Digilent Adept Serial Peripheral Interface (DSPI) Programmer's Reference Manual

Revision: July 16, 2012

www.digilentinc.com

1300 NE Henley Court, Suite 3
Pullman, WA 99163
(509) 334 6306 Voice | (509) 334 6300 Fax

Introduction

This document describes the programming interface to the Digilent Adept Serial Peripheral Interface (DSPI) subsystem for version 2 of the Digilent Adept software system. It describes the capabilities of the DSPI subsystem and the API functions used to access its features.

SPI is an industry standard synchronous serial communications protocol supported by many integrated circuit devices. It is made up of four signals: Slave Select (SS), Master Out/Slave In (MOSI), Master In/Slave Out (MISO), and Serial Clock (SCK). Some vendors use other signal names (e.g. CS, SDO, SDI, CLK). An SPI device can either be a master or a slave. The master generates the SS, MOSI, and SCK signals. The slave device generates the MISO signal. The SS signal is an active low enable used by the master to enable the slave. A Digilent Adept compatible SPI device is always an SPI master.

The master and slave devices each contain a shift register. The connections between the master and slave form a continuous shift register, such that when the byte in the master's shift register is shifted into the slave, the byte in the slave's shift register is simultaneously shifted into the master. In order for the master to read a byte from the slave, it must send a byte to the slave. The data can be shifted most-significant-bit (MSB) first or least-significant-bit (LSB) first. Some devices only support one shift direction but most support shifting in either direction. The master and the slave must agree on the shift direction.

Once a DSPI port has been enabled, it must be configured for SPI operating mode, shift direction, and clock frequency to values compatible with the slave device. The functions DspiSetMode and DspiSetSpeed are used for this purpose.

SPI Operating Mode

An SPI port can potentially operate in one of four modes. The SPI mode indicates on which clock edge the data will be sampled and at what level the clock signal will be when the port is idle. The following table describes the behavior of the four modes:

Mode	SCK Rising Edge	SCK Falling Edge	SCK Idle State
0	Sample data	Shift	Low
1	Shift	Sample data	Low
2	Shift	Sample data	High
3	Sample data	Shift	High

Port Properties

The port properties are used to indicate which optional parts of the DSPI interface are supported by a given port.

The following port properties are defined for this subsystem:

1. dprpSpiSetSpeed – indicates that the port supports setting SPI clock rate
2. dprpSpiShiftLeft – indicates that the port supports MSB first shift
3. dprpSpiShiftRight – indicates that the port supports LSB first shift
4. dprpSpiDelay – indicates that the port supports inter-byte delay
5. dprpSpiMode0 – indicates that the port supports SPI mode 0
6. dprpSpiMode1 – indicates that the port supports SPI mode 1
7. dprpSpiMode2 – indicates that the port supports SPI mode 2
8. dprpSpiMode3 – indicates that the port supports SPI mode 3
9. dprpSpiStartEndDelay – indicates that the port supports setting start and end delays

DSPI API Functions

The following API functions make up the SPI interface.

DspiGetVersion(char * szVersion)

Parameters

szVersion	- pointer to buffer to receive version string

This function returns a version number string identifying the version number of the DSPI DLL. The symbol cchVersionMax declared in dpcdecl.h defines the longest string that can be returned in *szVersion*.

DspiGetPortCount(HIF hif, INT32 * pcprt)

Parameters

hif	- open interface handle on the device
pcprt	- pointer to variable to receive count of ports

This function returns the number of DSPI ports supported by the device specified by *hif*.

DspiGetPortProperties(HIF hif, INT32 prtReq, DWORD * pdprp)

Parameters

hif	- open interface handle on the device
prtReq	- port number to query
pdprp	- pointer to variable to return port property bits

This function returns the port properties for the specified DSPI port. The port properties indicate the specific features of the DSPI specification implemented by the specified port.

DspiEnable(HIF hif)

Parameters

hif	- open interface handle on the device

This function is used to enable the default SPI port (port 0) on the specified device. This function must be called before any functions that operate on the SPI port may be called for the specified device.

DspiEnableEx(HIF hif, INT32 prtReq)

Parameters

hif	- open interface handle on the device
prtReq	- SPI port number

This function is used to enable a specific port on devices that support multiple SPI ports. This function must be called before any functions that operate on the SPI port may be called. The *prtReq* parameter specifies the port number of the SPI port to enable.

DspiDisable(HIF hif)

Parameters

hif	- open interface handle on the device

This function is used to disable and end access to the currently enabled SPI port on the specified interface handle.

DspiSetSelect(HIF hif, BOOL fSel)

Parameters

hif	- open interface handle on the device
fSel	- state to set Slave Select signal. FALSE = active, TRUE = inactive

This function sets the Slave Select signal (SS) on an enabled SPI port. If the fSel parameter is set to FALSE, the signal is set to the logic 0 state and the SPI port is activated. If the fSel parameter is set to TRUE, the signal is set to the logic 1 state and the SPI port is deactivated.

DspiSetSpiMode(HIF hif, DWORD idMod, BOOL fShRight)

Parameters

hif	- open interface handle on the device
idMod	- sets SPI mode (must be a value between 0 and 3
fShRight	- sets direction of bits are shifted in/out. TRUE = Right, FALSE = Left.

This function sets the mode and shift direction of an enabled SPI port. The mode is a value in the range 0-3 and is specified via the *idMod* parameter. The shift direction is specified by the *fShRight* parameter. The data is shifted right (i.e. LSB first) if *fShRight* is TRUE and left (i.e. MSB first) if *fShRight* is FALSE.

DspiGetSpeed(HIF hif, DWORD * pfrqCur)

Parameters

hif	- open interface handle on the device
pfrqCur	- variable to receive current SCK frequency

This function gets the current SCK frequency on an enabled SPI port and returns it by reference in the *pfrqCur* parameter. The value returned is the current SPI clock frequency in HZ.

DspiSetSpeed(HIF hif, DWORD frqReq, DWORD * pfrqSet)

Parameters

hif	- open interface handle on the device
frqReq	- requested value of SCK clock frequency
pfrqSet	- pointer to return actual set SCK frequency

This function is used to set the SCK frequency on an enabled SPI port. The desired frequency is HZ is specified in *frqReq*. The SPI clock frequency will be set to the highest supported frequency that doesn't exceed the requested value or the lowest supported frequency if the requested frequency is lower than the lowest supported frequency. The actual frequency obtained is returned in *pfrqSet*.

Note: This API call is only available to ports with the dprpSpiSetSpeed property.

DspiSetDelay(HIF hif, DWORD tusDelay)

Parameters

hif	- open interface handle on the device
tusDelay	- value of inter-byte delay to set

This function sets the inter-byte delay for an enabled SPI port. The port will wait this amount of time after sending a byte before sending the next byte. This can be used to allow time for the slave device to complete any necessary processing of a byte before the next one is sent. The delay value is in microseconds.

Note: This API call is only available to ports with the dprpSpiDelay property.

DspiGetDelay(HIF hif, DWORD * ptusDelay)

Parameters

hif	- open interface handle on the device
ptusDelay	- variable to receive current inter-byte delay setting

This function gets the delay between bytes for an enabled SPI port. The delay value is in micro seconds.

Note: This API call is only available to ports with the dprpSpiDelay property.

DspiSetStartEndDelay(HIF hif, DWORD tusStart, DWORD tusEnd)

Parameters

hif	- open interface handle on the device
tusStart	- value of the start delay to set
tusEnd	- value of the end delay to set

This function sets the delay that occurs between the slave select edge and the first byte (start delay) and the delay that occurs between the last byte and the slave select edge (end delay). The delay values are specified in microseconds.

Note: This API call is only available to ports with the dprpSpiStartEndDelay property. Calling DspiSetDelay after this function may result in the start and end delay being set to the same value as the inter-byte delay. If you wish to configure a port's inter-byte delay then call DspiSetDelay prior to calling this function.

DspiGetStartEndDelay(HIF hif, DWORD * ptusStart, DWORD * ptusEnd)

Parameters

hif	- open interface handle on the device
ptusStart	- variable to receive current start delay setting
ptusEnd	- variable to receive current end delay setting

This function gets the delay that occurs between the slave select edge and the first byte (start delay) and the delay that occurs between the last byte and the slave select edge (end delay) for an enabled SPI port. The delay value is in micro seconds.

Note: This API call is only available to ports with the dprpSpiStartEndDelay property.

DspiPutByte(HIF hif, BOOL fSelStart, BOOL fSelEnd, BYTE bSnd, BYTE * pbRcv, BOOL fOverlap)

Parameters

hif	- open interface handle on the device
fSelStart	- state to SS signal to before byte is shifted to slave
fSelEnd	- state to set SS signal to after byte is shifted to slave
bSnd	- byte to shift in to the slave
pbRcv	- pointer to receive byte shifted out of slave. If set to NULL, no byte is returned.
fOverlap	- TRUE if operation should be overlapped

This function sends a single byte to the slave device on an enabled port. It simultaneously reads a byte from the slave returns it in *pbRcv*. If a NULL value is specified for *pbRcv*, then no byte will be returned.

The parameters *fSelStart* and *fSelEnd* specify the state of the slave select signal at the start and at the end of the operation. The slave select signal will be set to the level specified by *fSelStart* before the data byte is sent to the slave. It will then be set to the level specified by *fSelEnd* after the byte has been sent to the slave. Specifying TRUE for these parameters sets the slave select signal to logic 1 and specifying FALSE sets it to logic 0.

DspiPut(HIF hif, BOOL fSelStart, BOOL fSelEnd, BYTE * rgbSnd, BYTE * rgbRcv, DWORD cbSnd, BOOL fOverlap)

Parameters

hif	- open interface handle on the device
fSelStart	- state to set SS signal at start of operation
fSelEnd	- state to set SS signal at end of operation
rgbSnd	- buffer of bytes to send to slave device
rgbRcv	- buffer to receive bytes read from slave device
cbSnd	- number of bytes to send
fOverlap	- TRUE if operation should be overlapped

This function sends the specified number of bytes to the slave on an enabled port. The data to be sent to the slave is in the buffer specified by *rgbSnd*. For each byte sent to the slave, a byte will be read from the slave. The data read from the slave will be returned in the buffer specified by *rgbRcv*. If the data read from the slave isn't needed, a NULL value can be specified for *rgbRcv*. The number of bytes to be sent to the slave is specified by *cbSnd*.

The parameters *fSelStart* and *fSelEnd* specify the state of the slave select signal at the start and at the end of the operation. The slave select signal will be set to the level specified by *fSelStart* before the first data byte is sent to the slave. It will then be set to the level specified by *fSelEnd* after the last byte has been sent to the slave. Specifying TRUE for these parameters sets the slave select signal to logic 1 and specifying FALSE sets it to logic 0.

DspiGet(HIF hif, BOOL fSelStart, BOOL fSelEnd, BYTE bFill, BYTE * rgbRcv, DWORD cbRcv, BOOL fOverlap)

Parameters

hif	- open interface handle on the device
fSelStart	- state to set SS signal to before byte buffer is shifted out
fSelEnd	- state to set SS signal to after byte buffer is shifted out
bFill	- level of MOSI during transaction. TRUE = '1', FALSE = '0'.
pbRcv	- buffer to receive bytes shifted out of the slave
cbSnd	- number of bytes to get
fOverlap	- TRUE if operation should be overlapped

This function reads the specified number bytes out of the slave on an enabled port. Data must be sent to the slave device in order to read from it. The parameter *bFill* specifies the data to be sent to the slave to allow each byte to be read from the slave. The data read from the slave will be placed in the buffer specified by *rgbRcv*. The number of bytes to be read is specified by *cbRcv*.

The parameters *fSelStart* and *fSelEnd* specify the state of the slave select signal at the start and at the end of the operation. The slave select signal will be set to the level specified by *fSelStart* before the first data byte is sent to the slave. It will then be set to the level specified by *fSelEnd* after the last byte has been sent to the slave. Specifying TRUE for these parameters sets the slave select signal to logic 1 and specifying FALSE sets it to logic 0.