



**Future Technology Devices International Ltd.**

## **Application Note AN\_109**

# **Programming Guide for High Speed FTCI2C DLL**

**Document Reference No.: FT\_000110**

**Version 1.1**

**Issue Date: 2009-03-18**

**This document provides details of the function calls required when using the High  
Speed FTCI2C.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](mailto:support1@ftdichip.com) Web: <http://www.ftdichip.com>

Copyright © 2009 Future Technology Devices International Limited

<b>1</b>	<b>Introduction.....</b>	<b>2</b>
<b>2</b>	<b>Application Programming Interface (API).....</b>	<b>3</b>
<b>2.1</b>	<b>Public Functions.....</b>	<b>3</b>
2.1.1	I2C_GetNumDevices.....	3
2.1.2	I2C_GetNumHiSpeedDevices.....	3
2.1.3	I2C_GetDeviceNameLocID.....	4
2.1.4	I2C_GetHiSpeedDeviceNameLocIDChannel.....	5
2.1.5	I2C_Open.....	6
2.1.6	I2C_OpenEx.....	7
2.1.7	I2C_OpenHiSpeedDevice.....	8
2.1.8	I2C_GetHiSpeedDeviceType.....	9
2.1.9	I2C_Close.....	10
2.1.10	I2C_CloseDevice.....	10
2.1.11	I2C_InitDevice.....	11
2.1.12	I2C_TurnOnDivideByFiveClockingHiSpeedDevice.....	12
2.1.13	I2C_TurnOffDivideByFiveClockingHiSpeedDevice.....	12
2.1.14	I2C_TurnOnThreePhaseDataClockingHiSpeedDevice.....	13
2.1.15	I2C_TurnOffThreePhaseDataClockingHiSpeedDevice.....	13
2.1.16	I2C_SetDeviceLatencyTimer.....	14
2.1.17	I2C_GetDeviceLatencyTimer.....	14
2.1.18	I2C_GetClock.....	15
2.1.19	I2C_GetHiSpeedDeviceClock.....	16
2.1.20	I2C_SetClock.....	16
2.1.21	I2C_SetLoopback.....	17
2.1.22	I2C_SetMode.....	19
2.1.23	I2C_SetGPIOs.....	20
2.1.24	I2C_SetHiSpeedDeviceGPIOs.....	21
2.1.25	I2C_GetGPIOs.....	22
2.1.26	I2C_GetHiSpeedDeviceGPIOs.....	24
2.1.27	I2C_Write.....	25
2.1.28	I2C_Read.....	28
2.1.29	I2C_GetDIIVersion.....	30
2.1.30	I2C_GetErrorCodeString.....	30
<b>3</b>	<b>Contact Information.....</b>	<b>32</b>
	<b>Appendix - Revision History.....</b>	<b>34</b>

---

## 1 Introduction

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

The FT2232 I2C 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 Inter-Integrated Circuit (I2C) synchronous serial protocol interface. The FT2232 I2C API will be contained within the **FT232RL.DLL**.

The FT232RL DLL has been created to allow application developers to use the FT2232D, FT2232H and FT4232H devices to create a USB to Inter-Integrated Circuit (I2C) protocol interface without any knowledge of the MPSSE command set. All of the functions in FT232RL.DLL can be replicated using calls to FT2XX.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 FT232RL drivers while channel A is being used for I2C 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 I2C 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 FT232RL drivers while channels A and B are being used for I2C communication.

This document lists all of the functions available in FT232RL.DLL.

---

## 2 Application Programming Interface (API)

### 2.1 Public Functions

#### 2.1.1 I2C\_GetNumDevices

FTC\_STATUS **I2C\_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 I2C\_GetNumHiSpeedDevices

FTC\_STATUS **I2C\_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 I2C\_GetDeviceNameLocID

FTC\_STATUS **I2C\_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 100 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 I2C\_GetHiSpeedDeviceNameLocIDChannel

FTC\_STATUS **I2C\_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 I2C\_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.

### **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\_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 I2C\_Open

FTC\_STATUS **I2C\_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 its assigned handle is still open then its 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 I2C\_OpenEx

FTC\_STATUS **I2C\_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 its assigned handle is still open then its 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 I2C\_OpenHiSpeedDevice

FTC\_STATUS **I2C\_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 its assigned handle is still open then its 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 I2C\_GetHiSpeedDeviceType

FTC\_STATUS I2C\_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



### 2.1.11 I2C\_InitDevice

FTC\_STATUS **I2C\_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 2.1.18.

Note: the frequency in Hz, represented by the divisor, is calculated using the following formula:  $\text{frequency} = 12\text{MHz} / ((1 + \text{dwClockDivisor}) * 2)$ .

#### 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 I2C\_TurnOnDivideByFiveClockingHiSpeedDevice

FTC\_STATUS **I2C\_TurnOnDivideByFiveClockingHiSpeedDevice** (FTC\_HANDLE fhandle)

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 I2C\_TurnOffDivideByFiveClockingHiSpeedDevice

FTC\_STATUS **I2C\_TurnOffDivideByFiveClockingHiSpeedDevice** (FTC\_HANDLE fhandle)

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 I2C\_TurnOnThreePhaseDataClockingHiSpeedDevice

FTC\_STATUS **I2C\_TurnOnThreePhaseDataClockingHiSpeedDevice** (FTC\_HANDLE ftHandle)

This function turns on 3 phase data clocking for a FT2232H dual hi-speed device or FT4232H quad hi-speed device. Three phase data clocking, ensures the data is valid on both edges of a clock.

### 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

## 2.1.15 I2C\_TurnOffThreePhaseDataClockingHiSpeedDevice

FTC\_STATUS **I2C\_TurnOffThreePhaseDataClockingHiSpeedDevice** (FTC\_HANDLE ftHandle)

This function turns off 3 phase data clocking for a FT2232H dual hi-speed device or FT4232H quad hi-speed device. The default is 2 phase data clocking ie the data is only valid for one edge of a clock.

### 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

## 2.1.16 I2C\_SetDeviceLatencyTimer

FTC\_STATUS **I2C\_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

## 2.1.17 I2C\_GetDeviceLatencyTimer

FTC\_STATUS **I2C\_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.

## 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`

### 2.1.18 I2C\_GetClock

`FTC_STATUS I2C_GetClock` (DWORD dwClockDivisor, LPDWORD lpdwClockFrequencyHz)

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

#### Parameters

dwClockDivisor

Specifies a divisor, which will be used to calculate 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.

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:  $\text{frequency} = 12\text{MHz} / ((1 + \text{dwClockDivisor}) * 2)$ .

#### Return Value

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

`FTC_INVALID_CLOCK_DIVISOR`



### 2.1.19 I2C\_GetHiSpeedDeviceClock

FTC\_STATUS **I2C\_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 <b>Hz</b> , 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:  $\text{frequency} = 60\text{MHz} / ((1 + \text{dwClockDivisor}) * 2)$ .

#### Return Value

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

FTC\_INVALID\_CLOCK\_DIVISOR

### 2.1.20 I2C\_SetClock

FTC\_STATUS **I2C\_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 <b>Hz</b> , 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:  $\text{frequency} = 12\text{MHz}/((1 + \text{dwClockDivisor}) * 2)$

For the FT2232H dual and FT4232H quad hi-speed devices the frequency in Hz, represented by the divisor, is calculated using the following formula:  $\text{frequency} = 60\text{MHz}/((1 + \text{dwClockDivisor}) * 2)$

## 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

### 2.1.21 I2C\_SetLoopback

FTC\_STATUS **I2C\_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

## 2.1.22 I2C\_SetMode

FTC\_STATUS **I2C\_SetMode** (FTC\_HANDLE ftHandle, DWORD dwCommsMode)

This function specifies the communications mode of 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 I2C synchronous protocol. Default is FAST\_MODE.

### Parameters

ftHandle	Handle of a FT2232D dual device.
dwCommsMode	Specifies the communications mode of an external device.

### Valid Communications Modes

STANDARD\_MODE  
FAST\_MODE  
STRETCH\_DATA\_MODE

### Return Value

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

FTC\_INVALID\_HANDLE  
FTC\_INVALID\_COMMS\_MODE

## 2.1.23 I2C\_SetGPIOs

FTC\_STATUS **I2C\_SetGPIOs** (FTC\_HANDLE ftHandle, PFTC\_INPUT\_OUTPUT\_PINS pHighInputOutputPinsData)

This function controls 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.
pHighInputOutputPinsData	Pointer to the structure that contains the data that is used to control 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\_OUTPUT\_BUFFER\_POINTER  
FTC\_FAILED\_TO\_COMPLETE\_COMMAND  
FTC\_IO\_ERROR

### Example:

```
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
```

## 2.1.24 I2C\_SetHiSpeedDeviceGPIOs

FTC\_STATUS **I2C\_SetHiSpeedDeviceGPIOs** (FTC\_HANDLE ftHandle,  
PFTH\_INPUT\_OUTPUT\_PINS  
pHighInputOutputPinsData)

This function controls the use of the 8 general purpose higher input/output pins (GPIOH1 – GPIOH8) of the FT2232H dual hi-speed device.

### Parameters

ftHandle	Handle of the FT2232H dual 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 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
```

### 2.1.25 I2C\_GetGPIOs

FTC\_STATUS **I2C\_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
```



## 2.1.26 I2C\_GetHiSpeedDeviceGPIOs

FTC\_STATUS **I2C\_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
```

## 2.1.27 I2C\_Write

FTC\_STATUS **I2C\_Write** (FTC\_HANDLE ftHandle, PWriteControlByteBuffer pWriteControlBuffer, DWORD dwNumControlBytesToWrite, BOOL bControlAcknowledge, DWORD dwControlAckTimeoutmSecs, BOOL bStopCondition, DWORD dwDataWriteTypes, PWriteDataByteBuffer pWriteDataBuffer, DWORD dwNumDataBytesToWrite, BOOL bDataAcknowledge, DWORD dwDataAckTimeoutmSecs, PFTC\_PAGE\_WRITE\_DATA pPageWriteData)

This function writes data to 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 I2C synchronous protocol.

### Parameters

ftHandle	Handle of a FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device.
pWriteControlBuffer	Pointer to buffer that contains the control and address data to be written to an external device. Listed below are four examples of control and address bytes:
Control Address byte, Address byte	
Control Address byte, Address byte 1, Address byte 0	
Control Address byte, Control byte, Address byte	
Control Address byte, Control byte 1 ... Control byte n	
dwNumControlBytesToWrite	Specifies the number of bytes in the write data buffer, to be written to an external device. Valid range 1 to 255 bytes.
bControlAcknowledge	Check for acknowledgement after every control byte is written to an external device, acknowledgement required(TRUE), acknowledgement not required(FALSE).
dwControlAckTimeoutmSecs	Timeout interval in milliseconds to wait for an acknowledgement after a control byte has been written to an external device. A value of INFINITE indicates, timeout never expires waiting for an acknowledgement. Only valid when bControlAcknowledge variable is TRUE.

---

bStopCondition	Send a Stop condition, after all control bytes have been written to an external device, send Stop condition(TRUE), do not send Stop condition(FALSE).
dwDataWriteTypes	Specifies the type of write to be used, when the data contained in the write data buffer is written to an external device. Write no data, write the data one byte at a time or write the data in pages, ex :- 8 pages of 8 bytes per page.
<b><u>Valid Data Write Types</u></b>	
NO_WRITE_TYPE	
BYTE_WRITE_TYPE	
PAGE_WRITE_TYPE	
pWriteDataBuffer	Pointer to buffer that contains the data to be written to external device.
dwNumDataBytesToWrite	Specifies the number of bytes in the write data buffer, to be written to an external device. Valid range 1 to 65535 ie 64K bytes. If NO_WRITE_TYPE specified, no data bytes will be written to external device. If BYTE_WRITE_TYPE specified, only one data byte will be written to external device.
bDataAcknowledge	Check for acknowledgement after every data byte is written to an external device, acknowledgement required(TRUE), acknowledgement not required(FALSE).
dwDataAckTimeoutmSecs	Timeout interval in milliseconds to wait for an acknowledgement after a data byte has been written to an external device. A value of INFINITE indicates, timeout never expires waiting for an acknowledgement. Only valid, if bDataAcknowledge variable is TRUE.
pPageWriteData	Pointer to a structure that contains the number of pages and the number of bytes per page to be written to an external 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_CONTROL_DATA_BUFFER_POINTER  
FTC_INVALID_NUMBER_CONTROL_BYTES  
FTC_CONTROL_ACKNOWLEDGE_TIMEOUT  
FTC_NULL_WRITE_DATA_BUFFER_POINTER  
FTC_INVALID_NUMBER_DATA_BYTES_WRITE  
FTC_DATA_ACKNOWLEDGE_TIMEOUT  
FTC_INVALID_WRITE_TYPE  
FTC_NUMBER_BYTES_TOO_SMALL_PAGE_WRITE  
FTC_NULL_PAGE_WRITE_BUFFER_POINTER  
FTC_FAILED_TO_COMPLETE_COMMAND  
FTC_IO_ERROR
```

## Example:

```
#define MAX_WRITE_CONTROL_BYTES_BUFFER_SIZE 256 // 256 bytes  
  
typedef BYTE WriteControlByteBuffer[MAX_WRITE_CONTROL_BYTES_BUFFER_SIZE];  
typedef WriteControlByteBuffer *PWriteControlByteBuffer;  
  
typedef struct FTC_Page_Write_Data {  
    DWORD    dwNumPages;  
    DWORD    dwNumBytesPerPage;  
} FTC_PAGE_WRITE_DATA *PFTC_PAGE_WRITE_DATA  
  
#define MAX_WRITE_DATA_BYTES_BUFFER_SIZE 65536 // 64K bytes  
  
typedef BYTE WriteDataByteBuffer[MAX_WRITE_DATA_BYTES_BUFFER_SIZE];  
typedef WriteDataByteBuffer *PWriteDataByteBuffer;
```

## 2.1.28 I2C\_Read

FTC\_STATUS **I2C\_Read** (FTC\_HANDLE ftHandle, PWriteControlByteBuffer pWriteControlBuffer, DWORD dwNumControlBytesToWrite, BOOL bControlAcknowledge, DWORD dwControlAckTimeoutmSecs, DWORD dwDataReadTypes, PReadDataByteBuffer pReadDataBuffer, DWORD dwNumDataBytesToRead)

This function reads data 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 I2C synchronous protocol.

### Parameters

ftHandle	Handle of a FT2232D dual device or FT2232H dual hi-speed device or FT4232H quad hi-speed device.
pWriteControlBuffer	Pointer to buffer that contains the control and address data to be written to an external device. Listed below are three examples of control address bytes:
Control Address byte, Address byte	
Control Address byte, Address byte 1, Address byte 0	
Control Address byte, Control byte, Address byte	
Control Address byte, Control byte 1 ... Control byte n	
dwNumControlBytesToWrite	Specifies the number of bytes in the write data buffer, to be written to an external device. Valid range 1 to 255 bytes.
bControlAcknowledge	Check for acknowledgement after every control byte is written to an external device, acknowledgement required(TRUE), acknowledgement not required(FALSE).
dwControlAckTimeoutmSecs	Timeout interval in milliseconds to wait for an acknowledgement after a control byte has been written to an external device. A value of INFINITE indicates, timeout never expires waiting for an acknowledgement. Only valid, if bControlAcknowledge variable is TRUE.
dwDataReadTypes	Specifies the type of read to be used, when data is to be read from an external device. Read the specified number of data bytes one byte at a time or read the specified number of data bytes in one continuous block.

### **Valid Data Read Types**

BYTE\_READ\_TYPE  
BLOCK\_READ\_TYPE

pReadDataBuffer	Pointer to buffer that contains the data read from an external device.
dwNumDataBytesToRead	Specifies the number of bytes to be read from an external device. Valid range 1 to 65535. I.E. 64K bytes. If BYTE_READ_TYPE specified, only one byte will be returned in the read data buffer.

### **Return Value**

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

FTC\_INVALID\_HANDLE  
FTC\_NULL\_CONTROL\_DATA\_BUFFER\_POINTER  
FTC\_INVALID\_NUMBER\_CONTROL\_BYTES  
FTC\_CONTROL\_ACKNOWLEDGE\_TIMEOUT  
FTC\_NULL\_READ\_DATA\_BUFFER\_POINTER  
FTC\_INVALID\_NUMBER\_DATA\_BYTES\_READ  
FTC\_INVALID\_READ\_TYPE  
FTC\_FAILED\_TO\_COMPLETE\_COMMAND  
FTC\_IO\_ERROR

### **Example:**

```
#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;
```

### 2.1.29 I2C\_GetDllVersion

FTC\_STATUS **I2C\_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

### 2.1.30 I2C\_GetErrorMessageString

FTC\_STATUS **I2C\_GetErrorMessageString** (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 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](mailto:sales1@ftdichip.com)  
E-mail (Support) [support1@ftdichip.com](mailto:support1@ftdichip.com)  
E-mail (General Enquiries) [admin1@ftdichip.com](mailto: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 8791 3570  
Fax: +886 (0) 2 8791 3576

E-mail (Sales) [tw.sales1@ftdichip.com](mailto:tw.sales1@ftdichip.com)  
E-mail (Support) [tw.support1@ftdichip.com](mailto:tw.support1@ftdichip.com)  
E-mail (General Enquiries) [tw.admin1@ftdichip.com](mailto: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](mailto:us.sales@ftdichip.com)  
E-Mail (Support) [us.admin@ftdichip.com](mailto: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](mailto:cn.sales@ftdichip.com)  
E-Mail (Support): [cn.support@ftdichip.com](mailto:cn.support@ftdichip.com)  
E-Mail (General Enquiries): [cn.admin1@ftdichip.com](mailto: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.

Vinculum is part of Future Technology Devices International Ltd. Neither the whole nor any part of the information contained in, or the product described in this manual, may be adapted or reproduced in any material or electronic form without the prior written consent of the copyright holder. This product and its documentation are supplied on an as-is basis and no warranty as to their suitability for any particular purpose is either made or implied. Future Technology Devices International Ltd will not accept any claim for damages howsoever arising as a result of use or failure of this product. Your statutory rights are not affected. This product or any variant of it is not intended for use in any medical appliance, device or system in which the failure of the product might reasonably be expected to result in personal injury. This document provides preliminary information that may be subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document. Future Technology Devices International Ltd, Unit 1, 2 Seaward Place, Centurion Business Park, Glasgow G41 1HH United Kingdom. Scotland Registered Number: SC136640

---

## Appendix - Revision History

### Revision History

Draft	Initial Draft	December, 2008
1.0	Initial Release	21 <sup>st</sup> January, 2009
1.1	Corrections to add missing commands Corrected Taiwan Address	18 <sup>th</sup> March 2009