Conceiving, Planning and Development in scientific electronics

FTD2XX.DLL DYNAMIC LIBRARY

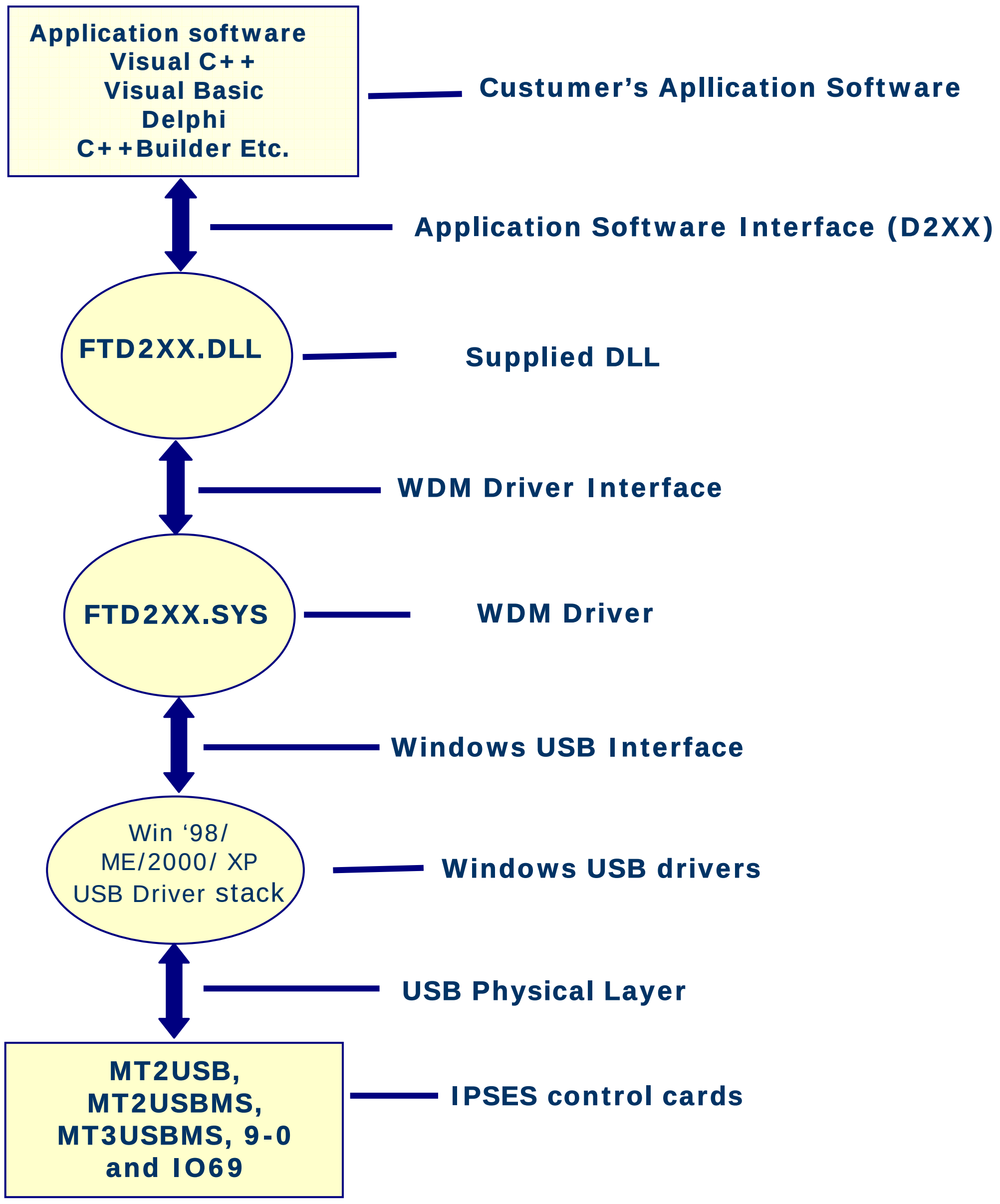
USER MANUAL

IPSES S.r.l. - Via Trieste, 48 - 20020 Cesate (MI) - ITALY
Tel. (+39) 02/99068453 Fax (+39) 02/700403170
http://www.ipses.com e-mail info@ipses.com

TABLE OF CONTENTS

The **FTD2XX.DLL** Dynamic Library for Windows allows you to write your application software to interface with the control card devices by **IPSES S.r.l.** using a DLL.
The architecture of the FTD2XX.DLL drivers consists of a Windows WDM driver that communicates with the device via the Windows USB Stack and a DLL which interfaces the Application Software (written in VC++, C++ Builder, Delphi, VB etc.) to the WDM driver. The FTD2XX.DLL interface provides a simple, easy to use, set of functions to access **MT2USB**, **MT2USBMS**, **MT3USBMS**, **9-0** and **IO-69** control card.

D2XX Driver Architecture



Variables

UCHAR	unsigned char (1 byte).
PUCHAR	pointer to unsigned char (4 bytes).
PCHAR	pointer to char (1 byte).
DWORD	unsigned long (4 bytes).
FT_HANDLE	**DWORD**.

Errors

UFT_STATUS (DWORD)

FT_OK = 0
FT_INVALID_HANDLE = 1
FT_DEVICE_NOT_FOUND = 2
FT_DEVICE_NOT_OPENED = 3
FT_IO_ERROR = 4
FT_INSUFFICIENT_RESOURCES = 5
FT_INVALID_PARAMETER = 6
FT_INVALID_BAUD_RATE = 7
FT_DEVICE_NOT_OPENED_FOR_ERASE = 8
FT_DEVICE_NOT_OPENED_FOR_WRITE = 9
FT_FAILED_TO_WRITE_DEVICE = 10
FT_EEPROM_READ_FAILED = 11
FT_EEPROM_WRITE_FAILED = 12
FT_EEPROM_ERASE_FAILED = 13
FT_EEPROM_NOT_PRESENT = 14
FT_EEPROM_NOT_PROGRAMMED = 15
FT_INVALID_ARGS = 16
FT_OTHER_ERROR = 17

DLL Functions

FT_ListDevices

Description Gets information concerning the devices currently connected. This function can return such information as the number of devices connected, and device strings such as *serial number* and product description.

Syntax FT_STATUS **FT_ListDevices** (PVOID *pvArg1*, PVOID *pvArg2,* DWORD *dwFlags*)

Parameters

pvArg1	meaning depend on the *dwFlags* value (see note below)
pvArg2	meaning depend on the dwFlags value (see note below)
dwFlags	Determines format of returned information (see note below)

Return Value FT_OK if successful, otherwise the return value is an FT error code

Note Remarks This function can be used in a number of ways to return different types of information.

In its simplest form, it can be used to return the number of devices currently connected. If **FT_LIST_NUMBER_ONLY** bit is set in *dwFlags*, the parameter *pvArg1* is interpreted as a pointer to a DWORD location to store the number of devices currently connected.

It can be used to return device string information. If FT_OPEN_BY_SERIAL_NUMBER bit is set in *dwFlags*, the *serial number* string will be returned from this function. If FT_OPEN_BY_DESCRIPTION bit is set in *dwFlags*, the product description string will be returned from this function. If neither of these bits is set, the *serial number* string will be returned by default. It can be used to return device string information for a single device. If FT_LIST_BY_INDEX bit is set in *dwFlags*, the parameter *pvArg1* is interpreted as the index of the device, and the parameter *pvArg2* is interpreted as a pointer to a buffer to contain the appropriate string. Indexes are zero-based, and the error code FT_DEVICE_NOT_FOUND is returned for an invalid index.

It can be used to return device string information for all connected devices. If FT_LIST_ALL bit is set in *dwFlags*, the parameter *pvArg1* is interpreted as a pointer to an array of pointers to buffers to contain the appropriate strings, and the parameter *pvArg2* is interpreted as a pointer to a DWORD location to store the number of devices currently connected. Note that, for *pvArg1*, the last entry in the array of pointers to buffers should be a NULL pointer so the array will contain one more location than the number of devices connected.

FT_Open

Description Opens the device and return a handle which will be used for subsequent accesses.

Syntax FT_STATUS **FT_Open** (int *iDevice*, FT_HANDLE **ftHandle*)

Parameters

iDevice	indicates the number of the device to be opened. Must be 0 if only one device is attached. For multiple devices 1, 2 etc.
ftHandle	Pointer to a variable of type FT_HANDLE where the handle will be stored. This handle must be used to access the device.

Return Value FT_OK if successful, otherwise the return value is an FT error code

Note Although this function can be used to open multiple devices by setting *iDevice* to 0, 1, 2 etc. there is no ability to open a specific device. To open named devices, use the function **FT_OpenEx.** With the **FT_OpenEx** function it is possible to open a device also trough its *serial number* or trough its description. For further information, please contact **IPSES S.r.l.**

FT_OpenEx

Description Opens the specified device and return a handle that will be used for subsequent accesses. The device can be specified by its *serial number*, device description or location. This function can also be used to open multiple devices simultaneously. Multiple devices can be opened at the same time if they can be distinguished by *serial number* or device description. Alternatively, multiple devices can be opened at the same time using location IDs – location information derived from their physical locations on USB. Location IDs can be obtained using the utility USBView and are given in hexadecimal format.

Syntax FT_STATUS **FT_OpenEx** (PVOID *pvArg1*, DWORD *dwFlags*, FT_HANDLE **ftHandle*)

Parameters

pvArg1	Meaning depends on *dwFlags*, but it will normally be interpreted as a pointer to a null terminated string.
dwFlags	*FT_OPEN_BY_SERIAL_NUMBER*, *FT_OPEN_BY_DESCRIPTION or FT_OPEN_BY_LOCATION*.
ftHandle	Pointer to a variable of type FT_HANDLE where the handle will be stored. This handle must be used to access the device.

Return Value FT_OK if successful, otherwise the return value is an FT error code.

Note The meaning of *pvArg1* depends on *dwFlags*: if *dwFlags* is *FT_OPEN_BY_SERIAL_NUMBER*, *pvArg1* is interpreted as a pointer to a null-terminated string that represents the *serial number* of the device; if *dwFlags* is *FT_OPEN_BY_DESCRIPTION*, *pvArg1* is interpreted as a pointer to a null-terminated string that represents the device description; if *dwFlags* is *FT_OPEN_BY_LOCATION*, *pvArg1* is interpreted as a long value that contains the location ID of the device. Please note that Windows CE and Linux do not support location IDs. *ftHandle* is a pointer to a variable of type FT_HANDLE where the handle is to be stored. This handle must be used to access the device. For further information, please contact **IPSES S.r.l.**

FT_Close

Description	Closes the communication with a open device.
Syntax	FT_STATUS **FT_Close** (FT_HANDLE *ftHandle*)

Parametres

ftHandle	pointer to the communication *handle* of the device to close.

Return Value	FT_OK if successful, otherwise the return value is an FT error code

FT_Read

Description	Reads a string from the device.
Syntax	FT_STATUS **FT_Read** (FT_HANDLE *ftHandle*, LPVOID *lpBuffer*, DWORD *dwBytesToRead*, LPDWORD *lpdwBytesReturned*)

Parameters

ftHandle	pointer to the communication *handle* of the device to read.
lpBuffer	pointer to the buffer that receives the data from the device.
DwBytesToRead	Number of bytes to be read from the device.
lpdwBytesReturned	Pointer to a variable of type DWORD which receives the number of bytes read from the device.

Return Value	FT_OK if successful, FT_IO_ERROR otherwise.

Note **FT_Read** always returns the number of bytes read in *lpdwBytesReturned*. This function does not return until *dwBytesToRead* have been read into the buffer. The number of bytes in the receive queue can be determined by calling **FT_GetStatus** or **FT_GetQueueStatus**, and passed to **FT_Read** as *dwBytesToRead* so that the function reads the device and returns immediately. When a read timeout value has been specified in a previous call to **FT_SetTimeouts**, **FT_Read** returns when the timer expires or *dwBytesToRead* have been read, whichever occurs first. If the timeout occurred, **FT_Read** reads available data into the buffer and returns FT_OK. An application should use the function return value and *lpdwBytesReturned* when processing the buffer. If the return value is FT_OK, and *lpdwBytesReturned* is equal to *dwBytesToRead* then **FT_Read** has completed normally. If the return value is FT_OK, and *lpdwBytesReturned* is less then *dwBytesToRead* then a timeout has occurred, and the read has been partially completed. Note that if a timeout occurred and no data was read, the return value is still FT_OK. A return value of FT_IO_ERROR suggests an error in the parameters of the function, or a fatal error like USB disconnect has occurred.

FT_Write

Description Writes a string to the device.

Syntax FT_STATUS **FT_Write** (FT_HANDLE *ftHandle*, LPVOID *lpBuffer*, DWORD *dwBytesToWrite*, LPDWORD *lpdwBytesWritten*)

Parameters

ftHandle	pointer to the communication *handle* of the device to write.
lpBuffer	pointer to the buffer which contains the bytes to be written in the device.
DwBytesToWrite	number of bytes to write to the device.
lpdwBytesWritten	pointer to a variable of type DWORD which receives the number of bytes written to the device

Return Value FT_OK if successful, otherwise the return value is an FT error code.

FT_ResetDevice

Description Sends a Reset command to the device.

Syntax FT_STATUS **FT_ResetDevice** (FT_HANDLE *ftHandle*)

Parameters

ftHandle	pointer to the communication *handle* of the device to reset .

Return Value FT_OK if successful, otherwise the return value is an FT error code.

FT_SetBaudRate

Description Sets the *baudrate* for the device.

Syntax FT_STATUS **FT_SetBaudRate** (FT_HANDLE *ftHandle*, DWORD *dwBaudRate*)

Parameters

ftHandle	pointer to the communication *handle* of the device to set out.
dwBaudRate	value of the *baudrate* to set out.

Return Value FT_OK if successful, otherwise the return value is an FT error code.

FT_SetBaudRate parameter needs to be set as listed below to communicate with **IPSES S.r.l.** devices.

Device	*dwBaudRate*
MT2USB	9600
MT2USBMS	9600
MT3USBMS	9600
9-0	19200

FT_SetDataCharacteristics

Description Sets the data characteristics for the device**.**

Syntax FT_STATUS **FT_SetDataCharacteristics** (FT_HANDLE *ftHandle*, UCHAR *uWordLength*, UCHAR *uStopBits*, UCHAR *uParity*)

Parameters

ftHandle	pointer to the communication *handle* of the device to set out .
uWordLength	number of *bits* per word. It must set as FT_BITS_8 (in the case of 8 bit schosen) or as FT_BITS_7 (in the case of 7 bits chosen).
uStopBits	number of stop *bits*. It must set as FT_STOP_BITS_1 (when one stop bit is requested) or as FT_STOP_BITS_2 (when two stop bits are requested).
uParity	number of parity *bits*. It must set as FT_PARITY_NONE (no parity bit) or as FT_PARITY_ODD (parity bit is odd) or as FT_PARITY_EVEN (parity bit is even) or as FT_PARITY_MARK (always high parity bit) or as FT_PARITY_SPACE (always low parity bit).

Return Value FT_OK if successful, otherwise the return value is an FT error code.

FT_SetDataCharacteristics parameters needs to be set as listed below to communicate with **IPSES S.r.l.** devices.

Device	*uWordLength*	*uStopBits*	*uParity*
MT2USB	8	1	0
MT2USBMS	8	1	0

MT3USBMS	8	1	0
9-0	8	1	0

FT_SetFlowControl

Description Sets the flow control the chip serial communication of chip USB/RS232.

Syintax FT_STATUS **FT_SetFlowControl** (FT_HANDLE *ftHandle*, USHORT *usFlowControl*, UCHAR *uXon*, UCHAR *uXoff*)

Parameters

ftHandle	pointer to the communication *handle* of the device to set out.
usFlowControl	set the kind of flow control. It must be set as FT_FLOW_NONE (no flow control) or as FT_FLOW_RTS_CTS (*hardware* RTS/CTS flow control) or as FT_FLOW_DTR_DSR (*hardware* DTR/DSR flow control) or as FT_FLOW_XON_XOFF (*software* XON/XOFF flow control)
uXon	shows the character uses as Xon signal. It must be set only when the flow control is *software* XON/XOFF kind (otherwise, it must be set as zero).
uXoff	shows the character uses as Xoff signal. It must be set only when the flow control is *software* XON/XOFF kind (otherwise, it must be set as zero).

Return Value FT_OK if successful, otherwise the return value is an FT error code.

FT_SetFlowControl parameters needs to be set as listed below to communicate with **IPSES S.r.l.** devices.

Device	*usFlowControl*	*uXon*	*uXoff*
MT2USB	NONE	0	0
MT2USBMS	NONE	0	0
MT3USBMS	NONE	0	0
9-0	NONE	0	0

FT_GetModemStatus

Description Gets the modem status from the device.

Syintax FT_STATUS **FT_GetModemStatus** (FT_HANDLE *ftHandle,* LPDWORD *lpdwModemStatus*)

Parameters

FtHandle	*Handle* of the device.
lpdwModemStatus	Pointer to a variable of type DWORD which receives the modem status from the device. Status lines are bit-mapped as follows:

CTS = 0x10
DSR = 0x20
RI = 0x40
DCD = 0x80

Return Value	FT_OK if successful, otherwise the return value is an FT error code.

FT_Purge

Description	This function purges receive and transmit buffers in the device.
Syintax	FT_STATUS **FT_Purge** (FT_HANDLE *ftHandle,* DWORD *dwMask*)
Parameters	
FtHandle	*Handle* of the device.
dwMask	Any combination of FT_PURGE_RX and FT_PURGE_TX.
Return Value	FT_OK if successful, otherwise the return value is an FT error code.

FT_SetTimeouts

Description	This function sets the read and write timeouts for the device.
Syintax	FT_STATUS **FT_SetTimeouts** (FT_HANDLE *ftHandle*, DWORD *dwReadTimeout*, DWORD *dwWriteTimeout*)
Parameters	
FtHandle	*Handle* of the device.
dwReadTimeout	Read timeout in milliseconds.
dwWriteTimeout	Write timeout in milliseconds.
Return Value	FT_OK if successful, otherwise the return value is an FT error code.

FT_GetQueueStatus

Description	Shows the number of characters in the receive queue.
Syntax	FT_STATUS **FT_GetQueueStatus** (FT_HANDLE *ftHandle*, LPDWORD *lpdwAmountInRxQueue*)
Parameters	
ftHandle	pointer to the communication *handle* of the device to set out .
lpdwAmountInRxQueue	Pointer to a variable of type DWORD which receives the number of characters in the receive queue.

Return Value FT_OK if successful, otherwise the return value is an FT error code.

FT_GetStatus

Description Shows the device status including number of characters in the receive queue, number of characters in the transmit queue, and the current event status.

Syntax FT_STATUS **FT_GetStatus** (FT_HANDLE *ftHandle*, LPDWORD *lpdwAmountInRxQueue* , LPDWORD *lpdwAmountInTxQueue*, LPDWORD *lpdwEventstatus*)

Parameters

ftHandle	pointer to the communication *handle* of the device to set out .
lpdwAmountInRxQueu	Pointer to a variable of type DWORD which receives the number of characters in the receive queue.
LpdwAmountInTxQueue	Pointer to a variable of type DWORD which receives the number of characters in the transmit queue.
lpdwEventstatus	Pointer to a variable of type DWORD which receives the current state of the event status.

Return Value FT_OK if successful, otherwise the return value is an FT error code.

FT_GetDeviceInfo

Description Get device information.

Syntax FT_STATUS **FT_GetDeviceInfo** (FT_HANDLE *ftHandle*, FT_DEVICE **pftType*, LPDWORD *lpdwID*, PCHAR *pcSerialNumber*, PCHAR *pcDescription*, PVOID *pvDummy*)

Parameters

ftHandl	pointer to the communication *handle* of the device to set out .
pftType	Pointer to unsigned long to store device type.
LpdwId	Pointer to unsigned long to store device ID.
pcSerialNumber	Pointer to buffer to store device *serial number* as a null terminated string.
pcDescription	Pointer to buffer to store device description as a null-terminated string.
pvDummy	Reserved for future use - should be set to NULL.

Return Value FT_OK if successful, otherwise the return value is an FT error code.

Note This function is used to return the device type, device ID, device description and *serial number*. The device ID is encoded in a DWORD - the most significant word contains the vendor ID, and the least significant word contains the product ID. So the returned ID 0x04036001 corresponds to the device ID VID_0403&PID_6001.

FT_ResetPort

Description Send a reset command to the port.

Sintax FT_STATUS **FT_ResetPort** (FT_HANDLE ft*Handle*)

Parameters

ftHandle Handle of the device.

Return Value FT_OK if successful, otherwise the return value is an FT error code.

Note This function is used to attempt to recover the port after a failure. It is not equivalent to an unplugreplug event. **Not available in Windows CE and Linux**.

FT_CreateDeviceInfoList

Description This function builds a device information list and returns the number of D2XX devices connected to the system. The list contains information about both unopen and open devices.

Sintax FT_STATUS **FT_CreateDeviceInfoList** (LPDWORD *lpdwNumDevs*)

Parameters

lpdwNumDevs Pointer to unsigned long to store the number of devices connected.

Return Value FT_OK if successful, otherwise the return value is an FT error code.

Note An application can use this function to get the number of devices attached to the system. It can then allocate space for the device information list and retrieve the list using **FT_GetDeviceInfoList** . If the devices connected to the system change, the device info list will not be updated until **FT_CreateDeviceInfoList** is called again.

FT_GetDeviceInfoList

Description This function returns a device information list and the number of D2XX devices in the list.

Sintax FT_STATUS FT_**GetDeviceInfo** (FT_DEVICE_LIST_INFO_NODE **pDest*, LPDWORD *lpdwNumDevs*)

Parameters

**pDest*	Pointer to an array of FT_DEVICE_LIST_INFO_NODE structures.
lpdwNumDevs	Pointer to the number of elements in the array.

Return Value FT_OK if successful, otherwise the return value is an FT error code.

Note This function should only be called after calling **FT_CreateDeviceInfoList** . If the devices connected to the system change, the device info list will not be updated until **FT_CreateDeviceInfoList** is called again. Location ID information is not returned for devices that are open when **FT_CreateDeviceInfoList** is called. The array of **FT_DEVICE_LIST_INFO_NODES** contains all available data on each device. The storage for the list must be allocated by the application. The number of devices returned by **FT_CreateDeviceInfoList** can be used to do this. When programming in Visual Basic, LabVIEW or similar languages, **FT_GetDeviceInfoDetail** may be required instead of this function. Please note that Windows CE and Linux do not support location IDs. As such, the Location ID parameter in the structure will be empty under Windows CE and Linux.

FT_GetDeviceInfoDetail

Description This function returns an entry from the device information list.

Sintax FT_STATUS **FT_GetDeviceInfoDetail** (DWORD *dwIndex*, LPDWORD *lpdwFlags*, LPDWORD *lpdwType*, LPDWORD *lpdwID*, LPDWORD *lpdwLocId*, PCHAR *pcSerialNumber*, PCHAR *pcDescription*, FT_HANDLE **ftHandle*)

Parameters

dwIndex	Index of the entry in the device info list.
lpdwFlags	Pointer to unsigned long to store the flag value.
lpdwType	Pointer to unsigned long to store device type.
lpdwID	Pointer to unsigned long to store device ID.
lpdwLocId	Pointer to unsigned long to store the device location ID.
pcSerialNumber	Pointer to buffer to store device *serial number* as a nullterminated string.
pcDescription	Pointer to buffer to store device description as a null-terminated string.
**ftHandle*	Pointer to a variable of type FT_HANDLE where the handle will be stored.

Return Value FT_OK if successful, otherwise the return value is an FT error code.

Note This function should only be called after calling **FT_CreateDeviceInfoList** . If the devices connected to the system change, the device info list will not be updated until **FT_CreateDeviceInfoList** is called again. The index value is zero-based. The flag value is a 4-byte bit map containing miscellaneous data. Bit 0 (least significant bit) of this number indicates if the port is open (1) or closed (0). The remaining bits (1 - 31) are reserved at this time. Location ID information is not returned for devices that are open when **FT_CreateDeviceInfoList** is called. To return the whole device info list as an array of **FT_DEVICE_LIST_INFO_NODE** structures, use **FT_GetDeviceInfoList** . Please note that Windows CE and Linux do not support location IDs. As such, the Location ID parameter in the structure will be empty under Windows CE and Linux.

FT_GetDriverVersion

Description This function returns the D2XX driver version number.
Sintax FT_STATUS **FT_GetDriverVersion** (FT_HANDLE *ftHandle*, LPDWORD *lpdwDriverVersion*)

Parameters

ftHandle	Handle of the device.
lpdwDriverVersion	Pointer to the driver version number.

Return Value FT_OK if successful, otherwise the return value is an FT error code.

Note A version number consists of major, minor and build version numbers contained in a 4-byte field (unsigned long). Byte0 (least significant) holds the build version, Byte1 holds the minor version, and Byte2 holds the major version. Byte3 is currently set to zero. For example, driver version "3.01.02" is represented as 0x00030102. Note that a device has to be opened before this function can be called. **Not available in Windows CE or Linux.**

FT_GetLibraryVersion

Description This function returns D2XX DLL version number.

Sintax FT_STATUS **FT_GetLibraryVersion** (LPDWORD *lpdwDLLVersion*)

Parameters
lpdwDLLVersion Pointer to the DLL version number.

Return Value FT_OK if successful, otherwise the return value is an FT error code.

Note A version number consists of major, minor and build version numbers contained in a 4-byte field (unsigned long). Byte0 (least significant) holds the build version, Byte1 holds the minor version, and Byte2 holds the major version. Byte3 is currently set to zero. For example, driver version "3.01.02" is represented as

0x00030102. Note that this function does not take a handle, and so it can be called without opening a device. **Not available in Windows CE or Linux.**