- **Introduction the Serial Communications**
- Huang Sections 9.2, 10.2, 11.2
- SCI Block User Guide
- SPI Block User Guide
- IIC Block User Guide
  - Parallel vs Serial Communication
  - Synchronous and Asynchronous Serial Communication
  - Description of the SPI (Serial Peripheral Interface) protocol
  - Description of the IIC (Inter-Integrated Circuit) protocol

Parallel Data Transfer

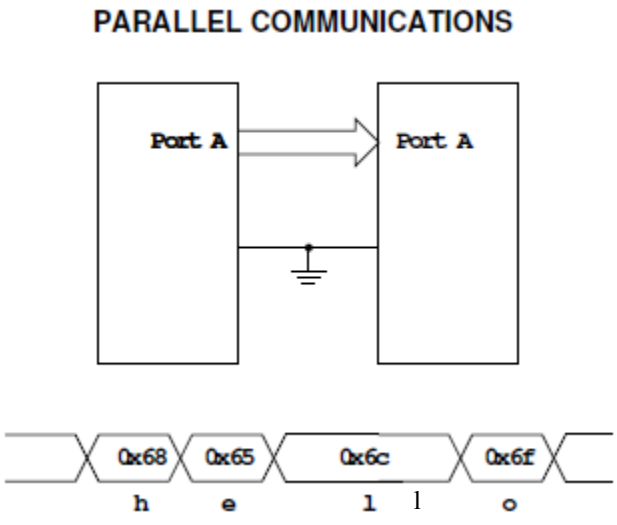
• Suppose you need to transfer data from one HCS12 to another. How can you do this?

• You could connect PORTA of the sending computer (set up as an output port) to PORTA of the receiving computer (set up as an input port).

• The sending computer puts the data on its PORTA, one byte at a time.

• The receiving computer reads the data on its PORTA.

• For example, want to sent the five bytes corresponding to the five characters ”hello”:



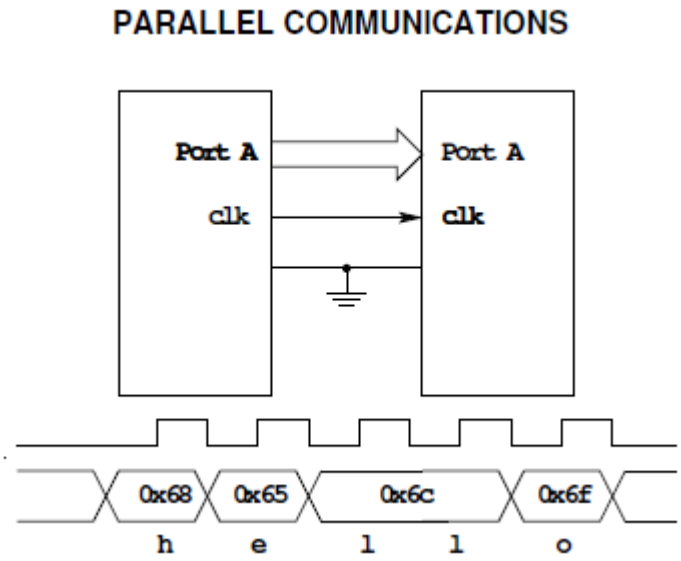
Need 9 wires to transmit 8 bits of data
How can receiver tell when it should read the data?

Parallel Data Transfer

• The sending computer needs to tell the receiving computer when to read the data.

• It can do this with another line used as a clock line.

• On the rising edge of the clock line, the receiving computer should read the data:



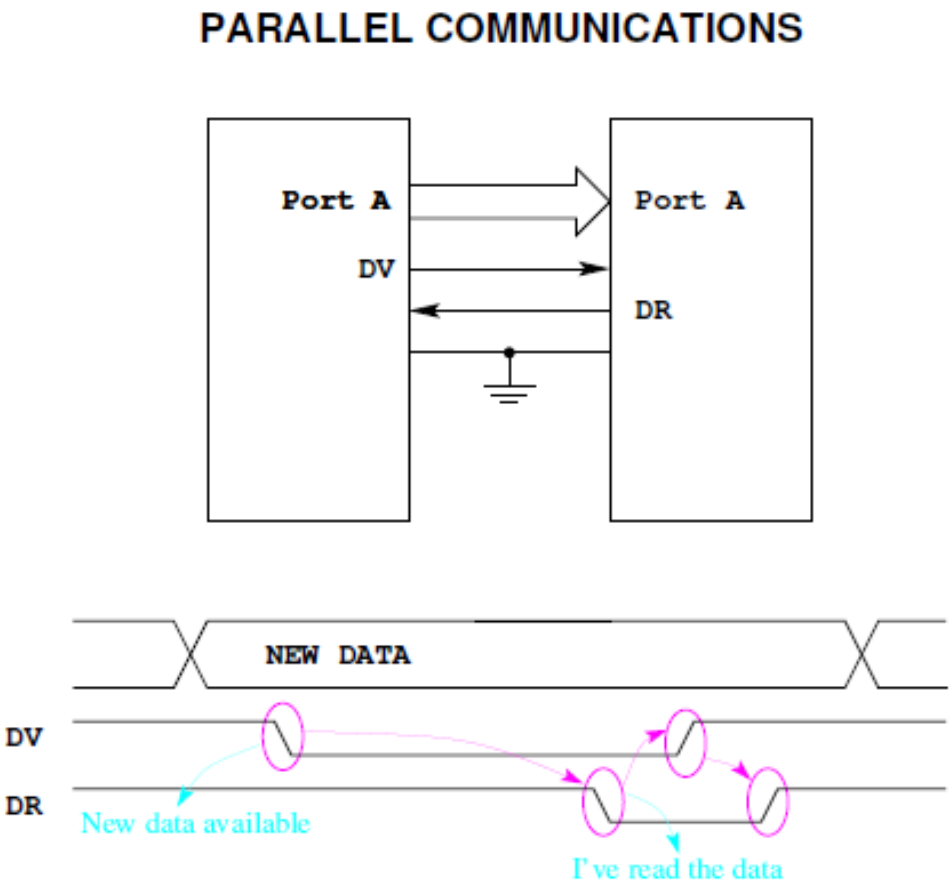
Need 10 wires to transmit 8 bits of data

Parallel Data Transfer

– How can the sending computer know that the receiving computer has received the data?

– Can use a method called **handshaking**.

* The sending computer uses a Data Valid line to tell the receiving computer that the data on the data lines is valid.
* The receiving computer uses a Data Received line to tell the sending computer that it has read the current data byte.



Use two lines –– Handshake –– sender knows when receiver is ready for new data.

Need 11 wires to transmit 8 bits of data

– In the above figure, the sending computer puts the data on the data lines and brings DV low to indicate new data is available.

– When the receiving computer sees the new data is available it reads the data on the data lines, then brings DR low to say that it has read the data.

– When the sending computer sees DR go low, it brings DV high.

– When the receiving computer sees DV go high, it brings DR high.

– Both computers are now ready for the next data transfer.

Serial Data Transfer

• Using parallel data transfer you can use 10 wires to transfer one byte at a time from one computer to another.

• Using 18 wires, you can transfer two bytes (16 bits) at a time.

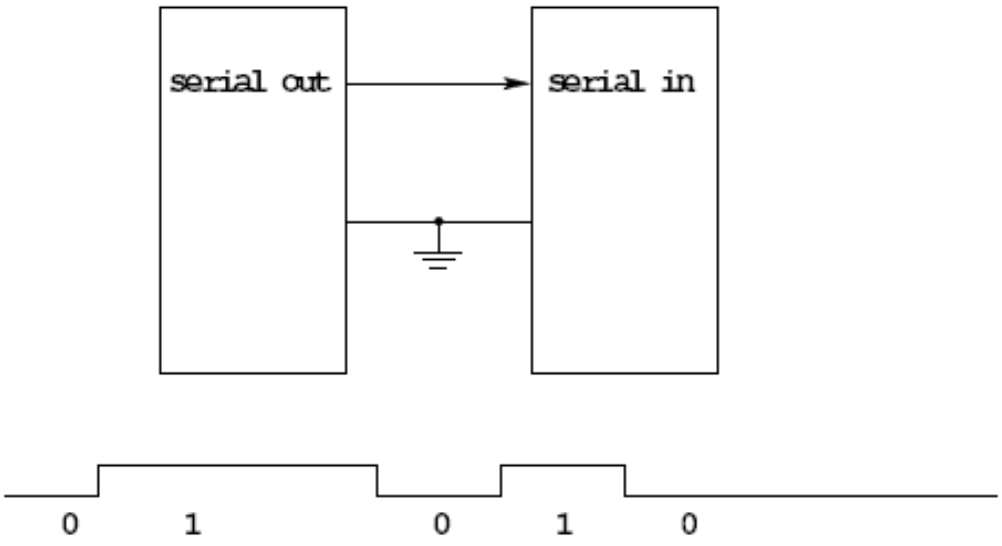
• Parallel data transfer is a very fast way to transfer data between two computers.

• There are two problems with parallel data transfer:
  – It takes a lot of wires between the computers.
  – It uses lots of I/O pins on the computers.

• Serial data transfer is a slower transfer mechanism, but it uses fewer wires and fewer I/O pins.

• Serial data transfer sends one bit at a time between two computers:

SERIAL COMMUNICATIONS



'h' = 0x68 = %01101000

Can't tell how many ones or zeros there are

Types of Serial Data Transfer

• Asynchronous serial (no clock between devices; uses internal clocks to synchronize data transfer)

– RS-232 (typically ± 12V), RS-485 (± 6V)

• Synchronous serial (clock line between devices to synchronize data transfer)

– SPI (Synchronous Peripheral Interface)
– IIC or I^2C (Inter-Integrated Circuit)

• Differential Serial (Typically ±3 V, clock and signal intermixed on same wires)

– CAN (Controller Area Network)
– Ethernet
– USB (Universal Serial Bus)

• LVDS (Low Voltage Differential Signaling) (±350 mV) LVDS is much faster than other methods, and uses much less power, because voltage swing is much smaller.

– Firewire
– Serial ATA
– PCI Express
– Many others – wave of the future

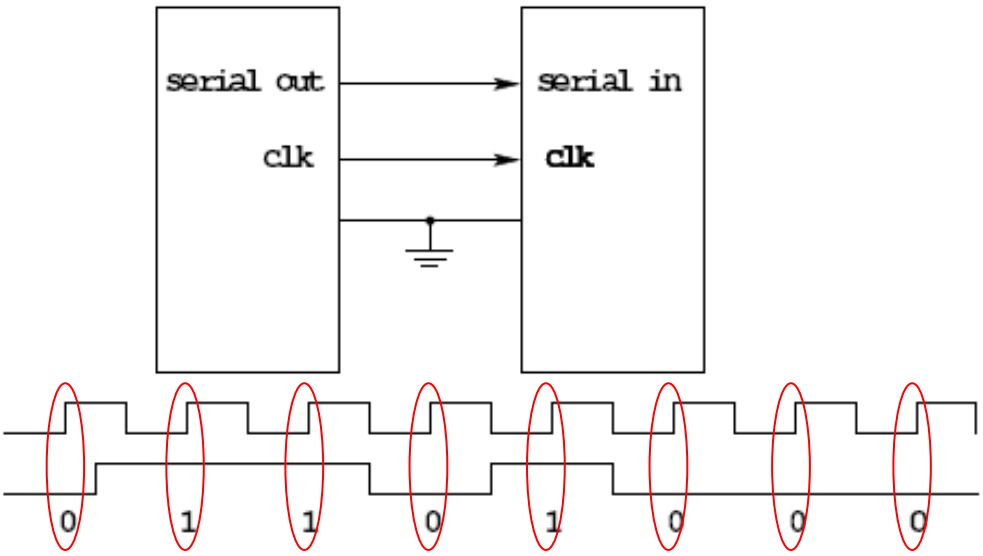
Synchronous Serial Data Transfer

• To use serial data transfer, you need to have a way for the receiving computer to know when the data bit is valid.

• There are two ways to do this:
– Synchronous Serial Data Transfers (Serial Peripheral Interface (SPI) on the HCS12)
– Asynchronous Serial Data Transfers (Serial Communication Interface (SCI) on the HCS12)

• Synchronous Serial Data Transfer uses a clock line between the two computers for the sending computer to tell the receiving computer when each data bit is valid:

SYNCHRONOUS SERIAL COMMUNICATIONS



'h' = 0x68 = B"01101000"

Need 3 wires to transmit 1 bit at a time

Synchronous Serial Data Transfer

• In synchronous serial data transfer, the sending computer puts the data byte it wants to send into an internal shift register.

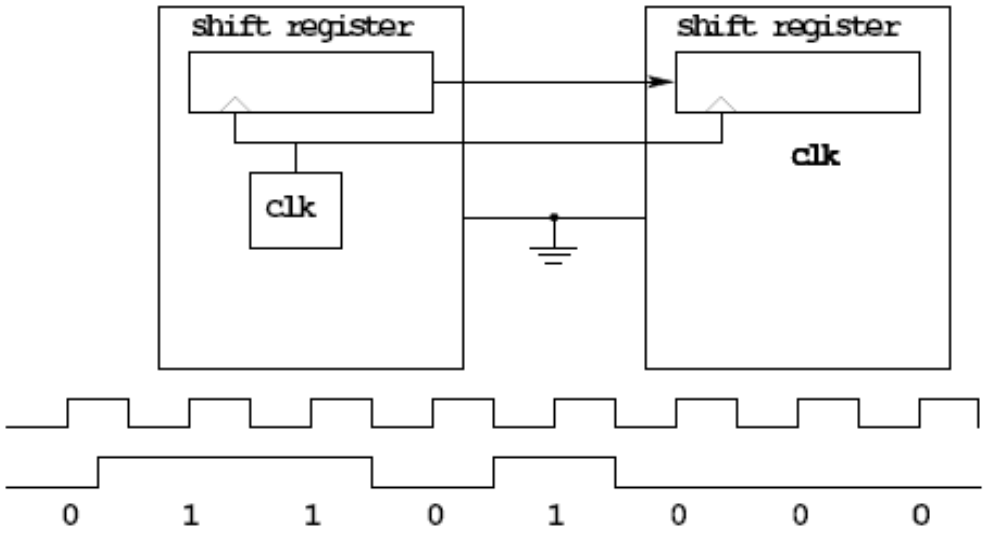
• The sending computer uses a clock to shift the 8 data bits out of the shift register onto an external data pin.

• The receiving computer puts the data from the sending computer on the input of an internal shift register.

• The receiving computer uses the clock from the sending computer to shift the data into its shift register.

• After 8 clock ticks, the data has been transferred from the sending computer to the receiving computer.

SYNCHRONOUS SERIAL COMMUNICATIONS



'h' = 0x68 = B"01101000"

Need 3 wires to transmit 1 bit at a time

The HCS12 Serial Peripheral Interface (SPI)

• The HCS12 has a Synchronous Serial Interface

• On the HCS12 it is called the Serial Peripheral Interface (SPI)

• If an HCS12 generates the clock used for the synchronous data transfer it is operating in Master Mode.

• If an HCS12 uses and external clock used for the synchronous data transfer it is operating in Slave Mode.

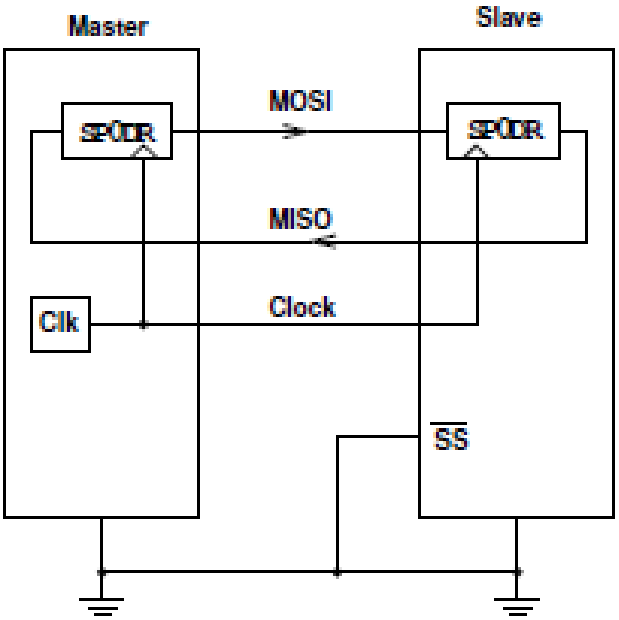
• If two HCS12's talk to each other using their SPI's one must be set up as the Master and the other as the Slave.

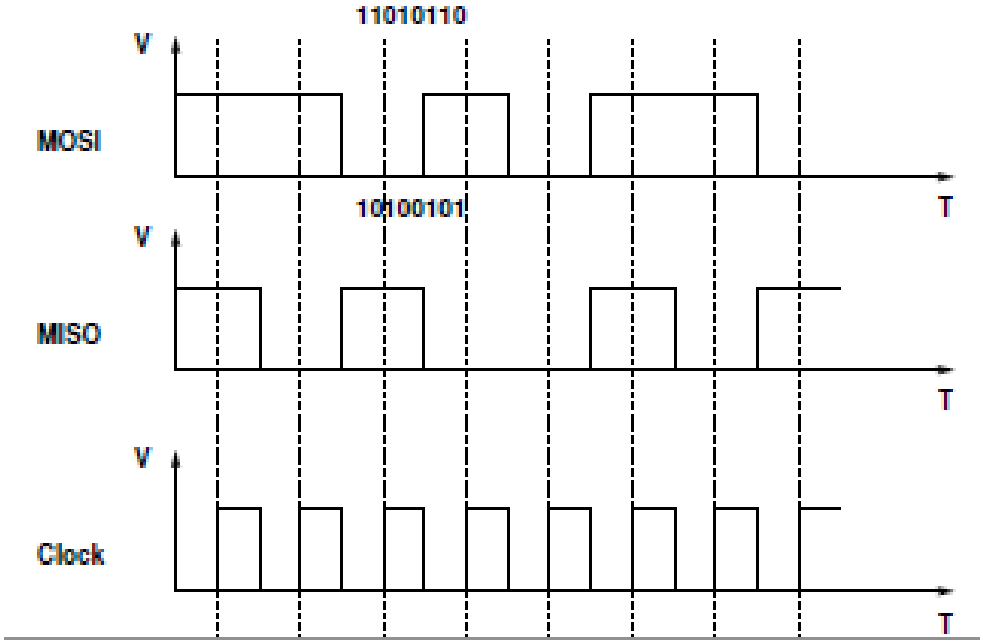
• The output of the Master SPI shift register is connected to the input of the Slave SPI shift register over the Master Out Slave In (MOSI) line.

• The input of the Master SPI shift register is connected to the output of the Slave SPI shift register over the Master In Slave Out (MISO) line.

• After 8 clock ticks, the data originally in the Master shift register has been transferred to the slave, and the data in the Slave shift register has been transferred to the Master.

Synchronous Serial Communications





Use of Slave Select with the HCS12 SPI

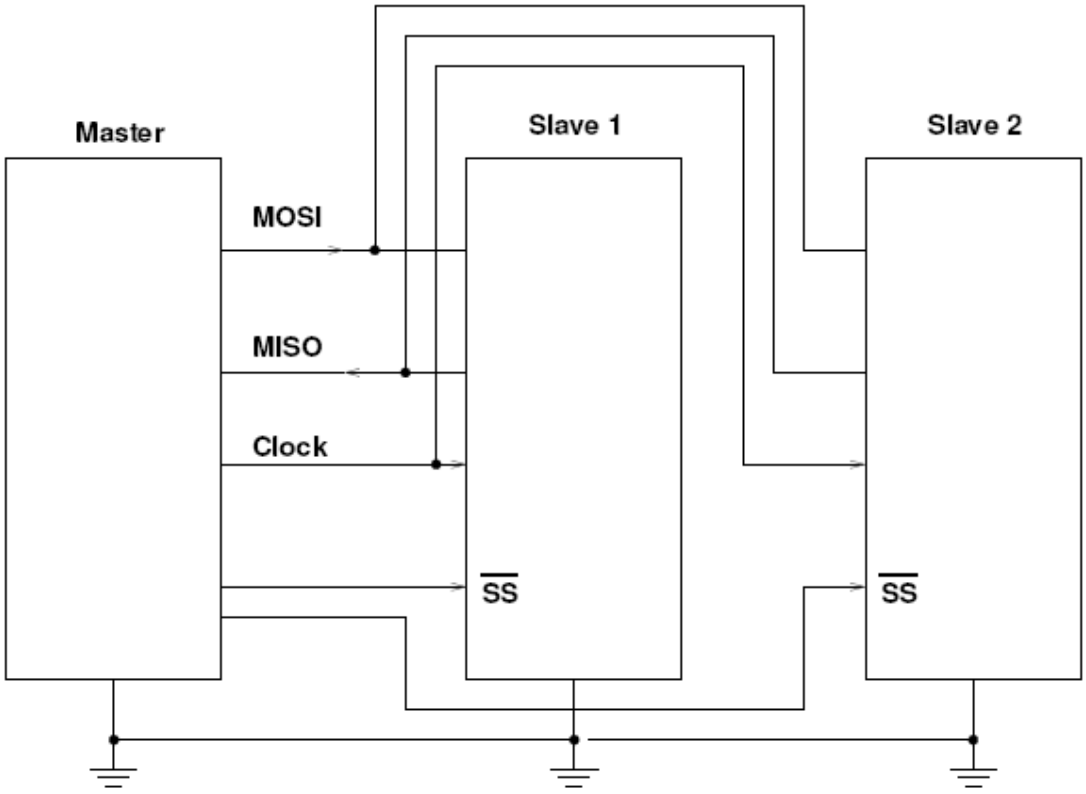
• A master HCS12 can talk with more than one slave HCS12's.

• A slave HCS12 uses its Slave Select (SS) line to determine if it is the one the master is talking with.

• There can only be one master HCS12, because the master HCS12 is the device which generates the serial clock signal.

• Need to have: Slave select (SS), Serial clock (SCK), Master out/slave in (MOSI), Master in/slave out (MISO).

SYNCHRONOUS SERIAL COMMUNICATIONS



With select lines, one master can communicate with more than one slave

Using the HCS12 SPI with other devices

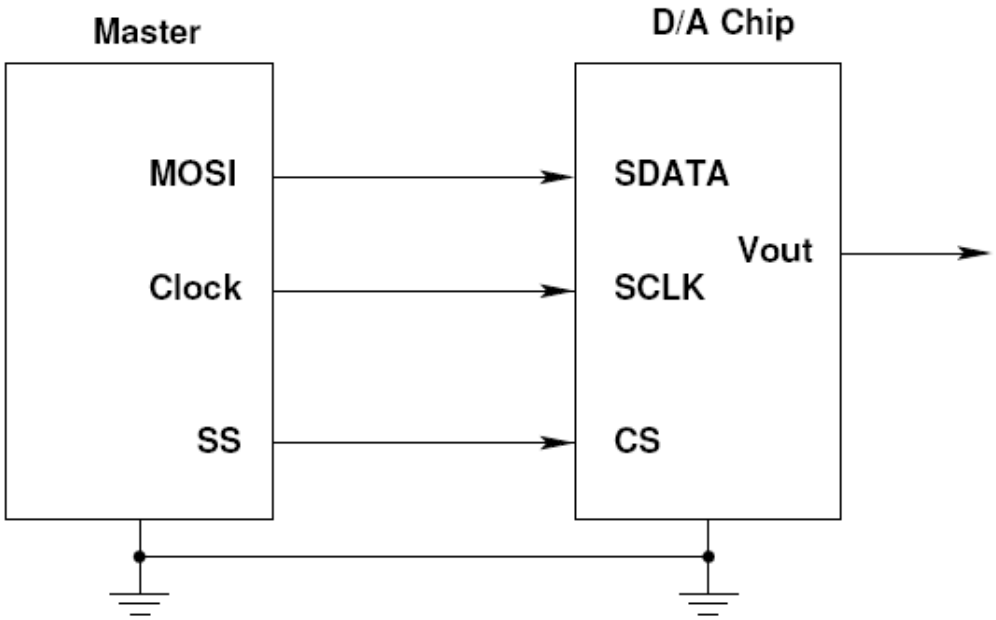
• The HCS12 can communicate with many types of devices using its SPI

• For example, consider a D/A (Digital-to-Analog) Converter

• The D/A converter has three digital lines connected to the HCS12:
  – Serial Data
  – Serial Clock
  – Chip Select

• The HCS12 can send a digital number to the D/A converter. The D/A converter will convert this digital number to a voltage.

SPI COMMUNICATION WITH A/D CONVERTER



The HCS12 IIC Interface

• The SPI requires three lines (Clock, MISO, MOSI), plus another line to select each device to talk to.

1. An SPI bus which controls four slaves will require seven lines.

• Another popular synchronous serial interface is the Inter-Integrated Circuit (IIC or I2C) bus

– The IIC bus can control multiple devices using only two wires

– The two wires are Clock and Data

*The devices connect to the wires using a *wired and* method

* The lines are normally high. Any device on the bus can bring them low.

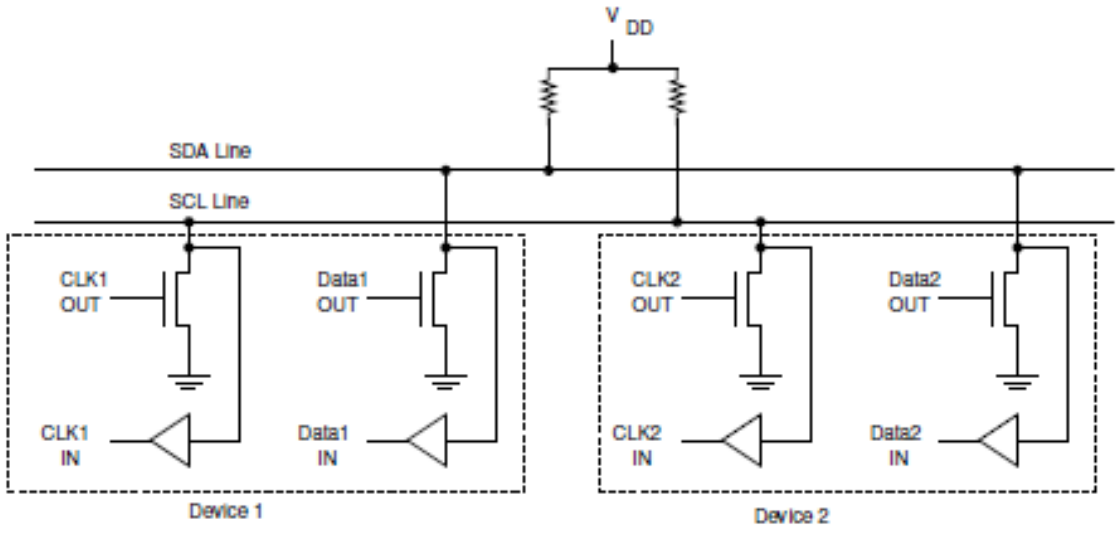
– Each device on the bus has a unique address.

– An IIC master starts the process by sending out a serial stream with the seven-bit address of the slave it wants to talk to, and an eight bit indicating if it wants to write to the slave or read from the slave.

– If it writes to the slave, it will continue to send data on the serial data line (toggling the clock on each data bit) until all the data is sent.

– If it reads from the slave, it will release the data line, but retain control of the clock line. The slave takes over the data line, and sends out its data in response to the clock provided by the master.

– After all the data is transferred, the master releases both the clock and the data lines



The IIC Interface

• Normally, both SDA (Serial Data Line) and SCL (Serial Clock) are high.

• When the master wants to talk to a slave, it brings SDA low (the Start condition). This gives the master control of the bus

• The master sends eight bits of data on the SDA bus, and toggles the SCL bus for each bit. The eight bits are the seven-bit address of the slave, and one bit for read-write

– a low indicates a write, and a high indicates a read. After the eighth data bit, the master releases the SDA line.

• The master sends a ninth clock pulse. The addressed slave responds with an acknowledge by bringing SDA low.

• If the master is writing data to the slave, it continues controlling both SDA and SCL. It sends eight bits of data, releases SDA, and waits for an acknowledge from the slave

• It the master is reading data from the slave, it controls the SCL line and the slave controls the SDA line. The master sends eight clock pulses on the SCL line, and the slave transfers one bit of data on each clock pulse. At the end of the eight bits, the master takes over the SDA bus and sends an acknowledge on all but the last byte it wants. After the last byte, the master sends an NACK (leaves SDA high).

• After the transfer is complete, the master sends a Stop condition to indicate that it is releasing the bus. The Stop condition is indicated by bringing SDA high which SCL is high.