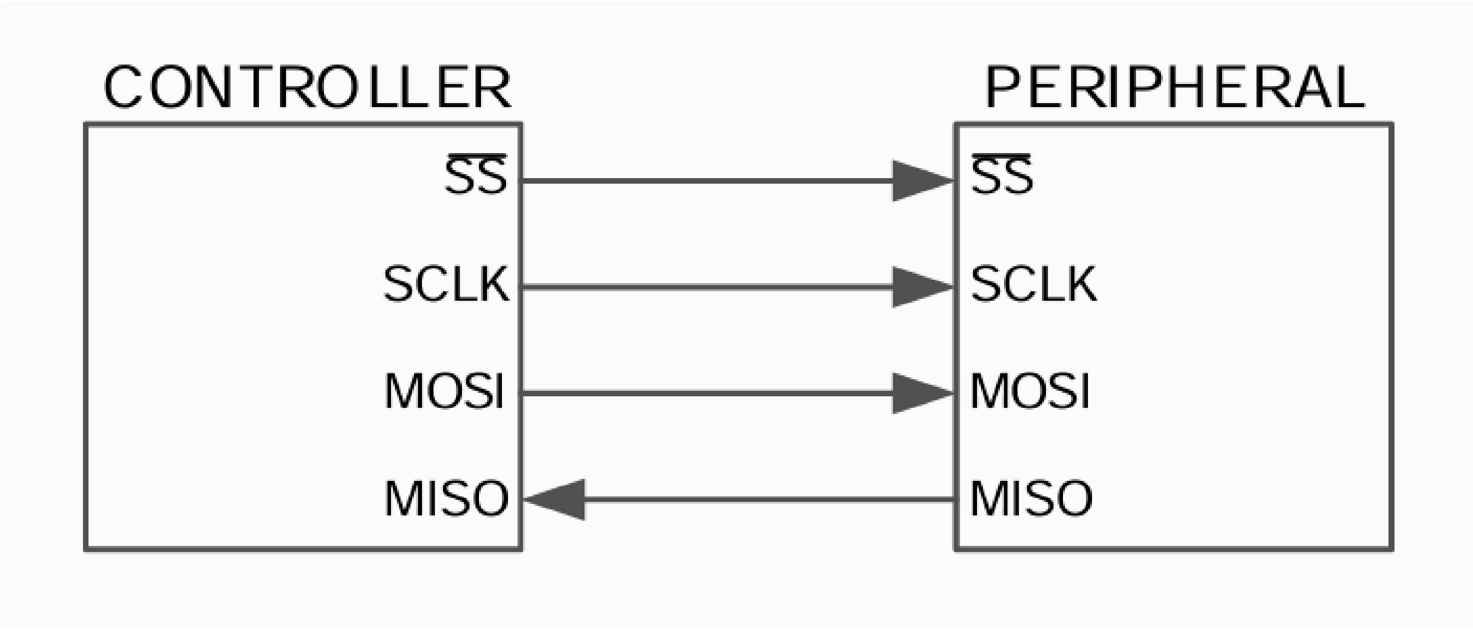
SPI PROTOCOL

PROTOCOL OVERVIEW

Serial Peripheral Interface

- Synchronous serial communication protocol.
- Basically for short distance wired communication between integrated circuits.
- Follows a master slave architecture where a master generates the clock signal, initiates communication and controls data flow.
- Communications are transmitted in binary, constructed from bits
- Transmission usually consists of 8 bits.



Key features

- Full duplex serial communication protocol where the transmission and response happen simultaneously.
- 8-bit Baud Clock generator
- Programmable character length (2 to 16 bits)
- Interrupt capability
- DMA support (read/write synchronization events)
- Up to 66 MHz operation
- Basically faster than I2C and UART.
- SPI do not use device addresses.

Abbr.	Name	Description
$\overline{SS}$	Slave Select	Active-low chip select signal from master to enable communication with a specific slave device
SCLK	Serial Clock	Clock signal from master
MOSI	Master Out Slave In	Serial data output from master
MISO	Master In Slave Out	Serial data output from slave

- Controller controls the peripheral select and the serial clock
- An SPI bus can have only one controller, but may control multiple slaves
- Each peripheral has a peripheral select for independent control
- Data can be transmitted from controller to peripheral or peripheral to controller that may be used as full duplex

Peripheral select SS

- Selects the peripheral device for communication
- Is often used as active low, which is often represented by an overbar
- When communication completes SS returns to ideal state.
- Also known as: SS, SSEL, CS, CS, SYNC, nSS, SS#

Serial clock SCLK

- After pulling SS to low then begins clock generation
- SCLK originates from controller and shared with all slaves
- Clock indicates when data should be sampled

MOSI (Master Out Slave In)

- Output from the Controller used to send data to the peripheral device
- Number of bytes depends on implementation
- Multiple bytes can be sent sequentially.
- Also known as: SIMO, MSTR; from the peripheral device: SDI, DI, DIN, SI; from the Controller device: SDO, DO, DOUT, SO

MISO (Master In Slave Out)

- Output from the peripheral device used to send data to the Controller
- Can be shared between all peripheral devices
- Peripheral output becomes high impedance when SS is not selected

- Also known as: SOMI; from the peripheral device: SDO, DO, DOUT, SO; to the controller device: SDI, DI, DIN, SI

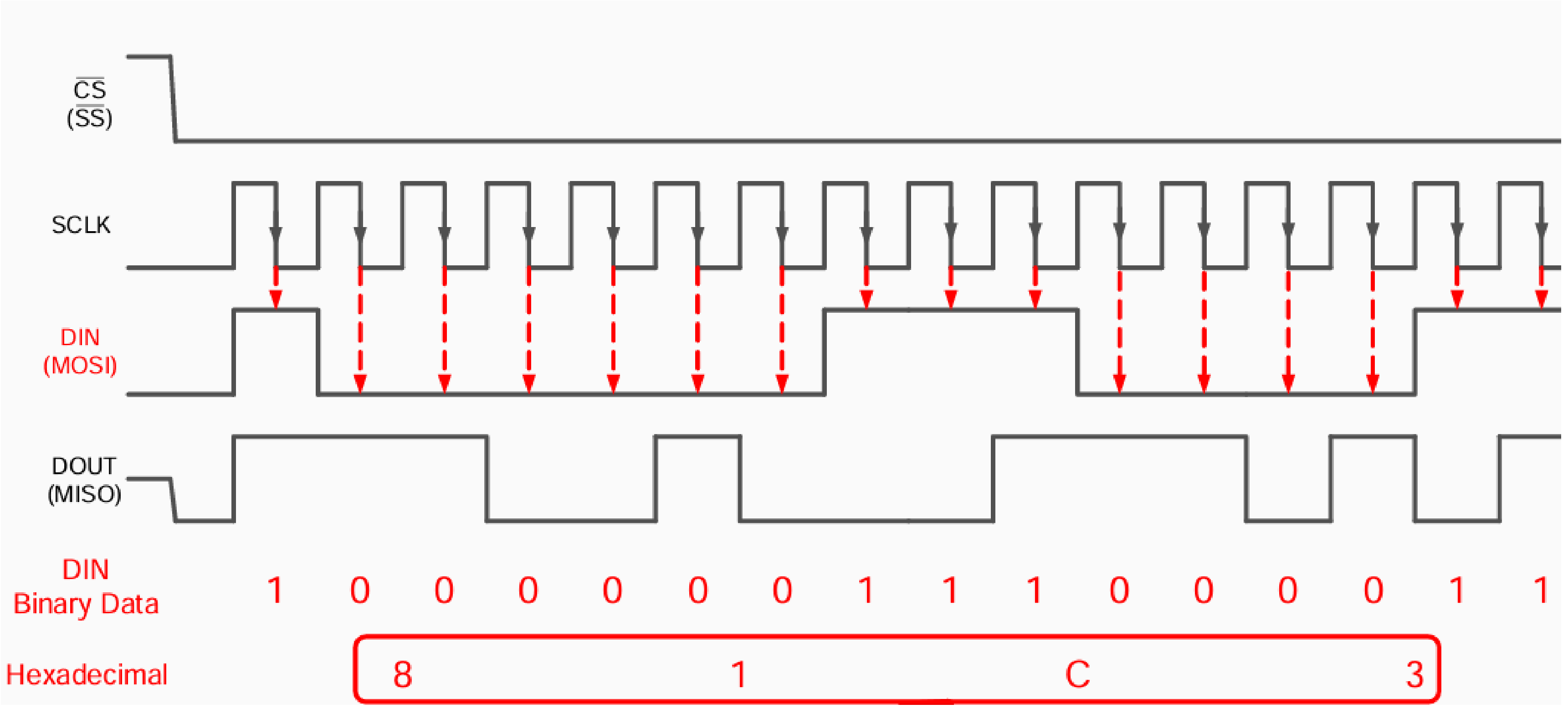
Clock Polarity and Phase

SPI mode	Clock polarity (CPOL)	Clock phase (CPHA)	Data is shifted out on	Data is sampled on
0	0	0	falling SCLK, and when $\overline{SS}$ activates	rising SCLK
1	0	1	rising SCLK	falling SCLK
2	1	0	rising SCLK, and when $\overline{SS}$ activates	falling SCLK
3	1	1	falling SCLK	rising SCLK

Practical Application

Analog to digital conversion - ADS118   16-Bit ADC

Uses Mode 1 - data sampled on the falling edge.



Hardware Requirements

- Master and slave must operate at a compatible logic voltage levels.

- If they differ bidirectional level shifters or voltage translators are required on the SPI lines to prevent damage or logic errors.

- Pull up or Pull down resistors needed on MISO/MOSI/SS lines depending on device specifications.

SSPSTAT - Status Register

Bit 7 SMP - Sample Bit - Master mode , Slave mode

Bit 6 CKE - SPI clock select bit

Bit 0 BF - Buffer Full Status bit(Recieve mode only)

I2C Only (bit 5-1)

D/A - Data/Address bit

P - Stop bit

S - Start bit

R/W - Read/Write

UA - Update Address bit

- The lower six bits are read only
- The upper two bits are read/write

SSPCON - Control Register

Bit 7	WCOL - Write collision detect bit
Bit 6	SSPOV - Recieve overflow indicator bit
Bit 5	SSPEN - Synchronous serial port enable bit
Bit 4	CKP - Clock polarity select bit
SSMP3 - SSMP0	- Synchronous Serial port mode select bits

SSPSR and SSBUF

- SSPSR is the shift register used for shifting data in or out.
- SSPBUF is the buffer register to which data bytes are written to or read from.
- In receive operations, SSPSR and SSPBUF together create a double-buffered receiver.
- When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

Initialization

- The SPI initialize Master function is used to start the SPI communication as the master.
- Inside this function we set the respective pins RC5 and RC3 as output pins.
- Then we configure the SSPSTAT and the SSPCON register to turn on the SPI communications.

```
void SPI_Initialize_Master()
{
TRISC5 = 0;
SSPSTAT = 0b00000000; //
SSPCON = 0b00100000; //
TRISC3 = 0; //Set as output for slave mode
}
```

```
void SPI_Initialize_Slave()
{
TRISC5 = 0;
SSPSTAT = 0b00000000;
SSPCON = 0b00100000;
TRISC3 = 1; //Set as in out for master mode
}
```

The SSPSTAT and the SSPCON is set in the same way for both the slave and the master.

Testing and debugging strategies

Basic data transfer

- Loopback test - Connect MISO to MOSI - Send and verify data
- Check if the pheripheral is enabled correctly, data transfer, clock , CPHA, CPOL.

Register inspection

- Set breakpoints in the code at points where data is transmitted/received.
- Monitor register values using MPLAB IDE
- If communication fails check WCOL,BF or clock edge bits for timing issues.

Mode compatibility

- Compare CPOL and CPHA settings between master and slave
- Adjust mode bits in SPI control registers to match slave timing