CyUSB Suite for Linux

Version 1.0.5

User Guide

Document Version History

Version	Date	Author	Notes
1.0	07/25/12	V.Radhakrishnan	Original version submitted to Cypress for review.
1.0.1	08/13/12	Cypress Semiconductor	Updated for general software release.
1.0.2	10/01/12	Cypress Semiconductor	Changes to fix false download failure to the FX3 device.
1.0.3	01/23/13	Cypress Semiconductor	Updated the library to allow more device ID pairs to be supported.
1.0.4	07/11/13	Cypress Semiconductor	Changes to the cyusb_linux utility to allow automated firmware download to EEPROM or SPI Flash. Added download_fx3 and download_fx2 example utilities to faciltate firmware download for FX3 and FX2LP devices.
1.0.5	05/08/15	Cypress Semiconductor	Added streamer functionality to the cyusb_linux utility. Added a new console utility to measure data transfer performance.

Chapter - 1 : Introduction

The CyUSB Suite for Linux provides a set of tools for users to program and communicate with Cypress EZ-USB devices over the USB port. The software functionality is analogous to the Cypress USBSuite software for Windows platforms, and is built around the open source libusb-1.0 library.

This document describes how to install, configure and use the CyUSB Suite for Linux software. This software allows you to download ("Program") firmware images onto the EZ-USB FX2, FX2LP and EZ-USB FX3 based devices, and test the various interfaces / alternate interfaces on the device.

This software does not include any firmware source code for the EZ-USB parts or tools for generating the firmware binaries. A set of pre-compiled firmware binaries that are extracted from the Software Development Kits (SDK) for FX2 and FX3 are included. The actual firmware itself is developed using other tools offered by Cypress, notably the SDKs.

There are two companion documents available for Linux :

1. This guide, also known as the CyUSB Suite for Linux User Guide, and
2. The CyUSB Suite for Linux - Programmers Reference Manual, which describes the driver for linux and how to build and integrate user-written applications with the driver.

The software itself has been developed on top of the opensource libusb driver available at

http://sourceforge.net/projects/libusb/files/libusb-1.0/

under which directory various versions are available. At the moment of this writing, the version used is libusb-1.0.16.tar.bz2.

This version of the CyUSB Suite for Linux has been developed and tested on the following environment :

1. Linux Kernel - 3.13.0 (stock kernel from http://www.kernel.org)
2. Fedora - 14 distribution, Ubuntu 11.04, Ubuntu 12.04, Ubuntu 14.04
3. Qt version 4.7
4. libusb-1.0.16

The software tools allow you to do the following :

1. View the device, configuration, interface, alternate-interface and endpoint descriptors of attached devices.
2. Select a specific interface+alt interface to communicate with.
3. Program the device (download firmware) as follows :
   a) For the FX2 based devices -> download into RAM (Internal/External), small EEPROM or a large EEPROM
   b) For the FX3 based devices -> download into RAM, I2C based EEPROM or SPI based flash
4. Test your own commands (Vendor Extensions) after downloading a specific firmware that implements your command(s).
5. Perform Bulk/Isochronous data transfers from/to OUT and IN endpoints on the USB devices.

Chapter - 2 : Installing the software

The software CyUSB Suite for Linux is available as a compressed tarball as follows :

cyusb_linux_1.0.5.tar.gz; where the 1.0.5 refers to the version number of the software

Copy the tar file to any of the local folder and extract it as shown below

[user@desktop /home/Cypress]$ tar zxvf cyusb_linux_1.0.5.tar

Once the archive is extracted, the following directory structure is created:

The ***bin*** directory contains the main executable program called cyusb_linux.

The ***configs*** directory contains 3 configuration files as under :

a) **cyusb.conf** -> Contains VID and PID of list of Cypress devices, which are of interest.User can add edit the file and add new VID - PID pairs to the list.

b) **88-cyusb.rules**-> Udev rules written for Cypress devices.

c) **cy_renumerate.sh**-> Udev script that runs whenever a Cypress device is connected to or removed from the Host machine. This script is the means for CyUSB based software tools to handle hotplug of devices of interest.

The ***docs*** subdirectory contains the documentation in PDF format (requires Acrobat Reader or equivalent).

The ***fx2_images*** subdirectory contain some standard firmware images for the FX2LP devkit

The ***fx3_images*** subdirectory contains some standard frmware images for the FX3 devkit

The ***gui_src*** subdirectory contains the source code for the CyUSB library and the CyUSB_Linux GUI.

The ***include*** subdirectory contains the header files controlcenter.h and cyusb.h

The ***lib*** subdirectory contains the library file libcyusb.so. This would need to be accessible for the main program to work.

The ***src*** folder contains a set of example programs that make use of the cyusb APIs to perform various types of USB data transfers.

The ***test_cases*** directory contains a simple program to generate test case data.

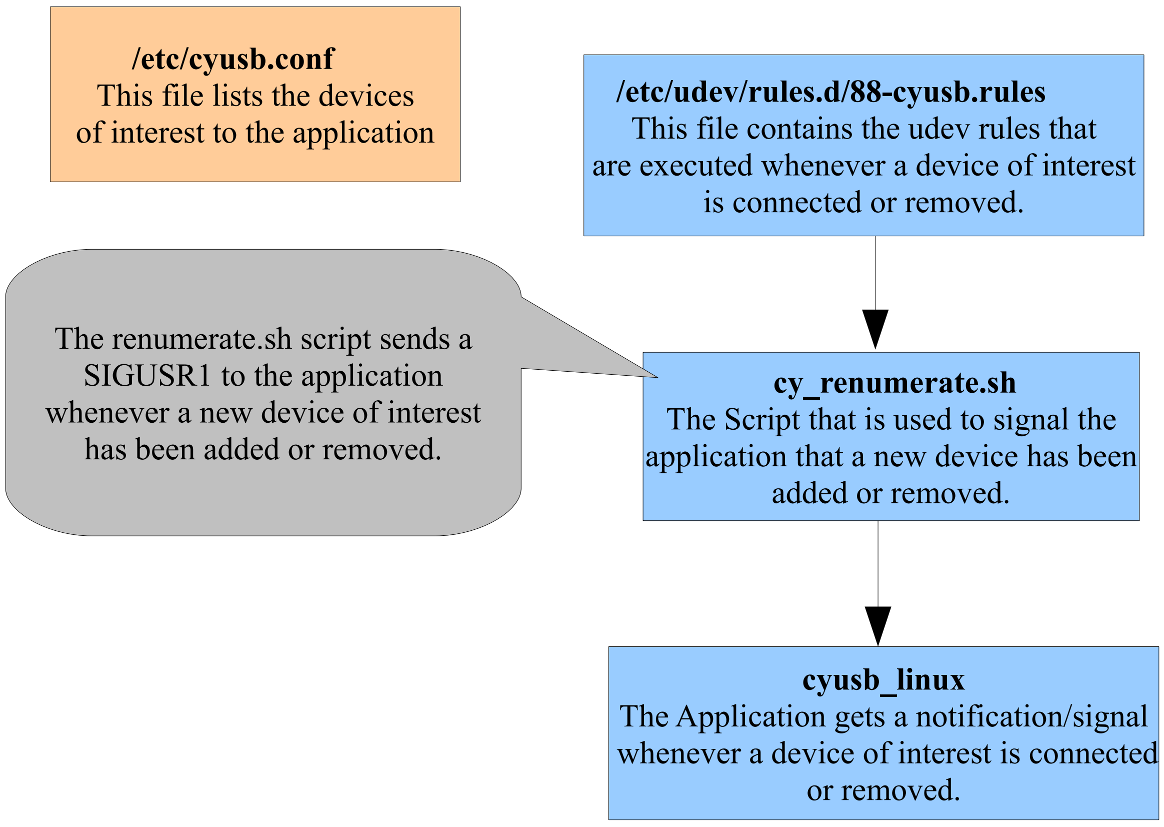
Firmware Download and Initialization

The Cypress EZ-USB FX2 and FX3 devices work on the principle of Re-Enumeration. i.e after a new firmware image is downloaded, the device would Re-Enumerate as a new device. We would need to detach the earlier device and attach to this new device, which is done automatically through the interaction of a custom udev script called **88-cyusb_rules,** and a custom shell script called **renumerate.sh.**

The **88-cyusb_rules** specify that the renumerate.sh script needs to be invoked by the udev layer whenever a device matching the rules in this file, is plugged in or disconnected from the host PC.

The file **cy_renumerate.sh** is a simple shell script that executes whenever a USB device of interest is added or removed from a host computer. This script is responsible for notifying the cyusb_linux application that a Cypress USB device has been connected or disconnected from the PC.

The Figure below gives the graphical representation of entire cyusb_linux:


/etc/cyusb.conf
This file lists the devices of interest to the application
/etc/udev/rules.d/88-cyusb.rules
This file contains the udev rules that are executed whenever a device of interest is connected or removed.
The renumerate.sh script sends a SIGUSR1 to the application whenever a new device of interest has been added or removed.
cy_renumerate.sh
The Script that is used to signal the application that a new device has been added or removed.
cyusb_linux
The Application gets a notification/signal whenever a device of interest is connected or removed.

- **Building and installing the software**

  Refer to the README file for pre-requisites required for the installation and functioning of this software. Ensure that all of the pre-requisite software modules are installed before starting to compile and install the CyUSB Suite.

  Once the files are extracted in the local directory, Open the terminal and point to the directory where files were extracted. Type the following commands to build and install the executable.

  For example: If the files were in /home/Cypress:
  [user@desktop /home/Cypress]$ make
  [user@desktop /home/Cypress]# ./install.sh (Requires super-user privileges to copy the configuration files into system folders like /etc and /usr/bin)

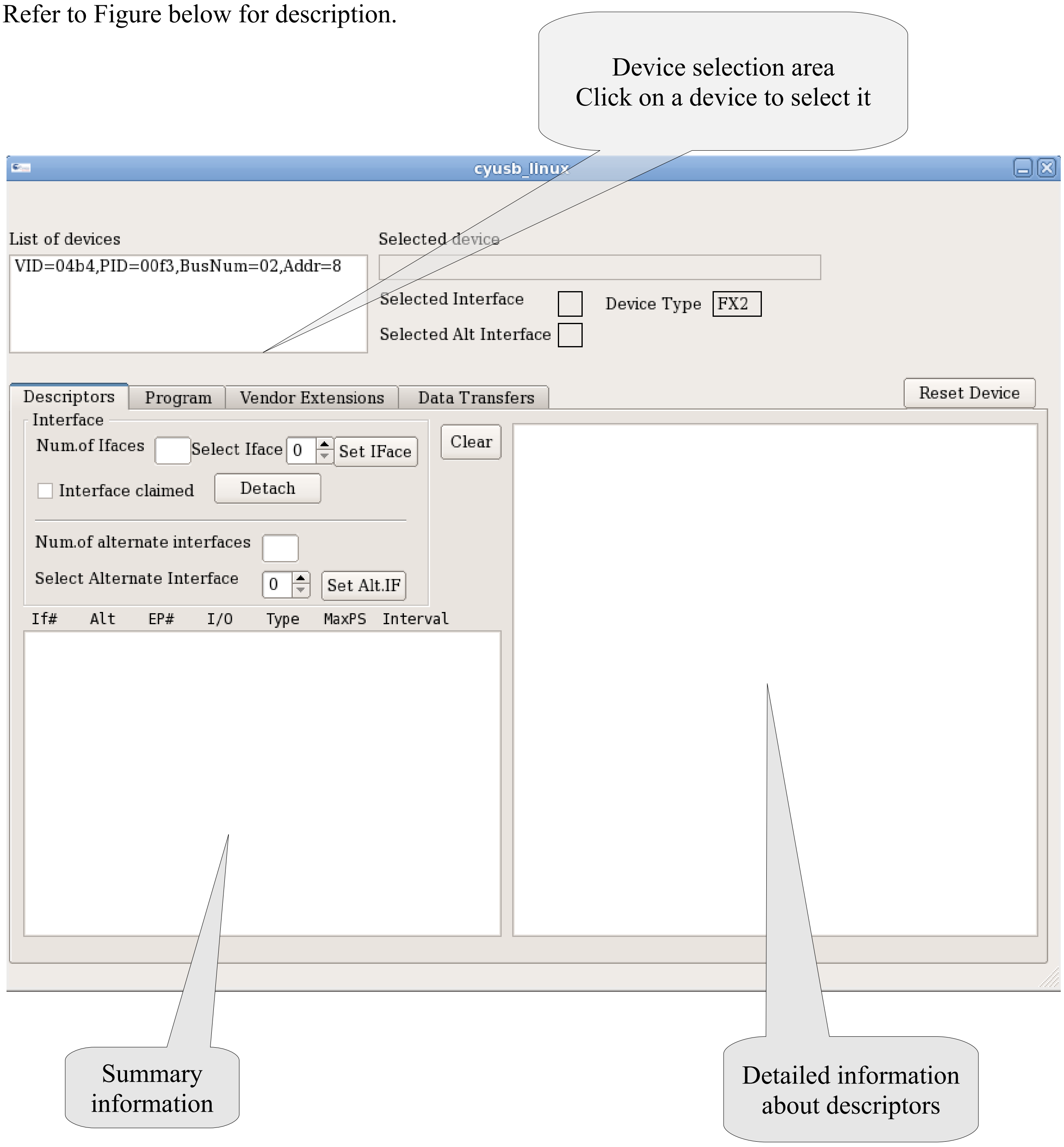
  This will install the cyusb_linux application, which can then be launched from the terminal. This application can be executed in normal user mode without super-user (root) privileges. This script also sets an environment variable called CYUSB_ROOT to point to the path where the files are located.

Chapter - 3 : Using the application

Open a terminal and launch the application by typing “cyusb_linux”.

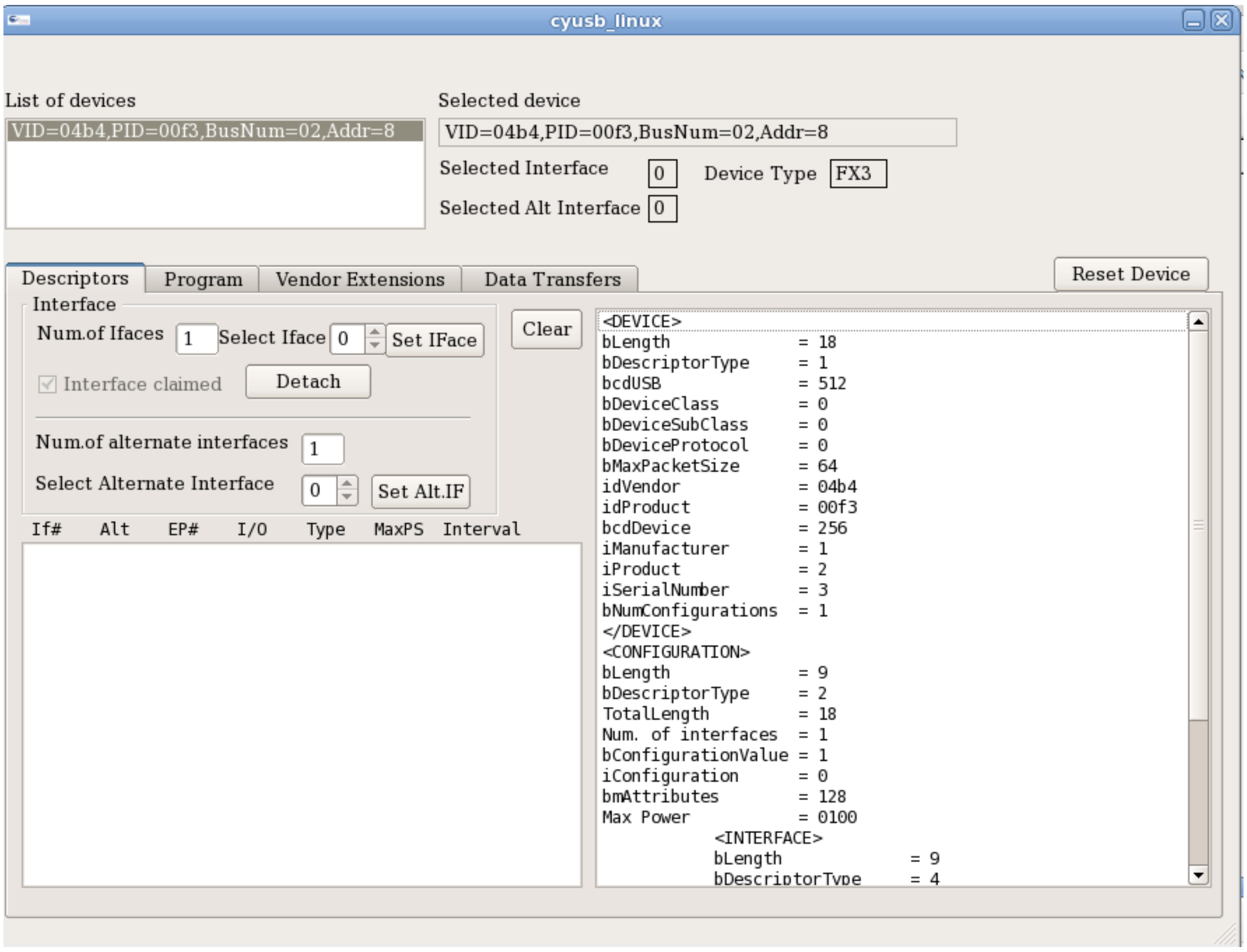
E.g. :[user@desktop /home]$ cyusb_linux.

**Make sure to run the application as normal user.
Refer to Figure below for description.



VID=04b4,PID=00f3,BusNum=02,Addr=8

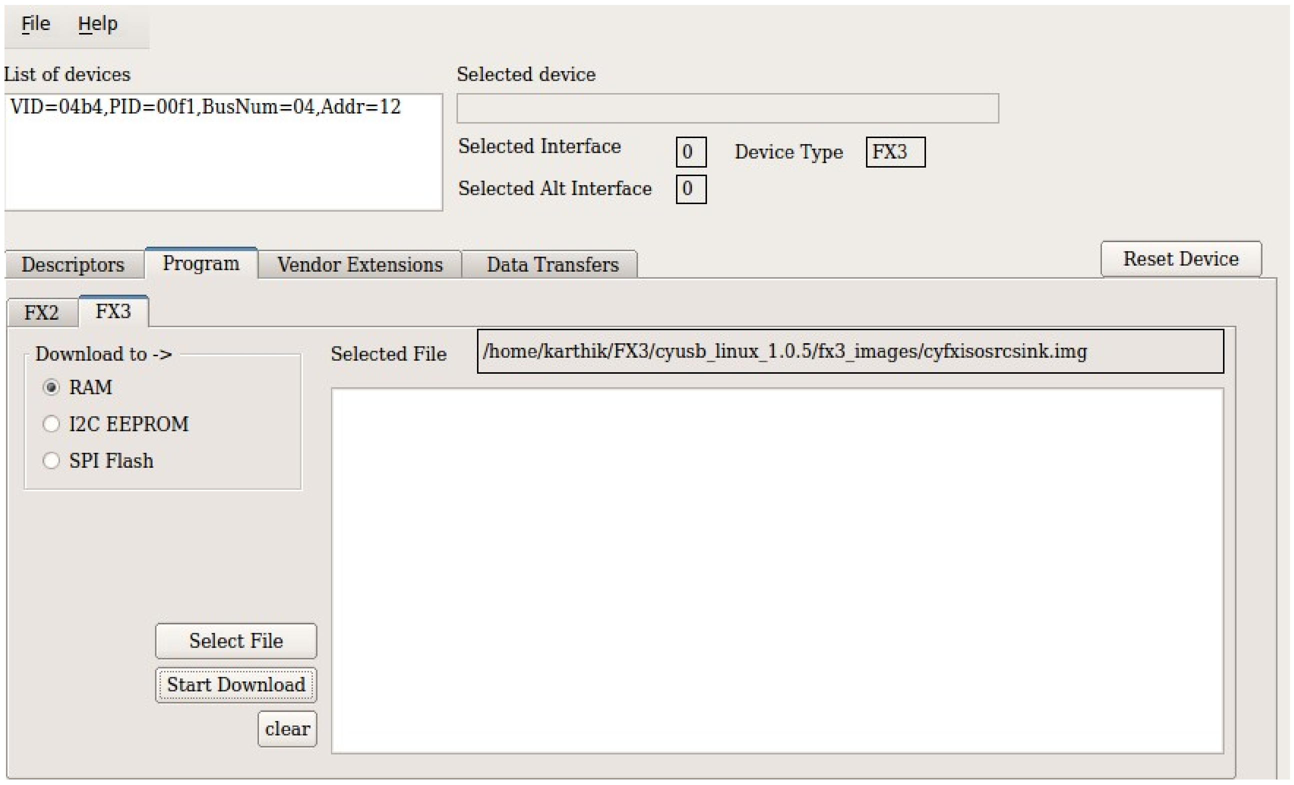
On selecting a device, detailed information is displayed about all its descriptors in the display area as shown in the following diagram :



```
<DEVICE>
bLength            = 18
bDescriptorType    = 1
bcdUSB             = 512
bDeviceClass       = 0
bDeviceSubClass    = 0
bDeviceProtocol    = 0
bMaxPacketSize     = 64
idVendor           = 04b4
idProduct          = 00f3
bcdDevice          = 256
iManufacturer      = 1
iProduct           = 2
iSerialNumber      = 3
bNumConfigurations = 1
</DEVICE>
<CONFIGURATION>
bLength            = 9
bDescriptorType    = 2
TotalLength        = 18
Num. of interfaces = 1
bConfigurationValue = 1
iConfiguration     = 0
bmAttributes       = 128
Max Power          = 0100
            <INTERFACE>
            bLength            = 9
            bDescriptorType    = 4
```

Programming the device :

On clicking the Program tab, we get the following screen :



List of devices
VID=04b4,PID=00f1,BusNum=04,Addr=12

Selected device

Selected Interface 0 Device Type FX3
Selected Alt Interface 0

Download to ->
[● RAM]
○ I2C EEPROM
○ SPI Flash

Selected File /home/karthik/FX3/cyusb_linux_1.0.5/fx3_images/cyfxisosrcsink.img

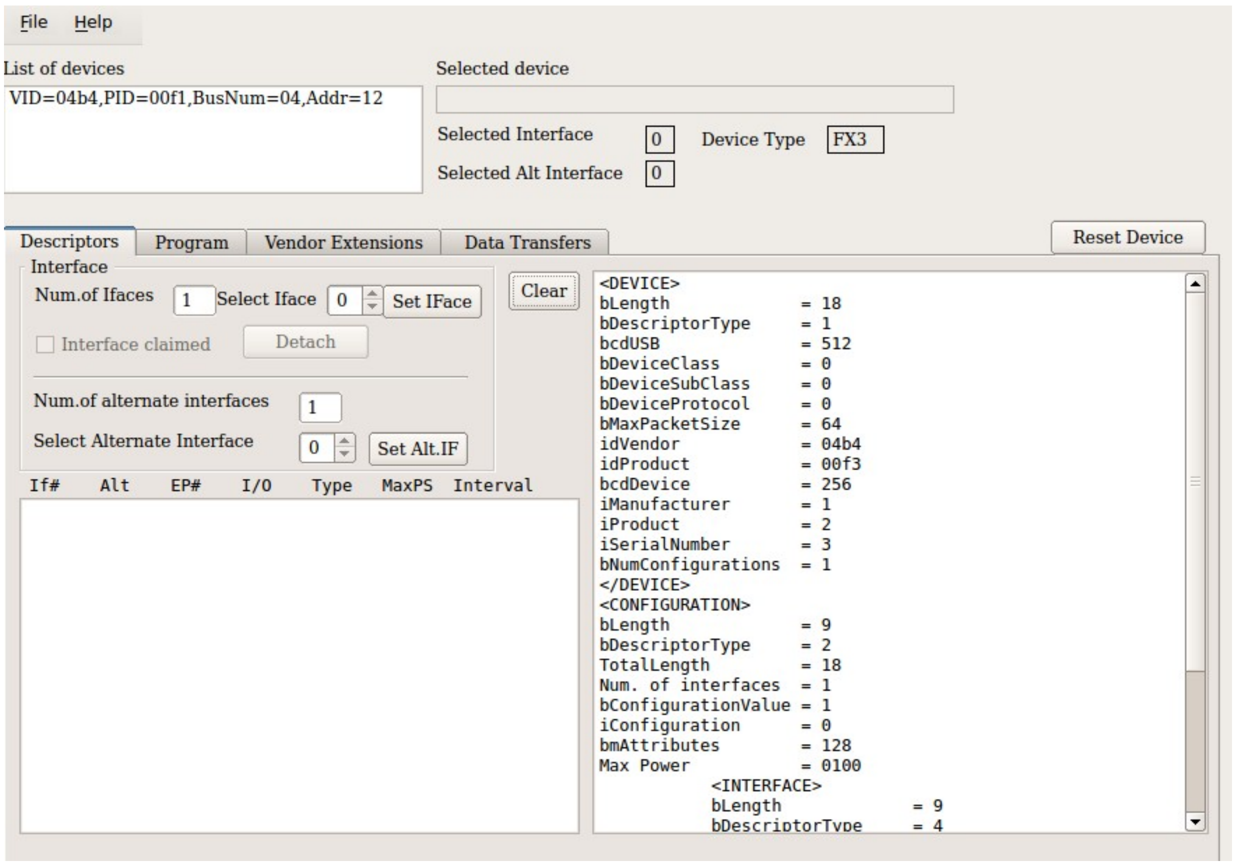
You can select to download a firmware image to either RAM, or an I2C EEPROM or an SPI flash on the target, but **REMEMBER** to make the necessary jumper adjustments !! (Read the h/w manual)

If you click on the Select File button, you get a standard file dialog box which allows you to navigate and select a firmware image to download.

Also, since this application can be used for both FX2 as well as FX3 devices, the application grays out the tab accordingly, i.e if an FX3 device has been selected for programming, the FX2 tab is grayed out and vice-versa.

Once a file image has been selected for download, clicking the **Start Download** button actually downloads the file into the target. Typically, the device would now Re-Enumerate as a new device and appear again with a new device address in the device list window shown earlier.

Going back to the **Descriptors** tab will show the details of the device, both in the detailed window as well as the sumary window.



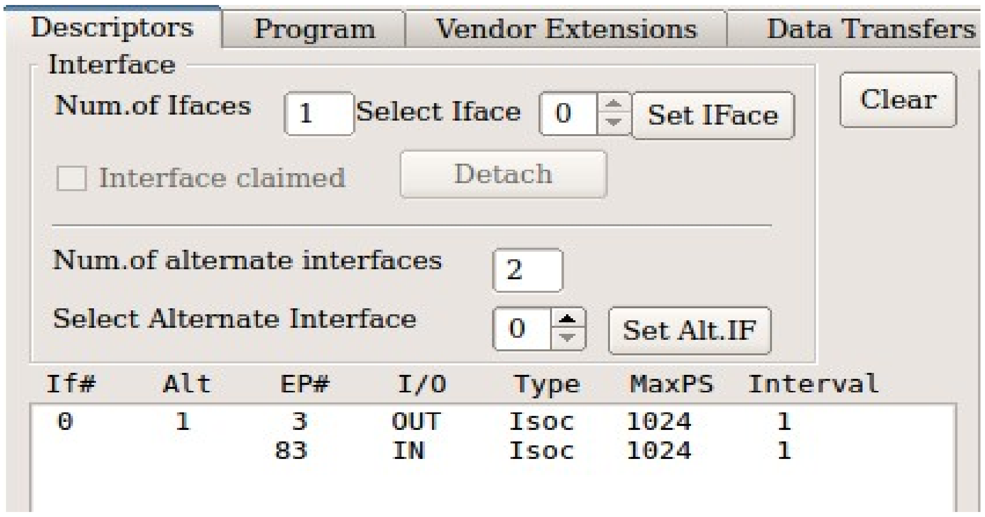
VID=04b4,PID=00f1,BusNum=04,Addr=12

Selected Interface: 0
Device Type: FX3
Selected Alt Interface: 0
Num.of Ifaces: 1
Select Iface: 0
Num.of alternate interfaces: 1
Select Alternate Interface: 0

```
If#    Alt    EP#    I/O    Type    MaxPS  Interval
```

```
<DEVICE>
bLength              = 18
bDescriptorType      = 1
bcdUSB               = 512
bDeviceClass         = 0
bDeviceSubClass      = 0
bDeviceProtocol      = 0
bMaxPacketSize       = 64
idVendor             = 04b4
idProduct            = 00f3
bcdDevice            = 256
iManufacturer        = 1
iProduct             = 2
iSerialNumber        = 3
bNumConfigurations   = 1
</DEVICE>
<CONFIGURATION>
bLength              = 9
bDescriptorType      = 2
TotalLength          = 18
Num. of interfaces   = 1
bConfigurationValue  = 1
iConfiguration       = 0
bmAttributes         = 128
Max Power            = 0100
           <INTERFACE>
           bLength              = 9
           bDescriptorType      = 4
```

The device may now present itself with multiple interfaces with different capabilities, and it is now very important to actually select the appropriate alternate interface for further interaction, such as for data transfers (bulk/isoc) etc. This can be done using the Set Alt. IF button as under :



Num.of Ifaces: 1
Select Iface: 0
Num.of alternate interfaces: 2
Select Alternate Interface: 0

```
If#    Alt    EP#    I/O    Type    MaxPS  Interval
 0      1      3     OUT    Isoc    1024      1
              83     IN     Isoc    1024      1
```

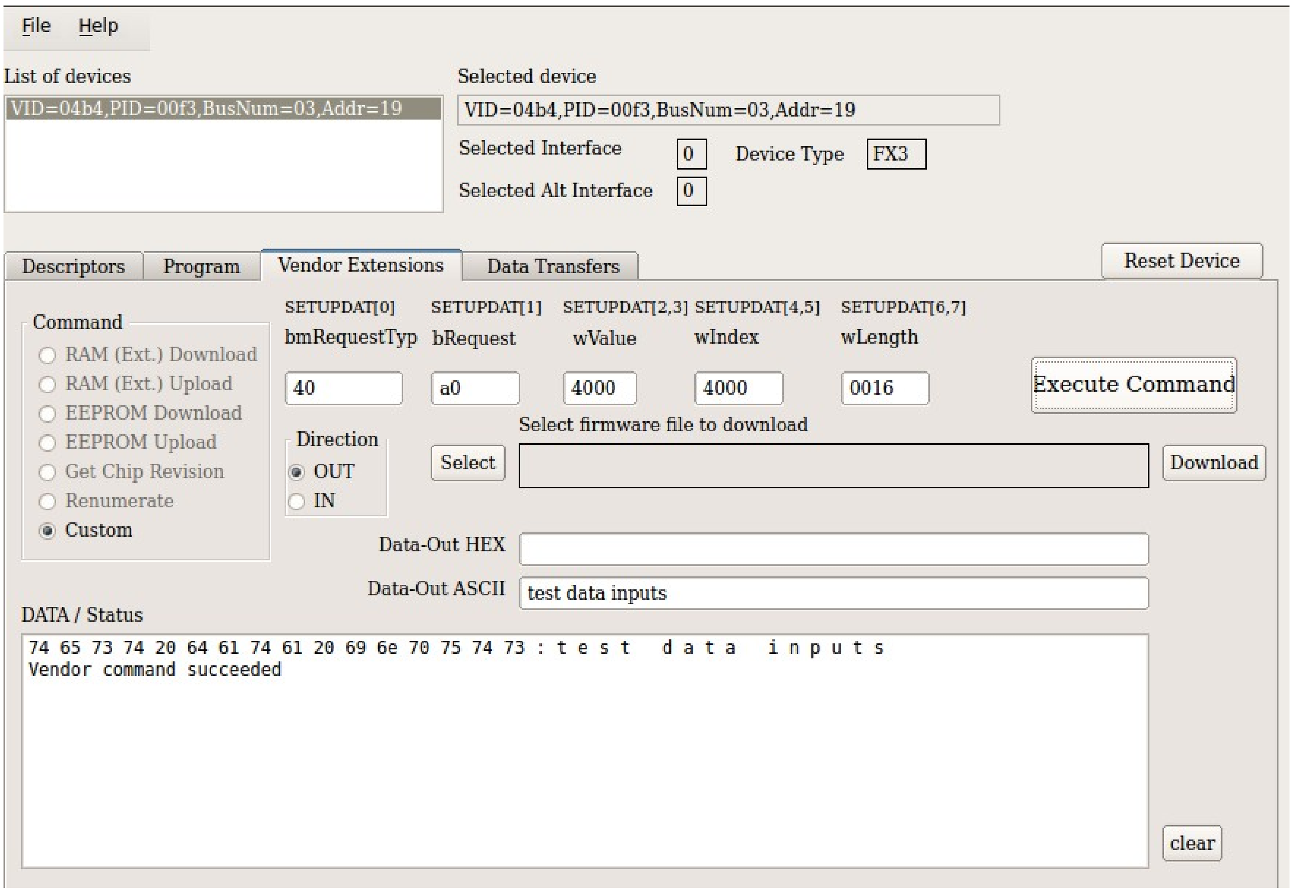
Using the Vendor Extensions tab :

This tab allows you to send custom control requests to the device.

For convenience, many standard options for the FX2 are already integrated into the display such as RAM (External) Download, RAM(External) Upload, EEPROM Download, EEPROM Upload, Get Chip Revision, and Renumerate.

The Custom option can be used to send any arbitrary command after setting appropriate values in the Control Setup packet fields such as bmRequestType, bRequest, wValue, wIndex, and wLength. In case the custom command also has an OUT data phase, the data can be input using the Data-Out HEX or Data-Out ASCII fields.

If the command has an IN data phase, the data and the command status are displayed in the DATA / Status text box.



List of devices: VID=04b4,PID=00f3,BusNum=03,Addr=19

Selected device: VID=04b4,PID=00f3,BusNum=03,Addr=19

Selected Interface: 0

Device Type: FX3

Selected Alt Interface: 0

Command: RAM (Ext.) Download / RAM (Ext.) Upload / EEPROM Download / EEPROM Upload / Get Chip Revision / Renumerate / [✓ Custom]

SETUPDAT[0] bmRequestTyp	SETUPDAT[1] bRequest	SETUPDAT[2,3] wValue	SETUPDAT[4,5] wIndex	SETUPDAT[6,7] wLength
40	a0	4000	4000	0016

Direction: [✓ OUT] IN

Select firmware file to download:

Data-Out HEX:

Data-Out ASCII: test data inputs

DATA / Status

```
74 65 73 74 20 64 61 74 61 20 69 6e 70 75 74 73 : t e s t   d a t a   i n p u t s
Vendor command succeeded
```

The Select button can be used to download a specific firmware image to test out a vendor command and the Execute Command button is used to actually execute the vendor command.

Performing BULK IN/OUT transfers :

After selecting a device and an appropriate interface+alt interface that supports Bulk transactions, when the Data Transfers tab is selected and Bulk Tab selected inside that tab, you will find that the appropriate Endpoints for OUT/IN are automatically available in the drop down combo boxes.

To send data out, we can use one of the following :

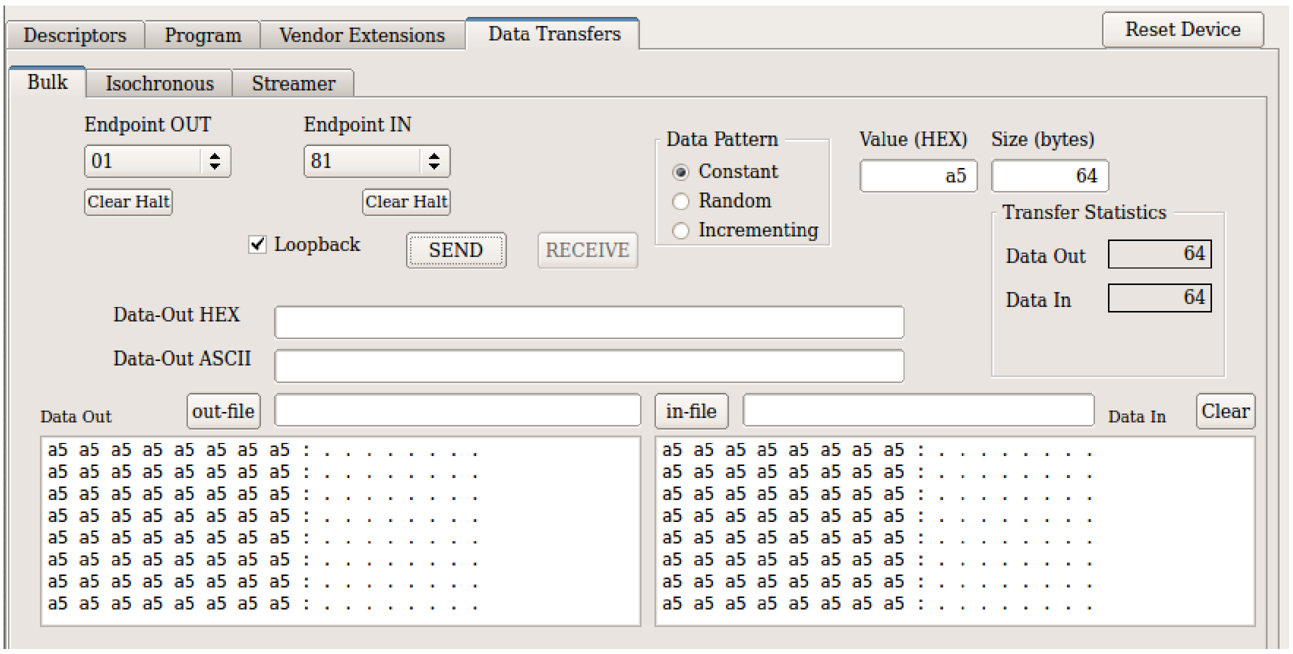
1. A constant data pattern, example 512 bytes of the character 'A', or
2. A random data pattern, using a random number generator program of the glibc library, or
3. An incrementing data pattern, with the start number being specified in the Value field, or
4. Select a file using the Out-File button and choosing any file from the system, or
5. Send selected values typed in through a text box in ASCII or HEX format.

To receive data from the device, we just set the data size to a multiple of the maximum packet size for the endpoint, and click the receive button. The actual data may be less than what we specified and this is reflected in the Data In transfer statistics fields.

Both data out and data in are shown in their respective windows.

In case the device has an appropriate loopback firmware image loaded, we can also click the loopback checkbox to automatically loopback the data after a write, i.e a **SEND** would automatically result in a **RECEIVE**.

For an example of a bulk out with a loopback and using a constant data of '0xA5' and sending 64 bytes, see picture below :



Endpoint OUT: 01
Endpoint IN: 81
[✓ Loopback]
Data Pattern: [✓ Constant] Random Incrementing
Value (HEX): a5
Size (bytes): 64
Transfer Statistics
Data Out: 64
Data In: 64

Data Out

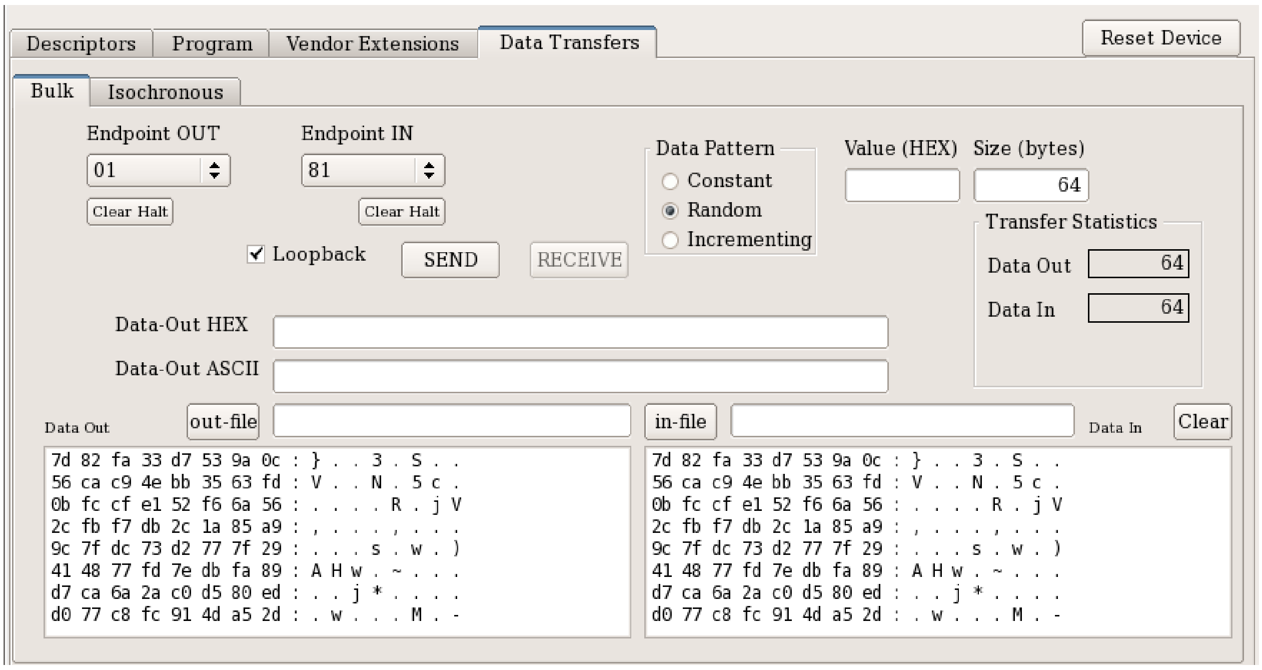
```
a5 a5 a5 a5 a5 a5 a5 a5 : . . . . . . . .
a5 a5 a5 a5 a5 a5 a5 a5 : . . . . . . . .
a5 a5 a5 a5 a5 a5 a5 a5 : . . . . . . . .
a5 a5 a5 a5 a5 a5 a5 a5 : . . . . . . . .
a5 a5 a5 a5 a5 a5 a5 a5 : . . . . . . . .
a5 a5 a5 a5 a5 a5 a5 a5 : . . . . . . . .
a5 a5 a5 a5 a5 a5 a5 a5 : . . . . . . . .
a5 a5 a5 a5 a5 a5 a5 a5 : . . . . . . . .
```

Data In

```
a5 a5 a5 a5 a5 a5 a5 a5 : . . . . . . . .
a5 a5 a5 a5 a5 a5 a5 a5 : . . . . . . . .
a5 a5 a5 a5 a5 a5 a5 a5 : . . . . . . . .
a5 a5 a5 a5 a5 a5 a5 a5 : . . . . . . . .
a5 a5 a5 a5 a5 a5 a5 a5 : . . . . . . . .
a5 a5 a5 a5 a5 a5 a5 a5 : . . . . . . . .
a5 a5 a5 a5 a5 a5 a5 a5 : . . . . . . . .
a5 a5 a5 a5 a5 a5 a5 a5 : . . . . . . . .
```

Note: For this example, we have downloaded cyfxbulklpautoenum.img image file, which loops back any data it receives on Bulk-IN to Bulk-OUT to the EZ-USB FX3 device.

As another example, using the same image, but now using a 64-byte randomly generated data with loopback, see image below :



Endpoint OUT: 01
Endpoint IN: 81
[✓] Loopback
Data Pattern: ○ Constant [● Random] ○ Incrementing
Value (HEX):
Size (bytes): 64
Transfer Statistics
Data Out: 64
Data In: 64
Data-Out HEX:
Data-Out ASCII:

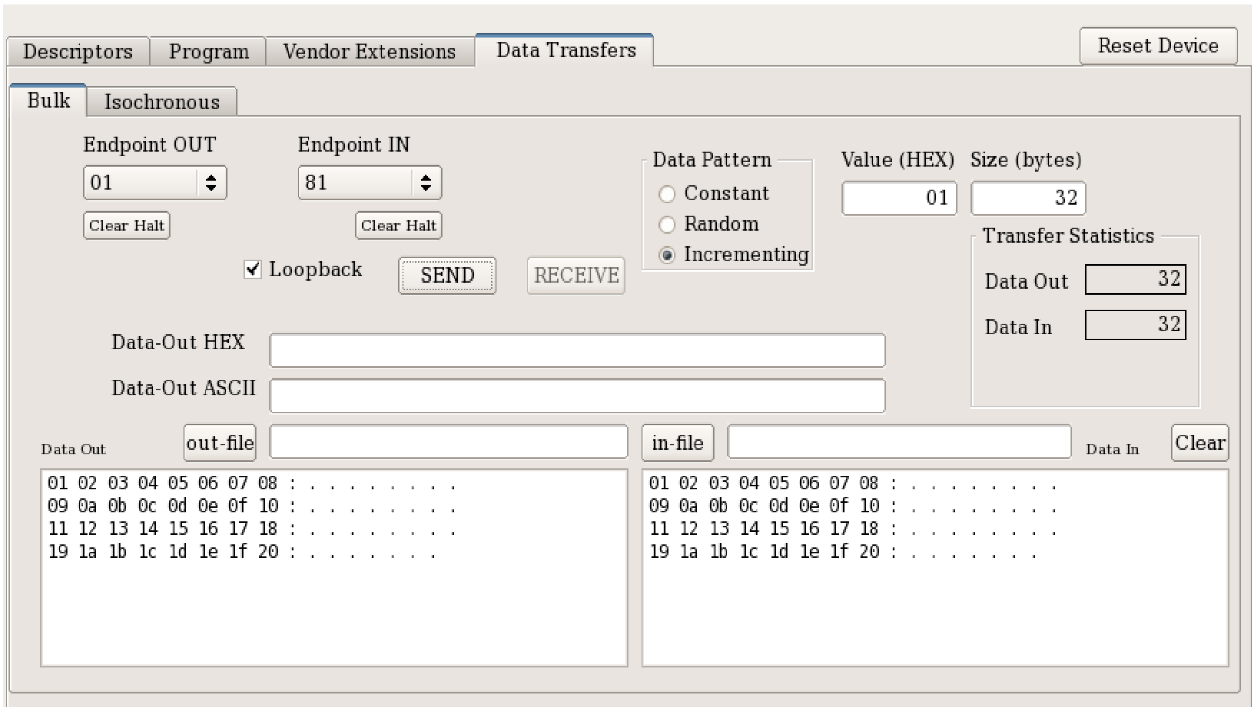
Data Out

```
7d 82 fa 33 d7 53 9a 0c : } . . 3 . S . .
56 ca c9 4e bb 35 63 fd : V . . N . 5 c .
0b fc cf e1 52 f6 6a 56 : . . . . R . j V
2c fb f7 db 2c 1a 85 a9 : , . . . , . . .
9c 7f dc 73 d2 77 7f 29 : . . . s . w . )
41 48 77 fd 7e db fa 89 : A H w . ~ . . .
d7 ca 6a 2a c0 d5 80 ed : . . j * . . . .
d0 77 c8 fc 91 4d a5 2d : . w . . . M . -
```

Data In

```
7d 82 fa 33 d7 53 9a 0c : } . . 3 . S . .
56 ca c9 4e bb 35 63 fd : V . . N . 5 c .
0b fc cf e1 52 f6 6a 56 : . . . . R . j V
2c fb f7 db 2c 1a 85 a9 : , . . . , . . .
9c 7f dc 73 d2 77 7f 29 : . . . s . w . )
41 48 77 fd 7e db fa 89 : A H w . ~ . . .
d7 ca 6a 2a c0 d5 80 ed : . . j * . . . .
d0 77 c8 fc 91 4d a5 2d : . w . . . M . -
```

As yet another example, using the same firmware image, but we now use an incrementing data pattern, with a start value of 0x01 and we send 32 bytes only :



Endpoint OUT: 01
Endpoint IN: 81
[✓] Loopback
Data Pattern: ○ Constant ○ Random [● Incrementing]
Value (HEX): 01
Size (bytes): 32
Transfer Statistics
Data Out: 32
Data In: 32
Data-Out HEX:
Data-Out ASCII:

Data Out

```
01 02 03 04 05 06 07 08 : . . . . . . . .
09 0a 0b 0c 0d 0e 0f 10 : . . . . . . . .
11 12 13 14 15 16 17 18 : . . . . . . . .
19 1a 1b 1c 1d 1e 1f 20 : . . . . . . . .
```

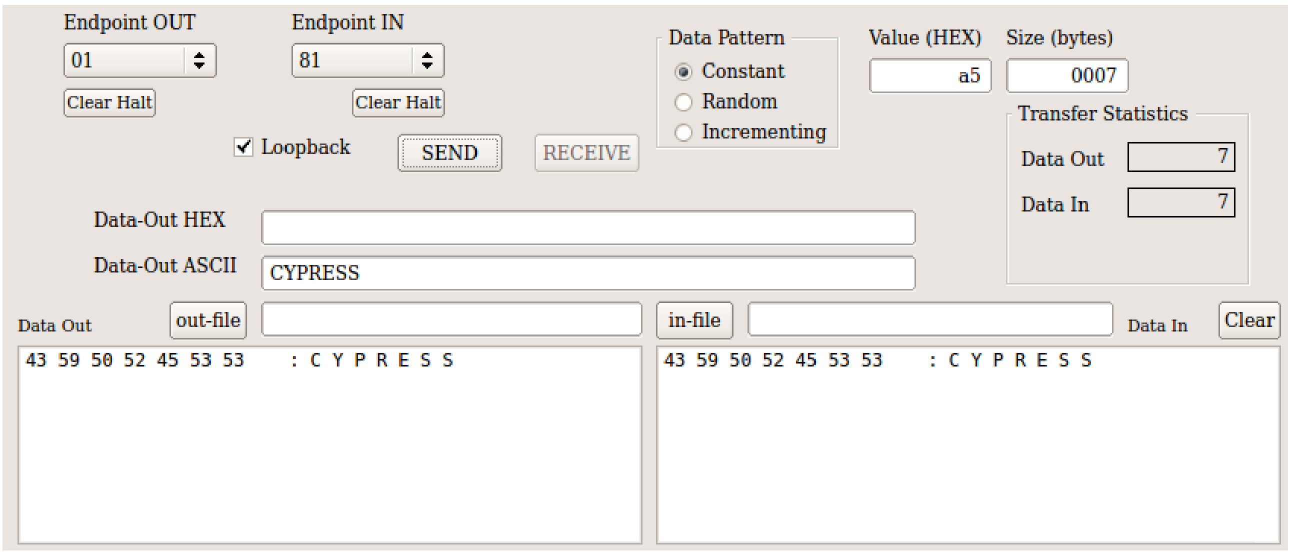
Data In

```
01 02 03 04 05 06 07 08 : . . . . . . . .
09 0a 0b 0c 0d 0e 0f 10 : . . . . . . . .
11 12 13 14 15 16 17 18 : . . . . . . . .
19 1a 1b 1c 1d 1e 1f 20 : . . . . . . . .
```

We can also use the out-file button to select a file from the computer to transmit and we can use the in-file button to capture the data sent out into a new file. We can use command line tools to do a diff to ensure that the data sent out is in fact correctly received without any data transmission errors.

You can also use the program in the test_cases directory to generate large amounts of data to store in a file and then send/receive and do further testing.

As a final example, we can enter random data directly in the Data-Out ASCII field or Data-Out HEX field and the system automatically calculates the length. If loopback is selected, we can see the data transferred out and back in :



Endpoint OUT: 01
Endpoint IN: 81
[✓ Loopback]
Data Pattern: [✓ Constant] Random Incrementing
Value (HEX): a5
Size (bytes): 0007
Transfer Statistics
Data Out: 7
Data In: 7
Data-Out HEX:
Data-Out ASCII: CYPRESS

Data Out
```
43 59 50 52 45 53 53     : C Y P R E S S
```

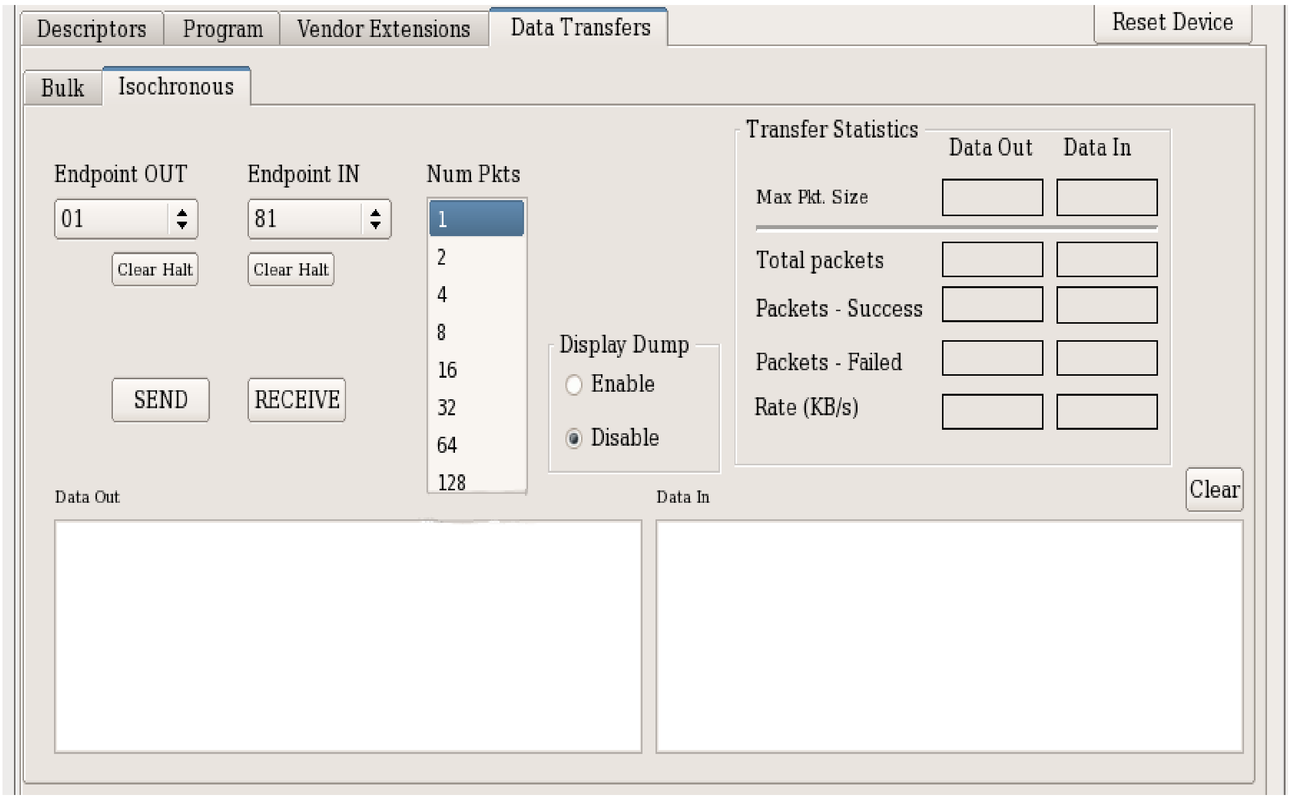
Data In
```
43 59 50 52 45 53 53     : C Y P R E S S
```

In the event of an endpoint Halt or Stall condition, we can clear the endpoint by issuing an explicit Clear Halt command by clicking the respective buttons under the Endpoint OUT and Endpoint IN drop down boxes.

Performing Isochronous Transfers :

After selecting a device and an appropriate interface+alt interface that supports Isoc transactions, when the Data Transfers tab is selected and Isochronous Tab selected inside that tab, you will find that the appropriate Endpoints for OUT/IN are automatically available in the drop down combo boxes.

The drop-down box for Num Pkts allows you to select the number of packets you wish to send or receive from the device. In the example shown below, we have used the cyfxisosrcsink.img which discards all data written to the OUT endpoint and generates a constant data pattern thru the IN endpoint.



Endpoint OUT: 01
Endpoint IN: 81
Num Pkts: 1, 2, 4, 8, 16, 32, 64, 128
Display Dump: ○ Enable ● Disable

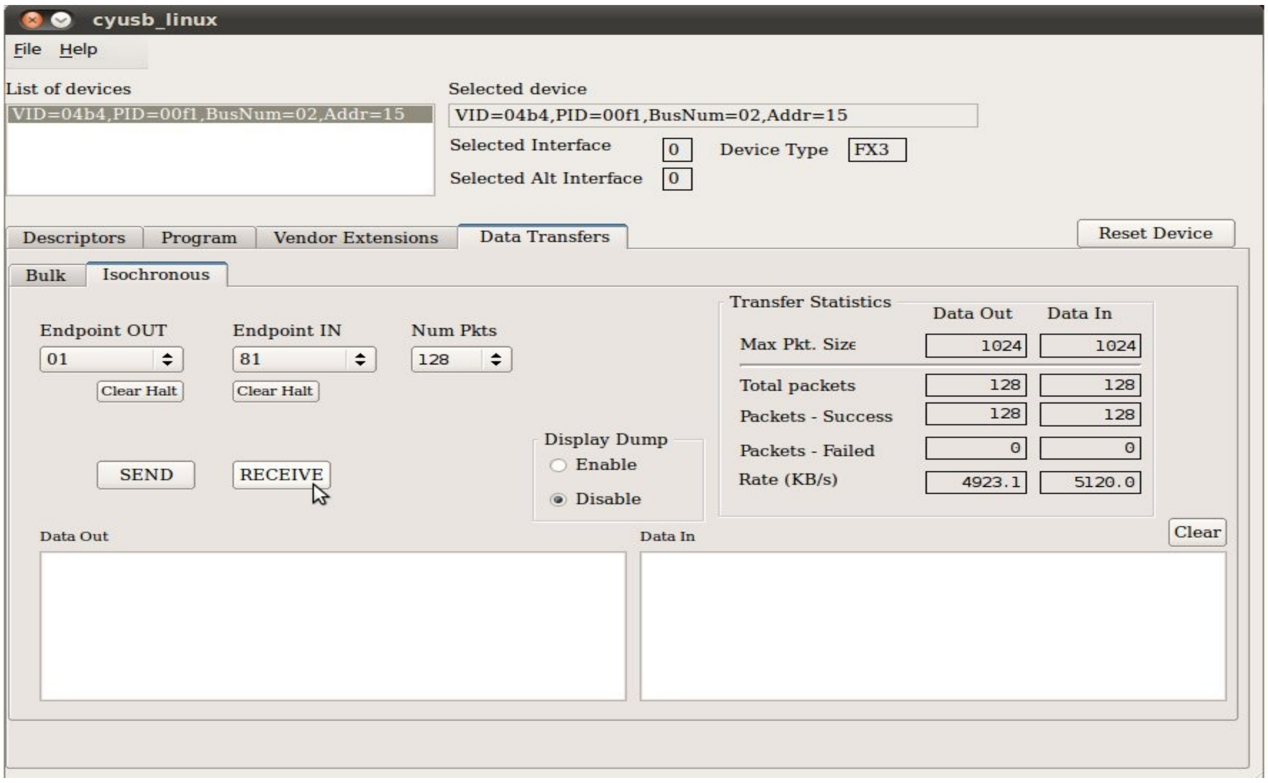
Transfer Statistics

	Data Out	Data In
Max Pkt. Size		
Total packets		
Packets - Success		
Packets - Failed		
Rate (KB/s)		

Data Out

Data In

In the example below we have selected 128 packets. The Max Packet Size field in the Transfer Statistics panel shows the data automatically retrieved from the device's descriptors, which in this case happens to be 1024 bytes. This means we are about to transfer 128 KB of data.



List of devices
VID=04b4,PID=00f1,BusNum=02,Addr=15

Selected device: VID=04b4,PID=00f1,BusNum=02,Addr=15
Selected Interface: 0   Device Type: FX3
Selected Alt Interface: 0

Endpoint OUT: 01
Endpoint IN: 81
Num Pkts: 128
Display Dump: ○ Enable ● Disable

Transfer Statistics

	Data Out	Data In
Max Pkt. Size	1024	1024
Total packets	128	128
Packets - Success	128	128
Packets - Failed	0	0
Rate (KB/s)	4923.1	5120.0

Data Out

Data In

Generally, for large isochronous transfers, we recommend that the Display Dump be disabled since it serves no purpose and also the objective is to estimate the speed of transfer and display would vitiate the timings.

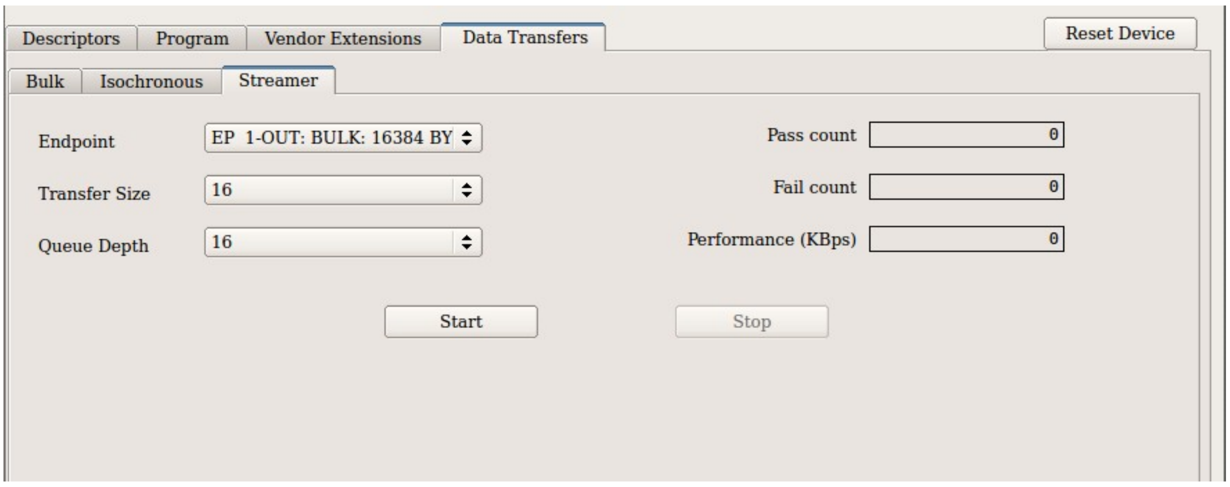
In the example above, we can see that 128 KB of data has been sent and 128 KB of data has been received (by clicking SEND and RECEIVE separately), and the data transfer rate is about 4.923 MB per sec and 5.120 MB per second respectively.

Measuring Transfer Performance:

The Streamer tab under Data Transfers can be used to perform continuous data transfers and measure the transfer performance. Since only one endpoint is selected for testing, the USB device implementation used should be capable of endlessly sending or receiving data on the selected endpoint.

For example, the cyfxbulksrcsink.img or cyfxisosrcsink.img files can be used to measure the data rates of bulk and isochronous transfers when using the FX3 device.

On selecting the streamer tab, you can see the following screen:

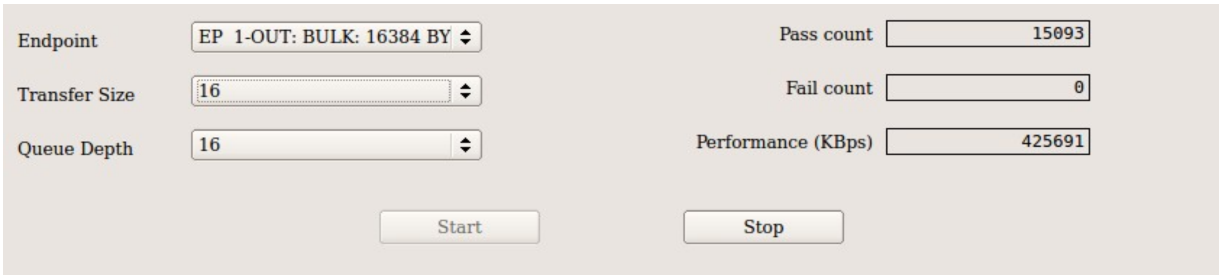


Endpoint: EP 1-OUT: BULK: 16384 BY
Transfer Size: 16
Queue Depth: 16
Pass count: 0
Fail count: 0
Performance (KBps): 0

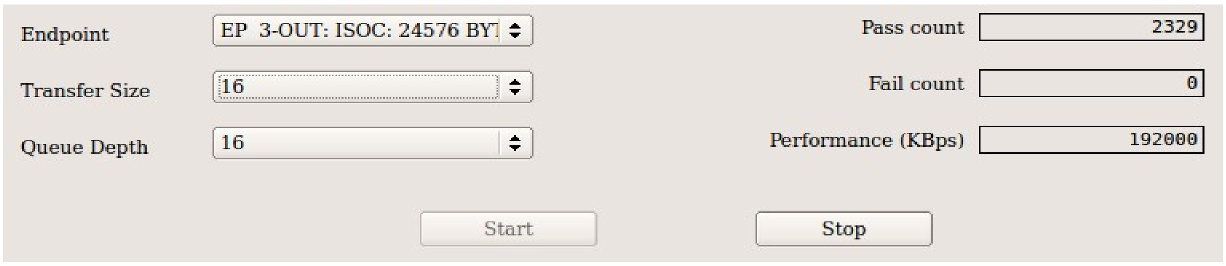
The drop down list next to Endpoint will list all of the bulk, interrupt and isochronous endpoints that are part of the active interface and alternate setting. The size listed is the product of the maximum packet size and the burst length for the endpoint. For isochronous endpoints, the number of bursts per frame (MULT) value is also considered when calculating the endpoint size.

The streamer application measures data rate by maintaining a queue of asynchronous data transfer requests. The Queue Depth setting specifies the number of elements in the queue, and the Transfer Size specifies the size of each request in terms of the endpoint size.

Once the parameters have been selected, hit the Start button. The number of Passing/Failing transfers and the measured performance are continously reflected in the output boxes on the right side of the screen. The following picture shows the output when measuring bulk transfer rate using the cyfxbulksrcsink firmware example:


Endpoint: EP 1-OUT: BULK: 16384 BY
Transfer Size: 16
Queue Depth: 16
Pass count: 15093
Fail count: 0
Performance (KBps): 425691

The following picture shows the output obtained when measuring isochronous transfer rate using the cyfxisosrcsink firmware example:


Endpoint: EP 3-OUT: ISOC: 24576 BYT
Transfer Size: 16
Queue Depth: 16
Pass count: 2329
Fail count: 0
Performance (KBps): 192000

Hit the Stop button to stop transfers and return to any other activity. The application will take a few seconds to stop the transfers and respond to further user input.

Chapter - 4 : Known Issues and Solutions

1. The cyusb library and application can only support 10 devices at a time. Please recompile the library and application after modifying the MAXDEVICES definition in cyusb.h, if a greater number devices is to be supported.
2. The cyusb library only looks at the first 100 VID/PID pair entries from the cyusb.conf file for driver binding. Please recompile the library and application after modifying the MAX_ID_PAIRS definition in cyusb.h, if this limit is to be changed.