Plus 2

USB Communication Interface

(v. 02, 10-2015)

Laserpoint srl – Via Burona, 51 – 20090 Vimodrone (Milano) – Italy
Phone +39 02 27 400 236 – Telefax +39 02 25 029 161
www.laserpoint.eu

Table of Contents

1 - Installation

Connect the ***Plus 2*** electronics through the USB port to the host PC device by a USB cable A to B type.

Include in your Code the FTD2XX.DLL Dynamic Library for Windows (described in Annex 1) in order to write your application.

2 - Input Commands and answer messages

When the ***Plus 2*** receives a valid input command, it confirms to the host device that the command has been received and it returns the answer as follows.

2.1 Input Commands

The format of a valid command is as follow:

*COMMANDNAME_NUMERICALVALUE:

where:

"*" : Start of command

":" : End of command

"_" : space character

COMMANDNAME : the instruction as described in the following table; it is an ASCII character sequence. The command name must be in capitals.

2.2 Answer messages

Plus 2 device sends a message through USB interface only if it receives an Input Command from the Host Device.

Maximum response time from ***Plus 2*** is ~50msec. Set a delay of 50ms between write and read function to wait for device reply.

The format of an answer is as follow:

ANSWER ;

where:

";" : End of answer

ANSWER : there are three kind of answer

String: ASCII character sequence

Int: integer number, numerical sequence (in ASCII code)

Float: floating point number, numerical sequence plus decimal point (in ASCII code); ex. the NumericValue 23.45 is codified with the 5 ASCII characters "23.45".

2.3 Commands & Answers description Table

Command Name	Description	Answer
HEADN	Displays the Head model	"H" + String 8 char
SERNU	Displays the Head serial number	"S" + Int 6 digit
WSENS	Displays Head sensitivity (mV/W)	"W" + Float 3int.5dec
PNOMW	Displays maximum power value the Head can withstand (W) (or maximum Energy value (J))	"R" + Float 5int.1dec
LAMBDA	Displays the selected Wavelength (nm). This is the wavelength currently selected on the **`Plus 2`** screen.	"LAMBDA" + Int 5 digit *Note: when an older (pre-2014) or custom sensor head is connected to the **`Plus 2`**, the answers to LAMBDA command is a 3 character String: CO2, ERB, YAG, LAD, VIS or EXC*
POWER	Set **`Plus 2`** in Power Meter mode (if available)	"ok"
FIT	Set **`Plus 2`** in "FIT" operation mode (if available)	"ok"
ENERGY	Set **`Plus 2`** in Energy Meter mode (if available)	"ok"
ZERO	Zeroing **`Plus 2`**	"ok"
OUTPM	Displays measured power (W) (or measured energy (J) if Energy mode is selected)	-/+ Float, up to 9 digit
TERMI	Thermistor availability: (1) yes, (2) no	"T" + Int 1 digit
TEMP	Displays Head temperature x 10 (°C)	"t" + Int 3 digit
STATUS	Displays the status byte: Bit 0: Head connected: (1) yes, (0) no Bit 1: thermistor connected: (1) yes, (0) no Bit 2: not used Bit 3: cool warning (1) Bit 4: battery: connected to AC (1) Bit 5: battery: charge in progress (1) Bit 6: overload warning (1) Bit 7: overflow warning (1) Bit 8: status "ready", for Fit/Energy mode (1) Bit 9: status "triggered", for Fit/Energy mode (1) Bit 10: status "wait", for Fit modes (1) Bit 11: not used Bit 12: overflow ADC gain G=x1 (1) Bit 13: overflow ADC gain G=x10 (1) Bit 14: overflow ADC gain G=x100 (1) Bit 15: not used	Int 5 digit, (to be converted in binary)
SETX1 0	Set x1 electronic amplifier gain	"ok"
SETX1 1	Set x10 electronic amplifier gain	"ok"
SETX1 2	Set x100 electronic amplifier gain	"ok"
SETX1 3	Set the automatic selection of the electronic amplifier gain	"ok"

X1D	Displays the selected electronic gain set up: 0: x1 gain 1: x10 gain 2: x100 gain 3: automatic gain, current x1 gain 4: automatic gain, current x10 gain 5: automatic gain, current x100 gain	Int 1 digit, from 0 to 5
FAST	Enables the acceleration algorithm	"FAST"
SLOW	Disables the acceleration algorithm, DO NOT use this command while in FIT or ENERGY mode.	"SLOW"
FASTSLOW	Returns the Fast/Slow current setting	"FAST" or "SLOW"

2.4 Error Messages

the following error message may be sent by the ***Plus 2*** if a communication error occurs: ??;

where:

?? :	USB communication error
";" :	End of answer

An error message may be sent for the following error conditions:

Input command not started with * character

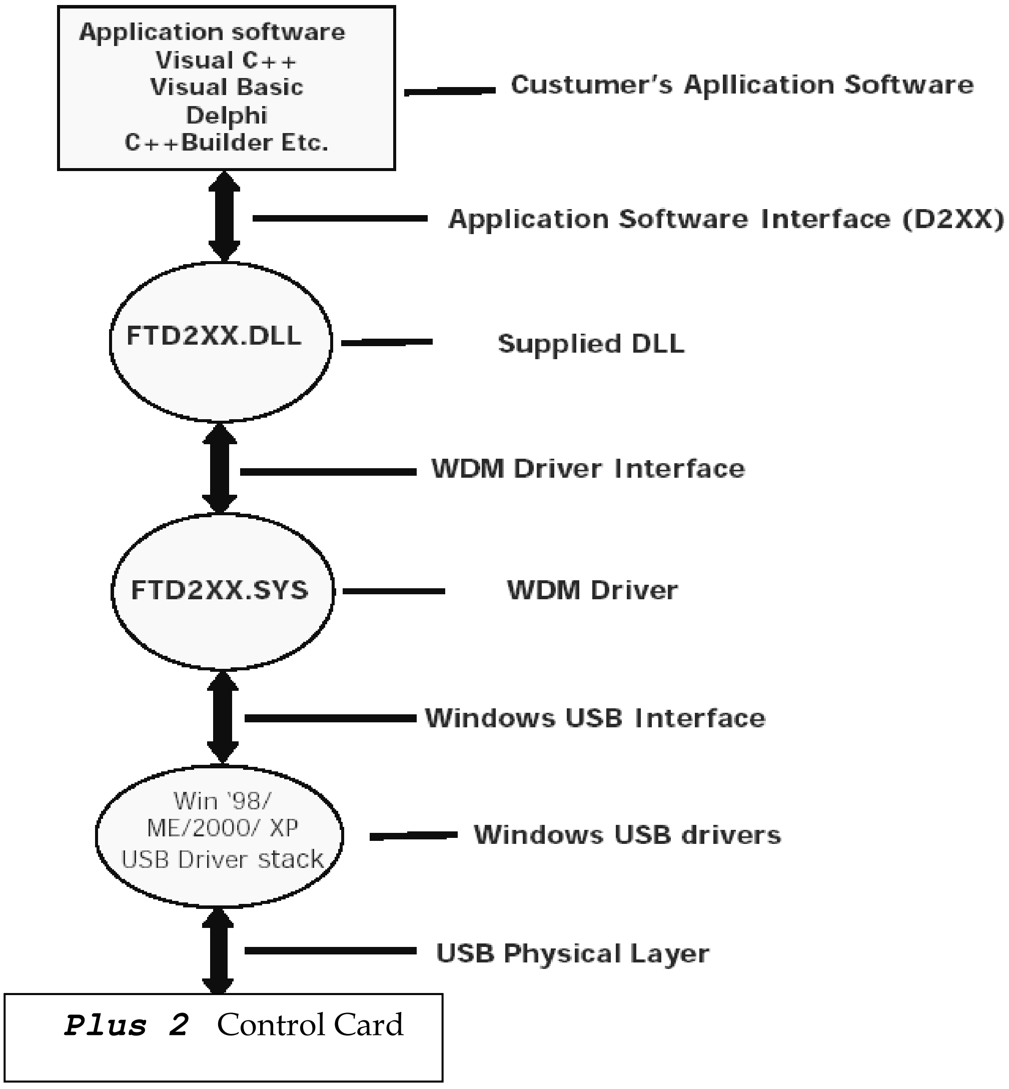
Input command does not correspond with the command list

Input command not in capitals

3 Annex 1: FTD2XX.DLL Dynamic Library

The **FTD2XX.DLL** Dynamic Library for Windows allows you to write your application.
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 ***`Plus 2`*** control card.

D2XX Driver Architecture



DLL Functions

A complete list and detailed description of FTDXXX functions is available as “D2XX Programmer's Guide” in this section of FTDI website:

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 zerobased, 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 (not described in this user manual) it is possible to open a device also trough its *serial number* or trough its description. For further information, please contact **LASERPOINT.srl.**

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 l**pdwBytesReturned** 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.

*Note: **Plus 2** Baud Rate value is 38400.*

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.

*Note: for **Plus 2** the DataCharacteristics must be set as FT_DATA_BITS_8, FT_STOP_BITS_1, FT_PARITY_NONE*

FT_SetFlowControl

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

Syntax FT_STATUS **FT_SetDataCharacteristics** (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.

*Note: for **Plus 2** the FlowControl must be set as FT_FLOW_NONE*

FT_SetDTR

Description Sets the Data Terminal Ready (DTR) control signal. (Data Terminal Ready).

Syntax FT_STATUS **FT_SetDTR** (FT_HANDLE *ftHandle*)

Parameters
ftHandle pointer to the communication *handle* of the DTR device to set out.

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

FT_ClrDTR

Description This function clears the Data Terminal Ready (DTR) control signal (*Data Terminal Ready*).

Syntax FT_STATUS **FT_ClrDTR** (FT_HANDLE *ftHandle*)

Parameters
ftHandle pointer to the communication *handle* of the DTR device to set out.

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

FT_SetRTS

Description Sets the Request To Send (RTS) control signal. (Request To Send).

Syntax FT_STATUS **FT_SetDTR** (FT_HANDLE *ftHandle*)

Parameters
ftHandle pointer to the communication *handle* of the RTS device to set out.

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

FT_ClrRTS

Description Clears the Request To Send (RTS) control signal (*Request To Send*).

Syntax FT_STATUS **FT_SetDTR** (FT_HANDLE *ftHandle*)

Parameters
FtHandle pointer to the communication *handle* of the RTS device to set out.

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

FT_SetTimeouts

Description Sets the read and write timeouts for the device.

Syntax FT_STATUS **FT_SetBaudRate** (FT_HANDLE *ftHandle*, DWORD *dwReadTimeout*, DWORD *dwWriteTimeout*)

Parameters
FtHandle pointer to the communication *handle* of the device to set out .
dwReadTimeout value of the Read timeout, in milliseconds, to set out.
dwWriteTimeout value of the Write timeout, in milliseconds, to set out.

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.*

Error codes

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

4 Examples and Notes

EXAMPLE 1

Here below are reported some examples of the main steps necessary to start the communication with a ***Plus 2*** device with FTDXXX functions. The programming language for this example is VB.NET.

```
' Get the number of the connected FTDI devices:
FT_Status = FT_GetNumberOfDevices(NumDevicesConnected, vbNullChar, FT_LIST_NUMBER_ONLY)
```

Browse all the connected FTDI devices to find Laserpoint Plus 2 device

```
' 1 Get the device description
FT_Status = FT_GetDeviceString(i, Description, FT_LIST_BY_INDEX Or FT_OPEN_BY_DESCRIPTION)

' 2 Shrink the description returned as 64 chars string to the correct number of chars.
' Ex: "Plus2                " --> "Plus2"
Description = Microsoft.VisualBasic.Left(Description, InStr(1, Description, vbNullChar) - 1)
```

NOTE: Description of Laserpoint Plus 2 device is "**Plus2**"

Get the serial number of Laserpoint device using the description

```
' Get serial number of device using index
FT_Status = FT_GetDeviceString(i, Serial, FT_LIST_BY_INDEX Or FT_OPEN_BY_SERIAL_NUMBER)

' Shrink the description from 64 chars to the correct number of chars
' Ex: "123456                " --> "123456"
Serial = Microsoft.VisualBasic.Left(Serial, InStr(1, Serial, vbNullChar) - 1)
```

```
' Open communication with device identified by its serial number. Function will return a communication Handle which will
' be used for all the following communications
FT_Status = FT_OpenBySerialNumber(Serial, FT_OPEN_BY_SERIAL_NUMBER, COM_Handle)
```

```
' Setting communication parameters
FT_Status = FT_SetBaudRate(COM_Handle, 38400)
FT_Status = FT_SetDataCharacteristics(COM_Handle, FT_DATA_BITS_8, FT_STOP_BITS_1, FT_PARITY_NONE)
FT_Status = FT_SetFlowControl(COM_Handle, FT_FLOW_NONE, 0, 0)
FT_Status = FT_SetTimeouts(COM_Handle, 500, 100)
FT_Status = FT_Purge(COM_Handle, FT_PURGE_RX Or FT_PURGE_TX)
```

```
' Read and write to serial port
FT_Status = FT_Write_String(COM_Handle, "*COMMAND:", 9, byteswritten)

' Set a delay of 50ms between write and read function to wait for device to reply
System.Threading.Thread.Sleep(50)

' Read the response from the device
FT_Status = FT_GetQueueStatus(COM_Handle, RXBytes)
ReadString = Space(RXBytes)
FT_Status = FT_Read_String(COM_Handle, ReadString, RXBytes, bytesread)
```

5 Useful links:

For a complete list and description of FTDXXX functions, please download the “FTD2XX Programmer's Guide” at this link:
http://www.ftdichip.com/Support/Documents/ProgramGuides.htm

To download the right libraries for your Operative System / architecture please check this section of FTDI website:
http://www.ftdichip.com/Drivers/D2XX.htm

For software examples with different programming languages please check this section:
http://www.ftdichip.com/Support/SoftwareExamples/CodeExamples.htm