Hardware Communication Protocols: UART, I2C and SPI

Electronics (Engn/Phys 208), winter 2019. Last modified: 25 Feb 2019

Serial/UART:

What *two* ***hardware lines*** required to implement serial communication?

What is ***synchronous*** vs. ***asynchronous*** communication?

What is the ***data frame***?

What are the ***start bit*** and ***stop bit*** used for?

What is the purpose of the ***parity bit***?

What is ***baud rate***, and how is it quantifiably different from the number of actual data bits/second transmitted/received?

What is a UART ***hardware buffer*** and what does it do?

What is ***bus contention***? How is it limiting and often problematic in terms of hardware connections?

https://learn.sparkfun.com/tutorials/serial-communication (good primer)

https://www.allaboutcircuits.com/technical-articles/back-to-basics-the-universal-asynchronous-receiver-transmitter-uart/ (another short and sweet primer)

I2C (Inter-Integrated Circuits):

What are ***SDA*** and ***SCL*** lines? What two fundamental pieces of information are transmitted?

What does ***open-drain*** mean? Where does it come into play with I2C hardware connections, i.e., what additional connections might you have to make between a host microcontroller and peripheral device?

Why is the open-drain configuration critical for I2C to work properly?

How does the ***communication protocol*** work, in general? Carefully inspect the timing diagram and explain what chunks of data are transmitted, in what order, and whether it is MSB or LSB first.

What are the ***start*** and ***stop conditions***?

What are the ***ACK/NACK bits*** used for?

How can multiple devices be connected to a single I2C port? What are potential limitations to consider?

https://i2c.info/ (short little primer, covers the basics)

https://learn.sparkfun.com/tutorials/i2c/all (good tutorial all-around, covers hardware implementation and timing diagram)

https://www.nxp.com/docs/en/user-guide/UM10204.pdf (straight from the horse's mouth, NXP's full guide to I2C protocols)

https://www.i2c-bus.org/i2c-primer/ (full on, super detailed information)

https://hackaday.com/2016/07/19/what-could-go-wrong-i2c-edition/ (covers common pitfalls with I2C)

https://learn.sparkfun.com/tutorials/raspberry-pi-spi-and-i2c-tutorial/introduction (RasPi tutorial)

https://www.pjrc.com/teensy/td_libs_Wire.html (Teensy I2C Wire library)

SPI (Serial Peripheral Interface):

What are the 4 lines of a ***SPI port***, and what is the purpose of each one?

How does the communication protocol work in general? Carefully inspect the timing diagram as to the timing of ***CS (SS), MISO (SDO), MOSI (SDO), SCK.***

Is the chip select line typically ***active high*** or ***low***?

What is the typical maximum ***clock rate*** in units of MHz?

What is the role of ***clock polarity*** and ***clock phase***—what are the possible options for each?

How can multiple devices be connected to a single SPI port?

https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html (nice intro, relatively easy to read)

https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi (simple, but no simpler intro)

https://hackaday.com/2016/07/01/what-could-go-wrong-spi/ (covers common pitfalls with SPI devices)

https://www.pjrc.com/teensy/td_libs_SPI.html (Teensy SPI library)