I2C

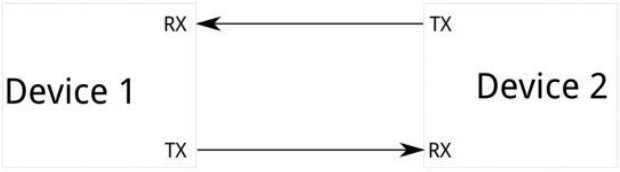
Introduction

The Inter-integrated Circuit (I^2C) Protocol is a protocol intended to allow multiple "slave" digital integrated circuits ("chips") to communicate with one or more "master" chips. Like the Serial Peripheral Interface (SPI), it is only intended for short distance communications within a single device. Like Asynchronous Serial Interfaces (such as RS-232 or UARTs), it only requires two signal wires to exchange information.

Why Use I2C?

To figure out why one might want to communicate over I^2C, you must first compare it to the other available options to see how it differs.

What's wrong with Serial Ports?



Because serial ports are **asynchronous** (no clock data is transmitted), devices using them must agree ahead of time on a data rate.

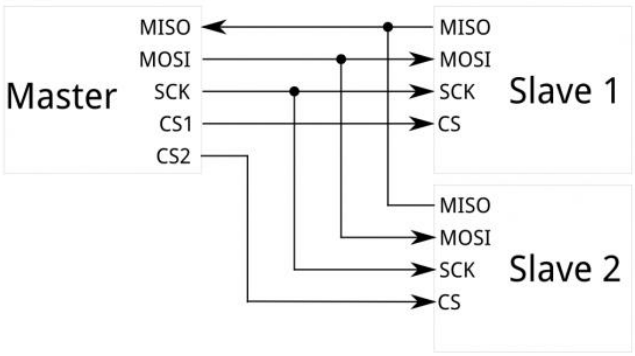
The two devices must also have clocks that are close to the same rate, and will remain so–excessive differences between clock rates on either end will cause garbled data.

Asynchronous serial ports require hardware overhead–the UART at either end is relatively complex and difficult to accurately implement in software if necessary. At least one start and stop bit is a part of each frame of data, meaning that 10 bits of transmission time are required for each 8 bits of data sent, which eats into the data rate.

Another core fault in asynchronous serial ports is that they are inherently suited to communications between two, and only two, devices. While it is *possible* to connect multiple devices to a single serial port, **bus contention** (where two devices attempt to drive the same line at the same time) is always an issue and must be dealt with carefully to prevent damage to the devices in question, usually through external hardware.

Finally, data rate is an issue. While there is no *theoretical* limit to asynchronous serial communications, most UART devices only support a certain set of fixed baud rates, and the highest of these is usually around 230400 bits per second.

What's Wrong with SPI?

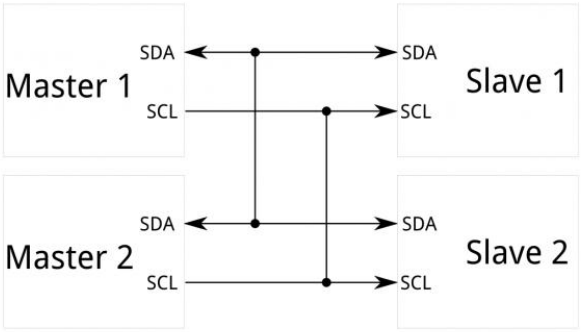


The most obvious drawback of SPI is the number of pins required. Connecting a single master to a single slave with an SPI bus requires four lines; each additional slave requires one additional chip select I/O pin on the master. The rapid proliferation of pin connections makes it undesirable in situations where lots of devices must be slaved to one master. Also, the large number of connections for each device can make routing signals more difficult in tight PCB layout situations. SPI only allows one master on the bus, but it does support an arbitrary number of slaves (subject only to the drive capability of the devices connected to the bus and the number of chip select pins available).

SPI is good for high data rate **full-duplex** (simultaneous sending and receiving of data) connections, supporting clock rates upwards of 10MHz (and thus, 10 million bits per second) for some devices, and the speed scales nicely.

The hardware at either end is usually a very simple shift register, allowing easy implementation in software.

Enter I^{2}C - The Best of Both Worlds!



I^{2}C requires a mere two wires, like asynchronous serial, but those two wires can support up to 1008 slave devices. Also, unlike SPI, I^{2}C can support a multi-master system, allowing more than one master to communicate with all devices on the bus (although the master 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^{2}C devices can communicate at 100kHz or 400kHz. There is some overhead with I^{2}C; for every 8 bits of data to be sent, one extra bit of meta data (the "ACK/NACK" bit, which we'll discuss later) must be transmitted.

The hardware required to implement I^{2}C is more complex than SPI, but less than asynchronous serial. It can be fairly trivially implemented in software.

I^2C - A Brief History

I^2C was originally developed in 1982 by Philips for various Philips chips. The original spec allowed for only 100kHz communications, and provided only for 7-bit addresses, limiting the number of devices on the bus to 112 (there are several reserved addresses, which will never be used for valid I^2C addresses). In 1992, the first public specification was published, adding a 400kHz fast-mode as well as an expanded 10-bit address space. Much of the time (for instance, in the ATMega328 device on many Arduino-compatible boards) , device support for I^2C ends at this point. There are three additional modes specified: fast-mode plus, at 1MHz; high-speed mode, at 3.4MHz; and ultra-fast mode, at 5MHz.

In addition to “vanilla” I^2C, Intel introduced a variant in 1995 call “System Management Bus” (SMBus). SMBus is a more tightly controlled format, intended to maximize predictability of communications between support ICs on PC motherboards. The most significant difference between SMBus is that it limits speeds from 10kHz to 100kHz, while I^2C can support devices from 0kHz to 5MHz. SMBus includes a clock timeout mode which makes low-speed operations illegal, although many SMBus devices will support it anyway to maximize interoperability with embedded I^2C systems.

Source: https://learn.sparkfun.com/tutorials/i2c