Detailed Report on UART (Universal Asynchronous Receiver Transmitter)

Ayush Yadav

Introduction

Universal Asynchronous Receiver Transmitter (UART) is a communication protocol used for serial data transmission. Unlike synchronous protocols, UART does not require a shared clock signal between the transmitter and receiver. Instead, it uses predefined settings like the baud rate to ensure synchronization.

Key Concept: Baud Rate

The baud rate determines the number of bits transmitted per second. For a given baud rate B, the bit period T_b is calculated as:

$$T_b = \frac{1}{B} \tag{1}$$

For example, with a baud rate of 9600 bps (bits per second):

$$T_b = \frac{1}{9600} = 104.17\,\mu s \tag{2}$$

Oversampling Technique

To improve accuracy, UART employs oversampling, where the bit period T_b is divided into smaller intervals. A common oversampling factor is 16×, dividing T_b into 16 parts.

Counter Value Calculation

The Baud Rate Generator (BRG) divides the system clock frequency f_{clk} to derive the baud rate using a counter. The maximum counter value C_{max} is given by:

$$C_{max} = \frac{f_{clk}}{B \times 16} \tag{3}$$

For example, with $f_{clk} = 100$ MHz and $B = 9600$ bps:

$$C_{max} = \frac{100 \times 10^6}{9600 \times 16} = 650 \tag{4}$$

UART Transmitter

The transmitter state machine operates through the following steps:

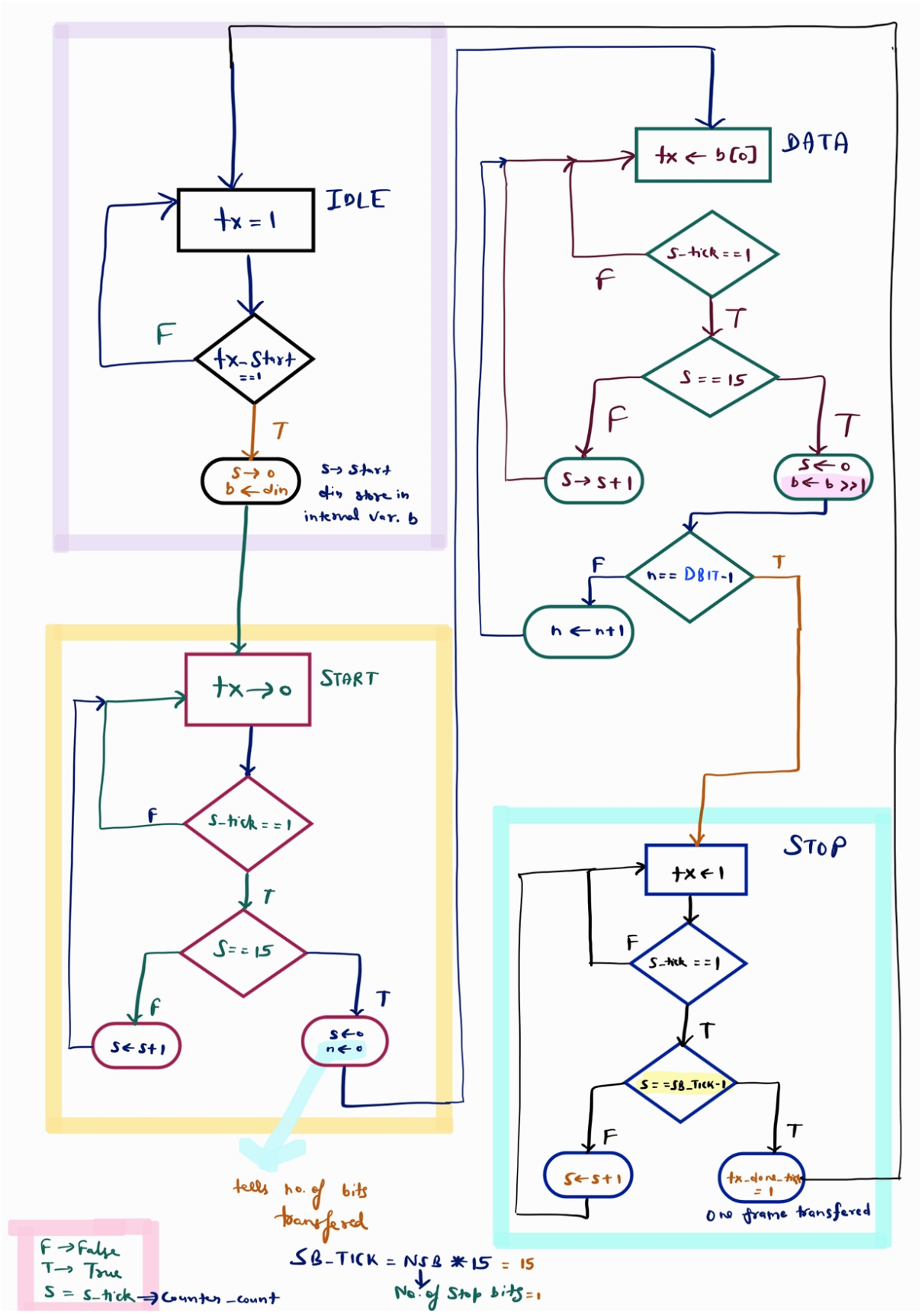

Figure 1: State Machine for UART Transmitter

Step 0a: Reset

Initialization of all variables and setting the transmitter to the idle state. This ensures that the system is prepared to transmit data without errors.

Step 0b: Idle State

The transmitter waits for a `tx_start` signal. During this state, no data transmission occurs, and the transmitter is ready to begin the next cycle.

Step 0c: Start Bit

The line is pulled low for one bit period T_b. This signals the start of a transmission and allows the receiver to synchronize with the transmitter.

Step 0d: Data Bits Transmission

Each data bit is transmitted serially, starting with the least significant bit (LSB). The transmitter shifts the data buffer and sends bits one by one until all bits are transmitted.

Step 0e: Stop Bit

The line is set high for one bit period T_b. This signals the end of the transmission and allows the receiver to validate the data.

Step 0f: Return to Idle

The transmitter resets and waits for the next `tx_start` signal. This ensures readiness for the next transmission cycle.

State Machine Table

State	Condition	Actions	Next State
Reset	Reset signal asserted	Initialize variables	Idle
Idle	`tx_start` asserted	Load data into buffer	Start
Start	-	Set `tx` line to 0	Data
Data	Bit counter $s < 7$	Transmit LSB, shift buffer	Data
Data	$s = 7$	Transmit last bit	Stop
Stop	-	Set `tx` line to 1	Idle

Table 1: Transmitter State Machine Table

UART Receiver

The receiver operates through the following steps:

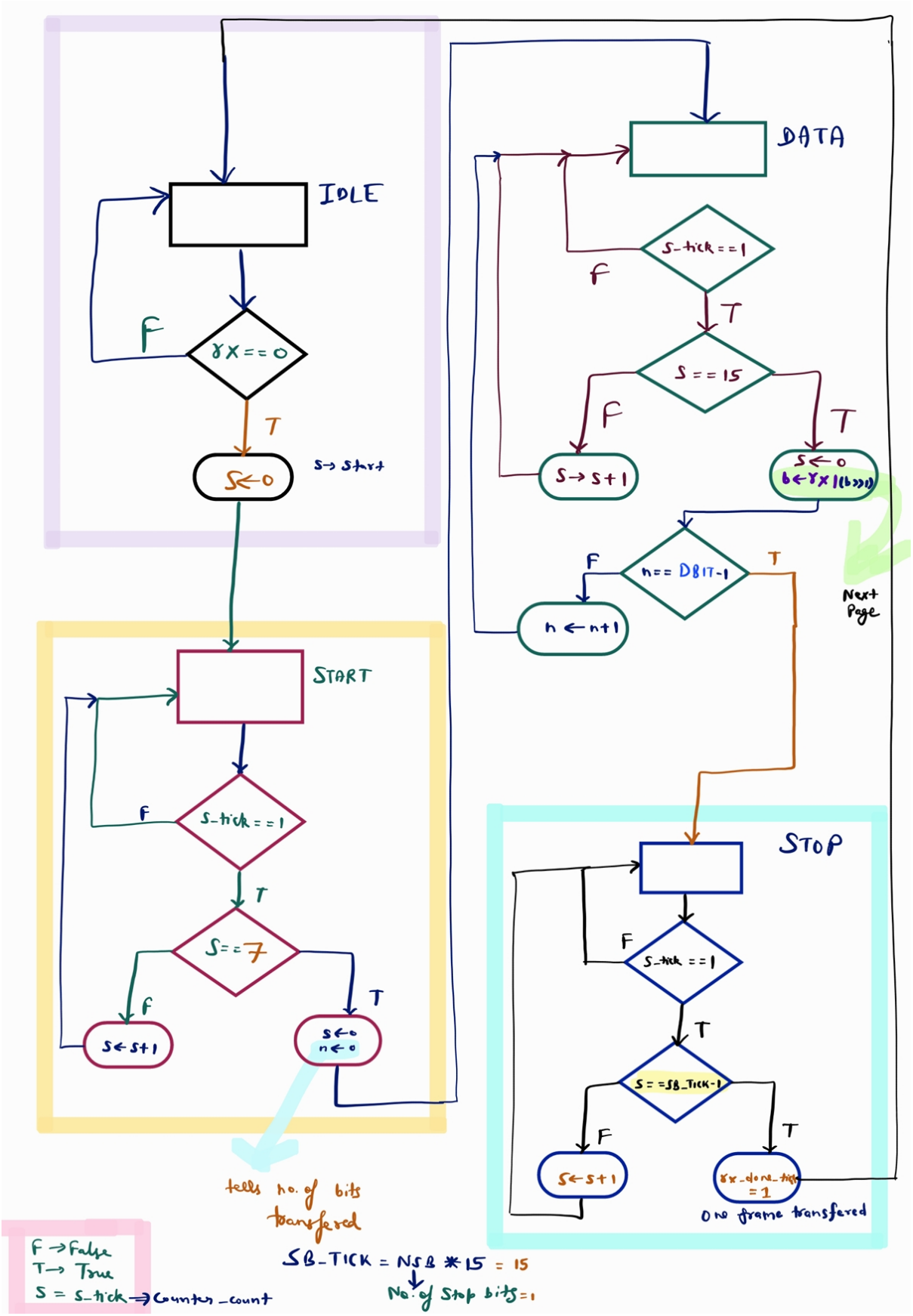

Figure 2: State Machine for UART Receiver

Step 0a: Idle

The receiver monitors the line for a low signal (start bit). This is the default state where no data reception is occurring.

Step 0b: Start Bit Detection

The line is sampled at the midpoint of T_b to validate the start bit. This ensures that the receiver has accurately detected the beginning of a transmission.

Step 0c: Data Bits Reception

The receiver samples and stores each bit at the middle of T_b. This process continues for all data bits, starting with the least significant bit (LSB).

Step 0d: Stop Bit Validation

The receiver validates the stop bit by checking if the line is high. This confirms that the transmission was completed correctly.

Step 0e: Return to Idle

The receiver resets and waits for the next start bit. This prepares the system for the next data reception cycle.

State Machine Table

State	Condition	Actions	Next State
Idle	`rx` line low	Reset bit counter	Start
Start	Midpoint of T_b	Validate start bit	Data
Data	Bit counter $s < 7$	Sample `rx` line, store bit	Data
Data	$s = 7$	Store last bit	Stop
Stop	`rx` line high	Validate stop bit	Idle

Table 2: Receiver State Machine Table

Summary

UART is a robust and widely used protocol for serial communication. By leveraging techniques like baud rate generation and oversampling, it ensures accurate data transmission and reception. The use of state machines and step-by-step processes allows the transmitter and receiver to operate efficiently, handling start bits, data bits, and stop bits in a structured manner.