Course Introduction

Purpose:

- The intent of this course is to familiarize you with one of the common serial communication modules found on S08-based microcontrollers, the Serial Peripheral Interface (SPI).

Objectives:

- Describe the uses and features of the SPI.
- Identify the steps to configure the SPI.
- Describe the master-slave data transmission process.

Content:

- 18 pages
- 3 questions

Learning Time:

- 25 minutes

Welcome to the HCS08 Serial Peripheral Interface (SPI) course. The intent of this course is to familiarize you with this common serial communication module found on S08-based microcontrollers. You will learn about the features of the SPI as well as how to configure and use the SPI module. You will examine the five SPI module registers used to select SPI options, control baud rate, report SPI status, and to hold transmit/receive data. More specifically, upon completing this course you should understand how to configure the SPI to operate in Master of Slave mode, and you should be able to describe the master-slave data transmission process.

SPI Uses and Features

Uses

- Full-duplex, synchronous serial communication with external peripherals
- Examples: high resolution analog-to-digital converters (ADCs), serial EEPROM memory, or other microcontrollers
- Communication within PCB board at relative high speeds

Features

- Low-cost communications module
- Data transmission pins
- Serial clock pin
- Slave select pin
- Master and Slave mode operation
- Maximum Master mode frequency = bus frequency / 2
- Maximum Slave mode frequency = bus frequency / 4
- Single-wire bidirectional mode
- Selectable MSB-first or LSB-first shifting
- Serial clock phase and polarity options

Let’s begin with a discussion of the uses and features of the SPI module.

The SPI is a full-duplex, synchronous serial communications module. It is intended primarily for communications within a PCB board at relatively high speeds. The SPI provides a cost-effective way to interface serially with external peripherals. Some examples are high resolution analog components such as analog-to-digital converters (ADCs) and digital-to-analog converters (DACs); serial EEPROM memory; LCD modules; and shift registers. The SPI can also allow communication to other MCUs or processors at relatively high speeds.

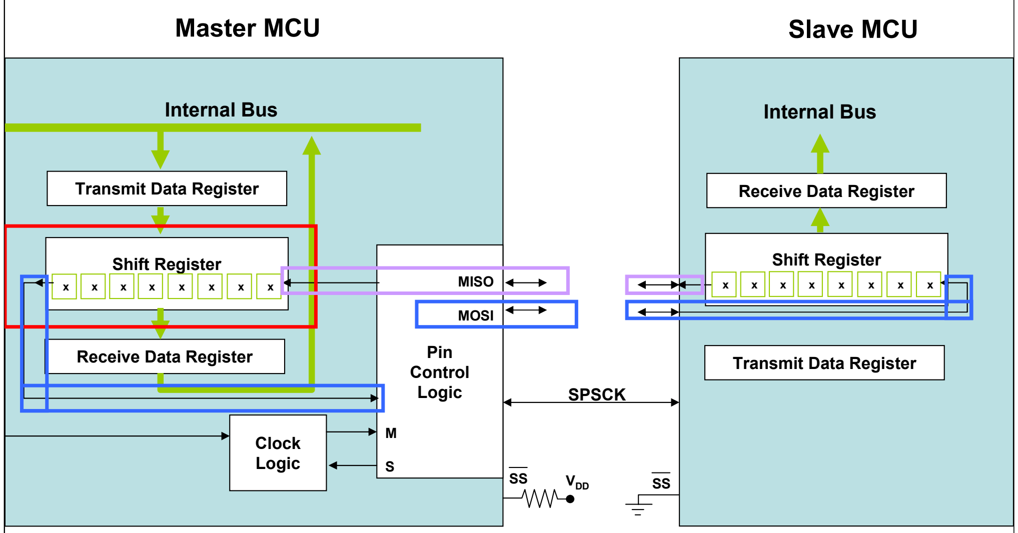
By communicating serially, rather than via parallel, the MCU and the peripherals require fewer pins. This results in smaller packages and makes the SPI a low-cost communication module.

The pins include two data transmission pins for full-duplex operation, a serial clock pin for synchronous communication, and an optional slave select pin that is used to initiate communication with a slave SPI, pending master/slave configurations.

The SPI operates in either Master mode or Slave mode. In Master mode, the SPI generates the synchronous communication clock by dividing the bus clock with a prescale divisor and prescaler selection. You will see a detailed description on selecting the baud rate later in this course. The maximum Master mode frequency is half of the bus frequency. This means that S08 MCUs operating at 20 MHz bus frequency allow a maximum Master mode clock rate of 10 MHz. S08s operating at low voltages and 8 MHz bus frequency provide a Master mode clock rate of 4 MHz. In Slave mode, the maximum frequency is equal to the bus frequency divided by 4 to allow for over-sampling and ensure accurate communication.

Additional features include the option to operate in a single-wire bidirectional mode, which frees up an additional pin in I/O count-sensitive applications; the option to shift the most significant bit (MSB) or least significant bit (LSB) first; and options for the serial clock phase and polarity for capturing each bit of data.

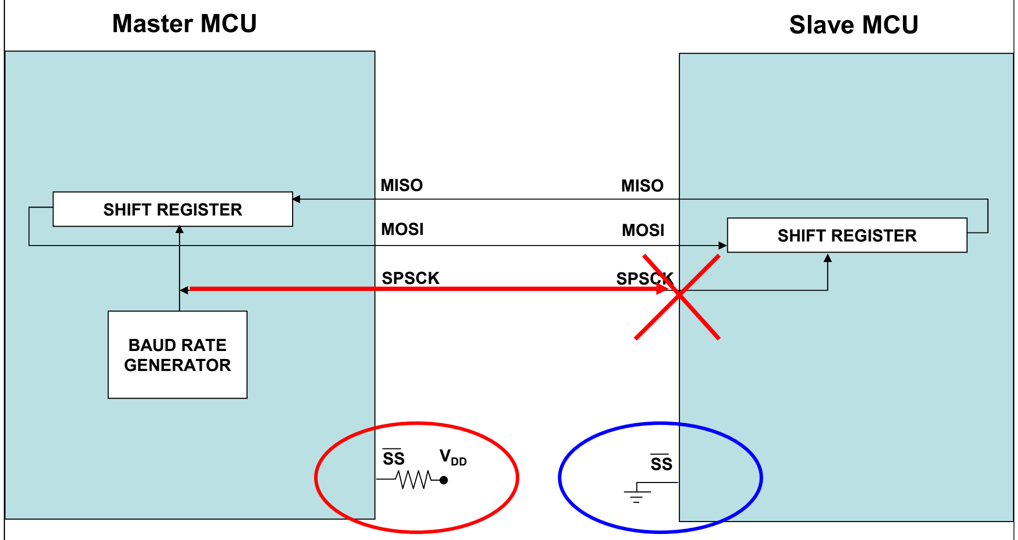
SPI Master-Slave Overview



The SPI is built around a double-buffered 8-bit shift register with both ends of the shift register brought out to MCU pins. One end of the shift register is connected to the master-in slave-out (MISO) pin. This pin acts as an input for the master SPI module and as an output for the slave SPI module. The other end of the shift register is connected to the master-out slave-in (MOSI) pin.

The CPU begins a serial SPI transfer by writing a byte of data to the master's SPI transmit data register (SPID). All 8 bits of data will be automatically transferred serially out the master's MOSI pin synchronized to the master's SPSCK clock output. For each bit shifted out of the master's MOSI pin, a bit is shifted in through the master's MISO pin. This allows full-duplex communication. A more detailed example will be shown after we review the SPI module further.

SPI Master-Slave Connections



Next, let’s look at the SPI master-slave connections. The SPI optionally shares four port pins. The function of these pins depends on the settings of the SPI control bits in the two SPI control registers, SPIC1 and SPIC2. When the SPI is disabled via one of its control register bits, these four pins revert to being general-purpose port I/O pins that are not controlled by the SPI. For the purpose of this course, we will assume that the SPI is enabled, and the shared ports reflect the SPI functions.

Here, you can see how the master MCU is connected to the slave. Note that before you enable the SPI modules, you must configure them as a master and slave SPI.

To enable the master SPI, tie the master’s slave select pin, SS_bar, to V_{DD}.

Before enabling the slave SPI, tie the slave’s SS_bar pin to the ground. This allows a master to select the slave. Typically, the master will use its I/O pins to drive the slave’s SS_bar pin. Note that if a slave’s SS_bar pin is set to 1, the slave’s MISO pin is placed in a high-impedance state. When this occurs, the slave will ignore all incoming SPSCK clocks, even if it is already in the middle of a transmission.

SPI Control Registers

SPIC1

	Bit 7	6	5	4	3	2	1	Bit 0
Read: / Write:	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
Reset:	0	0	0	0	0	1	0	0

SPIC2

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	MODFEN	BIDIROE	0	SPISWAI	SPCO
Write:	▨	▨	▨	MODFEN	BIDIROE	▨	SPISWAI	SPCO
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Let's start with a look at the SPI control registers. Here you can see the two SPI control registers, SPIC1 and SPIC2. These read/write registers include the SPI enable control, interrupt enables, and configuration options. Let's review each control bit in detail, starting with the SPIC1 register.

The SPIE bit is a read/write interrupt enable bit that allows CPU interrupt requests from two sources: receive buffer full or mode fault events. When the SPIE is set to 1, a hardware interrupt is requested when either the receive buffer is full represented by Receive Buffer Full Flag (SPRF)=1 in the SPI Status register (SPIS) or when the slave select of the master MCU is pulled low causing a mode fault (MODF=1).

The SPI System Enable bit (SPE) turns the SPI on or off. Setting SPE=0 disables the SPI, halts any transfer that is in progress, clears data buffers, and initializes internal state machines. The SPRF is cleared and the Transmit Data Buffer Empty Flag (SPTEF) in the SPIS is set to indicate that both buffers are empty.

The SPI Transmit Interrupt Enable bit (SPTIE) is a read/write bit that enables CPU interrupt requests to be generated when a byte transfers from the transmit data register to the shift register.

Bit 4 of the SPIC1 register is the SPI Master/Slave Mode Select bit (MSTR). This read/write bit configures the SPI in Master mode when set to 1, and configures the SPI in Slave mode when set to 0.

The Clock Polarity bit (CPOL) is a read/write bit that determines the logic state of the SPSCK pin between transmissions. It effectively places or removes an inverter in series with the clock signal from a master SPI or to a slave SPI device. To transmit data between two SPI modules, the SPI modules must have identical CPOL values.

The Clock Phase bit (CPHA) is a read/write bit that controls the timing relationship between the serial clock and SPI data. To transmit data between two SPI modules, the SPI modules must have identical CPHA values. We will examine phase and polarity later in the course.

The Slave Select Output Enable bit (SSOE) is used in combination with the mode fault enable bit (MODFEN) in SPIC2 and the MSTR to determine the function of the SS_bar pin.

The last bit in the SPIC1 control register is the Least Significant Bit First Enable (LSBFE). It allows you to select the shifter direction. Setting the LSBFE bit to 1 causes SPI serial data transfers to start with the LSB. With the bit set low, data transfers start with the MSB.

SPI Control Registers

SPIC2

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	**0**	**0**	**0**	**MODFEN**	**BIDIROE**	**0**	**SPISWAI**	**SPCO**
Write:	▨	▨	▨			▨		
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

MODFEN	**SSOE (bit 1 of SPIC1)**	**Master Mode**	**Slave Mode**
0	0	General-purpose I/O (not SPI)	Slave Select input
0	1	General-purpose I/O (not SPI)	Slave Select input
1	0	$\overline{SS}$ input for mode fault	Slave Select input
1	1	Automatic $\overline{SS}$ output	Slave Select input

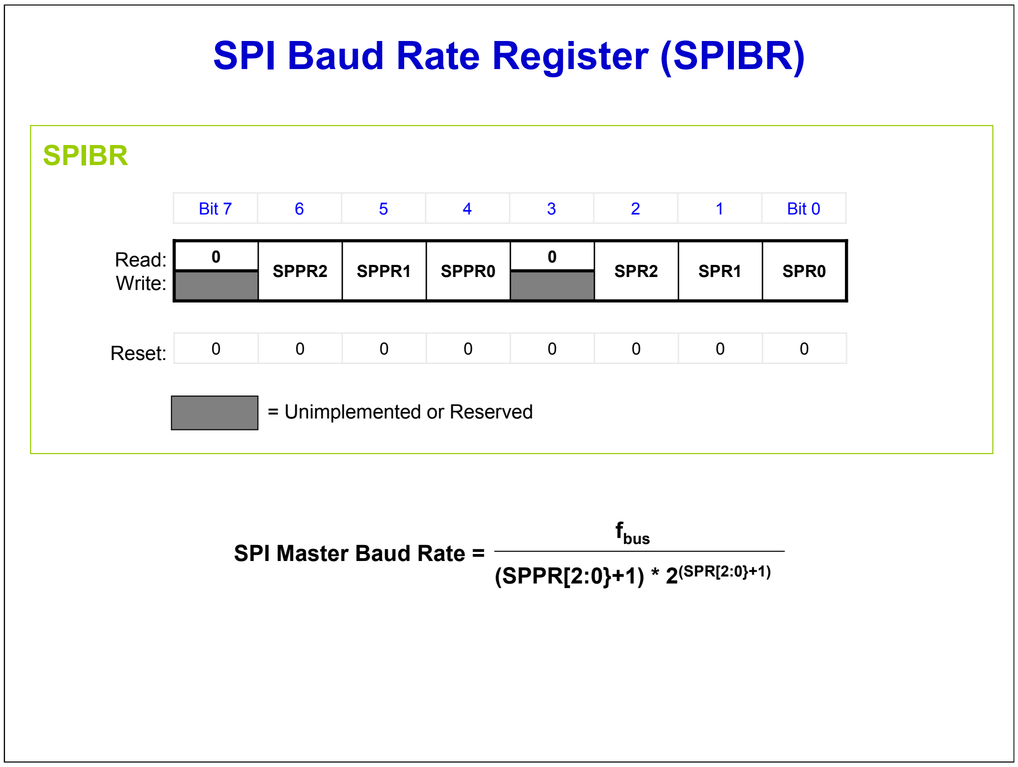
The second SPI control register, SPIC2, is used to control optional features on the SPI system. Bits 7, 6, 5, and 2 are not implemented and always read 0.

The Master Mode – Fault Function Enable bit (MODFEN) together with the SSOE bit and the MSTR bit, determine how the SS_bar pin is used. Note that MODFEN is a read/write bit, the SSOE bit is bit 1 of SPIC1, and the MSTR bit is bit 4 of SPIC1. This table shows you how the various combinations of MODFEN, SSOE, and MSTR bits assign functionality to the SS_bar pin. As you can see, when the SPI is configured to Slave mode, this bit has no meaning or effect.

The Bidirectional Mode Output Enable bit (BIDIROE) is used when the SPI is configured in Single-wire Bidirectional mode to enable the single bidirectional SPI I/O pin as an output driver. When the bit is set low, the pin acts as an input, and when it is set high the pin is enabled as an output. Depending on whether the SPI is configured as a master or a slave, it uses either the MOSI (as a MOMI) or MISO (as a SISO) pin, respectively, as the single SPI data I/O.

The SPI Stop in Wait Mode bit (SPISWAI) is a read/write bit that is used to either keep the SPI clocks running or stopped when the MCU enters WAIT mode. When this bit is set high, the SPI clocks stop when the MCU enters WAIT mode. This reduces overall power consumption.

Finally, the SPI Pin Control bit (SPC0) is a read/write bit that selects whether Single-wire Bidirectional mode or Full-duplex mode is enabled. Single-wire Bidirectional mode is selected when the bit is set high.



SPI Baud Rate Register (SPIBR)

SPIBR

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	**0**	**SPPR2**	**SPPR1**	**SPPR0**	**0**	**SPR2**	**SPR1**	**SPR0**
Write:	▨				▨			
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

$$\textbf{SPI Master Baud Rate} = \frac{f_{bus}}{(SPPR[2:0\}+1) * 2^{(SPR[2:0\}+1)}}$$

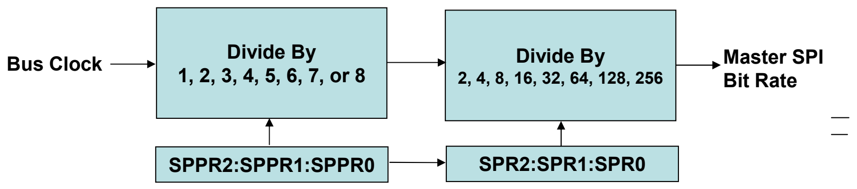
This figure shows the SPI Baud Rate Register (SPIBR). This register is used, in Master mode only, to set the prescaler and the bit rate divisor. With these settings, it determines the bit rate clock output for the SPI master. In Slave mode, this register has no meaning or effect.

SPPR2, SPPR1, and SPPR0 are the three bits that select one of eight divisors for the SPI baud rate prescaler. The output of the prescaler drives the input of the SPI baud rate divider.

SPR2, SPR1, and SPR0 are the three bits that select one of eight divisors for the SPI baud rate divider. The input to the divider comes from the SPI baud rate prescaler, and the output is the SPI bit rate clock (in Master mode).

This equation shows how the respective bit fields SPPR[2:0] and SPR[2:0] determine the master SPI baud rate. Next, we will take a closer look at how this register generates the Master mode clock rate, as well as the divisor values, yielded by the bit fields in this register.

SPIBR


Bus Clock
Divide By 1, 2, 3, 4, 5, 6, 7, or 8
Divide By 2, 4, 8, 16, 32, 64, 128, 256
Master SPI Bit Rate
SPPR2:SPPR1:SPPR0
SPR2:SPR1:SPR0

SPI Baud Rate Prescaler Divisor (=SPPR[2:0]+1)

SPPR2:SPPR1:SPPR0	Prescaler Divisor
0:0:0	1
0:0:1	2
0:1:0	3
0:1:1	4
1:0:0	5
1:0:1	6
1:1:0	7
1:1:1	8

SPI Baud Rate Divisor (=$2^{(SPR[2:0]+1)}$)

SPR2:SPR1:SPR0	Rate Divisor
0:0:0	2
0:0:1	4
0:1:0	8
0:1:1	16
1:0:0	32
1:0:1	64
1:1:0	128
1:1:1	256

Here, you can see the two divisors controlled by the SPIBR in Master mode only. The first divisor, the SPI baud rate prescaler, takes the bus clock and divides by one of eight values, based on the bit field, SPPR2:SPPR1:SPPR0. The corresponding values are shown in the table.

The output of the Prescale divisor then feeds the baud rate divisor. The bit field SPR2:SPR1:SPR0 selects the value with which the input is divided, according to the table. The output drives the Master SPI bit rate onto the SPSCLK pin.

As an example, let’s assume a bus frequency of 8 MHz. If both bit fields, SPPR2:SPPR1:SPPR0 and SPR2:SPR1:SPR0, read 0:0:0, the bus frequency is first divided by 1 by the prescaler then divided by 2 by the baud rate divisor, giving a master baud rate of 4 MHz.

Question

What is the master SPI baud rate if the bus frequency is set to 20 MHz and the SPI baud rate register is set to 0 1 0 0 0 0 1 1? Select the correct answer and then click Done.

a. 4 MHz

b. 0.25 MHz

c. 16 MHz

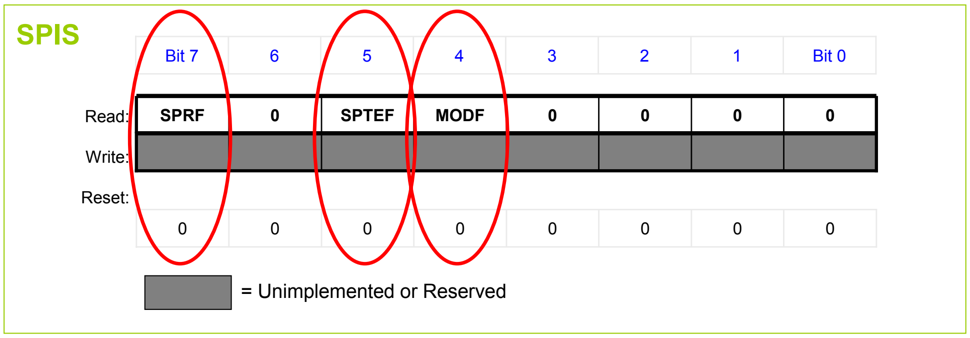
d. 0.3125 MHz

Take a few moments to check your understanding.

Correct.

The input to the baud rate divisor is the bus frequency divided by the prescaler divisor. That input is then divided by the baud rate divisor to give you the master baud rate. Since the prescaler divisor bit field SPPR[2:0] is set to 1 0 0, the prescale divisor is 5. Therefore, the input to the baud rate divisor is 20 MHz divided by five, or 4 MHz. The baud rate divisor bit field SPR[2:0] is set to 0 1 1, so the divisor is 16. Therefore, the master baud rate is 4 MHz divided by 16, or 0.25 MHz. Click the forward arrow to continue on to the next page.

SPI Status Register (SPIS)



SPIS

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	**SPRF**	**0**	**SPTEF**	**MODF**	**0**	**0**	**0**	**0**
Write:	▨	▨	▨	▨	▨	▨	▨	▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented or Reserved

Now, let’s examine the SPIS. It is a read-only register that lets the processor know if either the read buffer or write buffer is full, or if a Master mode fault error has been detected. Writes to this register have no meaning or effect.

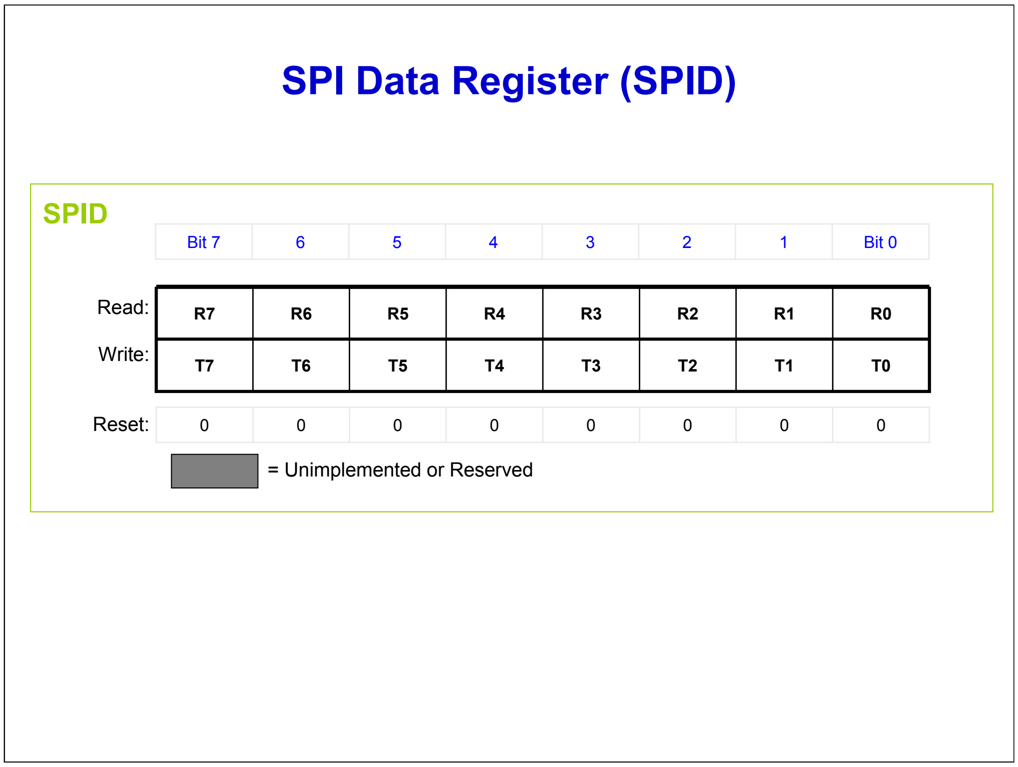
Bit 7 of the status register is the Read Buffer Full Flag (SPRF). It is set high at the completion of a transfer to indicate that received data may be read from the SPI data register. Clearing this bit is accomplished by reading the SPRF when it is set high, then reading the SPI data register.

The Transmit Buffer Empty Flag (SPTEF) is set when there is room in the transmit data buffer. This bit is cleared when the SPI status register is read with SPTEF set high, followed by writing a data value to the transmit buffer at the SPI Data register (SPID). The write to SPID will be ignored if the status register is not read with a high transmit buffer empty flag = 1. If the SPTEF bit goes high, a CPU interrupt request is initiated as long as the SPTIE bit in the SPIC1 is also set high.

When a data byte transfers from the transmit buffer into the transmit shift register, the SPTEF bit is automatically set. For an idle SPI with no data in the transmit buffer or the shift register and no transfer in progress, data written to the data register is transferred to the shifter almost immediately. Therefore, SPTEF is set within two bus cycles allowing a new value to be queued into the buffer.

After completion of the transfer of the value in the shift register, the queued value from the transmit buffer will automatically move to the shifter and the SPTEF will be set to indicate there is room for new data in the transmit buffer. If there is no data waiting in the buffer, the SPTEF bit simply remains high.

Finally, the Master Mode Fault Flag bit (MODF) is set high when the SPI is configured in Master mode, and the slave select input goes low, indicating some other SPI device is also configured as a master. The SS_bar pin will only act as a mode fault error input when the SPI is configured as a master, the MSTR bit = 0, the MODFEN bit = 0, and the SSOE bit = 0. Otherwise, MODF will never be set. Clearing the MODF bit is achieved by reading MODF when it is high and writing to the SPIC1.



SPI Data Register (SPID)

SPID

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R7	R6	R5	R4	R3	R2	R1	R0
Write:	T7	T6	T5	T4	T3	T2	T1	T0
Reset:	0	0	0	0	0	0	0	0

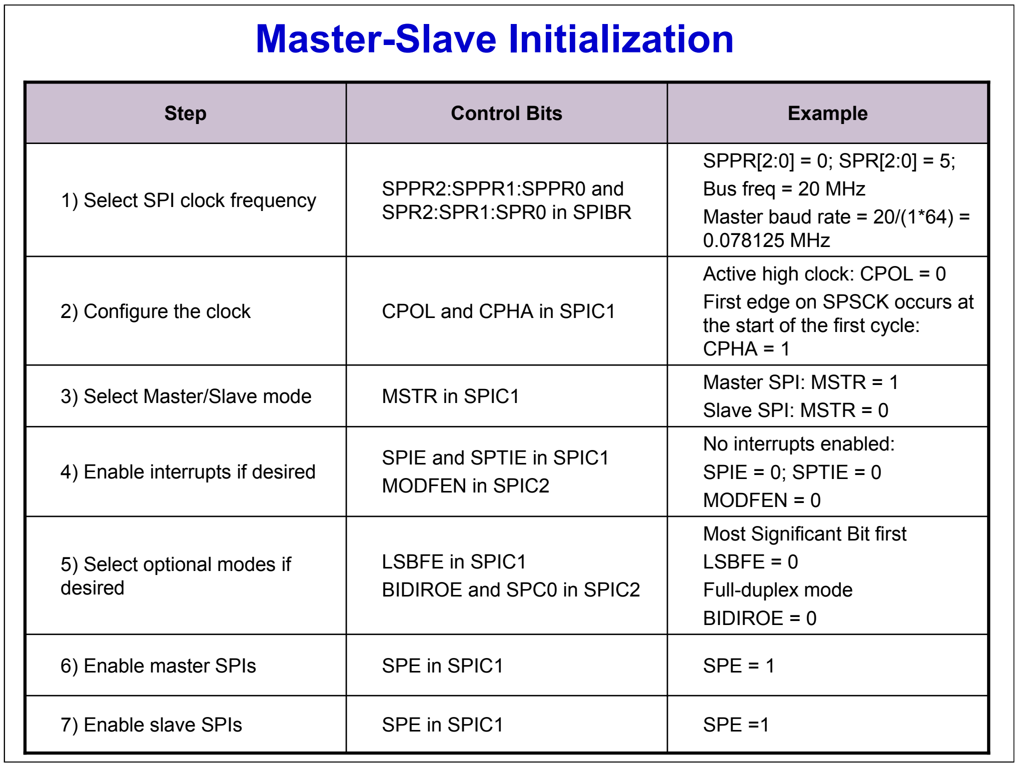
▨ = Unimplemented or Reserved

The last register directly associated with the SPI is the SPID, which holds data that is either received or to be transmitted. The two rows represent the receive and transmit buffers, respectively, as part of the SPI data register.

Reads of this register return the data read from the receive data buffer. Writes to this register write data to the transmit data buffer. In Master mode, writing data to the transmit data buffer initiates an SPI transfer. Remember that before data is written to the transmit data buffer, the SPTEF must be set high to indicate there is room in the transmit buffer to queue a new transmit byte.

Received data may be read from SPID any time after the SPRF is set and before another transfer is finished. Failure to read the data out of the receive data buffer before a new transfer ends causes a receive overrun condition and the data from the new transfer is lost.

Do not use read-modify-write instructions on the SPI data register since the register read is not the same as the register written. This concept is illustrated here by representing the received data buffer and the transmit data buffer as separate rows.



Master-Slave Initialization

Step	Control Bits	Example
1) Select SPI clock frequency	SPPR2:SPPR1:SPPR0 and SPR2:SPR1:SPR0 in SPIBR	SPPR[2:0] = 0; SPR[2:0] = 5; Bus freq = 20 MHz Master baud rate = 20/(1*64) = 0.078125 MHz
2) Configure the clock	CPOL and CPHA in SPIC1	Active high clock: CPOL = 0 First edge on SPSCK occurs at the start of the first cycle: CPHA = 1
3) Select Master/Slave mode	MSTR in SPIC1	Master SPI: MSTR = 1 Slave SPI: MSTR = 0

4) Enable interrupts if desired	SPIE and SPTIE in SPIC1
	MODFEN in SPIC2
		No interrupts enabled:
		SPIE = 0; SPTIE = 0
		MODFEN = 0

5) Select optional modes if desired	LSBFE in SPIC1
	BIDIROE and SPC0 in SPIC2
		Most Significant Bit first
		LSBFE = 0
		Full-duplex mode
		BIDIROE = 0

6) Enable master SPIs	SPE in SPIC1	SPE = 1
7) Enable slave SPIs	SPE in SPIC1	SPE =1

Next, let's review the steps to configure the SPI for data transmission using the SPIC1 and SPIC2 control registers and the SPIBR baud rate register.

First, select the SPI clock frequency by choosing a prescaler divisor and a baud rate divisor with the bit fields SPPR2:SPPR1:SPPR0 and SPR2:SPR1:SPR0 respectively.

Second, configure the clock with the CPOL and CPHA.

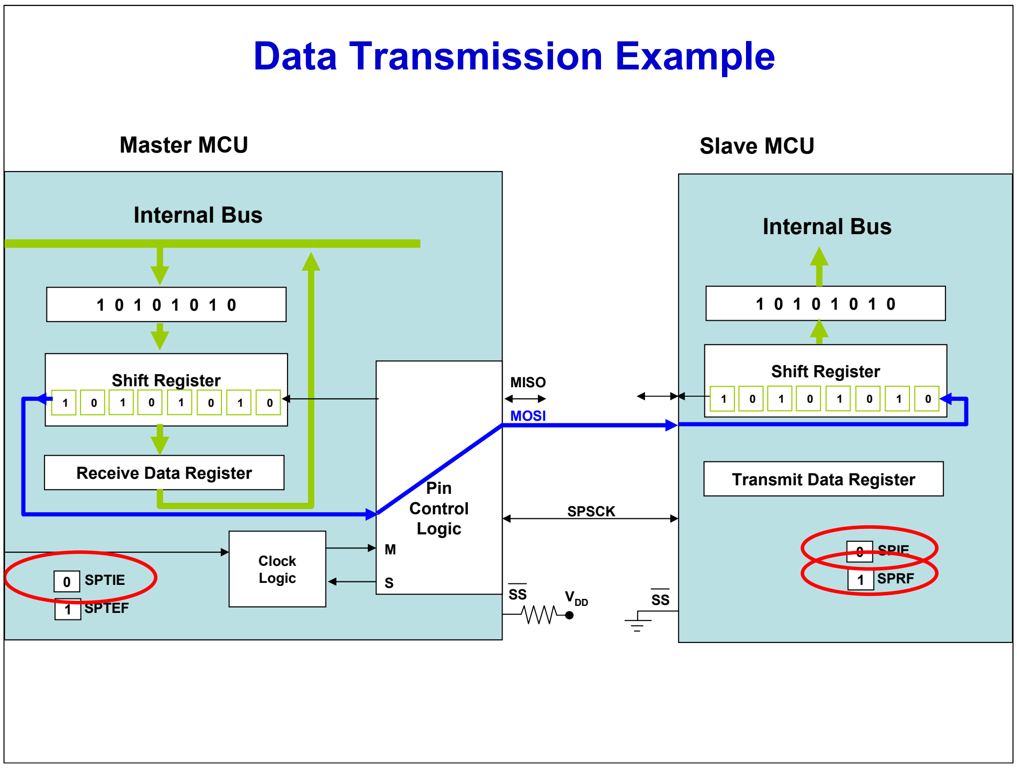
Third, for each SPI module in the system, select the mode using the MSTR. For the master SPI, set the bit to 1, and for slave SPIs, set the bit to 0.

Fourth, for an interrupt driven operation, you will need to enable interrupts using the SPTIE and the SPIE. This allows interrupts due to the SPI receive buffer being full or a mode fault event. MODFEN must be set high and SSOE set low to be able to detect a mode fault event.

Fifth, optional modes of operation should also be selected, if desired. The modes include the Least Significant Bit First, set by the LSBFE bit, and the bidirectional modes of operation, set by the BIDIROE and SPC0 bits.

Sixth, enable the master SPI system using SPE in the control register. Finally, make sure that the master SPI is enabled before the slave SPIs.

Take a moment to examine the example of initializing the SPIs.



Now that we've configured the SPI, let's look at an example of master-slave data transmission.

The SPTEF should be set to 1 showing that the transmit buffer is empty. The software then begins an SPI transmission by writing a byte to the SPI transmit data register.

If the shift register is empty, the byte is immediately transferred to the shift register. Otherwise, the transfer begins when the shift register finishes transferring the previous byte.

Once the byte is transferred from the SPI transmit register to the shift register, the SPTEF bit is set, indicating that another byte can be written to the SPI transmit data register. The SPTEF bit generates a CPU interrupt if the SPTIE bit is set to 1. This interrupt allows interrupt driven transmissions of multi-byte transfers. In this example, the SPTIE bit is set to 0 so that interrupts are disabled.

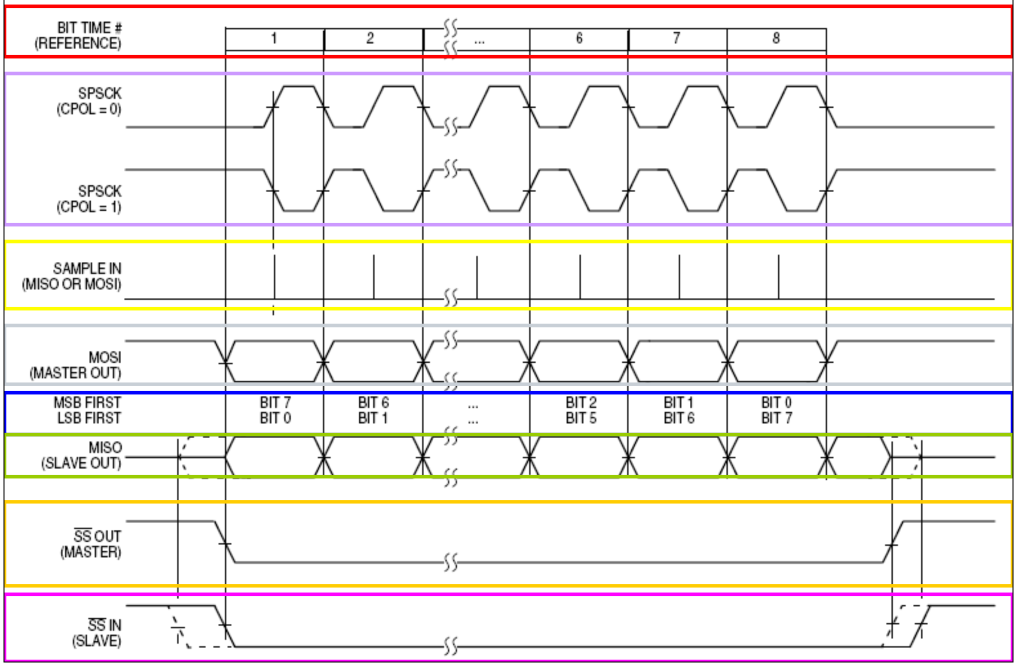
The byte begins shifting out a bit at a time on the MOSI pin that is synchronized with the master clock signal, SPSCK. The transfer will continue for 8 SPSCK clock cycles, transferring all 8-bits. As the byte shifts out from the master, another byte shifts in from the slave on the MISO pin. In this example, we will disregard the byte of information being shifted from the slave SPI to the master SPI.

The transmission ends when the whole byte is shifted out of the master SPI shift register and into the slave SPI shift register. The slave shift register is then automatically transferred to the slave SPI receive data register, if it is empty, allowing another byte to transfer, if one is pending.

The SPRF bit is set indicating that the SPI receive register is full and waiting to be read. If the SPIE bit is set to 1, an SPI receive interrupt request is also generated. In this example, the SPIE bit is set to 0.

To avoid overflow, the slave must read the receive data register prior to the master attempting to transfer more than one additional byte. For example, if the master has successfully transmitted a byte that is waiting to be read in the receive data register, the master may transfer an additional byte. This additional byte will be held in the shift register and not loaded into the receive register, since the previous byte has not been read by the CPU. If the master attempts to serially transmit another byte prior to reading the receive data register, the data in the slave shift register will be overwritten causing an overflow error condition. The byte in the receive data register is not overwritten.

Transmission Formats (CPHA = 0)



Software can select any of four combinations of the SPI serial clock phase and polarity using the CPOL and CPHA bits. The clock polarity selects an active high or low clock and has no significant effect on the transmission format. The clock phase selects one of two fundamentally different transmission formats. Let's examine SPI transmission for each clock phase beginning with the CPHA bit set to 0.

Here, you can see the eight bit times. Bit 1 starts as the slave is selected (the SS IN goes low), and bit 8 ends at the last SPSCK edge.

The MSB first and LSB first lines show the order of SPI data bits, depending on the setting in LSBFE. Both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL.

The SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The SS OUT waveform applies to the slave select output from a master provided MODFEN and SSOE = 1.

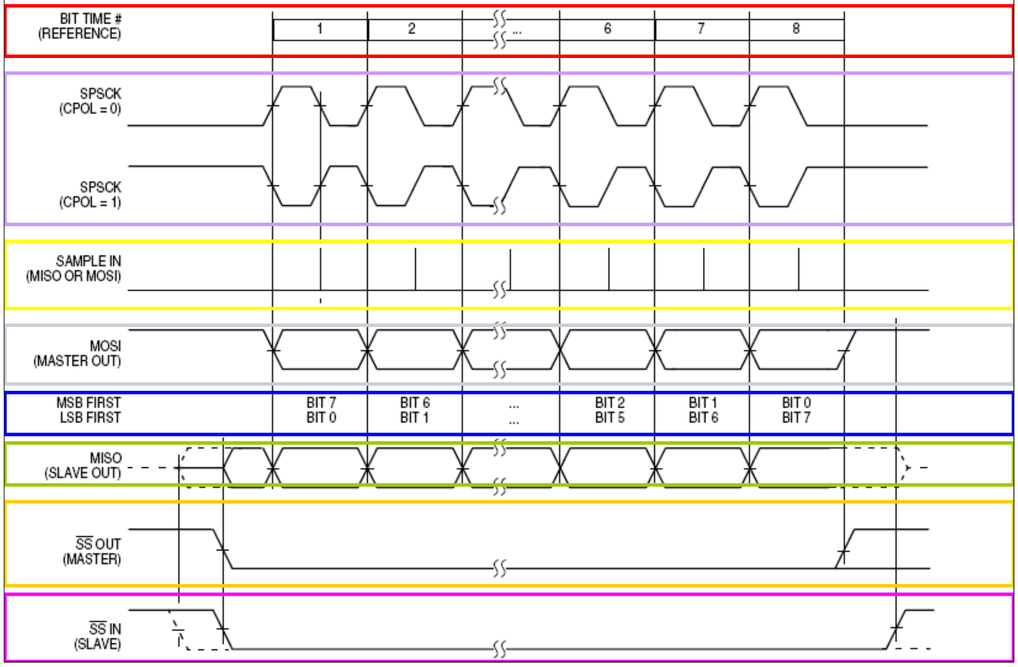
The master SS output goes to active low at the start of the first bit time of the transfer. It goes back to high one-half SPSCK cycle after the end of the eighth bit time of the transfer. The SS IN waveform applies to the slave select input of a slave.

When CPHA = 0, the slave begins to drive its MISO pin with the first data bit value (MSB or LSB depending on the LSBFE) when SS goes to active low. The first SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs respectively.

At the second SPSCK edge, the SPI shifter shifts one bit position, which shifts in the bit value that was just sampled. It also shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave respectively. This cycle continues until all 8 bits of data are transferred.

When CPHA = 0, the slave's SS input must go to its inactive high level between transfers.

Transmission Formats (CPHA = 1)



Here, you can see the clock formats when the clock phase bit, CPHA is set to 1. When CPHA=1, the first edge on SPSCK occurs at the start of the first cycle of an 8-cycle data transfer.

Take a look at the eight bit times shown. Bit 1 starts at the first SPSCK edge and bit 8 ends one-half a SPSCK cycle after the sixteenth SPSCK edge.

As when CPHA=0, the MSB first and LSB first lines show the order of SPI data bits, depending on the setting in LSBFE. Also, both variations of SPSCK polarity are shown, but only one of these waveforms applies for a specific transfer, depending on the value in CPOL.

Again, same as with CPHA=0 transmission, when CPHA=1 the SAMPLE IN waveform applies to the MOSI input of a slave or the MISO input of a master. The MOSI waveform applies to the MOSI output pin from a master and the MISO waveform applies to the MISO output from a slave. The SS OUT waveform applies to the slave select output from a master provided MODFEN and SSOE = 1.

Here CPHA=1 and CPHA=0 differ. When CPHA=1, the master SS output goes to active low one-half SPSCK clock cycle before the start of the transfer and goes back high at the end of the eighth bit time of the transfer. The SS IN waveform applies to the slave select input of a slave.

When CPHA=1, the slave begins to drive its MISO output when SS goes to active low, but the data is not defined until the first SPSCK edge. The first SPSCK edge shifts the value of the first bit of data from the shifter onto the MOSI pin of the master and the MISO pin of the slave.

The next SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the third SPSCK edge, the SPI shifter shifts one bit position, which shifts in the bit value that was just sampled. It also shifts the second data bit value out the other end of the shifter to the MOSI and MISO pins of the master and slave, respectively. This cycle continues until all 8-bits have been transferred.

When CPHA = 1, the slave's SS input is not required to go to its inactive high level between transfers.

Question

In SPI master-slave connections, the SPI optionally shares four port pins. The function of these pins depends on the settings in which registers? Select the correct answer and then click Done.

a. SPIC1 and SPIC2

b. SPIS and SPID

c. SPIBR and SPIC2

d. SPIC1 and SPIS

Let's review master-slave connections.

Correct.

The function of these four port pins depends on the settings in the two SPI control registers, SPIC1 and SPIC2. Click the forward arrow to continue on to the next page.

Question

Consider the following situation: the slave SPID register contains a byte of data previously transmitted but not yet read by the slave CPU. Keeping this information in mind, do you know what happens when the master SPI attempts to send another byte to the slave SPI? Select the correct answer and then click Done.

a. No transmission occurs.

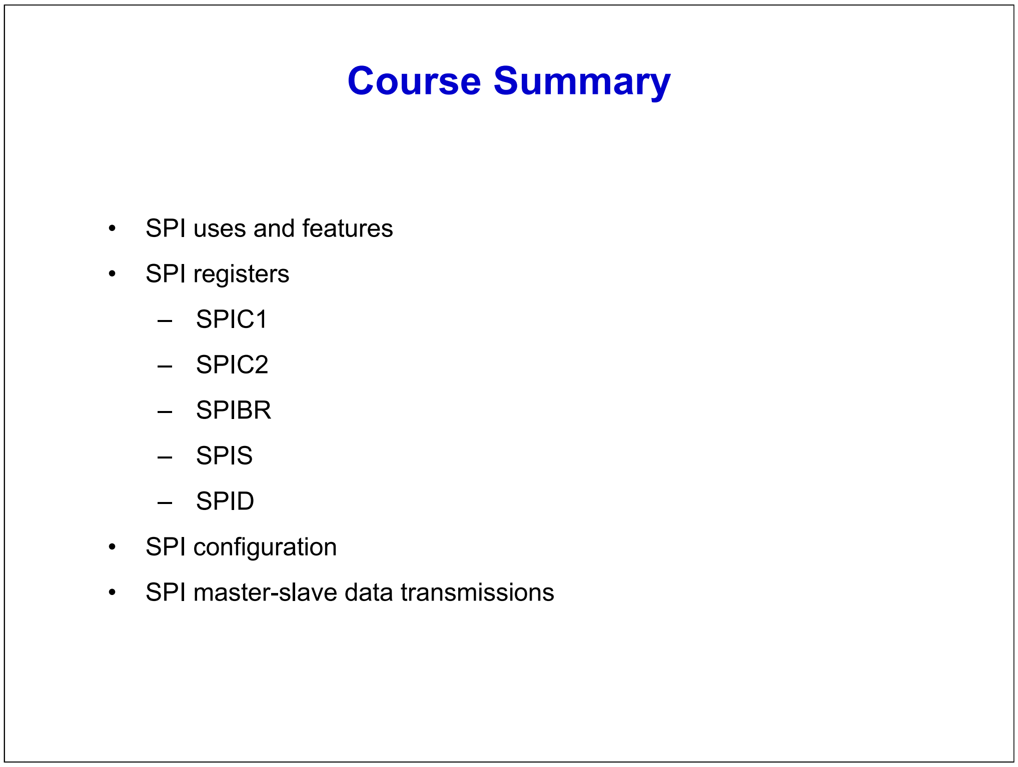
b. All 8-bits are transferred from the master’s shift register to the slave’s shift register while data in the slave’s receive data register is overwritten.

c. All 8-bits are transferred from the master’s shift register to the slave’s shift register while data in the slave’s receive data register is not overwritten until the CPU first reads the slave’s receive data register.

Consider this question about the master-slave data transmission process.

Correct.

All 8-bits are transferred from the master’s shift register to the slave’s shift register, while data in the slave’s receive data register is not overwritten until the CPU first reads the slave’s receive data register. This double buffering allows one extra byte to be transmitted prior to reading the previous transmission with no loss of data. You need to make sure that the data is read from the data register before a third byte is transmitted. This avoids an overflow error condition. In that situation, the data in the slave shift data register will be overwritten, but the data in the slave received data buffer will not. Click the forward arrow to continue on to the next page.



Course Summary

- SPI uses and features
- SPI registers
  - SPIC1
  - SPIC2
  - SPIBR
  - SPIS
  - SPID
- SPI configuration
- SPI master-slave data transmissions

In this course, you were presented with the uses and features of the HCS08 SPI module as well as how to configure and use the SPI module. The SPI module is particularly useful for high resolution ADCs, DACs, serial EEPROM, and other MCUs. You also examined the five SPI modules: SPIC1, SPIC2, SPIBR, SPIS, and SPID. These registers are used to select SPI options, control baud rate, report SPI status, and to hold transmit/receive data. Finally, you learned how to configure the SPI module for Master and Slave mode as well as how select the various modes of operation and transmission formats.