Recovery boot: procedures to use if USB connection is unresponsive

Terminology and tips

Terminology: USB FX2 unresponsive aka USB FX2 microcontroller stall

USB FX2 microcontroller stall term/definition is used to describe the situation where your TE USB FX2 module's FPGA works normally (you are still able to connect to FPGA using JTAG connection) but you are unable to connect to it via USB connection despite having the correct USB drivers installed.

TE USB FX2 modules are industrial products and if wrong firmware is written in EEPROM (=> USB unresponsive but FPGA's working) it can easily corrected (without using JTAG connection) with EEPROM switch and USB connection: the solution is quite easy and fast (5-10 seconds) under Windows (implicit two-step recovery boot) and it is also possible under Linux (explicit two-step recovery boot).

If the user is unsure about new firmware, he/she could run the firmware in RAM

If the user is unsure about the suitability of a new firmware (wherever it is the source), it's a good idea to load it into RAM first to make sure it is not totally broken.

A reference firmware (from Trenz Electronic GitHub) is suitalble and tested; it can be used to test the functionality of TE USB FX2 module and SW APIs. The only possible problem with reference firmware files is the corruption of the downloaded .iic file. If the user have problem with reference .iic file should consider to check the integrity of the file and/or download a new copy of desired .iic file from Trenz Electronic GitHub.

Recovery boot: bring back a TE USB FX2 module's USB connection from stall

Writing wrong (or corrupted) firmware to EEPROM will bring the USB FX2 microcontroller to stall. To bring it back out of a stall, the firmware recovery procedure differs upon the host operating system:

- use an Implicit Two-Step Recovery Boot (Windows);
- use an Explicit Two-Step Recovery Boot (Linux or Windows).

Recovery Boot

If USB connection is unresponsive, a wrong (or corrupted) firmware is probably loaded in FX2 microcontroller's EEPROM.

> To default, the FX2 hardware (i.e with EEPROM isolated =>hardware's first stage loader) enumerates the USB FX2 microcontroller chip as VID=0x04B4 (Cypress) and PID=0x8613 (FX2LP), and provides support for loading firmware into RAM.
>
> To default, using the FX2 hardware (i.e with EEPROM isolated => hardware's first stage loader) it is possible to write a new firmware in RAM but not in EEPROM.

The USB FX2 microcontroller hardware's first stage loader (supporting the 0xA0 vendor request) can't write into external memory (EEPROM for example). Configurations that put firmware into external memory thus need a second stage loader. For typical "flat" memory architectures, a loader supporting the 0xA3 vendor request is used to write into that memory. Similarly, a second stage loader that supports the 0xA2 or 0xA9 vendor request is needed when writing boot firmware into an I2C EEPROM. These 0xA2, 0xA9 and 0xA3 vendor commands are conventions defined by Cypress.

Implicit Two-Step Recovery boot Explicit Two-Step Recovery Boot

It is possible to write in EEPROM only using a 2nd stage loader firmware supporting EEPROM programming (aka intermediate good firmware) running in RAM; normally the bootloader Vend_Ax.hex is used.

- In the case of Implicit Two-Step Recovery Boot (at this time, only Windows) the 2nd stage (boot)loader firmware is loaded without the explicit intervention of the user.
- In the case of Explicit Two-Step Recovery Boot (mainly Linux) the 2nd stage (boot)loader firmware is loaded with the explicit intervention of the user.

Both recovery boots use exactly the same procedure (under the hood, they are one and the same), but in Windows OS case the 2nd stage (boot)loader firmware step is **hidden/implicit** in loadEEPROM() function used by CyConsole, CyControl and OpenFutNet.