USB 2 & 3 Loopback Plug Linux Programming Guide

Document Edition: 2.2
Date: 28 June 2017
Web site: http://www.passmark.com

Table of Contents

Introduction

This document will demonstrate how to communicate with PassMark USB2 and USB3 loop back plugs in Linux using the libusb library (http://www.libusb.info/). This document and demo are written using the 1.0 version of libusb (not the 0.1 version).

API documentation for libusb1.0 is available from http://libusb.sourceforge.net/api-1.0/.

The Linux_USB_API-Programming_Guide.zip file contains;

- The source code for the a demo project demonstrating how to use libusb
- This help document

Details about the USB2 loopback hardware can be found here,
http://www.passmark.com/products/usb2loopback.htm

Details about the USB3 loopback hardware can be found here,
http://www.passmark.com/products/usb3loopback.htm

This document complements the Windows USB2 API document
http://www.passmark.com/products/usb2api.htm

System Requirements

Linux environment with the 1.0 version of libusb installed.

USB3 plugs should be running the latest firmware (V2.3 or higher).
http://www.passmark.com/products/usb3updater.htm

Compiling

When compiling the demo project or other projects using libusb the library must be linked to, for example to compile the demo project use;

```
g++ usb_example.cpp –lusb-1.0
```

Data Types

Many of the required data types and constants are defined in PassMarkUSB.h in the demo project. These are outlined below.

Endpoints Addresses

```
#define USB_2_EPLOOPOUT 0x02        USB2 Loopback out
#define USB_2_EPLOOPIN 0x86         USB2 Loopback in
#define USB_2_EPBENCHOUT 0x04       USB2 Benchmark out
#define USB_2_EPBENCHIN 0x88        USB2 Benchmark in
#define USB_2_EPHISTORY 0x81        USB2 History report read
#define USB_3_EPLOOPOUT 0x01        USB3 Loopback out
#define USB_3_EPLOOPIN 0x81         USB3 Loopback in
```

Testing Modes

USB2

```
#define USB_2_LOOPBACK      1
#define USB_2_BENCHMARK     2
```

USB3

```
typedef enum _USB_3_TEST_MODE
{
        USB_3_TEST_LOOPBACK = 0,
        USB_3_TEST_BENCHMARK_READ,
        USB_3_TEST_BENCHMARK_WRITE,
        USB_3_TEST_BENCHMARK_RW
} USB_3_TEST_MODE;
```

When making a call to SendVendorCommand (from the demo) or libusb_control_transfer (from libusb) use one of these values to define the test type.

USB IDs

```
#define LOOPBACK_VENDOR_ID 0x0403
#define LOOPBACK_USB_2_PRODUCT_ID 0xff0a
#define LOOPBACK_USB_3_PRODUCT_ID 0xff0b
```

These are the vendor and product IDs for the PassMark USB2 and USB3 plug. These IDs are displayed when a plug is connected in system information sources such as dmesg and /proc/bus/usb/devices.

Test Pattern Types

```
typedef enum _DATA_PATTERN
{
        INCREMENTINGBYTE,
```

```
        RANDOMBYTE,
        CONSTANTBYTE
} DATA_PATTERN;
```

INCREMENTINGBYTE	Starts at a preset value and increases by 1 per byte sent
RANDOMBYTE	Uses a randomly generated value for each byte sent
CONSTANTBYTE	Uses a preset value for each byte sent

Use one of these values to define the test pattern type before starting a test loop, see line #155 in usb_example.cpp and several lines starting at line #164 controlling the pattern of data sent to the test plug.

When running a USB2 benchmark test only CONSTANTBYTE can be used.

Functions in usb_example.cpp

GetUSBPortsInfo

This function loads the contents of the /proc/bus/usb/devices or /sys/kernel/debug/usb/devices file into a buffer and scans through it looking for PassMark USB2 and USB3 plugs.

This can be used to build up a list of connected USB plugs and is used when connecting to a plug as well as matching a plug with which USB port it is connected to.

SendUSB_2_VendorCommand
SendUSB_3_VendorCommand

These are a wrapper over the libusb_control_transfer function that send different commands to the USB2 or USB3 plugs such as setting it to loopback mode and resetting the LED display of the plugs.

GetUSBDeviceInfo

This function makes several calls to libusb_get_string_descriptor_ascii and assembles the serial number and description from the USB plug.

ConnectUSBPlug

This will open a handle to the USB plug, get device information from the USB plug (serial number, description), set the USB plug into loopback mode and setup the input/output buffers.

When connecting to a USB3 plug this function must be called a second time after the initial connect with the bReconnect parameter set to true. This is because the USB3 plug will re-enumerate after begin set to loopback mode.

DisableUSB3LowPowerEntry

This will disable the entry of U1/U2 sleep modes. Disabling U1/U2 sleep modes entry during the test helps to prevent the surge currents needed when power cycling the USB3 PHY transceiver. These surge currents lead to sudden drop in the plug's supply voltage and could result in 8b/10b decoding errors.

EnableUSB3LowPowerEntry

This will enable the entry of U1/U2 sleep modes.

EnableUSB3ErrorCounts

This will enable the error counters of the USB transceiver.

GetUSB3GetErrorCounts

This will retrieve low-level error counters from a USB3 plug that is running firmware version 2.0 and higher, it is called during the loopback test. Before using it, the low power modes entry must be disabled and error counters must be enabled.

Starting A Test

Initialise libusb

Before any other libusb functions are called the usb library needs to be initialised first by calling libusb_init.

Find a PassMark USB plug

Call GetUSBPortsInfo to build up a list of USB plugs and choose a plug to test.

Connect to the USB Plug

The ConnectUSBPlug function gets information from the USB plug, such as the serial number, sets the plug into the loopback mode with SendUSB_2_VendorCommand or SendUSB_3_VendorCommand and sets up the input/output buffers associated with the plug.

After this call SendUSB_2_VendorCommand or SendUSB_3_VendorCommand to clear the bus error count and turn off any error LEDs currently on.

Cleaning Up

Once the test has been finished the input and output buffers are freed and libusb_release_interface should be called before using libusb_close on the USB plug device handle.

For Loopback Tests:

Select a Test Pattern

Select a test pattern (INCREMENTINGBYTE, RANDOMBYTE, CONSTANTBYTE) and the output buffer will be filled with data matching this pattern. The original value for the pattern buffer is initialised to 1 and this will be the starting point for each pattern.

Call the LoopbackTest function.

The Test Loop

At the start of the loop initialise the buffer for the current pattern. A packet number should be inserted at the start of the buffer.

The output buffer can then be written using libusb_bulk_transfer and the USB_2_EPLOOPOUT or USB_3_EPLOOPOUT endpoint, checking that the call succeeded and the number of bytes sent matches the number of bytes expected.

Now a packet should be read from the USB plug using libusb_bulk_transfer,, this read is part of a loop that checks the packet number for the read matches the previous write. If the read call is successful and the number of bytes read matches the number expected then the data is verified by comparing the input and output buffers.

If there is a mismatch the packet numbers are checked, if they match then there has been an error. If they do not match the data read should be ignored and usb_bulk_read called again to read what should be the correct packet. If after a set number of reads the packet numbers are still unmatched or there is no more data the loop should be exited and an error logged.

For Benchmark Tests:

USB2

The plug needs to be put into benchmark mode by calling SendUSB_2_VendorCommand with USB_2_BENCHMARK. There must be a 10msec delay after sending this command before starting to read data.

The output buffer can then be read using libusb_bulk_transfer and the USB_2_EPBENCHIN endpoint, checking that the call succeeded. Then a history report detailing the measured speeds can be read using the USB_2_EPHISTORY endpoint.

After a specified number of reads the loop will switch to sending data using using libusb_bulk_transfer and the USB_2_EPBENCHOUT endpoint.

USB3

See the USB_3_BenchmarkTest and USB_3_BenchmarkLoop functions in the example. First put the plug into benchmark mode using USB_3_TEST_BENCHMARK_RW, this can be done when calling ConnectUSBPlug and then calling ConnectUSBPlug again as changing the USB3 mode requires a re-enumeration.

Once connected the LCD display should be turned off using SendUSB_3_VendorCommand with USB_3_SET_DISP_MODE |

USB_3_DISPLAY_DISABLE. Then the benchmark test can begin by calling USB_3_BenchmarkLoop with the endpoint (USB_3_EPLOOPIN for write test, USB_3_EPLOOPOUT for read test), the number of packets to send and the size of the packets to send. The USB_3_BenchmarkTest function will run a write benchmark and then a read benchmark on the USB3 plug.