I^2C vs SPI

I^2C (for 'Inter-Integrated Circuit', protocol) that only requires 2 wires for connecting all the peripheral to a microcontroller

SPI (for 'Serial Peripheral Interface') is used to connect the microcontroller peripherals with 4 wires

Both protocols are well-suited for communications between integrated circuits,

I^{2}C BUS

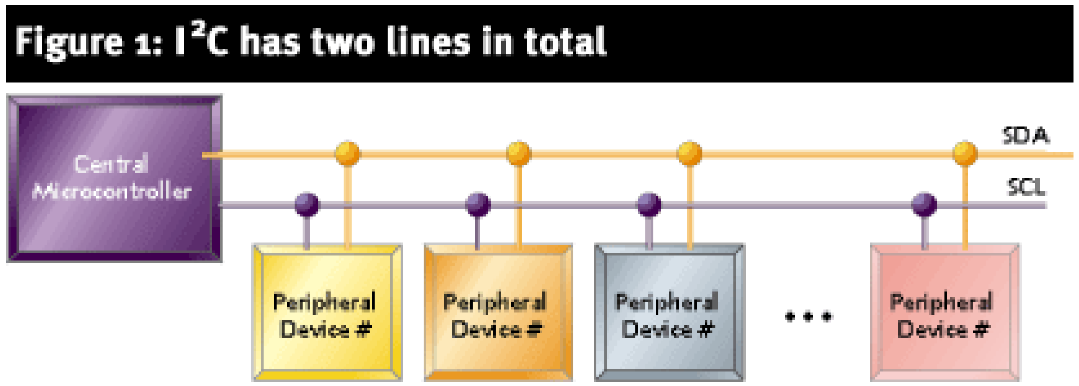
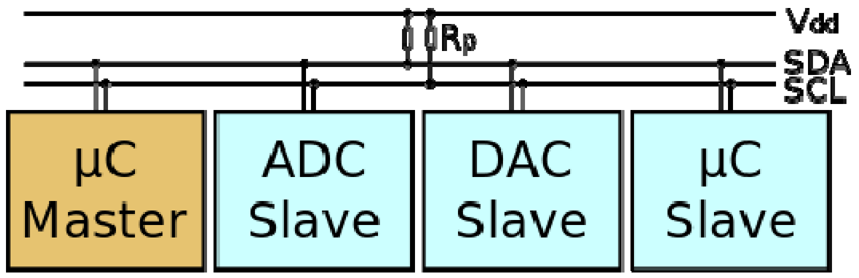


Figure 1: I^{2}C has two lines in total

- I^{2}C master is the CPU or microcontroller in the system
- I^{2}C is used for attaching low-speed peripherals to a motherboard, embedded system, or other digital electronic devices
- I^{2}C is a two-wire serial bus
- The two I^{2}C signals are serial data (SDA) and serial clock (SCL).
- Together, these above two signals make it possible to support serial transmission of 8-bit bytes of data-
7-bit slave device addresses plus control bits-over the two-wire serial bus.
- I^{2}C signaling protocol provides device addressing, a read/write flag, and a simple acknowledgement mechanism
- Standard I^{2}C devices operate up to 100Kbps while fast-mode devices operate at up to 400Kbps and 3.4Mbps

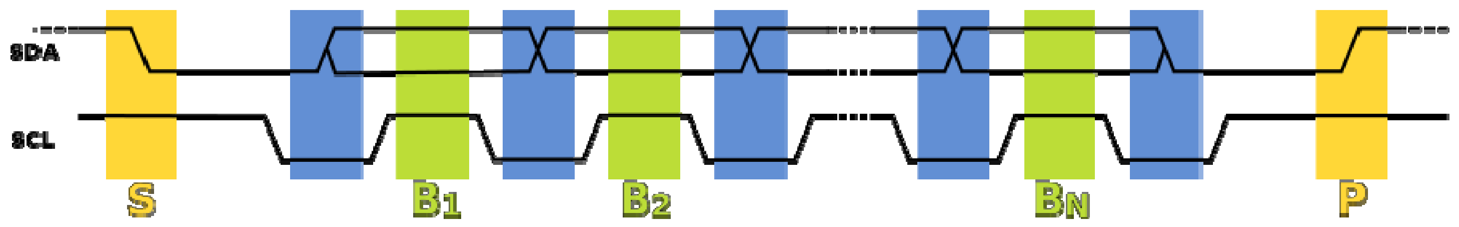
I²C Design



A sample schematic with one master (a microcontroller), three slave nodes (an ADC, a DAC, and a microcontroller), and pull-up resistors R_p
The I²C reference design has a 7-bit or a 10-bit (depending on the device used) slave address space

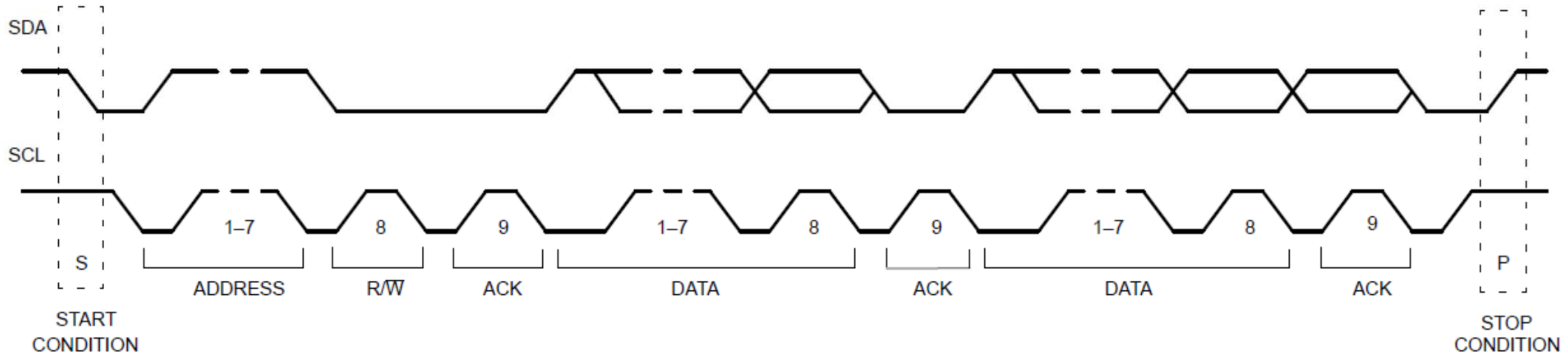
I²C uses only two bidirectional open-drain lines,
Serial Data Line (SDA) and
Serial Clock Line (SCL), pulled up with resistors.
Typical voltages used are +5 V or +3.3 V

I^2C Timing Diagram



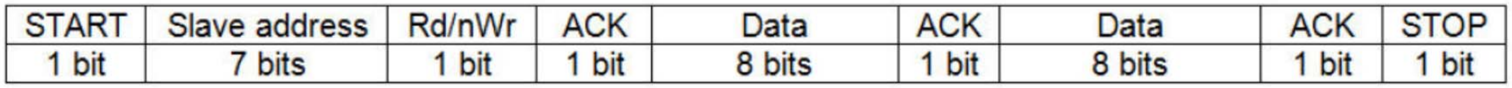
1.Data transfer is initiated with a START bit (S) signaled by SDA being pulled low while SCL stays high.
2.SDA sets the 1st data bit level while keeping SCL low (during blue bar time.) and the data is sampled (received) when SCL rises (green).
3.When the transfer is complete, a STOP bit (P) is sent by releasing the data line to allow it to be pulled high while SCL is kept high continuously.
4.In order to avoid false marker detection, the level on SDA is changed on the SCL falling edge and is sampled and captured on the rising edge of SCL.

A complete data transfer for I²C



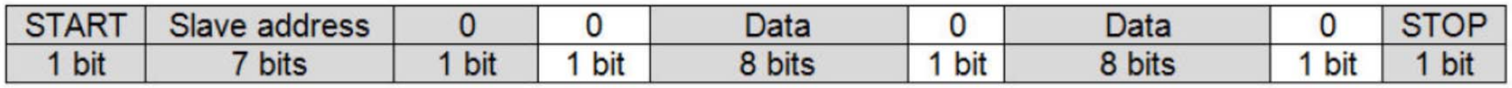
Data transfers follow the format shown in above. After the START condition (S), a slave address is sent. This address is 7 bits long followed by an eighth bit which is a data direction bit (R/W) — a ‘zero’ indicates a transmission (WRITE), a ‘one’ indicates a request for data (READ). A data transfer is always terminated by a STOP condition (P) generated by the master. However, if a master still wishes to communicate on the bus, it can generate a repeated START condition (Sr) and address another slave without first generating a STOP condition. Various combinations of read/write formats are then possible within such a transfer

Example, I^{2}C transfer with 2 bytes



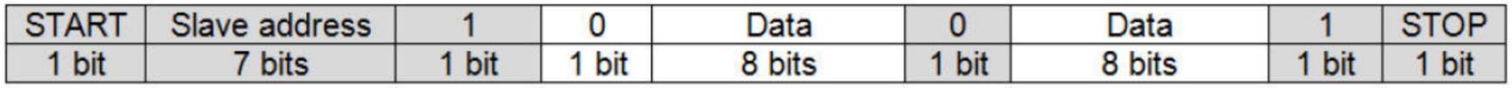
START	Slave address	Rd/nWr	ACK	Data	ACK	Data	ACK	STOP
1 bit	7 bits	1 bit	1 bit	8 bits	1 bit	8 bits	1 bit	1 bit

Example 1: writing 2 byte to a slave. The data put on the bus by the master are shaded.



START	Slave address	0	0	Data	0	Data	0	STOP
1 bit	7 bits	1 bit	1 bit	8 bits	1 bit	8 bits	1 bit	1 bit

Example 2: reading 2 bytes from a slave. The data put on the bus by the master are shaded.

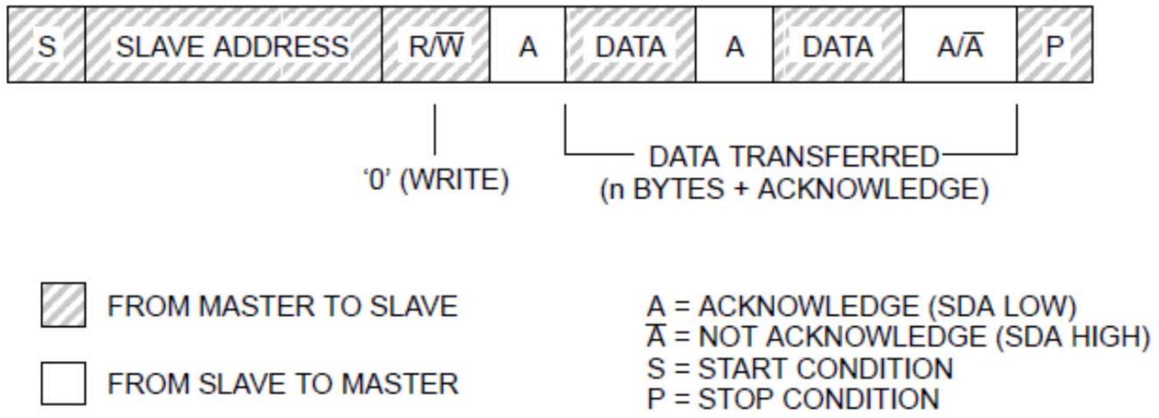


START	Slave address	1	0	Data	0	Data	1	STOP
1 bit	7 bits	1 bit	1 bit	8 bits	1 bit	8 bits	1 bit	1 bit

Typical I^{2}C transfer, with 2 bytes of data. The master initiates the transfer with a START condition, followed by the slave address and the transfer type (read or write) bit. The slave acknowledges its address. Each data byte is then transmitted and acknowledged by the receiver. When it receives data, the master can issue a not-acknowledge condition (NACK) if it received enough data. The bus is released when the master issues a STOP condition.

I²C Data Transfer Formats (3 format)

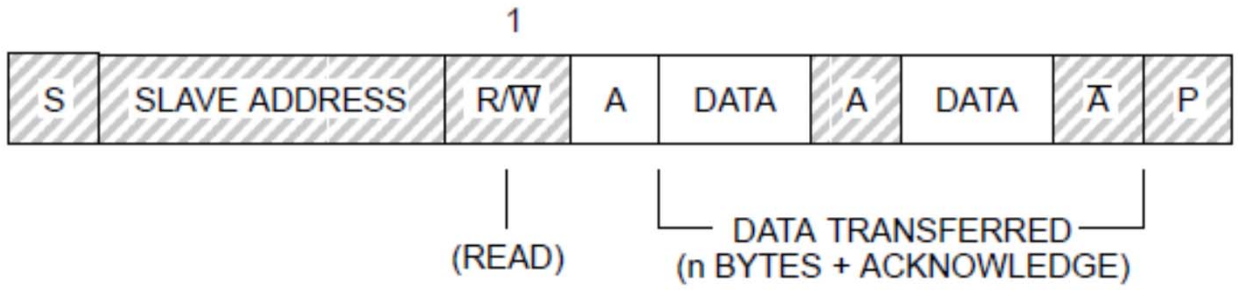
- Format 1: Master-transmitter transmits to slave-receiver. The transfer direction is not changed



A master-transmitter addresses a slave receiver with a 7-bit address.
The transfer direction is not changed

I²C Data Transfer Formats (3 format)

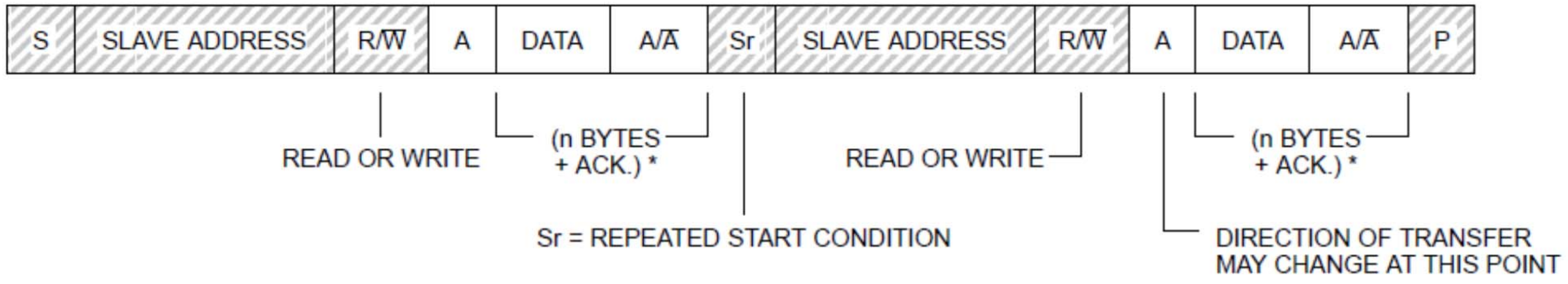
- Format 2: Master reads slave immediately after first byte



A master reads a slave immediately after the first byte

I^{2}C Data Transfer Formats (3 format)

- **Format 3: Combined format**

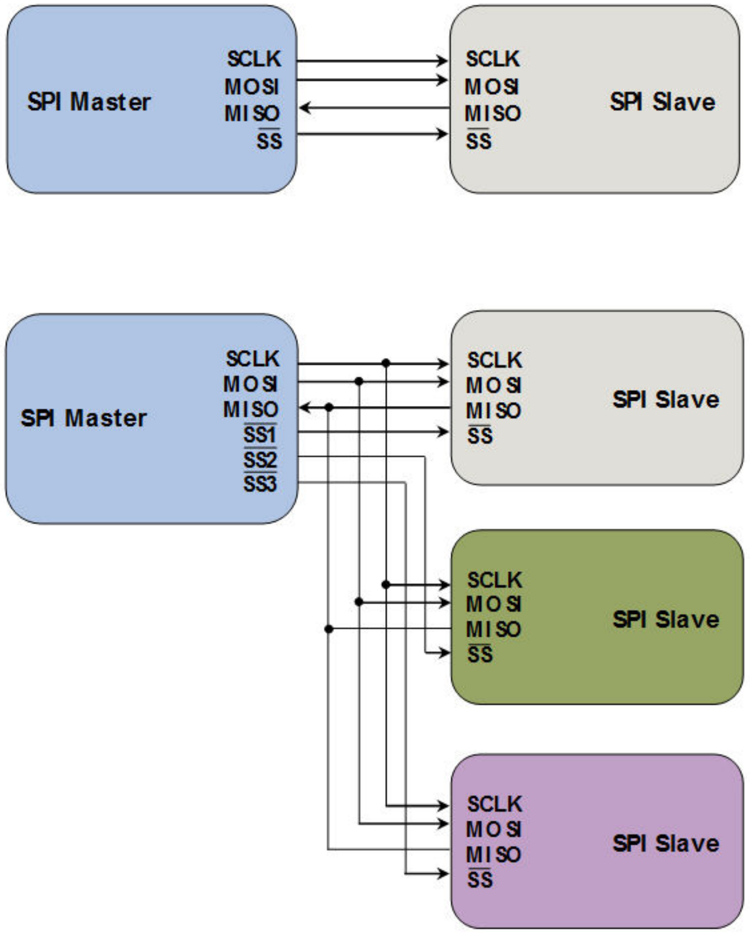


* TRANSFER DIRECTION OF DATA AND ACKNOWLEDGE BITS DEPENDS ON R/$\overline{W}$ BITS.

1. Combined formats can be used, for example, to control a serial memory. During the first data byte, the internal memory location has to be written. After the START condition and slave address is repeated, data can be transferred.
2. All decisions on auto-increment or decrement of previously accessed memory locations etc. are taken by the designer of the device.
3. Each byte is followed by an acknowledgement bit as indicated by the A or $(\text{not } A)$ blocks in the sequence.
4. I^{2}C-bus compatible devices must reset their bus logic on receipt of a START or repeated START condition such that they all anticipate the sending of a slave address.

SPI

Figure 1 : Two SPI busses topologies. The upper figure shows a SPI master connected to a single slave (point-to-point topology). The lower figure shows a SPI master connected to multiple slaves.



A protocol on 4 signal lines
- A clock signal named SCLK, sent from the bus master to all slaves; all the SPI signals are synchronous to this clock signal;
- A slave select signal for each slave, SSn, used to select the slave the master communicates with;
- A data line from the master to the slaves, named MOSI (Master Out-Slave In)
- A data line from the slaves to the master, named MISO (Master In-Slave Out).

SPI

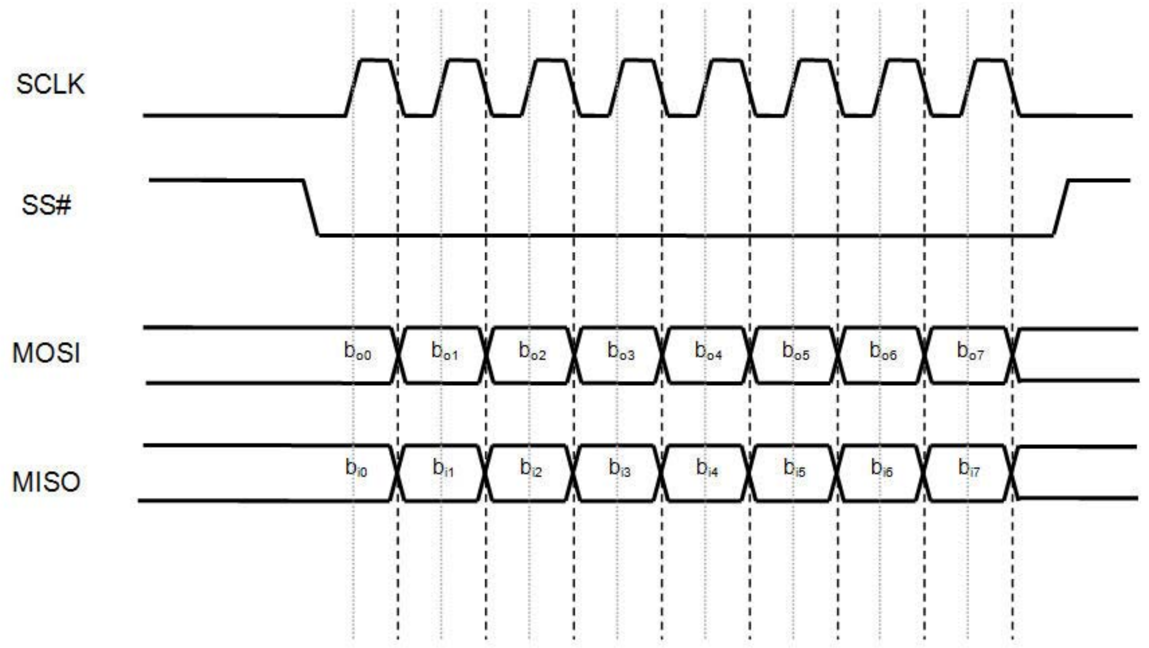

SCLK
SS#
MOSI
MISO
b_{o0} b_{o1} b_{o2} b_{o3} b_{o4} b_{o5} b_{o6} b_{o7}
b_{i0} b_{i1} b_{i2} b_{i3} b_{i4} b_{i5} b_{i6} b_{i7}

Figure 2 : A simple SPI communication. Data bits on MOSI and MISO toggle on the SCLK falling edge and are sampled on the SCLK rising edge. The SPI mode defines which SCLK edge is used for toggling data and which SCLK edge is used for sampling data.

SPI is a single-master communication protocol
SPI does not define any maximum data rate, not any particular addressing scheme; it does not have a acknowledgement mechanism to confirm receipt of data and does not offer any flow control.

Actually, the SPI master has no knowledge of whether a slave exists, unless ‘something’ additional is done outside the SPI protocol