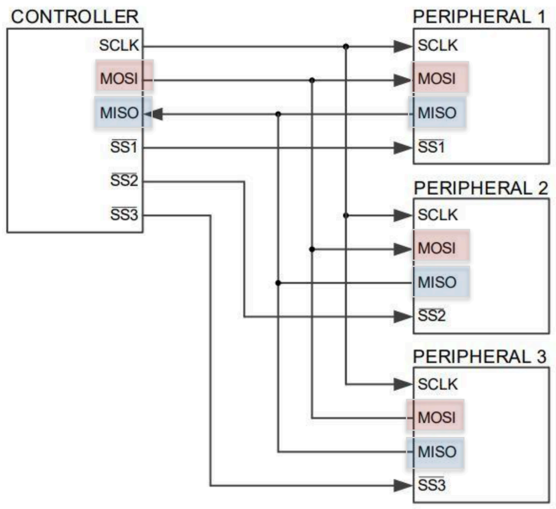
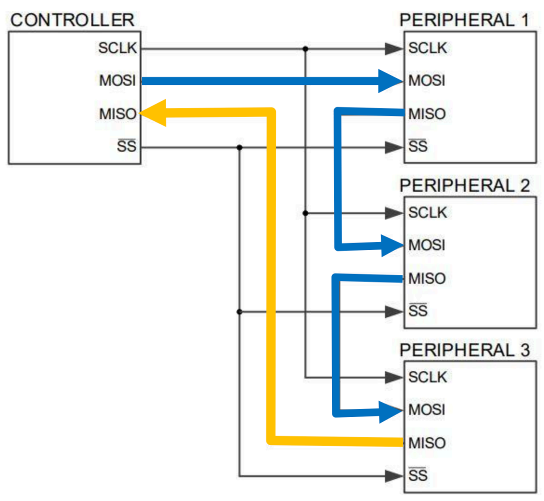
## Serial Peripheral Interface

- higher throughput than I2C
    - dedicated connection to each device (sub) from controller (main)
- no need for arbitration
- uses *push-pull outputs* and not *open drain*
- strobe-side (no ACK/NACK)
- 4 wires
    - Serial clock line (SCLK)
    - Main In, Sub Out (MISO)
        - send data to controller
        - floating when that device is not selected
    - Main Out, Sub In (MOSI)
        - send data to device (shared between all devices)
    - Sub select (SS) (active low)
- each line is ***uni-directional***
- **full-duplex** communication - can send and receive at the same time
    - data is sent on both MOSI and MISO
        - both lines will actively shift data even when one line is not needed
- No need for arbitration due to **single master**

### multidrop configuration



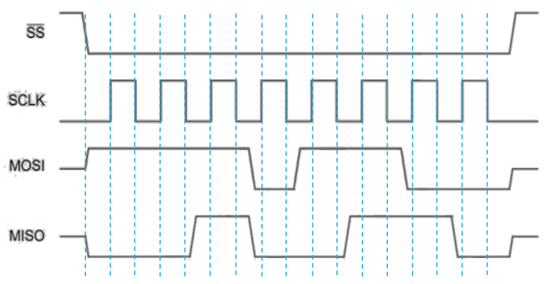
### daisy chain configuration



- more complex timing diagram; not supported by all devices

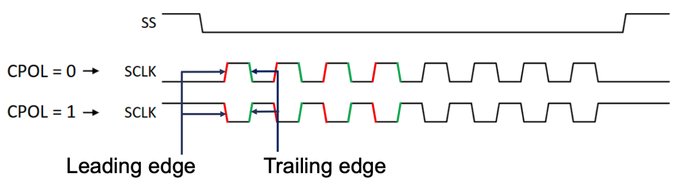
## Basic Communication

- controller selects a device by setting SSn low
  - data is sent 1 bit at a time, ***MSB first***
  - no limit on how many bits can be sent at a time
  - many devices use 8bit for consistency
  - Only the selected slave processes the data on the MOSI line. The other slaves ignore the data, even though they receive it.
  - Only the selected slave drives the MISO line. All other slaves keep their MISO lines in a high-impedance state (tri-state) to avoid contention.



## Clock Polarity (CPOL)

- **Clock polarity (CPOL)** determines the *idle value* of the clock
  - CPOL = 0 if SCLK is 0 when idle
  - CPOL = 1 if SCLK is 1 when idle
- first edge after idle is *leading edge* and second edge is *trailing edge*



## Clock Phase (CPHA)

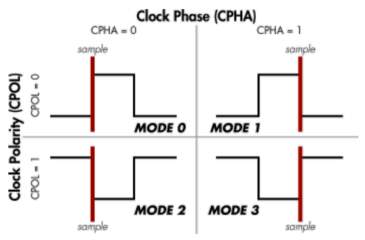
- **Clock phase (CPHA)** determine when the data is output
  - CPHA = 0
    - MSB is typically output right when SS goes low
    - data is sampled on the leading edge
  - CPHA = 1

- MSB is typically output on the first clock edge after SS goes low
- data is sampled on the trailing edge

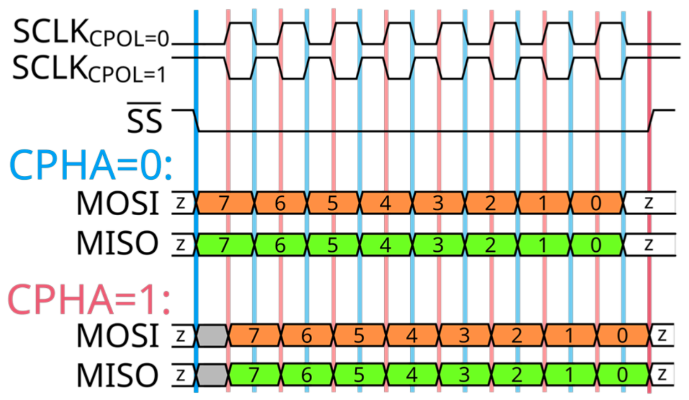
## SPI - 4 Modes

This allows for 4 SPI Mode settings
- Mode 0 – CPOL = 0, CPHA = 0
- Mode 1 – CPOL = 0, CPHA = 1
- Mode 2 – CPOL = 1, CPHA = 0
- Mode 3 – CPOL = 1, CPHA = 1

Generally, devices support Mode 0
- But you must check the data sheet to make sure!

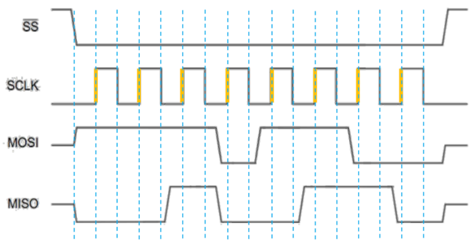


## Peripherals



### SPI example

- SCLK is 0 when idle. CPOL = 0
- Data changes at the same time as SS. CPHA = 0
  - So, data is sampled on the rising edge.



## Advantages of SPI over I2C

1. SPI supports higher speeds (45Mbits/s VS 1Mbits/s)
2. SPI devices do not need unique addresses (SSn)
3. I2C limited to 8 bits; SPI has no limit
4. SPI lower power than I2C due to less circuitry and no pull-ups (discussion below)

## Limitations of SPI

- limited to a single controller per 'bus' (single master unlike I2C)
- in multi-drop, number of connected device is limited by number of SS signals on master
- can be slow if device all require different modes
  - If multiple devices on the same bus require different modes, the master must reconfigure its SPI settings each time it switches between devices.
- both controller and device are always sending info
  - The slave may not always have meaningful data to send back to the master, but it is still required to send something (e.g., dummy bytes).

    - This constant exchange of data can waste bandwidth and increase power consumption, especially in low-power applications.
- no ACK/NACK; no guarantee data sent successfully

## Push-Pull VS Open Drain

Why open drain?

> The main reason is that it allows for multiple endpoints to coexist on the same line and transmit. The outputs can only pull one way, so the effect is like a ***wired OR gate***. If the outputs were push-pull, then the device that is asserting will fight against the ones that aren't.

> Open drain output has higher power consumption during active transfers due to the pull-up resistors that are used.

> Push/pull permits much faster signaling and requires no pulling resistor, so is the default in one driver cases.
>
> Open drain allows some easy low-effort "Wired OR" type of multiple driver scenarios.
>
> To do multiple drivers with push/pull, you need a scheme for making sure only one is enabled at any time.
>
> Signals which cross power enable or voltage domains require careful thought in either case, though a variant open drain driver made of a discrete grounded gate MOSFET can when correctly applied translate voltage levels and even allow the lower voltage side of the topology to be unpowered, at the cost of course of slew rate limited by the pullup resistor to each supply.

Summary:

- open drain high power consumption; but supports wiredOR (multiple endpoints coexist on same line)
- push pull works for SPI bc the wires are single-direction; would not have cases where one end high other end low
    - push pull is also faster & lower power consumption