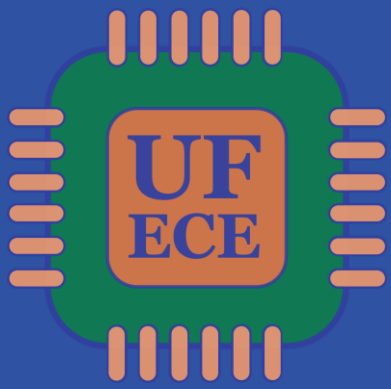
Microprocessor Applications

Universal Asynchronous Receiver/Transmitter (UART)



UF UNIVERSITY of FLORIDA

Dr. Eric Schwartz
Christopher Cr
Wesley Piard

1

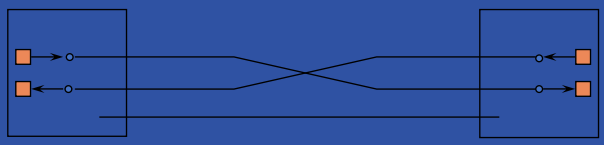
2

General Description of UART

- A Universal Asynchronous Receiver/Transmitter (**UART**) is a system or device that is designed to receive and transmit serial data asynchronously.
- As its name implies, a UART can be used in a variety of applications and can implement many different communication protocols.

Full Duplex vs. Half Duplex

- Full-duplex systems can transmit and receive data simultaneously. **UART is typically full-duplex.**

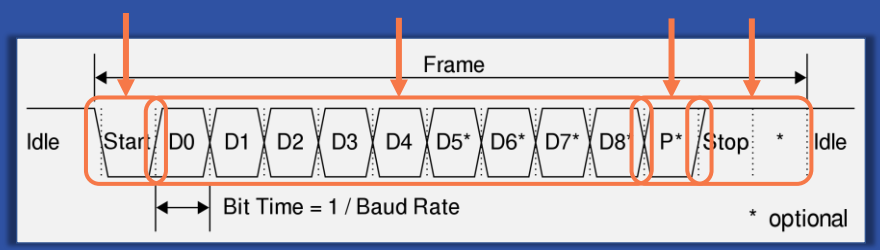


- Half-duplex systems can only either transmit or receive at any given time.

UART Transmission Characteristics

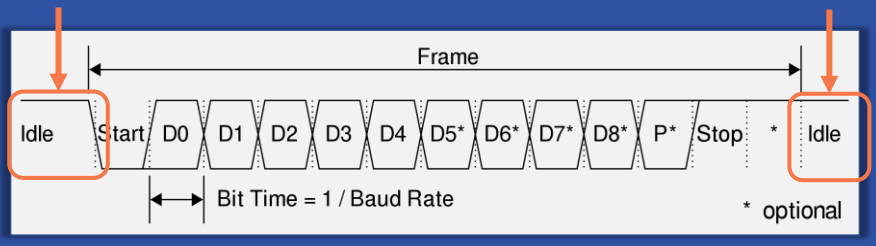
- A frame of data in a UART transmission is depicted below.

- Each frame typically consists of a start bit, an adjustable number of data bits, an optional bit for parity (error detection), and one or more stop bits.



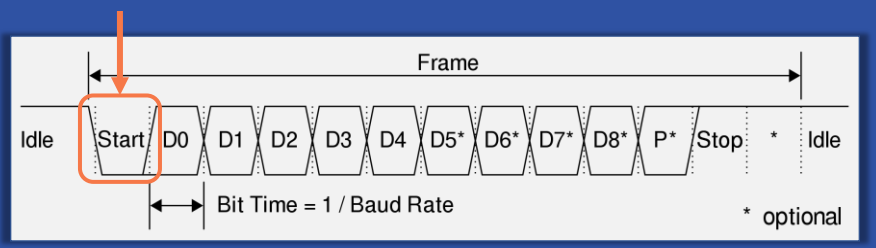
UART Transmission Characteristics – Idle State

- When there is no data being transmitted, a "high" voltage is present. This is the **idle** state.



UART Transmission Characteristics – Start Bit

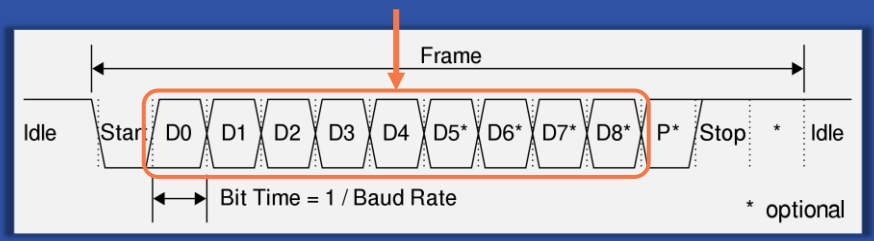
- The start bit is what indicates a new frame is being transmitted; it is a transition from the "high" idle state to a "low" or zero voltage state.



7

UART Transmission Characteristics – Data Bits

- A UART frame can be configured to transmit some number of data bits.
- Eight data bits is the most common configuration.
- Typically, the least-significant data bit (LSb) is transmitted first.

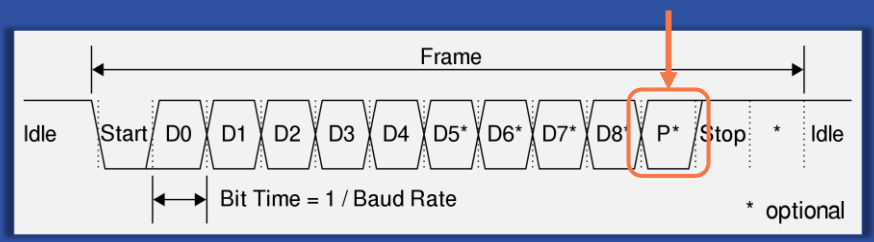


7

8

UART Transmission Characteristics - Parity

- In addition, there is an optional bit, known as the parity bit, that can be used for error detection.
- A parity bit normally follows the data bits.



8

UART Transmission Characteristics - Parity

The parity bit is used to ensure that none of the bits in a UART frame were corrupted during transmission.

- If a parity bit is utilized, even or odd parity can be chosen.
- For even parity, the parity bit should ensure that the total number of '1' data bits in the frame is even.
- For odd parity, the parity bit should ensure that the total number of '1' data bits in the frame is odd.
- It is the transmitter's job to properly set the parity bit, and it is the receiver's job to properly calculate if the parity bit is correct.

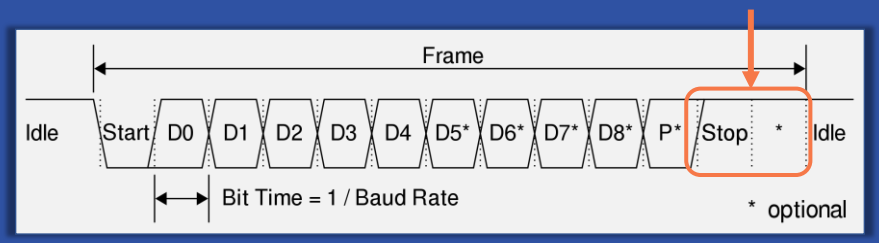
UART Transmission Characteristics - Parity

Example: Consider the 8-bit binary value **0b11001010**. Note that there are four '1' bits in this 8-bit value.

- Even parity:
  - The total number of '1' bits is already even, so the parity bit would be '0'.
- Odd parity:
  - The total number of '1' bits is even, so the parity bit would be '1'. This would make the total number of '1' bits five, which is odd.

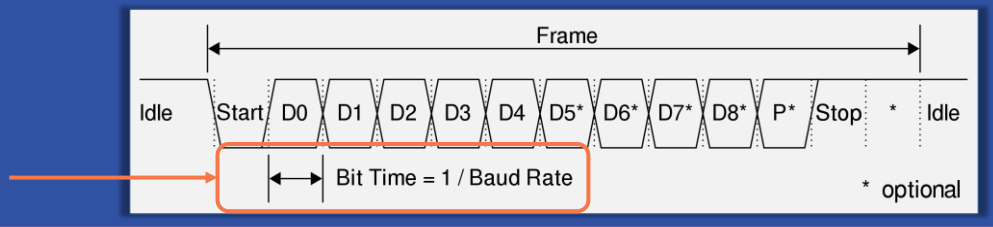
UART Transmission Characteristics – Stop Bit(s)

- Stop bit(s) indicate the end of a frame. A stop bit indicates a return to the idle state.
- For most UARTs, there can be one or more stop bits.



UART Baud Rate

- For UART, the baud rate is measured in bits per second. It must be the same for two UARTs to communicate.
- The time it takes to transmit each bit can be calculated as the inverse of the baud rate.

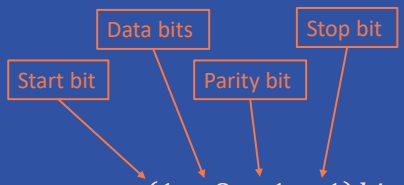


UART Transmission Time Example

- For some applications, it may be useful to calculate how long an entire transmission would take. Note that in a lot of cases, a full transmission may be more than a single 8-bit value.
- The time to transmit a certain amount of data via UART can be calculated if the following things are known:
  - Baud rate
  - Number of data bits
  - Parity (none, odd, or even)
  - Number of stop bits

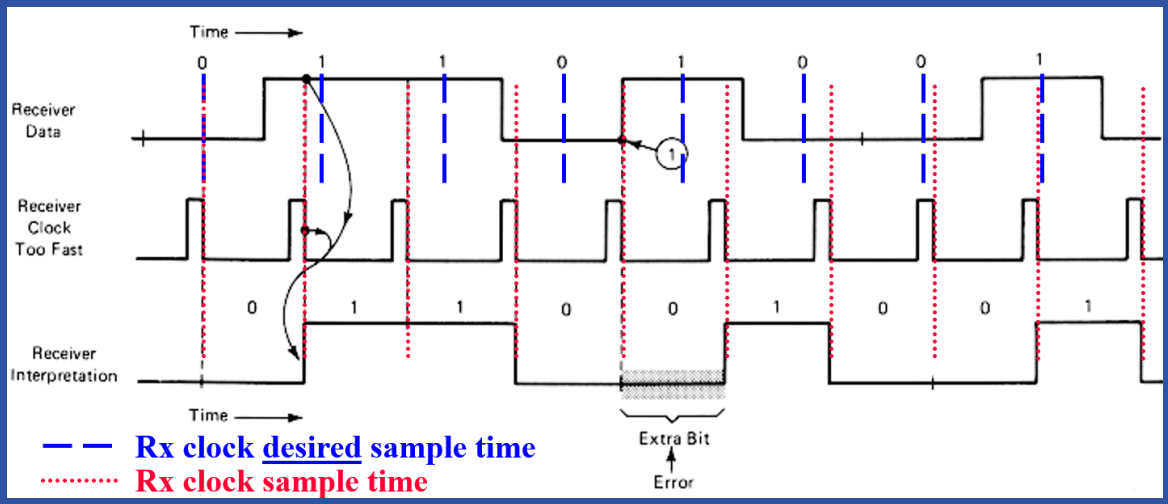
UART Transmission Time Example

Example: How long would it take to transmit 100 bytes of data via UART at **9,600 bps**? Assume **even parity** is used with **one stop bit**.

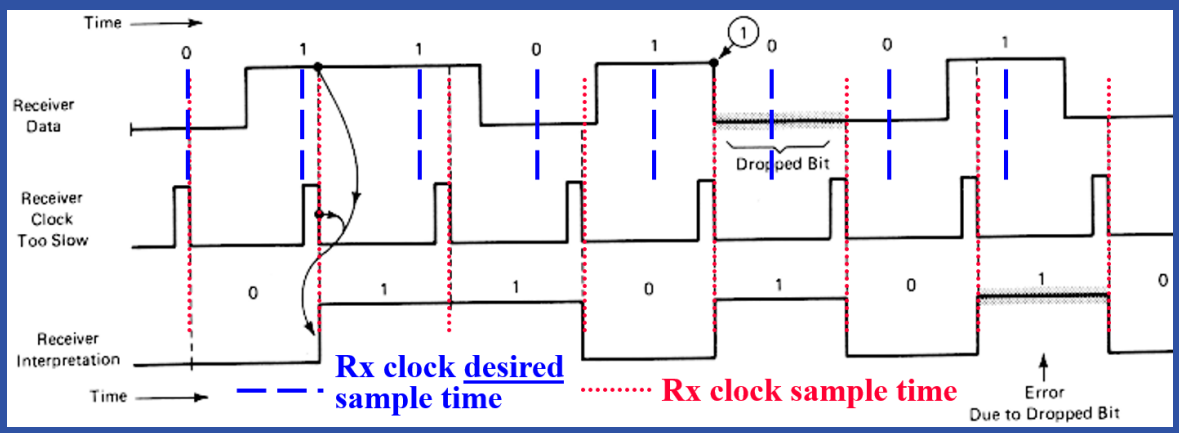


$$(100\ frames) * \frac{(1+8+1+1)bits}{1\ frame} * \frac{1\ sec}{9{,}600\ bits} = 114.5ms$$

Receiver Sampling - Underflow



Receiver Sampling - Overflow



Conclusion

- UARTs are commonly used in embedded applications for short and medium-length communication.
- They are simple and don't require a lot of connections to be fully functional.
- Having a good understanding of the way UARTs work is critical when trying to work with embedded systems in a timely manner.
- Being able to quickly adapt to a new project or system that uses existing UARTs is necessary for both legacy and modern embedded systems.