Follow up from yesterday

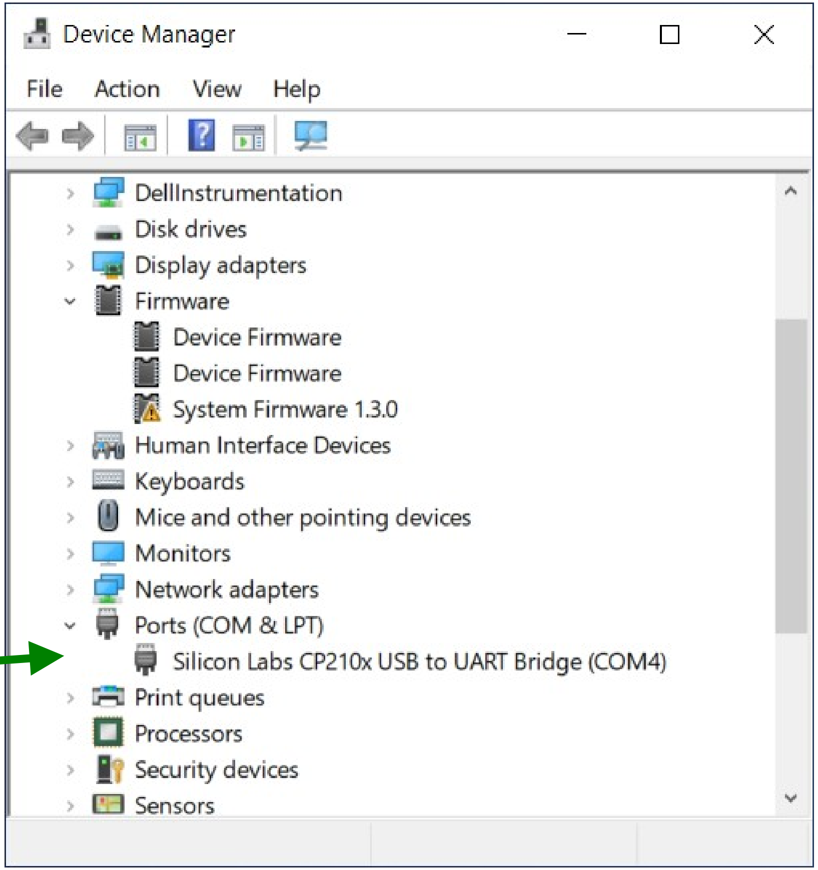
- Follow up on Exercises
  - Please read the documentation on the Web for the exercises
  - Go step by step and try to understand the steps: Otherwise you will waste a lot of time…

!!! GOING SLOWLY IS FASTER !!!

- Always try to understand commands and never blindly copy them. Ask if you do not understand the options
- Always read the output of the commands which indicate (more or less) well the problems

Follow up: exercises

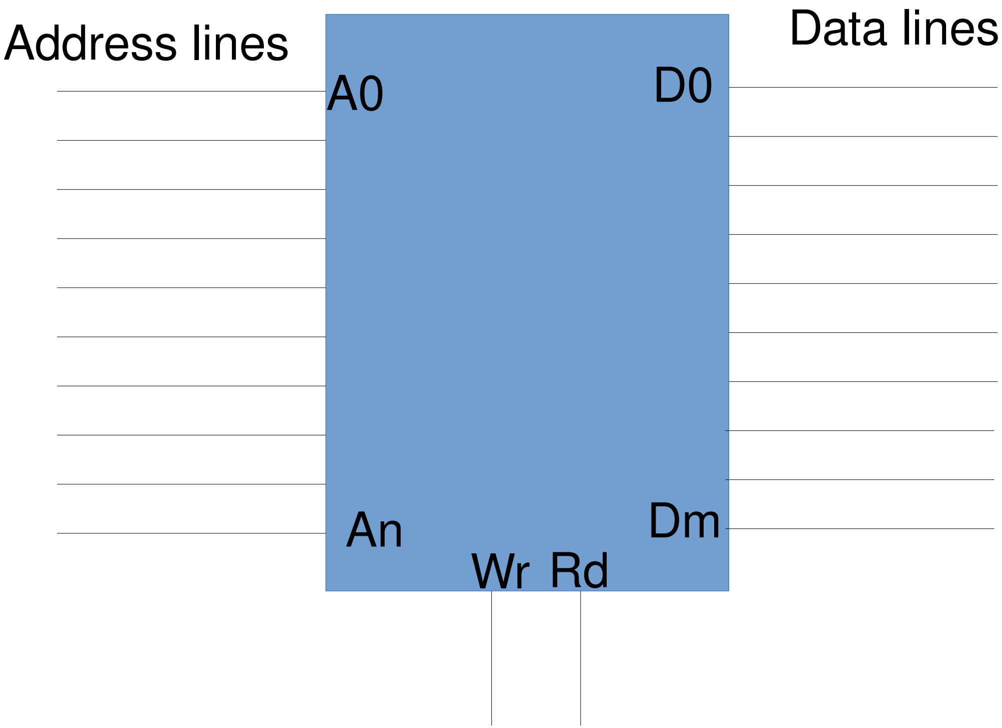
- Windows users had to install the driver for the USB to serial chip
  - (You remember from our lectures: “real” computers need “drivers” for the Peripheral devices)
  - The driver needs to handle a CP210x chip from “Silicon Labs”. (This is the chip soldered on the board to convert the USB stream to an UART stream)
  - You can download the driver here:
  - https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads
    Or directly from the website of the course (I added it to the references of the page)
  - Once installed the connected ESP should show up like this (Note down the COM port!):



- On Mac OS the device file is not called /dev/ttyUSB0 but /dev/tty.usbserial (or similar)

Follow up : Lookup Table

- A lookup table is a memory: 2^n words which are m bits wide. In a TLB n=m



Follow up: look up table

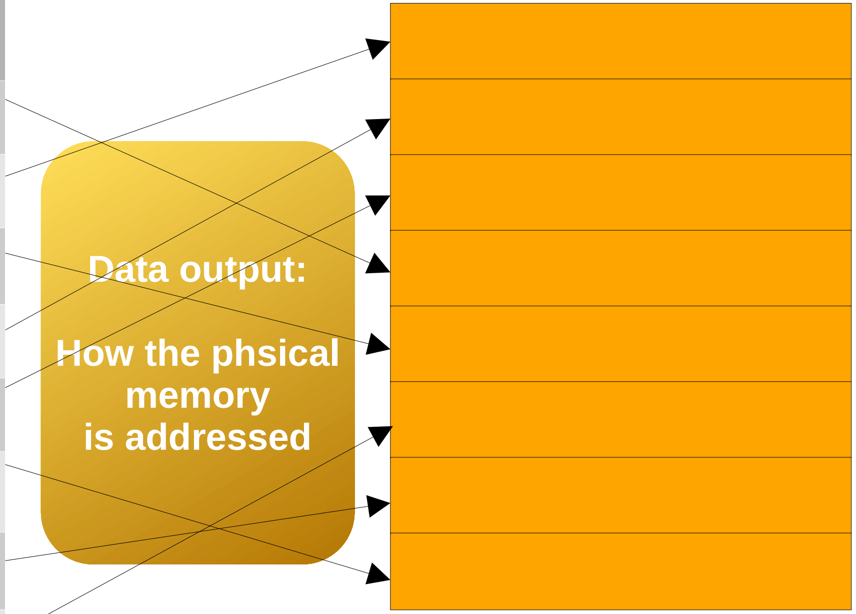
- Imagine a memory with the following contents

8 pages or frames

Adress side:

How the CPU addresses the memory

Adress of Lookup memory	Contents of Lookup Memory
0	3
1	0
2	4
3	1
4	2
5	7
6	6
7	5



Microcontrollers

Part II

Peripherals: UART, I2C, SPI

(Christoph.Schwick@cern.ch)

Serial interfaces : UART, I2C, SPI

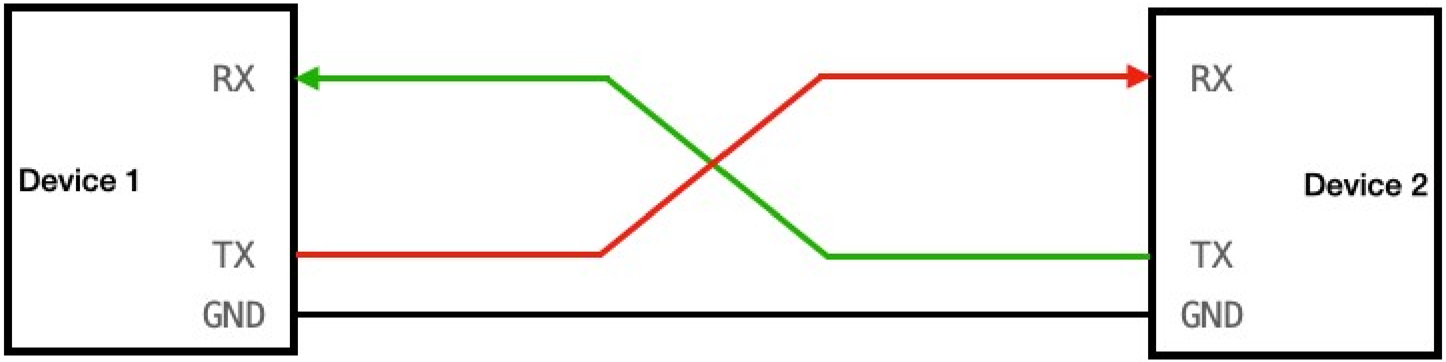
- Requirements for communication among chips in microcontroller systems:
  - Simple protocols (limited use of CPU power or hardware components)
  - Robustness (should work in cheap electronic environment)
  - Use few hardware resources (as little pins as possible)
    - Serial protocols are preferred: Send one bit after the other instead of entire bytes, otherwise we need to handle many signals in parallel
  - Consider speed: A display needs to have more data transferred than a temperature sensor
  - Consider use case:
    - Communication on the same board from chip to chip: controlled electrical requirement → allows faster speed (SPI and I2C)
    - Communication to the “outside world”: longer cables, interface different electrical environments → to make these connections robust you need to sacrifice speed (UART)

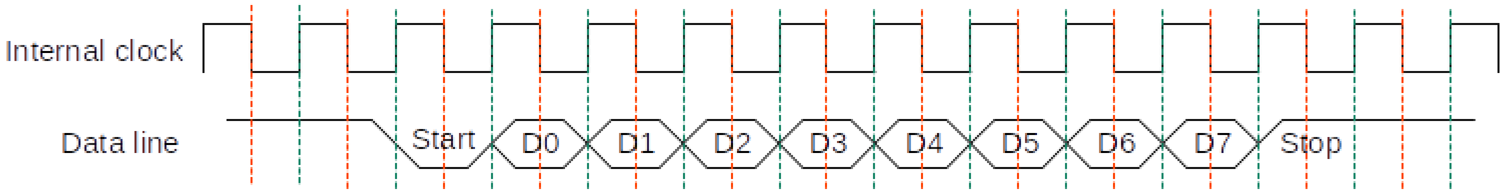
UART serial interface

- You know this from the previous lecture series (FPGA lectures)
- Defines the protocol (how bits are serialised and transferred), but not the electrical implementation (voltage levels, pullup resistors or not, ...)
- Features
  - Only 2 lines used (in the simplest case)
  - Full duplex (data transfer in both directions at the same time)
  - No need to transfer clock
  - As a consequence relatively slow (nowadays 115200 bits per second is typical)
    - In older times 9600 baud was the standard speed and still is the standard in rough environments
- Implementation “RS232” (on old computers)
  - High voltage levels → very slow but very robust

- Usage
  - Due to its robustness ideal to connect different systems also in rough environments (i.e. computer to microcontroller, measurement instrument to computer)

UART: functional diagram



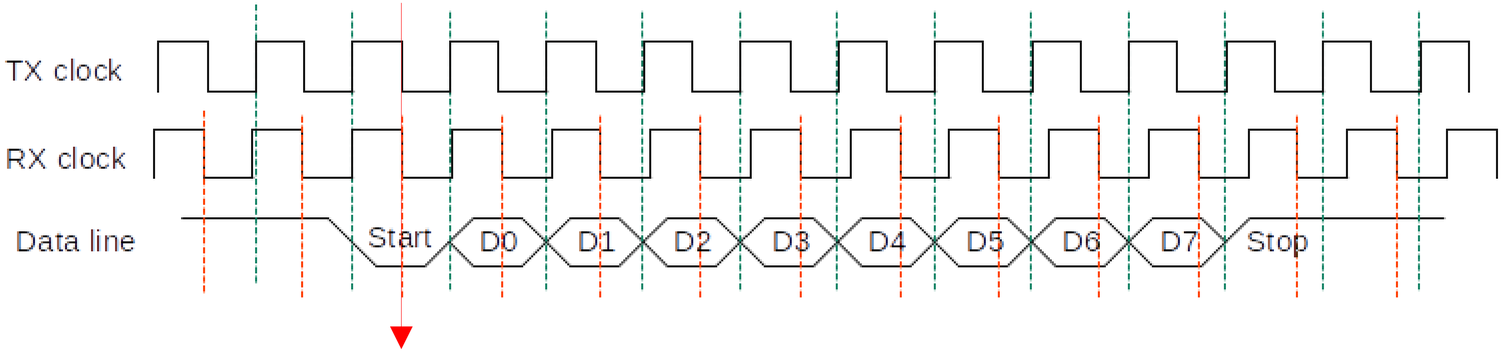


Sender changes level on data line on rising clock edge

Receiver samples data line at falling clock edge

UART: robust against diverging clocks

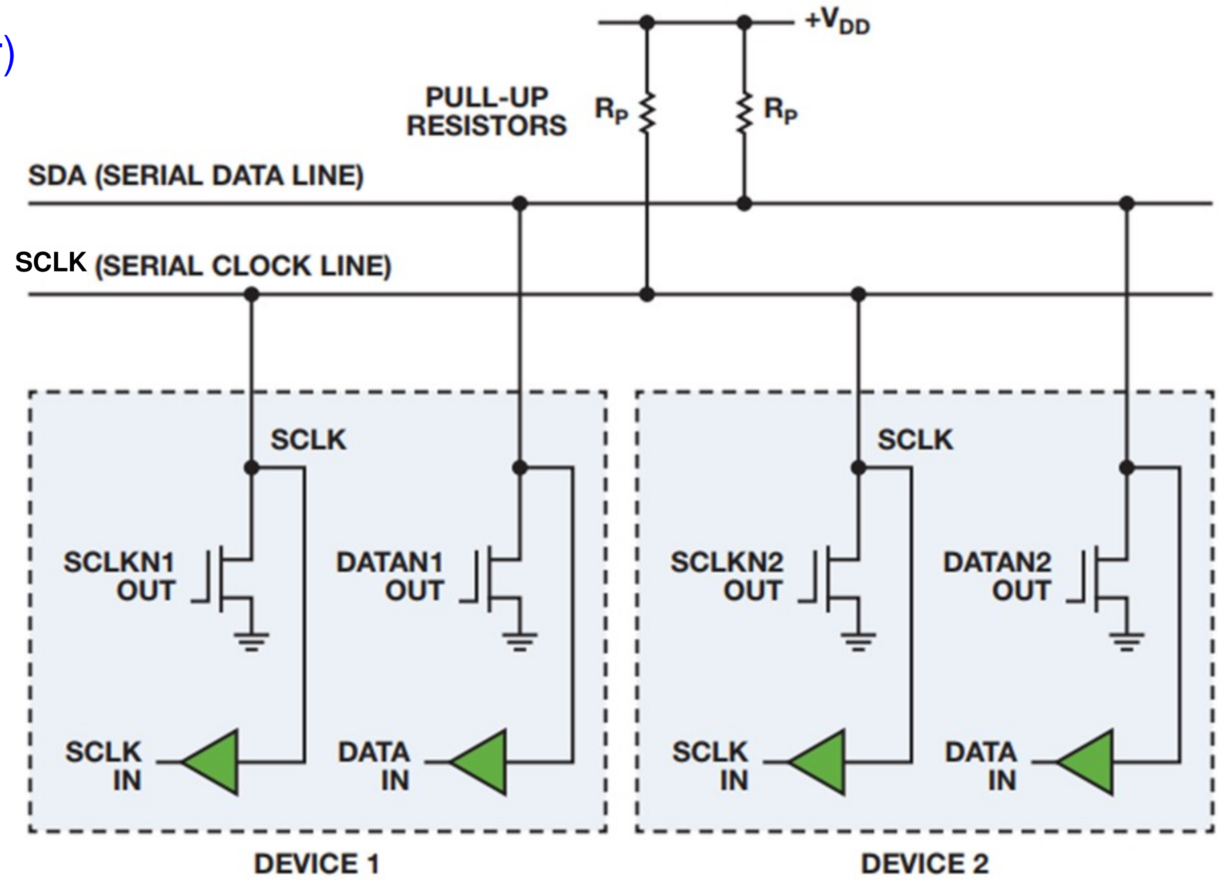
- Sender and receiver need to use an "approximately" equal clock.
- Robust against small clock differences due to re-synchronisation after each data word (stop and start bits)



UART protocol is tolerant of clock differences: clocks are re-synchronised at each Start bit. Clocks only need to be "more or less" in sync for 9 bits.

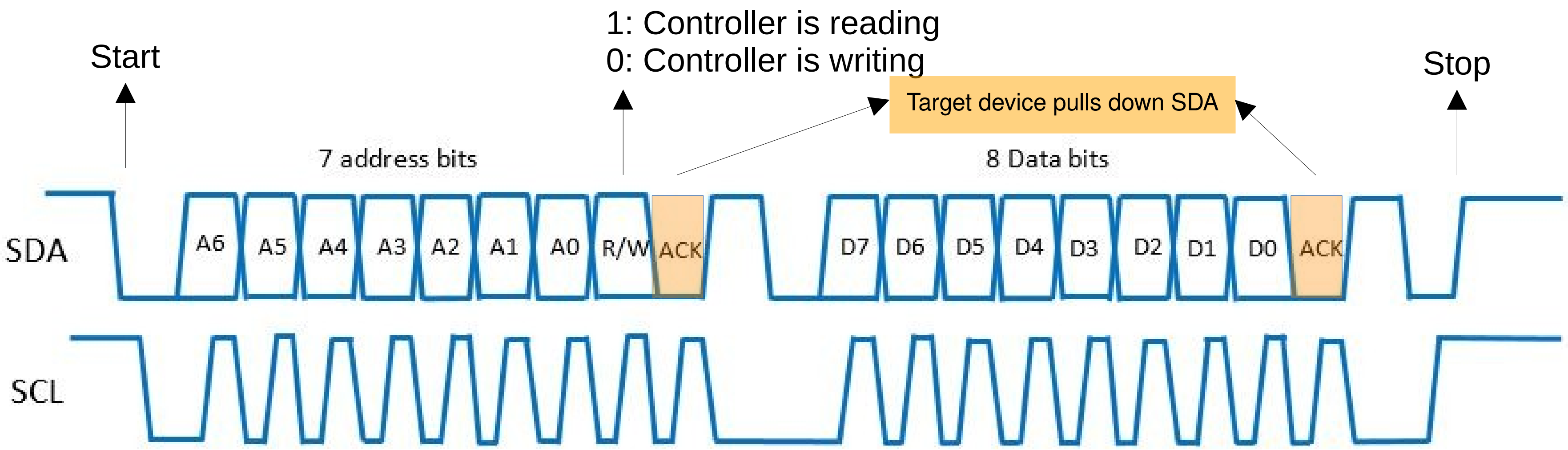
I²C: Communication to other chips
(Inter Integrated Circuit)

- Designed to connect chips on the same PCB (microcontroller to on-board sensor)
- Uses only 2 wires to connect many different chips
- “Reasonable” speed (up to 3.4Mbit/s maximal)
- Half duplex (data transfer in one direction at a time)
- Addressing scheme
  - Multiple controllers and targets
- Speed limitation
  - Pull up resistors
  - Bus with multiple taps (no termination)



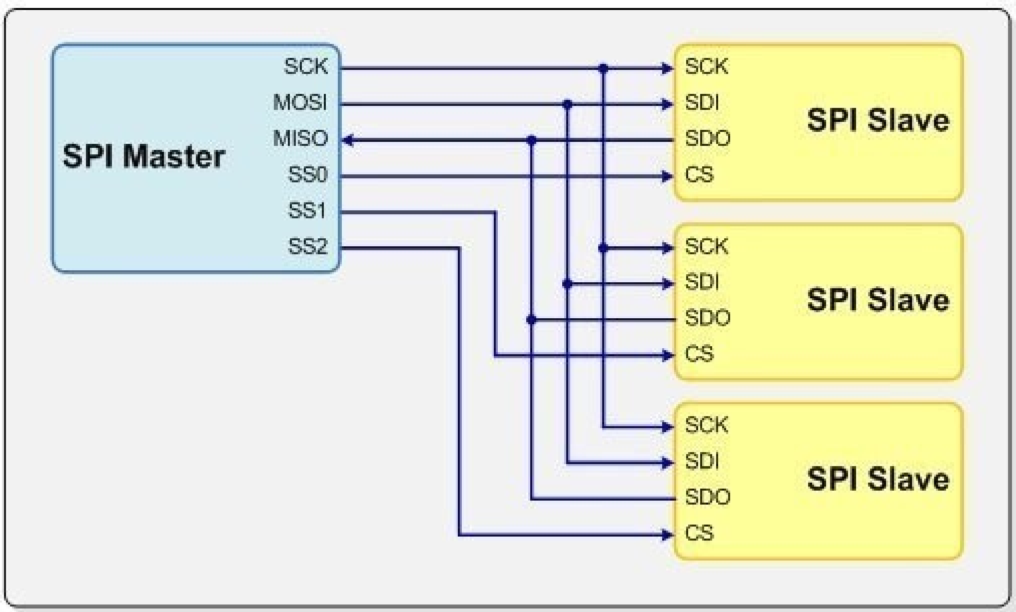
I2C communication protocol

- Each communication is initiated by the controller which “talks” to a target
- Communication is using frames: a frame is the transfer of a byte
- Each frame has to be acknowledged (or not, in case of problems...)
- Communication always starts with an address frame (selecting the target device)
- SDA only changes during SCL=’0’ : exception: Start and Stop condition



SPI communication

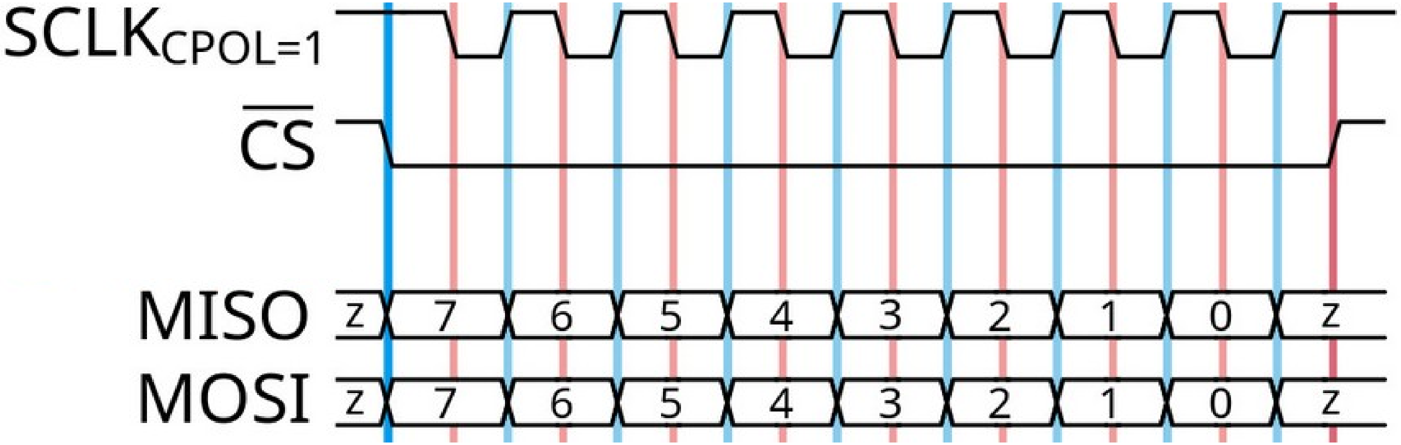
- Another protocol for “on board” communication between 2 devices
  - Full duplex in the variant using 3 wires (clock, 2 data lines for both directions)
  - Multiple devices can be attached to the same BUS: in this case one additional select line per device is needed to select the “slave” device which “talks” to the (one and only) controller or master:
  - NO addressing scheme!
- SPI is faster than I2C (up to 100Mbps)
  - No pull-up resistors but low and high levels are driven actively by drivers
- SPI is the simplest protocol
  - It only defines how bits and bytes are transferred



SPI data transfer

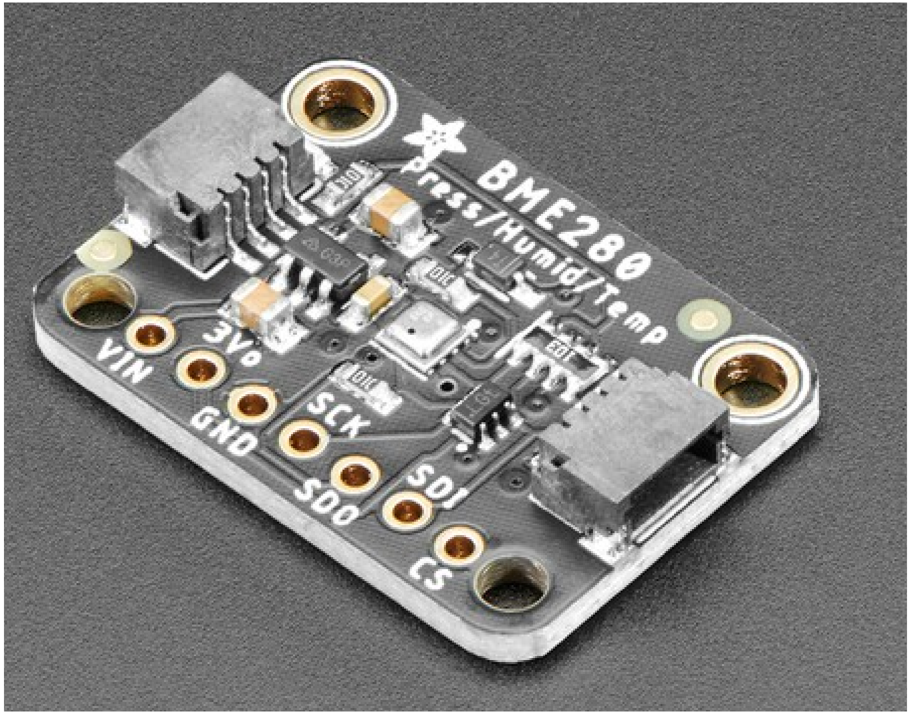
- The controller must activate the Chip Select line before the transfer. At this time also the first data bit is put on the data line by the sender. From then on:
  - The Sending side has to change the bits on rising clock edge
  - The receiving side samples the data line on the falling clock edge

- There exist 4 variants with different clock polarities (idle clock ='1' or '0') and Phase relations (when to change the data lines and when to sample them)



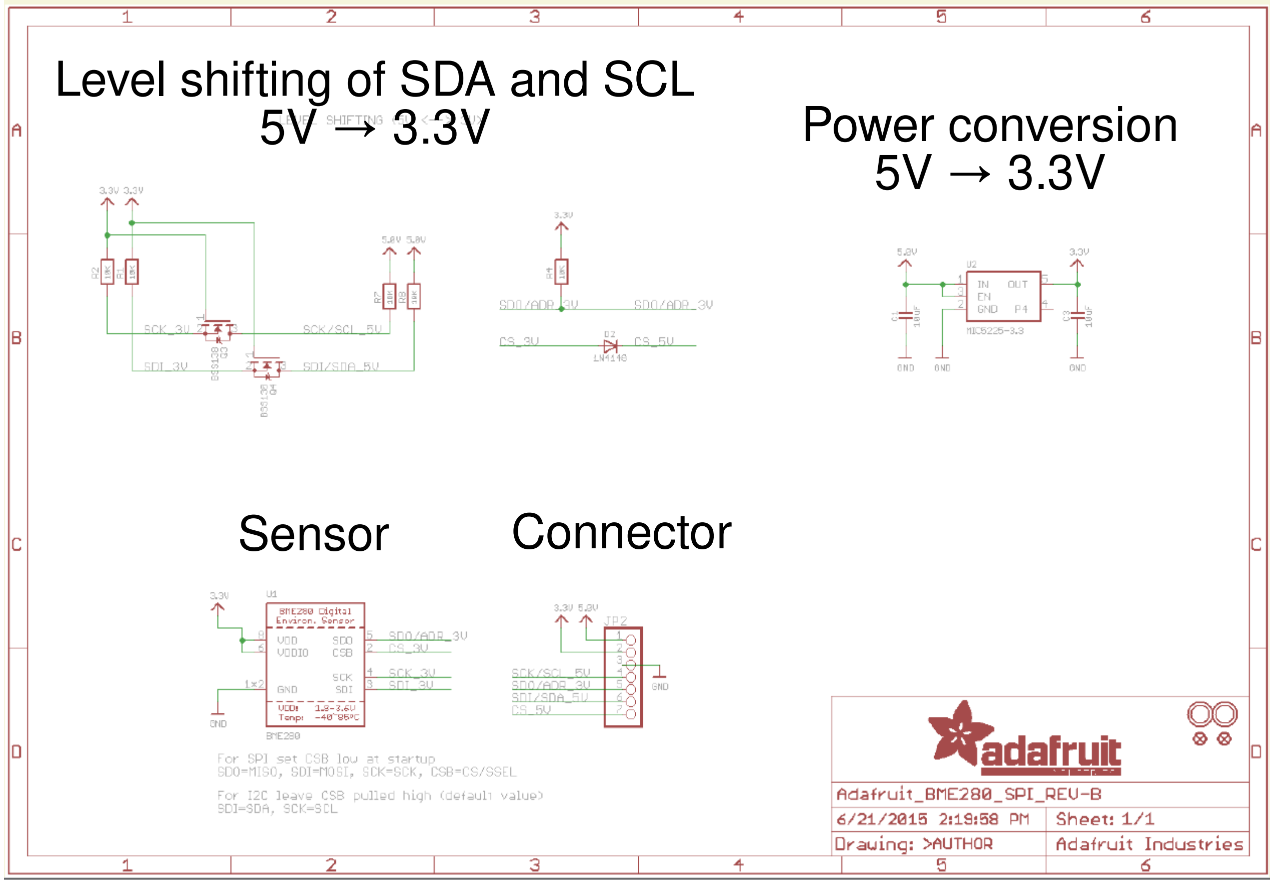
I2C Sensor

- BME 280 (Bosch)
  - Temperature, Humidity, Pressure
- Can be read with I2C or SPI
  - We will use I2C
  - I2C or SPI is selected at power up (See circuit diagram how this is done)
- Sensor is calibrated in factory
  - The calibration constants measured in the factory are needed to calculate correct sensor values. They are written into the chip and can be read out via I2C
  - The three different sensor values can be read out after a measurement cycle.
- Accuracy of the sensor
  - The accuracy of the sensor can be selected via different oversampling factors (multiple measurements are done and the mean is calculated to reduce noise)
- Formulas to calculate final values are provided in datasheet



BME280 circuit diagram

- The BME280 can be operated with maximal 3.6V.
  - A Power Converter from 5 to 3.3V allows to operate the breakout board with 5V.
  - For this also the SDA and SCL lines need to be converted to 3.3V . This is done with a “level shifter”



“Homework”

- Study the data sheet of the SH1107 controller of the OLED display we want to use in the next exercise. You find it as the first reference in :
  - https://microcontoller-course.web.cern.ch/Exercises/OLED_display/OLED.en/
- On the wepage above you find the paragraph “Important sections in the data sheet” which allow you to select the essentials of the data sheet.