Future Technology Devices International Ltd.

Application Note AN_111

Programmers Guide for High Speed FTCSPI DLL

Document Reference No.: FT_000112

Version 1.1

Issue Date: 2009-03-18

This document provides details of the function calls required when using the High Speed FTCSPI dll.

Future Technology Devices International Limited (FTDI)

Unit1, 2 Seaward Place, Centurion Business Park, Glasgow G41 1HH United Kingdom
Tel.: +44 (0) 141 429 2777 Fax: + 44 (0) 141 429 2758
E-Mail (Support): **support1@ftdichip.com** Web: http://www.ftdichip.com

TABLE OF CONTENTS

1 Introduction

The FT2232D, FT2232H and FT4232H devices contains FTDI's multi-protocol synchronous serial engine (MPSSE) controller, which may be used to interface to many popular synchronous serial protocols including JTAG, SPI and I2C.

The FT2232 SPI API will provide a set of function's to allow a programmer to control the FT2232D dual device MPSSE controller, the FT2232H dual device MPSSE hi-speed controller and the FT4232H quad device MPSSE hi-speed controller, to communicate with other devices using the Serial Peripheral Interface(SPI) synchronous serial protocol interface. The FT2232 SPI API will be contained within the **FTCSPI.DLL**. The FT2232D dual device can communicate with up to a maximum of 5 SPI devices. This is achieved because the **FTCSPI.DLL** supports 5 chip select pins on the FT2232D dual device.

The FTCSPI DLL has been created to allow application developers to use the FT2232D, FT2232H and FT4232H devices to create a USB to Serial Peripheral Interface (SPI) protocol interface without any knowledge of the MPSSE command set. All of the functions in FTCSPI.DLL can be replicated using calls to FTD2XX.DLL and sending the appropriate commands to the MPSSE.

The FT2232D MPSSE controller is only available through channel A of the FT2232D device; channel B of the FT2232D device does not support the MPSSE. Channel B may be controlled independently using FTDI's FTCD2XX drivers while channel A is being used for SPI communication.

The FT2232H MPSSE controller is available through channels A and B of the FT2232H device; both channels A and B can be used for SPI communication.

The FT4232H MPSSE controller is only available through channels A and B of the FT4232H device; channels C and D of the FT4232H device do not support the MPSSE. Channels C and D may be controlled independently using FTDI's FTCD2XX drivers while channels A and B are being used for SPI communication.

This document lists all of the functions available in FTCSPI.DLL.

2 Application Programming Interface (API)

2.1 Public Functions

2.1.1 SPI_GetNumDevices

FTC_STATUS SPI_GetNumDevices(LPDWORD lpdwNumDevices)

This function must be used, if more than one FT2232D dual device will be connected to a system. This function returns the number of available FT2232D dual device(s) connected to a system.

Parameters

lpdwNumDevices	Pointer to a variable of type DWORD which receives the actual number of available FT2232D dual device(s) connected to a system.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_IO_ERROR

2.1.2 SPI_GetNumHiSpeedDevices

FTC_STATUS SPI_GetNumHiSpeedDevices(LPDWORD lpdwTotalNumHiSpeedDevices)

This function must be used, if more than one FT2232H dual/FT4232H quad hi-speed devices will be connected to a system. This function returns the number of available FT2232H dual and FT4232H quad hi-speed device(s) connected to a system.

Parameters

lpdwTotalNumHiSpeedDevices	Pointer to a variable of type DWORD which receives the total number of available FT2232H dual and FT4232H quad hi-speed device(s) connected to a system.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_IO_ERROR

2.1.3 SPI_GetDeviceNameLocID

FTC_STATUS SPI_GetDeviceNameLocID(DWORD dwDeviceNameIndex, LPSTR lpDeviceNameBuffer, DWORD dwBufferSize, LPDWORD lpdwLocationID)

This function returns the name and the location identifier of the specified FT2232D dual device connected to a system.

Parameters

dwDeviceNameIndex	Index of the FT2232D dual device. Use the FT2232D_GetNumDevices function call, see section 2.1.1, to get the number of available FT2232D dual device(s) connected to a system. Example: if the number of a specific FT2232D dual device returned is 2 then valid index values will be 0 and 1.
lpDeviceNameBuffer	Pointer to buffer that receives the device name of the specified FT2232D dual device connected to a system. The string will be NULL terminated.
dwBufferSize	Length of the buffer created for the device name string. Set buffer length to a minimum of 50 characters.
lpdwLocationID	Pointer to a variable of type DWORD which receives the location identifier of the specified FT2232D dual device connected to a system.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_DEVICE_NOT_FOUND
FTC_INVALID_DEVICE_NAME_INDEX
FTC_NULL_ DEVICE_NAME_BUFFER_POINTER
FTC_ DEVICE_NAME_BUFFER_TOO_SMALL
FTC_IO_ERROR

2.1.4 SPI_GetHiSpeedDeviceNameLocIDChannel

FTC_STATUS SPI_GetHiSpeedDeviceNameLocIDChannel(DWORD dwDeviceNameIndex, LPSTR lpDeviceNameBuffer, DWORD dwDeviceNameBufferSize, LPDWORD lpdwLocationID, LPSTR lpChannelBuffer)

This function returns the name, location identifier and the channel of the specified FT2232H dual hi-speed device or FT4232H quad hi-speed device connected to a system.

Parameters

dwDeviceNameIndex	Index of the FT2232H dual hi-speed device or FT4232H quad hi-speed device. Use the SPI_GetNumHiSpeedDevices function call, see section 2.1.2, to get the number of available FT2232H dual and FT4232H quad hi-speed device(s) connected to a system. Example: if the number of FT2232H dual and FT4232H quad hi-speed device(s) returned is 2 then valid index values will be 0 and 1.
lpDeviceNameBuffer	Pointer to buffer that receives the device name of the specified FT2232H dual hi-speed device or FT4232H quad hi-speed device connected to a system. The string will be NULL terminated.
dwDeviceNameBufferSize	Length of the buffer created for the device name string. Set buffer length to a minimum of 100 characters.
lpdwLocationID	Pointer to a variable of type DWORD which receives the location identifier of the specified FT2232H dual hi-speed device or FT4232H quad hi-speed device connected to a system.
lpChannelBuffer	Pointer to a buffer that receives the channel of the specified FT2232H dual hi-speed device or FT4232H quad hi-speed device connected to a system. The buffer will only return a single character either A or B. The string will be NULL terminated.
dwChannelBufferSize	Length of the buffer created for the channel string. Set buffer length to a minimum of 5 characters.
lpdwHiSpeedDeviceType	Pointer to a variable of type DWORD which receives the actual type of hi-speed device, FT2232H dual hi-speed or FT4232H quad hi-speed. **<u>Valid Hi-Speed Device Types</u>** FT2232H_DEVICE_TYPE FT4232H_DEVICE_TYPE

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_DEVICE_NOT_FOUND
FTC_INVALID_DEVICE_NAME_INDEX
FTC_NULL_DEVICE_NAME_BUFFER_POINTER
FTC_ DEVICE_NAME_BUFFER_TOO_SMALL
FTC_NULL_CHANNEL_BUFFER_POINTER
FTC_CHANNEL_BUFFER_TOO_SMALL
FTC_IO_ERROR

2.1.5 SPI_Open

FTC_STATUS SPI_Open(FTC_HANDLE *pftHandle)

This function must only be used, if a maximum of one FT2232D dual device will be connected to a system.

This function first determines which attached application is invoking this function. If an attached application invokes this function again and it's assigned handle is still open then it's assigned handle will be returned again. If another application attempts to open this device, which is already in use, an error code is returned. This function first then determines if a FT2232D dual device is present then checks that an application is not already using this FT2232D dual device. If another application is not using this FT2232D dual device then an attempt is made to open it. If the open was not successful an error code will be returned. If the open is successful, the FT2232D dual device is initialized to its default state, see section 2.1.11. If the initialization was successful the handle is passed back to the application. If the initialization was not successful an error code will be returned.

Parameters

pftHandle	Pointer to a variable of type FTC_HANDLE where the handle to the open device will be returned. This handle must then be used in all subsequent calls to access this device.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_DEVICE_NOT_FOUND
FTC_DEVICE_IN_USE
FTC_TOO_MANY_DEVICES
FTC_FAILED_TO_SYNCHRONIZE_DEVICE_MPSSE
FTC_FAILED_TO_COMPLETE_COMMAND
FTC_IO_ERROR
FTC_INSUFFICIENT_RESOURCES

2.1.6 SPI_OpenEx

FTC_STATUS SPI_OpenEx (LPSTR lpDeviceName, DWORD dwLocationID, FTC_HANDLE *pftHandle)

This function first determines which attached application is invoking this function. If an attached application invokes this function again and it's assigned handle is still open then it's assigned handle will be returned again. If another application attempts to open this device, which is already in use, an error code is returned. This function first determines if the specified FT2232D dual device is present then checks that an application is not already using the specified FT2232D dual device. If another application is not using the specified FT2232D dual device then an attempt is made to open it. If the open was not successful an error code will be returned. If the open is successful, the specified FT2232D dual device is initialized to its default state, see section 2.1.11. If the initialization was successful the handle is passed back to the application. If the initialization was not successful an error code will be returned.

Parameters

lpDeviceName	Pointer to a NULL terminated string that contains the name of the specified FT2232D dual device to be opened.
dwLocationID	Specifies the location identifier of the specified FT2232D dual device to be opened.
pftHandle	Pointer to a variable of type FTC_HANDLE where the handle to the open device will be returned. This handle must then be used in all subsequent calls to access this device.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_NULL_DEVICE_NAME_BUFFER_POINTER
FTC_INVALID_DEVICE_NAME
FTC_INVALID_LOCATION_ID
FTC_DEVICE_NOT_FOUND
FTC_DEVICE_IN_USE
FTC_FAILED_TO_SYNCHRONIZE_DEVICE_MPSSE
FTC_FAILED_TO_COMPLETE_COMMAND
FTC_IO_ERROR
FTC_INSUFFICIENT_RESOURCES

2.1.7 SPI_OpenHiSpeedDevice

FTC_STATUS SPI_OpenHiSpeedDevice (LPSTR lpDeviceName, DWORD dwLocationID, LPSTR lpChannel, FTC_HANDLE *pftHandle)

This function first determines which attached application is invoking this function. If an attached application invokes this function again and it's assigned handle is still open then it's assigned handle will be returned again. If another application attempts to open this device, which is already in use, an error code is returned. This function first determines if the specified FT2232H dual hi-speed device or FT4232H quad hi-speed device is present then checks that an application is not already using the specified FT2232H dual hi-speed device or FT4232H quad hi-speed device. If another application is not using the specified FT2232H dual hi-speed device or FT4232H quad hi-speed device then an attempt is made to open it. If the open was not successful an error code will be returned. If the open is successful, the specified FT2232H dual hi-speed device or FT4232H quad hi-speed device is initialized to its default state, see section 2.1.11. If the initialization was successful the handle is passed back to the application. If the initialization was not successful an error code will be returned.

Parameters

lpDeviceName	Pointer to a NULL terminated string that contains the name of the specified FT2232H dual hi-speed device or FT4232H quad hi-speed device to be opened.
dwLocationID	Specifies the location identifier of the specified FT2232H dual hi-speed device or FT4232H quad hi-speed device to be opened.
lpChannel	Pointer to a NULL terminated string that contains the channel of the specified FT2232H dual hi-speed device or FT4232H quad hi-speed device to be opened. The channel identifier will be a single character either A or B.
pftHandle	Pointer to a variable of type FTC_HANDLE where the handle to the open device will be returned. This handle must then be used in all subsequent calls to access this device.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_NULL_DEVICE_NAME_BUFFER_POINTER
FTC_INVALID_DEVICE_NAME
FTC_INVALID_LOCATION_ID
FTC_INVALID_CHANNEL
FTC_DEVICE_NOT_FOUND
FTC_DEVICE_IN_USE
FTC_FAILED_TO_SYNCHRONIZE_DEVICE_MPSSE
FTC_FAILED_TO_COMPLETE_COMMAND
FTC_IO_ERROR
FTC_INSUFFICIENT_RESOURCES

2.1.8 SPI_GetHiSpeedDeviceType

FTC_STATUS SPI_GetHiSpeedDeviceType (FTC_HANDLE ftHandle, LPDWORD lpdwHiSpeedDeviceType)

This function returns the high speed device type detected. The type should either be FT2232H or FT4232H.

Parameters

ftHandle	Handle of the FT2232H dual hi-speed device or FT4232H quad hi-speed device opened.
lpdwHiSpeedDeviceType	Pointer to a variable of type DWORD which receives the device type.

Valid Hi-Speed Device Types

FT2232H_DEVICE_TYPE
FT4232H_DEVICE_TYPE

Return Value
Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_IO_ERROR

2.1.9 SPI_Close

This function closes a previously opened handle to a FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device.

Parameters

ftHandle	Handle of the FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device to close.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_IO_ERROR

2.1.10 SPI_CloseDevice

FTC_STATUS **SPI_CloseDevice** (FTC_HANDLE ftHandle, PFTC_CLOSE_FINAL_STATE_PINS pCloseFinalStatePinsData)

This function closes a previously opened handle to a FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device.

Parameters

ftHandle	Handle of the FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device to close.
pCloseFinalStatePinsData	Pointer to the structure that contains the data that is used to set the final state of output pins TCK, TDI, TMS

Return Value
Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_IO_ERROR

2.1.11 SPI_InitDevice

FTC_STATUS SPI_InitDevice(FTC_HANDLE ftHandle, DWORD dwClockDivisor)

This function initializes the FT2232D dual device, by carrying out the following in the following order:

- resets the device and purge device USB input buffer
- sets the device USB input and output buffers to 64K bytes
- sets the special characters for the device, disable event and error characters
- sets the device read timeout to infinite
- sets the device write timeout to 5 seconds
- sets the device latency timer to 16 milliseconds
- reset MPSSE controller
- enable MPSSE controller
- synchronize the MPSSE
- resets the device and purge device USB input buffer
- set data in and data out clock frequency
- set MPSSE loopback state to off (default)

- resets the device and purge device USB input buffer
- reset Test Access Port(TAP) controller on an external device
- set the Test Access Port(TAP) controller on an external device to test idle mode

Parameters

ftHandle	Handle of a FT2232D dual device.
dwClockDivisor	Specifies a divisor, which will be used to set the frequency that will be used to clock data in and out of a FT2232D dual device. Valid range is 0 to 65535. The highest clock frequency is represented by 0, which is equivalent to 6MHz, the next highest clock frequency is represented by 1, which is equivalent to 3MHz and the lowest clock frequency is represented by 65535, which is equivalent to 91Hz. To obtain the actual frequency in Hz, represented by the specified divisor, see section 2.1.16. Note: the frequency in Hz, represented by the divisor, is calculated using the following formula: frequency = 12MHz/((1 + dwClockDivisor) * 2).

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_INVALID_CLOCK_DIVISOR
FTC_FAILED_TO_SYNCHRONIZE_DEVICE_MPSSE
FTC_FAILED_TO_COMPLETE_COMMAND
FTC_IO_ERROR
FTC_INSUFFICIENT_RESOURCES

2.1.12 SPI_TurnOnDivideByFiveClockingHiSpeedDevice

FTC_STATUS **I2C_TurnOnDivideByFiveClockinghiSpeedDevice** (FTC_HANDLE fthandle)

This function turns on the divide by five for the MPSSE clock to allow the hi-speed devices FT2232H and FT4232H to clock at the same rate as the FT2232D device. This allows for backward compatibility.

Parameters

ftHandle	Handle of a FT2232H dual hi-speed device or FT4232H quad hi-speed device.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_IO_ERROR

2.1.13 SPI_TurnOffDivideByFiveClockingHiSpeedDevice

FTC_STATUS **SPI_TurnOffDivideByFiveClockinghiSpeedDevice** (FTC_HANDLE fthandle)

This function turns off the divide by five for the MPSSE clock to allow the hi-speed devices FT2232H and FT4232H to clock at the higher speeds. Maximum is 30Mbit/s

Parameters

ftHandle	Handle of a FT2232H dual hi-speed device or FT4232H quad hi-speed device.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_IO_ERROR

2.1.14 SPI_SetDeviceLatencyTimer

FTC_STATUS SPI_SetDeviceLatencyTimer(FTC_HANDLE ftHandle, BYTE timerValue)

This function sets the value in milliseconds of the latency timer for a FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device. The latency timer is used to flush any remaining data received from a FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device from the USB input buffer, when the latency timer times out.

Parameters

ftHandle	Handle of a FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device.
timerValue	Specifies the value, in milliseconds, of the latency timer. Valid range is 2 - 255.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_INVALID_TIMER_VALUE
FTC_IO_ERROR

2.1.15 SPI_GetDeviceLatencyTimer

FTC_STATUS SPI_GetDeviceLatencyTimer(FTC_HANDLE ftHandle, LPBYTE lpTimerValue)

This function gets the value in milliseconds of the latency timer for a FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device. The latency timer is used to flush any remaining data received from a FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device from the USB input buffer, when the latency timer times out.

Parameters

ftHandle	Handle of a FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device.
lpTimerValue	Pointer to a variable of type BYTE which receives the actual latency timer value in milliseconds. Valid range is 2 - 255.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_IO_ERROR

2.1.16 SPI_GetClock

FTC_STATUS SPI_GetClock (DWORD dwClockDivisor, LPDWORD lpdwClockFrequencyHz)

This function calculates the frequency in **Hz**, that data will be clocked in and out of a FT2232D dual device.

Parameters

dwClockDivisor	Specifies a divisor, which will be used to set the frequency that will be used to clock data in and out of a FT2232D dual device. Valid range is 0 to 65535. The highest clock frequency is represented by 0, which is equivalent to 6MHz, the next highest clock frequency is represented by 1, which is equivalent to 3MHz and the lowest clock frequency is represented by 65535, which is equivalent to 91Hz. To obtain the actual frequency in Hz, represented by the specified divisor, see section 2.1.16.
lpdwClockFrequencyHz	Pointer to a variable of type DWORD which receives the actual frequency in **Hz**, that data will be clocked in and out of a FT2232D dual device. Note: The frequency in Hz, represented by the divisor, is calculated using the following formula: frequency = 12MHz/((1 + dwClockDivisor) * 2).

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_CLOCK_DIVISOR

2.1.17 SPI_GetHiSpeedDeviceClock

FTC_STATUS SPI_GetHiSpeedDeviceClock (DWORD dwClockDivisor, LPDWORD lpdwClockFrequencyHz)

This function calculates the frequency in **Hz**, that data will be clocked in and out of a FT2232H dual hi-speed device or FT4232H quad hi-speed device.

Parameters

dwClockDivisor	Specifies a divisor, which will be used to set the frequency that will be used to clock data in and out of a FT2232H dual hi-speed device or FT4232H quad hi-speed device. Valid range is 0 to 65535. The highest clock frequency is represented by 0, which is equivalent to 30MHz, the next highest clock frequency is represented by 1, which is equivalent to 15MHz and the lowest clock frequency is represented by 65535, which is equivalent to 457Hz.
lpdwClockFrequencyHz	Pointer to a variable of type DWORD which receives the actual frequency in **Hz**, that data will be clocked in and out of a FT2232H dual hi-speed device or FT4232H quad hi-speed device.

Note: the frequency in Hz, represented by the divisor, is calculated using the following formula:

frequency = 60MHz/((1 + dwClockDivisor) * 2).

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_CLOCK_DIVISOR

2.1.18 SPI_SetClock

FTC_STATUS SPI_SetClock (FTC_HANDLE ftHandle, DWORD dwClockDivisor, LPDWORD lpdwClockFrequencyHz)

This function sets and calculates the frequency in **Hz**, that data will be clocked in and out of a FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device.

Parameters

ftHandle	Handle of a FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device.
dwClockDivisor	Specifies a divisor, which will be used to set the frequency that will be used to clock data in and out of a FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device. Valid range is 0 to 65535. The highest clock frequency is represented by 0, which is equivalent to 6MHz for the FT2232D dual device and 30MHz for the FT2232H dual and FT4232H quad hi-speed devices, the next highest clock frequency is represented by 1, which is equivalent to 3MHz for the FT2232D dual device and 15MHz for the FT2232H dual and FT4232H quad hi-speed devices and the lowest clock frequency is represented by 65535, which is equivalent to 91Hz for the FT2232D dual device and 457Hz for the FT2232H dual and FT4232H quad hi-speed devices.

lpdwClockFrequencyHz	Pointer to a variable of type DWORD which receives the actual frequency in **Hz**, that data will be clocked in and out of a FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device. For the FT2232D dual device the frequency in Hz, represented by the divisor, is calculated using the following formula:
	
	frequency = 12MHz/((1 + dwClockDivisor) * 2)
	
	For the FT2232H dual and FT4232H quad hi-speed devices the frequency in Hz, represented by the divisor, is calculated using the following formula:
	
	frequency = 60MHz/((1 + dwClockDivisor) * 2)

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_INVALID_CLOCK_DIVISOR
FTC_FAILED_TO_COMPLETE_COMMAND
FTC_IO_ERROR

2.1.19 SPI_SetLoopback

FTC_STATUS SPI_SetLoopback(FTC_HANDLE ftHandle, BOOL bLoopbackState)

This function controls the state of the FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device loopback. The FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device is set to loopback for testing purposes.

Parameters

ftHandle	Handle of the FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device.
bLoopbackState	Controls the state of the FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device loopback. To switch loopback on(TRUE) or off(FALSE).

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_FAILED_TO_COMPLETE_COMMAND
FTC_IO_ERROR

2.1.20 SPI_SetGPIOs

FTC_STATUS SPI_SetGPIOs(FTC_HANDLE ftHandle, PFTC_CHIP_SELECT_PINS pChipSelectsDisableStates, PFTC_INPUT_OUTPUT_PINS pHighInputOutputPinsData)

This function controls the disable states of the 5 chip select pins and the use of the 4 general purpose higher input/output pins (GPIOH1 – GPIOH4) of the FT2232D dual device.

Parameters

ftHandle	Handle of a FT2232D dual device.
pChipSelectsDisableStates	Pointer to the structure that contains the disable states for the 5 chip select pins of the FT2232D dual device.
pHighInputOutputPinsData	Pointer to the structure that contains the data that is used to control the use of the 4 general purpose higher input/output pins (GPIOH1 – GPIOH4) of the FT2232D dual device.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_NULL_CHIP_SELECT_BUFFER_POINTER
FTC_NULL_INPUT_OUTPUT_BUFFER_POINTER
FTC_FAILED_TO_COMPLETE_COMMAND
FTC_IO_ERROR

Example:

```
typedef struct FTC_Chip_Select_Pins {
  BOOL          bADBUS3ChipSelectPinState;   Set chip select pin to disable(TRUE),
enabled(FALSE)
  BOOL          bADBUS4GPIOL1PinState;       Set chip select pin to disable(TRUE),
enabled(FALSE)
  BOOL          bADBUS5GPIOL2PinState;       Set chip select pin to disable(TRUE),
enabled(FALSE)
  BOOL          bADBUS6GPIOL3PinState;       Set chip select pin to disable(TRUE),
enabled(FALSE)
  BOOL          bADBUS7GPIOL4PinState;       Set chip select pin to disable(TRUE),
enabled(FALSE)
} FTC_CHIP_SELECT_PINS *PFTC_CHIP_SELECT_PINS
```

```
typedef struct FTC_Input_Output_Pins {
  BOOL          bPin1InputOutputState;        Set pin1 to input mode(FALSE), set pin1
to output mode(TRUE)
  BOOL          bPin1LowHighState;     If pin1 is set to output mode, set pin1
low(FALSE), high(TRUE)
  BOOL          bPin2InputOutputState;        Set pin2 to input mode(FALSE), set pin2
to output mode(TRUE)
  BOOL          bPin2LowHighState;     If pin2 is set to output mode, set pin2
low(FALSE), high(TRUE)
  BOOL          bPin3InputOutputState;        Set pin3 to input mode(FALSE), set pin3
to output mode(TRUE)
  BOOL          bPin3LowHighState;     If pin3 is set to output mode, set pin3
low(FALSE), high(TRUE)
  BOOL          bPin4InputOutputState;        Set pin4 to input mode(FALSE), set pin4
to output mode(TRUE)
  BOOL          bPin4LowHighState;     If pin4 is set to output mode, set pin4
low(FALSE), high(TRUE)
} FTC_INPUT_OUTPUT_PINS *PFTC_INPUT_OUTPUT_PINS
```

2.1.21 SPI_SetHiSpeedDeviceGPIOs

FTC_STATUS SPI_SetHiSpeedDeviceGPIOs(FTC_HANDLE ftHandle, PFTC_CHIP_SELECT_PINS pChipSelectsDisableStates, PFTH_INPUT_OUTPUT_PINS pHighInputOutputPinsData)

This function controls the disable states of the 5 chip select pins and the 8 general purpose higher input/output pins (GPIOH1 – GPIOH8) of the FT2232H dual hi-speed device or the disable states of the 5 chip select pins of the FT4232H quad hi-speed device.

Parameters

ftHandle	Handle of the FT2232H dual hi-speed device or FT4232H quad hi-speed device.
pChipSelectsDisableStates	Pointer to the structure that contains the disable states for the 5 chip select pins of the FT2232H dual hi-speed device or FT4232H quad hi- speed device.
pHighInputOutputPinsData	Pointer to the structure that contains the data that is used to control the 8 general purpose higher input/output pins (GPIOH1 – GPIOH8) of the FT2232H dual hi-speed device. Note: The 8 general purpose higher input/output pins (GPIOH1 – GPIOH8) do not physically exist on the FT4232H quad hi-speed device.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_NULL_INPUT_OUTPUT_BUFFER_POINTER
FTC_FAILED_TO_COMPLETE_COMMAND
FTC_IO_ERROR

Example:

```
typedef struct FTC_Chip_Select_Pins {
  BOOL          bADBUS3ChipSelectPinState;   Set chip select pin to disable(TRUE),
enabled(FALSE)
  BOOL          bADBUS4GPIOL1PinState;       Set chip select pin to disable(TRUE),
enabled(FALSE)
  BOOL          bADBUS5GPIOL2PinState;       Set chip select pin to disable(TRUE),
enabled(FALSE)
  BOOL          bADBUS6GPIOL3PinState;       Set chip select pin to disable(TRUE),
enabled(FALSE)
  BOOL          bADBUS7GPIOL4PinState;       Set chip select pin to disable(TRUE),
enabled(FALSE)
} FTC_CHIP_SELECT_PINS *PFTC_CHIP_SELECT_PINS
```

```
typedef struct FTH_Input_Output_Pins {
  BOOL          bPin1InputOutputState;          Set pin1 to input mode(FALSE), set pin1
to output mode(TRUE)
  BOOL          bPin1LowHighState;     If pin1 is set to output mode, set pin1
low(FALSE), high(TRUE)
  BOOL          bPin2InputOutputState;          Set pin2 to input mode(FALSE), set pin2
to output mode(TRUE)
  BOOL          bPin2LowHighState;     If pin2 is set to output mode, set pin2
low(FALSE), high(TRUE)
  BOOL          bPin3InputOutputState;          Set pin3 to input mode(FALSE), set pin3
to output mode(TRUE)
  BOOL          bPin3LowHighState;     If pin3 is set to output mode, set pin3
low(FALSE), high(TRUE)
  BOOL          bPin4InputOutputState;          Set pin4 to input mode(FALSE), set pin4
to output mode(TRUE)
  BOOL          bPin4LowHighState;     If pin4 is set to output mode, set pin4
low(FALSE), high(TRUE)
  BOOL          bPin5InputOutputState;          Set pin5 to input mode(FALSE), set pin5
to output mode(TRUE)
  BOOL          bPin5LowHighState;     If pin5 is set to output mode, set pin5
low(FALSE), high(TRUE)
  BOOL          bPin6InputOutputState;          Set pin6 to input mode(FALSE), set pin6
to output mode(TRUE)
  BOOL          bPin6LowHighState;     If pin6 is set to output mode, set pin6
low(FALSE), high(TRUE)
  BOOL          bPin7InputOutputState;          Set pin7 to input mode(FALSE), set pin7
to output mode(TRUE)
  BOOL          bPin7LowHighState;     If pin7 is set to output mode, set pin7
low(FALSE), high(TRUE)
  BOOL          bPin8InputOutputState;          Set pin8 to input mode(FALSE), set pin8
to output mode(TRUE)
  BOOL          bPin8LowHighState;     If pin8 is set to output mode, set pin8
low(FALSE), high(TRUE)
} FTH_INPUT_OUTPUT_PINS *PFTH_INPUT_OUTPUT_PINS
```

2.1.22 SPI_GetGPIOs

FTC_STATUS SPI_GetGPIOs(FTC_HANDLE ftHandle, PFTC_LOW_HIGH_PINS pHighPinsInputData)

This function gets the input states(low or high) of the 4 general purpose higher input/output pins (GPIOH1 – GPIOH4) of the FT2232D dual device.

Parameters

ftHandle	Handle of a FT2232D dual device.
pHighPinsInputData	Pointer to the structure that contains the input states(low or high) of the 4 general purpose higher input/output pins (GPIOH1 - GPIOH4) of the FT2232D dual device.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_NULL_INPUT_BUFFER_POINTER
FTC_FAILED_TO_COMPLETE_COMMAND
FTC_IO_ERROR

Example:

```
typedef struct FTC_Low_High_Pins {
  BOOL          bPin1LowHighState;    Pin1 input state low(FALSE), high(TRUE)
  BOOL          bPin2LowHighState;    Pin2 input state low(FALSE), high(TRUE)
  BOOL          bPin3LowHighState;    Pin3 input state low(FALSE), high(TRUE)
  BOOL          bPin4LowHighState;    Pin4 input state low(FALSE), high(TRUE)
} FTC_LOW_HIGH_PINS *PFTC_LOW_HIGH_PINS
```

2.1.23 SPI_GetHiSpeedDeviceGPIOs

FTC_STATUS SPI_GetHiSpeedDeviceGPIOs(FTC_HANDLE ftHandle, PFTH_LOW_HIGH_PINS pHighPinsInputData)

This function gets the input states(low or high) of the 8 general purpose input/output pins (GPIOH1 – GPIOH8) of the FT2232H dual hi-speed device.

Parameters

ftHandle	Handle of the FT2232H dual hi-speed device.
pHighPinsInputData	Pointer to the structure that contains the input states(low or high) of the 8 general purpose higher input/output pins (GPIOH1 – GPIOH8) of the FT2232H dual hi-speed device. Note: The 8 general purpose higher input/output pins (GPIOH1 – GPIOH8) do not physically exist on the FT4232H quad hi-speed device.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_NULL_INPUT_OUTPUT_BUFFER_POINTER
FTC_FAILED_TO_COMPLETE_COMMAND
FTC_IO_ERROR

Example:

```
typedef struct FTH_Low_High_Pins {
  BOOL          bPin1LowHighState;    Pin1 input state low(FALSE), high(TRUE)
  BOOL          bPin2LowHighState;    Pin2 input state low(FALSE), high(TRUE)
  BOOL          bPin3LowHighState;    Pin3 input state low(FALSE), high(TRUE)
  BOOL          bPin4LowHighState;    Pin4 input state low(FALSE), high(TRUE)
  BOOL          bPin5LowHighState;    Pin5 input state low(FALSE), high(TRUE)
  BOOL          bPin6LowHighState;    Pin6 input state low(FALSE), high(TRUE)
  BOOL          bPin7LowHighState;    Pin7 input state low(FALSE), high(TRUE)
  BOOL          bPin8LowHighState;    Pin8 input state low(FALSE), high(TRUE)
} FTH_LOW_HIGH_PINS *PFTH_LOW_HIGH_PINS
```

2.1.24 SPI_Write

FTC_STATUS SPI_Write(FTC_HANDLE ftHandle, PFTC_INIT_CONDITION pWriteStartCondition, BOOL bClockOutDataBitsMSBFirst, BOOL bClockOutDataBitsPosEdge, DWORD dwNumControlBitsToWrite, PWriteControlByteBuffer pWriteControlBuffer, DWORD dwNumControlBytesToWrite, BOOL bWriteDataBits, DWORD dwNumDataBitsToWrite, PWriteDataByteBuffer pWriteDataBuffer, DWORD dwNumDataBytesToWrite, PFTC_WAIT_DATA_WRITE pWaitDataWriteComplete, PFTC_HIGHER_OUTPUT_PINS pHighPinsWriteActiveStates)

This function writes data to an external device ie a device attached to a FT2232D dual device. A FT2232D dual device communicates with an external device by simulating the SPI synchronous protocol.

Parameters

ftHandle	Handle of a FT2232D dual device.
pWriteStartCondition	Pointer to the structure that contains the start output states(low or high) of the clock, data out and signal out/chip select pins of the FT2232D dual device.
bClockOutDataBitsMSBFirst	Clock out control and data bits Most Significant Bit(MSB) first(TRUE), clock out control and data bits Least Significant Bit(LSB) first(FALSE)
bClockOutDataBitsPosEdge	Clock out control and data bits on positive clock edge(TRUE), clock out control and data bits on negative clock edge(FALSE)
dwNumControlBitsToWrite	Specifies the number of control bits to be written to an external device. Valid range 2 to 2040. 2040 bits is equivalent to 255 bytes.
pWriteControlBuffer	Pointer to buffer that contains the control data to be written to an external device. Listed below are three examples of control and address bytes: Two Control bytes Control Address byte 1, Control Address byte 2 Two Control bytes, Control Address byte 1, Control Address byte 2
dwNumControlBytesToWrite	Specifies the number of control bytes in the write control buffer, which contains all the specified bits to be written to an external device. Valid range 1 to 255 bytes.
bWriteDataBits	Write data bits to an external device(TRUE), do not write any data bits to an external device(FALSE)
dwNumDataBitsToWrite	Specifies the number of data bits to be written to an external device. Valid range 2 to 524280. 524280 bits is equivalent to 64K bytes.
pWriteDataBuffer	Pointer to buffer that contains the data to be written to an external device.
dwNumDataBytesToWrite	Specifies the number of data bytes in the write data buffer, which contains all the specified bits to be written to an external device. Valid range 1 to 65535 ie 64K bytes.
pWaitDataWriteComplete	Pointer to the structure that contains the data that controls whether the FT2232D device should wait until all data bytes have been written to an external device, before returning.

pHighPinsWriteActiveStates	Pointer to the structure that contains which of the 4 general purpose higher input/output pins (GPIOH1 - GPIOH4) of a FT2232D dual device, are to be used during a write to an external device. Each GPIO pin that is to be used during a write to an external device must have been previously configured as an output pin, see section 2.1.20.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_NULL_INITIAL_CONDITION_BUFFER_POINTER
FTC_INVALID_NUMBER_CONTROL_BITS
FTC_NULL_WRITE_CONTROL_BUFFER_POINTER
FTC_INVALID_NUMBER_CONTROL_BYTES
FTC_NUMBER_CONTROL_BYTES_TOO_SMALL
FTC_INVALID_NUMBER_WRITE_DATA_BITS
FTC_NULL_WRITE_DATA_BUFFER_POINTER
FTC_INVALID_NUMBER_WRITE_DATA_BYTES
FTC_NUMBER_WRITE_DATA_BYTES_TOO_SMALL
FTC_NULL_WAIT_DATA_WRITE_BUFFER_POINTER
FTC_NULL_OUTPUT_PINS_BUFFER_POINTER
FTC_INVALID_INIT_CLOCK_PIN_STATE
FTC_INVALID_FT2232D_CHIP_SELECT_PIN
FTC_INVALID_FT2232D_DATA_WRITE_COMPLETE_PIN
FTC_DATA_WRITE_COMPLETE_TIMEOUT
FTC_INVALID_CONFIGURATION_HIGHER_GPIO_PIN
FTC_FAILED_TO_COMPLETE_COMMAND
FTC_IO_ERROR

Example:

typedef struct FTC_Init_Condition {

BOOL	bClockPinState;	Set clock pin output low(FALSE), high(TRUE)
BOOL	bDataOutPinState;	Set data out pin output low(FALSE), high(TRUE)
BOOL	bChipSelectPinState;	Set chip select pin to disable, output low(FALSE), high(TRUE)
DWORD	dwChipSelectPin	Specifies which pin on the FT2232D dual device, will be used as the chip select pin.

<u>Valid FT2232D Pins</u>

ADBUS3ChipSelect

ADBUS4GPIOL1

ADBUS5GPIOL2

ADBUS6GPIOL3

ADBUS7GPIOL4

} FTC_INIT_CONDITION *PFTC_INIT_CONDITION

```
#define MAX_WRITE_CONTROL_BYTES_BUFFER_SIZE  256      // 256 bytes

typedef BYTE WriteControlByteBuffer[MAX_WRITE_CONTROL_BYTES_BUFFER_SIZE];
typedef WriteControlByteBuffer  *PWriteControlByteBuffer;

#define MAX_WRITE_DATA_BYTES_BUFFER_SIZE              65536  // 64K bytes

typedef BYTE WriteDataByteBuffer[MAX_WRITE_DATA_BYTES_BUFFER_SIZE];
typedef WriteDataByteBuffer  *PWriteDataByteBuffer;

typedef struct FTC_Wait_Data_Write {
```

BOOL	bWaitDataWriteComplete;	Wait until all data bytes have been written to an External device, wait(TRUE), do not wait(FALSE).
DWORD	dwWaitDataWritePin;	Specifies which pin on the FT2232D dual device, indicates, when all the data bytes have been written to an external device. If one of the 4 higher GPIO pins (GPIOH1 - GPIOH4) is selected, it must have been previously configured as an input pin, see section 2.1.20.

Valid FT2232D Pins

ADBUS2DataIn

ACBUS0GPIOH1

ACBUS1GPIOH2

ACBUS2GPIOH3

ACBUS3GPIOH4

BOOL	bDataWriteCompleteState;	Specifies what state indicates that all data bytes have been written to an external device, low(FALSE), high(TRUE).
DWORD	dwDataWriteTimeoutmSecs;	Timeout interval in milliseconds to wait for all data bytes to be written to an external device.

```
} FTC_WAIT_DATA_WRITE *PFTC_WAIT_DATA_WRITE

typedef struct FTC_Higher_Output_Pins {
```

BOOL	bPin1State;	Set pin1 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin1ActiveState;	Indicates the state, pin1 must be set to during a write to an external device, low(FALSE), high(TRUE)
BOOL	bPin2State;	Set pin2 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin2ActiveState;	Indicates the state, pin2 must be set to during a write to an external device, low(FALSE), high(TRUE)
BOOL	bPin3State;	Set pin3 to be used during a write to an external device, use(TRUE), not use(FALSE)

BOOL	bPin3ActiveState;	Indicates the state, pin3 must be set to during a write to an external device, low(FALSE), high(TRUE)
BOOL	bPin4State;	Set pin4 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin4ActiveState;	Indicates the state, pin4 must be set to during a write to an external device, low(FALSE), high(TRUE)

} FTC_HIGHER_OUTPUT_PINS *PFTC_HIGHER_OUTPUT_PINS

2.1.25 SPI_WriteHiSpeedDevice

FTC_STATUS SPI_WriteHiSpeedDevice(FTC_HANDLE ftHandle, PFTC_INIT_CONDITION pWriteStartCondition, BOOL bClockOutDataBitsMSBFirst, BOOL bClockOutDataBitsPosEdge, DWORD dwNumControlBitsToWrite, PWriteControlByteBuffer pWriteControlBuffer, DWORD dwNumControlBytesToWrite, BOOL bWriteDataBits, DWORD dwNumDataBitsToWrite, PWriteDataByteBuffer pWriteDataBuffer, DWORD dwNumDataBytesToWrite, PFTC_WAIT_DATA_WRITE pWaitDataWriteComplete, PFTH_HIGHER_OUTPUT_PINS pHighPinsWriteActiveStates)

This function writes data to an external device ie a device attached to a FT2232H dual hi-speed device or FT4232H quad hi-speed device. A FT2232H dual hi-speed device or FT4232H quad hi-speed device communicates with an external device by simulating the SPI synchronous protocol.

Parameters

ftHandle	Handle of a FT2232H dual hi-speed device or FT4232H quad hi-speed device.
pWriteStartCondition	Pointer to the structure that contains the start output states(low or high) of the clock, data out and signal out/chip select pins of the FT2232H dual hi-speed device or FT4232H quad hi-speed device.
bClockOutDataBitsMSBFirst	Clock out control and data bits Most Significant Bit(MSB) first(TRUE), clock out control and data bits Least Significant Bit(LSB) first(FALSE)
bClockOutDataBitsPosEdge	Clock out control and data bits on positive clock edge(TRUE), clock out control and data bits on negative clock edge(FALSE)
dwNumControlBitsToWrite	Specifies the number of control bits to be written to an external device. Valid range 2 to 2040. 2040 bits is equivalent to 255 bytes.
pWriteControlBuffer	Pointer to buffer that contains the control data to be written to an external device. Listed below are three examples of control and address bytes: Two Control bytes Control Address byte 1, Control Address byte 2 Two Control bytes, Control Address byte 1, Control Address byte 2
dwNumControlBytesToWrite	Specifies the number of control bytes in the write control buffer, which contains all the specified bits to be written to an external device. Valid range 1 to 255 bytes.
bWriteDataBits	Write data bits to an external device(TRUE), do not write any data bits to an external device(FALSE)
dwNumDataBitsToWrite	Specifies the number of data bits to be written to an external device. Valid range 2 to 524280. 524280 bits is equivalent to 64K bytes.
pWriteDataBuffer	Pointer to buffer that contains the data to be written to an external device.
dwNumDataBytesToWrite	Specifies the number of data bytes in the write data buffer, which contains all the specified bits to be written to an external device. Valid range 1 to 65535 ie 64K bytes.
pWaitDataWriteComplete	Pointer to the structure that contains the data that controls whether the FT2232D device should wait until all data bytes have been written to an external device, before returning.

pHighPinsWriteActiveStates	Pointer to the structure that contains which of the 8 general purpose higher input/output pins (GPIOH1 – GPIOH8) of a FT2232H dual hi-speed device, are to be used during a write to an external device. Each GPIO pin that is to be used during a write to an external device must have been previously configured as an output pin, see section 2.1.21. Note: The 8 general purpose higher input/output pins (GPIOH1 – GPIOH8) do not physically exist on the FT4232H quad hi-speed device.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_NULL_INITIAL_CONDITION_BUFFER_POINTER
FTC_INVALID_NUMBER_CONTROL_BITS
FTC_NULL_WRITE_CONTROL_BUFFER_POINTER
FTC_INVALID_NUMBER_CONTROL_BYTES
FTC_NUMBER_CONTROL_BYTES_TOO_SMALL
FTC_INVALID_NUMBER_WRITE_DATA_BITS
FTC_NULL_WRITE_DATA_BUFFER_POINTER
FTC_INVALID_NUMBER_WRITE_DATA_BYTES
FTC_NUMBER_WRITE_DATA_BYTES_TOO_SMALL
FTC_NULL_WAIT_DATA_WRITE_BUFFER_POINTER
FTC_NULL_OUTPUT_PINS_BUFFER_POINTER
FTC_INVALID_INIT_CLOCK_PIN_STATE
FTC_INVALID_FT2232D_CHIP_SELECT_PIN
FTC_INVALID_FT2232D_DATA_WRITE_COMPLETE_PIN
FTC_DATA_WRITE_COMPLETE_TIMEOUT
FTC_INVALID_CONFIGURATION_HIGHER_GPIO_PIN
FTC_FAILED_TO_COMPLETE_COMMAND
FTC_IO_ERROR

Example:

typedef struct FTC_Init_Condition {

BOOL	bClockPinState;	Set clock pin output low(FALSE), high(TRUE)
BOOL	bDataOutPinState;	Set data out pin output low(FALSE), high(TRUE)
BOOL	bChipSelectPinState;	Set chip select pin to disable, output low(FALSE), high(TRUE)
DWORD	dwChipSelectPin	Specifies which pin on the FT2232H dual hi-speed device or FT4232H quad hi-speed device, will be used as the chip select pin. **<u>Valid FT2232H/FT4232H Pins</u>** ADBUS3ChipSelect ADBUS4GPIOL1 ADBUS5GPIOL2 ADBUS6GPIOL3 ADBUS7GPIOL4

} FTC_INIT_CONDITION *PFTC_INIT_CONDITION

```
#define MAX_WRITE_CONTROL_BYTES_BUFFER_SIZE  256      // 256 bytes

typedef BYTE WriteControlByteBuffer[MAX_WRITE_CONTROL_BYTES_BUFFER_SIZE];
typedef WriteControlByteBuffer  *PWriteControlByteBuffer;

#define MAX_WRITE_DATA_BYTES_BUFFER_SIZE                65536  // 64K bytes

typedef BYTE WriteDataByteBuffer[MAX_WRITE_DATA_BYTES_BUFFER_SIZE];
typedef WriteDataByteBuffer  *PWriteDataByteBuffer;


typedef struct FTC_Wait_Data_Write {
```

BOOL	bWaitDataWriteComplete;	Wait until all data bytes have been written to an External device, wait(TRUE), do not wait(FALSE).
DWORD	dwWaitDataWritePin;	Specifies which pin on the FT2232H dual hi-speed device, indicates, when all the data bytes have been written to an external device. If one of the 8 higher GPIO pins (GPIOH1 – GPIOH8) is selected, it must have been previously configured as an input pin, see section 2.1.21.

Valid FT2232H Pins

ADBUS2DataIn

ACBUS0GPIOH1

ACBUS1GPIOH2

ACBUS2GPIOH3

ACBUS3GPIOH4

ACBUS4GPIOH5

ACBUS5GPIOH6

ACBUS6GPIOH7

ACBUS7GPIOH8

BOOL bDataWriteCompleteState; Specifies what state indicates that all data bytes have been written to an external device, low(FALSE), high(TRUE).

DWORD	dwDataWriteTimeoutmSecs;	Timeout interval in milliseconds to wait for all data bytes to be written to an external device.

```
} FTC_WAIT_DATA_WRITE *PFTC_WAIT_DATA_WRITE

typedef struct FTH_Higher_Output_Pins {
```

BOOL	bPin1State;	Set pin1 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin1ActiveState;	Indicates the state, pin1 must be set to during a write to an external device, low(FALSE), high(TRUE)
BOOL	bPin2State;	Set pin2 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin2ActiveState;	Indicates the state, pin2 must be set to during a

		write to an external device, low(FALSE), high(TRUE)
BOOL	bPin3State;	Set pin3 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin3ActiveState;	Indicates the state, pin3 must be set to during a write to an external device, low(FALSE), high(TRUE)
BOOL	bPin4State;	Set pin4 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin4ActiveState;	Indicates the state, pin4 must be set to during a write to an external device, low(FALSE), high(TRUE)
BOOL	bPin5State;	Set pin5 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin5ActiveState;	Indicates the state, pin5 must be set to during a write to an external device, low(FALSE), high(TRUE)
BOOL	bPin6State;	Set pin6 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin6ActiveState;	Indicates the state, pin6 must be set to during a write to an external device, low(FALSE), high(TRUE)
BOOL	bPin7State;	Set pin7 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin7ActiveState;	Indicates the state, pin7 must be set to during a write to an external device, low(FALSE), high(TRUE)
BOOL	bPin8State;	Set pin8 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin8ActiveState;	Indicates the state, pin8 must be set to during a write to an external device, low(FALSE), high(TRUE)

} FTH_HIGHER_OUTPUT_PINS *PFTH_HIGHER_OUTPUT_PINS

2.1.26 SPI_Read

FTC_STATUS SPI_Read (FTC_HANDLE ftHandle, PFTC_INIT_CONDITION pReadStartCondition, BOOL bClockOutControBitsMSBFirst, BOOL bClockOutControBitsPosEdge, DWORD dwNumControlBitsToWrite, PWriteControlByteBuffer pWriteControlBuffer, DWORD dwNumControlBytesToWrite, BOOL bClockInDataBitsMSBFirst, BOOL bClockInDataBitsPosEdge, DWORD dwNumDataBitsToRead, PReadDataByteBuffer pReadDataBuffer, LPDWORD lpdwNumDataBytesReturned, PFTC_HIGHER_OUTPUT_PINS pHighPinsReadActiveStates)

This function reads data from an external device ie a device attached to a FT2232D dual device. A FT2232D dual device communicates with an external device by simulating the SPI synchronous protocol.

Parameters

ftHandle	Handle of a FT2232D dual device.
pReadStartCondition	Pointer to the structure that contains the start output states(low or high) of the clock, data out and signal out/chip select pins of the FT2232D dual device.
bClockOutControlBitsMSBFirst	Clock out control bits Most Significant Bit(MSB) first(TRUE), clock out control bits Least Significant Bit(LSB) first(FALSE)
bClockOutControlBitsPosEdge	Clock out control bits on positive clock edge(TRUE), clock out control bits on negative clock edge(FALSE)
dwNumControlBitsToWrite	Specifies the number of control bits to be written to an external device. Valid range 0 to 2040. 2040 bits is equivalent to 255 bytes.
pWriteControlBuffer	Pointer to buffer that contains the control data to be written to an external device. Listed below is an example of control and address bytes: Control Address byte 1, Control Address byte 2
dwNumControlBytesToWrite	Specifies the number of control bytes in the write control buffer, which contains all the specified bits to be written to an external device. Valid range 1 to 255 bytes.
bClockInDataBitsMSBFirst	Clock in data bits Most Significant Bit(MSB) first(TRUE), clock in data bits Least Significant Bit(LSB) first(FALSE)
bClockInDataBitsPosEdge	Clock in data bits on positive clock edge(TRUE), clock in data bits on negative clock edge(FALSE)
dwNumDataBitsToRead	Specifies the number of bits to be read from an external device. Valid range 2 to 524280. 524280 bits is equivalent to 64K bytes.
pReadDataBuffer	Pointer to buffer that returns the data read from an external device. Size of buffer should be set to 65535.
lpdwNumDataBytesReturned	Pointer to a variable of type DWORD which receives the actual number of data bytes read from an external device. These bytes contain the specified number of bits read from an external device.
pHighPinsReadActiveStates	Pointer to the structure that contains which of the 4 general purpose higher input/output pins (GPIOH1 - GPIOH4) of a FT2232D dual device, are to be used during a read from an external device. Each GPIO pin that is to be used during a read from an external device must have been previously configured as an input pin, see section 2.1.20.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_NULL_INITIAL_CONDITION_BUFFER_POINTER
FTC_INVALID_NUMBER_CONTROL_BITS
FTC_NULL_WRITE_CONTROL_BUFFER_POINTER
FTC_INVALID_NUMBER_CONTROL_BYTES
FTC_NUMBER_CONTROL_BYTES_TOO_SMALL
FTC_INVALID_NUMBER_READ_DATA_BITS
FTC_NULL_READ_DATA_BUFFER_POINTER
FTC_NULL_OUTPUT_PINS_BUFFER_POINTER
FTC_INVALID_INIT_CLOCK_PIN_STATE
FTC_INVALID_FT2232D_CHIP_SELECT_PIN
FTC_INVALID_CONFIGURATION_HIGHER_GPIO_PIN
FTC_FAILED_TO_COMPLETE_COMMAND
FTC_IO_ERROR

Example:

typedef struct FTC_Init_Condition {

BOOL	bClockPinState;	Set clock pin output low(FALSE), high(TRUE)
BOOL	bDataOutPinState;	Set data out pin output low(FALSE), high(TRUE)
BOOL	bChipSelectPinState;	Set chip select pin to disable, output low(FALSE), high(TRUE)
DWORD	dwChipSelectPin	Specifies which pin on the FT2232D dual device, will be used as the chip select pin. **<u>Valid FT2232D Pins</u>** ADBUS3ChipSelect ADBUS4GPIOL1 ADBUS5GPIOL2 ADBUS6GPIOL3 ADBUS7GPIOL4

} FTC_INIT_CONDITION *PFTC_INIT_CONDITION

```
#define MAX_WRITE_CONTROL_BYTES_BUFFER_SIZE  256      // 256 bytes

typedef BYTE WriteControlByteBuffer[MAX_WRITE_CONTROL_BYTES_BUFFER_SIZE];
typedef WriteControlByteBuffer  *PWriteControlByteBuffer;

#define MAX_READ_DATA_BYTES_BUFFER_SIZE                65536  // 64K bytes

typedef BYTE ReadDataByteBuffer[MAX_READ_DATA_BYTES_BUFFER_SIZE];
typedef ReadDataByteBuffer  *PReadDataByteBuffer;
```

typedef struct FTC_Higher_Output_Pins {

BOOL	bPin1State;	Set pin1 to be used during a read from an external device, use(TRUE), not use(FALSE)
BOOL	bPin1ActiveState;	Indicates the state, pin1 must be set to during a read from an external device, low(FALSE), high(TRUE)
BOOL	bPin2State;	Set pin2 to be used during a read from an external device, use(TRUE), not use(FALSE)
BOOL	bPin2ActiveState;	Indicates the state, pin2 must be set to during a read from an external device, low(FALSE), high(TRUE)
BOOL	bPin3State;	Set pin3 to be used during a read from an external device, use(TRUE), not use(FALSE)
BOOL	bPin3ActiveState;	Indicates the state, pin3 must be set to during a read from an external device, low(FALSE), high(TRUE)
BOOL	bPin4State;	Set pin4 to be used during a read from an external device, use(TRUE), not use(FALSE)

BOOL	bPin4ActiveState;	Indicates the state, pin4 must be set to during a read from an external device, low(FALSE), high(TRUE)

} FTC_HIGHER_OUTPUT_PINS *PFTC_HIGHER_OUTPUT_PINS

2.1.27 SPI_ReadHiSpeedDevice

FTC_STATUS SPI_ReadHiSpeedDevice (FTC_HANDLE ftHandle, PFTC_INIT_CONDITION pReadStartCondition, BOOL bClockOutControBitsMSBFirst, BOOL bClockOutControBitsPosEdge, DWORD dwNumControlBitsToWrite, PWriteControlByteBuffer pWriteControlBuffer, DWORD dwNumControlBytesToWrite, BOOL bClockInDataBitsMSBFirst, BOOL bClockInDataBitsPosEdge, DWORD dwNumDataBitsToRead, PReadDataByteBuffer pReadDataBuffer, LPDWORD lpdwNumDataBytesReturned, PFTH_HIGHER_OUTPUT_PINS pHighPinsReadActiveStates)

This function reads data to an external device ie a device attached to a FT2232H dual hi-speed device or FT4232H quad hi-speed device. A FT2232H dual hi-speed device or FT4232H quad hi-speed device communicates with an external device by simulating the SPI synchronous protocol.

Parameters

ftHandle	Handle of a FT2232H dual hi-speed device or FT4232H quad hi-speed device.
pReadStartCondition	Pointer to the structure that contains the start output states(low or high) of the clock, data out and signal out/chip select pins of the FT2232H dual hi-speed device or FT4232H quad hi-speed device.
bClockOutControlBitsMSBFirst	Clock out control bits Most Significant Bit(MSB) first(TRUE), clock out control bits Least Significant Bit(LSB) first(FALSE)
bClockOutControlBitsPosEdge	Clock out control bits on positive clock edge(TRUE), clock out control bits on negative clock edge(FALSE)
dwNumControlBitsToWrite	Specifies the number of control bits to be written to an external device. Valid range 0 to 2040. 2040 bits is equivalent to 255 bytes.
pWriteControlBuffer	Pointer to buffer that contains the control data to be written to an external device. Listed below is an example of control and address bytes: Control Address byte 1, Control Address byte 2
dwNumControlBytesToWrite	Specifies the number of control bytes in the write control buffer, which contains all the specified bits to be written to an external device. Valid range 1 to 255 bytes.
bClockInDataBitsMSBFirst	Clock in data bits Most Significant Bit(MSB) first(TRUE), clock in data bits Least Significant Bit(LSB) first(FALSE)
bClockInDataBitsPosEdge	Clock in data bits on positive clock edge(TRUE), clock in data bits on negative clock edge(FALSE)
dwNumDataBitsToRead	Specifies the number of bits to be read from an external device. valid range 2 to 524280. 524280 bits is equivalent to 64K bytes.
pReadDataBuffer	Pointer to buffer that returns the data read from an external device. Size of buffer should be set to 65535.
lpdwNumDataBytesReturned	Pointer to a variable of type DWORD which receives the actual number of data bytes read from an external device. These bytes contain the specified number of bits read from an external device.

pHighPinsReadActiveStates	Pointer to the structure that contains which of the 8 general purpose higher input/output pins (GPIOH1 – GPIOH8) of a FT2232H dual hi-speed device, are to be used during a read from an external device. Each GPIO pin that is to be used during a read from an external device must have been previously configured as an input pin, see section 2.1.21. Note: The 8 general purpose higher input/output pins (GPIOH1 – GPIOH8) do not physically exist on the FT4232H quad hi-speed device.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_NULL_INITIAL_CONDITION_BUFFER_POINTER
FTC_INVALID_NUMBER_CONTROL_BITS
FTC_NULL_WRITE_CONTROL_BUFFER_POINTER
FTC_INVALID_NUMBER_CONTROL_BYTES
FTC_NUMBER_CONTROL_BYTES_TOO_SMALL
FTC_INVALID_NUMBER_READ_DATA_BITS
FTC_NULL_READ_DATA_BUFFER_POINTER
FTC_NULL_OUTPUT_PINS_BUFFER_POINTER
FTC_INVALID_INIT_CLOCK_PIN_STATE
FTC_INVALID_FT2232D_CHIP_SELECT_PIN
FTC_INVALID_CONFIGURATION_HIGHER_GPIO_PIN
FTC_FAILED_TO_COMPLETE_COMMAND
FTC_IO_ERROR

Example:

```
typedef struct FTC_Init_Condition {

BOOL          bClockPinState;Set clock pin output low(FALSE), high(TRUE)

BOOL          bDataOutPinState;       Set data out pin output low(FALSE), high(TRUE)

BOOL          bChipSelectPinState;    Set chip select pin to disable, output low(FALSE),
                                      high(TRUE)
DWORD         dwChipSelectPin         Specifies which pin on the FT2232H dual hi-speed
                                      device or FT4232H quad hi-speed device, will be
                                      used as the chip select pin.
                                      Valid FT2232H/FT4232H Pins
                                      ADBUS3ChipSelect
                                      ADBUS4GPIOL1
                                      ADBUS5GPIOL2
                                      ADBUS6GPIOL3
                                      ADBUS7GPIOL4
} FTC_INIT_CONDITION *PFTC_INIT_CONDITION

#define MAX_WRITE_CONTROL_BYTES_BUFFER_SIZE  256      // 256 bytes

typedef BYTE WriteControlByteBuffer[MAX_WRITE_CONTROL_BYTES_BUFFER_SIZE];
typedef WriteControlByteBuffer  *PWriteControlByteBuffer;

#define MAX_READ_DATA_BYTES_BUFFER_SIZE                  65536  // 64K bytes

typedef BYTE ReadDataByteBuffer[MAX_READ_DATA_BYTES_BUFFER_SIZE];
typedef ReadDataByteBuffer  *PReadDataByteBuffer;

typedef struct FTH_Higher_Output_Pins {

 BOOL         bPin1State;             Set pin1 to be used during a read from an
                                      external device, use(TRUE), not use(FALSE)
 BOOL         bPin1ActiveState;       Indicates the state, pin1 must be set to during a
                                      read from an external device, low(FALSE),
                                      high(TRUE)
 BOOL         bPin2State;             Set pin2 to be used during a read from an
                                      external device, use(TRUE), not use(FALSE)
 BOOL         bPin2ActiveState;       Indicates the state, pin2 must be set to during a
                                      read from an external device, low(FALSE),
                                      high(TRUE)
 BOOL         bPin3State;             Set pin3 to be used during a read from an
                                      external device, use(TRUE), not use(FALSE)
 BOOL         bPin3ActiveState;       Indicates the state, pin3 must be set to during a
                                      read from an external device, low(FALSE),
                                      high(TRUE)
 BOOL         bPin4State;             Set pin4 to be used during a read from an
                                      external device, use(TRUE), not use(FALSE)
```

BOOL	bPin4ActiveState;	Indicates the state, pin4 must be set to during a read from an external device, low(FALSE), high(TRUE)
BOOL	bPin5State;	Set pin5 to be used during a read from an external device, use(TRUE), not use(FALSE)
BOOL	bPin5ActiveState;	Indicates the state, pin5 must be set to during a read from an external device, low(FALSE), high(TRUE)
BOOL	bPin6State;	Set pin6 to be used during a read from an external device, use(TRUE), not use(FALSE)
BOOL	bPin6ActiveState;	Indicates the state, pin6 must be set to during a read from an external device, low(FALSE), high(TRUE)
BOOL	bPin7State;	Set pin7 to be used during a read from an external device, use(TRUE), not use(FALSE)
BOOL	bPin7ActiveState;	Indicates the state, pin7 must be set to during a read from an external device, low(FALSE), high(TRUE)
BOOL	bPin8State;	Set pin8 to be used during a read from an external device, use(TRUE), not use(FALSE)
BOOL	bPin8ActiveState;	Indicates the state, pin8 must be set to during a read from an external device, low(FALSE), high(TRUE)

} FTH_HIGHER_OUTPUT_PINS *PFTH_HIGHER_OUTPUT_PINS

2.1.28 SPI_ClearDeviceCmdSequence

FTC_STATUS SPI_ClearDeviceCmdSequence(FTC_HANDLE ftHandle)

This function clears the sequence of commands and associated data from the internal command buffer associated with a FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device.

Parameters

ftHandle	Handle of a FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE

2.1.29 SPI_AddDeviceWriteCmd

FTC_STATUS SPI_AddDeviceWriteCmd (FTC_HANDLE ftHandle, PFTC_INIT_CONDITION pWriteStartCondition, BOOL bClockOutDataBitsMSBFirst, BOOL DWORD bClockOutDataBitsPosEdge, DWORD dwNumControlBitsToWrite, PWriteControlByteBuffer pWriteControlBuffer, DWORD dwNumControlBytesToWrite, BOOL bWriteDataBits, DWORD dwNumDataBitsToWrite, PWriteDataByteBuffer pWriteDataBuffer, DWORD dwNumDataBytesToWrite, PFTC_HIGHER_OUTPUT_PINS pHighPinsWriteActiveStates)

This function adds a write command and associated data to the internal command buffer(size 131070 ie 128K bytes) of a FT2232D dual device. This enables a programmer to build up a sequence of commands ie write and read, before executing the sequence of commands, see section 2.1.33.

Warning: while constructing a sequence of commands, do not invoke SPI_Write or SPI_Read functions, as this will clear the sequence of commands and associated data from the internal command buffer.

Parameters

ftHandle	Handle of a FT2232D dual device.
pWriteStartCondition	Pointer to the structure that contains the start output states(low or high) of the clock, data out and signal out/chip select pins of the FT2232D dual device.
bClockOutDataBitsMSBFirst	Clock out control and data bits Most Significant Bit(MSB) first(TRUE), clock out control and data bits Least Significant Bit(LSB) first(FALSE)
bClockOutDataBitsPosEdge	Clock out control and data bits on positive clock edge(TRUE), clock out control and data bits on negative clock edge(FALSE)
dwNumControlBitsToWrite	Specifies the number of control bits to be written to an external device. Valid range 2 to 2040. 2040 bits is equivalent to 255 bytes.
pWriteControlBuffer	Pointer to buffer that contains the control data to be written to an external device. Listed below are three examples of control and address bytes: Two Control bytes Control Address byte 1, Control Address byte 2 Two Control bytes, Control Address byte 1, Control Address byte 2
dwNumControlBytesToWrite	Specifies the number of control bytes in the write control buffer, which contains all the specified bits to be written to an external device. Valid range 1 to 255 bytes.
bWriteDataBits	Write data bits to an external device(TRUE), do not write any data bits to an external device(FALSE)
dwNumDataBitsToWrite	Specifies the number of data bits to be written to an external device. Valid range 2 to 524280. 524280 bits is equivalent to 64K bytes.
pWriteDataBuffer	Pointer to buffer that contains the data to be written to an external device.
dwNumDataBytesToWrite	Specifies the number of data bytes in the write data buffer, which contains all the specified bits to be written to an external device. Valid range 1 to 65535 ie 64K bytes.

pHighPinsWriteActiveStates	Pointer to the structure that contains which of the 4 general purpose higher input/output pins (GPIOH1 – GPIOH4) of a FT2232D dual device, are to be used during a write to an external device. Each GPIO pin that is to be used during a write to an external device must have been previously configured as an output pin, see section 2.1.20.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_NULL_INITIAL_CONDITION_BUFFER_POINTER
FTC_INVALID_NUMBER_CONTROL_BITS
FTC_NULL_WRITE_CONTROL_BUFFER_POINTER
FTC_INVALID_NUMBER_CONTROL_BYTES
FTC_NUMBER_CONTROL_BYTES_TOO_SMALL
FTC_INVALID_NUMBER_DATA_BITS
FTC_NULL_WRITE_DATA_BUFFER_POINTER
FTC_INVALID_NUMBER_DATA_BYTES
FTC_NUMBER_DATA_BYTES_TOO_SMALL
FTC_INVALID_FT2232D_DATA_WRITE_COMPLETE_PIN
FTC_NULL_OUTPUT_PINS_BUFFER_POINTER
FTC_INVALID_INIT_CLOCK_PIN_STATE
FTC_INVALID_FT2232D_CHIP_SELECT_PIN
FTC_DATA_WRITE_COMPLETE_TIMEOUT
FTC_INVALID_FT2232D_DATA_WRITE_COMPLETE_PIN
FTC_INVALID_CONFIGURATION_HIGHER_GPIO_PIN
FTC_COMMAND_SEQUENCE_BUFFER_FULL

Example:

typedef struct FTC_Init_Condition {

BOOL	bClockPinState;	Set clock pin output low(FALSE), high(TRUE)
BOOL	bDataOutPinState;	Set data out pin output low(FALSE), high(TRUE)
BOOL	bChipSelectPinState;	Set chip select pin to disable, output low(FALSE), high(TRUE)
DWORD	dwChipSelectPin	Specifies which pin on the FT2232D dual device, will be used as the chip select pin.
		<u>Valid FT2232D Pins</u>
		ADBUS3ChipSelect
		ADBUS4GPIOL1
		ADBUS5GPIOL2
		ADBUS6GPIOL3
		ADBUS7GPIOL4

} FTC_INIT_CONDITION *PFTC_INIT_CONDITION

```
#define MAX_WRITE_CONTROL_BYTES_BUFFER_SIZE  256      // 256 bytes

typedef BYTE WriteControlByteBuffer[MAX_WRITE_CONTROL_BYTES_BUFFER_SIZE];
typedef WriteControlByteBuffer  *PWriteControlByteBuffer;

#define MAX_WRITE_DATA_BYTES_BUFFER_SIZE                 65536   // 64K bytes

typedef BYTE WriteDataByteBuffer[MAX_WRITE_DATA_BYTES_BUFFER_SIZE];
typedef WriteDataByteBuffer  *PWriteDataByteBuffer;
```

typedef struct FTC_Wait_Data_Write {

BOOL	bWaitDataWriteComplete;	Wait until all data bytes have been written to an External device, wait(TRUE), do not wait(FALSE).
DWORD	dwWaitDataWritePin;	Specifies which pin on the FT2232D dual device, indicates, when all the data bytes have been written to an external device. If one of the 4 higher GPIO pins (GPIOH1 – GPIOH4) is selected, it must have been previously configured as an input pin, see section 2.1.20.

Valid FT2232D Pins

ADBUS2DataIn

ACBUS0GPIOH1

ACBUS1GPIOH2

ACBUS2GPIOH3

ACBUS3GPIOH4

BOOL bDataWriteCompleteState; Specifies what state indicates that all data bytes have been written to an external device, low(FALSE), high(TRUE).

DWORD	dwDataWriteTimeoutmSecs;	Timeout interval in milliseconds to wait for all data bytes to be written to an external device.

} FTC_WAIT_DATA_WRITE *PFTC_WAIT_DATA_WRITE

typedef struct FTC_Higher_Output_Pins {

BOOL	bPin1State;	Set pin1 to be used as an output during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin1ActiveState;	Indicates the state pin1 must be set to during a write to an external device, low(FALSE), high(TRUE)
BOOL	bPin2State;	Set pin2 to be used as an output during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin2ActiveState;	Indicates the state pin2 must be set to during a write to an external device, low(FALSE), high(TRUE)
BOOL	bPin3State;	Set pin3 to be used as an output during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin3ActiveState;	Indicates the state pin3 must be set to during a write to an external device, low(FALSE), high(TRUE)
BOOL	bPin4State;	Set pin4 to be used as an output during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin4ActiveState;	Indicates the state pin4 must be set to during a write to an external device, low(FALSE), high(TRUE)

} FTC_HIGHER_OUTPUT_PINS *PFTC_HIGHER_OUTPUT_PINS

2.1.30 SPI_AddHiSpeedDeviceWriteCmd

FTC_STATUS SPI_AddHiSpeedDeviceWriteCmd (FTC_HANDLE ftHandle, PFTC_INIT_CONDITION pWriteStartCondition, BOOL bClockOutDataBitsMSBFirst, BOOL DWORD bClockOutDataBitsPosEdge, DWORD dwNumControlBitsToWrite, PWriteControlByteBuffer pWriteControlBuffer, DWORD dwNumControlBytesToWrite, BOOL bWriteDataBits, DWORD dwNumDataBitsToWrite, PWriteDataByteBuffer pWriteDataBuffer, DWORD dwNumDataBytesToWrite, PFTH_HIGHER_OUTPUT_PINS pHighPinsWriteActiveStates)

This function adds a write command and associated data to the internal command buffer(size 131070 ie 128K bytes) of a FT2232H dual hi-speed device or FT4232H quad hi-speed device. This enables a programmer to build up a sequence of commands ie write and read, before executing the sequence of commands, see section 2.1.33.

Warning: while constructing a sequence of commands, do not invoke SPI_Write or SPI_Read functions, as this will clear the sequence of commands and associated data from the internal command buffer.

Parameters

ftHandle	Handle of a FT2232H dual hi-speed device or FT4232H quad hi-speed device.
pWriteStartCondition	Pointer to the structure that contains the start output states(low or high) of the clock, data out and signal out/chip select pins of the FT2232H dual hi-speed device or FT4232H quad hi-speed device.
bClockOutDataBitsMSBFirst	Clock out control and data bits Most Significant Bit(MSB) first(TRUE), clock out control and data bits Least Significant Bit(LSB) first(FALSE)
bClockOutDataBitsPosEdge	Clock out control and data bits on positive clock edge(TRUE), clock out control and data bits on negative clock edge(FALSE)
dwNumControlBitsToWrite	Specifies the number of control bits to be written to an external device. Valid range 2 to 2040. 2040 bits is equivalent to 255 bytes.
pWriteControlBuffer	Pointer to buffer that contains the control data to be written to an external device. Listed below are three examples of control and address bytes: Two Control bytes Control Address byte 1, Control Address byte 2 Two Control bytes, Control Address byte 1, Control Address byte 2
dwNumControlBytesToWrite	Specifies the number of control bytes in the write control buffer, which contains all the specified bits to be written to an external device. Valid range 1 to 255 bytes.
bWriteDataBits	Write data bits to an external device(TRUE), do not write any data bits to an external device(FALSE)
dwNumDataBitsToWrite	Specifies the number of data bits to be written to an external device. Valid range 2 to 524280. 524280 bits is equivalent to 64K bytes.
pWriteDataBuffer	Pointer to buffer that contains the data to be written to an external device.

dwNumDataBytesToWrite	Specifies the number of data bytes in the write data buffer, which contains all the specified bits to be written to an external device. Valid range 1 to 65535 ie 64K bytes.
pHighPinsWriteActiveStates	Pointer to the structure that contains which of the 8 general purpose higher input/output pins (GPIOH1 – GPIOH8) of a FT2232H dual hi- speed device, are to be used during a write to an external device. Each GPIO pin that is to be used during a write to an external device must have been previously configured as an output pin, see section 2.1.21. Note: The 8 general purpose higher input/output pins (GPIOH1 – GPIOH8) do not physically exist on the FT4232H quad hi-speed device.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_NULL_INITIAL_CONDITION_BUFFER_POINTER
FTC_INVALID_NUMBER_CONTROL_BITS
FTC_NULL_WRITE_CONTROL_BUFFER_POINTER
FTC_INVALID_NUMBER_CONTROL_BYTES
FTC_NUMBER_CONTROL_BYTES_TOO_SMALL
FTC_INVALID_NUMBER_DATA_BITS
FTC_NULL_WRITE_DATA_BUFFER_POINTER
FTC_INVALID_NUMBER_DATA_BYTES
FTC_NUMBER_DATA_BYTES_TOO_SMALL
FTC_INVALID_FT2232D_DATA_WRITE_COMPLETE_PIN
FTC_NULL_OUTPUT_PINS_BUFFER_POINTER
FTC_INVALID_INIT_CLOCK_PIN_STATE
FTC_INVALID_FT2232D_CHIP_SELECT_PIN
FTC_DATA_WRITE_COMPLETE_TIMEOUT
FTC_INVALID_FT2232D_DATA_WRITE_COMPLETE_PIN
FTC_INVALID_CONFIGURATION_HIGHER_GPIO_PIN
FTC_COMMAND_SEQUENCE_BUFFER_FULL

Example:

typedef struct FTC_Init_Condition {

BOOL	bClockPinState;	Set clock pin output low(FALSE), high(TRUE)
BOOL	bDataOutPinState;	Set data out pin output low(FALSE), high(TRUE)
BOOL	bChipSelectPinState;	Set chip select pin to disable, output low(FALSE), high(TRUE)
DWORD	dwChipSelectPin	Specifies which pin on the FT2232H dual hi-speed device Or FT4232H quad hi-speed device, will be used as the chip select pin.

<u>Valid FT2232H/FT4232H Pins</u>

ADBUS3ChipSelect

ADBUS4GPIOL1

ADBUS5GPIOL2

ADBUS6GPIOL3

ADBUS7GPIOL4

} FTC_INIT_CONDITION *PFTC_INIT_CONDITION

#define MAX_WRITE_CONTROL_BYTES_BUFFER_SIZE 256 // 256 bytes

typedef BYTE WriteControlByteBuffer[MAX_WRITE_CONTROL_BYTES_BUFFER_SIZE];
typedef WriteControlByteBuffer *PWriteControlByteBuffer;

#define MAX_WRITE_DATA_BYTES_BUFFER_SIZE 65536 // 64K bytes

typedef BYTE WriteDataByteBuffer[MAX_WRITE_DATA_BYTES_BUFFER_SIZE];
typedef WriteDataByteBuffer *PWriteDataByteBuffer;

typedef struct FTC_Wait_Data_Write {

BOOL bWaitDataWriteComplete;	Wait until all data bytes have been written to an External device, wait(TRUE), do not wait(FALSE).
DWORD dwWaitDataWritePin;	Specifies which pin on the FT2232H dual device, indicates, when all the data bytes have been written to an external device. If one of the 8 higher GPIO pins (GPIOH1 – GPIOH8) is selected, it must have been previously configured as an input pin, see section 2.1.21.

Valid FT2232H Pins

ADBUS2DataIn

ACBUS0GPIOH1

ACBUS1GPIOH2

ACBUS2GPIOH3

ACBUS3GPIOH4

ACBUS4GPIOH5

ACBUS5GPIOH6

ACBUS6GPIOH7

ACBUS7GPIOH8

BOOL bDataWriteCompleteState; Specifies what state indicates that all data bytes have been written to an external device, low(FALSE), high(TRUE).

DWORD	dwDataWriteTimeoutmSecs;	Timeout interval in milliseconds to wait for all data bytes to be written to an external device.

} FTC_WAIT_DATA_WRITE *PFTC_WAIT_DATA_WRITE

typedef struct FTH_Higher_Output_Pins {

BOOL	bPin1State;	Set pin1 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin1ActiveState;	Indicates the state, pin1 must be set to during a write to An external device, low(FALSE), high(TRUE)
BOOL	bPin2State;	Set pin2 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin2ActiveState;	Indicates the state, pin2 must be set to during a

		write to An external device, low(FALSE), high(TRUE)
BOOL	bPin3State;	Set pin3 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin3ActiveState;	Indicates the state, pin3 must be set to during a write to an external device, low(FALSE), high(TRUE)
BOOL	bPin4State;	Set pin4 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin4ActiveState;	Indicates the state, pin4 must be set to during a write to an external device, low(FALSE), high(TRUE)
BOOL	bPin5State;	Set pin5 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin5ActiveState;	Indicates the state, pin5 must be set to during a write to an external device, low(FALSE), high(TRUE)
BOOL	bPin6State;	Set pin6 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin6ActiveState;	Indicates the state, pin6 must be set to during a write to an external device, low(FALSE), high(TRUE)
BOOL	bPin7State;	Set pin7 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin7ActiveState;	Indicates the state, pin7 must be set to during a write to an external device, low(FALSE), high(TRUE)
BOOL	bPin8State;	Set pin8 to be used during a write to an external device, use(TRUE), not use(FALSE)
BOOL	bPin8ActiveState;	Indicates the state, pin8 must be set to during a write to an external device, low(FALSE), high(TRUE)

} FTH_HIGHER_OUTPUT_PINS *PFTH_HIGHER_OUTPUT_PINS

2.1.31 SPI_AddDeviceReadCmd

FTC_STATUS SPI_AddDeviceReadCmd (FTC_HANDLE ftHandle, PFTC_INIT_CONDITION pReadStartCondition, BOOL bClockOutControlBitsMSBFirst, BOOL bClockOutControlBitsPosEdge, DWORD dwNumControlBitsToWrite, PWriteControlByteBuffer pWriteControlBuffer, DWORD dwNumControlBytesToWrite, BOOL bClockInDataBitsMSBFirst, BOOL bClockInDataBitsPosEdge, DWORD dwNumDataBitsToRead, PFTC_HIGHER_OUTPUT_PINS pHighPinsReadActiveStates)

This function adds a read command to the internal command buffer(size 131070 ie 128K bytes) of a FT2232D dual device. This enables a programmer to build up a sequence of commands ie write and read, before executing the sequence of commands, see section 2.1.33.

Warning: while constructing a sequence of commands, do not invoke SPI_Write or SPI_Read functions, as this will clear the sequence of commands and associated data from the internal command buffer.

Parameters

ftHandle	Handle of a FT2232D dual device.
pReadStartCondition	Pointer to the structure that contains the start output states(low or high) of the clock, data out and signal out/chip select pins of the FT2232D dual device.
bClockOutControlBitsMSBFirst	Clock out control bits Most Significant Bit(MSB) first(TRUE), clock out control bits Least Significant Bit(LSB) first(FALSE)
bClockOutControlBitsPosEdge	Clock out control bits on positive clock edge(TRUE), clock out control bits on negative clock edge(FALSE)
dwNumControlBitsToWrite	Specifies the number of control bits to be written to an external device. Valid range 0 to 2040. 2040 bits is equivalent to 255 bytes.
pWriteControlBuffer	Pointer to buffer that contains the control data to be written to an external device. Listed below is an example of control and address bytes: Control Address byte 1, Control Address byte 2
dwNumControlBytesToWrite	Specifies the number of control bytes in the write control buffer, which contains all the specified bits to be written to an external device. Valid range 1 to 255 bytes.
bClockInDataBitsMSBFirst	Clock in data bits Most Significant Bit(MSB) first(TRUE), clock in data bits Least Significant Bit(LSB) first(FALSE)
bClockInDataBitsPosEdge	Clock in data bits on positive clock edge(TRUE), clock in data bits on negative clock edge(FALSE)
dwNumDataBitsToRead	Specifies the number of bits to be read from an external device. Valid range 2 to 524280. 524280 bits is equivalent to 64K bytes.
pHighPinsReadActiveStates	Pointer to the structure that contains which of the 4 general purpose higher input/output pins (GPIOH1 – GPIOH4)of a FT2232D dual device, are to be used during a read from an external device. Each GPIO pin that is to be used during a read from an external device must have been previously configured as an input pin, see section 2.1.20.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_NULL_INITIAL_CONDITION_BUFFER_POINTER
FTC_INVALID_NUMBER_CONTROL_BITS
FTC_NULL_WRITE_CONTROL_BUFFER_POINTER
FTC_INVALID_NUMBER_CONTROL_BYTES
FTC_NUMBER_CONTROL_BYTES_TOO_SMALL
FTC_INVALID_NUMBER_DATA_BITS
FTC_NULL_OUTPUT_PINS_BUFFER_POINTER
FTC_INVALID_INIT_CLOCK_PIN_STATE
FTC_INVALID_FT2232D_CHIP_SELECT_PIN
FTC_INVALID_CONFIGURATION_HIGHER_GPIO_PIN
FTC_COMMAND_SEQUENCE_BUFFER_FULL
FTC_INSUFFICIENT_RESOURCES

Example:

typedef struct FTC_Init_Condition {

BOOL	bClockPinState;	Set clock pin output low(FALSE), high(TRUE)
BOOL	bDataOutPinState;	Set data out pin output low(FALSE), high(TRUE)
BOOL	bChipSelectPinState;	Set chip select pin to disable, output low(FALSE), high(TRUE)
DWORD	dwChipSelectPin	Specifies which pin on the FT2232D dual device, will be used as the chip select pin. **Valid FT2232D Pins** ADBUS3ChipSelect ADBUS4GPIOL1 ADBUS5GPIOL2 ADBUS6GPIOL3 ADBUS7GPIOL4

} FTC_INIT_CONDITION *PFTC_INIT_CONDITION

typedef struct FTC_Higher_Output_Pins {

BOOL	bPin1State;	Set pin1 to be used during a read from an external device, use(TRUE), not use(FALSE)
BOOL	bPin1ActiveState;	Indicates the state, pin1 must be set to during a read from an external device, low(FALSE), high(TRUE)
BOOL	bPin2State;	Set pin2 to be used during a read from an external device, use(TRUE), not use(FALSE)
BOOL	bPin2ActiveState;	Indicates the state, pin2 must be set to during a read from an external device, low(FALSE), high(TRUE)
BOOL	bPin3State;	Set pin3 to be used during a read from an external device, use(TRUE), not use(FALSE)

BOOL	bPin3ActiveState;	Indicates the state, pin3 must be set to during a read from an external device, low(FALSE), high(TRUE)
BOOL	bPin4State;	Set pin4 to be used during a read from an external device, use(TRUE), not use(FALSE)
BOOL	bPin4ActiveState;	Indicates the state, pin4 must be set to during a read from an external device, low(FALSE), high(TRUE)

} FTC_HIGHER_OUTPUT_PINS *PFTC_HIGHER_OUTPUT_PINS

2.1.32 SPI_AddHiSpeedDeviceReadCmd

FTC_STATUS SPI_AddHiSpeedDeviceReadCmd (FTC_HANDLE ftHandle, PFTC_INIT_CONDITION pReadStartCondition, BOOL bClockOutControlBitsMSBFirst, BOOL bClockOutControlBitsPosEdge, DWORD dwNumControlBitsToWrite, PWriteControlByteBuffer pWriteControlBuffer, DWORD dwNumControlBytesToWrite, BOOL bClockInDataBitsMSBFirst, BOOL bClockInDataBitsPosEdge, DWORD dwNumDataBitsToRead, PFTH_HIGHER_OUTPUT_PINS pHighPinsReadActiveStates)

This function adds a read command to the internal command buffer(size 131070 ie 128K bytes) of a FT2232H dual hi-speed device or FT4232H quad hi-speed device. This enables a programmer to build up a sequence of commands ie write and read, before executing the sequence of commands, see section 2.1.33.

Warning: while constructing a sequence of commands, do not invoke SPI_Write or SPI_Read functions, as this will clear the sequence of commands and associated data from the internal command buffer.

Parameters

ftHandle	Handle of a FT2232H dual hi-speed device or FT4232H quad hi-speed device.
pReadStartCondition	Pointer to the structure that contains the start output states(low or high) of the clock, data out and signal out/chip select pins of the FT2232H dual hi-speed device or FT4232H quad hi-speed device.
bClockOutControlBitsMSBFirst	Clock out control bits Most Significant Bit(MSB) first(TRUE), clock out control bits Least Significant Bit(LSB) first(FALSE)
bClockOutControlBitsPosEdge	Clock out control bits on positive clock edge(TRUE), clock out control bits on negative clock edge(FALSE)
dwNumControlBitsToWrite	Specifies the number of control bits to be written to an external device. Valid range 0 to 2040. 2040 bits is equivalent to 255 bytes.
pWriteControlBuffer	Pointer to buffer that contains the control data to be written to an external device. Listed below is an example of control and address bytes: Control Address byte 1, Control Address byte 2
dwNumControlBytesToWrite	Specifies the number of control bytes in the write control buffer, which contains all the specified bits to be written to an external device. Valid range 1 to 255 bytes.
bClockInDataBitsMSBFirst	Clock in data bits Most Significant Bit(MSB) first(TRUE), clock in data bits Least Significant Bit(LSB) first(FALSE)
bClockInDataBitsPosEdge	Clock in data bits on positive clock edge(TRUE), clock in data bits on negative clock edge(FALSE)
dwNumDataBitsToRead	Specifies the number of bits to be read from an external device. Valid range 2 to 524280. 524280 bits is equivalent to 64K bytes.

pHighPinsReadActiveStates	Pointer to the structure that contains which of the 8 general purpose higher input/output pins (GPIOH1 – GPIOH8) of a FT2232H dual device, are to be used during a read from an external device. Each GPIO pin that is to be used during a read from an external device must have been previously configured as an input pin, see section 2.1.21. Note: The 8 general purpose higher input/output pins (GPIOH1 – GPIOH8) do not physically exist on the FT4232H quad hi-speed device.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_NULL_INITIAL_CONDITION_BUFFER_POINTER
FTC_INVALID_NUMBER_CONTROL_BITS
FTC_NULL_WRITE_CONTROL_BUFFER_POINTER
FTC_INVALID_NUMBER_CONTROL_BYTES
FTC_NUMBER_CONTROL_BYTES_TOO_SMALL
FTC_INVALID_NUMBER_DATA_BITS
FTC_NULL_OUTPUT_PINS_BUFFER_POINTER
FTC_INVALID_INIT_CLOCK_PIN_STATE
FTC_INVALID_FT2232D_CHIP_SELECT_PIN
FTC_INVALID_CONFIGURATION_HIGHER_GPIO_PIN
FTC_COMMAND_SEQUENCE_BUFFER_FULL
FTC_INSUFFICIENT_RESOURCES

Example:

typedef struct FTC_Init_Condition {

BOOL	bClockPinState;	Set clock pin output low(FALSE), high(TRUE)
BOOL	bDataOutPinState;	Set data out pin output low(FALSE), high(TRUE)
BOOL	bChipSelectPinState;	Set chip select pin to disable, output low(FALSE), high(TRUE)
DWORD	dwChipSelectPin	Specifies which pin on the FT2232H dual hi-speed device or FT4232H quad hi-speed device, will be used as the chip select pin.
		<u>Valid FT2232H/FT4232H Pins</u>
		ADBUS3ChipSelect
		ADBUS4GPIOL1
		ADBUS5GPIOL2
		ADBUS6GPIOL3
		ADBUS7GPIOL4

} FTC_INIT_CONDITION *PFTC_INIT_CONDITION

typedef struct FTH_Higher_Output_Pins {

BOOL	bPin1State;	Set pin1 to be used during a read from an external device, use(TRUE), not use(FALSE)
BOOL	bPin1ActiveState;	Indicates the state, pin1 must be set to during a read from an external device, low(FALSE), high(TRUE)
BOOL	bPin2State;	Set pin2 to be used during a read from an external device, use(TRUE), not use(FALSE)
BOOL	bPin2ActiveState;	Indicates the state, pin2 must be set to during a read from an external device, low(FALSE), high(TRUE)
BOOL	bPin3State;	Set pin3 to be used during a read from an external device, use(TRUE), not use(FALSE)
BOOL	bPin3ActiveState;	Indicates the state, pin3 must be set to during a read from an external device, low(FALSE), high(TRUE)
BOOL	bPin4State;	Set pin4 to be used during a read from an external device, use(TRUE), not use(FALSE)
BOOL	bPin4ActiveState;	Indicates the state, pin4 must be set to during a read from an external device, low(FALSE), high(TRUE)
BOOL	bPin5State;	Set pin5 to be used during a read from an external device, use(TRUE), not use(FALSE)
BOOL	bPin5ActiveState;	Indicates the state, pin5 must be set to during a read from an external device, low(FALSE), high(TRUE)

BOOL	bPin6State;	Set pin6 to be used during a read from an external device, use(TRUE), not use(FALSE)
BOOL	bPin6ActiveState;	Indicates the state, pin6 must be set to during a read from an external device, low(FALSE), high(TRUE)
BOOL	bPin7State;	Set pin7 to be used during a read from an external device, use(TRUE), not use(FALSE)
BOOL	bPin7ActiveState;	Indicates the state, pin7 must be set to during a read from an external device, low(FALSE), high(TRUE)
BOOL	bPin8State;	Set pin8 to be used during a read from an external device, use(TRUE), not use(FALSE)
BOOL	bPin8ActiveState;	Indicates the state, pin8 must be set to during a read from an external device, low(FALSE), high(TRUE)

} FTH_HIGHER_OUTPUT_PINS *PFTH_HIGHER_OUTPUT_PINS

2.1.33 SPI_ExecuteDeviceCmdSequence

FTC_STATUS SPI_ExecuteDeviceCmdSequence (FTC_HANDLE ftHandle, PReadCmdSequenceDataByteBuffer pReadCmdSequenceDataBuffer, LPDWORD lpdwNumBytesReturned)

This function executes a sequence of commands, stored in the internal command buffer ie write, read, data to/from an external device ie a device attached to a FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device. A FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device communicates with an external device by simulating the SPI synchronous protocol.

Parameters

ftHandle	Handle of a FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device.
pReadCmdSequenceDataBuffer	Pointer to buffer that returns the data read from an external device. Size of buffer should be set to 131071.
lpdwNumBytesReturned	Pointer to a variable of type DWORD which receives the actual number of data bytes read from an external device. These bytes contain the total number of bits, read as specified in the sequence of read and write/read commands.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_INVALID_HANDLE
FTC_NO_COMMAND_SEQUENCE
FTC_NULL_READ_CMDS_DATA_BUFFER_POINTER
FTC_FAILED_TO_COMPLETE_COMMAND
FTC_IO_ERROR

Example:

```
#define MAX_READ_CMDS_DATA_BYTES_BUFFER_SIZE                131071          //
128K bytes

typedef BYTE
ReadCmdSequenceDataByteBuffer[MAX_READ_CMDS_DATA_BYTES_BUFFER_SIZE];
typedef ReadCmdSequenceDataByteBuffer  *PReadCmdSequenceDataByteBuffer;
```

2.1.34 SPI_GetDllVersion

FTC_STATUS SPI_GetDllVersion(LPSTR lpDllVersionBuffer, DWORD dwBufferSize)

This function returns the version of this DLL.

Parameters

lpDllVersionBuffer	Pointer to the buffer that receives the version of this DLL. The string will be NULL terminated.
dwBufferSize	Length of the buffer created for the device name string. Set buffer length to a minimum of 10 characters.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_NULL_DLL_VERSION_BUFFER_POINTER
FTC_DLL_VERSION_BUFFER_TOO_SMALL

2.1.35 SPI_GetErrorCodeString

FTC_STATUS SPI_GetErrorCodeString(LPSTR lpLanguage, FTC_STATUS StatusCode, LPSTR lpErrorMessageBuffer, DWORD dwBufferSize)

This function returns the error message for the specified error code, to be used for display purposes by an application programmer. The error code passed into this function must have been returned from a function within this DLL.

Parameters

lpLanguage	Pointer to a NULL terminated string that contains the language code. Default for this first version the default language will be English(EN).
StatusCode	Status code returned from a previous DLL function call.
lpErrorMessageBuffer	Pointer to the buffer that receives the error message. The error message represents the description of the status code. The string will be NULL terminated. If an unsupported language code or invalid status code is passed in to this function, the returned error message will reflect this.
dwBufferSize	Length of the buffer created for the error message string. Set buffer length to a minimum of 100 characters.

Return Value

Returns FTC_SUCCESS if successful, otherwise the return value will be one of the following error codes:

FTC_NULL_LANGUAGE_CODE_BUFFER_POINTER
FTC_INVALID_LANGUAGE_CODE
FTC_INVALID_STATUS_CODE
FTC_NULL_ERROR_MESSAGE_BUFFER_POINTER
FTC_ERROR_MESSAGE_BUFFER_TOO_SMALL

3 FTCSPI.h

```
/*++

Copyright (c) 2005 Future Technology Devices International Ltd.

Module Name:

    ftcspi.h

Abstract:

    API DLL for FT2232H and FT4232H Hi-Speed Dual Device and FT2232D Dual Device
setup to simulate the
    Serial Peripheral Interface(SPI) synchronous serial protocol.
    FTCSPI library definitions

Environment:

    kernel & user mode

Revision History:

    13/05/05    kra     Created.
    21/08/08    kra     Added new functions for FT2232H and FT4232H hi-speed
devices.

--*/


#ifndef FTCSPI_H
#define FTCSPI_H


// The following ifdef block is the standard way of creating macros
// which make exporting from a DLL simpler.  All files within this DLL
// are compiled with the FTCSPI_EXPORTS symbol defined on the command line.
// This symbol should not be defined on any project that uses this DLL.
// This way any other project whose source files include this file see
// FTCSPI_API functions as being imported from a DLL, whereas this DLL
// sees symbols defined with this macro as being exported.

#ifdef FTCSPI_EXPORTS
#define FTCSPI_API __declspec(dllexport)
#else
#define FTCSPI_API __declspec(dllimport)
#endif

typedef DWORD FTC_HANDLE;
typedef ULONG FTC_STATUS;

// Hi-speed device types
enum {
      FT2232H_DEVICE_TYPE = 1,
      FT4232H_DEVICE_TYPE = 2
};

#define ADBUS3ChipSelect 0
#define ADBUS4GPIOL1 1
#define ADBUS5GPIOL2 2
#define ADBUS6GPIOL3 3
#define ADBUS7GPIOL4 4
```

```
#define ADBUS2DataIn 0
#define ACBUS0GPIOH1 1
#define ACBUS1GPIOH2 2
#define ACBUS2GPIOH3 3
#define ACBUS3GPIOH4 4
#define ACBUS4GPIOH5 5
#define ACBUS5GPIOH6 6
#define ACBUS6GPIOH7 7
#define ACBUS7GPIOH8 8

#define FTC_SUCCESS 0 // FTC_OK
#define FTC_INVALID_HANDLE 1 // FTC_INVALID_HANDLE
#define FTC_DEVICE_NOT_FOUND 2 //FTC_DEVICE_NOT_FOUND
#define FTC_DEVICE_NOT_OPENED 3 //FTC_DEVICE_NOT_OPENED
#define FTC_IO_ERROR 4 //FTC_IO_ERROR
#define FTC_INSUFFICIENT_RESOURCES 5 // FTC_INSUFFICIENT_RESOURCES

#define FTC_FAILED_TO_COMPLETE_COMMAND 20          // cannot change, error code
mapped from FT2232c classes
#define FTC_FAILED_TO_SYNCHRONIZE_DEVICE_MPSSE 21  // cannot change, error code
mapped from FT2232c classes
#define FTC_INVALID_DEVICE_NAME_INDEX 22           // cannot change, error code
mapped from FT2232c classes
#define FTC_NULL_DEVICE_NAME_BUFFER_POINTER 23     // cannot change, error code
mapped from FT2232c classes
#define FTC_DEVICE_NAME_BUFFER_TOO_SMALL 24        // cannot change, error code
mapped from FT2232c classes
#define FTC_INVALID_DEVICE_NAME 25                 // cannot change, error code
mapped from FT2232c classes
#define FTC_INVALID_LOCATION_ID 26                 // cannot change, error code
mapped from FT2232c classes
#define FTC_DEVICE_IN_USE 27                       // cannot change, error code
mapped from FT2232c classes
#define FTC_TOO_MANY_DEVICES 28                    // cannot change, error code
mapped from FT2232c classes

#define FTC_NULL_CHANNEL_BUFFER_POINTER 29         // cannot change, error code
mapped from FT2232h classes
#define FTC_CHANNEL_BUFFER_TOO_SMALL 30            // cannot change, error code
mapped from FT2232h classes
#define FTC_INVALID_CHANNEL 31                     // cannot change, error code
mapped from FT2232h classes
#define FTC_INVALID_TIMER_VALUE 32                 // cannot change, error code
mapped from FT2232h classes

#define FTC_INVALID_CLOCK_DIVISOR 33
#define FTC_NULL_INPUT_BUFFER_POINTER 34
#define FTC_NULL_CHIP_SELECT_BUFFER_POINTER 35
#define FTC_NULL_INPUT_OUTPUT_BUFFER_POINTER 36
#define FTC_NULL_OUTPUT_PINS_BUFFER_POINTER 37
#define FTC_NULL_INITIAL_CONDITION_BUFFER_POINTER 38
#define FTC_NULL_WRITE_CONTROL_BUFFER_POINTER 39
#define FTC_NULL_WRITE_DATA_BUFFER_POINTER 40
#define FTC_NULL_WAIT_DATA_WRITE_BUFFER_POINTER 41
#define FTC_NULL_READ_DATA_BUFFER_POINTER 42
#define FTC_NULL_READ_CMDS_DATA_BUFFER_POINTER 43
#define FTC_INVALID_NUMBER_CONTROL_BITS 44
#define FTC_INVALID_NUMBER_CONTROL_BYTES 45
#define FTC_NUMBER_CONTROL_BYTES_TOO_SMALL 46
#define FTC_INVALID_NUMBER_WRITE_DATA_BITS 47
#define FTC_INVALID_NUMBER_WRITE_DATA_BYTES 48
#define FTC_NUMBER_WRITE_DATA_BYTES_TOO_SMALL 49
```

```
#define FTC_INVALID_NUMBER_READ_DATA_BITS 50
#define FTC_INVALID_INIT_CLOCK_PIN_STATE 51
#define FTC_INVALID_FT2232C_CHIP_SELECT_PIN 52
#define FTC_INVALID_FT2232C_DATA_WRITE_COMPLETE_PIN 53
#define FTC_DATA_WRITE_COMPLETE_TIMEOUT 54
#define FTC_INVALID_CONFIGURATION_HIGHER_GPIO_PIN 55
#define FTC_COMMAND_SEQUENCE_BUFFER_FULL 56
#define FTC_NO_COMMAND_SEQUENCE 57
#define FTC_NULL_CLOSE_FINAL_STATE_BUFFER_POINTER 58
#define FTC_NULL_DLL_VERSION_BUFFER_POINTER 59
#define FTC_DLL_VERSION_BUFFER_TOO_SMALL 60
#define FTC_NULL_LANGUAGE_CODE_BUFFER_POINTER 61
#define FTC_NULL_ERROR_MESSAGE_BUFFER_POINTER 62
#define FTC_ERROR_MESSAGE_BUFFER_TOO_SMALL 63
#define FTC_INVALID_LANGUAGE_CODE 64
#define FTC_INVALID_STATUS_CODE 65

#ifdef __cplusplus
extern "C" {
#endif

FTCSPI_API
FTC_STATUS WINAPI SPI_GetNumDevices(LPDWORD lpdwNumDevices);

FTCSPI_API
FTC_STATUS WINAPI SPI_GetNumHiSpeedDevices(LPDWORD lpdwNumHiSpeedDevices);

FTCSPI_API
FTC_STATUS WINAPI SPI_GetDeviceNameLocID(DWORD dwDeviceNameIndex, LPSTR
lpDeviceNameBuffer, DWORD dwBufferSize, LPDWORD lpdwLocationID);

FTCSPI_API
FTC_STATUS WINAPI SPI_GetHiSpeedDeviceNameLocIDChannel(DWORD dwDeviceNameIndex,
LPSTR lpDeviceNameBuffer, DWORD dwBufferSize, LPDWORD lpdwLocationID, LPSTR
lpChannel, DWORD dwChannelBufferSize, LPDWORD lpdwHiSpeedDeviceType);

FTCSPI_API
FTC_STATUS WINAPI SPI_Open(FTC_HANDLE *pftHandle);

FTCSPI_API
FTC_STATUS WINAPI SPI_OpenEx(LPSTR lpDeviceName, DWORD dwLocationID, FTC_HANDLE
*pftHandle);

FTCSPI_API
FTC_STATUS WINAPI SPI_OpenHiSpeedDevice(LPSTR lpDeviceName, DWORD dwLocationID,
LPSTR lpChannel, FTC_HANDLE *pftHandle);

FTCSPI_API
FTC_STATUS WINAPI SPI_GetHiSpeedDeviceType(FTC_HANDLE ftHandle, LPDWORD
lpdwHiSpeedDeviceType);

FTCSPI_API
FTC_STATUS WINAPI SPI_Close(FTC_HANDLE ftHandle);

typedef struct Ft_Close_Final_State_Pins{
  BOOL  bTCKPinState;
  BOOL  bTCKPinActiveState;
  BOOL  bTDIPinState;
  BOOL  bTDIPinActiveState;
  BOOL  bTMSPinState;
  BOOL  bTMSPinActiveState;
}FTC_CLOSE_FINAL_STATE_PINS, *PFTC_CLOSE_FINAL_STATE_PINS;
```

```
FTCSPI_API
FTC_STATUS WINAPI SPI_CloseDevice(FTC_HANDLE ftHandle,
PFTC_CLOSE_FINAL_STATE_PINS pCloseFinalStatePinsData);

FTCSPI_API
FTC_STATUS WINAPI SPI_InitDevice(FTC_HANDLE ftHandle, DWORD dwClockDivisor);

FTCSPI_API
FTC_STATUS WINAPI SPI_TurnOnDivideByFiveClockingHiSpeedDevice(FTC_HANDLE
ftHandle);

FTCSPI_API
FTC_STATUS WINAPI SPI_TurnOffDivideByFiveClockingHiSpeedDevice(FTC_HANDLE
ftHandle);

FTCSPI_API
FTC_STATUS WINAPI SPI_SetDeviceLatencyTimer(FTC_HANDLE ftHandle, BYTE
timerValue);

FTCSPI_API
FTC_STATUS WINAPI SPI_GetDeviceLatencyTimer(FTC_HANDLE ftHandle, LPBYTE
lpTimerValue);

FTCSPI_API
FTC_STATUS WINAPI SPI_GetClock(DWORD dwClockDivisor, LPDWORD
lpdwClockFrequencyHz);

FTCSPI_API
FTC_STATUS WINAPI SPI_GetHiSpeedDeviceClock(DWORD dwClockDivisor, LPDWORD
lpdwClockFrequencyHz);

FTCSPI_API
FTC_STATUS WINAPI SPI_SetClock(FTC_HANDLE ftHandle, DWORD dwClockDivisor,
LPDWORD lpdwClockFrequencyHz);

FTCSPI_API
FTC_STATUS WINAPI SPI_SetLoopback(FTC_HANDLE ftHandle, BOOL bLoopbackState);

typedef struct Ft_Chip_Select_Pins{
  BOOL  bADBUS3ChipSelectPinState;
  BOOL  bADBUS4GPIOL1PinState;
  BOOL  bADBUS5GPIOL2PinState;
  BOOL  bADBUS6GPIOL3PinState;
  BOOL  bADBUS7GPIOL4PinState;
}FTC_CHIP_SELECT_PINS, *PFTC_CHIP_SELECT_PINS;

typedef struct Ft_Input_Output_Pins{
  BOOL  bPin1InputOutputState;
  BOOL  bPin1LowHighState;
  BOOL  bPin2InputOutputState;
  BOOL  bPin2LowHighState;
  BOOL  bPin3InputOutputState;
  BOOL  bPin3LowHighState;
  BOOL  bPin4InputOutputState;
  BOOL  bPin4LowHighState;
}FTC_INPUT_OUTPUT_PINS, *PFTC_INPUT_OUTPUT_PINS;

FTCSPI_API
FTC_STATUS WINAPI SPI_SetGPIOs(FTC_HANDLE ftHandle, PFTC_CHIP_SELECT_PINS
pChipSelectsDisableStates,
                               PFTC_INPUT_OUTPUT_PINS pHighInputOutputPinsData);

typedef struct FTH_Input_Output_Pins{
  BOOL  bPin1InputOutputState;
  BOOL  bPin1LowHighState;
  BOOL  bPin2InputOutputState;
  BOOL  bPin2LowHighState;
  BOOL  bPin3InputOutputState;
  BOOL  bPin3LowHighState;
  BOOL  bPin4InputOutputState;
  BOOL  bPin4LowHighState;
  BOOL  bPin5InputOutputState;
  BOOL  bPin5LowHighState;
  BOOL  bPin6InputOutputState;
  BOOL  bPin6LowHighState;
  BOOL  bPin7InputOutputState;
  BOOL  bPin7LowHighState;
  BOOL  bPin8InputOutputState;
  BOOL  bPin8LowHighState;
}FTH_INPUT_OUTPUT_PINS, *PFTH_INPUT_OUTPUT_PINS;

FTCSPI_API
FTC_STATUS WINAPI SPI_SetHiSpeedDeviceGPIOs(FTC_HANDLE ftHandle,
PFTC_CHIP_SELECT_PINS pChipSelectsDisableStates,
                                            PFTH_INPUT_OUTPUT_PINS
pHighInputOutputPinsData);

typedef struct Ft_Low_High_Pins{
  BOOL  bPin1LowHighState;
  BOOL  bPin2LowHighState;
  BOOL  bPin3LowHighState;
  BOOL  bPin4LowHighState;
}FTC_LOW_HIGH_PINS, *PFTC_LOW_HIGH_PINS;

FTCSPI_API
FTC_STATUS WINAPI SPI_GetGPIOs(FTC_HANDLE ftHandle, PFTC_LOW_HIGH_PINS
pHighPinsInputData);

typedef struct FTH_Low_High_Pins{
  BOOL  bPin1LowHighState;
  BOOL  bPin2LowHighState;
  BOOL  bPin3LowHighState;
  BOOL  bPin4LowHighState;
  BOOL  bPin5LowHighState;
  BOOL  bPin6LowHighState;
  BOOL  bPin7LowHighState;
  BOOL  bPin8LowHighState;
}FTH_LOW_HIGH_PINS, *PFTH_LOW_HIGH_PINS;

FTCSPI_API
FTC_STATUS WINAPI SPI_GetHiSpeedDeviceGPIOs(FTC_HANDLE ftHandle,
PFTH_LOW_HIGH_PINS pHighPinsInputData);

#define MAX_WRITE_CONTROL_BYTES_BUFFER_SIZE 256    // 256 bytes

typedef BYTE WriteControlByteBuffer[MAX_WRITE_CONTROL_BYTES_BUFFER_SIZE];
typedef WriteControlByteBuffer *PWriteControlByteBuffer;

#define MAX_WRITE_DATA_BYTES_BUFFER_SIZE 65536    // 64k bytes

typedef BYTE WriteDataByteBuffer[MAX_WRITE_DATA_BYTES_BUFFER_SIZE];
typedef WriteDataByteBuffer *PWriteDataByteBuffer;

typedef struct Ft_Init_Condition{
  BOOL  bClockPinState;
  BOOL  bDataOutPinState;
  BOOL  bChipSelectPinState;
  DWORD dwChipSelectPin;
}FTC_INIT_CONDITION, *PFTC_INIT_CONDITION;

typedef struct Ft_Wait_Data_Write{
  BOOL  bWaitDataWriteComplete;
  DWORD dwWaitDataWritePin;
  BOOL  bDataWriteCompleteState;
  DWORD dwDataWriteTimeoutmSecs;
}FTC_WAIT_DATA_WRITE, *PFTC_WAIT_DATA_WRITE;

typedef struct Ft_Higher_Output_Pins{
  BOOL  bPin1State;
  BOOL  bPin1ActiveState;
  BOOL  bPin2State;
  BOOL  bPin2ActiveState;
  BOOL  bPin3State;
  BOOL  bPin3ActiveState;
  BOOL  bPin4State;
  BOOL  bPin4ActiveState;
}FTC_HIGHER_OUTPUT_PINS, *PFTC_HIGHER_OUTPUT_PINS;

FTCSPI_API
FTC_STATUS WINAPI SPI_Write(FTC_HANDLE ftHandle, PFTC_INIT_CONDITION
pWriteStartCondition, BOOL bClockOutDataBitsMSBFirst,
                            BOOL bClockOutDataBitsPosEdge, DWORD
dwNumControlBitsToWrite, PWriteControlByteBuffer pWriteControlBuffer,
                            DWORD dwNumControlBytesToWrite, BOOL bWriteDataBits,
DWORD dwNumDataBitsToWrite, PWriteDataByteBuffer pWriteDataBuffer,
                            DWORD dwNumDataBytesToWrite, PFTC_WAIT_DATA_WRITE
pWaitDataWriteComplete, PFTC_HIGHER_OUTPUT_PINS pHighPinsWriteActiveStates);

typedef struct FTH_Higher_Output_Pins{
  BOOL  bPin1State;
  BOOL  bPin1ActiveState;
  BOOL  bPin2State;
  BOOL  bPin2ActiveState;
  BOOL  bPin3State;
  BOOL  bPin3ActiveState;
  BOOL  bPin4State;
  BOOL  bPin4ActiveState;
  BOOL  bPin5State;
  BOOL  bPin5ActiveState;
  BOOL  bPin6State;
  BOOL  bPin6ActiveState;
  BOOL  bPin7State;
  BOOL  bPin7ActiveState;
  BOOL  bPin8State;
  BOOL  bPin8ActiveState;
}FTH_HIGHER_OUTPUT_PINS, *PFTH_HIGHER_OUTPUT_PINS;

FTCSPI_API
FTC_STATUS WINAPI SPI_WriteHiSpeedDevice(FTC_HANDLE ftHandle,
PFTC_INIT_CONDITION pWriteStartCondition, BOOL bClockOutDataBitsMSBFirst,
                                         BOOL bClockOutDataBitsPosEdge, DWORD
dwNumControlBitsToWrite, PWriteControlByteBuffer pWriteControlBuffer,
                                         DWORD dwNumControlBytesToWrite, BOOL
bWriteDataBits, DWORD dwNumDataBitsToWrite,
                                         PWriteDataByteBuffer pWriteDataBuffer,
DWORD dwNumDataBytesToWrite, PFTC_WAIT_DATA_WRITE pWaitDataWriteComplete,
                                         PFTH_HIGHER_OUTPUT_PINS
pHighPinsWriteActiveStates);
```

```
#define MAX_READ_DATA_BYTES_BUFFER_SIZE 65536    // 64k bytes

typedef BYTE ReadDataByteBuffer[MAX_READ_DATA_BYTES_BUFFER_SIZE];
typedef ReadDataByteBuffer *PReadDataByteBuffer;

FTCSPI_API
FTC_STATUS WINAPI SPI_Read(FTC_HANDLE ftHandle, PFTC_INIT_CONDITION
pReadStartCondition, BOOL bClockOutControlBitsMSBFirst,
                           BOOL bClockOutControlBitsPosEdge, DWORD
dwNumControlBitsToWrite, PWriteControlByteBuffer pWriteControlBuffer,
                           DWORD dwNumControlBytesToWrite, BOOL
bClockInDataBitsMSBFirst, BOOL bClockInDataBitsPosEdge,
                           DWORD dwNumDataBitsToRead, PReadDataByteBuffer
pReadDataBuffer, LPDWORD lpdwNumDataBytesReturned,
                           PFTC_HIGHER_OUTPUT_PINS pHighPinsReadActiveStates);

FTCSPI_API
FTC_STATUS WINAPI SPI_ReadHiSpeedDevice(FTC_HANDLE ftHandle, PFTC_INIT_CONDITION
pReadStartCondition, BOOL bClockOutControlBitsMSBFirst,
                                        BOOL bClockOutControlBitsPosEdge, DWORD
dwNumControlBitsToWrite, PWriteControlByteBuffer pWriteControlBuffer,
                                        DWORD dwNumControlBytesToWrite, BOOL
bClockInDataBitsMSBFirst, BOOL bClockInDataBitsPosEdge,
                                        DWORD dwNumDataBitsToRead,
PReadDataByteBuffer pReadDataBuffer, LPDWORD lpdwNumDataBytesReturned,
                                        PFTH_HIGHER_OUTPUT_PINS
pHighPinsReadActiveStates);

FTCSPI_API
FTC_STATUS WINAPI SPI_ClearDeviceCmdSequence(FTC_HANDLE ftHandle);

FTCSPI_API
FTC_STATUS WINAPI SPI_AddDeviceWriteCmd(FTC_HANDLE ftHandle, PFTC_INIT_CONDITION
pWriteStartCondition, BOOL bClockOutDataBitsMSBFirst,
                                        BOOL bClockOutDataBitsPosEdge, DWORD
dwNumControlBitsToWrite, PWriteControlByteBuffer pWriteControlBuffer,
                                        DWORD dwNumControlBytesToWrite, BOOL
bWriteDataBits, DWORD dwNumDataBitsToWrite,
                                        PWriteDataByteBuffer pWriteDataBuffer,
DWORD dwNumDataBytesToWrite,
                                        PFTC_HIGHER_OUTPUT_PINS
pHighPinsWriteActiveStates);

FTCSPI_API
FTC_STATUS WINAPI SPI_AddHiSpeedDeviceWriteCmd(FTC_HANDLE ftHandle,
PFTC_INIT_CONDITION pWriteStartCondition, BOOL bClockOutDataBitsMSBFirst,
                                               BOOL bClockOutDataBitsPosEdge,
DWORD dwNumControlBitsToWrite, PWriteControlByteBuffer pWriteControlBuffer,
                                               DWORD dwNumControlBytesToWrite,
BOOL bWriteDataBits, DWORD dwNumDataBitsToWrite,
                                               PWriteDataByteBuffer
pWriteDataBuffer, DWORD dwNumDataBytesToWrite,
                                               PFTH_HIGHER_OUTPUT_PINS
pHighPinsWriteActiveStates);

FTCSPI_API
FTC_STATUS WINAPI SPI_AddDeviceReadCmd(FTC_HANDLE ftHandle, PFTC_INIT_CONDITION
pReadStartCondition, BOOL bClockOutControlBitsMSBFirst,
                                       BOOL bClockOutControlBitsPosEdge, DWORD
dwNumControlBitsToWrite, PWriteControlByteBuffer pWriteControlBuffer,
                                       DWORD dwNumControlBytesToWrite, BOOL
bClockInDataBitsMSBFirst, BOOL bClockInDataBitsPosEdge,
                                       DWORD dwNumDataBitsToRead,
PFTC_HIGHER_OUTPUT_PINS pHighPinsReadActiveStates);

FTCSPI_API
FTC_STATUS WINAPI SPI_AddHiSpeedDeviceReadCmd(FTC_HANDLE ftHandle,
PFTC_INIT_CONDITION pReadStartCondition, BOOL bClockOutControlBitsMSBFirst,
                                              BOOL bClockOutControlBitsPosEdge,
DWORD dwNumControlBitsToWrite, PWriteControlByteBuffer pWriteControlBuffer,
                                              DWORD dwNumControlBytesToWrite,
BOOL bClockInDataBitsMSBFirst, BOOL bClockInDataBitsPosEdge,
                                              DWORD dwNumDataBitsToRead,
PFTH_HIGHER_OUTPUT_PINS pHighPinsReadActiveStates);

#define MAX_READ_CMDS_DATA_BYTES_BUFFER_SIZE 131071  // 128K bytes

typedef BYTE
ReadCmdSequenceDataByteBuffer[MAX_READ_CMDS_DATA_BYTES_BUFFER_SIZE];
typedef ReadCmdSequenceDataByteBuffer *PReadCmdSequenceDataByteBuffer;

FTCSPI_API
FTC_STATUS WINAPI SPI_ExecuteDeviceCmdSequence(FTC_HANDLE ftHandle,
PReadCmdSequenceDataByteBuffer pReadCmdSequenceDataBuffer,
                                               LPDWORD lpdwNumBytesReturned);

FTCSPI_API
FTC_STATUS WINAPI SPI_GetDllVersion(LPSTR lpDllVersionBuffer, DWORD
dwBufferSize);

FTCSPI_API
FTC_STATUS WINAPI SPI_GetErrorCodeString(LPSTR lpLanguage, FTC_STATUS
StatusCode,
                                         LPSTR lpErrorMessageBuffer, DWORD
dwBufferSize);


#ifdef __cplusplus
}
#endif
#endif  /* FTCSPI_H */
```

4 Contact Information

Head Office – Glasgow, UK

Future Technology Devices International Limited
Unit 1, 2 Seaward Place, Centurion Business Park
Glasgow G41 1HH
United Kingdom

Tel: +44 (0) 141 429 2777
Fax: +44 (0) 141 429 2758

E-mail (Sales) sales1@ftdichip.com
E-mail (Support) support1@ftdichip.com
E-mail (General Enquiries) admin1@ftdichip.com
Web Site URL http://www.ftdichip.com
Web Shop URL http://www.ftdichip.com

Branch Office – Taipei, Taiwan

Future Technology Devices International Limited (Taiwan)
2F, No 516, Sec. 1 NeiHu Road
Taipei 114
Taiwan, R.O.C.
Tel: +886 (0) 2 8797 1330
Fax: +886 (0) 2 8751 9737

E-mail (Sales) tw.sales1@ftdichip.com
E-mail (Support) tw.support1@ftdichip.com
E-mail (General Enquiries) tw.admin1@ftdichip.com
Web Site URL http://www.ftdichip.com

Branch Office – Hillsboro, Oregon, USA

Future Technology Devices International Limited (USA)
7235 NW Evergreen Parkway, Suite 600
Hillsboro, OR 97123-5803
USA
Tel: +1 (503) 547 0988
Fax: +1 (503) 547 0987

E-Mail (Sales) us.sales@ftdichip.com
E-Mail (Support) us.admin@ftdichip.com
Web Site URL http://www.ftdichip.com

Branch Office – Shanghai, China

Future Technology Devices International Limited (China)
Room 408, 317 Xianxia Road,
ChangNing District,
ShangHai, China

Tel: +86 (21) 62351596
Fax: +86(21) 62351595

E-Mail (Sales): cn.sales@ftdichip.com
E-Mail (Support): cn.support@ftdichip.com
E-Mail (General Enquiries): cn.admin1@ftdichip.com
Web Site URL: http://www.ftdichip.com

Distributor and Sales Representatives

Please visit the Sales Network page of the FTDI Web site for the contact details of our distributor(s) and sales representative(s) in your country.

Appendix A – Revision History

Revision History

Draft	Initial Draft	December, 2008
1.0	Initial Release	21st January, 2008
1.1	Added missing commands	18 March, 2009