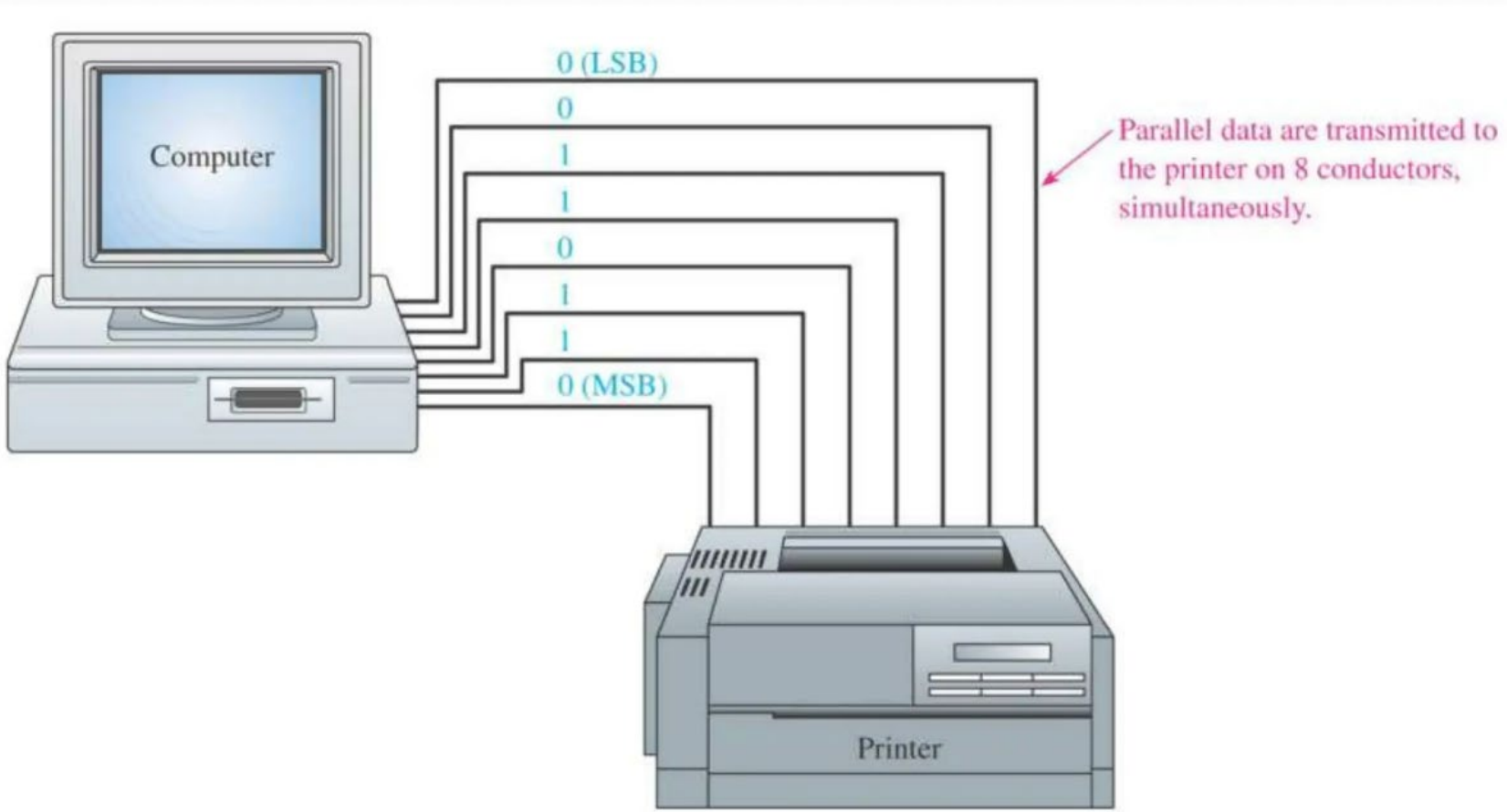
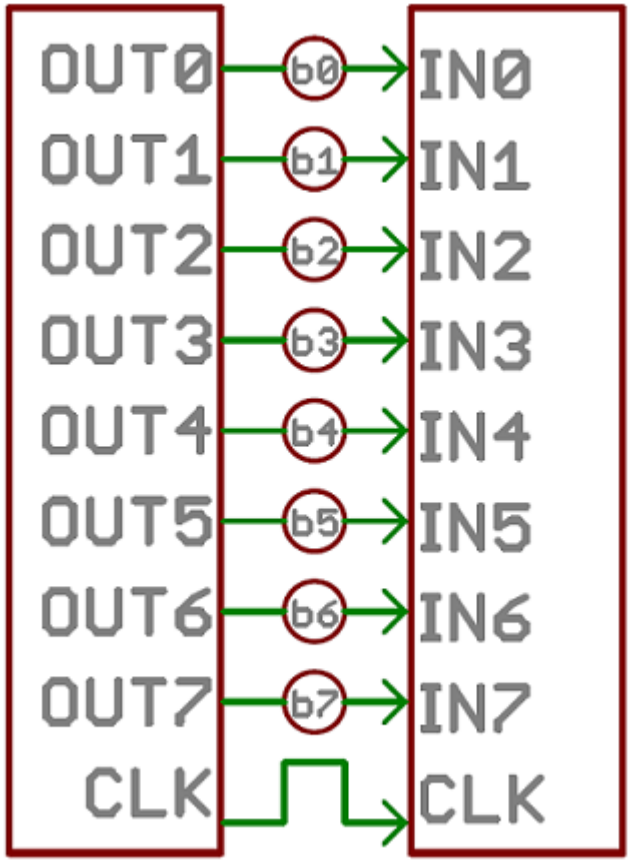
P524: Survey of Instrumentation and Laboratory Techniques

Week 4: Serial Communications

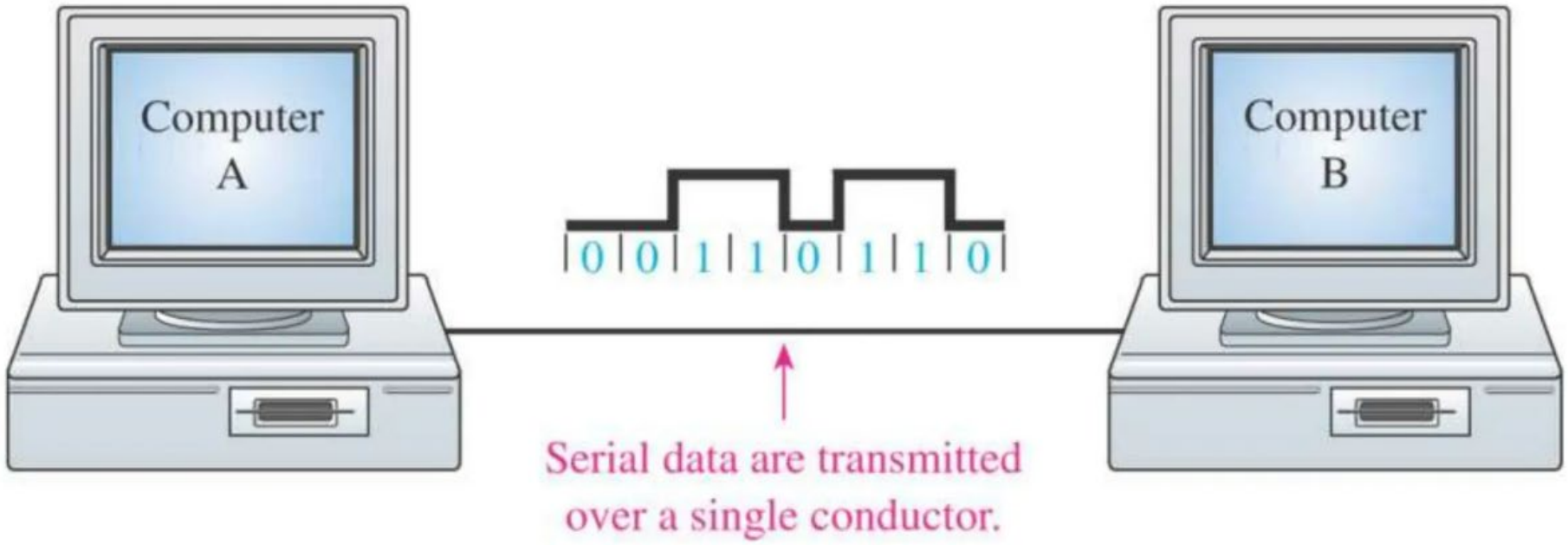
9/10/2024

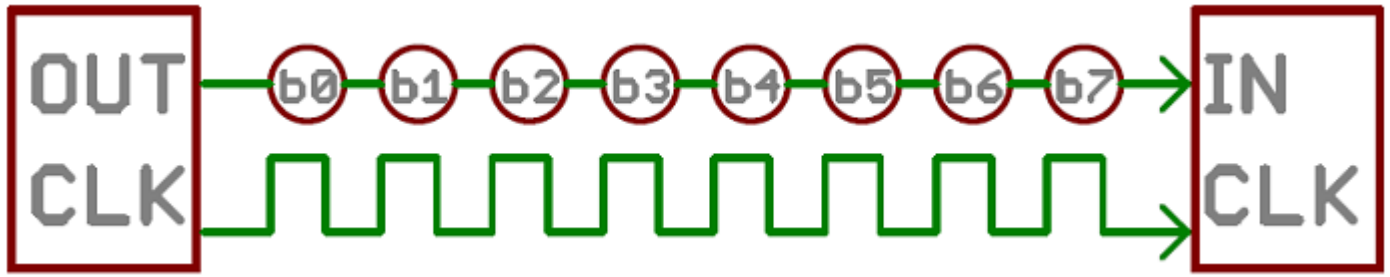
Parallel communication between a computer and a printer



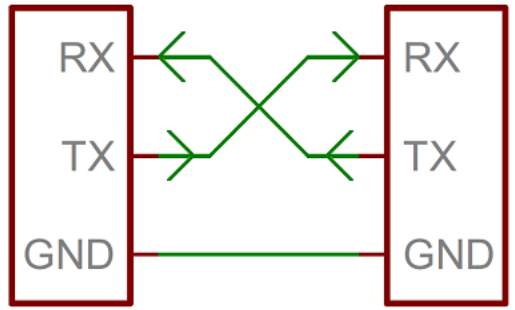


Serial communication between computers





Asynchronous serial protocol



Hardware wiring:
E.g.: Ultimate GPS, Bluetooth, serial LCD

- Baud Rate (bits-per-second): Baud rates can be just about any value within reason. The only requirement is that both devices operate at the same rate. **9600 bps, or 960 bytes per second** for non-critical applications. Other "standard" baud are 1200, 2400, 4800, 19200, 38400, 57600, and 115200.

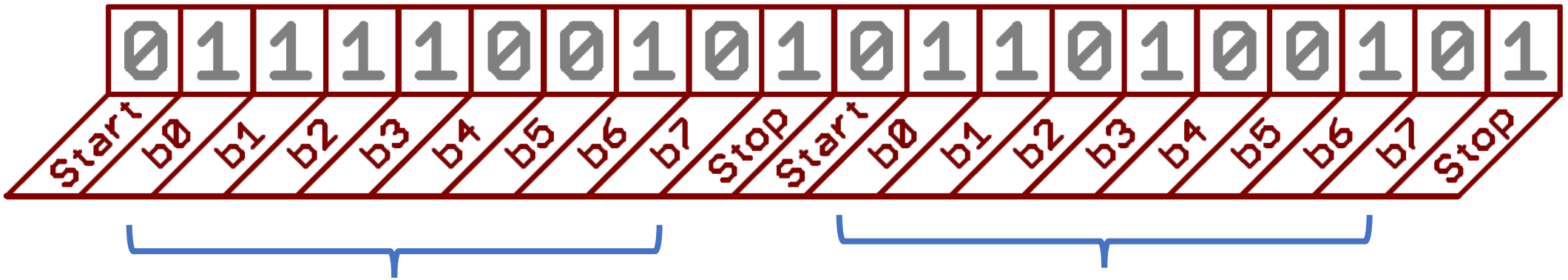
- Framing the data: Each block (usually a byte) of data transmitted is sent in a packet of frame of bits:

- Synchronization bits: The start and stop bits mark the beginning and end of a packet.
- Parity bit: sum 5-9 bits of data byte: even/odd for error checking

9600 8N1 Protocol: 9600 baud, 8 data bits, no parity, and 1 stop bit

one of the most commonly used serial protocols.

A device transmitting the ASCII characters 'O' and 'K' would have to create two packets of data:



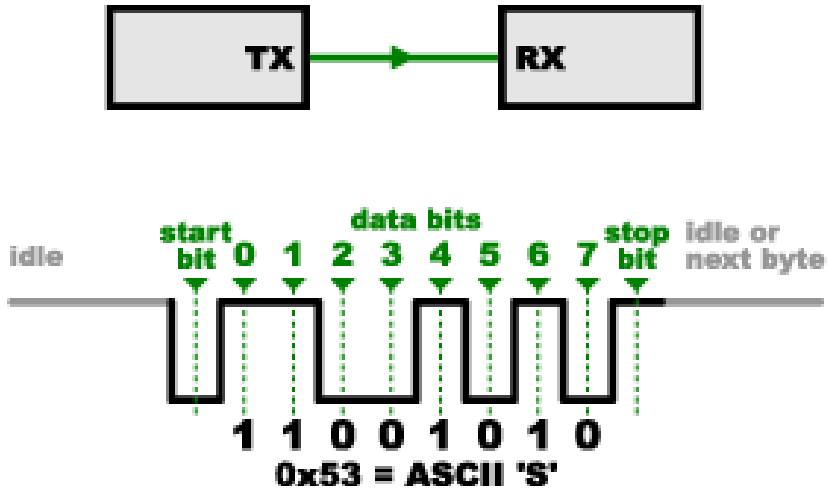
The ASCII value of *O* (that's uppercase) is 79, which breaks down into an 8-bit binary value of `01001111`.

K's binary value is 75, or binary: `01001011`.

Endianness: Is data sent most-significant bit (msb) to least, or vice-versa? If it's not otherwise stated, you can usually assume that data is transferred **least-significant bit (lsb) first**.

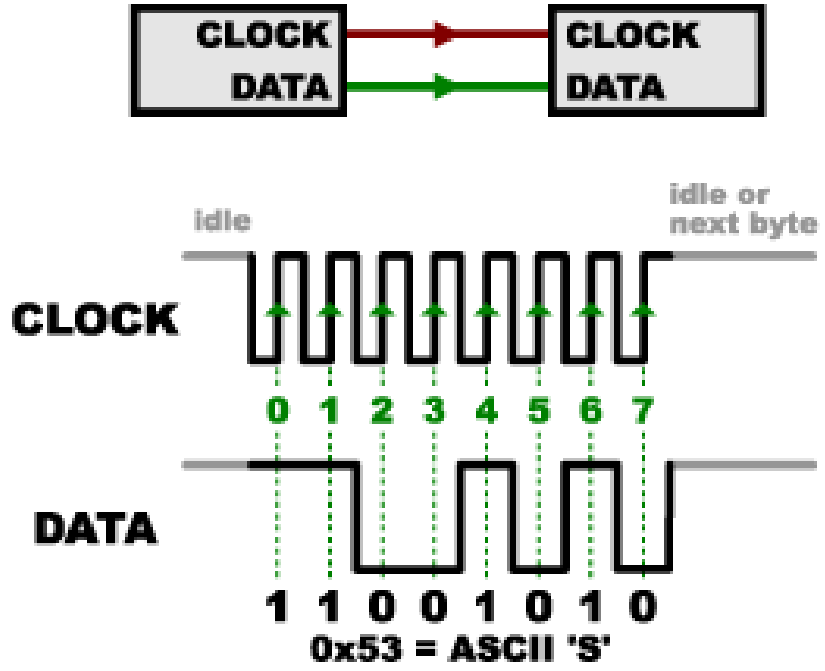
Asynchronous vs Synchronous serial protocols

Asynchronous:



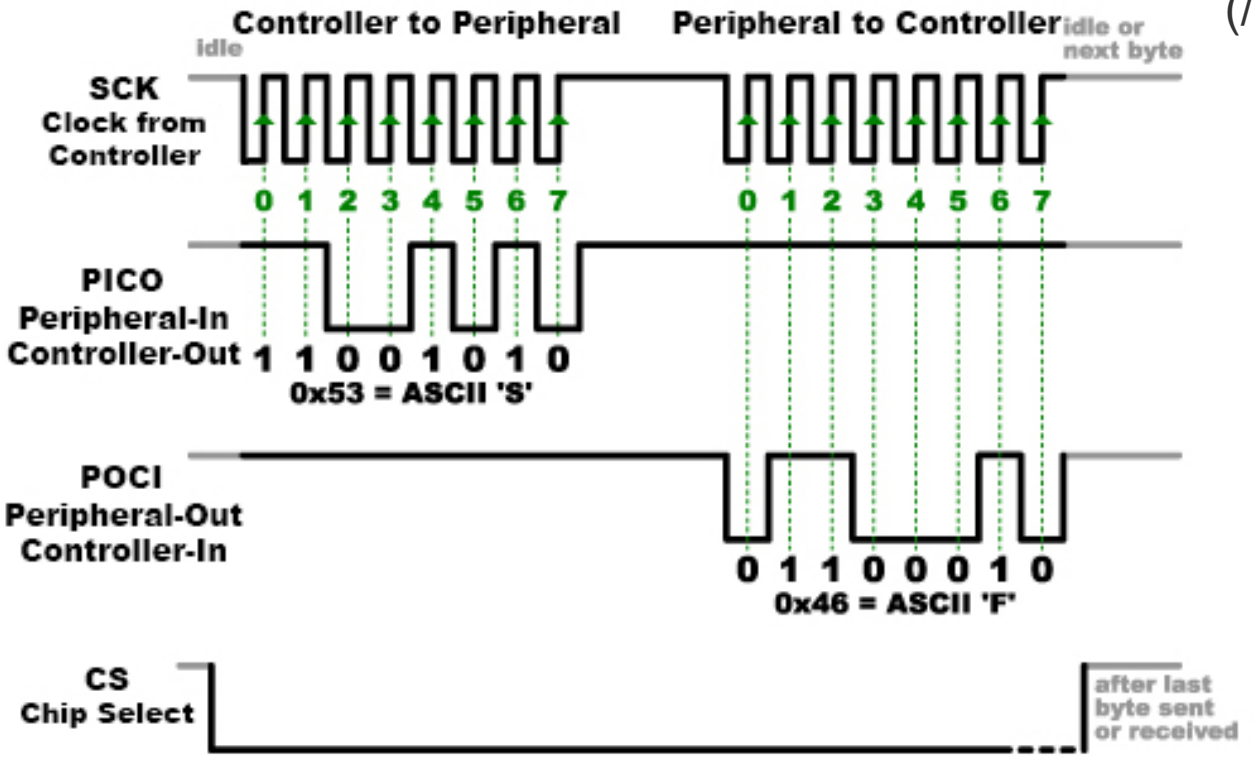
The start bit is always indicated by an idle data line going from 1 to 0, while the stop bit(s) will transition back to the idle state by holding the line at 1.

Synchronous:



- Because the clock is sent along with the data, specifying the speed isn't important
- Save the start and stop bits

Serial Peripheral Interface (SPI)



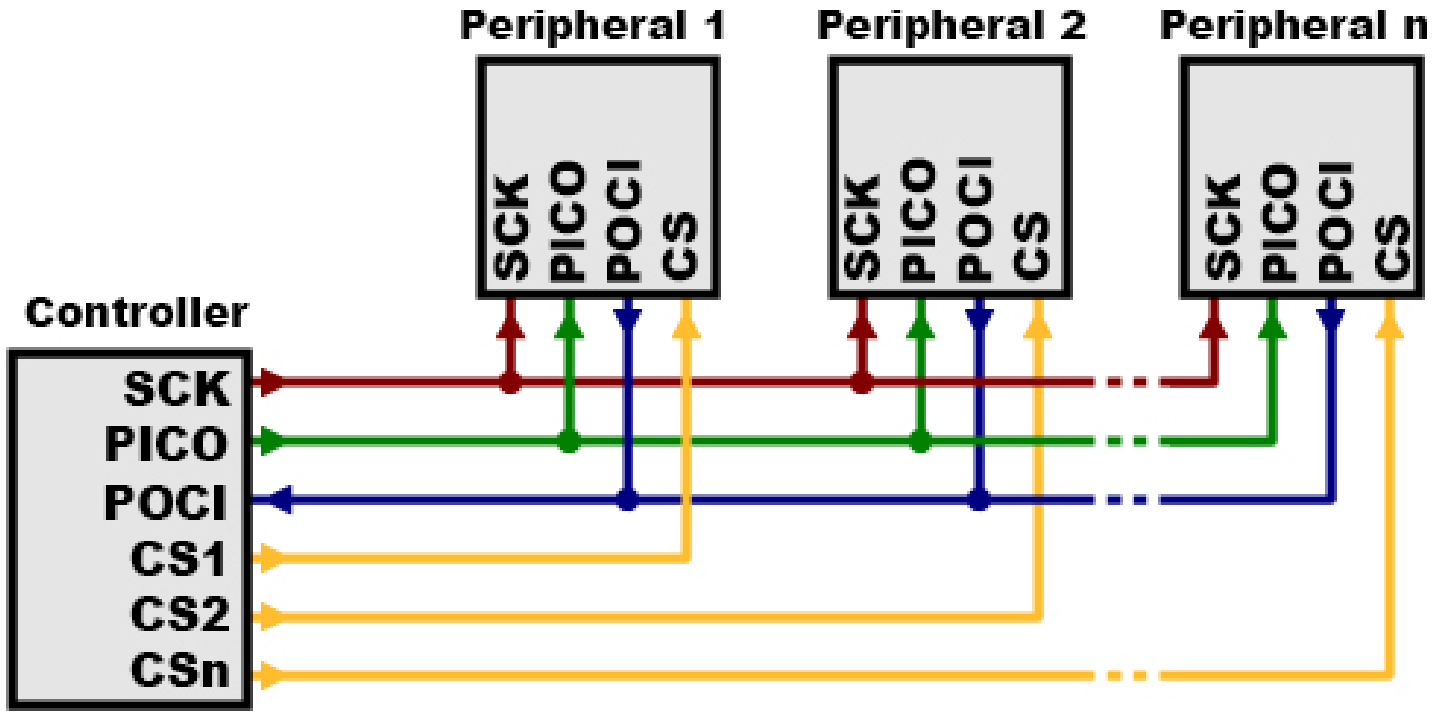
SPI can operate at extremely high speeds: ~ 10MHz.

This speed may be too fast for some devices. In the Arduino SPI library, the speed is set by the setClockDivider() function, which divides the controller clock (16MHz on most Arduinos) down to a frequency between 8MHz (/2) and 125kHz (/128).

Common Name	Replaced Name	notes
Master	Controller	
Slave	Peripheral	
MISO	POCI	Peripheral→ Controller
MOSI	PICO	Controller→ Peripheral
SS	CS	Chip select

SPI is good for high data rate **full-duplex** (simultaneous sending and receiving of data) connections.

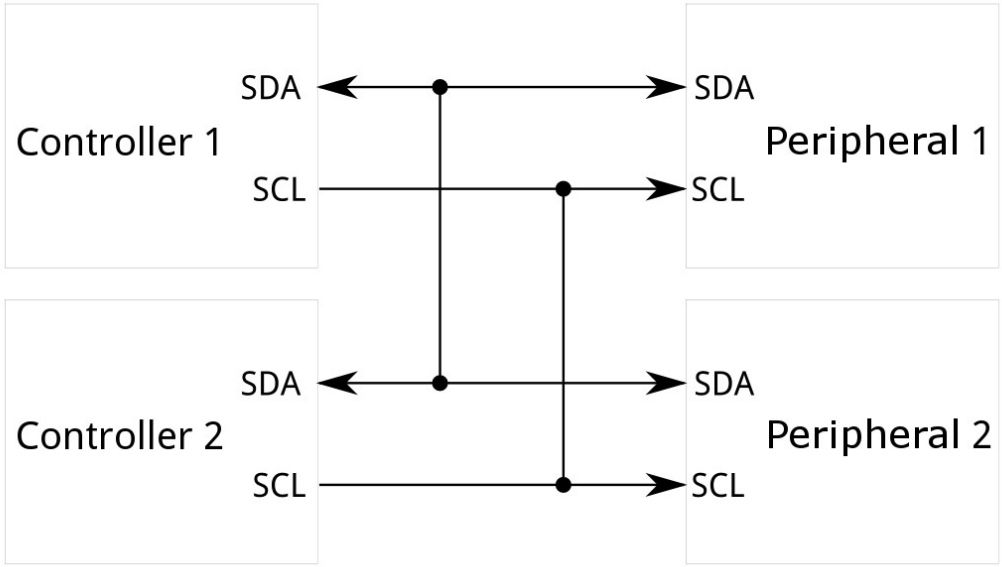
Multiple Peripherals



Shared lines for the clock, data in, data out,
But requires a separate ‘Chip Select’ line for each peripheral.

Inter-Integrated Circuit (I2C)

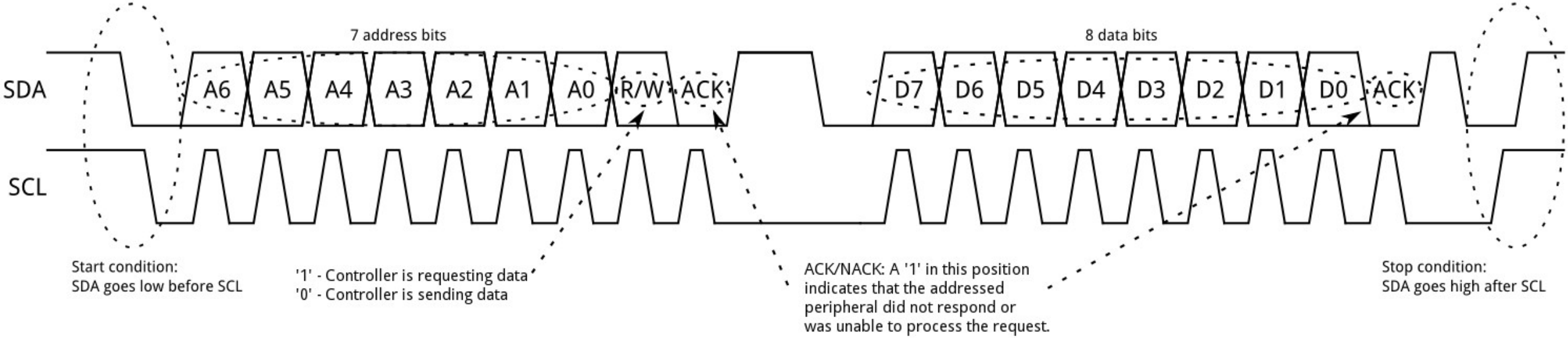
- I^2C requires a mere two wires, like asynchronous serial, but those two wires can support up to 1008 peripheral devices.
- Unlike SPI, I^2C can support a multi-controller system, allowing more than one controller to communicate with all peripheral devices on the bus (although the controller devices can't talk to each other over the bus and must take turns using the bus lines).



- Data rates fall between asynchronous serial and SPI; most I^2C devices can communicate at 100kHz or 400kHz.
  - There is some overhead with I^2C; for every 8 bits of data to be sent, one extra bit of meta data (the "ACK/NACK" bit, which we'll discuss later) must be transmitted.

I2C Data Frame

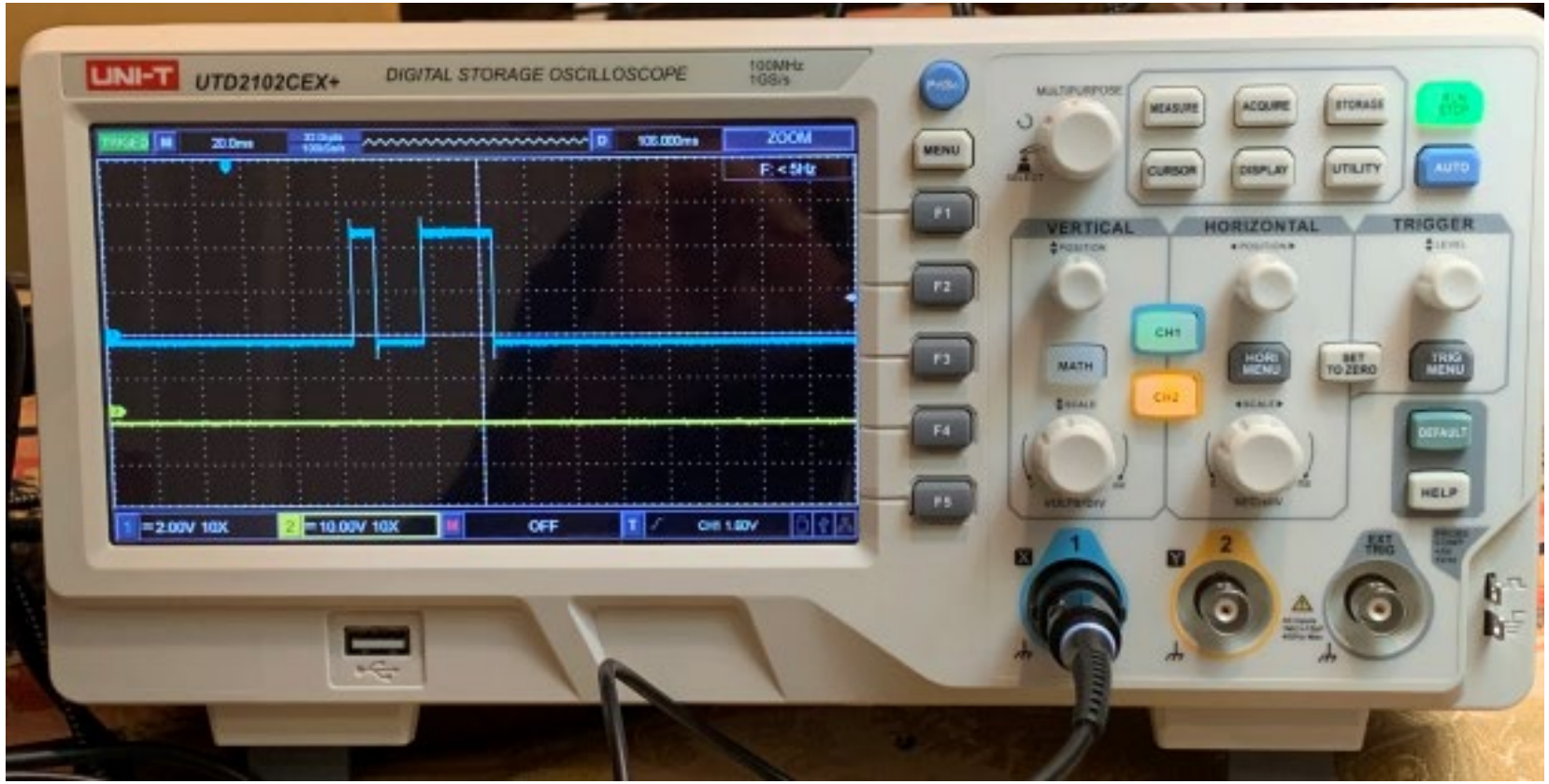
- **Start condition:** To initiate the address frame, the controller device leaves SCL high and pulls SDA low



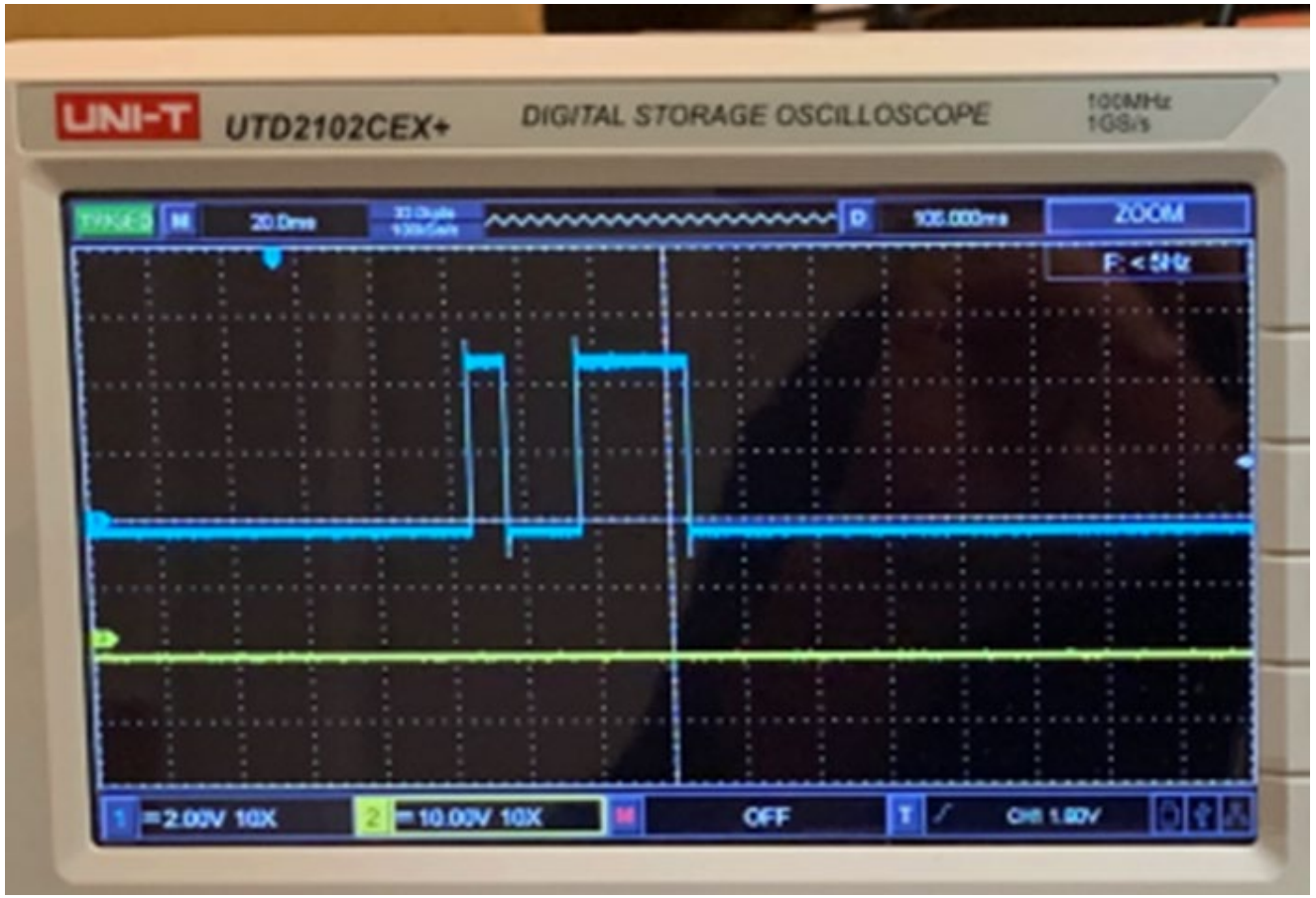
- Each peripheral needs to have a unique 7-bit address. 2^7=128. Advanced protocol allows 10-bit address: 2^10=1024.
- Stop condition: a 0→1 (low to high) transition on SDA *after* a 0→1 transition on SCL, with SCL remaining high.

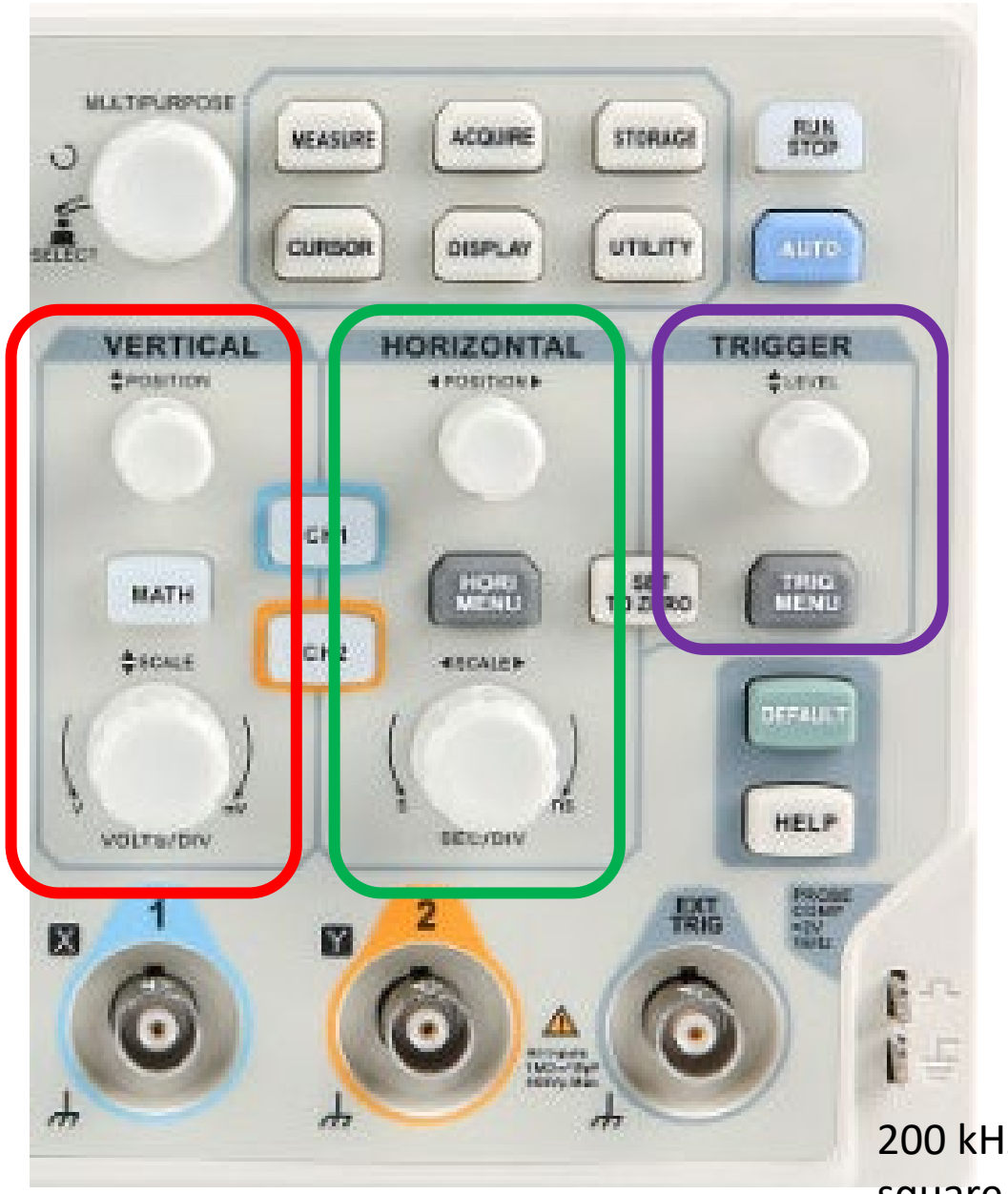
Oscilloscope

- let’s setup an oscilloscope (UNI-T UTD2102CEX+ “digital storage oscilloscopes)



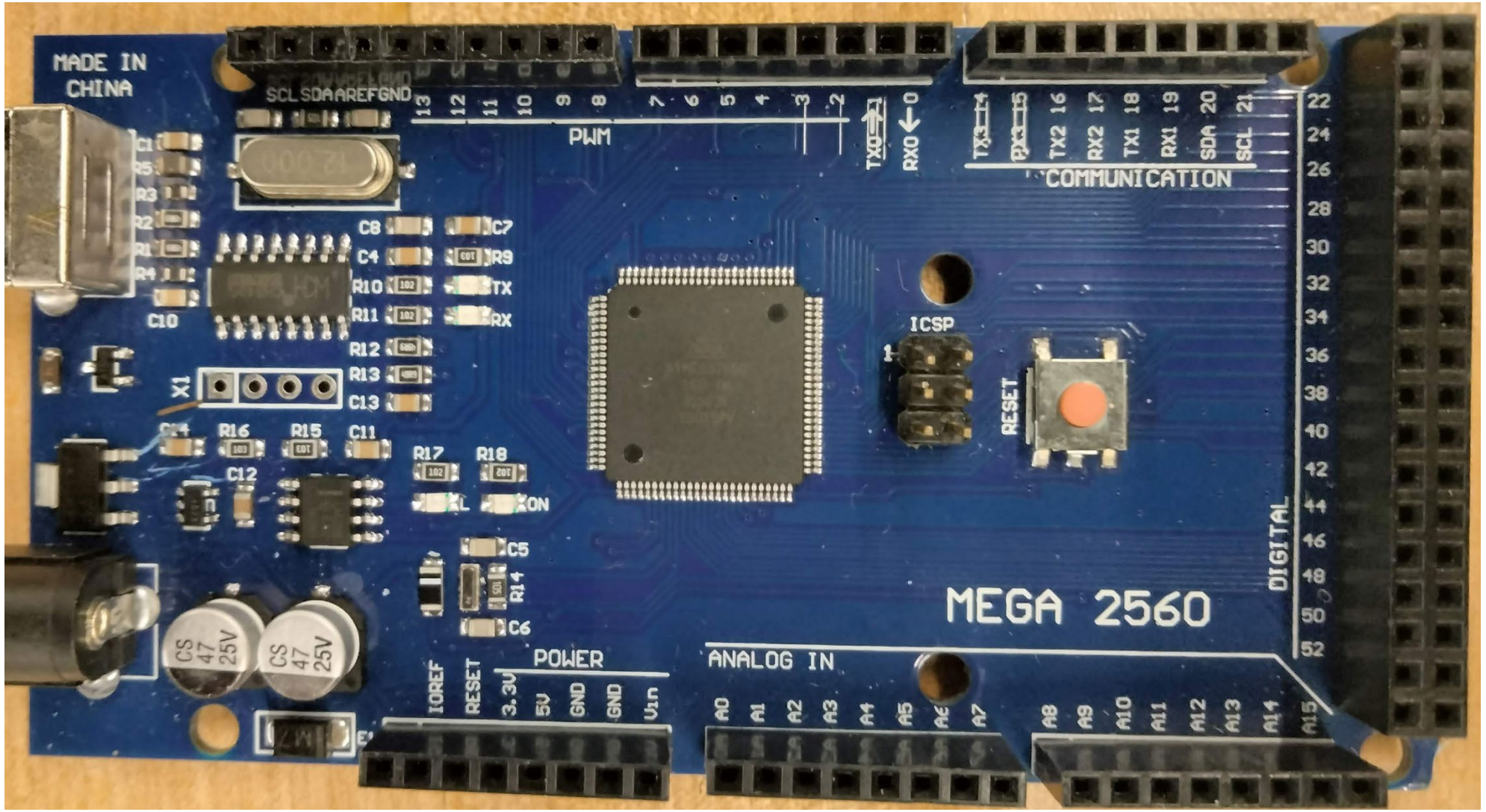
Scope Review





200 kHz square pulse

Serial communication pins on Arduino MEGA 2560



Asynchronous:

- TX0/RX0
- TX1/RX1
- TX2/RX2
- TX3/RX3

I2C:

- Pin 20: SDA
- Pin 21: SCL

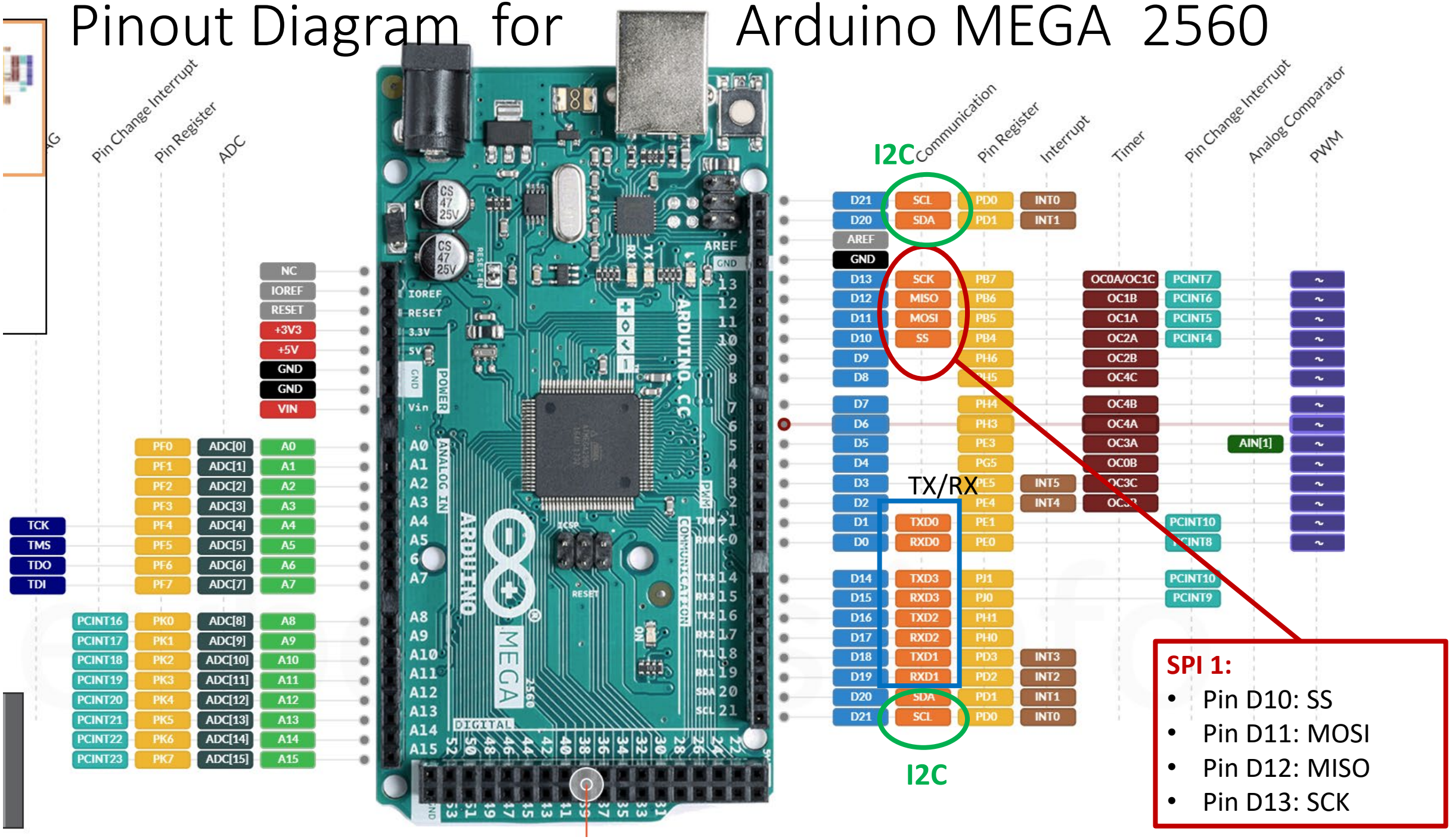
Also the top two pins on the left row.

SPI:

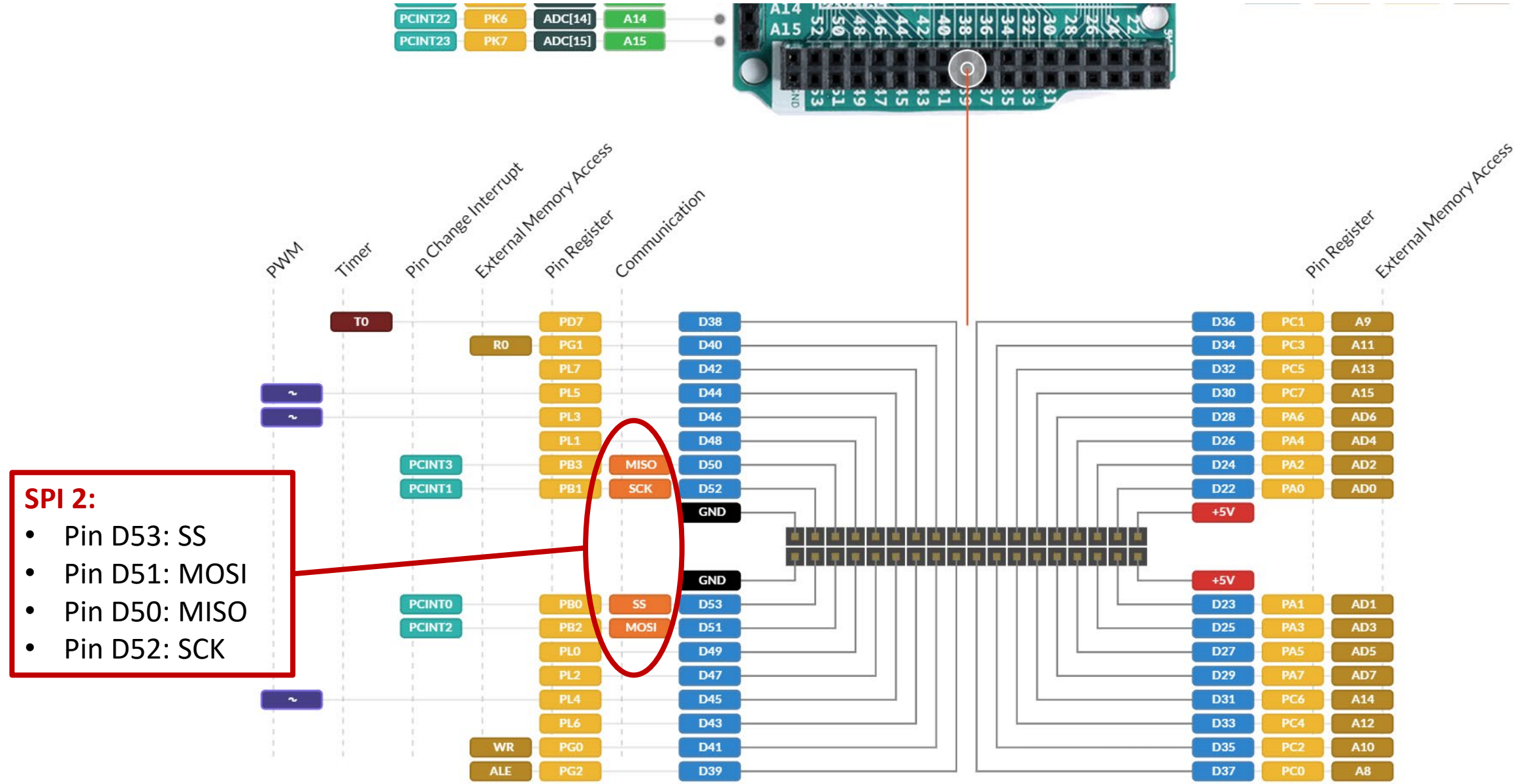
- You can specify any 4 GPIO pins for the required 4 lines. (Software serial at reduced speed)
- Hardware serial: Specific pins only

(Check the board pinout diagram)

Pinout Diagram for Arduino MEGA 2560



Extra pins on the MEGA 2560 board:

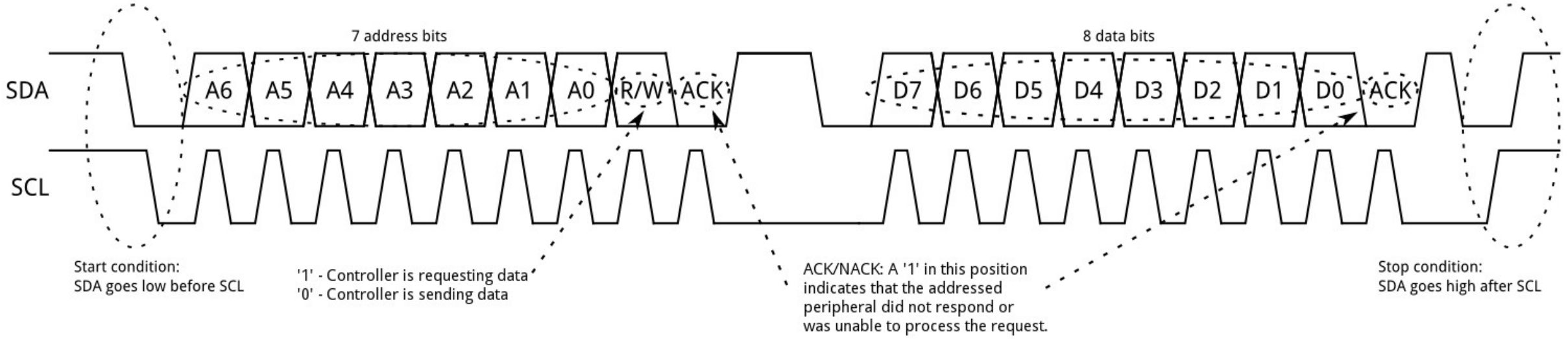


SPI 2:

- Pin D53: SS
- Pin D51: MOSI
- Pin D50: MISO
- Pin D52: SCK

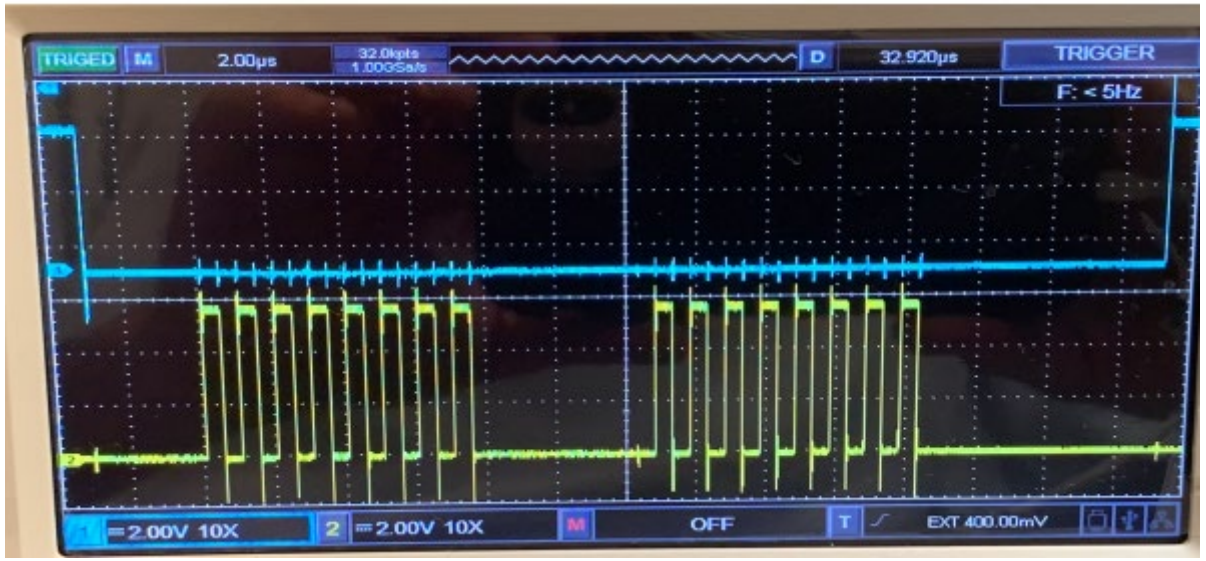
I2C decode

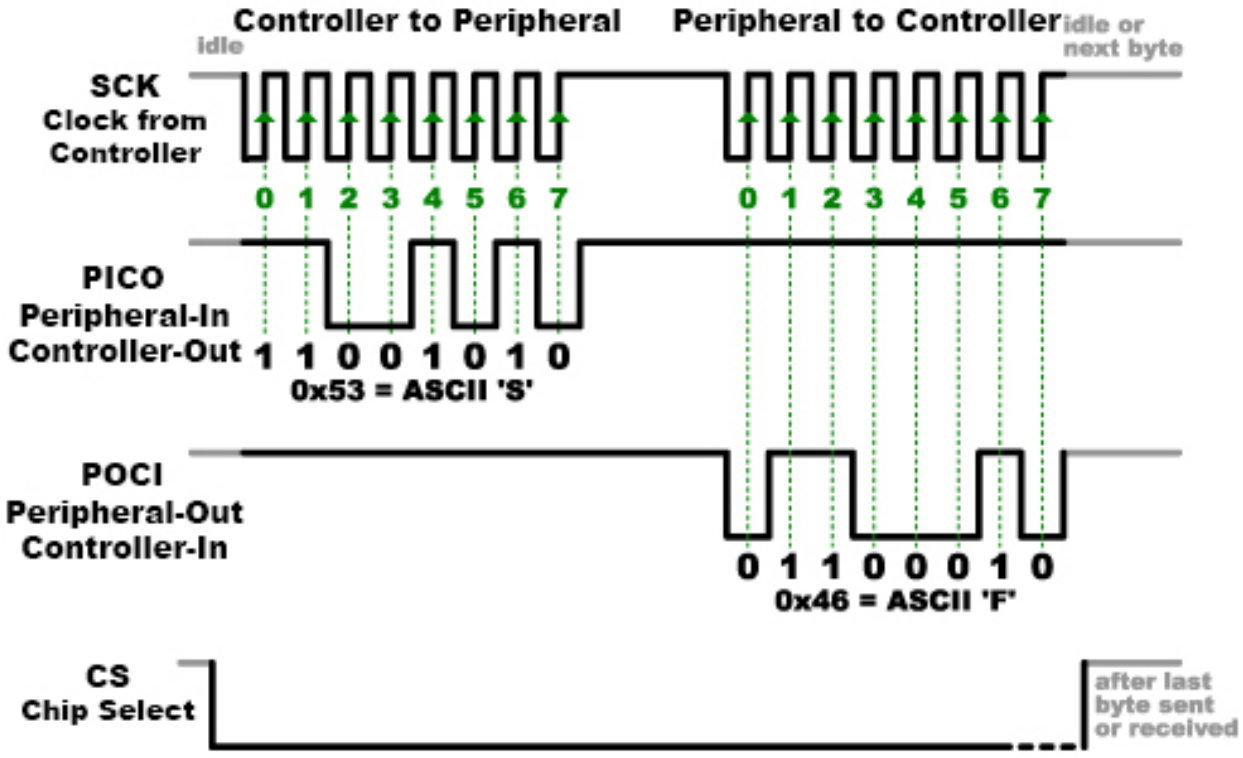
- What is the clock speed of I2C?
- On the right is what you should see on the scope. Can you decode the address?



SPI decode

- What is the clock speed of SPI?
- On the bottom is what you should see on the scope. Can you monitor PICO and decode its information?





Homework this week

- 1. SparkFun Please read all three SparkFun tutorials listed above, near the start of this unit’s material.
- 2. SPI vs I2C speeds:
  - Figure out how to disable the VOC measurement on your BME680s. Turn off all Serial.print statements in loop except for an initial print announcing your intentions and a final one giving your result. Compare how long it takes to read the BME680 100 times using I2C and SPI. Yeah, SPI is much faster, but only if your device’s time to perform a measurement is small compared to the time to execute an I2C or SPI instruction!