I2C Protocol

http://www.firmcodes.com/i2c-protocol/

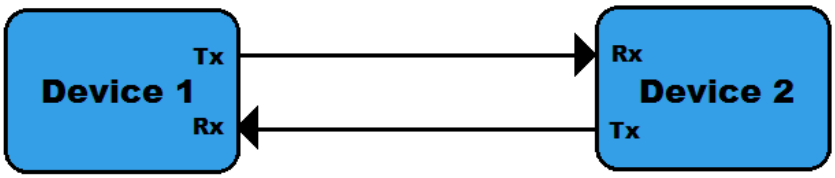
Introduction

Usually embedded systems are complex design which include multiple peripherals interconnected like LED, LCD, RTC, other microcontroller, Memory card, etc with main computer or master to expand its capabilities. All these peripheral's interfacing is done by standard rules which is called protocol. Protocol is a set of rules that defines how communication between systems and devices are done which include bit ordering, bit pattern meanings, creating data frames, error checking, etc. UART, SPI, I2C, USB and Ethernet are some of the protocols widely used in embedded systems for serial data communication. Here we are bound to I2C Protocol.

The **I**nter-**i**ntegrated **C**ircuit (I2C) Protocol was originally developed in 1982 by Philips(now known as NXP) used to connect multiple devices with only two wire. Its is also called two wire interface protocol.

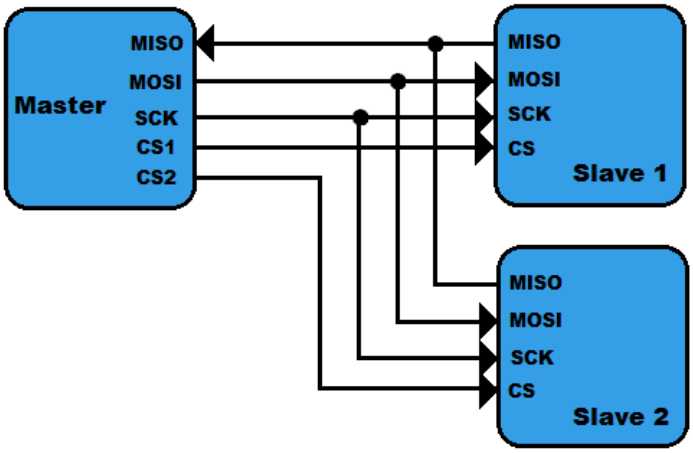
Why I2C Protocol ?

Comparing I2C with UART



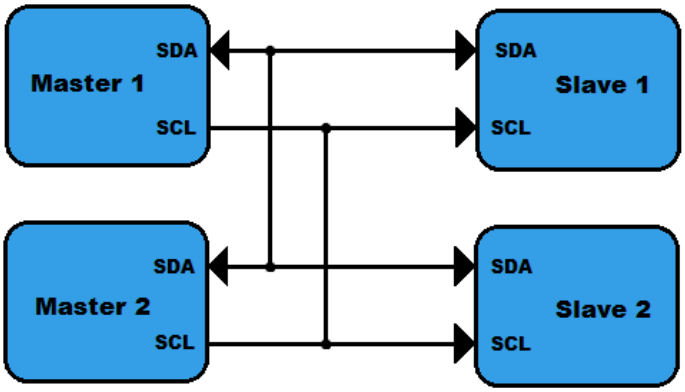
- In UART, separate clock line is not used with data transmission so both device should agree on same Baud-rate. If differences occur between Baud-rate on either end will cause garbled data while in I2C separate clock line is used.
- UART uses hardware overheads like start bit, stop bit, partiy bit or CRC bit in each frame of transmission which will lead to increase in transmission time while I2C use only acknowledgement bit(ACK) or no acknowledgement bit.
- Another drawback of UART is that they are suited to communications between two, and only two, devices while I2C can used with multiple devices and different modes like multi-master or single master, etc.
- Data transfer speed is also an issue for UART, they are limited to some extent and maximum baud-rate is around 230400 bps.

Comparing I2C with SPI



- The first drawback of SPI is pin used to connect multiple devices. If we connect two device with SPI standard it will use four pins MISO, MOSI, SCK, CS.
- In SPI protocol, there is only single master but it can support arbitrary number of slaves. While in I2C, multiple master can be possible.
- If we compare data transmission speed of SPI and I2C, then SPI will win definitely. SPI support speed upto 10MHz (10 Mbps).

What's good in I2C ?

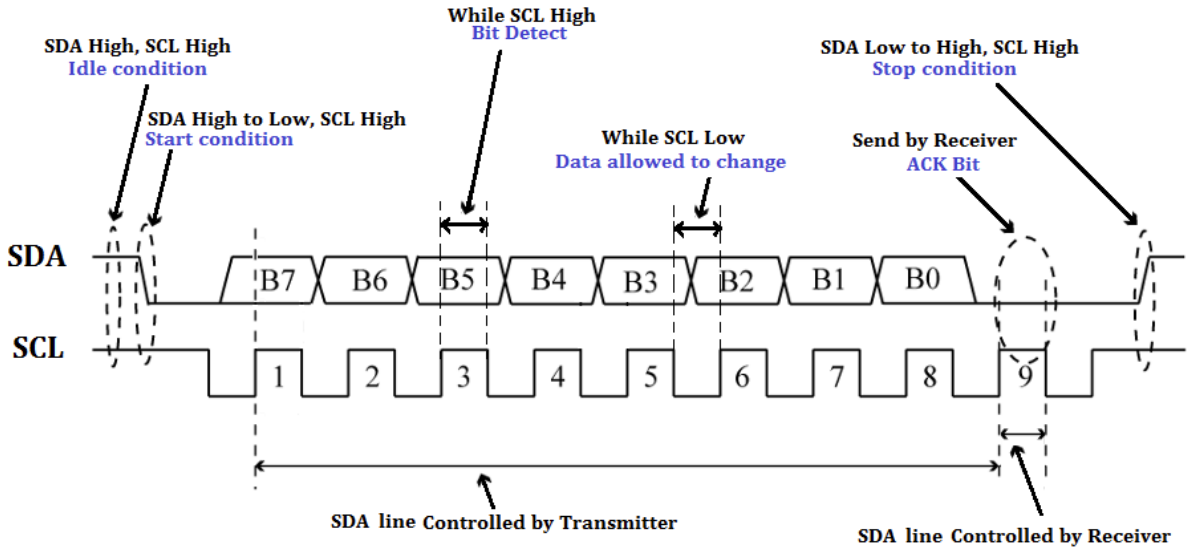


- I2C requires only two line(two wire), but those two wires can support up to 1008 slave devices.
- Unlike SPI, I2C can support a multi-master system, allowing more than one master to communicate with all devices on the bus (although the master devices can't talk to each other over the bus and must take turns using the bus lines).
- Data rates of I2C devices can communicate at the rate of 100kHz to 400kHz.

How I2C Protocol Works ?

To understand the working of I2C Protocol, Let we take some practical examples

Single Byte Transfer

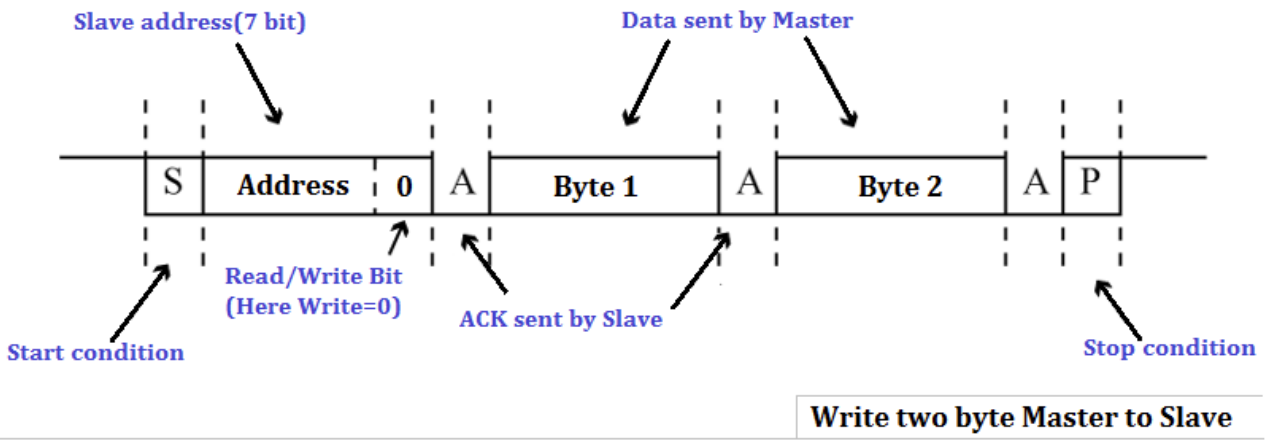


Single Byte Transaction

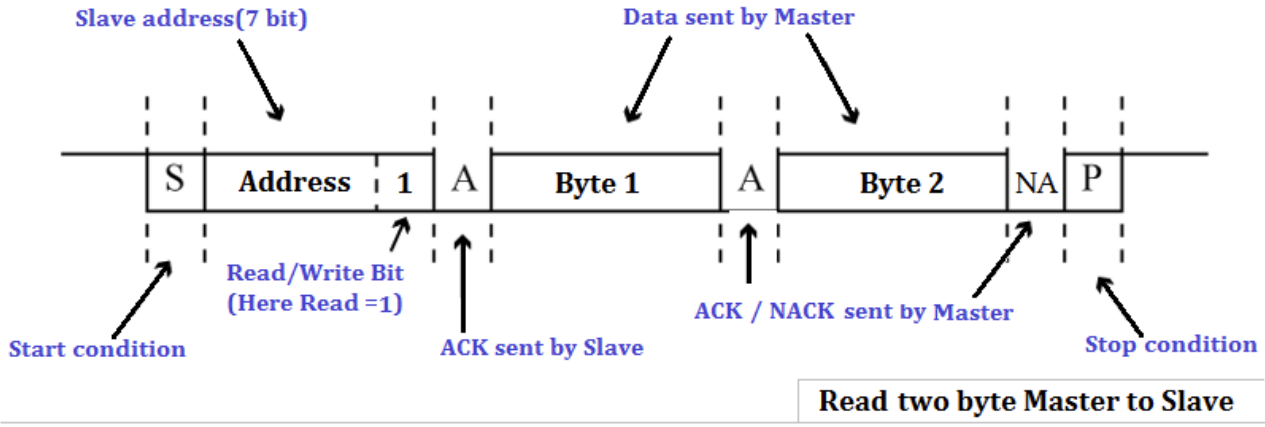
- **Idle condition**: As you can see in above figure, at the initial condition SDA and SCL both High because of both line pulled High with pull-up resister.
- **Start condition**: Data transfer is start when SDA line pulled down mean High to Low when SCL line is High.
- **Bit Detect**: Slave can understand bit reception when it receives single bit ‘1’ or ‘0’ when SCL line is High.
- **Data allowed to change**: Transition of bit from 1 to 0 or 0 to 1 is only allowed when SCL line is Low as seen in pick.
- **ACK Bit**: Transmitter must release the SDA line after transmitting 8 bit to allow the Receiver to pull SDA line Low to acknowledgement the previous 8 bit data reception.
- **Stop condition**: is only understood when SDA line goes Low to High while SCL line is High.

Note: In I2C Protocol Byte is transmitted MSB to LSB. And in multi-Byte transmission, every 8 bits has a 9th bit that is an acknowledge.

Write/Read Two Byte from Master to Slave



Write two byte Master to Slave



Read two byte Master to Slave

Start Condition

Every communication in I2C Protocol is begin with start condition which we already studied above how start condition form. This alerts all slave devices that a transmission is about to start.

Address Frame Byte

The address frame byte include device address which is of 7-bit long and Read(1)/Write(0) bit. The address frame byte is always sent first when any new communication sequence initiate.

ACK/NACK Bit

The 9th bit of the frame is the NACK/ACK bit(here NA or A). This is the case for all frames (data or address). Once the first 8 bits of the frame are sent, the receiving device is given control over SDA. If the receiving device does not pull the SDA line low before the 9th clock pulse, it can be inferred that the receiving device either did not receive the data or some error occur.

In multi-byte write process, Acknowledgement is send by slave at every single byte reception. But in mulit-byte read process, first byte(which is address of slave) Acknowledgement is send by slave and then Acknowledgement is send by master at every single byte reception and if master don't want more byte then it send No Acknowledgement to slave and generate stop condition.

Data Bytes

After the address frame byte has been sent, data bytes can begin being transmitted. The master will simply continue generating clock pulses at a regular interval, and the data will be placed on SDA by either the master or the slave, depending on whether the R/W bit indicated a read or write operation. The number of data byte is arbitrary, and most slave devices will auto-increment the internal register, meaning that subsequent reads or writes will come from the next register in line.

Stop condition

Every communication in I2C Protocol is end with stop condition which we already studied above. During normal data writing operation, the value on SDA should not change when SCL is high, to avoid false stop conditions.

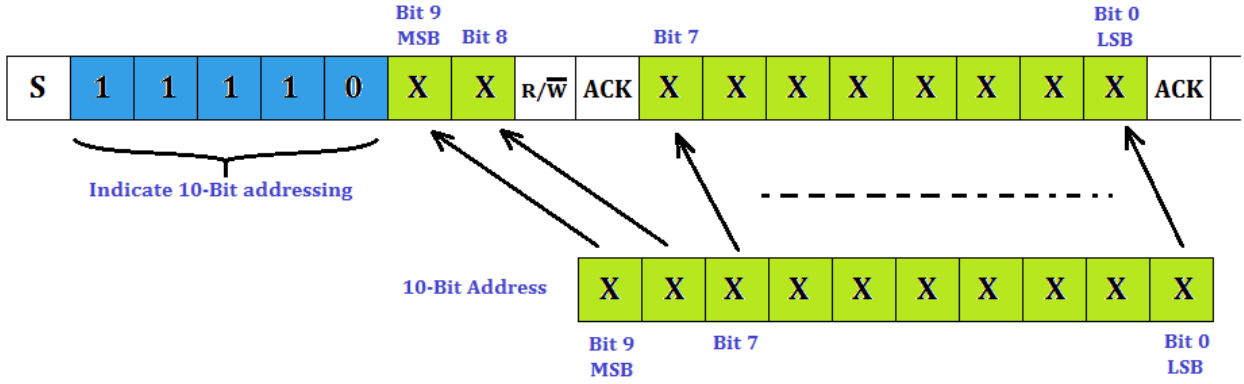
Speed of I2C Protocol

Common I²C bus speeds are the 100 kbit/s standard mode and the 10 kbit/s low-speed mode, but arbitrarily low clock frequencies are also allowed. Recent revisions of I²C can run at faster speeds typically 400 kbit/s Fast mode, 1 Mbit/s Fast mode plus or Fm+, and 3.4 Mbit/s High Speed mode.

More About I2C Protocol

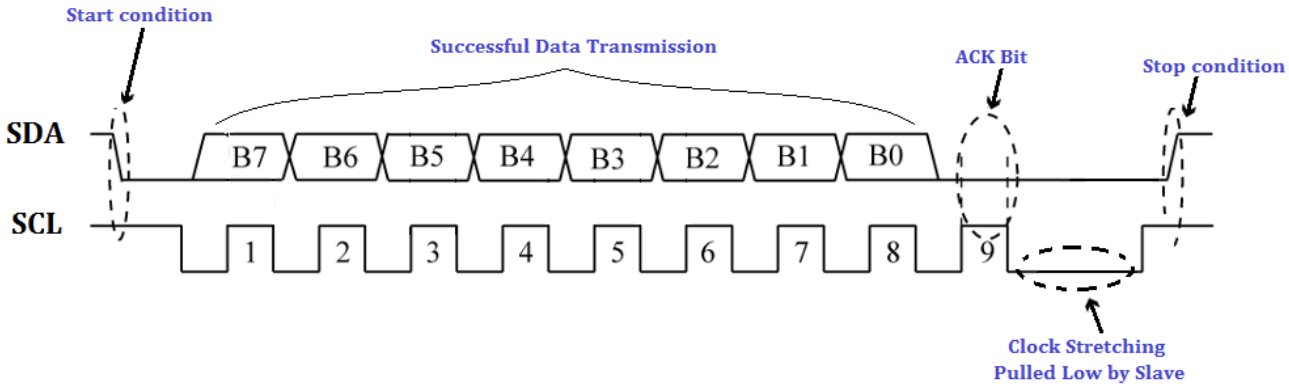
10-bit Addressing supports more number of devices

In order to increase device address capability, 10-bit addressing scheme is introduced to I2C Protocol by which we can connect more number of devices by using I2C Protocol.



In a 10-bit addressing system, two frames are required to transmit the slave address. The first frame will consist of the code 11110xyz, where ‘x’ is the MSB of the slave address, y is 8th bit of the slave address, and z is the Read/Write bit as seen above. The first frame’s ACK bit will be asserted by all slaves which match the first two bits of the address. As with a normal 7-bit transfer, another transfer begins immediately, and this transfer contains bit 0 to bit 7 of the address. At this point, the addressed slave should respond with an ACK bit. If it doesn’t, the failure mode is the same as a 7-bit system.

Clock stretching



- In an I2C communication the master device determines the clock speed. However, there are situations where an I2C slave is not able to co-operate with the clock speed given by the master and needs to slow down a little. This is done by a mechanism referred to as clock stretching.
- An I2C slave is stretching the clock by pull down the SCL if it needs to reduce the bus speed. The master on the other hand is required to read back the clock signal after releasing it to high state and wait until the line has actually gone high.

Repeated Start Conditions

- When we communicate through I2C bus, its obvious in most cases that first you need to send command and then you get response from slave, so each time you dont need to terminate communication by stop condition. You can use start condition reapetedly without terminating previous transaction. The I2C protocol defines a so-called repeated start condition.
- For example, After having sent the address byte (address and read/write bit) the master may send any number of bytes followed by a stop condition. Instead of sending the stop condition it is also allowed to send another start condition again followed by an address (and of course including a read/write bit) and more data. This is defined recursively allowing any number of start conditions to be sent.
- The purpose of this is to allow combined write/read operations to one or more devices without releasing the bus and thus with the guarantee that the operation is not interrupted.