

Basics of SPI: Serial Communications

TIPL 6001

TI Precision Labs – Digital Communications

Presented by Alex Smith

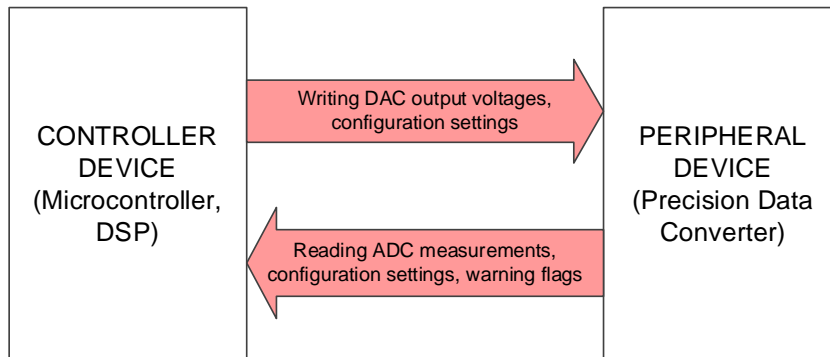
Prepared by Joseph Wu



1

Hello, and welcome to our in-depth look at communications with precision data converters. In this video, we describe digital communications and the basics of Serial Peripheral Interface (or SPI) communication. We'll discuss the communication structure and the required digital lines. Then we'll show different modes of SPI communication and finally give an example of how SPI data is transmitted to and received by a peripheral device.

Common Communication Standard



Many precision analog devices have a digital interface between a microcontroller, known as a controller and a controlled device, known as a peripheral. Using precision ADCs as an example, the peripheral ADC measures an input voltage, and the Controller microcontroller reads the data through digital communications. With precision DACs as the second example, the Controller microcontroller writes a digital word to the DAC to set an output voltage. In both types of devices, this communication may be bi-directional. Many precision data converters may require some configuration and programming.

Digital communications are needed to write to and read from the device. For there to be communication between the two devices, a common standard is required so that both the controller and peripheral understand each other.

Communication Through Bits

Communications are transmitted in binary, constructed from bits, and also be organized in hexadecimal

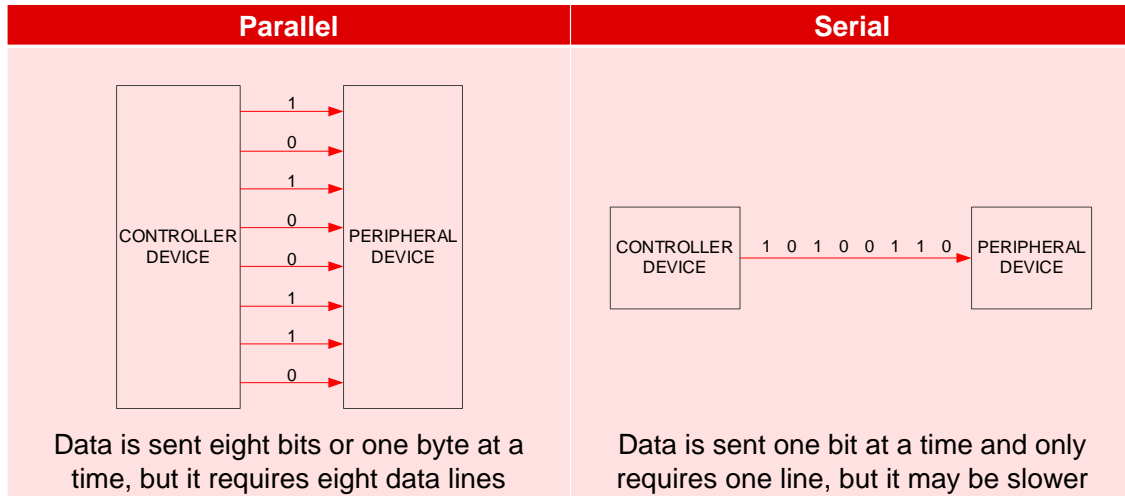
Eight bits is one byte → 1 0 1 0 0 1 1 0 →

Every four bits can be represented as hexadecimal → A 6 ←

Bits	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Communications are made with transmissions of words in binary notation. These communication bits of ones and zeros are often organized in groups of 8 known as a byte. For convenience, bytes can be organized into two digital hexadecimal numbers. Often several bytes are used to send data back and forth between the microcontroller controller and peripheral device. Multiple bytes may be used to represent the output data for ADCs, output voltages for DACs, or to send commands and configurations to the data converter.

Parallel vs Serial



Digital communications can take many forms, but there are two primary structures that are often used.

First, the communications can be parallel, where one line is used for each bit of data. For example, a single byte is transmitted over eight parallel digital lines at once.

Second, the communications can be serial, where all the data bits are on a single digital line. In a serial communication, bits are sent serially, one after another. Serial communications are more widely used in modern devices, reducing the number of communication lines between the Controller and the peripheral device. The figures here show the data going from the controller to the peripheral, but separate lines are required for transmitting data from the peripheral back to the controller.

For precision data converters, most use forms of serial communication. Most TI devices use one of two different serial interface standards, using either four-wire Serial Peripheral Interface (or SPI) or a two-wire I2C. This presentation discusses SPI and its implementation. I2C is discussed in a different presentation.

Voltage Levels

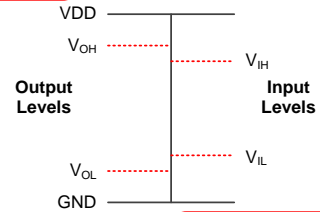
High Voltage = 1 →

Serial Data

Low Voltage = 0 →

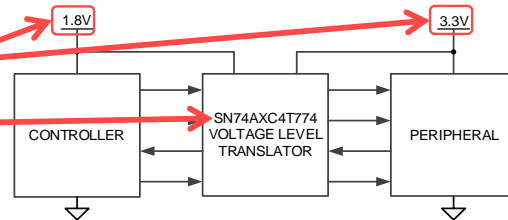
1 0 1 0 0 1 1 0

Datasheets will define the output driving range for digital outputs...



...And define the input range for high and low for digital inputs

If the controller and peripheral operate on different supply voltages, a **voltage level translator** may be required



The lines between the controller and the peripheral device are driven to one of two voltage levels. If the voltage is near zero, then the line is decoded as a digital zero. If the voltage is near the digital supply, then the line is decoded as a digital one.

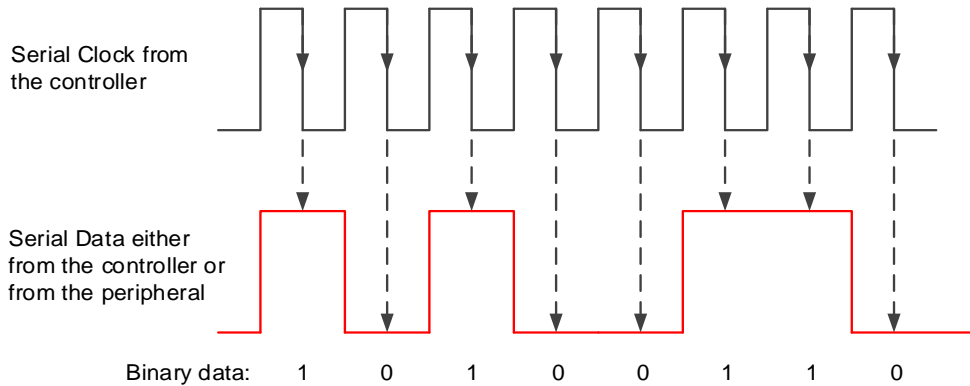
Note that both the controller and the peripheral need to agree what these levels are so that the communication is decoded correctly. Digital outputs must be driven high enough to be decoded as a digital one and driven low enough to be decoded as a digital zero. The datasheet defines the minimum output voltage high level and the maximum output voltage for a low level for a digital output. Additionally, the datasheet defines the minimum input voltage that defines a high level and the maximum input voltage that defines a low level.

The communication standards define these levels based on the voltage of the digital supply. If the controller and peripheral digital supplies do not match, then there may be problems with communications. A voltage level translation device may be required to bridge the communications from one supply level to another.

Common Timing

A serial clock is used to determine a common timing:

In this example, the data is read on the falling edge of the serial clock



Communications also require a common timing between the controller and the peripheral device. The two devices must agree on the timing, knowing when the voltage levels should be read by the peripheral device to define when the bit transmission occurs. Often, a serial clock from the controller is used to define the timing of communication. Reading the bit may occur on the rising or falling edges of the serial clock. In this example, the data is read on the falling edge of the serial clock. Timing is important to the communications and is described in a different video.

Serial Peripheral Interface (SPI)

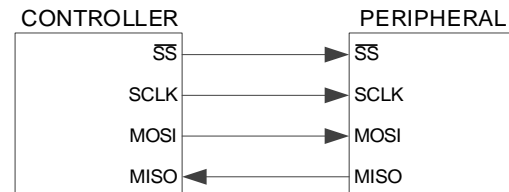
SPI

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



The most common serial interface used in precision data converters is a standard known as Serial Peripheral Interface abbreviated as SPI.

There are two control lines for SPI. The controller, usually a microcontroller or DSP, controls a peripheral select and the serial clock used for data synchronization. An SPI bus can control multiple peripherals but there can only be one controller. Each peripheral requires its own peripheral select line from the controller.

Additionally, there are two data lines. One line sends data from the controller to the peripheral and another sends data from the peripheral to the controller. The data transmission may be full duplex, meaning that data can be transmitted in both directions at the same time, using the same SCLK pulses for clocking data.

The next slides describe the four SPI lines.

SPI connections: SS

SS

Peripheral select: Selects the peripheral device for communication

Is often used as active low, which is often represented by an overbar

Also known as: \overline{SS} , SSEL, \overline{CS} , CS, \overline{SYNC} , nSS, SS#

Commonly labeled as \overline{CS} or \overline{SYNC} in TI data converters



SS is the peripheral SELECT. This line is used to select the peripheral device for communication. If there are multiple devices on the SPI bus, each device has its own peripheral select. peripheral Select is often used as active low, indicated by an overbar. When the device is selected, peripheral Select goes low to activate communication. When the device is not selected, the peripheral Select is set high. peripheral Select may also be labeled as SSEL, CS, CS bar, CE, nSS, or SS#. In many precision ADCs from TI, this chip select line is commonly labeled as CS bar, used as active low. In many TI precision DACs, this line is commonly labeled SYNC bar, also active low.

SPI connections: SCLK

SCLK

Serial Clock: Synchronizes data transmission between the controller and peripheral

SCLK originates from controller and shared with all slaves

Clock may idle high or low

Data is clocked in on either the rising or falling edge of the clock

Also known as: SCK



SCLK is the SERIAL CLOCK. As mentioned previously, SCLK is used by the controller to synchronize the data sent between the controller and the peripheral device. The SCLK originates from the controller and is sent to all peripheral devices. This serial clock may not always be in operation, and may be idle as a high or low signal. Data is clocked into the controller or peripheral at either the rising edge or falling edge of the serial clock. The SCLK line may also be labeled as SCK.

SPI connections: MOSI

MOSI

Controller Out, Peripheral In: Output from the Controller used to send data to the peripheral device

Can be shared between peripheral devices

Also known as: SIMO, MSTR; from the peripheral device: SDI, DI, DIN, SI; from the Controller device: SDO, DO, DOUT, SO

Commonly labeled as DIN in TI data converters



MOSI is the controller OUT/peripheral IN. This line is used to send data from the controller to the peripheral device. The peripheral device reads the MOSI line based on synchronization from the SCLK. Again, this line may have different names depending on the device manufacturer. On the peripheral device it may be known as SIMO, SDI, DI, DIN, or SI. This line is connected to the controller device on some serial out line labeled as SDO, DO, or DOUT. In precision data converters, if there are registers that require configuration, the configuration data are sent on this line. If the device is not active, as the controller has not selected the device to be active with peripheral select, this line is deactivated and the peripheral does not receive data on MOSI. In many precision data converters, TI commonly labels this digital input as DIN.

SPI connections: MISO

MISO

Controller In, Peripheral 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

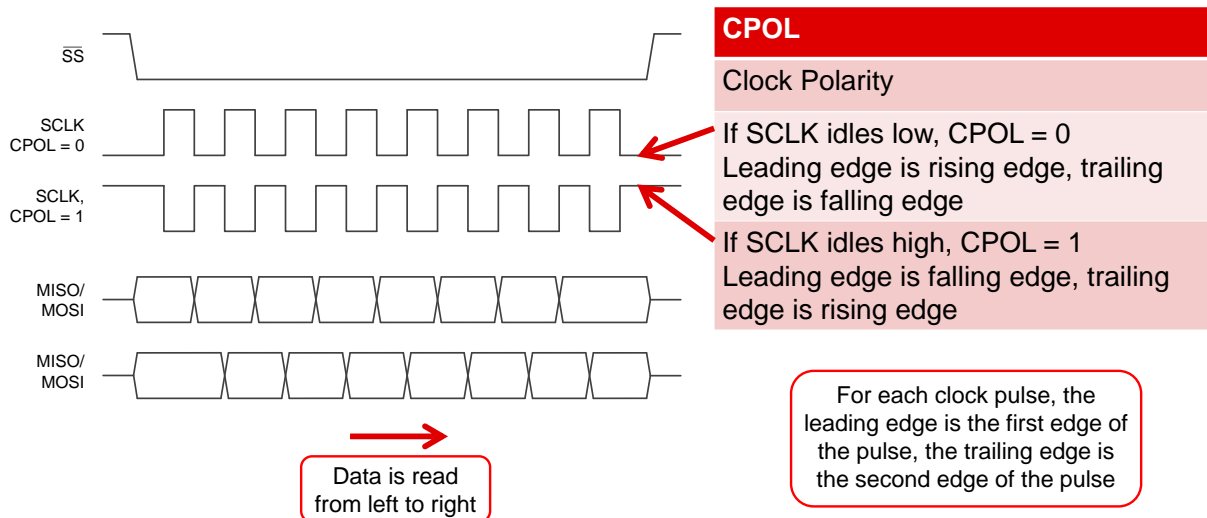
Also known as: SOMI; from the peripheral device: SDO, DO, DOUT, SO; to the controller device: SDI, DI, DIN, SI

Commonly labeled as DOUT in TI data converters



MISO is controller IN/peripheral OUT. This line is used to send data from the peripheral device out to the controller. If the controller has not selected the device to be active with peripheral select, this line becomes high impedance or Hi-Z, allowing for a system with multiple devices to share this line. The peripheral device sends data out on the MISO line and is synchronized with the SCLK. On the peripheral device, it may be known as SOMI, SDO, DO, DOUT, or SO. This line is connected to the controller device on some serial out line labeled as SDI, DI, or DIN. In many precision data converters, TI commonly labels this digital output as DOUT.

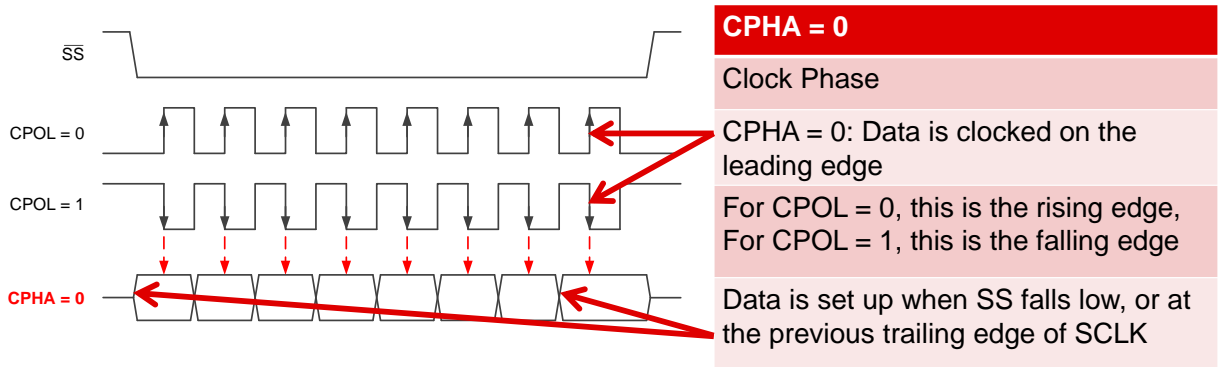
SPI Clock Polarity: CPOL



As previously mentioned, data from the controller and peripheral may be read on either the rising or falling edge of SCLK. Additionally, the clock polarity is important in defining the leading and trailing edges on which data is clocked in and out of the device. This enables several modes of SPI communication. There are two parameters used to determine modes of operation.

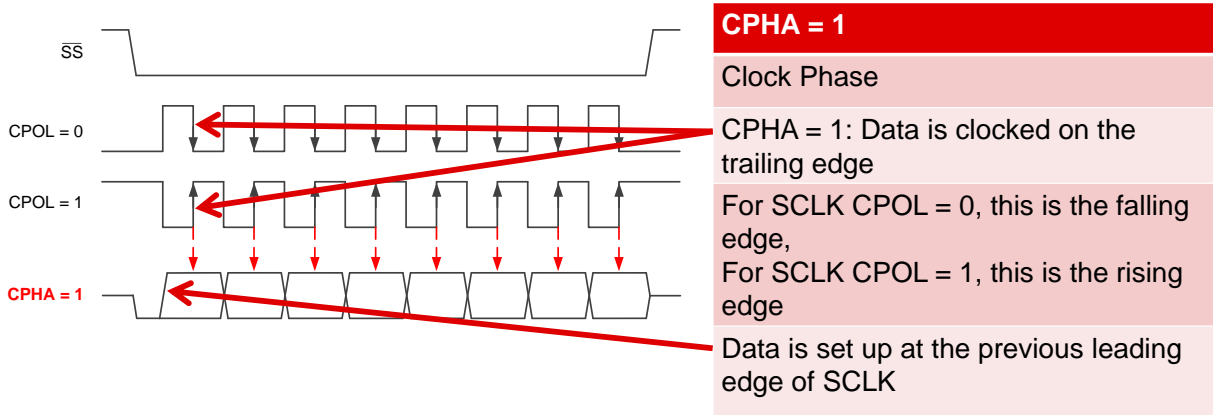
The first parameter CPOL, determines the clock polarity for the serial clock. When CPOL equals zero, then the clock idles low. The leading edge is a rising edge and the trailing edge is a falling edge. When CPOL is equal to one, the clock idles high. The leading edge is a falling edge and the trailing edge is a rising edge.

SPI Clock Phase, CPHA = 0



The second parameter, CPHA, determines the clock phase for which the data is acquired. For the CPHA equal to zero, data is captured on the leading edge of the clock pulse. Data is set up on the trailing edge of SCLK for the first bit, when SS bar is set low.

SPI Clock Phase, CPHA = 1



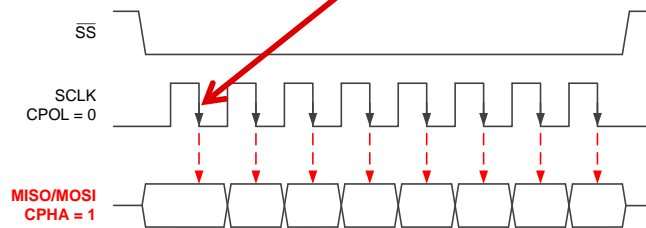
For CPHA equal to one, the data is captured on the trailing edge of SCLK. In both clock phase modes, data is set up on the previous clock edge.

SPI Mode Numbers

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

Example:

SPI Mode 1 – SCLK idles low, data is read on the trailing edge of the clock



Combinations of CPOL and CPHA allow for four modes of operation of SPI. The table in this slide shows SPI modes 0 through 3. Each defines a different clock polarity and phase for SPI communication. Note that different manufacturers of SPI devices and microcontrollers may have different definitions of SPI mode numbers. Consult the datasheets for both the controller and peripherals to determine the clock polarity and phase information.

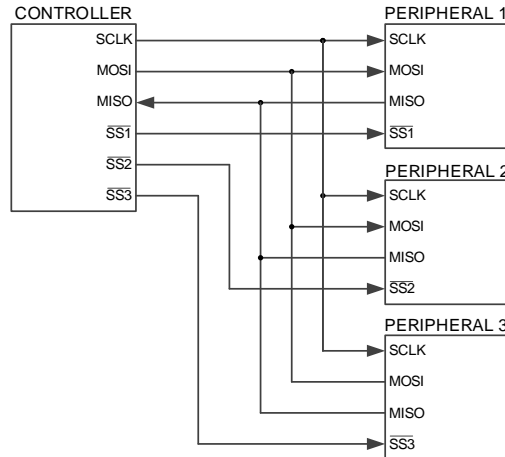
Controlling Multiple Peripherals: Multiple Peripheral Selects

Controlling Multiple Peripherals Method 1 – Multiple SS

Each device has an independent peripheral select

SCLK, MISO, MOSI are each shared between devices

If device is not selected MISO is becomes high impedance



SPI can be used to communicate with multiple devices. This can be done in two different ways.

First, because SPI uses peripheral select lines, the controller can communicate with multiple devices. Each peripheral has its own peripheral select coming from the controller, while the remaining lines are shared. When peripheral select is selected for the device, the remaining SPI lines are active, when the peripheral select is not selected, the SPI lines are ignored. When not selected, each peripheral MOSI and SCLK line is ignored and each MISO line becomes Hi-Z so that an active peripheral may drive the MISO without contention from other peripherals trying to drive the same line.

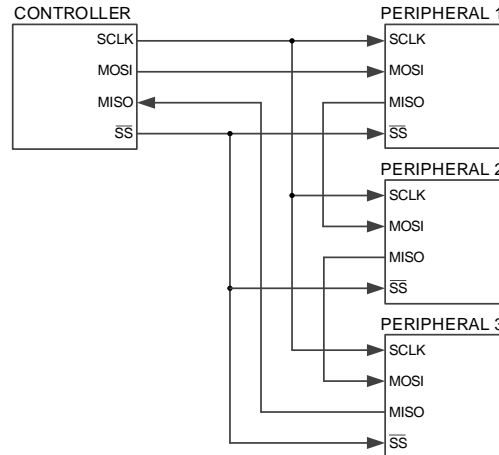
Controlling Multiple Peripherals: Daisy Chaining

Controlling Multiple Peripherals Method 2 – Daisy Chain

Single SS controls all peripheral devices

Data is sent from one device to the next as in a chain; MISO from one device connects to the MOSI of the next device

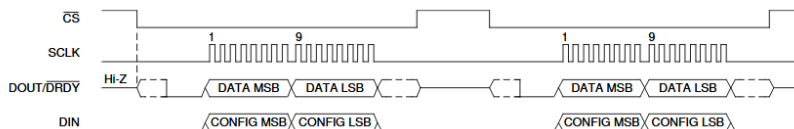
Not all devices support daisy chaining of communications



For some systems, a single SPI drives one peripheral, while other peripherals are daisy-chained together. In a daisy chained system, SCLKs and peripheral Selects may be shared by all peripherals, However, the controller connects to a single MOSI of the first peripheral. The MISO of the first peripheral connects to the MOSI of the next peripheral. This chains all of the peripheral together from one MISO to the next MOSI. For the last peripheral device, the MISO connects to the controller. All data from the devices are passed through the chain and collected at the end from the final peripheral.

Not all devices are able to be controlled and read through a daisy chained SPI connection. Devices must be specifically designed for this communication.

SPI Communication Example: ADS1118



ADS1118

16-bit ADC

Uses SPI Mode 1

Data is clocked in and out of the device at the same time

Configuration register is 16 bits

Diagram shows two transfers of data

BIT	FIELD	SETTING	DESCRIPTION
15	OS	1	Start conversion
14:12	MUX[2:0]	000	Differential input measurement, AINP = AIN0, AINM = AIN1, selection of the first channel
11:9	PGA[2:0]	000	FSR = ±6.144V, sets the ADC to be able to measure the full supply range of 0V to VDD
9	MODE	1	Operation in single-shot conversion mode
7:5	DR[2:0]	100	Data rate = 128SPS
4	TS_MODE	0	Measures analog inputs instead of the temperature sensor
3	PULL_UP_EN	0	Disables the internal pullup resistor on the DOUT/DRDY
2:1	NOPI[1:0]	01	Write 01 to these bits to ensure that the device writes to the configuration register
0	RESERVED	1	Always write 1 to this bit

In this example, the data sent to the device shown in the Setting column

Here the configuration register is set to: 1000 0001 1100 0011 (81C3h)



TEXAS INSTRUMENTS

18

Here is an example of SPI data communication. This comes from the communication of the ADS1118 precision ADC.

The ADS1118 is a 16-bit ADC that uses SPI Mode 1, with CPOL equal to 0 and CPHA equal to 1. Here the SCLK idles low and the data is clocked in and out of the device on the trailing edge of SCLK. In the 16-bit data transmission cycle, two bytes of data are clocked into a configuration register. At the same time, 16-bits of ADC data is clocked out of the device. The ADC has a 16-bit configuration register with settings showing in the table. These configuration register values program the device to the settings shown in the description column. For this example, the setting we want to write to the device is 81C3 in hexadecimal.

SPI Communication Example: ADS1118 Write

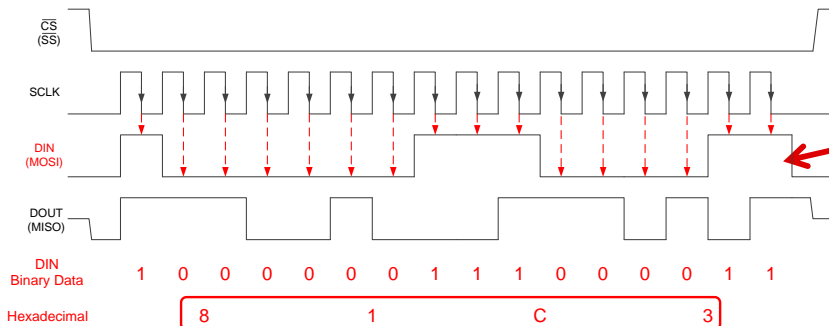
ADS1118 Example Write

CS selects the device

SCLK idles low and data is clocked on the falling edge

DIN sends 81C3h to the device

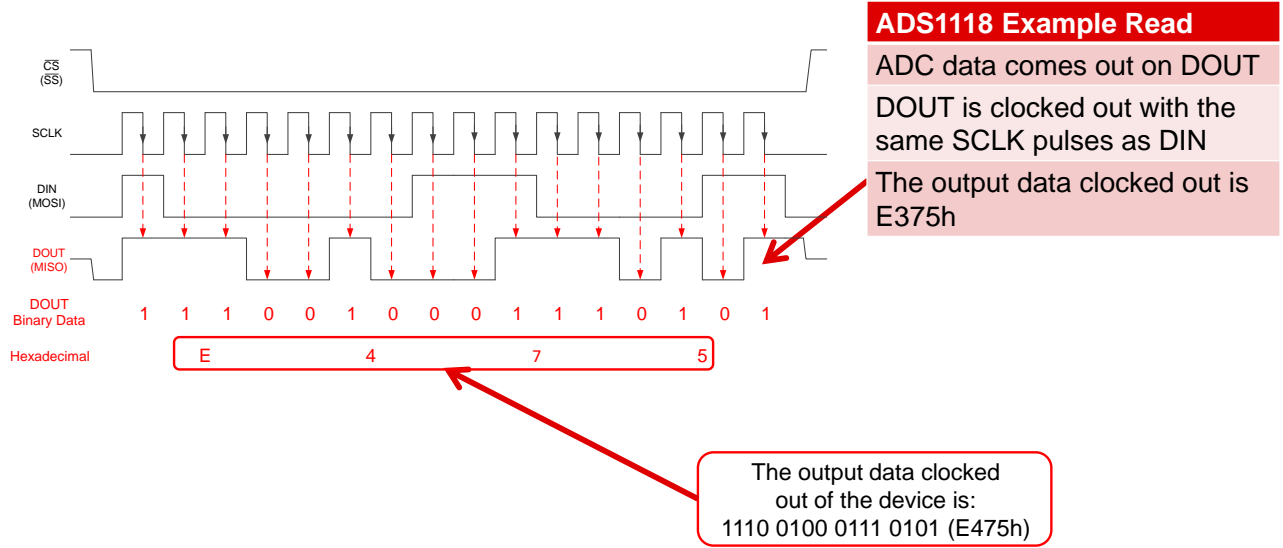
DOUT is clocked in with the same SCLK pulses



The configuration register is set to:
1000 0001 1100 0011 (81C3h)

Continuing with the example, this is the 16-bit data transmission cycle for the ADS1118 showing the SPI lines. Again, the SCLK idles low, and data is transmitted at the falling edge of SCLK. The write to the device is done on the MOSI, or on this device, the pin is labeled as DIN. The configuration register is set to 81C3 in hexadecimal.

SPI Communication Example: ADS1118 Read



At the same time, the ADC data is clocked out of the ADS1118. Here, a random output data word is clocked out of the device. Looking at the DOUT line, the binary output is read as E475 in hexadecimal.

**Thanks for your time!
Please try the quiz.**

That concludes this video – thank you for watching! Please try the quiz to check your understanding of this video’s content.

Quiz: Basics of SPI: Serial Communication

1. The following binary output is what value in hexadecimal?
 - a. 59
 - b. 6A
 - c. 95
 - d. A6

1 0 1 0 0 1 1 0

Quiz: Basics of SPI: Serial Communication

1. The following binary output is what value in hexadecimal?

- a. 59
- b. 6A
- c. 95
- d. A6

1 0 1 0 0 1 1 0

A

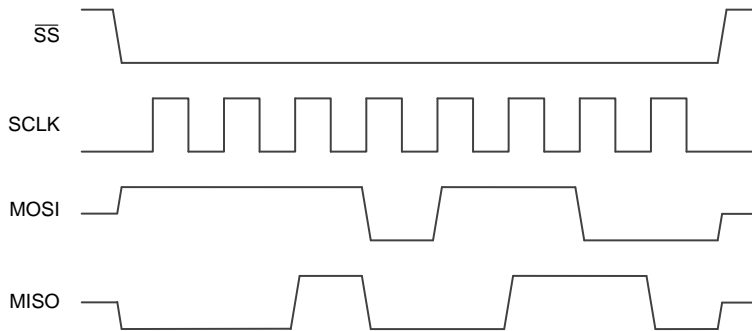
6

Bits	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Quiz: Basics of SPI: Serial Communication

2. The following diagram is CPOL = 0, CPHA = 1. What hexadecimal byte is the sent from the peripheral to the controller in this transaction?

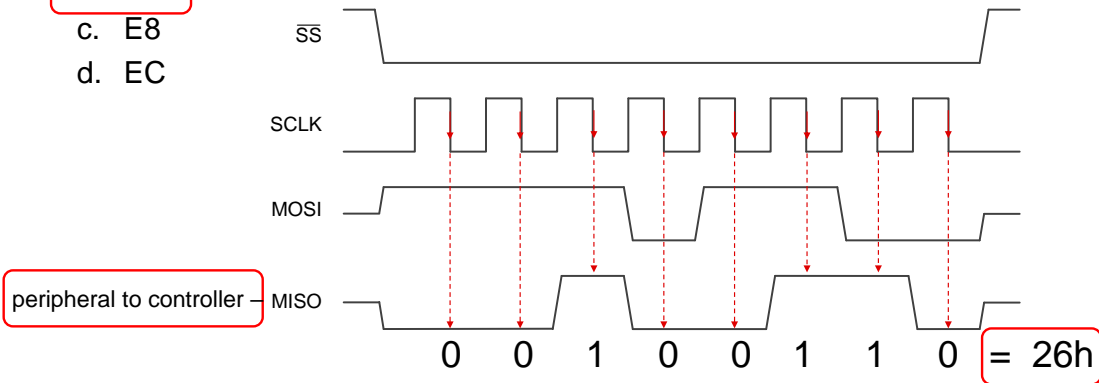
- a. 24
- b. 26
- c. E8
- d. EC



Quiz: Basics of SPI: Serial Communication

2. The following diagram is CPOL = 0, CPHA = 1. What hexadecimal byte is the sent from the peripheral to the controller in this transaction?

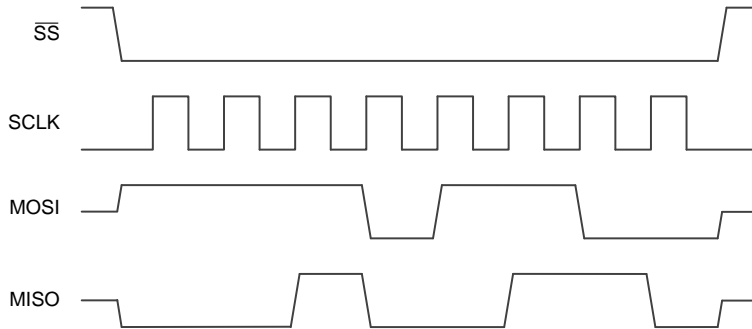
- a. 24
- b. 26
- c. E8
- d. EC



Quiz: Basics of SPI: Serial Communication

3. The following diagram is CPOL = 0, CPHA = 1. What hexadecimal byte is the sent from the controller to the peripheral in this transaction?

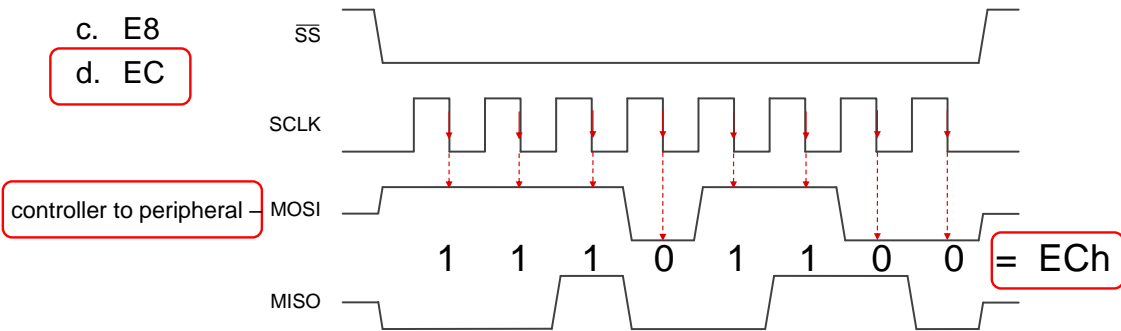
- a. 24
- b. 26
- c. E8
- d. EC



Quiz: Basics of SPI: Serial Communication

3. The following diagram is CPOL = 0, CPHA = 1. What hexadecimal byte is the sent from the controller to the peripheral in this transaction?

- a. 24
- b. 26
- c. E8
- d. EC**



Thanks for your ti



© Copyright 2020 Texas Instruments Incorporated. All rights reserved.

This material is provided strictly "as-is," for informational purposes only, and without any warranty.
Use of this material is subject to TI's [Terms of Use](#), viewable at TI.com

