

SW 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 is quite easy and well documented in user friendly porting guides.

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

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).

TE_USB_FX2_CyAPI.dll (C++)

Documentation:

- C++ TE_USB_FX2 API reference manual ([UM-TE_USB_API.cpp Reference Manual](#))
- DEWESoft C++ DLL to Trenz Electronic C++ DLL Porting Guide ([UM-TE_USB_API.cpp Porting Guide](#))

Application code examples:

- C++ software projects templates and reference applications are available [here](#).

Compiling C++ applications:

- 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++ applications using Qt can be easily compiled with Microsoft Visual Studio 2010 Professional and the [Qt Visual Studio Add-in](#) on QtProject.org.
- 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 ([here](#)). C++ applications for 64 bit Windows operating systems with Microsoft Visual Studio *Express* requires also *Microsoft Windows SDK 7.1*



The straightforward procedure to install both *Microsoft Windows SDK 7.1* and *Microsoft Visual Studio 2010 Express* on the same computer will fail: see [here](#). In this link, 2 different procedures are described; we have successfully tested both procedures.

TE_USB_FX2_CyUSB.dll (C#)

Documentation:

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

Application code examples:

- .NET software projects templates and reference applications are available [here](#).

Compiling C# applications:

- 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.

Simplified TE_USB_FX2_CyAPI.dll (C++)

Documentation:

- Dewesoft C++ DLL to Simplified Trenz Electronic C++ DLL ([UM-SimplifiedTE_USB_API.cpp Porting Guide](#))

Application code examples:

- Open_FUT (generation 3)

Compiling C++ applications:

- Pure *extern C* code library simplified version of TE_USB_FX2_CyAPI (C++) APIs.
- Ease binding to/from Python ([ctype](#)) or another programming language
- Impossible to directly access CyAPI.lib classes and functions to extend Trenz Electronic TE_USB_FX2_CyAPI.dll libraries
- It is not thread safe

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 library instead of the *simplified TE_USB_FX2_CyAPI library*;
- using .NET Trenz Electronic USB FX2 API instead 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

CPython
Open_FUT code

IronPython
Open_FUT code

Python ctype
Tkinter

IronPython
WF or WCF

simplified TE
C++ API

TE
.NET API

simp. TE
C++ DLL
(Win 32)

simp. TE*
C++ DLL
(Win 64)

TE
.NET DLL

Cypress
C++ DLL
CyAPI.lib
(Win 32)

Cypress
C++ DLL
CyAPI.lib
(Win 64)

Cypress
.NET DLL
CyUSB.dll

MS Win 32

MS Win 64

MS Win 32

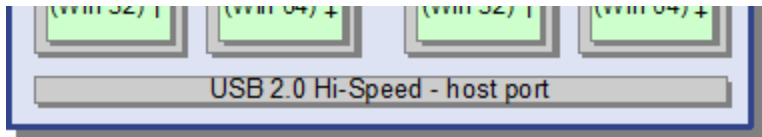
MS Win 64

TE
USB driver
(Win 32) +

TE
USB driver
(Win 64) +

TE
USB driver
(Win 32) +

TE
USB driver
(Win 64) +



Open_FUT block diagram - generation 3.