SPI Bus interface

Introduction:

Serial to Peripheral Interface (SPI) is a hardware/firmware communications protocol developed by Motorola and later adopted by others in the industry. Microwire of National Semiconductor is same as SPI. Sometimes SPI is also called a "four wire" serial bus.

The Serial Peripheral Interface or SPI-bus is a simple 4-wire serial communications interface used by many microprocessor/microcontroller peripheral chips that enables the controllers and peripheral devices to communicate each other. Even though it is developed primarily for the communication between host processor and peripherals, a connection of two processors via SPI is just as well possible.

The SPI bus, which operates at full duplex (means, signals carrying data can go in both directions simultaneously), is a synchronous type data link setup with a Master / Slave interface and can support up to 1 megabaud or 10Mbps of speed. Both single-master and multi-master protocols are possible in SPI. But the multi-master bus is rarely used and look awkward, and are usually limited to a single slave.

The SPI Bus is usually used only on the PCB. There are many facts, which prevent us from using it outside the PCB area. The SPI Bus was designed to transfer data between various IC chips, at very high speeds. Due to this high-speed aspect, the bus lines cannot be too long, because their reactance increases too much, and the Bus becomes unusable. However, its possible to use the SPI Bus outside the PCB at low speeds, but this is not quite practical.

The peripherals can be a Real Time Clocks, converters like ADC and DAC, memory modules like EEPROM and FLASH, sensors like temperature sensors and pressure sensors, or some other devices like signal-mixer, potentiometer, LCD controller, UART, CAN controller, USB controller and amplifier.

Data and control lines of the SPI and the basic connection:

An SPI protocol specifies 4 signal wires.

1. Master Out Slave In (MOSI) - MOSI signal is generated by Master, recipient is the Slave.
2. Master In Slave Out (MISO) - Slaves generate MISO signals and recipient is the Master.
3. Serial Clock (SCLK or SCK) - SCLK signal is generated by the Master to synchronize data transfers between the master and the slave.
4. Slave Select (SS) from master to Chip Select (CS) of slave - SS signal is generated by Master to select individual slave/peripheral devices. The SS/CS is an active low signal.

There may be other naming conventions such as Serial Data In [SDI] in place of MOSI and Serial Data Out [SDO] for MISO.

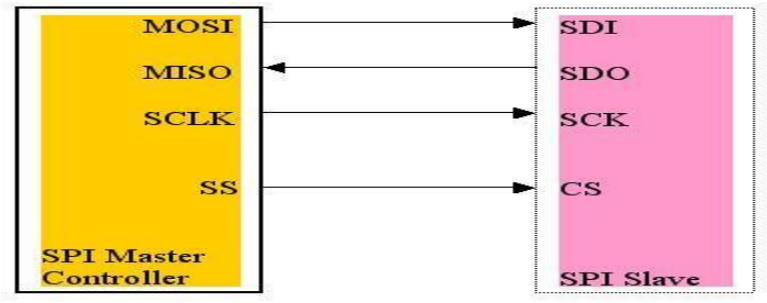

Fig-1 (Single master, single slave SPI implementation)

Among these four logic signals, two of them MOSI & MISO can be grouped as data lines and other two SS & SCLK as control lines.

As we already know, in SPI-bus communication there can be one master with multiple slaves. In single-master protocol, usually one SPI device acts as the SPI Master and controls the data flow by generating the clock signal (SCLK) and activating the slave it wants to communicate with slave-select signal (SS), then receives and or transmits data via the two data lines. A master, usually the host micro controller, always provides clock signal to all devices on a bus whether it is selected or not.

The usage of these each four pins may depend on the devices. For example, SDI pin may not be present if a device does not require an input (ADC for example), or SDO pin may not be present if a device does not require an output (LCD controllers for example). If a microcontroller only needs to talk to 1 SPI Peripheral or one slave, then the CS pin on that slave may be grounded. With multiple slave devices, an independent SS signal is needed from the master for each slave device.

How do they communicate?

The communication is initiated by the master all the time. The master first configures the clock, using a frequency, which is less than or equal to the maximum frequency that the slave device supports. The master then select the desired slave for communication by pulling the chip select (SS) line of that particular slave-peripheral to "low" state. If a waiting period is required (such as for analog-to-digital conversion) then the master must wait for at least that period of time before starting to issue clock cycles.

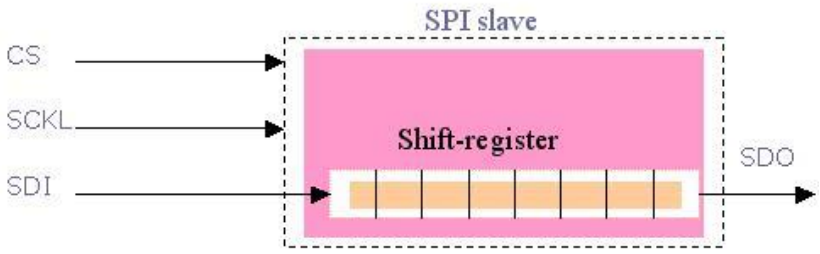

Fig-2 (The shift register used)

The slaves on the bus that has not been activated by the master using its slave select signal will disregard the input clock and MOSI signals from the master, and must not drive MISO. That means the master selects only one slave at a time.

Most devices/peripherals have tri-state outputs, which goes to high impedance state (disconnected) when the device is not selected. Devices without this tri-state outputs cannot share SPI bus with other devices, because such slave's chip-select may not get activated.

A full duplex data transmission can occur during each clock cycle. That means the master sends a bit on the MOSI line; the slave reads it from that same line and the slave sends a bit on the MISO line; the master reads it from that same line.

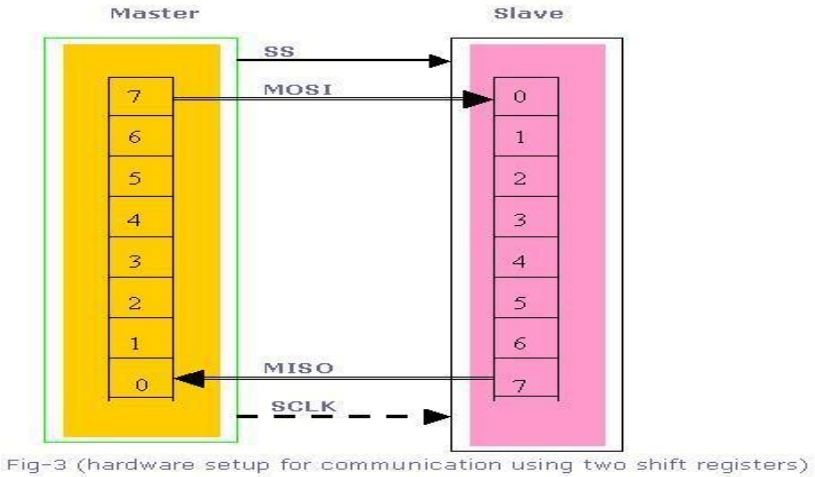

Fig-3 (hardware setup for communication using two shift registers)

Data transfer is organized by using Shift register with some given word size such as 8- bits (remember, its not limited to 8-bits), in both master and slave. They are connected in a ring. While master shifts register value out through MOSI line, the slave shifts data in to its shift register.

Data are usually shifted out with the MSB first, while shifting a new LSB into the same register. After that register has been shifted out, the master and slave have exchanged their register values. Then each device takes that value and does the necessary operation with it (for example, writing it to memory). If there are more data to be exchanged, the shift registers are loaded with new data and the process is repeated. When there are no more data to be transmitted, the master stops its clock. Normally, it then rejects the slave.

There is a "multiple byte stream mode" available with SPI bus interface. In this mode the master can shift bytes continuously. In this case, the slave select (SS) is kept low until all stream process gets finished.

SPI devices sometimes use another signal line to send an interrupt signal to a host CPU. Some of the examples for these type of signals are pen-down interrupts from touch-screen sensors, thermal limit alerts from temperature sensors, alarms issued by real time clock chips, and headset jack insertions from the sound codec in a cell phone.

Significance of the clock polarity and phase:

Another pair of parameters called clock polarity (CPOL) and clock phase (CPHA) determine the edges of the clock signal on which the data are driven and sampled.
That means, in addition to setting the clock frequency, the master must also configure the clock polarity (CPOL) and phase (CPHA) with respect to the data. Since the clock serves as synchronization of the data communication, there are four possible modes that can be used in an SPI protocol, based on this CPOL and CPHA.

SPI Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

If the phase of the clock is zero (i.e. CPHA = 0) data is latched at the rising edge of the clock with CPOL = 0, and at the falling edge of the clock with CPOL = 1.
If CPHA = 1, the polarities are reversed. Data is latched at the falling edge of the clock with CPOL = 0, and at the rising edge with CPOL = 1.

The micro-controllers allow the polarity and the phase of the clock to be adjusted. A positive polarity results in latching data at the rising edge of the clock. However data is put on the data line already at the falling edge in order to stabilize. Most peripherals, which can only be slaves, work with this configuration. If it should become necessary to use the other polarity, transitions are reversed.

Different types of configurations:

Suppose a master-microcontroller needs to talk to multiple SPI Peripherals. There are 2 ways to set things up:

1. Cascaded slaves or daisy-chained slaves
2. Independent slaves or parallel configuration

Daisy-chained slave configuration:

In cascaded slave configuration, all the clock lines (SCLK) are connected together. And also all the chip select (CS) pins are connected together. The data flows out the microcontroller, through each peripheral in turn, and back to the microcontroller. The data output of the preceding slave-device is tied to the data input of the next, thus forming a wider shift register. So the cascaded slave-devices are evidently looked at as one larger device and receive therefore the same chip select signal. This means, only a single SS line is required from the master, rather than a separate SS line for each slave.

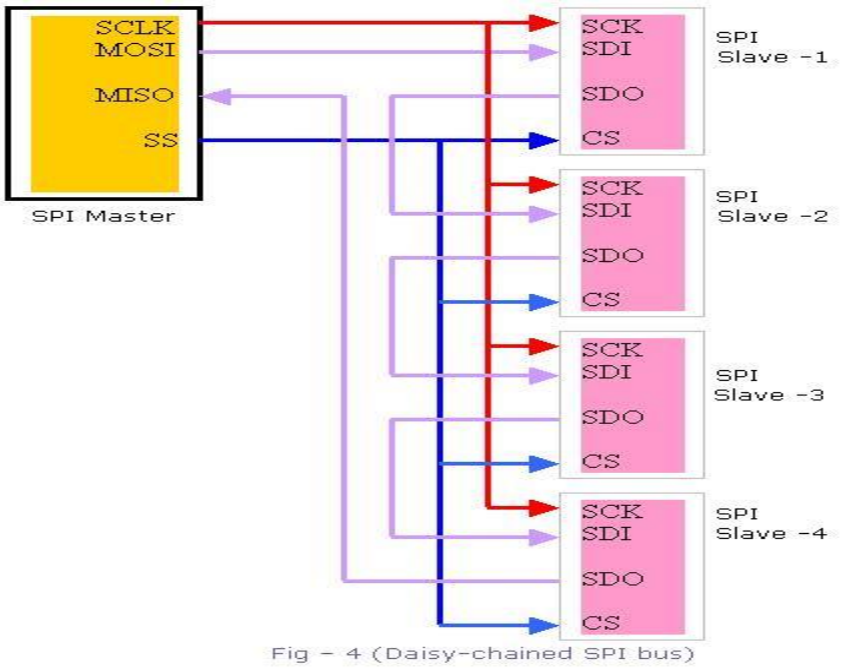

Fig – 4 (Daisy-chained SPI bus)

But we have to remember that the daisy-chain will not work with devices which support or require multiple bytes operation.

Independent slave configuration:

This is the typical SPI-bus configuration with one SPI-master and multiple slaves/peripherals. In this independent or parallel slave configuration,

1.All the clock lines (SCLK) are connected together.
2. All the MISO data lines are connected together.
3. All the MOSI data lines are connected together.
4. But the Chip Select (CS) pin from each peripheral must be connected to a separate Slave Select (SS) pin on the master-microcontroller.

Queued Serial Peripheral Interface (QSPI)

The queued serial peripheral interface (QSPI) is another type of SPI controller, not another bus type. Or in other words it is just an extension to the SPI-bus.
The difference is that it uses a data queue with programmable queue pointers that allow some data transfers without CPU intervention. It also has a wrap-around mode that allows continuous transfers to and from the queue with no CPU intervention. As a result, the peripherals or the slaves appear to the CPU as memory-mapped parallel devices. This feature is useful in applications such as control of an Analog to Digital converter.
The QSPI has got some more programmable features like chip selects and transfer length/delay.

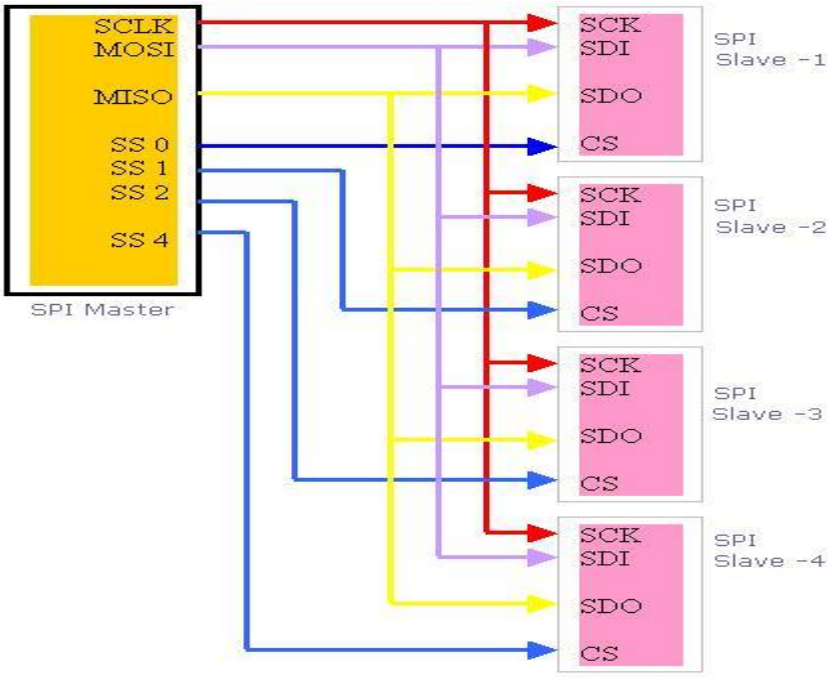

Fig – 4 (SPI bus in independent slave configuration)

Advantages of SPI

1. Full duplex communication
2. Higher throughput than I²C protocol
3. Not limited to 8-bit words in the case of bit-transferring
4. Arbitrary choice of message size, contents, and purpose
5. Simple hardware interfacing
6. Typically lower power requirements than I²C due to less circuitry.
7. No arbitration or associated failure modes.
8. Slaves use the master's clock, and don't need precision oscillators.
9. Transceivers are not needed.
10. At most one "unique" bus signal per device (CS); all others are shared

Disadvantages of SPI

1. Requires more pins on IC packages than I²C
2. No in-band addressing. Out-of-band chip select signals are required on shared busses.
3. No hardware flow control
4. No slave acknowledgment
5. Multi-master busses are rare and awkward, and are usually limited to a single slave.
6. Without a formal standard, validating conformance is not possible
7. Only handles short distances compared to RS-232, RS-485, or CAN.