**Digital Communication**

Table of Contents

[Serial-Parallel Communication 2](#_Toc64556607)

[Difficulties with Serial Ports 2](#_Toc64556608)

[Serial Peripheral Interface 3](#_Toc64556609)

[Master Slave Relationships 3](#_Toc64556610)

[Steps of Data Transmission 4](#_Toc64556611)

[Multiple Slaves 6](#_Toc64556612)

[Benefits and Drawbacks 7](#_Toc64556613)

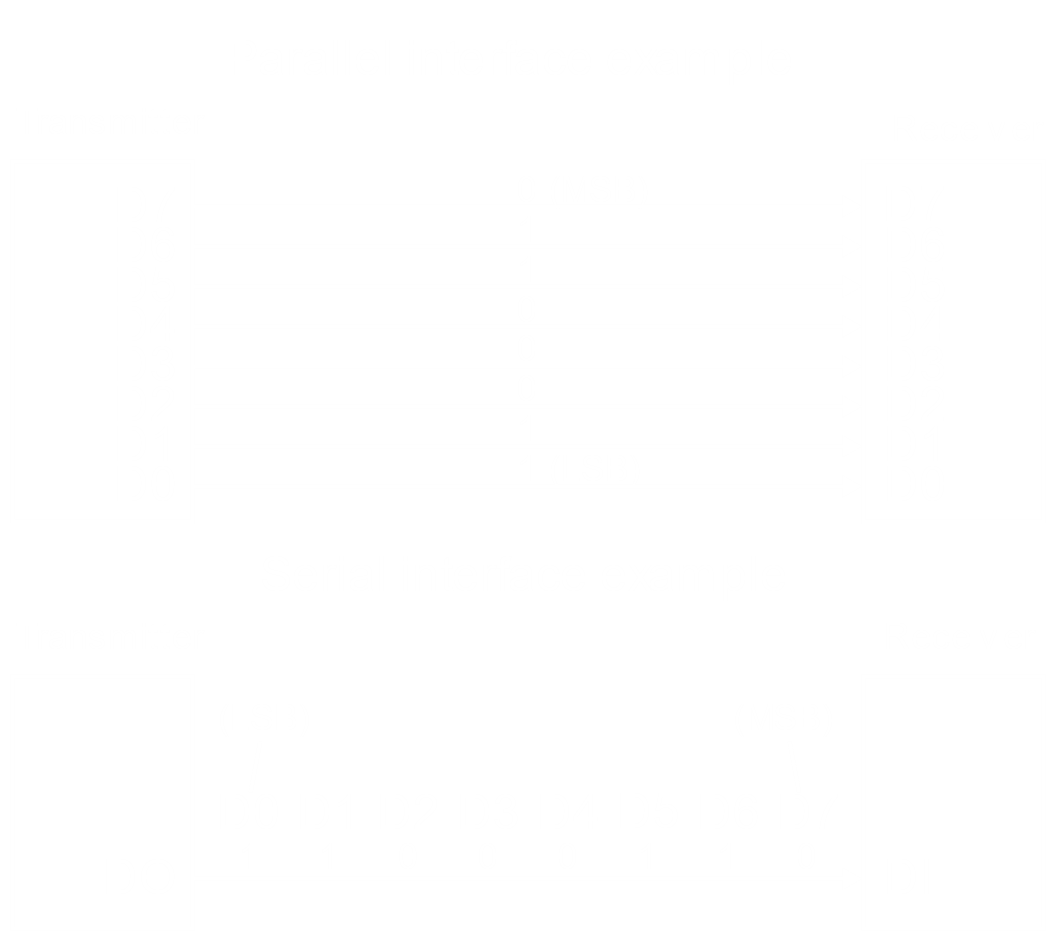
[I2C Communication Protocol 8](#_Toc64556614)

[Bus Connections 9](#_Toc64556615)

[Terminology 9](#_Toc64556616)

## Serial-Parallel Communication

When two devices want to communicate with each other by transferring bits, they can do so in either serial or parallel form. In serial communication, the bits are sent one by one along just one wire, so we need more clock cycles to send a certain number of bits. In parallel communication, many bits are sent at once via parallel wires, so we need fewer clock cycles. In either case, the sender and receiver clocks need to be synchronized for the receiver to be able to accurately detect what bits are being transmitted.



### Difficulties with Serial Ports

As mentioned before, it is essential that the sender and receiver clocks be synchronized for the receiver to be able to correctly detect all the data being sent. However, it is not always possible to do this. We need to deal with asynchronous transmission. In such cases, the sender attaches start and stop bit patterns to the beginning and end of each byte. The receiver is aware of these patterns and uses them to detect when a particular byte starts or stops and synchronize itself accordingly. Asynchronous communication does work, but it has a lot of overhead.

## Serial Peripheral Interface

The serial peripheral interface (SPI) is used to transfer data between integrated circuits using a reduced number of data lines. It is a common configuration used by many devices, such as SD card modules, RFID card reader modules etc. to communicate with microcontrollers.

One unique benefit of SPI is the fact that data can be transferred without interruption. Any number of bits can be sent or received in a continuous stream.

Some key features of SPIs include:

* Read, write and full-duplex (simultaneous read and write) data transactions
* SPI master and SPI slave configuration
* Configurable SPI bus clock polarity, phase and frequency
* Configurable number of bits to transfer
* Communication with serial external devices like ADCs, DACs, temperature sensors, pressure sensors, LCD controllers, etc.

### Master Slave Relationships

Devices communicating via SPIs are in a master-slave relationship. The master is the controlling device, usually a microcontroller, while the slave is usually something like a sensor, display or memory chip which takes instructions from the master. In the simplest configuration, there is a single master and a single slave, but one master can control multiple slaves.



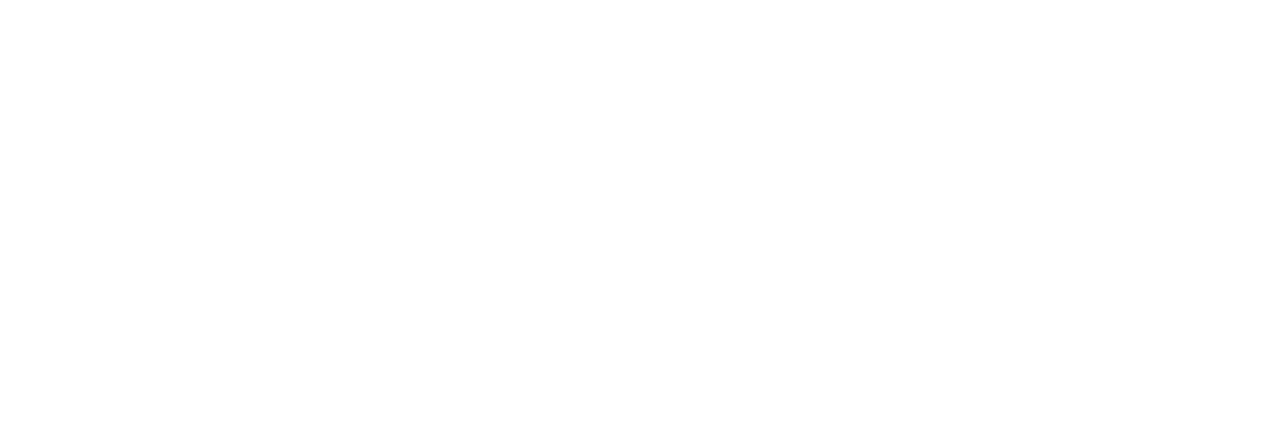
In the above configuration, the pins are as follows:

* Master Output Slave Input (MOSI) – Line from the master to send data to the slave
* Master Input Slave Output (MISO) – Line for the slave to send data to the master
* System Clock (SCLK) – Line for clock signal
* Slave Select or Chip Select (SS/CS) – Line for the master to select which slave to send data to. It is normally kept high and is switched to low to make a selection. Thus, it is active low.

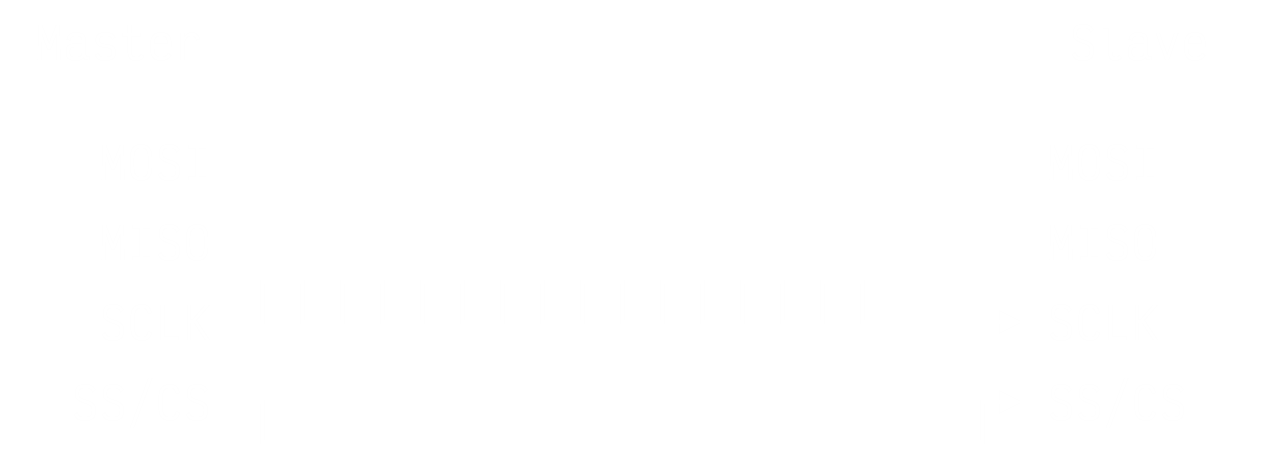
In an SPI configuration, the clock signal is generated by the master. The master sends data to the slave bit by bit through the MOSI line and the slave receives the data at the MOSI pin. This data is usually sent with the most significant bit first. The slave can also send data back to the master via the MISO line. This data is usually sent with the least significant bit first. Generally, the slave only sends an acknowledgement that the data was received.

### Steps of Data Transmission

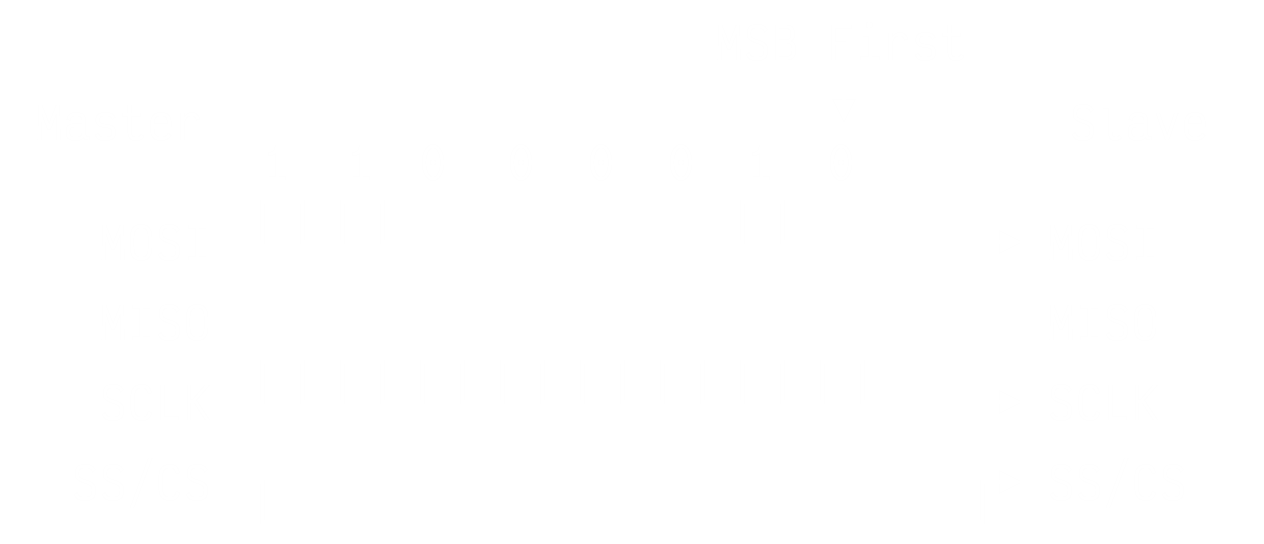
1. The master outputs the clock signal.



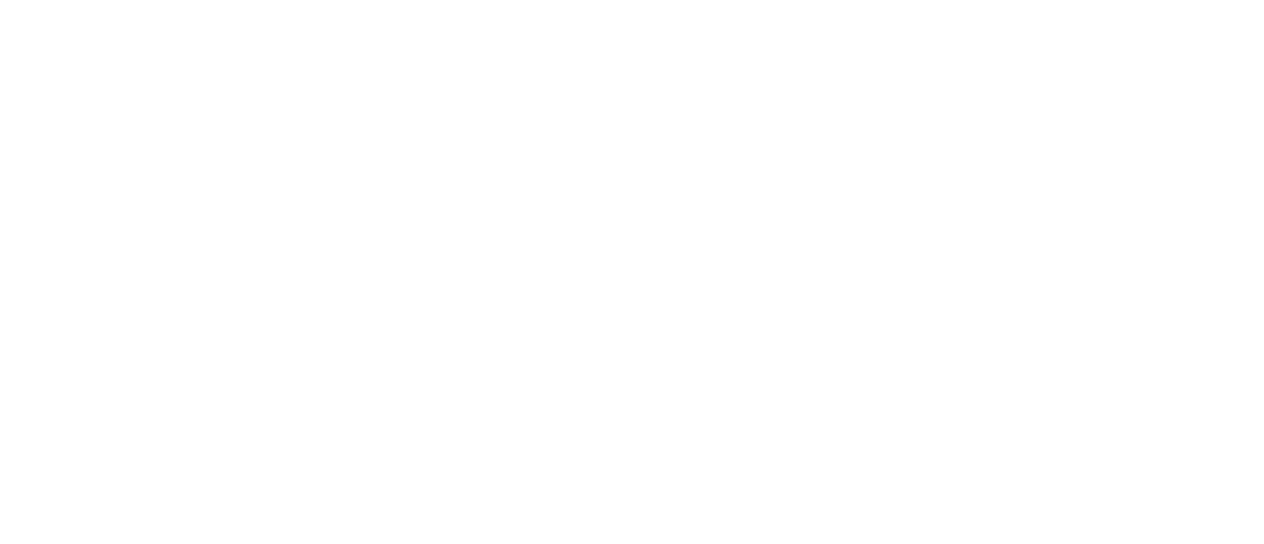
1. The master switches the SS/CS pin to a low voltage, which activates the slave.



1. The master sends the data one bit at a time along the MOSI line.

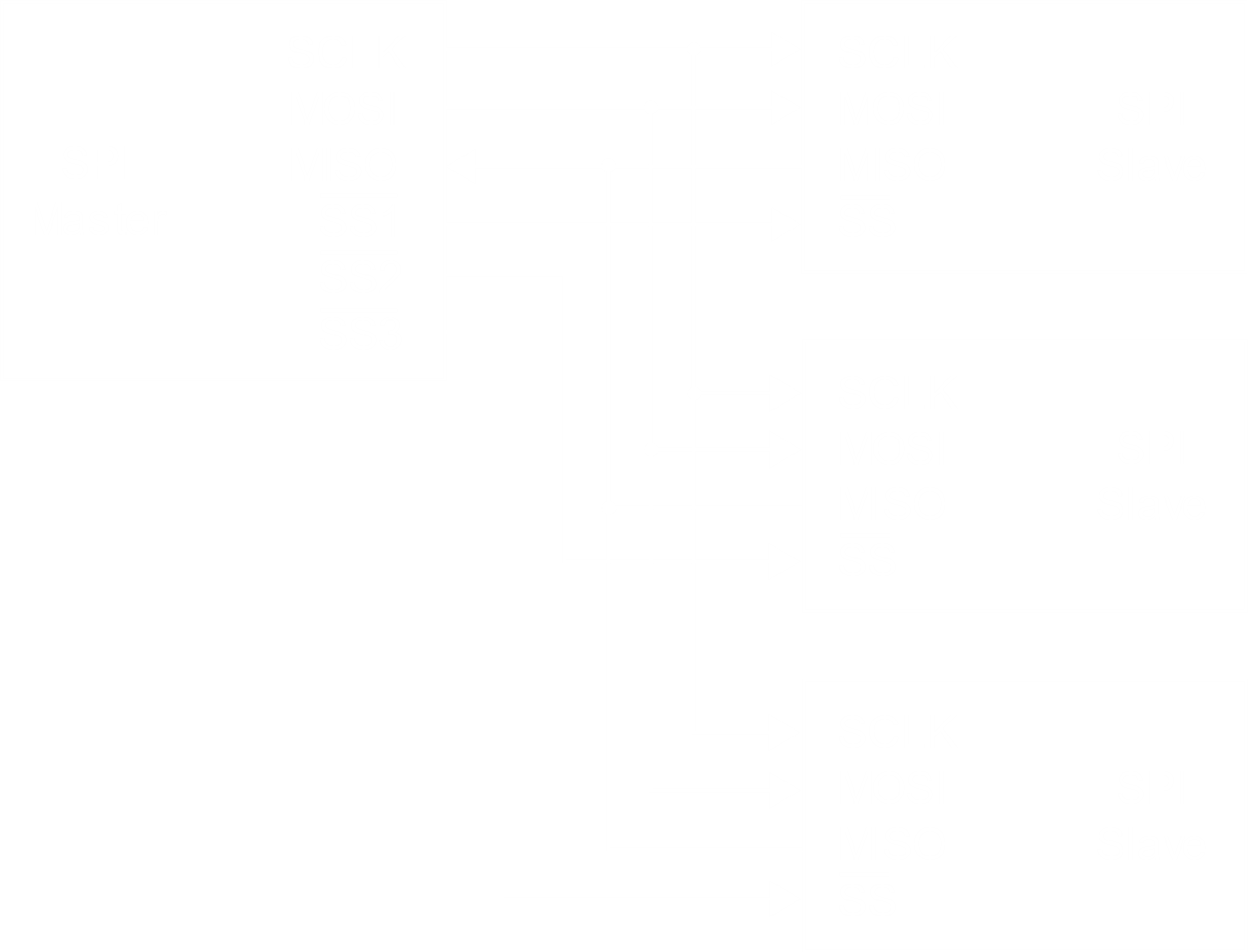


1. If a response is required, the slave sends an acknowledgement one bit at a time along the MISO line.

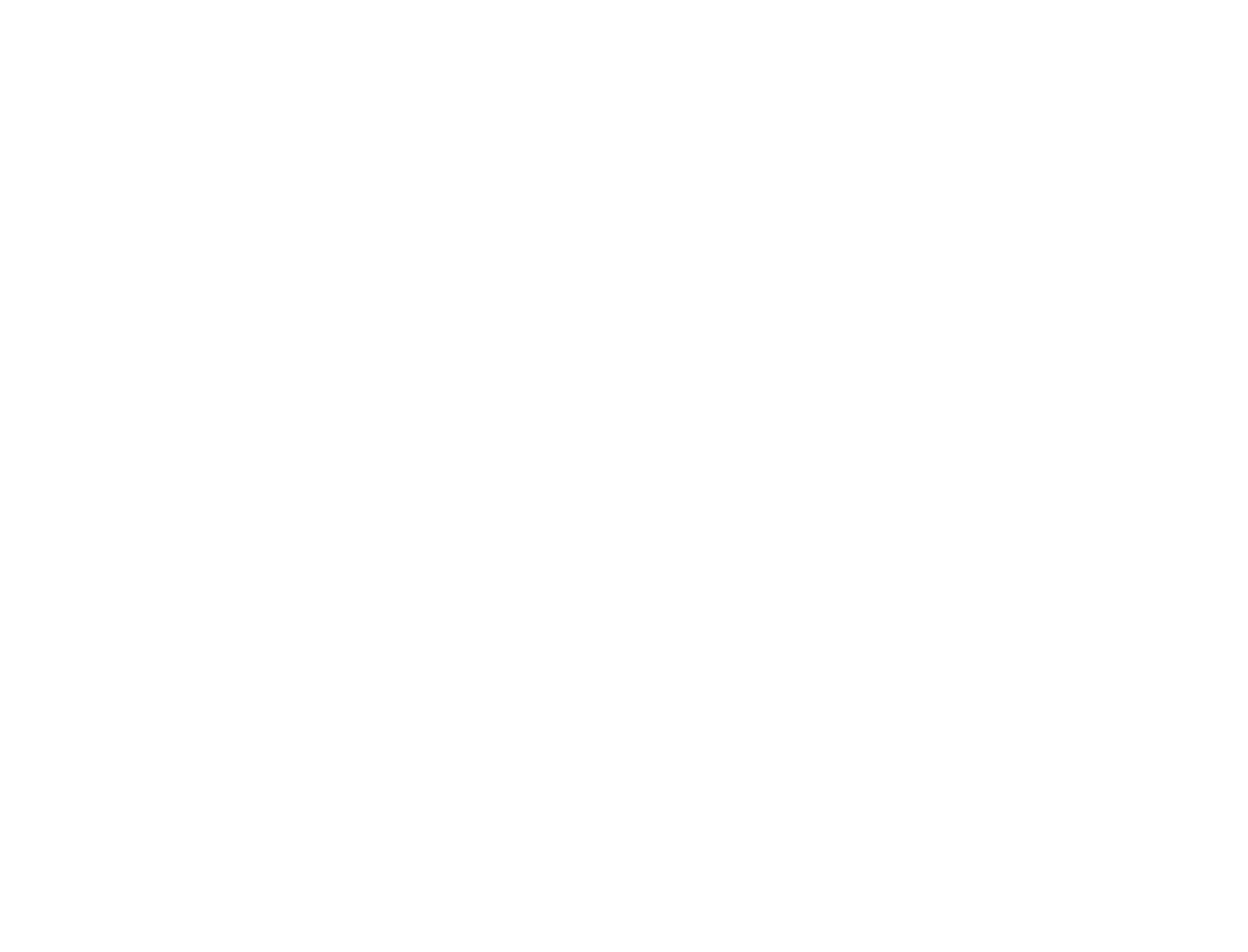


### Multiple Slaves

There are two ways for one master to control multiple slaves. One way is for the master to have multiple slave-select pins.



The other way is to have a single slave select pin daisy chained over all the slaves.



### Benefits and Drawbacks

☑ SPI is considered the fastest synchronous serial data transfer interface

☑ SPI is a very simple communication protocol

☑ It supports full-duplex (continuous, bi-directional) communication

☒ It requires more traces on the board, meaning the hardware is larger

☒ There is no hardware flow control

☒ There is no slave acknowledgement

☒ They may be prone to noisy spikes which cause faulty communication

## I2C Communication Protocol

The main problem with SPI is the large number of pins required. Connecting a single master to a single slave requires four pins, while every extra slave after that requires one extra pin. This makes the device rather large and undesirable in situations where space is limited. Additionally, we can only have one master on the bus.

To solve this problem, we can use the I2C communication protocol. This is a serial protocol for two-wire interface to connect low-speed devices like

* Microcontrollers
* EEPROMS
* AD and DA converters
* I/O interfaces
* Other peripherals in embedded systems

The I2C bus is popular because of

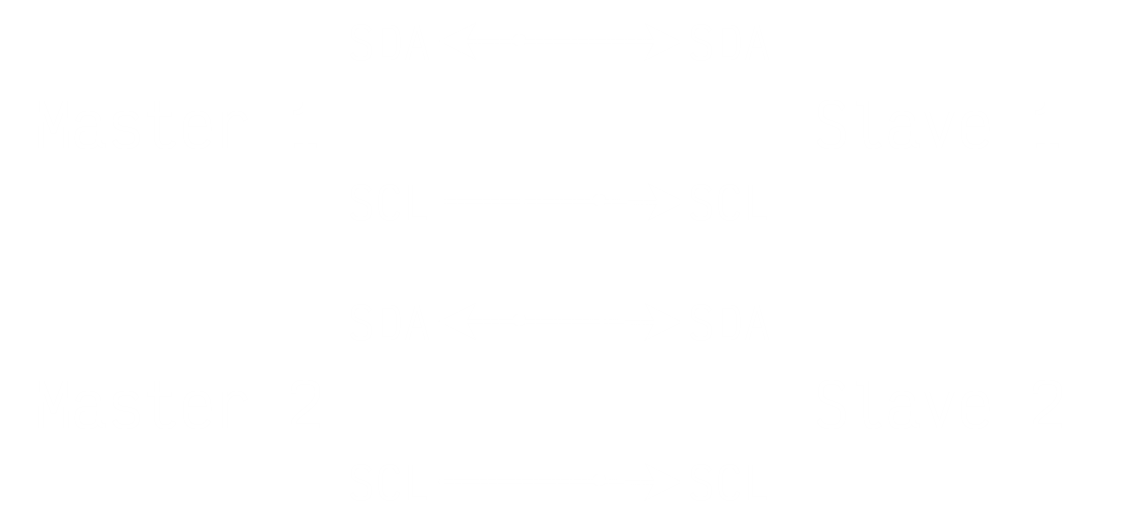
* Its simplicity
* The fact that there can be more than one master
* Only the upper bus speed is defined
* Only two wires are needed with pull-up resistors to connect an almost unlimited number of I2C devices

We can use I2C with even slower microcontrollers with general purpose I/O pins since they only need to generate the correct Start and Stop conditions in addition to functions for reading and writing a byte.

### Bus Connections

In I2C, we only need two wires, but these two wires can support up to 1008 slaves. We can also have more than one master, which is more realistic. For example, computers often have multiple cores in a processor, or even multiple processors. This is one of the ways in which we can achieve that.

Most I2C devices can communicate at or . There is some overhead, since for every bits of data, there will be bit for the ACK/NAK. The hardware required is also more complex than SPI, but less than asynchronous serial. It is fairly trivial to implement in software.



### Terminology

* **Transmitter** – The device that transmits data to the bus.
* **Receiver** – The device that receives data from the bus.
* **Master** – The device that generates clock cycles, starts communication, since I2C commands and stop communication.
* **Slave** – The device that listens to the bus and is addressed by the master.
* **Multi-master** – The scenario where there is more than one master.
* **Synchronization** – A process to synchronize the clocks of two or more devices.
* **Arbitration** – A process to determine which of the masters on the bus can use it when multiple masters need to use the bus.