USB3 Vision®

version 1.2

April, 2022

900 Victors Way, Suite 140 • Ann Arbor, Michigan 48108 USA • www.automate.org

USB3 Vision Licensing and Logo Usage

This standard was designed so that users of the technology can quickly and easily identify USB3 Vision compliant products that will interoperate and “plug and play” with each other. All commercial products developed using the USB3 Vision standard must license the standard and qualify for the right to use the IP of the standard, standard name and/or logo. To qualify, each product must have the proper paperwork submitted to the A3 and must pass USB3 Vision compliance testing. More information on licensing USB3 Vision can be found at

https://www.automate.org/a3-content/vision-standards

The USB3 Vision logo may be used only in conjunction with licensed products which have passed USB3 Vision compliance testing.

Table of Contents

1.0 Introduction

2.0 USB Related Requirements

3.0 Device Identification

4.0 Device Control

5.0 Streaming Data

6.0 Mechanical

7.0 Testing

8.0 Document History

List of Tables

List of Figures

USB3 Vision - v1.2 List of Requirements

1.0 Introduction

1.1 Purpose

USB3 Vision® is a communication interface for vision applications based on USB technology. It allows for easy interfacing between USB3 Vision transmitter devices and hosts using standard USB hardware.

The aim of this standard is to define the protocols used by USB3 Vision compliant products to communicate with each other. The standard builds upon USB 3.0 technology. As such, it does not cover the physical description of the transport media. Nor is any specific implementation of a specialized high-performance driver part of this standard.

USB has become the de facto standard for peripheral connections to PCs. Nearly every single PC in production at the time of this standard's initial development had at least two USB 2.0 ports. Additionally, USB 3.0 ports were beginning to show up on desktop and laptop PCs. USB 3.0 can theoretically transfer at 5 Gbps, which is a great amount of bandwidth for current vision applications.

Even though the name of this standard specifically refers to USB 3.x, USB3 Vision can be implemented using USB 2.0 as well as higher versions. Additionally, within a major version of the USB standard, newer versions replace previous ones completely while still retaining backwards compatibility. As such, references to a specific minor version of USB within this specification, such as USB 3.0, should be assumed to be referring to the latest version of the USB 3.x standard as documented in the reference list.

IMPORTANT NOTICE

The name USB3 Vision® and the distinctive logo design are registered trademarks owned by the A3 and may only be used under license for products registered with the A3. Any commercial use of this standard or the technologies therein, in whole or in part, requires a separate license obtained from the A3. Simple possession of the document does not convey any rights to use the standard. Information on the product registration and licensing program may be obtained from the A3.

The A3 shall not be responsible for identifying all patents for which a license may be required by this standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

1.2 Liability Disclaimer

This standard is provided "as is" and without any warranty of any kind, expressed or implied. Without limitation, there is no warranty of non-infringement, no warranty of merchantability, and no warranty of fitness for a particular purpose. All warranties are expressly disclaimed.

The user assumes the full risk of using this standard. In no event shall the A3, members of the technical committee, or their companies, be liable for any actual, direct, indirect, punitive, or consequential damages arising from such use, even if advised of the possibility of such damages.

1.3 Technical Committee

The USB3 Vision Standard Committee was formed in September 2011 to define an open transport platform based on USB 3.0 for high-performance vision applications. The committee is sponsored by the A3: https://www.visiononline.org/.

1.3.1 Version 1.1

Work on version 1.1 of the standard was initiated in 2015. The following individuals have participated in the technical meetings leading to version 1.1.

Bob McCurrach, Director of Standards Development

Eric Gross, Chair

Uwe Hagmaier, Vice Chair

Thomas Hopfner, Cable Subcommittee Chair

Lutz Koschorreck, Test Framework Subcommittee Chair

Takahiro Akahori
Bjoern Albrecht
Marino Barberio
Stefan Battmer
Jan Becvar
Ray Berst
Eric Bourbonnais
Eric Carey
Paul Carroll
Gri Ryong Choi
Karsten Christensen
Intial Deng
Thomas Detjen
Art Didyk
Fritz Dierks
Sascha Dorenbeck
Marco Draeger
Katie Ensign
JaeSeong Eom
James Falconer
Anna Fang
Werner Feith
Ron Folkeringa
Matt Gara
Fancois Gobeil
Tim Handschack
Sarah Jansen
Daniel Kaestner
Takayuki Kawaguchi
Martin Kersting
SeungHyun Kim
Kazunari Kudo
Max Larin
Sam Liu
Peter Lloyd
Sergey Loginovskikh
Thomas Lueck
Yoshi Masuda
Stephane Maurice
Roman Moie
Steve Mott
Thies Moeller
Marcel Naggatz
Damian Nesbitt
Quang Nhan Nguyen
Jinhee Oh
Kazuyuki Ohata
Neerav Patel
Ondrej Pindroch
Merlin Plock
Rick Ragnini
David Reaves
Lisa Ripke
Andreas Rittinger
Ryan Robe
Silvu Adrian Sasu
Matthias Schaffland
Takeichiro Sekiya
Matthew Simons
Manisha Singh
Rupert Stelz
Shankar Subramanian
Chendra Hadi Suryanto

Hirohiko Tada

Koji Takano

Hiroyuki Takano

Tom Tang

Timo Teifel

Yohei Tomoyasu

Sadafumi Torri

Akimi Tsukui

Tim Vlarr

Silvio Voitzsch

Masanori Yasuda

Lewis Zhang

Christoph Zierl

Juraj Zopp

The following technical member companies voted on version 1.1 of the standard:

A&B Software

Allied Vision

Alysium-Tech GmbH

Basler AG

Baumer Optronic GmbH

Cognex Corporation

Components Express, Inc.

EverPro Technologies Company Ltd.

FLIR Systems, Inc.

Hamamatsu Corporation

Hirakawa Hewtech Corp.

Hitachi Kokusai Electric America Ltd.

IDS Imaging Development Systems, Inc.

Intercon 1 - A Division of Nortech Systems, Inc.

JAI

Lucid Vision Labs

MathWorks

MATRIX VISION GmbH

Matrox Imaging

MVTec Software GmbH

National Instruments

NEC Platforms, Ltd.

Newnex Technology Corp.

Omron Sentech Co., Ltd.

PixeLINK

Pleora Technologies

Sensor To Image GmbH

STEMMER IMAGING

SVS-VISTEK GmbH

Teledyne DALSA

Toshiba Teli Corporation

Vieworks Co., Ltd.

XIMEA

1.3.2 Version 1.2

The following technical member companies voted on version 1.2 of the standard:

A&B Software

Allied Vision

Alysium-Tech GmbH

Basler AG

Baumer Optronic GmbH

China Daheng Group, Inc.

Cognex Corporation

Components Express, Inc.

Euresys

Everpro Technologies Company Ltd.

Hamamatsu Corporation

Hirakawa Hewtech Corp.

Hitachi Kokusai Electric America Ltd.

IDS Imaging Development Systems, Inc.

Intercon 1 - A Division of Nortech Systems, Inc.

JAI

Lucid Vision Labs

MACNICA, Inc.

MathWorks

MATRIX VISION GmbH

Matrox Imaging

Microview Science & Technology (TianJin) Co., Ltd.

MVTec Software GmbH

NI

NEC Platforms, Ltd.

Newnex Technology Corp.

Omron Sentech Co., Ltd.

PixeLINK

Pleora Technologies

Sensor To Image GmbH

STEMMER IMAGING AG

SVS-VISTEK GmbH

Teledyne DALSA

Teledyne FLIR

Toshiba Teli Corporation

Vieworks Co., Ltd.

XIMEA

1.4 Definitions and Acronyms

1.4.1 Definitions

Application	USB3 Vision control application software running on a host. Typically a software application running on a PC but can also be of another nature such as microcode running on a field programmable gate array (FPGA).
Bootstrap Register	A standard-defined register which allows an application to control the basic functions of a device.
Control Protocol	USB3 Vision Control Protocol (U3VCP) defining commands supported by USB3 Vision compliant devices.
Descriptor	Applications can query devices for their descriptors. See the USB standard documents for more information.
Device	USB3 Vision compliant controllable device. Typically a camera but can also be a non-streamable device such as a USB3 Vision strobe light controller.
Endpoint	A buffer in the device that is the source or sink of data.
Event Generator	Entity generating event messages according to the USB3 Vision standard.
Event Receiver	Entity receiving and capable of interpreting event messages according to the USB3 Vision standard.
Host	Entity that acts as a USB host, this is most often going to be a PC or laptop, but could also be a receiving device. It controls the device and receives the frames.
Interconnect	Cabling and infrastructure like repeaters and hubs between a host and a device.
Interface	Describes functions or features that are implemented in a device.
KB	Kilobytes or 1024 bytes.
Link	The physical connection that transports packets from a source to a destination.
Stream Host Receiver	Entity receiving and capable of de-encapsulating one or more streams of data according to the USB3 Vision Streaming Protocol.
Stream Transmitter Device	Entity producing one or more streams of data according to the USB3 Vision Streaming Protocol.

1.4.2 Requirements Terminology

This standard uses the conventions shown in Table 1-1 to list requirements.

Each requirement is represented by a unique number. Each number is composed of up to 3 elements:

Requirement Type: Absolute **R**equirement, **C**onditional **R**equirement, Absolute **O**bjective, **C**onditional **O**bjective

Sequence number (sn): Unique number identifying the requirement or objective. The sequence numbers are attributed sequentially as new requirements and objectives are added to the standard

Suffix: Identifies if the requirement or objective is applicable to hosts or devices, control or streaming, mechanical or some combination of these according to the terminology presented in Table 1-2.

Table 1-1: Requirements Terminology

Term	Description	Representation
Absolute Requirement	Feature that MUST be supported by the product. It is mandatory to support the feature to ensure interoperability.	R-<sn><suffix>
Conditional Requirement	Feature that MUST be supported IF another feature is present. It is mandatory to support the feature when another feature is supported.	CR-<sn><suffix>
Absolute Objective	Feature that SHOULD be supported by the product. It is recommended, but not essential.	O-<sn><suffix>
Conditional Objective	Feature that SHOULD be supported IF another feature is present. It is recommended, but not essential.	CO-<sn><suffix>

Table 1-2: Suffix Terminology

Suffix	Meaning	Description
c	**c**ontrol	Represents a control-related requirement or objective applicable to both devices and hosts.
ch	**c**ontrol **h**ost	Represents a control-related requirement or objective exclusive to hosts.
cd	**c**ontrol **d**evice	Represents a control-related requirement or objective exclusive to devices.
s	**s**tream	Represents a requirement or objective applicable to both stream transmitter devices and host receivers.
sr	**s**tream **r**eceiver	Represents a requirement or objective exclusive to stream host receivers.
st	**s**tream **t**ransmitter	Represents a requirement or objective exclusive to stream transmitter devices.
m	**m**echanical	Represents a mechanical and/or electrical requirement or objective applicable to hosts, devices and interconnects.
mi	**m**echanical **i**nterconnect	Represents a mechanical and/or electrical requirement or objective applicable to interconnects.
mh	**m**echanical **h**ost	Represents a mechanical and/or electrical requirement or objective applicable to hosts.
mhi	**m**echanical **h**ost and **i**nterconnect	Represents a mechanical and/or electrical requirement or objective applicable to hosts and interconnects at the host.

Table 1-2: Suffix Terminology(Continued)

md	**m**echanical **d**evice	Represents a mechanical and/or electrical requirement or objective applicable to devices.
mdi	**m**echanical **d**evice and **i**nterconnect	Represents a mechanical and/or electrical requirement or objective applicable to devices and the interconnects at the devices.

For example, **R-61st** is requirement number 61 that only applies to USB3 transmitter devices.

1.4.3 Acronyms

ABRM	Technology Agnostic Bootstrap Register Map
AOI	Area of Interest
BRM	Bootstrap Register Map
CFA	Color Filter Array
DCI	Device Control Interface
DEI	Device Event Interface
IAD	Interface Association Descriptor
ID	Identifier
IIDC	Instrumentation & Industrial Digital Camera
GFSI	GenDC Streaming Interface
GFSIRM	GenDC Flow Streaming Interface Register Map
GSIRM	GenDC Streaming Interface Register Map
GUID	Globally Unique Identifier
lsb	Least Significant Bit
LSB	Least Significant Byte
msb	Most Significant Bit
MSB	Most Significant Byte
PC	Personal Computer
PFNC	Pixel Format Naming Convention
ROI	Region of Interest
SBRM	Technology Specific Bootstrap Register Map
SCD	Specific Command Data
SFNC	GenICam™ Standard Features Naming Convention

SI	Streaming Interface
SIRM	Streaming Interface Register Map
SSECD	SuperSpeed Endpoint Companion Descriptors
U3V	USB3 Vision
U3VCP	USB3 Vision Control Protocol
USB	Universal Serial Bus
USB-IF	USB Implementers Forum

1.5 Reference Documents

USB Standards	
USB 2.0	Universal Serial Bus Specification, revision 2.0
USB 3.2	Universal Serial Bus Specification, revision 3.2
USB Type-C Locking Connector Specification	USB Type-C Locking Connector Specification, March 9, 2016, part of USB 3.2 Specification
Other Standards	
GenICam Pixel Format Naming Convention	Pixel Format Naming Convention (PFNC) version 2.2
GenICam	GenApi Standard Document, version 2.1.1 GenICam is a trademark of the European Machine Vision Association (EMVA).
GenCP	GenICam GenCP Generic Control Protocol, EMVA, version 1.3 GenICam is a trademark of the European Machine Vision Association.
GenICam SFNC	GenICam Standard Features Naming Convention, EMVA, Version 2.4
GenDC	GenICam Generic Data Container. EMVA, Version 1.1

1.6 Document Typographic Convention

This standard document uses the following typographic conventions:

Bootstrap registers

Bootstrap registers are highlighted using Arial font.

Ex: Device Version

Field names in packets and bootstrap registers

The names of the fields in a packet are expressed using lowercase *italic* letters, except for acronyms. When a field name contains an acronym or a reserved word, then the acronym can also use uppercase *italic* letters. When multiple words are used, they are separated with an underscore (‘_’).

Ex: *request_id*

Feature names

Feature names are expressed by concatenating all the words forming the feature name and using uppercase for the first letter of each word.

Ex: AcquisitionStart

U3VCP Commands

All U3VCP command names are expressed in uppercase with no space. Underscore (‘_’) are used to separate the CMD (command) or ACK (acknowledge) suffix.

Ex: READMEM_CMD (from GenCP standard)

USB3 Vision Status Codes

All USB3 Vision status codes are expressed using uppercase with underscore (‘_’) to separate multiple words. They start with “U3V_STATUS”.

Ex: U3V_STATUS_RESEND_NOT_SUPPORTED

USB3 Vision Events

All USB3 Vision events are expressed using uppercase with underscore (‘_’) to separate multiple words. They start with “U3V_EVENT”. Note that only the test event value is defined in the current version of this standard, meaning all other events used will be specified purely by the GenICam XML file used with the camera.

Ex: U3V_EVENT_TEST

1.7 System Overview

USB3 Vision systems are rather simple. The simplest one is a point-to-point connection between a PC and a USB3 Vision video streaming device.

Figure 1-1 demonstrates the traditional point-to-point camera to host receiver control (receiving application) use case, while Figure 1-2 demonstrates a more complex use case.

Figure 1-1: Point-To-Point Connection Between Device And Host

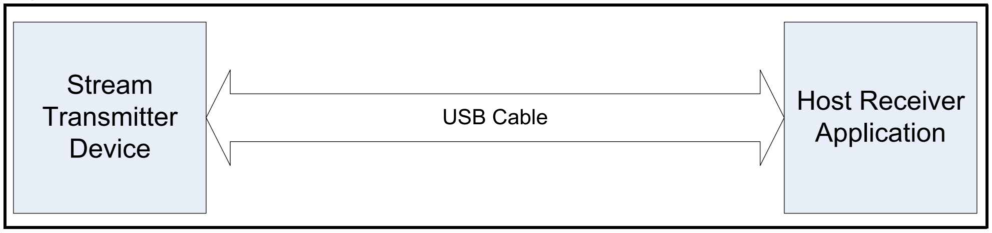
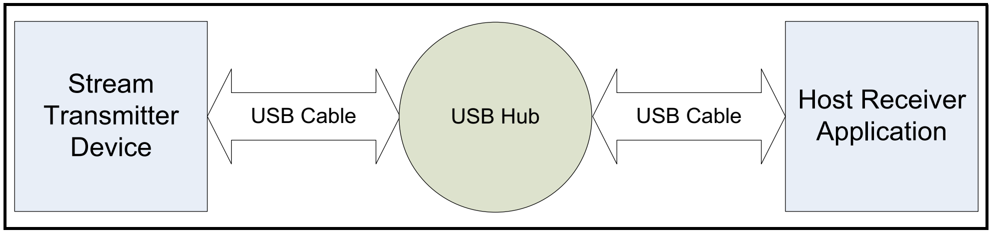


Figure 1-2: Connection Between Device and Host Via a Hub



The main concepts in a USB3 Vision system product are that:

1. A product must be an application on a host (e.g. software or a driver), a device (e.g. a camera), or a host device (e.g. a PC receiving images from a camera).
2. A product does not need to have a stream interface.

Because of this, the following device classes are defined:

3. ***Stream Transmitter Device***: A stream transmitter device is a device capable of streaming data. It includes one or more stream transmitters. For instance, this can be a camera.
4. ***Host***: A host is a device capable of controlling a USB3 Vision device. It includes one or more U3VCP controllers. For instance, this can be a PC controlling multiple USB3 Vision Devices.
5. ***Host Receiver***: A host receiver is a device capable of receiving streams. It includes one or more stream receivers. For instance, this can be a PC receiving data from a USB3 Vision Camera.
6. ***Peripheral***: A peripheral is a device that cannot transmit or receive streams. However, it can execute tasks controlled using the USB3 Vision control protocol. For instance, this can be a USB3 Vision strobe light controller.

It should be noted that a product can act as the host of more than one transmitter device. However, the parameters of a device can only be controlled by a single control application since a transmitter device can only be connected to one host.

1.8 Byte Ordering and Bit Numbering

Everything within the USB3 Vision standard uses the little-endian byte order (data, chunk data, header fields, etc.). For more details, see the GenCP standard.

2.0 USB Related Requirements

This section focuses on the USB3 Vision requirements associated with the Universal Serial Bus 3.0 Specification. This standard is available from the USB Implementers Forum at:

http://www.usb.org/developers/docs/

2.1 USB 3.x SuperSpeed Compliance

The USB3 Vision standard is written to support the development of cameras that support the USB 3.x interface.

More details on USB 3.x SuperSpeed compliance, requirements and testing can be found here:

https://usb.org/compliance

2.2 USB 2.0 High-Speed Compatibility

It is recommended that a device provides full functionality at reduced performance levels when connected to a USB 2.0 High-Speed bus.

There are no requirements or recommendations for either low- or full-speed compatibility.

O-116cd
When connected to a host, a device SHOULD first try enumerate using the 3.x SuperSpeed connection before attempting using the 2.0 HighSpeed connection. This ensures compatibility with specialized cable solutions that only connect the 3.x wire pairs.

2.3 Boot Mechanism

When connected to a host or hub, a device must independently boot into a fully functional state. It cannot rely on the presence of host software or other external devices to provide firmware or other essential data required for initialization or function.

3.0 Device Identification

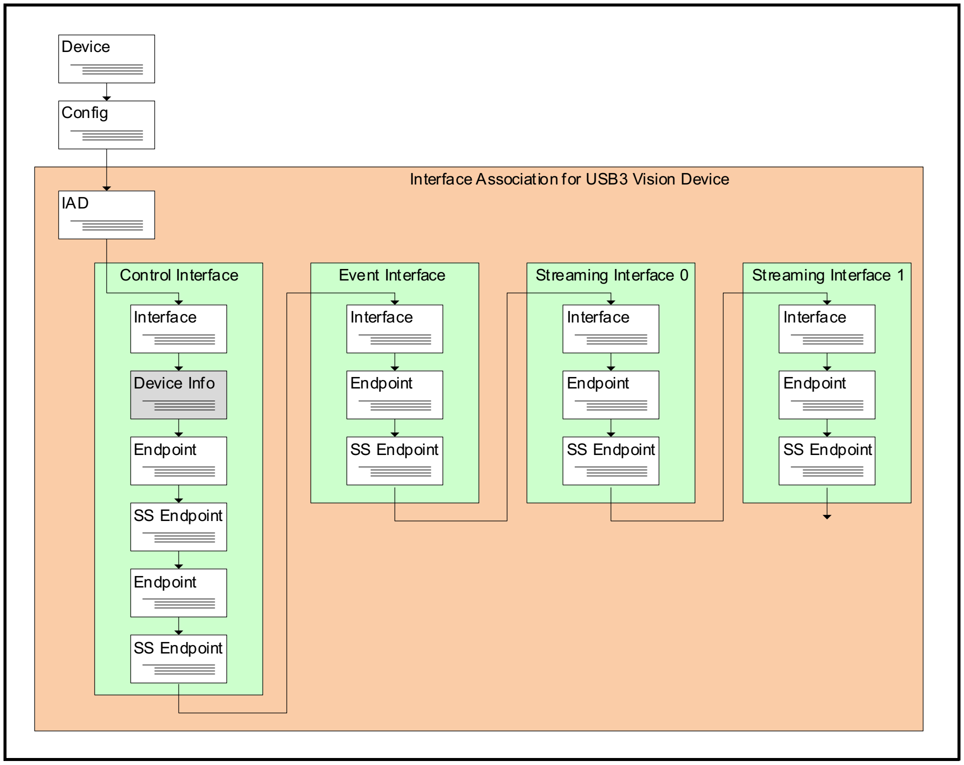
3.1 Identification Overview

3.1.1 Overview

A USB3 Vision device is identified by a USB-IF assigned class and subclass. Generic host drivers are matched to the USB3 Vision class/subclass pair.

A USB3 Vision device contains multiple interfaces, to partition the control, event, and streaming functions. These interfaces are associated through the Interface Association Descriptor (IAD) – see Figure 3-1. The IAD informs the host that interfaces are grouped into a device function. This is described in section 9.6.4 of the Universal Serial Bus 3.0 Specification.

Figure 3-1: Associated interfaces of a USB3 Vision device



To assist host software in selecting a device, some of the information found in the bootstrap registers is also made available in a class-specific descriptor called the *Device Info* descriptor.

3.1.2 USB3 Vision GUID

A USB3 Vision unique ID allows consistent identification of devices, regardless of vendor. The Globally Unique Identifier (GUID) is a 12-character hexadecimal number string formed from the USB-IF assigned vendor ID and a 32-bit unique value assigned by each vendor:

{ USB-IF assigned vendor ID } { Unique value assigned by each vendor }

R-1cd
The USB3 Vision GUID is a string, 12 characters long, of uppercase hexadecimal digits. The first 4 characters are the USB-IF vendor ID, with leading zeros if required. The last 8 characters are assigned by a vendor and MUST be unique to each device.

For example: “ABCD12345678”, where 0xABCD is the vendor ID, and 0x12345678 is a unique value found on only one device from this vendor.

CR-2cd
If the USB3 Vision GUID is printed on a device label, it MUST be prefixed by the text “U3V” followed by a space.

For example: “U3V ABCD12345678” is printed on a device label or box.

3.2 Descriptors

3.2.1 Device Descriptor

The USB3 Vision functionality is not located at the device level. It is found at the interface level.

R-3cd
The USB Device Descriptor MUST use the *Multi-interface Function Device Class Codes*, to indicate an *Interface Association Descriptor* is present.

R-4cd
The *Device Descriptor* values listed in Table 3-1 are mandatory.

Table 3-1: Specified fields of Device Descriptor

Field	Specified Value
bDeviceClass	Constant value EFh. (Miscellaneous Device Class)
bDeviceSubClass	Constant value 02h. (Common Class)
bDeviceProtocol	Constant value 01h. (Interface Association Descriptor)

Because the values of the Device Descriptor fields iManufacturer, iProduct, and iSerialNumber are not specified by this standard, these values will not be consistent between manufacturers.

R-5ch
USB3 Vision drivers MUST NOT use the Device Descriptor fields iManufacturer, iProduct, and iSerialNumber in place of the fields in the Class-Specific Camera Info Descriptor defined in Table 3-5 when identifying a USB3 Vision device instance.

It is allowed to create a USB3 Vision device which contains configurations or interfaces unrelated to the USB3 Vision standard.

3.2.2 Configuration Descriptor

This standard does not limit or specify the use of Configuration Descriptors.

3.2.3 Interface Association Descriptor

Each independent USB3 Vision device is described by a set of associated interfaces: a *Device Control Interface*, zero or one *Device Event Interfaces*, and zero or more *Device Streaming Interfaces*. These interfaces are preceded by an *Interface Association Descriptor (IAD)*. See Figure 3-1.

R-6cd
The device MUST contain an *Interface Association Descriptor* immediately before the *Device Control Interface.*

R-7cd
The *Interface Association Descriptor* values listed in Table 3-2 are mandatory.

Table 3-2: Specified fields of Interface Association Descriptor

Field	Specified Value
bFunctionClass	Constant value EFh. (Miscellaneous Class)
bFunctionSubClass	Constant value 05h (USB3 Vision)
bFunctionProtocol	Constant value 00h
iFunction	The index of a string that equals “USB3 Vision Device”

Other USB devices might be described in the set of associated interfaces. This permits a device vendor to provide custom functionality without breaking the standard. Such additional functionality may either be provided as additional interfaces within the same interface set as the USB3 Vision functionality or in a different set.

R-8ch
Host drivers are required to operate if non-USB3 Vision devices are part of the set described by the Interface Association Descriptor. The host driver may choose to ignore any non-USB3 Vision interfaces present.

3.2.4 Device Control Interface Descriptor

One Device Control Interface is required. This is described by a Standard Interface Descriptor.

R-9cd
The *Device Control Interface Descriptor* values listed in Table 3-3 are mandatory.

Table 3-3: Specified fields of Device Control Interface Descriptor

Field	Specified Value
bInterfaceClass	Constant value EFh (Miscellaneous Class)
bInterfaceSubClass	Constant value 05h (USB3 Vision).
bInterfaceProtocol	Constant value 00h (Device Control)

R-10cd
The default Device Control interface MUST be the first IAD interface. This means the bInterfaceNumber field of the default interface descriptor MUST equal the bFirstInterface field of the IAD.

The default Device Control interface requires two endpoints for bi-directional communication.

R-11cd
For the default Device Control interface, the associated *Endpoint Descriptors* and *SuperSpeed Endpoint Companion Descriptors* (SSECD) MUST describe exactly two endpoints: one bulk OUT, and one bulk IN.

R-12cd
For the two default Device Control interface endpoints, the *SuperSpeed Endpoint Companion Descriptor* values listed in Table 3-4 are mandatory.

Table 3-4: Specified fields of SSECD of the Device Control Interface

Field	Specified Value
bMaxBurst	Constant value 00h. This restricts bandwidth consumed by the control channel to one burst.

R-13cd
A USB3 Vision device MUST NOT have alternate Device Control interfaces.

An alternate interface may be introduced in a future version of this standard; host drivers need to be prepared for this possibility.

R-14ch
Alternate Device Control interfaces MUST be ignored by the host driver.

3.2.5 Class-Specific Device Info Descriptor

The *Device Info* descriptor shown in Table 3-5 provides some of the identity information available in the bootstrap registers.

R-15cd
A single instance of the *Device Info* descriptor MUST occur after the default *Device Control* Interface descriptor.

R-16cd
All strings referenced in the descriptor MUST be able to be represented as 7-bit ASCII strings.

Table 3-5: Device Info Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Descriptor size=20d. This descriptor may grow in length in future versions of the USB3 Vision Standard. This descriptor will remain backwards compatible. New fields will be added only to the end of the descriptor.
1	bDescriptorType	1	Constant	U3V_INTERFACE = 24h
2	bDescriptorSubtype	1	Constant	U3V_DEVICEINFO = 1h

Table 3-5: Device Info Descriptor(Continued)

3	bGenCPVersion	4	Number	GenCP version. Must be identical to the value in the bootstrap register *GenCP Version*.
7	bU3VVersion	4	Number	USB3 Vision version. Must be identical to the value in the bootstrap register *U3V Version*.
11	iDeviceGUID	1	Index	Index to USB3 Vision GUID string.
12	iVendorName	1	Index	Index to Vendor Name string. Must be identical to the string in the bootstrap register *Manufacturer Name*.
13	iModelName	1	Index	Index to Model Name string. Must be identical to the string in the bootstrap register *Model Name*.
14	iFamilyName	1	Index	Index to Family Name string. Must be identical to the string in the bootstrap register *Family Name*. May be zero if not supported.
15	iDeviceVersion	1	Index	Index to Device Version string. Must be identical to the string in the bootstrap register *Device Version*.
16	iManufacturerInfo	1	Index	Index to Manufacturer Info string. Must be identical to the string in the bootstrap register *Manufacturer Info*.
17	iSerialNumber	1	Index	Index to Serial Number string. Must be identical to the string in the bootstrap register *Serial Number*.
18	iUserDefinedName	1	Index	Index to User Defined Name string. Must be identical to the string in the bootstrap register *User Defined Name*. May be zero if not supported.

Table 3-5: Device Info Descriptor(Continued)

<table>
<tr>
<td>19</td>
<td>bmSpeedSupport</td>
<td>1</td>
<td>Bitmask</td>
<td>Bitmask indicating the bus speeds supported by this device:
<table>
<tr><th>Bit offset
(lsb &lt;&lt; x)</th><th>Width
(bits)</th><th>Description</th></tr>
<tr><td>0</td><td>1</td><td>Low-Speed</td></tr>
<tr><td>1</td><td>1</td><td>Full-Speed</td></tr>
<tr><td>2</td><td>1</td><td>High-Speed</td></tr>
<tr><td>3</td><td>1</td><td>SuperSpeed Gen1x1/5 Gbps</td></tr>
<tr><td>4</td><td>1</td><td>SuperSpeedPlus Gen2x1/10 Gbps</td></tr>
<tr><td>5</td><td>1</td><td>SuperSpeedPlus Gen1x2/10 Gbps</td></tr>
<tr><td>6</td><td>1</td><td>SuperSpeedPlus Gen2x2/20 Gbps</td></tr>
<tr><td>7</td><td>1</td><td>Reserved, must be 0</td></tr>
</table>
The device must function as a USB3 Vision device at the indicated speeds. Software might use this to compare against the device’s current speed and determine if it should generate an error.</td>
</tr>
</table>

3.2.6 Device Event Interface Descriptor

The Event Interface is optional on the device.

CR-17cd

If a USB3 Vision device supports events, the device MUST provide the Device Event Interface (DEI) described by a Standard Interface Descriptor. This interface is used for the transmission of asynchronous events from the device to the host through the event pipe.

CR-18cd

The Device Event *Interface Descriptor* values specified in Table 3-6 are mandatory for the event interface.

Table 3-6: Specified fields of Device Event Interface Descriptor

Field	Specified Value
bInterfaceClass	Constant value EFh (Miscellaneous Class)
bInterfaceSubClass	Constant value 05h (USB3 Vision)
bInterfaceProtocol	Constant value 01h (Device Events)

The default Device Event Interface requires one bulk endpoint for data sent to the host.

CR-19cd

If events are supported on the device, for the default Device Event Interface, the associated *Endpoint Descriptor* and *SuperSpeed Endpoint Companion Descriptor*

MUST describe exactly one bulk IN endpoint.

CR-20cd
For the Device Event Interface endpoint, the *SuperSpeed Endpoint Companion Descriptor* values listed in Table 3-7 are mandatory.

Table 3-7: Specified fields of SSECD of the Event Interface

Field	Specified Value
bMaxBurst	Constant value 00h. This restricts bandwidth consumed by the event channel to one burst.

R-21cd
A vendor MUST NOT create alternate Device Event Interfaces.

An alternate interface may be introduced in a future version of this standard; host drivers need to be prepared for this possibility.

R-22ch
Alternate Device Event Interfaces MUST be ignored.

3.2.7 Device Streaming Interface Descriptor

The Device Streaming Interface is optional. If present the stream interface is described by the Standard Interface Descriptor.

CR-23cd
The device streaming *Interface Descriptor* values specified in Table 3-8 are mandatory if a device has a streaming interface.

Table 3-8: Specified fields of Device Streaming Interface Descriptor

Field	Specified Value
bInterfaceClass	Constant value EFh (Miscellaneous Class)
bInterfaceSubClass	Constant value 05h (USB3 Vision)
bInterfaceProtocol	Constant value 02h (Device Streaming)

The default Device Streaming Interface requires one bulk endpoint for sending data to the host.

CR-24cd
If a device has a streaming interface the default Device Streaming Interface, the associated *Endpoint Descriptor* and *SuperSpeed Endpoint Companion Descriptor* MUST specify exactly one bulk IN endpoint.

R-25cd
A vendor MUST NOT create alternate Device Streaming Interfaces.

CR-123st
If multiple streaming interfaces are present, the order of the streaming interfaces in the descriptor list MUST match the stream index and the order of the SIRM register regions for each streaming index.

An alternate interface may be introduced in a future version of this standard; host drivers need to be prepared for this possibility.

R-26ch
Alternate Device Streaming interfaces (i.e. bAlternateSetting != 0) MUST be ignored.

4.0 Device Control

4.1 Control And Event Transport Layer

4.1.1 Overview

USB3 Vision compliant devices provide different interfaces for different functions:

> **R-27cd**
> To control the behavior of a USB3 Vision device, the device MUST provide the following endpoint/interface:
>
> *Control endpoint 0*: This is the default path for controlling the standard USB aspects of the device.
>
> *Device Control Interface* (DCI): This interface is used to control the USB3 Vision aspects of the device.

In addition, the devices may provide an optional event channel for the transmission of asynchronous events from the device to the host.

Configuration and settings of the interfaces are specified in the Chapter 3.

> **R-28ch**
> Host software controlling the device MUST set-up and enable the Device Event Interface (DEI) if present on a device and MUST support events sent by the device and MUST be able to receive data from the Device Event Interface whenever the host accesses features on the device.

> **R-29ch**
> Host software controlling the device MUST use the GenICam standard event mechanism to interpret event data sent by the device if the GenICam standard is used to provide access to the device's features.

4.1.2 Control and Event Channel Protocol

The DCI and DEI operate according to the Generic Control Protocol (GenCP). GenCP is part of the GenICam standard.

NOTE: No acknowledgement is used for events, because only a single, unidirectional endpoint is used. The underlying transport layer is reliable and will perform as many retries as specified by the USB standard.

> **R-30cd**
> For USB3 Vision devices the byte order of bootstrap registers and protocol fields is little-endian, which is the byte ordering used by USB.

4.1.3 Data Transfer Layout

GenCP defines a generic layout for the command/acknowledge data transfer. It is divided into the following four sections:

Prefix
Common Command Data (CCD)
Specific Command Data (SCD)
Postfix

4.1.3.1 Prefix and Postfix

The content of Prefix is technology specific and is not defined by GenCP. USB3 Vision defines the Prefix as 0x43563355 for the control channel and 0x45563355 for the event channel. The content of Postfix is technology specific and is not used by USB3 Vision.

R-31c
The USB3 Vision device and the host MUST use a valid USB3 Vision Prefix and MUST NOT use a Postfix.

The *Prefix* is a 32-bit magic value to ensure data integrity.

For the control channel the *Prefix* is defined as follows:

Width (Bytes)	Offset (Bytes)	Description
4	0	0x43563355 (Control Prefix) This *Prefix* contains the ASCII string “U3VC” standing for USB3 Vision Control (with ‘U’ in the LSB).

For the event channel the *Prefix* is defined as follows:

Width (Bytes)	Offset (Bytes)	Description
4	0	0x45563355 (Event Prefix) This *Prefix* contains the ASCII string “U3VE” standing for USB3 Vision Event (with ‘U’ in the LSB).

4.1.3.2 Common Command Data

GenCP defines a technology agnostic Common Command Data (CCD) section. The section can include a command request or a command acknowledge.

R-32c
The DCI and DEI command data layout MUST follow the format described below:

Width (Bytes)	Offset (Bytes)	Description
Prefix = 0x43563355 or 0x45563355 (Control or Event prefix)		
2	0	flags
2	2	command_id

2	4	length
2	6	request_id
SCD		

R-33c
The DCI acknowledge data layout MUST follow the format described below:

Width (Bytes)	Offset (Bytes)	Description
Prefix = 0x43563355		
2	0	status code
2	2	command_id
2	4	length
2	6	request_id
SCD		

GenCP specifies two command request flags, one for *command resends* and one for *request acknowledge.* Only *request acknowledge* is supported by USB3 Vision.

R-134cd
A device receiving a command with a set command resend flag where prefix and command_id are valid MUST respond with an acknowledge packet where the status code is set to U3V_STATUS_RESEND_NOT_SUPPORTED.

A USB3 Vision device uses *commands* specified by GenCP. Other *commands* are not supported and must be ignored by the USB3 Vision device and the host application.

R-34cd
A command with request_id set to 0 MUST reset the request_id sequence and be accepted by a USB3 Vision device.

R-35ch
Controlling host software MUST start with a request_id set to 0 after connecting to a USB3 Vision device.

R-124cd
A device receiving an unsupported GenCP command id MUST respond with a GENCP_NOT_IMPLEMENTED error. If the command uses an incorrect prefix or format, the device MUST ignore the command completely.

GenCP specifies *EVENT_CMD* as a *command_id* for events. The *EVENT_ACK command_id* is not used by USB3 Vision.

R-36ch
The host software MUST support the PENDING_ACK command.

4.1.3.2.1 Status Codes

GenCP specifies a status code field within the acknowledge packet layout and the related status codes. A USB3 Vision device must use the GenCP status codes as stated by GenCP.

USB3 Vision defines technology specific status codes in addition to the generic GenCP status codes. The USB3 Vision status codes apply to control as well as to data streaming.

R-37cd
The status code format used MUST match the GenCP standard in the Table 4-1:

Table 4-1: Status Codes

Bit offset (lsb << x)	Width (bits)	Description
0	12	Status Code
12	1	Reserved Set to 0
13	2	Namespace 0 = GenCP Status Code 1 = Technology Specific Code 2 = Device Specific Code
15	1	Severity 0 = Warning/Info 1 = Error

R-38cd
USB3 Vision status codes MUST set the *Namespace* field to 1.

Table 4-2 and Table 4-3 specify the USB3 Vision status codes:

Table 4-2: Command Status Codes

Status Code	Name	Description
0xA001	U3V_STATUS_RESEND_NOT_SUPPORTED	This status code must be used in an Acknowledge Packet in response to a Command Packet where: - Prefix is valid - *command_id* is valid - Flags field is set to 0xC000 (CommandResend)
0xA002	U3V_STATUS_DSI_ENDPOINT_HALTED	This status code must be used in an Acknowledge Packet in response to a Command Packet when Stream Enable flag of SI Control Register is set to 1 and the endpoint of the Device Streaming Interface is halted.

Table 4-2: Command Status Codes(Continued)

0xA005	U3V_STATUS_DEI_ENDPOINT_HALTED	This status code must be used in an Acknowledge Packet in response to a Command Packet when Event Enable flag of EI Control Register is set to 1 and the endpoint of the Device Event Interface is halted.
0xA003	U3V_STATUS_SI_PAYLOAD_SIZE_NOT_ ALIGNED	This status code must be used in an Acknowledge Packet in response to a Command Packet when: The value written to the SI streaming size registers is not aligned to Payload Size Alignment value of the SI Info register. Applicable registers: - SI Payload Transfer Size - SI Payload Final Transfer1 Size - SI Payload Final Transfer2 Size - SI Maximum Leader Size - SI Maximum Trailer Size
0xA004	U3V_STATUS_SI_REGISTERS_ INCONSISTENT	This status code must be used in an Acknowledge Packet in response to a Command Packet trying to set the Stream Enable bit in the SI Control Register which was not successful because values within the stream interface registers were not consistent or legal.

Table 4-3: Data Block Status Codes

Status Code	Name	Description
0xA100	U3V_STATUS_DATA_DISCARDED	Data discarded in the current block. This status code must be used in the trailer when: - Some data in the block has been discarded.
0xA101	U3V_STATUS_DATA_OVERRUN	Device is unable to send all data. This status code must be used in the trailer when the Device cannot send all data because the data does not fit within the programmed SIRM register settings.

4.1.3.3 Specific Command Data

GenCP defines a technology agnostic Specific Command Data section. The section can include data depending on the content of the corresponding Common Command Data section.

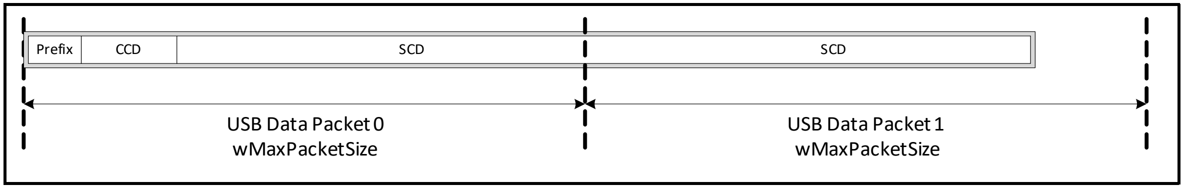
4.1.3.3.1 Control Channel Specific Command Data

The length and content of *Specific Command Data* (*SCD*) depends on the *command_id* used. USB3 Vision devices and the host software use the *SCDs* as defined by GenCP.

Depending on the length of the *SCD*, a command or acknowledge transfer can be larger than a USB data packet specified by *wMaxPacketSize.*

R-39c
A USB3 Vision device and a host MUST support command and acknowledge transfers which are distributed over multiple USB data packets

Figure 4-1: Command or Acknowledge Transfer Distributed Over Two USB Data Packets



R-40cd
A device MUST implement the Maximum Command Transfer Length register specifying the maximum supported command transfer length in bytes, where all data of a command packet are included. The minimum value of the Maximum Command Transfer Length is wMaxPacketSize represented by the descriptor of the corresponding endpoint.

R-41ch
A host MUST NOT send more data per command transfer than specified in the Maximum Command Transfer Length register. See Section 4.2.2.4.

R-42cd
A device MUST implement the Maximum Acknowledge Transfer Length register specifying the maximum supported acknowledge transfer length in bytes, where all data of an acknowledge packet are included. The minimum value of the Maximum Acknowledge Transfer Length is wMaxPacketSize represented by the descriptor of the corresponding endpoint.

R-43cd
A USB3 Vision device MUST NOT send more data than specified in the Maximum Acknowledge Transfer Length register. See Section 4.2.2.5.

NOTE: The maximum USB Transfer Size might be host-dependent and could further limit the maximum size of command or acknowledgment transfers used.

R-117ch
A host MUST always submit transfers on the IN endpoint of the Device Control Interface equal to the calculated maximum total returned size of the possible acknowledgments, including all GenCP headers and any requested payload data. This value may be optionally rounded up to wMaxPacketSize of the endpoint.

R-118cd
A device MUST terminate transfers of the acknowledgment response on the IN endpoint of the Device Control Interface by sending a zero-length-packet if the total transferred size is a multiple of wMaxPacketSize of the endpoint and less than the calculated maximum total returned size of the possible acknowledgments, including all GenCP headers and any requested payload data.

4.1.3.3.2 Event Channel Specific Command Data

Event Channel Specific Command Data is defined to be processed by GenICam via the host application, but the mapping of this data into GenICam is not defined. The length and content of the *SCD* for the event channel depends on the *command_id* used. A USB3 Vision device and the host software use the SCDs defined by GenCP

Data from the SCD is mapped into the address space of individual Event Ports using the following scheme:.

Width (Bytes)	Offset (Bytes)	Description
Prefix = 0x45563355 (U3VE)		
CCD (command_id = EVENT_CMD)		
2	0	reserved, must be 0
2	2	event_id
8	4	timestamp
X	12	data

The mapping starts immediately after the CCD section and all fields past this section are within the addressable space of the Event Port. As an example, a node that is reading the timestamp of a given event would use an offset of 4.

The definitions of the various fields are defined by GenCP. The timestamp fields must be interpreted using the definition of the timestamp mechanism defined by the GenCP standard.

R-119ch
A host MUST always submit transfers on the IN endpoint of the Device Event Interface equal to the maximum size specified by Maximum Event Transfer Length rounded up to the wMaxpacketSize advertised on the endpoint.

4.1.3.4 U3V Event IDs

Table 4-4: USB3 Vision Technology Specific Event IDs

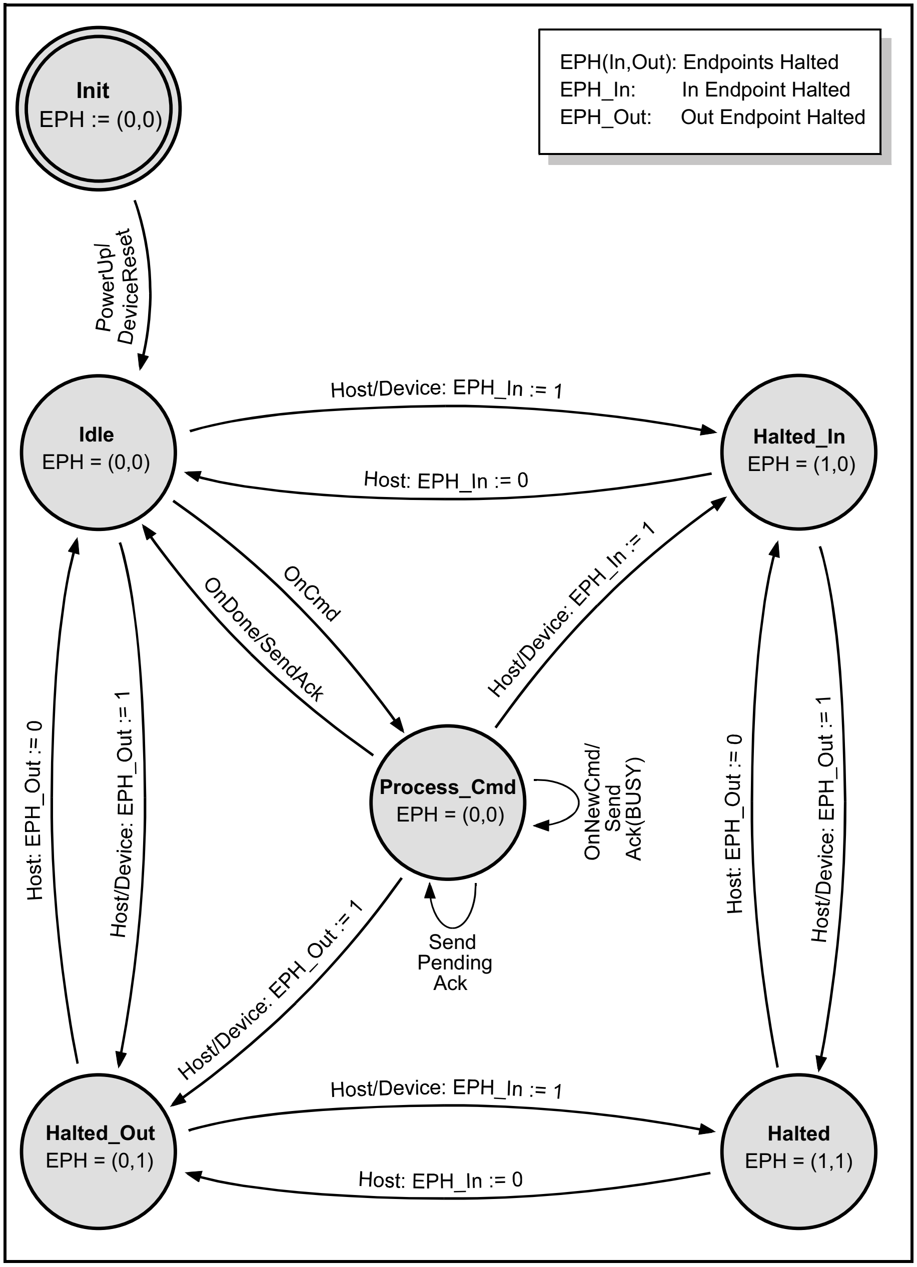
Event ID	Name	Description
0x4FFF	U3V_EVENT_TEST	This event id must be used for the Test event controlled by TriggerEventTest. This event has no data.

4.1.4 Controlling a USB3 Vision Device

4.1.4.1 Device Control Interface

The OUT endpoint of the Device Control Interface (DCI) is used by the host to send a command packet to the device. The IN endpoint of the DCI is used by the device to send an acknowledge packet to the host as response to a command packet.

Figure 4-2: Device Control Interface State Diagram



The state diagram shown in Figure 4-2 illustrates the behavior of a USB3 Vision device using its DCI.

NOTE:A device must accept transfers queued to the IN and OUT endpoints in any order. For example, a host driver might queue one or more acknowledgment transfers to the IN endpoint before queuing a request to the OUT endpoint.

R-44cd
On the Device Control Interface, the device MUST accept the host queuing an IN endpoint acknowledge request in any order in relation to sending the OUT endpoint command request.

4.1.4.1.1 Normal Usage

Initially the device starts in the **Idle** state with its IN and OUT endpoints enabled (EPH=(0,0)).

The host sends a command putting the device into **Process_Cmd** state. When processing is done the device sends an acknowledge returning into the **Idle** state.

R-45cd
If processing takes longer than the time specified in the bootstrap register (Maximum Device Response Time), the device MUST send a pending acknowledge.

If a new command is received while processing a previous one, the device will send an acknowledge with a Busy status code as defined by GenCP.

4.1.4.1.2 Re-synchronization

To stop the device while it is processing or to connect to a device without knowing its state, the host has to set the ENDPOINT_HALT (EPH_x:=1) feature on both endpoints, putting the device into any of the **Halted** states. Clearing the ENDPOINT_HALT (EPH_x:=0) feature on both endpoints returns the device to the **Idle** state.

4.1.4.2 Device Event Interface

A USB3 Vision device uses the DEI to send events to the host. To control the operation of the DEI the device uses a register block, which is called Event Interface Register Map (EIRM).

CR-46cd
A USB3 Vision device MUST implement the EIRM if it supports events.

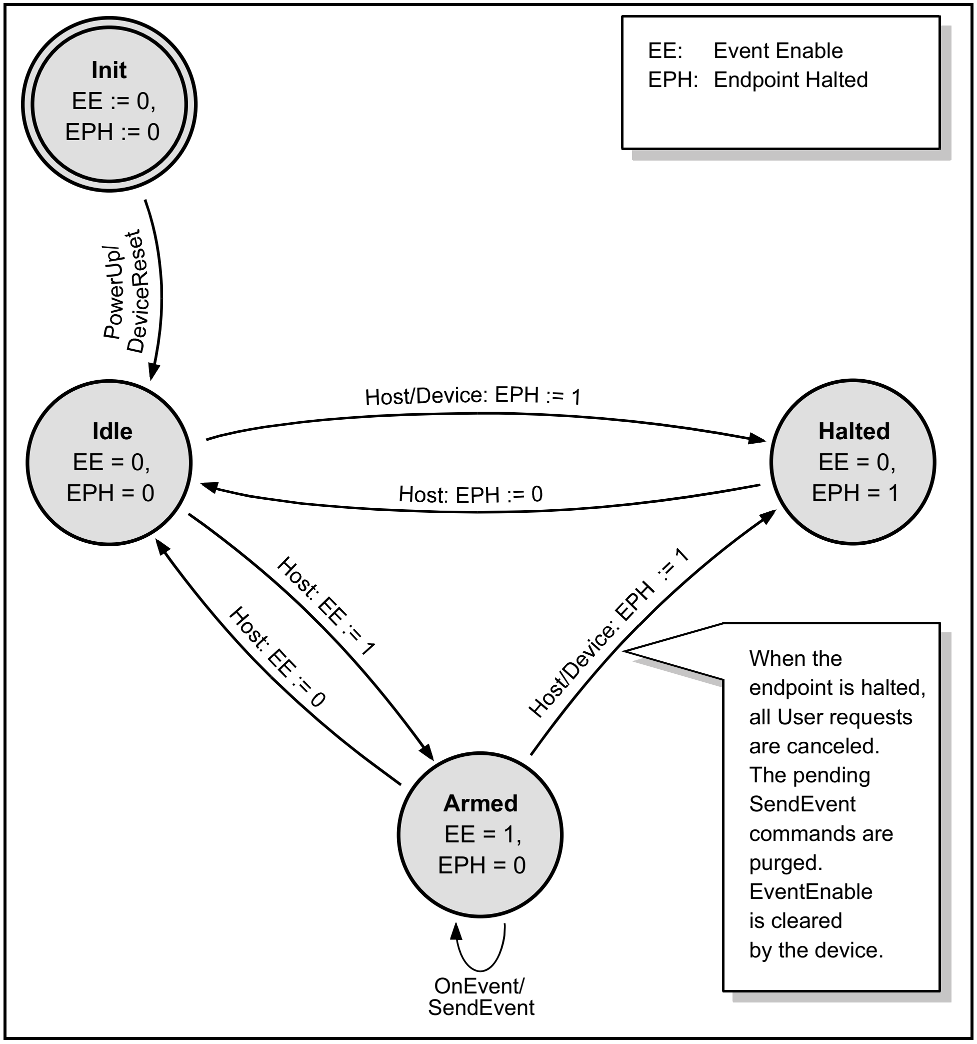
CR-47cd
A USB3 Vision device MUST set the address of the EIRM in the corresponding register of the SBRM if events are supported.

4.1.4.2.1 Device Event Interface State Diagram

The IN endpoint of the DEI is used by a USB3 Vision device to send event packets to the host.

The state diagram shown in Figure 4-3 illustrates the behavior of a USB3 Vision device using its DEI.

Figure 4-3: Device Event Interface State Diagram



4.1.4.2.2 Normal Usage

Initially the USB3 Vision device starts in the **Idle** state and the Event Enable (EE) is cleared. Setting EE to “1” enables the transmission of events and the **Armed** state is entered. On an event an event command (EVENT_CMD) is sent and the device remains in the **Armed** state. By setting EE to “0”, the device will not send any event commands to the host and returns to the **Idle** state.

4.1.4.2.3 Re-synchronization

If a host or a device sets the ENDPOINT_HALT feature on the endpoint, the device switches to the **Halted** state and stops generating events and the register EE is set to zero by the device. When the host clears the ENDPOINT_HALT feature, the device returns to the **Idle** state.

4.1.4.2.4 Event Interface Register Map

The following registers, shown in Table 4-5 are associated with the event channel, if the device supports events. Registers are listed relative to the start address of the Event Interface Register

Map block (EIRM Address). The address and size of the EIRM is part of the device´s bootstrap registers (EIRM Address and EIRM Length). See Section 4.2.2.

The EIRM registers marked as “R” may only be read by the host. An attempt to write to a register marked “R” must return an error.

The following shows the definitions for each column in Table 4-5:

- Address: Address of the register in the Event Interface RM

- Name: Name of the register for further reference

- Support: M=Mandatory/R=Recommended/CM=Conditional Mandatory

- Access: R=Read only/W=Write only/RW=Read Write

- Length: Length of the register in bytes

- Description: Brief description of the register

Table 4-5: Event Interface Register Map (EIRM)

Address	Name	Support	Access	Length	Description
EIRM Address + 0x00	EI Control	M	RW	4	Event Interface Control Register
EIRM Address + 0x04	Maximum Event Transfer Length	M	R	4	Specifies the maximum supported event transfer length of the device.
EIRM Address + 0x08	Event Test Control	M	RW	4	Control the generation of test events.

NOTE: EIRM Address is defined in the Technology Specific Bootstrap Registers.

4.1.4.2.5 EI Control

This register is used to control the event operation.

Address	EIRM + 0x00
Length	4
Access Type	RW
Data Type	UINT32
Support	M
Factory Default	0

Bit offset (lsb << x)	Width (bits)	Description

0	1	**Event Enable** When set to “1”, the device is able to send events. When set to “0”, the device is unable to send events to the host. If the Device Event Interface endpoint is halted, it must return a U3V_STATUS_DEI_ENDPOINT_HALTED error when Event Enable is set.
1	31	reserved, must be 0.

4.1.4.2.6 Maximum Event Transfer Length

This register specifies the maximum supported event transfer length of the device in bytes.

CR-48cd
A USB3 Vision device MUST implement the Maximum Event Transfer Length register specifying the maximum supported event transfer length in bytes, where all data of an event packet are included. The minimum value of the Maximum Event Transfer Length is wMaxPacketSize represented by the descriptor of the corresponding endpoint.

CR-49cd
A USB3 device MUST NOT send more event data per event packet than specified in the Maximum Event Transfer Length register.

Address	EIRM + 0x04
Length	4
Access Type	R
Support	M
Data Type	UINT32
Factory Default	Device specific

Bit offset (lsb << x)	Width (bits)	Description
0	32	Maximum Event Transfer Length Specifies the maximum supported event transfer length of the device in bytes.

4.1.4.2.7 Event Test Control

This register is used to control the generation of test events.

Address	EIRM + 0x08
Length	4
Access Type	RW
Data Type	UINT32

Support	M
Factory Default	0

Bit offset (lsb << x)	Width (bits)	Description
0	1	**TriggerEventTest** When set to “1” and the Event Enable flag is “1”, a test event with id U3V_EVENT_TEST is sent. This flag is self-clearing.
1	31	reserved, must be 0.

4.1.5 Manifest

The Manifest section of GenCP describes a way to store multiple GenICam files in the device. Each file has a separate table entry to specify the file properties.

R-50cd
A USB3 Vision device MUST provide a valid SHA1 hash value for each file in the Manifest section.

CR-51ch
Host software that caches a downloaded file MUST re-download the file if the hash of its cached file does not match the one listed in the manifest.

4.2 Bootstrap Register Map (BRM)

Based on the GenCP standard the total BRM is divided into two parts, the Technology Agnostic Bootstrap Register Map (ABRM), and the Technology Specific Bootstrap Register Map (SBRM). The next sections describe the bootstrap registers which are defined by this standard.

R-125cd
A device MUST return a GENCP_INVALID_ADDRESS error if a host accesses any bootstrap registers within the ABRM or SBRM that are marked reserved or are listed as unused by this specification

4.2.1 Technology Agnostic Bootstrap Register Map (ABRM)

The first 64 KB of the bootstrap register map is defined as part of the GenCP standard. Table 4-6 shows the ABRM that shall be incorporated.

The following shows the definitions for each column in Table 4-6:

- Address: Address of the register in the devices BRM

- Name: Name of the register for further reference

- Support: M=Mandatory/R=Recommended/CM=Conditional Mandatory

- Access: R=Read only/W=Write only/RW=Read Write

- Length: Length of the register in bytes

- Description: Brief description of the register

All registers marked as "CM" (Conditional Mandatory) are conditional based on the relevant bit set in the Device Capability Register.

Table 4-6: Technology Agnostic Bootstrap Register Map (ABRM)

Address	Name	Support	Access	Length	Description
0x00000	GenCP Version	M	R	4	Complying GenCP Version
0x00004	Manufacturer Name	M	R	64	String containing the name of the manufacturer
0x00044	Model Name	M	R	64	String containing the name of the device model
0x00084	Family Name	CM	R	64	String containing the name of the family of this device
0x000C4	Device Version	M	R	64	String containing the version of this device
0x00104	Manufacturer Info	M	R	64	String containing additional manufacturer specific information
0x00144	Serial Number	M	R	64	String containing the serial number of the device
0x00184	User Defined Name	CM	RW	64	String containing the user defined name of the device
0x001C4	Device Capability	M	R	8	Bit field describing the device’s capabilities
0x001CC	Maximum Device Response Time	M	R	4	Maximum response time in ms
0x001D0	Manifest Table Address	M	R	8	Address of the Manifest Table
0x001D8	SBRM Address	CM	R	8	Address of the Technology Specific Bootstrap Register Map
0x001E0	Device Configuration	M	RW	8	Bit field describing the device’s configuration
0x001E8	Heartbeat Timeout	CM	RW	4	Heartbeat Timeout in ms. **Not used for this standard.**
0x001EC	Message Channel channel_id	CM	RW	4	channel_id used for the message channel. **Not used for this standard.**
0x001F0	Timestamp	CM	R	8	Current device time in ns
0x001F8	Timestamp Latch	CM	W	4	Timestamp Latch
0x001FC	Timestamp Increment	CM	R	8	Timestamp Increment
0x00204	Access Privilege	CM	RW	4	Access Privilege**. Not used for this standard.**

Table 4-6: Technology Agnostic Bootstrap Register Map (ABRM)(Continued)

0x00208	Protocol Endianess	CM	R	4	Endianess of protocol fields and bootstrap registers. Only little-endian is supported by this standard.
0x0020C	Implementation Endianess	CM	R	4	Endianess of device implementation registers Only little-endian is supported by this standard.
0x00210	Device Software Version	CM	R	64	Version of the public software interface of the device.
0x00250	Reserved	M	no	64944	Reserved Register Space

The following features in the Device Capability register are mandated to indicate the following:

- Access Privilege Available—Unavailable (Not used by this standard)
- Timestamp Supported—Available (Required by this standard)
- String Encoding—0x0 ASCII (Required by this standard)
- SBRM Supported—Available (Required by this standard)
- Written Length Field Supported—Available (Required by this standard)

R-111cd
A device MUST set the fields in the Device Capability register to indicate support for the optional GenCP features that are listed as mandatory for this specification.

R-115cd
A device MUST set the GenCP Version register to a minimum version matching the GenCP specification referenced in section 1.5, but MAY report and implement any newer minor version of the specification.

4.2.2 Technology Specific Bootstrap Register Map (SBRM)

The SBRM MUST start on a free manufacturer-specific address space after address 0xFFFF. Table 4-7 shows the SBRM that shall be incorporated.

All registers marked as "CM" (Conditional Mandatory) are conditional based on the relevant bit set in the U3VCP Capability Register.

The following shows the definitions for each column in Table 4-7:

- Address: Address of the register in the devices BRM
- Name: Name of the register for further reference
- Support: M=Mandatory/R=Recommended/CM=Conditional Mandatory
- Access: R=Read only/W=Write only/RW=Read Write
- Length: Length of the register in bytes

- Description: Brief description of the register

Table 4-7: Technology Specific Bootstrap Register Map (SBRM)

Address	Name	Support	Access	Length	Description
SBRM	U3V Version	M	R	4	Version of the USB3 Vision standard this Bootstrap Register Map complies with.
SBRM + 0x04	U3VCP Capability Register	M	R	8	Bit field specifying the device's U3V features and capabilities.
SBRM + 0x0C	U3VCP Configuration Register	M	RW	8	Configures additional features on the control channel.
SBRM + 0x14	Maximum Command Transfer Length	M	R	4	Specifies the maximum supported command transfer length of the device.
SBRM + 0x18	Maximum Acknowledge Transfer Length	M	R	4	Specifies the maximum supported acknowledge transfer length of the device.
SBRM + 0x1C	Number of Stream Channels	M	R	4	Number of Stream Channels and the corresponding Streaming Interface Register Maps (SIRM)
SBRM + 0x20	SIRM Address	CM	R	8	Address of the first Streaming Interface Register Map.
SBRM + 0x28	SIRM Length	CM	R	4	Specifies the length of each SIRM.
SBRM + 0x2C	EIRM Address	CM	R	8	Address of the Event Interface Register Map.
SBRM + 0x34	EIRM Length	CM	R	4	Specifies the length of the EIRM.
SBRM + 0x38	IIDC2 Address	CM	R	8	Address of the IIDC2 register set.
SBRM + 0x40	Current Speed	M	R	4	Specifies the current speed of the USB link.
SBRM + 0x44	Reserved	M	-	65468	Reserved Register Space

4.2.2.1 U3V Version

Version of the USB3 Vision standard this Bootstrap Register Map complies with.

Address	SBRM
Length	4
Access Type	R

Support	M
Data Type	2 x UINT 16 fields
Factory Default	Implementation specific

Bit offset (lsb << x)	Width (bits)	Description
0	16	Minor Version This field represents the minor version of the USB3 Vision standard.
16	16	Major Version This field represents the major version of the USB3 Vision standard.

4.2.2.2 U3VCP Capability Register

This register reports the additional features and bootstrap registers supported by this device on its control channel.

Address	SBRM + 0x04
Length	8
Access Type	R
Support	M
Data Type	Bit field
Factory Default	Implementation specific

Bit offset (lsb << x)	Width (bits)	Description
0	1	SIRM Available Set if the device supports at least one device streaming interface. SIRM Address and SIRM Length register must be implemented.
1	1	EIRM Available Set if the device supports at least one device event interface. EIRM Address and EIRM Length register must be implemented.
2	1	**IIDC2 Available** Set if the device supports IIDC2 register map. IIDC2 register must be implemented.
3	61	Reserved, must be 0.

CR-112cd
If the EIRM Available bit is set in the U3V Capability Register, the corresponding Message Channel capability bit MUST be set in the Device Capability register in the ABRM.

4.2.2.3 U3VCP Configuration Register

This register provides the ability to configure additional features to control the device. These additional features must be indicated by the U3VCP capability register.

Address	SBRM + 0x0C
Length	8
Access Type	RW
Support	M
Data Type	Bit field
Factory Default	Device specific

Bit offset (lsb << x)	Width (bits)	Description
0	64	Reserved, must be 0.

4.2.2.4 Maximum Command Transfer Length

This register specifies the maximum supported command transfer length of the device in bytes. Note that the maximum USB Transfer Size depends on the operating system.

Address	SBRM + 0x14
Length	4
Access Type	R
Support	M
Data Type	UINT32
Factory Default	Device specific

Bit offset (lsb << x)	Width (bits)	Description
0	32	Maximum Command Transfer Length Specifies the maximum supported command transfer length of the device in bytes.

4.2.2.5 Maximum Acknowledge Transfer Length

This register specifies the maximum supported acknowledge transfer length of the device in bytes. Note that the maximum USB Transfer Size depends on the operating system.

Address	SBRM + 0x18
Length	4
Access Type	R
Support	M

Data Type	UINT32
Factory Default	Device specific

Bit offset (lsb << x)	Width (bits)	Description
0	32	Maximum Acknowledge Transfer Length Specifies the maximum supported acknowledge transfer length of the device in bytes.

4.2.2.6 Number of Stream Channels

This register specifies the number of Stream Channels. If a device does not support streaming capabilities, the register must be set to 0.

R-52cd
A device MUST support the Number of Stream Channels register.

Address	SBRM + 0x1C
Length	4
Access Type	R
Support	M
Data Type	UINT32
Factory Default	Device specific

Bit offset (lsb << x)	Width (bits)	Description
0	32	Number of Stream Channels Specifies the number of Stream Channels. 0, if no stream channel is supported.

4.2.2.7 Streaming Interface Register Map (SIRM) Address

The register contains the address of the entry address of the first Streaming Interface Register Map. If the device supports more than one Stream Channel, the second SIRM starts at SIRM Address + SIRM Length.

CR-53cd
A device MUST support the SIRM Address register, if the SIRM Available flag is set in the capability register.

Address	SBRM + 0x20
Length	8
Access Type	R
Support	CM

Data Type	UINT64
Factory Default	Device specific

Bit offset (lsb << x)	Width (bits)	Description
0	64	SIRM Address Specifies the address of the Entry Address of Streaming Interface Register Map.

4.2.2.8 SIRM Length

The register specifies the length of each Streaming Interface Register Map.

CR-54cd
A device MUST support the SIRM Length register, if the SIRM Available flag is set in the capability register.

Address	SBRM + 0x28
Length	4
Access Type	R
Support	CM
Data Type	UINT32
Factory Default	Device specific

Bit offset (lsb << x)	Width (bits)	Description
0	32	SIRM Length Specifies the length in bytes of each Streaming Interface Register Map.

4.2.2.9 EIRM Address

The register contains the address of the entry address of the Event Interface Register Map. If the device supports more than one Event Channel the second EIRM starts at EIRM Address + EIRM Length.

NOTE: This version of the standard only supports one Event Channel.

CR-55cd
A device MUST support the EIRM Address register, if the EIRM Available flag is set in the capability register.

Address	SBRM + 0x2C
Length	8

Access Type	R
Support	CM
Data Type	UINT64
Factory Default	Device specific

Bit offset (lsb << x)	Width (bits)	Description
0	64	EIRM Address Specifies the address of the Entry Address of Event Interface Register Map.

4.2.2.10 EIRM Length

The register specifies the length of the Event Interface Register Map.

CR-56cd
A device MUST support the EIRM Length register, if the EIRM Available flag is set in the capability register.

Address	SBRM + 0x34
Length	4
Access Type	R
Support	CM
Data Type	UINT32
Factory Default	Device specific

Bit offset (lsb << x)	Width (bits)	Description
0	32	EIRM Length Specifies the length of each Event Interface Register Map.

4.2.2.11 IIDC2 Address

The register contains the address of the entry address of the IIDC2 Register Set. IIDC2 is a camera control register layout standardized by the Japan Industrial Imaging Association (JIIA).

CR-57cd
A device MUST support the IIDC2 Address register, if the IIDC2 Available flag is set in the capability register.

Address	SBRM + 0x38
Length	8

Access Type	R
Support	CM
Data Type	UINT64
Factory Default	Device specific

Bit offset (lsb << x)	Width (bits)	Description
0	64	IIDC2 Address Specifies the address of the Entry Address of IIDC2 Register Set.

4.2.2.12 Current Speed

The register indicates the current speed of the device. The layout of bits in this register match the first 7 bits of “bmSpeedSupport” field and the full 32 bits of “bmSpeedSupportExtended” in the Device Info Descriptor but only indicate the current speed rather than all supported speeds.

R-58cd
A device MUST indicate its currently connected speed in the Current Speed register.

Address	SBRM + 0x40
Length	4
Access Type	R
Support	M
Data Type	UINT32
Factory Default	Device specific

Bit offset (lsb << x)	Width (bits)	Description
0	1	Low-Speed
1	1	Full-Speed
2	1	High-Speed
3	1	SuperSpeed Gen1x1/5 Gbps
4	1	SuperSpeedPlus Gen2x1/10 Gbps
5	1	SuperSpeedPlus Gen1x2/10 Gbps
6	1	SuperSpeedPlus Gen2x2/20 Gbps
7	25	Reserved, must be 0

4.3 Standard Features List for Cameras

The USB3 Vision standard relies on the GenICam standard (http://www.genicam.org) to describe the features supported by the camera. This description takes the form of an XML device description file respecting the syntax defined by the GenApi module of the GenICam standard.

R-59cd
Any USB3 Vision device MUST provide an XML device description file compliant to the GenApi module of the GenICam standard. That XML MUST be compliant with GenICam schema version 1.1 and up.

This XML file is retrieved and interpreted by the host software to enumerate the features supported by the device. The XML device description file provides the mapping between a device feature and the device registers supporting it.

Since a large portion of USB3 Vision devices are cameras, this section provides a list of mandatory features that must be present in the USB3 Vision camera XML description file. Note that non-camera devices are not covered by this section. Any camera providing an XML device description file compliant to the syntax of the GenApi module of the GenICam standard and including the mandatory features presented in this section is considered suitable for USB3 Vision.

NOTE: The requirements in this section only apply to camera devices. Therefore, all requirements and objectives below are conditional as they only apply to this type of USB3 Vision devices.

4.3.1 XML Description File Mandatory Features for Cameras

CR-60st
If a USB3 Vision device is a camera, then it MUST support the features provided in Table 4-8 in its XML description files and MUST comply with the GenICam Standard Feature Naming Convention (SFNC) regarding the behavior of those features.

Table 4-8: USB3 Vision Mandatory Features for the Camera XML Description File

Feature Name	Name	Access	Units	Description
Width	IInteger	R/(W)	Pixels	Width of the image output by the camera on the stream channel.
Height	IInteger	R/(W)	Pixels	Height of the image output by the camera on the stream channel. For linescan sources, this represents the maximum height of the frame created by combining consecutive lines.
PixelFormat	IEnumeration	R/(W)	N/A	Format of the pixel output by the camera on the stream channel. See Figure 5.5.7.

Table 4-8: USB3 Vision Mandatory Features for the Camera XML Description File(Continued)

PayloadSize	IInteger	R	Bytes	Maximum number of bytes transferred for each image or block on the stream channel. This is the maximum total size of data payload for a block. Leader and Trailer are not counted. This is mainly used by the host software to determine size of buffers to allocate (largest buffer possible for current mode of operation).
AcquisitionMode	IEnumeration	R/(W)	N/A	Used by host software to set the mode of acquisition. This feature indicates how a sequence of images is generated (continuously, single shot, multi-shot, …) Only a single value is mandatory for USB3 Vision. {“Continuous”} *Continuous*: Configures the camera to stream an infinite sequence of images that must be stopped explicitly by the application. Note that the AcquistionMode can have more than this single entry. The “AcquisitionStart” and “AcquisitionStop” features are used to control the acquisition state.
AcquisitionStart	ICommand	(R)/W	N/A	Start image acquisition using the specified acquisition mode.
AcquisitionStop	ICommand	(R)/W	N/A	Stop image acquisition using the specified acquisition mode.

NOTE: Access modes given between parentheses are optional.

This list is only intended for camera devices. Non-camera devices do not have to support the above features.

4.3.2 Other Features

Recommended feature names for other device functionalities are provided in the “GenICam Standard Features Naming Convention” (available from http://www.genicam.org). This naming convention should be respected when implementing such functionalities to guarantee interoperability. Proposals for new recommended feature names shall be addressed to the GenICam committee.

CO-61cd

When features that match functionality described in the "GenICam Standard Feature Naming Convention" (SFNC) are implemented in a device, those features

SHOULD follow the names and behavior outlined within the SFNC in order to provide interoperability.

5.0 Streaming Data

5.1 Streaming Transport Layer

5.1.1 Preface

The streaming transport layer standard describes objectives and solutions for USB compliant, reliable, fast, and low overhead data transfer for vision devices.

This version of the standard only covers streams going from a device (typically a camera or a similar device) to a host. The terms **device** and **host** used here relate to the USB standard which describes the role of a device and a host in great detail.

As the entire USB protocol is host centric all transfers are described as seen from the host's perspective.

Therefore an "in" transaction means that data is transferred from the device to the host.

This version of the standard uses Bulk mode as the standard transport mechanism for the streaming interface. Future versions of this standard may also consider Isochronous mode as an option.

5.1.2 Zero Copy Mechanism

As USB 3.0 is capable of data transfer at high rates, it is especially important to design the streaming protocol in a way that ensures maximum efficiency on both the device and host. It is particularly important to avoid unnecessary copying of data on the host and the related waste of CPU resources and memory bandwidth.

Accordingly, provisions must be made to make a user´s payload buffer available to a USB driver and fill it directly with the data coming from the bus. To this end, the layout of data sent over the bus must ensure that data transferred to the user's payload buffer are clearly separated from ancillary data which is processed within the lower layers of the transport layer software.

Additionally, mechanisms are provided to allow the host and device to co-operate on splitting GenDC-formatted payloads such that each Flow and the GenDC Descriptor can all be transferred to different locations in memory on the host. The details of using this format are discussed in Section 5.2.2, GenDC Transfer Modes.

Within the present version of this standard, this efficient behavior is called "zero copy".

5.1.3 Alignment Restrictions

In order to allow USB3 Vision devices, host controllers, and software to make effective use of zero-copy data transfers, the specification allows large flexibility in terms of splitting transfers into multiple blocks as needed by the host. However, a device may not support single-byte granularity/alignment of transfer sizes. For example, a seven-byte buffer may need to be transmitted from a device that only supports 4-byte transmission alignment.

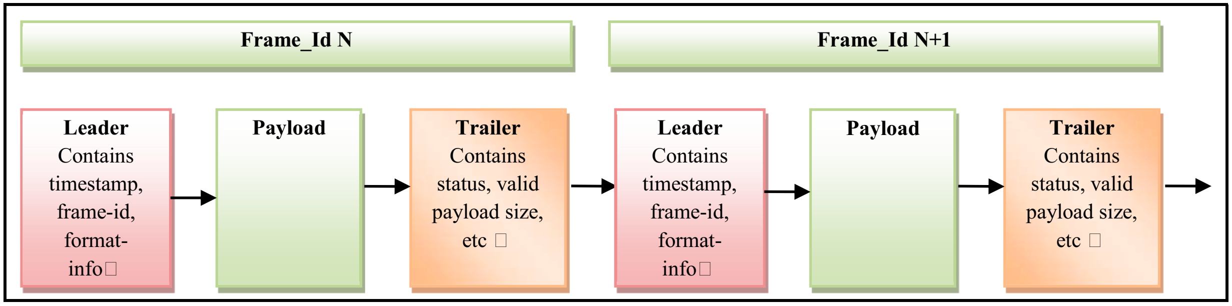
The solution is for the device to provide a mechanism to expose its required alignment characteristics (via Payload Size Alignment) and the host must use that as a minimum alignment size when programming any streaming interface transfer registers. The host may also take into account further restrictions that it may have of its own (typically for performance reasons) and calculate the least common multiple between the two restrictions.

It is the host's responsibility to take this above alignment into account when allocating buffers for transfer and programming the transfer registers. The transfer splitting mechanisms provided by this specification allow the host to make this necessary alignment invisible to the end application buffers (if desired) by bounce-buffering a minimal amount of the transfer.

5.2 Streaming Mechanism

To allow the separation of user requested payload data and USB3 Vision protocol specific data, this standard introduces the concept of a leader - payload - trailer data sequence. This sequence is shown in Figure 5-1.

Figure 5-1: Streaming Sequence


Frame_Id N
Frame_Id N+1
Leader
Contains timestamp, frame-id, format-info□
Payload
Trailer
Contains status, valid payload size, etc □
Leader
Contains timestamp, frame-id, format-info□
Payload
Trailer
Contains status, valid payload size, etc □

There must be exactly one transfer per leader/trailer but there can be several transfers for the payload buffer. The device provides registers which the host software fills with the amount and data size of the transfers it uses for the leader/payload/trailer. The leader and trailer must each fit within a single transfer.

5.2.1 Standard Payload Transfer

This transfer mode is used for simple payloads that do not use GenDC format. This is the original format first introduced in USB3 Vision version 1.0.

5.2.1.1 Control of Streaming Transfers

A dedicated set of bootstrap registers allows the control operation of the streaming interface. This set of registers is called Stream Interface Register Map (SIRM). The registers are described in detail in Section 5.4.

The registers are implemented as bootstrap registers to allow the low level part of the host software to interact with them without the need to use GenICam.

The registers describe the transfers expected from the host side software and control details of streaming such as expected payload sizes and stream enable/disable.

5.2.1.2 Leader Transfer

The leader must be sent as a single transfer. The maximum length of the leader is written into the SI Maximum Leader Size Register by the host.

R-62sr
Host software MUST set the SI Maximum Leader Size Register to a value large enough to contain a complete leader data block as defined by USB3 Vision. A larger value than the minimum required for the leader may be set by the host.

5.2.1.3 Payload Sequence

A payload sequence consists of one or more transfers. The structure of this sequence is defined by the host.

A set of registers in the SIRM are used to specify payload transfers. These registers allow the host software to tailor the payload transfers to its needs and specific restrictions. These restrictions regarding the USB transfer size might originate from the driver being used, OS, USB host chipset, or the device (via the Payload Size Alignment register field). The Final Transfer 1 / 2 blocks enable the host to handle such limitations while maximizing the use of zero copy.

The SI Payload Transfer Size, SI Payload Transfer Count and SI Payload Final Transfer1 Size and SI Payload Final Transfer2 Size registers describe the host´s requirements concerning payload data layout. If the device has less payload data to send than expected by the host it must complete the outstanding transfer(s) using short or zero-length packets.

The payload is divided into “equal sized blocks” + “final transfer1”+ “final transfer2”.

Properties of “equal sized blocks” are described by SI Payload Transfer Size and SI Payload Transfer Count.

The maximum size of payload data in bytes the device may send can be calculated as follows: maximum_size_of_payload_data_in_bytes = SI Payload Transfer Size * SI Payload Transfer Count + SI Payload Final Transfer1 Size + SI Payload FinalTransfer2 Size

Size of “final transfer1” is described by SI Payload Final Transfer1 Size.

Size of “final transfer2” is described by SI Payload Final Transfer2 Size.

For instance, given a limitation that the transfer size needs to be a multiple of 1KB but less than or equal to 64KB (limitations enforced by the WinUSB driver on Windows), the host software could break up a 326,680 byte transfer into 4 equally sized transfers of 64KB, then a final transfer 1 of 63KB, then a final transfer 2 of 1KB into a temporary buffer (copying 24 bytes into the user's buffer).

The stream receiver can determine the valid payload data size from specific fields within the trailer. See Section 5.5.3.

R-63st
The device MUST never send more than SI Payload Transfer Size * SI Payload Transfer Count + SI Payload Final Transfer1 Size + SI Payload FinalTransfer2 bytes of payload data.

R-64st
Payload transfers MUST be sent in the order: “Equal sized blocks” → “final transfer1”→ “final transfer2”.

R-65st
If the SI Payload Transfer Size Register is zero, “equal sized blocks” transfers are omitted from the Payload.

R-66st
If the SI Payload Transfer Count Register is zero, “equal sized blocks” transfers are omitted from the Payload.

R-67st
If the SI Payload Final Transfer1 Size Register is zero, “final transfer1” transfer is omitted.

R-68st
If the SI Payload Final Transfer2 Size Register is zero, “final transfer2” transfer is omitted.

R-69st
If the device has less payload data to send than expected by the host it MUST complete the outstanding transfer(s) using short packets, zero-length packets, or padded data.

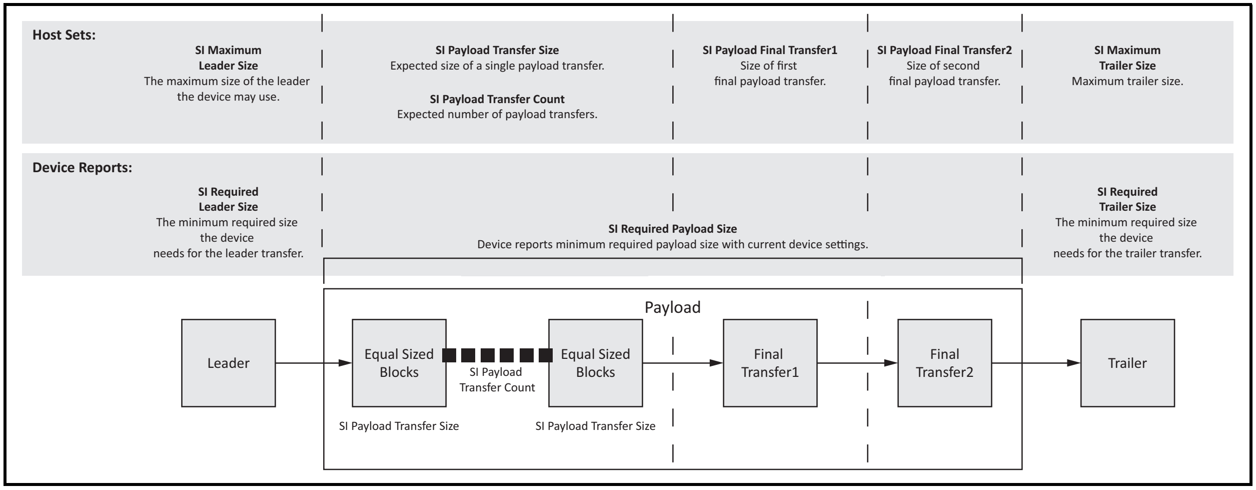
R-126st
If the device has less payload data to send than expected by the host, the *valid_payload_size* field in the trailer MUST not contain a length beyond the first short or zero-length packet.

R-127sr
Within a given payload sequence for a single buffer, the stream receiver must discard any payload data received in subsequent transfers after a transfer is terminated by a short or zero-length packet.

R-70st
The device MUST store the effective payload size in the trailer field valid_payload_size.

See Figure 5-2 for a standard transmission sequence from the host and device perspective.

Figure 5-2: Payload Sequence


Host Sets:
SI Maximum Leader Size
The maximum size of the leader the device may use.
SI Payload Transfer Size
Expected size of a single payload transfer.
SI Payload Transfer Count
Expected number of payload transfers.
SI Payload Final Transfer1
Size of first final payload transfer.
SI Payload Final Transfer2
Size of second final payload transfer.
SI Maximum Trailer Size
Maximum trailer size.
Device Reports:
SI Required Leader Size
The minimum required size the device needs for the leader transfer.
SI Required Payload Size
Device reports minimum required payload size with current device settings.
SI Required Trailer Size
The minimum required size the device needs for the trailer transfer.
Payload
Leader
Equal Sized Blocks
SI Payload Transfer Count
Equal Sized Blocks
Final Transfer1
Final Transfer2
Trailer
SI Payload Transfer Size
SI Payload Transfer Size

5.2.1.4 Trailer Transfer

The trailer must be sent as a single transfer. The maximum length of the trailer is written into the SI Maximum Trailer Size Register by the host.

R-71sr
Host software MUST set SI Maximum Trailer Size Register to a value large enough to contain a complete trailer data block as defined by the SI Required Trailer Size register. A larger value than the minimum required for the trailer may be set by the host.

5.2.2 GenDC Transfer Modes

GenDC is the standard mechanism to transfer more complex payloads that cannot be represented using a simple PFNC-encoded buffer and/or Chunk Data, such as 3D data. There are multiple mechanisms to support GenDC transfers. These modes are: Container Transfer Mode, Serial Flow Transfer Mode, and Parallel Flow Transfer Mode. Each of these modes are explained in detail in the sections below.

The Serial Flow and Parallel Flow modes allow the individual elements of the GenDC Container to be split into different buffers on the host, utilizing the flow mechanism described in GenDC, while the Container mode can only send the data to a single buffer, using only a single flow. In all modes, it is possible for the host to re-assemble a contiguous GenDC Container without copying the payload portion of the data.

From the host’s perspective, the Serial Flow and Parallel Flow Transfer modes satisfy the exact same purpose, allowing the device to zero-copy multiple buffer segments, each potentially variable-sized, into individual, arbitrary buffers on the host. Typically, a device would only need to implement one or the other depending on whether its individual component data is generated in parallel or not. The Parallel Flow mode does presume some additional capabilities on the host (Bulk streams) that are only present in newer operating systems/USB APIs, so this might be a reason a device could decide to support both.

In both Serial Flow and Parallel Flow GenDC transfer modes, the leader and trailer of the U3V streaming mechanism is used to transport the GenDC Descriptor alone in Flow 0, while the Component payloads starts at Flow 1. The rationale for this is to allow the GenDC descriptor and payload to be transferred to different locations in host memory (if desired), while also not adding additional USB transfer overhead for simple cases. If the transport layer driver is expected to deliver complete GenDC payloads to the software layers above it, it is expected to configure the transfers such that Flows 1 through N are arranged sequentially in the Container buffer, leaving room for the GenDC Descriptor ahead of it. After transmission is complete, the host must then copy the GenDC Descriptor from the location within the U3V Leader or Trailer into the complete GenDC Container buffer.

CR-135st
Devices supporting GenDC MUST support at least one of the GenDC Serial Flow or Parallel Flow Transfer modes.

CR-136st
Devices supporting GenDC Serial Flow Transfer Mode must also support GenDC Container Transfer Mode

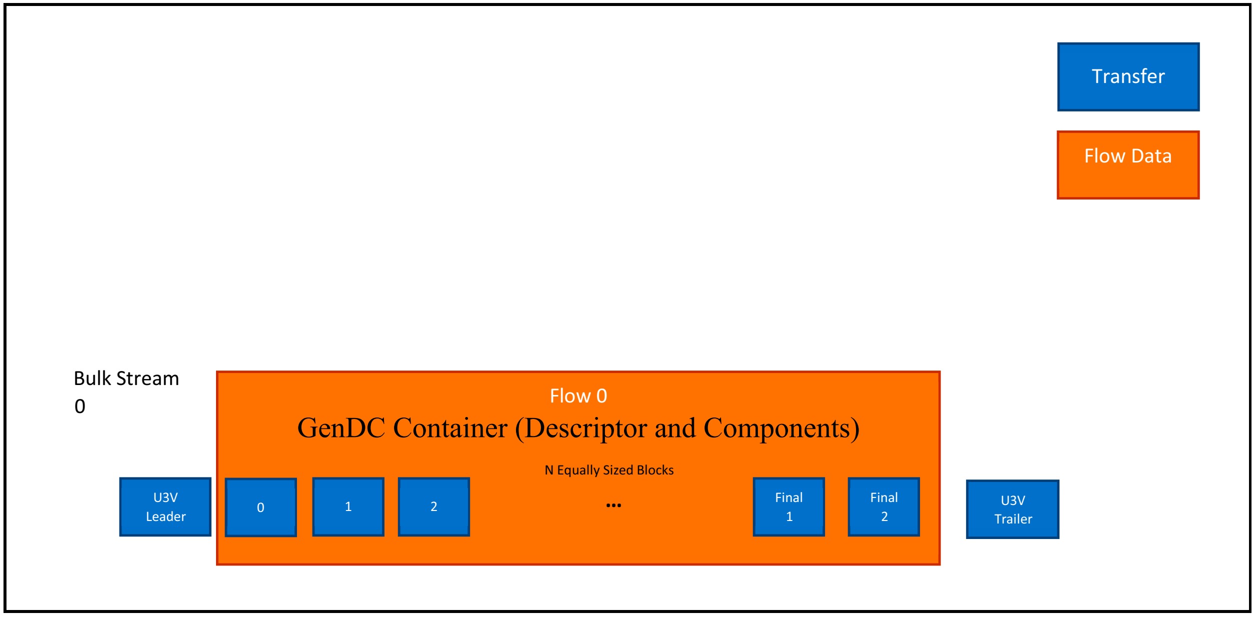
CR-137sr
Hosts supporting GenDC MUST support at least one of the GenDC Serial Flow Transfer or GenDC Container Transfer Modes

CO-138sr
Hosts supporting GenDC SHOULD support GenDC Parallel Flow Transfer Mode unless running on a system where bulk streams are not supported

5.2.2.1 GenDC Container Transfer Mode

This mode is identical to the Standard Payload Transfer mode, but the payload portion of the sequence contains a complete GenDC Container instead of a standard payload. All requirements from the Standard Payload Transfer mode apply to this mode as well. The purpose of this mode is to allow simplified usage of the device from the host when a contiguous GenDC container is desired. Note that only a single flow is supported and thus none of the extended GenDC SIRM register sets are used. See Figure 5-3.

Figure 5-3: GenDC Container Transfer Sequence

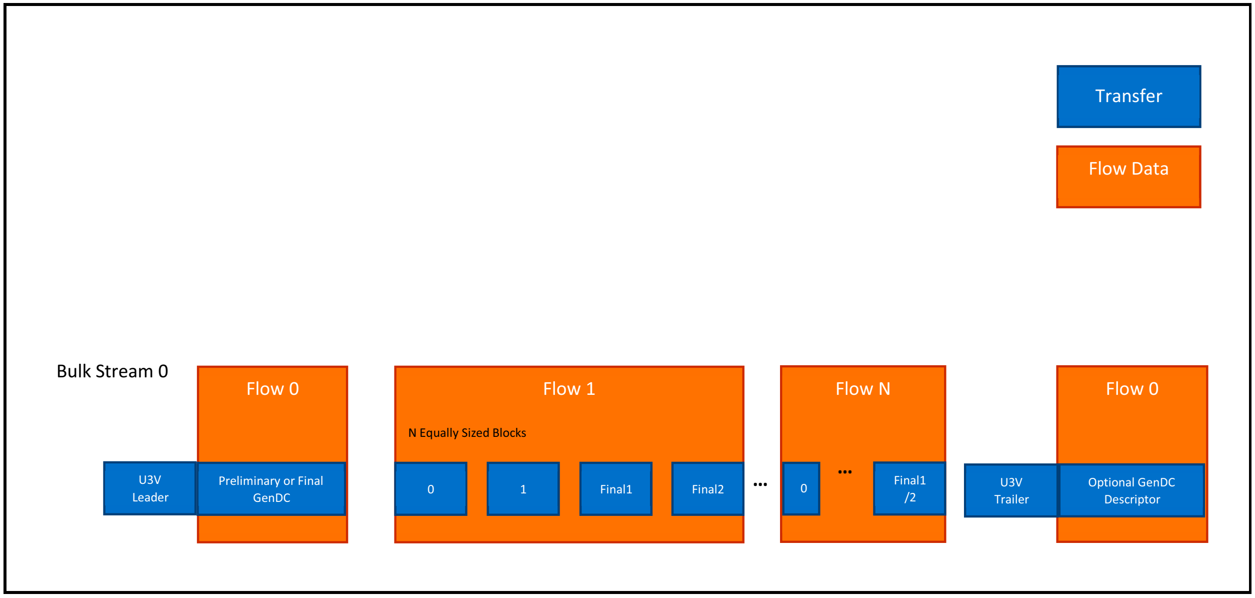


5.2.2.2 GenDC Serial Flow Transfer Mode

This mechanism uses a single bulk stream, like Standard Payload Transfers. Flow 0 of the GenDC Container, containing only the GenDC Descriptor, is transmitted in the Leader, Trailer, or both transfers.

Flows 1 to N, containing the GenDC Payloads, are transmitted in sequence by repeating the standard payload sequence for each flow (excluding the leader and trailer portions), but using each flow's GenDC SIRM subset instead of the standard SIRM registers. The transport layer driver can determine the sizes of each flow from the flow table to program each independent SIRM subset. See Figure 5-4.

Figure 5-4: GenDC Serial Flow Transfer Sequence

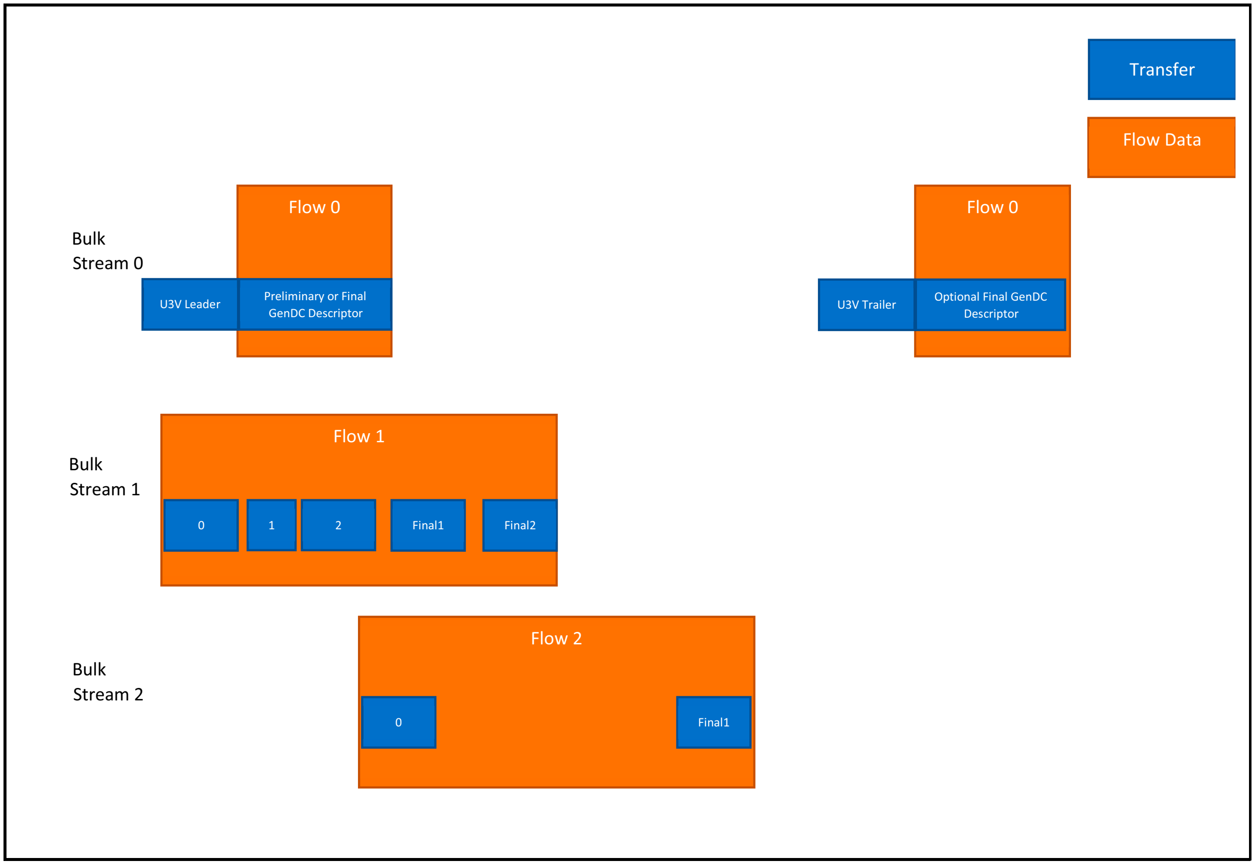


5.2.2.3 GenDC Parallel Flow Transfer Mode

This mechanism is identical to the GenDC Serial Flow Transfer Mode, with the exception that all flows are sent over different USB bulk streams (on the same endpoint), with flow N corresponding to bulk stream N.

This approach allows a device to alternate sending packet data for individual buffer elements of a single payload at its own discretion, while still directing the data to the correct location on the host. The host is expected to queue buffers to all the individual streams in the payload simultaneously such that the device can send packets to them in any order. See Figure 5-5.

Figure 5-5: Gen DC Parallel Flow Transfer Sequence



5.3 Streaming Operation States

5.3.1 Normal Usage

After startup or device reset the streaming interface reaches the **Idle** state waiting for streaming to be enabled by the host using the Stream Enable bit. When Stream Enable (Host:SE := 1) is set by the host the device prepares for streaming using the SIRM registers contents. After setting Stream Enable to 1 (Host:SE := 1), changing the SIRM registers marked as RW* have no effect on streaming until the next stream start cycle triggered by setting the Stream Enable bit (Host:SE := 1). The device must capture the content of these registers internally when starting the stream.

The device is now in the **Streaming** state and is prepared to stream data over the stream endpoint.

When the host wants to stop streaming, it must clear Stream Enable (Host:SE := 0). The device then switches to the transient state **Idle_n**. The device must stop streaming as soon as possible and flush all internal buffers. When all buffers are flushed (NDB=0) the device returns to the **Idle** state.

5.3.2 Device Cannot Directly Flush its Internal Buffers

If the device is not able to flush all buffers after the Stream Enable bit has been cleared (Host:SE := 0), it must switch to the **Halted** state (Device:EPH:=1) putting the endpoint into stall. If that happens the host must clear the halt condition on the endpoint to switch the device back to the **Idle** state. When the halt condition has been cleared on the streaming endpoint by the host (Host:EPH := 0), the device returns to the **Idle** state.

5.3.3 Host Re-synchronization

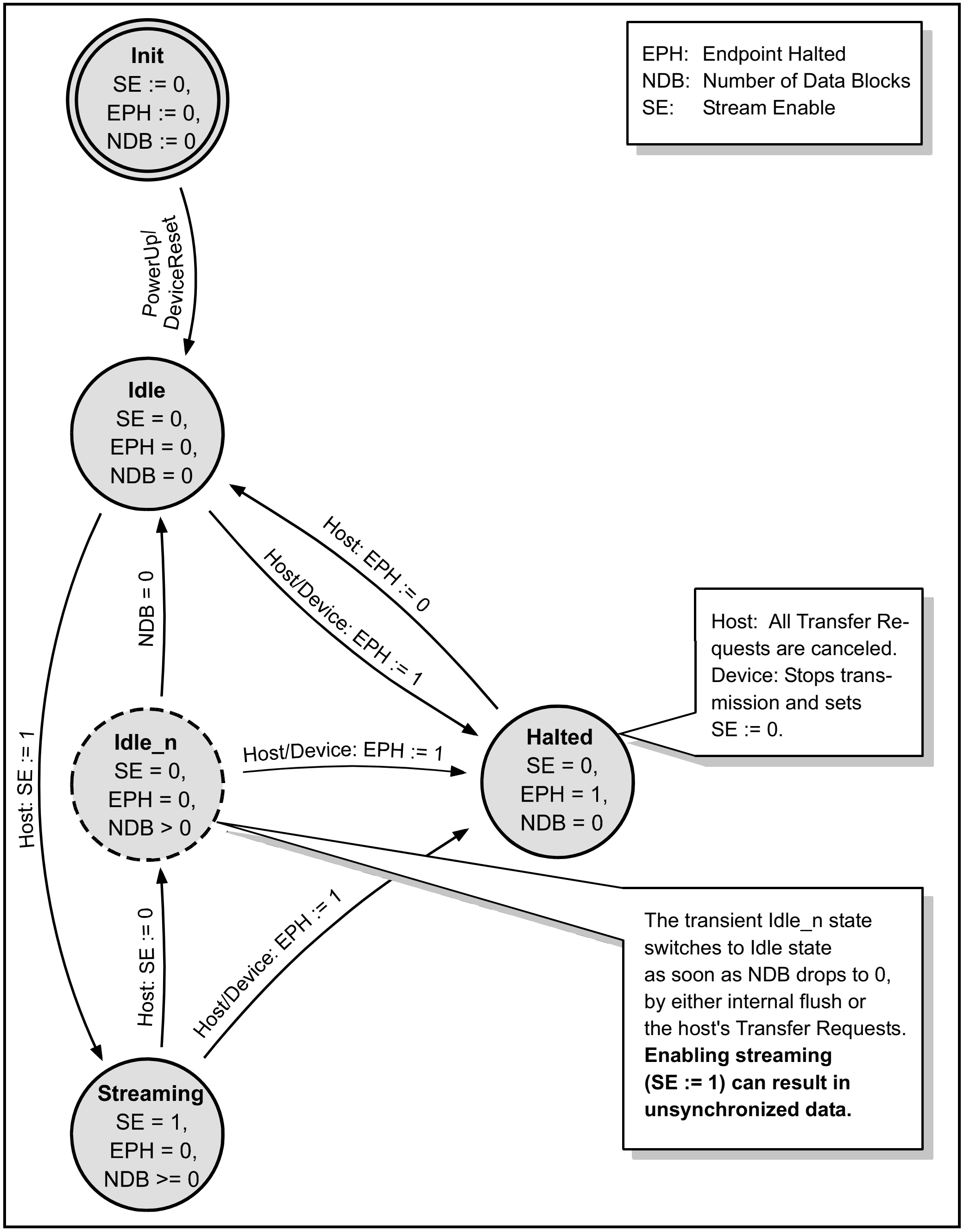
If the host gets out of sync within the streaming process it must halt the streaming endpoint (Host:EPH := 1). This can happen from **Idle** or **Streaming** states. When streaming is active (Host:SE=1) and the host sets the halt condition on the streaming endpoint (Host:EPH:=1), the device switches to the **Halted** state and the Stream Enable bit is cleared by the device itself (Device:SE:=0). When the halt condition has been cleared on the streaming endpoint by the host (Host:EPH := 0), the device returns to the **Idle** state.

5.3.4 Streaming Transmission State Diagram

Figure 5-6 shows the various states involved in the streaming operation. This diagram describes the low level functionality of the streaming transport layer.

Note that this does not include higher-level concepts covered by the GenICam SFNC document. In particular, a user may use host software to control a transfer via TransferStart/Stop independently of the state of the underlying USB3 Vision stream. For more information concerning the control of streaming from a users perspective, please refer to the transfer model description within the GenICam SFNC document (See the Device Acquisition Model / Acquisition Control / Transfer Control chapters).

Figure 5-6: Transmission States



5.4 Streaming Control Registers

Aside from starting and stopping streams, the device needs to know the leader/payload/trailer layout.

This is controlled by registers accessed through the device control protocol. See Chapter 4.

A device can have zero or more streaming interfaces.

This standard only describes stream transmitters. The following registers, shown in Table 5-1, are associated with a transmitter´s streaming channels (IN stream from host side view). Registers are listed relative to the start address of the SIRM register block (SIRM Address). The address and size of the SIRM register map is part of the device´s bootstrap registers (SIRM Address). See Section 4.2.2.

R-72cd
The SIRM registers marked as “R” may only be read by the host. An attempt to write to a register marked “R” MUST return an error.

The SIRM registers marked as “RW” may be written and read back at any time by the host.

The SIRM registers marked as “RW*” may be written and read back at any time by the Host independent of the state Stream Enable. However, the Device must use a copy of these registers while Stream Enable is enabled. The Device must capture the register values when Stream Enable is enabled and use them until Stream Enable is disabled. Changes to the Streaming registers while Stream Enable is enabled must not be used by the device. This means the host cannot change the streaming buffer layout while Stream Enable is enabled. However, values written into these registers will become active the next time Stream Enable is set to 1.

R-73st
The Device MUST capture the R/W* registers shown in Table 5-1 when Stream Enable is enabled and use them until Stream Enable is disabled. Changes to the Streaming registers while Stream Enable is enabled MUST NOT become active until Stream Enable is enabled again.

- Address: Address of the register in the devices BRM

- Name: Name of the register for further reference

- Support: M=Mandatory/R=Recommended/CM=Conditional Mandatory

- Access: R=Read only/W=Write only/RW=Read Write/RW*=like RW but content is internally captured when streaming is enabled.

- Length: Length of the register in bytes

- Description: Brief description of the register

Table 5-1: Streaming Interface Register Map Registers

Address	Name	Support	Access	Length	Description
SIRM Address + 0x00	SI Info	M	R	4	Device reports information about stream interface
SIRM Address + 0x04	SI Control	M	RW	4	Controls the stream interface
SIRM Address + 0x08	SI Required Payload Size	M	R	8	Device reports minimum required payload size with current settings
SIRM Address + 0x10	SI Required Leader Size	M	R	4	Device reports minimum required leader size
SIRM Address + 0x14	SI Required Trailer Size	M	R	4	Device reports minimum required trailer size

Table 5-1: Streaming Interface Register Map Registers(Continued)

Address	Name	Support	Access	Length	Description
SIRM Address + 0x18	SI Maximum Leader Size	M	RW*	4	Maximum leader size
SIRM Address + 0x1C	SI Payload Transfer Size	M	RW*	4	Expected Size of a single Payload Transfer
SIRM Address + 0x20	SI Payload Transfer Count	M	RW*	4	Expected Number of Payload Transfers
SIRM Address + 0x24	SI Payload Final Transfer1 Size	M	RW*	4	Size of first final Payload transfer
SIRM Address + 0x28	SI Payload Final Transfer2 Size	M	RW*	4	Size of second final Payload transfer
SIRM Address + 0x2C	SI Maximum Trailer Size	M	RW*	4	Maximum trailer size
SIRM Address + 0x30	SI Supported Payload Modes	CM	R	4	Indicates which payload streaming modes are supported
SIRM Address + 0x34	SI Payload Mode	CM	RW*	4	Controls the payload streaming mode
SIRM Address + 0x38	SI GSIRM Address	CM	R	8	Indicates the address of the GenDC SIRM Register map used for GenDC transfer modes
SIRM Address + 0x40	SI GSIRM Size	CM	R	4	Indicates the size of the GenDC SIRM Register Map

5.4.1 Description of Stream interface Registers

5.4.1.1 SI Info

Reports details about the device streaming.

Address	SIRM + 0x00
Length	4
Access Type	R
Support	M
Data Type	UINT32
Factory Default	Device specific

Bit offset (lsb << x)	Width (bits)	Description
0	1	Supports Additional Payload Modes Indicates that the *SI Supported Payload* Modes register is supported to indicate the presence of additional payload modes. When modes besides the Standard Transfer Mode are supported, this must be set to 1, otherwise 0.
1	23	reserved, must be 0.
24	8	**Payload Size Alignment** Limit SI Streaming Size Registers to $2^{(\text{Payload Size Alignment})}$ Bytes size alignment. Example: When Payload Size Alignment=0 the size alignment is 1 byte. If Payload Size Alignment=2 Alignment is 4 bytes. This means the "SI Size" register values must be a multiple of 4 bytes.

5.4.1.2 SI Control

Controls streaming operation.

Address	SIRM + 0x04
Length	4
Access Type	RW
Data Type	UINT32
Support	M
Factory Default	0

Bit offset (lsb << x)	Width (bits)	Description
0	1	**Stream Enable** When “1”, enable Streaming. Switching to “0” while a Stream is active will abort the stream. If the device cannot accept the set of parameters programmed into the streaming registers, it must return a U3V_STATUS_SI_REGISTERS_INCONSISTENT error when Stream Enable is set. If the Device Streaming Interface endpoint is halted, it must return a U3V_STATUS_DSI_ENDPOINT_HALTED error when Stream Enable is set.
1	31	Reserved, must be 0.

5.4.1.3 SI Required Payload Size

Reports the minimum required payload size the host needs to provide with the current device settings. This is the minimum the host can set to the SI registers for the total transfer size. A device may only change this as a result of a configuration change (writing a register, loading another configuration, etc.).

The device implementation must guarantee that a read of SI Required Payload Size after such a change reflects the correct new value.

R-114st
The device MUST guarantee that the value of SI Required Payload Size always matches the PayloadSize feature in the device’s XML description file.

In the case of an unknown payload length (e.g. because of streaming compressed data), the device must report a SI Required Payload Size which will always be sufficient.

R-74st
The device MUST guarantee that a read of SI Required Payload Size reflects the current device configuration.

This register is only informative for the host to allocate buffers of sufficient size.

The host may decide to use another (larger or smaller) buffer size. Therefore, the device must always observe the "SI Payload" registers when doing the actual streaming.

Address	SIRM + 0x08
Length	8
Access Type	R
Support	M
Data Type	UINT64
Factory Default	Device Specific

Bit offset (lsb << x)	Width (bits)	Description
0	64	**SI Required Payload Size** Minimum required payload size in bytes with current settings required by the device.

5.4.1.4 SI Required Leader Size

Device reports its minimum required size in bytes it needs for the leader transfer. This may change depending on current device configuration. This must not change while the device is streaming. The device may transfer less than this size.

The host must do a single transfer for the leader of at least SI Required Leader Size bytes and must also tell the device using the SI Maximum Leader Size Register.

Address	SIRM + 0x10
Length	4
Access Type	R
Support	M
Data Type	UINT32
Factory Default	Device Specific

Bit (lsb << x)	Width (bits)	offsetDescription
0	32	**SI Required Leader Size** Device reports minimum required size of leader transfer.

5.4.1.5 SI Required Trailer Size

Device reports its minimum required size in bytes it needs for the trailer transfer. This may change depending on current device configuration. This must not change while the device is streaming. The device may transfer less than this size.

The host must do a single transfer for the trailer of at least SI Required Trailer Size bytes and must also tell the device using the SI Maximum Trailer Size Register.

Address	SIRM + 0x14
Length	4
Access Type	R
Support	M
Data Type	UINT32
Factory Default	Device Specific

Bit offset (lsb << x)	Width (bits)	Description
0	32	**SI Required Trailer Size** Device reports minimum required size of trailer transfer.

5.4.1.6 SI Maximum Leader Size

Defines the maximum size of the leader the device may use.

The device may transfer up to SI Maximum Leader Size bytes for the leader. The host must make sure the leader fits within a single bulk transfer.

Size alignment restrictions from SI Info[Payload Size Alignment] must be observed. See description of SI Info register (Section 5.4.1.1).

Address	SIRM + 0x18
Length	4
Access Type	RW*
Support	M
Data Type	UINT32
Factory Default	0

Bit offset (lsb << x)	Width (bits)	Description
0	32	**SI Maximum Leader Size** Maximum leader size in bytes the device may use.

5.4.1.7 SI Payload Transfer Size

This register contains the size of regular payload bulk transfers ("equally sized transfers"). This register is used only for Standard Transfer and GenDC Container Transfer modes.

Address	SIRM + 0x1C
Length	4
Access Type	RW*
Data Type	UINT32
Support	M
Factory Default	0

Bit offset (lsb << x)	Width (bits)	Description
0	32	**SI Payload Transfer Size** Maximum Size in bytes of one payload transfer ("equally sized transfers") which a device may use.

5.4.1.8 SI Payload Transfer Count

This register contains the number of regular payload data bulk transfers. This register is used only for Standard Transfer and GenDC Container Transfer modes.

Address	SIRM + 0x20
Length	4
Access Type	RW*
Support	M
Data Type	UINT32
Factory Default	0

Bit offset (lsb << x)	Width (bits)	Description
0	32	**SI Payload Transfer Count** Count of regular payload transfers ("equally sized transfers") which a device must use.

5.4.1.9 SI Payload Final Transfer1 Size

This register contains the size of the Final Transfer 1 payload bulk transfer. This is needed in situations where the overall payload length is not divisible by SI Payload Transfer Size amounts.

Size alignment restrictions from SI Info[Payload Size Alignment] must be observed. See description of the SI Info register (Section 5.4.1.1).

This register is used only for Standard Transfer and GenDC Container Transfer modes.

Address	SIRM + 0x24
Length	4
Access Type	RW*
Support	M
Data Type	UINT32
Factory Default	0

Bit offset (lsb << x)	Width (bits)	Description
0	32	**SI Payload Final Transfer1 Size** Max size in bytes of final payload transfer 1 device may send.

5.4.1.10 SI Payload Final Transfer2 Size

This register contains the size of the Final Transfer 2 payload bulk transfer. This may be useful in situations where a scratch buffer is needed on the host to reduce the amount of data to copy.

Size alignment restrictions from SI Info[Payload Size Alignment] must be observed. See description of the SI Info register (Section 5.4.1.1).

This register is used only for Standard Transfer and GenDC Container Transfer Modes.

Address	SIRM + 0x28
Length	4
Access Type	RW*
Support	M
Data Type	UINT32
Factory Default	0

Bit offset (lsb << x)	Width (bits)	Description
0	32	**SI Payload Final Transfer2 Size** Max size in bytes of final payload transfer 2 device may send.

5.4.1.11 SI Maximum Trailer Size

Defines the maximum size of the trailer the device may use.

The device may transfer up to SI Maximum Trailer Size bytes for the trailer.

The host must make sure the trailer fits within a single bulk transfer.

Size alignment restrictions from SI Info[Payload Size Alignment] must be observed. See description of the SI Info register (Section 5.4.1.1).

Address	SIRM + 0x2C
Length	4
Access Type	RW*
Support	M
Data Type	UINT32
Factory Default	0

Bit offset (lsb << x)	Width (bits)	Description
0	32	**SI Maximum Trailer Size** Maximum trailer size in bytes the device may use.

R-120ch
Host software MUST set SI Maximum Leader Size, SI Maximum Trailer Size, SI Payload Transfer Size, SI Payload Final Transfer1 Size and SI Payload Final Transfer2 Size registers to values that are a multiple of the value specified by the device in the Payload Size Alignment register.

R-121cd
A device MUST return U3V_STATUS_SI_PAYLOAD_SIZE_NOT_ALIGNED if SI Maximum Leader Size, SI Maximum Trailer Size, SI Payload Transfer Size, SI Payload Final Transfer1 Size and SI Payload Final Transfer2 Size registers are set to values that are not multiples of the Payload Size Alignment register.

R-122cd
A device MUST allow any arbitrary transfer size written into SI Maximum Leader Size, SI Maximum Trailer Size, SI Payload Transfer Size, SI Payload Final Transfer1 Size and SI Payload Final Transfer2 Size registers if they meet the alignment criteria provided by the device. Transfers that are larger than the device requires for the given payload may be terminated early by the device while transfers that are smaller than the device requires for the given payload MUST be flagged with a U3V_STATUS_DATA_OVERRUN status.

5.4.1.12 SI Supported Payload Modes

Indicates which payload streaming modes are supported by the device. This register must be implemented if the “Supports Additional Payload Modes” bit is set in the SI Info register.

Address	SIRM + 0x30
Length	4
Access Type	R
Support	CM
Data Type	UINT32
Factory Default	Device Specific

Bit offset (lsb << x)	Width (bits)	Description
0	1	Standard Transfer (non-GenDC)
1	1	GenDC Container Transfer Mode
2	1	GenDC Serial Flow Transfer Mode
3	1	GenDC Parallel Flow Transfer Mode
4	28	Reserved (0)

R-139st
A streaming device must support at least one streaming mode.

5.4.1.13 SI Payload Mode

Controls which Payload Mode is in use. The layout of the bits in this register match the SI Supported Payload Modes register, but only one bit is allowed to be set to select a single mode.

This register must be implemented if the “Supports Additional Payload Modes” bit is set in the SI Info register.

Address	SIRM + 0x34
Length	4
Access Type	RW*
Support	CM
Data Type	UINT32
Factory Default	Device Specific

Bit offset (lsb << x)	Width (bits)	Description
0	1	Standard Transfer (non-GenDC)
1	1	GenDC Container Transfer Mode
2	1	GenDC Serial Flow Transfer Mode
3	1	GenDC Parallel Flow Transfer Mode
4	28	Reserved (0)

5.4.1.14 SI GSIRM Address

Indicates the address of the GenDC SIRM Register Map for this streaming channel. This register map is only used when one of the GenDC transfer modes is used. This register is required when a GenDC transport mode is supported by the device.

Address	SIRM + 0x38
Length	8
Access Type	R
Support	CM
Data Type	UINT64
Factory Default	Device Specific

Bit offset (lsb << x)	Width (bits)	Description
0	64	SI GSIRM Address Address of GSIRM Register Map

5.4.1.15 SI GSIRM Size

Indicates the size of the GenDC SIRM Register Map

This register is required when a GenDC transport mode is supported by the device.

Address	SIRM + 0x40
Length	4
Access Type	R
Support	CM
Data Type	UINT32
Factory Default	Device Specific

Bit offset (lsb << x)	Width (bits)	Description
0	32	SI GSIRM Size Size of GSIRM Register Map

5.4.2 GenDC Streaming Interface Register Map

This register map is required when at least one GenDC transport mode is supported by the device.

Table 5-2: GenDC Streaming Interface Register Map Registers

GSIRM Address + 0x00	SI GenDC Flow Mapping Table Address	CM	R	8	Address of the GenDC Flow Mapping Table
GSIRM Address + 0x08	SI GenDC Flow Mapping Table Size	CM	R	4	Size of the GenDC Flow Mapping Table in bytes
GSIRM Address + 0x0C	SI Number of GFSIRM Register Maps	CM	R	4	Device reports the number of GenDC Flow SIRM Register Maps supported in current configuration
GSIRM Address + 0x10	SI GFSIRM Address	CM	R	8	Address of the first GenDC Flow SIRM Register Map (GFSIRM)
GSIRM Address + 0x18	SI GFSIRM Size	CM	R	4	Indicates the size of each GenDC Flow SIRM Register Map (GFSIRM)
GSIRM Address + 0x1C	SI GenDC Descriptor Size	CM	R	4	Indicate the size of the GenDC Prefetch descriptor
GSIRM Address + 0x20	SI GenDC Descriptor Address	CM	R	8	Indicates the address of the GenDC Prefetch descriptor

5.4.2.1 SI GenDC Flow Mapping Table Address

Indicates the address of the standard GenDC Flow Mapping Table. This table is used to read the size of individual flows used for streaming.

Address	GSIRM + 0x00
Length	8
Access Type	R
Support	CM
Data Type	UINT64
Factory Default	Device Specific

Bit offset (lsb << x)	Width (bits)	Description
0	64	SI GenDC Flow Mapping Table Address Address of GenDC Flow Mapping Table

5.4.2.2 SI GenDC Flow Mapping Table Size

Indicates the size of the standard GenDC Flow Mapping Table.

Address	GSIRM + 0x08
Length	4
Access Type	R
Support	CM
Data Type	UINT32
Factory Default	Device Specific

Bit offset (lsb << x)	Width (bits)	Description
0	32	SI GenDC Flow Mapping Table Size Size of GenDC Flow Mapping Table

5.4.2.3 SI Number of GFSIRM Register Maps

Indicates the number of GenDC Flow SIRM Register Maps supported in the current configuration.

Address	GSIRM + 0x0C
Length	4
Access Type	R
Support	CM
Data Type	UINT32
Factory Default	Device Specific

Bit offset (lsb << x)	Width (bits)	Description
0	32	SI Number of GFSIRM Register Maps Number of GFSIRM register maps supported

5.4.2.4 SI GFSIRM Address

Indicates the address of the first GenDC Flow SIRM Register Map (GFSIRM).

Address	GSIRM + 0x10
Length	8
Access Type	R
Support	CM
Data Type	UINT64
Factory Default	Device Specific

Bit offset (lsb << x)	Width (bits)	Description
0	64	SI GFSIRM Address Address of first GFSIRM register map

5.4.2.5 SI GFSIRM Size

Indicates the size of each GenDC Flow SIRM Register Map (GFSIRM).

Address	GSIRM + 0x18
Length	4
Access Type	R
Support	CM
Data Type	UINT32
Factory Default	Device Specific

Bit offset (lsb << x)	Width (bits)	Description
0	32	SI GFSIRM Size Size of each GFSIRM register map

5.4.2.6 SI GenDC Descriptor Size

Indicates the size of the GenDC Prefetch descriptor. This can be used by the device to provide a sample GenDC descriptor similar to what the device will send in the stream. Software might use this to prepare the transport layer buffer configuration.

Address	GSIRM + 0x1C
Length	4
Access Type	R
Support	CM
Data Type	UINT32
Factory Default	Device Specific

Bit offset (lsb << x)	Width (bits)	Description
0	32	SI GenDC Descriptor Size Size of GenDC Prefetch Descriptor

5.4.2.7 SI GenDC Descriptor Address

Indicates the address of the GenDC Prefetch descriptor. This can be used by the device to provide a sample GenDC descriptor similar to what the device will send in the stream. Software might use this to prepare the transport layer buffer configuration.

Address	GSIRM + 0x20
Length	8
Access Type	R
Support	CM
Data Type	UINT64
Factory Default	Device Specific

Bit offset (lsb << x)	Width (bits)	Description
0	64	SI GenDC Descriptor Address Address of GenDC Prefetch Descriptor

5.4.3 GenDC Flow Streaming Interface Register Map

Table 5-3: GenDC Flow Streaming Interface Register Map Registers

GFSIRM Address + 0x00	GFSI Payload Transfer Size	CM	RW*	4	Size of a single Payload Transfer for this flow
GFSIRM Address + 0x04	GFSI Payload Transfer Count	CM	RW*	4	Number of Payload transfers for this flow
GFSIRM Address + 0x08	GFSI Payload Final Transfer1 Size	CM	RW*	4	Size of first final Payload transfer for this flow
GFSIRM Address + 0x0C	GFSI Payload Final Transfer2 Size	CM	RW*	4	Size of second final Payload transfer for this flow

5.4.3.1 GFSI Payload Transfer Size

This sets the expected size of a single payload transfer in this flow. This register serves the same purpose and behaves the same as SI Payload Transfer Size, but programmed independently for each flow 1 through N.

Address	GFSIRM + 0x00
Length	4
Access Type	RW*
Support	CM
Data Type	UINT32
Factory Default	0

Bit offset (lsb << x)	Width (bits)	Description
0	32	GFSI Payload Transfer Size Size of a single Payload Transfer for this flow

5.4.3.2 GFSI Payload Transfer Count

This sets the expected number of equally sized payload transfers in this flow. This register serves the same purpose and behaves the same as SI Payload Transfer Count, but is programmed independently for each flow 1 though N.

Address	GFSIRM + 0x04
Length	4
Access Type	RW*
Support	CM
Data Type	UINT32
Factory Default	0

Bit offset (lsb << x)	Width (bits)	Description
0	32	GFSI Payload Transfer Count Expected count of Payload Transfers for this flow

5.4.3.3 GFSI Payload Final Transfer1 Size

This sets the first final transfer for this flow. This register serves the same purpose and behaves the same as SI Payload Final Transfer1, but is programmed independently for each flow 1 though N.

Address	GFSIRM + 0x08
Length	4
Access Type	RW*
Support	CM
Data Type	UINT32
Factory Default	0

Bit offset (lsb << x)	Width (bits)	Description
0	32	GFSI Payload Final Transfer1 Size Size of first final Payload transfer for this flow

5.4.3.4 GFSI Payload Final Transfer2 Size

This sets the second final transfer for this flow. This register serves the same purpose and behaves the same as SI Payload Final Transfer2, but is programmed independently for each flow 1 though N.

Address	GFSIRM + 0x0C
Length	4
Access Type	RW*
Support	CM
Data Type	UINT32
Factory Default	0

Bit offset (lsb << x)	Width (bits)	Description
0	32	GFSI Payload Final Transfer2 Size Size of second final Payload transfer for this flow

5.4.4 Endpoint for Streaming

To stream data via a USB3 Vision device, the device must provide one or more of the following interface/endpoints:

- Device Streaming Interface Descriptor
  This interface implements exactly one Bulk IN endpoint. This endpoint is used to stream data from the device to the host.

Configuration and settings of the specific endpoints are specified in the Chapter 3.

5.5 Stream Formats

5.5.1 Data Block

Information passed on the stream channel is divided into data blocks. For instance, a USB3 Vision device fits each captured image into a data block. A receiver would thus be able to track images by looking at the block ID associated with each data block.

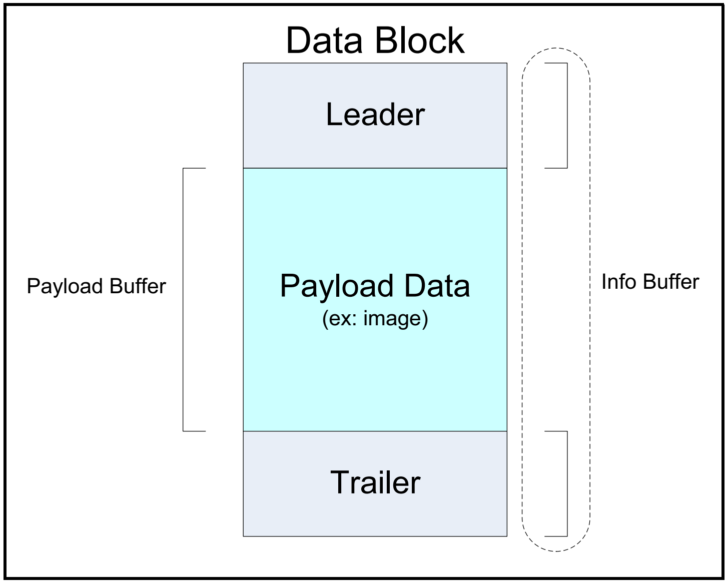
R-75st
All sections of the data block MUST use little-endian byte ordering unless explicitly defined by the specification for the payload type in use.

A data block is divided into 3 sections to provide the receiver with the information necessary for decoding.

1. Leader
2. Payload Data
3. Trailer

The leader starts the data block and provides information about the payload type of the block, including a block ID acting as a counter. The leader can also provide additional information to help decode the block. The leader is followed by the payload data, which contains the actual data to be streamed (typically the pixel data). Finally, the data block is completed by a trailer indicating the end of the data block. This is illustrated in the Figure 5-7.

Figure 5-7: Data Block



On the receiver side, the data block can be split in two sections:

1. Info Buffer
2. Payload Buffer

The info buffer receives the leader and trailer.

> **NOTE**: On the receiver side, the info buffer is typically implemented as either one buffer (leader and trailer are copied contiguously into this buffer) or two separate buffers (one for the leader and one for the trailer). The data block is divided in such a way that the leader and trailer can be easily extracted from the payload information.

The payload buffer is typically received using zero-copy and only contains the actual data (ex: the image) with no ancillary information.

Data blocks are provided sequentially on a given stream channel.

As defined in Section 5.1, the use of bulk transfer offers guaranteed delivery. The link is considered reliable and no separate mechanism is offered to support data retransmission.

5.5.2 Generic Leader

The leader is streamed before the payload data and contains ancillary data that is used to describe the content of the payload data. All payload data types are based on a generic leader shown in Table 5-4.

Table 5-4: Generic Leader Content

Offset (Bytes)	Width (Bytes)	Description
0	4	magic_key Hard-coded magic key equal to 0x4C563355 used for validation. This contains the ASCII string "U3VL" standing for USB3 Vision Leader (with 'U' in the LSB).
4	2	reserved Set to 0 on transmission, ignore on reception.
6	2	leader_size Total number of bytes in the leader.
8	8	block_id ID of the data block. Sequential and incrementing starting at 0. *block_id* wraps-around to 0 when it reaches its maximum value. For a transmitter, the *block_id* is reset to 0 each time the stream channel is enabled.
16	2	**reserved** Set to 0 on transmission, ignore on reception.
18	2	**payload_type** Refer to Table 5-6 for a list of supported payload types.
20	N	Additional info for the leader. This is payload type specific. Refer to the specific payload type for additional information. (Section 5.5.5)

R-76st
The leader MUST be located before the payload data.

In order to enable a receiver to determine if data blocks have been lost at the beginning of a new connection (i.e. when the stream channel is enabled), it is necessary for the transmitter to reset the *block_id* so it starts with a known value.

R-77st
A transmitter MUST reset the *block_id* field associated with a stream channel to a value of 0 when this stream channel is opened by the host via Stream Enable in the SI Control Register

R-78st
Successive data blocks that have been acquired MUST present successive *block_id* values, incrementing by one.

Note the above requirement applies to acquired blocks. Transmitted blocks are allowed to use non-successive block_ids when some of them are dropped due to an overflow condition, as explained in Section 5.5.4.

5.5.3 Generic Trailer

The trailer contains ancillary information appearing after the payload data and provides additional information about the state of the payload data once it has been transferred. All payload types are based on a generic trailer shown in Table 5-5.

Table 5-5: Generic Trailer Content

Offset (Bytes)	Width (Bytes)	Description
0	4	**magic_key** Hard-coded magic key equal to 0x54563355 used for validation. This contains the ASCII string “U3VT” standing for USB3 Vision Trailer (with ‘U’ in the LSB).
4	2	**reserved** Set to 0 on transmission, ignore on reception.
6	2	**trailer_size** Total number of bytes in the Trailer.
8	8	**block_id** ID of the data block. Sequential and incrementing starting at 0. *block_id* wraps-around to 0 when it reaches its maximum value. For a transmitter, the *block_id* is reset to 0 each time the stream channel is opened.
16	2	**status** Status of the streaming operation. (see Table 4-3)
18	2	**reserved** Set to 0 on transmission, ignore on reception.
20	8	**valid_payload_size** Size in bytes of valid data in the payload data section. Additional bytes that might be transmitted in the payload data section must be considered as irrelevant; they are not counted by this field.
28	N	Additional info for the trailer. This is payload type specific. Refer to the specific payload type for additional information. (see Section 5.5.5)

R-79st
The trailer MUST be transmitted after the payload data.

R-80st
Every payload data block which is transmitted MUST have a corresponding leader and trailer transmitted.

The leader and trailer must always be transmitted as a pair, even if some payload data is missing, as explained in the next section.

5.5.4 Device Overflow Handling

It is possible for a device to discard payload data when more data is acquired than can be transmitted or for some other reasons. The following scenarios exist:

5.5.4.1 Discarding Some Payload Data

After a leader has been transmitted, if payload data has been discarded in the data block of this leader, then all subsequent payload data within that data block is transmitted as zero-length or short packets to maintain synchronization with the receiver before sending the trailer. The U3V_STATUS_DATA_DISCARDED status code is put in the trailer to announce the partial discarding of data. Since the leader is transmitted, the corresponding trailer must also be transmitted.

A device may choose to use the mechanism described above even if all the payload data has been discarded. This means transmitting only the leader, followed by zero-length or short packets for payload data, followed by the trailer. In this case, there will be no gap in *block_id* values.

5.5.4.2 Payload Data Overflow

After a leader has been transmitted, if payload data is discarded after the completion of the data block due to the data block being too small, the U3V_STATUS_DATA_OVERRUN status code is put in the trailer to announce that the device had more data than the data block could hold. Since the leader is transmitted, the corresponding trailer must also be transmitted.

5.5.4.3 Discarding a Full Data Block

When a full data block is discarded by the transmitter, the *block_id* of that block is skipped to allow the receiver to determine how many data blocks have been discarded. The receiver is responsible for monitoring the *block_id* to determine if one or more data blocks have been discarded.

5.5.5 Payload Types

To efficiently transport information, USB3 Vision defines several payload types that can be streamed out of a transmitter. These payload types are listed in Table 5-6.

Table 5-6: Payload Types

Payload Type	Value	Description
Image	0x0001	Uncompressed image data.
Image Extended Chunk	0x4001	Support for Image extended chunk data. In this case, the Image must be the first chunk in the payload data.
Chunk	0x4000	Generic chunk mode where the first chunk is not derived from any payload type. Used to transmit any combination of chunks.
GenDC Container	0x1001	Used by GenDC Container Transfer Mode, the payload data contains an entire GenDC Container
GenDC Component Data	0x1002	Used by both GenDC Serial Flow and Parallel Flow Transfer modes, the payload data contains only the GenDC Component Data payload, and the GenDC Descriptor is contained within the Leader and/or Trailer

USB3 Vision supports self-described data. Therefore, the leader of all supported non-GenDC payload types contains all the information necessary for the receiver to correctly decode the payload data. For GenDC formats, the GenDC Descriptor that is transferred is used to decode the payload data.

5.5.5.1 Image Payload Type

This payload type is used to transmit uncompressed images in raster-scan format.

CR-81st
If the image payload type is supported, a stream using the image payload type MUST provide pixel data in raster-scan format.

This means the image is reconstructed, if necessary, in the transmitter memory before being transmitted from left to right, then top to bottom. This is typical with single-tap sensor.

For images, multiple regions of interest (ROI) from the same frame must be provided on the stream channel as separate data blocks. Each ROI must be transmitted with a different block ID. In this case, identical timestamps must be used to facilitate matching the ROI to a given exposure. The leader indicates ROI offset in the full size image.

CR-82st
If an image source supports multiple ROIs, overlapping data from multiple ROIs MUST be provided with each ROI so that each image is complete in its data block.

5.5.5.1.1 Image Leader

Each image data block starts with a leader.

CR-84st
If the image payload type is supported, its leader MUST follow the layout shown in Table 5-7.

Table 5-7: Image Leader Content

Offset (Bytes)	Width (Bytes)	Description
0	4	**magic_key** Refer to Generic Leader.
4	2	**reserved** Refer to Generic Leader.
6	2	**leader_size** Refer to Generic Leader.
8	8	**block_id** Refer to Generic Leader.
16	2	**reserved** Refer to Generic Leader.
18	2	**payload_type** Refer to Generic Leader. Set to 0x0001 (Image)
20	8	**timestamp** 64-bit timestamp, in ns, representing when the block of data was generated. Image blocks captured at the same time (such as multiple ROIs from the same exposure) use the same timestamp. Images captured at different times must use different timestamps.

Table 5-7: Image Leader Content(Continued)

Offset (Bytes)	Width (Bytes)	Description
28	4	**pixel_format** This field gives the pixel format of payload data. See Section 5.5.7.
32	4	**size_x** Width in pixels of the image transported in the payload data section of this data block.
36	4	**size_y** Height in lines of the image transported in the payload data section of this data block. For variable frame size, the transmitter may transmit less than this value. If this is the case, the trailer must provide the actual number of lines transmitted in its *size_y* field.
40	4	**offset_x** Horizontal offset in pixels from image origin. Used for ROI support. When no ROI is defined this field must be set to 0. For Color Filter Array (CFA), it is recommended to have *offset_x* increment by the full width of the CFA. This is to ensure the *pixel_format* field matches the PixelFormat feature configured by the receiver. Other pixel formats might pose restrictions on the value taken by this field.
44	4	**offset_y** Vertical offset in lines from image origin. Used for ROI support. When no ROI is defined this field must be set to 0. For CFA, it is recommended to have *offset_y* increment by the full height of the CFA. This is to ensure the *pixel_format* field matches the PixelFormat feature configured by the receiver. Other pixel formats can pose restrictions on the value taken by this field.
48	2	**padding_x** Horizontal padding expressed in bytes. Number of extra bytes provided at the end of each line (i.e. after the last pixel of the line) to facilitate image alignment in buffers. This can be used to have 32-bit aligned image lines. Set to 0 when no horizontal padding is used.
50	2	**reserved** Set to 0 on transmission, ignore on reception.

CR-83st

If an image source supports multiple ROIs, different ROI from the same exposure MUST use the same timestamp.

CR-110st

Images that are exposed at different times, regardless of internal timer resolution, MUST use unique timestamps for each image.

5.5.5.1.2 Image Payload Data

The Image Payload Data directly represents the image information (i.e. the pixels of the images). It is provided as a contiguous buffer using the Pixel Format indicated in the image leader.

5.5.5.1.3 Image Trailer

An image data block terminates with a trailer section.

CR-85st
If the image payload type is supported, its trailer MUST follow the layout shown in Table 5-8.

Table 5-8: Image Trailer Content

Offset (Bytes)	Width (Bytes)	Description
0	4	**magic_key** Refer to Generic Trailer.
4	2	**reserved** Refer to Generic Trailer.
6	2	**trailer_size** Refer to Generic Trailer.
8	8	**block_id** Refer to Generic Trailer.
16	2	**status** Refer to Generic Trailer.
18	2	**reserved** Refer to Generic Trailer.
20	8	**valid_payload_size** Refer to Generic Trailer.
28	4	**size_y** This field is the actual height, in lines, for this particular data block. This is done to support variable frame size image sources once the actual number of lines provided has been confirmed. Only image height may be variable (i.e. image width must be fixed). This value must be smaller or equal to the *size_y* announced in the leader.

5.5.5.1.4 Variable Image Payload Size

For variable frame size, the transmitter can transmit less data than indicated by the size_y field of the image leader. When this is the case, the size_y field of the trailer provides the actual number of lines transmitted for this image.

Support of variable image payload size is achieved by:

1. Sending zero-length or short packets to cover for image information not present due to the smaller actual image height than announced in the leader.
2. Setting the *size_y* field of the trailer to the actual image height.

CR-86st
When a transmitter supports variable payload size, it MUST indicate the actual *size_y* of the image in the trailer. This value MUST be smaller or equal to the size_y field of the image leader.

> **NOTE**: The size_y field of the image leader provides the worst case image height that can be transmitted (i.e. largest number of lines). The worst case payload size must be accounted for by the mandatory PayloadSize feature.

5.5.5.2 Image Extended Chunk Payload Type

Chunks can be appended to the image payload type by setting bit 14 of the *payload_type* field (the extended chunk mode bit). This metadata is attached as Chunk Data following the rules presented in Section 5.5.6.

> **NOTE**: Extended chunk mode is defined in a generic way so it can be used for future payload types other than the Image payload type. Bit 14 of the *payload_type* field is used to indicate that extended chunk is activated.

When extended chunk mode is selected:

1. The leader remains the same with the exception of the extended chunk mode bit.
2. The payload data contains chunk data, with the first chunk being equivalent to the selected payload format (i.e. image payload type).
3. The trailer must be present and it is appended with one additional field:
    a. *chunk_layout_id*

5.5.5.2.1 Image Extended Chunk Leader

Each Image Extended Chunk data block starts with a leader.

CR-87st
When image extended chunk payload type is supported and enabled, then the transmitter MUST provide a leader that follows the layout shown in Table 5-9.

Table 5-9: Image Extended Chunk Leader Content

Offset (Bytes)	Width (Bytes)	Description
0	4	**magic_key** Refer to Generic Leader.
4	2	**reserved** Refer to Generic Leader.
6	2	**leader_size** Refer to Generic Leader.
8	8	**block_id** Refer to Generic Leader.
16	2	**reserved** Refer to Generic Leader.
18	2	payload_type Refer to Generic Leader. Set to 0x4001 (Image Extended Chunk).
20	8	**timestamp** Refer to Image Leader.

Table 5-9: Image Extended Chunk Leader Content(Continued)

Offset (Bytes)	Width (Bytes)	Description
28	4	pixel_format Refer to Image Leader.
32	4	**size_x** Refer to Image Leader.
36	4	size_y Refer to Image Leader.
40	4	offset_x Refer to Image Leader.
44	4	offset_y Refer to Image Leader.
48	2	padding_x Refer to Image Leader.
50	2	reserved Set to 0 on transmission, ignore on reception.

5.5.5.2.2 Image Extended Chunk Payload Data

The Image Extended Chunk Payload Data directly represents the image information (i.e. the pixels of the images) in the first chunk followed by other chunks. The image is provided as a contiguous data using the Pixel Format indicated in the image leader.

5.5.5.2.3 Image Extended Chunk Trailer

An Image Extended Chunk data block terminates with a trailer section.

CR-88st

When image extended chunk mode is supported and enabled, then the transmitter MUST provide a trailer that follows the layout shown in Table 5-10.

Table 5-10: Image Extended Chunk Trailer Content

Offset (Bytes)	Width (Bytes)	Description
0	4	**magic_key** Refer to Generic Trailer.
4	2	**reserved** Refer to Generic Trailer.
6	2	**trailer_size** Refer to Generic Trailer.
8	8	**block_id** Refer to Generic Trailer.
16	2	**status** Refer to Generic Trailer.

Table 5-10: Image Extended Chunk Trailer Content(Continued)

Offset (Bytes)	Width (Bytes)	Description
18	2	**reserved** Refer to Generic Trailer.
20	8	**valid_payload_size** Refer to Generic Trailer.
28	4	**size_y** Refer to Image Trailer.
32	4	**chunk_layout_id** This field serves as an indicator to notify the host the chunk layout has changed and the receiver should re-parse the chunk layout in the buffer. It is an arbitrary value chosen by the camera and changes whenever the layout is changed. A value of 0 is reserved and indicates this dynamic functionality is not support. The algorithm used to compute the *chunk_layout_id* is left as quality of implementation.

CR-113st

When a chunk layout (availability or position of individual chunks) changes since the last block, the transmitter MUST change the *chunk_layout_id*. As long as the chunk layout remains the same, the transmitter MUST keep the *chunk_layout_id* identical. When switching back to a layout which was already used before, the transmitter can use the same ID that was used then or use a new ID. A *chunk_layout_id* value of 0 is invalid. It is reserved to be used by transmitters not supporting the layout ID functionality. If the transmitter does not support the *chunk_layout_id* functionality, it MUST set this field to 0 for all blocks.

5.5.5.2.4 Variable Image Extended Chunk Payload Size

It is possible for Image Extended Chunk Payload type to transmit images of variable size (in the first chunk) by respecting the same provision indicated in Section 5.5.5.1.4 for Image Payload Type.

5.5.5.3 Chunk Payload Type

This payload type is used to transmit chunks.

5.5.5.3.1 Chunk Leader

Each Chunk data block starts with a leader.

CR-89st

If the Chunk payload type is supported, its leader MUST follow the layout shown in Table 5-11.

Table 5-11: Chunk Leader Content

Offset (Bytes)	Width (Bytes)	Description
0	4	**magic_key** Refer to Generic Leader.

Table 5-11: Chunk Leader Content(Continued)

Offset (Bytes)	Width (Bytes)	Description
4	2	**reserved** Refer to Generic Leader.
6	2	**leader_size** Refer to Generic Leader.
8	8	**block_id** Refer to Generic Leader.
16	2	**reserved** Refer to Generic Leader.
18	2	**payload_type** Refer to Generic Leader. Set to 0x4000 (Chunk).
20	8	**timestamp** Refer to Image Leader.

5.5.5.3.2 Chunk Payload Data

The Chunk Payload Data directly represents the chunk information. It is provided as a contiguous buffer hosting the various chunks present in the Data Block. For further information see Section 5.5.6.

5.5.5.3.3 Chunk Trailer

A chunk data block terminates with a trailer section.

CR-90st

If the chunk payload type is supported, its trailer MUST follow the layout shown in Table 5-12.

Table 5-12: Chunk Trailer Content

Offset (Bytes)	Width (Bytes)	Description
0	4	**magic_key** Refer to Generic Trailer.
4	2	**reserved** Refer to Generic Trailer.
6	2	**trailer_size** Refer to Generic Trailer.
8	8	**block_id** Refer to Generic Trailer.
16	2	**status** Refer to Generic Trailer.
18	2	**reserved** Refer to Generic Trailer.

Table 5-12: Chunk Trailer Content(Continued)

Offset (Bytes)	Width (Bytes)	Description
20	8	**valid_payload_size** Refer to Generic Trailer.
28	4	**chunk_layout_id** Refer to Image Extended Chunk Payload type.

5.5.5.4 GenDC Container Type

This payload type is used to transmit a complete GenDC Container as a monolithic payload. It is used by the GenDC Container Transfer Mode. In this format, the GenDC Descriptor is part of the payload.

5.5.5.4.1 GenDC Container Leader

Table 5-13: GenDC Container Leader Content

Offset (Bytes)	Width (Bytes)	Description
0	4	**magic_key** Refer to Generic Leader.
4	2	**reserved** Refer to Generic Leader.
6	2	**leader_size** Refer to Generic Leader.
8	8	**block_id** Refer to Generic Leader.
16	2	**reserved** Refer to Generic Leader.
18	2	**payload_type** Refer to Generic Leader. Set to 0x1001 (GenDC Container)
20	8	**timestamp** 64-bit timestamp, in ns, representing when the block of data was generated.

5.5.5.4.2 GenDC Container Payload

The GenDC Container Payload Data directly represents a complete GenDC Container. It is provided as a contiguous buffer with the GenDC Descriptor and associated payloads contained within it.

5.5.5.4.3 GenDC Container Trailer

Table 5-14: GenDC Container Trailer Content

Offset (Bytes)	Width (Bytes)	Description
0	4	**magic_key** Refer to Generic Trailer.
4	2	**reserved** Refer to Generic Trailer.
6	2	**trailer_size** Refer to Generic Trailer.
8	8	**block_id** Refer to Generic Trailer.
16	2	**status** Refer to Generic Trailer.
18	2	**reserved** Refer to Generic Trailer.
20	8	**valid_payload_size** Refer to Generic Trailer.

5.5.5.5 GenDC Component Data Type

This payload type is used to transmit a complete GenDC Container as multiple flows using the GenDC Serial Flow or Parallel Flow Transfer modes. In this format, the GenDC Descriptor is attached as part of the Leader and/or Trailer.

5.5.5.5.1 GenDC Component Data Leader

Table 5-15: GenDC Component Data Leader Content

Offset (Bytes)	Width (Bytes)	Description
0	4	**magic_key** Refer to Generic Leader.
4	2	**reserved** Refer to Generic Leader.
6	2	**leader_size** Total number of bytes in the leader**, including the GenDC Descriptor**.
8	8	**block_id** Refer to Generic Leader.
16	2	**reserved** Refer to Generic Leader.
18	2	**payload_type** Refer to Generic Leader. Set to 0x1002 (GenDC Component Data)
20	4	**descriptor_size** Size of the GenDC descriptor in bytes. Must be set to zero when no GenDC descriptor is transmitted in the leader packet.
24	N	**gendc_descriptor** The GenDC descriptor data.
24+N	P	**post_padding** Must be set to zero. May be used to align the total size to an arbitrary boundary defined by the device.

5.5.5.5.2 GenDC Component Data Payload

The GenDC Component Data Payload contains all the payload data for a given GenDC flow, except for the GenDC descriptor.

5.5.5.5.3 GenDC Component Data Trailer

Table 5-16: GenDC Component Data Trailer Content

Offset (Bytes)	Width (Bytes)	Description
0	4	**magic_key** Refer to Generic Trailer.
4	2	**reserved** Refer to Generic Trailer.
6	2	**trailer_size** Refer to Generic Trailer.
8	8	**block_id** Refer to Generic Trailer.
16	2	**status** Refer to Generic Trailer.
18	2	**reserved** Refer to Generic Trailer.
20	8	**valid_payload_size** Refer to Generic Trailer. This must be the total payload size of all flows transmitted.
28	4	**number_of_flows** Indicates the number of flows transferred in this block. Must always be equal to the number of flows indicated in use by the flow table.
32+(N*8)	8	**flow_valid_payload_size** Indicates the valid transferred payload bytes for each flow N.
32+(**number_of_flows***8)	4	**final_descriptor_size** **Size** of the final GenDC descriptor in bytes. Must be set to zero when no final GenDC descriptor is transmitted in the trailer packet.
32+(**number_of_flows***8) + 4	4	**padding** Must be set to zero.
32+(**number_of_flows***8) + 8	N	**final_gendc_descriptor** The final GenDC descriptor data.
32+(**number_of_flows***8) + 8 + N	P	**post_padding** Must be set to zero. May be used to align the total size to an arbitrary boundary defined by the device.

5.5.5.5.4 Variable GenDC Payload Size

For variable GenDC payload size in each flow, the transmitter can transmit less data than it indicates as the maximum size indicated by the Flow Mapping Table.

The host can determine how much valid data was transferred for each flow by reading the **flow_valid_payload_size** fields in the trailer for each flow.

The device can transmit less data for each flow by sending zero-length or short packets within the payload sequence for each flow.

5.5.6 Chunk Data

Chunks are tagged blocks of data that can be grouped within a data block to transmit metadata. Example chunks are:

- Image
- Data extracted from image
- AOI / pixel format
- State of I/O lines
- Exposure time

Each chunk consists of the chunk data and a trailing tag. The tag contains:

- A unique chunk identifier, which identifies the structure of the chunk data and the chunk features associated with this chunk.
- The length of the chunk data.

As the chunk tags (*chunk ID* and *length* fields) are headers embedded in the payload of chunk format block, their byte order is little-endian.

CR-91st
If Chunk Data is supported, all chunk tags MUST use little-endian byte order.

Endian type of the chunk data itself is defined in the XML device description file (if applicable to the data type). Table 5-17 describes the general structure of any chunk.

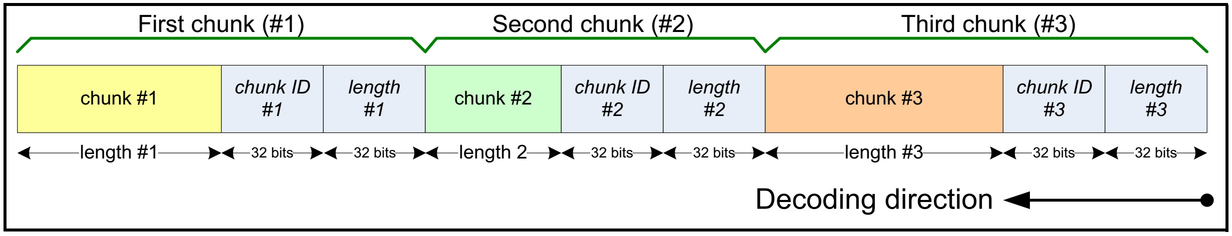
Table 5-17: Chunk Data Content

Offset (Bytes)	Width (Bytes)	Description
0	N	data The data that the chunk is transporting. This section must be a multiple of 4 bytes. If it is not, the data has to be padded with zeros to a multiple of four bytes such that N is a multiple of 4 bytes. This ensures *chunk ID* and *length* fields are 32-bit aligned within the chunk.
N	4	chunk ID The chunk identifier.
N+4	4	length The length of the data (in bytes, must be a multiple of 4).

The chunk structure with the tag behind the data allows transmitters to output data in chunk format as a stream, even in cases of variable length data (as occur in linescan applications). Decoding starts from the end of the buffer.

A chunk data block consists of a sequence of chunks in chunk data format as shown in Figure 5-8.

Figure 5-8: Chunk Data Diagram



A block in chunk format is decoded with the help of a GenICam chunk parser. To let the chunk parser distinguish between the chunks added by multiple enabled chunk features, each chunk carries a *chunk ID*. The *ID* for each chunk is transferred just before the chunk's *length* information.

CO-92st
If Chunk Data is supported, the chunks SHOULD be defined in the XML device description file.

The chunk parser decodes a block in chunk format beginning with the last received data walking to the beginning of the block. The chunk parser exposes the chunk data as read-only features.

5.5.6.1 Byte Ordering Example for Chunk Data

Table 5-18 shows the breakdown of the payload of a specific USB3 Vision block in chunk format into byte fields.

Assume the following parameters stored in 3 chunks:

1. Image with a 1 x 1 AOI

- Mono8 gray value: 0x83
- Chunk ID:0x617D18DB

2. a 32-bit frame counter with the value 0x00000008.

- Chunk ID: 0x8C0F1CD8

3. Image information

- ChunkOffsetX: 3
- ChunkOffsetY: 6
- ChunkWidth: 1
- ChunkHeight: 1

- Chunk ID: 0x230F1C23

Table 5-18: Example of Chunk Content

	Byte 0	Byte 1	Byte 2	Byte 3	Description
Image chunk	0x83	0x00	0x00	0x00	Image data + padding bytes
	0xdb	0x18	0x7d	0x61	Chunk ID for image data
	0x04	0x00	0x00	0x00	Length of the chunk
Frame counter chunk	0x08	0x00	0x00	0x00	Frame counter
	0xd8	0x1c	0x0f	0x8c	Chunk ID for frame counter
	0x04	0x00	0x00	0x00	Length of the chunk
Image info chunk	0x03	0x00	0x06	0x00	ChunkOffsetX, ChunkOffsetY
	0x01	0x00	0x01	0x00	ChunkWidth, ChunkHeight
	0x23	0x1c	0x0f	0x23	Chunk ID for image info
	0x08	0x00	0x00	0x00	Length of the chunk

5.5.6.2 GenICam Chunk Definition Example

The GenICam fragment to define the ImageInfo chunk in the above example is shown in Table 5-19:

Table 5-19: GenICam Chunk Definition Example

```
<RegisterDescription>
…
<MaskedIntReg Name="ChunkOffsetXValue">
<Address>0x00</Address>
<Length>4</Length>
<AccessMode>RO</AccessMode>
<pPort>ImageInfoPort</pPort>
<LSB>0</LSB>
<MSB>15</MSB>
<Sign>Unsigned</Sign>
<Endianess>LittleEndian</Endianess>
</MaskedIntReg>
<MaskedIntReg Name="ChunkOffsetYValue">
<Address>0x00</Address>
<Length>4</Length>
<AccessMode>RO</AccessMode>
<pPort>ImageInfoPort</pPort>
<LSB>16</LSB>
<MSB>31</MSB>
<Sign>Unsigned</Sign>
<Endianess>LittleEndian</Endianess>
</MaskedIntReg>
<MaskedIntReg Name="ChunkWidthValue">
<Address>0x04</Address>
```

Table 5-19: GenICam Chunk Definition Example(Continued)

```
<Length>4</Length>
<AccessMode>RO</AccessMode>
<pPort>ImageInfoPort</pPort>
<LSB>0</LSB>
<MSB>15</MSB>
<Sign>Unsigned</Sign>
<Endianess>LittleEndian</Endianess>
</MaskedIntReg>
<MaskedIntReg Name="ChunkHeightValue">
<Address>0x04</Address>
<Length>4</Length>
<AccessMode>RO</AccessMode>
<pPort>ImageInfoPort</pPort>
<LSB>16</LSB>
<MSB>31</MSB>
<Sign>Unsigned</Sign>
<Endianess>LittleEndian</Endianess>
</MaskedIntReg>
<Port Name="ImageInfoPort">
<ChunkID>230f1c23</ChunkID>
</Port>
</RegisterDescription>
```

Refer to the GenICam standard for additional information about Chunk Data.

5.5.7 Pixel Formats

This section describes the various pixel formats supported by USB3 Vision. These layouts are based on the GenICam Pixel Format Naming Convention (PFNC). The mechanism to select the pixel format is supplied in the XML device description file.

CR-93st
USB3 Vision transmitters SHOULD use the GenICam Pixel Format Naming Convention to name and describe pixels formats where applicable.

To optimize processing time on the receiver, the pixel data is little-endian by default.

USB3 Vision uses **Image Padding** as defined in the Pixel Format Naming Convention. Image Padding requires that no artificial padding is inserted at the end of a line. Therefore, for some packed pixel formats, it is possible that pixels from different lines are combined in the same bytes.

NOTE: To avoid this grouping of pixels from different lines, a camera can choose to use a coarser increment for the Width standard feature.

Each Pixel Format has an associated pixel_format_id value that is used in the Image Payload leader to announce the Pixel Format of the payload data.

O-94s
A transmitter or a receiver SHOULD support a subset of the Pixel Formats listed

in Table 5-20.

Table 5-20: Recommended Pixel Formats

Pixel Name	pixel_format_id	Note
Mono1p	0x01010037	Combine 8 monochrome pixels in one byte
Mono2p	0x01020038	Combine 4 monochrome pixels in one byte
Mono4p	0x01040039	Combine 2 monochrome pixels in one byte
Mono8	0x01080001	8-bit pixel in one byte
Mono10	0x01100003	10-bit pixel padded to 16 bits
Mono10p	0x010a0046	It takes 4 pixels (packed over 5 bytes) to re-align on byte boundary
Mono12	0x01100005	12-bit pixel padded to 16 bits
Mono12p	0x010c0047	It takes 2 pixels (packed over 3 bytes) to re-align on byte boundary
Mono14	0x01100025	14-bit pixel padded to 16 bits
Mono16	0x01100007	16-bit pixel in two bytes
BayerGR8	0x01080008	8-bit pixel in one byte
BayerGR10	0x0110000C	10-bit pixel padded to 16 bits
BayerGR10p	0x010A0056	It takes 4 pixels (packed over 5 bytes) to re-align on byte boundary
BayerGR12	0x01100010	12-bit pixel padded to 16 bits
BayerGR12p	0x010C0057	It takes 2 pixels (packed over 3 bytes) to re-align on byte boundary
BayerGR16	0x0110002E	16-bit pixel in two bytes
BayerRG8	0x01080009	8-bit pixel in one byte
BayerRG10	0x0110000D	10-bit pixel padded to 16 bits
BayerRG10p	0x010A0058	It takes 4 pixels (packed over 5 bytes) to re-align on byte boundary
BayerRG12	0x01100011	12-bit pixel padded to 16 bits
BayerRG12p	0x010C0059	It takes 2 pixels (packed over 3 bytes) to re-align on byte boundary
BayerRG16	0x0110002F	16-bit pixel in two bytes
BayerGB8	0x0108000A	8-bit pixel in one byte
BayerGB10	0x0110000E	10-bit pixel padded to 16 bits
BayerGB10p	0x010A0054	It takes 4 pixels (packed over 5 bytes) to re-align on byte boundary
BayerGB12	0x01100012	12-bit pixel padded to 16 bits
BayerGB12p	0x010C0055	It takes 2 pixels (packed over 3 bytes) to re-align on byte boundary
BayerGB16	0x01100030	16-bit pixel in two bytes
BayerBG8	0x0108000B	8-bit pixel in one byte

Table 5-20: Recommended Pixel Formats(Continued)

Pixel Name	pixel_format_id	Note
BayerBG10	0x0110000F	10-bit pixel padded to 16 bits
BayerBG10p	0x010A0052	It takes 4 pixels (packed over 5 bytes) to re-align on byte boundary
BayerBG12	0x01100013	12-bit pixel padded to 16 bits
BayerBG12p	0x010C0053	It takes 2 pixels (packed over 3 bytes) to re-align on byte boundary
BayerBG16	0x01100031	16-bit pixel in two bytes
BGR8	0x02180015	8-bit color component in one byte
BGR10	0x02300019	10-bit color component padded to 16 bits
BGR10p	0x021E0048	It takes 4 pixels (packed over 15 bytes) to re-align on byte boundary
BGR12	0x0230001B	12-bit color component padded to 16 bits
BGR12p	0x02240049	It takes 2 pixels (packed over 9 bytes) to re-align on byte boundary
BGR14	0x0230004A	14-bit color component padded to 16 bits
BGR16	0x0230004B	16-bit color component in two bytes
BGRa8	0x02200017	8-bit color component in one byte
BGRa10	0x0240004C	10-bit color component padded to 16 bits
BGRa10p	0x0228004D	It takes 1 pixel (packed over 5 bytes) to re-align on byte boundary
BGRa12	0x0240004E	12-bit color component padded to 16 bits
BGRa12p	0x0230004F	It takes 1 pixel (packed over 6 bytes) to re-align on byte boundary
BGRa14	0x02400050	14-bit color component padded to 14 bits
BGRa16	0x02400051	16-bit color components in two bytes
YCbCr8	0x0218005B	8-bit color component in one byte
YCbCr422_8	0x0210003B	8-bit color component in one byte, YCbYCr component sequence
YCbCr411_8	0x020C005A	8-bit color component in one byte, YYCbYYCr component sequence

* some additional padding bits might be necessary if the number of pixels is not a multiple of the “# of pixels to realign” column.

Custom pixel formats can be realized by following the rules listed in the PFNC for 32-bit pixel format ID values. This is achieved by setting the most significant bit of the pixel_format_id to 1 and using the remaining 31 bits as manufacturer-specific. Custom pixel formats should only be used when no standard pixel format exists as they prevent interoperability.

5.5.8 Bandwidth Allocation

Since a USB3 Vision device relies on bulk transfer to stream data, a specific method should be provided to throttle the maximum bandwidth usage on the device's connection. This is accomplished by using two features offered by GenICam Standard Feature Naming Convention (SFNC).

1. The maximum bandwidth allocated to the device for streaming is provided by the **DeviceLinkThroughputLimit** feature.
2. Enabling bandwidth management is controlled through the **DeviceLinkThroughputLimitMode** feature.

> **O-95st**
> A USB3 Vision transmitter SHOULD support both DeviceLinkThroughputLimit and DeviceLinkThroughputLimitMode features, as specified by GenICam SFNC.

Refer to the GenICam SFNC for the definition of **DeviceLinkThroughputLimit** and **DeviceLinkThroughputLimitMode**. USB3 Vision devices have a single link (as defined by SFNC). As such, no link selector is needed for these 2 features.

6.0 Mechanical

6.1 Objective

The mechanical specification has been developed with the following objectives:

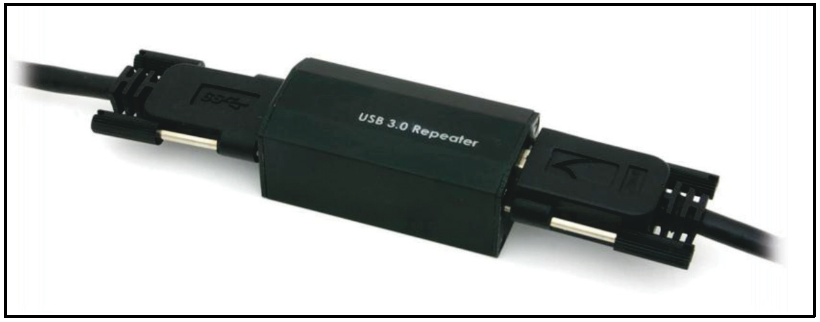
- Specification of locking connectors which are necessary for the machine vision market, but are not considered part of the USB standard
- Compatibility with the USB 3.2 standard
- Compatibility with existing device designs
- Interoperability between cables and devices
- Extended cable length and flexibility

Please note that electrical and environmental requirements are also part of the mechanical standard.

6.2 Rationale

Secure connections are especially important or required in machine vision and other high movement/force applications. In some applications, the mating force between the plug and receptacle is not sufficient to prevent the plug from moving or unplugging and therefore losing connection. This is especially important for thicker cables or harsh environments. Locking screws are widely used to secure the connection between the plug and receptacle. Figure 6-1 shows a typical example of the use of locking screws.

Figure 6-1: USB 3.0 Standard-A and Standard-B Locking Connectors with Repeater



The Micro-B connector is first defined by the USB 3.0 standard for small handheld devices and is used for most USB3 Vision devices. Interconnection devices used in machine vision systems like host cards, hubs and repeaters typically use Standard-A, Standard-B, or Powered-B connectors. All of these connectors potentially need to be secured.

With the growing use of Type-C connectors in consumer applications, more industrial use is expected.

To accomplish extended needs of industrial applications it might be necessary to use cables that are larger in diameter and therefore do not fit the standard maximum overmold dimensions. For such applications an additional set of maximum overmold dimensions is specified as **USB3 Vision Standard+**.

6.3 Locking Connectors

The following locking connectors are defined by this standard:

USB3 Vision Standard connectors:

- Micro-B
- Standard-A
- Standard-B / Powered-B
- Type-C

USB3 Vision Standard+ connectors:

- Micro-B+
- Standard-A+
- Type-C+

The Standard+ connectors have bigger overmold dimensions that are required by larger diameter cables. Note that this extended overmold will lead to problems if there is not enough space available and may make stacking impossible. But to maintain backwards compatibility, device support for Standard+ connectors is optional.

In this standard, the Standard-B and Powered-B connectors are considered the same as they share common mechanical dimensions. These dimensions already fit the larger cable diameters and therefore it is not necessary to provide Standard+ dimensions.

In general, the USB standard defines only a minimum distance from the tip of the connector to the overmold. For locking connectors it is also necessary to specify a maximum distance. Otherwise, the mechanical force that can be applied when tightening the screws can damage the device.

O-96md
To provide plug-and-play experience to the user, USB3 Vision devices should use the USB3 Vision locking connectors.

R-97mi
USB3 Vision interconnection devices like hubs and repeaters MUST use the USB3 Vision locking connectors.

The customer experience with USB3 Vision will be tarnished if the customer breaks a device while screwing in a USB3 Vision connector. Therefore, if a custom locking mechanism is used it must not be confused with the USB3 Vision defined mechanism.

CR-98mdi
If a USB3 Vision device uses a custom locking mechanism it MUST not be damaged by attaching a USB3 Vision connector.

R-99mdi
A USB3 Vision device utilizing A, B, or Micro-B connectors may not place any thread in the location of the M2 threads as defined by Figure 6-3, unless the

thread is M3 or larger.

All tolerances in drawings in this chapter are +/- 0.10mm unless otherwise specified.

6.3.1 Screw and Thread Position

See Figure 6-2 and Figure 6-3 for the positioning of the locking screws and threads. These are centered relative to the plug and the socket to ensure proper mating regardless of the cut out and overmold dimensions.

Figure 6-2: Standard-A, Standard/Powered-B, and Micro-B Plugs With Screw Positions

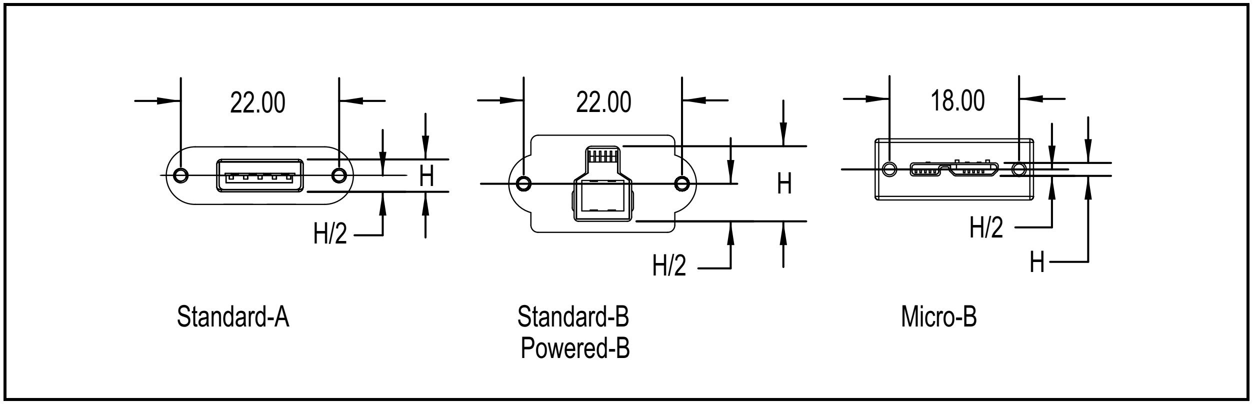
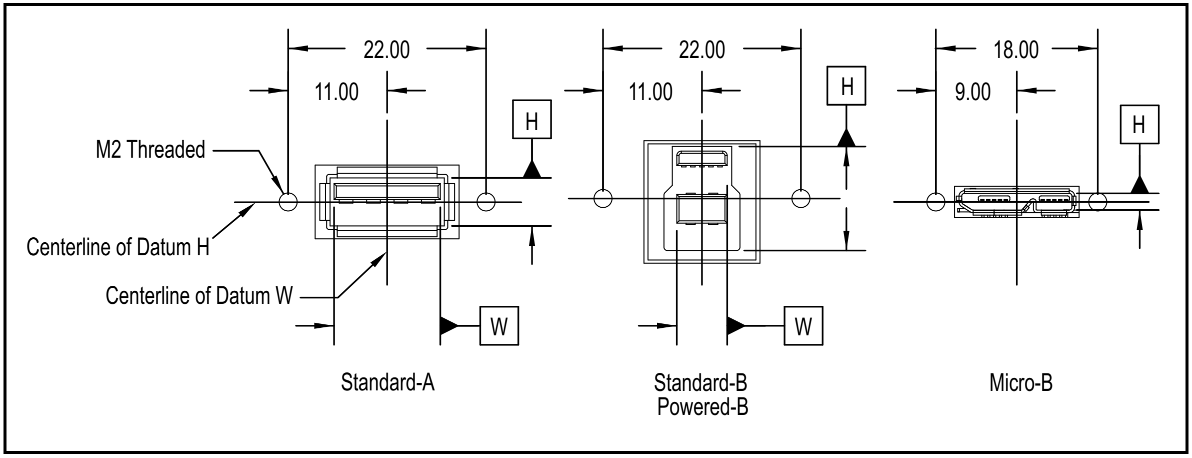


Figure 6-3: Standard-A, Standard/Powered-B, and Micro-B Receptacles With Screw Positions



Diagrams showing the position of locking screws for Type-C connectors may be found within the *USB Type-C Locking Connector Specification Revision 1.0.*

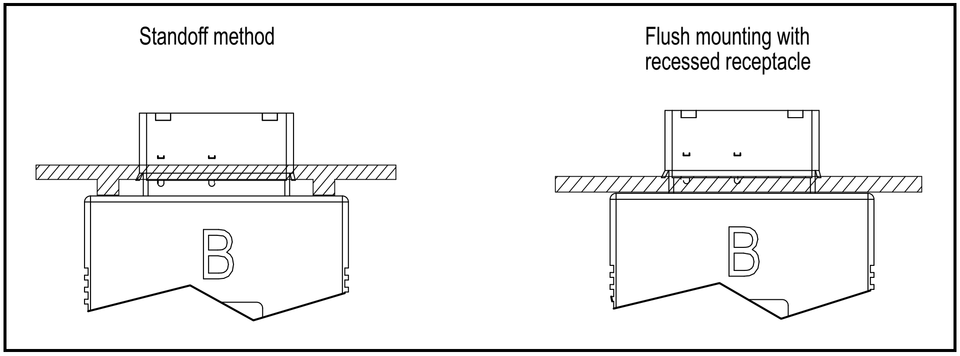
6.3.2 Mating

Existing devices utilizing MicroB connectors use two mating methods to adapt the overmold to the faceplate of the device:

- Stand offs
- Flush mounting

These mating methods are shown in Figure 6-4.

Figure 6-4: Mating Methods



In order not to limit the freedom of the device designer to choose one or the other (or a design in between), the depth of the receptacle relative to the faceplate is not specified. This is also necessary as the receptacle dimensions are not fully specified by the USB standard. The insertion tunnel, in particular, is only given as reference dimension. Therefore, it is up to the device manufacturer to properly recess the receptacle as needed. When the receptacle is not properly placed, excessive force can act on it when using locking connectors and therefore damage to the device can occur. Figure 6-4 shows that the relative position of the mated plug and receptacle is independent of the chosen mating method.

CR-100mdi
Any USB3 Vision device which uses the USB3 Vision locking mechanism MUST make sure the USB receptacle is placed in a way that it cannot be damaged mechanically by screwing in a USB3 Vision connector and MUST also have proper electrical contact.

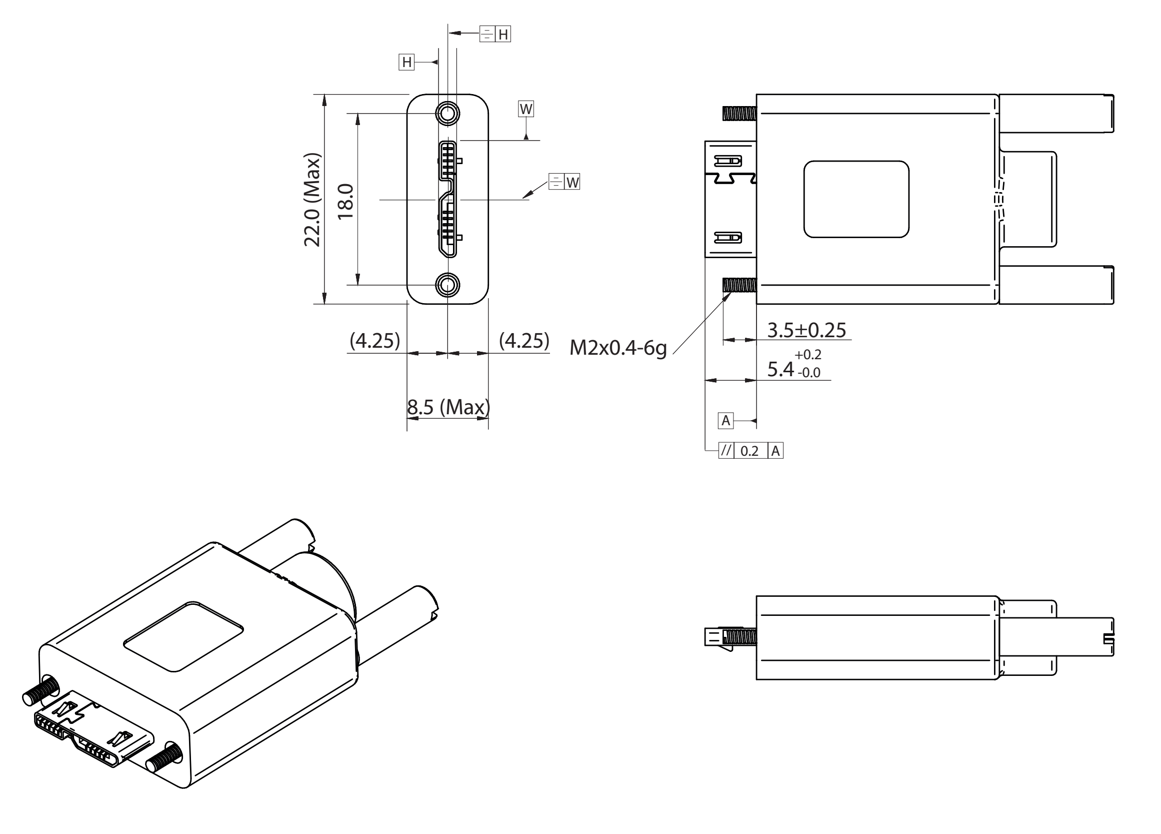
CO-101mi
USB3 Vision locking connectors for types A, B, and Micro-B should have a mating thread depth of at least 1.60 mm, with a standard M2 coarse thread pitch of 0.4 mm.

CR-102mi
USB3 Vision locking connectors for types A, B, and Micro-B MUST have the locking screws protruding from the overmold completely threaded.

6.3.3 Micro-B Locking Connector

The USB3 Vision Micro-B locking connector has locking screws 18 mm apart. This allows for enough clearance between the sides of the receptacle and each screw to have a female M2 threaded feature on either side of the receptacle on a threaded plane. The locking screw is dimensioned from the overmold to ensure there is enough mating overlap with the female thread. See Figure 6-5.

Figure 6-5: USB3 Vision Micro-B Locking Connector



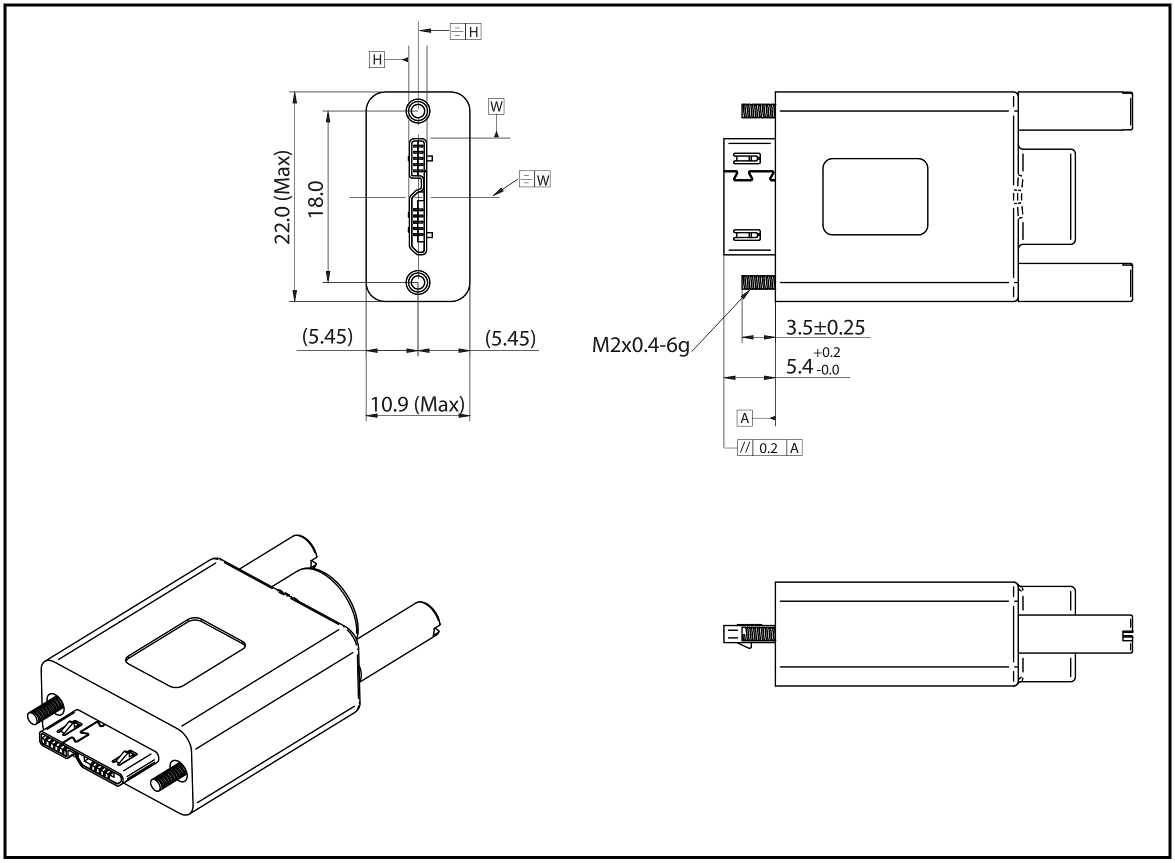
R-103mi

The USB3 Vision Micro-B locking connector MUST match the specifications of Figure 6-5.

6.3.4 Micro-B+ Locking Connector

The USB3 Vision Micro-B+ locking connector allows an overmold of 10.9 mm height instead of the 8.5mm of the standard Micro-B locking connector. No other changes are made. See Figure 6-6.

Figure 6-6: USB3 Vision Micro-B+ Locking Connector



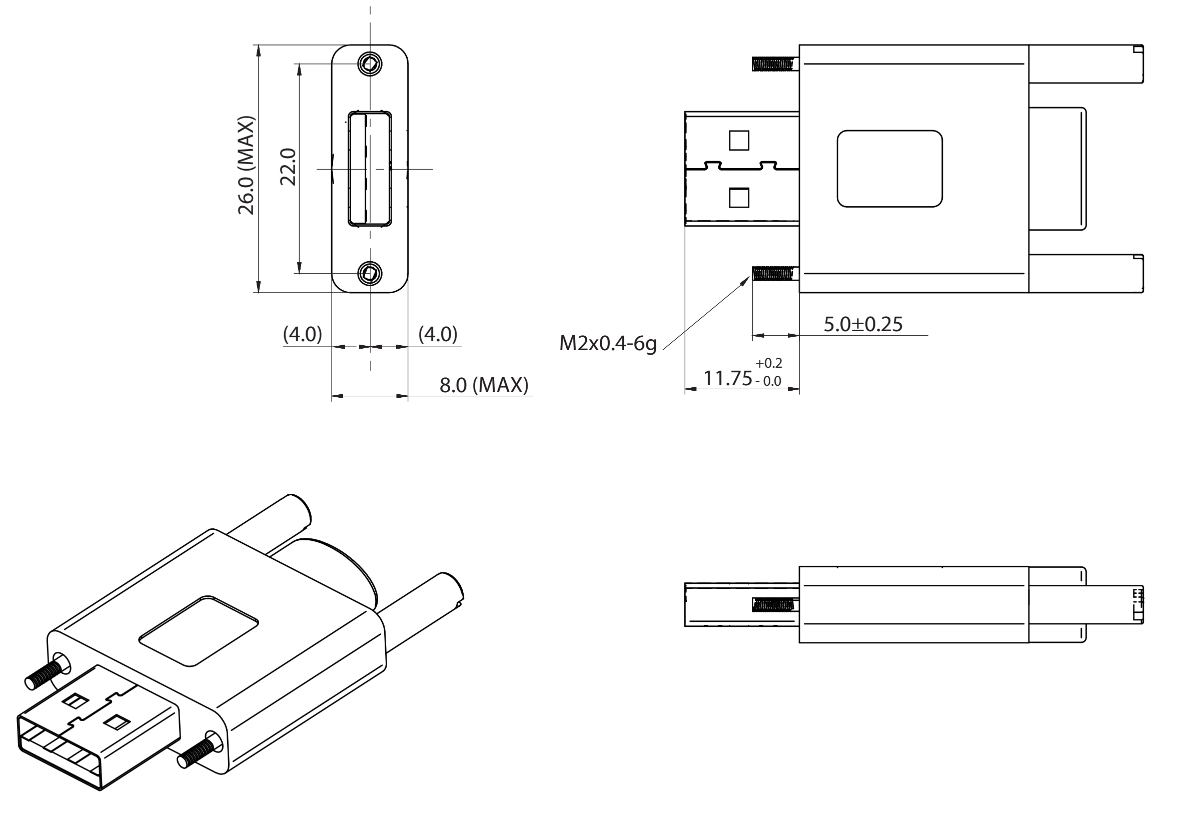
R-128mi

The USB3 Vision Micro-B+ locking connector MUST match the specifications of Figure 6-6.

6.3.5 Standard-A Locking Connector

The USB3 Vision Standard-A locking connector has locking screws 22 mm apart. This allows for enough clearance between the sides of the receptacle and each screw to have a female M2 threaded feature on either side of the receptacle on a threaded plane. The locking screw is dimensioned from the overmold to ensure there is enough mating overlap with the female thread. See Figure 6-7.

Figure 6-7: USB3 Vision Standard-A Locking Connector



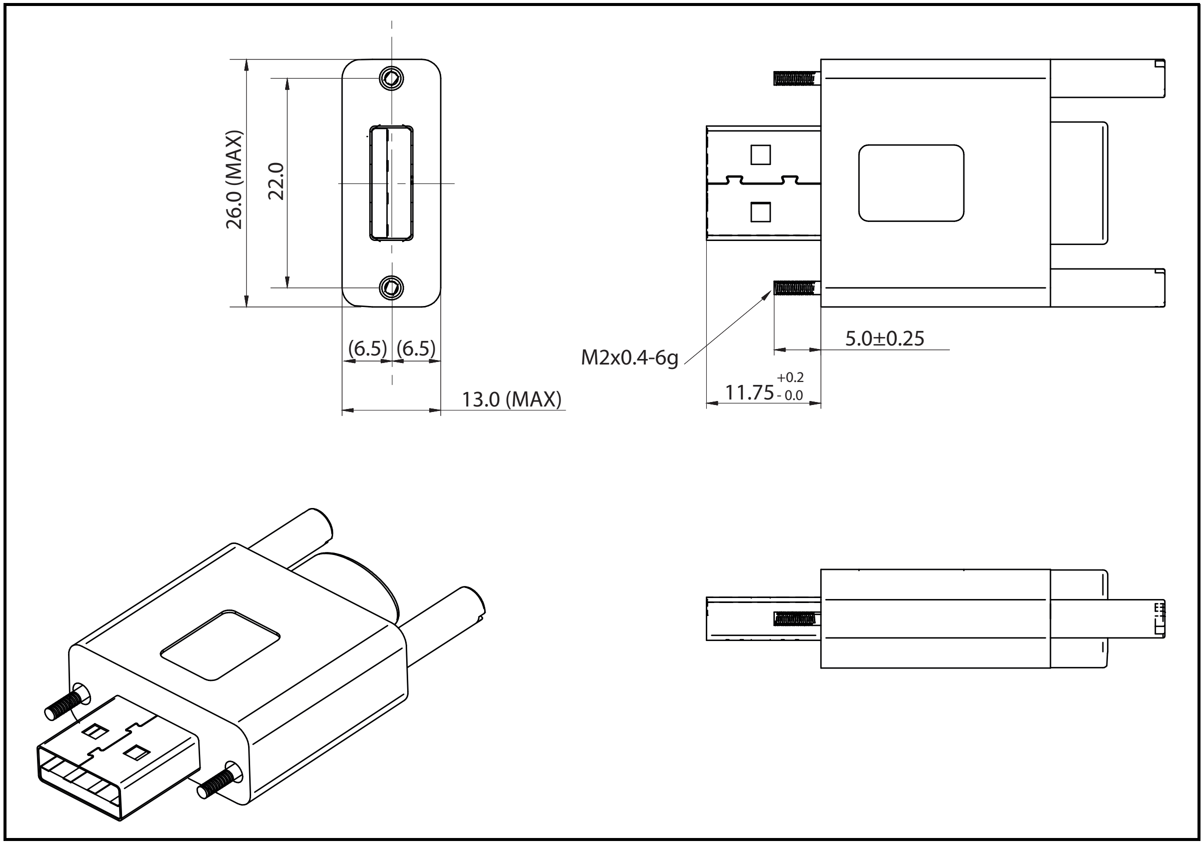
R-104mi

The USB3 Vision Standard-A locking connector MUST match the specifications of Figure 6-7.

6.3.6 Standard-A+ Locking Connector

The USB3 Vision Standard-+ locking connector allows an overmold of 13mm height instead of the 8 mm of the Standard-A locking connector. No other changes are made. See Figure 6-8.

Figure 6-8: USB3 Vision Standard-A+ Locking Connector



R-129mi

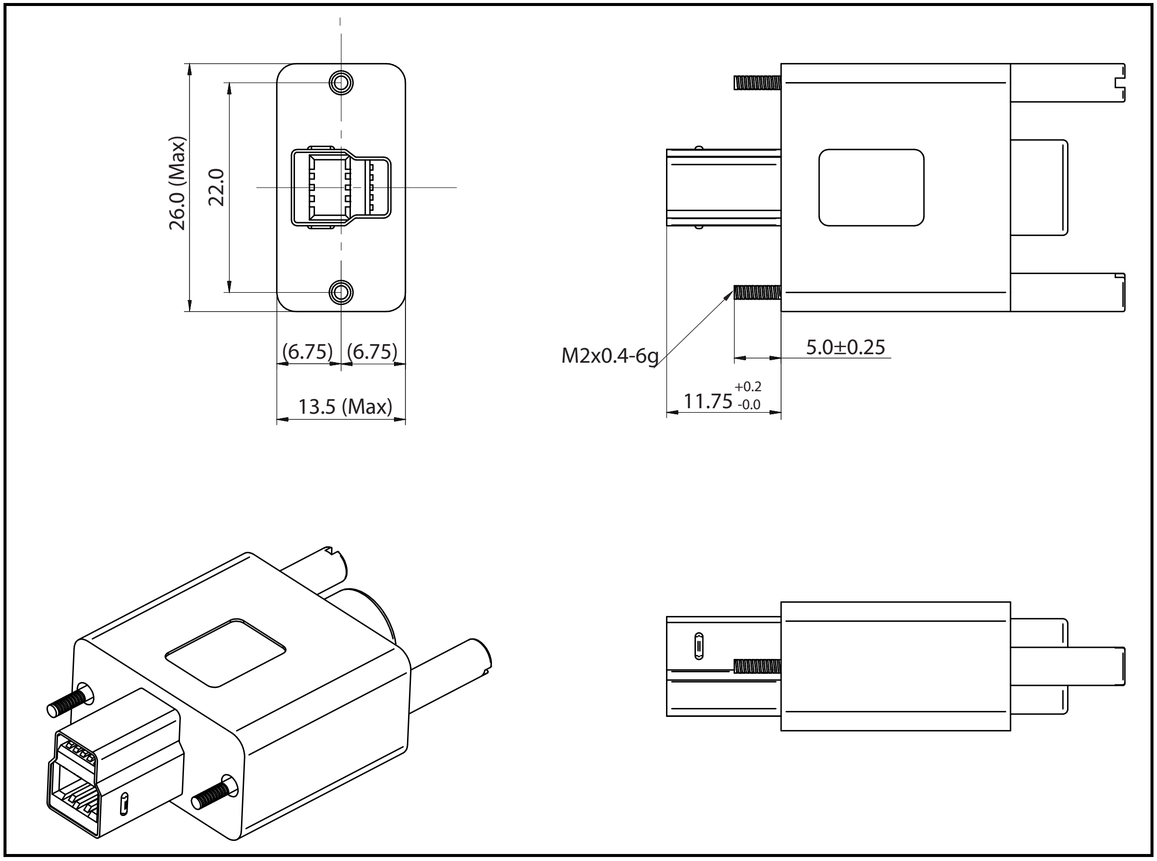
The USB3 Vision Standard-A+ locking connector MUST match the specifications of Figure 6-8.

6.3.7 Standard and Powered-B Locking Connectors

The mechanical dimensions for the USB 3.0 Standard-B and Powered-B connectors are identical (see USB 3.0 standard 5.2.1.3.).The dimensions for the Standard-B connector therefore apply also to the Powered-B connector.

The USB3 Vision Standard-B and Powered-B locking connectors have locking screws 22 mm apart. This allows for enough clearance between the sides of the receptacle and each screw to have a female M2 threaded feature on either side of the receptacle on a threaded plane. The locking screw is dimensioned from the overmold to ensure there is enough mating overlap with the female thread. See Figure 6-9.

Figure 6-9: Standard B and Powered-B Locking Connectors



R-105mi
The USB3 Vision Standard-B/Powered-B locking connector MUST match the specifications of Figure 6-9.

6.3.8 Type-C Locking Connectors

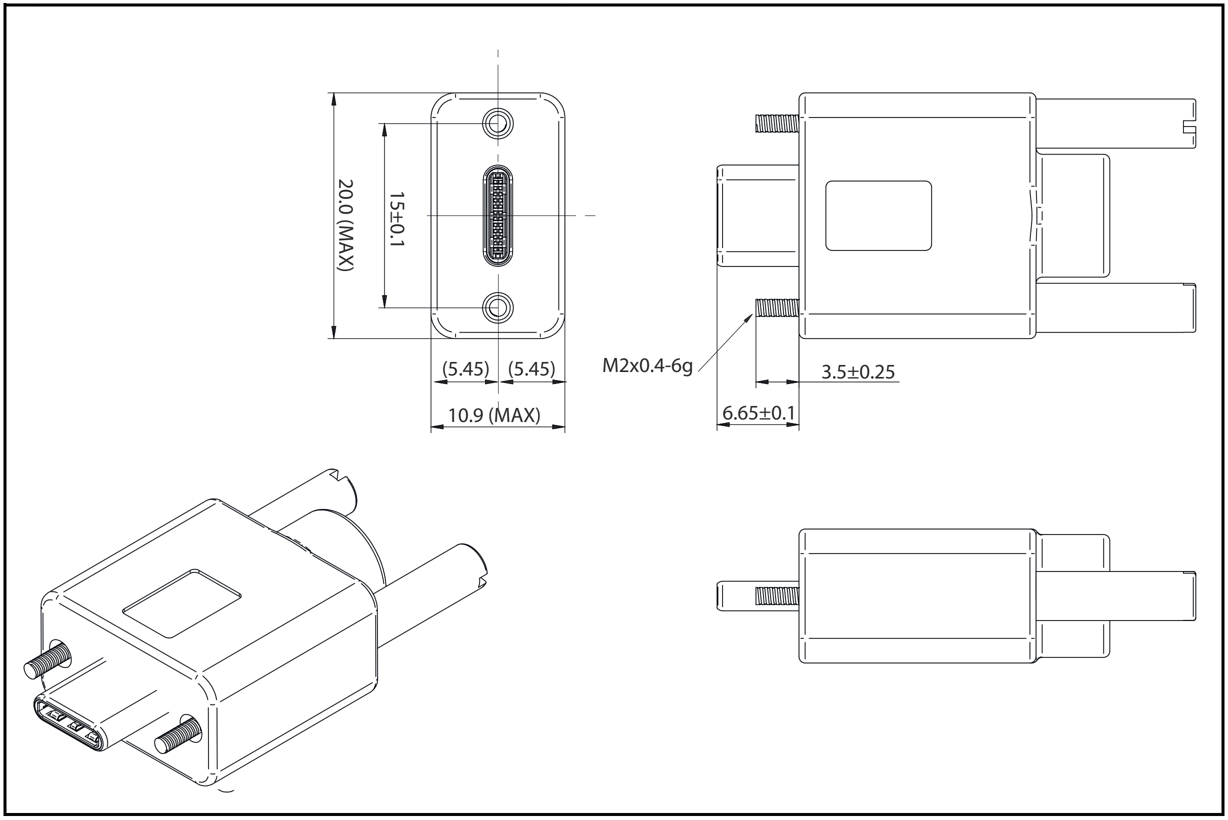
The specification for Type-C locking connectors is defined by the USB-IF Device Working Group in the USB Type-C Locking Connector Specification Revision 1.0.

R-130mdi
A USB3 Vision device utilizing Type-C locking connectors MUST follow the requirements and guidelines defined in the *USB Type-C Locking Connector Specification 1.0*.

6.3.9 Type-C+ Locking Connector

The USB3 Vision Dual Screw USB Type-C+ locking connector allows an overmold of 10.9mm height instead of the 6.5mm of the Dual Screw USB Type-C locking connector. No other changes are made. See Figure 6-10 for details.

Figure 6-10: Dual Screw USB Type-C+ Locking Connectors



R-131mdi
A USB3 Vision device utilizing Type-C+ locking connectors MUST follow the requirements and guidelines defined in the *USB Type-C Locking Connector Specification* 1.0 with the exception of the overmold dimensions as defined by this specification in Figure 6-10.

6.4 Cable Assemblies

To receive USB3 Vision certification the cable assemblies must pass USB3 Vision compliancy testing which is documented in a separate compliancy test document available from the A3.

R-106mi
USB3 Vision interconnection cable assemblies MUST use the Standard/Powered-B locking connector specified in Figure 6-9.

R-132mi
USB3 Vision Standard device cable assemblies MUST use the Micro-B locking connector specified in Figure 6-5 or the USB Type-C locking connectors specified in the *USB Type-C Locking Connector Specification 1.0*.

R-133mi
USB3 Vision Standard+ device cable assemblies MUST use the Micro-B+ locking connector specified in Figure 6-6 or the USB Type-C+ locking connectors specified in Figure 6-10.

7.0 Testing

7.1 Standard Compliance

The USB3 Vision standard compliance is self-certified by device and software vendors. This means all applicable requirements of this standard must be met. For devices this can be partly proved by successfully running the USB3 Vision test suite.

7.2 Design for Testability

USB3 Vision devices are designed to be testable. The test suite for checking compliance with the USB3 Vision standard needs to run without external interaction. Functionality and configuration registers which are needed for this purpose are part of the standard.

7.3 Device Features for Testing

Device features which are necessary for testing can be found in the GenICam XML description file of the device. The features which must be implemented may depend on the capabilities of the device. Some of the features are taken from the SFNC, but the features have been modified to be mandatory or conditional mandatory for USB3 Vision.

Table 7-1: Mandatory Device Features Required For Testing

Feature Name	Interface	Access	Units	Values	Description
DeviceReset	ICommand	W	-	-	Reset the device to its power up state. After receiving a DeviceReset command, the device sends an acknowledge immediately and waits MaxDeviceResponseTime before the command is executed to allow the acknowledge to be received.
DeviceFirmwareVersion	IString	R	-	NULL-terminated string (7-bit ASCII)	Version of the firmware in the device.

Table 7-1: Mandatory Device Features Required For Testing(Continued)

TestPendingAck	IInteger	RW	[msec]	>=0 0 [default]	Test PENDING_ACK feature. On write, the device waits a time period of TestPendingAck ms before acknowledging the write. If this time period is longer than the value in the Maximum Device Response Time register, the device must use PENDING_ACK during the completion of this request. On reads the device returns the current value without any additional wait time.

R-108cd
Any USB3 Vision device MUST support the features shown in Table 7-1.

Table 7-2: Conditional Mandatory Event Generator Device Features Required For Testing

Feature Name	Interface	Access	Units	Values	Description
EventTest	IInteger	R	-	0x4FFF	Returns the unique id of the Test event (U3V_EVENT_TEST) . It can be used to register a callback function to be notified of the event occurrence.
EventTestTimestamp	IInteger	R	[ns]	-	Returns the timestamp of the Test event (U3V_EVENT_TEST).
TriggerEventTest (DEPRECATED FEATURE NAME)	ICommand	W	-	-	Generates the Test event (TriggerEventTest in the Event Test Control register) if the event channel is enabled.
TestEventGenerate	ICommand	(R)/W	-	-	Generates the SFNC-described Test event with event ID U3V_EVENT_TEST

NOTE: Refer to the Event Control chapter of the GenICam SFNC for a detailed definition of the EventTestData category and the features listed in the above table.

CR-109cd

Any USB3 Vision device with an event channel must support the features shown in Table 7-2.

8.0 Document History

8.1 Table of Changes by Version

Table 8-1: USB3 Vision Document History

Version	Date	Editor	Description of Changes
1.0	January 2013	Bob McCurrach	Initial release from USB3 Vision technical Committee
1.0.1 Draft 1	September 2014	Eric Gross, Mark Breaux	Addressed following tickets: #193 Cleaned up list of figures table layout
			#229 Table 5-8 Image Extended Chunk Trailer clarifications
			#230 Table 5-11 Chunk Data Content style/ consistency improvements
			#256 RC4.1 section 4.1.3.4 - clarified TriggerEventTest
			#266 Unnecessary usage of "endianess" removed
			#272 Maximum Event Transfer Length descriptions clarified
			#273 RC4.11 5.4.1.6 SI Maximum Leader Size description clarified
			#274 5.4.1.11: SI Maximum Leader Size should follow 5.4.1.6 style
			#279 Requirement R-16d suffix fixed
			#281 Title Page links and ALT entry fixed
			#283 Table 52: block_id stream channel enable terminology clarified
			#284 New status-code U3V_STATUS_DEI_ENDPOINT_HALTED added for ambiguous case
			#297 R-62 suffix fixed
			#301 Timestamps for multiple ROIs clarified
			#305 Make length_written field of WRITEMEM_ACK mandatory
			#306 Clarified Message Channel ID register not used
			#307 Message Channel Supported GenCP capability bit and EIRM relationship clarified
			#309 Add requirement for U3V-required bits/ capabilities to be listed in Device Capability register
			#316 Add 'U3V_STATUS_DATA_OVERRUN' status code in addition to U3V_STATUS_DATA_DISCARDED
			#329 Address Pixel Format Naming Convention duplication
			#280 Added more cases for U3V_STATUS_SI_PAYLOAD_SIZE_NOT_ALIGNED status code
			#282 Clarified SI Required Payload Size wording

Table 8-1: USB3 Vision Document History(Continued)

			#286 Add requirement for SI Required Payload Size to match GeniCam "PayloadSize"
			#275 Match other A3 standards and capitalize requirements keyworks ("MUST")
			#304 Make R-2cd (U3V device label requirements) a conditional requirement
1.0.1 Draft 2	January 2015	Eric Gross, Mark Breaux	Addressed following tickets: #263 Clarify what is meant by the term product
			#277 Clarify alignment restrictions for payload transfers
			#296 Require host application to use device's reported Maximum Transfer Lengths
			#333 Add new "SuperSpeed 10Gbps" bit in speed enumerations
			#334 Add recommendation for devices to enumerate on 3.0 before 2.0
			#336 Address incorrect factory-default values for some bootstrap registers
			#345 Add requirement for device GenCP version to be the minimum defined by the specification but allow higher minor versions to be used
			#349 Change header on each page of spec to be version 1.0.1
			#350 Fix table-of-content links for new 1.0.1 requirements
			#352 Add change history section to document
1.0.1	March 2015	Eric Gross, Mark Breaux	-Removed spurious word "Trailer" in 5.4.1.6 SI Maximum Leader Size description
			Released as 1.0.1
1.1 Drafts 1-5	May 2018	Eric Gross	#291 Clarify that the stream receiver must discard all data after short or zero-length packet #263 Clarify what is meant by the term product #251 Status Code when accessing GenCP Heartbeat Timeout #248 Clarify error for unsupported commands #372 Clarify wording for R-44cd #358 Fix typo between Stream and Event registers and errors -Add USB 3.1 and 3.1 speeds/terminology -Add references to Type-C locking specification -Update referenced versions of dependent standards -Add support for multiple independent stream endpoints
1.1 Draft 6	May 2018	Thomas Hopfner	-Fully include Type-C locking connectors. -Add USB3 Vision Standard+ connectors which have a bigger overmold to allow longer and more flexible cables: Standard-A+, Micro-B+, Single/Dual Screw Type-C+.

Table 8-1: USB3 Vision Document History(Continued)

1.1 Draft 7	September 2018	Bob McCurrach	Transferring edits into final publication software.
1.1 RC0	March, 2019	Bob McCurrach	-Removed single-screw, type-C+ connector from specification -Updated external specification references
1.1 RC1	June 2019	Bob McCurrach	-Updated KB reference for consistency -Updated mechanical drawings for consistency, provided by Alysium -Removed reference to U3VSP USB3 Vision Streaming Protocol as this is not used anymore -Updated ABRM table to GenCP version 1.3 -Updated contributors list -Updated link to USB website in Ch 2. -Deleted line numbers in index -Updated R-126st to reflect actual field name
1.1 RC2	June 2019	Bob McCurrach	-Corrected verbiage in R-115cd -Corrected title of Figure 6-10 to include “+” -Revised sections 6.2 and 6.3 to more explicitly state that device support for Standard+ connectors is optional
1.1 Final	September 2019	Bob McCurrach	-Labeled as v1.1 for release based on successful Technical Committee (TC) vote.
1.2 Draft 1	June 2020	Eric Gross	-Added USB4 speed references -Added GenDC support
1.2 Draft 2	October 2020	Eric Gross	-Completed TODOs on register descriptions -Addressed feedback from meeting
1.2 Draft 3	November 2020	Eric Gross	-Reworked GenDC payload transfer mode requirements to consolidate them -Made some language regarding USB 3.0 be more generic -Addressed typos regarding bit field widths
1.2 Draft 4	February 2021	Eric Gross	-Remove USB4 speed references -Add disclaimer for USB 3.0 references -Change parallel mode support for hosts to optional
1.2 Draft 5	May 2021	Eric Gross Lutz Koschorreck Lisa Ripke	-Renamed GenDC transfer modes to better align with GenDC spec -Various clarifications
1.2 RC0	October 2021	Bob McCurrach	-Imported updates from v1.2 Draft 5 into Framemaker

Index