Application Note

USB 2.0 Compliance Mode for TUSB7320/7340 in Linux

Mike Yu

ABSTRACT

This application note is intended for hardware design engineers, signal integrity (SI) engineers, test technicians, and system integrators responsible for the electrical validation and compliance certification of systems integrating the TUSB7320 or TUSB7340 controllers. A foundational understanding of the Linux command line, the PCI Express (PCIe) configuration space, and general xHCI architecture is assumed.

Table of Contents

Trademarks

1 Introduction

Purpose and Background

During the hardware development phase of platforms utilizing the TUSB7320 or TUSB7340 xHCI host controllers, performing USB 2.0 electrical compliance testing—including eye-diagram, jitter, and signal quality analysis—is crucial to ensure strict adherence to the USB-IF physical layer specifications. To facilitate these measurements, physical test instruments, such as high-bandwidth oscilloscopes, require the downstream ports of the host controller to continuously transmit specific test patterns (e.g., Test Packet, Test J, Test K).

In a Windows® environment, developers typically utilize the official USB-IF Hardware Start Of Frame Evaluation and Test Tool (HSETT) to automate this process via a graphical user interface (GUI). However, the HSETT utility is exclusive to Windows environments and cannot be executed on Linux®-based platforms, such as embedded systems, industrial PCs, or enterprise servers. Consequently, on a Linux operating system, entering USB 2.0 Compliance Mode must be achieved manually through low-level command-line interfaces to directly manipulate the host controller internal registers. This application report provides a standardized, command-based methodology to successfully trigger USB 2.0 compliance test patterns under Linux.

Description and Scope

The procedures and configurations outlined in this document are applicable to the following Texas Instruments SuperSpeed USB 3.0 Host Controllers:

- TUSB7320: 2-Port USB 3.0 xHCI Host Controller
- TUSB7340: 4-Port USB 3.0 xHCI Host Controller

Because these devices share a highly compatible register architecture and memory-mapped I/O (MMIO) space configuration, the register-level commands detailed herein apply equally to both devices. This guide focuses on utilizing native Linux kernel interfaces and standard, open-source command-line utilities, eliminating the dependency on proprietary software or vendor-specific drivers.

2 Hardware Configuration and Setup

Test Setup Overview

To capture accurate waveforms for USB 2.0 electrical compliance testing, a proper physical connection between the Device Under Test (DUT), the test fixture, and the measurement equipment is required. The test configuration must isolate the USB 2.0 signaling lines (D+ and D-) to allow the oscilloscope to probe the continuous test patterns generated by the TUSB7320 or TUSB7340 downstream ports.

Required Equipment and Materials

The following hardware components are necessary to perform the evaluation: Host System (DUT): A hardware platform integrating the TUSB7320 or TUSB7340 host controller, running a functional Linux distribution.

- USB 2.0 Compliance Test Fixture: A standard USB-IF certified test board (such as the USB 2.0 Signal Quality Test Board) to break out the D+ and D- lines.
- Digital Storage Oscilloscope: A high-bandwidth oscilloscope (typically 2 GHz bandwidth for USB 2.0 High-Speed compliance) equipped with compliance analysis software.
- Probes/Cables: High-quality SMA cables or matched differential probes with appropriate bandwidth ratings to connect the fixture to the oscilloscope channels.

Hardware Connection Procedure and Requirements

To verify the TUSB7320/TUSB7340 controller successfully transitions into the commanded compliance state, the physical connection sequence must adhere to the following steps:

1. De-energize and Ground the System: Ensure the DUT and the oscilloscope share a common chassis ground to prevent Electrostatic Discharge (ESD) damage or ground loops from distorting the high-speed signaling measurements.
2. Connect the Test Fixture: Attach the USB-IF compliance test fixture directly to the target downstream USB port of the TUSB7320 or TUSB7340 being evaluated.
3. Establish Measurement Probing: Connect the SMA outputs or the differential probes from the test fixture's test points (corresponding to the selected port) to the input channels of the oscilloscope.
4. Verify Test Fixture Termination: Ensure that the test fixture is configured with the standard 50 Ω termination (or 90 Ω differential termination) as required by the specific test suite (for example, Test Packet or Test J/K).

Critical Hardware Note

The TUSB7320 and TUSB7340 utilize internal port logic to detect line state transitions and attach events. The physical test fixture must be connected to the port under test prior to executing the Linux software commands. If the controller does not detect the correct termination load on the D+ / D- lines, the port configuration logic can reject the test state transition or automatically reset the port, preventing the generation of the compliance pattern.

3 Software Preparation and Environment Setup

Embedded Utility Requirements (BusyBox Setup)

In many embedded Linux distributions, enterprise appliances, or minimalist OS configurations running on platforms with the TUSB7320 or TUSB7340, standard core utilities may be stripped down. To perform low-level memory manipulation, the system requires the BusyBox suite, which includes the essential devmem applet.

Before proceeding to the register configuration, engineers must check for BusyBox availability and install it if it is missing from the native system image.

1. Verify Existing BusyBox and devmem Applet. Check if BusyBox is already present on the platform and inspect its built-in command list by executing:

   ```
   busybox --help
   ```

2. Install BusyBox via Command Line. If the deployment platform has internet connectivity or access to local package repositories, execute the standard package manager command to fetch and install the complete BusyBox environment:

   ```
   sudo apt-get install busybox
   ```

3. Path Verification. Once installed, verify that the utility responds correctly to standard execution paths. If system symlinks are not automatically generated by the package manager, the applet can always be called explicitly from the command line using the busybox prefix:

   ```
   sudo busybox devmem --help
   ```

   Verify that the command outputs the syntax usage for reading and writing physical memory addresses before moving to the hardware mapping steps.

4 Memory-Mapped I/O (MMIO) Register Configuration

To force the TUSB7320/TUSB7340 downstream ports into USB 2.0 Compliance Mode without relying on high-level kernel drivers, software can directly manipulate the xHCI Operational Registers mapped into system memory. This section utilizes the busybox devmem utility to perform low-level read/write operations directly to the physical memory addresses.

Execute lspci to Locate the Host Controller on the PCI Bus

Before interacting with the hardware memory space, scan the PCI bus topology to identify the specific Bus, Device, and Function (BDF) address assigned to the Texas Instruments controller by the system architecture.

Execute the basic lspci utility to list all peripheral components connected to the PCI lanes:

```
lspci
```

Once the target entry is located (e.g., 01:00.0), re-run the command with the specific BDF slot and verbose hexadecimal dumping parameters to extract the 64-bit Base Address Register (BAR) mapping:

```
sudo lspci -s <BUS:DEV.FUN> -vxx
```

Identify the 64-bit memory base address corresponding to the xHCI Operational Registers from the output fields. Note that the controller allocates multiple memory regions; **select the address entry specified with [size=64K]** (e.g., Memory at f79f0000). This address serves as the foundational Base Address for all subsequent register offset calculations.

Halt the xHCI Controller

Before forcing individual ports into electrical compliance test modes, the xHCI controller must be halted to prevent operational software loops or interrupt routines from overriding the register states.

Modify the USB Command (USBCMD) register (Offset 0x20 from the base address) by clearing the Run/Stop bit to 0. Execute the following 8-bit devmem command:

```
sudo busybox devmem <Base_Address + 0x20> 8 4
```

Power Off the Target Downstream Port

To properly cycle the port logic and prepare the transceiver for compliance mode transition, the power to the specific downstream port must be negated.

Locate the Port Status and Control (PORTSC) register offset for the target port. Write to clear the port power bits utilizing a 32-bit width command:

```
sudo busybox devmem <Base_Address + Port_Offset> 32 0x0A000080
```

Transmit the USB 2.0 Test Pattern

Once the port is successfully sequenced, select the desired compliance test mode by writing to the Port Test Control field within the associated test register space (Offset + 0x4 from the base Port Offset).

To transmit a continuous USB 2.0 High-Speed Test Packet pattern (required for electrical eye-diagram analysis), execute the following 32-bit command:

```
sudo busybox devmem <Base_Address + Port_Offset + 0x4> 32 0x40000000
```

Upon successful execution, the physical layer (PHY) of the targeted port will immediately initiate continuous pattern generation.

Physical Ports Address

To test alternative physical ports, shift the target memory addresses based on the following table:

Table 4-1. TUSB7320/TUSB7340 Downstream Port Register Offsets

Port	Control Address	Test Command Address
1	0x0420	0x0424
2	0x0430	0x0434
3	0x0440 (1)	0x0444 (1)
4	0x0450 (1)	0x0454 (1)

1. TUSB7340 only

Execution Example

This section provides an actual implementation on Port1. The walkthrough is captured from a live test environment utilizing a standard Ubuntu platform.

1. Running the lspci discovery command yields:

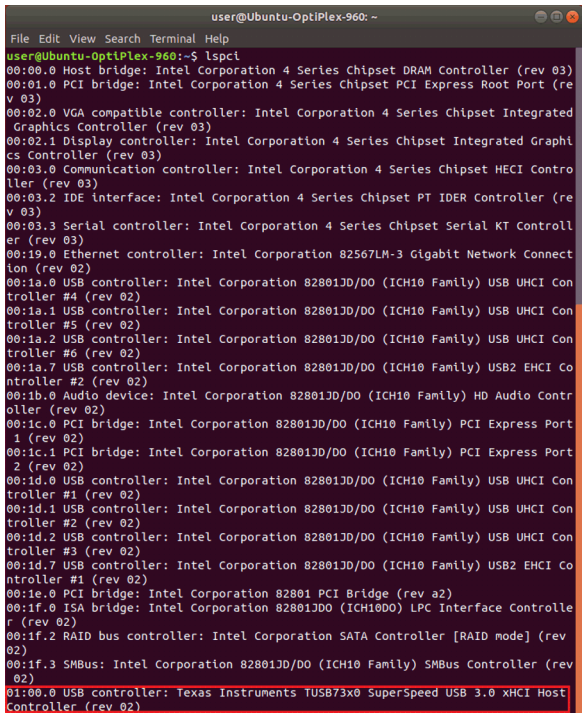

Figure 4-1. Discovery Command for PCI Address

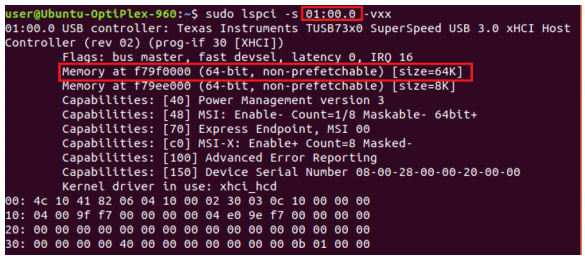

Figure 4-2. Command for Base Address

Key Terminal Output Extracted:

- PCI Address (BDF): **01:00.0**
- 64-bit Base Address (BAR): **0xf79f0000** (from Memory at f79f0000)

2. Full Test Packet Trigger Command Sequence. Using the extracted Base Address (0xf79f0000), the exact register execution log to trigger Port 1 (Control Address 0x420, Test Command address 0x424) into Test Packet Mode is detailed below:

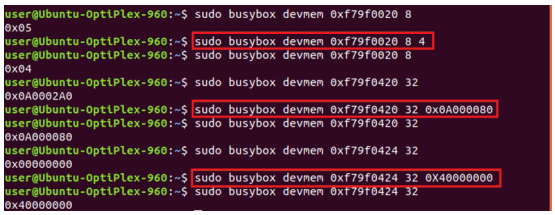

Figure 4-3. Command for Triggering Test Packet

5 Summary

Performing USB 2.0 electrical compliance validation on Linux platforms where standard graphical utilities like HSETT are unavailable requires a robust, low-level approach. By utilizing native PCI utilities (lspci) and direct memory manipulation through BusyBox (devmem), hardware validation and signal integrity engineers can consistently trigger the mandatory test patterns across the entire TUSB73x0 host controller family. Following the structural sequence outlined in this application report verifies repeatable, precise measurement workflows, paving a clear path toward successful USB-IF certification.

6 References

1. Texas Instruments, *TUSB7320 SuperSpeed USB 3.0 2–Port xHCI Host Controller*, product page.
2. Texas Instruments, *TUSB7340 SuperSpeed USB 3.0 4–Port xHCI Host Controller*, product page.
3. Keysight, *Understanding the USB 2.0 PID/VID Method for Testing High Speed Embedded Hosts*, webpage.
4. USB, *USB 2.0 Electrical Compliance Test Specification*, webpage.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale, TI’s General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Last updated 10/2025