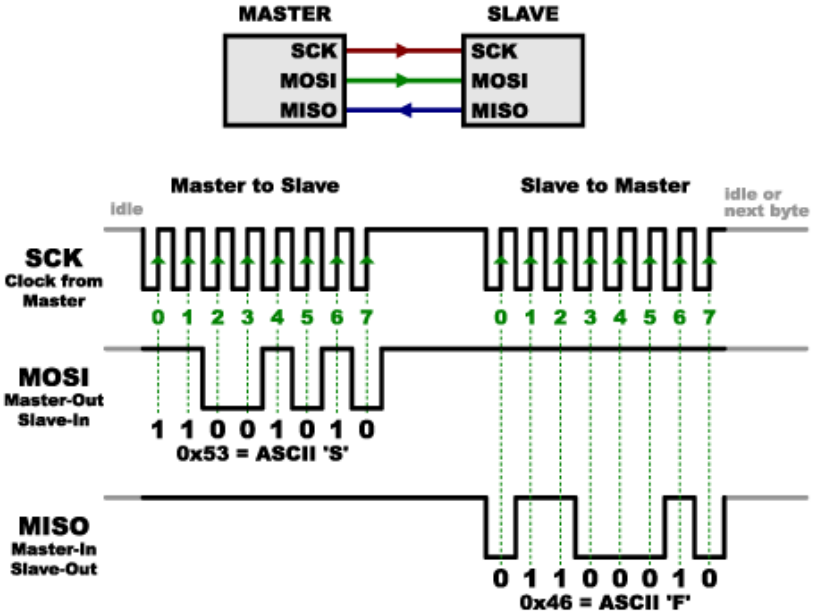
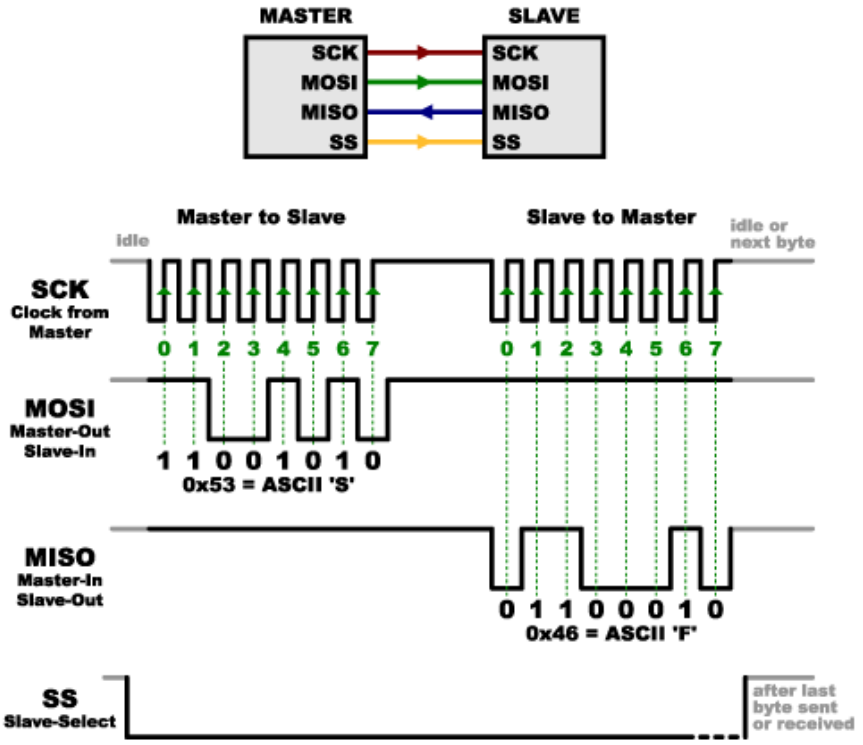
SERIAL PERIPHERAL INTERCONNECT (SPI)

- Master/slave point to point serial interface
- Optional selection signals (one for each peripherals)
- **Specifications** may **vary** widely (active levels and sampling edges)
- SPI is “full duplex” (separate send and receive lines) so, optionally, data can be transmit and received at the same time
- May be implemented simply in hardware with a **shift register** or through **bit banging**



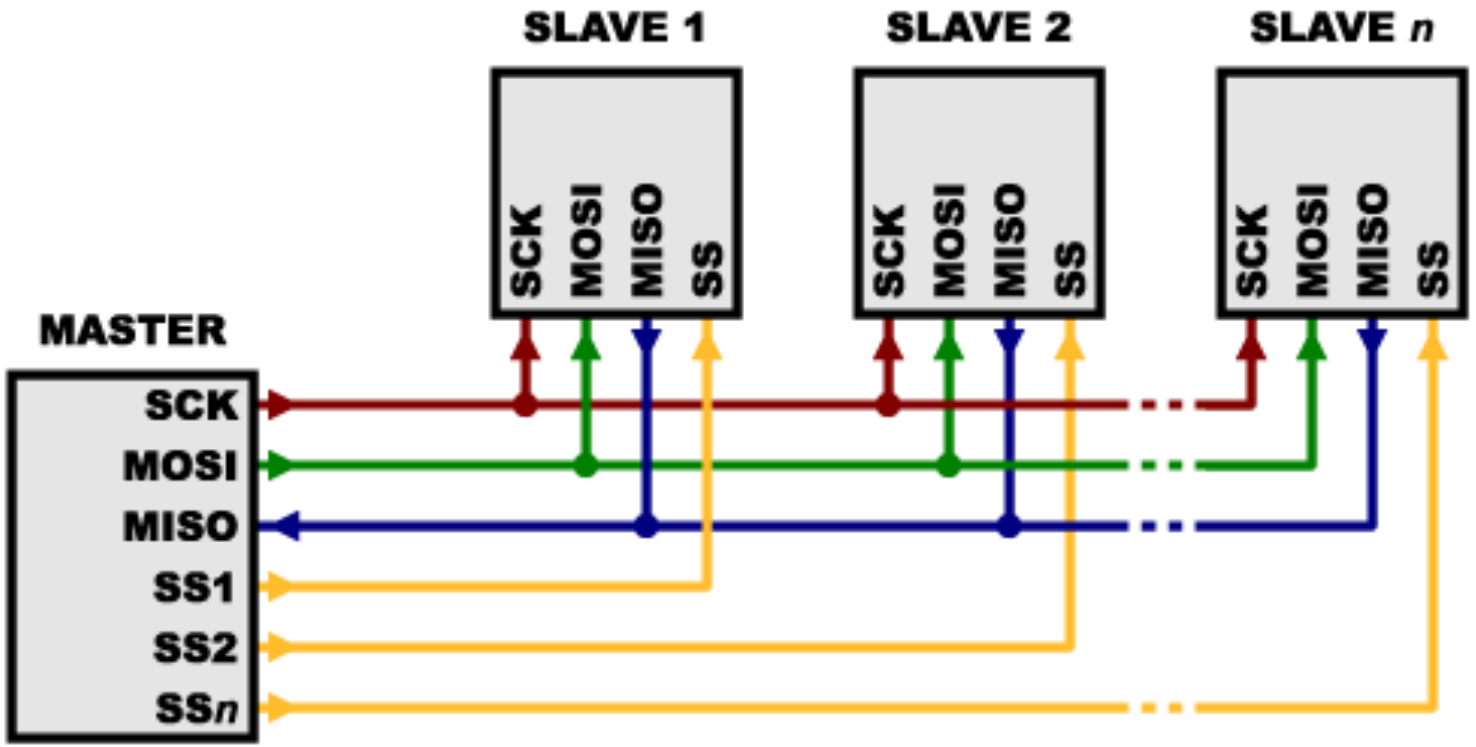
SERIAL PERIPHERAL INTERCONNECT (SPI)

- Single slave
  - Slave Select



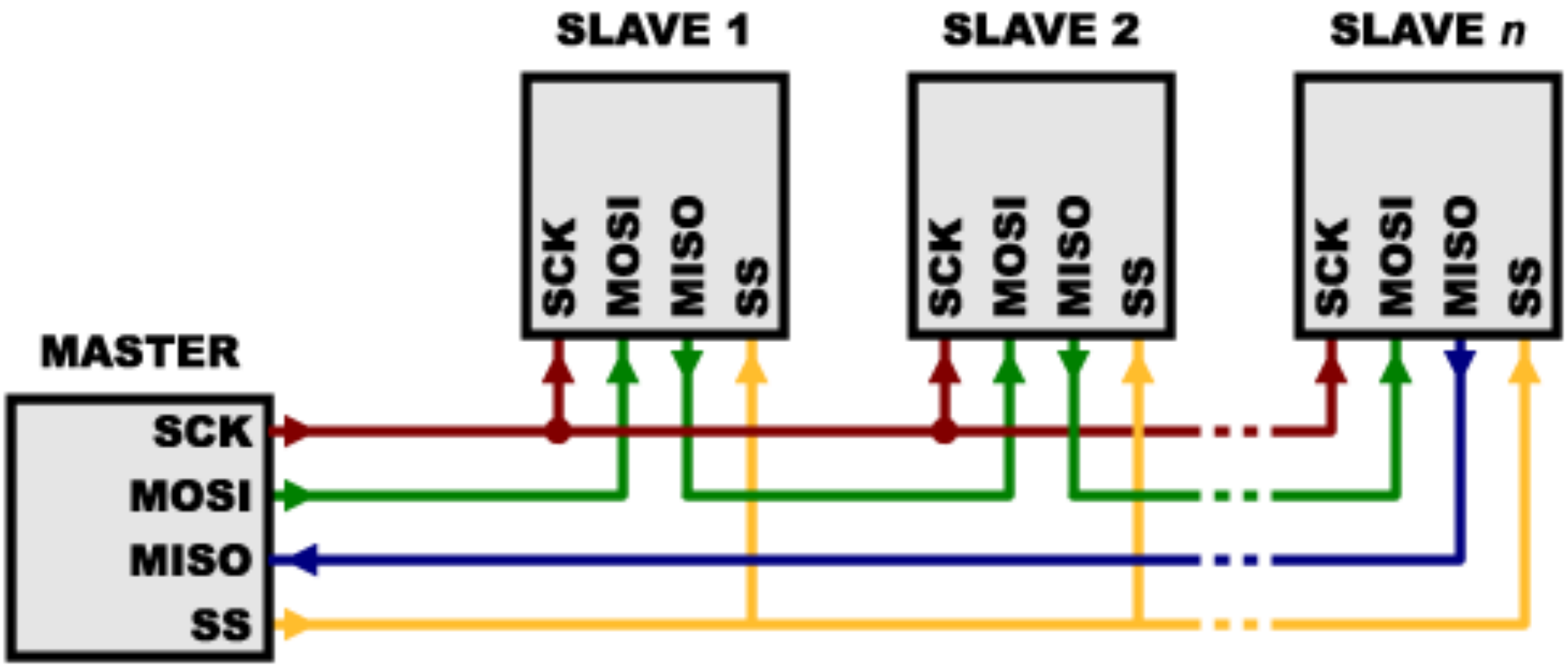
SERIAL PERIPHERAL INTERCONNECT (SPI)

- Multiple slaves
  - Multiple Slave Select (SS): each slave is addressed independently



SERIAL PERIPHERAL INTERCONNECT (SPI)

- Multiple slaves
  - Daisy-chaining: a single SS line is shared by all the slaves. Once all the data is sent, the SS line is raised, which causes all the chips to be activated simultaneously (Daisy-chained shift registers)

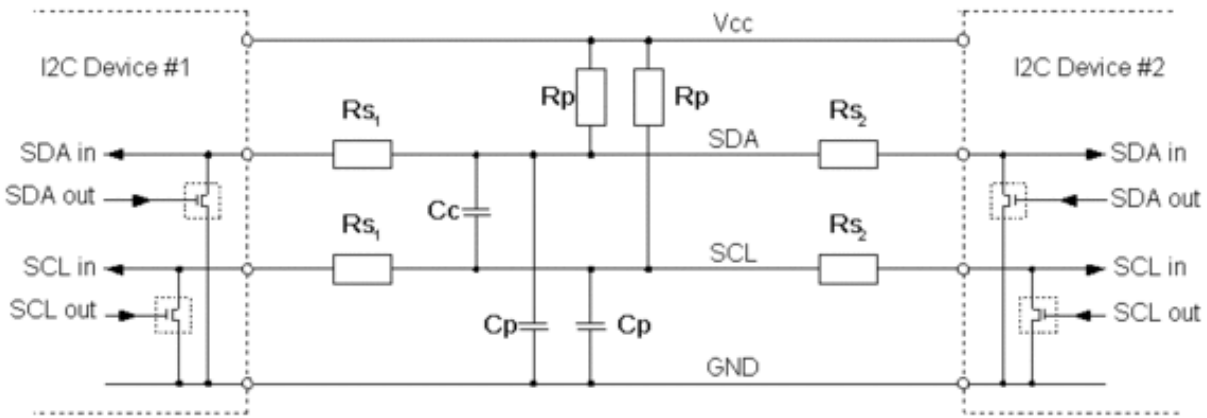


I²C BUS

- Synchronous, multi-master, multi-slave, packet switched, single-ended, serial computer bus invented by Philips Semiconductor (now NXP Semiconductors)
- Used for attaching lower-speed peripheral ICs to processors and microcontrollers in short-distance, intra-board communication
- Each device connected to the bus is software-addressable by a unique address
- Provided with arbitration and collision detection
- No strict baud rate requirements, bus clock is generated by the master
- Original communication speed was defined with a maximum of 100 kbit per second (Standard mode)
- Later modes
  - Fastmode (400 kbit/s)
  - High speed mode (3.4 Mbit/s) with additional logic
  - Fast mode plus (up to 1 MHz max frequency, 1 Mbit/s) for intermediate transfer rates with no additional logic.
  - Ultra fast mode UFM (up to 5 Mbit/s) attained through major changes (single master, no acknowledges, no multi-master arbitration, push-pull connections)

I²C BUS

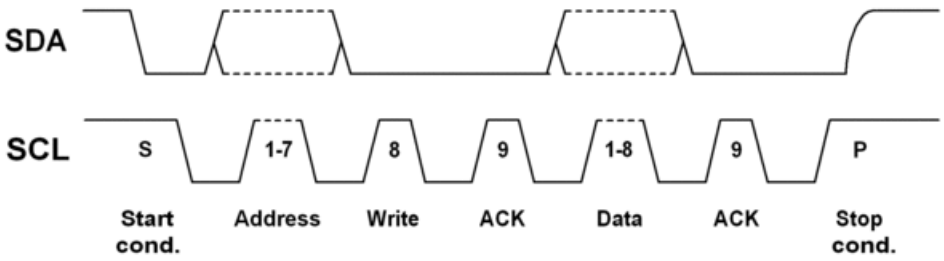
- Two wires carry data (SDA) and clock signals (SCL)
- Open drain connections allows for:
  - concurrent operation of more than one I2C master (if they are multi-master capable)
  - clock stretching (slaves can slow down communication by holding down SCL)



VCC	I2C supply voltage (1.2 V - 5.5 V)
GND	Common ground
SDA	Serial data (I2C data line)
SCL	Serial clock (I2C clock line)
Rp	Pull-up resistance (I2C termination)
Rs	Serial resistance
Cp	Wire capacitance
Cc	Cross channel capacitance

I²C BUS

- Addressing of multiple slaves on the same bus
- Open drain connections allows for:
  - concurrent operation of more than one I2C master (if they are multi-master capable)
  - clock stretching (slaves can slow down communication by holding down SCL)
- All, using a single, bidirectional, data line

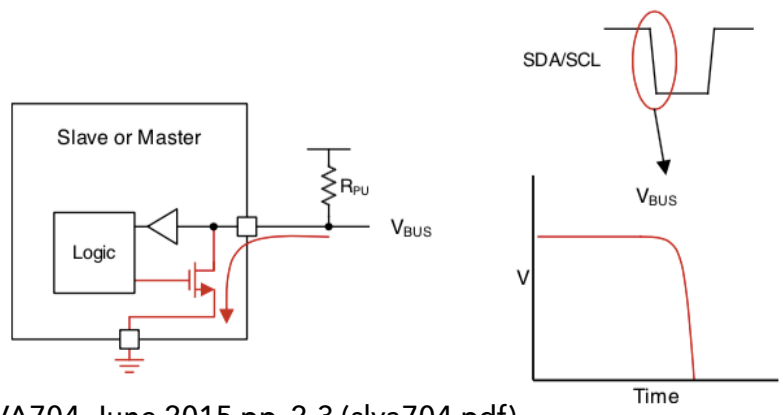


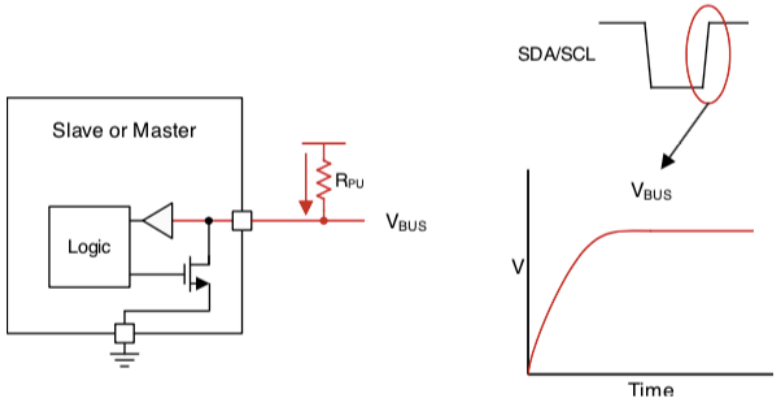
Address	Purpose
0000000 0	General Call
0000000 1	Start Byte
0000001 X	CBUS addresses
0000010 X	Reserved for Different Bus Formats
0000011 X	Reserved for future purposes
00001XX X	High-Speed Master Code
11110XX X	10-bit Slave Addressing
11111XX X	Reserved for future purposes

SERIAL BUSES

I²C bus

- Open-Drain for Bidirectional Communication
  - To transmit a **low** the logic **activates** the **pull-down FET**
    - the **line** is **shorted to ground** (pulled to low)
  - To transmit a **high** the logic may only release the bus by turning off the pull-down FET
    - The **line** is left **floating**, and the **pull-up resistor** pulls the voltage up to the voltage rail

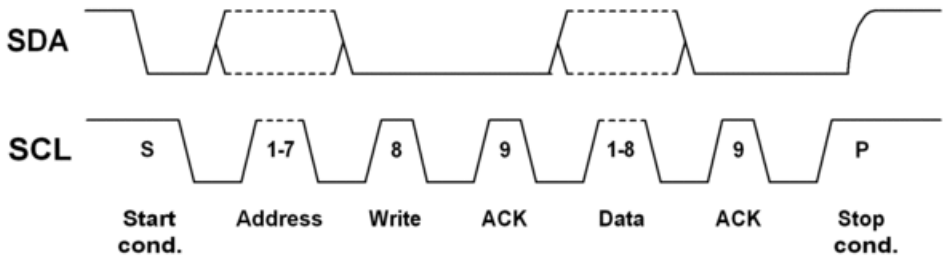




TI - SLVA704–June 2015 pp. 2-3 (slva704.pdf)

I²C BUS

- Standard bidirectional interface that uses a controller (master) to communicate with slave devices
- A slave may not transmit data unless it has been addressed by the master
- Each device on the I2C bus has a specific device address to differentiate between other devices that are on the same I2C bus
- Many slave devices will require configuration upon startup to set the behavior of the device
- This is typically done when the master accesses the slave's internal register maps, which have unique register addresses
- A device can have one or multiple registers where data is stored, written, or read

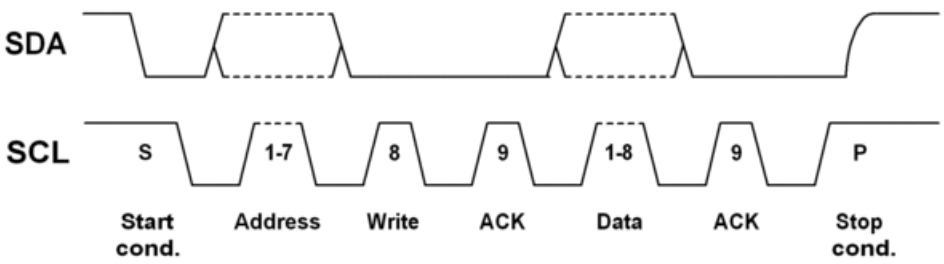


Address	Purpose
0000000 0	General Call
0000000 1	Start Byte
0000001 X	CBUS addresses
0000010 X	Reserved for Different Bus Formats
0000011 X	Reserved for future purposes
00001XX X	High-Speed Master Code
11110XX X	10-bit Slave Addressing
11111XX X	Reserved for future purposes

TI - SLVA704–June 2015 p. 3 (slva704.pdf)

I²C BUS

- General procedure for a **master** to **send** data to a **slave** device:
  - Master-transmitter sends a START condition and addresses the slave-receiver
  - Master-transmitter sends data to slave-receiver
  - Master-transmitter terminates the transfer with a STOP condition

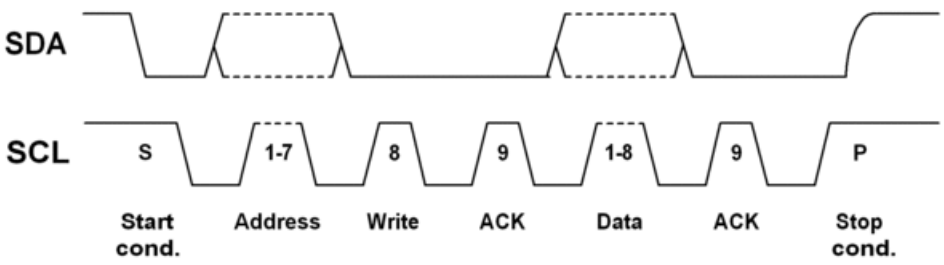


Address	Purpose
0000000 0	General Call
0000000 1	Start Byte
0000001 X	CBUS addresses
0000010 X	Reserved for Different Bus Formats
0000011 X	Reserved for future purposes
00001XX X	High-Speed Master Code
11110XX X	10-bit Slave Addressing
11111XX X	Reserved for future purposes

TI - SLVA704–June 2015 p. 3 (slva704.pdf)

I²C BUS

- General procedure for a **master** to **receive/read** data from a **slave**:
  - Master-receiver sends a START condition and addresses the slave-transmitter
  - Master-receiver sends the requested register to read to slave-transmitter
  - Master-receiver receives data from the slave-transmitter
  - Master-receiver terminates the transfer with a STOP condition

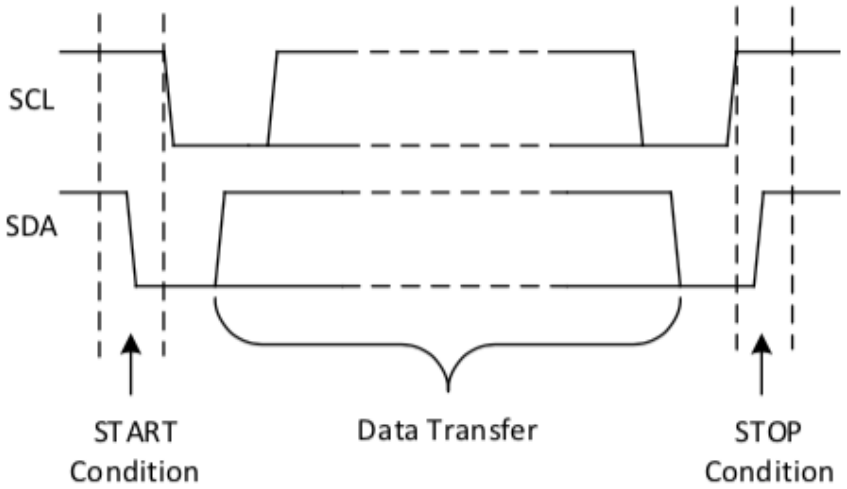


TI - SLVA704–June 2015 p. 3 (slva704.pdf)

Address	Purpose
0000000 0	General Call
0000000 1	Start Byte
0000001 X	CBUS addresses
0000010 X	Reserved for Different Bus Formats
0000011 X	Reserved for future purposes
00001XX X	High-Speed Master Code
11110XX X	10-bit Slave Addressing
11111XX X	Reserved for future purposes

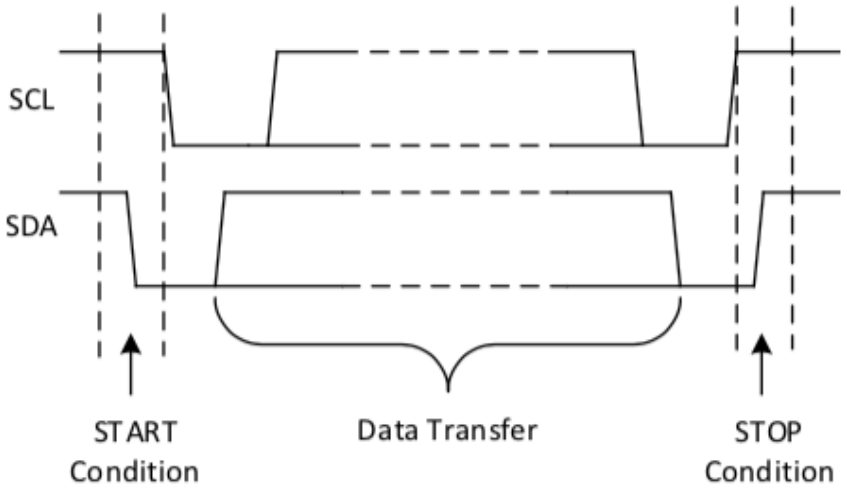
I²C BUS

- Communication is initiated by the master sending a **START** condition and terminated by the master sending a **STOP** condition.
- A **high-to-low** transition on the SDA line while the SCL is high defines a **START** condition
- A **low-to-high** transition on the SDA line while the SCL is high defines a **STOP** condition



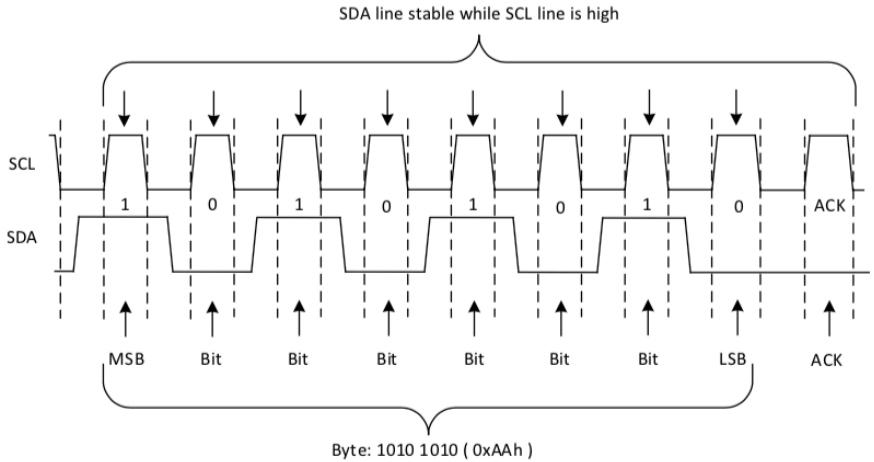
I²C BUS

- A **repeated START** condition is similar to a **START** condition and is used in place of a **back-to-back STOP** then **START** condition
  - It looks identical to a START condition, but differs from a START condition because it happens before a STOP condition (when the bus is not idle)
  - Useful for when the master wishes to start a new communication, but does not wish to let the bus go idle with the STOP condition, which has the chance of the master losing control of the bus to another master (in multi-master environments)



I²C BUS

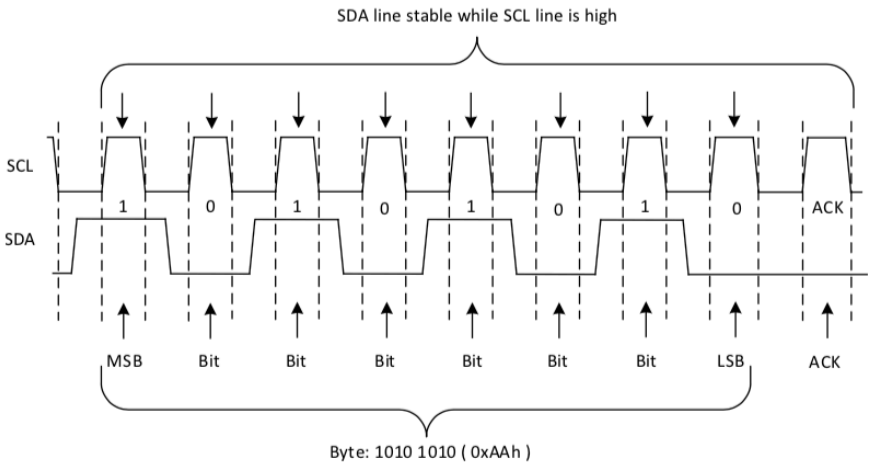
- **One data bit** is transferred during **each** clock **pulse** of the **SCL**
- One byte is comprised of **eight bits on** the **SDA** line
- A byte may either be a device address, register address, or data written to or read from a slave
- Data is transferred **Most Significant Bit** (MSB) **first**
- **Any number of data bytes can be transferred** from the master to slave **between** the **START** and **STOP** conditions
- Data on the **SDA line must remain stable** during the **high phase** of the **clock** period, as changes in the data line when the **SCL is high are** interpreted as **control commands** (START or STOP)



TI - SLVA704–June 2015 pp. 4-5 (slva704.pdf)

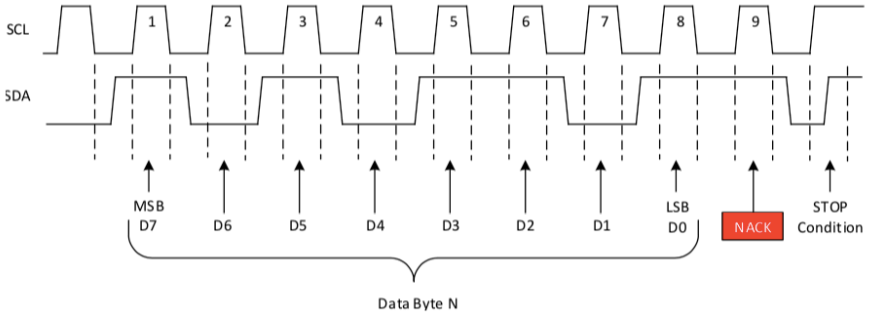
I²C BUS

- Each byte of data (including the address byte) is followed by one **ACK** (Acknowledge) bit from the receiver. The ACK signals that the byte was successfully received and another byte may be sent
- Before the **receiver** can send an ACK, the transmitter must release the SDA line
- To **send** an ACK bit, the **receiver** pulls down the SDA line during the low phase of period 9, so that the **SDA** line **is stable low** during the high phase of the ACK clock period



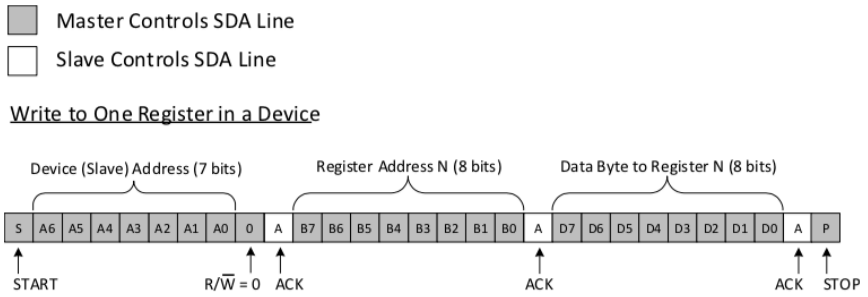
I²C BUS

- When the **SDA** line **remains high** during period 9, this is interpreted as a **Not Acknowledge** (**NACK)**
- Several conditions lead to the generation of a NACK
  - The receiver is unable to receive or transmit because it is performing some real-time function and is not ready to start communication with the master
  - During the transfer, the receiver gets data or commands that it does not understand
  - During the transfer, the receiver cannot receive any more data bytes
  - A master-receiver is done reading data and indicates this to the slave through a NACK



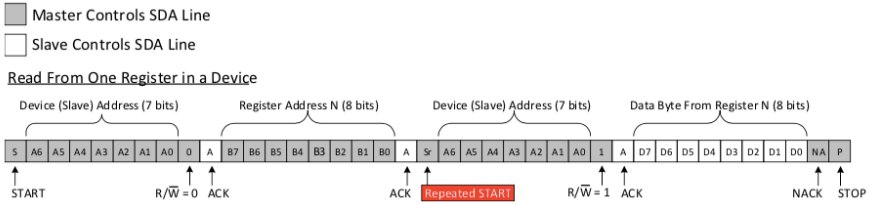
I²C BUS

- **To write** on the bus, the master sends a start condition with the slave's address followed by the R/W bit set to **0,** then:
- The slave sends the acknowledge bit
- The master sends the register address of the register it wishes to write to
- The slave acknowledges again, if it is ready
- The master starts sending the register data (one or more bytes) to the slave
- The master terminates the transmission with a STOP condition



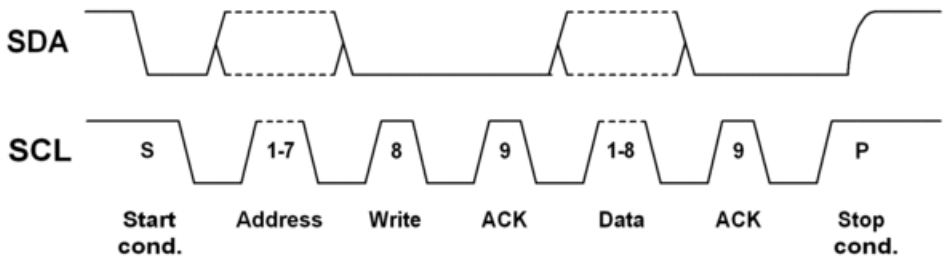
I²C BUS

- **To read** from a slave the master sends a **START condition** followed by the address of the slave with the R/W bit set to 0 and the register address it wishes to read from
- If the slave acknowledges the register address, the master sends a **START condition** again (**Repeated START**), followed by the slave address with the R/W bit set to 1
- If the slave acknowledges the read request, the master releases the SDA bus but continues supplying the clock to the slave. The master becomes the **master-receiver** and the slave the **slave-transmitter**
- The master releases the SDA line and continues sending out clock pulses, so that the slave can transmit data. At the end of every byte of data, the master sends an **ACK** to the slave, letting the slave know that it is ready for more data. Once the master receives the number of bytes expected, it sends a **NACK**, signaling to the slave to halt communications and release the bus. The master then sends a **STOP condition**



I²C BUS

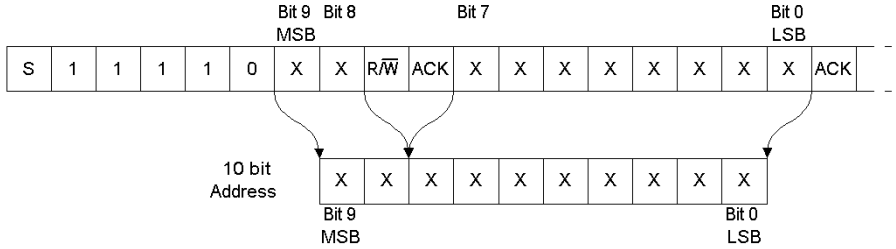
- Slow Start byte
  - Microcontrollers not provided with I2C controller circuitry have to observe the I2C lines permanently to detect an I2C transmission (polling). To reduce the waste of CPU power, an I2C transfer can be established with a slower arbitration method.
  - For this, the master transmits the START condition, followed by the start byte ('00000001'), a dummy acknowledge pulse and a repeated start condition. The observing microcontroller has to detect only one of the seven zeros on SDA to detect an I2C transmission. This can be done with a relatively slow polling rate. As soon as the controller detects that SDA is low, it can switch to a higher polling rate in order to await the repeated start condition and the following I2C transfer.
  - After the transfer has ended, it can switch back to the CPU-power saving slow polling rate in order to detect the next transmission.



Address	Purpose
0000000 0	General Call
0000000 1	Start Byte
0000001 X	CBUS addresses
0000010 X	Reserved for Different Bus Formats
0000011 X	Reserved for future purposes
00001XX X	High-Speed Master Code
11110XX X	10-bit Slave Addressing
11111XX X	Reserved for future purposes

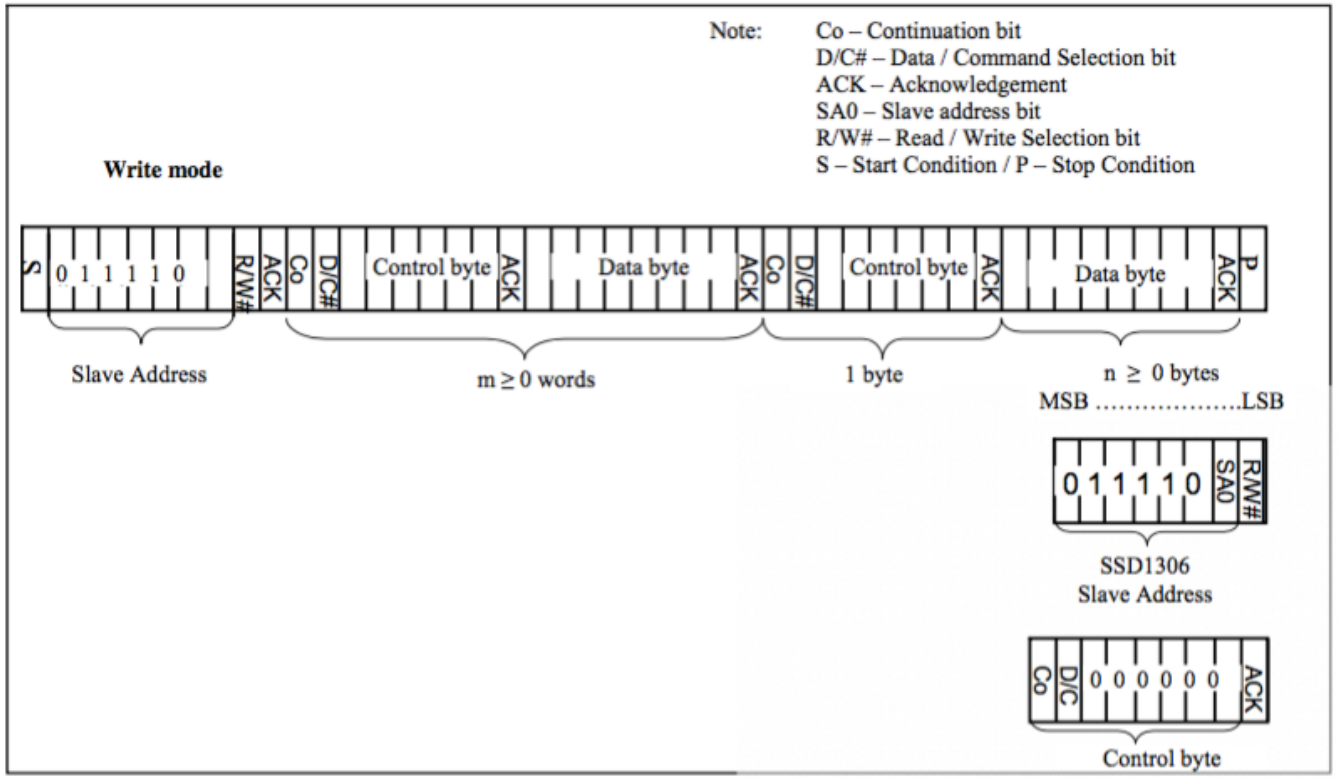
I²C BUS

- 10-bit Addressing
  - In order to prevent address clashes, due to the limited range of the 7 bit addresses, a new 10 bit address scheme has been introduced. This enhancement can be mixed with 7 bit addressing and increases the available address range about ten times. After the start condition, a leading ‘11110’ introduces the 10 bit addressing scheme. The last two address bits of the first byte concatenated with the eight bits of the second byte of the whole 10 bit address. Devices which only use 7 bit addressing simply ignore messages with the leading ‘11110’.



I²C BUS

- Devices implement vendor-specific protocols for data transfer and command
  - The SSD1306 OLED Display is provided with several interfaces (I2C, SPI, 6800 and 8080 buses)



SSD1306 OLED Display Datasheet - (SSD1306 Datasheet for 096 OLED.pdf) page 20