Universal Serial Bus

Peripheral Development Kit Documentation

USB Single Step Transaction (USBSSTD)

User Manual

Revision 0.1

1. Overview

USB Transaction Single Step (USBSSTD) is a simple, low-level tool for exercising USB peripherals. USBSSTD is targeted to the needs of initial integration of a new USB peripheral. During initial hardware integration, it is advantageous to have complete control and visibility into the USB subsystem. USBSSTD provides this level of control and visibility. It provides an easy to use, graphical interface which allows the user to construct, submit, and evaluate the status of individual USB transactions. USBSSTD accommodates interactions with Host Controllers which conform to the Universal Host Controller Interface (UHCI).[1]

Although the granularity of control is per-transaction, the tool provides a number of aids to help the user to construct both individual, and sequences of transfers. These aids help the user to quickly setup and issue complete USB message traffic, or simply issue sequences of USB input or output transactions.

USBSSTD works independently of the USB software stack. It interacts directly with the host controller and schedule, to issue transfers and send/receive data with the addressed USB device. If the USB subsystem has a number of active devices, the user should take care in the use of this tool.

The following sections are organized in a top-down fashion, beginning with a description of the USBSSTD interface followed by an intuitive walk through of its use.

2. USBSSTD Interface

The primary dialog, illustrated in Figure 1, is organized into four sections. The top section contains controls for constructing a transfer descriptor. A transfer descriptor is used to instruct the USB Host Controller how to perform bus transaction.

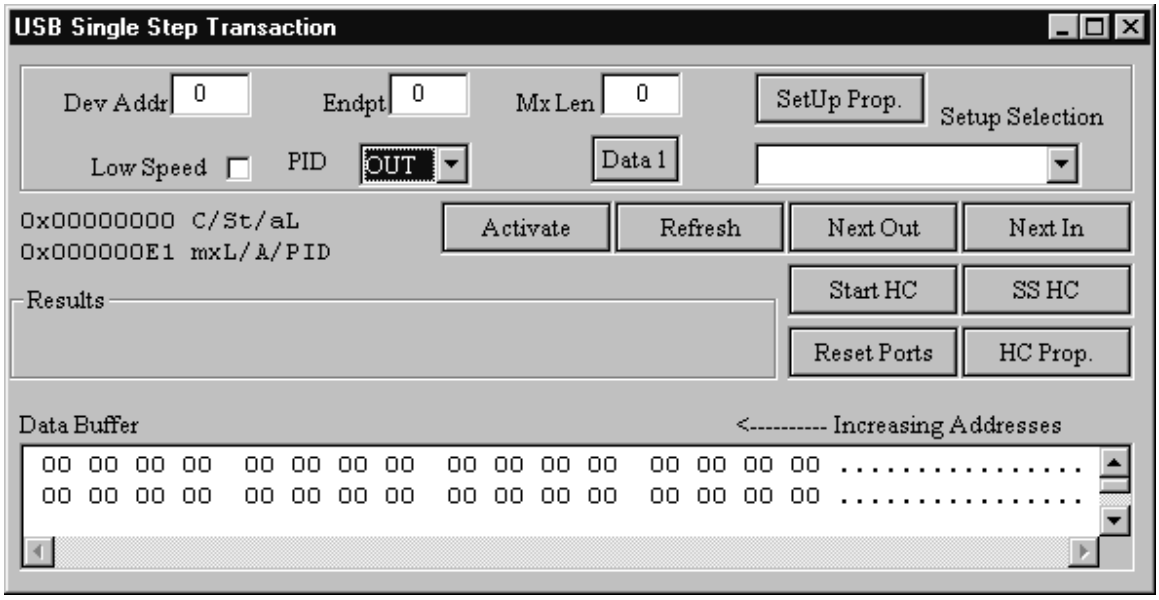

Figure 1: USBSSTD Main Dialog

[1] Please refer to the UHCI Specification for details.

There are two portions to the middle section. The left-hand middle contains transaction setup and execution status information. As the user constructs a new transfer, the control bits of the transfer appear in the Transfer Descriptor Window. Just below the Transfer Descriptor Window is the “Results” window. This window displays the latest transaction status using intuitive English phrases. At the same time, the Transfer Descriptor Window always shows the original bits from which the Results Window’s phrases are derived.

The right-hand middle contains action buttons for both transaction and host controller control. The details of their use is explained in following sections. Finally, the bottom of the dialog is a scrollable window into the data buffer associated with the transfer descriptor.

This interface is used to modify two data structures, and to interact directly with the USB Host Controller. The two data structures are defined below.

1. A data buffer used as data source or sink for the transaction (visible in bottom window of the main dialog).
2. A transaction descriptor which is set up via this interface and executed by the Host Controller. The interesting portions of the transfer descriptor are displayed in the Transfer Descriptor Window.

The usage paradigm of this interface is based on a three step process.

1. Initialization the transfer descriptor and data buffer. Transfer Descriptor initialization includes setting device addressing, data toggle, PID selection and expected data transfer length. Setting up the data buffer involves editing the data buffer to contain the correct data, if the transfer is Host to Device. Otherwise, the data buffer need not be modified by the user.
2. Transfer descriptor activation. Once the transfer descriptor and data buffer are initialized, the user must tell the USB Host Controller to issue the transaction to the USB device.
3. Analysis of transaction execution. Once the Host Controller has completed executing the transfer descriptor, the results are displayed in both the Transfer Descriptor and Results windows.

This simple three-step process is repeated for every transaction the user desires to execute. The following sub-sections cover the details of how the interface supports each of these steps.

2.1 Step 1: Transfer Descriptor/Data Buffer Initialization

The first step in issuing a transaction to the Host Controller is initialization of the transfer descriptor and data buffer. The top section of the main dialog (Figure 1) is surrounded by a box and contains controls for initialization of both the transfer descriptor and data buffer. These are called the initialization controls.

All but two of the controls provide specific capabilities for specific fields in the transfer descriptor. The other two controls provide easy shortcuts for initializing the transfer descriptor and data buffer for standard SETUP transactions.

There are three edit boxes, marked from left to right as: *Dev Addr*, *Endpt*, and *MxLen*. The first two edit boxes allow the user to specify the address of the USB device endpoint. These boxes accept hexadecimal numbers only, and are range-checked to ensure valid values are provided by the user. The *MxLen* edit box is used to set the Max Length field in the transfer descriptor. This value represents the number of bytes the Host Controller should send or expect to receive during the data portion of the bus transaction.

If the USB peripheral is a low speed device, the check box labeled as such should be checked. Checking this box sets a bit in the transfer descriptor informing the Host Controller that the USB device is a Low Speed device, and therefore should use low speed signaling protocol.

The user may initialize the transfer descriptor to perform an IN (Function to Host), OUT (Host to Function), or SETUP (Host to Function) transaction. The drop-down list control labeled *PID* is used by the user to set the transfer descriptor appropriately.

A button is provided to allow the user control over the DATA0/DATA1 bit, per transaction. This button is located next to the *PID* selection drop list. The text on the button reflects the action which will be taken if the button is pressed. For example, if the button text is “Data 1”, then pressing the button will set the data toggle to a ‘1’, and the button text will change to “Data 0”.

The final two controls in the initialization box are shortcuts for initializing the transfer descriptor and data buffer for SETUP transactions. Pressing the *SetUp Prop.* Button produces the dialog box illustrated in Figure 2. The Setup Data Properties dialog is a tabbed properties dialog which provides the user edit controls for each type of standard setup command. SendVendor and SendClass are not supported via this interface.

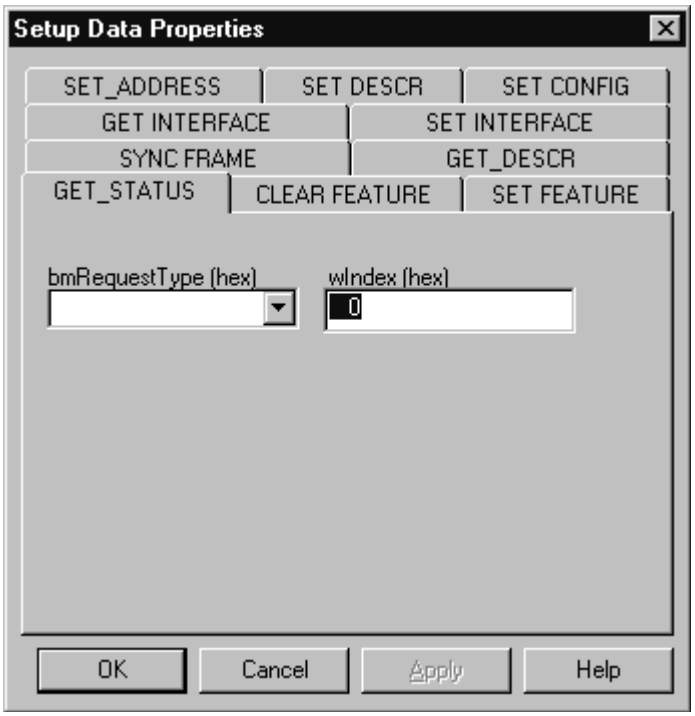

Figure 2: Setup Properties Dialog

After the user has initialized one or more of the standard setup properties, the ‘OK’ button is used to return to the main dialog. At this point, the user then uses the *Setup Selection* drop-down list, located below the *SetUp Prop*. Button to select one of the standard setup commands. The result of this selection is the transfer descriptor *and* data buffer are correctly initialized for the type of setup command selected.

Just as the *SetUp Prop*. Button and *Setup Selection* drop-down list are short-cuts for initializing for a SETUP transaction, there are two other buttons, located in the middle section of the main dialog, which provide short cuts for initializing IN and OUT transactions. These buttons are labeled: *Next Out* and *Next In*. The big benefit of these controls is that they automatically manage the DATA0/DATA1, and are synchronized with *Setup Selection*. For example, if the data portion of a control message is Host to Function, then each press of the *Next Out* will correctly initialize the transfer descriptor for the next OUT transaction (correct management of DATA0/DATA1, but the user must explicitly set the correct *MxLen*). The first press of *Next In* will initialize the transfer descriptor for the status stage of the control message, including setting of *MxLen* to issue a NULL data packet. Likewise, if the data portion were Function to Host, then the first press of the *Next Out* button would initialize the transfer descriptor to the status stage.

There will be situations where the shortcuts do not provide all of the proper initialization information. In these situations, the specific edit controls in the initialization window may be used to override values in the transfer descriptor. In addition, the data buffer may be directly edited at any time.

The data buffer window is the last edit box, extending across the entire bottom of the main dialog box. The bytes of the data buffer are displayed 16 bytes per line, read right-to-left (i.e. with least significant byte on the right).

After the transfer descriptor and data buffer are initialized, the user is ready to progress to step 2.

2.2 Step 2: Transfer Descriptor Activation

Once the transfer descriptor and data buffer are initialized, the next step is to have the Host Controller use the transfer descriptor to issue a USB Bus Transaction to the USB peripheral. Directly center on the main dialog box is a button marked *Activate*. Pressing this button causes the Host Controller to execute the transfer descriptor when it sees it during its scan of the schedule. USBSSTD does not provide any guarantees in which frame, or where in the frame the transaction will be issued.

When the *Activate* button is pressed, USBSSTD then deactivates the button. The button will remain deactivated until either the Host Controller has executed and de-activated the transfer descriptor, or the transfer descriptor is reloaded via the *Setup Selection*, *Next In*, or *Next Out* edit controls.

2.3 Step 3: Analysis of Transfer Results

USB Host Controller is an independent machine executing in parallel with the system process associated with USBSSTD. USBSSTD implements no synchronization mechanisms with the Host Controller. Therefore, once the transfer descriptor has been activated (via the *Activate* button), there is no feedback path from the Host Controller to USBSSTD to signal that it has changed the state of the transfer descriptor.

In order to evaluate the current state of the transfer descriptor, a *Refresh* button is provided to the user. This button simply provides an explicit mechanism for refreshing the display from the actual transfer descriptor and data buffer. This button is conveniently located next to the *Activate* button.

Between the press of the *Activate* button and a new transfer descriptor initialization, a press of the *Refresh* button will additionally cause USBSSTD to interpret the transfer descriptor's status bits, and display an appropriate phrase, with the current byte count to the Results Window (middle-left portion of main dialog).

The following list enumerates the phrases and interpretations the user will observe in the Results Window.

Result Phrase	Interpretation
SUCCESS	Transaction completed normally, all expected bytes transferred.
NACK	Transaction is still active, USB device is issuing a NACK response to the transaction.
STALLED/BITSTUFF	Transaction failed. Host controller detected a bitstuff error in the data sent by the USB device.
STALLED/TIMEOUT	Transaction failed. No response from device
STALLED/BABBLE	Transaction failed. Host controller received more bytes than expected from the device. The expected number of bytes was MxLen. The actual number of bytes received is also desplayed.
STALLED/FIFOERR	UHCI-specific error where there was either a FIFO overrun due to inability to get to PCI bus during a Function to Host transfer, or underrun for a similar reason during a Host to Function transfer.
STALLED	The device response to the transaction was STALL, not NACK or ACK.

2.4 Host Controller Interface

USBSSTD provides four controls for the Host Controller. ***These should be used with extreme caution when integrating on a populated USB system***. The Host Controller control buttons are located middle-right of main dialog, below *Next Out* and *Next In*.

The first button explicitly turns the host controller on or off. The text on the button reflects the action to be taken when the button is pressed. For example, if the button text says: “Start HC”, then pressing the button will cause the host controller to be started, and the button text will change to “Stop HC”.

Next to the Start/Stop HC button is the “single-step” host controller button. Prior to using this

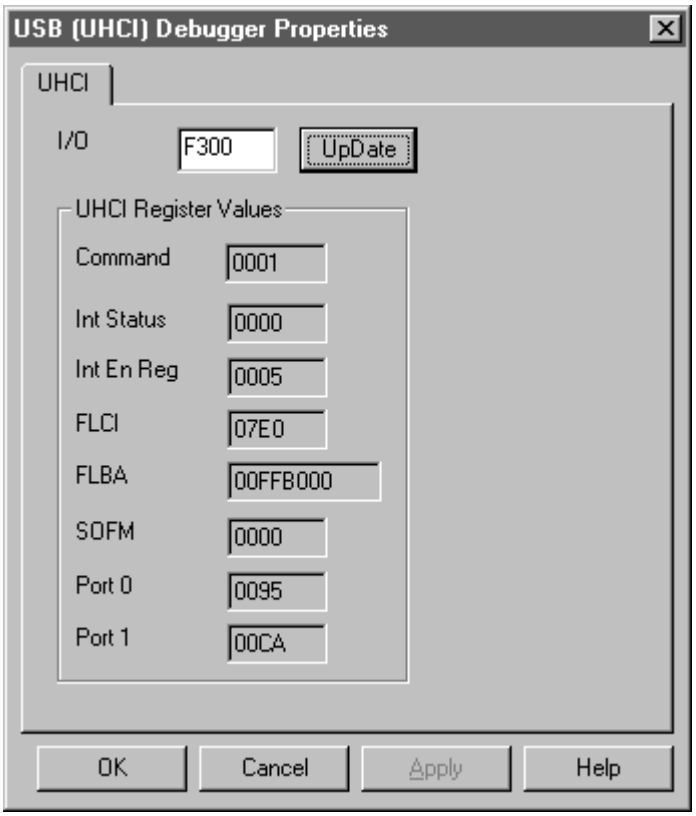

Figure 3: Host Controller Register Dialog

button, the user should make sure the host controller has been stopped (using the Start/Stop HC button). Pressing this button turns the Host Controller “on” until it successfully executes a transfer descriptor. At completion of the transaction, the Host Controller will turn itself off.

Directly below the Start/Stop HC button is a button for resetting the Host Controller’s port registers. This button will deliver a reset signal to all ports, and any devices connected to any subtree will be indiscriminately reset. All ports are re-enabled after the reset has been asserted for proper amount of time (i.e. > 25ms).

The final button in this group is marked as *HC Prop*. Pressing this button produces a dialog of the form illustrated in Figure 3. This dialog prints out the current values of the host controller’s registers. Detailed definitions of these registers are available in the UHCI Specification.

Typically, the user can use this dialog to verify the host controller is running, and the port to which the device under test is attached is enabled. Pressing the *Update* button on this dialog refreshes the display directly from the Host Controller’s registers. The Host Controller is running normally if the Command Register has a value of 0x0001, and on each press of the *Update* button, the value in the FLCI (Frame List Current Index) register changes. This is a verification that the Host Controller is running a walking the schedule, as expected.

In addition, the state of the port register connected to the device under test must have a value of 0x0095 if the connected device is a high speed and TBD if it is low speed. This really applies only if the device under test is connected directly to one of the root hub ports. If connected to a separate hub, then the port register(s) will reflect the state of the connection to the appropriate device(s).