Universal Serial Bus
Mass Storage Class

USB Attached SCSI Protocol (UASP)

Revision 1.0
June 24, 2009

Document Status

Revision History			
Rev	**Date**	**Author**	**Description**
1.0	June 24, 2009	Curtis E. Stevens	1) Initial Revision

The authors of this specification would like to recognize the following people who participated in the UASP Specification technical working groups. We would also like to thank others in the Promoter companies and throughout the industry who contributed to the development of this specification.

UAS Contributors		
Name	**E-Mail**	**Company**
Amit Nanda		Cypress Semiconductor
Hans van Antwerpen		Cypress Semiconductor
Chuck Trefts	chuck.trefts@ellisys.com	Ellisys
Mario Pasquali	mario.pasquali@ellisys.com	Ellisys
Yuji Oishi	bigstone.gm@hscjpn.co.jp	Hagiwara Sys-Com
Steve McGowan	steve.mcgowan@intel.com	Intel
Boris Dinkevich		Jungo
Joel Silverman	joel@k-micro.us	Kawasaki Microelectronics
Jason R. Oliver		MCCI
Cristian Chis	cchis@mcci.com	MCCI
John Garney	john.garney@mcci.com	MCCI
Jim Bovee	jibove@microsoft.com	Microsoft
Kiran Bangalore	kiran.bangalore@microsoft.com	Microsoft
Eugene Gryazin	eugene.gryazin@nokia.com	Nokia
Matthew Stephens	mstephens@plxtech.com	PLX/Oxford
Shigekatsu Tateno	stateno@plxtech.com	PLX/Oxford
Thomas Friend	tfriend@plxtech.com	PLX/Oxford
Dave Landsman	Dave.Landsman@sandisk.com	SanDisk
Martin Furuhjelm	martin.r.furuhjelm@seagate.com	Seagate Technology
Tony Priborsky	Tony.Priborsky@seagate.com	Seagate Technology
Cindy Lee		SMSC
Shannon Cash		SMSC
Gideon Intrater	Gideon.Intrater@symwave.com	Symwave
Biao Jia	Biao.Jia@symwave.com	Symwave
Rudy Liang	Rudy.Liang@symwave.com	Symwave
Christopher Thomas	chris.thomas@symwave.com	Symwave
Grant Ley	g-ley@ti.com	Texas Instruments
Ed Beeman		USB-IF
Paul E. Berg		USB-IF
Curtis E. Stevens	Curtis.Stevens@wdc.com	Western Digital Technologies

Please send comments via electronic mail to:

uasp@usb.org

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Purpose

The purpose of this class specification is to provide implementation information on how to use the [UAS] transport standard in a [USB2] or [USB3] system.

1.2 Scope

This specification provides usage examples as well as system recommendations for implementing [UAS] on [USB2] and [USB3] systems.

If there are conflicts between this specification and [UAS], then [UAS] takes precedence on all issues of conflict.

1.3 Related Documents

[SAM4] ISO/IEC 14776-414, SCSI Architecture Model-4 (SAM-4) (ANSI INCITS 447:2008)

[SAS] Serial Attached SCSI Revision 1.1 (SAS 1.1), (ANSI INCITS 417:2006)

[SPC] SCSI Primary Commands - 3 (SPC-3) (ANSI INCITS 408:2005)

[SBC] SCSI Block Commands - 2 (SBC-2) (ANSI INCITS 405:2005)

[RBC] Reduced Block Command Set (RBC) (ANSI INCITS 330:2000)

[MMC] MultiMedia Command Set - 5 (MMC-5) (ANSI INCITS 430:2007)

[SSC] SCSI Stream Commands - 2 (SSC-2) (ANSI INCITS 380:2003)

[OSD] Object-Based Storage Devices (OSD) (ANSI INCITS 400:2004)

[UAS] D2095 - USB Attached SCSI, INCITS T10. Download from www.t10.org.

[USB2] Universal Serial Bus Specification Revision 2.0, April 27, 2000. Download from www.usb.org.

[USB3] Universal Serial Bus 3.0 Specification Revision 1.0, November 12, 2008. Download from www.usb.org

[BOT] Universal Serial Bus Bulk Only Transport Rev 1.0, September 31, 1999. Download from www.usb.org.

[SATA] Serial ATA Rev 2.6 (SATA 2.6), February 15, 2007. Download from www.sata-io.org.

[AHCI] Advanced Host Controller Interface Rev 1.3, June 26, 2008. Download from developer.intel.com

[EHCI] Enhanced Host Controller Interface Rev 1.0, March 12, 2002. Download from www.intel.com

[xHCI] Extensible Host Controller Interface, still in development. Download from www.intel.com

For further information regarding ANSI standards, contact ANSI Customer Service Department at 212-642-4900 (phone), 212-302-1286 (fax) or via the World Wide Web at http://www.ansi.org.

2 Management Overview

The Bulk Only Transport used by [USB2] Mass Storage Class devices is simple and inexpensive to implement in microcontrollers – and hence suitable for inexpensive flash based storage products.

[BOT] is single threaded in that every transaction initiated by the host has to be completed by the device and completion status passed to the initiating process before the next transaction can be started. This creates a significant overhead (about 20%) of the total data transfer. [USB3] increases the data transfer bandwidth from 480 Mb/s to 5 Gb/s, but if the [BOT] protocol remains, analysis suggests that the actual transfer rate will be approximately 250 MB/s on a 2.4 GHz Core Duo™ with 12% CPU usage.

UASP (USB Attached SCSI Protocol) will improve efficiency in [USB2], and utilizing the full duplex capability of [USB3] allow for data transfers in excess of 400MB/s. The new protocol will require new host software and device firmware. Backwards compatibility is achieved though the device supporting both [BOT] and [UAS]. This document defines the implementation of the [UAS] protocol in [USB2] and [USB3].

When these standards ([UAS] and UASP) have been implemented in the host and device, the device will be accessed using the host SCSI software stack, and the USB interface will function as another SCSI host adapter in the host stack. Devices will implement the [SAM4] architecture model and thus be able to queue commands in the device, with the corresponding increase in performance.

3 Functional Characteristics

See [UAS], “USB Class Specific Requests” for class specific commands.

4 Standard Descriptors

See [UAS], “USB Descriptors” for UASP descriptor descriptions. Table 1 lists the descriptors defined in UAS.

Table 1 — Descriptors

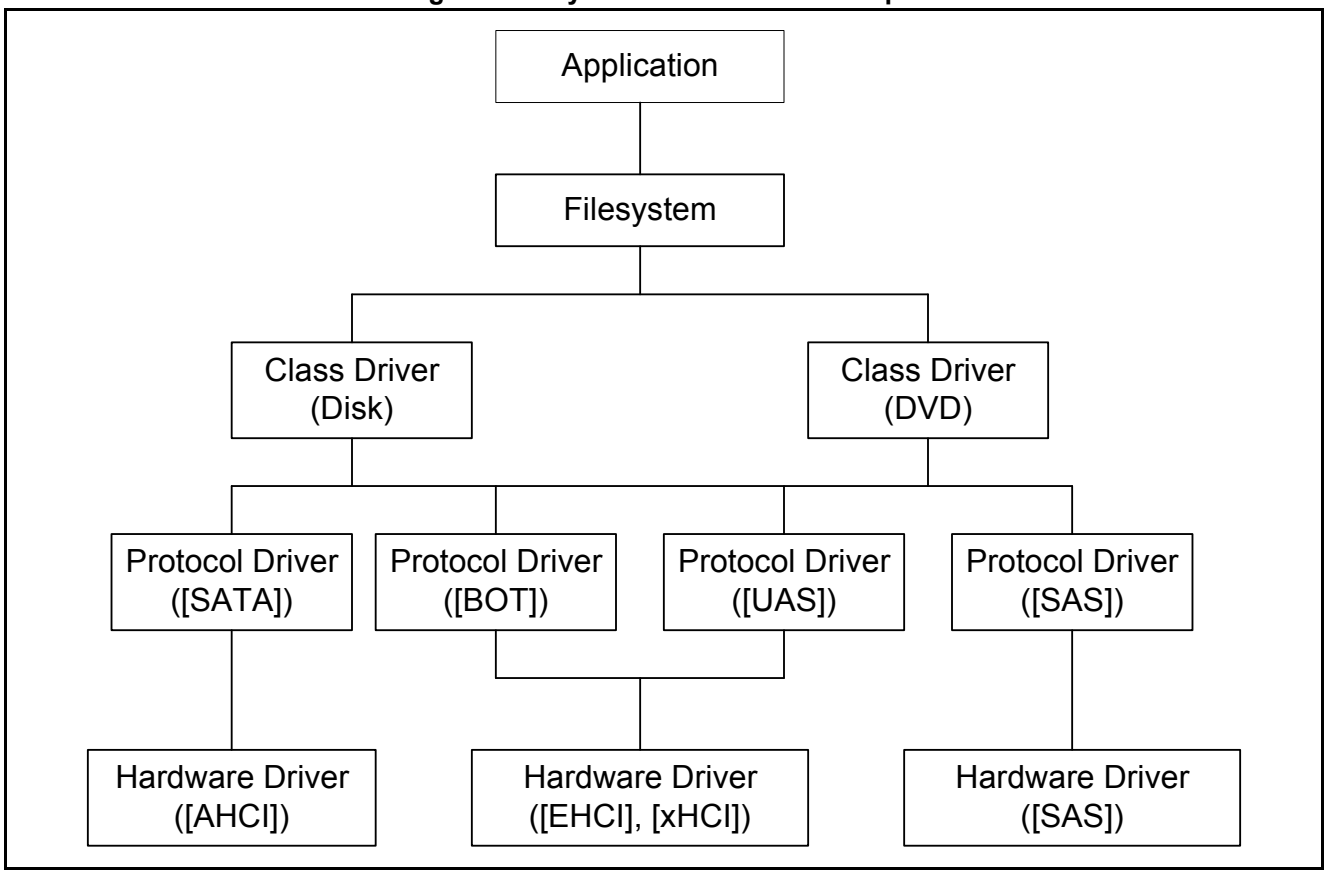
Descriptor Name
Device Descriptor
Configuration Descriptor
Interface Descriptor
Endpoint Descriptor
Pipe Usage Class Specific Descriptor

5 [USB3] Operation

5.1 Overview

Subclauses 5.3 and 5.4 provide examples of sample communications for non-data, data-in, data-out, and bi-directional data transfers. These examples are based on the system architecture described in figure 1.

Figure 1 — System Driver Stack Example



In figure 1, there is an application which reads or writes a file. The application uses the file system to open a file on a logical device. The file system uses the logical device information to locate the class driver. The class driver provides parameters which allows the file system to make requests for data transfer within the boundaries of the logical device. The class driver processes requests from the file system and creates commands or command sequences which are then sent to the target device using the protocol driver. Once the command sequences have been created, the class driver uses the logical device information to select the protocol driver. The protocol driver prepares the command sequence for transmission by performing any protocol specific operations necessary for sending the command. Finally, the protocol driver uses the logical device information to pass the request to the hardware driver. There may be multiple interactions between the protocol driver and the hardware driver to complete a request from the class driver.

There are a variety of implementation possibilities using the System Driver Stack Example. In some implementations, the class and protocol drivers are merged into a single driver. In other implementations, all three layers may be merged into a single driver.

[UAS] is a protocol standard architected to work with any SCSI command set standard (e.g., [SPC], [SBC], [RBC], [MMC], [SSC], and [OSD]). The protocol processing part of the system prepares command set specific requests and has an understanding of the underlying physical layer architecture.

The ladder diagrams in 5.3 and 5.4 start with the UAS Protocol driver receiving a command and show the communications flow as the command is placed into proper protocol format and then converted into [USB2] and [USB3] link layer requests for transmission to the device. These ladders do not document the physical layer.

Each diagram starts with the UAS driver receiving a command, the command is formatted and the transaction requests are sent to the hardware driver. In the ladder diagrams, the hardware driver is the xHCI or EHCI driver. The xHCI or EHCI driver then causes the xHCI or EHCI hardware to send transaction packets. The transaction packets are received by the device and contain the command, data, or status information necessary to complete the request. The ladder diagrams do not show layers (Link layer, Protocol, and Command) which would be used to interpret and complete the request from the host.

5.2 Figure Notation

Notation used in the diagrams and diagram descriptions (See 5.3 and 5.4) is as follows.

a) ITr (Pipe): Input Transfer request on the specified pipe.
b) OTr (Pipe): Output Transfer request on the specified pipe.
c) ITc (Pipe): Input Transfer request completion on the specified pipe.
d) OTc (Pipe): Output Transfer request completion on the specified pipe.
e) NAK (Pipe): [USB2] NAK response on the specified pipe.
f) DATA (Pipe): [USB2] Data transaction packet on the specified pipe.
g) OUT (Pipe): [USB2] output transaction on the specified pipe.
h) IN (Pipe): [USB2] input transaction request.
i) ACK (Pipe): [USB2] and [USB3] acknowledge transaction on the specified pipe.
j) IN ACK (Pipe): [USB3] input acknowledgment on the specified pipe.
k) IN ACK (Pipe, Prime): [USB3] input acknowledgment on the specified pipe priming the first operation (SID = Prime).
l) NRDY (Pipe): [USB3] primitive used by the device to indicate that the device is not ready to transfer data on the specified pipe.
m) NRDY (Pipe, Prime): [USB3] primitive used by the device to indicate that the device is not ready to transfer data on the specified pipe responding to the Prime pipe operation (SID = Prime).
n) DP (Pipe): [USB3] data packet.
o) DP (Pipe, Prime): [USB3] output data packet on the specified pipe priming the first operation (SID = Prime).
p) ERDY (Pipe): [USB3] primitive used by the device to initiate on input or output operation.
q) SIU: [UAS] Sense Information Unit.
r) RRIU: [UAS] Read Ready Information Unit.
s) WRIU: [UAS] Write Ready Iinformation Unit.
t) CIU: [UAS] Command Information Unit.
u) SID: [USB3] Stream ID.

Bracketing in these diagrams is implementation dependent. Bracketing requirements may change if the system does not follow the model described in these ladders. [UAS] supplies the protocol ordering requirements.

5.3 High-speed UASP

5.3.1 Overview

This subclause describes the High-speed packet exchanges required to support the UASP. High-speed devices emulate a subset of the [USB3] Streams protocol. Refer to subclauses 4.4.6.4 and 8.12.4.1 of the [USB3] specification for more information on the Streams protocol.

These figures are examples of a system using high speed operation through an EHCI driver. They also apply to a system doing high speed through an xHCI driver.

5.3.2 Non-data transfer

Non-data transfers are the simplest UASP operation. Non-data transfers only utilize the Command and Status pipes. A Command IU (CIU) is transmitted by the host on the Command pipe, and the device responds with a Response IU (RIU) on the Status pipe.

Figure 2 — Example High Speed UASP Non-Data Transfer

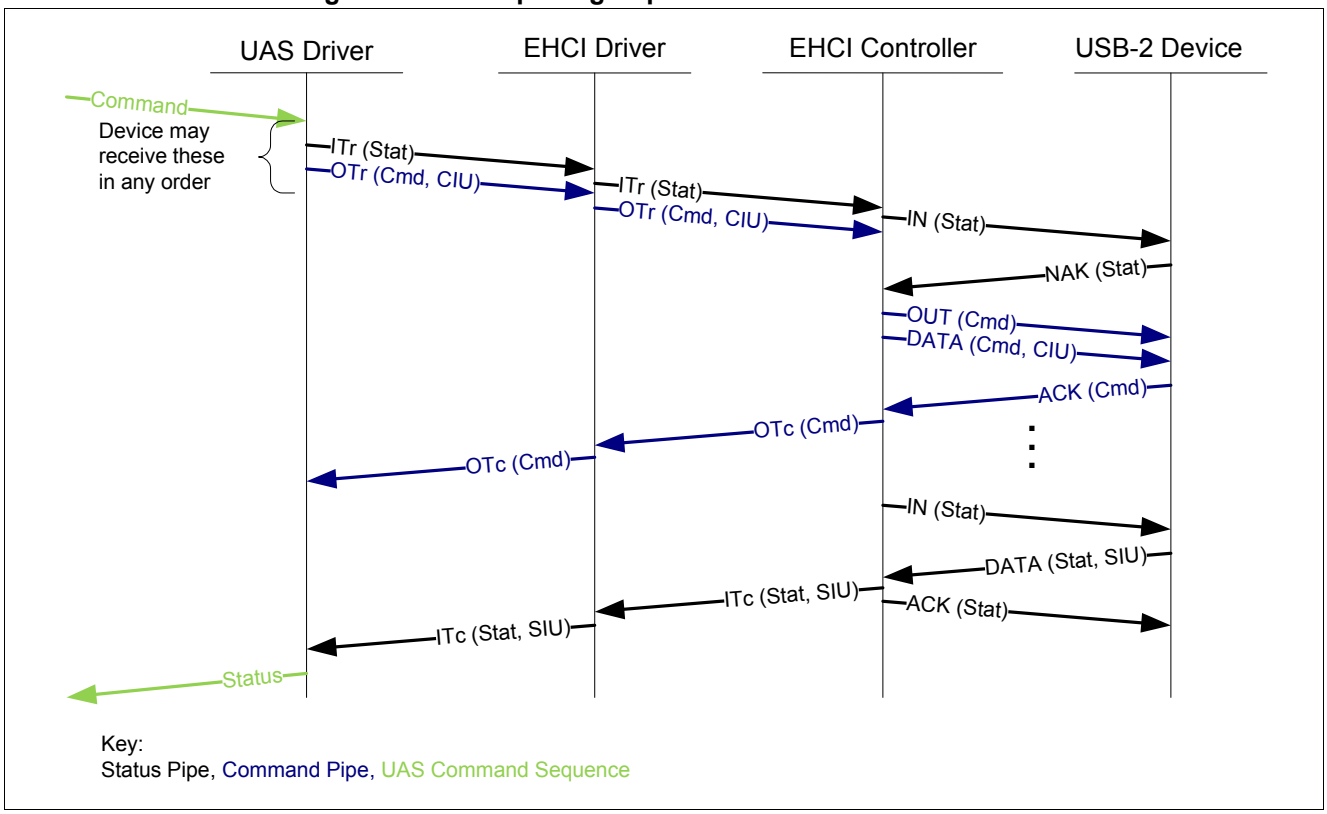


Figure 2 illustrates the execution of a High-speed UASP Non-Data Transfer. A Non-data Transfer Command to the UAS Driver causes the UAS drive to issue ITr (Stat) to the EHCI drive to allow the status pipe to receive the SIU for the command, and an OTr (Cmd, CIU) to transmit the CIU to the device. The CIU contains the tag used to match the SIU to the CIU.

The ITr (Stat) causes the EHCI Controller to generate an IN (Stat). In response to the OTr (Cmd) the EHCI contoller sends the CIU to the device by transmitting an OUT (Cmd), followed by a DATA (Cmd) which contains the CIU.

In response to the IN (Stat) packet the device returns NAK (Stat) packets until the device has status to return. When the device receives the DATA (Cmd, CIU), it responds with an ACK (Cmd), acknowledging the successful reception of the CIU. The CIU contains the command tag which is used to match the SIU with the CIU when the command completes.

The reception of the ACK (Cmd) for the CIU completes the OTr (Cmd) for the EHCI controller, which generates an OTc (Cmd) to the EHCI driver. The EHCI driver then returns OTc (Cmd) to the UAS driver indicating that the command has successfully been delivered to the device.

When the command is complete, the device responds to the IN (Stat) with a DATA (Stat, SIU) that contains the SIU.

The EHCI controller responds to the DATA (Stat) packet with an ACK (Stat), acknowledging the successful reception of the SIU. The reception of the DATA (Stat) packet completes the ITr (Stat) for the device, which generates an In ITc (Stat, SIU) to the EHCI controller. This in turn generates an ITc (Stat, SIU) to the UAS driver, indicating that the command has been completed.

The reception of the OTc (Cmd) for the CIU and the ITc (Stat, SIU) for the SIU, causes the UAS driver to compare the tag in the SIU with the tag in the CIU and then complete the command sequence by returning status to the layer above the UAS driver.

5.3.3 Data-in transfer

Figure 3 — Example High Speed UASP Data-in

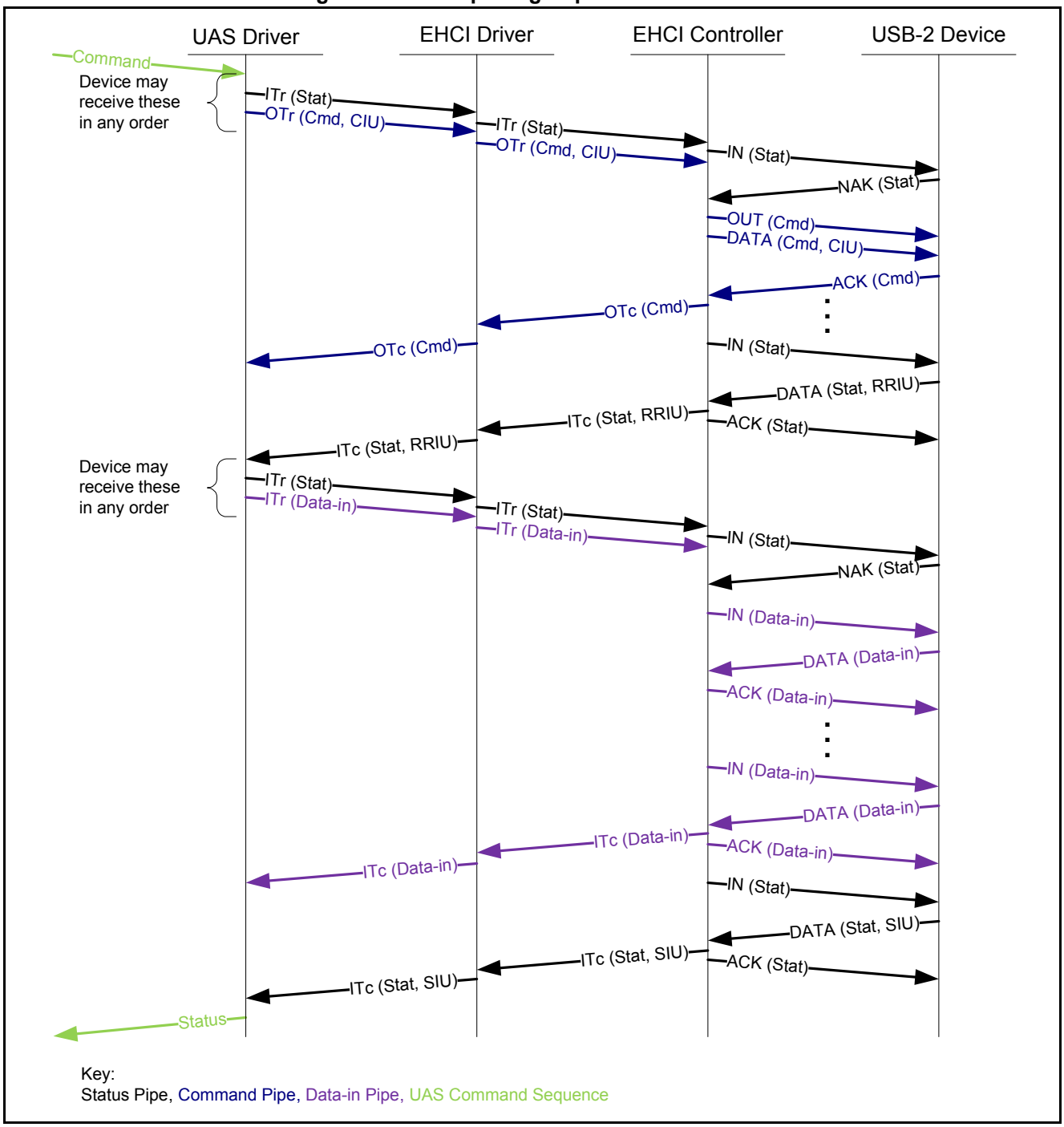


Figure 3 illustrates the execution of a High-speed UASP Data-in Transfer. A Data-in Transfer Command from the UAS Driver causes the EHCI driver to initiate an ITr (Stat) which allows the EHCI controller to receive the RRIU for the command, and an OTr (Cmd, CIU) to transmit the CIU to the device.

The ITr (Stat) causes the EHCI Controller to generate an IN (Stat). In response to the OTr (Cmd, CIU) the EHCI contoller sends the CIU to the device by transmitting an OUT (Cmd), followed by a DATA (Cmd, CIU).

In response to the IN (Stat) packet the device returns NAK (Stat) packets until the device has status to return. When the device receives the DATA (Cmd), it responds with an ACK (Cmd), acknowledging the successful reception of the CIU. The CIU contains the command tag which is used to match the SIU with the CIU when the command completes.

The reception of the ACK (Cmd) for the CIU completes the OTr (Cmd) for the EHCI controller, which generates an OTc (Cmd) to the EHCI driver. The EHCI driver then returns OTc (Cmd) to the UAS driver indicating that the command has successfully been delivered to the device.

When the device is ready to send the data for a command, it responds to the IN (Stat) polling with a DATA (Stat, RRIU).

The EHCI controller responds to the DATA (Stat, RRIU) packet with an ACK (Stat), acknowledging the successful reception of the RRIU. The reception of the DATA (Stat, RRIU) packet completes the ITr (Stat) for the EHCI controller, which generates an ITc (Stat, RRIU) to the EHCI driver. The EHCI driver returns an ITc (Stat, RRIU) to the UAS driver indicating that there is status available. The UAS driver evaluates the RRIU and checks the tag to determine which command the device is responding to with data. The UAS driver then issues an ITr (Stat) to the EHCI driver which enables the next status return by the device. The UAS driver then issues an ITr (Data-in) using the buffer indicated by the tag in the RRIU.

In response to the ITr (Stat), the EHCI driver issues an ITr (Stat) which schedules a data transfer on the status pipe. In response to the ITr (Stat), the EHCI controller issues IN (Stat) packets to the device and the device returns NAK (Stat) packets until the device has status to return.

In response to the ITr (Data-in), the EHCI driver schedules a data transfer on the Data-in pipe.

The EHCI controller begins receiving data from the device by transmitting an IN (Data-in).

When the device receives an IN (Data-in), the device responds with a DATA (Data-in) which contains read data.

When the EHCI controller receives the DATA (Data-in) packet on the Data-in pipe, it responds with an ACK (Data-in), acknowledging the successful reception of the read data.

The steps described in the previous three paragraphs repeat until the Data-in transfer is complete.

The reception of the last Data (Data-in) by the EHCI controller (e.g., the host buffer is full or a short packet is received) completes the ITr (Data-in) for the EHCI controller, which generates an ITc (Data-in) to the EHCI driver. This in-turn generates an ITc (Data-in) completion to the UAS driver.

When the command is complete, the device responds to the IN (Stat) with a DATA (Stat, SIU). The SIU contains the tag of the CIU.

The EHCI controller responds to the DATA (Stat, SIU) with an ACK (Stat) packet, acknowledging the successful reception of the SIU. The reception of the DATA (Stat, SIU) packet completes the ITr (Stat) for the EHCI controller, which generates an In ITc (Stat, SIU) to the EHCI driver. This in turn causes the EHCI driver to generate an ITc (Stat, SIU) to the UAS driver.

The reception of the OTc (Cmd) for the CIU, the ITc (Data-in) for the read data, and the ITc (Stat, SIU) for the SIU, all with the same tag, enables the UAS driver to complete the command sequence by returning status to the layer above the UAS driver. If the ITc (Data-in) has not been received by the UAS driver, then an error has occurred and the UAS driver should terminate the ITc (Data-in) before returning status.

5.3.4 Data-out transfer

Figure 4 — Example High Speed UASP Data-out Transfer

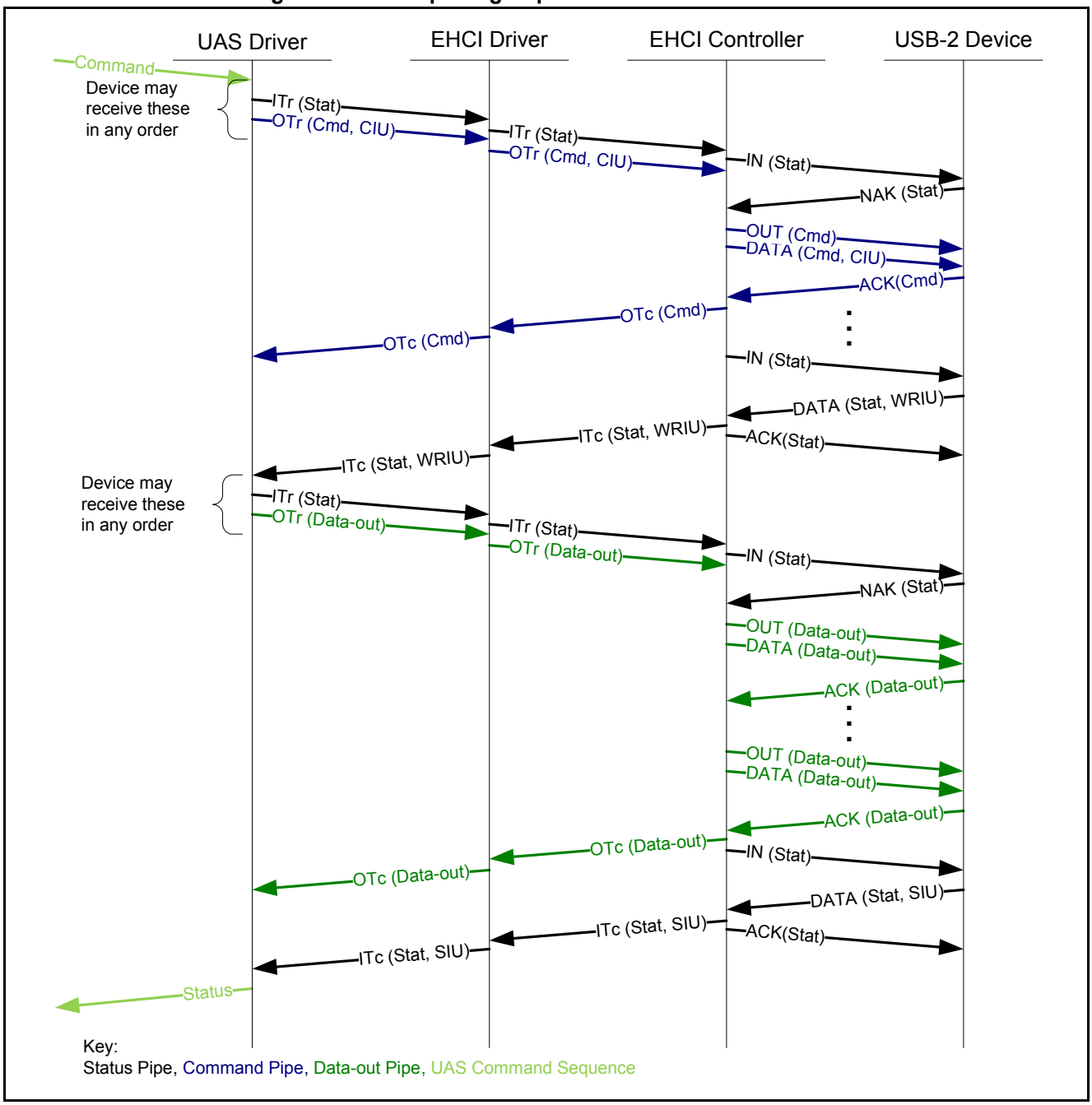


Figure 4 illustrates the execution of a High-speed UASP Data-out Transfer. A Data-out Transfer Command from the UAS Driver causes the EHCI driver to initiate an ITr (Stat) which allows the EHCI controller to receive the WRIU for the command, and an OTr (Cmd, CIU) to transmit the CIU to the device.

The ITr (Stat) causes the EHCI Controller to generate an IN (Stat). In response to the OTr (Cmd, CIU) the EHCI contoller sends the CIU to the device by transmitting an OUT (Cmd), followed by a DATA (Cmd, CIU).

In response to the IN (Stat) packet the device returns NAK (Stat) packets until the device has status to return. When the device receives the DATA (Cmd, CIU), it responds with an ACK (Cmd), acknowledging the successful reception of the CIU. The CIU contains the command tag which is used to match the SIU with the CIU when the command completes.

The reception of the ACK (Cmd) for the CIU completes the OTr (Cmd) for the EHCI controller, which generates an OTc (Cmd) to the EHCI driver. The EHCI driver then returns OTc (Cmd) to the UAS driver indicating that the command has successfully been delivered to the device.

When the device is ready to receive the data for a command, it responds to the IN (Stat) polling with a DATA (Stat, WRIU).

The EHCI controller responds to the DATA (Stat, WRIU) packet with an ACK (Stat), acknowledging the successful reception of the WRIU. The reception of the DATA (Stat, WRIU) packet completes the ITr (Stat) for the EHCI controller, which generates an ITc (Stat, WRIU) to the EHCI driver. The EHCI driver returns an ITc (Stat, WRIU) to the UAS driver indicating that there is status available. The UAS driver evaluates the WRIU and checks the tag to determine which command the device is requesting data for. The UAS driver then issues an ITr (Stat) to the EHCI driver which enables the next status return by the device. The UAS driver then issues an OTr (Data-out) using the buffer indicated by the tag in the WRIU.

In response to the ITr (Stat), the EHCI driver issues an ITr (Stat) which schedules a data transfer on the status pipe. In response to the ITr (Stat), the EHCI controller issues IN (Stat) packets to the device and the device returns NAK (Stat) packets until the device has status to return.

In response to the OTr (Data-out), the EHCI driver schedules a data transfer on the Data-out pipe.

The EHCI controller begins send the data to the device by transmitting an OUT (Data-out) followed by a DATA (Data-out).

When the device receives a DATA (Data-out), the device responds with an ACK (Data-out) which indicates that the write data has been received.

The steps described in the previous two paragraphs repeat until the Data-out transfer is complete.

The reception of the last ACK (Data-out) by the EHCI controller (e.g., the host buffer is empty or a short packet is sent) completes the OTr (Data-out) for the EHCI controller, which generates an OTc (Data-out) to the EHCI driver. This in-turn generates an OTc (Data-out) completion to the UAS driver.

When the command is complete, the device responds to the IN (Stat) with a DATA (Stat, SIU). The SIU contains the tag of the CIU.

The EHCI controller responds to the DATA (Stat, SIU) with an ACK (Stat) packet, acknowledging the successful reception of the SIU. The reception of the DATA (Stat, SIU) packet completes the ITr (Stat) for the EHCI controller, which generates an In ITc (Stat, SIU) to the EHCI driver. This in turn causes the EHCI driver to generate an ITc (Stat, SIU) to the UAS driver.

The reception of the OTc (Cmd) for the CIU, the OTc (Data-out) for the write data, and the ITc (Stat, SIU) for the SIU, all with the same tag, enables the UAS driver to complete the command sequence by returning status to the layer above the UAS driver. If the OTc (Data-out) has not been received by the UAS driver, then an error has occurred and the UAS driver should terminate the OTc (Data-out) before returning status.

5.3.5 Bi-directional data transfer

In the Bi-directional Data Transfer presented by figure 5 the write data portion of the Transfer starts and ends before the read data portion. UASP imposes no ordering requirements on the relative start or end times of the read or write data portions of a Bi-directional Transfer.

Figure 5 — Example High Speed UASP Bi-directional Data Transfer

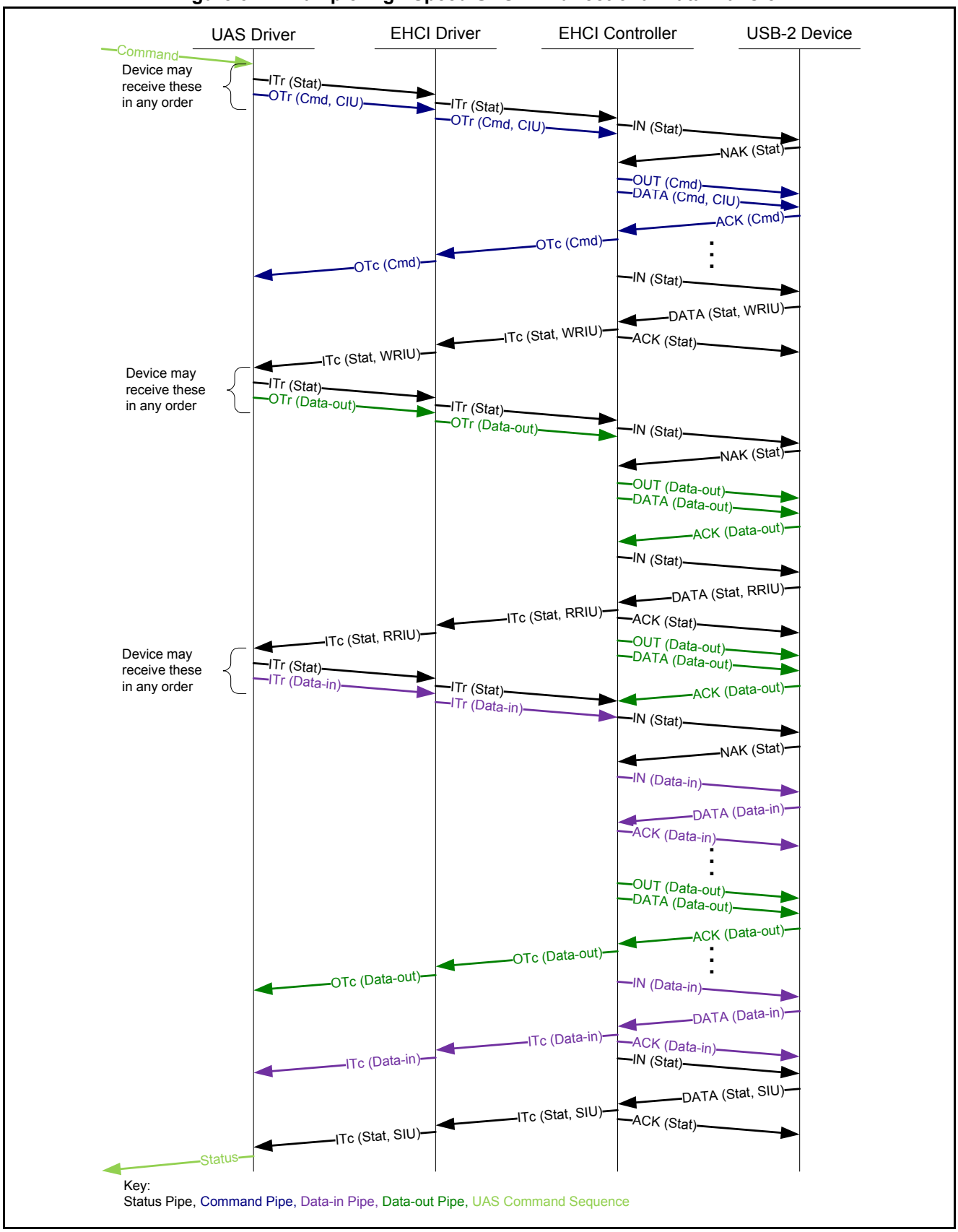


Figure 5 illustrates the execution of a High-speed UASP Bi-directional Data Transfer. A Bi-directional Transfer Command from the UAS Driver causes the EHCI driver to initiate an ITr (Stat) which allows the EHCI controller

to receive the WRIU or the RRIU for the command, and an OTr (Cmd, CIU) to transmit the CIU to the device.

The ITr (Stat) causes the EHCI Controller to generate an IN (Stat). In response to the OTr (Cmd, CIU) the EHCI contoller sends the CIU to the device by transmitting an OUT (Cmd), followed by a DATA (Cmd, CIU).

In response to the IN (Stat) packet the device returns NAK (Stat) packets until the device has status to return. When the device receives the DATA (Cmd, CIU), it responds with an ACK (Cmd), acknowledging the successful reception of the CIU. The CIU contains the command tag which is used to match the SIU with the CIU when the command completes.

The reception of the ACK (Cmd) for the CIU completes the OTr (Cmd) for the EHCI controller, which generates an OTc (Cmd) to the EHCI driver. The EHCI driver then returns OTc (Cmd) to the UAS driver indicating that the command has successfully been delivered to the device.

When the device is ready to receive the data for a command, it responds to the IN (Stat) polling with a DATA (Stat, WRIU).

The EHCI controller responds to the DATA (Stat, WRIU) packet with an ACK (Stat), acknowledging the successful reception of the WRIU. The reception of the DATA (Stat, WRIU) packet completes the ITr (Stat) for the EHCI controller, which generates an ITc (Stat, WRIU) to the EHCI driver. The EHCI driver returns an ITc (Stat, WRIU) to the UAS driver indicating that there is status available. The UAS driver evaluates the WRIU and checks the tag to determine which command the device is requesting data for. The UAS driver then issues an ITr (Stat) to the EHCI driver which enables the next status return by the device. The UAS driver then issues an OTr (Data-out) using the buffer indicated by the tag in the WRIU.

In response to the ITr (Stat), the EHCI driver issues an ITr (Stat) which schedules a data transfer on the status pipe. In response to the ITr (Stat), the EHCI controller issues IN (Stat) packets to the device and the device returns NAK (Stat) packets until the device has status to return.

In response to the OTr (Data-out), the EHCI driver schedules a data transfer on the Data-out pipe.

The EHCI controller begins receiving data from the device by transmitting an OUT (Data-out) followed by a DATA (Data-out)

When the device receives a DATA (Data-out), the device responds with an ACK (Data-out) which indicates that the write data has been received.

The steps described in the previous three paragraphs repeat until the Data-out transfer is complete.

While the data is being transferred from the host to the device, the EHCI controller responds to a DATA (Stat, RRIU) packet with an ACK (Stat), acknowledging the successful reception of an RRIU. The reception of the DATA (Stat, RRIU) packet completes the ITr (Stat) for the EHCI controller, which generates an ITc (Stat, RRIU) to the EHCI driver. The EHCI driver returns an ITc (Stat, RRIU) to the UAS driver indicating that there is status available. The UAS driver evaluates the RRIU and checks the tag to determine which command the device is responding to with data. The UAS driver then issues an ITr (Stat) to the EHCI driver which enables the next status return by the device. The UAS driver then issues an ITr (Data-in) using the buffer indicated by the tag in the RRIU.

In response to the ITr (Stat), the EHCI driver issues an ITr (Stat) which schedules a data transfer on the status pipe. In response to the ITr (Stat), the EHCI controller issues IN (Stat) packets to the device and the device returns NAK (Stat) packets until the device has status to return.

In response to the ITr (Data-in), the EHCI driver schedules a data transfer on the Data-in pipe.

The EHCI controller begins receiving data from the device by transmitting an IN (Data-in).

When the device receives an IN (Data-in), the device responds with a DATA (Data-in) which contains read data.

When the EHCI controller receives the DATA (Data-in) packet on the Data-in pipe, it responds with an ACK (Data-in), acknowledging the successful reception of the read data.

The steps described in the previous three paragraphs repeat until the Data-in transfer is complete.

The reception of the last ACK (Data-out) by the EHCI controller (e.g., the host buffer is empty or a short packet is sent) completes the OTr (Data-out) for the EHCI controller, which generates an OTc (Data-out) to the EHCI driver. This in-turn generates an OTc (Data-out) completion to the UAS driver.

The reception of the last Data (Data-in) by the EHCI controller (e.g., the host buffer is full or a short packet is received) completes the ITr (Data-in) for the EHCI controller, which generates an ITc (Data-in) to the EHCI driver. This in-turn generates an ITc (Data-in) completion to the UAS driver.

When the command is complete, the device responds to the IN (Stat) with a DATA (Stat, SIU). The SIU contains the tag of the CIU.

The EHCI controller responds to the DATA (Stat, SIU) with an ACK (Stat) packet, acknowledging the successful reception of the SIU. The reception of the DATA (Stat, SIU) packet completes the ITr (Stat) for the EHCI controller, which generates an ITc (Stat, SIU) to the EHCI driver. This in turn causes the EHCI driver to generate an ITc (Stat, SIU) to the UAS driver.

The reception of the OTc (Cmd) for the CIU, the OTc (Data-out) for the write data, the ITc (Data-in) for the read data and the ITc (Stat) for the SIU, all with the same tag, enables the UAS driver to complete the command sequence by returning status to the layer above the UAS driver. If the ITc (Data-in) or the OTc (Data-out) has not been received by the UAS driver, then an error has occurred and the UAS driver should terminate the outstanding ITc (Data-in) or OTc (Data-out) before returning status.

5.4 SuperSpeed UASP

5.4.1 Overview

This section describes the SuperSpeed packet exchanges required to support the UASP. This section assumes a familiarity with the [USB3] Streams protocol. Refer to sections 4.4.6.4 and 8.12.4.1 of [USB3] for more information on the Streams protocol.

The Cmd pipe does not use [USB3] streams. The Data-in, Data-out, and Stat pipes require a stream ID for each transaction. The stream ID is either Prime (as shown in the figure notation) or the stream ID provided by the appropriate CIU.

5.4.2 Non-data transfer

Non-data transfers are the simplest UASP operation. Non-data transfers only utilize the Command and Status pipes. A Command IU (CIU) is transmitted by the host on the Command pipe, and the device responds with a Response IU (RIU) on the Status pipe.

Figure 6 — Example SuperSpeed Non-Data Transfer

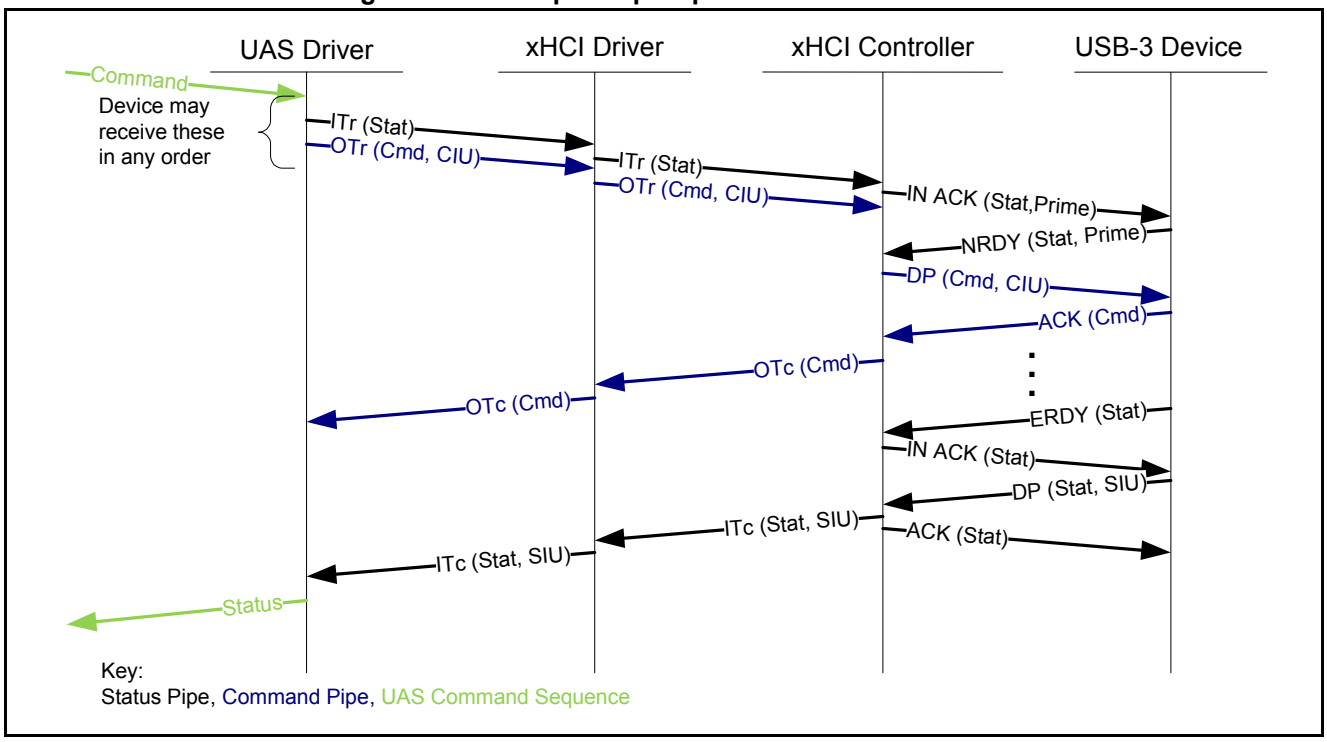


Figure 6 illustrates the execution of a SuperSpeed UASP Non-Data Transfer. A Non-data Transfer Command received by the UAS Driver causes the UAS driver to send an ITr (Stat) to receive the SIU for the command, and an OTr (Cmd, CIU) to transmit the CIU to the xHCI driver. Since the Status pipe is a Streams pipe, the ITr (Stat) includes the Tag associated with the transfer.

The ITr (Stat) causes the xHCI controller to generate an IN ACK (Stat, Prime). This packet transitions the Streams state machine of the Status pipe to the Prime Pipe state, notifying the device that the host is ready to receive an SIU. In response to the OTr (Cmd, CIU) the xHCI controller transmits a DP (Cmd, CIU).

In response to the IN ACK (Stat, Prime), the device returns an NRDY (Stat, Prime). This action transitions the Status pipe back to the Idle state. When the device receives the DP (Cmd, CIU), it responds with an ACK (Cmd), acknowledging the successful reception of the CIU.

The reception of the ACK (Cmd) for the CIU completes the OTr (Cmd, CIU) for the xHCI controller, which generates an OTc (Cmd) to the xHCI driver. The reception of the OTc (Cmd) to by the xHCI driver generates an OTc (Cmd) to the UAS driver.

When the command is complete, the device generates an ERDY (Stat) with the SID set to the Tag value provided by the CIU Tag field. This transitions the pipe to the Start Stream state and informs the xHCI controller that the device is ready to transfer the SIU on the Status pipe.

In response to the ERDY (Stat), the xHCI controller sends an IN ACK (Stat) to the device with the SID set to the ERDY Tag value.

In response to the IN ACK (Stat), the device returns a DP (Stat, SIU) which contains the SID set to value returned in the IN ACK (Stat).

The xHCI controller responds to the DP (Stat, SIU) with an ACK (Stat), acknowledging the successful reception of the SIU. The reception of the DP (Stat, SIU) completes the ITr (Stat) for the xHCI controller, which generates an ITc (Stat, SIU) to the xHCI driver. When the xHCI driver receives the ITc (Stat, SIU) from the xHCI controller, the xHCI driver generates an ITc (Stat, SIU) to the UAS driver.

The reception of the OTc (Cmd) for the CIU and the ITc (Stat) for the SIU, both with the same tag, causes the UAS driver to complete the command sequence by returning status to the layer above the UAS driver.

5.4.3 Data-in transfer

Figure 7 — Example SuperSpeed UASP Data-in transfer

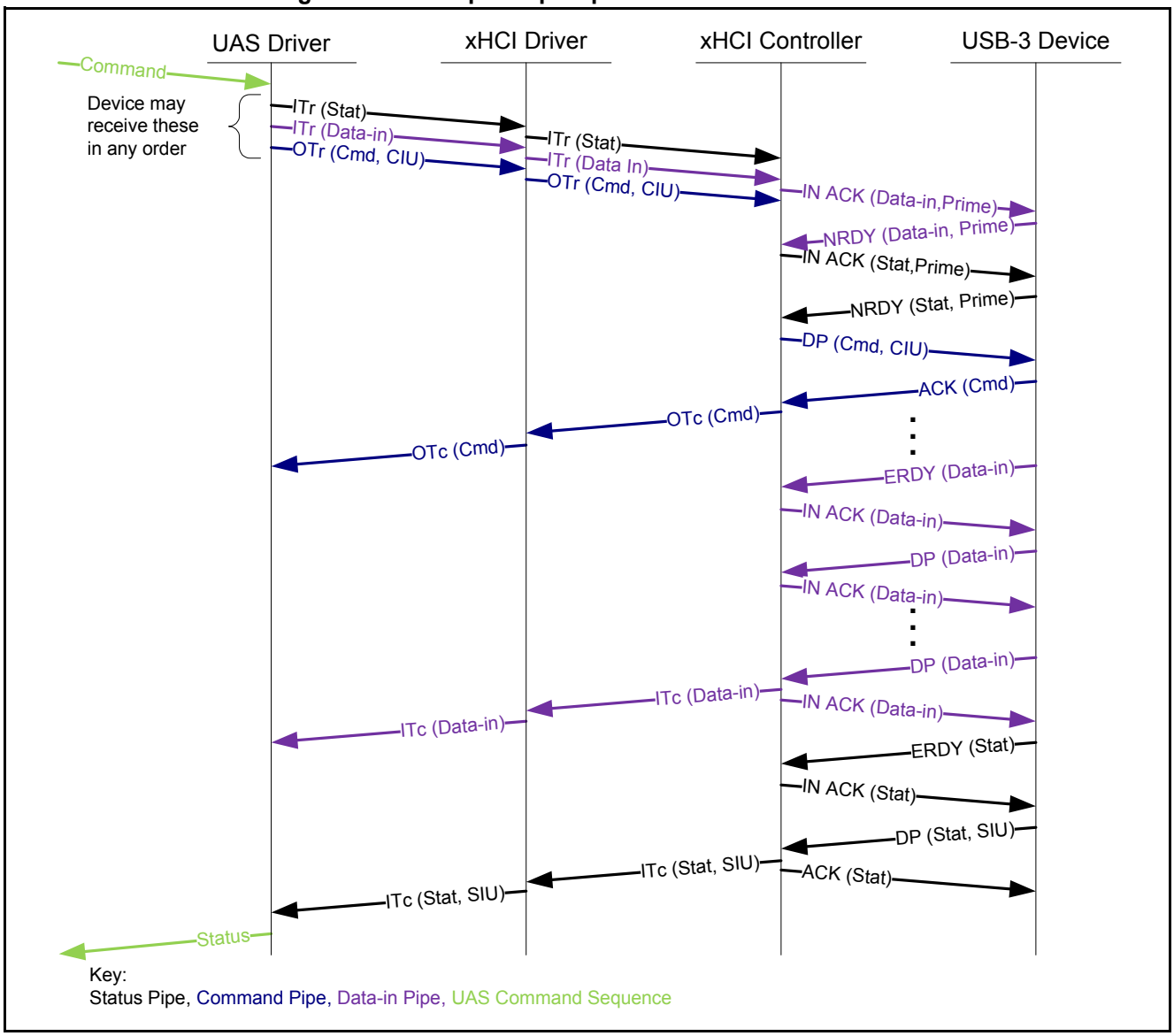


Figure 7 illustrates the execution of a SuperSpeed UASP Data In Transfer. A Data-in Transfer Command received by the UAS Driver causes the UAS driver to send an ITr (Stat) to receive the SIU for the command, an ITr (Data-in) to receive the data, and an OTr (Cmd, CIU) to transmit the CIU to the xHCI driver. Since the Status pipe is a Streams pipe, the ITr (Stat) includes the Tag associated with the transfer.

The ITr (Stat) causes the xHCI controller to generate an IN ACK (Stat, Prime). This packet transitions the Streams state machine of the Status pipe to the Prime Pipe state, notifying the device that the host is ready to receive an SIU. The ITr (Data-in) causes the xHCI controller to generate an IN ACK (Data-in, Prime). This packet transitions the Streams state machine of the Data-in pipe to the Prime Pipe state, notifying the device that the host is ready to receive data. In response to the OTr (Cmd, CIU) the xHCI controller transmits a DP (Cmd, CIU).

In response to the IN ACK (Stat, Prime), the device returns an NRDY (Stat, Prime). This action transitions the Status pipe back to the Idle state. In response to the IN ACK (Data-in, Prime), the device returns an NRDY (Data-in, Prime). This action transitions the Data-in pipe back to the Idle state. When the device receives the DP (Cmd, CIU), it responds with an ACK (Cmd), acknowledging the successful reception of the CIU.

The reception of the ACK (Cmd) for the CIU completes the OTr (Cmd, CIU) for the xHCI controller, which generates an OTc (Cmd) to the xHCI driver. The reception of the OTc (Cmd) to by the xHCI driver generates an OTc (Cmd) to the UAS driver.

When the device is ready to receive the data for a command, the device generates an ERDY (Data-in) with the SID set to the Tag value provided by the CIU Tag field. This transitions the pipe to the Start Stream state and informs the xHCI controller that the device is ready to transfer data.

In response to the ERDY (Data-in) the xHCI controller sends an IN ACK (Data-in) to the device with the SID set to the ERDY Tag value. This action transitions the data Stream state machine from the Start Stream to the Move Data state, and begins the data transfer.

In response to the IN ACK (Data-in), the device returns a DP (Data-in) that contains the read data and the SID set to the SID value from the IN ACK (Data-in).

The xHCI controller responds to the DP (Data-in) with an ACK (Data-in), acknowledging the successful reception of the read data.

The steps described in the previous two paragraphs repeat until the data transfer is complete.

When the command is complete, the device generates an ERDY (Stat) with the SID set to the Tag value provided by the CIU Tag field. This transitions the pipe to the Start Stream state and informs the xHCI controller that the device is ready to transfer the SIU on the Status pipe.

In response to the ERDY (Stat), the xHCI controller sends an IN ACK (Stat) to the device with the SID set to the ERDY Tag value.

In response to the IN ACK (Stat), the device returns a DP (Stat, SIU) which contains the SIU and the SID set to value returned in the IN ACK (Stat).

The xHCI controller responds to the DP (Stat, SIU) with an ACK (Stat), acknowledging the successful reception of the SIU. The reception of the DP (Stat, SIU) completes the ITr (Stat, SIU) for the xHCI controller, which generates an ITc (Stat, SIU) to the xHCI driver. When the xHCI driver receives the ITc (Stat, SIU) from the xHCI controller, the xHCI driver generates an ITc (Stat, SIU) to the UAS driver.

The reception of the OTc (Cmd) for the CIU, the ITc (Data-in) for the data, and the ITc (Stat) for the SIU, all with the same tag, causes the UAS driver to complete the command sequence by returning status to the layer above the UAS driver. If the ITc (Data-in) has not been received by the UAS driver, then an error has occurred and the UAS driver should terminate the ITc (Data-in) before returning status.

5.4.4 Data-out transfer

Figure 8 — Example SuperSpeed UASP Data-out transfer

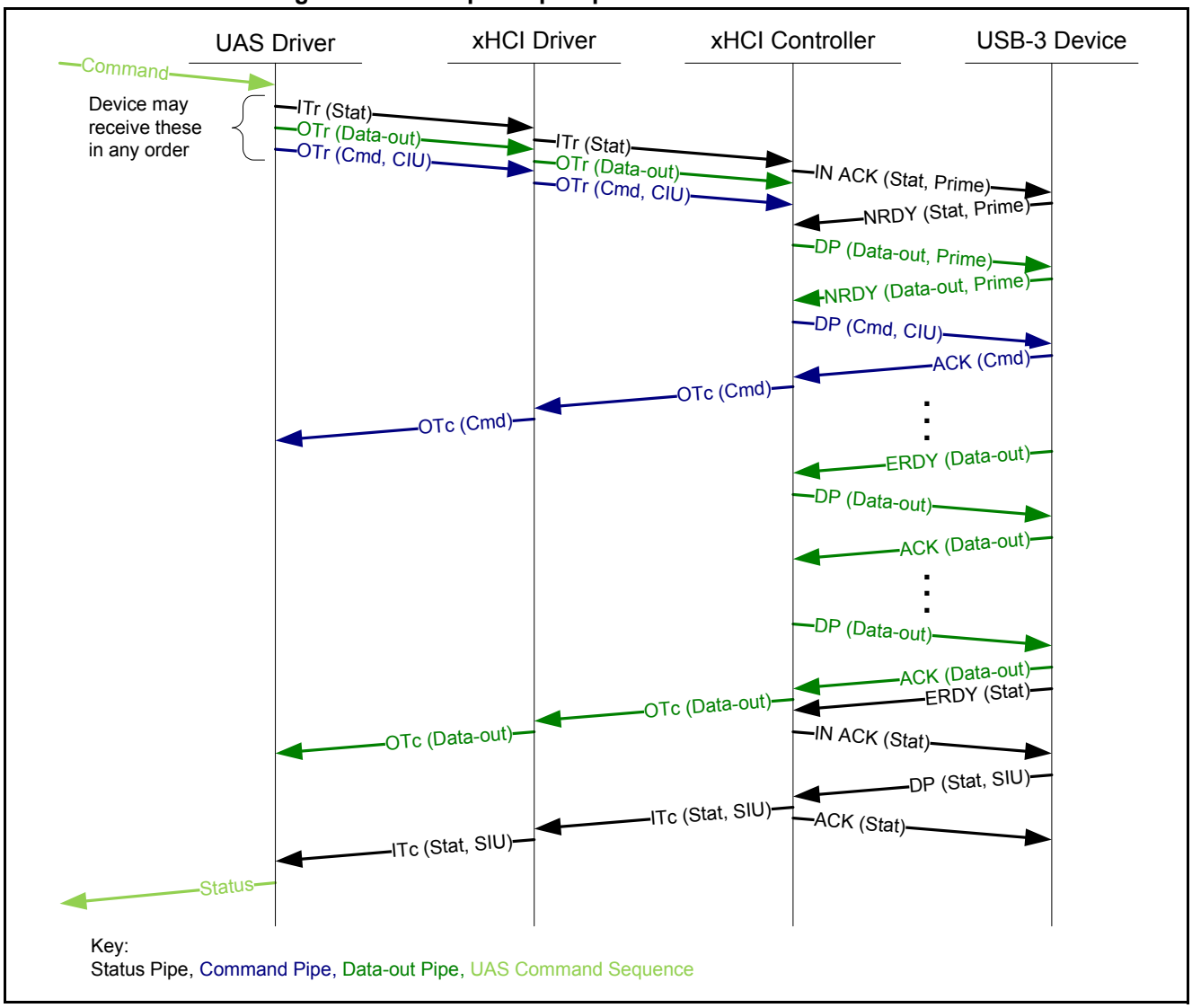


Figure 8 illustrates the execution of a SuperSpeed UASP Data Out Transfer. A Data-out Transfer Command received by the UAS Driver causes the UAS driver to send an ITr (Stat) to receive the SIU for the command, an OTr (Data-out) to send the data, and an OTr (Cmd, CIU) to transmit the CIU to the xHCI driver. Since the Status pipe is a Streams pipe, the ITr (Stat) includes the Tag associated with the transfer.

The ITr (Stat) causes the xHCI controller to generate an IN ACK (Stat, Prime). This packet transitions the Streams state machine of the Status pipe to the Prime Pipe state, notifying the device that the host is ready to receive an SIU. In response to the OTr (Data-out) the xHCI controller generates a DP (Data-out, Prime). This packet transitions the Streams state machine of the Data-out pope to the Prime pipe state, notifying the device that the host is ready to send data. In response to the OTr (Cmd, CIU) the xHCI controller transmits a DP (Cmd, CIU).

In response to the IN ACK (Stat, Prime), the device returns an NRDY (Stat, Prime). This action transitions the Status pipe back to the Idle state. In response to the DP (Data-out, Prime), the device returns an NRDY (Data-out, Prime). This action transitions the Data-out pipe back to the Idle state. When the device receives the DP (Cmd, CIU), it responds with an ACK (Cmd), acknowledging the successful reception of the CIU.

The reception of the ACK (Cmd) for the CIU completes the OTr (Cmd, CIU) for the xHCI controller, which generates an OTc (Cmd) to the xHCI driver. The reception of the OTc (Cmd) to by the xHCI driver generates an OTc (Cmd) to the UAS driver.

When the device is ready to receive the data for a command, the device generates an ERDY (Data-out) with the SID set to the Tag value provided by the CIU Tag field. This transitions the pipe to the Start Stream state and informs the xHCI controller that the device is ready to receive data.

In response to the ERDY (Data-out) the xHCI controller sends a DP (Data-out) to the device with the SID set to the ERDY Tag value. This action transitions the data Stream state machine from the Start Stream to the Move Data state, and begins the data transfer.

In response to the DP (Data-out), the device returns an ACK (Data-out) acknowledging the successful reception of the data. The ACK (Data-out) contains a SID set to the SID value from the DP (Data-out).

The xHCI controller responds to the ACK (Data-in) with a DP (Data-in).

The steps described in the previous two paragraphs repeat until the data transfer is complete.

When the data transfer is complete, the device generates an ERDY (Stat) with the SID set to the Tag value provided by the CIU Tag field. This transitions the pipe to the Start Stream state and informs the xHCI controller that the device is ready to transfer the SIU on the Status pipe.

In response to the ERDY (Stat), the xHCI controller sends an IN ACK (Stat) to the device with the SID set to the ERDY Tag value.

In response to the IN ACK (Stat), the device returns a DP (Stat, SIU) which contains the SIU and the SID set to value returned in the IN ACK (Stat).

The xHCI controller responds to the DP (Stat, SIU) with an ACK (Stat), acknowledging the successful reception of the SIU. The reception of the DP (Stat, SIU) completes the ITr (Stat) for the xHCI controller, which generates an ITc (Stat, SIU) to the xHCI driver. When the xHCI driver receives the ITc (Stat, SIU) from the xHCI controller, the xHCI driver generates an ITc (Stat, SIU) to the UAS driver.

The reception of the OTc (Cmd) for the CIU, the OTc (Data-out) for the data, and the ITc (Stat) for the SIU, all with the same tag, causes the UAS driver to complete the command sequence by returning status to the layer above the UAS driver. If the OTc (Data-out) has not been received by the UAS driver, then an error has occurred and the UAS driver should terminate the OTc (Data-out) before returning status.

5.4.5 Bi-directional data transfer

In the Bi-directional Data Transfer presented by figure 9 the write data portion of the Transfer starts and ends before the read data portion. UASP imposes no ordering requirements on the relative start or end times of the read or write data portions of a Bi-directional Transfer.

Figure 9 — Example SuperSpeed UASP Bi-directional Data Transfer

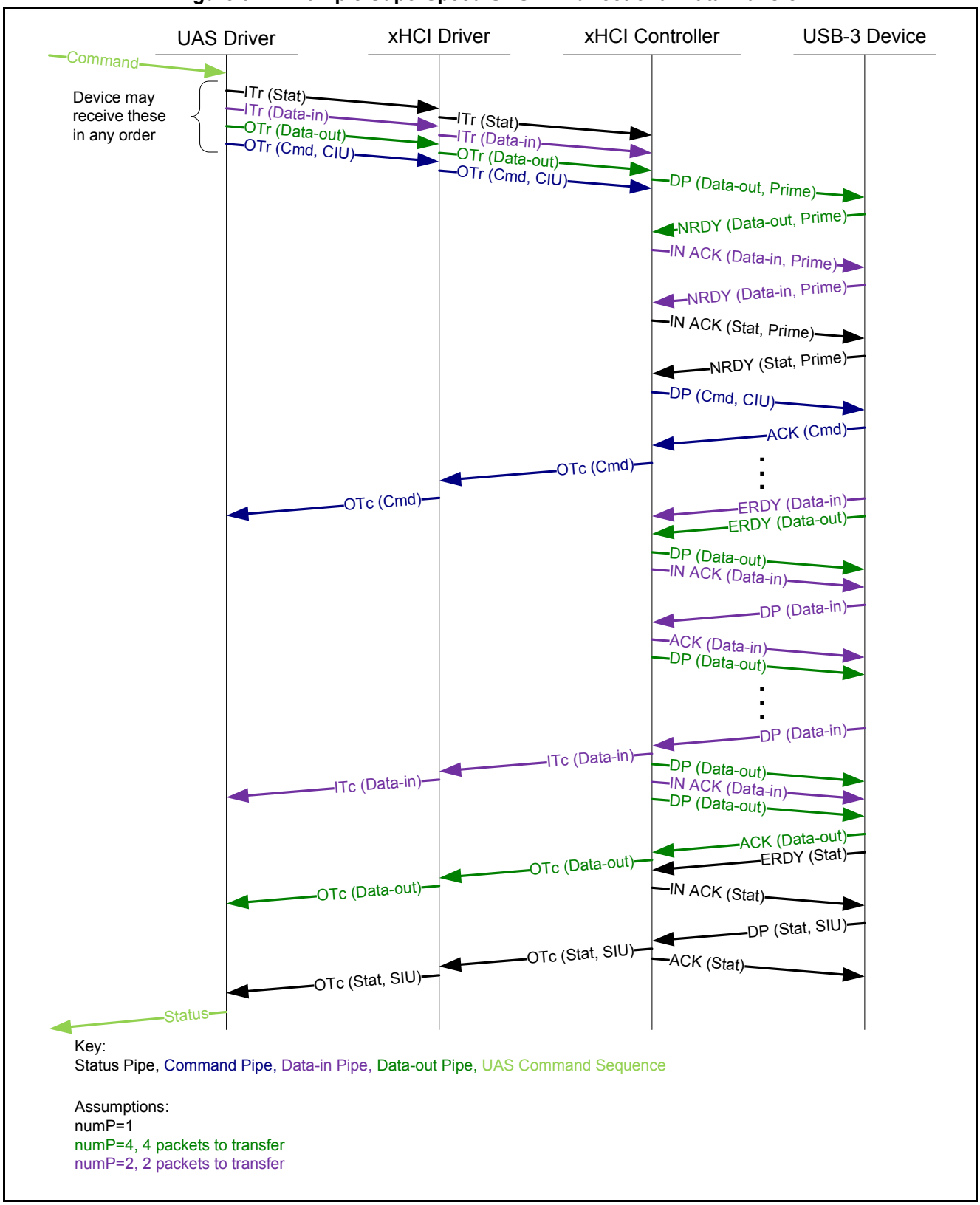


Figure 9 illustrates the execution of a SuperSpeed UASP Bi-directional Data Transfer. A bi-directional data transfer Command received by the UAS Driver causes the UAS driver to send an ITr (Stat) to receive the SIU for the command, an ITr (Data-in) to receive the data, an OTr (Data-out) to receive the data, and an OTr (Cmd, CIU) to transmit the CIU to the xHCI driver. Since the Status pipe is a Streams pipe, the ITr (Stat) includes the Tag associated with the transfer.

The ITr (Stat) causes the xHCI controller to generate an IN ACK (Stat, Prime). This packet transitions the Streams state machine of the Status pipe to the Prime Pipe state, notifying the device that the host is ready to receive an SIU. The ITr (Data-in) causes the xHCI controller to generate an IN ACK (Data-in, Prime). This packet transitions the Streams state machine of the Data-in pipe to the Prime Pipe state, notifying the device that the host is ready to receive data. The OTr (Data-out) causes the xHCI controller to generate a DP (Data-out, Prime). This packet transitions the Streams state machine of the Data-out pipe to the Prime Pipe state, notifying the device that the host is ready to send data. In response to the OTr (Cmd, CIU) the xHCI controller transmits a DP (Cmd, CIU).

In response to the IN ACK (Stat, Prime), the device returns an NRDY (Stat, Prime). This action transitions the Status pipe back to the Idle state. In response to the IN ACK (Data-in, Prime), the device returns an NRDY (Data-in, Prime). This action transitions the Data-in pipe back to the Idle state. In response to the DP (Data-out, Prime), the device returns an NRDY (Data-out, Prime). This action transitions the Data-out pipe back to the Idle state. When the device receives the DP (Cmd, CIU), it responds with an ACK (Cmd), acknowledging the successful reception of the CIU.

The reception of the ACK (Cmd) for the CIU completes the OTr (Cmd) for the xHCI controller, which generates an OTc (Cmd) to the xHCI driver. The reception of the OTc (Cmd) to by the xHCI driver generates an OTc (Cmd) to the UAS driver.

When the device is ready to receive the data for a command, the device generates an ERDY (Data-in) with the SID set to the Tag value provided by the CIU Tag field. This transitions the pipe to the Start Stream state and informs the xHCI controller that the device is ready to transfer data.

In response to the ERDY (Data-in) the xHCI controller sends an IN ACK (Data-in) to the device with the SID set to the ERDY Tag value. This action transitions the data Stream state machine from the Start Stream to the Move Data state, and begins the data transfer.

In response to the IN ACK (Data-in), the device returns a DP (Data-in) that contains the read data and the SID set to the SID value from the IN ACK (Data-in).

The xHCI controller responds to the DP (Data-in) with an ACK (Data-in), acknowledging the successful reception of the read data.

The steps described in the previous two paragraphs repeat until the data transfer is complete.

While the data is being transferred from the device to the host, the device becomes ready to receive the data for the same command, the device generates an ERDY (Data-out) with the SID set to the Tag value provided by the CIU Tag field. This transitions the data-out pipe to the Start Stream state and informs the xHCI controller that the device is ready to receive data.

In response to the ERDY (Data-out) the xHCI controller sends a DP (Data-out) to the device with the SID set to the ERDY Tag value. This action transitions the data Stream state machine from the Start Stream to the Move Data state, and begins the data transfer.

In response to the DP (Data-out), the device returns an ACK (Data-out) acknowledging the successful reception of the data. The ACK (Data-out) contains a SID set to the SID value from the DP (Data-out).

The xHCI controller responds to the ACK (Data-in) with a DP (Data-in).

The steps described in the previous two paragraphs repeat until the data transfer is complete.

When the command is complete, the device generates an ERDY (Stat) with the SID set to the Tag value provided by the CIU Tag field. This transitions the pipe to the Start Stream state and informs the xHCI controller that the device is ready to transfer the SIU on the Status pipe.

In response to the ERDY (Stat), the xHCI controller sends an IN ACK (Stat) to the device with the SID set to the ERDY Tag value.

In response to the IN ACK (Stat), the device returns a DP (Stat, SIU) which contains the SID set to value returned in the IN ACK (Stat).

The xHCI controller responds to the DP (Stat, SIU) with an ACK (Stat), acknowledging the successful reception of the SIU. The reception of the DP (Stat, SIU) completes the ITr (Stat) for the xHCI controller, which generates

an ITc (Stat, SIU) to the xHCI driver. When the xHCI driver receives the ITc (Stat, SIU) from the xHCI controller, the xHCI driver generates an ITc (Stat, SIU) to the UAS driver.

The reception of the OTc (Cmd) for the CIU, the ITc (Data-in) for the read data, the OTc (Data-out) for the write data, and the ITc (Stat) for the SIU, all with the same tag, causes the UAS driver to complete the command sequence by returning status to the layer above the UAS driver. If the ITc (Data-in) or the OTc (Data-out) have not been received by the UAS driver, then an error has occurred and the UAS driver should terminate the outstanding ITc (Data-in) or OTc (Data-out) before returning status.

Annex A
(Informative)

Backward Compatibility

A.1 Overview

For [USB2] backward compatibility, the device shall present [BOT] as alternate interface zero (primary) and [UAS] as alternate interface one (secondary). A device which does not need backward compatibility with [BOT] shall present [UAS] as alternate interface zero. In [USB2] systems, the [BOT] driver or an associated filter driver may need to issue a SET INTERFACE request for alternate interface one and then allow the [UAS] driver to load.