TE USB FX2 Technology Stack Overview

Authors: Sergio Pavesi, Fabio De Riccardis
Revision: 1.1
Date: 05-Jul-2013 16:52

Table of Contents

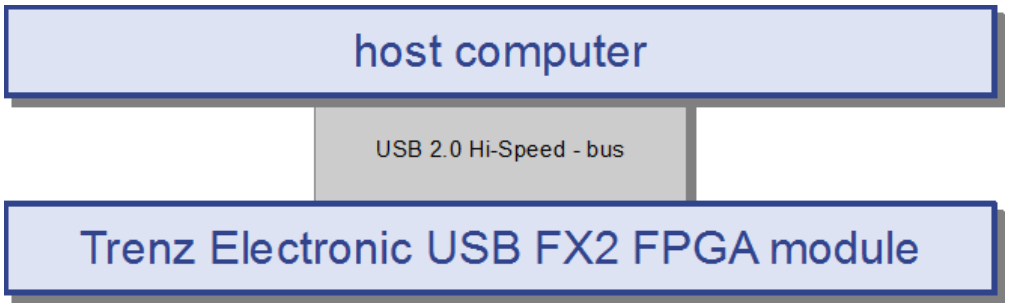
This guide provides an outline of the Trenz Electronic USB FX2 Technology Stack and a quick guide to its resources. A PDF export of this document is available in the Trenz Electronic download area.

1 Document Change History

date	revision	authors	description
2013-04-17	1.1	Fabio De Riccardis	Added pictures and captions.
2013-04-08	1.0	Sergio Pavesi, Fabio De Riccardis	Initial release.
	All	Sergio Pavesi, Fabio De Riccardis	

2 Technology Stack Outline

Trenz Electronic USB FX2 FPGA modules are devices that support USB 2.0 Hi-Speed communication with a host computer.



TE USB FX2 system overview.

This document gives an overview of the USB FX2 technology stack supported by Trenz Electronic FPGA modules equipped with Cypress EZ-USB FX2 microcontroller (currently: TE0300, TE0320 and TE0630).

2.1 Generations

There are two generations of Trenz Electronic USB FX2 FPGA modules. The following table summarizes the main differences.

generation	2	3
hardware	same	same
reference architecture	same	same
firmware	same	same
VID	*0x0547*	*0x0BD0*
PID	*0x1002*	*0x0300*
device driver family	*DEWESoft*	*Cypress EZ-USB*
API(s) family	*DEWESoft (C++)*	*Cypress (C++, .NET)*
reference application	*DEWESoft (C++)*	*Trenz Electronic (C++, .NET)*
recovery USB firmware tools	Cypress USB Console, Cypress USB Control Center	Cypress USB Console, Cypress USB Control Center
regular USB firmware tools	DEWESoft FUT Open_FUT (generation 2)	Cypress USB Console, Cypress USB Control Center Open_FUT (generation 3)
recovery FPGA bitstream tool	Xilinx iMPACT	Xilinx iMPACT

generation	2	3
regular FPGA bitstream tool	Xilinx iMPACT, DEWESoft FUT, Open_FUT (generation 2)	Xilinx iMPACT, Open_FUT (generation 3)

Technology stack generation comparison table.

Trenz Electronic modules can be used with both couples of driver/API:

- DEWESoft device driver + DEWESoft API,
- Cypress device drivers + Trenz Electronic API(s),

but **not** a mix of the two:

- DEWESoft device driver + Trenz Electronic API(s),
- Cypress device driver + DEWESoft API.

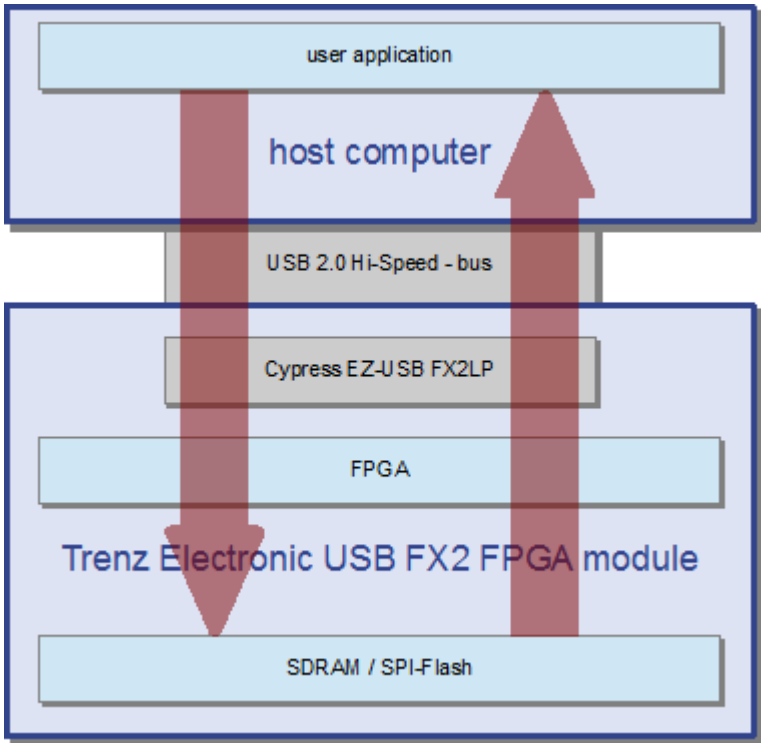
Modules of both generations are factory programmed and tested with an open source **reference architecture**.

2.2 Capabilities

The reference architecture allows users to

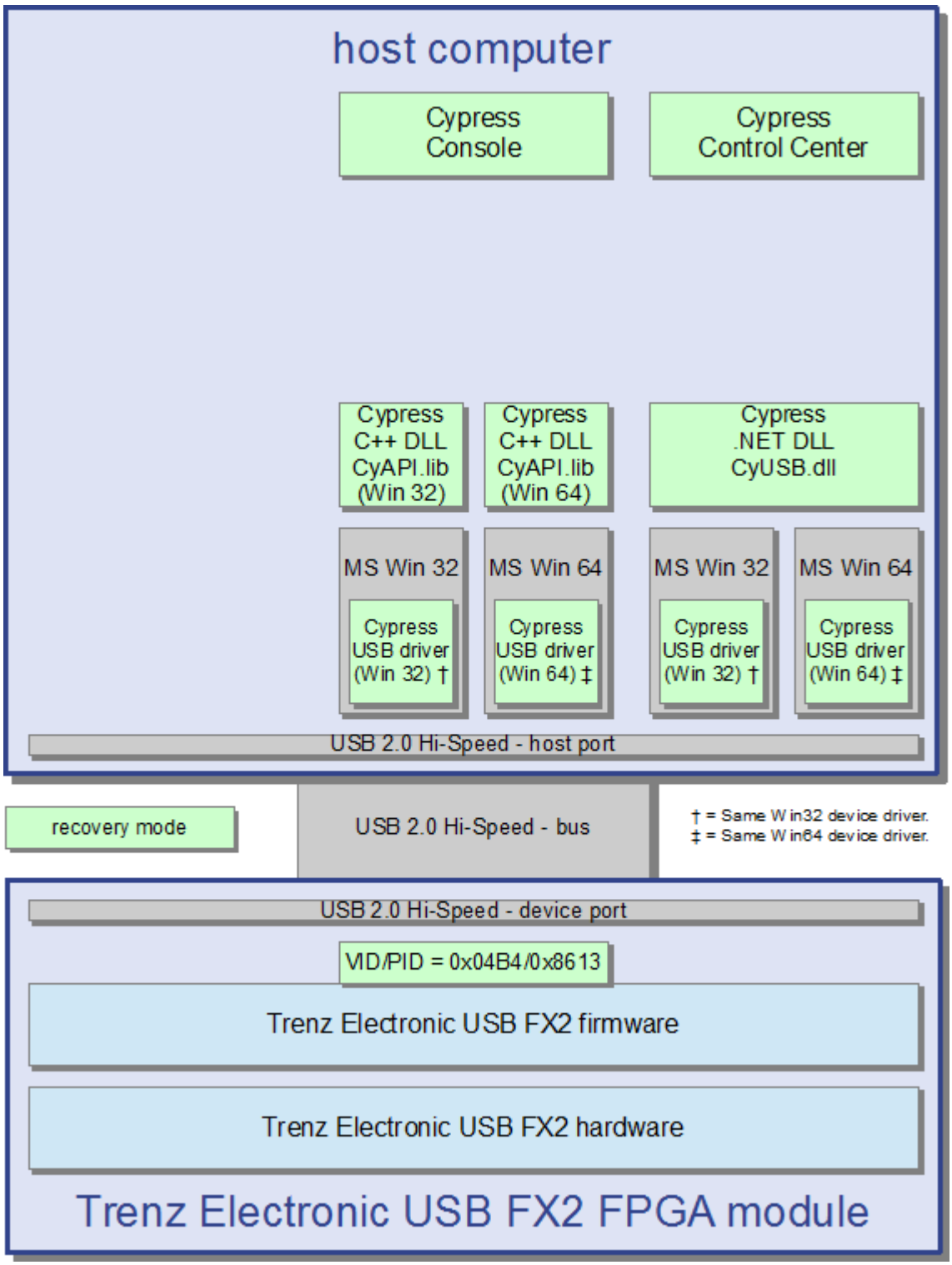
- read and write the module SDRAM,
- read and write the Flash EEPROM,
- read and write the USB microcontroller EEPROM,
- read firmware version,
- reconfigure the FPGA

from a host application.



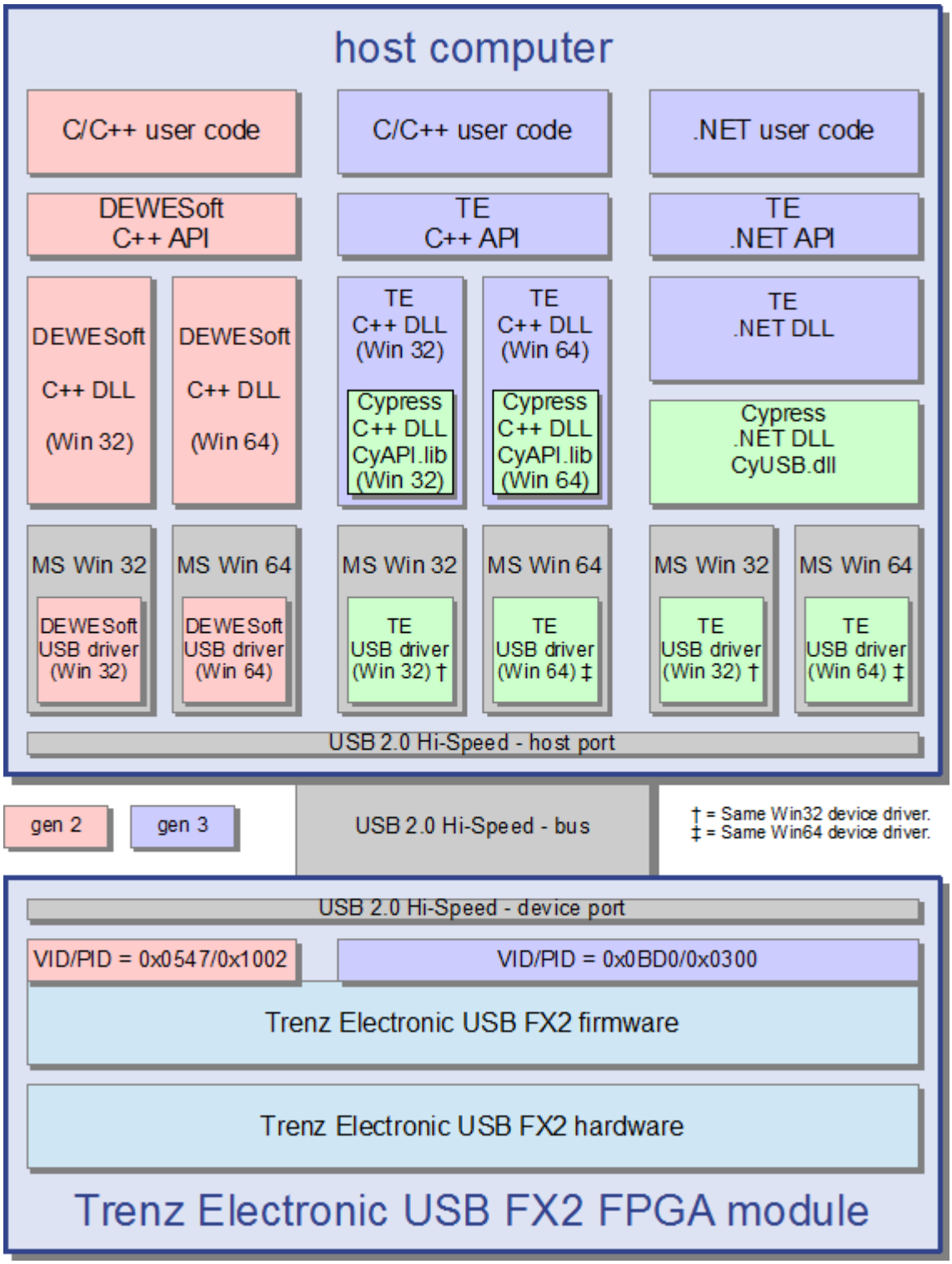
Sample application block diagram.

2.3 Technology Stack Overview (Recovery Mode)



Technology stack block diagram - recovery mode.

2.4 Technology Stack Overview (Regular Mode)



Technology stack block diagram - regular mode.

3 Hardware Layer (Generation 2 = Generation 3)

All TE USB FX2 modules feature the same Cypress EZ-USB FX2LP USB 2.0 microcontroller and high speed USB peripheral controller.
Generation 2 and generation 3 technology stacks share the same hardware.

3.1 TE0300 Series

FPGA:

- Xilinx Spartan-3E (1200, 1600)

Baseboards:

- TE0303: Prototyping Baseboard for Trenz Electronic TE0300 and TE0630 Micromodules
- TE0304: Application Baseboard for Trenz Electronic TE0300 and TE0630 Micromodules

Documentation:

- http://www.trenz-electronic.de/te0300
- http://www.trenz-electronic.de/download/d0/Trenz_Electronic/d1/TE0300_series.html

Reference designs using Xilinx ISE 13.3 and Xilinx EDK 13.3:

- https://github.com/Trenz-Electronic/TE03XX-Reference-Designs/
- http://www.trenz-electronic.de/download/d0/Trenz_Electronic/d1/TE0300_series/d2/TE0300/d3/reference_designs.html

3.2 TE0320 Series

FPGA:

- Xilinx Spartan-3A DSP (1800, 3400)

Baseboards:

- TE0323 Prototyping Baseboard for Trenz Electronic TE0320 Micromodules

Documentation:

- http://www.trenz-electronic.de/te0320
- http://www.trenz-electronic.de/download/d0/Trenz_Electronic/d1/TE0320_series.html

Reference design using Xilinx ISE 13.3and Xilinx EDK 13.3:

- https://github.com/Trenz-Electronic/TE03XX-Reference-Designs

- http://www.trenz-electronic.de/download/d0/Trenz_Electronic/d1/TE0320_series/d2/TE0320/d3/reference_designs.html

3.3 TE0630 Series

FPGA:

- Xilinx Spartan-6 LX (45, 75, 100, 150)

Documentation:

- http://www.trenz-electronic.de/te0630
- http://www.trenz-electronic.de/download/d0/Trenz_Electronic/d1/TE0630_series.html

Baseboards:

- TE0303 Prototyping Baseboard for Trenz Electronic TE0300 and TE0630 Micromodules
- TE0304 Application Baseboard for Trenz Electronic TE0300 and TE0630 Micromodules

Documentation:

- http://www.trenz-electronic.de/download/d0/Trenz_Electronic/d1/TE0630_series.html

Reference designs using Xilinx ISE 13.2 and Xilinx EDK 13.2:

- https://github.com/Trenz-Electronic/TE063X-Reference-Designs/
- http://www.trenz-electronic.de/download/d0/Trenz_Electronic/d1/TE0630_series/d2/TE0630/d3/reference_designs.html

3.4 Difference Between TE0300 and TE0320/TE0630

The amount of data that the user can transfer in each packet (PacketSize) depends on the Trenz Electronic USB FX2 module used:

- TE0300: PacketSize 51,200 bytes;
- TE0320, TE0630: PacketSize 102,400 bytes.

If the user wishes to transfer data or files larger than the above sizes, he/she must decompose them into smaller packets before transfer and recompose them after transfer.

4 Reference Architecture Layer (Generation 2 = Generation 3)

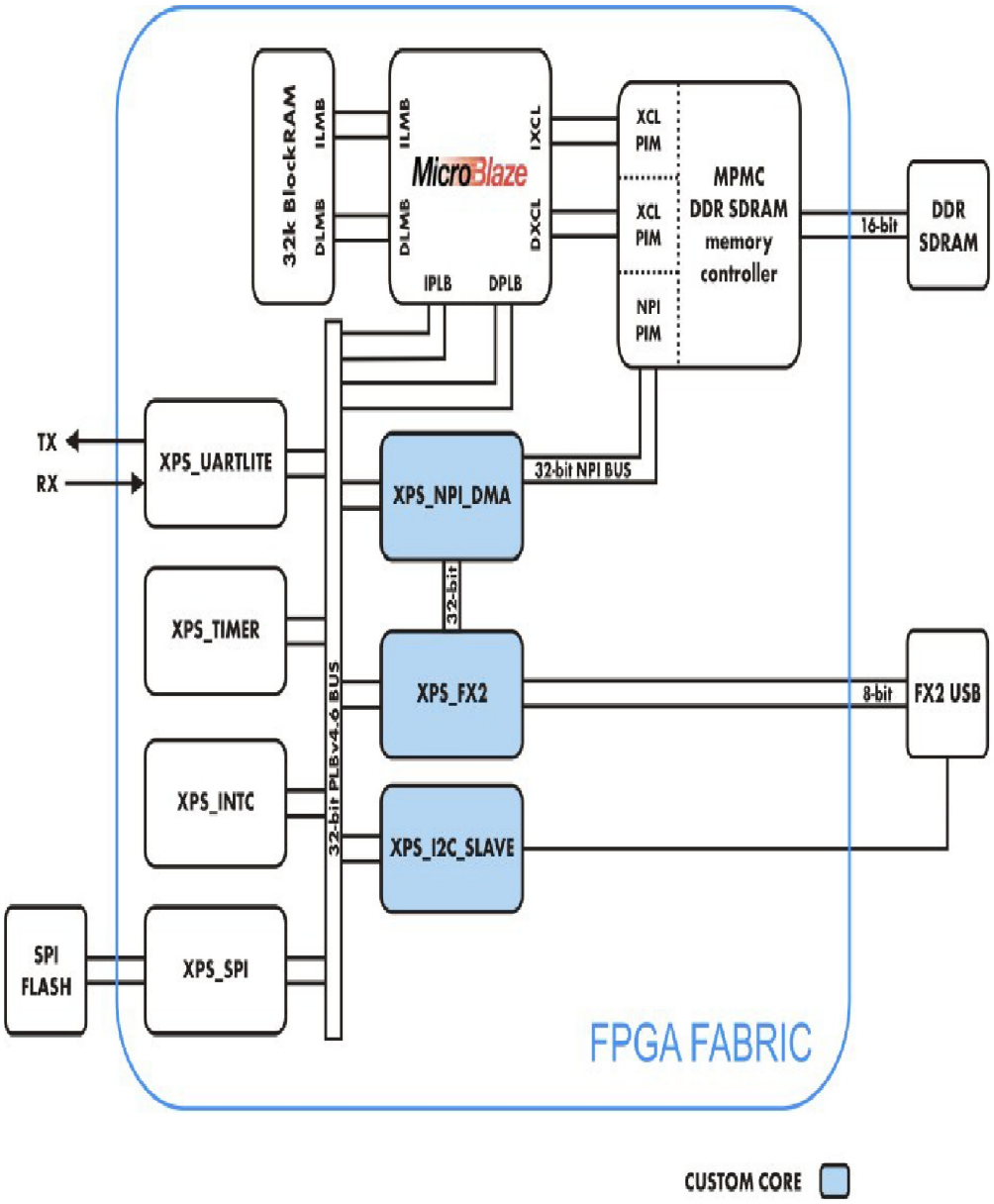
The Xilinx FPGA itself on the Trenz Electronic USB FX2 modules is blank when powered off. To define an FPGA functionality, a logic architecture should be defined and loaded into the device at power-on.

The reference design architecture was built using Xilinx Embedded Development Kit (EDK). Basically, it is an embedded system on a chip with a Xilinx MicroBlaze 32-bit soft microprocessor. The MicroBlaze i(MB) nitializes and sets up the system.

Standard EDK cores are used to implement a serial interface (XPS_UARTLITE), an SPI FLASH interface (XPS_SPI), a timer / counter block (XPS_TIMER) and an interrupt controller (XPS_INTC).

Besides standard IP cores, the reference architecture contains three custom IP cores:

1. XPS_I2C_SLAVE: forwards commands coming from the USB bus towards the MicroBlaze;
2. XPS_NPI_DMA: custom DMA between DDR SDRAM and other multiple sources;
3. XPS_FX2: used for high speed bidirectional communication between the FPGA and a host PC.



Reference architecture block diagram.

4.1 XPS_I2C_SLAVE custom IP block

It is a logic block for low speed bidirectional communication between the FPGA and a host PC. It is usually used for command, settings and status communication. It contains 6 × 32-bit memory mapped registers:

- 3 for *host => FPGA* communication (FX22MB registers)
- 3 for *FPGA => host* communication (MB2FX2 registers)

When the host sends commands to the MicroBlaze (MB) soft embedded processor, an interrupt is triggered. When the MB writes data to MB2FX2_reg0, the interrupt (INT0) is sent to the Cypress EZ-USB FX2LP USB microcontroller. When the FX2 microcontroller receives an interrupt, it reads all MB2FX2 registers.

4.2 XPS_NPI_DMA custom IP block

It is a high speed DMA (direct memory access) engine which connects to the MPMC (Multi-Port Memory Controller) VFBC (Video Frame Buffer Controller) port. It enables high speed data streaming to/from external memory (DDR SDRAM) and multiple sources. It can be controlled by a processor using 6 × 32-bit memory mapped registers attached to the PLB (peripheral local bus). For more information about registers, see *Xilinx LogiCORE IP Multi-Port Memory Controller (MPMC)* data sheets (Xilinx DS643), *Video Frame Buffer Controller PIM* section.
When data is sent from the USB-host to a USB FX2 module high-speed endpoint (high speed communication channel), it is automatically stored into the RAM by the custom built DMA engine (XPS_NPI_DMA) at a specified buffer location. The reference design software running on the MicroBlaze verifies the transferred data at the end of transmission and sends to the USB host a notification about the data test (passed / failed).

4.3 XPS_FX2 custom IP block

It is a logic block for high speed bidirectional communication between the FPGA and a host PC. It contains 2 kbyte FIFOs for data buffering. More information about the 5 × 32-bit memory mapped registers is contained in the *#project_root#/pcores/xps_fx2_v1_00_a/doc/* folder of the reference design project folder.
When data is sent form a USB FX2 module high-speed endpoint to the USB host, it is automatically fetched from the RAM via the custom DMA engine (XPS_NPI_DMA)and forwarded to the XPS_FX2 core in 1-kbyte packets. MicroBlaze throttles the throughput to prevent XPS_FX2 TX FIFO overflow.

5 Firmware Layer (Generation 2 Generation 3)

Generation 2 and generation 3 technology stacks share the same firmware but with different VIDs/PIDs.
The following firmware tools

- DEWESoft FUT (firmware upgrade tool)
- Open_FUT (generation 2)
- Open_FUT (generation 3)

work only if the reference architecture, or a derived architecture, is running in the module. The reference/derived architecture is necessary because the three tools make use of *API Commands* (see the corresponding API reference manual) executed by the Xilinx MicroBlaze soft embedded processor of the reference/derived architecture. These tools are therefore able to update the firmware of the USB FX2 microcontroller and the FPGA configuration file (bitstream).
The following firmware tools

- Cypress USB Console
- Cypress USB Control Center

work also if the reference/derived architecture is not running in the module. he reference/derived architecture is not necessary because they do not make use of *API Commands*. Conversely, they directly make use of CyAPI.lib and CyUSB.dll, respectively. These two tools are therefore only able to update the firmware of the USB FX2 microcontroller, but not the FPGA configuration file (bitstream).

5.1 Generation 2

VID/PID:

- 0x0547/0x1002

Firmware files:

- TE-USB-FX2_current_DW.iic https://github.com/Trenz-Electronic/TE-USB-Suite/tree/master/TE_USB_FX2.firmware/ready_for_download

Firmware update tools:

- DEWESoft Firmware Upgrade Tool:It requires DEWESoft device driver and DEWESoft API.It work only if the reference/derived architecture is running in the module.DEWESoft FUT is able to update the firmware of the USB FX2 microcontroller and the FPGA configuration file (fpga.bin bitstream).
- Open_FUT for generation 2:It requires DEWESoft device driver, Python and DEWESoft API.It works only if the reference/derived architecture is running in the module.Open_FUT is able to update the firmware of the USB FX2 microcontroller and the FPGA configuration file (bitstream fpga.bin).

- CyConsole = Cypress USB Console:It requires Cypress generic USB device driver and CyAPI.lib static library.It works also if the reference/derived architecture is not running in the module.It is used as recovery USB firmware tool (recovery boot)It requires to boot with EEPROM switched off (DEWESoft device is registered as Cypress device), then back on (EEPROM can be written).Some documentation in UM-USB-Firmware.pdf.Some examples here (firmware restore only).
- CyControl = Cypress USB Control Center (requires .NET Framework):It requires Cypress generic USB device driver, .Microsoft NET Framework (version 4.0.30319) and CyUSB.dll dynamic library.It works also if the reference/derived architecture is not running in the module.It is used as recovery USB firmware tool (recovery boot)It requires to boot with EEPROM switched off (DEWESoft device is registered as Cypress device), then back on (EEPROM can be written).Some examples here (firmware restore only).

Documentation

- CyConsole = Cypress USB Console: http://www.trenz-electronic.de/download/d0/Trenz_Electronic/d1/TE-USB-Suite/d2/generation_2/d3/documents.html

5.2 Generation 3

VID/PID:

- 0x0BD0/0x0300

Firmware files:

- TE-USB-FX2_current_TE.iic https://github.com/Trenz-Electronic/TE-USB-Suite/tree/master/TE_USB_FX2.firmware/ready_for_download

Firmware update tools:

- Open_FUT for generation 3:It requires TE USB FX2 device driver, Python and a pure *extern C* code library version of TE_USB_FX2_CyAPI APIs (codename: *simplified TE_USB_FX2_CyAPI APIs*).It works only if the reference/derived architecture is running in the module.Open_FUT is able to update the firmware of the USB FX2 microcontroller and the FPGA configuration file (bitstream fpga.bin).
- CyConsole = Cypress USB Console:It requires Cypress generic USB device driver and CyAPI.lib static library.It works also if the reference/derived architecture is not running in the module.Boot with EEPROM switched off is not required.Some documentation in UM-USB-Firmware.pdf.Some examples here (firmware restore) and here (firmware update).
- CyControl = Cypress USB Control Center (requires .NET Framework);It requires Cypress generic USB device driver, .Microsoft NET Framework (version 4.0.30319) and CyUSB.dll dynamic library.It works also if the reference/derived architecture is not running in the module.Boot with EEPROM switched off is not required.Some examples here (firmware restore) and here (firmware update).

Documentation

- CyConsole = Cypress USB Console: http://www.trenz-electronic.de/download/d0/Trenz_Electronic/d1/TE-USB-Suite/d2/generation_2/d3/documents.html

5.3 Recovery Boot

A *recovery boot* is a multi-step boot operation:

1. module is off and connected;
2. EEPROM switch is set off;
3. module is powered on;
4. module enumerates in recovery mode (VID = 0x04B4, PID = 0x8613);the module is served by the Cypress generic USB device driver;the Cypress generic USB device driver allows Cypress firmware update tools (Cypress Console, Cypress Control Center) to work with the Cypress EZ-USB FX2 microcontroller on the module;
5. EEPROM switch is set back on;Cypress firmware update tools can read and write the EEPROM.

Using Cypress firmware update tools with generation **2** modules does require a recovery boot, in order to force enumeration as Cypress generic USB device driver.
On the other side, using Cypress firmware update tools with generation **3** modules does not require a recovery boot. This is possible because the original Cypress generic USB device driver (in case of a recovery boot) and the Trenz Electronic USB FX2 device driver (in case of a regular boot) are both Cypress driver. Trenz Electronic USB FX2 driver derives from the original Cypress generic USB device driver.
EEPROM switch shall always be on during EEPROM programming.
The firmware update procedure is similar for both generations. The following table summarizes the main differences.

generation	**2: recovery boot**	**3: recovery boot**	**3: regular boot**
boot mode (EEPROM switch)	recovery (off, then on)	recovery (off, then on)	regular (always on)
VID	04B4	04B4	0BD0
PID	8613	8613	0300
device	Cypress generic USB	Cypress generic USB	TE USB FX2
required driver	recovery driver	recovery driver	regular driver
update applications	Cypress USB Console Cypress USB Control Center		

Firmware update comparison table.
Further information is shown in the following video play lists.

Generation 2 to generation 3 migration (firmware + driver)  video play list.	Migration from the second generation (aka DEWESoft) to the third generation (aka TE USB FX2) of Trenz Electronic USB FX2 technology stack (firmware and driver).
	TE0300: migration from 2nd to 3rd generation (firmware + driver) This video shows how to upgrade a Trenz Electronic TE0300 device form the second to the third generation (firmware and driver). The video has been recorded on a Microsoft Windows 7 (64 bit) operating system with a TE0300-01 FPGA module connected to a USB port, but the procedure is almost the same for any Trenz Electronic USB FX2 device. In this video, power-on reset is used. Please note that TE0300 has master reset, thus a powered reset (S2 switch) is possible.
	TE0320: migration from 2nd to 3rd generation (firmware + driver) This video shows how to upgrade a Trenz Electronic TE0320 device form the second to the third generation (firmware and driver). The video has been recorded on a Microsoft Windows 7 (64 bit) operating system with a TE0320 FPGA module connected to a USB port, but the procedure is almost the same for any Trenz Electronic USB FX2 device. In this video, power-on reset is used. Please note that TE0320 has master reset, thus a powerered reset (S1D switch to ON) is possible.
	TE0630: migration from 2nd to 3rd generation (firmware + driver) This video shows how to upgrade a Trenz Electronic TE0630 device form the second to the third generation (firmware and driver). The video has been recorded on a Microsoft Windows 7 (64 bit) operating system with a TE0630-00 FPGA module connected to a USB port, but the procedure is almost the same for any Trenz Electronic USB FX2 device. Please note that, unlike TE0300 and TE0320, TE0630-00 has no -master reset-, thus a power-on reset is required.

	Cypress USB Console used for 2nd to 3rd generation update. This video explains how to use -Cypress USB Console- instead of "Cypress Control Center". This video assumes that you have followed the first 6 minutes of "generation 2 to generation 3" videos (for anyone of the Trenz Electronic USB FX2 modules). This video is an extension to 6:00 to 6:40 of "generation 2- to -generation 3" videos.

Generation 3 TE USB FX2 (firmware update)  video play list	Update of TE USB FX2´s firmware (generation 3) from an old one to a new one.
	Update USB FX2 microcontroller firmware (generation 3), using "Cypress USB Console" This video shows how to update a Trenz Electronic device form generation 3 (old firmware) to generation 3 (new firmware), using "Cypress USB Console". The video has been recorded on a Microsoft Windows 7 (64 bit) operating system with a TE0630-00 FPGA module connected to a USB port, but the procedure is almost the same for any Trenz Electronic USB FX2 device. Please note that unlike TE0300 and TE0320, TE0630-00 has no master-reset, thus a power-on reset is required.
	Update USB FX2 microcontroller firmware (generation 3) using "Cypress USB Control Center" This video shows how to update a Trenz Electronic device form generation 3 (old firmware) to generation 3 (new firmware), using "Cypress USB Control Center". The video has been recorded on a Microsoft Windows 7 (64 bit) operating system with a TE0630-00 FPGA module connected to a USB port, but the procedure is almost the same for any Trenz Electronic USB FX2 device. Please note that unlike TE0300 and TE0320, TE0630-00 has no master-reset, thus a power-on reset is required.

	Powered Reset procedure for TE0300 and TE0320 (based on "Update USB FX2 microcontroller firmware" videos) This video assumes that you have followed the "Update USB FX2 microcontroller firmware (generation 3) using Cypress USB Control Center" video and implements the "powered reset" as reset (1:31 to 2:14 of the previous video). This video shows how to realize a powered reset for a TE0320 or a TE0300 device. The video has been recorded on a Microsoft Windows 7 (64 bit) operating system with a TE0300-01 FPGA module connected to a USB port. Please note that, unlike TE0300 and TE0320, TE0630 has no master reset switch, thus a power-on reset is required.

6 Device Driver Layer

6.1 Generation 2

6.1.1 Device driver files

- TE03xx-USB_32-64.zip

Generation 2 device drivers are provided by DEWESoft and can also be downloaded from here or here.

6.2 Generation 3

6.2.1 Device driver files

- TE_USB_FX2-drivers.zip

6.2.2 Documentation

- *USB Drivers Installation - User Manual* (UM-drivers-TE_USB_FX2.pdf)
- video play list: Microsoft Windows XP 32 bit and 64 bit;
- video play list: Microsoft Windows Vista 32 bit;
- video play list: Microsoft Windows 7 64 bit;
- video play list: Microsoft Windows 8 64 bit.

generation 3 TE USB FX2 - Win XP 32  video play list	Installation and removal of a Trenz Electronic USB FX2 (third generation) device driver on a Microsoft Windows XP (32 bit) operating system connected to Trenz Electronic USB FX2 device with third generation v. 3.x firmware. Please note how Windows XP (32 bit) is less demanding than Windows XP (64 bit) with regard to device driver signatures.
	TE USB FX2 device driver - Windows XP 32 - first installation • operating system: Microsoft Windows XP (32 bit) • VID / PID = 0x0BD0 / 0x0300 • driver: TE_USB_FX2 (Trenz Electronic USB FX2 device driver) • note: first-time installation on a fresh operating system

	TE USB FX2 device driver - Windows XP 32 - following installations • operating system: Microsoft Windows XP (32 bit) • VID / PID = 0x0BD0 / 0x0300 • driver: TE_USB_FX2 (Trenz Electronic USB FX2 device driver) • note: following installations on a fresh operating system
	TE USB FX2 device driver - Windows XP 32 - removal • operating system: Microsoft Windows XP (32 bit) • VID / PID = 0x0BD0 / 0x0300 • driver: TE_USB_FX2 (Trenz Electronic USB FX2 device driver) • note: removal from a fresh operating system

generation 3 TE USB FX2 - Win XP 64  video play list	Installation and removal of a Trenz Electronic USB FX2 (third generation) device driver on a Microsoft Windows XP (64 bit) operating system connected to a Trenz Electronic USB FX2 device with third generation v. 3.x firmware. Please note how Windows XP (64 bit) is more demanding than Windows XP (32 bit) with regard to device driver signatures.
	TE USB FX2 device driver - Windows XP 64 - first installation • operating system: Microsoft Windows XP (64 bit) • VID / PID = 0x0BD0 / 0x0300 • driver: TE_USB_FX2 (Trenz Electronic USB FX2 device driver) • note: first-time installation on a fresh operating system
	TE USB FX2 device driver - Windows XP 64 - following installations • operating system: Microsoft Windows XP (64 bit) • VID / PID = 0x0BD0 / 0x0300 • driver: TE_USB_FX2 (Trenz Electronic USB FX2 device driver) • note: following installations on a fresh operating system
	TE USB FX2 device driver - Windows XP 64 - removal • operating system: Microsoft Windows XP (64 bit) • VID / PID = 0x0BD0 / 0x0300 • driver: TE_USB_FX2 (Trenz Electronic USB FX2 device driver) • note: removal from a fresh operating system

generation 3 TE USB FX2 - Win Vista 32  video play list	Installation and removal of a Trenz Electronic USB FX2 (third generation) device driver on a Microsoft Windows Vista (32 bit) operating system connected to a Trenz Electronic USB FX2 device with third generation v. 3.x firmware. The device driver on a Microsoft Windows Vista (64 bit) behaves like on a Microsoft Windows Vista (32 bit).
	TE USB FX2 device driver - Windows Vista 32 - manual installation • operating system: Microsoft Windows Vista (32 bit) • VID / PID = 0x0BD0 / 0x0300 • driver: TE_USB_FX2 (Trenz Electronic USB FX2 device driver) • note: manual installation on a fresh operating system
	TE USB FX2 device driver - Windows Vista 32 - removal • operating system: Microsoft Windows Vista (32 bit) • VID / PID = 0x0BD0 / 0x0300 • driver: TE_USB_FX2 (Trenz Electronic USB FX2 device driver) • note: removal from a fresh operating system

generation 3 TE USB FX2 - Win 7 64  video play list	Installation and removal of a Trenz Electronic USB FX2 (third generation) device driver on a Microsoft Windows 7 (64 bit) operating system connected to Trenz Electronic USB FX2 device with third generation v. 3.x firmware.
	TE USB FX2 device driver - Windows 7 64 - installation settings • operating system: Microsoft Windows 7 (64 bit) • VID / PID = 0x0BD0 / 0x0300 • driver: TE_USB_FX2 (Trenz Electronic USB FX2 device driver) • note: recommended driver installation settings
	TE USB FX2 device driver - Windows 7 64 - first automatic installation • operating system: Microsoft Windows 7 (64 bit) • VID / PID = 0x0BD0 / 0x0300 • driver: TE_USB_FX2 (Trenz Electronic USB FX2 device driver)

	• note: first-time automatic installation on a fresh operating system
	TE USB FX2 device driver - Windows 7 64 - following automatic installations • operating system: Microsoft Windows 7 (64 bit) • VID / PID = 0x0BD0 / 0x0300 • driver: TE_USB_FX2 (Trenz Electronic USB FX2 device driver) • note: following automatic installations on a fresh operating system
	TE USB FX2 device driver - Windows 7 64 - removal • operating system: Microsoft Windows 7 (64 bit) • VID / PID = 0x0BD0 / 0x0300 • driver: TE_USB_FX2 (Trenz Electronic USB FX2 device driver) • note: removal from a fresh operating system

generation 3 TE USB FX2 - Win 8 64  video play list	Installation and removal of a Trenz Electronic USB FX2 (third generation) device driver on a Microsoft Windows 8 (64 bit) operating system connected to a Trenz Electronic USB FX2 device with third generation v. 3.x firmware.
	TE USB FX2 device driver - Windows 8 64 - automatic installation • operating system: Microsoft Windows 8 (64 bit) • VID / PID = 0x0BD0 / 0x0300 • driver: TE_USB_FX2 (Trenz Electronic USB FX2 device driver) • note: automatic installation on a fresh operating system
	TE USB FX2 device driver - Windows 8 64 - manual installation • operating system: Microsoft Windows 8 (64 bit) • VID / PID = 0x0BD0 / 0x0300 • driver: TE_USB_FX2 (Trenz Electronic USB FX2 device driver) • note: manual installation on a fresh operating system
	TE USB FX2 device driver - Windows 8 64 - removal • operating system: Microsoft Windows 8 (64 bit) • VID / PID = 0x0BD0 / 0x0300 • driver: TE_USB_FX2 (Trenz Electronic USB FX2 device driver) • note: removal from a fresh operating system

6.2.3 Description

The driver package contains many folders. During installation, the user shall specify the folder according to the following table.

host operating system (Microsoft Windows)	device driver folder	automatic/online installation/update
2000	MS-Windows-2000.recommended	not available
XP	MS-Windows-XP.recommended	not available
Vista	MS-Windows-Vista+7	not available
7	MS-Windows-Vista+7	available
8	MS-Windows-Vista+7	available

Reccommended device driver folder depending on operating system.
Trenz Electronic device drivers are derived from the original Cypress generic USB device drivers, distributed with the Cypress SuiteUSB 3.4 (or later).
Driver installation on Microsoft Window 7 and Microsoft Window 8 can be performed automatically (on-line). It can happen that the first online driver installation attempt fails; in this case the user shall just force a retry (the second attempt normally works!). See section *2.1.2.3.2 Force the online search* of USB Drivers Installation - User Manual.

6.3 Coexistence

Installation of both device drivers on the same host computer is normally possible.
Installation of both device drivers on the same host is not recommended under Microsoft Widows XP.
Please read *USB Drivers Installation - User Manual* (UM-drivers-TE_USB_FX2.pdf) for further details.

7 API Layer

APIs require a device driver of the same generation.
Beyond the APIs (used for the communication between the host computer and the USB FX2 microcontroller), another set of API exists: they are called "API commands". The application sends API commands to the Xilinx MicroBlaze embedded processor by means of the following API functions:

- generation 2: TE0300_*SendCommand(…, command, …)*;
- generation 3: TE_USB_FX2{_}*SendCommand(..., command, …)*.

The Xilinx MicroBlaze soft embedded processor executes then the command. Skilled users can extend the API command set themselves.

7.1 Generation 2

DLL files:

- DEWESoft_API

Documentation:

- UM-TE_USB_API.pdf

This API

- uses a C interface, and therefore can be easily bound to Python;
- **uses handles;**
- cannot set any *time-out* for SetData() function. In fact, it is internally set to 1000 ms.
- cannot set the driver buffer dimension;
- is closed source, and therefore cannot be extended or enhanced.

A binding from Python to DEWESoft TE USB API is possible through the *ctypes* function library.

7.2 Generation 3

Trenz Electronic USB FX2 APIs

1. TE_USB_FX2_CyAPI API for C/C++
2. TE_USB_FX2_CyUSB API for .NET Framework

are based on the Cypress CyUSB.sys device driver (renamed as TE_USB_FX2_xx.sys) and respectively on

1. Cypress CyAPI APIs for C/C++
2. Cypress CyUSB API for .NET Framework

distributed with the Cypress SuiteUSB 3.4 (or later).
Trenz Electronic USB FX2 APIs do not use handles.

7.2.1 TE_USB_FX2_CyAPI APIs (C++)

API files:

- TE_USB_FX2_CyAPI
  TE_USB_FX2_CyAPI.dll is open source but CyAPI.lib is not. Nevertheless, its source code can be requested from Cypress under non disclosure agreement.

Documentation:

- UM-TE_USB_API.cpp.pdf
- UM-TE_USB_API.cpp.PortingGuide.pdf

Cypress *CyAPI* (CyAPI.lib) APIs for C/C++:

- are static libraries (embedded in TE_USB_FX2_CyAPI.dll and therefore **not** required by the user program);
- can be used by both managed (with .NET) C++ code and by unmanaged (without .NET) C++ code;
- have one version for Microsoft Windows 32 bit;
- have one version for Microsoft Windows 64 bit;
- have same name to ease code portability.

Trenz Electronic TE_USB_FX2_CyAPI.dll's for C/C++:

- are dynamic libraries;
- can be used by both managed (with .NET) C++ code and by unmanaged (without .NET) C++ code;
- have one version for Microsoft Windows 32 bit;
- have one version for Microsoft Windows 64 bit;
- have same name to ease code portability;
- are open source (can be modified and extended).

TE_USB_FX2_CyAPI APIs are **not** pure *extern C* code libraries because an external creation of a class instance defined in CyAPI.h (CyAPI.lib) is required.
Advantage:

- possible to directly access CyAPI.lib classes and functions to extend Trenz Electronic TE_USB_FX2_CyAPI.dll libraries with other classes and functions
- support of multi-thread and multi-process programming.

Disadvantage:

- A binding from Python to TE_USB_FX2_CyAPI APIs is possible by resorting to a wrapper (e.g. through SWIG) rather than just the *ctypes* function library. This wrapper is not provided by Trenz Electronic.

A pure *extern C* code library version of TE_USB_FX2_CyAPI APIs (codename: *simplified TE_USB_FX2_CyAPI APIs*) is available for developers.
Advantage:

- binding to Python requires just the *ctypes* function library rather than by resorting to a wrapper.

Disadvantages:

- impossible to directly access CyAPI.lib classes and functions to extend Trenz Electronic TE_USB_FX2_CyAPI.dll libraries with other classes and functions;
- no support of multi-thread programming (but still support of multi-process programming).

7.2.2 TE_USB_FX2_CyUSB API (.NET)

API files:

- TE_USB_FX2_CyUSB
  TE_USB_FX2_CyUSB.dll is open source but CyUSB.dll is not. Nevertheless, its source code can be requested from Cypress under non disclosure agreement.

Documentation:

- UM-TE_USB_API.cs.pdf
- UM-TE_USB_API.cs.PortingGuide.pdf

Cypress *CyUSB* (CyUSB.dll) API for .NET Framework:

- is a dynamic library (**not** embedded in TE_USB_FX2_CyAPI.dll and therefore required by the user along with TE_USB_FX2_CyAPI.dll);
- classes and functions can be directly accessed from the application to extend Trenz Electronic TE_USB_FX2_CyUSB.dll libraries with other classes and functions;
- has only one version for both Microsoft Windows 32 and 64 bit.
- is not a COM (Component Object Model) object, so it cannot be called by C++ or Python code;
- is a managed Microsoft .NET class library that can be used with any programming language of the .NET Framework (Visual Basic, Managed C++, C#, …);

Trenz Electronic TE_USB_FX2_CyUSB.dll for .NET Framework:

- is a dynamic library;
- has only one version for both Microsoft Windows 32 and 64 bit;
- is not a COM (Component Object Model) object, so it cannot be called by C++ or Python code;
- is a managed Microsoft .NET class library that can be used with any programming language of the .NET Framework (Visual Basic, Managed C++, C#, …);
- is open source (can be modified and extended).

TE_USB_FX2_CyUSB.dll has no binding to Python because it is pure .NET CLR code that cannot be compiled as a COM object. It is a managed Microsoft .NET class library that can be used with all languages of the NET Framework (Visual Basic, Managed C++, C#, …) or other .NET infrastructures (such as IronPython).
If the user application is mainly made of C# code, it is convenient to use IronPython.If the user application is mainly made of Python, it is convenient to use Python for .NET package.

7.2.3 Differences Between TE_USB_FX2_CyAPI (C++) APIs and TE_USB_FX2_CyUSB (.NET) API

TE_USB_FX2_CyAPI (C++) APIs and TE_USB_FX2_CyUSB (.NET) API do not have the same signature although they are very similar. The signature of the pure *extern C* code library version of TE_USB_FX2_CyAPI APIs available for developers (codename: *simplified TE_USB_FX2_CyAPI APIs*) is slightly different from the two signatures above.
TE_USB_FX2_CyUSB (.NET) API is intrinsically cleaner than TE_USB_FX2_CyAPI (C++) APIs because CyUSB.dll is intrinsically cleaner than CyAPI.lib.
Please note that CCyUSBDevice Cypress class in the Cypress CyAPI C++ API is actually the list of all USB devices currently served by a Cypress USB driver derivative (e.g. Cypress generic USB device or Trenz Electronic USB FX2 device). For ease of understanding and convenience, Trenz Electronic named the corresponding variable as USBdevList.

API	Cypress class name	TE USB FX2 variable name
C++ API	CCyUSBDevice	USBdevList
.NET API	USBDeviceList	USBdevList

Device list naming comparison.

API	Cypress class name	TE USB FX2 variable name
C++ API	(not available)	(not available)
.NET API	CyUSBDevice	TE_USB_FX2_USBDevice

Single device of the above device lists.
Most functions, parameters and variables have maintained name and meaning across APIs. A notable exceptions is recalled in the following table.

TE_USB_FX2_CyAPI (C++) APIs	TE_USB_FX2_CyUSB (.NET) API
TE_USB_FX2_SetData_InstanceDriverBuffer() TE_USB_FX2_SetData()	(not available) TE_USB_FX2_SetData()
TE_USB_FX2_GetData_InstanceDriverBuffer() TE_USB_FX2_GetData()	(not available) TE_USB_FX2_GetData()

SetData/GetData naming exception,

7.2.4 Possible Future Work

A pure *extern C* code library version of TE_USB_FX2_CyAPI (C++) APIs and TE_USB_FX2_CyUSB (.NET) API would

- ease binding from Python or another programming language, but also
- make impossible to directly access CyAPI.lib classes and functions to extend Trenz Electronic TE_USB_FX2_CyAPI.dll libraries or CyUSB.lib classes and methods to extend Trenz Electronic TE_USB_FX2_CyUSB.dll library.

8 Application Layer

Generation 2 and generation 3 technology stacks are very similar at the application layer. Porting an application from generation 2 to generation 3 technology stack in quite easy and well documented in user friendly porting guides.

8.1 Generation 2

Users can write their application code in C/C++ using the DEWESoft TE USB API. The application code shall use **handles**.
Sample application projects:

- https://github.com/Trenz-Electronic/TE-USB-Suite/tree/master/TE_USB_FX2.gen_2

8.2 Generation 3

Users can write their application code

- in C/C++ using the TE_USB_FX2_CyAPI APIs or
- in a language of the .NET Framework (Visual Basic, Managed C++, C#, ...) using the TE_USB_FX2_CyUSB API.

The application code does **not** use handles.
If the user wants to develop and launch an application or system service (e.g. a Plug and Play application or system service), he/she can call any useful functions or methods from CyAPI (in C++ only), TE_USB_FX2_CyAPI (in C++ only), CyUSB (in any .NET programming language) or TE_USB_FX2_CyUSB (in any .NET programming language).

8.2.1 TE_USB_FX2_CyAPI.dll (C++)

Documentation:

- C++ TE_USB_FX2 API reference manual (UM-TE_USB_API.cpp.pdf)
- DEWESoft C++ DLL to Trenz Electronic C++ DLL Porting Guide (UM-TE_USB_API.cpp.PortingGuide.pdf)

Users can write their own C++ applications by including the FilesToImportForApplicationCpp.zip package. C++ application code can access CyAPI.lib classes and functions directly to extend TE_USB_FX2_CyAPI APIs.
C++ software projects templates and reference applications are available here.
There is no difference between compiling C++ applications for 32 bit Windows operating systems with Microsoft Visual Studio *Express* and Microsoft Visual Studio *Professional.*
There is some difference between compiling C++ applications for 64 bit Windows operating systems with Microsoft Visual Studio *Express* and Microsoft Visual Studio *Professional.* Such differences are explained in the C++ TE_USB_FX2 API reference manual (UM-TE_USB_API.cpp.pdf).

C++ applications using Qt can be easily compiled with Microsoft Visual Studio 2010 Professional and the Qt Visual Studio Add-in on QtProject.org.

8.2.2 TE_USB_FX2_CyUSB.dll (C#)

Documentation:

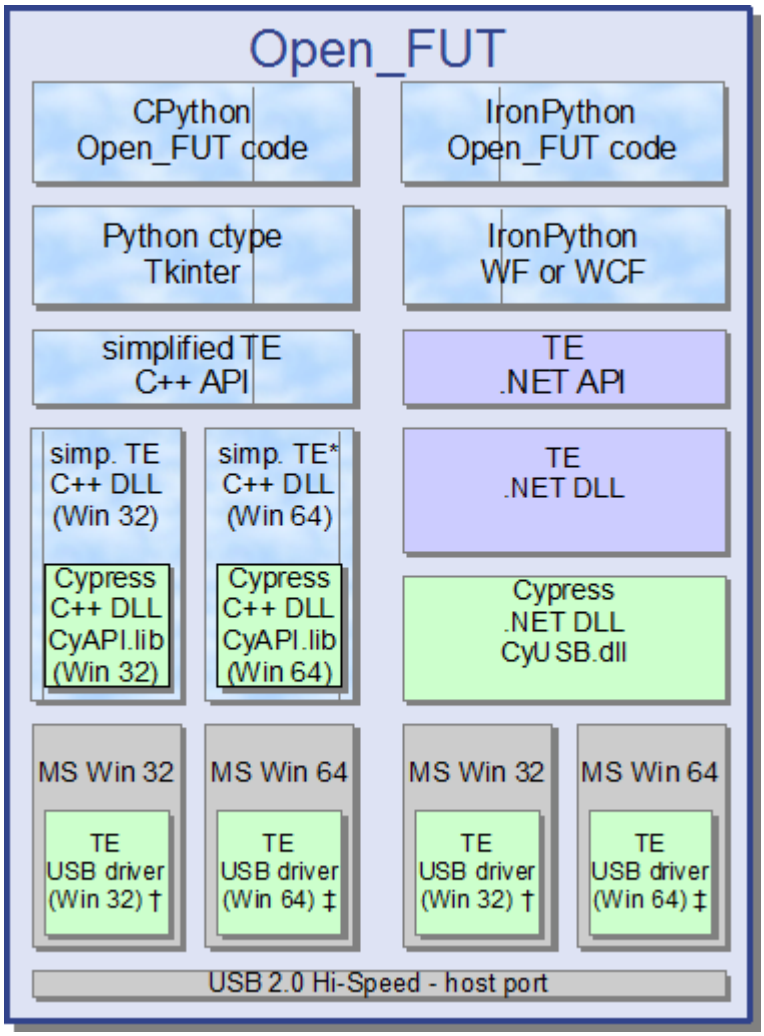
- C# TE_USB_FX2 API reference manual (UM-TE_USB_API.cs.pdf)
- DEWESoft C++ DLL to Trenz Electronic C# DLL Porting Guide (UM-TE_USB_API.cs.PortingGuide.pdf)

Users can write their own .NET applications by including the FilesToImportForApplicationCsharp.zip package.
.NET application code can access CyUSB.dll classes and methods directly to extend TE_USB_FX2_CyUSB API.
.NET software projects templates and reference applications are available here.

8.2.3 Open_FUT (generation 3)

Open_FUT (generation 3) has been developed in CPython by using a pure *extern C* code library version of TE_USB_FX2_CyAPI APIs (codename: *simplified TE_USB_FX2_CyAPI APIs*).
Open_FUT could be ported to .NET by

- using TE_USB_FX2_CyUSB librariesinstead of the *simplified TE_USB_FX2_CyAPI libraries*;
- using .NET Trenz Electronic USB FX2 APIinstead of the *simplified C/C++ Trenz Electronic USB FX2 APIs*;
- using Microsoft WF (Windows Forms) or Microsoft WCF (Windows Communication Foundation)instead of Tkinter (Tkinter is not well supported by .NET);
- using IronPython (or Python for .NET package)instead of Python ctype with CPython.



Open_FUT block diagram - generation 3.

9 Quick Migration Guide

9.1 Documentation

- Migration from the 2nd to the 3rd generation – firmware and device driver (UM-TE-USB-FX2-gen2_to_gen3.pdf)
- video play list: generation 2 to generation 3 migration (firmware + driver)

Generation 2 to generation 3 migration (firmware + driver)  video play list.	Migration from the second generation (aka DEWESoft) to the third generation (aka TE USB FX2) of Trenz Electronic USB FX2 technology stack (firmware and driver).
	TE0300: migration from 2nd to 3rd generation (firmware + driver) This video shows how to upgrade a Trenz Electronic TE0300 device form the second to the third generation (firmware and driver). The video has been recorded on a Microsoft Windows 7 (64 bit) operating system with a TE0300-01 FPGA module connected to a USB port, but the procedure is almost the same for any Trenz Electronic USB FX2 device. In this video, power-on reset is used. Please note that TE0300 has master reset, thus a powered reset (S2 switch) is possible.
	TE0320: migration from 2nd to 3rd generation (firmware + driver) This video shows how to upgrade a Trenz Electronic TE0320 device form the second to the third generation (firmware and driver). The video has been recorded on a Microsoft Windows 7 (64 bit) operating system with a TE0320 FPGA module connected to a USB port, but the procedure is almost the same for any Trenz Electronic USB FX2 device. In this video, power-on reset is used. Please note that TE0320 has master reset, thus a powerered reset (S1D switch to ON) is possible.
	TE0630: migration from 2nd to 3rd generation (firmware + driver) This video shows how to upgrade a Trenz Electronic TE0630 device form the second to the third generation (firmware and driver). The video has been recorded on a Microsoft

	Windows 7 (64 bit) operating system with a TE0630-00 FPGA module connected to a USB port, but the procedure is almost the same for any Trenz Electronic USB FX2 device. Please note that, unlike TE0300 and TE0320, TE0630-00 has no -master reset-, thus a power-on reset is required.
	Cypress USB Console used for 2nd to 3rd generation update. This video explains how to use -Cypress USB Console- instead of "Cypress Control Center". This video assumes that you have followed the first 6 minutes of "generation 2 to generation 3" videos (for anyone of the Trenz Electronic USB FX2 modules). This video is an extension to 6:00 to 6:40 of "generation 2- to -generation 3" videos.

9.2 Description

When migrating from the second generation of the software stack for Trenz Electronic FPGA modules with USB interface (aka DEWESoft firmware and DEWESoft driver derivative) to the third generation (aka Trenz Electronic firmware and Cypress driver derivative), we recommend the following:

- A) flash latest firmware into the module:

https://github.com/Trenz-Electronic/TE-USB-Suite/tree/master/TE_USB_FX2.firmware/ready_for_download/
(currently: TE-USB-FX2_v03.02_TE.iic)

- B) fully uninstall (i.e. with "delete files" option activated on Windows Vista or later versions, in Windows XP the deletion must be done manually: see section 4) old driver and install new driver:

http://www.trenz-electronic.de/download/d0/Trenz_Electronic/d1/TE-USB-Suite/d2/generation_3/d3/drivers.html
and, of course, read carefully the manual (UM-drivers-TE_USB_FX2.pdf)
Driver installation on Microsoft Window 7 and Microsoft Window 8 can be performed automatically (on-line). It is possible that the first online driver installation fails; in this case the developer shall force a retry (the second attempt normally works). See UM-Drivers-TE_USB_FX2 manual at section *2.1.2.3.2 Force the online search*.
Please compare your *Device Manager* window with the pictures shown in the manual cited above. In particular, check that the picture shown in section *2.1.3 Common to Windows XP/7/8 (Final Part)* matches both device class name and driver class name in your *Device Manager*. If not, this is probably because the operating system links/loads the old driver class and driver file(s). In this case, it might be useful to go to C:\WINDOWS\system32\drivers (or the like) and delete TE03xx* files. The intended driver file names are "TE_USB_FX2_xx" instead:

- TE_USB_FX2.cat
- TE_USB_FX2.inf
- TE_USB_FX2_32.sys

- TE_USB_FX2_64.sys

Warning: Generation 3 device drivers with version date prior to 06 September 2012 must *not* be used.

- C) download latest C++/C# API: https://github.com/Trenz-Electronic/TE-USB-Suite/

On the GitHub repository, the developer can find the current C++/C# API. The code is associated with Microsoft Visual Studio Express 2010 project. This project can also be opened in Visual Studio Professional 2010.
The C++ DLL and sample code is preset for Microsoft Windows 32 bit operating systems. If you desire to compile the solution for 64 bit operating systems, the procedure in Microsoft Visual Studio Express and Microsoft Visual Studio Professional has some differences and is specified in *Appendix A : Open the Visual Studio 2010 project* of C++ TE_USB_FX2 API - reference manual published here: http://www.trenz-electronic.de/download/d0/Trenz_Electronic/d1/TE-USB-Suite/d2/generation_3/d3/APIs.html
The differences are the following

- Express requires the installation of *Microsoft Windows SDK 7.1* and it must be selected under Platform Toolset as *Windows7.1SDK*;
- Professional uses the same v100 (10.0) runtime components as in Platform Toolset. The developer does not need to install *Microsoft Windows SDK 7.1*.

It is possible to download some *Microsoft Visual Studio 2010 Express* projects already correctly configured (for *Microsoft Visual Studio 2010 Express* 32/64 bit and *Microsoft Visual Studio 2010 Professional* 64 bit) from here:

- http://www.trenz-electronic.de/download/d0/Trenz_Electronic/d1/TE-USB-Suite/d2/generation_3/d3/reference_designs.html

10 References

- Trenz Electronic USB Suite resource center
  http://www.trenz-electronic.de/download/d0/Trenz_Electronic/d1/TE-USB-Suite.html
- Trenz Electronic GitHub repositories
  https://github.com/Trenz-Electronic/
- Trenz Electronic USB Suite GitHub repository
  https://github.com/Trenz-Electronic/TE-USB-Suite
- Cypress SuiteUSB 3.4 - USB Development tools
  http://www.cypress.com/?rID=34870
- Cypress CyAPI Programmer's Reference
  http://www.cypress.com/?docID=41365
- Cypress CyUSB .NET DLL Programmer's Reference
  http://www.cypress.com/?docID=41366
- Cypress USBSuite Application Development
  http://www.cypress.com/index.cfm?docID=39566
- IronPython (the Python programming language for the .NET Framework)
  http://ironpython.net/
- Python for .NET
  http://pythonnet.sourceforge.net/
- SWIG (Simplified Wrapper and Interface Generator)
  http://www.swig.org/
- Qt Visual Studio Add-in
  http://qt-project.org/search/tag/qt~visual~studio~add-in