Chapter 4

Universal Asynchronous Receiver/Transmitted (UART)

Chapter Introduction

In this chapter we will cover the following topics:

- the UART Protocol
-

Universal Asynchronous Receiver/Transmitter (UART) is a simple asynchronous serial protocol commonly used for textual terminal communication; that is, a text-based user interface.

Chapter Objectives

In this chapter, students will learn:

- How to interpret UART exchanges and transactions

4.1 UART Preliminaries

A typical, bidirectional UART channel, shown in Fig. X connects only two peers that we will refer to as peer A and peer B and consists of only three wires:

- a wire used as a **data signal** to transmit data from peer A to peer B, which is connected between the transmit (Tx) output of peer A to the receive (Rx) input of peer B,
- a wire used as a **data signal** to transmit data from peer B to peer A, which is connected between the transmit (Tx) output of peer B to the receive (Rx) input of peer A, and
- a common ground wire.

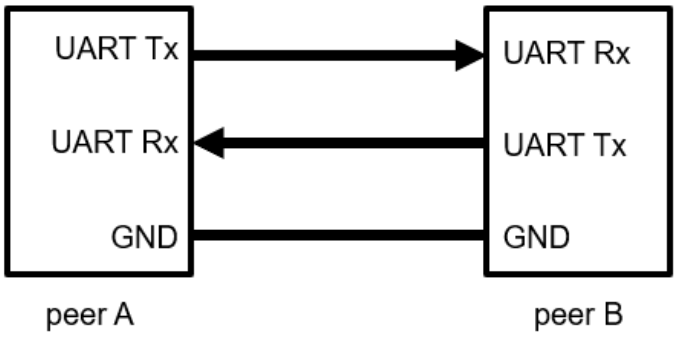


Fig. X: A UART channel

You will notice a few differences as compared to SPI:

- there is no clock wire, since UART is an asynchronous protocol,

- there is no equivalent to the slave-select wire, which signals that the channel is being used for communication because UART communication is signaled using the data wire (specifically using the **start bit** and **stop bit**), and
- there is a common ground wire, because SPI channels assume that all peers share a common ground using connections outside the channel.

In a UART channel, data bits are generally transmitted on each data signal from least significant bit to most significant bit, although this can be changed if both peers agree to using the opposite ordering.

4.2 UART Timing

UART channels rely on each peer having an internal clock for changing the state of the output signal and sampling the input signal. The frequency of these clocks must be pre-arranged by the designer before the channel begins operating.

Since there is no way to avoid sampling the data within the aperture time, UART channels typically use a slow clock from 9600 Hz to 19200 Hz, which is called the channel **baud rate**. This minimizes the probability that the receiver will become metastable.

The two peers using that channel are expected to always hold the states for its respective output data signal for the duration of each **bit period**, which is $\frac{1}{baud\ rate}$. For a 9600 baud channel, the bit period is $\frac{1}{9600\frac{bits}{second}} = 104\frac{\mu s}{bit}$.

4.3 UART Throughput

UART channels normally transmit 8 bits in each transmission, although like the baud rate, this is another parameter that can be selected by the designer prior to channel operation. In this case, you might expect that the channel will require 8 cycles to transmit 8 bits, but this is not the case, because each UART transmission includes additional bits for signaling the beginning and end of a transmission called the **start bit** and **stop bits**. Another extra bit, called the **parity bit**, is optionally transmitted to allow the receive to recognize any transmission errors (including errors caused by metastability in the receiver).

Because each transmission includes both data bits and the start, stop, and parity bits, the **throughput** of the channel, or the maximum rate at which data can be transferred over the channel, is not the same as the baud rate, but can be computed as:

$$throughput = \frac{data\ bits\ per\ transmission}{total\ bits\ per\ transmission} \times baud\ rate$$

Thus, if the baud rate is 9600 and there are 8 data bits, one start bit (note that there is always exactly one start bit), two stop bits, and one parity bit, then the throughput is computed as:

$$\frac{8}{8+1+2+1} \times 9600 = \frac{8}{12} \times 9600 = 6400\frac{bits}{second}$$

In a UART channel, each of the data signals operate independently. That is, there is no coordination between data flowing in each direction. In fact, UART channels can be operated with only one data signal to allow for one-way communication.

4.4 Start and Stop Bits

The UART signal wire idles at logic level high. In other words, the state of the signal wire is logic-1 when there are no transmissions actively occurring on the corresponding signal wire.

Unlike SPI, which uses a dedicated wire to signal the start of a transmission, UART does this by injecting a **start bit** into the data signal. Since the idle state of each data signal is logic-1, the start bit is signified by driving the data signal to logic-0 for one bit period.

Since the start bit signals the start of a transmission, the next *N* bit periods are expected to hold the *N* bits being transmitted, where *N* = the number of data bits for which the channel is configured, which like the baud rate must be pre-arranged between both peers connected to the channel.

After the data bits are transmitted, the sender may optionally send a parity bit, which we will describe below. After this, the end of the transmission is signaled with the **stop bit** or **stop bits**. The stop bit is normally a single bit but can be configured as two bits. When there is one stop bit, it is signaled when the transmitter drives the data signal to logic-1 for one bit period. When there is two stop bits, it is signaled when the transmitter drives the data signal to logic-1 for two bit periods.

Stop bits have two objectives. The first is to guarantee that the data signal exhibits exactly one transition (from logic-1 to logic-0) between every two consecutive transmissions. The second is to allow the receiver time to process the previous transmission. Two stop bits extends this processing time, although modern systems generally do not require extra time, it is most often the case the only one stop bit is used.

Fig. X shows the timing of a single data signal when using eight data bits, one stop bit, and no parity bits.

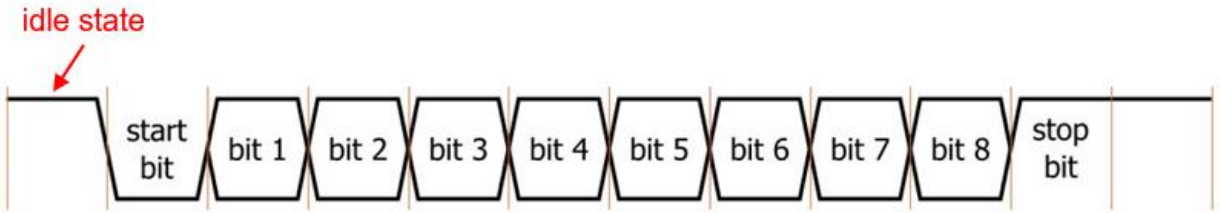
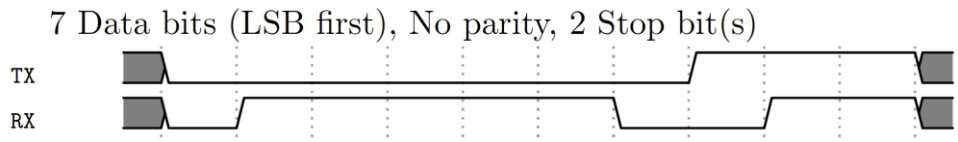


Figure X: UART data signal timing with one stop bit and no parity bits.

Exercise: In the following UART transmission, what bytes are exchanged? Assume that data bits are transmitted from most significant to least significant, that there are 7 data bits, two stop bits, and no parity bits.



Answer: TX: 1000000, RX: 0011111

4.5 Parity

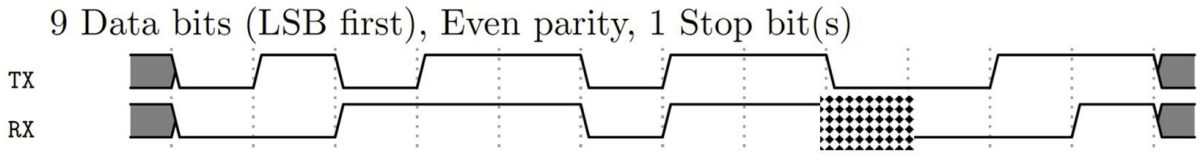
UART channels can optionally include a **parity bit**. If enabled, the parity bit is transmitted immediately after the data bits. The parity bit adds the ability for the receiver to detect certain types of transmission errors that might have occurred due to channel noise or a metastability condition at the receiver. The parity bit is set by the transmitter to force the total number of one bits transmitted between the start and stop bits to be even or odd, depending on which parity mode for which the channel is configured.

For example, if the parity mode is set to even parity, the parity bit is set to one if there is an odd number of bits in the transmitted data bits. This way, the total number of one bits within both the data bits and the parity bit will be even. Likewise, if the parity mode is set to odd parity, the parity bit is set to one if there is an even number of bits in the transmitted data bits. This way, the total number of one bits within both the data bits and the parity bit will be odd.

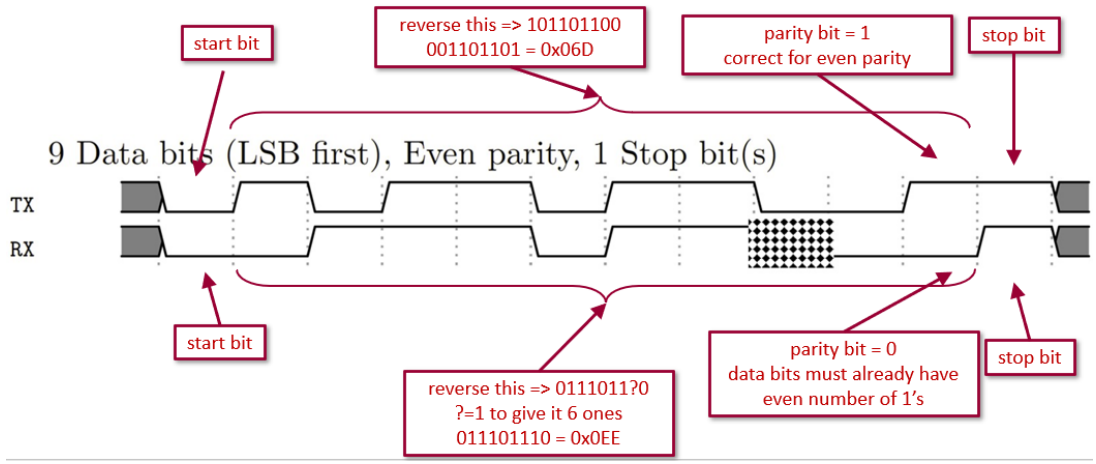
In this case, any single bit among the data and parity bits that is misinterpreted at the receiver as the opposite bit value will cause the receiver to detect a transmission error since the total number of one bits among the data bits and parity bit will not be even or odd as guaranteed by the parity setting. Note that parity does not protect against errors affecting the start and stop bits.

Additionally, if one of the data or parity bits is not received at all, which is sometimes called an **erasure**, the receive can infer its value based on which value will set the number of one-bits to be consistent with the parity configuration.

Exercise: In the following UART transmission, what bytes are exchanged? Assume 9 data bits, even parity, one stop bit, and the hidden bit period indicating an erasure.



Answer:



TX: 001101101

RX: 0X1101110, where X = 1, since there are 5 one-bits in the data payload and the parity bit is 0, so there must be one additional one-bit to make the total number of one-bits even,

---

Exercise: Assume a UART channel operates at baud rate 9600 bps and has 7 data bits, one parity bit, and one stop bit. What is its throughput?

Answer: $9600 \frac{total\ bits}{second} \times \frac{7\ data\ bits}{10\ total\ bits} = 6720\ \frac{data\ bits}{second}$

4.6 Chapter Summary

This chapter covered the fundamental concepts of UART, a bidirectional asynchronous serial protocol, including:

1. UART timing
2. UART throughput
3. UART start/stop bits
4. UART parity
5. Example UART transmissions