

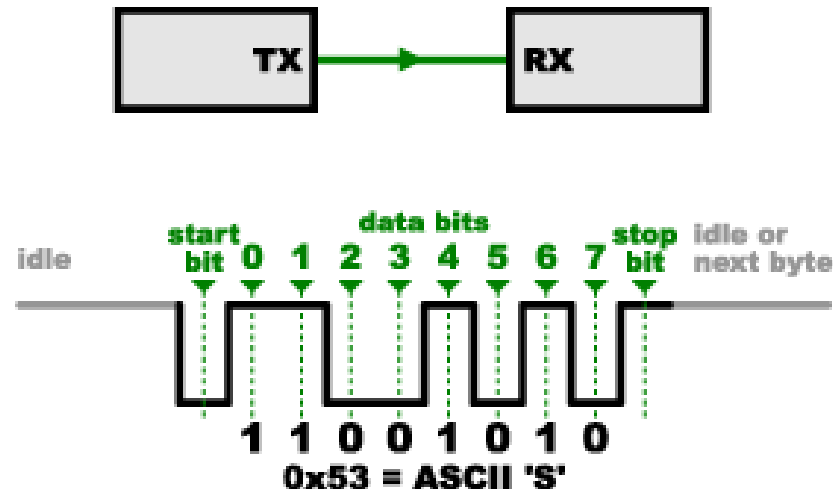
# P524: Survey of Instrumentation and Laboratory Techniques

## Week 7.5: Serial Communications, Oscilloscope (cont.)

10/9/2025

# Asynchronous vs Synchronous serial protocols

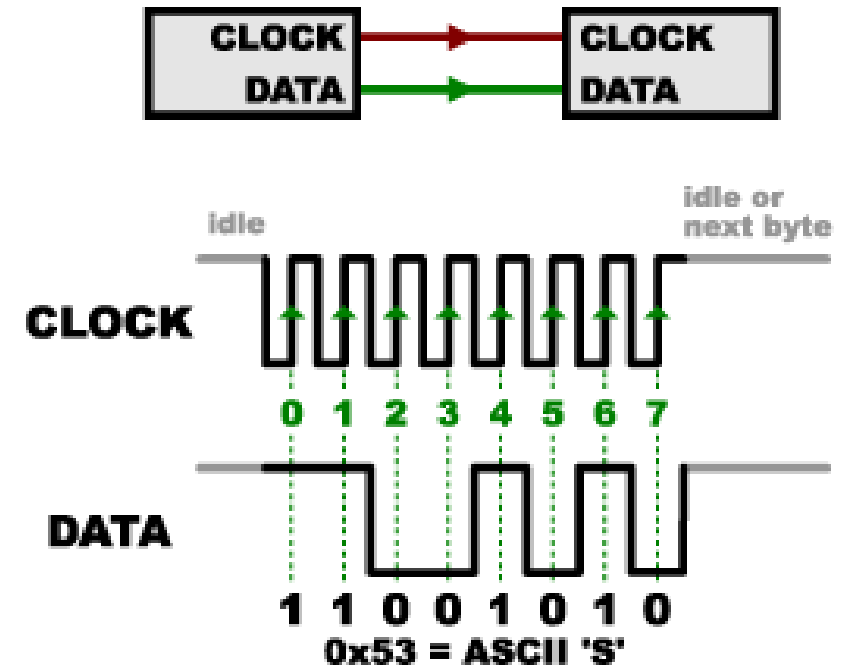
## Asynchronous:



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.

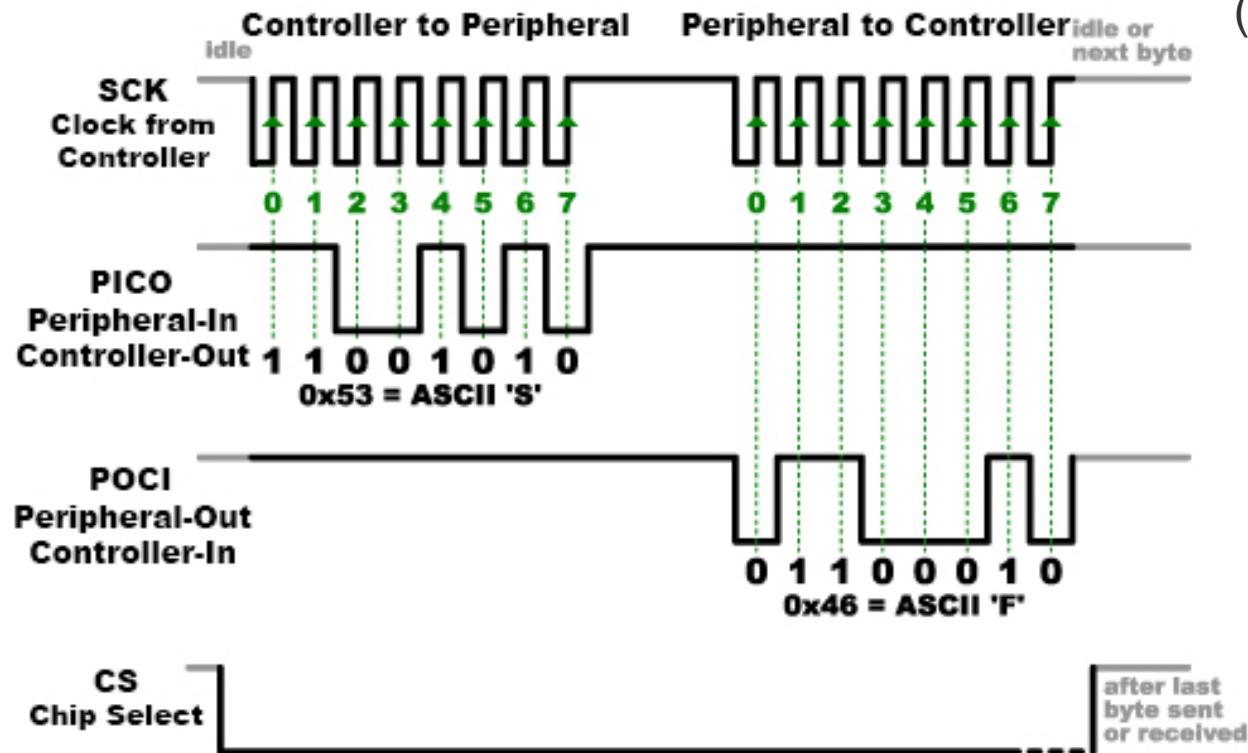
Complex hardware to strip data bits

## Synchronous:



- Because the clock is sent along with the data, specifying the speed isn't important
- Save the start and stop bits; simple hardware converts to parallel

# Serial Peripheral Interface (SPI)



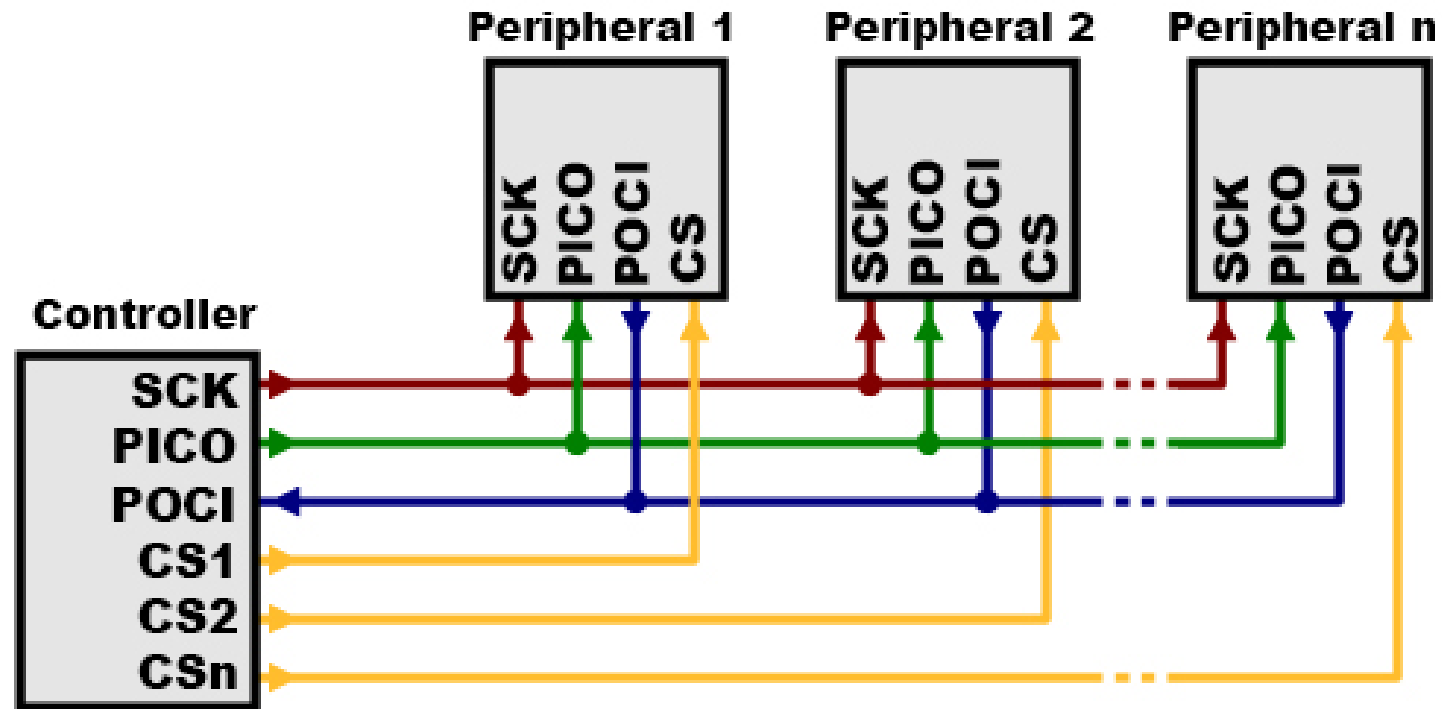
**SPI can operate at extremely high speeds: ~ 10MHz.**

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.

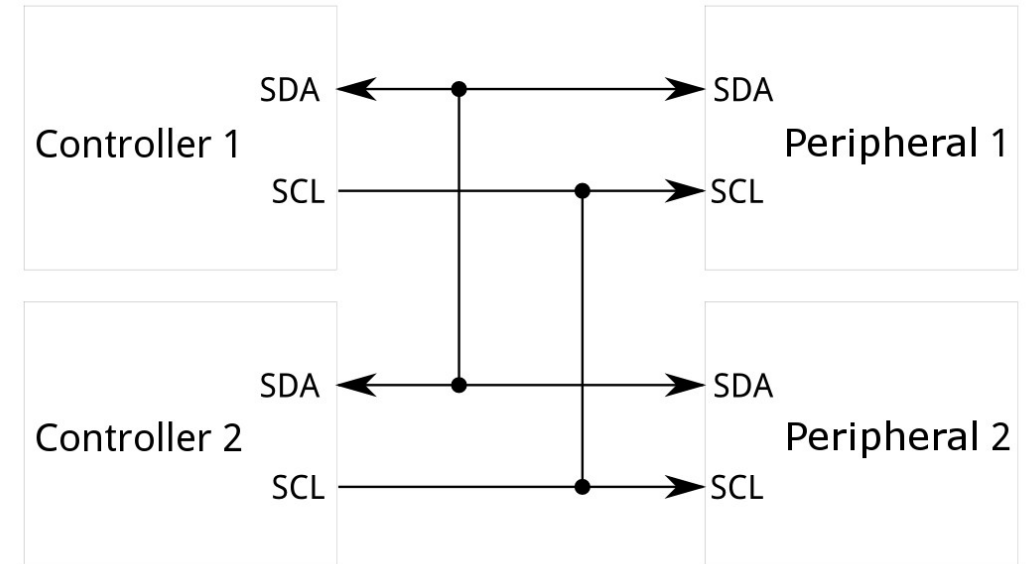
# Multiple Peripherals



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

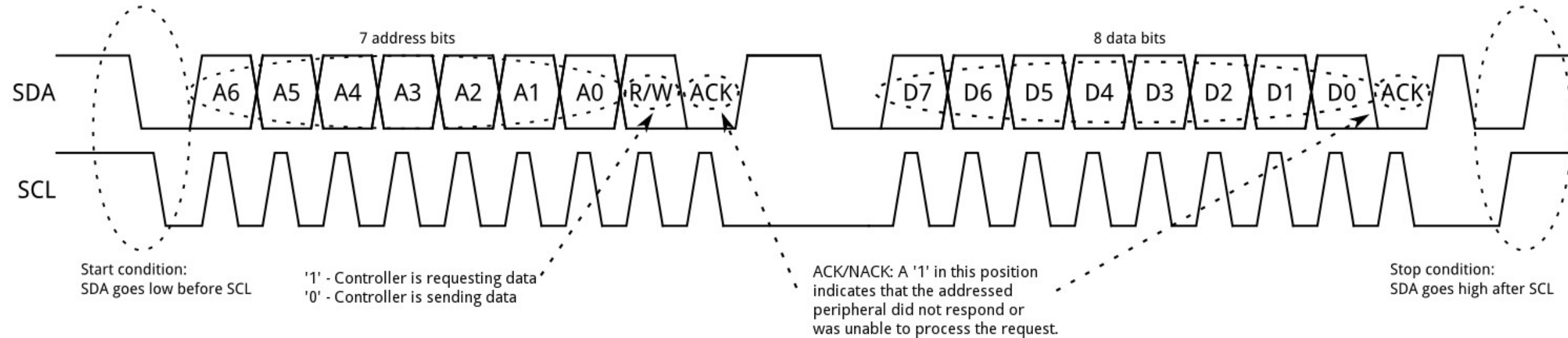
# Inter-Integrated Circuit (I2C)

- I<sup>2</sup>C requires a mere two wires, like asynchronous serial, but those two wires can support up to 1008 peripheral devices.
- Unlike SPI, I<sup>2</sup>C 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<sup>2</sup>C devices can communicate at 100kHz or 400kHz.
  - There is some overhead with I<sup>2</sup>C; for every 8 bits of data to be sent, one extra bit of meta data (the "ACK/NACK" bit) must be transmitted.



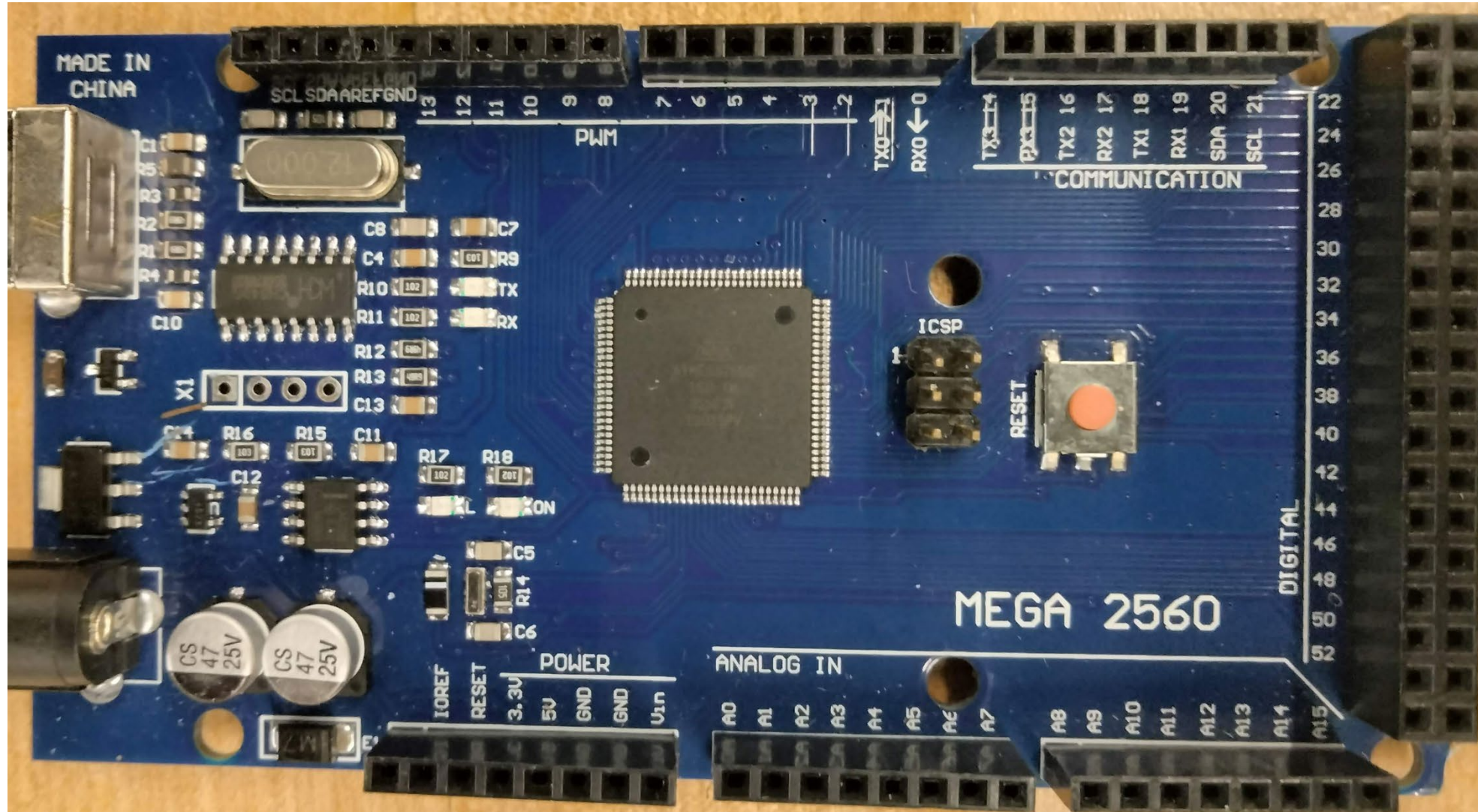
# I2C “Frame”

- **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  $0 \rightarrow 1$  (low to high) transition on SDA *after* a  $0 \rightarrow 1$  transition on SCL, with SCL remaining high.

# Serial communication pins on Arduino MEGA 2560



## Asynchronous:

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

## I2C:

- Pin 20: SDA
- Pin 21: SCL

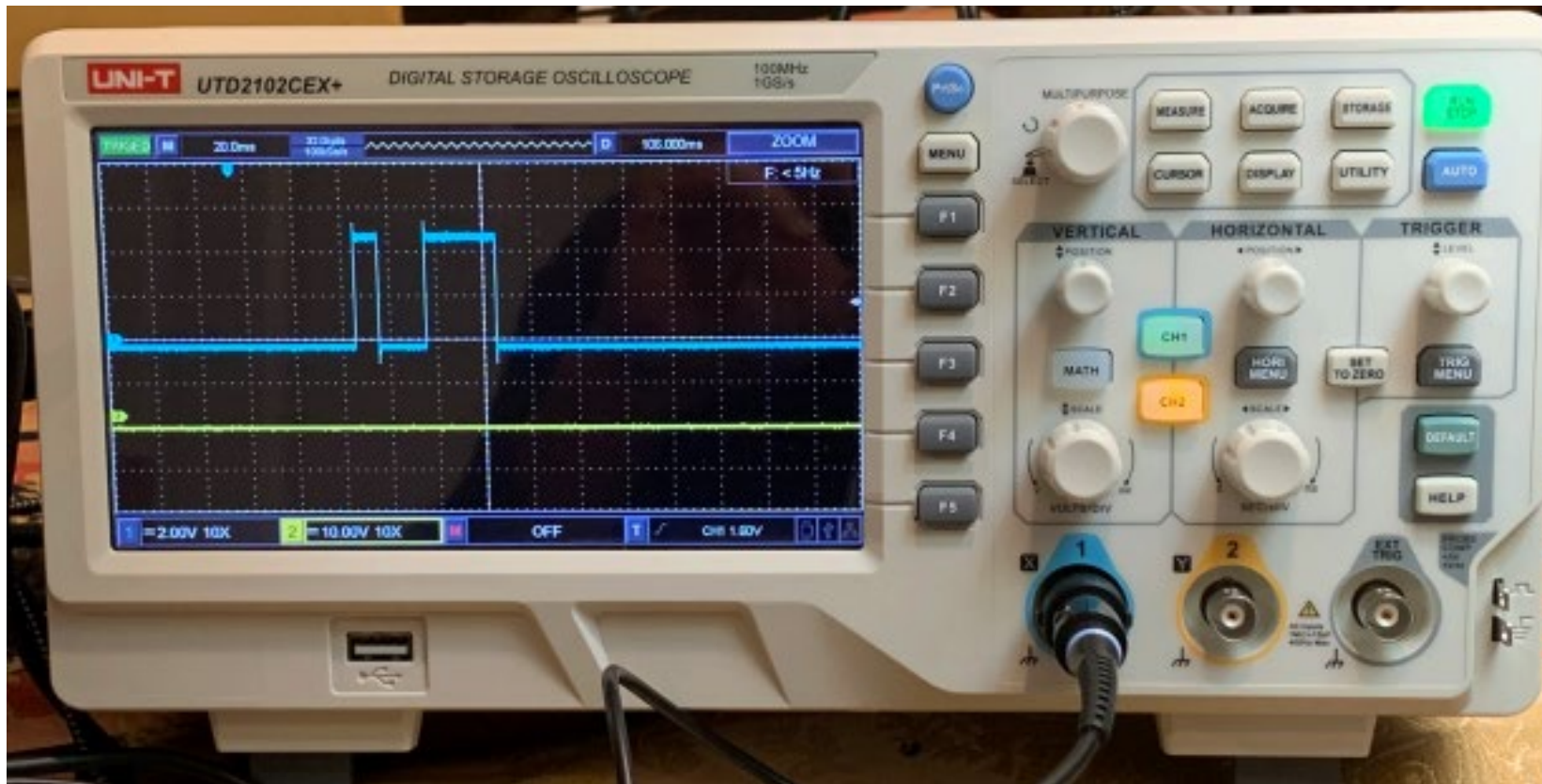
Also the top two pins on the left row.

## SPI:

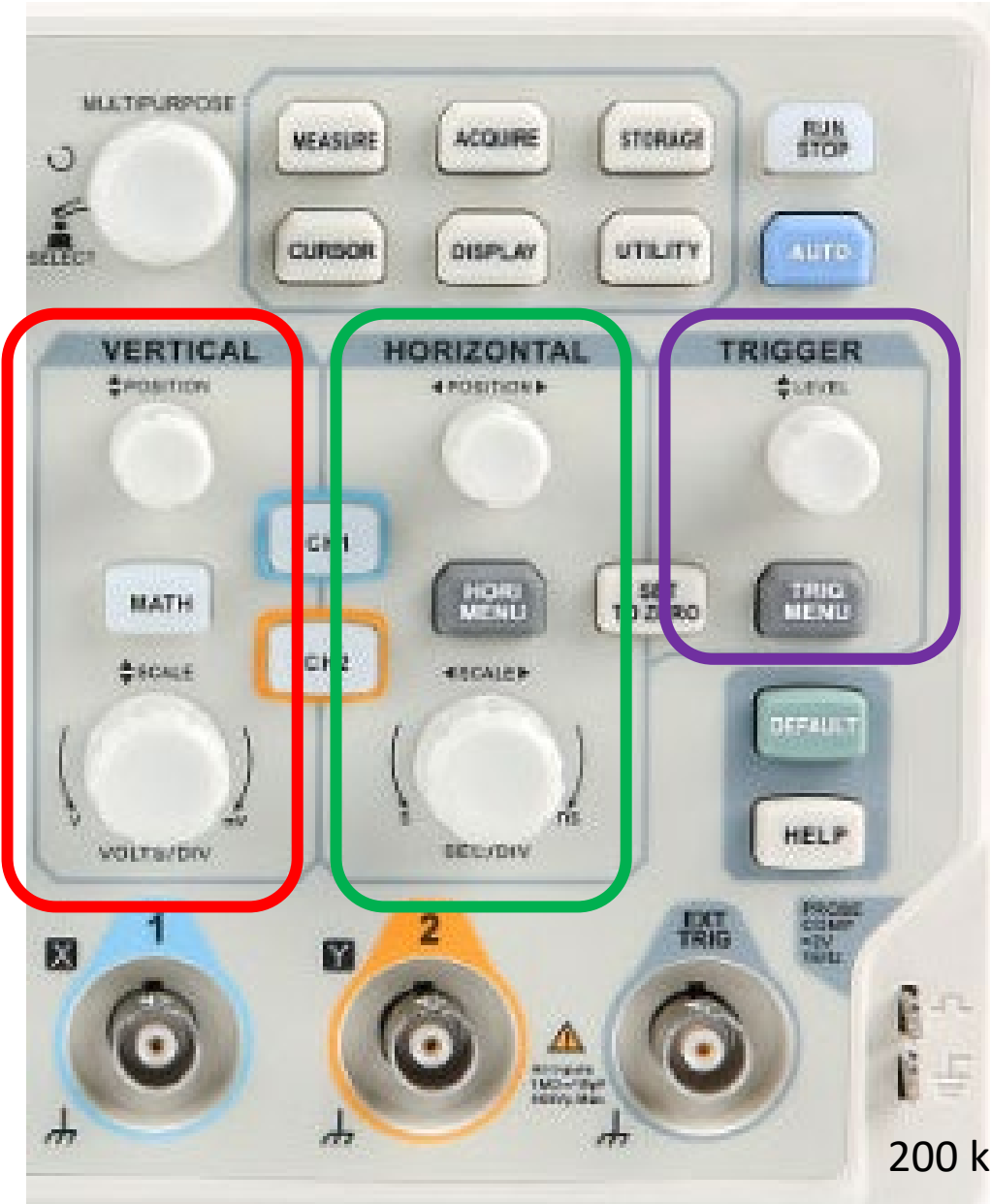
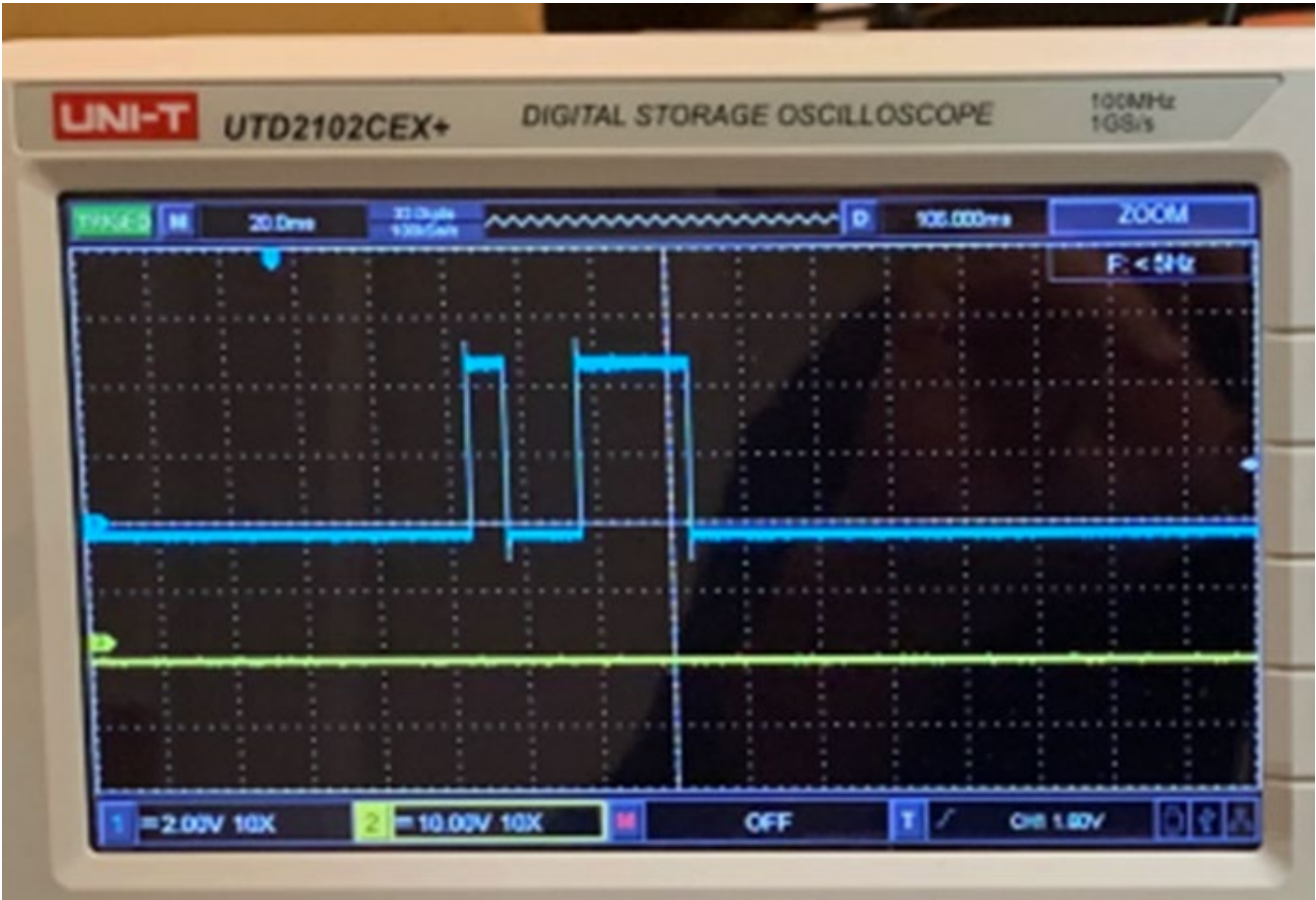
- You can specify any 4 GPIO pins for the required 4 lines. (Software serial at reduced speed)
- Hardware serial: Specific pins only (Check the board pinout diagram)

# Oscilloscope

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



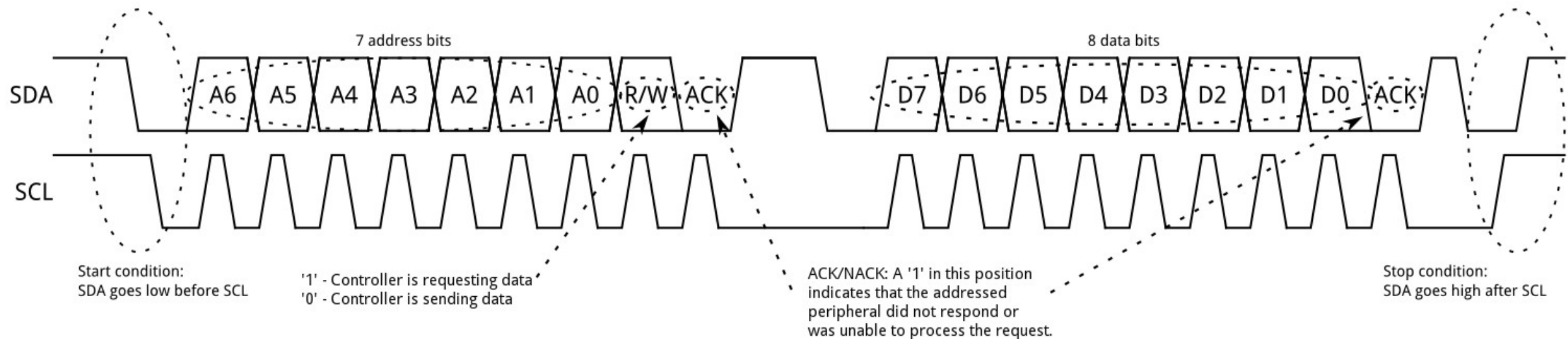
# Scope Review



200 kHz  
square pulse

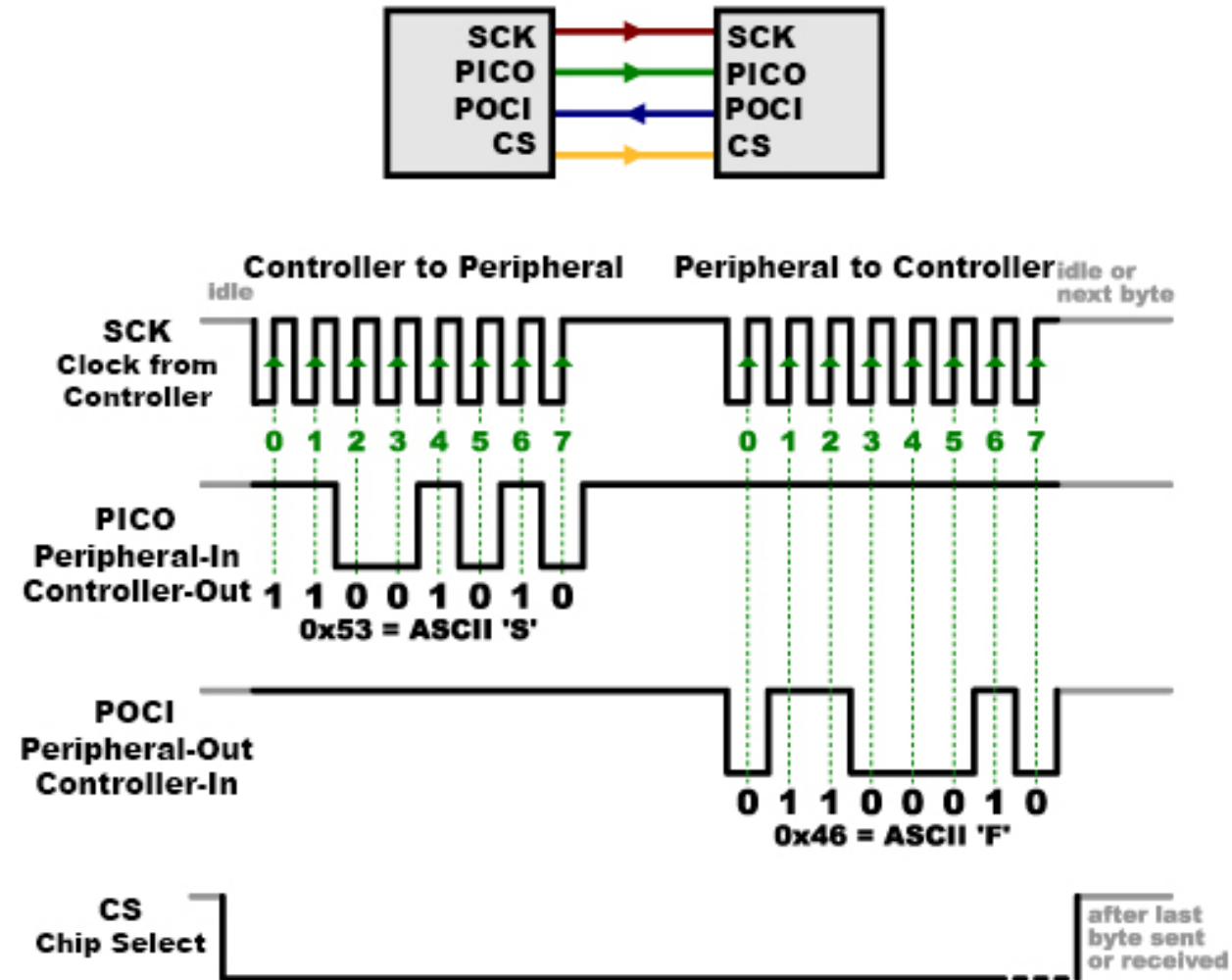
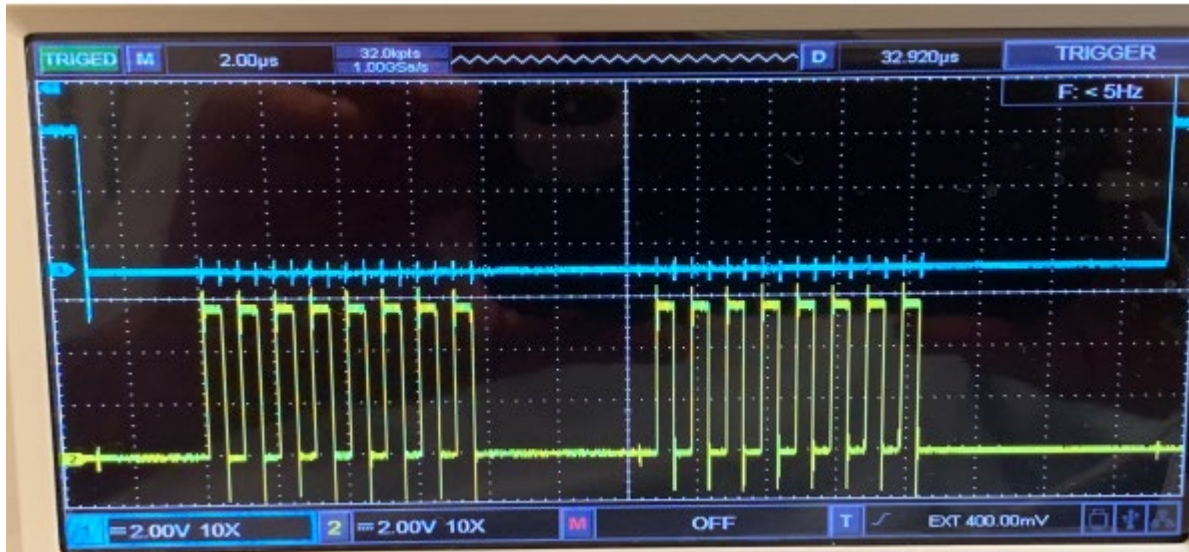
# I2C decode

- 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 (due 10/16)

- 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. 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.