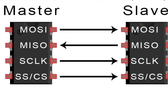
SPI - Serial Peripheral Interface

When you connect a microcontroller to a sensor, display, or other module, do you ever think about how the two devices talk to each other? What exactly are they saying? How are they able to understand each other?

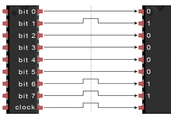


Communication between electronic devices is like communication between humans. Both sides need to speak the same language. In electronics, these languages are called communication protocols. Luckily for us, there are only a few communication protocols we need to know when building most embedded systems projects. Three most common protocols are Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I2C), and Universal Asynchronous Receiver/Transmitter (UART).
SPI, I2C, and UART are quite a bit slower than protocols like USB, ethernet, Bluetooth, and WiFi, but they're a lot more simple and use less hardware and system resources. SPI, I2C, and UART are ideal for communication between microcontrollers and between microcontrollers and sensors where large amounts of high speed data don't need to be transferred.

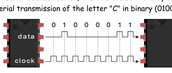
Serial Communication vs. Parallel Communication

Electronic devices talk to each other by sending bits of data through wires physically connected between devices. A bit is like a letter in a word, except instead of the 26 letters (in the English alphabet), a bit is binary and can only be a 1 or 0. Bits are transferred from one device to another by quick changes in voltage. In a system operating at 5 V, a 0 bit is communicated as a short pulse of 0 V, and a 1 bit is communicated by a short pulse of 5 V. The bits of data can be transmitted either in parallel or serial form.

In parallel communication, the bits of data are sent all at the same time, each through a separate wire. The following diagram shows the parallel transmission of the letter "C" in binary (01000011):



In serial communication, the bits are sent one by one through a single wire. The following diagram shows the serial transmission of the letter "C" in binary (01000011):



SPI is a common communication protocol used by many different devices. For example, SD card modules, RFID card reader modules, shift registers, sensors, real-time clocks (RTCs), analog-to-digital converters, and 2.4 GHz wireless transmitter/receivers all use SPI 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 as opposed to I2C and UART where data is sent in packets, limited to a specific number of bits, requiring start and stop conditions to define the beginning and end of each packet thus resulting into data interruption during transmission.

Devices communicating via SPI are in a master-slave relationship. The master is the controlling device (usually a microcontroller), while the slave (usually a sensor, display, or memory chip) takes instruction from the master. The simplest configuration of SPI is a single master, single slave system, but one master can control more than one slave.

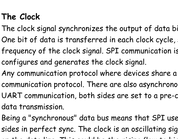
MOSI (Master Output/Slave Input) - Line for the master to send data to the slave.
MISO (Master Input/Slave Output) - Line for the slave to send data to the master.
SCLK (Clock) - Line for the clock signal.
SS/CS (Slave Select/Chip Select) - Line for the master to select which slave to send data to.

Table of Specifications

Wires Used	4
Maximum Speed	Up to 10 Mbps
Synchronous or Asynchronous	Synchronous
Serial or Parallel	Serial
Maximum number of masters	1
Maximum number of slaves	Theoretically Unlimited*

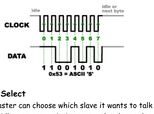
*In practice, the number of slaves is limited by the load capacitance of the system, which reduces the ability of the master to accurately switch between voltage levels.

The Serial Peripheral Interface (SPI) bus was developed by Motorola in the mid-1980s and has become a de facto standard. SPI provides full-duplex synchronous serial communication between master and slave devices.



The Clock

The clock signal synchronizes the output of data bits from the master to the sampling of bits by the slave. One bit of data is transferred in each clock cycle, so the speed of data transfer is determined by the frequency of the clock signal. SPI communication is always initiated by the master since the master configures and generates the clock signal.
Any communication protocol where devices share a clock signal is known as synchronous. SPI is a synchronous communication protocol. There are also asynchronous methods that don't use a clock signal. For example, in UART communication, both sides are set to a pre-configured baud rate that dictates the speed and timing of data transmission.
Being a "synchronous" data bus means that SPI uses separate lines for data and a "clock" that keeps both sides in perfect sync. The clock is an oscillating signal that tells the receiver exactly when to sample the bits on the data line. This could be the rising (low to high) or falling (high to low) edge of the clock signal; the datasheet will specify which one to use. When the receiver detects that edge, it will immediately look at the data line to read the next bit (see the arrows in the below diagram). Because the clock is sent along with the data, specifying the speed isn't important, although devices will have a top speed at which they can operate (We'll discuss choosing the proper clock edge and speed in a bit).

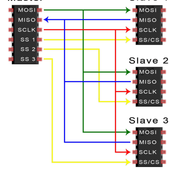


Slave Select

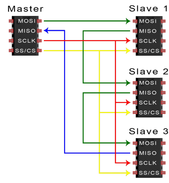
The master can choose which slave it wants to talk to by setting the slave's CS/SS line to a low voltage level. In the idle, non-transmitting state, the slave select line is kept at a high voltage level. Multiple CS/SS pins may be available on the master, which allows for multiple slaves to be wired in parallel. If only one CS/SS pin is present, multiple slaves can be wired to the master by daisy-chaining.

Multiple Slaves

SPI can be set up to operate with a single master and a single slave, and it can be set up with multiple slaves controlled by a single master. There are two ways to connect multiple slaves to the master. If the master has multiple slave select pins, the slaves can be wired in parallel like in the picture below.
this:



If only one slave select pin is available, the slaves can be daisy-chained like this:



MOSI and MISO

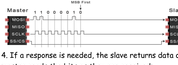
The master sends data to the slave bit by bit, in serial through the MOSI line. The slave receives the data sent from the master at the MOSI pin. Data sent from the master to the slave is usually sent with the most significant bit first.
The slave can also send data back to the master through the MISO line in serial. The data sent from the slave back to the master is usually sent with the least significant bit first.

Steps of SPI data transmission

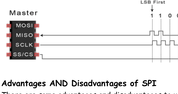
1. The master outputs the clock signal:

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

3. The master sends the data one bit at a time to the slave along the MOSI line. The slave reads the bits as they are received:



4. If a response is needed, the slave returns data one bit at a time to the master along the MISO line. The master reads the bits as they are received:



Advantages AND Disadvantages of SPI

There are some advantages and disadvantages to using SPI, and if given the choice between different communication protocols, you should know when to use SPI according to the requirements of your project:

Advantages

- No start and stop bits, so the data can be streamed continuously without interruption
- No complicated slave addressing system like I2C
- Higher data transfer rate than I2C (almost twice as fast)
- Separate MISO and MOSI lines, so data can be sent and received at the same time

Disadvantages

- Uses four wires (I2C and UARTs use two)
- No acknowledgement that the data has been successfully received (I2C has this)
- No form of error checking like the parity bit in UART
- Only allows for a single master