GLOBAL COORDINATION OF MACHINE VISION STANDARDIZATION

Japan Industrial Imaging Association Standard

日本インダストリアルイメージング協会規格

Version 2.1

JIIA CXP-001-2021

Revised on Jan 13, 2021

2021 年 01 月 13 日 改正

The Standardization Committee — CoaXPress Working Group
Japan Industrial Imaging Association

標準化委員会 — CoaXPress 分科会
一般社団法人 日本インダストリアルイメージング協会

This document is provided “as it is” with this current version.

None of JIIA, its members including contributions to the standard made by members of the G3 Group of Associations, the subsidiary companies of its members, or the affiliates of its members are liable, whether express or implied, for any kinds of warranties of merchantability, fitness for a particular purpose, and non-infringement of third-party property rights.

JIIA, its members including Contributions to the standard made by members of the G3 Group of Associations, the subsidiary companies of its members, or the affiliates of its members are exempted from taking responsibility and held harmless under the applicable law, whether express or implied, for any kinds of damages or losses arising from a course of usage of this document, including but not limited to passive damages, derivative damages, and incidental damages. This is applied even when any of JIIA, its members including Contributions to the standard made by members of the G3 Group of Associations, the subsidiary companies of its members, or the affiliates of its members was informed of the possibilities of damages in advance.

JIIA, its members including Contributions to the standard made by members of the G3 Group of Associations, the subsidiary companies of its members, or the affiliates of its members are exempted from taking responsibility and held harmless for any litigation, including but not limited to on defending, on cooperating, and on compensating, and infringement of intellectual property rights with third party, incurred by the usage of this document.

この書面は“現状のまま”の状態で提供されます。

JIIA，または G3 加盟協会会員による規格化協力会員，会員の子会社もしくは会員の関係会社のいずれも，この書面の内容に関して，商品性，特定の目的への適合性，非侵害の保証を含め，いかなる保証も，明示たると黙示たるとを問わず一切行いません。

JIIA，または G3 加盟協会会員による規格化協力会員，会員の子会社もしくは会員の関連会社のいずれも，この書面の使用，または使用不能から生ずるいかなる損害（逸失利益，及びその他の派生的，または付随的な損害を含むが，これらに限定されない全ての損害を言います。）について，適用法で認められる限り，一切の責任を負わないものとします。

たとえ JIIA，または G3 加盟協会会員による規格化協力会員，会員の子会社もしくは会員の関連会社がかかわる損害の可能性について知らされていた場合でも同様です。

JIIA，または G3 加盟協会会員による規格化協力会員，会員の子会社もしくは会員の関連会社のいずれも，この書面に起因して第三者との間に生じた，または生じうる知的財産権に関する紛争について，防御，協力，または補償する責任を負わないものとします。

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

Foreword

JIIA (Japan Industrial Imaging Association) is the Japanese machine vision standards body.

The initial development of CoaXPress was by the CoaXPress Consortium. The JIIA CoaXPress Technical Committee is responsible for the preparation and maintenance of this standard. Contributions to the standard are also made by members of the G3 Group of machine vision associations.

1 Scope

The CoaXPress digital interface was developed for high speed image data transmission and intended mainly for Machine Vision applications. The interface is also suitable for other imaging applications and for high speed data transmission in other fields.

CoaXPress uses 75 Ω coaxial cable as a physical medium.

2 References

2.1 Normative References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

- IEC 61169-8 Ed. 1.0:2007, Radio-frequency connectors — Part 8: Sectional specification — RF coaxial connectors with inner diameter of outer conductor 6,5 mm (0,256 in) with bayonet lock — Characteristic impedance 50 Ohm (type BNC).
- IEC 61169-8 Ed. 1.0 Annex A.:2007, Information for interface dimensions of 75 Ω characteristic impedance connector with unspecified reflection factor.
- IEC 61169-29 Ed. 1.0:2005, Radio-frequency connectors — Part 29: Sectional specification — Miniature radio frequency coaxial connectors model screw, snap-on, push-pull or quick-lock, slide-in (rack and panel applications) – Characteristic impedance 50 Ω (type 1,0/2,3) – 50 Ω and 75 Ω applications.
- GenICam™ standard from the European Machine Vision (EMVA) website (http://www.emva.org). See the GenICam section in this standard for version requirements. This standard references the following GenICam modules:
    - GenApi Also called the “GenICam Standard”
    - GenDC GenICam Data Container
    - GenTL GenICam Transport Layer
    - PFNC Pixel Format Naming Convention
    - SFNC Standard Features Naming Convention
- Byte oriented DC balanced 8B/10B partitioned block transmission code. US Patent 4486739.

2.2 General References

- Ref 1 ZIP format as defined on the PKWARE website (https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT).
- Ref 2 Recommendation ITU-R BT.601-6 “Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios”.
- Ref 3 Recommendation ITU-R BT.709-5 “Parameter values for the HDTV standard for production and international programme exchange”.
- Ref 4 IIDC2 Digital Camera Control Specification from the Japan Industrial Imaging Association (JIIA) website (http://www.jiia.org).
- Ref 5 Pixel Format Naming Convention (PFNC) from the EMVA website (http://www.emva.org).
- Ref 6 CoaXPress product registration and test procedures from the Japan Industrial Imaging Association (JIIA) website (http://www.jiia.org).

3 References and Definitions of Terms

3.1 Definition of Terms

Device — Can be any type of system that generates and transmits images or high speed data, as long as it is further responding and acting according to the CoaXPress standard. In most cases the Device will be a digital camera. A Device can also be on a frame grabber to allow for daisy chaining or data forwarding.

Host — Can be any type of system that receives, records, processes or displays high speed data, as long as it is further responding and acting in accordance with the CoaXPress standard. In most cases the Host will be a simple interface card, or more powerful frame grabber or image processor.

Connection — A single coaxial cable that connects a Device to a Host, which has the ability to provide a high speed connection in one direction, a low speed connection in the opposite direction, and power.

Link — One or more connections, comprising one Master connection and zero or more extension connections.

3.2 Abbreviations

CXP	CoaXPress
DT	Device Transceiver circuit
Gbps	Giga bit per second, 10^9 bits per second
HT	Host Transceiver circuit
LSB	Least significant bit
Mbps	Mega bit per second, 10^6 bits per second
MSB	Most significant bit
OCP	Over Current Protection
PoCXP	Power over CoaXPress
ppm	parts per million
PRU	Power Receiving Unit
PTU	Power Transmitting Unit
ROI	Region of Interest

3.3 Conventions

“Shall” means a mandatory requirement.

“Can” means an optional feature.

Comments written in italic text do not form part of the standard, but are intended to help understand the requirements of the standard.

4 Introduction to CoaXPress

4.1 Overview

CoaXPress is an interface to connect Devices (typically cameras) to Hosts (typically frame grabbers). It combines the simplicity of coaxial cable with state of the art high speed serial data technology, allowing up to 12.5 Gbps data rate per cable, plus device control and power in the same cable.

CoaXPress is a point to point scalable interface. The physical medium between the Device and Host is 75Ω coaxial cable.

An interface consists of one master connection and optional extension connections, which together form a link. Each connection is associated with a coax cable. At the Device connections are numbered; 0 for Master, and 1 to (*n*-1) for (*n*-1) extension connections as shown in the following figure:

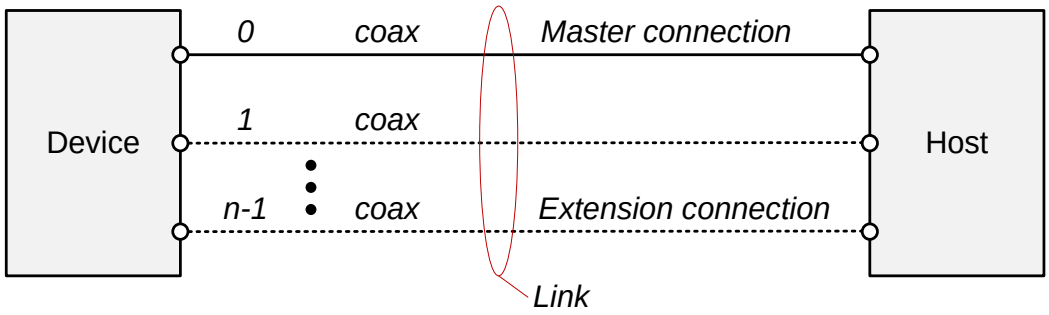


Figure 1 — CoaXPress physical topology

Each connection provides the following signaling functions:

- High speed serial (usually Device to Host downconnection), which can be up to 12.5 Gbps.
- Low speed serial (usually Host to Device upconnection), which can be up to 41.6˙ Mbps [1].
- Power (Host to Device), up to 13W.

A dedicated high speed upconnection from the Host to the Device is allowed for high speed triggers and camera control. This connection does not support power.

[1] Note – the dot after the number 6 means “6 recurring”, i.e. the bitrate is 41.666666… Mbps.

The link protocol defines the transfer of triggers, control data and high speed streaming data over a link, as shown in the following figure:

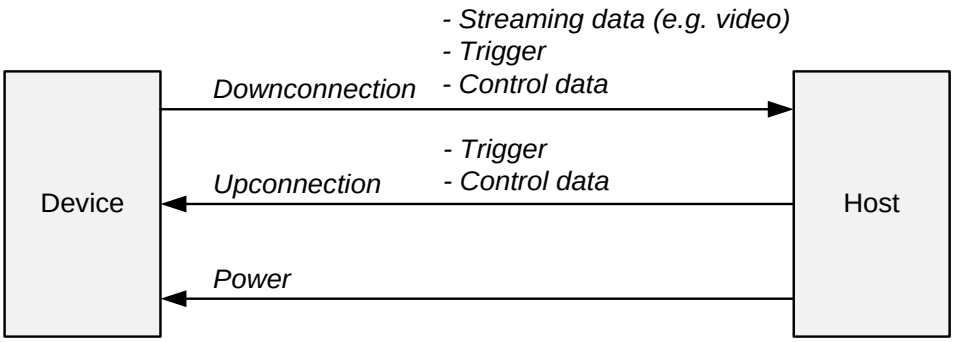


Figure 2 — Signaling connections and data flow

Both the upconnection and downconnection use 8B/10B coding.

Comment: 8B/10B is an industry-standard code that maps 8 bit data to 10 bit data to achieve DC balance on the connection, while also allowing clock recovery by ensuring regular transitions.

4.2 Use of Coax Cable

There are several reasons for specifying coax:

- Coax is acknowledged to be the best electrical medium for high speed data cables.
- Coax does not suffer from intra-pair skew. Intra-pair skew is an important performance limiter for differential cables at high speeds.
- Optimal bandwidth: a single-ended medium has less than half the loss of a shielded differential pair of the same diameter.
- Very good EMI/EMC performance.
- A large variety of possible cables allowed: thick ones for large distance, thin or flexible for shorter distance.
- It is a low cost solution.
- It is easy to install and terminate connectors in the field.
- Terminating several coaxial cables in a common connector is relatively easy to achieve; terminating several shielded twisted pairs in a common connector is an expensive manual time consuming task.
- There is a legacy cable infrastructure.

In this way using coax cabling, whether a single coax or a set of coax cables is used, optimal performance is always achieved.

4.3 CoaXPress Speeds

CoaXPress can operate at the speeds defined in the following table:

Table 1 — Supported high speed connection bit rates

CXP Speed	Bit Rate (Gbps)
CXP-1	1.250
CXP-2	2.500
CXP-3	3.125
CXP-5	5.000
CXP-6	6.250
CXP-10	10.000
CXP-12	12.500

Comment: Additional speeds, both lower and higher, may be defined in future versions of the standard.

These speeds are defined in section 7.6.

4.4 Connection Aggregation

CoaXPress Devices can use more than one coax cable in order to increase the maximum speed as described in section 4.1 above. The standard does not limit the number of cables which makes CoaXPress a future proof interface.

Comment: CoaXPress Devices operating at any speed can be used with more than one cable. e.g. In some situations two cables running at 3.125 Gbps may be preferred to one cable at 6.25 Gbps as it will allow for longer cable length and may result in a lower system cost (because lower cost components can be used).

4.5 Maximum Cable Length

The maximum cable length between Device and Host is dependent on the bit rate and the type of coax cable. For this reason CoaXPress certified cables are clearly marked with the maximum speed they support.

Example: At 6.25 Gbps the maximum cable length is around 25 meters for thin or highly flexible cables, or around 15 meters at 12.5 Gbps. At the other extreme, lengths of over 200 meters are possible with thicker cables at 1.25 Gbps.

Comment: The connection test mode gives an indication of the error rate on the connection as a part of a diagnostics test, or for testing if an existing cable in an analog system can be reused for CoaXPress.

4.6 Data Communication

Trigger, control data and streaming data share bandwidth. Different types of data are transferred using a priority scheme. As a result of this priority scheme the bandwidth available for low priority data is automatically limited by the amount of used bandwidth for high priority data.

4.7 Trigger Accuracy

Trigger sequences have the highest priority. Triggers on a high speed connection have an accuracy equal to ten transmitter bit periods, that is, better than ± 2 ns at 3.125 Gbps.

Triggers on a low speed connection use an accurate timing correction value in the trigger words that allows for a low fixed trigger latency. This is 3.4 µs, with an accuracy of ± 4 ns with a 20.83˙ Mbps upconnection (used for downconnection speeds up to and including 6.25 Gbps), or 1.7 µs, with an accuracy of ± 2 ns with a 41.6˙ Mbps upconnection (used for downconnection speeds over 6.25 Gbps).

4.8 Power over Cable

The available power per cable is 13W at the Device, at a nominal 24V.

It is anticipated that 13W should be enough for most Devices with one CoaXPress coax connector; likewise 26W should be sufficient for Devices with two connectors (up to 25 Gbps); 39W for three connectors, etc.

Should more power be needed, or if the Device is not designed to be powered over the cable, a separate power input on the Device can be used.

CoaXPress Hosts are equipped with Device detection and short circuit protection to minimize the risk of damage should a device other than a CoaXPress Device be connected.

4.9 Data Integrity

The control protocol (i.e. everything except for data) is designed to be tolerant of single bit errors. The data itself is protected by a CRC32 checksum which signals data errors.

Comment: Data resend is not supported because CoaXPress uses a physical medium that is designed to be nominally error free. Adding a resend capability would significantly increase the complexity of the protocol, add costs to the Devices that support it and decrease the real-time performance. CoaXPress can however inform the application whether there are bit errors and then an appropriate course of action taken.

For off-line testing of the complete imaging chain (Device – Cable(s) – Host), CoaXPress has a connection test mode. In this mode the Device transmits a known data sequence at a programmable bit rate that gives an indication of the error rate on the connection.

This mode can be used for a quick self-test during power-up, but also as a diagnostics tool (to identify unreliable cables or connectors).

Also the connection test mode can be used to verify connection robustness in combination with legacy coax cables (for example when upgrading from analog to digital cameras using existing cables).

4.10 Plug and Play

4.10.1 General

CoaXPress is designed to be plug-and-play.

CoaXPress contains mechanisms for automatic link setup (bit rate, link configuration scheme, Device detection) and Device and Host setup (image format, bit depth, data packing format, etc).

The link is designed to automatically recover after a loss of connectivity.

Comment: These mechanisms make CoaXPress hot-pluggable, and the protocol supports it. However this does not imply that an application will recover if a Device is unplugged and then reconnected.

4.10.2 GenICam

CoaXPress products support GenICam as described in section 13.

4.11 Unified Time Stamping

4.11.1 Introduction

Unified time stamping allows reporting of events coming from Devices, Hosts and software related events into a *unified time reference*. It also provides a convenient method to detect, at the Host side, missed or delayed triggers.

4.11.2 Accuracy

The CoaXPress standard allows time-stamping implementations providing 10 ns accuracy. An accuracy of 10 µs is considered as a minimum requirement for area-scan applications. In line-scan applications, an accuracy of about 10 ns is targeted.

4.11.3 Principle

The Device and the Host both maintain a local time clock. Both clocks are independent and free-running.

The Device uses its local time clock to timestamp any internal event. We say that the Device timestamps are expressed “in the time of the Device”, also abridged as “t-dev”.

Similarly, the Host uses its local time clock to timestamp any internal event. We say that the Host timestamps are expressed “in the time of the Host”, called “t-host”.

The Device sends periodically a heartbeat message to the Host (see section 9.7.1). The heartbeat message contains the “t-dev” time when the message was sent by the Device.

The Host uses the Device heartbeats to track the drift between the Device clock and its internal clock. At any time, the Host maintains the relationship between its internal clock and the Device clock with enough accuracy according to the application requirements.

The relationship between t-dev and t-host can be maintained by a simple first-order equation:

$Time_{T\text{-}HOST} = (Time_{T\text{-}DEV} * CSF) + COF$

where:

- $Time_{T\text{-}HOST}$ is the time expressed in the t-host reference
- $Time_{T\text{-}DEV}$ is the time expressed in the t-dev reference

- CSF is the Clock Scaling Factor
- COF is the Clock Offset

Based on this relationship, the Host can translate at any time a timestamp expressed into t-dev to a timestamp expressed in t-host. This mechanism avoids the need to send time messages from the Host to the Device, which would be more difficult with just the low speed upconnection.

In this way, all hardware events coming from both Device and Host can be ordered in the right sequence and precisely reported in a unified time reference.

4.11.4 Clock Tracking Basics

The Host uses heartbeat points to update its knowledge about the two parameters: the clock scaling factor (CSF) and the clock counter offset (COF).

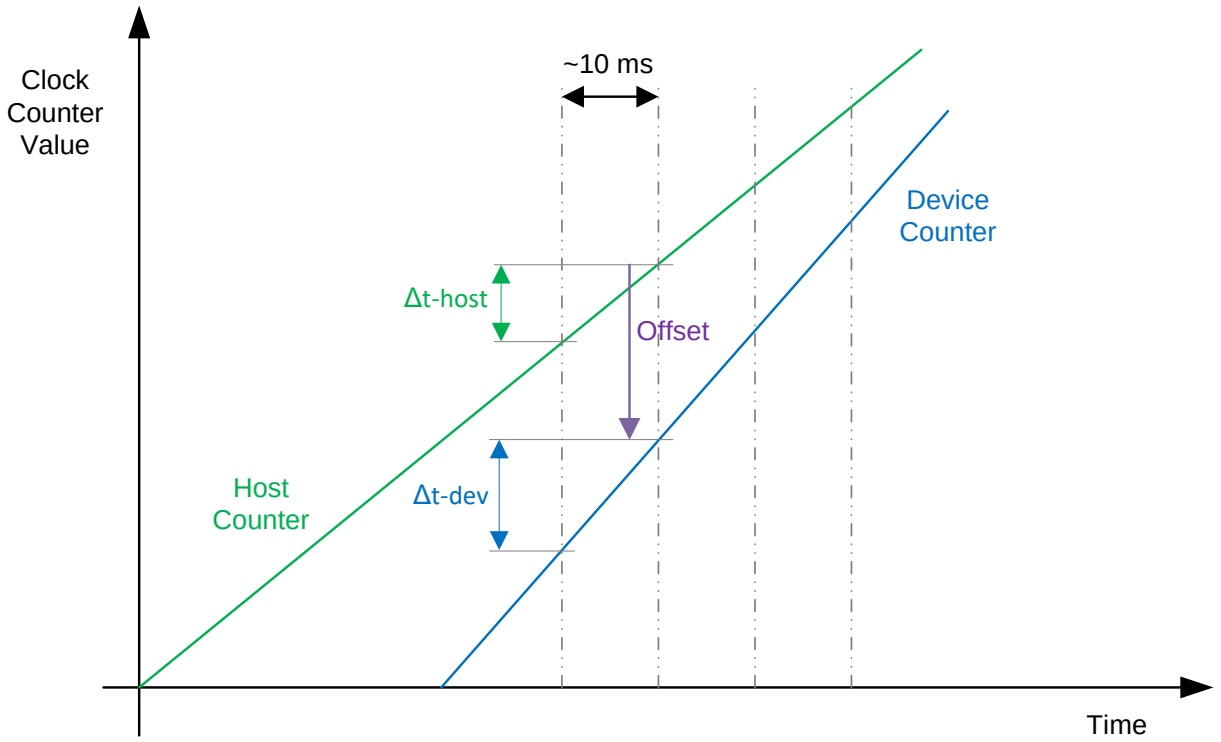


Figure 3 — Timestamp clock tracking

At each heartbeat message the COF is given by the difference between the clock counters and the CSF can be calculated by the ratio of Δt-host over Δt-dev.

This is the general principle. In practice, averaging techniques and careful fixed-point calculation must be used to achieve good accuracy.

The aim is to track very low speed frequency variations (like the one due to temperature), so the heartbeat rate may be slow.

The recommended heartbeat message interval (see section 9.7) is 10 milliseconds.

Note – this is not an implementation of IEEE 1588 (PTP), however a Host which does implement that standard can map the Device's time to its PTP time.

5 Labeling and Use of the CoaXPress Logos

The use of CoaXPress logos on products or marketing material is only allowed for products that have been registered with JIIA. The registered products shall have completed the compliance test procedure and have met the requirements, in accordance with the CoaXPress Compliance Product Certification Program.

Comment: JIIA provides compliance test procedures (section 2.2 ref 6).

5.1 Official CoaXPress Logo

The design of the official CoaXPress trademarked logo is shown in the following figure:

Figure 4 — Official CoaXPress Logo

CoaXPress is a registered trademark in Japan, the USA, the EU and other countries and the property of the trademark belongs to Japan Industrial Imaging Association (JIIA). You need a trademark license to use the logo on any product or in commercial documents. If you want to use the trademark or have any questions about using trademarks, please contact a JIIA representative (sec@jiia.org).

5.2 CoaXPress Logo Varieties

Varieties of CoaXPress logos are shown in the following figure. The first row is the standard color logo, the second row a variant for use on black backgrounds, and the third row the monochrome version. The second column is the short-form logo to be used where space does not permit the full version.

Figure 5 — CoaXPress logos

Artwork is available from JIIA.

5.3 Product Labeling

CoaXPress compliant products shall be labeled with the official Logo and indication according to the following table:

Table 2 — Labeling of CoaXPress products depending on their allowed bit rate

Maximum operational Bit Rate per Coax (Gbps)	Compliance Labeling
1.250	CXP-1
2.500	CXP-2
3.125	CXP-3
5.000	CXP-5
6.250	CXP-6
10.000	CXP-10
12.500	CXP-12

This label indicates that the product meets the CoaXPress standard, with a maximum supported bit rate per coax cable as indicated. It applies to all CoaXPress products, including Hosts, Devices and CXP-Cables.

Cable example: A given length of CXP-Cable marked as CXP-5 conforms to all the standard requirements for a maximum bit rate of 5 Gbps per coax. It will qualify for the lower bit rates as well. A longer length, but using the same cable type, might only qualify as CXP-3.

Comment: This labeling is purely indicating CoaXPress compliance. The product vendor is free to label their products to indicate other features in addition to the compliance label.

A Device with more than one coax connector shall label the master connector with an arrowhead ▶ pointing towards the connector.

5.4 Product Feature Bar

It is recommended that CoaXPress product information (e.g. in brochures, adverts and websites) is labeled with a feature bar in the style shown in the following figure:

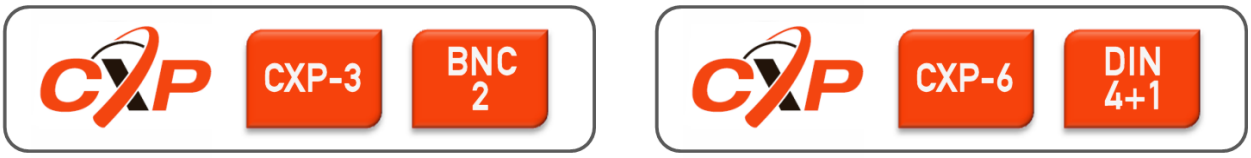


Figure 6 — Feature bar

The first box gives the maximum speed, in the format listed in section 5.3, and the second box gives the connector type and number of connections. The connector type shall be “BNC” for BNC type connectors, “μBNC” for Micro-BNC connectors and “DIN” for DIN 1.0/2.3 type connectors.

Examples: The left hand feature bar above is for a product with two BNC downconnections running at a maximum of 3.125 Gbps, and the right hand one for a product with DIN 1.0/2.3 connectors, with four downconnections and a high speed upconnection all running at a maximum of 6.25 Gbps.

Artwork is available from JIIA.

Comment: This gives a simple “at a glance” guide to help end users and system integrators choose compatible cameras, cables and frame grabbers.

6 Cable and Connector Specification

6.1 Introduction

This section defines CoaXPress cables and connectors from the point of view of a manufacturer of a Device or Host. See "Annex A: Detailed Cable Specification" for detailed cable requirements for cable manufacturers.

CoaXPress uses 75 Ω coaxial ("coax") cables.

A cable compliant with this standard is called a **CXP-cable**; a bundle of coax cables including connectors is called a **CXP-multi-cable**. At cable ends one shall have **CXP-connectors**, or when grouped at ends of a CXP-multi-cable, can have **CXP-multi-connectors**. The connectors at the Device and the Host side are the receiving **jack CXP-connectors** and **jack CXP-multi-connectors**.

Comment: The use of an approved CoaXPress cable is the simplest way to ensure reliable operation, particularly at high speeds and/or long cable lengths. However at lower speeds, and/or with shorter cables, many legacy 75Ω coax cables should operate perfectly.

6.2 Connectors

Comment: The connector standards do not define insertion loss requirements; however this requirement is included in the overall cable and electrical characteristics defined in this standard.

6.2.1 Single Connectors

Three types of connector are defined for CoaXPress. See JIIA Whitepaper CXPR-006-2018 for recommendations on the choice of connector for a given application.

6.2.1.1 BNC

BNC connectors shall be the 75 Ω type, as defined in IEC 61169-8 annex A. They shall also comply with all other requirements in IEC 61169-8 other than the requirement for 75 Ω instead of 50 Ω.

The connectors on the Device and Host shall be a socket-center contact type, and that on the cable shall be a plug-center contact type.

6.2.1.2 Micro-BNC

Micro-BNC (also known as HD-BNC) connectors shall be the 75 Ω type. This connector is a de-facto standard, originated by Amphenol®. Therefore an example drawing is provided in "Annex C: Micro-BNC".

The connectors on the Device and Host shall be a socket-center contact type, and that on the cable shall be a plug-center contact type.

A fixing nut, or panel, case etc, shall always be flush or lower than the top of the threaded part of the Device or Host connector.

Comment: This ensures that the cable connector does not foul the fixing nut or panel, case etc.

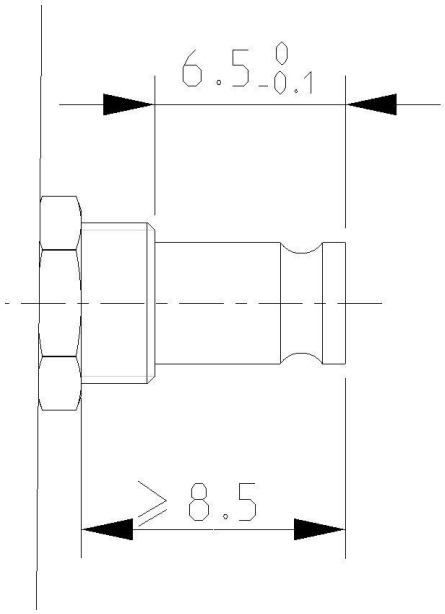
6.2.1.3 DIN 1.0/2.3

DIN 1.0/2.3 shall be the 75 Ω push-pull type, as defined by the IEC 61169-29 standard.

The connectors on the Device and Host shall be a female type (socket), and that on the cable shall be a male type (plug).

The DIN 1.0/2.3 connector shall only be used at speeds up to and including CXP-6.

Additionally the minimum thread length in the following figure shall be met:



Dimensions in mm

Figure 7 — DIN Connector Thread Length

Comment: This ensures that the cable connector does not foul the fixing nut. It also allows use of screwlock cable plugs. IEC 61169-29 does not specify this dimension.

6.2.2 Multi-Connectors

A multiway connector is formed with ‘n’ downconnection channels (a master plus n-1 extension connections), plus optionally one high speed upconnection channel, and shall use either Micro-BNC or DIN 1.0/2.3 connectors as defined in sections 6.2.1.2 and 6.2.1.3.

Connection positions for a Device, and dimensions for both Device and Host shall be as shown in the following figure:

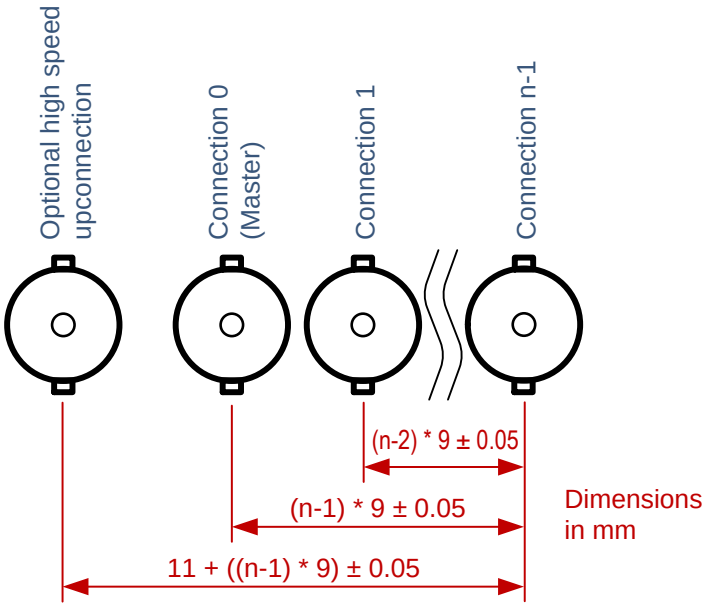


Figure 8 — Device / Host Multi-Connector

The connections will be in the same order at the Host if a CXP-multi-cable is used.

For a Micro-BNC multi-connector, the coupling studs shall be orientated as shown in the figure.

Comment: The master connection and high speed upconnection may need to be routed to the same SERDES in the Host to allow them to both run at the same bitrate, even if other Host connections are routed to different Devices at different bitrates. Therefore some Hosts may only support the high speed upconnection if the master connection is connected to the connector adjacent to the high speed upconnection (which will always be the case if a CXP-multi-cable is in use).

Comment: If individual connectors are used on a Host or Device to form a multi-connector, a jig may be needed to achieve the tolerances shown.

A CXP-multi-cable for DIN 1.0/2.3 connectors has a nominally 1mm (min) locking slot in a defined position (see section A.7). Device and Host vendors can supply a bracket that fixes to Device or Host and engages in the slot to provide an additional mechanical locking function by preventing the release tab from being moved. For ease of reference part of the diagram showing the CXP-multi-cable is duplicated here to show the slot:

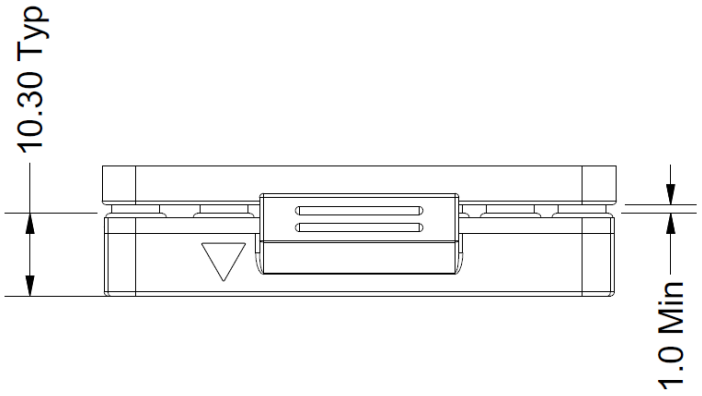


Figure 9 — DIN 1.0/2.3 Multi-Connector Locking Slot

6.2.3 Couplers

CoaXPress cables can be joined with inline 75 Ω couplers, however these will reduce the usable cable length.

Comment: Couplers rated to at least the operational frequency should be used (e.g. 3.125 GHz for 6.25 Gbps), however even the best may reduce cable length by around 5m at 6.25 Gbps.

6.2.4 Contact Material

For BNC connectors it is recommended that the center contact is gold plated, and the outer contact either nickel or white bronze plated.

For Micro-BNC connectors it is recommended that the center contact is gold plated, and the outer contact either nickel or gold plated.

For DIN 1.0/2.3 connectors it is recommended that both the center and outer contacts are gold plated.

Comment: These are proven contact materials. Nickel and white bronze (an alloy of copper, tin and zinc) are compatible materials, i.e. they can be used together. Additional approved materials may be added in future versions of the standard.

6.3 Multiple Cables for Connection Aggregation

When more than one physical cable is used to form a link, all the cables shall be nominally the same type and length. The difference in length of the cables shall be less than 1m.

Comment: This is to limit the skew in data packets between the multiple cables, and hence to minimize the FIFO requirements at the Host to realign the data. A 1m difference corresponds to approximately a 5ns skew, which is 4 words at 25 Gbps, so easily handled by a FIFO.

6.4 Connector Indicator Lamps

It is recommended that Devices and Hosts have an indicator lamp by each connector. If fitted, these lamps shall include the following indications:

Table 3 — Connector indicator lamp states

State	Indication
No power	Off
System booting	Solid orange
Powered, but nothing connected (not applicable to a Device reliant on PoCXP power)	*Flash_1* red
Connection detection in progress[2], PoCXP active	*AlternateFlash_12_5* green / orange Shown for a minimum of 1s even if the connection detection is faster
Connection detection in progress, PoCXP not in use	*Flash_12_5* orange Shown for a minimum of 1s even if the connection detection is faster
Device / Host incompatible, PoCXP active	*AlternateFlash_0_5* red / green
Device / Host incompatible, PoCXP not in use	*AlternateFlash_0_5* red / orange
PoCXP over-current (Host only)	Solid red
Device / Host connected, but no data being transferred	Solid green
Device / Host connected, waiting for event (e.g. trigger)	*Flash_1* orange
Device / Host connected, data being transferred	*Flash_12_5* green
Error during data transfer (e.g. CRC error, single bit error detected)	500ms red pulse In case of multiple errors, there shall be at least two green *Flash_12_5* pulses before the next error is indicated
Connection test packets being sent	*AlternateFlash_0_5* green / orange
Compliance test mode enabled (Device only)	*AlternateFlash_0_5* red / green / orange
System error (e.g. internal error)	*Flash_12_5* red

[2] For a Host it is recommended that this indication is shown from the point at which it detects initial low-level connection lock following a *ConnectionReset,* until discovery is complete. For a Device it is recommended that this indication is shown from the point at which it receives a *ConnectionReset* until discovery is complete.

Table 4 — Connector indicator lamp timings

Indication	Frequency (±20%)	Duty Cycle (on) (±20%)
Flash_12_5	12.5 Hz	25% (20ms on, 60ms off)
Flash_1	1 Hz	20% (200ms on, 800ms off)
Flash_0_5	0.5 Hz	50% (1s on, 1s off)
AlternateFlash_12_5	12.5 Hz	25% (20ms on color 1, 60ms off, 20ms on color 2, 60ms off)
AlternateFlash_0_5	0.5 Hz	50% (1s on color 1, 1s off, 1s on color 2, 1s off etc)

Recommendations:

- It is recommended that error states have the highest priority.
- It is recommended to only change indicator lamp states aligned to a timing transition defined in Table 4.
- It is recommended that both Device and Host have an option to turn off the indicator lamps, for example to allow use in darkened environments. An additional recommended option is to only show error conditions on the lamps.

Examples: If the Device is waiting for trigger, and the Host is ready to accept data, the Device will indicate a Flash_1 orange, and the Host solid green. If the triggering was done instead at the Host, with the Device continuously streaming data, the Device would show Flash_12_5 green and the Host a Flash_1 orange.

Comment: These indications are designed to help a user understand what the system is (or more importantly, isn't) doing. They will also help Technical Support staff to diagnose customer issues.

Comment: The indications do not try to show the data rate being sent on the connection, only that data is being transferred. Additional indicators can be fitted to show data rate.

7 Electrical Specification

7.1 Introduction

This section defines the electrical parameters from the point of view of a manufacturer of a Device or Host. Reference to datasheets of an approved CoaXPress transceiver may be useful.

See "Annex B: Chipset Requirements" for detailed requirements for chipset designers (parts of which may also be useful to Device and Host manufacturers as part of product compliance testing).

7.2 Top Level Overview

Figure 10 shows the electrical block diagram of one connection in a CoaXPress system, with the Device on the left linked to the Host on the right by a coax cable with a characteristic impedance of 75 Ω. The figure also defines the test points Tp1 to Tp4 used in this section.

The high speed connection is usually the downconnection from the Device to the Host, and the low speed connection is in the opposite direction, so usually the upconnection from the Host to the Device. In systems using the optional high speed upconnection these are reversed, but for clarity this section describes the usual case. A high speed upconnection shall implement a Host Transceiver in the Device, and a Device Transceiver in the Host according to this section.

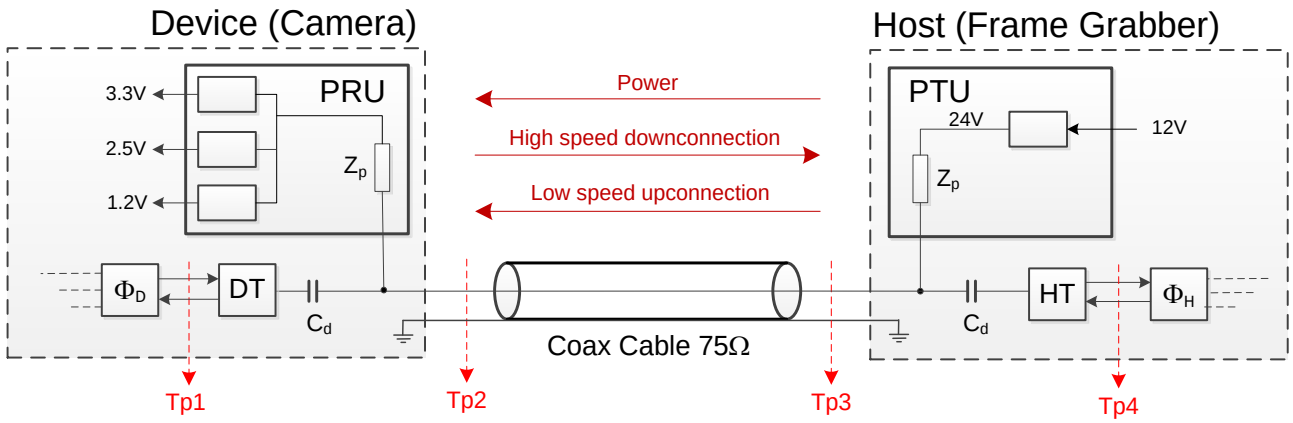


Figure 10 — Electrical block diagram

Comment: Voltages shown at the output of the PRU and the input to the PTU are typical examples.

The Device has a Phy Φ_D (including a high speed serializer and low speed deserializer) connected to a Device Transceiver (DT) that transmits and receives the bidirectional full duplex data over the coax cable. The Device shall use a Device Transceiver compliant with Annex B.

Comment: The Device Transceiver splits the incoming low-speed data signal and the outbound high speed data signal. It also compensates for low frequency loss in the received low speed data, so restoring the original digital bit stream at Tp1.

The I/O pin of the Device Transceiver shall be coupled to the coax cable through a capacitor C_d that AC couples the data between this I/O pin and the center contact of the coax connector.

The Host has a Phy Φ_H (including a low speed serializer and high speed deserializer) connected to a Host Transceiver (HT) that transmits and receives the bidirectional full duplex data over the coax cable. The Host shall use a Host Transceiver compliant with Annex B.

Comment: The Host Transceiver splits the incoming high-speed data signal and the outbound low speed data signal. It also compensates for frequency dependent attenuation in the coax cable, so restoring the original digital bit stream at Tp4.

The I/O pin of the Host Transceiver shall be coupled to the coax cable through a capacitor C_d that AC couples the data between this I/O pin and the center connection point of the coax connector.

Connections using Power over CoaXPress (PoCXP) shall also implement a Power Transmit Unit (PTU) or Power Receive Unit (PRU), as described in section 8 on PoCXP.

7.3 Capacitor C_d

Capacitor C_d at the Host Transceiver side shall have a value of between 80nF and 500nF.

Capacitor C_d at the Device Transceiver side shall have a value of between 25nF and 500nF.

Comment: The 25nF and 80nF limits allow it to couple the low speed data through. A lower capacitor value at the Device side limits the stress on the inputs of the Device Transceiver (DT) when the 24V power supply is switched. The actual value used should be based on the recommendations of the transceiver vendor.

The breakdown voltage V_{BRCd} of capacitor C_d at both the Host and the Device side shall be at least 50V.

Comment: This is higher than the PoCXP voltage of 24V, to withstand possible overshoots when the power is ramped up.

7.4 Termination

A resistor of 75 Ω ± 15% shall terminate the coax transmission line at both the Host and the Device side.

Comment: This wide tolerance allows the terminators to be implemented within the Device Transceiver (DT) or Host Transceiver (HT), which gives much better high frequency performance than discrete resistors, without violating the return loss requirements (section 7.8). However transceivers using discrete resistors may need to use tighter tolerance terminators, as recommended by the transceiver vendor. At several GHz the termination will not be a pure resistance – this is allowed for in the return loss requirements.

7.5 Impedance Z_p

Impedance Z_p shall be as high as possible with respect to the 75 Ω transmission line impedance for the high bit rate signal frequency, and shall be controlled for the low bit rate signal frequency (2 – 10 MHz at 20.83˙ Mbps and 4 – 20 MHz at 41.6˙ Mbps).

To get this controlled impedance for the low bit rate signal frequency, the inductive part L_p of the impedance Z_p at both the Host and the Device side shall have a value of 11.5 µH (± 30 %).

Comment: This can be obtained by placing an inductor of 10 µH in series with a ferrite bead. In that case, at low bit rate, the inductor is mainly determining the impedance, whilst at high bit rate this inductor represents

a low impedance due to its parasitic self-capacitance. The ferrite bead in series then makes the impedance high at the high bit rate.

Comment: In practice, this requirement is covered by the return loss specification.

7.6 High Speed Connection

The high speed connection bit rate shall be certain multiples of 625 Mbps starting at 1.250 Gbps as listed in the following table:

Table 5 — Supported high speed connection bit rates

Supported Bit Rates	Unit Interval (UI)	Discovery Rate
1.250 Gbps	800 ps	✓
2.500 Gbps	400 ps	
3.125 Gbps	320 ps	✓
5.000 Gbps	200 ps	
6.250 Gbps	160 ps	
10.000 Gbps	100 ps	
12.500 Gbps	80 ps	

A Device (camera) shall support at least one of the defined bit rates in normal operation. A Device shall support one discovery rate (see section 12.1.2).

Comment: Often the operating bit rate will be fixed, and will be the lowest speed that can be used to suite the Device's performance. Unnecessary use of higher speeds will result in higher power consumption and will also reduce maximum cable length. The number of discovery bit rates is restricted to a small number of values to speed up the discovery process. More than one discovery bit rate is defined because a Device's SERDES may not support 1.25 Gbps, especially as higher speeds are added in future.

A CoaXPress Host (frame grabber) shall support all the defined bit rates between 1.25 Gbps and the highest rate it supports.

CoaXPress products shall be labeled to indicate their maximum speed as described in section 5.3.

The relative tolerance of the bit rate shall be ≤ ±100 ppm for speeds up to and including 6.25 Gbps, and ≤ ± 50 ppm for speeds above 6.25 Gbps.

A Device with multiple connections shall derive the high speed connection data from high speed clocks all derived from one common sub rate master clock, e.g. a 125 MHz clock.

A Device or a Host implementing a high speed upconnection, shall use a CoaXPress compliant Device Transceiver DT such that the high-speed output waveform is compliant with "Annex B: Chipset Requirements".

A Host, or a Device implementing a high speed upconnection, shall use a CoaXPress compliant Host Transceiver HT such that the Phy Φ_H can reconstruct the original bit stream by compensating for attenuation in the cable as defined in "Annex B: Chipset Requirements".

The jitter at Tp2 produced by the combination of the Phy Φ_D and the Device Transceiver (DT) shall not be more than 20% of the Unit Interval (UI). This jitter is shown as Tj in the following eye diagram at Tp2:

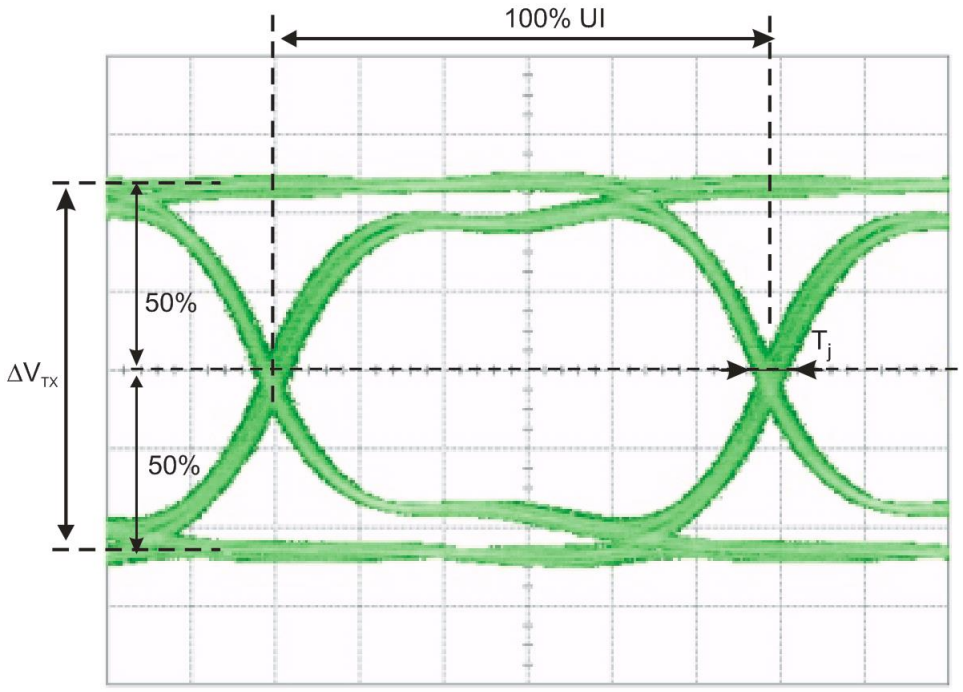


Figure 11 — Definition of the jitter in the transmit EYE at Tp2

Comment: Jitter is defined with respect to the Unit interval, resulting in relaxed normative absolute parameters for lower speed Devices. This jitter will mainly be a result of the clock source to the Phy Φ_D and the phy itself, but that contributed by the Device Transceiver must be allowed for.

7.7 Low Speed Connection

The low speed connection bit rate shall be certain fractions of 125 Mbps as listed in the following table.

Comment: This allows easy clock generation from the master clock used for generating and locking to the high bit rate clocks. A SERDES function at these speeds is easily implemented in FPGA fabric as a state machine.

Table 6 — Supported low speed connection bit rates

Supported Bit Rates	Unit Interval (UI)	Used at Connection Speeds
20.83˙ Mbps	48.000 ns	CXP-1 to CXP-6
41.6˙ Mbps	24.000 ns	CXP10, CXP-12

The relative tolerance of the bit rate shall be ≤ ±100 ppm.

A Host, or a Device implementing a high speed upconnection, shall use a CoaXPress compliant Host Transceiver HT such that the low-speed output waveform is compliant with “Annex B: Chipset Requirements”.

A Device, or a Host implementing a high speed upconnection, shall use a CoaXPress compliant Device Transceiver DT such that the Phy Φ_D can reconstruct the original bit stream by compensating for attenuation in the cable as defined in “Annex B: Chipset Requirements”.

The jitter at Tp3 produced by the combination of the Phy Φ_H and the Host Transceiver (HT) shall not be more than 5ns. This jitter is shown as T_{jLF} in the following eye diagram at Tp3:

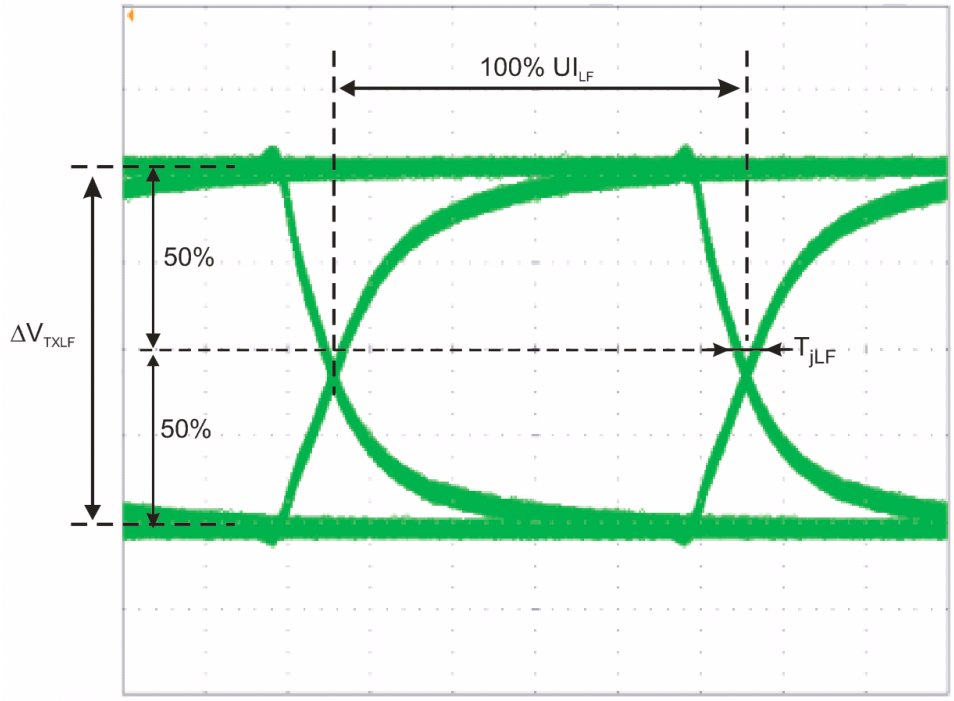


Figure 12 — Definition of the jitter in the low speed transmit EYE at Tp3

Note 1: *For clarity this eye diagram is shown with no simultaneous flow of high speed connection data.*

Note 2: *For clarity this eye diagram is shown with no baseline wandering due to the low-frequency loss from the impedance Z_p. (see section B.4.1).*

Comment: *Some CoaXPress transceivers will use clock and data recovery (CDR) functions. As a result, bit errors or even loss of lock may be seen on the low speed upconnection for up to 10ms after the corresponding high speed downconnection has been enabled, or has changed bitrate (see section B.4.3).*

7.8 Return Loss at Connectors

In order to have a measure for the quality of the high frequency front ends of the Device and the Host, the Return Loss is specified at the jack CXP-connectors at Tp2 and Tp3.

Comment: This return loss requirement is achieved by using suitable grade components (including the connector) in the CoaXPress circuitry, and by correct PCB design. See the important guidance notes below.

In order to evaluate whether all this has been done in a way according to this CoaXPress standard, Table 7 and Table 8 define frequency ranges in which the return loss shall be better than 15 dB, 10 dB, 7 dB or 4 dB. These frequency ranges depend on the specified highest bit rate of the connected Device or Host under test. Poor high frequency design (e.g. including stubs, not providing ground plane cut-outs below ferrite beads), or using the wrong inductors or ferrite beads will result in a worse return loss behavior in at least one of the frequency ranges, bringing it out of specification.

Table 7 — Normative return loss frequency ranges for Host

Host side			
Highest bit rate	**Frequency range where Return Loss shall be better than -15 dB**	**Frequency range where Return Loss shall be better than -10 dB**	**Frequency range where Return Loss shall be better than -7 dB**
1.250 Gbps	5 MHz – 312 MHz	312 MHz – 625 MHz	-
2.500 Gbps	5 MHz – 625 MHz	625 MHz – 1.25 GHz	-
3.125 Gbps	5 MHz – 625 MHz	625 MHz – 1.25 GHz	1.25 GHz – 1.62 GHz
5.000 Gbps	5 MHz – 625 MHz	625 MHz – 1.5 GHz	1.5 GHz – 2.50 GHz
6.250 Gbps	5 MHz – 625 MHz	625 MHz – 1.5 GHz	1.5 GHz – 3.2 GHz
10.000 Gbps	5 MHz – 625 MHz	625 MHz – 3.2 GHz	3.2 GHz – 5.0 GHz
12.500 Gbps	5 MHz – 625 MHz	625 MHz – 3.2 GHz	3.2 GHz – 6.25 GHz

Table 8 — Normative return loss frequency ranges for Device

Device Side			
Highest bit rate	**Frequency range where Return Loss shall be better than -10 dB**	**Frequency range where Return Loss shall be better than -7 dB**	**Frequency range where Return Loss shall be better than -4 dB**
1.250 Gbps	5 MHz – 312 MHz	312 MHz – 625 MHz	-
2.500 Gbps	5 MHz – 312 MHz	312 MHz – 1.25 GHz	-
3.125 Gbps	5 MHz – 312 MHz	312 MHz – 1.25 GHz	1.25 GHz – 1.62 GHz
5.000 Gbps	5 MHz – 312 MHz	312 MHz – 1.5 GHz	1.5 GHz – 2.50 GHz
6.250 Gbps	5 MHz – 312 MHz	312 MHz – 1.5 GHz	1.5 GHz – 3.2 GHz
10.000 Gbps	5 MHz – 625 MHz	625 MHz – 3.2 GHz	3.2 GHz – 5.0 GHz
12.500 Gbps	5 MHz – 625 MHz	625 MHz – 3.2 GHz	3.2 GHz – 6.25 GHz

Comment: *Other industry standards exist that have a more stringent return loss requirement at the highest frequency range. These standards typically allow signal transmission over a series coupling of a cable and a PCB backplane. In these cases stringent return loss is required to control local resonances of data patterns that could destroy signal integrity.*

In the CoaXPress standard it is assumed that the HT and DT circuits are located very close to the connector. This relaxes the return loss requirements, allowing easier support of higher bit rates and also to include PoCXP through Z_p.

Important Guidance Notes:

*CoaXPress is likely to be the fastest PCB design that many companies have implemented, therefore some degree of "learning curve" may be needed even though very few components are needed in a typical CoaXPress circuit. The simplest way of meeting these return loss requirements is to **precisely** follow the component and layout recommendations from the manufacturer of the CoaXPress compatible Host Transceiver HT and Device Transceiver DT, and to take advantage of any design review services offered by the manufacturer or their distributors. Note that at multi-Gigabit speeds, using "equivalent" components or small PCB layout changes (even moving a via or changing the physical size of a passive component) can have significant detrimental effects.*

8 Power over CoaXPress: PoCXP

8.1 Overview

The available power per cable is 13W, at a nominal 24V. It is anticipated that 13W should be sufficient for most Devices with one CoaXPress coax connector; likewise 26W should be sufficient for most Devices with two connectors, 39W for three connectors etc.

Should more power be needed, a separate power input to the Device can be used, but the intention is that PoCXP is the usual method of powering a CoaXPress Device, so avoiding the higher system cost of a "power brick" for the Device.

Comment: 24V is chosen to allow sufficient power to be available to power most Devices over the long cable lengths that CoaXPress supports. The use of 12V would mean either much lower available power because of I^2R losses in the cable, or much reduced cable lengths. While the use of 24V means that most frame grabbers will need a 12V to 24V supply, this should cost less much than the power brick that a Device would otherwise need, so minimizing system cost.

PoCXP shall not be implemented on a high speed upconnection.

PoCXP implements Device detection and short circuit protection to minimize the risk of damage should a Device other than a CoaXPress Device be accidentally connected to a CoaXPress Host.

Inductors in both the Host and Device prevent supply noise corrupting image data. Both the transmitter and receiver need to be AC coupled anyway, so they are unaffected by the power signal. The inductors and AC coupling requirements are described in section 7.

8.2 Simplified Example Block Diagram

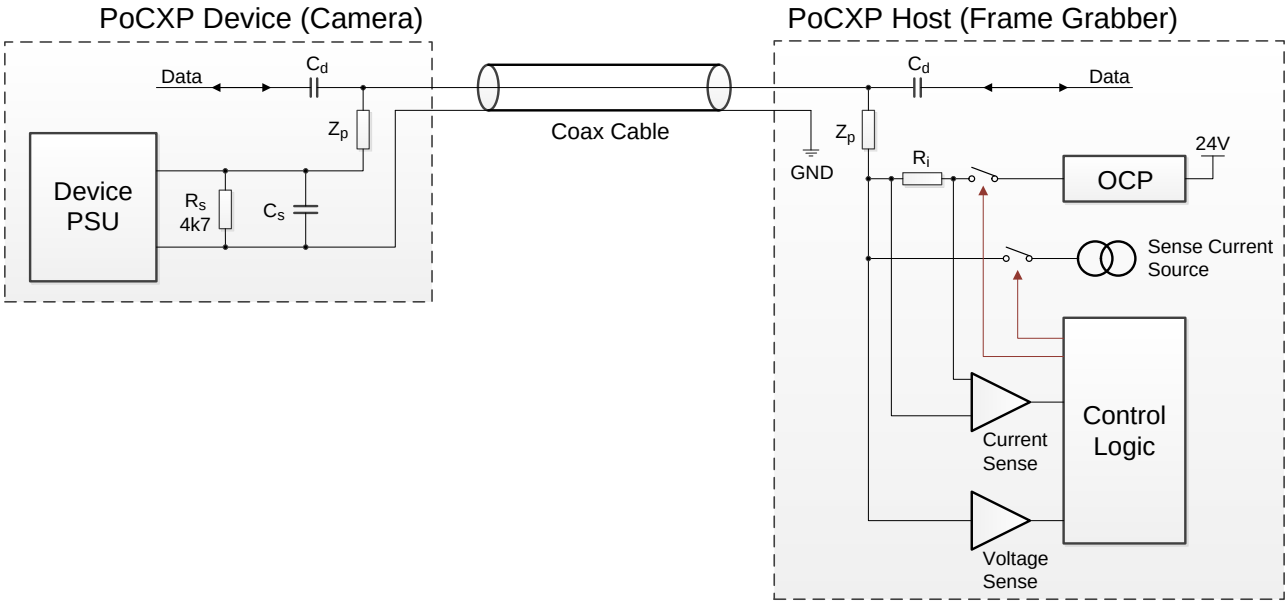


Figure 13 — PoCXP block diagram

Table 9 — Key to Host block diagram

Host	
OCP	Over current protection circuit (see section 8.4.2.3).
R_i	Low value sense resistor for current sensing (see section 8.4.4).
Z_p	Inductive filter between power and data (see section 7.5).
C_d	Data coupling capacitor for the high and low speed connections.

Table 10 — Key to Device block diagram

Device	
C_d	Data coupling capacitor for the high and low speed connections.
Z_p	Inductive filter between power and data (see section 7.5).
C_s	Device input capacitance (see section 8.3.3.2).
R_s	Device sense resistance (see section 8.3.3.1).

8.3 Device Requirements

8.3.1 General Requirements

It is highly recommended that all Devices that consume less than 13W per CoaXPress coax connector draw power from the Host via PoCXP (although the Device can have an auxiliary power connector). An exception is that a Device that is inherently self-powered, such as a daisy-chain port on a frame grabber, need not implement PoCXP.

Devices that may consume more than 13W per CoaXPress coax connector shall draw power from an auxiliary connector.

The PoCXP section in a Device is a Power Receiving Unit (see section 7.2).

Comment: Note that the Compliance Test Specification for CoaXPress Devices requires that a sample test camera must be provided that is not powered by PoCXP to allow certain tests including eye diagram checks to be made.

8.3.2 Operating Requirements

8.3.2.1 Voltage

The Device shall operate over a range of 18.5V DC to 26V DC.

Comment: This allows for a 3.5V round-trip drop in the cable compared to the nominal 24V ± 2V output by the Host, corresponding to a maximum round-trip cable resistance of 5Ω. "Round-trip" means the total effect of the center and outer conductors in the coax cable.

The Device shall not be damaged by continuous application of 30V DC and short term startup overshoot up to 50V.

Comment: 30V is the maximum voltage allowable by the "1394 over coax" standard.

8.3.2.2 Power

The Device shall draw a maximum of 13W per cable.

Comment: 13W gives a current of 541mA at the nominal 24V, or 703mA at the minimum 18.5V.

The Device shall not draw more than 50mA until 25ms after its input voltage has reached 15V.

Note: The initial charging current into C_s is excluded from this 50mA figure.

Comment: This defines a startup / brownout requirement, such that the Device does not draw 13W at a voltage much less than 18.5V, and so over-current the Host. The 25ms requirement prevents reflections on the cable causing off-on-off switching, and means that the input voltage should exceed 18.5V by the time the Device PSU starts.

8.3.3 Support for CoaXPress Detection

All CoaXPress Devices that take power from a CoaXPress coax connector shall implement the following requirements. These requirements apply to each connector that takes power.

8.3.3.1 Input Resistance

The Device shall present a sense resistance of 4k7Ω ± 5% whenever the input voltage at the Device is greater or equal to 2.2V and less than or equal to 5.5V.

Comment: This equates to a sense current of between 446µA (2.2V, 4k7Ω + 5%) and 1.23mA (5.5V, 4k7Ω – 5%). This wording also allows the sense resistance to be disconnected when the Device is powered at 24V. It also allows the use of an active circuit to implement the resistance, so allowing easy disconnection at 24V. However if left connected, the 4k7Ω sense resistance will only result in additional 123mW of power dissipation in the Device at 24V. The sense resistance is labeled "R_S" in Figure 13.

8.3.3.2 Input Capacitance

The Device shall have a maximum input capacitance C_S on the CoaXPress coax connector of 57µF.

Comment: The value of 57µF allows a 47µF 20% component to be used. This capacitor is labeled "C_S" in Figure 13.

The Device shall discharge this input capacitance to less than 1V within 500ms of power being removed.

Comment: This allows the Host to correctly sense a camera that gets unplugged and reconnected.

8.3.3.3 Minimum Load Power

The Device shall draw a minimum current of 15mA from the CoaXPress coax connector starting not more than 0.25s after power is applied.

Comment: This allows the Host to detect if the Device is disconnected. 15mA at 24V corresponds to 0.36W.

For avoidance of doubt, this requirement does apply to a Device with more than one CoaXPress coax connector, and consuming more than 13W: It shall draw a minimum of 15mA from a powered connector, even if the other(s) do not yet have power applied.

Comment: The PoCXP state machines in a Host do not know that they are being connected to one camera, so they may be out of sync. Without this requirement it is possible that one connection gets switched off before another one is powered up.

8.3.4 Devices Over 13W

A Device with more than one CoaXPress coax connector, and consuming more than 13W, shall meet the following additional requirements:

The Device's power supply shall be designed so that it does not draw more than 13W per cable.

Comment: This requirement means that the Device needs to detect when a sufficient number of CoaXPress coax connectors have power and only then start up the Device's power supply.

Comment: The Device can take power from any of its connectors, but if it supports ConnectionConfig modes with a limited number of connections, then in general those connections should also be the ones that it takes power from.

The Device's power supply shall be designed to isolate the CoaXPress coax connectors, i.e. power applied to one connector shall not be injected into any other ones.

8.3.5 Additional Power Connectors

A CoaXPress Device can have an auxiliary power connector to allow it to be used in a system with a separate Device power supply.

Comment: This should only be needed on Devices that consume more than 13W per cable, in which case they would not implement sense resistance R_s.

8.3.5.1 Non-PoCXP CoaXPress Devices

On a Device designed only for use with the auxiliary power connector, it shall not have an input resistance that meets the requirements of section 8.3.3.1. In non-PoCXP CoaXPress Devices the PRU can be omitted.

Comment: Typically the only load provided by a non-PoCXP Device will be one side of the data coupling capacitor C_d.

8.3.5.2 Dual Power CoaXPress Devices

A Device can be designed so that it is powered by either PoCXP or by the auxiliary power connector. In this case:

The Device shall be designed to isolate the power sources. In particular the auxiliary connector(s) shall not inject power into the Host, and the CoaXPress coax connector(s) shall not inject power into the auxiliary connector(s).

Comment: This prevents damage in the event that both a PoCXP Host and the separate power supply are used concurrently.

Whenever power is applied to the auxiliary power connector, the Device shall be powered from the auxiliary power connector and shall disconnect the sense resistance R_s, such that it shall not have an input resistance that meets the requirements of section 8.3.3.1.

Comment: This requirement prevents a PoCXP Host from continuously applying power to the Device, not sensing the minimum current flowing (see section 8.4.4), and disconnecting power again.

8.4 Host Requirements

8.4.1 General Requirements

All CoaXPress Hosts shall implement PoCXP for each CoaXPress coax connector using a Host Transceiver.

The PoCXP section in a Host is a Power Transmitting Unit (see section 7.2).

8.4.2 Operating Requirements

8.4.2.1 Voltage

When PoCXP is enabled, the Host shall supply 24V DC ± 2V to the CoaXPress coax connector.

Comment: The Host needs to allow for the voltage drop in the OCP, switching circuits, sense circuits and Z_p.

8.4.2.2 Power

The Host shall be capable of supplying 17W per cable over the full voltage range defined in section 8.4.2.1.

Comment: This 17W requirement allows for the loss in the cable. A multi-output Host may need to have an auxiliary connector to directly connect to the computer's power supply if the power consumption exceeds that allowed through the bus connector.

8.4.2.3 Edge Rate

As power is applied to the Device, the Host shall limit the rise time on the 0V to 24V (nominal) transition to a maximum of 2V/μs. This requirement shall be checked without a Device using a 4k7Ω load attached to the output of the Host.

Similarly, when applying the PoCXP sense current to the Device, the Host shall limit the rise time on the 0V to 7V (maximum) transition to a maximum of 2V/μs. This requirement shall be checked without any load or Device connected.

Comment: These requirements are to prevent damage to Host or Device transceivers that could result from repeated application of a fast edge.

8.4.3 Over-Current Protection: OCP

All PoCXP Hosts shall implement an over-current protection circuit with the following characteristics:

- A holding current over the Host's operating temperature range of at least 790mA.
- A nominal trip current over the Host's operating temperature range of not more than 5A.

It is recommended that a UL approved component is used to implement the OCP.

Comment: The OCP prevents serious damage or fire risk in the event of a failure in the Device, damage to the cable, or accidental coupling to a non-CoaXPress Device that happens to satisfy the PoCXP detection system. A PTC resettable fuse should satisfy this requirement providing its operating voltage is at least 26V.

Additionally, PoCXP Hosts shall be able to withstand a short circuit during normal operation without damage.

Comment: During normal operation it is possible that due to a damaged cable for example, a short circuit can occur. A Host should be able to withstand this short circuit without damage of its internal components. Note that the trip time of a PTC resettable fuse is usually too slow to prevent damage to electronic components.

8.4.4 Support for CoaXPress Detection

All CoaXPress Hosts shall implement the following requirements.

These requirements apply to each connector.

- The Host shall sense that a PoCXP Device is connected before applying power. It can do this by sensing the Device's nominal 4k7Ω input resistance "R_S", in which case it shall use a sense current of between 550μA and 1mA.

  Comment: This gives a sensed voltage of 2.46V to 4.94V after allowing for the RC time constant of 0.28s from the 57μF maximum capacitance "C_S" with the 4k7Ω input resistance. The sense current range of the Host is deliberately narrower than that implied by section 8.3.3.1 to give a noise margin to the system.

- Once power has been applied to a PoCXP Device, the Host shall monitor the power supply current consumed. If at any time after the initial 0.3s the average current is less than 8mA for 0.5s, the Host shall remove power from the Device.

  Comment: This causes power to be removed if the Device or cable is unplugged. It also allows the state machine controlling PoCXP to return to its "Sense" state ready to detect a new Device. The current sensing circuit can also be used to monitor the actual power consumed by the Device (and cable), and switch off power should the current exceed certain limits, but this is not a requirement of the

standard. The figure of 8mA for the Host, compared to 15mA for the Device, gives a noise margin to the system.

- Once power has been applied to a PoCXP Device, the Host shall monitor the voltage on the CoaXPress coax connector. If for more than 20ms this voltage drops to a level indicating that the OCP has tripped, the Host shall remove power from the Device.
- The Host shall specifically be designed not to apply power to the following non-PoCXP Devices:
  - Non-PoCXP CoaXPress Device, or analog Device, where the Device has a coupling capacitor as the only load.
  - 50Ω or 75Ω load to ground.
  - Short circuit to ground.
  - Large capacitative load such that the sensed voltage is not tending towards a stable value allowing for a maximum C_s of 57µF.
- The Host shall not exceed the following values while sensing for a PoCXP Device:
  - Maximum voltage with no load connected: 7V.
  - Maximum short circuit current: 2mA.

Comment: These 7V and 2mA limits are safety requirements, however if the no-load voltage exceeds 5.5V then some Devices using active sense circuits might not be powered based on section 8.3.3.1.

The following figure gives a simplified example state diagram for PoCXP Device detection:

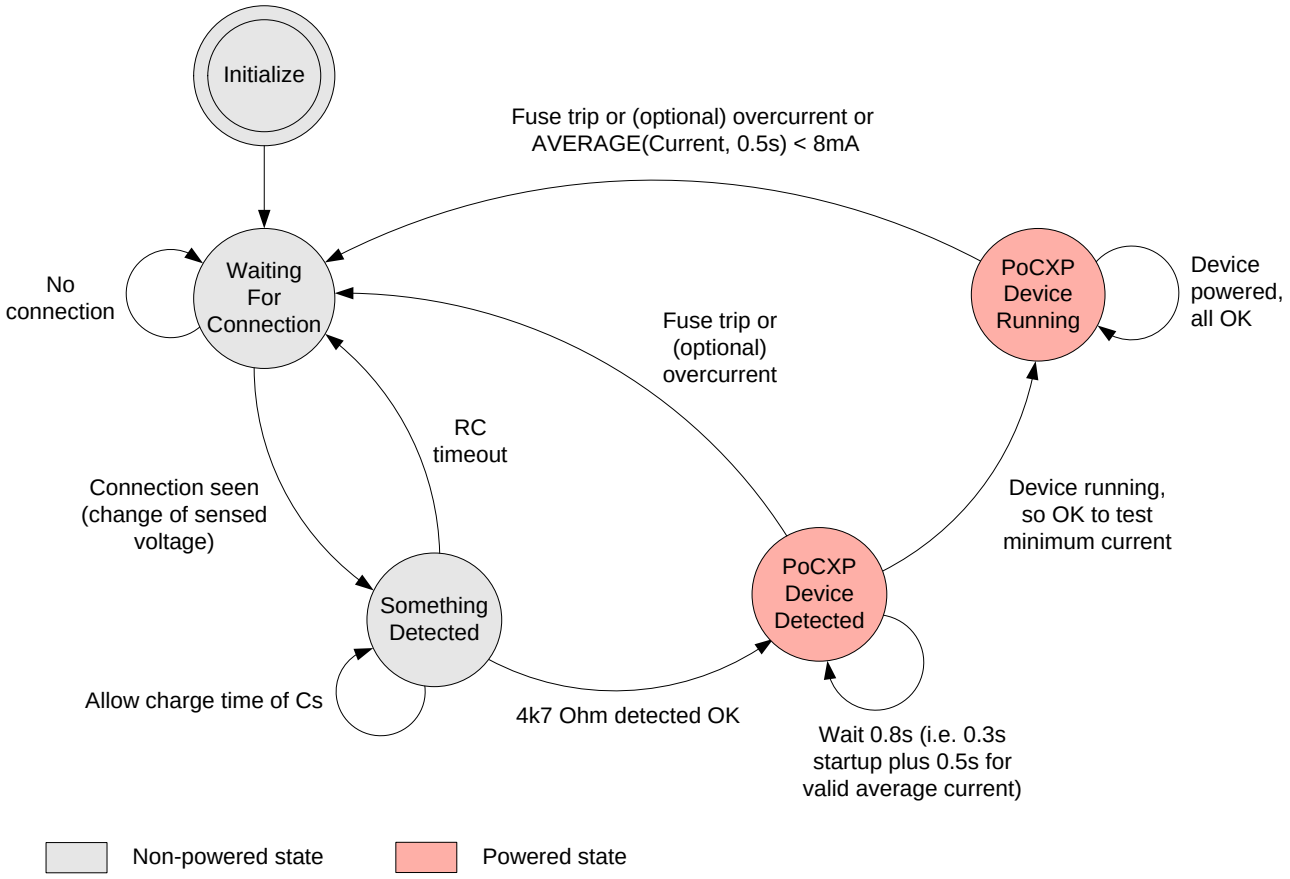


Figure 14 — Simplified example state diagram

8.4.5 Additional Features

It is recommended that the Host supports a mode to allow power to be disconnected from the Device under user control, e.g. to enable standby modes, or to perform a “hard reset” on the Device.

Similarly it is recommended that the Host supports a mode to allow power to be re-connected to the Device, by forcing the Host to re-detect if a PoCXP Device is connected.

Comment: Note that the Host cannot support a mode where it always applies power, as this would violate section 8.4.4.

9 Link Protocol

9.1 Overview

The link protocol defines a point to point interface between a Host and Device. The link consists of a Master physical connection and optional extension connections and a high speed upconnection.

All data is transferred using 8B/10B coded packets.

For each connection the protocol defines a set of logical channels carrying specific data such as stream data (e.g. images), real time triggers and Device control.

The Device is controlled by the Host via registers. Host register access is provided via a control channel.

When no high speed upconnection is in use, write access and therefore control is exclusively provided via the master connection making the connected Host connection the control master. Extension connections provide read-only access allowing Device and connection discovery. When the high speed upconnection is in use it replaces the master low speed upconnection to provide Device control.

The logical channels when no high speed upconnection is in use are shown in the following figure:

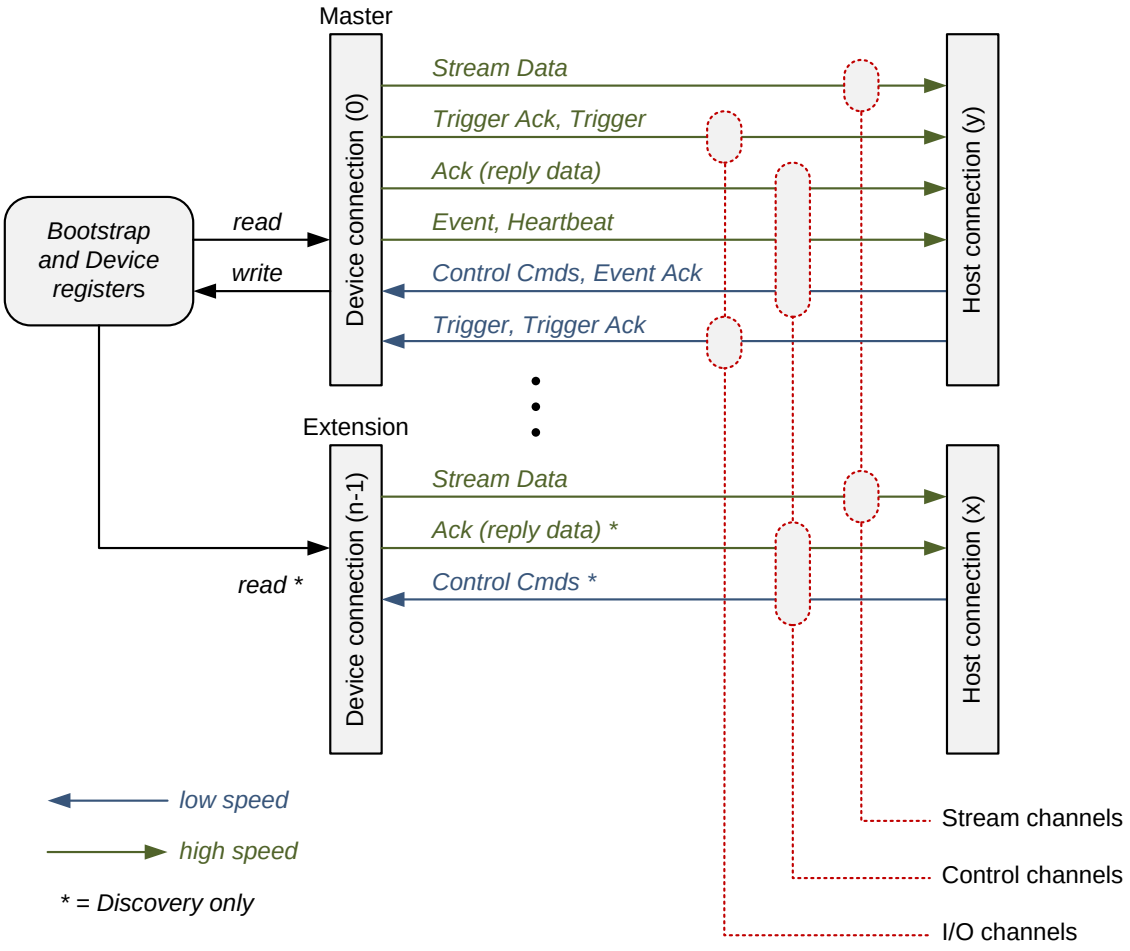


Figure 15 — Link Overview

The logical channels when the high speed upconnection is in use are shown in the following figure:

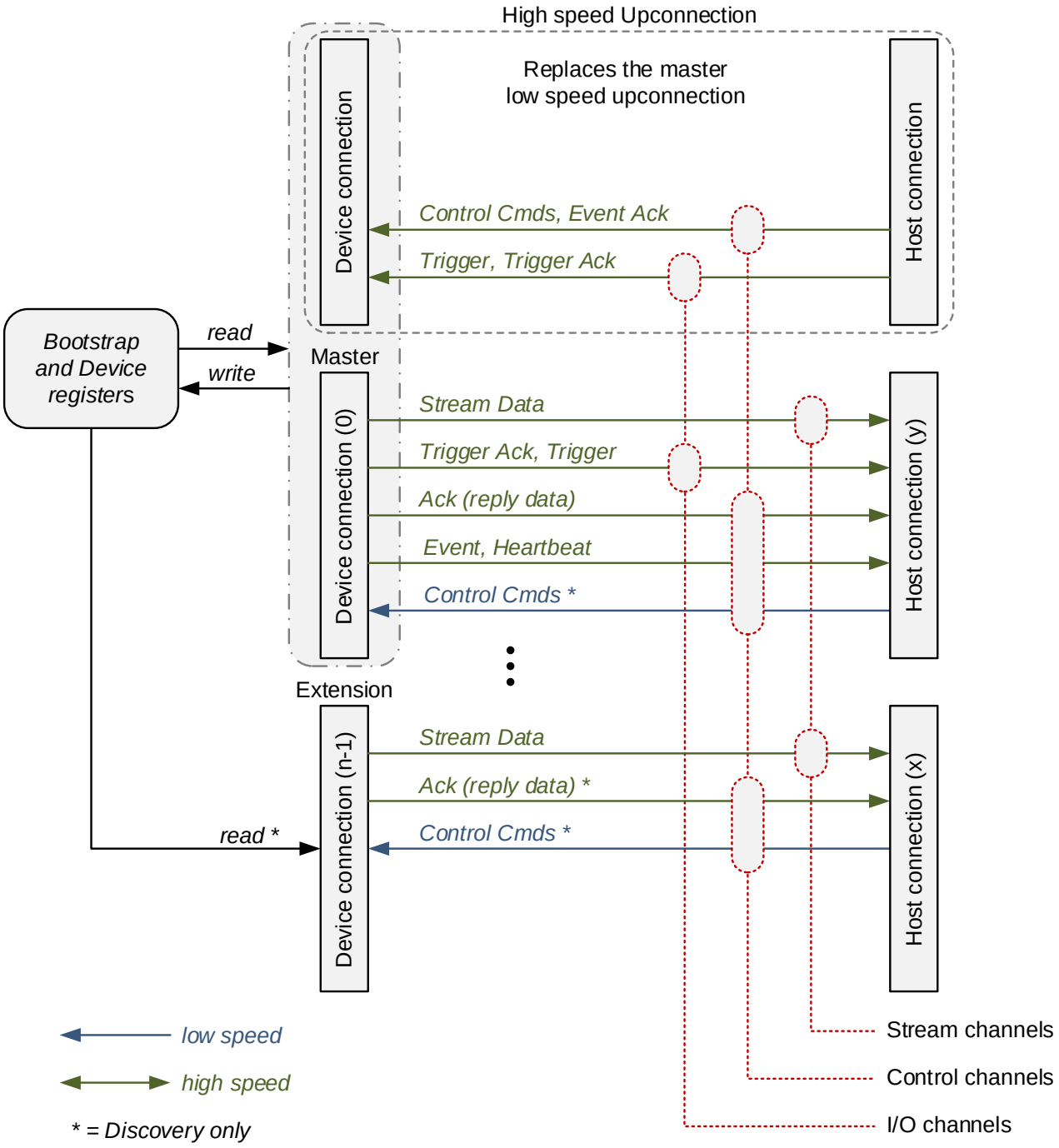


Figure 16 — Link Overview with High Speed Upconnection

Comment: These figures are best viewed in color.

9.2 Transport Layer

9.2.1 Packet Transmission Format

Both the upconnection and downconnection shall use 8B/10B coding as defined in section 2.1.

Besides the 256 data characters D0.0 – D31.7, K-codes are used for formatting and indication purposes. An overview of K-code usage is listed in the following table:

Table 11 — K-code usage

K-code	Function
K27.7	Start of data packet indication
K28.6	I/O acknowledgment
K28.1	Used for alignment
K28.2	Trigger indication
K28.3	Stream marker – see section 10.2
K28.4	Trigger indication
K28.5	Used for alignment
K29.7	End of data packet indication

Transmission is organized in groups of 4 characters called a word, except for low speed connection trigger packets which are six characters.

A word consists of 4 consecutive 8B/10B characters labeled *P0*, *P1*, *P2* and *P3* and are transmitted in the stated order (i.e. P0 first). The receiver shall perform word alignment in order to successfully decode the packets.

The bit transmission order shall be according to the 8B/10B coding scheme: 8 bits labeled { ‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’, ‘G’, ‘H’ } and a data/control selection are coded into a 10 bit symbol with bits labeled { ‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘i’, ‘f’, ‘g’, ‘h’, ‘j’ }. Prior to coding the bit labeled ‘A’ is the LSB. After coding the bit labeled ‘a’ is transmitted first. This order is shown in the following figure:

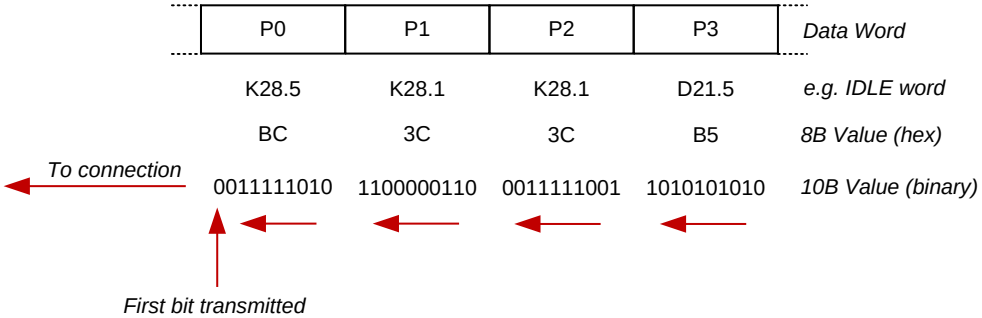


Figure 17 — Bit Order

Unique single values represented by multiple bytes (e.g. a 32 bit address) shall be transmitted using big endian byte order.

9.2.2 Error Handling

The link protocol (i.e. packet structure, overall control) is designed to be immune to single bit errors by a combination of careful choice of packet coding, and sending key information multiple times.

Any errors in data transfer are detectable by a CRC.

Error counters provide feedback of errors that the protocol can detect.

9.2.2.1 Duplicated Characters

Many words duplicate the same value in P0, P1, P2 and P3. The receiver shall decode these multiple characters in a way that provides immunity to single bit errors.

Comment: In general information is sent four times rather than the usual three times. This is simply to maintain word alignment, and three of the characters are sufficient to "vote" and cope with single bit errors. However having four duplicates may allow sophisticated receivers to decode the packet correctly in the event of multiple bit errors, but that is a quality of implementation decision beyond the requirements of the standard.

9.2.2.2 CRC

A CRC is used to protect stream data packets and control packets.

For control packets the receiver shall decode these CRCs and in the event of a CRC error shall send a negative acknowledgment indicating the error.

Comment: How this acknowledgment is handled is beyond the scope of this standard, but it is recommended that the command is resent up to three times, and if the third retry fails an error message would be passed up to the application to deal with.

For stream data packets the receiver shall decode these CRCs for error monitoring purposes.

Comment: How CRC errors are handled is beyond the scope of this standard, but typically an error message would be passed up to the application to deal with, either after every error, or after the number of errors exceeds a threshold.

All CRCs defined in the link protocol are 32 bit CRCs calculated over a specified block of data.

Polynomial: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Seed: 0xFFFFFFFF

Comment: This is the CRC-32 polynomial defined by IEEE-802.3 (Ethernet). Note that CoaXPress does not use the CRC as defined by IEEE-802.3 – the CRC calculation is defined below.

The CRC shall be calculated over the data using the following bit order:

- Data bit 0 (LSB) first within a character; bit 7 (MSB) last.
- Character P0 first within a word; character P3 last.
- Word 0 first within a packet; word *N*-1 last, where *N* is the number of words over which the CRC needs to be calculated (see individual packet definitions).

The calculated CRC shall be sent over the connection in the following bit order: The MSB of the CRC sent in character P0 bit 0 through to the LSB in character P3 bit 7.

Comment: This ensures that the MSB is sent first over the serial connection, to correctly give a zero value in the receiver CRC register after both data and the transmitted CRC have been processed.

The bit order is shown in the following figure:

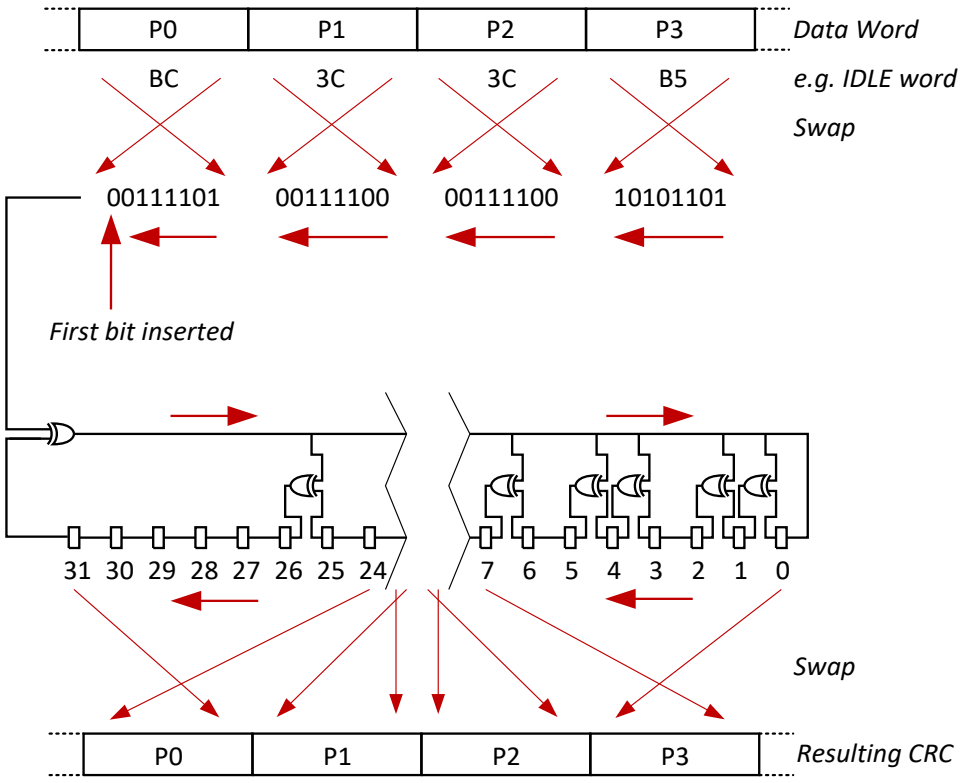


Figure 18 — CRC Calculation

Comment: For clarity, this shows a serial implementation.

The CRC calculation shall not include IDLE words used to stretch data transmission. The K28.3 stream marker shall be treated as a D28.3 for CRC calculation, i.e. the K-Code flag is ignored.

Comment: To aid understanding, a complete control command packet without tag (a read of address 0) is shown here, with the resulting CRC shown in red: K27.7 K27.7 K27.7 K27.7 0x02 0x02 0x02 0x02 0x00 0x00 0x00 0x04 0x00 0x00 0x00 0x00 0x56 0x86 0x5D 0x6F K29.7 K29.7 K29.7 K29.7.

9.2.2.3 Error Counters

The receiver shall maintain counters to provide reporting of errors. The number of bits used for the counters is a quality of implementation decision, although 32 bits is recommended.

A dedicated overflow bit shall be set when a counter overflows.

Comment: For receivers that have limited FPGA resources, the overflow bit could be implemented as a sticky error status bit, in which case there would be no need for a counter register.

The following error counters are defined:

StreamDataPacketCrc[CxpConnectionSelector] shall be incremented when a CRC error is detected in a stream data packet.

ControlPacketCrc[CxpConnectionSelector] shall be incremented when a CRC error is detected in a control packet.

EventPacketCrc[CxpConnectionSelector] shall be incremented when a CRC error is detected in an event packet.

DuplicatedCharactersCorrected[CxpConnectionSelector] shall be incremented when the values of P0, P1, P2, and P3 are not equal but the receiver is able to decode the value.

DuplicatedCharactersUncorrected[CxpConnectionSelector] shall be incremented when the values of P0, P1, P2, and P3 are not equal and the receiver is not able to decode the value.

Encoding[CxpConnectionSelector] shall be incremented when one of the following occurs:

- The running disparity rules are violated.
- The received 10-bit character cannot be mapped to a valid 8B/10B character.

ConnectionLockLoss[CxpConnectionSelector] shall be incremented when connection lock is lost.

9.2.3 Packet Types

Information is transferred in both directions using packets.

Packet types defined by the link protocol are listed in the following table:

Table 12 — Packet types

Channel	Packet Type	Usage	
		Downconnection	Upconnection
I/O	I/O packets = { Trigger, Ack }	✓	✓
Stream	Data packet, type = { Stream }	✓	-
Control	Data packet, type = { Control data, Connection Test }	✓	✓
Control	Data packet, type = { Event, Heartbeat }	✓	

Comment: In principle stream data could be sent on the upconnection, but the current discovery procedure does not allow for this, and the standard is written with only the Device generating stream data.

The start of a packet is indicated by a word containing K-codes:

- I/O trigger packets are transmitted as a specific start indication (K28.2 or K28.4), followed by fixed sized data content.
- I/O acknowledgements use a dedicated start indication (K28.6) followed by a single word code.
- All other packet types use a more generic data packet consisting of a start indication (K27.7), the packet payload and an end indication (K29.7).

9.2.4 Packet Transmission Order

The link protocol defines three levels of priority as indicated in the following table:

Table 13 — Packet transmission priority

Priority Level	Packet type
0 (highest)	Trigger
1	I/O acknowledgment (for trigger packet)
2 (lowest)	All other packets

To maintain guaranteed trigger timing, trigger packets and trigger acknowledges have a defined higher priority. A high priority packet shall be inserted into a lower priority packet that is being transmitted. The insertion shall take place at a word boundary with an exception for the low speed connection trigger packet which shall take place at a character boundary.

Comment: Allowing character insertion of the low speed trigger packet allows minimal trigger latency given the relatively low speed of the connection.

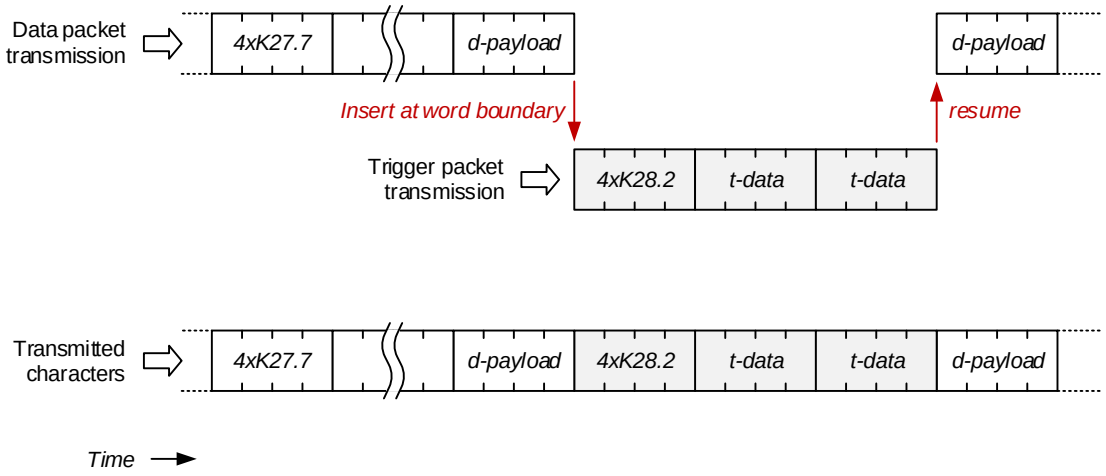
Transmission of the lower priority packet shall be resumed after the transmission of the inserted packet is completed.

A packet shall not be inserted into another packet at the same priority level. However, on completion of a packet at priority level two, Devices and Hosts are free to choose the next level two packet to send to optimize performance.

Comment: A Device may choose to send a small control packet acknowledgement as soon as the next image data packet finishes, to allow good GUI response in a user application. However if the control packet was to read a large data block, and its image data FIFOs were nearly full, it may continue to send image data and delay the control acknowledge.

Note: Section 9.5.3 requires packets within one stream to be transmitted in order.

An example of priority-based transmission on the high speed connection is illustrated in the following figure for a trigger packet being inserted into a data packet:



Explanation of terms used:
d-payload = Data packet payload
t-data = Trigger packet data

Figure 19 — High speed connection packet transmission example

The following figure shows the same sequence on the low speed connection, where the trigger packet is inserted on a character boundary:

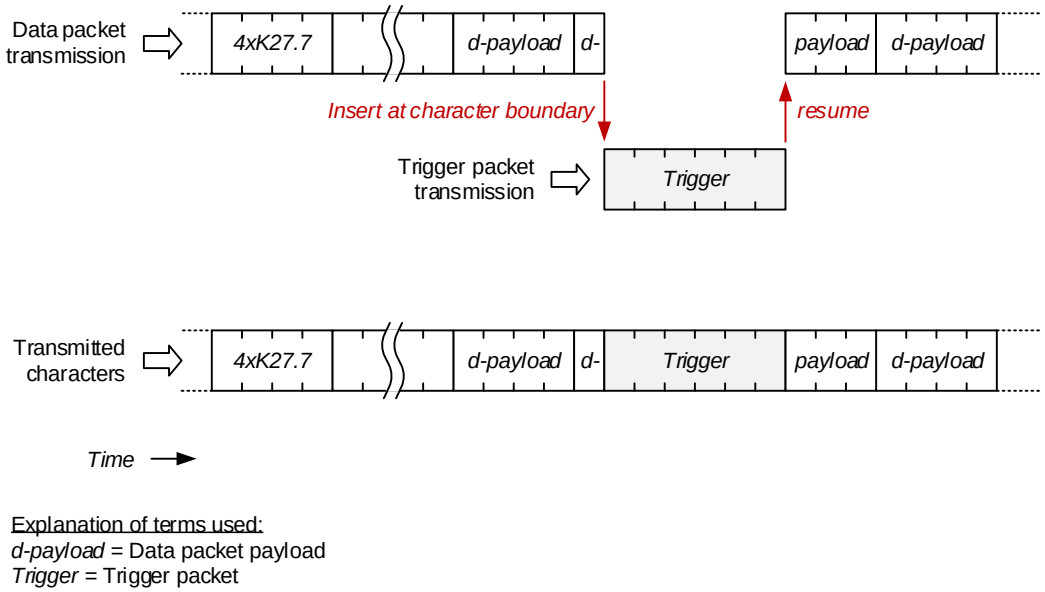


Figure 20 — Low speed connection packet transmission example

The following figure shows an additional example on the low speed connection, with two trigger packets inserted on a character boundary:

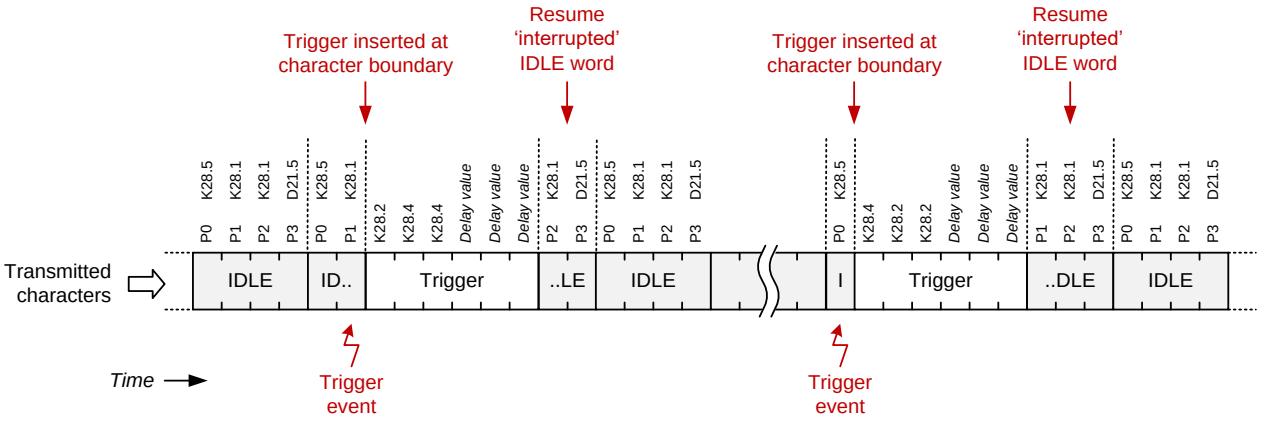


Figure 21 — Low speed connection trigger insertion example

9.2.5 Idle Transmission

A connection is idle when no packet transmission takes place. This idle time shall be filled with IDLE words. The format of this word is defined in the following table:

Table 14 — IDLE word format

Word	P0	P1	P2	P3
IDLE	K28.5	K28.1	K28.1	D21.5

Comment: The IDLE word serves the following purposes at the receiver:

- *Allows rapid bit clock lock by means of the high frequency (D21.5) data content.*
- *Provides low frequency (K28.5, K28.1) and high frequency (D21.5) data content that can serve the line equalizer.*
- *Allows character and word alignment using the K28.5 character that is only used as character P0 in the idle word.*
- *Allows monitoring of the alignment status once alignment is achieved.*

9.2.5.1 Idle Usage

Both high and low speed connection transmitters shall send the IDLE word when the connection is idle. For the avoidance of doubt, low speed connection transmitters shall continue to send the IDLE word on extension connections, and on the master connection if the optional high speed upconnection is in use.

For proper operation of a high speed connection an IDLE word shall be transmitted at least once every 100 words.

Comment: Therefore 99 words can be sent between IDLE words. Therefore in general IDLE words will be inserted into packets in the transmitting device, but they will be stripped out at the receiving device before the packet is decoded.

For proper operation of a low speed connection an IDLE word shall be transmitted at least once every 10,000 words.

Comment: Although the 10,000 word rule makes it less common than with the high speed connection, the rule still means that IDLE words may similarly be inserted into packets in the transmitting device.

To maintain trigger performance, IDLE words shall not be inserted into trigger or trigger acknowledge packets.

9.2.5.2 Optional Uses of Idle

Additonal IDLE words can be added to packets as defined in this section. These IDLE words should be ignored by the receiver and are not included in the CRC.

The transmission of a packet on the high speed connection (other than trigger or trigger acknowledge) can be stretched by inserting additional IDLE words.

Comment: The possibility to insert IDLE words relaxes transmitter buffer requirements – i.e. the entire packet does not need to be available when the packet transmission commences. This could be useful with control channel reads. However for data stream packets it is expected (but not mandatory) that the entire packet would be buffered, so no additional IDLE words would need to be inserted.

Additional IDLE words to stretch packets shall not be inserted into a packet on the low speed connection.

Comment: There is no need to stretch low speed connection packets because of the slow data rate, and banning this simplifies implementation.

IDLE words can also be inserted into the data stream to simplify data alignment in a system using an internal bus wider than 32 bits.

Comment: Inserting other values such as 0x0000 would not result in a CoaXPress-compliant implementation. Note that this statement does not conflict with section 10.4.2 where the end of a line in a CXP Data Stream is padded with zero bits.

9.3 I/O Channel

The I/O channel is defined for the Master connection (connection 0) only.

9.3.1 Trigger

Trigger can be in 2 directions: From Host to Device, and from Device to Host.

Comment: The common use case is for trigger from Host to Device.

Triggers have the highest priority and shall be inserted into any lower priority transmission packet at the appropriate transmission boundary.

Trigger packets shall be acknowledged (see section 9.3.2).

Note: CoaXPress uses a serial link so If a system has multiple trigger sources which could occur very close together, or even simultaneously, then it is not possible to guarantee that all these trigger sources can be sent with the latency defined by this standard. In that case it is a system architecture decision to prioritize how the system manages the triggers.

9.3.1.1 Low Speed Connection Trigger

The event that results in a trigger being sent could occur at any time, but the trigger packet cannot be sent until the next transmission boundary. For the low speed connection the trigger packet can start at the next character boundary (see section 9.2.4).

Comment: Allowing character alignment of the low speed trigger packet allows minimal trigger latency given the relatively low speed of the connection.

To minimize trigger jitter, the time between the trigger event and the trigger packet being sent is coded into the packet as a delay value. The receiver can then use this delay to recreate the trigger event with low jitter and a fixed latency.

For the low speed connection the delay value is 239 minus the number of units of 1/24 the low speed connection bit interval (2.00 ns at 20.83˙ Mbps, or 1.00 ns at 41.6˙ Mbps) between the trigger event and the start of the trigger packet. See Figure 22.

Comment : The following explains the value 239 given above, using 20.83˙ Mbps as an example.

Transmission timing accuracy is limited to character units. A character contains 10 bit intervals (after 8B/10B coding).

*A bit interval equals 1 / 20.83˙ Mbps = 48 ns. The character transmission time is therefore 10 * 48 ns.*

The trigger timing correction delay value is coded in units of 2 ns. A full character interval is therefore 480 / 2 = 240 delay value codes wide.

Therefore to encode the trigger delay correction value, a range of 0 to 239 is needed.

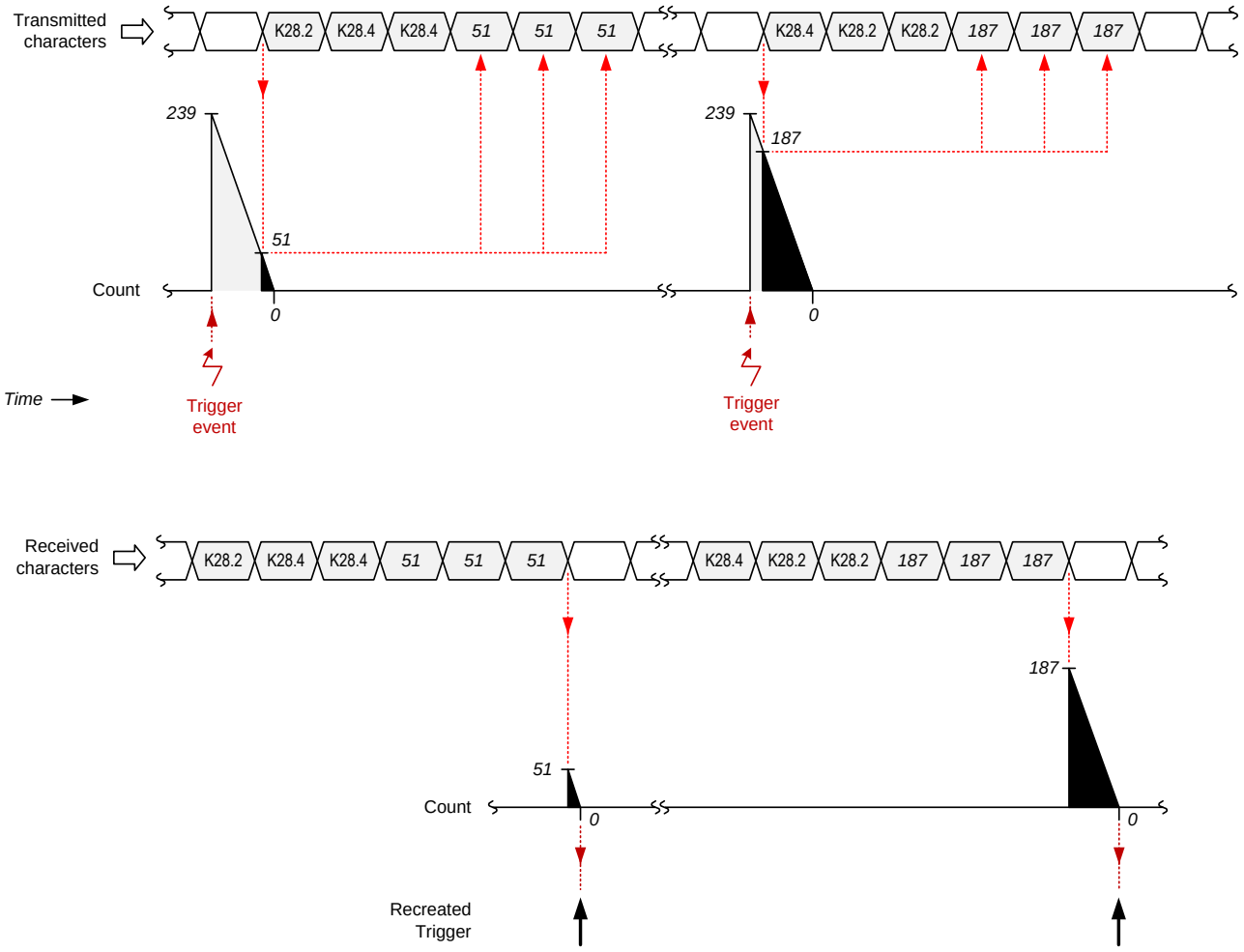


Figure 22 — Low speed trigger example

The packet definition in Table 15 is used by CoaXPress version 2.0 and earlier, and in version 2.1 and higher if *FeatureControlRegister* bit *ExtraLsTriggerEnable* is set to 0:

Table 15 — Low speed connection trigger packet format (default)

Char	Content	Description
0 – 2	*Either* K28.2 K28.4 K28.4	Trigger packet indication – LinkTrigger0
	or K28.4 K28.2 K28.2	Trigger packet indication – LinkTrigger1
3 – 5	3x Delay	The delay value is 239 minus the number of units of 1/24 the low speed connection bit interval between the trigger event and the start of the trigger packet. The Host can use a coarser time unit by setting lower bits of this value fixed at 0 or giving it a bias. The Device can use a coarser time unit by using less bits of this value (discard non used lower bits of this value or use rounding). The chosen accuracy and usage of this timing correction value is left as a quality of implementation at both the Host and Device.

Packet size: 6 characters (6 bytes).

Comment: In CXP v1.x, LinkTrigger0 was called "rising edge" and LinkTrigger1 "falling edge".

The packet definition in Table 16 is used by CoaXPress version 2.1 and higher if *FeatureControlRegister* bit *ExtraLsTriggerEnable* is set to 1, to allow additional LinkTrigger modes by re-purposing the LSB of the delay value:

Table 16 — Low speed connection trigger packet format (ExtraLsTrigger)

Char	Content	Description
0 – 2	*Either* K28.2 K28.4 K28.4	If the LSB of the delay value is set to 0: Trigger packet indication – LinkTrigger0 If the LSB of the delay value is set to 1: Trigger packet indication – LinkTrigger2
	or K28.4 K28.2 K28.2	If the LSB of the delay value is set to 0: Trigger packet indication – LinkTrigger1 If the LSB of the Delay value is set to 1: Trigger packet indication – LinkTrigger3
3 – 5	3x Delay	The LSB of the delay value is re-purposed to define additional LinkTrigger modes – see the definitions above of characters 0 – 2. In processing the delay value to correct trigger jitter, the Device shall set the LSB of this value fixed at 0. Apart from these conditions, the definition of the delay is identical to Table 15 above.

Packet size: 6 characters (6 bytes).

Comment : Additional LinkTrigger modes are required in a typical line scan implementation using GenDC, if pseudo-frame trigger, line trigger and exposure control are required using the low speed link. See section 10.5.3.

9.3.1.2 High Speed Connection Trigger

Similarly for the high speed connection the next transmission boundary is the next word (4 characters), and the delay value is three minus the number of whole characters between the trigger event and the start of the trigger packet.

Table 17 — High speed connection trigger packet format

Word	Content	Description
0	4x K28.2	Trigger packet indication
1	4x Delay	The delay value is three minus the number of whole characters between the trigger event and the start of this trigger packet. Usage is optional, when not used set to 0.
2	4x LinkTriggerN	A number N between 0 and 15 which defines the trigger, corresponding to LinkTrigger0 to LinkTrigger15.

Packet size: 3 words (12 bytes).

Comment : The addition of the LinkTriggerN field to this packet means that it is not compatible with CXP v1.x, but as there are no CXP v1.x products registered with JIIA that support high speed connection triggers, simply changing the packet avoids the complexity of a version-based conditional packet format.

9.3.2 I/O Acknowledgments

Trigger packets shall be acknowledged with an I/O acknowledgment packet.

Table 18 — I/O acknowledgment packet format

Word	Content	Description
0	4x K28.6	I/O acknowledgment packet indication
1	4x Code	Acknowledgment code, repeated 4 times: 0x01 Trigger packet received OK

Packet size: 2 words (8 bytes).

The Device or Host transmitting a trigger packet shall use the following rules:

- After completion of a trigger packet transmission it shall not send a new trigger packet until it has received an acknowledgment from the Host or Device receiving the packet.
- However if it does not receive an acknowledgment for more than a defined time (transmission timeout, see below) it can resend the last trigger packet or can send a new trigger packet.

Transmission timeout rules:

- Trigger packet sent on low speed connection, acknowledgment on high speed connection (therefore I/O is from Host to Device): The timeout shall be one character on the low speed connection.
- Trigger packet sent on high speed connection, acknowledgment on low speed connection (therefore I/O is from Device to Host): The time to send the acknowledgment depends on the usage on the low speed connection (see comment below). Therefore it is recommended that the Device has a register controlling the timeout, set to a default value by the Device based on its I/O usage, but that can be overridden by the Host.
- Trigger packet sent on high speed connection, acknowledgment on high speed connection (therefore high speed upconnection in use): The timeout shall be 480ns.

Comment: The decision on what to do in the event of a trigger packet being lost is system-dependent, and likely to be controlled by the Host. A Device may need a register that allows the Host to determine what it does.

Comment: Note that low speed connection has limited bandwidth, and the system designer needs to choose trigger rates in both directions such that the bandwidth is not exceeded. For instance if it is necessary to run the Host to Device trigger at near to the maximum capacity (e.g. for a line trigger on a line scan device), then the Device should not send triggers to the Host, as these would require trigger acknowledgments which there would not be time to send. This situation would be detected by transmission timeouts in the Device. Of course this is not a limitation when using a high speed upconnection which dedicates its bandwidth to the transmission of triggers.

9.4 Data Packet Definition

Table 19 — Data packet transmission format

Word	Content	Description
0	4x K27.7	Start of packet indication
1	4x Type	Data packet type: 0x00 Reserved for extended types 0x01 Indicates a stream data packet 0x02 Indicates control command with no tag 0x03 Indicates control acknowledge with no tag 0x04 Indicates a connection test packet 0x05 Indicates control command with tag 0x06 Indicates control acknowledge with tag 0x07 Indicates event packet 0x08 Indicates event acknowledge 0x09 Indicates a heartbeat packet Other values are reserved for future use.
2… *D*+1	Data (4 bytes)	Data packet payload of *D* data words
D+2	4x K29.7	End of packet indication

Packet size: *D* + 3 words ((*D* * 4) + 12 bytes).

The following sections define these data packet payloads.

9.5 Stream Data Packet

Stream data packets are used to send stream data (see section 10.1) across the link.

The packet transfer layer is responsible for forming packets from a stream by chopping the stream into blocks and adding a stream packet header and trailer to form a packet. This is shown diagrammatically in the following figure:

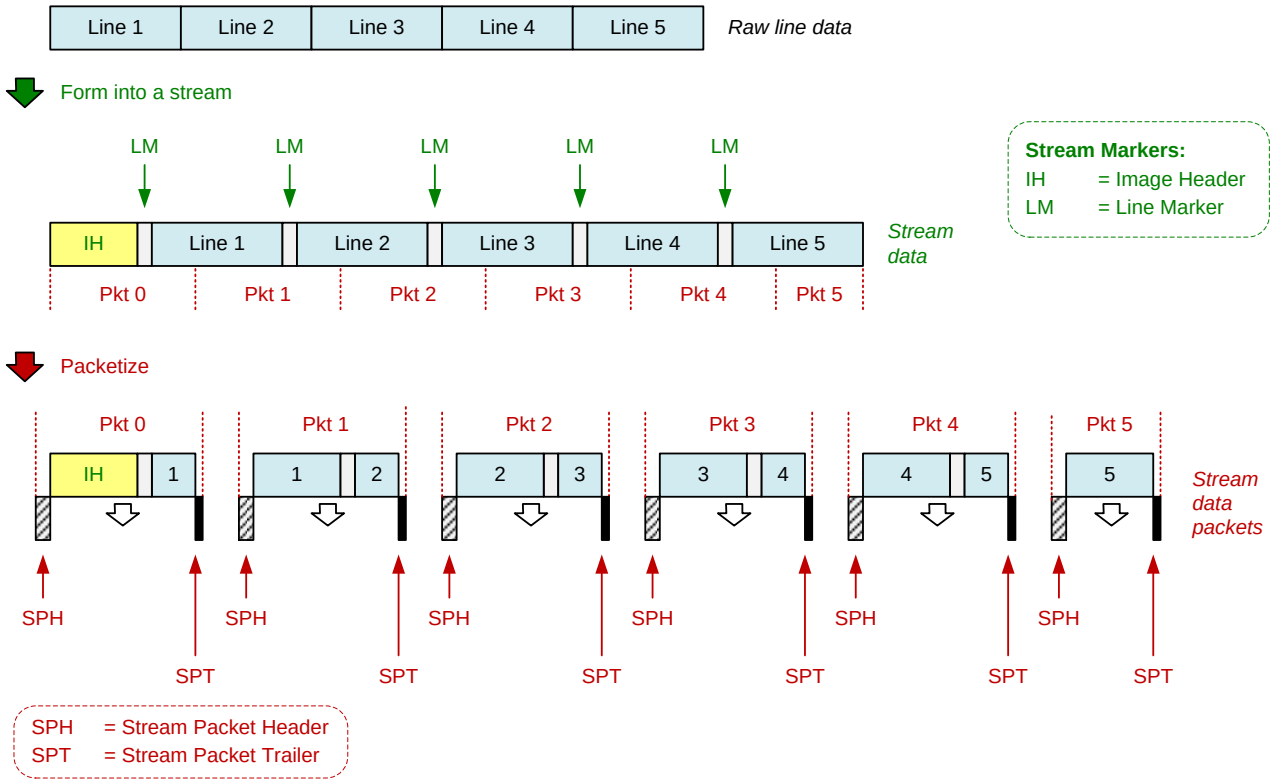


Figure 23 — Packet formation in Device – image data example

Comment: This figure is best viewed in color.

Comment: If a new frame immediately followed this one, then the image header for the new frame would typically go into packet 5, to maximize link utilization.

The packet transfer layer is also responsible for sharing all the packets from the Device across the available connections, as described in the following sections.

9.5.1 Stream Data Payload Format

The payload format is shown in the following table:

Table 20 — Stream data payload format

Word	Content	Description	
	4x K27.7	*Start of packet indication (see section 9.4).*	SPH
	4x 0x01	*Stream data packet indication.*	
0	4x Stream ID	Unique stream ID that this packet contains data for.	
1	4x Packet Tag	8 bit tag. Incremented for each packet with this stream ID, wraparound to 0 at 0xFF.	
2	4x DsizeP(15:8)	16 bit value representing the number of stream data words *N* per packet.	
3	4x DsizeP(7:0)		
4 to *N*+3	Stream data	*N* words of stream data.	
N+4	CRC	32-bit CRC calculated over the stream data 4 to *N*+3.	SPT
	4x K29.7	*End of packet indication.*	

Packet size: *N* + 8 words ((*N* * 4) + 32 bytes).

Note: The Stream Packet Header (SPH) and Stream Packet Trailer (SPT) are shown in the right hand column.

Comment: The Stream ID is the same as in the image header (e.g. section 10.4.6.2). This allows the packet decoder in the Host to read the Stream ID and therefore send the packet's contents to the appropriate buffer for that stream. The Stream ID will still be in that buffer (in the image header), and will be needed to process the stream.

9.5.2 Packet Size

The total stream packet size (i.e. from the first K27.7 to the last K29.7) shall not be more than the size set in *StreamPacketSizeMax* during Device discovery (see section 12.1.6).

The Device can use any packet size it wants to up to this size.

Table 21 — Packet Overhead

Packet Size	Packet Overhead
2 Kbyte	8/512 = 1.5%
4 Kbyte	8/1024 = 0.8%
8 Kbyte	8/2048 = 0.4%

Comment: The largest supported packet size is recommended for maximum efficiency. A packet size of around 2 Kbytes gives a reasonably low packet overhead combined with practical buffer sizes in FPGAs. Depending on the image size compared to the packet size, note that the last data packet in an image could be very small – potentially only transferring 1 word. Also note that bootstrap register StreamPacketSizeMax is set to zero during Device discovery; therefore the Device cannot send any packets until Device discovery completes and the register is set to the correct value.

9.5.3 Packet Order and Packet Tag

Packets within one stream shall be transmitted in order, i.e. the first packet in a stream shall have packet tag 0, then the data immediately following shall be the next packet with tag 1, then tag 2 etc.

Comment: The tag allows the Host to check for missing packets, and to confirm the correct packet order when multiple connections are in use. Given that the physical layer is deterministic, the protocol does not need to allow for packets arriving out of order.

The packet tag shall only be reset back to zero as part of a *ConnectionReset* (see section 12.3.33) or a write to *ConnectionConfig* (see section 12.3.38).

Comment: Therefore the packet tag is not reset for example by an AcquisitionStart or AcquisitionStop, a start or end of a frame, or a change of image size.

9.5.4 Combining Multiple Streams

After completing one packet transmission, the Device can send the next packet from any stream with available data.

It is the responsibility of the Device to prioritize packet transmission order based on its available buffer sizes.

Comment: A simple Device may only have one stream, so does not need this process.

This is shown in the following figure:

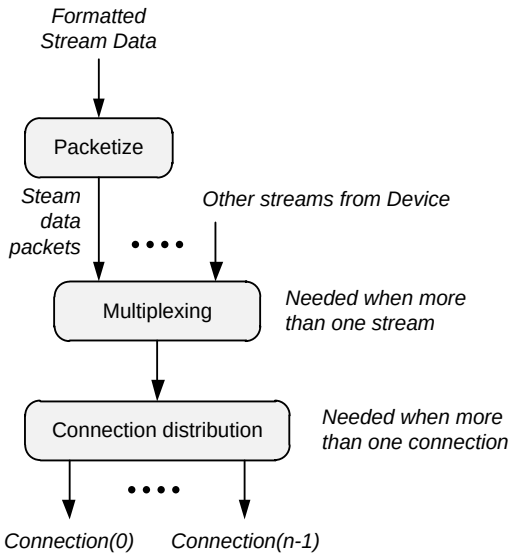
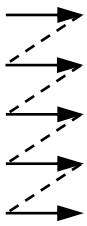


Figure 24 — Packet processing in Device

9.5.5 Packet Transfer over Multiple Connections

Successive packets in a stream, or a combined stream (see section 9.5.4), shall be written to the next connection in ascending order of Connection ID, starting at connection 0, as shown in the following table. Also see Figure 24.

Table 22 — Packet order on connections – example

Connection 0	Connection 1	Connection 2	Connection 3	Time
Packet 0	Packet 1	Packet 2	Packet 3	
Packet 4	Packet 5	Packet 6	Packet 7	
Packet 8	Packet 9	Packet 10	Packet 11	
Packet 12	Packet 13	Packet 14	Packet 15	
…	…	…	…	

Note that the Host knows the connection order from the bootstrap registers *DeviceConnectionID* it reads during Device discovery (see section 12.1.3).

Comment: A simple Device may only have one connection, so does not need this process. The defined order means that the Host knows which connection the next packet will be on, without needing to read the packet header to find out.

Note: Section 9.5.3 requires packets within one stream to be transmitted in order. This means that when for example a Control Acknowledge is sent on Connection 0 which delays the next stream packet on that connection, that the related stream packets on the other connections must also be delayed by at least the same amount to maintain the required packet order.

The packet writing order shall only be reset to start at connection 0 as part of a *ConnectionReset* (see section 12.3.33) or a write to *ConnectionConfig* (see section 12.3.38).

Comment: Therefore the packet writing order is not reset for example by an AcquisitionStart or AcquisitionStop, a start or end of a frame, or a change of image size.

9.6 Control Channel

The control channel is used to provide register access through specific control commands, transmitted from Host to Device, and the resulting acknowledgment messages, transmitted from the Device to the Host.

The following command messages are defined:

- Memory read of *B* bytes.
- Memory write of *B* bytes.

Acknowledgment messages contain an acknowledgment code. This code indicates:

- Success, meaning that the command executed (or is still executing) OK, and in the case of a memory read, the requested data is attached.
- Logical Errors, meaning that the command packet was received OK but attempted an invalid operation.
- Physical Errors, meaning that the command packet was corrupt.

Comment: The processing of logical and physical errors by the Host, and hence the application, may be very different.

Control commands shall only be sent by the Host. Acknowledgment messages shall only be sent by the Device as a response to a received control command.

The master connection control channel shall provide full register access while extension connection control channels shall be restricted to read-only access during Device discovery for registers shown with an “E” in Table 54.

Comment: The control tasks defined in the current standard are limited to Device register access by the Host.

Two versions of control packets are defined, with and without a tag field. The version with a tag was added in CXP version 2.0 to provide better recovery from error conditions. The original version without tag is still defined, and is needed for compatibility with version 1.x products and for discovery.

9.6.1 Control Transactions

A control transaction is composed of transmitted command, the execution of that command, and finally[3] an acknowledgment as shown schematically in the following figure:

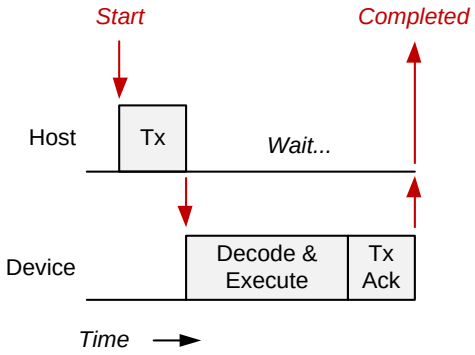


Figure 25 — Control transaction – standard

If the transaction will take longer than the allowed timeout (200ms, defined in section 9.6.1.1) a single wait acknowledgment can be sent to specify the required time (maximum 10 seconds, defined in section 9.6.3), as shown schematically in the following figure:

[3] There are two exceptions: When the connection speed is changed by writing to *ConnectionConfig*, the Device is required to acknowledge the write *before* changing the speed. See section 12.3.38. Also a write to *ConnectionReset* is “fire and forget”. See section 12.3.33.

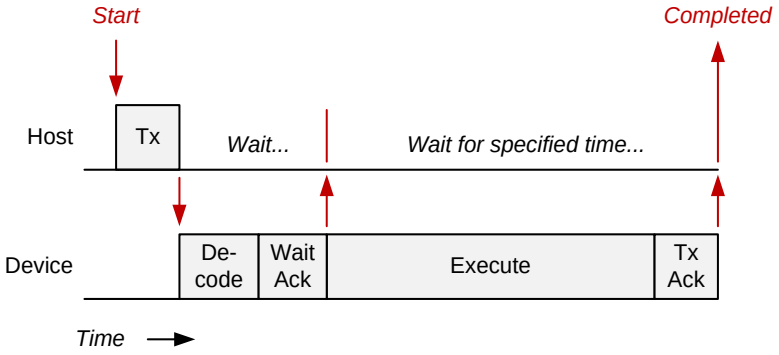


Figure 26 — Control transaction – wait acknowledgment

Comment: If a Device needs more than 10 seconds for a certain action, or does not know how long an action will take, it can use one control transaction to start the action, and then poll a register to check for completion. A GenICam command supports this method, and suitable XML code can specify the polling register and if necessary disable other Device features until the command has completed. See the GenICam standard for further information.

9.6.1.1 Transaction Rules

The Host shall implement the following rules for control transactions:

- After completion of command transmission the Host shall not send a new command until it has received a final acknowledgment from the Device.
- However the Host can resend the last command or can send a new command:
  - If the Host does not receive an acknowledgment (final acknowledgment or wait acknowledgment) for more than 200ms (transmission timeout), or
  - If following a wait acknowledgment the Host does not receive a final acknowledgment within the time given in the wait acknowledgement.

The Device shall implement the following rules for control transactions:

- On receiving a valid command the Device shall execute the command and transmit a final acknowledgment after command execution.
- On receiving an invalid command the appropriate acknowledgment shall be transmitted immediately and the command shall be discarded.
- Command execution time and the time required to transmit the final acknowledgment shall not exceed 200ms, which is the transmission timeout.
- However if the Device needs more than 200ms it shall transmit one wait acknowledgment within 200ms to inform the Host that it is processing the last command. The wait acknowledgment shall specify the maximum time that the Host should wait for the final acknowledgement after receiving the wait acknowledgment. The Device shall then send one final acknowledgment within the specified time, and no more wait acknowledgments shall be sent before the final acknowledgment.

Comment: Wait acknowledgments are not allowed for accesses to bootstrap registers. See section 12.3.3.

9.6.1.2 Tag Rules

The Host and Device shall implement the following rules for using tags in control transactions:

Basic rules:

- If bootstrap register *VersionUsed* indicates 1.0 or 1.1, or the Device does not support 2.0 or greater and therefore does not implement *VersionUsed*, then communications shall be without tag.
- If bootstrap register *VersionUsed* indicates 2.0 or greater, then communications shall be with tag.

Comment: This means that version 2.0 and greater systems always use the tag, after starting without tag during discovery to allow backwards compatibility. See discovery section 12.1.4.

- A Device shall respond to a Control Command packet without tag by an Acknowledgement packet without tag.
- A Device shall respond to a Control Command packet with tag by an Acknowledgement packet with tag.

Device and Host using Control Command and Acknowledgement packets with tag shall use the following rules:

- The tag shall be a free running number incremented for each new command.
- A Device shall send the tag in the acknowledge packet with the same tag it received in the command packet.
- The Host shall increment the tag by one for a new command. It shall wrap to 0x00 from 0xFF.
- The packet tag shall only be reset back to zero as part of a *ConnectionReset* (see section 12.3.28).

Error Recovery rules and recommendations:

- If a Host does not receive a valid acknowledgement with the correct tag it is recommended for the Host to try up to 3 resends, using the original tag value. Options after 3 failures could be:
  - *ConnectionReset* by the Host, considering this as a Link loss.
  - Assume the command was lost. Use tag + 1 for the next command.

Comment: In this case the Host does not know if the previous command was acted on.

Comment: This also means that a Device must be able to cope with a tag incrementing by more than 1, indicating that the previous command was lost.

- A write command received by a Device with the same tag as the previous command shall not be re-executed.

Comment: This means that the acknowledgment packet was lost, but the Device knows that it previously actioned the write.

- A read command received by a Device with the same tag as the previous command shall be re-executed, but it is Device-dependent if the previous data value is re-sent, or an updated value is sent.

Comment: Similarly, this means that the acknowledgment packet was lost.

9.6.2 Control Command Message Format

9.6.2.1 Without Tag

The payload of CoaXPress version 1.x compatible control commands without tag is specified in the following table:

Table 23 — Control command payload format – without tag

Word	Content	Description
	4x K27.7	*Start of packet indication (see section 9.4).*
	4x 0x02	*Control command indication – without tag.*
0	Cmd	Control operation code (in character P0): 0x00 Memory read 0x01 Memory write Others values are reserved for future use.
	Size	24 bit value, stored in consisting of 3 characters P1,P2,P3, that specifies the number of data bytes *B* to be read or written. *B* does not include any dummy bytes for padding. This value shall be ≥ 1 (at least 1 byte to read/write).
1	Addr	32 bit address of the memory location to read from / write to. When more than one byte must be read / written the successive addresses are in increasing order starting from this address.
2 to *N*+1	Write Data	Data to be written. This field is omitted for a read operation (Cmd=0x00). The size of this data section shall be an integer number of words *N*, where *N* = Ceil(*B* / 4) * When *B* is not a multiple of 4, a number of dummy bytes (*4 N – B)* shall be padded after the last data byte. Padded dummy bytes shall have the value 0.
N+2	CRC	32 bit CRC calculated over words 0 to *N*+1.
	4x K29.7	*End of packet indication.*

* Ceil(x) returns the smallest integer ≥ x

Packet size: *N* + 6 words ((*N* * 4) + 24 bytes).

9.6.2.2 With Tag

The payload of CoaXPress version 2.x compatible control commands with tag is specified in the following table:

Table 24 — Control command message format – with tag

Word	Content	Description
	4x K27.7	*Start of packet indication (see section 9.4).*
	4x 0x05	*Control command indication – with tag.*
0	4x Tag	8 bit tag. Incremented for each new Command as defined in section 9.6.1.2.
1	Cmd	Control operation code (in character P0): 0x00 Memory read 0x01 Memory write Others values are reserved for future use.
	Size	24 bit value, stored in consisting of 3 characters P1,P2,P3, that specifies the number of data bytes *B* to be read or written. *B* does not include any dummy bytes for padding. This value shall be ≥ 1 (at least 1 byte to read/write).
2	Addr	32 bit address of the memory location to read from / write to. When more than one byte must be read / written the successive addresses are in increasing order starting from this address.
3 to *N*+2	Write Data	Data to be written. This field is omitted for a read operation (Cmd=0x00). The size of this data section shall be an integer number of words *N*, where *N* = Ceil(*B* / 4) * When *B* is not a multiple of 4, a number of dummy bytes (*4 N – B)* shall be padded after the last data byte. Padded dummy bytes shall have the value 0.
N+3	CRC	32 bit CRC calculated over words 0 to *N*+2.
	4x K29.7	*End of packet indication.*

* Ceil(x) returns the smallest integer ≥ x

Packet size: *N* + 7 words ((*N* * 4) + 28 bytes).

9.6.3 Acknowledgment Message Format

The payload of CoaXPress version 1.x compatible acknowledgment messages without tag is specified in Table 25 and that of CoaXPress version 2.x compatible acknowledgment messages with tag is specified in Table 26:

9.6.3.1 Without Tag

Table 25 — Acknowledgment message format – without tag

Word	Content	Description
	4x K27.7	Start of packet indication (see section 9.4).
	4x 0x03	Control acknowledge indication – without tag.
0	4x Code	Acknowledgment code (repeated 4 times): *Success:* 0x00 Final, command executed OK, reply data is appended (i.e. acknowledgment of read command). 0x01 Final, command executed OK, No reply data is appended (i.e. acknowledgment of write command). 0x04 Wait. The time for the Host to wait shall be sent as a 4 byte integer in the reply data field, using the same packet structure as for a 0x00 acknowledgement. The value shall be in milliseconds, with a range of 100ms to 10s. *Logical Errors (final acknowledgments):* 0x40 Invalid address. 0x41 Invalid data for the address. 0x42 Invalid control operation code. 0x43 Write attempted to a read-only address. 0x44 Read attempted from a write-only address. 0x45 Size field too large – command message (write) or acknowledgment message (read) would exceed packet size limit. 0x46 Incorrect size received, message size is inconsistent with message size indication. 0x47 Malformed packet. *Physical Errors (final acknowledgments):* 0x80 Failed CRC test in last received command. Other values are reserved for future use. For all acknowledgment codes other than 0x00 or 0x04, the Size, Data and CRC fields shall be omitted and the end of packet indication shall be transmitted after the acknowledgment code.
1	Size	Number of appended reply data bytes B, which shall be the same as the size field in the corresponding Control Command packet. B does not include any dummy bytes for padding.
2 to *N*+1	Data	Reply data. The size of this data section shall be an integer number of words N, where N = Ceil(B / 4) * When B is not a multiple of 4, a number of dummy bytes (4 N – B) shall be padded after the last data byte. Padded dummy bytes shall have the value 0.
N+2	CRC	32 bit CRC calculated over data words 0 to N+1.
	4x K29.7	End of packet indication.

Packet size: *N* + 6 words ((*N* * 4) + 24 bytes).

9.6.3.2 With Tag

Table 26 — Acknowledgment message format – with tag

Word	Content	Description
	4x K27.7	Start of packet indication (see section 9.4).
	4x 0x06	Control acknowledge indication – with tag.
0	4x Tag	8 bit tag, matching the command being acknowledged.
1	4x Code	Acknowledgment code (repeated 4 times): *Success:* 0x00 Final, command executed OK, reply data is appended (i.e. acknowledgment of read command). 0x01 Final, command executed OK, No reply data is appended (i.e. acknowledgment of write command). 0x04 Wait. The time for the Host to wait shall be sent as a 4 byte integer in the reply data field, using the same packet structure as for a 0x00 acknowledgement. The value shall be in milliseconds, with a range of 100ms to 10s. *Logical Errors (final acknowledgments):* 0x40 Invalid address. 0x41 Invalid data for the address. 0x42 Invalid control operation code. 0x43 Write attempted to a read-only address. 0x44 Read attempted from a write-only address. 0x45 Size field too large – command message (write) or acknowledgment message (read) would exceed packet size limit. 0x46 Incorrect size received, message size is inconsistent with message size indication. 0x47 Malformed packet. *Physical Errors (final acknowledgments):* 0x80 Failed CRC test in last received command. Other values are reserved for future use. For all acknowledgment codes other than 0x00 or 0x04, the Size, Data and CRC fields shall be omitted and the end of packet indication shall be transmitted after the acknowledgment code.
2	Size	Number of appended reply data bytes B, which shall be the same as the size field in the corresponding Control Command packet. B does not include any dummy bytes for padding.
3 to *N*+2	Data	Reply data. The size of this data section shall be an integer number of words N, where N = Ceil(B / 4) * When B is not a multiple of 4, a number of dummy bytes (4 N – B) shall be padded after the last data byte. Padded dummy bytes shall have the value 0.
N+3	CRC	32 bit CRC calculated over data words 0 to N+2.
	4x K29.7	End of packet indication.

Packet size: *N* + 7 words ((*N* * 4) + 28 bytes).

9.6.4 Control Transaction Block Size

The total control packet size (i.e. from the first K27.7 to the last K29.7) shall not be more than the size set in bytes in *ControlPacketSizeMax* during Device discovery (see section 12.1.6).

9.7 Heartbeat

To allow the Host to compute the tracking parameters linking the Host time clock and the Device time clock (see section 4.11), the Device shall send a heartbeat message at regular interval to the Host. Heartbeat packets shall only be sent over the master connection. The Device shall not send heartbeats if bootstrap register *VersionUsed* indicates 1.0 or 1.1 or during Connection Test (see section 9.9.4).

The heartbeat message contains a snapshot of the Device time taken at the time the start of the message is sent by the Device to the Host. The Device time should be captured as late as possible before the packet is sent, and in a way that the time between Device time capture and the packet transmission does not vary significantly between packets.

Comment: For the best accuracy the time needs to be captured in the SERDES clock domain, without the usage of elastic buffers in the SERDES. However the target accuracy of 10ns for line scan may be achievable even if these options are not practical.

The recommended nominal heartbeat message interval is 10 milliseconds, and the heartbeat message interval shall not be greater than 100 milliseconds.

Comment: It is not critical that the heartbeat message is sent exactly every 10 milliseconds. The message must just be sent often enough to help the host to keep good clock tracking accuracy.

Comment: The heartbeat message is a low priority message and can be sent when the downstream channel is free, for instance between two image data packets.

9.7.1 Heartbeat Payload Format

The payload of Heartbeat messages is specified in the following table:

Table 27 — Heartbeat payload format

Word	Content	Description
	4x K27.7	*Start of packet indication (see section 9.4).*
	4x 0x09	*Heartbeat packet indication.*
0	4x ID(31:24)	32 bit value MasterHostConnectionID (see section 12.3.35).
1	4x ID(23:16)	
2	4x ID(15:8)	
3	4x ID(7:0)	
4	4x Time(63:56)	64 bit value representing the value of the Device time when the first bit of the first character of the Heartbeat Data Packet is sent by the Device to the Device Transceiver. The Device time is expressed in nanoseconds, and shall increment monotonically but not necessarily in steps of 1ns. It shall wrap around to 0x0000 0000 0000 0000 when it reaches 0xFFFF FFFF FFFF FFFF.
5	4x Time(55:48)	
6	4x Time(47:40)	
7	4x Time(39:32)	
8	4x Time(31:24)	
9	4x Time(23:16)	
10	4x Time(15:8)	
11	4x Time(7:0)	
	4x K29.7	*End of packet indication.*

Packet size: 15 words (60 bytes).

Comment: Assuming that heartbeat messages are sent every 10 milliseconds, the bandwidth usage is 0.0006% of the available bandwidth at 6.25 Gbps.

The Host shall detect mismatches in the time field and reject corrupted device time values, therefore avoiding wrong time tracking updates.

Comment: Heartbeat messages can occasionally be rejected or ignored by the Host without causing significant trouble to the clock tracking. The tracking parameters will keep valid values and will be updated by the next heartbeat packet. This will only slightly affect the tracking accuracy.

It is recommended to keep the greatest number of least significant bits naturally available in the Device implementation at reasonable cost. This will keep the best accuracy in the whole chain.

Comment: As an example, with an FPGA clock of 125 MHz, the last three least significant bits can be set to zero, so giving a resolution of eight ns.

The Host shall ensure the coherency of events.

Comment: A 64-bit value wraps around after 584 years, so a wrap is unlikely to occur.

The system should minimize the total jitter of the heartbeat message. This can be done by:

- It is recommended that the counter used for the Device time should be in the same clock domain as the CoaXPress core. In such a way, there is no jitter due to clock domain crossing.
- Host jitter can be reduced by averaging techniques.

- The Device implementation shall take care of the word alignment mechanism in the 10B character stream. Generally, the alignment is done on a word of four 10B characters. The counter used for the Device time shall be sampled at the right time to avoid any character jitter effect.

Comment: The Host implementation typically has an unavoidable of +/- one 10B character from the SERDES data alignment.

9.8 Event Channel

The event channel is used to provide the Device with a mechanism to asynchronously send messages and status updates to a Host. This is accomplished through specific event packets, transmitted from Device to Host, and the resulting event acknowledgment packet, transmitted from the Host to the Device.

Event packets shall only be sent by the Device. Event acknowledgments shall only be sent by the Host as a response to a received event packet.

Comment: Events sent from a Host to a Device are not supported. All Host to Device communications use the control channel.

Event packets and event acknowledgments shall only be sent over the master connection. Extension connections shall not carry event packets or event acknowledgments.

The Device shall not send events if bootstrap register *VersionUsed* indicates 1.0 or 1.1.

Multiple events per packet are supported. As such, the event packet acts as a container and is composed of one or more event messages. The event acknowledgment indicates reception of the entire event packet, including all event messages contained in the originating event packet.

9.8.1 Event Transactions

An event transaction is composed of transmitted event packet and an event acknowledgment.

9.8.1.1 Transaction Rules

Comment: These rules are very similar to those for Commands, but are explicitly stated for clarity.

The Device shall implement the following rules for event packets:

- The Device shall not send event packets before it has sent a Heartbeat packet.

Comment: The event packet includes a timestamp, so a heartbeat packet is needed to set the time reference in the Host.

- After completion of an event packet the Device shall not send a new event packet until it has received an acknowledgment from the Host.
- However the Device can resend the last event packet or can send a new event packet:
  - If the Device does not receive an acknowledgment for more than 200ms (transmission timeout).

Comment: The timeout is set to 200ms for consistency with other timeouts used in the standard, but a Host needs to reply quickly to event packets to support expected use cases such as checking trigger packets.

The Host shall implement the following rules for event packets:

- On receiving a valid event packet the Host shall transmit an event acknowledgment.

- The time required to transmit the event acknowledgment shall not exceed 200ms, which is the transmission timeout.

Comment: The timeout is set to 200ms for consistency with other timeouts used in the standard, but a Device needs to reply quickly to event acknowledgments to support expected use cases such as checking trigger packets.

Device and Host shall implement the following rules for tags:

- The tag shall be a free running number incremented for each new event packet.
- A Host shall send the tag in the acknowledge packet with the same tag it received in the event packet.
- The Device shall increment the tag by one for a new event. It shall wrap to 0x00 from 0xFF.
- The packet tag shall only be reset back to zero as part of a *ConnectionReset* (see section 12.3.28).

Error Recovery rules and recommendations:

- If a Device does not receive a valid acknowledgement with the correct tag it is recommended for the Device to try up to 3 resends, using the original tag value. Options after 3 failures could be:
  - To drop the link (i.e. Link loss) forcing a *ConnectionReset* by the Host.
  - Assume the event was lost. Use tag + 1 for the next event.

Comment: In this case the Device does not know if the previous event was received.

Comment: This also means that a Host must be able to cope with a tag incrementing by more than 1, indicating that the previous event was lost.

9.8.2 Event Payload Format

The payload of Event packets is specified in the following table. These packets are used to send event messages across the link. One or more event messages can be transmitted in the event messages section of the packet. The payload format is specified in the following table:

Table 28 — Event payload format

Word	Content	Description
	4x K27.7	*Start of packet indication (see section 9.4).*
	4x 0x07	*Event packet indication.*
0	4x ID(31:24)	32 bit value MasterHostConnectionID (see section 12.3.35).
1	4x ID(23:16)	
2	4x ID(15:8)	
3	4x ID(7:0)	
4	4x Packet Tag	8 bit tag. Incremented for each new Event packet as defined in section 9.8.1.1.
5	4x DsizeP(15:8)	16 bit value representing the number of event data words *N*. Maximum size = 256 words.
6	4x DsizeP(7:0)	
7 to *N*+6	Event message data	*N* words with one or more event messages, as defined in Table 29.
N+7	CRC	32-bit CRC calculated over the event data 7 to *N*+6.
	4x K29.7	*End of packet indication.*

Packet size: *N* + 11 words ((*N* * 4) + 44 bytes).

Comment: The event message data is limited to 256 words to keep the event packet compact and fast. If a specific event needs more data, then this can be read by the Host using command packets in response to receiving the event. DsizeP is a 16 bit value so a larger value could be used in future.

Table 29 — Event message format

Word	Content	Description
0	Event_size	16 bit value in bits 31:16 (characters P0, P1) representing the number of data bytes *B* for this event message, i.e. over words 0 to M+2. *B* does not include any dummy bytes for padding.
	Namespace	2 bit field in bits 15:14 specifying event namespace (see below): 0 = GenICam Event ID. 1 = CoaXPress Specific Event ID (reserved for future use). 2 = Device Specific Event ID. 3 = Reserved for future use.
	Reserved	2 bit field in bits 13:12 set to 0.
	EventID	12 bit field in bits 11:0 specifying the event source (see below).
1	Timestamp(63:32)	64 bit value representing the value of the Device time when the Device detected the situation resulting in this event. See section 4.11.
2	Timestamp(31:0)	
3 to *M*+2	Data	The size of this optional data section shall be an integer number of words *M*, where *M* = Ceil(*B* / 4) * When *B* is not a multiple of 4, a number of dummy bytes (*4 M – B)* shall be padded after the last data byte. Padded dummy bytes shall have the value 0.

* Ceil(x) returns the smallest integer ≥ x

Comment: The Event_size is limited to 256 words because of the size restriction of the event packet. If multiple events are sent in one packet then the sum of all the messages in that packet is limited to 256 words.

The Namespace shows where the event is defined. The value 0 shall indicate a GenICam event which is defined in the SFNC. The value 1 shall indicate an event that is defined in this standard (reserved for future use). The value 2 shall indicate an event that is defined by the Device.

The EventID is the unique identifier for the event. For GenICam events shall be defined in the Device's XML file. For CoaXPress events it is defined in this standard (reserved for future use). For Device specific events it shall be defined in the Device's documentation.

9.8.3 Event Acknowledgment Format

The payload of an event acknowledgment is specified in the following table.

Table 30 — Event acknowledgment format

Word	Content	Description
	4x K27.7	*Start of packet indication (see section 9.4).*
	4x 0x08	*Event acknowledgment indication.*
0	4x Packet Tag	8 bit tag, matching the event being acknowledged.
	4x K29.7	*End of packet indication.*

Packet size: 4 words (32 bytes).

9.9 Connection Test

The Host and Device shall provide connection test facilities to test the quality of the connection. Connection test in both directions is initiated and controlled by the Host.

9.9.1 Connection Test Methodology

Connection test makes use of test data packets between a Test Generator and a Test Receiver as depicted schematically in the following figure:

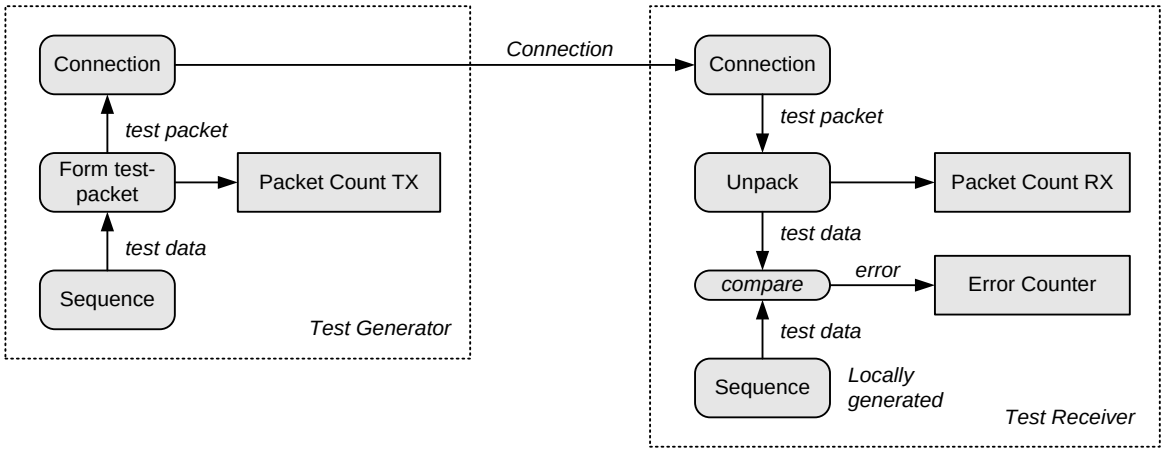


Figure 27 — Connection test methodology

The Test Generator shall transmit a Test Data Packet containing a known test pattern produced by a Sequence Generator, and increment the packet counter for each test packet transmitted. The Test Receiver shall compare the received test data packet content against its local Sequence Generator. The Test Receiver shall increment the Error Counter for each word that is different in the data packet, and increment the packet counter for each test packet received.

Comment: The test packet counters show how many test packets have been sent and received, so allowing a judgment to be made on the statistical meaning of the value in the error counter.

Comment: Both Device to Host and Host to Device connection tests can be run at the same time.

9.9.2 Connection Test Data Payload Format

The payload of a Test Data Packet content shall consist of a continuous 4096 byte sequence generated by counting 16 times from 0 to 255 (4096 bytes, 1024 words) as illustrated in the following table:

Table 31 — Connection test payload format

Word	Content				Description
	P0	P1	P2	P3	
	K27.7	*K27.7*	*K27.7*	*K27.7*	Start of packet indication (see section 9.4).
	0x04	*0x04*	*0x04*	*0x04*	Connection test indication.
0	0x00	0x01	0x02	0x03	Test data
1	0x04	0x05	0x06	0x07	
…					
63	0xFC	0xFD	0xFE	0xFF	
64	0x00	0x01	0x02	0x03	
…					
1023	0xFC	0xFD	0xFE	0xFF	
	K29.7	*K29.7*	*K29.7*	*K29.7*	End of packet indication.

Packet size: 1027 words (4108 bytes).

Comment: Using a PRBS (pseudo-random binary sequence) is of no use since 8B/10B line coding is applied to the generated data stream. For the purpose at hand a simple counter works equally well and greatly eases implementation.

The transmitter shall give priority to Control Data over Test Data for packet transmission.

Comment: This ensures that the control messages to start and stop connection test have priority over the test data.

9.9.3 Host to Device Connection Test

The Host shall implement a Test Generator and the Device a Test Receiver as shown in section 9.9.1.

The Error Counter in the Device shall be implemented as bootstrap registers *TestErrorCount[m]* (where *m* is the connection number 0 .. *n*-1, or the value *n* for the optional high speed upconnection). The Host shall have read and write access to these registers allowing it to reset them. Similarly, the Device shall implement bootstrap registers *TestPacketCountRx[m]* to show the total packet count received during the test, which the Host shall have read and write access to.

The Host shall use a spacing of at least one IDLE word between each test data packet, and shall insert IDLE words into the test sequence to meet the IDLE requirements (see section 9.2.5.1).

The Device shall process a received test data packet from the Host at all times regardless of the setting of the *TestMode* bootstrap register.

To test the Host to Device connections the Host shall execute the following operations:

- Reset the Device Error Counters and Packet Counters by writing the value 0 to *TestErrorCount[m]* and *TestPacketCountRx[m]* registers.
- Reset the Packet Counters at the Host transmitters that are involved in the test.
- Run the test for a pre-determined amount of time providing the desired coverage for each connection involved in the test.
- Read and process the Device *TestErrorCount[m]* and *TestPacketCountRx[m]* register values, and the Host Transmit Packet Counters, for each connection involved in the test.

9.9.4 Device to Host Connection Test

The Device shall implement a Test Generator and the Host a Test Receiver as shown in section 9.9.1.

The Test Generator shall be controlled via the downconnection test control bootstrap register *TestMode*. The Host shall have read and write access to this register. When the register is set to 1 for Test Mode the Device shall transmit test data packets at a regular interval.

The spacing between test data packets shall be at least 16 word intervals allowing room for control data packets. The Device shall insert IDLE words into the test sequence to meet the IDLE requirements (see section 9.2.5.1), but shall not transmit data other than connection test packets or control packets.

Comment: Note that Heartbeat packets should not be sent during connection test – see section 9.7.

To test the Device to Host connections the Host shall execute the following operations:

- Stop the Device from transmitting streaming data if applicable (stop image acquisition).
- Reset the Device Packet Counters by writing the value 0 to *TestPacketCountTx[m]* registers (where *m* is the connection number 0 .. *n*-1, or the value *n* for the optional high speed upconnection).
- Reset the Error Counters and Packet Counters at the Host receivers that are involved in the test.
- Put the Device in Test Mode by writing 1 to the *TestMode* bootstrap register.
- Run the test for a pre-determined amount of time providing the desired coverage for each connection involved in the test.
- Read and process the Host Error Counter and Receive Packet Counter values, and the Device *TestPacketCountTx[m]* registers, for each connection involved in the test.
- Restore normal Device operation by writing 0 to the *TestMode* bootstrap register.

10 Data Streams

10.1 Overview

Stream channels shall be used to transfer data from Device to Host. Data shall be formed into streams before being split into packets to send over the physical connection(s), as described in section 9.

CoaXPress defines its own data stream formats for rectangular images, but other types may be defined in future, such as for auxiliary data, compressed images and audio.

If a Device generates multiple images (e.g. multiple ROIs), each image is formed into a separate stream (see section 10.4.3).

Similarly, data from multiple taps are formed into separate streams (see section 10.4.5).

CoaXPress also supports streams of GenDC data, as defined by the GenICam GenDC standard.

10.2 Stream Format

A stream comprises a header giving the stream type and information about the following data, then the data itself.

A stream uses K-code K28.3 as a stream marker, in addition to the K-codes used for packet transfer (see section 9.2.1). This K-code is used to identify the stream header and important points in the data stream, such as start-of-line (for an image stream).

Comment: The format of a stream is designed to give a low overhead in terms of the number of additional characters needed for the stream header and stream markers, so allowing very high speed transfer of very small image sizes.

Following the K28.3 is a code indicating the type of stream data. The following types are defined:

Table 32 — Data Stream Types

Code	Usage
0x01	Rectangular image header indication
0x02	Rectangular line marker
0x03	Reserved
0x04	Reserved
0x05	GenDC data indication
0x06	GenDC final descriptor indication
0x07	GenDC preliminary descriptor indication

10.3 Stream ID

Each stream from a Device shall have a unique stream ID. The Device shall have a fixed set of streams it can produce, each with an allocated static stream ID.

Comment: The GenICam XML file and product documentation give the stream information for the Device.

Stream IDs range from 0 to 255.

It is recommended that the primary rectangular stream from a Device has stream ID = 0.

Stream ID rules for GenDC streams are defined in section 10.5.1.

10.4 CoaXPress Image Streams

Image streams defined by CoaXPress are formed from lines. An image header is used to inform the Host about the format of succeeding image lines. A line marker separates data from successive lines.

A typical image stream is illustrated in the following figure:

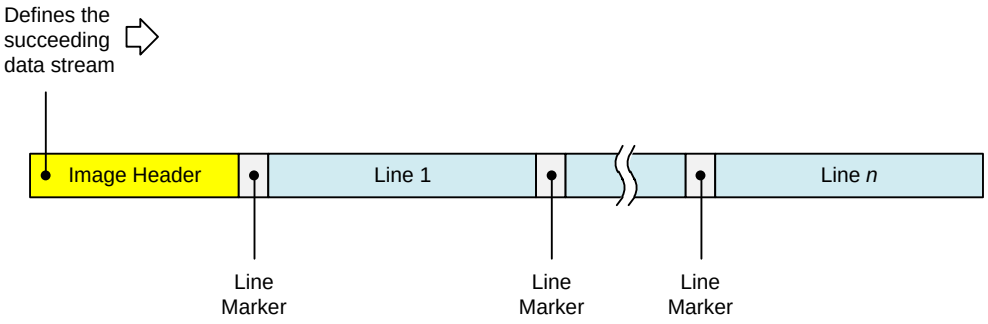


Figure 28 — Image stream structure

10.4.1 Pixel Types

This standard is based on the pixel format names and definition of the Pixel Format Naming Convention (PFNC) (section 2.2 ref 5), which is part of the GenICam standard.

The Device shall use pixel formats defined in Table 34, and its XML file shall list the formats it supports using the names given in the “PFNC Name” column (see section 13.2.2.6). Bit level coding of the pixel formats and packing over the CoaXPress link is CoaXPress specific (i.e. not as defined by the PFNC), as are some Planar formats which are not defined in the PFNC.

Comment: The CoaXPress v1.1 standard required the use of the suffix “pmsb” on all the packed format names, to comply with the GenICam PFNC and SFNC. This requirement was ignored by most vendors, and the PFNC has been revised to remove this requirement. CoaXPress v1.1.1 onwards reflect this change to the PFNC.

The Host can map the Device’s data to any pixel format in the PFNC, and the resulting image generated by the Host shall be precisely as defined by the PFNC.

Both Devices and Hosts shall support at least one of the types listed in Table 34 over the CoaXPress link.

Comment: A typical frame grabber Host would support many types.

Comment: YUV and YCbCr are clearly defined to avoid the usual uncertainty with these formats.

10.4.1.1 Pixel Format Code PixelF

The pixel format code *PixelF* is formed as shown in the following table. This code is used in the *PixelF* field in the image headers. Note that the value 0x0000 is reserved for “raw” data that does not match any defined format, such as user-specific formats.

Table 33 — PixelF coding

Bits	Usage
15..8	Data Type
7..4	Sub-type
3..0	Data Width

Comment: Many format codes are spare to allow future formats and data widths to be added.

Similarly, the format name is concatenated from the Data Type, Sub-type, and data width. e.g. BayerGR8 is Bayer data, with order GR, and 8 bits per component.

The data types and widths are defined in the following section, and the resulting *PixelF* values for most formats (only a selection of planar formats are listed for clarity) are shown in the following table, along with the PFNC name as used by GenICam. PFNC names are shown in italics where the PFNC defines the naming convention, but a PFNC pixel value has not yet been defined.

Table 34 — PixelF values

Pixel format	PFNC Name	PixelF Code
Raw	-	0x0000
Mono8	Mono8	0x0101
Mono10	Mono10	0x0102
Mono12	Mono12	0x0103
Mono14	Mono14	0x0104
Mono16	Mono16	0x0105
Planar1_8	-	0x0211
Planar1_10	-	0x0212
Planar1_12	-	0x0213
Planar1_14	-	0x0214
Planar1_16	-	0x0215
Planar2_8	-	0x0221
Planar2_10	-	0x0222
Planar2_12	-	0x0223
Planar2_14	-	0x0224
Planar2_16	-	0x0225
…	-	…
Planar15_8	-	0x02F1
Planar15_10	-	0x02F2
Planar15_12	-	0x02F3
Planar15_14	-	0x02F4
Planar15_16	-	0x02F5

continued on next page …

PixelF values continued…

Pixel format	PFNC Name	PixelF Code
BayerGR8	BayerGR8	0x0311
BayerGR10	BayerGR10	0x0312
BayerGR12	BayerGR12	0x0313
BayerGR14	BayerGR14	0x0314
BayerGR16	BayerGR16	0x0315
BayerRG8	BayerRG8	0x0321
BayerRG10	BayerRG10	0x0322
BayerRG12	BayerRG12	0x0323
BayerRG14	BayerRG14	0x0324
BayerRG16	BayerRG16	0x0325
BayerGB8	BayerGB8	0x0331
BayerGB10	BayerGB10	0x0332
BayerGB12	BayerGB12	0x0333
BayerGB14	BayerGB14	0x0334
BayerGB16	BayerGB16	0x0335
BayerBG8	BayerBG8	0x0341
BayerBG10	BayerBG10	0x0342
BayerBG12	BayerBG12	0x0343
BayerBG14	BayerBG14	0x0344
BayerBG16	BayerBG16	0x0345
RGB8	RGB8	0x0401
RGB10	RGB10	0x0402
RGB12	RGB12	0x0403
RGB14	RGB14	0x0404
RGB16	RGB16	0x0405
RGBA8	RGBa8	0x0501
RGBA10	RGBa10	0x0502
RGBA12	RGBa12	0x0503
RGBA14	RGBa14	0x0504
RGBA16	RGBa16	0x0505
YCbCr411_8	YCbCr411_8	0x0611
YCbCr411_10	*YCbCr411_10*	0x0612
YCbCr411_12	*YCbCr411_12*	0x0613
YCbCr411_14	*YCbCr411_14*	0x0614
YCbCr411_16	*YCbCr411_16*	0x0615
YCbCr422_8	YCbCr422_8 or YUV422_8[4]	0x0621

continued on next page …

[4] These two formats have the same PFNC definition. For new products YCbCr422_8 is recommended. YUV422_8 is kept as an option to maintain compatibility with products designed to earlier versions of the standard.

PixelF values continued…

Pixel format	PFNC Name	PixelF Code
YCbCr422_10	YCbCr422_10	0x0622
YCbCr422_12	YCbCr422_12	0x0623
YCbCr422_14	*YCbCr422_14*	0x0624
YCbCr422_16	*YCbCr422_16*	0x0625
YCbCr444_8	YCbCr8_CbYCr	0x0631
YCbCr444_10	YCbCr10_CbYCr	0x0632
YCbCr444_12	YCbCr12_CbYCr	0x0633
YCbCr444_14	*YCbCr14_CbYCr*	0x0634
YCbCr444_16	*YCbCr16_CbYCr*	0x0635
YCbCr601_411_8	YCbCr601_411_8	0x0711
YCbCr601_411_10	*YCbCr601_411_10*	0x0712
YCbCr601_411_12	*YCbCr601_411_12*	0x0713
YCbCr601_411_14	*YCbCr601_411_14*	0x0714
YCbCr601_411_16	*YCbCr601_411_16*	0x0715
YCbCr601_422_8	YCbCr601_422_8	0x0721
YCbCr601_422_10	YCbCr601_422_10	0x0722
YCbCr601_422_12	YCbCr601_422_12	0x0723
YCbCr601_422_14	*YCbCr601_422_14*	0x0724
YCbCr601_422_16	*YCbCr601_422_16*	0x0725
YCbCr601_444_8	YCbCr601_8_CbYCr	0x0731
YCbCr601_444_10	YCbCr601_10_CbYCr	0x0732
YCbCr601_444_12	YCbCr601_12_CbYCr	0x0733
YCbCr601_444_14	*YCbCr601_14_CbYCr*	0x0734
YCbCr601_444_16	*YCbCr601_16_CbYCr*	0x0735
YCbCr709_411_8	YCbCr709_411_8	0x0811
YCbCr709_411_10	YCbCr709_411_10	0x0812
YCbCr709_411_12	YCbCr709_411_12	0x0813
YCbCr709_411_14	*YCbCr709_411_14*	0x0814
YCbCr709_411_16	*YCbCr709_411_16*	0x0815
YCbCr709_422_8	YCbCr709_422_8	0x0821
YCbCr709_422_10	YCbCr709_422_10	0x0822
YCbCr709_422_12	YCbCr709_422_12	0x0823
YCbCr709_422_14	*YCbCr709_422_14*	0x0824
YCbCr709_422_16	*YCbCr709_422_16*	0x0825
YCbCr709_444_8	YCbCr709_8_CbYCr	0x0831
YCbCr709_444_10	YCbCr709_10_CbYCr	0x0832
YCbCr709_444_12	YCbCr709_12_CbYCr	0x0833
YCbCr709_444_14	*YCbCr709_14_CbYCr*	0x0834
YCbCr709_444_16	*YCbCr709_16_CbYCr*	0x0835

10.4.1.2 Data Width / Packing

This is the width in bits of each pixel, or for color formats the width in bits of each color component of the pixel. The width also defines the packing mode for the data. This is defined in the following table:

Table 35 — Data width definition

Data Width	Bit Definition
8 bit	0001
10 bit	0010
12 bit	0011
14 bit	0100
16 bit	0101

10.4.1.3 Mono Format

This is used for luminance data. This has no sub-types. This is defined in the following table:

Table 36 — Mono definition

Data Type	Bit Definition	Sub-type	Bit Definition
Mono	00000001	-	0000

10.4.1.4 Planar Format

This is used for planar data, such as individual red, green or blue planes, additional alpha (overlay) planes, or the separate planes in YUV420. The use of each plane by the Device is implementation dependent and beyond the scope of this standard, but planar RGB and YUV420 images shall follow the standard usage given. This is defined in the following table:

Table 37 — Planar definition

Data Type	Bit Definition	Sub-type	Bit Definition
Planar	00000010	Plane 1 Standard usage: R, Y	0001
		Plane 2 Standard usage: G, U, Cb	0010
		Plane 3 Standard usage: B, V, Cr	0011
		Plane 4	0100
		…	…
		Plane 15	1111

10.4.1.5 Bayer Format

This is used for Bayer data. This is defined in the following table:

Table 38 — Bayer definition

Data Type	Bit Definition	Sub-type	Bit Definition
Bayer	00000011	GR 1st line transmission order G, R 2nd line transmission order B, G	0001
		RG 1st line transmission order R, G 2nd line transmission order G, B	0010
		GB 1st line transmission order G, B 2nd line transmission order R, G	0011
		BG 1st line transmission order B, G 2nd line transmission order G, R	0100

10.4.1.6 RGB Format

This is used for RGB data, transmitted in the order red, green, blue. This has no sub-types. This is defined in the following table:

Comment: No other component orders (e.g. BGR) are supported by CoaXPress – they can be generated in the Host if required. A commonly used format like RGB24 is simply this RGB format with 8 bit data per color.

Table 39 — RGB definition

Data Type	Bit Definition	Sub-type	Bit Definition
RGB	00000100	-	0000

10.4.1.7 RGBA Format

This is used for RGBA data, where “A” is the alpha (or overlay) plane, transmitted in the order red, green, blue, alpha. This has no sub-types. This is defined in the following table:

Comment: No other component orders (e.g. ABGR) are supported by CoaXPress.

Table 40 — RGBA definition

Data Type	Bit Definition	Sub-type	Bit Definition
RGBA	00000101	-	0000

10.4.1.8 Generic YCbCr/YUV Format

This is used for generic YCbCr or YUV data, which for purposes of this standard has the full range of $0..2^{n-1}$, rather than being limited (e.g. 0..255 rather than 16..235), and shall be as defined by the formulae in the comment below. This is defined in the following table:

Table 41 — YCbCr definition

Data Type	Bit Definition	Sub-type	Bit Definition
YCbCr	00000110	411 Transmission order Y, Y, U, Y, Y, V	0001
		422 Transmission order Y, U, Y, V	0010
		444 Transmission order Y, U, V	0011

Comment: This data type was previously named YUV in this standard, but has been renamed to YCbCr for consistency with the PFNC standard. The definition is unchanged.

Comment: Strictly there is not a digital YCbCr/YUV standard, but the following formulae are often used to map RGB data with a range 0..255 to YCbCr/YUV data, and is used in JFIF. In the resulting 8 bit data, Y has the range 0..255; Cb/U and Cr/V are signed with a range 0..255 and 128 representing zero.

$$
\begin{aligned}
Y &= 0.299\,R + 0.587\,G + 0.114\,B \\
Cb\ (U) &= -0.169\,R - 0.331\,G + 0.499\,B + 128 \\
Cr\ (V) &= 0.499\,R - 0.418\,G - 0.0813\,B + 128
\end{aligned}
$$

10.4.1.9 YCbCr601 Format

This is used for YCbCr data, as specified by ITU-R BT.601 (section 2.2 ref 2) for 8 and 10 bit data. This is defined in the following table:

Table 42 — YCbCr601 definition

Data Type	Bit Definition	Sub-type	Bit Definition
YCbCr601	00000111	411 Transmission order Y, Y, Cb, Y, Y, Cr	0001
		422 Transmission order Y, Cb, Y, Cr	0010
		444 Transmission order Y, Cb, Cr	0011

Comment: This is the standard format for standard definition broadcast video. For 8 bit data, Y has the range 16..235; Cb and Cr are signed with a range 16..240 and 128 representing zero. The ITU-R BT.601 formulae can be simplified to the following for gamma corrected RGB data with a range 16..240:

$$
\begin{aligned}
Y &= 0.299\,R + 0.587\,G + 0.114\,B \\
Cb &= -0.173\,R - 0.339\,G + 0.511\,B + 128 \\
Cr &= 0.511\,R - 0.428\,G - 0.0832\,B + 128
\end{aligned}
$$

10.4.1.10 YCbCr709 Format

This is used for YCbCr data, as specified by ITU-R BT.709 (section 2.2 ref 3). This is defined in the following table:

Table 43 — YCbCr709 definition

Data Type	Bit Definition	Sub-type	Bit Definition
YCbCr709	00001000	411 Transmission order Y, Y, Cb, Y, Y, Cr	0001
		422 Transmission order Y, Cb, Y, Cr	0010
		444 Transmission order Y, Cb, Cr	0011

Comment: This is the standard format for high definition broadcast video. For 8 bit data, Y has the range 16..235; Cb and Cr are signed with a range 16..240 and 128 representing zero. The ITU-R BT.709 formulae can be simplified to the following for gamma corrected RGB data with a range 16..240:

$$
\begin{aligned}
Y &= 0.213\,R + 0.715\,G + 0.0722\,B \\
Cb &= -0.117\,R - 0.394\,G + 0.511\,B + 128 \\
Cr &= 0.511\,R - 0.465\,G - 0.0469\,B + 128
\end{aligned}
$$

10.4.2 Pixel Packing

The link protocol enforces maximum density packing in order to maximize throughput for the available bandwidth. Packing schemes are defined in following tables.

Pixels shall be packed within a line. The first pixel of a new line shall be stored into P0.

For some packing schemes the packed data size may not be an integer number of words across a line. In such a case the number of data words shall be rounded up to the nearest integer value, and the unused bits of the last data word shall be set 0. Note that both the number of words per line and pixels per line are provided by the preceding header or line marker.

The following figures show packing definitions for 8, 10, 12, 14 and 16 bit pixels (in the case of mono or planar formats) or components (in the case of multi-component formats such as RGB):

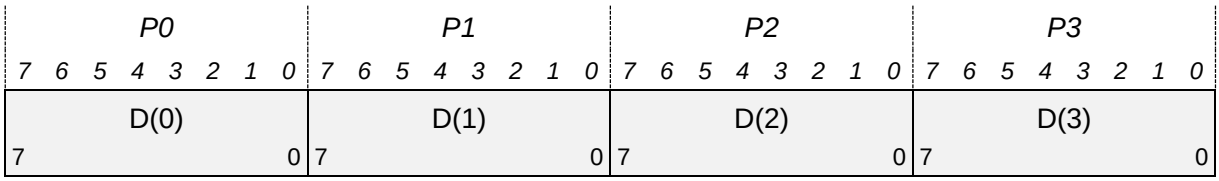


Figure 29 — Packing of 8 bit pixels or components

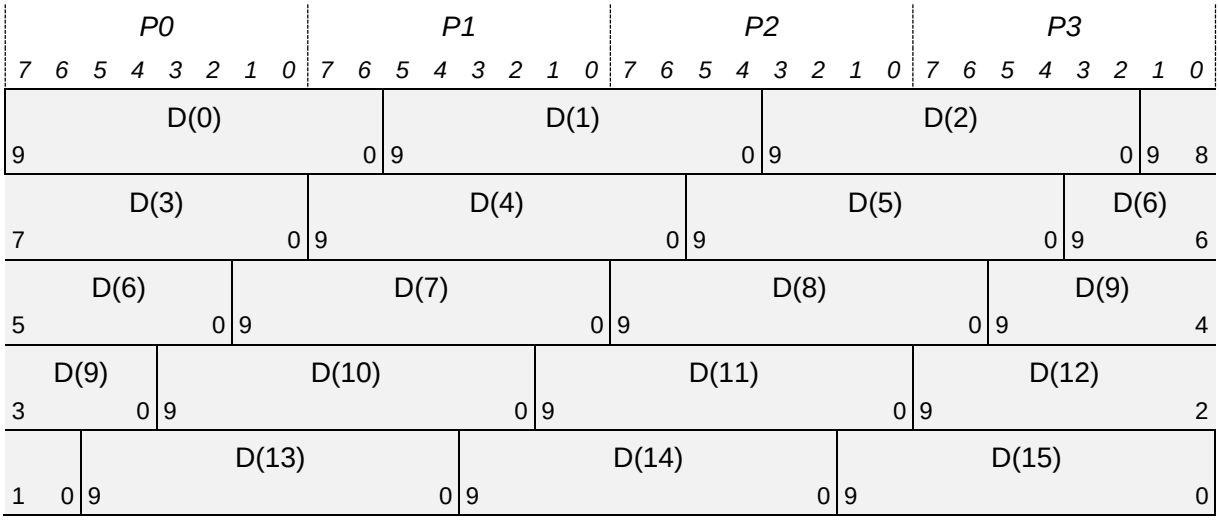


Figure 30 — Packing of 10 bit pixels or components

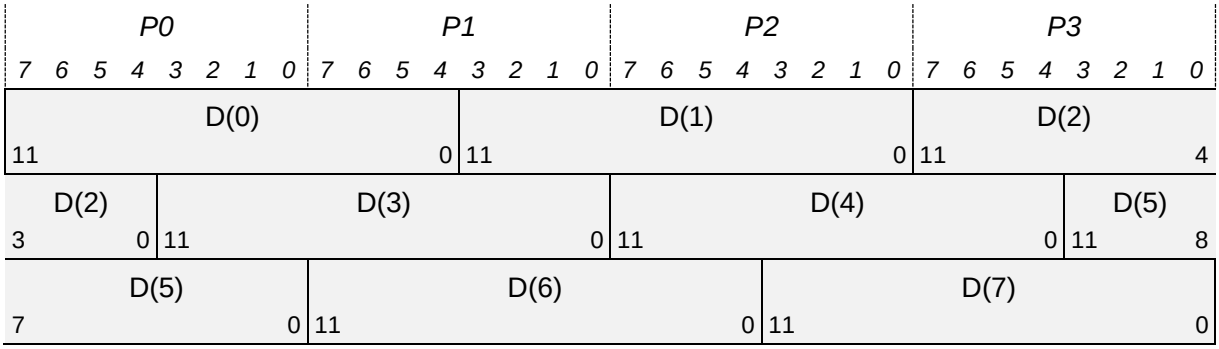


Figure 31 — Packing of 12 bit pixels or components

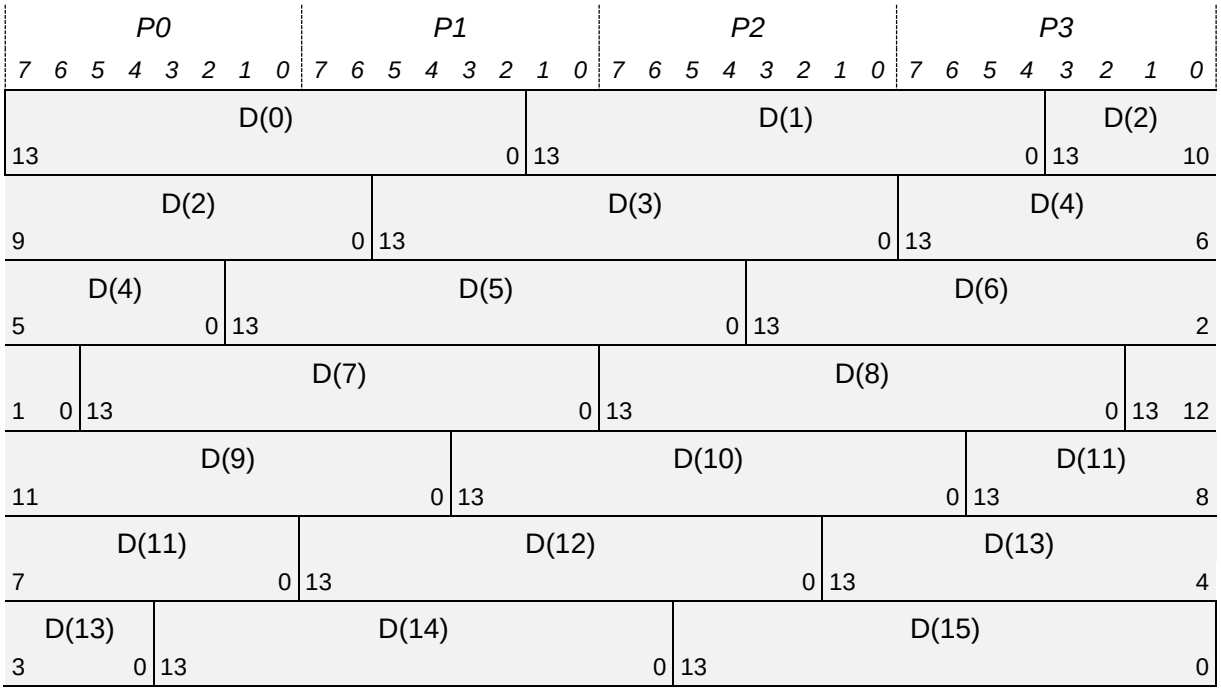


Figure 32 — Packing of 14 bit pixels or components

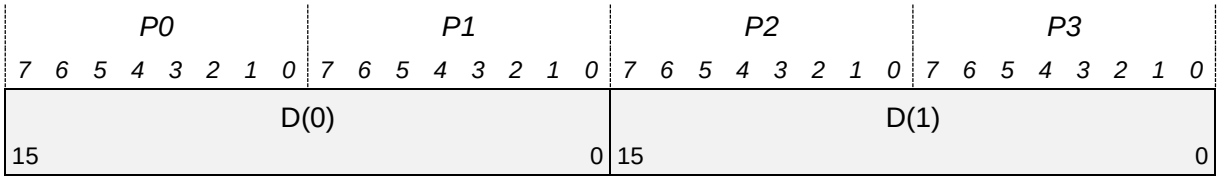


Figure 33 — Packing of 16 bit pixels or components

Comment: These packing modes define the data transmitted over CoaXPress. A frame grabber in the Host may process this data and provide different formats to the Host memory and/or display.

Pixel sizes that are in-between the defined packing sizes shall be MSB aligned into the next larger defined size. The unused least significant bits of the resulting pixel can either be filled with zeros or a dither pattern (implementation dependent). An example of 15 bit data being fitted into 16 bits prior to packing is shown in the following figure:

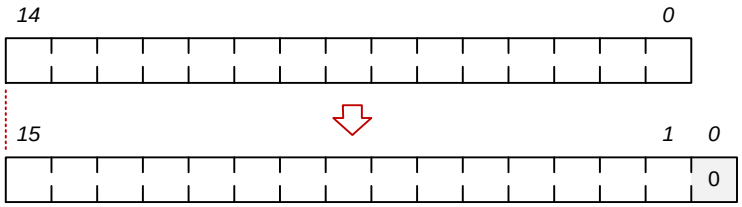


Figure 34 — Packing of 15 into 16 bits

Comment: Again this is for the data transmitted over CoaXPress, and simplifies hardware handling of varying width pixel data.

10.4.3 Multiple Images

A Device that generates more than one image shall form each image into a separate stream.

Comment: This allows for Devices that output several small regions of interest (ROI), and Devices that have more than one sensor (e.g. visible and IR).

10.4.4 Image Scan Format

Horizontal image scanning shall be fixed: Left to right.

Vertical image scanning can be intermixed using the concept of taps.

Comment: Devices that use multi-tap image sensors having opposite horizontal scan directions can relatively easily perform pixel rearrangement to obtain a horizontal single scan direction output. Using intermixed horizontal scan directions leads to complex definitions of how an image stream must be decoded at the Host, especially when combined with dynamic variable ROI capabilities. This is avoided by demanding horizontal rearrangement to be performed by the Device. Vertical multi tap scanning rearrangement on the other hand is better done at the Host giving an overall lower latency delay. Furthermore it prevents the need for frame size buffers at the Device to do the vertical rearrangement.

10.4.5 Tap Concept

Taps apply to vertical scanning only (see comment above). The number of taps shall be read or set by the Host via a Device register. The tap geometry naming follows the GenICam SFNC Tap Geometry naming, as shown in the following table:

Table 44 — Vertical tap formats

Name	Valid Values	Description
ZoneY	1, 2	The number of zones across the vertical direction.
TapY	Null, 2	The number of consecutive lines in the vertical direction that output pixels simultaneously from a zone. 'Null' means the default of a single line.
ExtractorY	Null, E	The location of the pixel extractors across vertical direction. 'Null' means the default of pixel extractors all at the top line. 'E' means pixel extractors are at both top and bottom lines.

The tap format is defined: 1X-<*ZoneY*>Y<*TapY*><*ExtractorY*>

The following tap scan geometries are supported:

- 1X-1Y (default single tap format, also used for line scan)
- 1X-1Y2
- 1X-2YE

Both Devices and Hosts shall support at least one of these tap formats.

Comment: A typical frame grabber Host would support all tap formats.

Scanning examples for both supported multi-tap geometries are illustrated in the following figure:

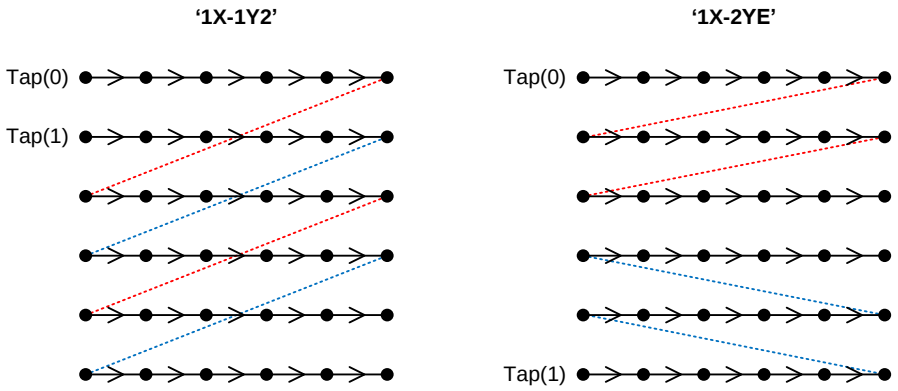


Figure 35 — Scanning example

Each tap shall form a separate stream.

The tap format code is formed as shown in the following table. This code is used in the TapG field in the image headers.

Table 45 — TapG coding

Bits	Usage	Bit Definition
15..12	ThisTap	The tap number this stream is from. “N-1” coding, i.e. 0000: Tap 1. 0001: Tap 2. Other values are reserved.
11..10	Reserved	00
9..6	ZoneY	“N-1” coding, i.e. 0000: 1. 0001: 2. Other values are reserved.
5..2	TapY	“N-1” coding, i.e. 0000: Null. 0001: 2. Other values are reserved.
1..0	ExtractorY	00: Null. 01: E. Other values are reserved.

The resulting TapG values for the defined formats are shown in the following table:

Table 46 — TapG values

Tap format	TapG Code
1X-1Y	0x0000
1X-1Y2, tap 1	0x0004
1X-1Y2, tap 2	0x1004
1X-2YE, tap 1	0x0041
1X-2YE, tap 2	0x1041

10.4.6 Rectangular Image Stream

This section applies to standard uncompressed rectangular area scan or line scan images.

10.4.6.1 Transmission Rules

A rectangular image is defined as one where:

- The size of the image (pixels per line, lines per frame) is known by the Device before it sends the image, so that it can generate a header.
- Each line in the image is the same length.
- Each line in the image has the same horizontal offset with respect to the left hand pixel of the full Device image.
- Each pixel in the image is the same type (e.g. color, mono etc).
- The image can be progressive scan or interlaced.
- The image can be Area Scan or Line Scan.

Comment: Successive images in the stream can be different sizes, because they have their own image header.

The following rules shall be used to form a rectangular image stream from an area scan Device:

- A rectangular image header shall be sent before the first line of each area scan image.

  Comment: The header therefore acts as a frame marker. The header contains critical information needed by the Host to decode succeeding image line data. The contents of the header are therefore redundantly encoded to guarantee proper decoding in case of a single bit error during transmission.

- A rectangular image line marker shall be sent before each line.
- Image data shall not be packed across line boundaries. Therefore the first pixel of a new line shall be stored into P0. See section 10.4.2.

The following rules shall be used to form a rectangular image stream from a line scan Device:

- A rectangular image header shall be sent before the first line from a line scan Device. Subsequent lines can be preceded by a rectangular image line marker or a rectangular image header.

  Comment: The line marker reduces overhead, particularly with short lines.

- Additionally, a line scan Device shall send a rectangular image header at least every 200ms, or once per line if the time between lines is more than 200ms.

  Comment: This allows recovery in the event that serious multiple bit errors corrupt the initial header.

- Image data shall not be packed across line boundaries. Therefore the first pixel of a new line shall be stored into P0. See section 10.4.2.

10.4.6.2 Rectangular Image Header

The header contains critical information needed by the Host to decode succeeding image line data. The contents of the header are therefore redundantly encoded to allow proper decoding in case of a single bit error during transmission.

The image header shall be as shown in the following table:

Table 47 — Rectangular image header

Word	Content	Description
1	4x K28.3	Stream Marker.
2	4x 0x01	Rectangular Image Header indication.
3	*4x StreamID*	Unique stream ID.
4	*4x SourceTag(15:8)*	16 bit source image index. Incremented for each transferred image header, wraparound to 0 at 0xFFFF. The same number shall be used by each stream containing data relating to the same image (area scan) or group of lines (line scan).
5	*4x SourceTag (7:0)*	
6	*4x Xsize(23:16)*	24 bit value representing the image width in pixels.
7	*4x Xsize(15:8)*	
8	*4x Xsize(7:0)*	
9	*4x Xoffs(23:16)*	24 bit value representing the horizontal offset in pixels of the image with respect to the left hand pixel of the full Device image. *Comment: Xoffs and Yoffs tell the Host where this image is with respect to the full Device image size.*
10	*4x Xoffs(15:8)*	
11	*4x Xoffs(7:0)*	
12	*4x Ysize(23:16)*	24 bit value representing the image height in pixels. This value shall be set to 0 for line scan images.
13	*4x Ysize(15:8)*	
14	*4x Ysize(7:0)*	
15	*4x Yoffs(23:16)*	24 bit value representing the vertical offset in pixels of the image with respect to the top line of the full Device image. This value shall be set to 0 for line scan images. *Comment: Xoffs and Yoffs tell the Host where this image is with respect to the full Device image size. As it represents the full Device image, it is the same for both fields of an interlaced image.*
16	*4x Yoffs(15:8)*	
17	*4x Yoffs(7:0)*	
18	*4x DsizeL(23:16)*	24 bit value representing the number of data words per image line.
19	*4x DsizeL(15:8)*	
20	*4x DsizeL(7:0)*	

continued on next page …

Word	Content	Description
21	4x PixelF(15:8)	16 bit value representing the pixel format (see section 10.4.1.1).
22	4x PixelF(7:0)	
23	4x TapG(15:8)	16 bit value representing the tap geometry (see section 10.4.5).
24	4x TapG(7:0)	
25	4x Flags	Image flags: (8 bits, repeated 4 times) Bits: Usage: 1:0 Interlacing: 0x0 None (Progressive scan or line scan) 0x1 Interlaced, first line after header from field 1 * 0x2 Interlaced, first line after header from field 2 0x3 Reserved for future use 7:2 Reserved for future use, set to 0

* Field 1 is defined as the field that includes the top image line.

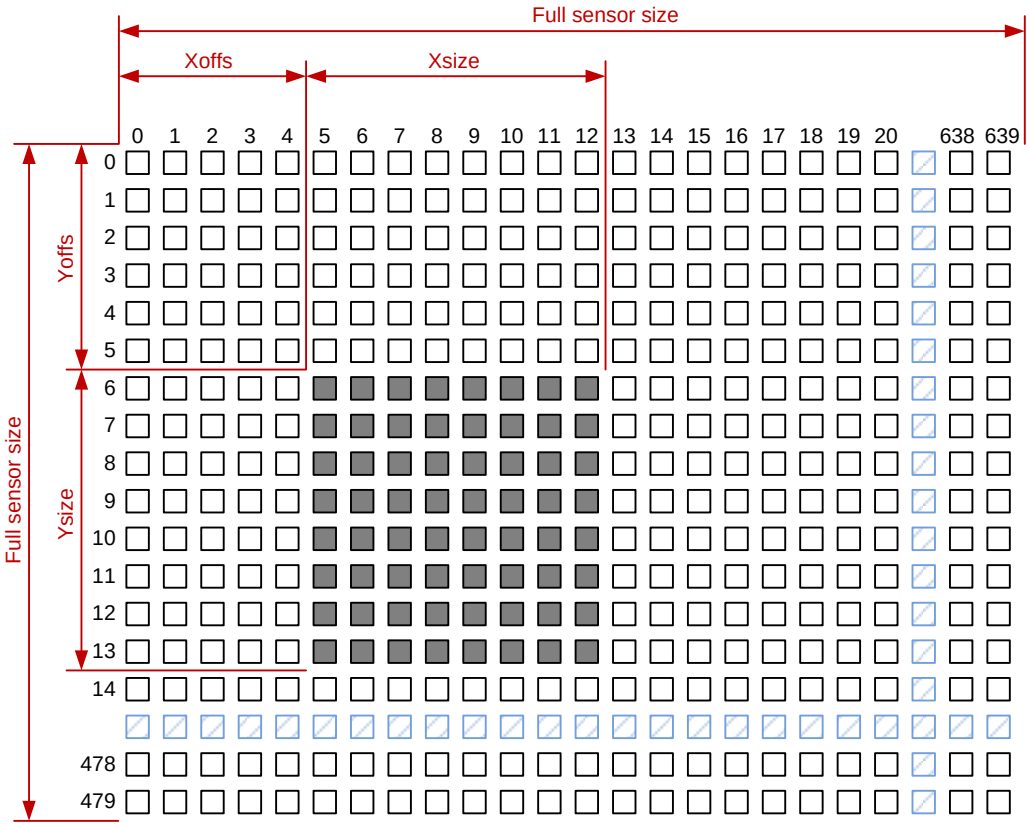


Figure 36 — Example 8x8 ROI in a VGA sized sensor

10.4.6.3 Rectangular Image Line Marker

The line marker shall be as shown in the following table:

Table 48 — Rectangular image line marker

Word	Content	Description
1	4x K28.3	Stream Marker.
2	4x 0x02	Rectangular line marker.

10.5 GenDC Streams

CoaXPress can be used to transmit GenICam GenDC data. This is an optional mode, controlled by *CapabilityRegister* bit *GenDcSupported* and *FeatureControlRegister* bit *GenDcEnable*.

Comment: Note that when GenDCEnable is set to 1, all image streaming must use GenDC. See section 12.3.47.

10.5.1 GenDC Transmission Rules

A GenDC Stream Header (section 10.5.2) shall be used to send content defined by the GenDC standard. Three types of GenDC Stream Header are defined:

GenDcType = 'GenDC data indication' shall be sent before the GenDC container's data. Only one data indication header shall be sent for one GenDC Data Container.

GenDcType = 'GenDC final descriptor indication' shall be sent before the GenDC container's final descriptor.

Comment: This will either be sent before the container's data if the data structure is completely known at the start, so the descriptor will not change, or sent after the container's data if a preliminary descriptor was sent. Note that GenDC does not allow a final descriptor to be sent after the container's data if a final descriptor was sent before the data; therefore if the data might change then a preliminary descriptor must be sent.

GenDcType = 'GenDC preliminary descriptor indication' shall be sent before the GenDC container's preliminary descriptor.

Comment: This is sent before the container's data if the descriptor may change. If a preliminary descriptor is sent then GenDC states that a final descriptor must be sent after the data.

Note that a preliminary descriptor may be used if the size of the GenDC data may be less than the value specified in the descriptor. In that case the transmission of the final descriptor indicates the completion of the container.

The multiple flows in GenDC can be mapped to multiple CoaXPress streams, or a single stream can be used to send all the CoaXPress data.

Comment: If only one stream is used, that may require the Device to buffer a significant amount of data. Using multiple streams avoids this buffering, providing the Host can accept the required number of streams.

If a single stream is used, it shall use stream 0. If multiple streams are used, stream 0 shall be used for the descriptors, and streams 1..N for GenDC flows as indicated by the GenDC FlowId.

The *SourceTag* shall be the same for all streams in one GenDC Data Container.

Example use cases resulting from these rules are shown below, where the figures have a blue background for content defined by the CoaXPress standard, and orange for content defined by the GenDC standard.

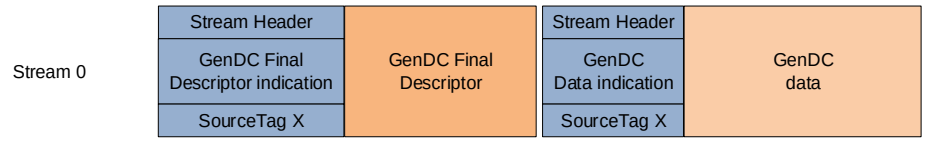


Figure 37 — Single stream without preliminary descriptor

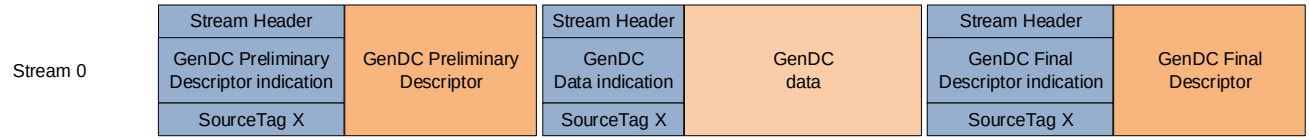


Figure 38 — Single stream with preliminary descriptor

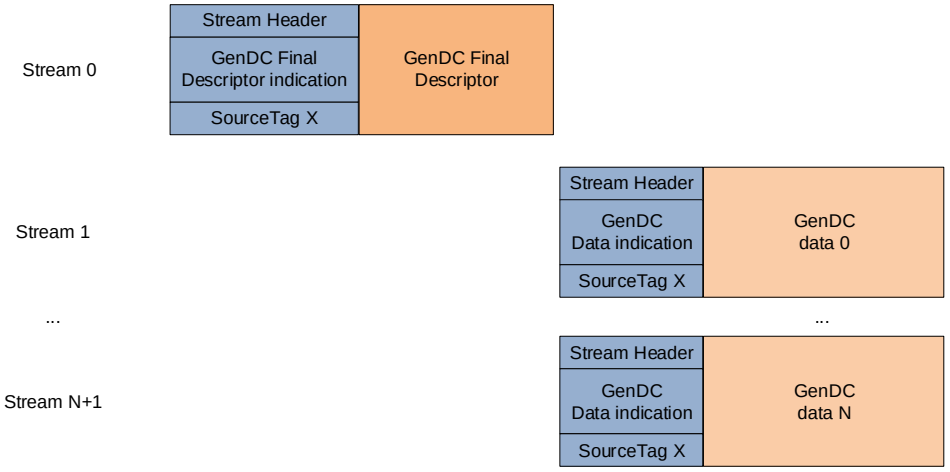


Figure 39 — Multiple streams without preliminary descriptor

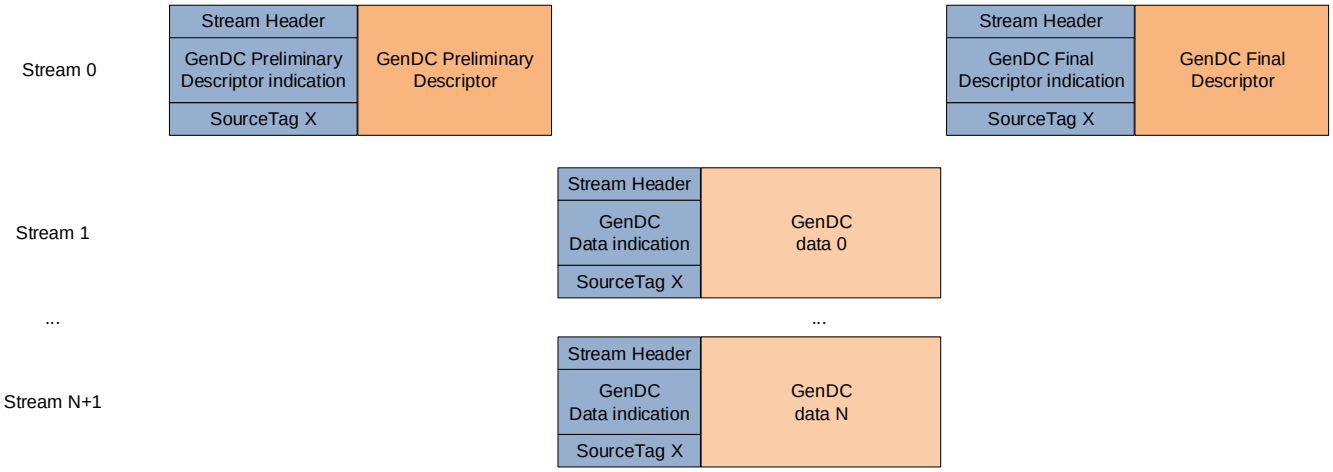


Figure 40 — Multiple streams with preliminary descriptor

10.5.2 GenDC Stream Header

The GenDC stream header shall be as shown in the following table:

Table 49 — GenDC stream header

Word	Content	Description
1	4x K28.3	Stream Marker.
2	4x GenDcType	This can have one of three values indicating the type of GenDC data that follows: 0x05 GenDC data indication 0x06 GenDC final descriptor indication 0x07 GenDC preliminary descriptor indication
3	*4x StreamID*	Unique stream ID.
4	*4x SourceTag(15:8)*	16 bit source image index. Incremented for each transferred GenDC Data Container, wraparound to 0 at 0xFFFF. The same number shall be used by each stream containing data relating to the same GenDC Data Container.
5	*4x SourceTag (7:0)*	
6	*4x GenDcDsize(47:40)*	48 bit value representing the number of data words in this stream following this header (i.e. starting at word 12) until the last word before the start of the next GenDC stream header. For GenDC data following a preliminary descriptor, this value represents the maximum number of data words; the data may be terminated by a final descriptor.
7	*4x GenDcDsize(39:32)*	
8	*4x GenDcDsize(31:24)*	
9	*4x GenDcDsize(23:16)*	
10	*4x GenDcDsize(15:8)*	
11	*4x GenDcDsize(7:0)*	

Note: The GenDcDsize field allows simple Hosts to receive and pass on the GenDC data without having to understand the GenDC descriptor.

10.5.3 GenDC Triggers – Line Scan

GenDC requires a line scan device to form pseudo-frames, therefore CoaXPress may be required to send triggers from the Host to the Device to indicate both lines and pseudo-frames, as well as for exposure control.

Comment: For comparison, in rectangular line scan image streams CoaXPress uses the Host to form pseudo-frames, therefore only line triggers need to be sent over the CoaXPress link.

CoaXPress therefore supports an additional trigger mode on the low speed upconnection to allow up to four different trigger types, LinkTrigger0 to LinkTrigger3, to be sent. See Table 16.

If the optional high speed upconnection is in use there are 16 different trigger types available, LinkTrigger0 to LinkTrigger15. See table Table 17.

10.5.4 GenDC Bootstrap Registers

Bootstrap registers provide the address and size of the GenDC prefetch descriptor and the GenDC flow table, as well as the size of the complete GenDC container. See sections 12.3.19 to 12.3.23.

11 Link Sharing

This section covers Link Sharing, which optionally allows one Device to stream data to more than one Host via multiple links.

Comment: This is mainly to support the use case where the Device is faster than the Host, such that the data bandwidth needs to be shared between multiple Hosts. It can also be used to build redundant systems where the same data is sent to more than one Host.

Link Sharing is optional. Its support is indicated by the *LinkSharingSupported* bit of the *CapabilityRegister* and enabled by the *LinkSharingEnable* bit of the *FeatureControlRegister*.

11.1 Concepts and Terminology

A Link Sharing system comprises one Device and two or more Hosts.

Comment: For a typical computer-based system, the Hosts may often be in the same computer, but this does not need to be the case.

A Device designed for use in Link Sharing systems has more than one link. These links shall be standard CoaXPress links comprising one master connection and optional extension connections. Each link is defined as a sub-Device, and provides a sub-set of the data from the Device. One sub-Device is the Master sub-Device, and this is the connection that controls the Device. The other sub-Devices are Slave sub-Devices. These concepts are shown in the following figure:

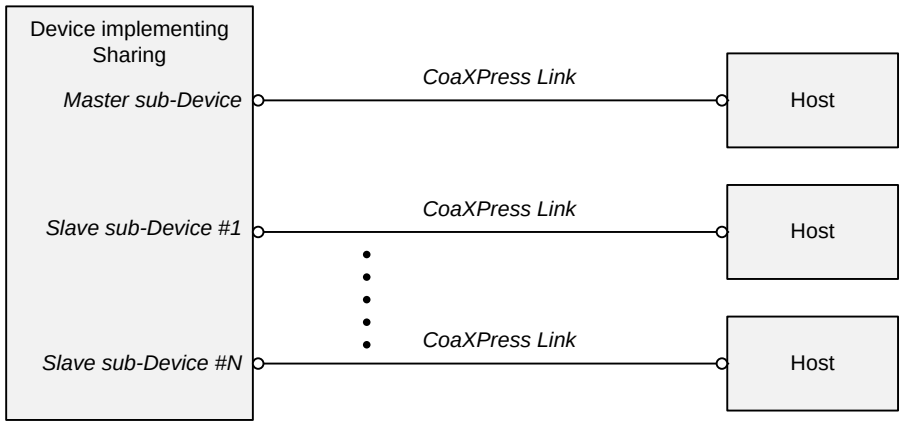


Figure 41 — Sharing Terminology

The configuration of multiple links on a Device is expected to be fixed as part of the Device's design, to minimize complexity, in particular for discovery.

Comment: Nothing in the standard prevents a Device supporting more than one link configuration via e.g. firmware options or a non-volatile register in manufacturer-specific space, but the standard does not support multiple link configurations to be changed 'on-the-fly'.

A Device with 'N' sub-Devices can require all 'N' links to be connected to Hosts before it can stream data.

Comment: This also reduces complexity in the Device, so it is not required to dynamically switch the mapping of sensor data to links based on how many Hosts are connected.

A Device shall use a unique set of stream IDs across all 'N' sub-Devices.

Comment: Therefore only the primary stream from the master sub-Device shall use stream ID = 0.

11.2 Discovery and Device Control

Each sub-Device shall conform to the standard rules for discovery defined in section 12.1.

Comment: This means that a Host does not need any changes to support discovery for a link sharing system.

Bootstrap register *LinkSharingStatus* indicates to the Host if it is connected to the Master sub-Device or a Slave sub-Device, and indicates which of its sub-Devices are discovered and ready to stream data. The Device shall not allow *AcquisitionStart* to be actioned until its required number of sub-Devices are discovered and ready to stream data.

11.3 Sharing Modes for Rectangular Image Streams

Rectangular image streams are shared by dividing the image into stripes. If a Device produces multiple rectangular streams, then the same stripe settings shall be used for each stream.

Comment: More complex Devices are expected to use GenDC, so this is not expected to be a significant restriction.

Bootstrap registers are used to read, and optionally control, the striping mode used by the Device. These registers describe the number of horizontal stripes, the number of vertical stripes, and overlap values between the stripes.

Comment: The best choice of striping mode for a Device is application-specific and beyond the scope of this standard. As a general rule horizontal stripes are easy to implement in Devices, but may give inefficient Host data transfer in a typical PCI Express environment, and vertical stripes may require additional buffering in Devices but are efficient for Hosts.

Comment: Overlap between stripes allows for image processing functions to be run that require access to adjacent pixels.

These bootstrap registers are defined in section 12.3 and are listed below:

LinkSharingHorizontalStripeCount

LinkSharingVerticalStripeCount

LinkSharingHorizontalOverlap

LinkSharingVerticalOverlap

LinkSharingDuplicateStripe

The Device shall form the *Rectangular Image Header* for each stripe with *Xsize* and *Ysize* giving the size of the stripe, and with *Xoffs* and *Yoffs* defining the location of the stripe within the entire Device image.

Comment: Xsize and Ysize defining the stripe size means that the Host does not need any special code to process the stripe data, and the Xoffs and Yoffs allow the entire image to be correctly reconstructed.

Example use cases are listed below, with the corresponding register settings. In each figure, one color is used for the data from one sub-Device.

11.3.1 Vertical Striping

The image in split into ‘v’ vertical regions, typically where ‘v’ is the number of sub-Devices. Each region shall be the same width, even if *LinkSharingVerticalOverlap* is non-zero.

LinkSharingHorizontalStripeCount = 0

LinkSharingVerticalStripeCount = v

LinkSharingDuplicateStripe = 0

The following example is for LinkSharingVerticalStripeCount = 4.

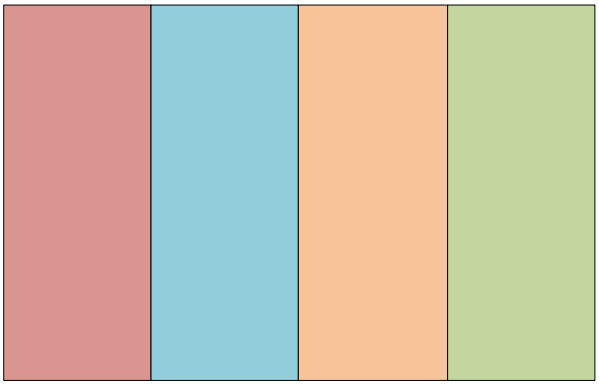


Figure 42 — Vertical Striping Example

11.3.2 Horizontal Striping

The image in split into ‘h’ horizontal regions, typically where ‘h’ is the number of sub-Devices. Each region shall be the same height, even if *LinkSharingHorizontalOverlap* is non-zero.

LinkSharingHorizontalStripeCount = h

LinkSharingVerticalStripeCount = 0

LinkSharingDuplicateStripe = 0

The following example is for LinkSharingHorizontalStripeCount = 2.

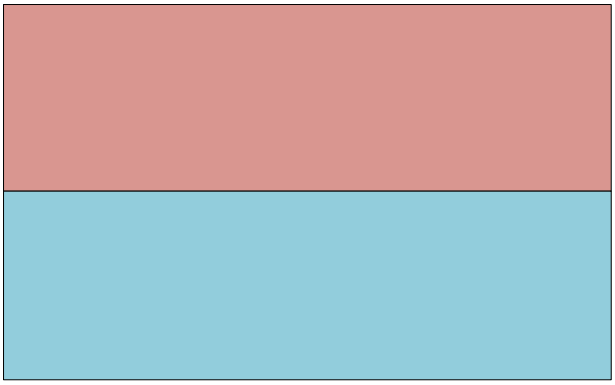


Figure 43 — Horizontal Striping Example

11.3.3 Line Interleaving

This is a limiting case of horizontal striping.

LinkSharingHorizontalStripeCount = ImageHeight / 2

LinkSharingVerticalStripeCount = 0

LinkSharingDuplicateStripe = 0

In the example below, there are only two sub-Devices, so the lines alternate between the sub-Devices.

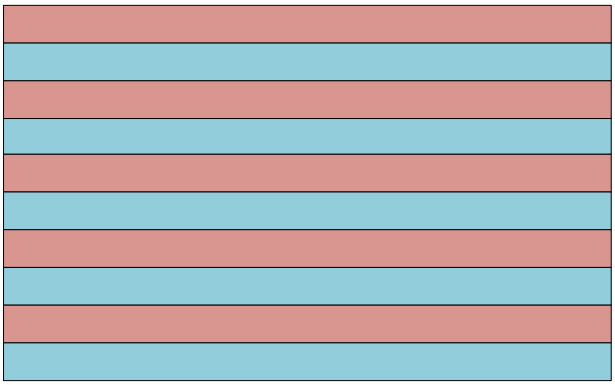


Figure 44 —Line Interleaving Example

11.3.4 Frame Interleaving

Alternate frames are sent to each sub-Device in turn.

LinkSharingHorizontalStripeCount = 1

LinkSharingVerticalStripeCount = 0

LinkSharingDuplicateStripe = 0

11.3.5 Image Duplication

The full frame is sent to all ‘d’ sub-Devices.

LinkSharingHorizontalStripeCount = 1

LinkSharingVerticalStripeCount = 0

LinkSharingDuplicateStripe = d

11.3.6 Combination of Modes

The register set allows for many combinations of striping and image duplication, some of which would not be common use cases. Examples are given here to help explain the meaning of the registers.

Four vertical columns, eight sub-Devices, each vertically striped image duplicated.

LinkSharingHorizontalStripeCount = 0

LinkSharingVerticalStripeCount = 4

LinkSharingDuplicateStripe = 2

11.3.7 Stripe Overlap

Bootstrap registers *LinkSharingHorizontalOverlap* and *LinkSharingVerticalOverlap* control the overlap of stripes to allow for image processing.

Comment: Note that the definitions of LinkSharingHorizontalStripeCount and LinkSharingVerticalStripeCount require all stripes to be the same size, even if overlap is enabled. As an example for horizontal striping, the number of non-overlapped pixels plus the number of overlapped pixels is the same for all stripes. This means that the left and right stripes contain more non-overlapped pixels. Similarly for vertical striping.

11.4 GenICam Support

The Master sub-Device shall present a standard GenICam XML file representing the entire image. Slave sub-Devices shall present read-only copies of the same XML file.

Comment: The read-only XML files give Hosts all the information they need to understand the setup of the Device.

For systems where all Hosts are in one computer unit, the GenTL Producer can reconstruct the full image so that the user need not be aware of the use of link sharing.

Comment: The GenTL Producer can only be expected to reconstruct common use cases, such as some of those listed above. Also note that vertical striping widths could clash with DMA alignment restrictions.

12 Device Setup

This section covers Device discovery and bootstrap registers.

12.1 Device Discovery

Device discovery is initiated by the Host and consists of the following tasks:

- Match Device transmitter and Host receiver bit rates to obtain register access (section 12.1.2).
- Discover Devices and the connection topology (section 12.1.3).
- Negotiate the CoaXPress version to use (section 12.1.4).
- Detect the high speed upconnection (not supported in this version of the standard) (section 12.1.5)
- Negotiate maximum packet sizes (section 12.1.6).
- Set the connection speed after reading *ConnectionConfigDefault* (section 12.1.7).

Discovered Devices need further configuration for operational use. Device configuration comprises bootstrap and Device register programming. Device configuration falls outside the scope of this standard.

Note: It is recommended that the implementation of the Device discovery process allows for users connecting and re-connecting cables while a system is powered up.

12.1.1 Connection State

Each transmitter connection has an associated connection state (Activated or Deactivated) as defined in the following table:

Table 50 — Transmitter connection state conditions

Connection state	Condition
Deactivated	No IDLE words are sent *Comment: In general this would be achieved by shutting down the SERDES for the connection, but if this is not practical, it is sufficient to leave the SERDES active, but not sending any IDLE words. Code D00.0 is recommended.*
Activated	The connection is at a defined bit rate (see Table 5 and Table 6) sending IDLE words as defined in section 9.2.5.

Each receiver connection has an associated connection state (Detected or Undetected) indicating the conditions defined in the following table:

Table 51 — Receiver connection state conditions

Connection state	Condition
Undetected	Data reception not possible: • Host connection not physically connected to a Device. • Host connection physically connected to a Device but no low level connection lock.
Detected	Data reception possible: • Host connection physically connected to a Device and low level connection lock achieved.

Comment: Connection state detection is closely related to receiver bit clock lock, word alignment status, running disparity error flagging, etc. The implementation of connection state detection is implementation specific and outside the scope of this standard, however IDLE checking is recommended – see section 12.2.

12.1.2 Match Device and Host Bit Rates

Without prior knowledge the Device and Host connection scheme is unknown and the bit rates of the Device transmitters and Host receivers may not match. However the low speed Host to Device connection has a fixed standard bit rate of 20.83˙ Mbps during discovery allowing transfer of control packets from Host to Device. During Device discovery the Host shall use a maximum control packet size of 128 bytes (see section 12.1.6).

The process to match the downconnection bit rates is described below and shown in Figure 45.

- For each Host connection the Host shall write a connection reset command by writing the value 1 to the Device *ConnectionReset* bootstrap register.
- A Device receiving this connection reset command via its Master connection (connection 0) shall execute connection reset and activate its discovery connection configuration within 200ms (see section 12.3.33).

  Comment: A Device receiving this connection reset command via an Extension connection shall ignore it. See section 12.3.28.
- The Host shall wait 200ms to allow for the Device to complete connection configuration.
- The Host shall initialize its receiver to one of the discovery bit rates it supports (see section 4.3).
- The Host shall monitor the connection state.
- If the connection state is Detected within a timeout period then the Host and Device have established a connection.

  Comment: A timeout period allows the Host receiver to attain lock and alignment. The required time is implementation specific and therefore not specified here.
- If the connection state is Undetected after the timeout period then the Host may repeat the process using another discovery bit rate.

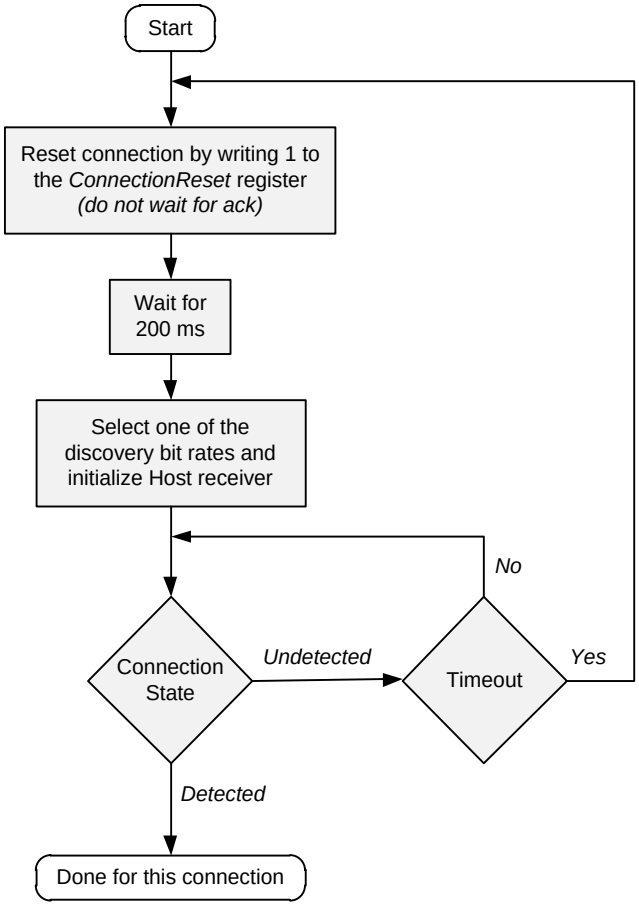


Figure 45 — Matching Device and Host bit rates

12.1.3 Discover Devices and Connection Topology

The Host shall read the *ConnectionConfigDefault* register to find the number of expected connections. It shall then write to the *ConnectionConfig* register to enable the number of connections read from *ConnectionConfigDefault*. However it shall not change from the discovery rate at this stage.

Comment: The Host needs to know how many connections the Device requires. It is of little use if only three connections are connected on a camera that needs four connections – the Host needs to report that state as a discovery failure. Similarly the Host must not complete discovery after finding two connections, if the user is still connecting cables and four are needed.

If the Host does not support the number of connections specified by *ConnectionConfigDefault*, or the required number of connections cannot be discovered within a timeout period, then the Host shall maintain the master connection at the discovery speed.

Comment: This allows a user to reprogram ConnectionConfigDefault (if this is supported by the Device). The value of the timeout period is beyond the scope of this standard, and may be infinite.

Connection schemes may vary and can be as simple as a single link connection of one Device to one Host connection, or as complicated as multiple Devices with multiple connections connected to a many connection Host as shown in the following figure:

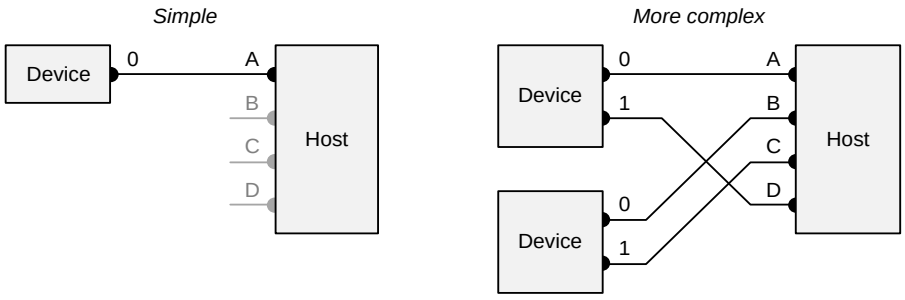


Figure 46 — Examples of Device – Host connection schemes

Note that a Host may not support all theoretically possible connection topologies.

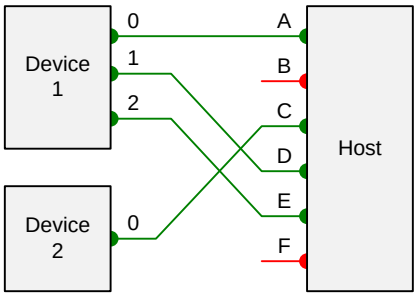
Comment: If all connection topologies are supported, then the topology discovery process means that the user does not need to plug in cables in a specific order. However supporting a subset of topologies may allow a Host to be more cost-effective.

The first step in connection topology discovery is to find out for each Detected Host connection whether it is connected to a Device Master connection (connection 0) or an extension connection (connection > 0).

This is accomplished by reading the *DeviceConnectionID* bootstrap registers for each Detected Host connection:

- Returned Device connection ID = 0: This Host connection goes to a Device Master connection.
- Returned Device connection ID > 0: This Host connection goes to a Device Extension connection.

An example is shown in the following figure:



Obtained information:

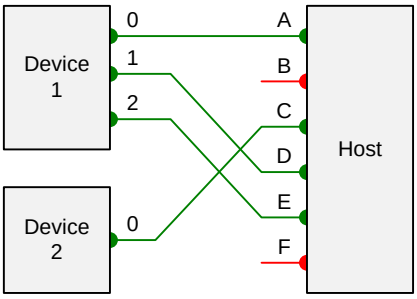
Host Connection	Connection State	Read DeviceConnectionId
A	Detected	0 (Master)
B	Undetected	-
C	Detected	0 (Master)
D	Detected	1 (Extension)
E	Detected	2 (Extension)
F	Undetected	-

Figure 47 — Connection topology example part 1

The next step is to find out the connection topology: which extension connection belongs to which master connection.

For each Host connection connected to a Device Master connection the Host shall write a Host connection identification number to the *MasterHostConnectionID* bootstrap registers. The Host connection identification number uniquely identifies the connection within the Host context. Any 32 bit number is allowed except for 0 which is reserved to indicate an unknown Host ID.

Finally by reading the *MasterHostConnectionID* bootstrap registers via all Host Connections the connection scheme is known, as shown in the following figure:



Obtained information:

Host Connection	Connection State	Read DeviceConnectionId	Write MasterHostConnectionId	Read MasterHostConnectionId
A	Detected	0 (Master)	HostID of A, *(e.g. 1)*	HostID of A *(1)*
B	Undetected	-	-	-
C	Detected	0 (Master)	HostID of C, *(e.g. 2)*	HostID of C *(2)*
D	Detected	1 (Extension)	-	HostID of A *(1)*
E	Detected	2 (Extension)	-	HostID of A *(1)*
F	Undetected	-	-	-

Figure 48 — Connection topology example part 2

12.1.4 Negotiate the CoaXPress Version Used

The Host shall read the *Revision* register of the Device.

Comment: Note that a Host can read the Revision register before this stage of discovery.

- If the version is 1.0 then the Host shall use 1.0 protocol, or shall report Device/Host incompatibility to the user if it cannot support 1.0.
- If the version is 1.1 then the Host shall use 1.1 protocol, or shall report Device/Host incompatibility to the user if it cannot support 1.1.
- If the version is 2.0 or greater, then a 2.0 or greater compatible Host shall read the *VersionsSupported* register of the Device. The Host can then compare the available versions supported by the Device with those that it supports. The Host shall select the highest common version, write that to register *VersionUsed*, and then switch to that version itself. If there is not a common version supported, the Host shall report Device/Host incompatibility to the user.

Comment: If the Revision register indicates 2.0 or greater, but the Host is not 2.0 compatible, then the default value of register VersionUsed ensures that revision 1.1 of the protocol is used by default. Note that this means that the Device must support version 1.1.

12.1.5 Detection of High Speed Upconnection

If the Host supports a high speed upconnection, it needs to discover if the Device also supports one, and if so, if there is a cable present to form the connection. A Host supporting a high speed upconnection shall therefore perform the following process.

- The Host shall read the *CapabilityRegister* bit *HsUpconnectionSupported* via the low speed connection.
- If the value of bit 0 of the register is “1”, then the Host shall enable its high speed upconnection at the current downconnection speed and send IDLE words on it. It shall also set the *HsUpconnectionEnable* bit in the *FeatureControl* register to 1.
- When the Device achieves connection state “Detected” (i.e bit and word lock) on its high speed upconnection, it shall switch all upconnection communications from the low speed connection to the high speed connection. Until the Device achieves connection state “Detected” it shall continue to use the low speed connection.
- The Host shall wait 200ms for the Device to achieve lock on the high speed connection.
- The Host shall then read the *CapabilityRegister* via the high speed connection. If the read succeeds, then the Host shall switch all upconnection communications to the high speed connection from the low speed connection. If the read fails then the Host shall disable its high speed upconnection (so not sending IDLE words), and continue to use the low speed upconnection. It shall also set the *HsUpconnectionEnable* bit in the *FeatureControl* register back to 0.

Comment: Even though both Host and Device support a high speed upconnection, the cables in use may not. The Host can report this as an error, but this is beyond the scope of the standard.

Comment: To save power, the Device can only enable its high speed upconnection receiver when the HsUpconnectionEnable bit is set to 1, providing it can enable it and achieve lock in 200ms.

Comment: Note that the Host is required to maintain IDLE words on the low speed upconnection even when it is using the high speed connection (see section 9.2.5.1). Therefore a switch back to the low speed connection is instantaneous.

12.1.6 Negotiate Maximum Packet Sizes

For each discovered Device the following packet sizes shall be negotiated:

- Control packet maximum size (see sections 9.6.4 and 12.3.36).
- Stream packet maximum size (see sections 9.5.2 and 12.3.37).

Control packet maximum size negotiation shall use the rules in the following table:

Table 52 — Control packet size negotiation

Device	Host
At all times: • Shall support a control packet size of up to its maximum supported control packet size. • Shall report the maximum supported control packet size, which shall be at least 132 bytes, in the *ControlPacketSizeMax* bootstrap register.	*Prior to Device discovery:* • Shall assume a maximum control packet size of 128 bytes for compatibility with version1.x compliant Devices where 128 bytes was the defined minimum value. *After discovering a Device:* • Shall read the *ControlPacketSizeMax* bootstrap register from a discovered Device. • May decide to use a control packet size up to the reported maximum value.

Stream packet maximum size negotiation shall use the rules in the following table:

Table 53 — Stream packet size negotiation

Device	Host
Connection reset: • Shall initialize the bootstrap register *StreamPacketSizeMax* to 0 indicating “not configured”. *While StreamPacketSizeMax=0:* • Shall not transmit stream packets. *When StreamPacketSizeMax>0 after a write from the Host:* • Transmission of stream packets allowed with a packet size up to the maximum size set by the *StreamPacketSizeMax* bootstrap register.	*After discovering a Device:* • Shall write the maximum stream packet size supported by the Host to the *StreamPacketSizeMax* bootstrap register.

12.1.7 Setting the Operating Bit Rate

The Host shall read the *ConnectionConfigDefault* register to find the required bit rate and number of connections operating at this bit rate, and shall then use the process in section 12.1.7.1 to set that rate and number of connections.

The bit rate and number of connections operating at this bit rate can also be changed at any point after Discovery that a new bit rate is needed on a link, in which case the process in section 12.1.7.1 shall be used. In that case the required new value of *ConnectionConfig c*an be found from GenICam (see section 13) or system documentation.

All connections forming one link, including a high speed upconnection, shall operate at the same bit rate.

12.1.7.1 Setting the Bit Rate

- The Host shall write the required *ConnectionConfig* value to the Device and wait for the acknowledgement at the original connection speed.
- The Device shall acknowledge the *ConnectionConfig* access, then shall change all connections specified by *ConnectionConfig* to the specified bit rate. If the new high speed connection bit rate requires a change in low speed connection bit rate, it shall also change the low speed upconnection speed to the value defined in Table 6.
- The Host shall write to its corresponding register(s) to set the corresponding connections to the same bit rate. If the new high speed connection bit rate requires a change in low speed connection bit rate, it shall also change the low speed upconnection speed to the value defined in Table 6.
- After it is sending a stable low speed upconnection at the defined rate the Host shall wait 200ms to allow the Device to complete connection re-configuration.
- The Host shall monitor the connection state.
- If the connection state is Detected within a timeout period the Host and Device have re-established the connection.
- If the connection state is Undetected after a timeout period then the Host may repeat the process or return to the Discovery procedure.

Comment: This section requires that a Device completes re-configuration within 200ms after a change of ConnectionConfig, so that the Host can detect a Detected connection state. However some Devices may require longer than 200ms to be ready to perform certain operations, such as image streaming. In that case the Device can send a wait acknowledgement in response to any command (except for accessing the Bootstrap registers) that it is not ready to perform – see section 9.6.1. Alternatively GenICam supports a transaction method by using suitable XML code that can specify a polling register.

See the following figure:

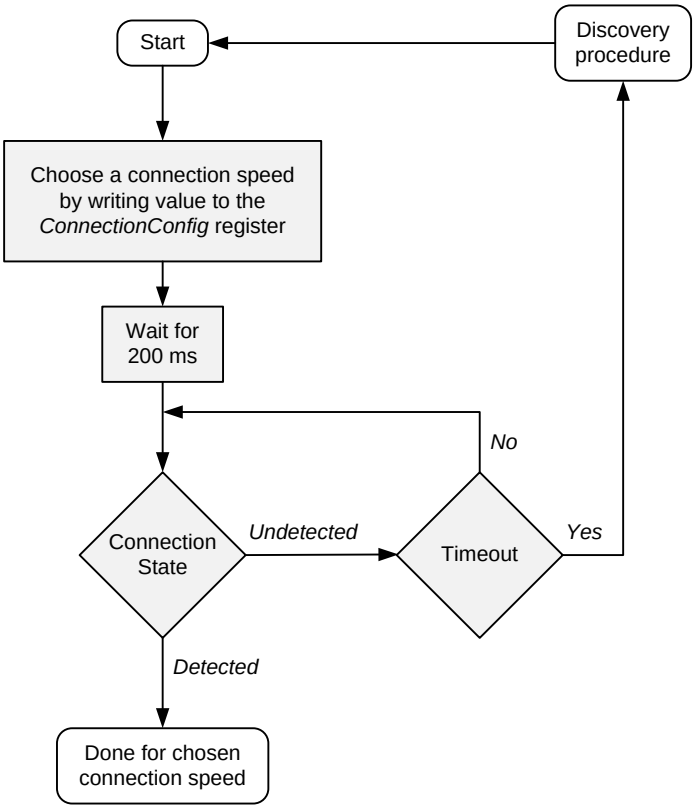


Figure 49 — Setting the bit rate

12.2 Detection of Loss of Lock

It is recommended that the presence of IDLE is checked on each active connection at the rates specified in section 9.2.5.1. If IDLE is not seen for a consecutive number of occasions set by a threshold, it is recommended that first the receiver for that connection is reset (so e.g. character and word alignment re-established).

If at the Host this does not recover lock, the Host shall reset the Device by writing to *ConnectionReset* and the discovery process re-tried (i.e. on all connections).

If at the Device this does not recover lock, the Device shall deactivate the corresponding downconnection and switch to 20.83˙ Mbps upconnection speed.

Comment: Remember that all low speed upconnections are activated and therefore send IDLEs.

This will force the Host to detect loss of downconnection lock and rediscover the Device via a *ConnectionReset*.

Additionally, in the case of the master connection, the Device shall also deactivate all extension connections and in the case of the High Speed upconnection, the Device shall deactivate the master and all extension connections.

Comment: A connection that is not activated shall not send IDLE words. See section 12.1.1.

Comment: The IDLE word shows that the connection is functioning, and confirms that the receiver is correctly word aligned. The value for the threshold may be system-dependent, but should not be set so low that "burst" type errors, e.g. as a result of EMC/EMI type events, cause the connection to reset. There is some correlation between PoCXP events (see section 8) and loss-of-lock detection. E.g. if the Host switches off Device power then IDLEs will not be received and the connection will get reset, but note that if a Device is connected that does not use PoCXP, a connection will be established without any PoCXP event.

12.3 Device Memory Space and Bootstrap Registers

Device memory space allows both registers to control the Device, and larger blocks of memory such as for the XML file needed for GenICam. The start of Device memory space has defined bootstrap registers.

12.3.1 Strings

Strings stored in the bootstrap and manufacturer-specific registers space shall be NULL-terminated, encoded ASCII. The NULL terminator is included in the string length. However when a register stores a string which is the maximum length of that register, it does not have a NULL terminator.

Comment: This is standard GenICam behavior.

12.3.2 Feature Size

Reads and writes of bootstrap and manufacturer-specific registers with a size up to and including 104 bytes shall be implemented as one Control Command message.

Comment: This simplifies updating non-volatile memory in a Device, and ensures that most numerical values can be read or written in one access. 104 bytes is the payload size of the minimum value of ControlPacketSizeMax (128 bytes for version 1.x Devices, or 132 bytes for version 2.x and higher Devices), therefore all CoaXPress products will support a read or write of this size.

Comment: In general it will be the role of the Host software to ensure that this requirement is met.

Device memory space shall be implemented so that reads and writes with a size over 104 bytes can be read or written either as one Control Command message, or as multiple word aligned messages of data size between 4 bytes and the defined register size, subject to the access not exceeding that allowed by *ControlPacketSizeMax*.

Comment: Large areas of Device memory such as XML files will in general exceed the maximum control packet size so reads and writes must be split into multiple accesses.

Note: Zipped XML files must be the correct size for the unzip process to work, and this size may not be a multiple of 4 bytes. Therefore Devices using zipped XML files, and all Hosts, shall support read accesses that are not a multiple of 4 bytes.

12.3.3 Miscellaneous Rules

Bootstrap registers shall be defined in the XML Device configuration file and respect the rules defined in SFNC (Standard Features Naming Convention) of the GenICam standard (see section 13).

All Device registers shall be 32-bit aligned, big-endian.

The Device shall not send a wait acknowledgement to an access to a bootstrap register, with the exception that a wait acknowledgement is allowed when writing to a writable GenICam group register.

Unused bootstrap register bits shall be set to 0.

Manufacturer-specific registers can start after the bootstrap register memory-space.

It is recommended that writable registers in the manufacturer-specific space are also readable.

Comment: This is better for high-reliability systems.

12.3.4 Bootstrap Registers

CoaXPress compliant Devices shall implement all mandatory bootstrap registers defined in this section.

The bootstrap registers are listed in Table 54, which uses the following indications:

Group:	Support:	This register is needed to support other standards, such as GenICam.
	GenICam:	This register implements a GenICam feature of the same name.
	CXP:	This is a register defined for the CoaXPress standard.
Support:	M:	Mandatory.
	O:	Optional.
	X:	Recommended to include in the Device's XML file.
	E:	Access possible via extension connections. See register description for details. No other registers shall be accessed via an extension connection.
Access:	R:	Read and/or
	W:	Write.
		Access modes given in parentheses are optional.
Default value:		This is the value after a Connection Reset.
		Comment: Note that a connection reset occurs at power up, so this is also the power-up value.

Table 54 — Bootstrap registers

Address	Name	Group	Support	Access	Length (bytes)	Register Interface
0x00000000	Standard	Support	M	R	4	Integer
0x00000004	Revision	Support	M, X	R	4	Integer
0x00000008	XmlManifestSize	Support	M	R	4	Integer
0x0000000C	XmlManifestSelector	Support	M	R/W	4	Integer
0x00000010	XmlVersion [XmlManifestSelector]	Support	M, X	R	4	Integer
0x00000014	XmlSchemaVersion [XmlManifestSelector]	Support	M	R	4	Integer
0x00000018	XmlUrlAddress [XmlManifestSelector]	Support	M	R	4	Integer
0x0000001C	Iidc2Address	Support	M	R	4	Integer
0x00002000	DeviceVendorName	GenICam	M, X	R	32	String
0x00002020	DeviceModelName	GenICam	M, X	R	32	String
0x00002040	DeviceManufacturerInfo	GenICam	M, X	R	48	String
0x00002070	DeviceVersion	GenICam	M, X	R	32	String
0x000020B0	DeviceSerialNumber	GenICam	M, X	R	16	String
0x000020C0	DeviceUserID	GenICam	M, X	R/W	16	String
0x000020D0	GenDcPrefetchDescriptorAddress	GenICam	O, X	R	4	Integer
0x000020D4	GenDcPrefetchDescriptorSize	GenICam	O, X	R	4	Integer
0x000020D8	GenDcFlowTableAddress	GenICam	O, X	R	4	Integer
0x000020DC	GenDcFlowTableSize	GenICam	O, X	R	4	Integer
0x000020E0	GenDcContainerSize	GenICam	O, X	R	4	Integer
0x00003000	WidthAddress	CXP	M	R	4	Integer
0x00003004	HeightAddress	CXP	M	R	4	Integer
0x00003008	AcquisitionModeAddress	CXP	M	R	4	Integer
0x0000300C	AcquistionStartAddress	CXP	M	R	4	Integer
0x00003010	AcquistionStopAddress	CXP	M	R	4	Integer
0x00003014	PixelFormatAddress	CXP	M	R	4	Integer
0x00003018	DeviceTapGeometryAddress	CXP	M	R	4	Integer
0x0000301C	Image1StreamIDAddress	CXP	M	R	4	Integer
0x00003018 + <n> x 4	Image<n>StreamIDAddress	CXP	M	R	4	Integer

continued on next page …

Bootstrap registers continued…

Address	Name	Group	Support	Access	Length (bytes)	Interface
0x00004000	ConnectionReset	CXP	M, E	W/(R)	4	Integer
0x00004004	DeviceConnectionID	CXP	M, E	R	4	Integer
0x00004008	MasterHostConnectionID	CXP	M, E	R/W	4	Integer
0x0000400C	ControlPacketSizeMax	CXP	M	R	4	Integer
0x00004010	StreamPacketSizeMax	CXP	M	R/W	4	Integer
0x00004014	ConnectionConfig	CXP	M, X	R/W	4	Integer
0x00004018	ConnectionConfigDefault	CXP	M, X	R	4	Integer
0x0000401C	TestMode	CXP	M, X	R/W	4	Integer
0x00004020	TestErrorCountSelector	CXP	M, X	R/W	4	Integer
0x00004024	TestErrorCount [TestErrorCountSelector]	CXP	M, X	R/W	4	Integer
0x00004028	TestPacketCountTx [TestErrorCountSelector]	CXP	M, X	R/W	8	Integer
0x00004030	TestPacketCountRx [TestErrorCountSelector]	CXP	M, X	R/W	8	Integer
0x00004038	ElectricalComplianceTest	CXP	O	R/W	4	Integer
0x0000403C	CapabilityRegister	CXP	M	R	4	Integer
0x00004040	FeatureControlRegister	CXP	M	R/W	4	Integer
0x00004044	VersionsSupported	CXP	M, X	R	4	Integer
0x00004048	VersionUsed	CXP	M, X	R/W	4	Integer
0x0000404C	LinkSharingStatus	CXP	O	R	4	Integer
0x00004050	LinkSharingHorizontalStripeCount	CXP	O, X	R/(W)	4	Integer
0x00004054	LinkSharingVerticalStripeCount	CXP	O, X	R/(W)	4	Integer
0x00004058	LinkSharingHorizontalOverlap	CXP	O, X	R/(W)	4	Integer
0x0000405C	LinkSharingVerticalOverlap	CXP	O, X	R/(W)	4	Integer
0x00004060	LinkSharingDuplicateStripe	CXP	O, X	R/(W)	4	Integer
0x00006000	Start of manufacturer-specific register space	-	-	-	-	-

12.3.5 Standard

This register shall provide a magic number indicating the Device implements the CoaXPress standard. The magic number shall be 0xC0A79AE5.

Comment: The magic number is an approximation of "CoaXPress".

12.3.6 Revision

This register shall provide the version of the CoaXPress standard implemented by this Device. Note that the register *VersionsSupported* added in version 2.0 provides more information about what the Device can support.

Bit Offset (LSB << x)	Width (bits)	Name	Example
16	16	Major version	v1.5 has major version 1, therefore these bits would hold the value 0x0001
0	16	Minor version	v1.5 has minor version 5, therefore these bits would hold the value 0x0005

Therefore Devices compliant to this version 2.1 of the standard shall return 0x00020001.

Comment: The sub-minor version is not coded in this register.

12.3.7 XmlManifestSize

This register shall provide the number of XML manifests available. At least one manifest shall be available.

12.3.8 XmlManifestSelector

This register shall select the required XML manifest registers. It shall hold a number between 0 and *XmlManifestSize* – 1.

Comment: A connection reset sets the value 0x00000000.

12.3.9 XmlVersion[XmlManifestSelector]

This register shall provide the version number for the XML file given in the manifest referenced by register *XmlManifestSelector*.

Bit Offset (LSB << x)	Width (bits)	Name	Description
24	8	Reserved	Shall be 0
16	8	XMLMajorVersion	The major version number of the XML file
8	8	XMLMinorVersion	The minor version number of the XML file
0	8	XMLSubMinorVersion	The sub-minor version number of the XML file

12.3.10 XmlSchemaVersion[XmlManifestSelector]

This register shall provide the GenICam schema version for the XML file given in the manifest referenced by register *XmlManifestSelector*.

Bit Offset (LSB << x)	Width (bits)	Name	Description
24	8	Reserved	Shall be 0
16	8	SchemaMajorVersion	The major version number of the schema used by the XML file
8	8	SchemaMinorVersion	The minor version number of the schema used by the XML file
0	8	SchemaSubMinorVersion	The sub-minor version number of the schema used by the XML file

12.3.11 XmlUrlAddress[XmlManifestSelector]

This register shall provide the address of the start of the URL string referenced by register *XmlManifestSelector*. This address shall be in manufacturer-specific register space (i.e above 0x6000). The string shall be in the format defined in by GenTL (see section 13).

12.3.12 Iidc2Address

For Devices that support the IIDC2 protocol (section 2.2 ref 4), this register shall provide the address of the start of the IIDC2 register space.

For Devices that do not support IIDC2, this register shall be set to the value 0x00000000.

Comment: IIDC2 can be supported to allow existing IIDC2 applications to be used.

Comment: Note that GenICam support is mandatory, therefore the Device needs an XML file that describes the IIDC2 registers. This XML file may be common to many IIDC2 products.

12.3.13 DeviceVendorName

This register shall provide the name of the manufacturer of the Device as a string.

12.3.14 DeviceModelName

This register shall provide the model name of the Device as a string.

12.3.15 DeviceManufacturerInfo

This register shall provide extended manufacturer-specific information about the Device as a string.

12.3.16 DeviceVersion

This register shall provide the version of the Device as a string.

12.3.17 DeviceSerialNumber

This register shall provide the serial number for the Device as a NULL-terminated string.

Comment: In v1.0 of the standard this was named DeviceID. It has been renamed because the GenICam feature it maps to has been renamed to DeviceSerialNumber.

12.3.18 DeviceUserID

This register shall provide a user-programmable identifier for the Device as a string. The Device shall provide persistent storage for this register so the value is maintained when power is switched off.

12.3.19 GenDcPrefetchDescriptorAddress

This register shall only be supported if GenDC is supported, as indicated by *CapabilityRegister* bit *GenDcSupported*.

This register shall provide the address of the start of the GenDC prefetch descriptor, as defined by the GenDC specification.

12.3.20 GenDcPrefetchDescriptorSize

This register shall only be supported if GenDC is supported, as indicated by *CapabilityRegister* bit *GenDcSupported*.

This register shall provide the size in bytes of the GenDC prefetch descriptor.

12.3.21 GenDcFlowTableAddress

This register shall only be supported if GenDC is supported, as indicated by *CapabilityRegister* bit *GenDcSupported*.

This register shall provide the address of the start of the GenDC flow table, as defined by the GenDC specification.

12.3.22 GenDcFlowTableSize

This register shall only be supported if GenDC is supported, as indicated by *CapabilityRegister* bit *GenDcSupported*.

This register shall provide the size in bytes of the GenDC flow table.

12.3.23 GenDcContainerSize

This register shall only be supported if GenDC is supported, as indicated by *CapabilityRegister* bit *GenDcSupported*.

This register shall provide the size in bytes of the complete GenDC Container.

Comment: This is the equivalent of SFNC PayloadSize and permits to setup the receiver buffer in advance at low level without reading that information from the GenDC Descriptor (useful in GenTL for example).

12.3.24 WidthAddress

12.3.25 HeightAddress

12.3.26 AcquisitionModeAddress

12.3.27 AcquisitionStartAddress

12.3.28 AcquisitionStopAddress

12.3.29 PixelFormatAddress

12.3.30 DeviceTapGeometryAddress

12.3.31 Image1StreamIDAddress

12.3.32 Image<n>StreamIDAddress

These registers shall provide the address in the manufacturer-specific register space of the use-case feature with the corresponding name (see section 13.2.1). *Image<n>StreamIDAddress* shall be set to 0 for stream numbers that are not supported by the Device.

Comment: This allows non-GenICam applications, or "black-box" type format converters, to support the standard use-case and allow continuous acquisition and display of images.

12.3.33 ConnectionReset

Writing the value 0x00000001 to this register shall provide Device connection reset. The Device shall also execute a connection reset after power-up.

ConnectionReset is always sent at the upconnection speed 20.83˙ Mbps.

Comment: If the upconnection speed in the Device was 41.6˙ Mbps then it would not recognize the ConnectionReset (sent at 20.83˙ Mbps), but it will quickly detect link loss and drop back to 20.83˙ Mbps. See section 12.2.

A Device receiving this connection reset command via its Master connection (connection 0) shall execute connection reset and activate its discovery connection configuration within 200ms. The Device shall clear the register back to 0x00000000 when it has activated its discovery connection configuration. The write by the Host should be regarded "fire and forget" without waiting for an acknowledgment.

Connection reset shall comprise:

- Activate the master connection. Extension connections shall not be activated.

Comment: A connection that is not activated shall not send IDLE words. See section 12.1.1.

- Initialize the bit rate for the master connection to the lowest discovery bit rate it supports (see section 4.3), with the corresponding value stored in *ConnectionConfig*, and the corresponding low speed upconnection receiver speed of 20.83˙ Mbps.
- *MasterHostConnectionID* shall be set to 0x00000000, indicating unknown Host Connection ID.

- *StreamPacketSizeMax* shall be set to 0x00000000, indicating “not initialized”, and preventing any data packets being sent. The Device therefore sends just IDLE characters on the master connection.
- The stream control in the Device shall be reset so that the Packet Tag in the stream data packet shall start from 0, and for a multi-connection Device, the first stream packet sent shall be on connection 0.
- Reset the tag in both Command packets and Event packets to 0.
- *TestMode* and *TestErrorCountSelector* shall be set to 0x00000000.
- All test mode counters shall be set to 0.
- *ElectricalComplianceTest* shall be set to 0x00000000 when a connection reset is received from the Host, but it shall not be changed by the connection reset automatically executed after power-up.
- *XmlManifestSelector* shall be set to 0.
- *FeatureControlRegister* shall be set to 0x00000000.
- *VersionUsed* shall be set to 0x00010001.

Comment: Therefore the Device will use Command packets without tag.

Note: The Device shall ignore a connection reset received on a Device extension connection.

Comment: A Host does not initially know what it has been connected to, so it will send connection reset commands to all connections. The Device needs to ignore those on extension connections or discovery will fail.

Comment: In general it is not possible to read this register while it has the value 0x00000001.

12.3.34 DeviceConnectionID

This register shall provide the ID of the Device connection via which this register is read, as shown in Figure 47.

Comment: A Connection ID of 0 means that the connection is a Master connection. This is a static register, but with a different value depending on which connection it is read from.

12.3.35 MasterHostConnectionID

This register shall hold the Host Connection ID of the Host connection connected to the Device Master connection. The value 0x00000000 is reserved to indicate an unknown Host ID.

Note: The Device shall ignore a MasterHostConnectionID write to a Device extension connection.

Comment: This register is set by the Host to a system wide (Host) unique 32 bit ID number as part of the Device discovery process (see section 12.1.3). A connection reset sets the value 0x00000000.

12.3.36 ControlPacketSizeMax

This register shall provide the maximum control packet size the Host can read from the Device, or write to the Device. The size is defined in bytes, and shall be a multiple of 4 bytes. The defined size is that of the entire packet, not just the payload.

Comment: The control packet size is at least 132 bytes for version 2.x Devices. See section 12.1.6.

12.3.37 StreamPacketSizeMax

This register shall hold the maximum stream packet size the Host can accept. The size is defined in bytes, and shall be a multiple of 4 bytes. The Device can use any packet size it wants to up to this size. The defined size is that of the entire packet, not just the payload.

Comment: This register is set by the Host as part of the Device discovery process. A connection reset sets the value 0x00000000. See section 12.1.6.

12.3.38 ConnectionConfig

This register shall hold a valid combination of the Device connection speed and number of active downconnections. Writing to this register shall set the connection speeds on the specified connections. It may also result in a corresponding speed change of the low speed upconnection. If the new *ConnectionConfig* value results in a change of connection speed, the Device shall acknowledge the *ConnectionConfig* access at the original connection speed. Therefore it shall acknowledge the access before changing connection speed. See section 12.1.7.1.

Comment: Not all theoretical combinations of connection speed and number of connections may be usable. One register is used to ensure that the two variables are set simultaneously. The XML file and product documentation give valid combinations for the Device. A connection reset sets the value corresponding to the selected discovery rate and one connection.

Bit Offset (LSB << x)	Width (bits)	Name	Description
16	16	Number of connections	Number of activated Device downconnections (e.g. 1 for one connection, 2 for two connections, etc.)
0	16	Connection speed	Bit rate selection code, see Table 55.

Table 55 — Bit rate codes

Bit Rate (Gbps)	Bit Rate Code
1.250	0x28
2.500	0x30
3.125	0x38
5.000	0x40
6.250	0x48
10.000	0x50
12.500	0x58

Comment: A connection that is not activated shall not send IDLE words. See section 12.1.1.

A write to *ConnectionConfig* shall reset the stream control in the Device so that the Packet Tag in the stream data packet shall start from 0, and for a multi-connection Device, the first stream packet sent shall be on connection 0. The stream control shall be reset even if the *ConnectionConfig* value is unchanged.

Comment: The tags in the command and event packets are not reset by writing to ConnectionConfig – see sections 9.6.1.2 and 9.8.1.1.

12.3.39 ConnectionConfigDefault

This register shall provide the value of the *ConnectionConfig* register that allows the Device to operate in its recommended mode.

Comment: The recommended mode would usually be the mode which gives the maximum performance.

If a Device can operate in more than one *ConnectionConfig* mode, it shall use the User Set control mechanism as defined by GenICam to allow a user to change *ConnectionConfigDefault*.

Comment: This allows a user to e.g. set up a two connection camera to operate on one connection if that is appropriate for the user's system. See section 12.1.3.

Comment: The User Set mechanism ensures that a coherent set of registers are saved to the Device, e.g. with the frame rate set to a value appropriate for the ConnectionConfigDefault value.

When using the *UserSetSave* command of the User Set control mechanism, the current *ConnectionConfig* value shall be saved to *ConnectionConfigDefault* of the selected User Set.

Example: Set ConnectionConfig to the desired (future) default configuration. Set UserSetSelector to the desired User Set. Execute UserSetSave which will save the value of ConnectionConfig to ConnectionConfigDefault of the desired User Set. Set UserSetDefault to the desired User Set. After a (re-)discovery the new ConnectionConfigDefault selected by UserSetDefault will be used.

12.3.40 TestMode

Writing the value 0x00000001 to this register shall enable test packets transmission from Device to Host. The value 0x00000000 shall allow normal operation. When the value is changed from 0x00000001 to 0x00000000 the Device shall complete the packet of 1024 test words currently being transmitted. See section 9.9.4.

Comment: A connection reset sets the value 0x00000000.

12.3.41 TestErrorCountSelector

This register shall select the required test count [TestErrorCountSelector] register. It shall hold a valid Device Connection ID 0 .. *n*-1, or *n* for the optional high speed upconnection.

Comment: A connection reset sets the value 0x00000000.

12.3.42 TestErrorCount[TestErrorCountSelector]

This register shall provide the current connection error count for the connection referred to by register *TestErrorCountSelector*.

Writing 0x00000000 to this register shall reset to zero the connection error count for the connection referred to by register *TestErrorCountSelector*.

Comment: A connection reset sets all connection test counters to zero. The error count is the number of incorrect words that have been received in test packets (see section 9.9.3).

12.3.43 TestPacketCountTx[TestErrorCountSelector]

This register shall provide the current transmitted connection test packet count for the connection referred to by register *TestErrorCountSelector*. See section 9.9.3.

Writing 0x00000000 to this register shall reset to zero the transmitted connection packet count for the connection referred to by register *TestErrorCountSelector*.

Comment: A connection reset sets all connection test counters to zero.

12.3.44 TestPacketCountRx[TestErrorCountSelector]

This register shall provide the current received connection test packet count for the connection referred to by register *TestErrorCountSelector*. See section 9.9.3.

Writing 0x00000000 to this register shall reset to zero the received connection packet count for the connection referred to by register *TestErrorCountSelector*.

Comment: A connection reset sets all connection test counters to zero.

12.3.45 ElectricalComplianceTest

If implemented, this shall be a non-volatile register to support formal compliance testing of the Device. It shall not be used at any other time. Writing the value 0x00000000 shall allow normal operation. Writing a valid *ConnectionConfig* value shall result in the following behavior when the Device is next powered up and no commands are sent on the upconnection.

- The connection speed and number of connections shall be set according to the value written to this register.
- Test packets as defined in section 9.9.2 shall be output on these connections.
- No other packets shall be sent on the connections other than Control Acknowledges.
- The Connector Indicator Lamps (section 6.4), if fitted, shall indicate compliance test mode.
- The Device shall not require any packets or IDLE words to be sent on the upconnection.

Comment: This allows the Device to be plugged into an analyzer for physical layer compliance testing. The Device can be setup for the bit rate to be tested, then when it is unplugged from the Host and plugged into the analyzer it will immediately output test data at the required bit rate. However when connected to a Host, the usual ConnectionReset access during discovery resets this register to 0x00000000, so any accidental access to this register (e.g. by a crashed Host PC) will automatically recover.

Comment: Alternatively, a manufacturer can supply a Device with dedicated firmware to allow physical layer compliance testing.

12.3.46 CapabilityRegister

This register shall indicate Device support of the optional features. It is used in conjunction with the *FeatureControlRegister*.

Bit Offset (LSB << x)	Width (bits)	Name	Description
4	28	Reserved	Shall be 0
3	1	GenDcSupported	Shall be 1 if the Device supports GenDC streaming (see section 10.5), otherwise shall be 0
2	1	ExtraLsTriggerSupported	Shall be 1 if the Device supports additional LinkTrigger modes on the low speed upconnection (see Table 16), otherwise shall be 0
1	1	LinkSharingSupported	Shall be 1 if the Device supports link sharing (see section 11), otherwise shall be 0
0	1	HsUpConnectionSupported	Shall be 1 if the Device supports a high speed upconnection, otherwise shall be 0

12.3.47 FeatureControlRegister

This register shall be used to enable or disable optional Device features. It is used in conjunction with the *CapabilityRegister*.

Bit Offset (LSB << x)	Width (bits)	Name	Description
4	28	Reserved	Shall be 0
3	1	GenDcEnable	Shall be 1 to enable GenDC streaming (see section 10.5), otherwise shall be 0. When this register is set to 1, all image streaming shall use GenDC.
2	1	ExtraLsTriggerEnable	Shall be 1 to enable additional LinkTrigger modes on the low speed upconnection (see Table 16), otherwise shall be 0
1	1	LinkSharingEnable	Shall be 1 to enable link sharing (see section 11), otherwise shall be 0
0	1	HsUpConnectionEnable	Shall be 1 to enable to high speed upconnection (via the discovery process defined in section 12.1.5), otherwise shall be 0

Comment: A connection reset sets the value 0x00000000.

12.3.48 VersionsSupported

This register shall provide the version(s) of the CoaXPress standard supported by the Device.

Bit Offset (LSB << x)	Width (bits)	Name	Description
4	28	Reserved	Shall be 0
3	1	Version2.1Supported	Shall be 1 if the Device supports CXP v2.1
2	1	Version2.0Supported	Shall be 1 if the Device supports CXP v2.0
1	1	Version1.1Supported	Shall be 1 if the Device supports CXP v1.1. Note that v1.1 support is mandatory to allow backwards compatibility – see section 12.1.4.
0	1	Version1.0Supported	Shall be 1 if the Device supports CXP v1.0

Therefore Devices able to work using versions 2.0 and 1.1 of the standard shall return 0x00000006.

Also see registers *Revision* and *VersionUsed*.

Comment: The sub-minor version is not coded in this register.

12.3.49 VersionUsed

This register shall provide the version of the CoaXPress standard used to communicate between the Device and Host. The register is set during Discovery (see section 12.1.4).

Bit Offset (LSB << x)	Width (bits)	Name	Description
16	16	MajorVersionUsed	e.g. v1.5 has major version 1, therefore these bits would hold the value 0x0001
0	16	MinorVersionUsed	e.g. v1.5 has minor version 5, therefore these bits would hold the value 0x0005

Therefore if Device and Host are using version 2.1 of the standard, this shall return 0x00020001.

Also see registers *Revision* and *VersionsSupported*.

Comment: ConnectionReset sets the value 0x00010001.

Comment: The sub-minor version is not coded in this register.

12.3.50 LinkSharingStatus

This register shall only be supported if link sharing is supported, as indicated by *CapabilityRegister* bit *LinkSharingSupported*.

This register shall provide status information for Devices supporting Link Sharing (see section 11).

Bit Offset (LSB << x)	Width (bits)	Name	Description
31	1	Master sub-Device	Shall be 1 if this sub-Device is the Master sub-Device. Shall be 0 if this sub-Device is a Slave sub-Device.
All other bits			Shall be 0
2 * n	2	Slave sub-Device #n status (n = 1 ..N)	Shall be 0 if the Slave sub-Device #n is not ready to stream data. Shall be 1 if the Slave sub-Device #n is ready to stream data.
0	2	Master sub-Device status	Shall be 0 if the Master sub-Device is not ready to stream data. Shall be 1 if the Master sub-Device is ready to stream data.

12.3.51 LinkSharingHorizontalStripeCount

This register shall only be supported if link sharing is supported, as indicated by *CapabilityRegister* bit *LinkSharingSupported*.

This register shall provide the number of horizontal stripes that the Device implements (see section 11 on Link Sharing). Optionally the register can be written to set the number of stripes used.

A value of zero means that horizontal striping is not used.

A value of 1 means that the horizontal stripe is the full image size, which is a valid value for frame interleaving or image duplication for redundant systems.

For a value of two or greater, each stripe shall contain the same number of lines. This requirement shall apply even if *LinkSharingHorizontalOverlap* is non-zero.

Comment: Therefore the number of non-overlapped lines plus the number of overlapped lines is the same for all stripes. This means that the top and bottom stripes contain more non-overlapped lines.

12.3.52 LinkSharingVerticalStripeCount

This register shall only be supported if link sharing is supported, as indicated by *CapabilityRegister* bit *LinkSharingSupported*.

This register shall provide the number of vertical stripes that the Device implements (see section 11 on Link Sharing). Optionally the register can be written to set the number of stripes used.

A value of zero means that vertical striping is not used.

A value of 1 is not valid for vertical striping.

For a value of two or greater, each stripe shall contain the same number of pixels. This requirement shall apply even if *LinkSharingVerticalOverlap* is non-zero.

Comment: Therefore the number of non-overlapped pixels plus the number of overlapped pixels is the same for all stripes. This means that the left and right stripes contain more non-overlapped pixels.

12.3.53 LinkSharingHorizontalOverlap

This register shall only be supported if link sharing is supported, as indicated by *CapabilityRegister* bit *LinkSharingSupported*.

This register shall provide the overlap in pixels for horizontal stripes (see section 11 on Link Sharing). Optionally the register can be written to set the overlap.

A value of zero means that there is no overlap.

12.3.54 LinkSharingVerticalOverlap

This register shall only be supported if link sharing is supported, as indicated by *CapabilityRegister* bit *LinkSharingSupported*.

This register shall provide the overlap in pixels for vertical stripes (see section 11 on Link Sharing). Optionally the register can be written to set the overlap.

A value of zero means that there is no overlap.

12.3.55 LinkSharingDuplicateStripe

This register shall only be supported if link sharing is supported, as indicated by *CapabilityRegister* bit *LinkSharingSupported*.

This register shall provide the duplicate count in striped system (see section 11 on Link Sharing).

Optionally the register can be written to set the duplicate count.

A value of zero means that there is no image duplication.

A non-zero value sets the number of duplicate images sent to sub-Devices.

13 GenICam Support

13.1 Introduction

CoaXPress products shall support GenICam. See section 2.1 for details of the GenICam standard.

On the Device side this means an XML file shall be provided with the Device description compatible with the GenApi module of GenICam. The bootstrap registers in the Device provide a means to access the XML file. The features of the Device shall follow GenICam's SFNC if applicable, by using the name and type provided in the GenICam SFNC.

The Device vendor is free to use any suitable register layout in the Device, such as IIDC2 compliant or vendor specific, as long as an XML file is provided describing the layout and the features exposed, and providing the bootstrap registers defined in section 12.3.4 are supported.

To reduce the size of the XML file, a compressed version of the file can be stored in the Device. A compressed file shall use the DEFLATE and STORE methods of the ZIP format (section 2.2 ref 1).

Comment: This is mainly relevant to files stored in the Device, where non-volatile memory space may be limited. Note that the Device does not need to generate or understand the file, only store it.

On the Host side this means implementing a GenTL Producer to allow access to the Device's XML file, provide a control interface and a streaming interface.

The following figure shows how GenICam (and in particular GenTL) terminology, shown in blue, maps to a CoaXPress Device and Host:

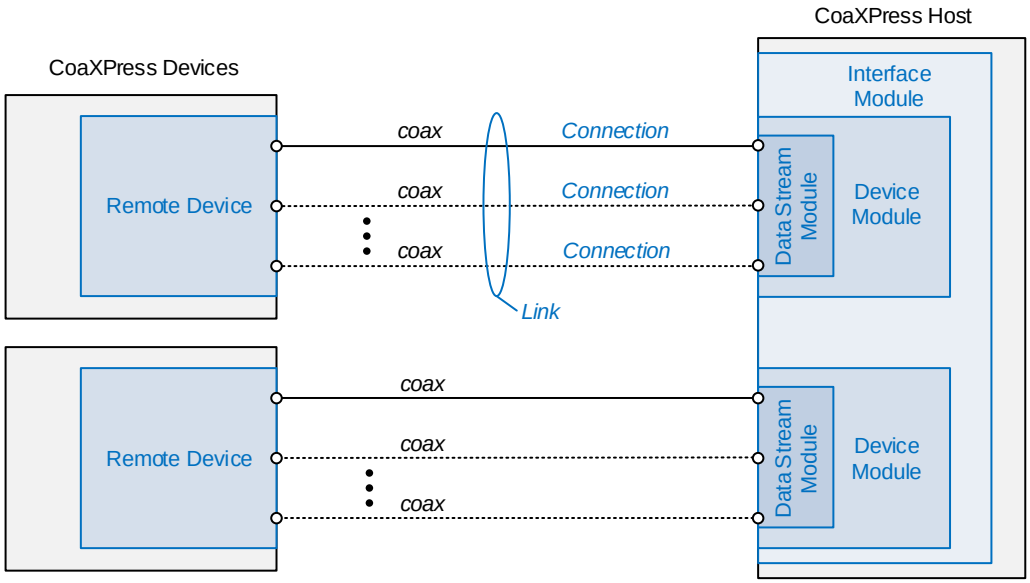


Figure 50 — GenICam Terminology

One of the goals of CoaXPress is to allow a user to buy a CoaXPress camera Device and easily interface it to any CoaXPress compliant Host (i.e. framegrabber and application software). In order to guaranty a

minimal level of interoperability, the CoaXPress standard addresses the use cases of the Device and Host to allow continuous acquisition and display of images.

Comment: The functionality defined in the "Use Case" is only a subset of what a CoaXPress Device can do.

13.2 Device Requirements for GenICam

CoaXPress compliant Devices shall follow the GenICam compliancy rules for the GenICam GenApi standard. These rules also imply the compliancy with the GenICam SFNC/PFNC naming conventions.

CoaXPress v2.1 Devces shall implement the following GenICam versions:

- GenApi V3.3 or higher, plus
- SFNC V2.6 or higher, plus
- GenDC V1.1 or higher (if GenDC supported by the Device)

Comment: At the time of publication, these components are expected to be in the first GenICam release of 2021.

13.2.1 Mandatory Device features

The following GenICam features shall be defined:

- Those needed for the minimal use case (see section 13.2.2).
- The bootstrap registers that are listed with "Support" = "X" in Table 53.
- The CXP-specific Device features listed as recommended in the CXP section of the SFNC.
- General SFNC features listed as mandatory, including TLParamsLocked.
- DeviceScanType for a camera Device.

Note: The feature names and functionality of all these features is defined in the SFNC, and the Device shall comply with the SFNC when implementing the features.

Comment: The Device will in general implement many more features than this mandatory minimum set. It is recommended to comply with the SFNC where possible for all features that are implemented.

13.2.2 Use Case of Device

To support the minimal use case, all CoaXPress camera Devices shall support the features listed in Table 56 in their XML description files, with corresponding registers in the manufacturer-specific space. The table uses the following indications:

Group:	GenICam:	This feature implements a GenICam feature of the same name.
	CXP:	This is a feature defined (or redefined) for the CoaXPress standard.
Access:	R:	Read and/or
	W:	Write.
		Access modes given in parentheses are optional.

Table 56 — Mandatory use case features

Name	Group	Access	Interface
Width	GenICam	R/(W)	IInteger
Height	GenICam	R/(W)	IInteger
AcquisitionMode	GenICam	R/(W)	IEnumeration
AcquisitionStart	GenICam	W/(R)	ICommand
AcquisitionStop	GenICam	W/(R)	ICommand
PixelFormat	GenICam	R/(W)	IEnumeration
DeviceTapGeometry	GenICam	R/(W)	IEnumeration
Image<n>StreamID	CXP	R	IInteger

Note: This list is only intended for camera Devices. Non-camera Devices do not have to support the list.

Comment: Bootstrap registers give the address of these registers to allow the Use Case to be supported by non-GenICam software. See section 12.3.24.

These features are described in the following sections.

13.2.2.1 Width

This feature provides the image width in pixels, as defined by the GenICam SFNC.

13.2.2.2 Height

This feature provides the image height in pixels, as defined by the GenICam SFNC.

13.2.2.3 AcquisitionMode

This feature controls the acquisition mode of the Device, as defined by the GenICam SFNC. For the mandatory use case, only “Continuous” needs to be supported, where images are captured continuously until stopped with an *AcquisitionStop* command.

13.2.2.4 AcquisitionStart

This feature starts acquisition from the Device, as defined by the GenICam SFNC. To allow the use of bootstrap register *AcquisitionStartAddress*, the register controlling this feature shall be single-shot, with the value 0x00000001 written to it to start acquisition.

Comment: If a Device has low level firmware that requires a value other than 0x00000001 to be written to its start register, then the Device will need to implement a virtual register that accepts the value 0x00000001 and then writes the required value to the start register.

13.2.2.5 AcquisitionStop

This feature stops acquisition from the Device, as defined by the GenICam SFNC. To allow the use of bootstrap register *AcquisitionStopAddress*, the register controlling this feature shall be single-shot, with the value 0x00000001 written to it to stop acquisition.

Comment: If a Device has low level firmware that requires a value other than 0x00000001 to be written to its stop register, then the Device will need to implement a virtual register that accepts the value 0x00000001 and then writes the required value to the stop register.

Comment: Note that the SFNC defines that AcquisitionStop should complete transmission of the current image.

13.2.2.6 PixelFormat

This feature provides the pixel format. The 32 bit value is defined by the GenICam PFNC. It is the responsibility of the Device to map this to a valid PixelF code to send over CoaXPress. See section 10.4.1.

13.2.2.7 DeviceTapGeometry

This feature provides the tap geometry. The value is defined by the GenICam SFNC. It is the responsibility of the Device to map this to a valid TapGeometry code to send over CoaXPress.

13.2.2.8 Image<n>StreamID

This gives the Stream ID of the *n*th image stream from the Device. e.g. Image2StreamID gives the ID for the second stream. It is recommended that the primary image stream from the Device (Image1StreamID) has a stream ID of 0 (see section 10.3).

Comment: Multiple stream ID features are part of the use case to allow for display of images from any Device with more than one stream, e.g. a two-tap camera Device.

13.2.3 XML File Location

The XML file can be stored in two types of location, and shall be provided in at least one of them:

- Non-volatile memory in the Device (recommended).
- The Device vendor's website.

The *XmlManifestTable* provides a method to access multiple options of the XML file, both to allow for different locations, and for different versions of the XML file and/or GenApi.

The URL format shall be as specified in the GenTL standard. Example URLs are shown below.

Example: "*Local: MyFilename.zip;B8000;33A?SchemaVersion=1.0.0*" is a ZIP file starting at address 0xB8000 in the Device with a length of 0x33A bytes. The filename is "MyFilename.zip", and the XML file uses GenICam schema v1.0.0.

Comment: The filename is not used here, but may be useful when comparing the file with other versions on the Host system.

Example: "*Web:http://www.mycompany.com/xml/MyFilename.xml*" is a text XML file found at *http://www.mycompany.com/xml*. The filename is "MyFilename.xml".

Note: The Host does not know the length of the string, so it must read it 4 bytes at a time until it sees the NULL terminator.

13.3 Host Requirements for GenICam

CoaXPress compliant Hosts shall follow the GenICam compliancy rules for both the GenICam GenApi and the GenICam GenTL standard. These rules also imply the compliancy with the GenICam SFNC/PFNC and GenTL SFNC naming conventions.

CoaXPress v2.1 Hosts shall implement the following GenICam versions:

- GenApi V3.3 or higher, plus
- SFNC V2.6 or higher, plus
- GenTL V1.6 or higher, plus
- GenTL SFNC V1.2 or higher, plus
- GenDC V1.1 or higher (if GenDC supported by the Host)

Comment: At the time of publication, these components are expected to be in the first GenICam release of 2021.

CoaXPress compliant Hosts shall provide a GenICam GenTL Producer API module.

This GenTL Producer shall implement the acquisition engine to enable image streaming from camera to host, as well as the Port functionality. The Host shall therefore provide a set of XML files to describe its features (system, interface, device, datastream).

Comment: The XML files would typically be embedded into the GenTL Producer.

13.3.1 Mandatory Host features

The following GenICam features shall be defined:

- The CXP-specific Host features listed as recommended in the CXP section of the SFNC.
- The GenTL features listed as mandatory in the GenTL SFNC.

Note: The feature names and functionality of all these features is defined in the SFNC, and the Host shall comply with the SFNC when implementing the features.

Comment: The Host will in general implement many more features than this mandatory minimum set. It is recommended to comply with the SFNC where possible for all features that are implemented.

13.4 Trigger Model for GenICam

The use of LinkTrigger*N* in the trigger packet definitions allows a direct mapping to the GenICam trigger model. As an example, for ExposureStart, TriggerSource could be set to LinkTrigger0.

13.5 GenDC Support

Support for GenDC is optional, controlled by the capability and feature control registers. See section 10.5 for the GenDC implementation in CoaXPress.

Annex A: Detailed Cable Specification

A.1 Introduction

This annex defines detailed cable requirements needed by cable manufacturers to design and verify a cable. It should be read in conjunction with section 6. Whether a given cable will work reliably depends on its DC and AC frequency behavior.

This annex is also needed by a CoaXPress user who chooses to make their own cables. They need to choose a 75 Ω coaxial cable type, cut it to the desired length and terminate it with CXP-connectors. In that way, a large set of coaxial cables and coaxial cable types can be used, including legacy ones. These cables are field-installed and are called **custom-cables**. It shall then be the responsibility of the installer to check whether the system has enough operational margin for the given application.

A.2 Impedance

The cable used in CXP-cables and in the separate lanes of a CXP-multi-cable shall be coaxial with a characteristic impedance of 75 Ω ± 4 Ω. The connectors used in CXP-cables and CXP-multi-cables shall be of the 75 Ω type.

Comment: Impedance errors in connectors are allowed for in the return loss requirements.

When a series coupling of CXP-cables is considered, all of the connectors used have to be of the 75 Ω type, including any inline couplers.

A.3 Return Loss of Cables

A CXP-cable and the separate lanes of a CXP-multi-cable shall have a return loss better than or equal to the one given in the following table:

Annex A: Table 1 — Return loss specification depending on frequency range

Frequency range	Return Loss	Cable Rating
0 - 500 MHz	-20 dB	All cables
500 MHz – 3.2 GHz	-15 dB	Cables rated up to CXP-6 speeds
500 MHz – 6.25 GHz	-15 dB	Cables rated up to CXP-12 speeds

A.4 Cable Length Limitations

The maximum length of a CoaXPress cable is the lowest figure from three different requirements: power supply voltage drop, high speed connection requirements and low speed connection requirements.

A.4.1 Power Supply Voltage Drop

A CXP-cable and the separate lanes of a CXP-multi-cable shall each have a total DC roundtrip resistance of less than 4.98 Ω for each of the coax cables.

Comment: As explained in section 8.3.2.1, there is for the CXP-cable a total voltage drop budget of 3.5V. In the worst case current condition of 703mA (see section 8.3.2.2), this leaves the CXP-cable with an allowed roundtrip resistance of 4.98Ω. This roundtrip resistance includes the connector resistances, the resistances of the core of the coax and of the shield of the coax. In practice, most of this resistance is determined by the core of the cable, however, the shield resistance shall not be neglected.

A.4.2 High Speed Connection Requirements

A CXP-cable and the separate lanes of a CXP-multi-cable that are specified for a given bit rate shall have an attenuation that is less or equal to the attenuation in the following table at its corresponding frequency.

Annex A: Table 2 — Specification of the maximum attenuation for a cable

Bit Rate (Gbps)	Maximum Attenuation (dB)	@ Frequency (GHz)	Belden 1694A (m)
1.250	-21.2	0.6250	*130*
2.500	-26.0	1.2500	*110*
3.125	-26.8	1.5625	*100*
5.000	-20.9	2.5000	*60*
6.250	-15.8	3.1250	*40*
10.000	-20.8	5.000	*40*
12.500	-17.9	6.250	*30*

Comment: As an example, the length of Belden 1694A cable that can be associated with that attenuation level at the given frequency is also shown in this table. Belden 1694A is used as a readily available reference; longer lengths are possible with lower attenuation cables.

Comment: Note that a greater attenuation is allowed at CXP-10 and CXP-12 speeds than at CXP-6. This is because CXP-10 and CXP-12 require a higher performance equalizer than was required at speeds up to CXP-6 in the version 1.x standard.

A series coupling of cables, possibly using inline couplers, shall have an attenuation in total of less than, or equal to, the attenuation in this table.

A.4.3 Low Speed Connection Requirements

A CXP-cable and the separate lanes of a CXP-multi-cable shall have a signal attenuation at 30 MHz of less than, or equal to, -4.74dB.

Comment: The performance of the low speed connection at 20.83˙ Mbps is determined on how well the edges are preserved during transmission. Edge preservation is mainly based on preserving the third harmonic component during transmission. In an 8B/10B coded system at 20 Mbps, the third harmonic is located at 30 MHz. The attenuation of -4.74dB @ 30 MHz equals to the attenuation of a 130m Belden 1694A cable.

Additionally, a CXP-cable for operation at CXP-10 and CXP-12 speeds shall have a signal attenuation at 60 MHz of less than, or equal to, -2.5dB.

A.5 Cable Current Capacity

A CXP-cable and the separate lanes of a CXP-multi-cable shall each be designed to carry 1A in normal operation.

A.6 Cable Attenuation

Comment: The receiver circuit in the Host (HT) includes an equalizer expecting to receive a signal that has suffered from frequency dependent losses that are characteristic for a coaxial cable, with skin-effect losses and possibly also dielectric losses. Therefore the attenuation characteristic of a CXP-cable has to closely resemble that of an ideal coaxial cable having only skin-effect and dielectric losses. This is referred to as "cable-like behavior". Using curve fitting it can be determined whether the cable attenuation characteristic of a possible CXP-cable is close enough to such an ideal cable.

A CXP-cable and the separate lanes of a CXP-multi-cable shall have an attenuation versus frequency characteristic exhibiting cable-like behavior over the frequency ranges indicated in the following table. A series coupling of cables shall also fulfill this requirement as if it is one cable including all of its connectors and inline couplers.

Annex A: Table 3 — Frequency range in which cable-like behavior is to be guaranteed

Cable Rating (Gbps)	Frequency Range from (MHz)	Frequency Range to (GHz)
1.250	1	0.6250
2.500	1	1.2500
3.125	1	1.5625
5.000	1	2.5000
6.250	1	3.1250
10.000	1	5.0000
12.500	1	6.2500

The following formula shall be used for fitting, where *f* is the frequency in Hz:

$$\text{Attenuation (dB)} = -A0 - \text{SQRT}(f / A1) - f / A2 \quad \textit{(formula 1 [5])}$$

This formula has three fitting parameters:

- A0 is linked with DC-attenuation.
- A1 is linked with skin effect losses.
- A2 is linked with possible dielectric losses in the cable.

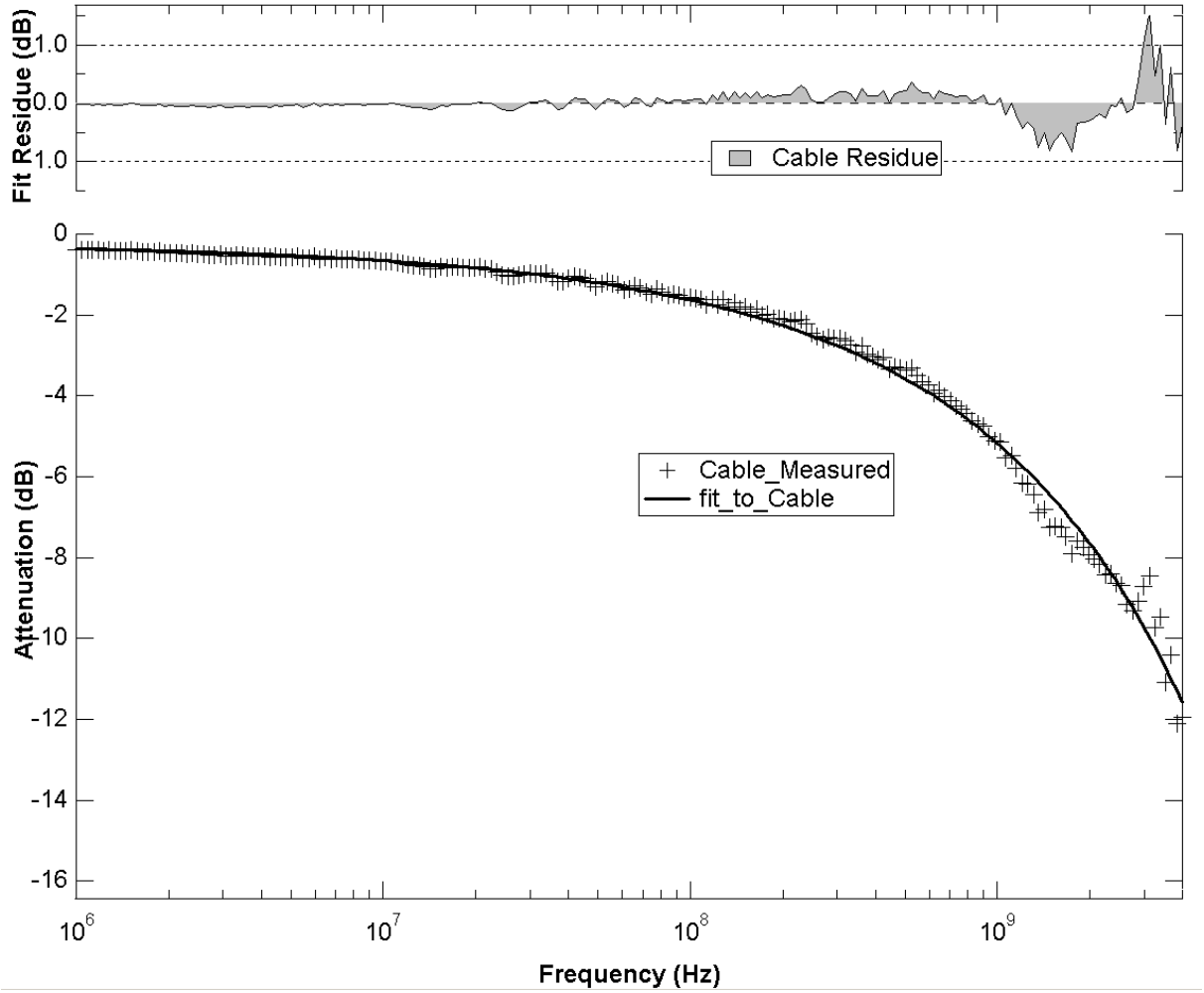
The fitting is based on maximum likelihood estimation, with equal weights for all measurements. During the fitting process, A0, A1, and A2 are free to change.

[5] As an example, for 100 meter Belden 1694A, the coefficients are A0=0.379; A1=2.92x10^6; A2=4.811x10^8 in agreement (+/-0.15 dB) with the manufacturer's data.

For fitting, between 40 and 50 measurement points for each decade shall be used, nominally logarithmically spaced over the entire frequency range.

The maximum difference between the measurement and the fitted attenuation curve shall never be more than 1 dB.

Example: Two 10m cables, coupled together with a 50 Ω (instead of the correct 75 Ω) inline coupler, are measured in the lab from 1 MHz to 3.125GHz:



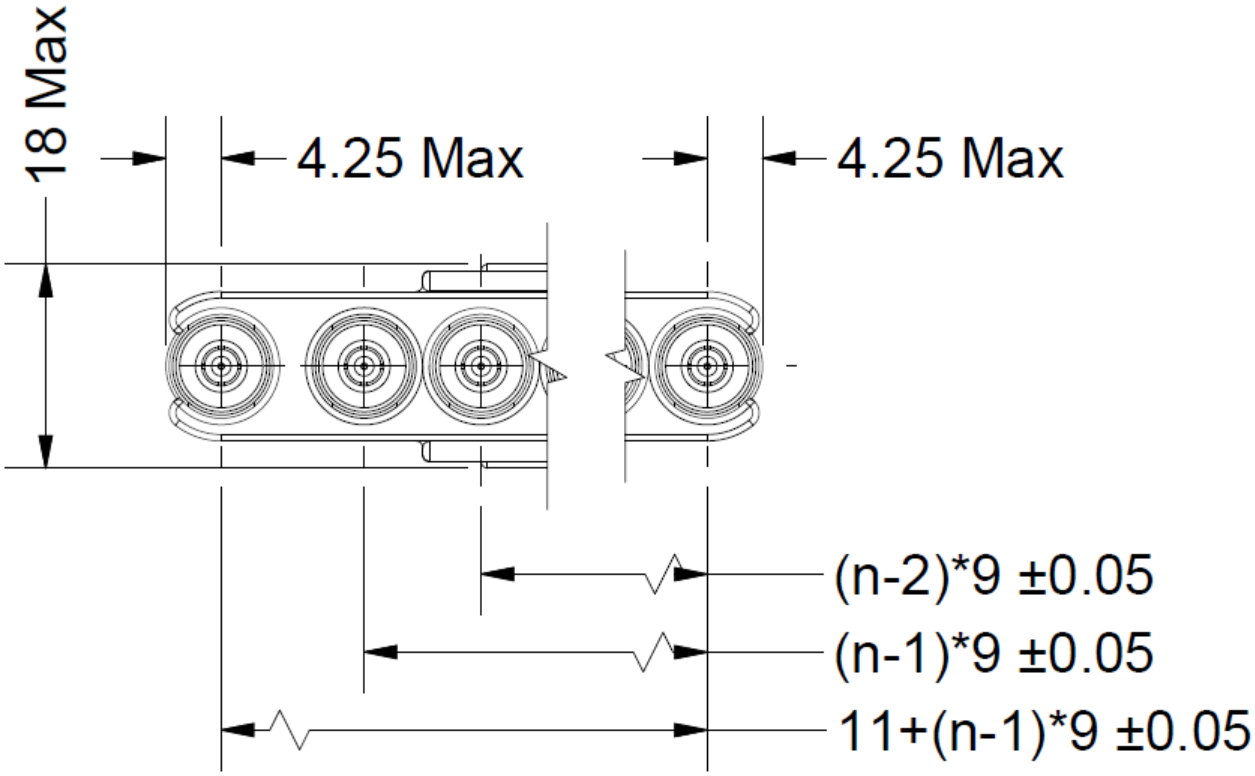
Annex A: Figure 1 — Deviation from normal attenuation behavior versus frequency

This cable combination has a residue of more than the 1 dB between 3 GHz and 3.5 GHz (as a result of the incorrect 50 Ω inline coupler!). Therefore it does not pass at a bit rate of 6.25 Gbps, however despite the 50 Ω connector it does exhibit cable-like behavior at 5 Gbps and lower bit rates.

A.7 Multi-Cable Requirements

A CXP-multi-cable shall have a 1:1 mapping between the connectors at each end of the cable, i.e. the connection order shall not be crossed within the cable.

The following figure shows the rules for the connector on a CXP-multi-cable (dimensions in mm):



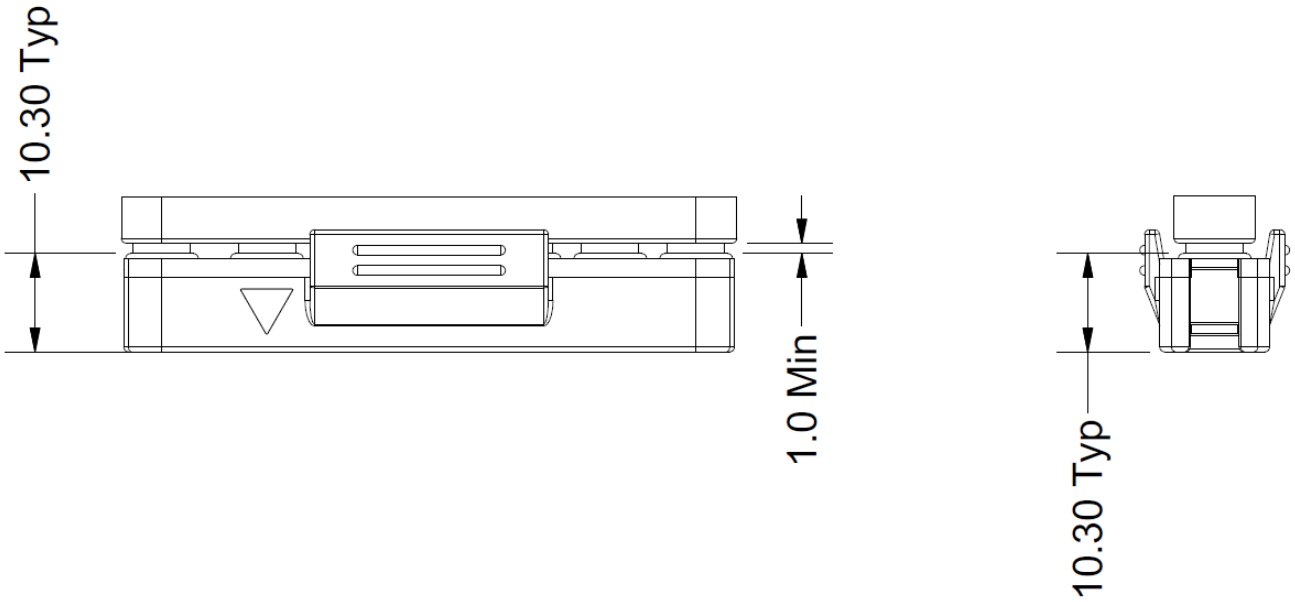
Annex A: Figure 2 — CXP-multi-cable rules

Note that this figure shows an example connector design. Cable vendors can use any design of housing that accepts Micro-BNC or DIN 1.0/2.3 connectors and complies with the dimensions shown.

A CXP-multi-cable shall be marked to indicate the master connection at each end of the cable.

Comment: The arrow shown in the figure below is an example of suitable marking.

A DIN 1.0/2.3 CXP-multi-cable shall have a 1mm (min) slot as shown in the figure below to provide an additional mechanical locking function by preventing the release tab from being moved. This can be a tab supplied with the cable, or for critical applications it allows the Device or Host vendor to supply a bracket that fixes to the Device or Host and engages in the slot. See section 6.2.2.



Annex A: Figure 3 — DIN 1.0/2.3 multi-cable slot

Annex B: Chipset Requirements

B.1 Introduction

This annex defines electrical requirements needed by chipset manufacturers to design and verify new transceivers. It should be read in conjunction with section 7.

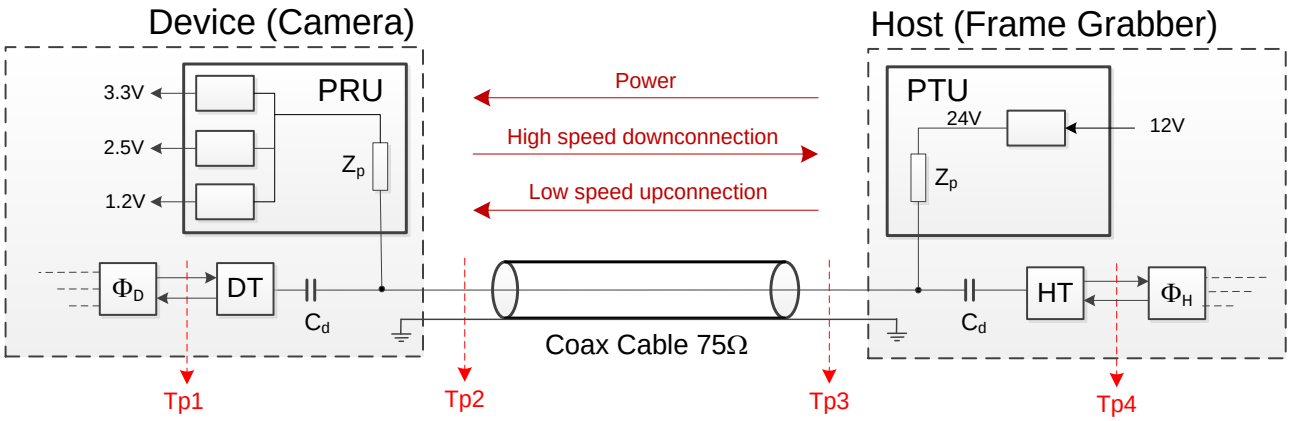
Some of this annex may also be useful to Device and Host manufacturers as part of product compliance testing.

The high speed connection is usually the downconnection from the Device to the Host, and the low speed connection is in the opposite direction, so usually the upconnection from the Host to the Device. In systems using the optional high speed upconnection these are reversed, but for clarity this annex describes the usual case. A high speed upconnection shall implement a Host Transceiver in the Device, and a Device Transceiver in the Host according to this annex.

B.2 Top Level Overview

Annex B: Figure 1 (duplicated from section 7.2 for ease of reference) shows the electrical block diagram of one connection in a CoaXPress system, with the Device on the left connected to the Host on the right by a coax cable with a characteristic impedance of 75 Ω. The figure also defines the test points Tp1 to Tp4 used in this section.

The high speed connection is usually the downconnection from the Device to the Host, and the low speed connection is in the opposite direction, so usually the upconnection from the Host to the Device.



Annex B: Figure 1 — Electrical block diagram

Comment: Voltages shown at the output of the PRU and the input to the PTU are typical examples.

Connected to the Phy Φ_D in the Device there shall be a Device Transceiver (DT) circuit that transmits and receives the bidirectional full duplex data over the coax cable. The Device Transceiver circuit shall split the incoming low-speed data signal and the outbound high speed data signal. The low speed data will

have lost some of its lower frequency components, resulting in baseline wandering, so the Device Transceiver circuit shall compensate for this, restoring the signal to the original digital bit stream at Tp1.

The I/O pin of the Device Transceiver circuit shall be coupled to the coax cable through a capacitor C_d that AC couples the data between this I/O pin and the center contact of the coax connector.

Connected to the Phy Φ_H in the Host there shall be a Host Transceiver (HT) circuit that receives and transmits the bidirectional full duplex data over the coax cable at the Host side. The Host Transceiver circuit shall split the incoming high-speed data signal and the outbound low-speed data signal.

The incoming high-speed data signal may have suffered from frequency dependent attenuation in the coax cable, therefore the Host Transceiver circuit shall also equalize the incoming signal and restore the analog signal to the original digital bit stream, presented at Tp4 to the Phy Φ_H.

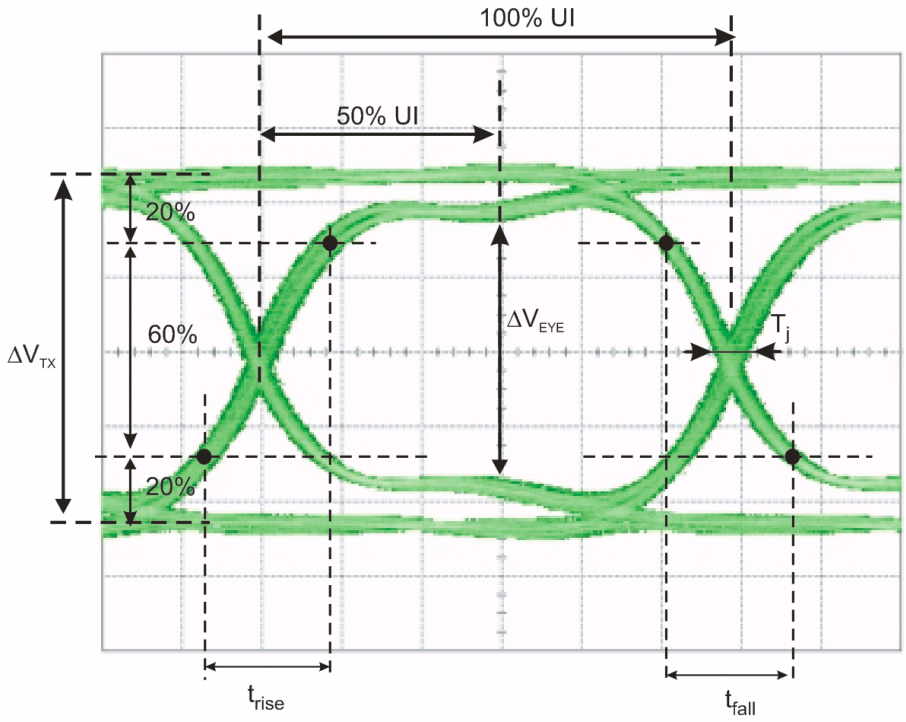
The I/O pin of the Host Transceiver circuit shall be coupled to the coax cable through a capacitor C_d that AC couples the data between this I/O pin and the center contact of the coax connector.

Systems using Power over CoaXPress (PoCXP) shall also implement a Power Transmit Unit (PTU) or Power Receive Unit (PRU), as described in section 8 on PoCXP.

B.3 High Speed Connection

B.3.1 High Speed Connection Cable Driver

The high speed connection Device Transceiver requirements are specified by the following eye diagram at Tp2:



Annex B: Figure 2 — Definition of the parameters in the transmit EYE at Tp2

This eye defines the normative parameters measured at Tp2, at the Device connector when in the test state (see section 9.9) outputting 8B/10B test data. The transmit amplitude, the maximum rise and fall times, and the maximum amount of jitter shall be as indicated in the following table:

Annex B: Table 1 — High speed connection parameters at the transmit (Device) side

Symbol	Parameter	Min.	Typ.	Max.	Units	Notes
ΔV_{TX}	Transmit amplitude	450	600	700	mV	At Tp2 into 75Ω.
		366	480	552	mV	At Tp2 into 50Ω. See note 1 below.
$\Delta V_{EYE}/\Delta V_{TX}$	Relative EYE opening	0.7			-	At Tp2 into 75Ω or 50Ω. See note 3 below.
t_{rise}, t_{fall}	Rise, fall time at CXP-1 to CXP-6		as short as possible	90	ps	At Tp2 into 75Ω or 50Ω, between 20% and 80% of amplitude. See note 3 below.
	Rise, fall time at CXP-10 to CXP-12		as short as possible	45	ps	
T_j	Transmit jitter		as low as possible	20	%UI	At Tp2 into 75Ω or 50Ω. See notes 2 and 3 below.

Comment: *Jitter is defined with respect to the Unit interval, resulting in relaxed normative absolute parameters for lower speed Devices.*

Note 1: *This parameter is quoted at 50 Ω to allow practical measurement with high speed oscilloscopes which will have a 50 Ω input. The remaining parameters are quoted at 75 Ω because these can be measured using a high-speed converter. The 50 Ω figure assumes that the driver has an internal 75 Ω series terminator.*

Note 2: *The jitter figure T_j is a system figure comprising the combination of jitter from both the Phy Φ_D and the Device Transceiver (DT). To allow practical implementation of the Phy Φ_D, the jitter contribution from the Device Transceiver should be minimized.*

Note 3: *The best method of testing these parameters is specified in the Compliance Test Specification document (section 2.2 ref 6). That document will determine whether to test into 50 Ω or into 75 Ω, and decide on all the specific elements in the test set-up. This may even be dependent on the connector type.*

The high speed connection Device Transceiver shall not apply pre-emphasis or de-emphasis.

Comment: *Pre-emphasis would make it more difficult for the cable receiver to do its job, because the combination of pre-emphasis and cable attenuation may result in an attenuation / frequency plot that does not have cable-like behavior. See section A.6.*

B.3.2 High Speed Connection Cable Receiver

At the receive side, the Host shall be able to receive the 8B/10B signals from a compliant Device over a length of the Belden 1694A coaxial cable having the attenuation given in the following table at the indicated frequency for the bit rates supported by the Host:

Annex B: Table 2 — High speed connection parameters at the receiver (Host) side

Bit Rate (Gbps)	Attenuation (dB)	@ Frequency (GHz)	Equivalent Belden 1694A (m)	Conditions
1.250	-22.0	0.625	135 m	8B/10B coding, full ΔV_{TX} range, power transmission & low speed connection active, BER $<10^{-12}$
2.500	-27.2	1.250	115 m	
3.125	-28.1	1.5625	105 m	
5.000	-22.6	2.500	65 m	
6.250	-17.8	3.125	45 m	
10.000	-23.4	5.000	45 m	
12.500	-20.9	6.250	35 m	

Comment: Belden 1694A is used as a readily available reference; longer lengths are possible with lower attenuation cables.

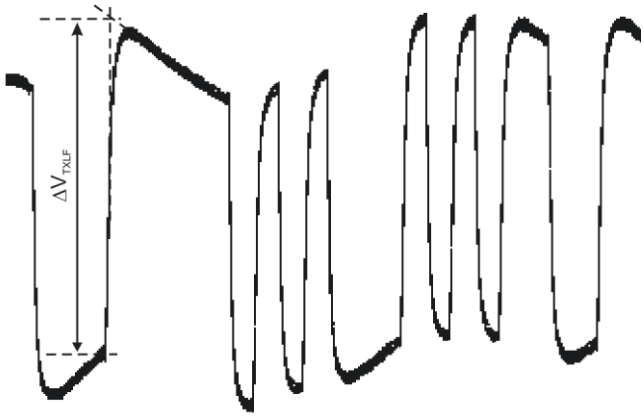
Comment: Note that a greater attenuation is allowed at CXP-10 and CXP-12 speeds than at CXP-6. This is because CXP-10 and CXP-12 require a higher performance equalizer than was required at speeds up to CXP-6 in the version 1.x standard.

Adaptive equalization shall restore the original bit-stream, followed by the lock of the clock and data recovery in the Phy Φ_H. When a Host is able to receive at different bit rates, for each of these bit rates, it shall be able to receive through a length of Belden 1694A cable having the attenuation given this table at the indicated frequency.

B.4 Low Speed Connection

B.4.1 Low Speed Connection Baseline Wandering

Due to the high-pass filtering induced by Z_p at the Host, the signal measurable at Tp3 will show baseline wandering as shown in the following figure:



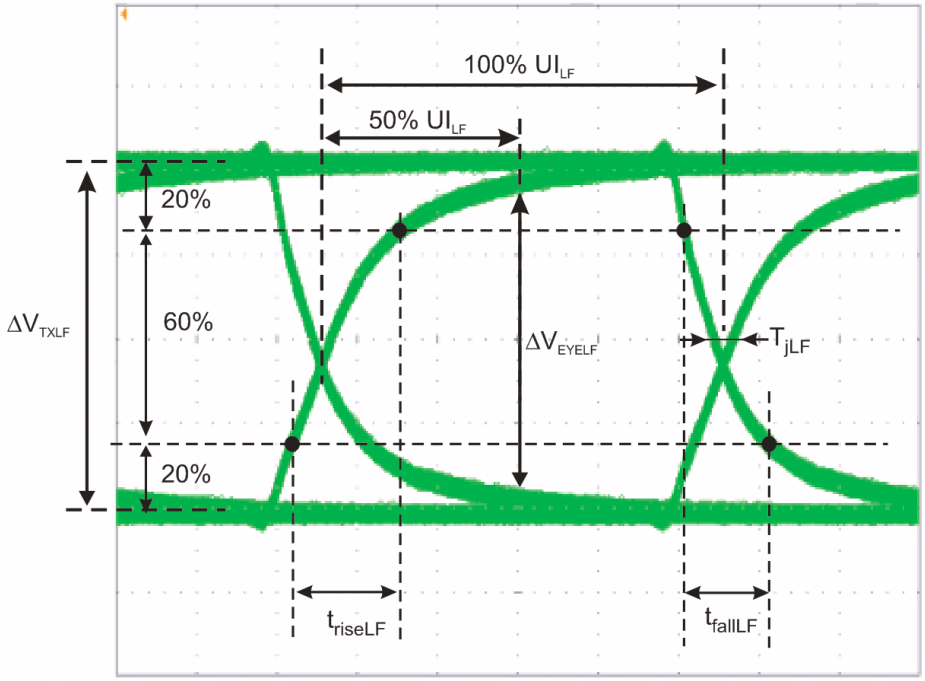
Annex B: Figure 3 — Signal transmitted at Tp3 by the Host showing baseline wander

This baseline wandering shall be taken care of in the Device Transceiver circuit.

Comment: By fixing the inductive part of Z_p to 11.5 µH (see section 7.5), base wander shall be present in a predictable way. In principle this base wander can be diminished by using an inductor of 40 µH or more. However, inductors of that value (and that also resist higher current levels), are physically large components! The 11.5 µH value is a nice compromise, leading to a physically small inductor, with a baseline wander that can still be restored by the receiver in the Device Transceiver circuit. Using a fixed value also makes interoperability more easily achievable since a known baseline wander can easily be restored.

B.4.2 Low Speed Connection Cable Driver

The low speed connection cable driver requirements (part of the Host Transceiver) are specified by the following eye diagram at Tp3:



Annex B: Figure 4 — Definition of the parameters in the low speed transmit EYE at Tp3

This eye defines the normative parameters measured at Tp3, at the Host connector when outputting 8B/10B test data. The transmit amplitude, the minimum and maximum rise and fall times, and the maximum amount of jitter are indicated in the following table:

Annex B: Table 3 — Low speed connection parameters at the transmit (Host) side

Symbol	Parameter	Min.	Typ.	Max.	Units	Notes
ΔV_{TXLF}	Transmit amplitude	90	130	180	mV	At Tp3 into 75Ω.
$\Delta V_{EYELF}/\Delta V_{TXLF}$	Relative EYE opening	0.75	-	-	-	At Tp3 into 75Ω.
t_{riseLF}, t_{fallLF}	Rise, fall time at 20.83˙ Mbps	3.5	-	20	ns	At Tp3 into 75Ω, between 20% and 80% of amplitude.
	Rise, fall time at 41.6˙ Mbps	3.5	-	12	ns	
T_{jLF}	Transmit jitter 20.83˙ Mbps	-	as low as possible	5	ns	At Tp3 into 75Ω.
	Transmit jitter 41.6˙ Mbps	-	as low as possible	2.5	ns	See note 4 below.

Comment: These parameters are quoted at 75 Ω because at this speed it is easy to make measurements with a 75 Ω terminator and a high impedance oscilloscope input.

Comment: The definition of the minimum rise and fall times are important, since unneeded steepness must be avoided to achieve maximum crosstalk tolerance with the high speed connection data.

Note 1: *The Power Transmitting Unit (PTU) may be present in the Host, and connected through Z_p. During this measurement it is however not required to use the Power Transmitting Unit for providing power to the Device.*

Note 2: *To verify the correct eye pattern, signal processing is needed, either in the oscilloscope or off-line, to equalize the low-frequency loss due to the impedance L_p, or Z_p must be disconnected.*

Note 3: *For clarity the eye diagram is shown with no simultaneous flow of high speed connection data since that would corrupt the EYE parameters severely.*

Note 4: *The jitter figure T_{jLF} is a system figure comprising the combination of jitter from both the Phy Φ_H and the Host Transceiver (HT). To allow practical implementation of the Phy Φ_H, the jitter contribution from the Host Transceiver should be minimized.*

B.4.3 Low Speed Connection Cable Receiver

At the receive side, the Device shall be able to receive the 8B/10B signals from a compliant Host over a minimum length of the well-specified Belden 1694A coaxial cable.

Annex B: Table 4 — Low speed connection parameters at the receive (Device) side

Bit Rate (Mbps)	Attenuation (dB)	@ Frequency (MHz)	Equivalent Belden 1694A (m)	Conditions
20.83•	-4.9	30	135 m	8B/10B coding, full ΔV_{TXLF} range power transmission & high speed connection active, BER < 10^{-12}
41.6•	-2.6	60	45 m	

The low speed upconnection signal from the receiver shall be stable within 10ms of the corresponding high speed downconnection being enabled, or changing bitrate.

Comment: *This requirement is to allow for the usage of clock and data recovery (CDR) logic in the receiver.*

B.5 Guidance Notes

Given that good PCB layout and component choice is critical to achieve CoaXPress performance (see section 7.8), and for many firms CoaXPress will be the highest speed design they have implemented, it is recommended that chipset manufacturers provide:

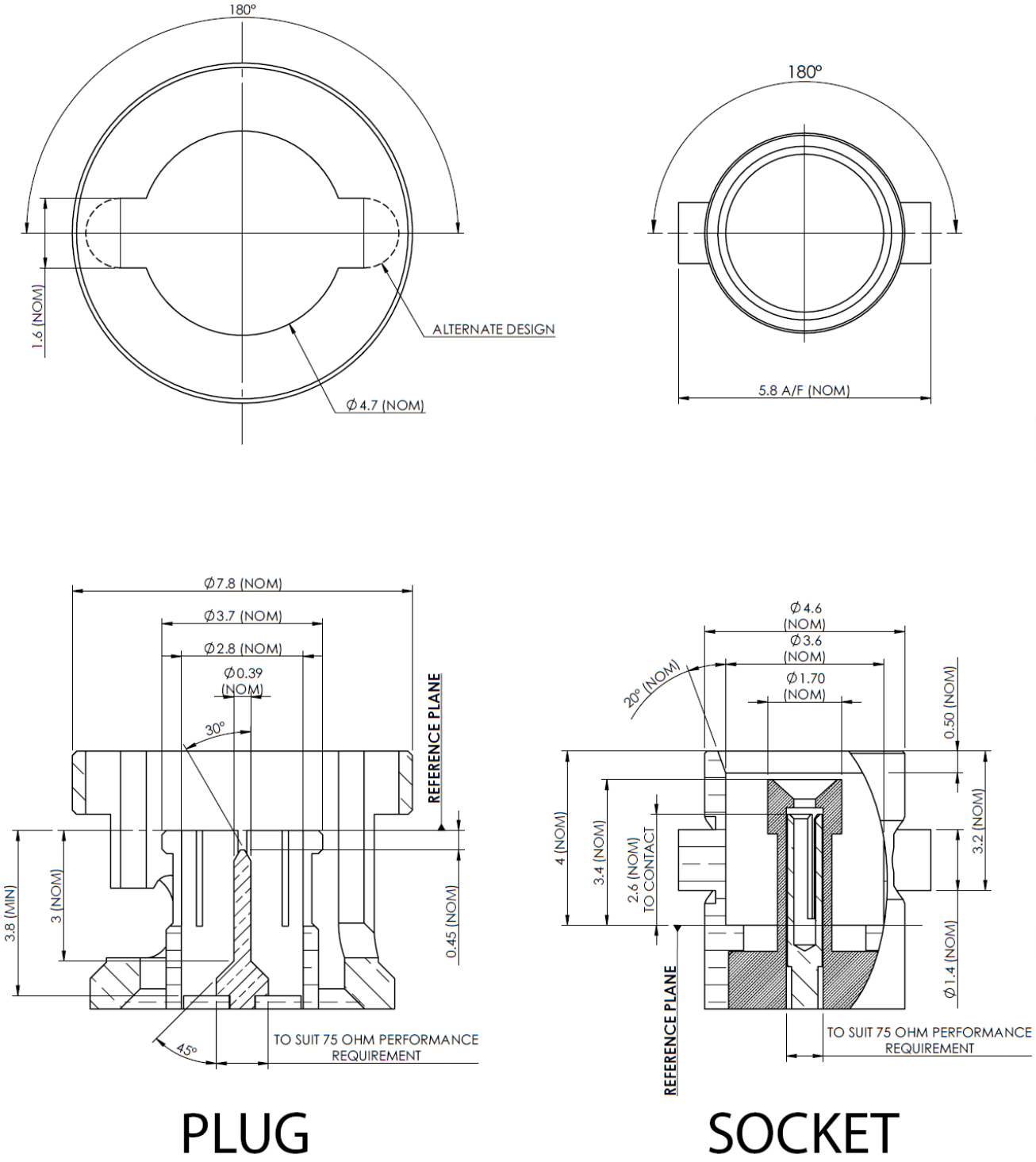
- Sample PCB artwork.
- Detailed layout guidance.
- Component choice recommendations.

Annex C: Micro-BNC

C.1 Introduction

The Micro-BNC connector (also known as HD-BNC) is a de-facto standard, originated by Amphenol®. Therefore the following drawing gives typical dimensions of Micro-BNC connectors. It is intended to be sufficient to identify compatible connectors, and does not fully define the connector.

C.2 Drawing



Annex C: Figure 1 — Nominal Micro-BNC Connector Typical Dimensions

(dimensions in mm)

Annex D: Descriptive Clause

D.1 Background of Standardization

D.1.1 Background of the Original CoaXPress Technology

The main technologies behind CoaXPress were developed as a part of MEDEA+ projects ASIC-CCD (2A206) and TritonZ (2A207). The MEDEA+ program is a EUREKA cluster focusing on micro and nanoelectronics, which are key enabling technologies for electronics and Information and Communications Technology (ICT).

The MEDEA+ label and the funding of the R&D by the local authorities in Belgium and the Netherlands were key for making what today is called CoaXPress.

A group of companies, the CoaXPress Consortium, developed the technology and initial standard for CoaXPress.

D.1.2 Handover from the CoaXPress Consortium to JIIA

In June 2009, the CoaXPress Consortium proposed that JIIA became the organization for the Global Standardization of a high-speed interface using a coaxial 75 Ω cable named “CoaXPress” and developed by the Consortium, and then in September 2009 JIIA accepted this proposal. In December 2009 the CoaXPress Consortium edition of the standard was handed over to JIIA, leading to the CoaXPress Standard version 1.0 being released by the JIIA CoaXPress Working Group in December 2010.

In 2007, AIA (formerly Automated Imaging Association) based in the North America, EMVA (European Machine Vision Association) based in Europe and JIIA had created the “Cooperation Agreement on Global Coordination of Machine Vision Standardization” (the “G3”) in order to avoid the duplication of similar standards development. The JIIA CoaXPress standard was accepted by the G3 as a world standard.

D.2 Version Changes

D.2.1 v1.0 to v1.1

Section numbers listed refer to the v1.1 standard.

v1.1 adds support for CXP-multi-connectors and CXP-multi-cables based on the DIN 1.0/2.3 connector (see sections 5.2 and A.7). It also allows for a high speed upconnection to allow very fast data transfer and triggers to a Device (see section 10.1, in particular section 10.1.4). These additions are part of the roadmap to allow CoaXPress to remain at the leading edge of performance. v1.1 is English-only; a Japanese translation may be released in future.

Note that v1.1 Hosts additionally need to support some v1.0 features to allow operation with v1.0 Devices, and v1.1 Devices additionally need to support some v1.0 features to allow operation with v1.0 Hosts.

v1.1 also includes the following key changes:

- One coax cable – a “Link” – renamed to “Connection”. A master connection and extension connections now form a “Link”. This is for consistency with the GenICam SFNC, and results in the renaming of some bootstrap registers. See section 4.1.
- Some changes to the connector indicator lamp indications to improve clarity. See section 5.4.
- Slightly relaxed physical layer parameters (identical to the Specification Change Notice for v1.0 published by JIIA in document JIIA CXPR-004-2012). See section 6 and Annex B.
- Additional requirements for PoCXP in Devices: A startup/brownout current limit is defined; a discharge time for C_s is defined; clarification of minimum load power requirements for multi-connection Devices. See sections 7.3.2.2 and 7.3.3.2.
- A simplified packet priority scheme. See section 8.2.4.
- GPIO has been removed to allow a future version of CoaXPress to implement an event packet using K28.0. Note that this event packet could include GPIO as one event type. See section 8.3.1.
- Tentative acknowledgements have been replaced by a wait acknowledgement, which allows the Host to stay in control of the command. See section 8.6.1.
- Improvements to the link test system: New counters are required to show the number of packets sent and received, including new bootstrap registers in the Device. See section 8.7.
- The mapping of the GenICam PixelFormat feature to the CoaXPress PixelF value is now fully defined. See sections 9.4.1.1 and 11.2.1.6.
- Rewording of the discovery process, with additional requirements defined for bootstrap register *LinkConfigDefault* (now called *ConnectionConfigDefault*) for both the Host and Device, and a changed value of *LinkConfig* (now called *ConnectionConfig*) following *LinkReset* (now called *ConnectionReset*). See sections 10.1, 10.3.28 and 10.3.34.
- Additional Device memory space access requirements defined. See section 10.3.

- Additional bootstrap registers defined to allow support of the basic use case without the need for GenICam. See sections 10.3.19 to 10.3.27. This also results in new requirements for the features AcquisitionStart and AcquisitionStop. See sections 11.2.1.4 and 11.2.1.5.
- Additional optional bootstrap register to more easily support electrical compliance test for a Device. See section 10.3.40.
- Much of the GenICam sections have been deleted, as these the requirements are now fully defined in the GenICam (in particular GenTL) standard.
- GenICam requirements merged into one section, section 11.

v1.1 also includes detail wording changes to clarify the meaning of the standard, in particular:

- Improved wording of CRC requirements. See section 8.2.2.2.
- StreamPacketSizeMax (formerly StreamPacketDataSize) and ControlPacketSizeMax (formerly ControlPacketDataSize) now clearly stated to be in units of bytes, and to apply to the entire packet size. See sections 10.1.5, 10.3.31 and 10.3.32.
- Bootstrap registers specified using the “lsb << x” notation, for consistency with GenCP.
- Improved wording of the impedance requirements for CXP-cables. See section A.2.

D.2.2 v1.1 to v1.1.1

Section numbers listed refer to the v1.1.1 standard.

v1.1.1 has a limited number of changes to clarify some points in the v1.1 standard. The main clarifications are:

- Clarification that the Packet Tag and connection order should only be reset as part of a *ConnectionReset* or a write to *ConnectionConfig*. See sections 8.5.3, 8.5.5 and 10.3.33.
- In the Control Transactions section, the rules cross-reference the existing requirement in section 10.3.3 not to use Wait Acknowledgements for bootstrap registers, plus the wording has been improved to make it clearer that only one final acknowledgement should be sent. See section 8.6.1.1.
- The Pixel Types section has been reworded following changes in the PFNC. Devices are no longer required to use the “pmsb” suffix for packed pixel formats, which many vendors had ignored anyway. See section 9.4.1.
- The Discovery section clarifies the point at which a Host is expected to enable extension connections. See sections 10.1.3 and 10.1.6.
- The Discovery section also shows revised versions of Figure 37 and Figure 38 which show a Device with more than two connections, to clarify that *DeviceConnectionID* can be greater than 1 (as already defined by section 10.3.29).
- The bootstrap registers that should be listed in the XML file are now indicated. See section 10.3.4.

D.2.3 v1.1.1 to v2.0

Section numbers listed refer to the v2.0 standard.

v2.0 adds support for CXP-10 and CXP-12 speeds, with the corresponding 41.6˙ Mbps upconnection speed, and several protocol enhancements.

Key changes:

- The high speed upconnection has been temporarily removed to allow an improved discovery protocol for it to be added in v2.1. See section 4.1.
- Unified time stamping has been added. See sections 4.11 and 9.7.
- Micro-BNC connector added. See section 6.2.1.2 and Annex C.
- New speeds CXP-10 and CXP-12 added. See section 7, Annex A and Annex B.
- New 41.6˙ Mbps low speed connection speed added for use at CXP-10 and CXP-12. See section 7.7.
- New edge rate and over-current requirements added for PoCXP Hosts. See sections 8.4.2.3 and 8.4.3.
- Additional error reporting features. See section 9.2.2.3.
- Rising and falling trigger types replaced by LinkTriggerN. For the low speed connection packet this is only a change of name. For the high speed connection packet the LinkTriggerN word has been added. See sections 9.3.1.1 and 9.3.1.2.
- Control channel reset deprecated.
- New control and control acknowledgement packets added with a tag field to improve robustness. See sections 9.6.1.2, 9.6.2.2 and 9.6.3.2.
- New heartbeat packet added. See section 9.7.
- New event packet added. See section 9.8.
- New optional link sharing section 11 added.
- Improved version detection in discovery. See section 12.1.4.
- Revised loss-of-lock wording to support the new 41.6˙ Mbps low speed connection speed. See section 12.2.
- Additional bootstrap registers to support the additional features in v2.0. These include a *CapabiityRegister* replacing *HsUpConnection*, and a corresponding *FeatureControlRegister* to manage optional features. See section 12.3.4 and sections 12.3.41 to 12.3.50.
- Additional *ConnectionReset* and *ConnectionConfig* requirements to support the additional features in v2.0. See sections 12.3.28 and 12.3.33.
- Revised wording of *ConnectionConfigDefault* based around the GenICam User Set mechanism. See section 12.3.34.

Detail changes:

- Detail wording improvements throughout the document.
- Labeling and logo requirements have been taken out of the introduction into a new section 5. This allows the Introduction to have no mandatory requirements listed.
- The description of the lamp timings in section 6.4 have been revised for clarity, but there is no functional difference.
- Added a note that it is planned to deprecate the Arbitrary Image stream in the next release. See section 10.4.7.

D.2.4 v2.0 to v2.1

Section numbers listed refer to the v2.1 standard.

v2.1 adds support for GenDC, as well as reinstating support for a high speed upconnection. There are also several clarifications.

Key changes:

- Arbitrary image streams deprecated, as pre-announced in v2.0.
- GenDC support added, with new section 10.5, new variant of the low speed connection trigger packet in Table 16, additional bootstrap registers in sections 12.3.19 to 12.3.23, and additional *CapabilityRegister* bits in section 12.3.46 and *FeatureControlRegister* bits in section 12.3.47.
- Support for the high speed upconnection restored. See section 12.1.5.
- Wait acknowledge now allowed for a write to a GenICam group bootstrap register. See section 12.3.3.

Detail changes:

- “CoaXPress Specification” changed to “CoaXPress Standard” in the document, to match the official JIIA title.
- Additional recommendations for connector indicator lamps and typo corrected. See section 6.4.
- Correction to the sense current mentioned in the comment in section 8.3.3.1.
- Additional comment regarding the 7V PoCXP voltage sense limit. See section 8.4.4.
- Rewording of section 9.2.4 to clarify the packet priority.
- Clarifications to Table 19 and Table 20.
- Clarifications to the IDLE usage in sections 9.2.5.1 and 9.2.5.2.
- Clarification to packet transfer over multiple connections. See section 9.5.5 and Table 22.
- Clarification to event packet transaction rules. See section 9.8.1.1.
- Byte/word size correction in the comment under Table 29, plus usage of words vs bytes made consistent in all of section 9.8.2.
- Slight restructuring of section 10 as part of adding GenDC, including adding Table 32 to section 10.2 to clearly list all data stream types.

- Table 34 updated to align with the latest PFNC convention.
- Section 10.4.1.8 renamed to include YCbCr in the name as well as YUV, again to align with the PFNC.
- Clarification to the use of the SourceTag for Line Scan Devices in Table 47.
- Clarification to the 200ms wait during bit rate changes. See section 12.1.7.
- Clarifications to the loss of lock detection. See section 12.2.
- Clarification to *ConnectionReset* and *ElectricalComplianceTest*. See section 12.3.33.
- This section D.2 updated to always refer to section numbers corresponding to the document being described rather than the equivalent section number in the latest standard.
- Plus other minor wording improvements throughout the document.

D.3 Members Involved

D.3.1 The Main Authors of the Original CoaXPress Standard

The main authors of the original CoaXPress standard were:

- Active Silicon Ltd. — Chris Beynon
- Adimec Advanced Image Systems B.V. — René Weltje
- EqcoLogic N.V. — Maarten Kuijk

In addition, the following people contributed to original CoaXPress standard:

- Active Silicon Ltd. — Martin Bone, Colin Pearce
- Adimec Advanced Image Systems B.V. — Jochem Herrmann, Tjeerd Veenstra, Frederik Voncken
- Components Express Inc. — Steve Mott
- EqcoLogic N.V. — Bram Devuyst

D.3.2 Contributors for v2.0

The v2.0 standard was edited by Chris Beynon of Active Silicon Ltd.

The following people have contributed to the v2.0 standard:

- Active Silicon Ltd. — Chris Beynon, Matthew Bridges
- Adimec Advanced Image Systems B.V. — Frederik Voncken
- AVAL DATA Corporation — Masahide Matsubara
- Basler AG — Frank Krehl
- Bitflow, Inc. — Dan Frisch
- Coax Connectors Ltd. — David Mills
- EASii IC SAS — Benjamin Pussacq
- e2v — Pascal Pellet
- Euresys s.a. — Denis Conard, Paulo Possa, Jean-Michel Wintgens
- MACOM Technology Solutions Holdings, Inc. — Wim Cops
- Matrox Electronic Systems Ltd. — Eric Boisvert, Jonathan Lavergne
- Microchip Technology Inc. — Maarten Kuijk
- Microteam Oy — Tero Kiiskinen
- Mikrotron GmbH — Andreas Ertl
- Optronis GmbH — Sébastien Gendreau
- Samtec, Inc. — Dylan Lang
- Sensor to Image GmbH — Werner Feith
- Silicon Software GmbH — Markus Grebing, Björn Röger
- TM-Research Inc. — Masaru Naganuma

D.3.3 Contributors for v2.1

The v2.1 standard was edited by Chris Beynon of Active Silicon Ltd.

The following people have contributed to the v2.1 standard:

- Active Silicon Ltd. — Chris Beynon
- Adimec Advanced Image Systems B.V. — Frederik Voncken
- AVAL DATA Corporation — Masahide Matsubara
- Basler AG — Frank Krehl

• EASii IC SAS	Benjamin Pussacq
• Euresys s.a.	Paulo Possa
• Matrox Electronic Systems Ltd.	Eric Boisvert, Jonathan Lavergne, Stephane Maurice
• Sensor to Image GmbH	Werner Feith
• Silicon Software GmbH	Björn Röger
• TM-Research Inc.	Masaru Naganuma

This page intentionally blank

The standards which the Japan Industrial Imaging Association publishes are enacted whether it infringes or not, on industrial property such as patents and new utility designs and so on.

一般社団法人日本インダストリアルイメージング協会が発行している規格類は，工業所有権（特許，実用新案など）に関する抵触の有無に関係なく制定されています。

The Japan Industrial Imaging Association is not responsible for any industrial property rights related to the contents of this standard.

一般社団法人日本インダストリアルイメージング協会は，この規格類の内容に関する工業所有権に対して，一切の責任を負いません。

JIIA CXP-001-2021

Published on Jan 13, 2021

2021 年 01 月 13 日発行

Published by

Japan Industrial Imaging Association

2-21-27, Hyakunin-cho, Shinjyuku-ku, Tokyo 169-0073 Japan

TEL/FAX: +81-3-3361-6880

発行

一般社団法人 日本インダストリアルイメージング協会

〒169-0073 東京都新宿区百人町 2-21-27

TEL/FAX：03-3361-6880