



Conceiving, Planning and Development in scientific electronics

FTD2XX.DLL DYNAMIC LIBRARY

USER MANUAL



IPSES s. r. l. - Via Trieste, 20/5 - 20020 Cesate (MI) ITALY – VAT #: 03999740966
Tel. (+39) 02/99068453 Fax (+39) 02/700403170
<http://www.ipses.com> e-mail info@ipses.com

Information provided in this manual is property of IPSES S.r.l. and must be considered and treated as confidential.

This publication can only be reproduced, transmitted, transcribed or translated into any human or computer language with the written consent of IPSES S.r.l.

Information in this documentation has been carefully checked and is believed to be accurate as of the date of publication; however, no responsibility is assumed of inaccuracies. IPSES will not be liable for any consequential or incidental damages arising from reliance on the accuracy of this documentation.

Information contained in this manual is subject to change without notice and does not represent a commitment on the part of IPSES.

All brand or product names are trademarks or registered trademarks of their respective holders.

This manual in English is the original version.

Printed in Italy

Copyright © 2006-09 IPSES S.r.l.



All rights reserved.

TABLE OF CONTENTS

Manual Revision History.....	4
D2XX Driver Architecture	5
Variables	6
Status errors.....	6
Constants.....	6
DLL Functions	8
FT_ListDevices.....	8
FT_Open.....	9
FT_OpenEx	9
FT_Close.....	10
FT_Read	10
FT_Write.....	11
FT_ResetDevice	12
FT_SetBaudRate	12
FT_SetDataCharacteristics.....	13
FT_SetFlowControl	13
FT_GetModemStatus	14
FT_Purge	15
FT_SetTimeouts.....	15
FT_GetQueueStatus	15
FT_GetStatus	16
FT_GetDeviceInfo	16
FT_ResetPort.....	17
FT_CreateDeviceInfoList	17
FT_GetDeviceInfoList.....	18
FT_GetDeviceInfoDetail.....	18
FT_GetDriverVersion	19



FT_GetLibraryVersion 20
 FT_SetBitMode 20
 FT_GetBitMode 21
 CONTACTS 22
 SUPPORT INFORMATION..... 23
 PROBLEM REPORTING 23
 ENGINEERING PROBLEM REPORT..... 24

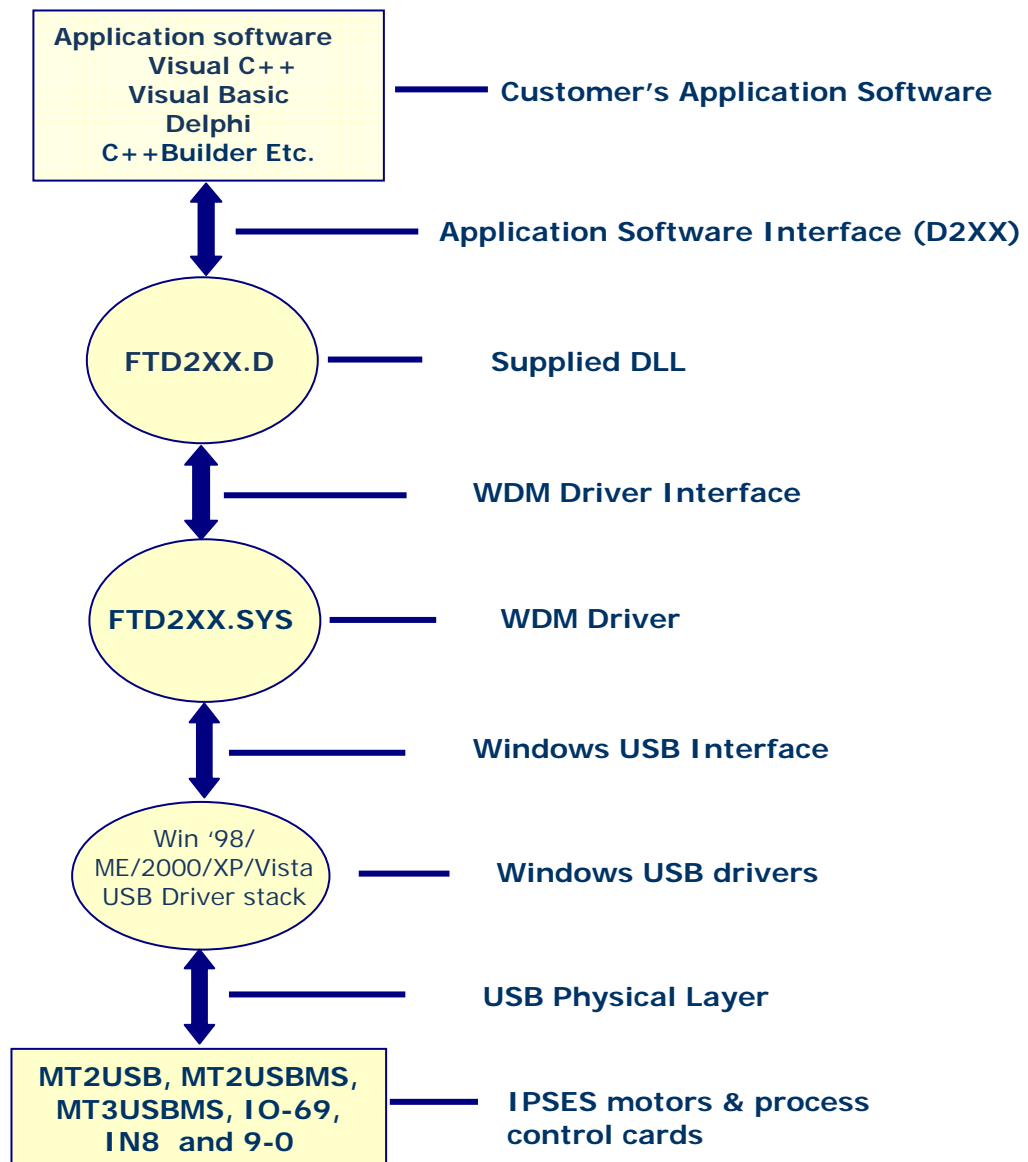
Manual Revision History

Revision/ Date	Change description	Author
01.00.0000 July, 2006	First version Released	Barbera D.
01.01.0000 April, 2009	Included "FT_SetBitMode" and "FT_GetBitMode" function's description. Other minor upgrade.	Rivolta A.



D2XX Driver Architecture

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**, **IO-69**, **IN8** and **9-0** control cards.



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**.

Status errors

FT_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

Constants

```
FT_DEVICE_LIST_INFO_NODE:  typedef struct _ft_device_list_info_node
                           {
                               DWORD      Flags;
                               DWORD      Type;
                               DWORD      ID;
                               DWORD      LocID;
                               char        SerialNumber[16];
                               char        Description[64];
                               FT_HANDLE   ftHandle;
                           } FT_DEVICE_LIST_INFO_NODE;
```



FT_LIST_NUMBER_ONLY	0x80000000
FT_LIST_BY_INDEX	0x40000000
FT_LIST_ALL	0x20000000
FT_OPEN_BY_SERIAL_NUMBER	1
FT_OPEN_BY_DESCRIPTION	2
FT_OPEN_BY_LOCATION	4
FT_BITS_8	8
FT_BITS_7	7
FT_STOP_BIT_1	0
FT_STOP_BIT_2	2
FT_PARITY_NONE	0
FT_PARITY_ODD	1
FT_PARITY_EVEN	2
FT_PARITY_MARK	3
FT_PARITY_SPACE	4
FT_FLOW_NONE	0x0000
FT_FLOW_RTS_CTS	0x0100
FT_FLOW_DTR_DSR	0x0200
FT_FLOW_XON_XOFF	0x0400
FT_PURGE_RX	1
FT_PURGE_TX	2



DLL Functions

FT_ListDevices

Description

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

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

This function can be used in a number of ways to return different types of information. A more powerful way to get device information is to use the FT_CreateDeviceInfoList, FT_GetDeviceInfoList and FT_GetDeviceInfoDetail functions as they return all the available information on devices.

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 information: if *FT_OPEN_BY_SERIAL_NUMBER* bit is set in *dwFlags*, the serial number string will be returned; if *FT_OPEN_BY_DESCRIPTION* bit is set in *dwFlags*, the product description string will be returned; if *FT_OPEN_BY_LOCATION* bit is set in *dwFlags*, the Location ID will be returned; if none 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* and *FT_OPEN_BY_SERIAL_NUMBER* or *FT_OPEN_BY_DESCRIPTION* bits are 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* and *FT_OPEN_BY_SERIAL_NUMBER* or *FT_OPEN_BY_DESCRIPTION* bits are 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.



The location ID of a device is returned if *FT_LIST_BY_INDEX* and *FT_OPEN_BY_LOCATION* bits are set in *dwFlags*. In this case the parameter *pvArg1* is interpreted as the index of the device, and the parameter *pvArg2* is interpreted as a pointer to a variable of type long to contain the location ID. Indexes are zero-based, and the error code *FT_DEVICE_NOT_FOUND* is returned for an invalid index. Please note that Windows CE and Linux do not support location IDs.

The location IDs of all connected devices are returned if *FT_LIST_ALL* and *FT_OPEN_BY_LOCATION* bits are set in *dwFlags*. In this case, the parameter *pvArg1* is interpreted as a pointer to an array of variables of type long to contain the location IDs, and the parameter *pvArg2* is interpreted as a pointer to a DWORD location to store the number of devices currently 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 Index of the device to open. Indices are 0 based.
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 through its *serial number* or through its description.

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 specified by serial number, device descriptor or location ID (location information derived from the physical locations of a device on USB). Location IDs for devices connected to a system can be obtained by calling *FT_GetDeviceInfoList* or *FT_ListDevices* with the appropriate flags.



Syntax

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

Parameters

pvArg1 Pointer to an argument whose type depends on the value of *dwFlags*. It is 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 parameter specified in *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. *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.

FT_Close**Description**

Closes an open device and releases its resources.

Syntax

FT_STATUS **FT_Close** (FT_HANDLE *ftHandle*)

Parametres

ftHandle Handle of the device.

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

<i>ftHandle</i>	Handle of the device.
<i>lpBuffer</i>	Pointer to the buffer that receives the data from the device.
<i>DwBytesToRead</i>	Number of bytes to be read from the device.
<i>lpdwBytesReturned</i>	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 than *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

<i>ftHandle</i>	Handle of the device.
<i>lpBuffer</i>	Pointer to the buffer which contains the data to be written to the device.
<i>DwBytesToWrite</i>	Number of bytes to write to the device.
<i>lpdwBytesWritten</i>	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

This function sends a reset command to the device.

Syntax

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

Parameters

ftHandle Handle of the device.

Return Value

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

FT_SetBaudRate

Description

This function sets the *baud rate* for the device.

Syntax

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

Parameters

ftHandle Handle of the device.
dwBaudRate Baud rate value.

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 devices provided by **IPSES S.r.l.**:

Device	<i>dwBaudRate</i>
MT2USB	9600
MT2USBMS	9600
MT3USBMS	9600
IO-69	9600
9-0	19200



FT_SetDataCharacteristics

Description

This function sets the data characteristics for the device.

Syntax

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

Parameters

ftHandle Handle of the device.

uWordLength Number of bits per word. It must set as *FT_BITS_8* (in the case of 8 bit chosen) 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 devices provided by **IPSES** S.r.l.:

Device	<i>uWordLength</i>	<i>uStopBits</i>	<i>uParity</i>
MT2USB	8	1	0
MT2USBMS	8	1	0
MT3USBMS	8	1	0
IO-69	8	1	0
9-0	8	1	0

FT_SetFlowControl

Description

This function sets the flow control for the device.

Syntax

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



Parameters

ftHandle Handle of the device.
usFlowControl set the kind of flow control. It must be one of FT_FLOW_NONE (no flow control), FT_FLOW_RTS_CTS (hardware RTS/CTS flow control), FT_FLOW_DTR_DSR (hardware DTR/DSR flow control) or FT_FLOW_XON_XOFF (software XON/XOFF flow control).
uXon Character used to signal Xon. Only used if flow control is FT_FLOW_XON_XOFF.
uXoff Character used to signal Xoff. Only used if flow control is FT_FLOW_XON_XOFF.

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 devices provided by **IPSES** S.r.l.:

Device	<i>usFlowControl</i>	<i>uXon</i>	<i>uXoff</i>
MT2USB	NONE	0	0
MT2USBMS	NONE	0	0
MT3USBMS	NONE	0	0
IO-69	NONE	0	0
9-0	NONE	0	0

FT_GetModemStatus

Description

Gets the modem status and line status from the device.

Syntax

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.

Syntax

FT_STATUS **FT_Purge** (FT_HANDLE *ftHandle*, ULONG *uMask*)

Parameters

<i>ftHandle</i>	Handle of the device.
<i>uMask</i>	Any combination of <i>FT_PURGE_RX</i> and <i>FT_PURGE_TX</i> .

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.

Syntax

FT_STATUS **FT_SetTimeouts** (FT_HANDLE *ftHandle*, DWORD *dwReadTimeout*, DWORD *dwWriteTimeout*)

Parameters

<i>ftHandle</i>	Handle of the device.
<i>dwReadTimeout</i>	Read timeout in milliseconds.
<i>dwWriteTimeout</i>	Write timeout in milliseconds.

Return Value

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

FT_GetQueueStatus

Description

Gets the number of bytes in the receive queue.

Syntax

FT_STATUS **FT_GetQueueStatus** (FT_HANDLE *ftHandle*, LPDWORD *lpdwAmountInRxQueue*)

Parameters

<i>ftHandle</i>	Handle of the device.
<i>lpdwAmountInRxQueue</i>	Pointer to a variable of type DWORD which receives the number of bytes in the receive queue.



Return Value

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

FT_GetStatus

Description

Get 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

<i>ftHandle</i>	Handle of the device.
<i>lpdwAmountInRxQueue</i>	Pointer to a variable of type DWORD which receives the number of characters in the receive queue.
<i>lpdwAmountInTxQueue</i>	Pointer to a variable of type DWORD which receives the number of characters in the transmit queue.
<i>lpdwEventstatus</i>	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 for an open device.

Syntax

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

Parameters

<i>ftHandle</i>	Handle of the device.
<i>pftType</i>	Pointer to unsigned long to store device type.
<i>lpdwId</i>	Pointer to unsigned long to store device ID.
<i>pcSerialNumber</i>	Pointer to buffer to store device <i>serial number</i> as a null terminated string.
<i>pcDescription</i>	Pointer to buffer to store device description as a null-terminated string.
<i>pvDummy</i>	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.

Syntax

FT_STATUS **FT_ResetPort** (FT_HANDLE *ftHandle*)

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 unplug-replug event.

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.

Syntax

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** or **FT_GetDeviceInfoDetail**. 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.

Syntax

```
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.

Information is not available for devices which are open in other processes. In this case, the *Flags* parameter of the **FT_DEVICE_LIST_INFO_NODE** will indicate that the device is open, but other fields will be unpopulated.

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.

FT_GetDeviceInfoDetail

Description

This function returns an entry from the device information list.

Syntax

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



Parameters

<i>dwIndex</i>	Index of the entry in the device info list.
<i>lpdwFlags</i>	Pointer to unsigned long to store the flag value.
<i>lpdwType</i>	Pointer to unsigned long to store device type.
<i>lpdwID</i>	Pointer to unsigned long to store device ID.
<i>lpdwLocId</i>	Pointer to unsigned long to store the device location ID.
<i>pcSerialNumber</i>	Pointer to buffer to store device serial number as a null terminated string.
<i>pcDescription</i>	Pointer to buffer to store device description as a null-terminated string.
<i>*ftHandle</i>	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). Bit 1 indicates if the device is enumerated as a high-speed USB device (2) or a full-speed USB device(0). The remaining bits (2 - 31) are reserved.

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_CreateDeviceInfoList** .

FT_GetDriverVersion

Description

This function returns the D2XX driver version number.

Syntax

```
FT_STATUS FT_GetDriverVersion (FT_HANDLE ftHandle, LPDWORD lpdwDriverVersion)
```

Parameters

<i>ftHandle</i>	Handle of the device.
<i>lpdwDriverVersion</i>	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.

FT_GetLibraryVersion

Description

This function returns D2XX DLL version number.

Syntax

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). Byte 0 (least significant) holds the build version, byte 1 holds the minor version, and byte 2 holds the major version. byte 3 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.

FT_SetBitMode

Description

Enables the CBUS Bit Bang Mode on MT2USB and MT2USBMS devices.

Syntax

FT_STATUS **FT_SetBitmode** (FT_HANDLE *ftHandle*, UCHAR *ucMask*, UCHAR *ucMode*)

Parameters

<i>ftHandle</i>	Handle of the device.
<i>ucMask</i>	Required value for bit mode mask. The upper nibble of this value controls which pins are inputs and outputs: a bit value of 0 sets the corresponding pin to an input, a bit value of 1 sets the corresponding pin to an output. The lower nibble controls which of the outputs are high (bit sets to 1) and low (bit sets to 0).
<i>ucMode</i>	Mode value. Can be one of the following: 0x00 = Reset 0x20 = CBUS Bit Bang Mode



Return Value

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

Note

This function is available only for **MT2USB** and **MT2USBMS** devices. The CBUS mode allows to control directly four physical pins (C3 to C0) of the USB chip on the device. For the devices provided by **IPSES** S.r.l. are available only pins C3 and C2, while pins C1 and C0 are reserved. So, the ucMask should be sets as "XX00YY00", where X will be 0 (input) or 1 (output) and Y will be the pins status. For further information, please contact **IPSES** S.r.l.

FT_GetBitMode**Description**

Gets the instantaneous values of CBUS Bit Bang Mode on MT2USB and MT2USBMS devices.

Syntax

FT_STATUS **FT_GetBitmode** (FT_HANDLE *ftHandle*, PCHAR *pucMode*)

Parameters

<i>ftHandle</i>	Handle of the device.
<i>pucMode</i>	Pointer to unsigned char to store the instantaneous data bus value. The lower nibble contains the current values of the pins, both those which are inputs and those which are outputs.

Return Value

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

Note

This function is available only for **MT2USB** and **MT2USBMS** devices. For further information, please contact **IPSES** S.r.l.



CONTACTS

IPSES S.r.l. conceives, designs, and markets electronic and scientific instruments. The customized design of our devices allows us to address specific needs for integration into embedded systems. **IPSES** customers enjoy access to a dedicated project engineering team, available as needed.

Our staff consists of highly competent professionals whose experience in the field is extremely strong. Thanks to constant training, process and technical development, **IPSES** is a leading company, combining the dynamism of a young group into the competence and reliability of a qualified staff.

IPSES S.r.l.

Research and development office:

via Trieste, 48
20020 Cesate (MI)
Italy



tel. +39 02 99068453
fax +39 02 700403170
e-mail: info@ipses.com
http://www.ipses.com



SUPPORT INFORMATION

The customer can contact the relevant engineer at IPSES S.r.l. directly.

A call can be logged in a variety of ways:

Telephone	:	+39 02 99068453
Fax	:	+39 02 700403170
Email	:	support@ipses.com

PROBLEM REPORTING

In case you encounter a problem using an IPSES product, we kindly ask you to report it by filling the form in the next page and sending it by fax to +39 02 700403170. The form can also be scanned and sent by e-mail to support@ipses.com . An electronic form is available on our web site (www.ipses.com)



ENGINEERING PROBLEM REPORT**Problem describer**

Name		IPSES S.r.l. Via Trieste, 48 Cesate (MI) Italy Fax ++39 02/700403170 e-mail <i>support@ipses.com</i>
Company		
Date	Tel.	

Product

Name	Version	Serial No.
------	---------	------------

Report Type (bug, change request or technical problem)

Major bug	<input type="checkbox"/>	Urgency:	
Minor bug	<input type="checkbox"/>	High	<input type="checkbox"/>
Change request	<input type="checkbox"/>	Medium	<input type="checkbox"/>
Technical problem	<input type="checkbox"/>	Low	<input type="checkbox"/>

Problem Description

--

Reproduction of Problem

--

IPSES S.r.l. Action notes

Received by	Date	Report No.	Action
-------------	------	------------	--------



Code FTD2XX.DLL

IPSES S.r.l.

Via Trieste, 48
20020 CESATE (MI) - ITALY
Tel. (+39) 02/99068453
Fax (+39) 02/700403170
e-mail: info@ipses.com
support@ipses.com

