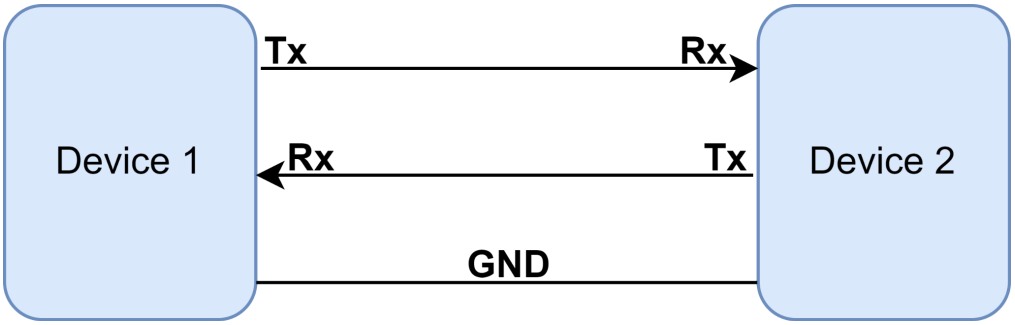
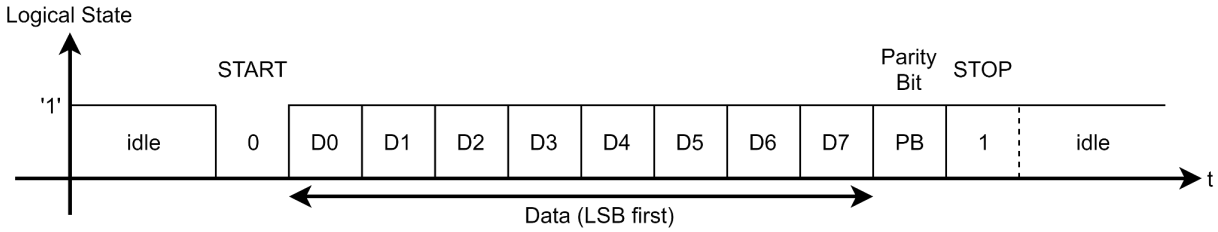
Lab 2 - UART



Implementation:

We will be implementing a Full-Duplex communication scheme (see above image). *Each device* will have a channel to send *frames*. Like in the previous lab, data is transmitted bit by bit. However, the details are different.
In asynchronous communication, the data is *framed*:



The frame implemented in this lab should use:

1. '0' as the START bit.
2. '1' as the STOP bit. The stop bit should be one "`BIT_TIME`" (1/50 sec like in the previous lab).
3. The Parity Bit (PB) should be Odd Parity (completing the number of ones in the byte into an odd number).
4. Data - 8 bits of data, the LSB sent first. Note that the number of bits will vary in the following labs, so make sure you can change the number of bits swiftly (by using defines, up to 12 bits).

You are required to create the following two functions. **Your work should transmit and receive indefinitely (non-stop)**:

1. **uart_tx()** - sends data to the other device in the above structure. After the transmission, the transmitter will not send information for a random amount of time. **Do not use any sleep/delay/while/for commands.**
2. **uart_rx()** - reads the channel pin, looking for the first '0'. If found, a transmission is in progress. Each bit should be sampled "`SAMP_NUM=3`" times. If the transmission is corrupt (wrong parity, bad bit, START ≠ 0, STOP ≠ 1), drop the frame and ignore the channel for the *rest of the frame time*.

You cannot use "Busy Waiting" in your implementation. Each call to "`uart_tx()`" is followed by "`uart_rx()`" (or vice-versa).

General Tips:

1. **Read all general tips; they might help you with your code, questions, and defenses.**
2. Some tips from Lab 1 are relevant to this lab; you may want to re-read them.
3. When the sampled bit's buffer (after masking) is not 0 or 0b1110, the bit is considered a "bad bit."
4. Don't wait for the entire frame to test the bits. You should test them "on the fly" (including START and parity).
5. For a modular code that works with any "`SAMP_NUM`," you can create a global variable named "`mask`" and save the corresponding binary mask into it.
   In this case, the mask is calculated *once* inside "`setup()`."
6. "rest of the frame time" depends on the current state and the number of received bits up to now.
7. **Your code should be well documented!**

Tools That May Help:

Simulator (login with BGU via Google account):
https://www.tinkercad.com

Arduino Reference:
https://www.arduino.cc/reference/en/

Odd Parity Calculator:
https://www.toolmenow.com/33/Odd-Parity-Calculator