

# SW API Layer

Software (SW) APIs require a device driver of the same generation.

Beyond the SW APIs (used for the communication between the host computer and the USB FX2 microcontroller), another set of firmware (FW) API exists: they are called "API Commands". The SW program running on host computer sends API commands to the USB FX2 microcontroller and/or Xilinx MicroBlaze embedded processor by means of the following SW API functions:

- generation 2: `TE0300_SendCommand(..., command, ...)`;
- generation 3: `TE_USB_FX2_SendCommand(..., command, ...)`.

The USB FX2 microcontroller and/or Xilinx MicroBlaze soft embedded processor is(/are) able to execute the command. Skilled users can extend the FW API Command modifying USB FX2 microcontroller firmware (and MicroBlaze SW if necessary, but in many case is not necessary).



Even in the case the API command is executed by Xilinx MicroBlaze, it is USB FX2 microcontroller to receive and dispatch the command to Xilinx MicroBlaze. For more information see [here](#).

## 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.

## 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.

## 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 Reference Manual](#)
- [UM-TE\\_USB\\_API.cpp Porting Guide](#)

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 (for example Plug and play functions and asynchronous multithread bulk data transfers).
- 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).

## 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 Reference Manual](#)
- [UM-TE\\_USB\\_API.cs Porting Guide](#)

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).

Advantage:

- Possible to directly access CyUSB.dll classes and functions to extend Trenz Electronic TE\_USB\_FX2\_CyUSB.dll libraries with other classes and functions (for example Plug and play functions and asynchronous multithread bulk data transfers)
- Support of multi-thread and multi-process programming.
- 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).

Disadvantage:

- 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. 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.

## 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() | (not available)             |
| TE_USB_FX2_SetData()                      | TE_USB_FX2_SetData()        |

|                                           |                      |
|-------------------------------------------|----------------------|
| TE_USB_FX2_GetData_InstanceDriverBuffer() | (not available)      |
| TE_USB_FX2_GetData()                      | TE_USB_FX2_GetData() |

**SetData/GetData naming exception,**

## 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.

At this moment a simplified [TE\\_USB\\_FX2\\_CyAPI \(C++\) API](#) is used with [Open\\_FUT \(gen3\)](#) tool but it is not thread safe.

## Libusb and libusbx libraries



### libusb and libusbx merged

[libusb](#) and [libusbx](#) release will be officially merged from [v1.0.19](#). These two project/libraries are already actually ("unofficially") merged.

For Windows, see section "WinUSB driver and Zadig (Generation2, Generation 3, Cypress default and/or custom firmware)" of [Device Driver Layer](#).

For Linux, see [Linux\\_FUT](#).

For Mac OS X, see "[AN74505](#) - EZ-USB® FX2LP™ - Developing USB Application on MAC OS X using LIBUSB".



These libraries are not yet officially supported under Windows, but they are already used with TE USB FX2 modules under Linux (see the community contribution [Linux\\_FUT](#)).