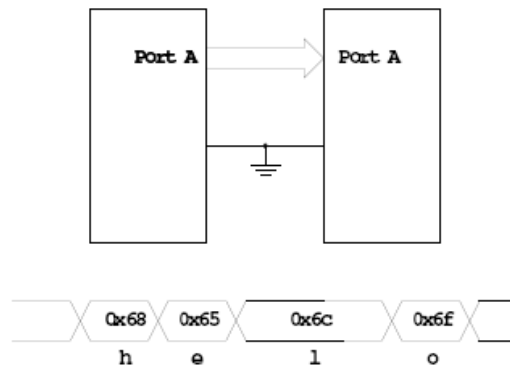


- **Introduction the Serial Communications**
- Huang Sections 9.2, 10.2
- SCI Block User Guide
- SPI Block User Guide

### 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":

### PARALLEL COMMUNICATIONS

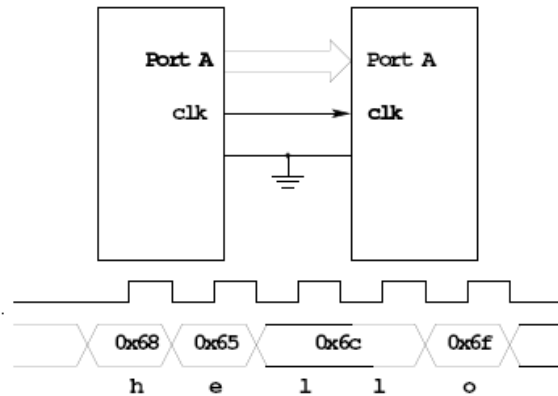


**Need 9 wires to transmit 8 bits of data**

**How can receiver tell when it should read the data?**

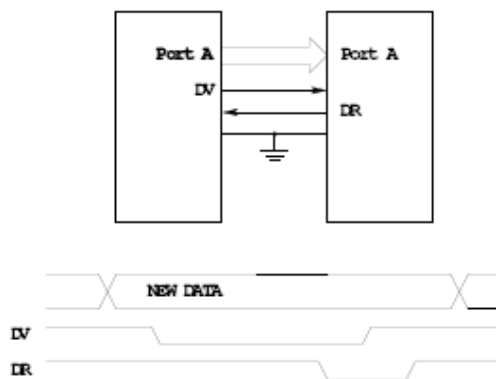
### Parallel Data Transfer

- 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**

- 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**

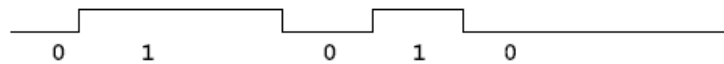
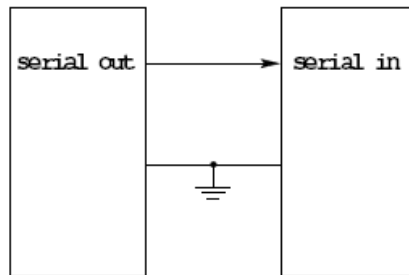
- In the above figure, the sending computer puts the data on the data lines and brings DV low to indicate new data is available.

- 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



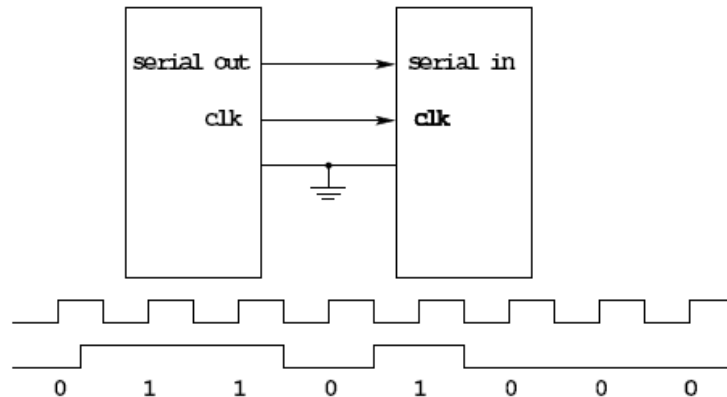
'h' = 0x68 = %01101000

Can't tell how many ones or zeros there are

## 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"

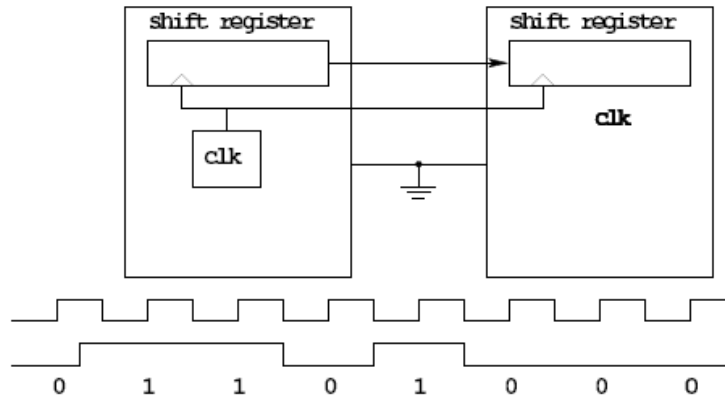
Need 3 wires to transmit 1 bit at a time

## 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



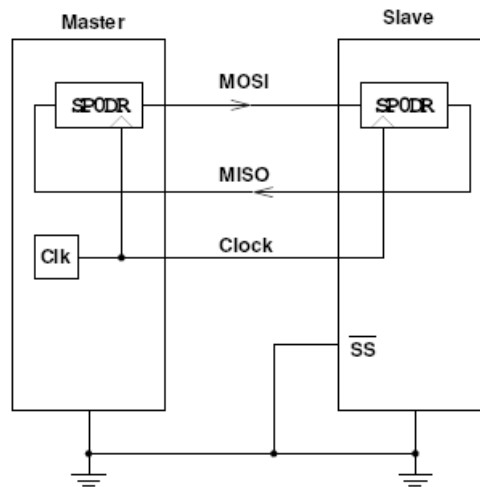
'h' = 0x68 = B"01101000"

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 an 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.

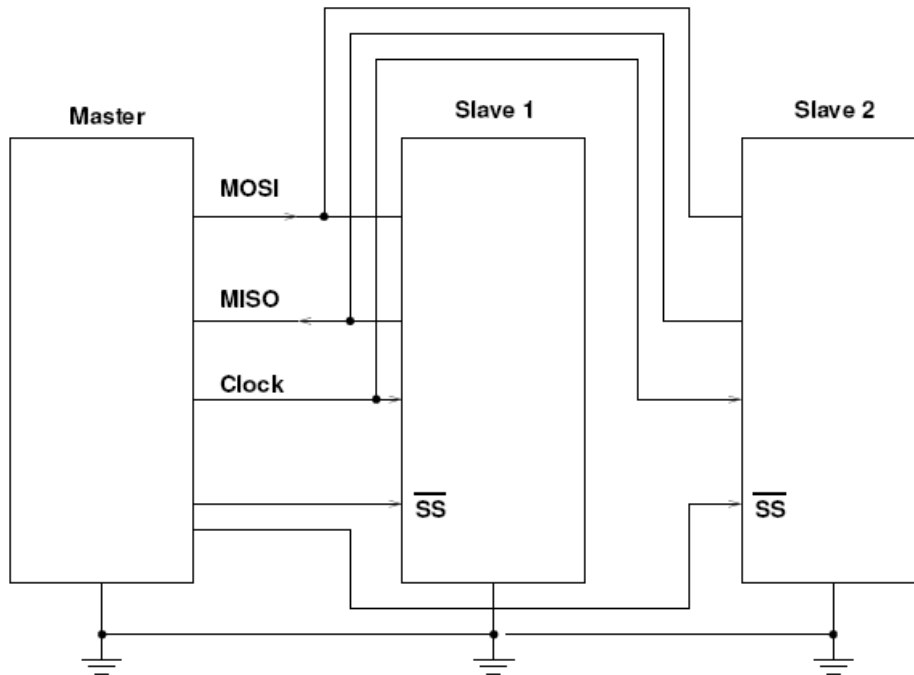
- After 8 clock ticks, the data originally in the Master shift register has been transferred to the slave, and the data in the Slave shift register has been transferred to the Master.



### 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

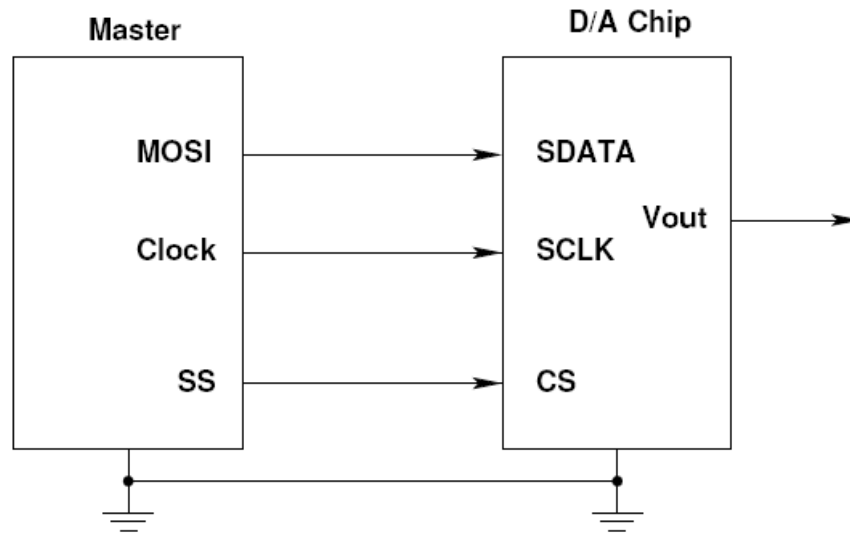


With select lines, one master can communicate with more than one slave

### 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

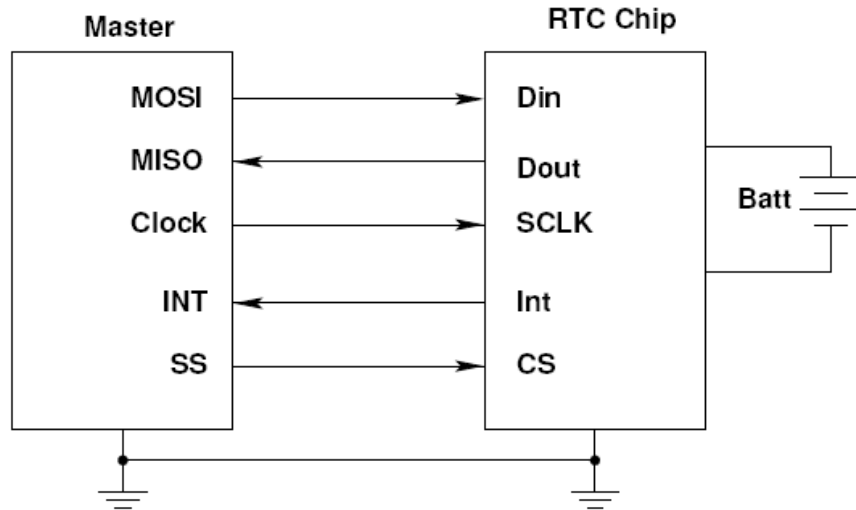


### Using the HCS12 SPI with other devices

- Another type of device the HCS12 can talk to is a Real Time Clock (RTC).
- An RTC keeps track of the time (year, month, day, hour, minute, second).
- An RTC can be programmed to generate an alarm (interrupt) at a particular time (i.e. 07:00), or can generate a periodic interrupt at a regular interval (once a second, once an hour, etc.).
- The HCS12 initially tells the RTC what the correct time is.
- The RTC keeps track of time from then on.



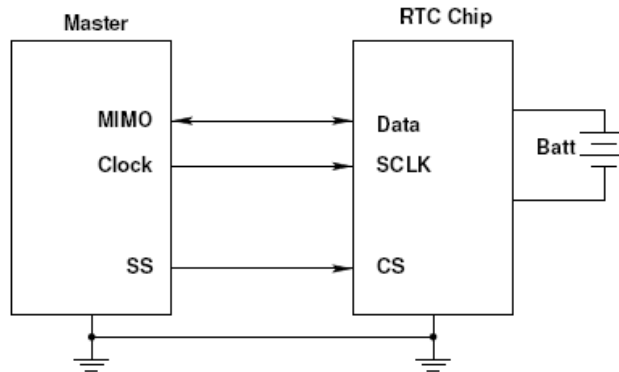
## SPI COMMUNICATION WITH A REAL TIME CLOCK



### Using the HCS12 SPI with other devices

- An interface with even fewer wires can be implemented by using one data line in bidirectional mode.
- In bidirectional mode, a single data line functions both as serial data in and serial data out.
- In lab, we will connect our 9S12 to a Dallas Semiconductor DS1302 Real Time Clock, which uses a three-wire serial interface with a bidirectional data line.
- The MOSI line on the 9S12 becomes a MOMI (Master Out Master In) line.
  - When the 9S12 wants to write data to the DS1302, it makes MIMO an output.
  - When the 9S12 wants to read data from the DS1302, it makes MIMO an input.

## BIDIRECTIONAL (3-WIRE) SPI COMMUNICATION SPI WITH A REAL TIME CLOCK



When used as a master in bidirectional mode, the Master Out Slave In pin becomes the Master In Master Out Pin.

In a system, an HCS12 can communicate with many different devices over its SPI interface.

- It uses the same data and clock lines, and selects different devices by using GPIO lines as slave selects.

