FTDI Programmer User Manual
β Version

Kunal Chakraborty
kunalc@cdot.in

January 22, 2019

Contents

0.1 Introduction

FTDI-programmer is an application written in pure python to access/program on board devices through FTDI devices using JTAG/SPI/I2C/GPIO interfaces. JTAG programming is done through svf file. It can be used in place of any external emulator. The application is built on PyFtdi driver.

The source code is compatible with both Windows and Linux system. However this manual is written from a windows user's point of view.

This is a β-version and hence the application has couple of limitations as listed in section 0.3

0.2 Features

- On board device programming through JTAG interface. It needs svf file for JTAG programming.
- TI UCD device programming through I2C interface. It needs SMBUS csv file programming. This file can be generated by Fusion tool.
- SPI Flash programming.(to be implemented)
- On board device access thhrough SPI/I2C/GPIO interfaces. (to be implemented)

0.3 Limitation

1. I2C mode
   (a) FTDI devices does not support multimaster and clock stretching. Hence this application will not work with slave devices which requires clock stretching. In that case application can be run on lower frequency so that target device may get enough time to respond. However there is a workaround given by FTDI by connecting I2C SCL to a separate gpio of FTDI device so that FTDI device can read the clock line. This feature is not supported in present version FTDI-programmer.
   (b) Highest I2C slave address supported is 0x78. It is the limitation from pyftdi driver. No workaround is provided.
2. JTAG mode
   (a) Target device must be the only device in JTAG chain because SVF parser of FTDI-programmer does not support header and trailer addition. This is kept as Future Development.

(b) SVF file verification takes longer than expected. This is because pyftdi driver takes long time to send data to application after reading from usb. This is kept as Future Development.

(c) Only Max-V CPLD programming is supported. If any other device is detected, FTDI-programmer will flag a warning with device id and exit.

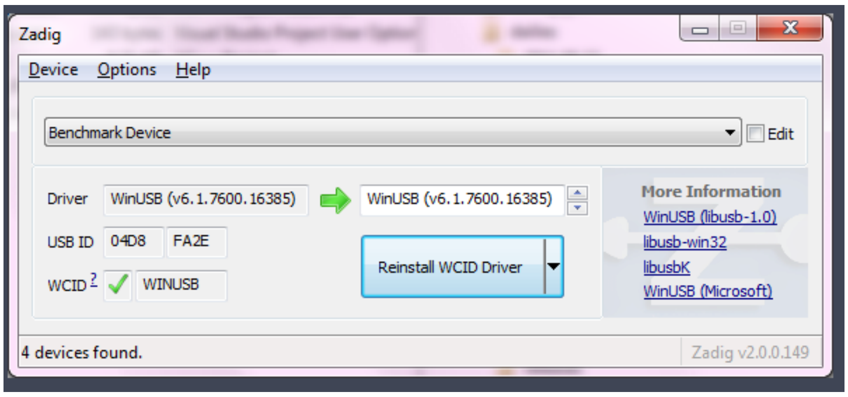
3. Others

   (a) MS Windows detects every channel of FTDI device as a separate usb device with same VID and PID. Hence FTDI-programmer can access only single channel of FTDI device. So user must enable a single lib-usb driver as per FTDI channel number as discussed in section 0.4

0.4 Installation

All the required python packages are compiled and supplied with FTDI-programmer. However it needs a low level backend driver libusb to talk to FTDI device. You must have admin privilege to carry out below steps. Step by step installation guide is given below.

1. **libusb installation:** An easy way to install libusb backend on windows is Zadig. Download the latest version of Zadig from `https://zadig.akeo.ie/` Run zadig.exe and you should get a dialog as shown in Fig1.



Benchmark Device

Driver WinUSB (v6.1.7600.16385) → WinUSB (v6.1.7600.16385)

USB ID 04D8 FA2E

WCID ? ✓ WINUSB

4 devices found.

Zadig v2.0.0.149

Figure 1: zadig main window

Go to Options – List All Devices. Then select FTDI-channel0 from drop down option. Next select libusb-win32 as driver.

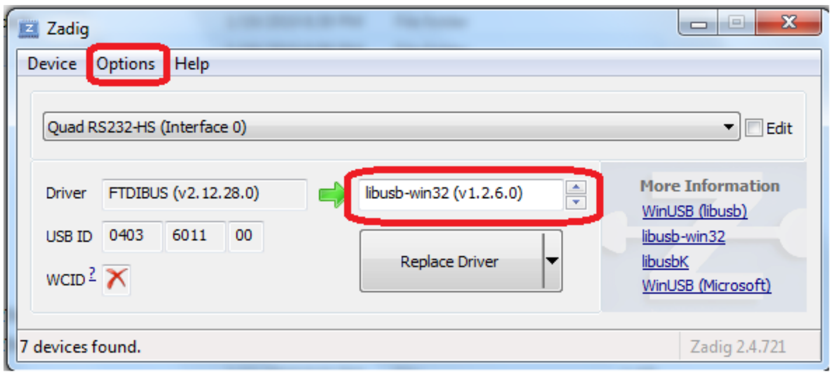


Figure 2: libusb installation

Finally, click install driver. Follow the same step to install driver on FTDI-channel1

2. **Verify libusb driver:** Go to windows device manager. You should be able to see libusb device. Under libusb, FTDI devices (1 for each channel) should be shown for which libusb driver is installed(refer to figure 3. Disable all channels except one channel(the one you want to use for programming).

3. **Get FTDI-programmer:** Download a fresh copy of FTDI-programmer from `https://github.com/kunalcdot/FTDI-programmer`. The application is under /bin folder. Run FTDI-programmer.exe to start the application.

0.5 Programming Guide

0.5.1 Programming file

- For JTAG programming standard svf file needs to be used.
- For UCD programming, SMBUS flash script in csv format needs to be generated. The script generation procedure is discussed in detail in section 0.5.2

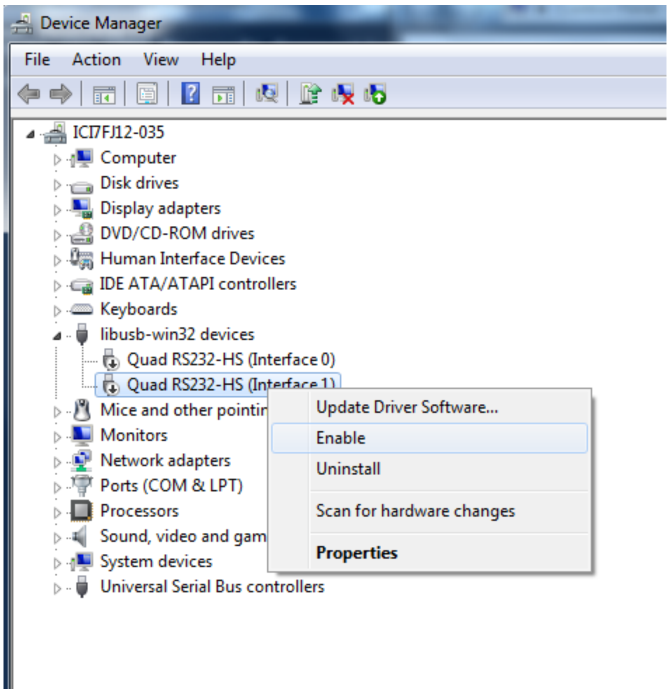


Figure 3: Enabling FTDI channel from device manager

0.5.2 SMBUS script generation

UCD devices run on SMBUS protocol. FTDI-programmer sends SMBUS commands towards UCD device for programming. TI Fusion tool can be used to generate SMBUS flash script. Note that flash script can only be generated in online mode. Also note that the UCD device used to generate flash script should have the same firmware as target device. This will also be verified by FTDI-programmer.

Open Fusion GUI tool in online mode. Then go to File – Export option. Under export option select **'Data Flash Script'**. Select below mentioned options in the script settings:

- Script Style: SMBUS
- Add PEC byte
- File format: CSV
- Hex format: 0xAABB

- Comment Style: 'Comment'
- Multiple byte: Compact
- Read verification: As per your choice

A screenshot of script settings is shown in figure 4. Export the file with .csv extension.

0.5.3 FTDI Channel selection

Due to limitation from MS Windows, PC running FTDI-programmer must have only one channel active. Libusb driver can be installed on multiple channels. However ensure that all the channels except the desired channel are disabled. This can be done in windows device manager utility. Refer to Figure 3. The user must have administrator privilege.

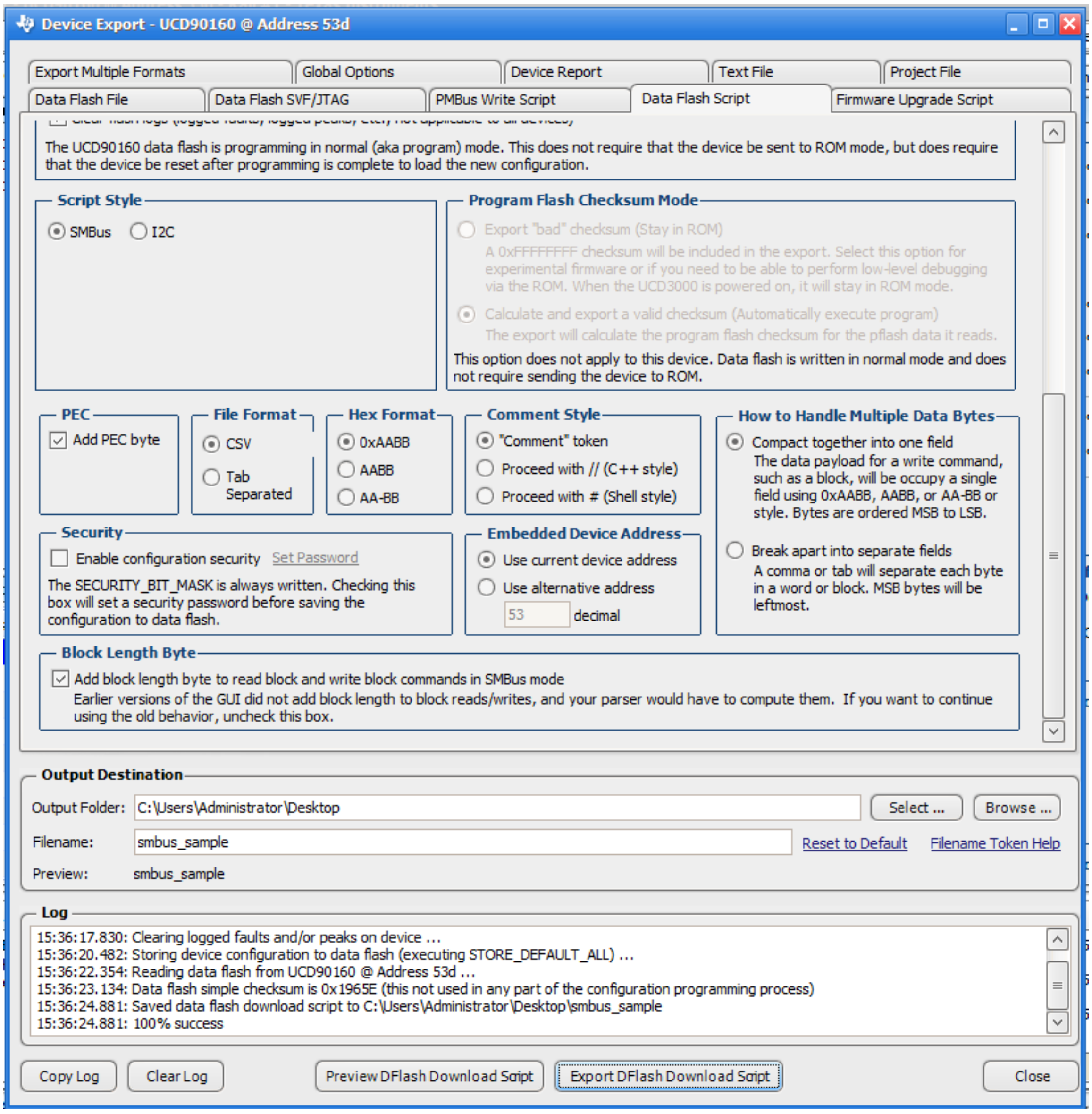
0.5.4 Run FTDI-programmer

Download a fresh copy of FTDI-programmer from `https://github.com/kunalcdot/FTDI-programmer`. The application is under /bin folder. Run FTDI-programmer.exe to start the application. Ensure all supporting files(dll and windows api) are present in same directory.

There are 2 programming modes: JTAG(standard svf programming) and I2C(customized UCD device programming). Each mode supports default as well as advance settings. Most of the time default setting is the desired one. In advance mode, you can change programming frequency and frequency optimization.

For JTAG default frequency is 3MHz. FTDI-programmer ignores svf file frequency command. In case of I2C default frequency is 5KHz. As FTDI device does not support clock stretching, default frequency is kept low to communicate with UCD device.

Next browse the file to be progrmmed and programming will be started.



The UCD90160 data flash is programming in normal (aka program) mode. This does not require that the device be sent to ROM mode, but does require that the device be reset after programming is complete to load the new configuration.

Script Style

[✓ SMBus] I2C

Program Flash Checksum Mode

Export "bad" checksum (Stay in ROM)
A 0xFFFFFFFF checksum will be included in the export. Select this option for experimental firmware or if you need to be able to perform low-level debugging via the ROM. When the UCD3000 is powered on, it will stay in ROM mode.

[✓ Calculate and export a valid checksum (Automatically execute program)]
The export will calculate the program flash checksum for the pflash data it reads.

This option does not apply to this device. Data flash is written in normal mode and does not require sending the device to ROM.

PEC

[✓ Add PEC byte]

File Format

[✓ CSV]
Tab Separated

Hex Format

[✓ 0xAABB]
AABB
AA-BB

Comment Style

[✓ "Comment" token]
Proceed with // (C++ style)
Proceed with # (Shell style)

How to Handle Multiple Data Bytes

[✓ Compact together into one field]
The data payload for a write command, such as a block, will be occupy a single field using 0xAABB, AABB, or AA-BB or style. Bytes are ordered MSB to LSB.

Break apart into separate fields
A comma or tab will separate each byte in a word or block. MSB bytes will be leftmost.

Security

Enable configuration security Set Password

The SECURITY_BIT_MASK is always written. Checking this box will set a security password before saving the configuration to data flash.

Embedded Device Address

[✓ Use current device address]
Use alternative address
53 decimal

Block Length Byte

[✓ Add block length byte to read block and write block commands in SMBus mode]
Earlier versions of the GUI did not add block length to block reads/writes, and your parser would have to compute them. If you want to continue using the old behavior, uncheck this box.

Output Destination

Output Folder:	C:\Users\Administrator\Desktop	Select ... Browse ...
Filename:	smbus_sample	Reset to Default Filename Token Help
Preview:	smbus_sample	

Log

```
15:36:17.830: Clearing logged faults and/or peaks on device ...
15:36:20.482: Storing device configuration to data flash (executing STORE_DEFAULT_ALL) ...
15:36:22.354: Reading data flash from UCD90160 @ Address 53d ...
15:36:23.134: Data flash simple checksum is 0x1965E (this not used in any part of the configuration programming process)
15:36:24.881: Saved data flash download script to C:\Users\Administrator\Desktop\smbus_sample
15:36:24.881: 100% success
```

Figure 4: SMBUS Flash Script for UCD device

0.6 Future Development

1. All programming file parser to be updated with 'regular expression' module. It should support all svf standard command.
2. JTAG read time needs to be reduced.
3. SPI Flash Programming option to be developed
4. An utility needs to be provided to read/write on board devices through I2C/SPI/GPIO etc

0.7 Troubleshooting

Ensure only one FTDI device is connected and single ftdi channel is enabled. Refer to section 0.4 for more details.

Error: No ftdi device – Ensure pc is connected to FTDI device and the device is out of reset

Error: Unknown device detected(In jtag mode): Ensure board is powered up and there is only one device(MAX-V) in JTAG chain.

Error: File not found: Ensure there is no white space in your file path.