Universal Serial Bus
Still Image Capture
Device Definition

Revision 1.0

July 11, 2000

Revision History

Rev	Date	Filename	Comments
1.0	July 11, 2000	usb_still_img10.pdf	Approvedby the DWG

Please send comments via electronic mail to:
timothy.whitcher@kodak.com

INTELLECTUAL PROPERTY DISCLAIMER

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE.

A LICENSE IS HEREBY GRANTED TO REPRODUCE AND DISTRIBUTE THIS SPECIFICATION FOR INTERNAL USE ONLY. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY OTHER INTELLECTUAL PROPERTY RIGHTS IS GRANTED OR INTENDED HEREBY.

AUTHORS OF THIS SPECIFICATION DISCLAIM ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF PROPRIETARY RIGHTS, RELATING TO IMPLEMENTATION OF INFORMATION IN THIS SPECIFICATION. AUTHORS OF THIS SPECIFICATION ALSO DO NOT WARRANT OR REPRESENT THAT SUCH IMPLEMENTATION(S) WILL NOT INFRINGE SUCH RIGHTS.

All product names are trademarks, registered trademarks, or servicemarks of their respective owners.

Contributors

Anderson, Blair	Ajilon
Armstrong, Frank	Eastman Kodak Company
Coppola, Stephen	Eastman Kodak Company
Fontani, Paolo	Hewlett-Packard Company
Hong, Fang	Seiko-Epson
Hsieh, William	Microsoft Corporation
Lawrence, David	Smart Technology Enablers, Inc.
Melville, John	Eastman Kodak Company
Myers, Paul	Questra Corporation
Parsons, Dave	Microsoft Corporation
Whitcher, Timothy	Eastman Kodak Company

Table of Contents

1 Introduction

1.1 Purpose

This document specifies the behavior of USB Still Image Capture Devices. The document was designed with digital still cameras in mind, but it may be applicable to other simular devices.

1.2 Scope

This document is intended to provide enough information for the following:

To allow software developers to create a device driver capable of connecting any compatible USB Still Image Capture Device to host-based image capture software.

1.3 Related Documents

Compaq, Intel, Microsoft, NEC, Universal Serial Bus Specification, Revision 1.1, September 23, 1998, http://www.usb.org/developers/data/usbspec.zip

PIMA 15740 Photography – Electronic still picture imaging – Picture Transfer Protocol, (PTP) for Digital Still Photography Devices, May 1, 2000, http://www.pima.net/standards/it10/IT10_POW.htm

1.4 Terms and Abbreviations

Table 1.4-1: Terms and Abbreviations

Term	Description
Configuration	A collection of one or more interfaces that may be selected on a USB device
Descriptor	Data structure used to describe a USB device capability or characteristic
Device	A USB peripheral
Driver	Host software that connects other drivers, DLLs or applications to USBDI.
Endpoint	Source or sink of data on a USB device
HCD	Acronym for Host Controller Driver, the Driver used to manage a host controller
HCDI	Acronym for HCD Interface, the programming interface used by USBD to interact with HCD
Host	A computer system where a Host Controller is installed
Host Controller	Hardware that connects a Host to USB
Host Software	Generic term for a collection of drivers, DLLs and/or applications that provide operating system support for a Device
IHV	Acronym for Independent Hardware Vendor
Interface	Collection of zero or more endpoints that present functionality to a host
OHCI	Acronym for Open Host Controller Interface, a hardware register specification defined by Compaq and Microsoft for a Host Controller
UHCI	Acronym for Universal Host Controller Interface, a hardware register specification defined by Intel for a Host Controller
USB	Acronym for Universal Serial Bus, a bus used to connect devices to a host
USBD	Acronym for Universal Serial Bus Driver, the Driver used to manage and use Devices among multiple Device Drivers
USBDI	Acronym for USBD Interface, the USBD programming interface

2 Management Overview

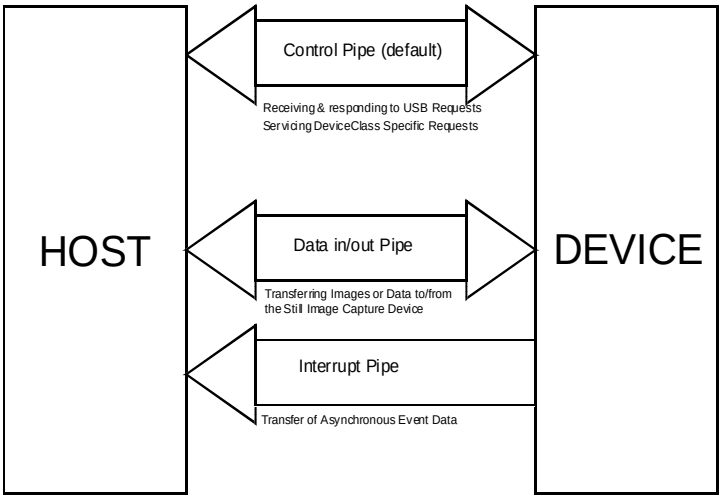
This document specifies the interface and behavior of a USB Still Image Capture Device from the perspective of the USB host. The device uses a bulk pipe for the transfer of image data.

The features and commands supported by a Still Image Capture Device are derived from the PIMA 15740 Standard, which addresses the requirements for interconnecting digital photography devices.

USB Still Image Capture Devices also use the bulk pipe to adjust device controls. Specific PIMA 15740 operations, the GetDevicePropertyDescriptor, GetDevicePropertyValue, and SetDevicePropertyValue, are used to manage the controls and mode setting of the device.

When an asynchronous event occurs in the device, such as a low battery indication or the removal of a memory card, the asynchronous event is reported over an interrupt pipe.

Figure 2.0-1 Device Configuration



3 Assumptions and Constraints

3.1 Compliance

This section describes assumptions and constraints related to compatibility.

3.1.1 Specification

The device shall be compliant with the USB Specification.

3.1.2 Device Framework

The device shall support standard and device-specific requests as described in the USB Specification.

3.1.3 PIMA 15740

The devices shall be compliant with the PIMA 15740 Standard Photography – Electronic Still Picture Imaging – Requirements for interconnecting digital photography devices.

3.2 Functional Overview

3.2.1 Image Data

Image data from the device is delivered to the host system through a bulk pipe. The format of the data is identified by an ObjectFormat data item in the PIMA 15740 ObjectInfo Dataset.

3.2.2 Device Controls and Status

As defined in PIMA 15740, the GetDevicePropDesc describes device controls, and the GetDevicePropValue and SetDeviceProp operations manage operational modes.

4 Device Characteristics

4.1 Configuration

The device configuration shall support at least one interface for image and data transfer. The device class code, device sub-class code, and the device protocol code in the Device Descriptor shall all be set to zero.

4.2 Interface

As noted above, the Still Image Capture Device shall support at least one interface for image and data transfer. As with all USB devices, the default endpoint shall be included in all interfaces by implication.

The Still Image Capture Device Interface Descriptor shall use a class code of Image (0x06), a sub-class code of Still Image Capture Device (0x01) and a protocol code of (0x01) to identify that the device uses protocol defined in this class specification.

4.3 Endpoints

The device shall contain at least four endpoints: default, Data-In, Data-Out, and an Interrupt endpoint.

4.3.1 Default

The default endpoint shall use control transfers as defined in the USB Specification. The default endpoint shall be used to send standard, class and vendor-specific requests to the device, an interface or an endpoint. The endpoint number must be zero (0x00).

4.3.2 Data-In

The Data-In endpoint shall be used to receive image and non-image data (such as a script) from the device intended for delivery to an imaging application on the host. The Data-In endpoint shall use bulk transfers. The endpoint number may be any value between one (1) and fifteen (15) that is not used by another endpoint on the device. The direction shall be IN. The maximum packet size is implementation specific and may vary for different alternate settings.

4.3.3 Data-Out

The Data-Out endpoint shall be used to send image and non-image data from the host to the device. The Data-Out endpoint shall use bulk transfers. The endpoint number may be any value between (1) and fifteen (15) that is not used by another endpoint on the device. The direction shall be OUT. The maximum packet size is implementation specific and may vary for different alternate settings.

4.3.4 Interrupt

The Interrupt endpoint associated with the Still Image Interface shall be used to send event data to the host from the device. The Interrupt endpoint shall use interrupt transfers. The endpoint number may be any value between one (1) and fifteen (15) that is not used by another endpoint on the device. The direction shall be IN. The maximum packet size is implementation specific.

4.4 Data Characteristics

The USB still image capture device may support one or more image data formats. The data formats are identified in the PIMA 15740 DeviceInfo Dataset.

4.5 PIMA 15740 Event Handling

USB Suspend and Resume signaling will affect the way a Still Image Capture Device that supports PIMA 15740 will handle PIMA 15740 Events. Likewise the handling of events will be affected if the device supports the Remote Wakeup feature and whether or not the Remote Wakeup feature is disabled.

PIMA Events shall be handled as described by the cases that follow:

Case 1. USB not suspended, Normal PIMA 15740 Event handling

1. PIMA Events are reported to the host via an interrupt where the data format is described in Clause 7.3.1 of this document.

Case 2. USB Suspended, the Remote Wakeup feature is enabled

1. Device signals remote wakeup upon PIMA 15740 Event detection
2. Device issues PIMA Events when bus signaling has resumed.

Case 3. USB Suspended, the Remote Wakeup feature is disabled, and no PIMA 15740 Events occur

1. When host or upstream hubs resume, the device continues, the PIMA 15470 session is undisturbed.

Case 4. USB Suspended, the Remote Wakeup feature is disabled, and PIMA 15740 events do occur

1. When host or upstream hubs resume, the device then posts the PIMA 15740 "UnreportedStatus" Event. The PIMA 15740 session remains open.
2. The host (Initiator) must examine the device. This includes checking the DeviceInfo dataset, checking object handles and ObjectInfo datasets, and checking the device status. The host (Initiator) may optionally close the session and restart.

The host may disable the remote wakeup feature at any time. However if a session is open, the awkward "UnreportedStatus" Event might occur.

This approach does not require the device to queue events, avoiding memory depth problems. After a suspension there will be at most one event.

Note: USB devices that follow this device class definition are PIMA 15740 single session devices. This class definition does not support multsession PIMA 15740 devices.

5 Device Requests

5.1 Standard Requests

The Device shall support the standard USB device requests as described below. The device shall return STALL if any unrecognized or unsupported standard request is received.

5.1.1 Clear Feature

The Device shall return STALL for any unrecognized or unsupported Clear Feature request.

5.1.2 Get Configuration

The Device shall support the Get Configuration request. The Device shall return zero if the device is unconfigured or the bConfiguration value as defined in the Configuration Descriptor.

5.1.3 Get Descriptor

The Device shall support Get Descriptor requests for standard descriptors (Device, Configuration and String). The Device may support Get Descriptor requests for Class or Vendor-specific descriptors. The Device shall return STALL if a Get Descriptor request is made for an unrecognized or unsupported descriptor.

5.1.4 Get Interface

The Device shall support a Get Interface request for Interface 0 when configured by returning an Alternate Setting of zero. The Device shall return STALL for a Get Interface request for any other Interface or any Get Interface request before the Device is configured.

5.1.5 Get Status

The Device shall support a Get Status request directed at the device, Interface 0 or any defined endpoint (default, Data-In, or Data-Out). The Device shall return STALL if a Get Status request is received for Interface 0 or any defined Endpoint before the Device is configured. The Device shall return STALL if a Get Status request is received for any unrecognized or unsupported recipient.

5.1.6 Set Address

The Device shall support a Set Address request to change the Device Address from the default address (zero) to a unique address. The Device may return STALL if any subsequent Set Address request is received to change the Device Address from a non-zero value to any value (including zero).

5.1.7 Set Configuration

The Device shall support the Set Configuration request to set the Device Configuration to zero (unconfigured) or the bConfiguration value defined in a Configuration Descriptor. The Device shall return STALL if a Set Configuration request is received with any other value.

5.1.8 Set Descriptor

The Device may support Set Descriptor requests for any defined Descriptor (Device, Configuration, Interface, Endpoint, String, Class or Vendor-Specific). The Device shall return STALL if a Descriptor may not be updated, is unrecognized, or is unsupported.

5.1.9 Set Feature

The Device shall return STALL for any unrecognized or unsupported Set Feature request.

5.1.10 Set Interface

When configured, the Device shall support a Set Interface request to Interface 0 for defined Alternate Settings.

5.1.11 Synch Frame

The Device shall return STALL for any Synch Frame request.

5.2 Class-Specific Requests

This Device Definition defines these class-specific requests. The device shall return STALL if an unrecognized or unsupported device-specific request is received.

5.2.1 Cancel Request

The Still Image Capture device shall accept the Cancel Request from the host, which is a control write sequence to the device's control endpoint. The data stage transfers to the device information that identifies the transaction over the Bulk Pipe that was cancelled by the host.

5.2.1.1 Cancel Set Up

The host is responsible for establishing the values passed in the fields listed in Table 5.2-1. The setup data packet has eight bytes.

Table 5.2-1 Format of Setup Data for the Cancel Request

Offset	Field	Size	Value	Description
0	bmRequestType	1	bitmap	00100001 Host-to-Device, Class-Specific, Recipient-Interface
1	bRequest	1	code	Cancel_Request (0x64, for this request)
2	wValue	2	value	value equal to zero.
4	wIndex	2	value	value equal to zero.
6	wLength	2	count	Value = 0x0006

5.2.1.2 Format of Cancel Data

The data stage of the Cancel Request has the following format.

Table 5.2-2 Format of Cancel Request Data

Offset	Field	Size	Value	Description
0	Cancellation Code	2	code	Value=0x4001, identifier for cancellation
2	TransactionID	4	number	An unsigned 32-bit field containing the PIMA 15740 TransactionID

5.2.2 Get Extended Event Data

The Still Image Capture device may accept the Get Extended Event Data Request from the host, which is a control, read sequence from the device's control endpoint. The data stage transfers to the host extended information regarding an asynchronous event or vendor condition.

5.2.2.1 Get Event Set Up

Table 5.2-3 defines the 8-byte set up data for the Get_Extended_Event_Data request.

Table 5.2-3 Format of Setup Data to retrieve the Extended Event Data

Offset	Field	Size	Value	Description
0	bmRequestType	1	bitmap	10100001 Device-to-Host, Class-Specific, Recipient-Interface
1	bRequest	1	code	Get_Extended_Event_Data (0x65, code for this request)
2	wValue	2	value	value equal to zero
4	wIndex	2	value	value equal to zero
6	wLength	2	count	Size of the host buffer allocated for the Extended Event Data

5.2.2.2 Format of Extended Event Data

The data stage of the Get Event Data Request has the following format.

Table 5.2-4 Data Format of Get Extended Event Data Request

Offset	Field	Size	Value	Description
0	Event Code	2	code	The PIMA15740 Event Code or Vendor Code
2	TransactionID	4	number	An unsigned 32-bit field containing the PIMA15740 TransactionID. This field is 0x00000000 if a TransactionID does not apply to this event.
6	Number of Parameters	2	number	This field contains a number that indicates the number of event parameters associated with the event code
8	Size of Parameter 1	2	number	This field contains a number that indicates the size in bytes of the corresponding event parameter.
10	Parameter 1	??	value	This field contains the actual event parameter. The format and meaning of the parameter is described in the description of the event.
??	Size of Parameter N	2	number	This field contains a number that indicates the size in bytes of the corresponding event parameter.
??	Parameter N	??	value	This field contains the actual event parameter. The format and meaning of the parameter is described in the description of the event.

5.2.3 Device Reset Request

The Still Image Capture device shall accept the Device Reset Request from the host, which is a non-data control sequence to the device's control endpoint. The Device Reset Request is used by the host to return the Still Image Capture Device to the Idle state after the Bulk-pipe has stalled. The request may also be used to clear any vendor specified suspend conditions.

5.2.3.1 Device Reset Set Up

The host is responsible for establishing the values passed in the fields listed in Table 5.2-5. The setup data packet has eight bytes.

Table 5.2-5 Format of Setup Data for the Device Reset Request

Offset	Field	Size	Value	Description
0	bmRequestType	1	bitmap	00100001 Host-to-Device, Class-Specific, Recipient-Interface
1	bRequest	1	code	Device_Reset_Request (0x66, for this request)
2	wValue	2	value	value equal to zero.
4	wIndex	2	value	value equal to zero.
6	wLength	2	count	Value of 0x0000, there is no data associated with this request

5.2.4 Get Device Status Request

The Still Image Capture device shall accept the Get Device Status Request from the host, which is a control, read sequence from the device's control endpoint. The data stage transfers to the host information regarding the status or protocol state of the

device. This request is used by the host to retrieve information needed to clear halted endpoints that result from a device initiated data transfer cancellation.

5.2.4.1 Get Device Status Set Up

Table 5.2-6 defines the 8-byte set up data for the Get Device Status Request.

Table 5.2-6 Format of Setup Data to retrieve the Extended Event Data

Offset	Field	Size	Value	Description
0	bmRequestType	1	bitmap	10100001 Device-to-Host, Class-Specific, Recipient-Interface
1	bRequest	1	code	Get_Device_Status (0x67, code for this request)
2	wValue	2	value	value equal to zero
4	wIndex	2	value	value equal to zero
6	wLength	2	count	Size of the host buffer allocated for the Extended Event Data

5.2.4.2 Format of Device Status Data

The data stage of the Get Event Data Request has the following format.

Table 5.2-7 Data Format of Get Device Status Request

Offset	Field	Size	Value	Description
0	wLength	2	number	This field specifies the total length of the status data
2	Code	2	code	The PIMA 15740 Response Code or Vendor Code: 0x2001 Status OK 0x2019 Device Busy 0x201F Transaction Cancelled
4	Parameter 1	??	value	The format and meaning of the parameter depends on the Status Code. For device initiated cancels this parameter contains an endpoint number.
??	Parameter N	??	value	The format and meaning of the parameter depends on the Status Code. For device initiated cancels this parameter contains an endpoint number.

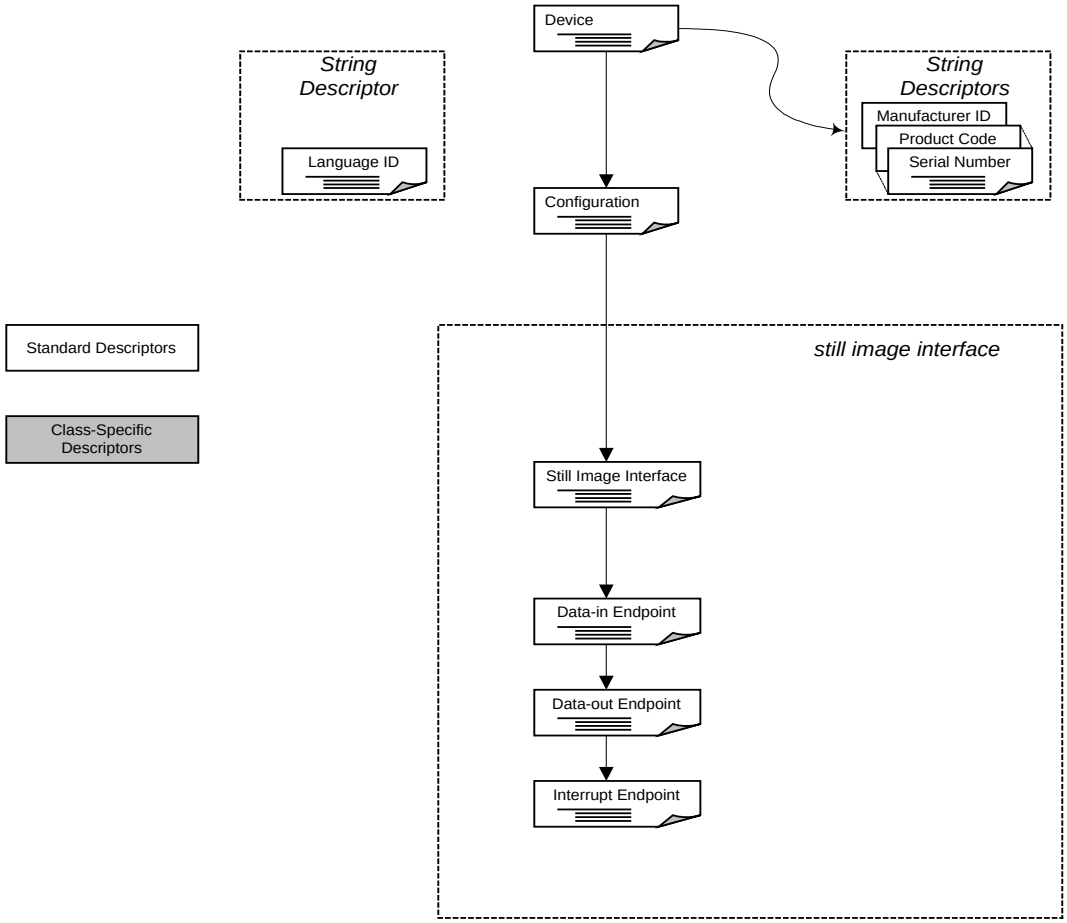
5.3 Vendor-Specific Requests

The Device may support vendor-specific requests. The device shall return STALL if an unrecognized or unsupported device-specific request is received.

6 Descriptors

The following figure shows the relationship among the descriptors in the Still Image Capture Device Class.

Figure 6.0-1 Descriptor Tree



6.1 Standard Descriptors

The Device shall support the standard USB descriptors as described below. The device shall return STALL if a request is received for any unrecognized or unsupported standard descriptor.

6.1.1 Device

The device shall return a Device Descriptor with the following values:

Table 6.1-1 Device Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x12	The length of this descriptor
1	bDescriptorType	1	0x01	Device Descriptor Type
2	bcdUSB	2	0x110	USB Specification Release Number
4	bDeviceClass	1	0x00	Class information may be found at the interface level
5	bDeviceSubClass	1	0x00	bDeviceSubClass is zero
6	bDeviceProtocol	1	0x00	bDeviceProtocol is zero
7	bMaxPacketSize0	1	number	Implementation specific, may be set to 8, 16, 32 or 64
8	idVendor	2	ID	Vendor ID assigned to IHV by USB-IF
10	idProduct	2	ID	Product ID assigned by IHV
12	bcdDevice	2	BCD	Device release number in BCD assigned by IHV to this release of model
14	iManufacturer	1	index	Index of string descriptor describing IHV. If set to zero (0), there is no man ufacturer string.
15	iProduct	1	index	Index of string describing this product. If set to zero (0), there is no product string.
16	iSerialNumber	1	index	Index of string descriptor with this spe cific device's serial number. If set to zero (0), this device does not have a serial number.
17	bNumConfigurations	1	number	Device has this number of configura tions.

6.1.2 Configuration Descriptor

The device shall return one or more Configuration Descriptors and other configuration related descriptors as described below:

Table 6.1-2 Configuration Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	The length of this descriptor
1	bDescriptorType	1	0x02	Configuration Descriptor Type
2	wTotalLength	2	number	Total length of data returned for this configuration. Includes this descriptor and all of the descriptors that follow (interface, endpoint and vendor spe cific, if present).
4	bNumInterfaces	1	number	This configuration has <#> interfaces
5	bConfigurationVa lue	1	number	Implementation specific value used as an argument to Set Configuration to select this configuration
6	iConfiguration	1	index	Index of string describing this configu ration. If set to zero (0), there is no configuration string.
7	bmAttributes	1	bitmap	Configuration characteristics as defined by USB Specification
8	MaxPower	1	mA	Maximum power draw from the bus by this device when this configuration is selected.

6.1.3 Still Image Interface Descriptor

Table 6.1-3 Still Image Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	The length of this descriptor
1	bDescriptorType	1	0x04	Interface Descriptor Type
2	bInterfaceNumber	1	number	Number of interface in configuration
3	bAlternateSetting	1	0x00	Default setting for this interface
4	bNumEndpoints	1	number	Number of endpoints in this interface, not including the default endpoint.
5	bInterfaceClass	1	0x06	Image interface
6	bInterfaceSubClass	1	0x01	Still Image Capture Device
7	bInterfaceProtocol	1	0x01	This field indicates the protocol sup ported by this device. PIMA 15740 compliant devices use Bulk-only protocol
8	iInterface	1	index	Index of string describing this inter face. If set to zero (0), there is no inter face string.

6.1.4 Data-In Endpoint Descriptor

Table 6.1-4 Data-In Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	7	The length of this descriptor
1	bDescriptorType	1	5	Endpoint Descriptor Type
2	bEndpointAddress	1	0x8?	Any endpoint number not used by another endpoint. The direction shall be set to IN. (The endpoint number may be the same as that of the Data-Out endpoint.)
3	bmAttributes	1	0x02	The endpoint uses bulk transfers
4	wMaxPacketSize	2	number	Maximum packet size used by this endpoint. This must be less than or equal to 64 bytes.
6	bInterval	1	number	Ignored

6.1.5 Data-Out Endpoint Descriptor

Table 6.1-5 Data-Out Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x07	The length of this descriptor
1	bDescriptorType	1	0x05	Endpoint Descriptor Type
2	bEndpointAddress	1	0x0?	Any endpoint number not used by another endpoint. The direction shall be set to OUT. (The endpoint number may be the same as that of the Data-In endpoint.)
3	bmAttributes	1	0x02	The endpoint uses bulk transfers
4	wMaxPacketSize	2	number	Maximum packet size used by this endpoint. This must be less than or equal to 64 bytes.
6	bInterval	1	number	Ignored

6.1.6 Interrupt Endpoint Descriptor

Table 6.1-6 Interrupt Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x07	The length of this descriptor
1	bDescriptorType	1	0x05	Endpoint Descriptor Type
2	bEndpointAddress	1	0x8?	Any endpoint number not used by another endpoint. The direction shall be set to IN.
3	bmAttributes		0x03	The endpoint uses interrupt transfers
4	wMaxPacketSize	2	number	Maximum packet size used by this endpoint. This must be less than or equal to 64 bytes.
6	bInterval	1	number	Interval for polling endpoint for data transfers

6.1.7 String Descriptors

The device may include strings describing the manufacturer, product, serial number, configuration and interface.

The Still Image Capture Device uses the following string descriptors to support these defined strings.

6.1.7.1 Manufacturer ID Code Descriptor

The string descriptor of the *iManufacturer* field of the Device Descriptor shall support up to 120 UNICODE characters, which describe the manufacturer's code name. Each character in the UNICODE string shall be alphanumeric and printable.

Table 6.1-7 Manufacturer ID Code Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0xF2	The length of this descriptor
1	bDescriptorType	1	3	String Descriptor Type
2	wString1	2	uchar	Man ID Code UNICODE character 1
4	wString2	2	uchar	Man ID Code UNICODE character 2
240	wString120	2	uchar	Man ID Code UNICODE character 120

6.1.7.2 Product ID Code Descriptor

The string descriptor of the *iProduct* field of the Device Descriptor shall support up to 122 UNICODE characters that describe the Model name.

Table 6.1-8: Product ID Code Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0xF2	The length of this descriptor
1	bDescriptorType	1	3	String Descriptor Type
2	wString1	2	uchar	Model UNICODE character 1
4	wString2	2	uchar	Model UNICODE character 2
122	wString120	2	uchar	Model UNICODE character 120

6.1.7.3 Serial Number Descriptor

The string descriptor of the *iSerialNumber* field of the Device Descriptor shall support up to 126 UNICODE characters, which encode the serial number in a vendor specific format. Each character in the in the UNICODE string shall be alphanumeric and printable.

Table 6.1-9 Serial Number Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0xFE	The length of this descriptor
1	bDescriptorType	1	3	String Descriptor Type
2	wString1	2	uchar	Serial Num UNICODE character 1
4	wString2	2	uchar	Serial Num UNICODE character 2
252	wString126	2	uchar	Serial Num UNICODE character 126

6.1.7.4 Language ID Descriptor

The string descriptor of Index=0, a Language ID Descriptor with a valid Language ID code, is mandatory for PIMA15740 compatible devices. String Descriptors are used to describe information required by PIMA 15740.

Table 6.1-10 Language ID Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0xFE	The length of this descriptor
1	bDescriptorType	1	3	String Descriptor Type
2	wLangID[0]	2	code	Language ID code 0
N+2	wLangID[N]	2	code	Language ID code N

6.1.7.5 Vendor Information Descriptor

The string descriptor of Index=*iVendorInformation* shall be used in Vendor Specific Descriptors to support PIMA15740 Vendor Dependent Information. A vendor may use up to 126 UNICODE characters in a vendor specific format. Typically, this field is used for firmware version information. Each of the UNICODE characters shall be alpha numeric and printable.

Table 6.1-11 Vendor Information Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0xFE	The length of this descriptor
1	bDescriptorType	1	3	String Descriptor Type
2	wString1	2	uchar	Vendor UNICODE character 1
4	wString2	2	uchar	Vendor UNICODE character 2
252	wString126	2	uchar	Vendor UNICODE character 126

6.2 Class-Specific Descriptors

There are no Class-Specific Descriptors defined for a Still Image Capture Device.

6.3 Vendor-Specific Descriptors

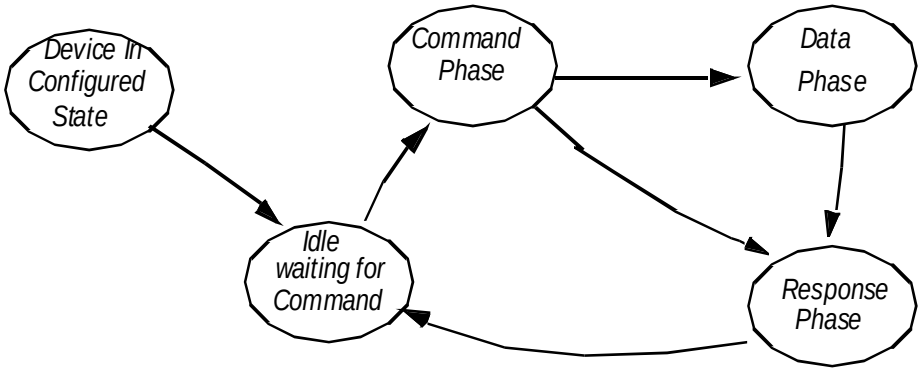
The device may support vendor-specific descriptors. The device shall return STALL if a request is received for an unrecognized or unsupported vendor-specific descriptor.

7 Still Image Capture Device Class-Specific Protocol

Still Image Capture Devices that are compatible to this annex support a Bulk-pipe interface used for image transfer or data transfer. This interface is used to provide transfer of coherent data objects such as image thumbnails, descriptors of the data objects, an image file, a code file, meta-data file, or any opaque data file. This interface is also used to manage stores and other items that affect data access in the device and it is used to manage the controls and modes of operation of the device. This contrasts with block data access provided by a mass storage interface.

A Still Image Capture Device that is compatible to PIMA 15740 adheres to a protocol model that involves three phases of operation execution. These are *Command*, *Data*, and *Response*. This three-phase operation execution model applies to the Bulk-Only Protocol defined in this Class specification.

Figure 7.0-1 Operation Phase State Diagram



In the *Command* phase the host transfers to the Still Image Capture Device a *Command Block* that defines the PIMA15740 operation that the host is requesting the device to perform. The specific contents of the *Command Block* correspond to an operation defined in the PIMA15740 specification. When a device is configured and idle, it is ready to receive a command. A device enters the *Command* phase when a *Command Block* is sent to the device. A device determines that a *Command Block* has been completely received when the device accepts from the host the number of bytes specified in the first four bytes of the *Command Block* that coincides with a short or null packet. A short packet or a NULL packet indicates the end of a *Command* phase. If the number of bytes specified in the first four bytes of the *Command Block* are an integral multiple of the wMaxPacketSize field of the Endpoint Descriptor the *Command* phase will end in a NULL packet. If the number of bytes transferred in the *Command* phase is less than that specified in the first four bytes of the *Command Block* then the device has received an invalid command and should STALL the Bulk-Pipe (refer to Clause 7.2).

After accepting and interpreting a command, a device optionally enters the *Data* phase. In the *Data* phase the data to be transferred is contained in a *Data Block.* The first four bytes of a *Data Block* describe the length in bytes of the data to be transferred. The

operation code in the *Command Block* determines if the operation requires data transfer. The operation code also determines the direction of data transfer (host to device - data out, or device to host - data in). The *Data* phase ends when the number of bytes transferred equals the number of bytes specified in the first four bytes of the *Data Block* that coincide with a short or a NULL packet. A short packet or a NULL packet indicates the end of the end of a *Data* phase. If the number of bytes specified in the first four bytes of the Data Block are an integral multiple of the wMaxPacketSize field of the Endpoint Descriptor the *Data* phase will end in a NULL packet.

If the number of bytes transferred in the *Data* phase is less than that specified in the first four bytes of the *Data Block* and the data receiver detects this condition before the initiation of the *Response* phase the data transfer may be cancelled (refer to Clause 7.2).

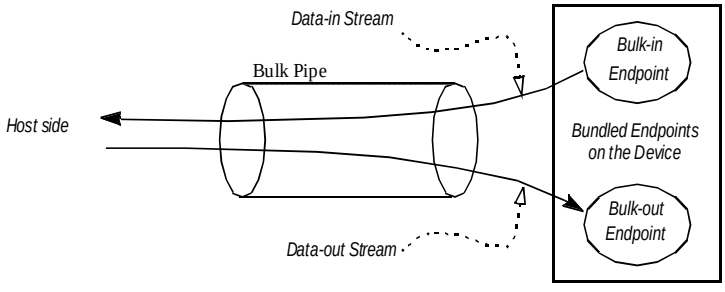
After the *Command* phase in operations without data transfer, or after the *Data* phase in operations with data transfer, the device enters the *Response* phase. The *Response* phase returns operation completion information to the host in a container called the *Response Block*. The first four bytes of a *Response Block* describe the length in bytes of the command completion data to be transferred.

The host determines that a *Response Block* has been completely received when the device sends a non-maximum length data packet. The maximum packet size is determined by the value set in the wMaxPacketSize field of the Endpoint Descriptor corresponding to the endpoint utilized in the status information transfer. Additionally, the host may also determine that a *Response Block* has been completely received when the device sends a NULL packet. This method will be used when the *Response Block* size in bytes is an integral number of maximum data packets.

Endpoint Utilization in Still Image Bulk-only Protocol

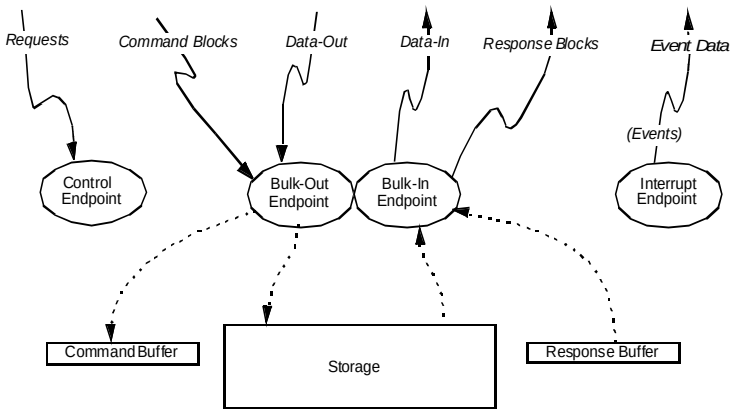
The Bulk-Only Protocol defined in this annex bundles together a Bulk-out Endpoint and a Bulk-in Endpoint into a single logical entity that services a thread of PIMA 15740 operations. This configuration may be abstracted to have a "Data-in Stream" that transfers bytes from the device to the host and a "Data-out Stream" that transfers bytes from the host to the device as shown in Figure D.7.0-2.

Figure 7.0-2 Bulk-Only Protocol Streams



The Data-out Stream transfers operations, operation parameters, and any data from the host to the device (*Command Blocks* and *Data(out) Blocks*). The Data-in Stream transfers operation responses and any data from the device to the host (*Data(in) Blocks* and *Response Blocks*). A short packet or a NULL packet sent by the data source indicates the end of the transfer's phase.

Figure 7.0-3 A USB Still Image Capture Device



7.1 Bulk-Pipe Containers

In a Still Image Capture Device the bulk pipe is used for the PIMA15740 operations that deal with file transfer and management. The bulk pipe has an input stream and an output stream connected to a common endpoint address. This common endpoint address is what leads to the abstraction of bundled endpoints. It is not necessary that a common endpoint address be used but it is recommended. The PIMA15740 information is encapsulated into the streams using containers. A container has four types; *Command Block*, *Data Block*, *Response Block, and Event Block*. The *Event Blocks* are not used on the Bulk-Pipe but rather the Interrupt Pipe for encapsulating asynchronous event data.

7.1.1 Generic Container Structure

The generic structure of the containers used by a PIMA 15740 compatible Still Image Capture Device follows.

Table 7.1-1 Generic Container Structure

Byte Offset	Length (Bytes)	Field Name	Description
0	4	Container Length	This field encodes as an unsigned integer the number of bytes in this container. A Still Image Capture Device uses this field to determine the size of the container.
4	2	Container Type	This field describes the type of the container: 0 *undefined* 1 *Command Block* 2 *Data Block* 3 *Response Block* 4 *Event Block* otherwise *reserved*
6	2	Code	This field contains the PIMA 15740 OperationCode, ResponseCode, or EventCode. The Data Block will use the OperationCode from the Command Block
8	4	TransactionID	This is a host generated number that associates all phases of an PIMA15740 operation.
12	??	Payload	The contents of this field depend on the operation and phase of the PIMA15740 operation.

Notes:

The OUT Stream can have *Command Blocks* and *Data Blocks*. The IN Stream can have *Data Blocks* and *Response Blocks*.

The Still Image Bulk-only Protocol, which is based on the PIMA15740 protocol, does not allow queuing of operations. Consequently when a *Command Block* that is optionally followed by a *Data Block* is sent by the host on the OUT Stream, a *Response Block* must be returned from the device on the IN Stream before the next *Command Block* can be set by the host to the device on the OUT Stream.

The data in the containers is in little endian format.

7.1.2 Command Block Payload Structure

The structure of the payload of a *Command Block* used by a Still Image Capture Device compatible with this annex follows.

Table 7.1-2 Command Block Payload Structure

Relative Byte Offset	Length (Bytes)	Field Name	Description
0	4	Parameter 1	This field contains an operation parameter. The format and meaning of the parameter is described in the operation description of each operation (PIMA 15740 Operation)..
??	4	Parameter N	This field contains an operation parameter. The format and meaning of the parameter is described in the operation description of each operation (PIMA 15740 Operation).

Notes:

Parameters are always 4 bytes in length. The number of parameters can be determined from the Container length. (Length – 12)/4

The *Command Block* payload does not have a transfer length parameter. This prevents the responding device from knowing a priori the amount of any data associated with an operation.

7.1.3 Data Block Payload Structure

The structure of the payload of a *Data Block* used by a Still Image Capture Device compatible with this annex is not defined. The actual structure of the data in a *Data* phase depends on the operation associated with the data.

7.1.4 Response Block Payload Structure

The structure of the payload of a *Response Block* used by a Still Image Capture Device compatible with this annex follows.

Table 7.1-3 Response Block Payload Structure

Relative Byte Offset	Length (Bytes)	Field Name	Description
0	4	Parameter 1	This field contains a response parameter. The format and meaning of the parameter is described in the response description of each operation (PIMA15740 Operation).
??	4	Parameter N	This field contains a response parameter. The format and meaning of the parameter is described in the response description of each operation (PIMA15740 Operation).

Notes:

Parameters are always 4 bytes in length. The number of parameters can be determined from the Container length. (Length – 12)/4

7.2 Still Image Bulk-only Protocol

This clause describes the implementation of Bulk-only protocol on a Still Image Capture Device.

Still Image Capture Devices defined in this annex support only a single thread of operation. The devices have one command buffer that processes commands sequentially. The bInterfaceProtocol field in the Still Image Interface Descriptor shall indicate Bulk-Only protocol.

7.2.1 Data Transfer Cancellation in Still Image Bulk-only Protocol

The information transfer over the Bulk-pipe can be cancelled at any time by either the host or the Still Image Capture Device. Normally only the *Data* phase is cancelled as the *Command* phase and the *Response* phase are short. An exception is when the device receives an invalid operation that will cause it to STALL the Bulk-Pipe. Consequently, a container may be only partially transferred by a stream or the data transfer may stop between containers. The Still Image Bulk-only Protocol enables detection of a cancelled information transfer condition allowing for recovery.

7.2.1.1 Rule Set

The cancellation data transfer over the Bulk-Pipe involves the use of two methods. One method is used by device-initiated cancels and the other method is used by host-initiated cancels.

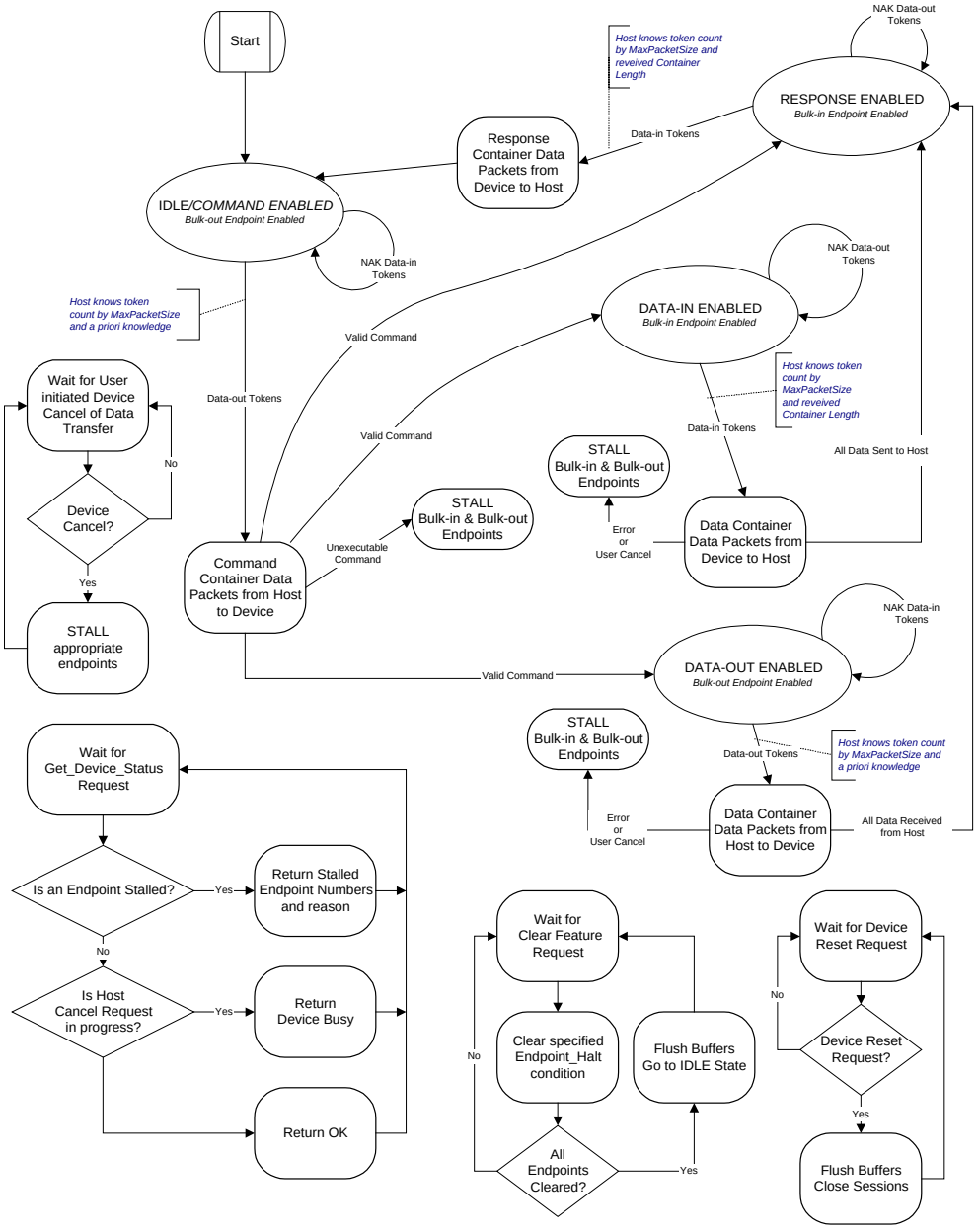
The Still Image Device shall cancel a data transfer by stalling the Bulk-Pipe endpoints. The host shall then determine the reason for the STALL (ENDPOINT_HALT condition) and the stalled endpoint numbers by issuing the class-specific Get_Device_Status Request (refer to Clause 5.2.4). The host shall then issue standard Clear_Feature Requests to clear the ENDPOINT_HALT condition on the affected endpoints. After the Clear_Feature Requests the device shall return OK status on subsequent Get_Device_Status Requests to indicate to the host that the device is ready to resume operations. The OK status corresponds to PIMA15740 Response Code 0x2001.

The host shall cancel a data transfer by no longer issuing tokens on the Bulk-Pipe and then issuing a class-specific Cancel Request to the device. The host shall then poll the device with Get_Device_Status Requests and when the device returns OK status commands may be resumed.

7.2.1.2 Device Behavior

The following flowchart describes the behavior of a device that implements Still Image Bulk-only Protocol.

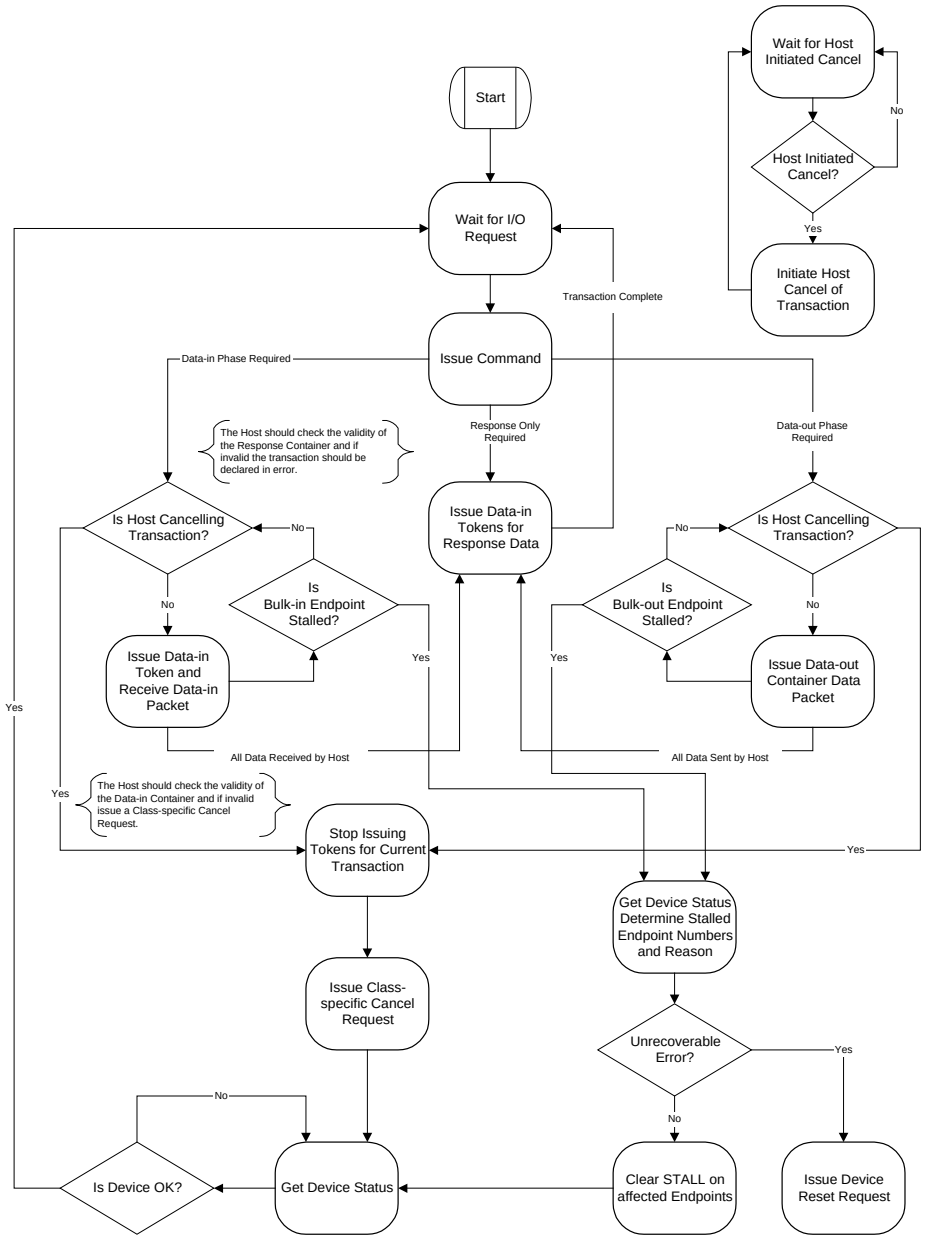
Figure 7.2-1 Still Image Protocol Device Behavior



7.2.1.3 Host Behavior

The following flowchart describes the behavior of a host that implements Still Image Bulk-only Protocol.

Figure 7.2-2 Still Image Protocol Host Behavior



7.3 Asynchronous Event Notification

A Still Image Capture Device that is compliant to PIMA 15740 shall provide a means to notify the host of the occurrence of certain events. Events such as the removal of a memory card from the Still Image Capture Device while actively connected to the host, is an example. A USB Still Image Capture Device uses the interrupt endpoint associated with the Still Image Interface for this purpose. The device returns to the host interrupt data with zero or more (up to three) parameters that identify the asynchronous event by PIMA15740 Event Code. For Asynchronous Events that require a large amount of data, the class-specific Get Extended Event Data Request shall be used.

The Get Extended Event Data Request is issued by the host when a Check Device Condition Asynchronous Event is received by via the Interrupt Pipe.

7.3.1 Asynchronous Event Interrupt Data Format

The device shall return PIMA 15740 interrupt data formatted as follows:

Table 7.3-1 Format of Asynchronous Event Interrupt Data

Offset	Field	Size (Bytes)	Value	Description
0	Interrupt Data Length	4	number	This field indicates the length in bytes of the interrupt data.
4	Container Type	2	0x0004	Container Type = Event
6	Event Code	2	code	The PIMA 15740 Event Code.
8	TransactionID	4	code	An unsigned 32-bit field containing the PIMA 15740 TransactionID. This field is 0x00000000 if a TransactionID does not apply to this event.
12	Event Parameter 1	4	variable	This field contains the 1st parameter associated with the event if needed.
16	Event Parameter 2	4	variable	This field contains the 2nd parameter associated with the event if needed.
20	Event Parameter 3	4	variable	This field contains the 3rd parameter associated with the event if needed.

Notes:

Event parameters are always 4 bytes in length. The number of parameters can be determined from the Interrupt Data Length. (Length – 12)/4

Annex A. Structure of the PIMA 15740 Datasets *[Normative]*

Four PIMA15740 datasets, namely the Deviceinfo Dataset, the ObjectInfo Dataset, the Device Property Describing Dataset, and the StorageInfo Dataset are transferred on the Bulk-pipe in data containers. All data items of these datasets are transferred together in one data container in one operation. Therefore the binary structure of these datasets needs to be defined. One should refer to the PIMA 15740 specification for the definitions of the fields identified in the structures. The definitions of generic types are as follows.

The PIMA 15740 specification identifies the following data types, which can correspond to a particular field within a dataset.

PIMA 15740 defines the following integer types.

Table A.0-1 Structure of PIMA15740 Integers

Type	Byte Size
INT8	1
UINT8	1
INT16	2
UINT16	2
INT32	4
UINT32	4
INT64	8
UINT64	8

All PIMA15740 Strings have the following structure.

Table A.0-2 Structure of PIMA15740 Strings

Field	Byte Size
Number of Characters	1
Characters	variable

Note: PIMA15740 uses two-byte UNICODE characters. All characters are packed little Endian. The strings are terminated with two null bytes (0x00 00) and the Number of Characters field contains a count that includes the two null bytes.

All PIMA 15740 Arrays have the following structure. When a byte size is context dependent, the byte size is determined by the element type which is determined by the PIMA 15740 Operation Code.

Table A.0-3 Structure of PIMA15740 Arrays

Field	Byte Size
Number of Array Elements	4
ArrayElement[0]	*context dependent*
ArrayElement[1]	*context dependent*
...	
ArrayElement[NumberElements-1]	*context dependent*

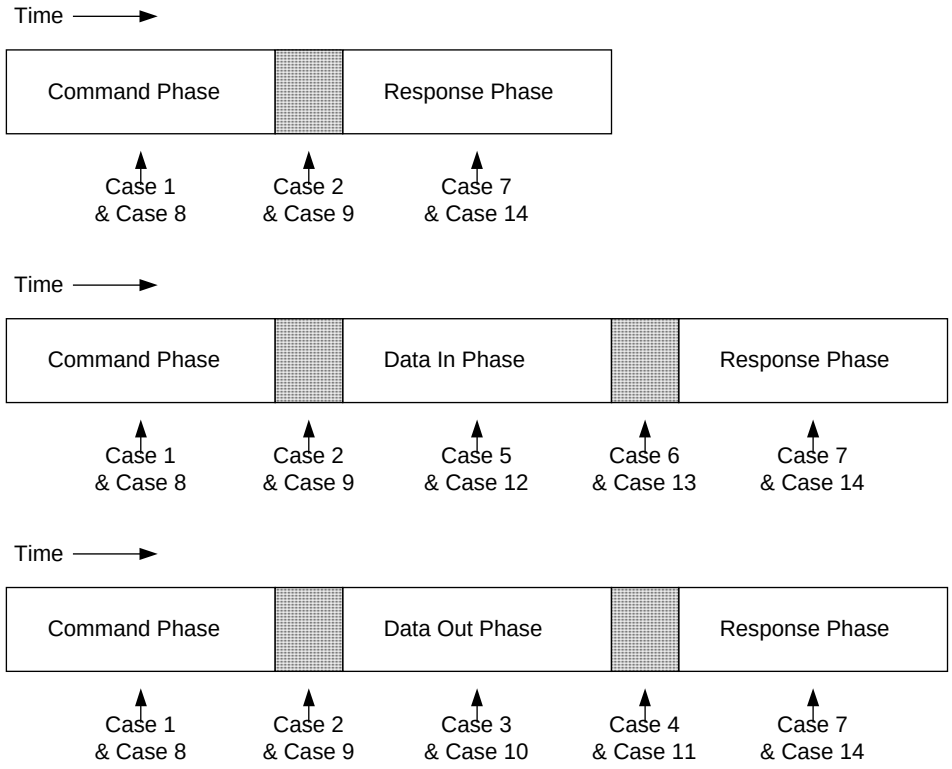
The PIMA 15740 specification does not define byte significance of the fields within a dataset. This is commonly known as "Endianness". Little Endian systems place the least significant byte of a multi-byte field first in a stream or at the lowest address in a memory buffer. Big Endian systems place the most significant byte of a multi-byte field first in a stream or at the lowest address of a memory buffer.

The Universal Serial Bus is a little Endian system. All fields are packed accordingly. The normative PIMA 15740 specification does specify the order of the various fields that comprise the DeviceInfo Dataset, the ObjectInfo Dataset, the Device Property Describing Dataset, and the StorageInfo Dataset.

Annex B. Still Image Bulk-only Protocol Cancellation Examples *[Informative]*

This annex presents specific cancellation examples in the Still Image Bulk-only Protocol. The different cases presented correspond to the indicated positions in the following figure that depicts the three phases of the Still Image Bulk-only Protocol.

Figure B-1 Cancellation Cases



Note: "just finished receiving" means "just received a short or NULL data packet" which terminates a Block.

Case 1: The HOST cancels when the DEVICE is receiving the Command Block, or

Case 2: The HOST cancels when the DEVICE has just finished receiving the Command Block, or

Case 3: The HOST cancels when the DEVICE is receiving the Data(out) Block, or

Case 4: The HOST cancels when the DEVICE has just finished receiving the Data(out) Block, or

Case 5: The HOST cancels when the HOST is receiving the Data(in) Block, or

Case 6: The HOST cancels when the HOST has just finished receiving the Data(in) Block.

Normally the Command Phase is not cancelled because it is short. If it is cancelled, then the following occurs:

1. The HOST stops issuing tokens.

2. The HOST sends to the DEVICE the class-specific Cancel Request.

3. The DEVICE enters a DEVICE_BUSY condition that is reported when the HOST issues a Get Device Status Request. DEVICE BUSY is reported by using the PIMA15740 Response Code 0.2019

4. The DEVICE clears its command buffer, goes to the *Idle/Command Enabled* state, and enters an OK condition.

5. The HOST polls the DEVICE with the Get Device Status Request. When the DEVICE returns OK status the cancel is complete.

Case 7: The HOST cancels when the HOST is receiving the Response Block.

1. The HOST does not cancel a Response Block as the Response Phase has the completion information of a completed transaction.

Case 8: The DEVICE cancels when the DEVICE is receiving the Command Block, or

Case 9: The DEVICE cancels when the DEVICE has just finished receiving the Command Block, or

Case 10: The DEVICE cancels when the DEVICE is receiving the Data(out) Block, or

Case 11: The DEVICE cancels when the DEVICE has just finished receiving the Data(out) Block, or

Case 12: The DEVICE cancels when the HOST is receiving the Data(in) Block, or

Case 13: The DEVICE cancels when the HOST has just finished receiving the Data(in) Block.

Since the Command Block is short and may only be one data packet in length the DEVICE usually does not cancel the Command Block. However, if an invalid command is received the DEVICE may cancel in Command phase.

1. The DEVICE places an Endpoint_Halt condition as indicated by the PIMA15740 Response Code, TransactionCancelled, 0x201F, on both the Bulk-in and Bulk-out endpoints. Consequently, any tokens issued by the HOST to these endpoints will return STALL.

2. The HOST issues a Get Device Status Request to determine the reason for the STALL and the endpoint numbers of the endpoints in an Endpoint_Halt condition.

3. The HOST may the issue Clear Feature Requests to clear the Endpoint_Halt condition on the endpoints returning STALL.

4. The DEVICE clears its command buffer, goes to the *Idle/Command Enabled* state, and enters an OK condition.

5. The HOST polls the DEVICE with the Get Device Status Request. When the DEVICE returns OK status the cancel is complete.

alternatively

3. The HOST sends to the DEVICE the class-specific Device Reset Request

4. The DEVICE clears its command buffer, closes all open sessions, and returns to the Configured State.

Case 14: The DEVICE cancels when the HOST is receiving the Response Block.

1. The Device shall not cancel a Response Block

Annex C. Architectual Framework *[Informative]*

The PIMA15740 Specification provides top-level requirements for digital Electronic Photography Devices. These requirements are in the form of commands, event handling requirements, device control, and information interchange protocol. The data transfer interface of the USB Still Image Capture Device Class addresses the PIMA15740 command and protocol requirements. The interrupt endpoints address the event handling requirements.

PIMA15740 enables a common device model to be developed for devices that support different interfaces. A device developer needs to adhere to the requirements of both PIMA15740 and the corresponding transport class specification

Figure C-1