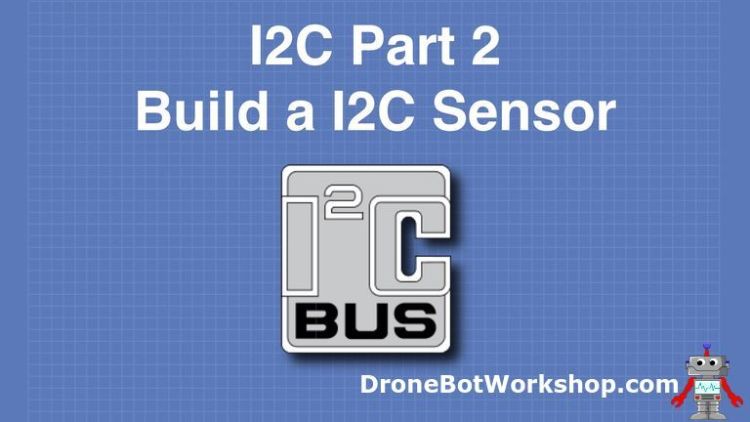
Inside I2C

As you’ll recall from the last article and video, I2C is a method of exchanging data between integrated circuits, sensors, microcomputers and microcontrollers. It uses two wires for communications and another two for power and ground.



The two communications connections are as follows:

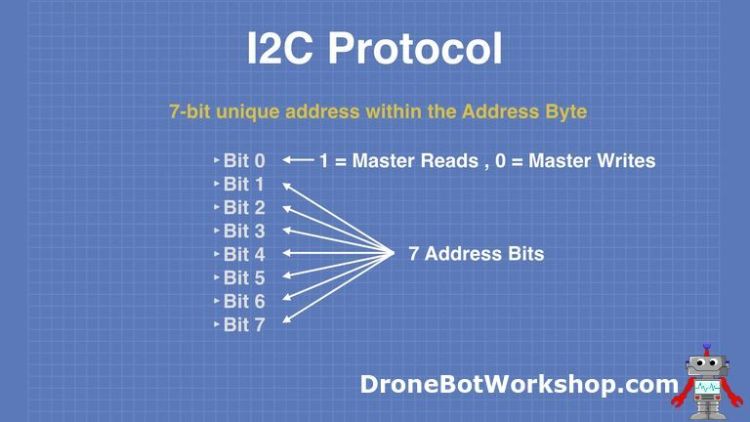
* **SDA** – This is the serial data line. I2C is a unidirectional system, so data can only travel in one direction at any given moment.
* **SCL** – The clock signal, which synchronizes the data.

I2C works on the principle of Masters and Slaves, the Master provides the clock signal and orchestrates all of the communications. Only one device can be master at any given moment, there can be several Slaves.

I2C Addressing

Every I2C Slave has an address that is unique on the I2C bus.  Masters do not have addresses. The address is used by the Master to communicate with a Slave, one device at a time.

Most I2C devices use a 7-bit addressing scheme, some newer devices use a 10-bit address.



In the more common 7-bit addressing system the lower bit (bit 0) is used to determine if the master wants to send data to the slave or read data from the slave.

* If Bit 0 is HIGH then the master is requesting to read data from the slave.
* If Boit 0 is LOW then the master is going to write data to the slave.

The remaining 7-bits are the I2C address of the slave that the master wants to communicate with.

Reserved Addresses

Using 7-bits for addressing will, in theory, allow a maximum of 128 addresses. Probably more than you’ll need for your project.

However, that isn’t actually true.

There are really “only” 112 addresses available on an I2C bus, as there are several reserved addresses. A couple of these addresses are used by the Master when making a call to all I2C devices, one is used to change into 10-bit addressing mode and several are reserved for future purposes.

The [I2C Bus Organization](https://www.i2c-bus.org/) has provided the date for the following table. If you are building an I2C device you should avoid these addresses, as they are reserved:

|  |  |
| --- | --- |
| Address in Binary, MSB on the left | Purpose |
| 0000000 0 | [General Call](https://www.i2c-bus.org/addressing/general-call-address/) |
| 0000000 1 | [Start Byte](https://www.i2c-bus.org/addressing/start-byte/) |
| 0000001 X | [CBUS Addresses](https://www.i2c-bus.org/addressing/cbus-addresses/) |
| 0000010 X | [Reserved for Different Bus Formats](https://www.i2c-bus.org/addressing/different-bus-formats/) |
| 0000011 X | Reserved for future purposes |
| 00001XX X | [High-Speed Master Code](https://www.i2c-bus.org/addressing/high-speed/) |
| 11110XX X | [10-bit Slave Addressing](https://www.i2c-bus.org/addressing/10-bit-addressing/) |
| 11111XX X | Reserved for future purposes |

Assigned Addresses

In addition to the reserved addresses, you need to avoid using the address of another I2C device that is used on your I2C bus implementation.

Usually, this is just a matter of determining what addresses are in use, what ones you might possibly use in the future and then just picking an address for your home-brew device that doesn’t conflict.

If, however, you are creating an I2C device that will be used by other people then address selection becomes a bit more challenging.

If you are building a commercial product then the I2C Bus Committee at NXP can assign you an address, for a fee.

Otherwise for hobby and non-profit projects you are pretty well on your own.

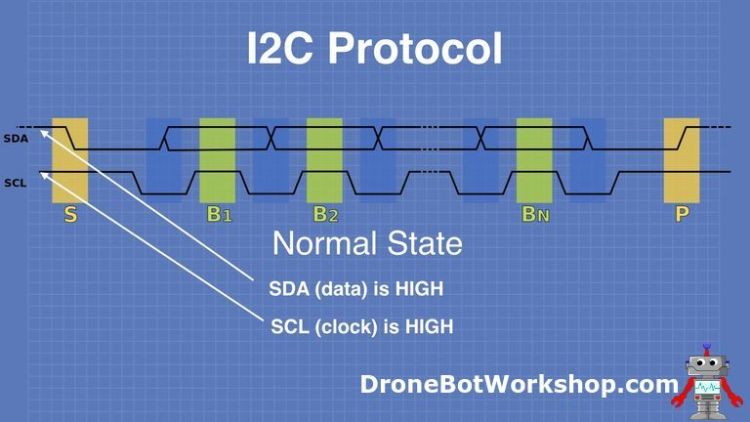
Oddly enough, there is no “master list” of I2C bus assignments. The closest thing I have been able to find is [Adafruit’s I2C Address Compilation](https://www.i2c-bus.org/addressing/10-bit-addressing/).

I2C Protocol

As I’ve already mentioned, the Master device initiates the communications and supplies the clock signal. It is not possible for a Slave device to initiate communications, it needs to wait until it is called by the Master.

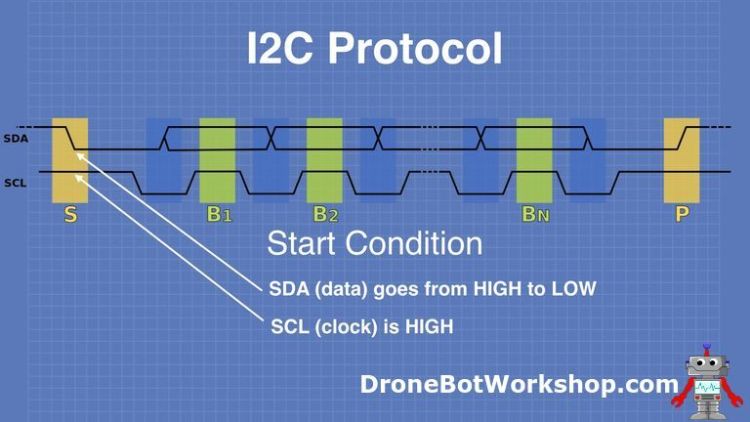
The sequence of communications is as follows:

Normal State

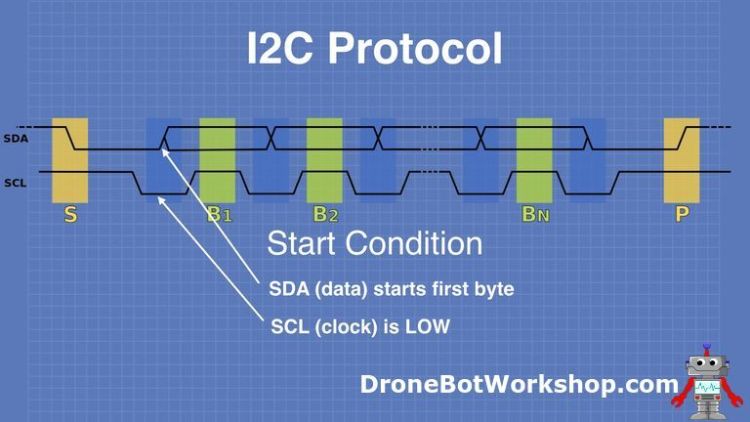


When the I2C bus is idle both the SDA (data) and SCL (clock) lines are held HIGH by the Master.

Start Condition

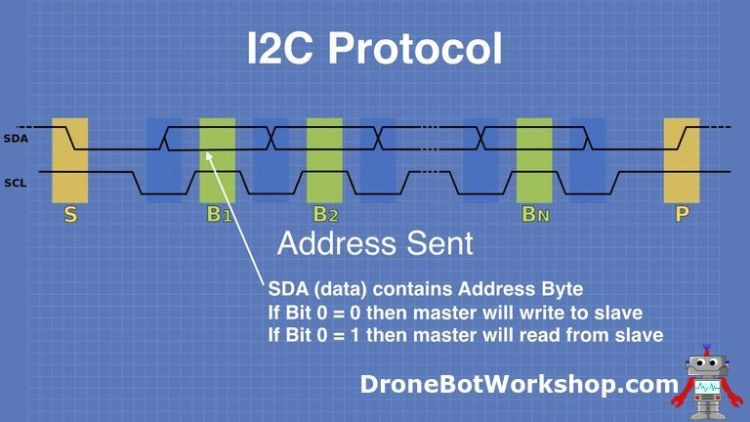


In order the begin exchanging data the Master puts the bus into Start Condition. It signifies this by changing the state of the DSDA line from HIGH to LOW. At this point, the SCL line is still held HIGH.



In the second phase of the Start Condition, the SCL line drops LOW. The SDA line now transmits the first byte of data, which is the beginning of the address.

Address Sent

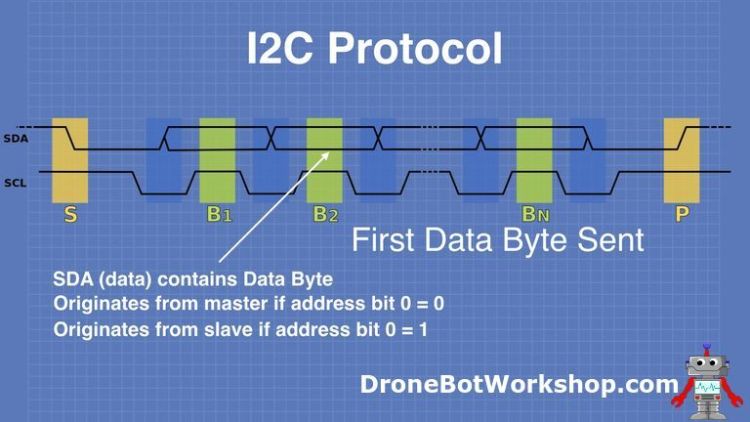


The Address is sent and is received by all of the attached slaves. It can be one of two types of addresses:

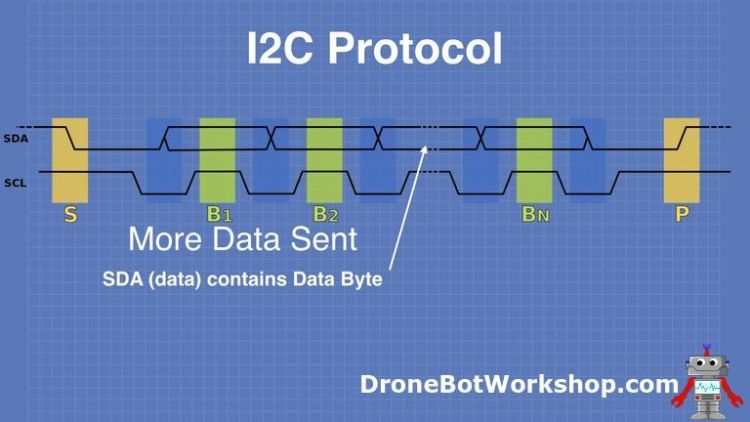
* A General Call (all zeros), which is a broadcast message to all of the slave devices.
* A specific slave address.

The first bit of the address byte is examined to see if the master is about to send data or if it is requesting data. This will determine the source of data for the next several operations.

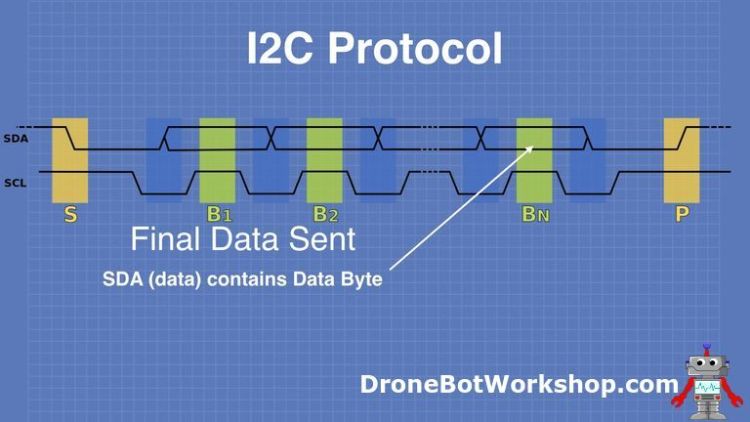
Sending Data



The data is now transferred on the I2C bus, it’s origin depends upon the previous address byte.



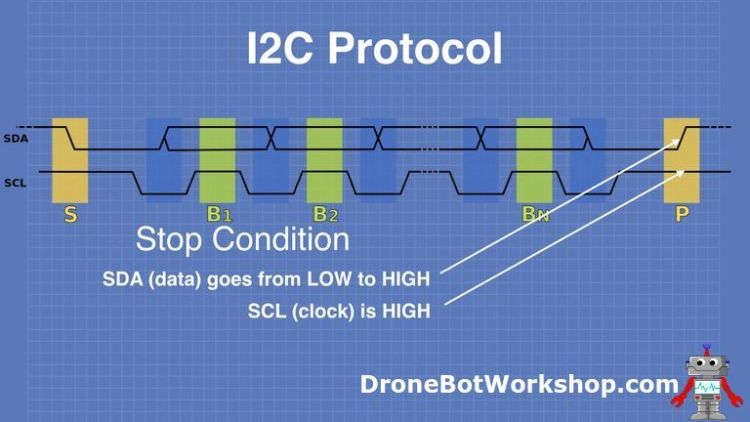
This may be one or many bytes of data sent, these bytes are synchronized by the clock signal provided by the Master on the SCL line.



The last byte of data is no different than the previous bytes. If the slave is sending data them the Master should have already determined how much data it expects, so it knows when it is the last byte. The number of bytes will vary depending upon the slave device.

After the last byte is sent the SDA line will be held LOW.

Stop Condition



The final stage is the Stop Condition. This is signified by the SDA line transitioning from LOW to HIGH, the opposite sequence from the Start Condition.  The SCL clock line will go HIGH.