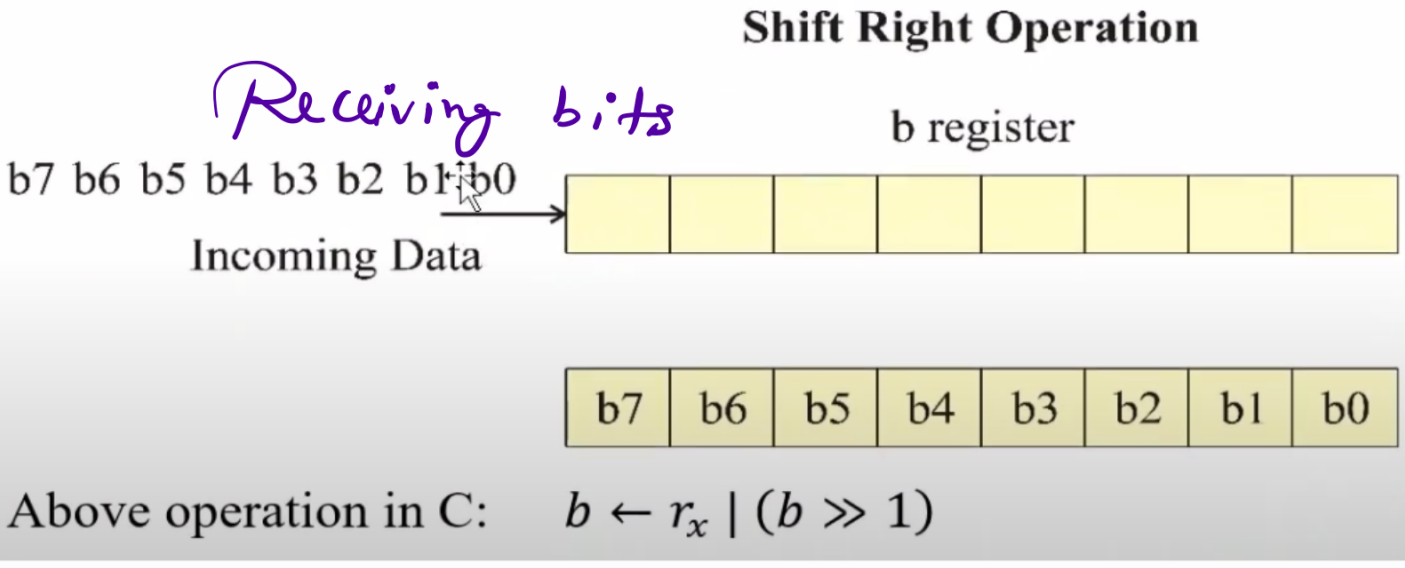


Receiver

State Machine


IDLE
rx == 0
F
T
S <- 0
S -> start
START
S_tick == 1
F
T
S == 7
F
T
S <- S+1
S <- 0
n <- 0
tells no. of bits transfered
DATA
S_tick == 1
F
T
S == 15
F
T
S -> S+1
S <- 0
b <- rx | (b >>)
n == DBIT-1
F
T
n <- n+1
Next Page
STOP
S_tick == 1
F
T
S == SB_TICK-1
F
T
S <- S+1
rx_done_tick = 1
One frame transfered
F -> False
T -> True
S = S_tick -> Counter_count
SB_TICK = NSB * 15 = 15
No. of Stop bits = 1

Shift Right Register Implementation



Above operation in C: $b \leftarrow r_x \mid (b \gg 1)$

Incoming data is : $b_0\ b_1 \ldots\ldots b_7$

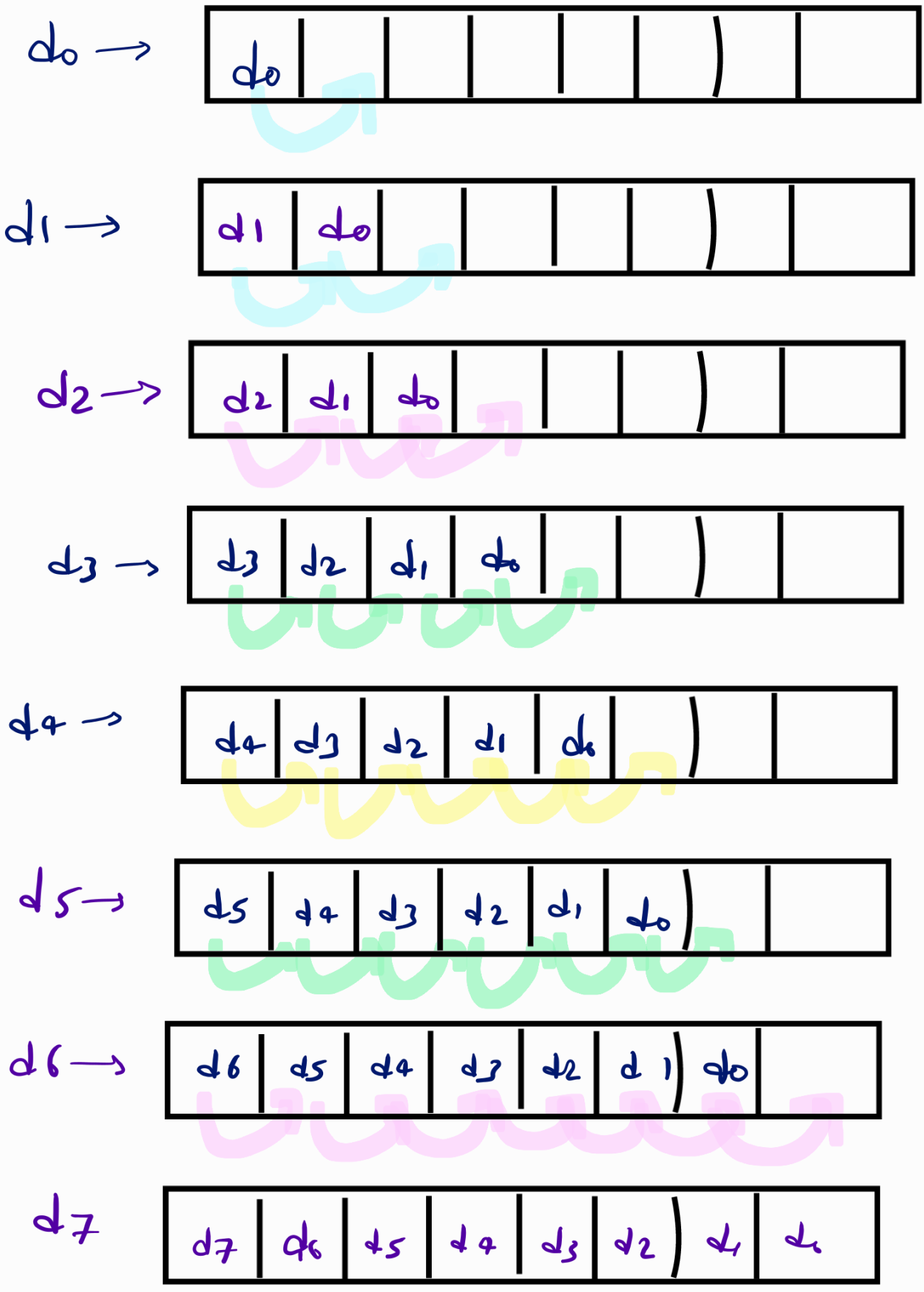
(b_0 received first and b_7 at last)

r_x bit is receiving the bits $b_0\ b_1, \ldots$ ..... b_7 serially.

After executing above instruction, b register contains the bits as :

$b = b_7\ b_6 \_\_\_\_\_ b_0$

means Receive & Shift


d0 →
d0
d1 →
d1
d0
d2 →
d2
d1
d0
d3 →
d3
d2
d1
d0
d4 →
d4
d3
d2
d1
d0
d5 →
d5
d4
d3
d2
d1
d0
d6 →
d6
d5
d4
d3
d2
d1
d0
d7
d7
d6
d5
d4
d3
d2
d1
d0