# CAB202 Supplementary Material
## Serial Protocols and UART

## Overview of Serial Protocols

A serial protocol refers to any type of communication interface that allows two or more devices to communicate using a serial interface. A serial interface is any interface where the data been sent or received by a device is passed in sequentially (i.e. one bit after another). There are also parallel interfaces where all bits, for a piece of data, are passed in at the same time. Parallel interfaces will not be studied in this unit. There are a number of different types of serial protocols, with the following three the most predominant:

- USART / UART
- Serial Peripheral Interface (SPI)
- I2C (also known as Two Wire Interface)

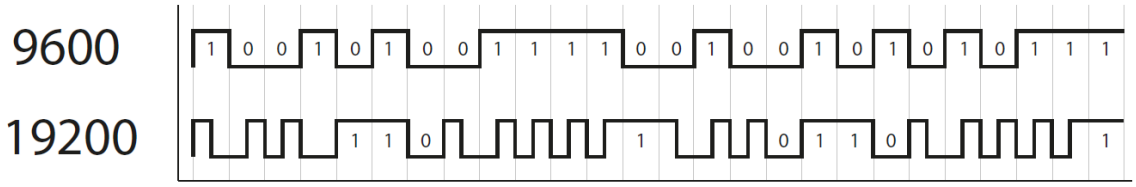
Each of these three protocols has their own unique method of communicating serial data. The protocol you will be exploring in this unit is the USART/UART interface which is generally the easiest to understand of the three. If you wish to read more on serial interfaces and the advantages and disadvantages of each, please go through the information on this website.

## USART/UART

USART and UART are similar to the extent that they are often confused, but there is a distinct difference between the two. UART (Universal Asynchronous Receiver Transmitter) only has two communication lines; a line for the transmission of data (**Tx**) and a line for receiving data (**Rx**). Alternatively, USART (Universal Synchronous/Asynchronous Receiver Transmitter) can also use a communication line that sends the clock signal when in synchronous modes. This process allows higher speeds in comparison with basic UART. In asynchronous modes validation of the data through the clock is not necessary because all communicating devices are told the speed at which data will be received (this speed is known as the Baud rate). While the Teensy is capable of performing communication in synchronous mode, this unit will focus on asynchronous communication. For a more detailed understanding of these communication methods, look through the data sheet, lecture slides, and other online literature.

### Baud Rate

The first key component of serial communication is the speed at which the two Teensy microcontrollers communicate with each other. This is known as the baud rate, and refers to how many **`bits`** are sent per second. It is essential that the two devices are communicating at the same speed. Consequently, baud rate mismatches are one of the most common issues encountered when debugging serial communications. A visual representation of baud rates and a table of the most commonly used baud rates are shown below:




*Baud rate mismatch example*

| | | |
|---|---|---|
| 4800 | 9600 | 19200 |
| 38400 | 56700 | 115200 |

*Commonly used baud rates*

The above figure illustrates the potential confusion arising from mismatched baud rates. Assuming the device is set to 9600, each bit it sends will line up between one of the vertical lines. However, if the computer side is misconfigured for 19200 baud, only some of the bits will fit nicely into these 'slots', and the Teensy will be unable to read most of the bits accurately, typically resulting in random gibberish. In other words, **if you are getting seemingly random characters from your Teensy it is most a result of incorrectly configured Baud rates!**

## Configuring the Atmega32u4 for UART

Like any hardware, before it can perform as desired it must first be initialised and configured correctly. Consequently, before using a microcontroller's UART functionality, it must first be configured with the appropriate set of commands. This is done by writing values to specific bits in certain registers. The basic UART registers you are required to configure can be found in the Atmega32u4 data sheet on Blackboard. It is **strongly recommended** that you completely read through Section 18 – USART (on pages 186-213). While it is not necessary to understand every single detail, it is important to get a broad overview of what is capable and understand exactly what you are doing in this prac. The following table contains a breakdown of the relevant sections in the data sheet:

| Register | Description | Data Sheet Pages |
|---|---|---|
| UBRRn | Register containing the Baud Rate, as determined by the formula on page 189 in table 18.1 in the datasheet | 209 |
| USRnA | Used to check to see if a transmission or receive is complete, as well as some other error checking procedures. | 205 |
| UCSRnB | Used to enable and disable the UART receiver and transmitter, as well as interrupt vectors | 206 |
| USRnC | Used to configure the UART protocol (i.e. number of Data Bits, Parity and Stop Bits) | 207 |
| UDRn | A buffer that holds the data that needs to be transmitted, or the data that was just received | 205 |