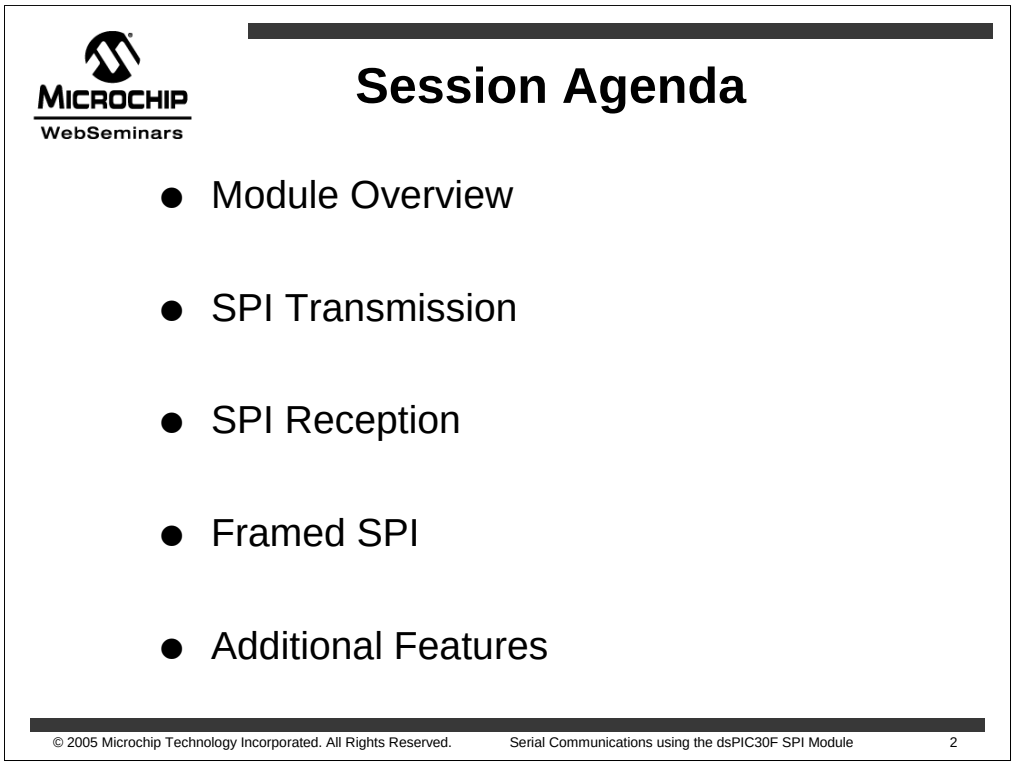
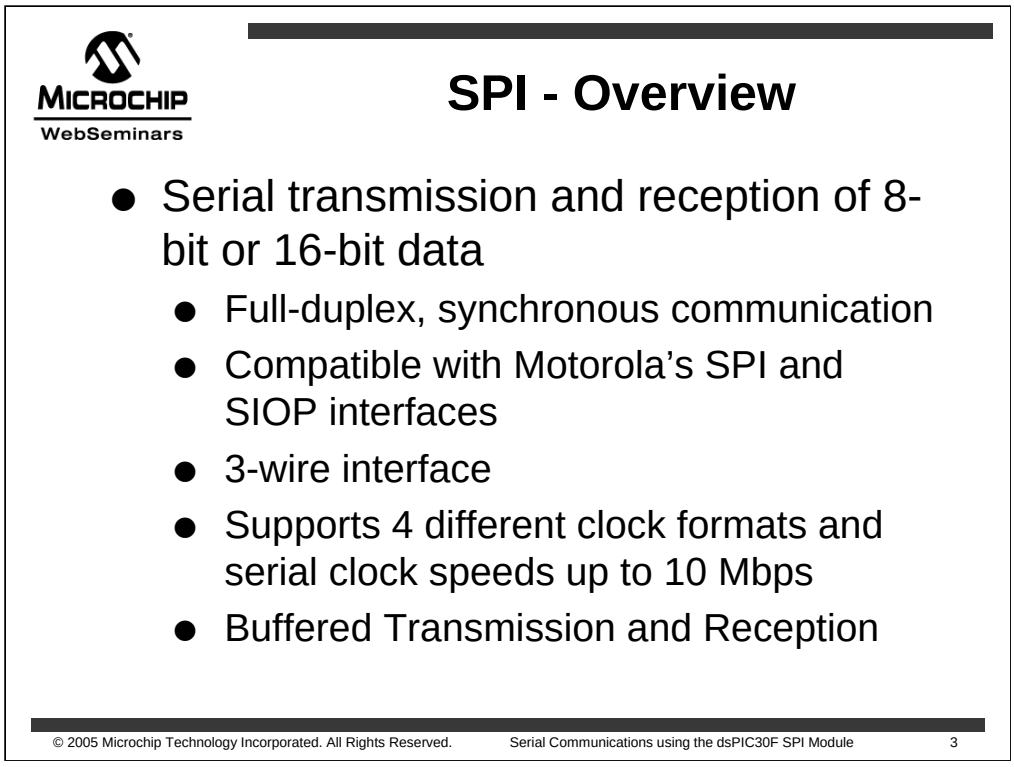
Hello and welcome to the “Serial Communications using the dsPIC30F SPI Module” web seminar. We’re glad that you’re here to learn more about Microchip’s family of Digital Signal Controllers.



My name is Ralph Fulchiero and I'm a Principle Applications Engineer with Microchip Technology. In today's session we're going to learn about the Serial Peripheral Interface, commonly known as the "SPI" module.

We will start by outlining some key features of the SPI in the dsPIC30F family of devices. We will gain an understanding of data transmission and reception through the SPI. This will be followed by a discussion of the special Framed SPI mode of operation. Lastly, we will study some advanced features of the SPI module.



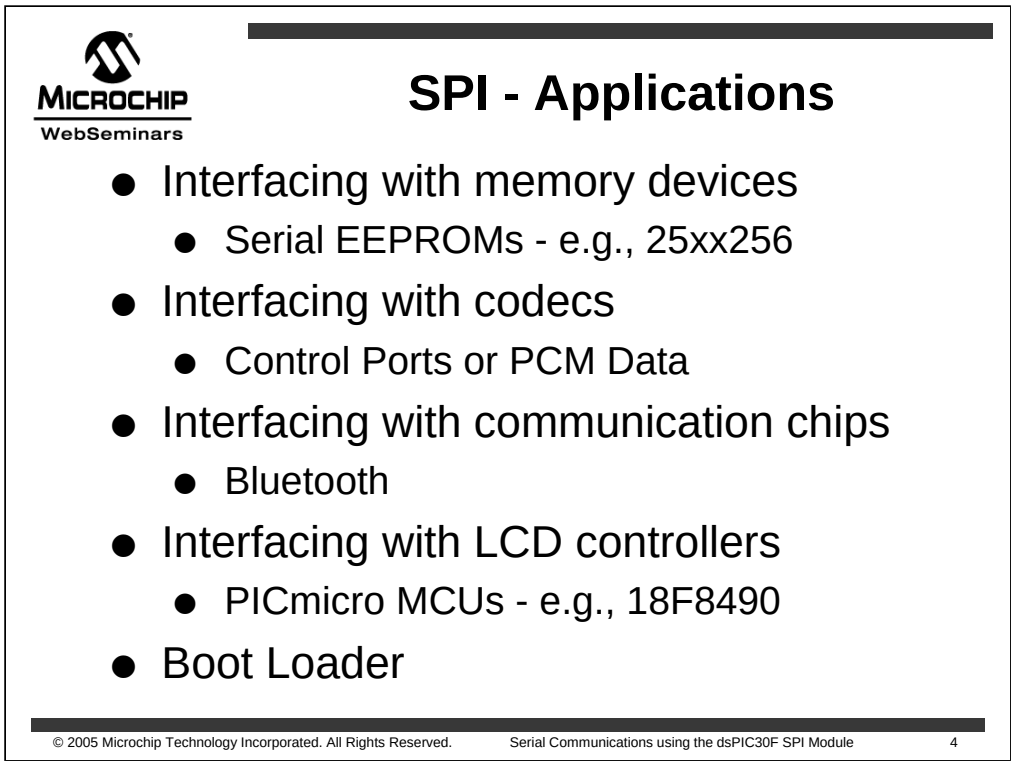
The SPI module is used for transmission and reception of bytes or words over a 3-wire interface consisting of a serial clock line, SCK, a serial data input line, SDI, and a serial data output line, SDO. Many dsPIC30F devices contain two SPI modules and therefore have two sets of SCK, SDI and SDO pins.

SPI communication is full-duplex, which means that a transmission and a reception proceed simultaneously. It is also an example of Synchronous communication, as both communicating entities use the same communication clock signal.

The SPI module may be used for communicating with external peripherals such as Analog-to-Digital or Digital-to-Analog Converters, display drivers, external memory devices such as Serial EEPROMs, or other microcontrollers. It supports all clock and data formats specified by Motorola's SPI and SIOP standard interfaces.

Serial data transfer speeds of up to 10 Mbps are supported by the SPI module. Transmissions and receptions are independently buffered, enabling back-to-back data transfers.

The SPI module in dsPIC30F devices also supports additional features such as Frame Synchronization, Slave Select, and device wake-up from SLEEP.



The SPI module can be utilized to interface and communicate with a wide variety of external peripheral devices.

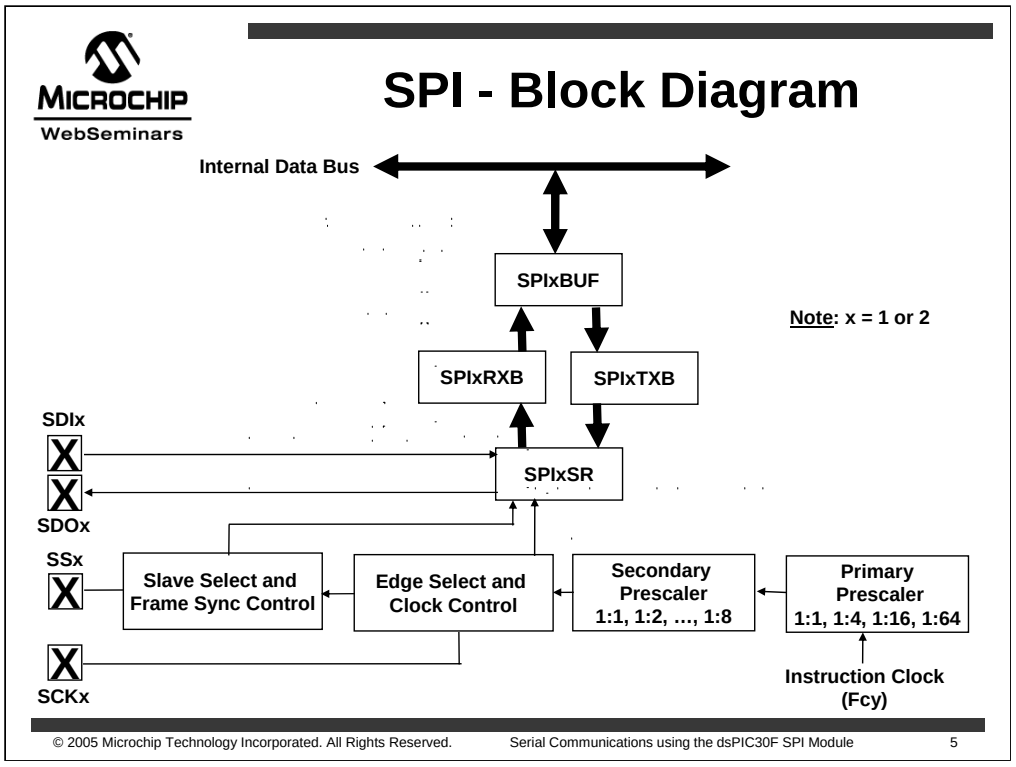
For example, it can be used to program and read external Serial EEPROM devices, such as the 25/256 family of devices from Microchip.

It can also be used to interface with some low-cost speech codec devices for sending speech samples, or as a control channel for more advanced audio codecs. The Framed SPI mode described later is an ideal configuration for this purpose.

Another popular use of the SPI module is to exchange Pulse-Code-Modulated data with dedicated communication processors running a Bluetooth protocol stack for wireless communications.

SPI can also be used to send commands and characters to LCD Controllers, such as the PIC18F8490.

Last but not the least, SPI is a very efficient communication channel for a Boot Loader program running on the dsPIC, to exchange command and response messages with an external Host device.

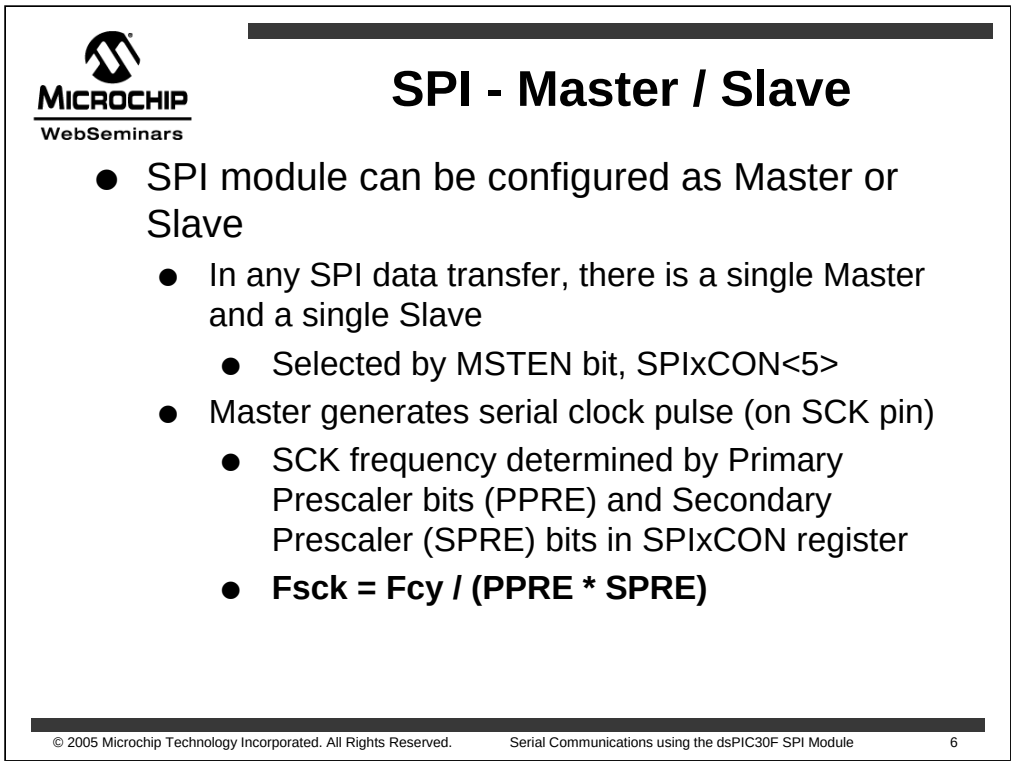


This is a simplified Block Diagram of the SPI module in dsPIC30F devices. Note that the basic communications interface consists of only 3 pins: SPI Serial Data Input, SPI Serial Data Output and SPI Serial Clock.

As shown in the figure, the transmitted and received data are buffered. A single SPI Shift Register is used to shift out data through the SDO pin and also shift in data through the SDI pin. Data written to the SPI Transmit Buffer is transferred to the SPI Shift Register, and data received in the SPI Shift Register is transferred to the SPI Receive Buffer. Keep in mind that the Transmit and Receive Buffers are mapped to the same user-accessible register called SPI Buffer.

A fourth pin, Slave Select, is used to support the Slave Select functionality as well as Framed SPI communications.

Depending on which dsPIC you are using, you will either have 1 or 2 SPIs to work with.

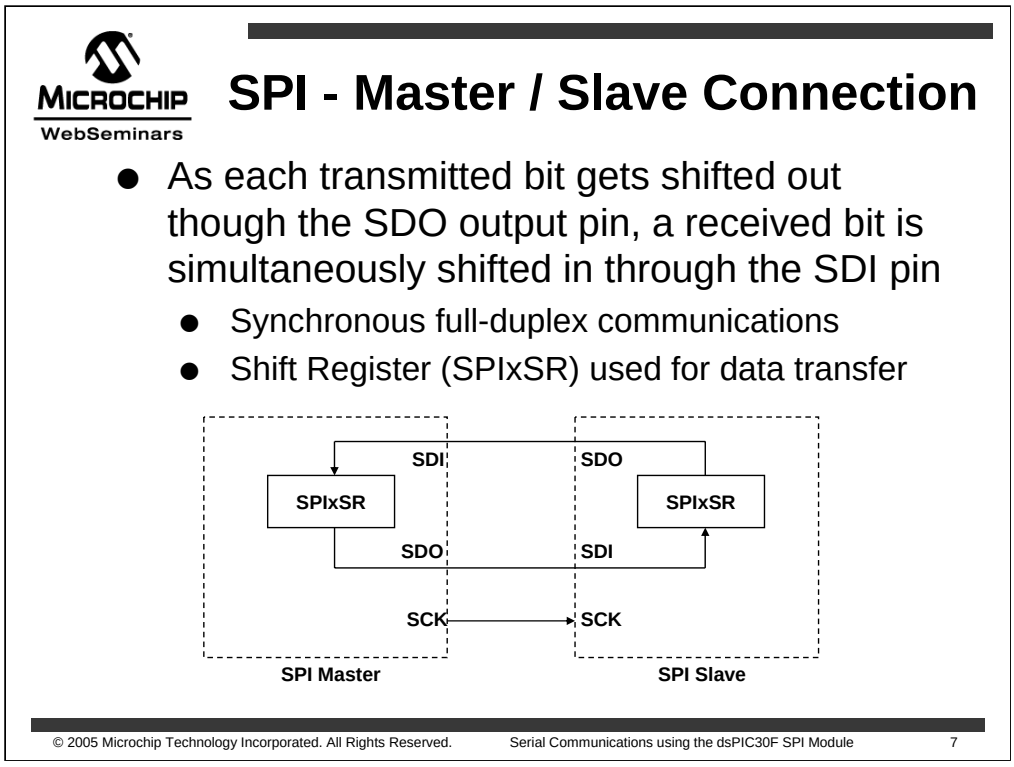


For an SPI data transfer to take place between two devices, one device must be configured as an SPI Master and the other as an SPI Slave. Master mode is enabled by setting the Master Enable bit in the SPI Control Register.

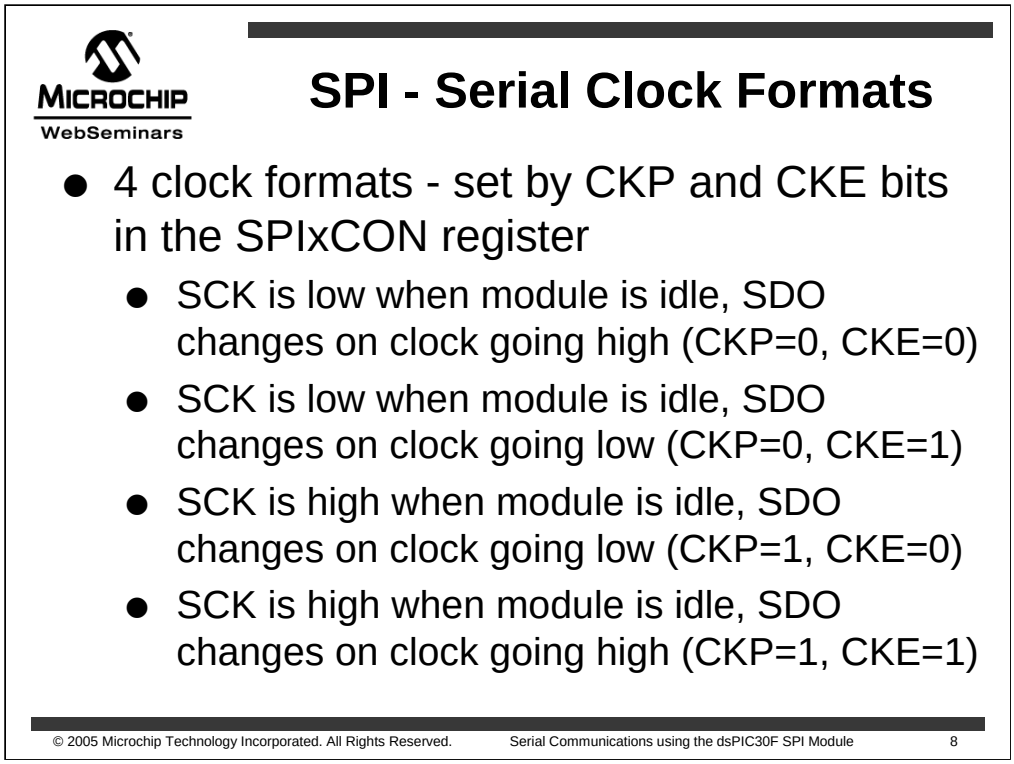
There are two main differences between Master and Slave functionality.

First, the Master generates the SPI Serial Clock which controls the rate at which data bits get transferred. The Slave receives this clock through its SPI Serial Clock pin. The Serial Clock frequency is derived from the device instruction frequency, Fcy, by dividing it by the user-programmable Primary and Secondary Prescaler values. Thus, Fcy can be divided by 1 to obtain the maximum possible Baud Rate for a given instruction frequency, and can even be divided by 512 to obtain the minimum possible Baud Rate.

Second, it is the Master that initiates a data transfer, as we will see shortly.



Two devices communicating through SPI are inter-connected with their SCK pins tied together, the SDI pin of the Master tied to the SDO pin of the Slave, and the SDO pin of the Master connected to the SDI pin of the Slave. The data transfer occurs simultaneously in both directions through the two Shift Registers.



The SPI module supports 4 different Serial Clock formats. The user can select one of these 4 formats by configuring the Clock Polarity Select, or CKP, and Clock Edge Select, or CKE, bits in the SPI Control Register.

The CKP bit determines whether the Serial Clock is at a high logic level or low logic level when the SPI module is in an idle state.

If CKP is set, then SCK is high when idle, which means the SCK signal is interpreted as ‘active-low’.

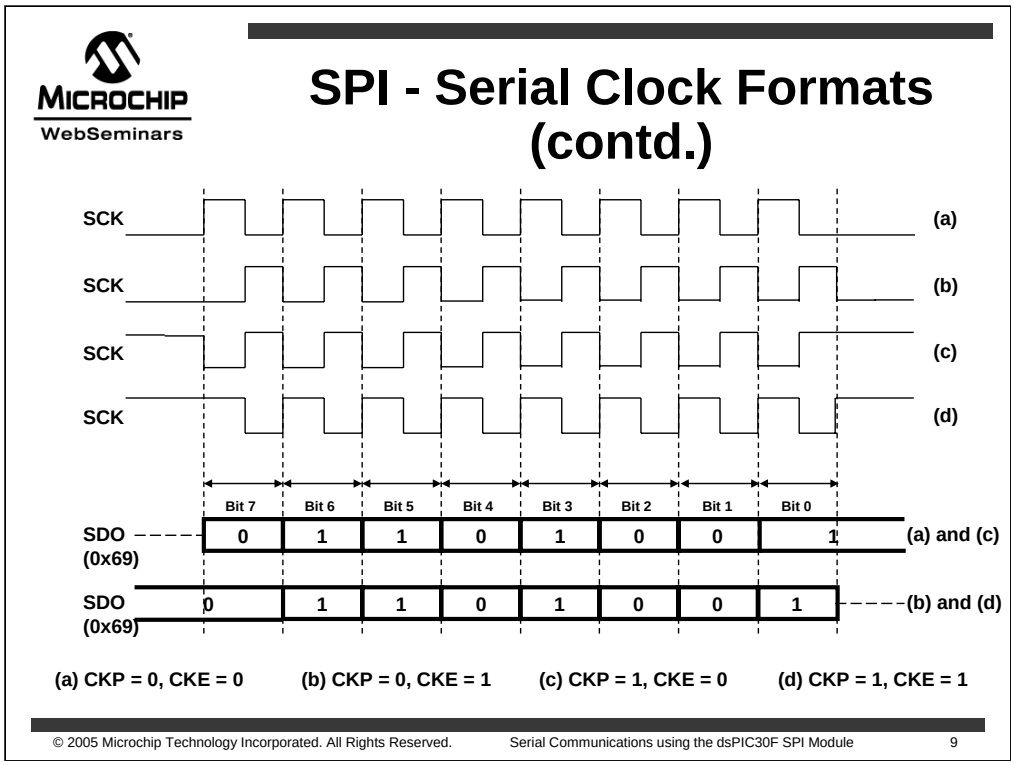
If CKP is cleared, then SCK is high when active, which means the SCK signal is interpreted as ‘active-high’.

The CKE bit determines whether the Serial Data Output changes its state on an idle-to-active transition of the Serial Clock or on an active-to-idle transition of the Serial Clock.

If CKE is set, SDO changes on an active-to-idle SCK transition.

If CKE is cleared, SDO changes on an idle-to-active SCK transition.

And… on the next slide we’ll see what these different clocking schemes look like.

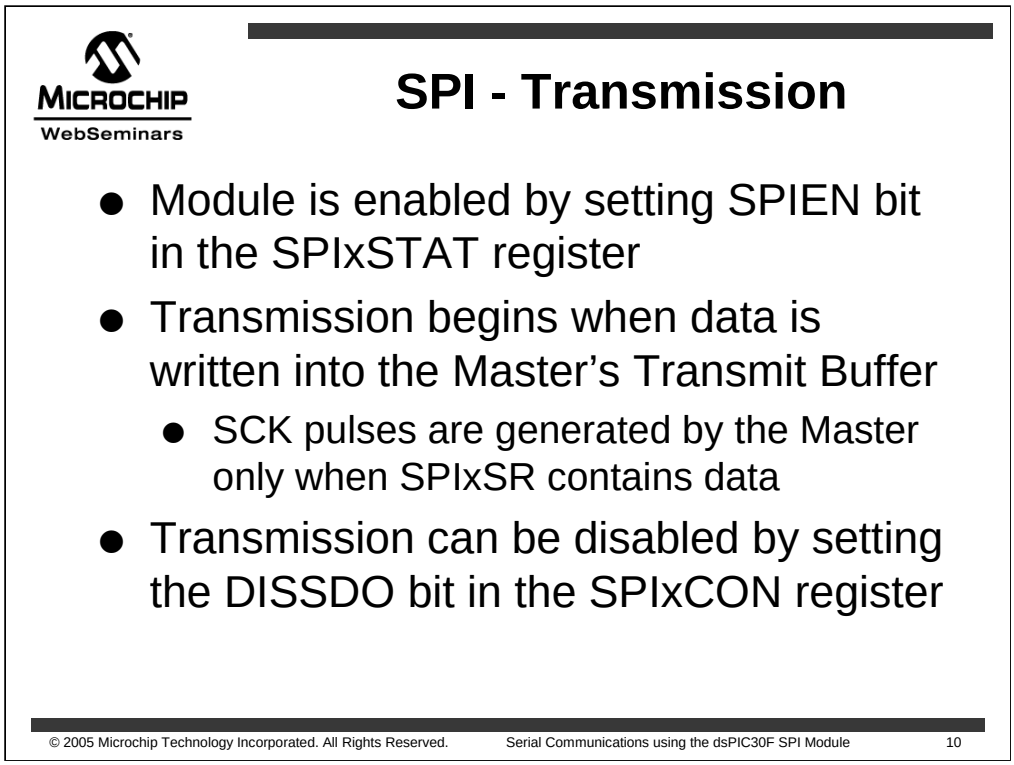


As you can see in this timing diagram, when the CKP and CKE bits are used in concert, one can choose between four different Serial Clock formats, depending on the application requirements.

Here, the data value hex “6” “9” is being transmitted. For the 4 different clock settings, you can see how the data changes in relation to the SCK clock line.

For instance, in the top diagram labeled “a”, the polarity and edge control bits are both low.

This means the clock is low for the idle state and the output changes on the low to high clock transition.

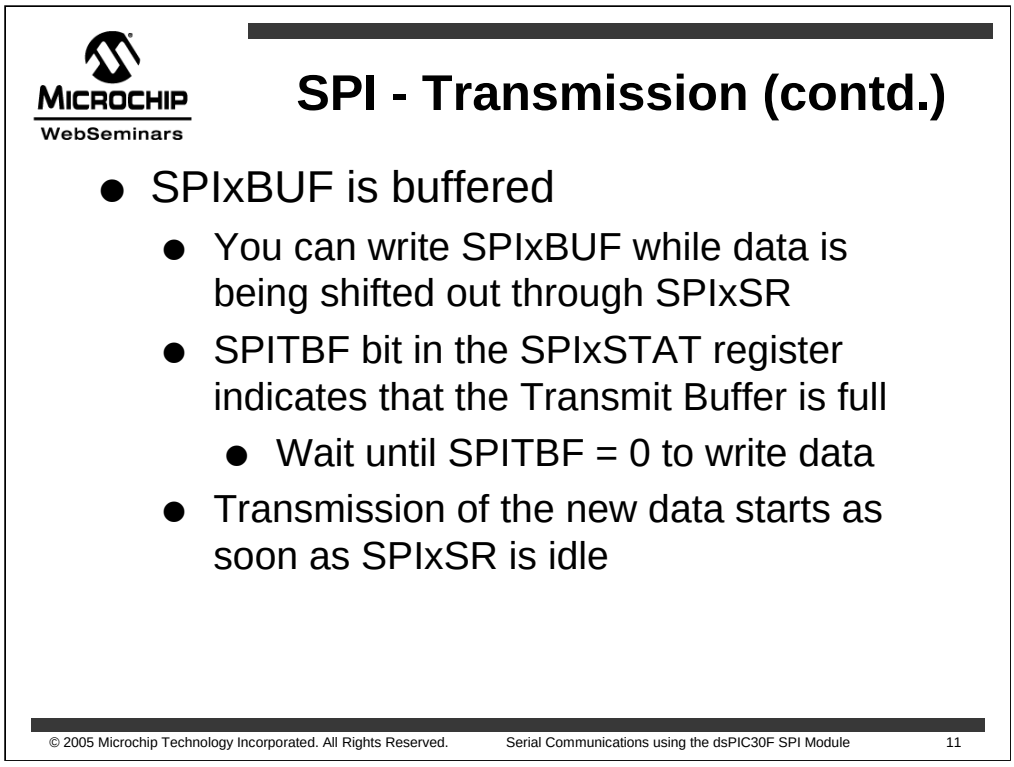


Before performing any transmission or reception, the SPI module must be enabled by setting the SPI Enable bit in the SPI Status Register.

As we discussed earlier, data transmission is always initiated by the SPI Master, by writing data to the SPI Buffer. This implies that for full-duplex operation, the Slave should write to its SPI Buffer before the Master writes to its SPI Buffer, so that both Master and Slave buffers contain valid data when data transfer begins.

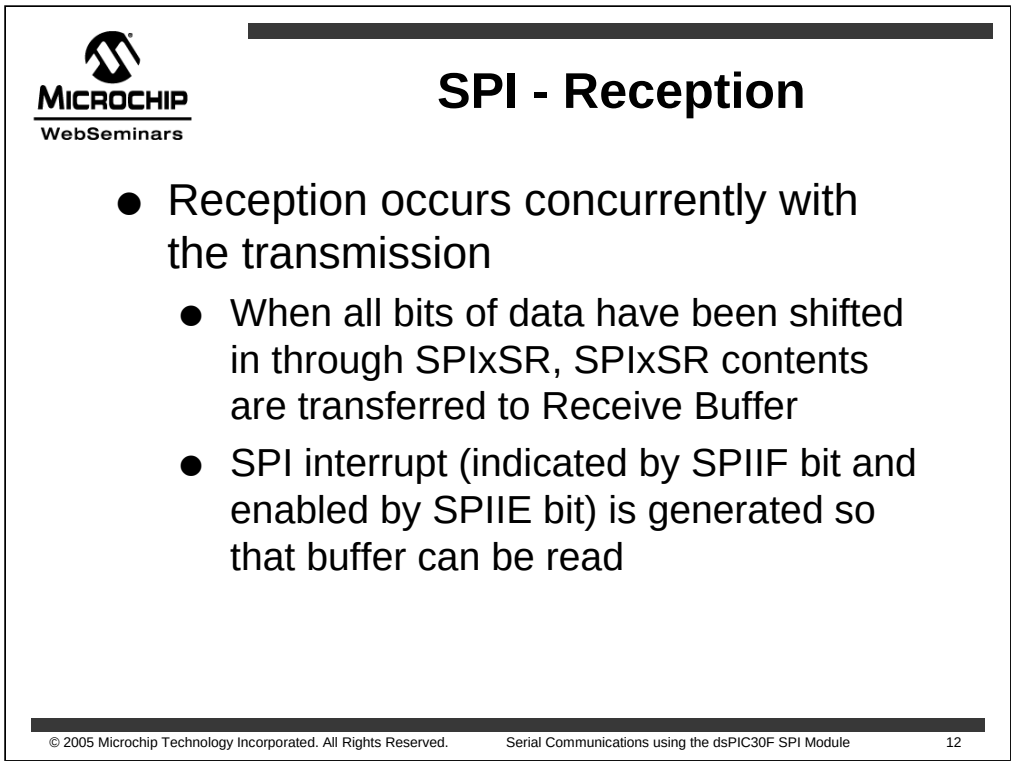
If there is no data transfer in progress and a new byte of data is written, the data written to the SPI Buffer is immediately transferred to the Shift Register, and the Master starts generating Serial Clock pulses to shift out each bit of data through the SDO pin.

The user has the option of disabling the Serial Data Output line by setting the Disable Serial Data Output bit in the SPI Mode Register.



To facilitate using the SPI module, the SPI Buffer is buffered. This means that the Master can write to the SPI Buffer while the Shift Register is shifting data. New data written to the SPI Buffer is transferred to the Shift Register when the Shift Register becomes empty.

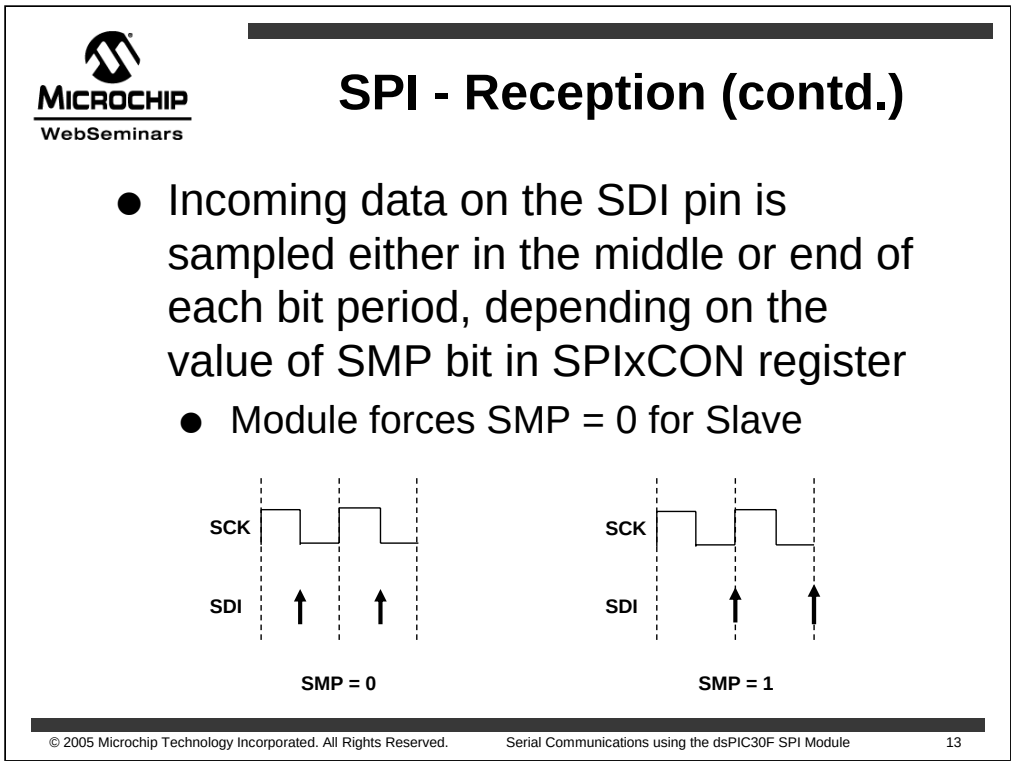
This buffering enables back-to-back data transmissions. The only precaution the user needs to take in order to prevent data corruption is to verify that the SPI Transmit Buffer Full bit in the SPI Status Register is clear before writing data to the SPI Buffer.



For both SPI Master and Slave, data is shifted in through the Serial Data Input line concurrently with the shifting out of data through the Serial Data Output line.

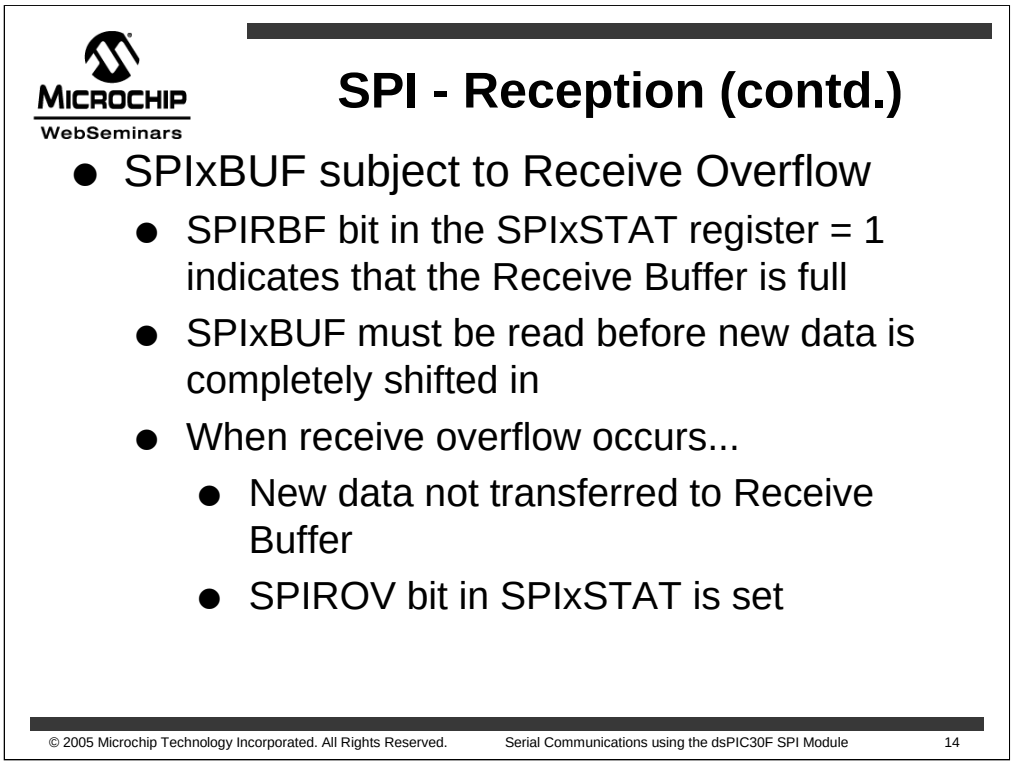
When all bits of the received data have been shifted in, the data is transferred from the Shift Register to the SPI Buffer, and an SPI Interrupt is generated. The interrupt is serviced if it is enabled by setting the SPI Interrupt enable bit and the SPI interrupt priority is higher than the current CPU interrupt priority.

After the SPI interrupt flag is set, the contents of the SPI Buffer can be read by your software.



The SPI Data Input Sample Phase, or SMP, bit in the SPI Control Register determines if the Serial Data Input line is sampled by the Master in the middle of the data output time for each bit or at the end of the data output time.

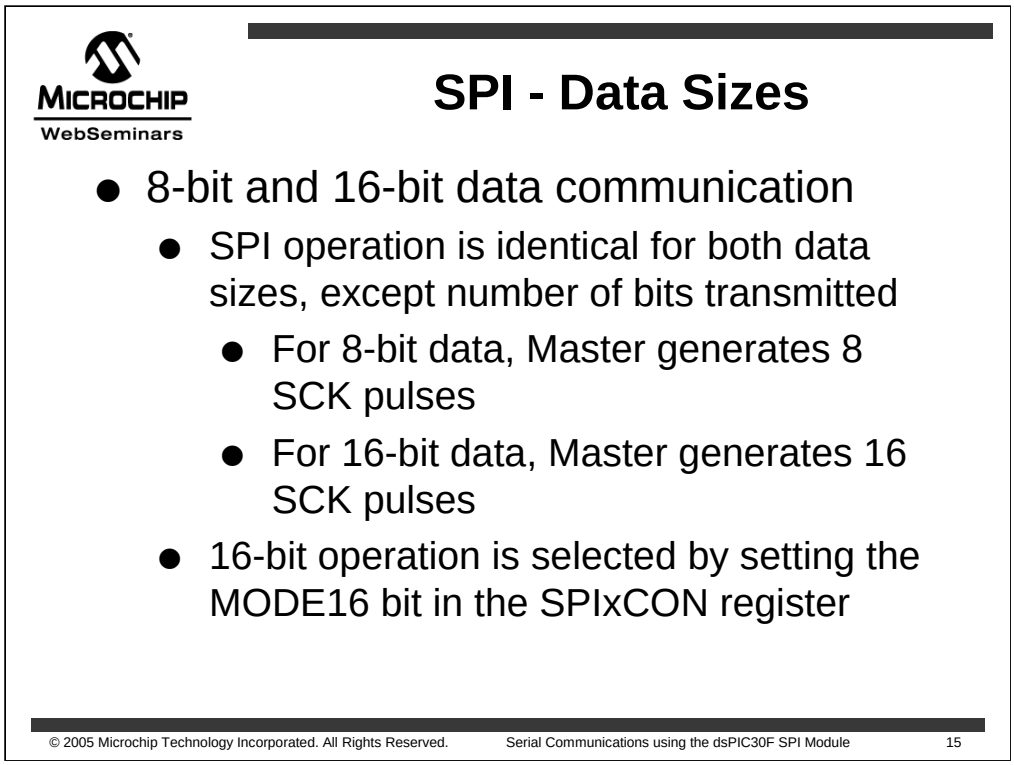
If the SMP bit is set, then the input sampling is done at the end of the bit output time, whereas if the SMP bit is cleared, then the input sampling is done at the middle of the bit output time. In the case of the SPI Slave, the hardware forces the SMP bit to zero, so the input data is always sampled at the middle of each bit output time.



Due to Receive Buffering, the SPI Buffer may contain unread received data while the Shift Register is shifting in new data. You must read the SPI buffer before all bits of the new data have been completely shifted in through the Shift Register.

Failure to do so will result in an SPI Receive Overflow error. When this occurs, the SPI Receive Overflow bit in the SPI Status Register is set, and the new data is discarded by the hardware.

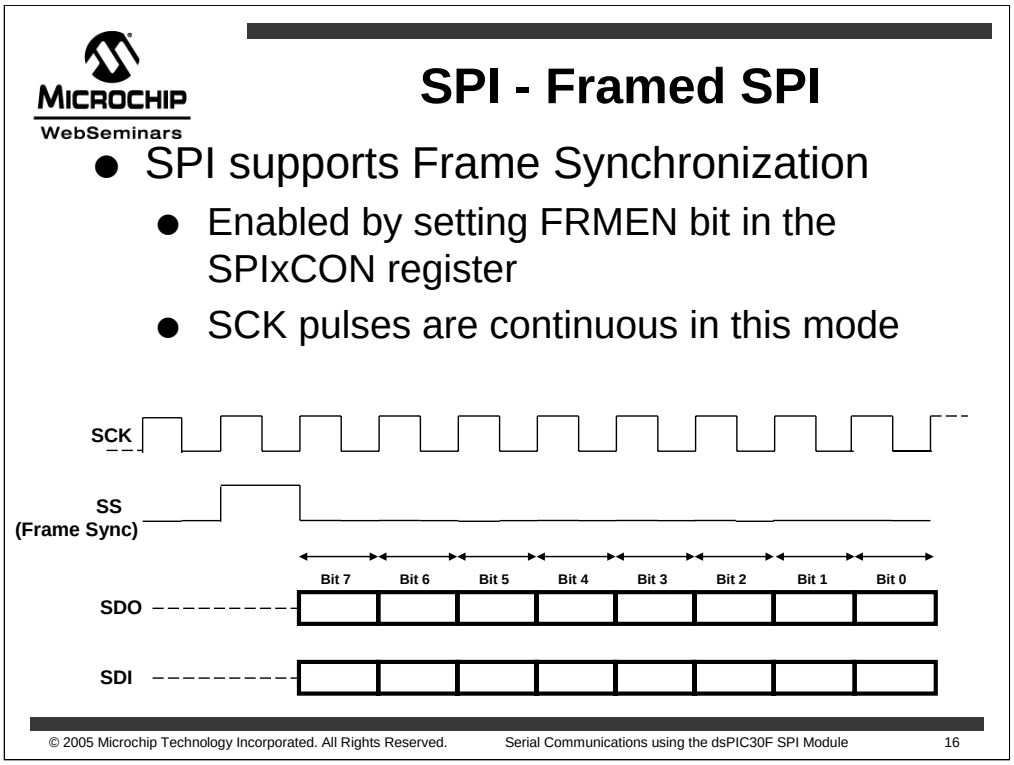
Your software can monitor the presence of new received data in the SPI Buffer by inspecting the SPI Receive Buffer Full bit in the SPI Status Register.



The SPI module supports transfer of both byte-sized and word-sized data.

The module functionality and communication sequence is the same for both data sizes; the only difference is the number of Serial Clock pulses generated by the Master, 8 in the case of bytes and 16 in the case of words.

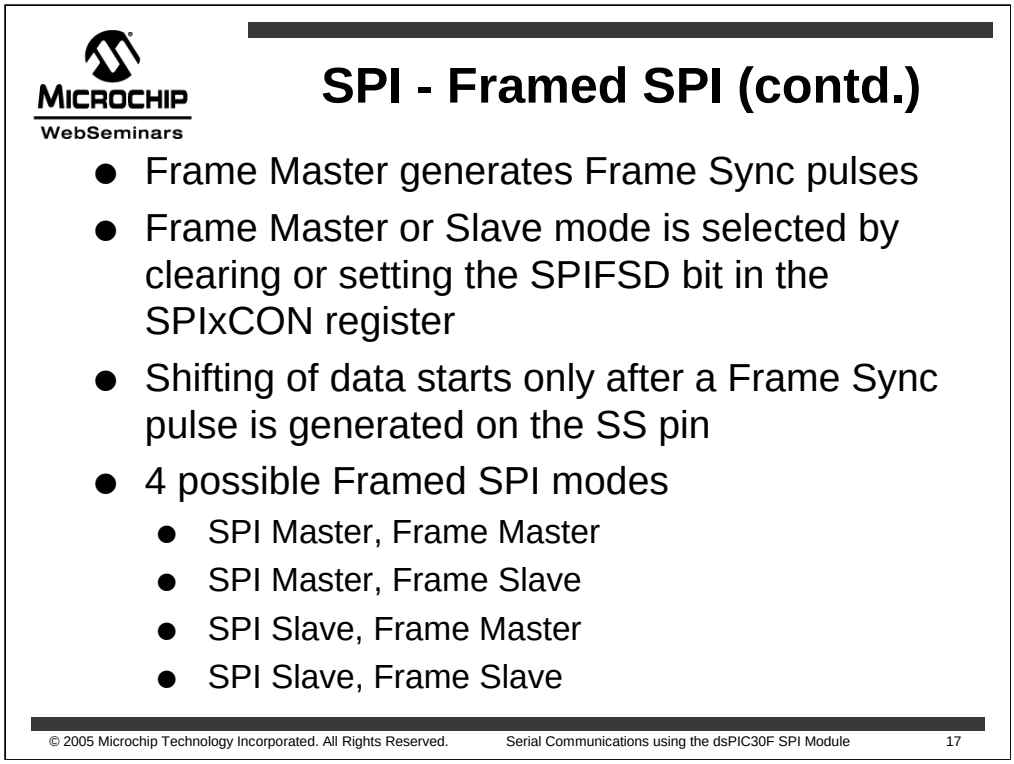
The 16-bit mode is enabled by setting the Word/Byte Communication Select bit, MODE16, in the SPI Control Register.



The SPI module supports a basic 4-wire Framed SPI protocol which enables efficient interfacing to certain codecs. The Framed SPI mode is enabled by setting the Framed SPI Enable bit in the SPI Control Register.

This timing diagram depicts the basic operation of the Framed SPI mode.

Observe that unlike in the normal SPI mode, in Framed SPI mode the Serial Clock is continuous rather than being generated only during the actual data transfer.



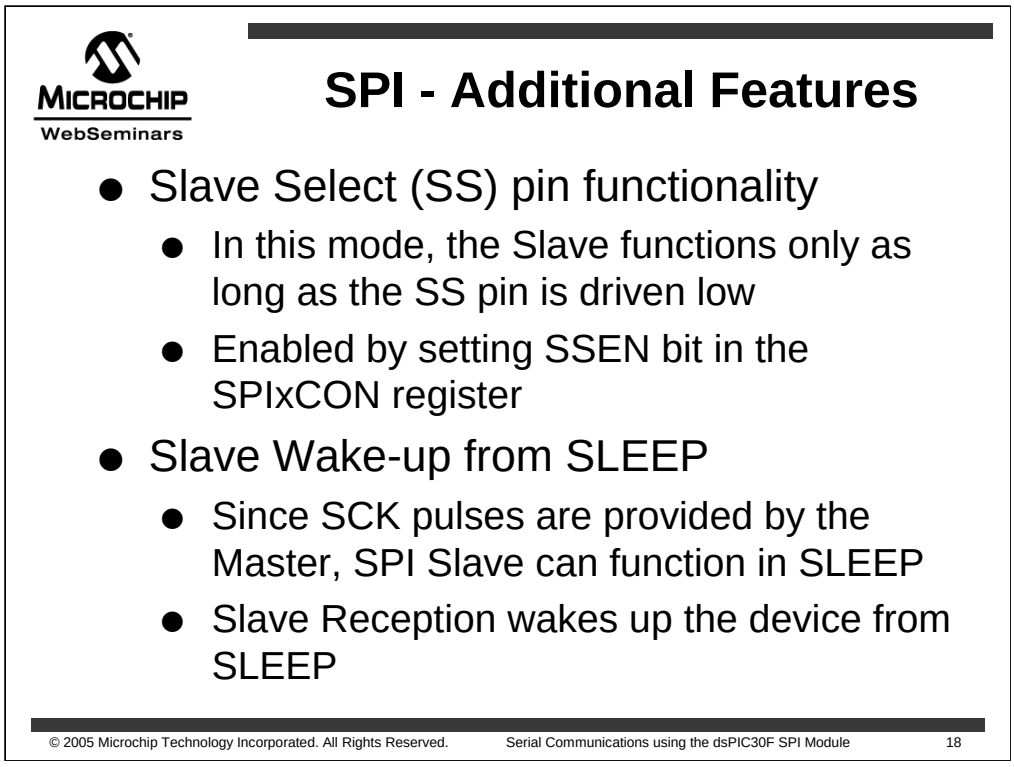
In the Framed SPI mode, a device can be a Frame Master, which implies that it generates the Frame Synchronization pulses on its Slave Select pin, or a Frame Slave, which implies that it uses Frame Synchronization pulses generated by a Frame Master.

The direction of the Frame Synchronization pulse is determined by the SPI Frame Sync Pulse Direction bit in the SPI Control Register.

Irrespective of which device is the SPI Master and which device is the SPI Slave, a Framed SPI data transfer begins as soon as the Frame Master device writes data to the SPI Buffer. Obviously, for full-duplex operation the Frame Slave device should write to its SPI Buffer before the Frame Master does, in order to ensure that the data is ready at both ends when data transfer begins.

A Framed SPI data transfer begins with the Frame Master generating a Frame Synchronization pulse of 1 bit-time duration. As soon as the Frame Synchronization pulse has been transmitted, the data bits start shifting out through the Shift Register.

Corresponding to various combinations of SPI Master Enable and Frame Sync Direction bit states, there are 4 possible configurations of a communicating device, as listed here. In each case, if one device is a Frame Master then the other must be a Frame Slave.



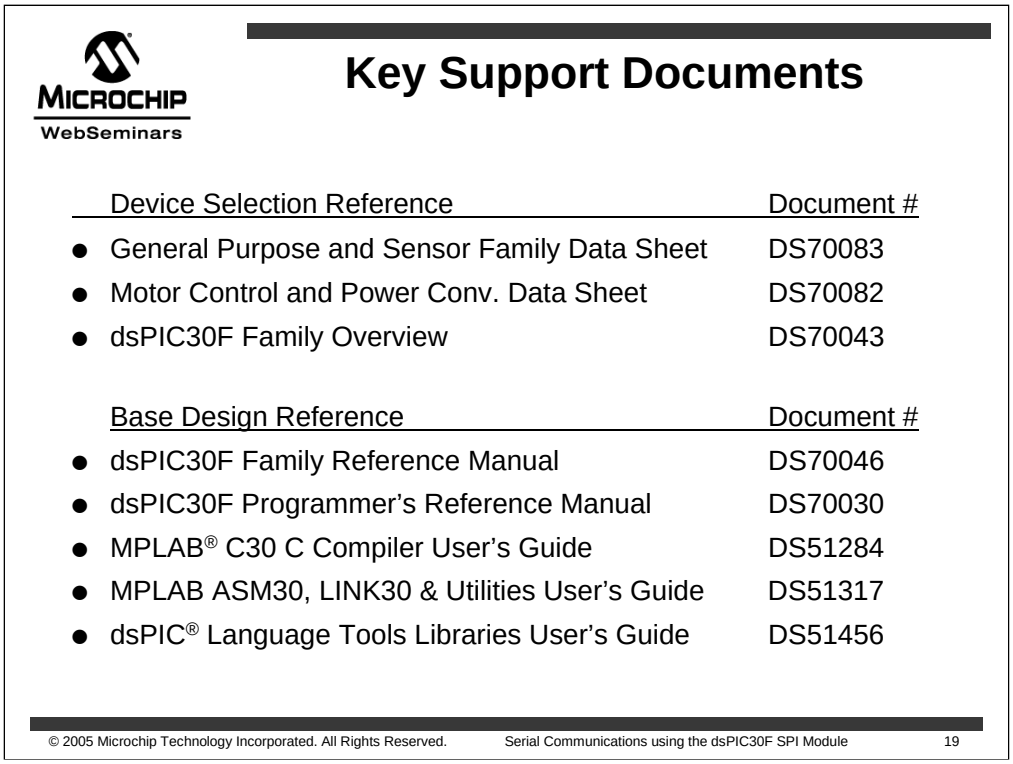
Let us round up our discussion of the SPI module with a couple of special SPI features.

The Slave Select pin can be used as a switch to enable or disable the function of an SPI Slave device. In this mode, the SPI Slave functionality is enabled only when the Slave Select pin goes to a low logic level. At other times, the Slave is effectively disabled and does not communicate. This feature helps to support multiple SPI slaves in a system. The Slave Select function is enabled by setting the Slave Select Enable bit in the SPI Control Register.

Note that the Slave Select and Frame Synchronization features must not be enabled simultaneously, as the same pin is used for both functions.

Also, if CKE is set, then the Slave Select functionality must be enabled and the Slave Select pin brought low.

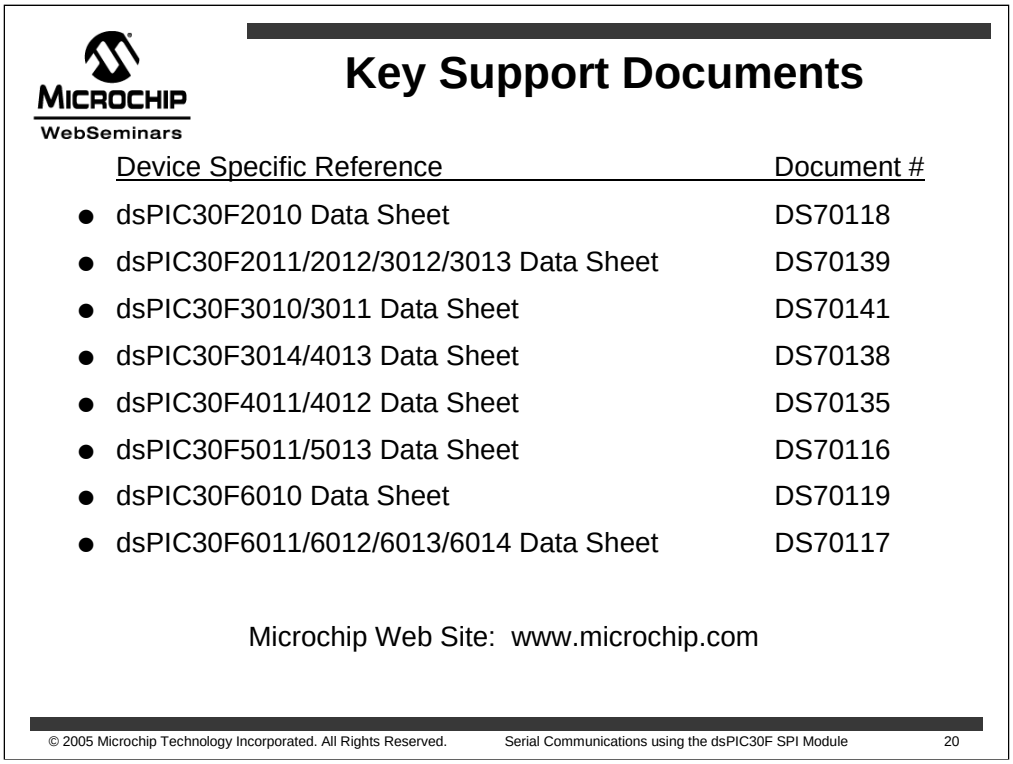
An SPI module configured as an SPI Slave can function even in SLEEP mode, because the Serial Clock signal is provided by an external SPI Master. As a result, reception of data by a Slave SPI can wake up the dsPIC30F device from SLEEP with no loss of data.



Well this wraps up today's session on the Serial Peripheral Interface. We hope you found the presentation useful and informative.

For more information about the dsPIC, here are references to some important documents that contain a wealth of information about the dsPIC30F family of devices.

The Family Reference Manual contains detailed information about the architecture and peripherals, whereas the Programmer's Reference Manual contains a thorough description of the instruction set.



For device-specific information such as pinout diagrams, packaging and electrical characteristics, the device datasheets listed here are the best source of information.

All these documents can be obtained from the Microchip web site shown, by clicking on the “dsPIC® Digital Signal Controllers” or “Technical Documentation” link.

Thanks again for attending today’s training session and your interest in the dsPIC30F family of Digital Signal Controllers.