Japan Industrial Imaging Association Standard JIIA CP-001-2011 /
1394 Trade Association Specification TS2011001

IIDC2 Digital Camera Control Specification Ver.1.0.0
January 26th, 2012

Sponsored by:

Japan Industrial Imaging Association (JIIA) / 1394 Trade Association (1394TA)

Accepted for publication by

This specification has been accepted by the JIIA and 1394TA Board of Directors.

Abstract

The purpose of this document is to act as a design guide for imaging device / host makers that wish to use a digital interface as the device-to-host interconnect. Adherence to the design specifications contained herein do not guarantee, but will promote interoperability for this class of device. The device registers, fields within those registers, video formats, modes of operation and controls for each are specified. Area has been left for growth. To make application for additional specification, contact the Japan Industrial Imaging Association Next Generation Camera Protocol Working Group or the 1394 Trade Association Industrial and Instrumentation Working Group. IIDC2 is designed for many kinds of digital interfaces.

JIIA

Japan Industrial Imaging Association
The Standardization Committee –
Next Generation Camera Protocol Working Group

This document is offered “as is” in this current version.

JIIA, its members, subsidiary companies of its member nor the affiliates of its members, do not guarantee any impact on trade, fitness for specific purpose, nor infraction of property rights.

JIIA, its members, subsidiary companies of its member nor the affiliates of the its members, do not take any responsibility for damage originating from the use of this text, or unavailability of this text (not limited to the passive damage and other derivative, incidental damage.) under applicable law.

This counts also if the damage is related to JIIA, its members, subsidiary companies of its member nor the affiliates of the its members.

JIIA, its members, subsidiary companies of its member nor the affiliates of the its members do not take the responsibility for defending not to compensate for potential disputes between JIIA and third party which may originate from intellectual property in this text.

The standard 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.

The Japan Industrial Imaging Association is not responsible for any industrial property rights related to the contents of this standard.

Information contained in this standard is subject to the Copyright Act of Japan, and the copyright protection controlled by international convention. When reprinting whole or a part of this standard, the permission from the publisher (JIIA) shall be obtained, except when it is for private use and or when it is met the Copyright Act.

Japan Industrial Imaging Association
2-10-15 Nakameguro, Yamate Ave. K Bldg.,
Meguro Tokyo 153-0061 Japan

1394 Trade Association Specification

1394 Trade Association Specifications are developed within Working Groups of the 1394 Trade Association, a non-profit industry association devoted to the promotion of and growth of the market for IEEE 1394-compliant products. Participants in Working Groups serve voluntarily and without compensation from the Trade Association. Most participants represent member organizations of the 1394 Trade Association. The specifications developed within the working groups represent a consensus of the expertise represented by the participants.

Use of a 1394 Trade Association Specification is wholly voluntary. The existence of a 1394 Trade Association Specification is not meant to imply that there are not other ways to produce, test, measure, purchase, market or provide other goods and services related to the scope of the 1394 Trade Association Specification. Furthermore, the viewpoint expressed at the time a specification is accepted and issued is subject to change brought about through developments in the state of the art and comments received from users of the specification. Users are cautioned to check to determine that they have the latest revision of any 1394 Trade Association Specification.

Comments for revision of 1394 Trade Association Specifications are welcome from any interested party, regardless of membership affiliation with the 1394 Trade Association. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally, questions may arise about the meaning of specifications in relationship to specific applications. When the need for interpretations is brought to the attention of the 1394 Trade Association, the Association will initiate action to prepare appropriate responses.

Comments on specifications and requests for interpretations should be addressed to:

Editor, 1394 Trade Association
315 Lincoln, Suite E
Mukilteo, WA 98275 USA

1394 Trade Association Specifications are adopted by the 1394 Trade Association without regard to patents which may exist on articles, materials or processes or to other proprietary intellectual property which may exist within a specification. Adoption of a specification by the 1394 Trade Association does not assume any liability to any patent owner or any obligation whatsoever to those parties who rely on the specification documents. Readers of this document are advised to make an independent determination regarding the existence of intellectual property rights, which may be infringed by conformance to this specification.

Contents

Tables

Figures

Annexes

No table of contents entries found.

Foreword

(This foreword is not part of JIIA Standard JIIA CP-001-2011 / 1394TA Specification TS2011001)

This specification defines IIDC2 Digital Camera Control Specification Ver.1.0.0 for industrial use. This document describes control and status registers definition and its control procedures.

The 1st version of IIDC (before IIDC Ver.1.32) attributes to 1394 Trade Association (hereinafter called 1394TA). The 2nd version of IIDC (IIDC2) attributes to Japan Industrial Imaging Association (hereinafter called JIIA), and be copyrighted by JIIA and 1394TA.

This specification was accepted by the Board of Directors of the JIIA and 1394TA. Board of Directors acceptance of this specification does not necessarily imply that all board members voted for acceptance. At the time it accepted this specification, the JIIA. Board of Directors had the following members:

BOD of JIIA

Post	*Organization Represented*	*Name of Representative*
Chair	Toshiba Teli	Shigeo Oka
Vice-Chair	GRAPHIN	Tomoaki Kurosawa
Vice-Chair	Hitachi Kokusai Electric	Kazuki Suyama
Vice-Chair	CCS	Shigeki Masumura
Secretary	Symco	Sachio Kiura
BOD	FAST	Shioji Kodama
BOD	Optart	Kazuo Fujiyama
BOD	Optronics	Naoki Ueno
BOD	Advanced Communication Media	Norichika Yui
BOD	ADSTEC	Hiroshi Yako
BOD	AVAL DATA	Hirofumi Moriyama

BOD of 1394TA

Post	*Organization Represented*	*Name of Representative*
Chair	Littelfuse	Max Bassler,
Vice-Chair	TC Applied Technologies	Morten Lave
Secretary	LSI	Dave Thompson
BOD	IPRA	Richard Davis
BOD	TI	Toni Ray
BOD	DapTechnology	Richard Mourn

The Next Generation Camera Protocol Working Group in JIIA and the Industrial and Instrumentation Working Group in 1394TA, which developed and reviewed this specification, had the following members:

BOD of Next Generation Camera Protocol WG in JIIA

Sadafumi Torii	Hamamatsu Photonics	Chair
Yoshitsugu Nakasone	Toshiba Teli	Vice-chair
Masaki Horikawa	GRAPHIN	Vice-chair
Koji Tsuchiya	Techno Scope	Vice-chair

BOD of 1394TA

Max Bassler	Littelfuse	Chair
Morten Lave	TC Applied Technologies	Vice-chair
Richard Mourn	DapTechnology	BOD

BOD of Industrial and Instrumentation WG in 1394TA

Dave Thompson	LSI	Chair
Rod Barman	Point Grey Research	Chair of Digital Camera sub-WG

Other members

Shinichi Itokawa	Toshiba Teli
Junji Kishi	Toshiba Teli
Yasutoshi Onishi	Hamamatsu Photonics
Takayuki Inoue	Hamamatsu Photonics
Masato Tsujie	Sony
Akira Taroda	Sony
Hirofumi Takeuchi	Sony
Malcolm Steenburgh	Point Grey Research
Kazuhiro Igarashi	Hitachi Kokusai Electric
Akiyoshi Sugawara	Hitachi Kokusai Electric
Hiroki Kosaka	Hitachi Kokusai Electric
Tsuyoshi Yamaura	Advas
Shinichi Wakabayashi	System Garden
Tsuneharu Iizuka	Daitodenso
Koji Yahagi	Primetech Engineerting
Naoki Ogawa	Symco
Masahide Matsubara	AVAL DATA
Sam Liu	Newnex
Jan de Vries	Dap Technology
Garold M. Yurko	TE connectivity

Version history

Version 1.0.0 (January 13th, 2012) first version

IIDC2 Digital Camera Control Specification Ver.1.0.0

1 Scope and purpose

1.1 Scope

This document specifies the IIDC2 Digital Camera Control Specification Ver.1.0.0 for industrial use. This document describes control and status register definitions and control procedures.

1.2 Purpose

The purpose of this specification is to describe IIDC2 Digital Camera Control Specification Ver.1.0.0 for Digital Camera and industrial peripheral devices.

This specification is defines a set of addressed register. This specification was designed to be easily applied to different digital interfaces (such as IEEE1394,CoaXPress, CamerLink, USB and etc...) and adopted by their upper level protocols.

2 Definitions and notation

2.1 Definitions

2.1.1 Conformance

Several keywords are used to differentiate levels of requirements and optionally, as follows:

1 MAY: A keyword that indicates flexibility of choice with no implied preference.

2 SHALL / SHALL NOT: A keyword that indicates a mandatory requirement. Designers are required to implement all such mandatory requirements to assure interoperability with other products conforming to this specification.

3 SHOULD / SHOULD NOT: A keyword that denotes flexibility of choice with a strongly preferred alternative. Equivalent to the phrase “is recommended.”

4 ignored: A keyword that describes objects (bits, bytes, quadlets or fields) whose values are not checked by the recipient.

5 reserved: A keyword used to describe objects or the code values assigned to these objects in cases where either the object or the code value is set aside for future standardization. Usage and interpretation MAY be specified by future extensions to this or other specifications. A reserved object SHALL be zeroed or, upon development of a future specification, set to a value specified by such a specification. The recipient of a reserved object SHALL ignore its value. The recipient of an object defined by this specification as other than reserved SHALL inspect its value and reject reserved code values.

6 not-used: A Keyword used to describe objects or the code whose values are defined but not used. A not-used object SHALL be zeroed. The recipient of a not-used object SHALL ignore its value.

7 read-only: A keyword used to describe objects who has read-only values. The recipient of a read-only object SHALL ignore its value.

2.1.2 Glossary

The following terms are used in this specification:

1 quadlet: four bytes of data

2.1.3 Abbreviations

The following are abbreviations that are used in this specification:

CSR	Control and status register

As exemplified by CSR, abbreviations MAY cite a bibliographic reference.

2.2 Notation

2.2.1 Numeric values

Decimal and hexadecimal are used within this specification. By editorial convention, decimal numbers are most frequently used to represent quantities or counts. Addresses are uniformly represented by hexadecimal numbers. Hexadecimal numbers are also used when the value represented has an underlying structure that is more apparent in a hexadecimal format than in a decimal format.

Decimal numbers are represented by Arabic numerals without subscripts or by their English names. Hexadecimal numbers are represented by digits from the character set 0 – 9 and A – F with prefix code of 0x (C-language style). For the sake of legibility hexadecimal numbers are separated into groups of four digits separated by spaces.

As an example, 42 and 0x2A both represent the same numeric value.

2.2.2 Bit, byte and quadlet ordering

This specification uses the facilities of interface and therefore uses the ordering conventions of interface in the representation of data structures. In order to promote interoperability with memory buses that MAY have different ordering conventions, this specification defines the order and significance of bits within quadlet, bytes within quadlets and quadlets within several quadlets area in terms of their relative position and not their physically addressed position.

At bit ordering within a quadlet, the most significant bit (msb) is left bound bit, and least significant bit (lsb) is right bound bit. Other else bit fields are same as it.

msb lsb

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 1 – Bit ordering within a quadlet

There are two types of byte ordering within a quadlet: One is Big Endian, the other is Little Endian. Which ordering to use is depend on the specification of interface.

msb lsb

1st byte	2nd byte	3rd byte	4th byte

Figure 2 – Byte ordering within a quadlet at Big Endian

msb lsb

4th byte	3rd byte	2nd byte	1st byte

Figure 3 – Byte ordering within a quadlet at Little Endian

Similarly, there are two types of quadlet ordering within a several quadlets area.

msb lsb

1st quadlet
2nd quadlet
3rd quadlet
4th quadlet

Figure 4 –Quadlet ordering within a several quadlets area at Big Endian

msb

4th quadlet
3rd quadlet
2nd quadlet
1st quadlet

Figure 5 –Quadlet ordering within a several quadlets area at Little Endian

3 Digital device control register

3.1 IIDC2Entry address

IIDC2 register space starts from **IIDC2Entry**

3.2 General structure

IIDC2Entry has the address offset list of **CategoryBlocks**. Each **CategoryBlock** has some **FeatureCSRs** which are categorized. Every feature has one bunch of registers (**FeatureCSR**). **IIDC2** has two groups of categories. One is for **BasicCategoryBlock** and the other is for **ExpandedCategoryBlock**. From **CategoryBlock0** to **CategoryBlock31** are **BasicCategoryBlocks**, and upper of **CategoryBlock32** are **ExpandedCategoryBlocks**. **BasicCategoryBlocks** contain **BasicCSRs** which provide the way to handle standard device functions. **ExpandedCategoryBlocks** contain **ExpandedCSRs** which provide the Expanded handling methods. **ExpandedCSRs** are chained from **BasicCSRs** by using **OffsetForExpanded** which points the address of **ExpandedCSR**. Devices MAY use **ExpandedCSRs** for these proprietary specifications.

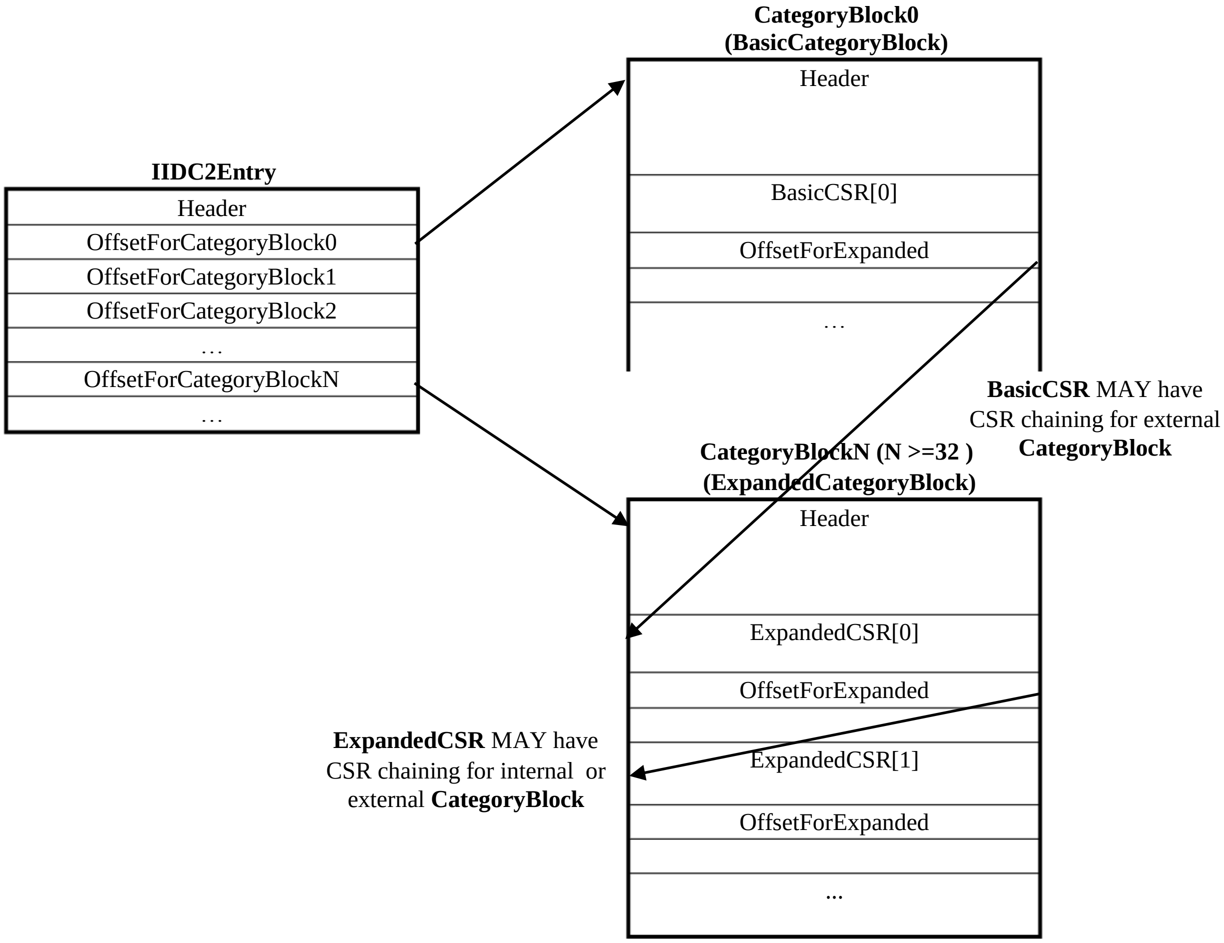

Figure 6 – General Structure

3.3 IIDC2Entry (offset list of CategoryBlock)

IIDC2Entry contains the list of quadlet offsets for **CategoryBlock.** Each feature has one bunch of control registers and status registers(**CSR**). All Features are categorized and located inside each **CategoryBlock**.

Device that don't implement the **CategoryBlock** which is not necessary MAY set the quadlet offset to zeros.

Offset	Name	Description	
+0x000	Keyword	0x4949 4443 ("IIDC")	
+0x004	Version	[31..24]	reserved
		[23..16]	Major version number
		[15..8]	Minor version number
		[7..0]	Sub-minor version number
+0x008	NumberOfCategoryBlocks	[31..16]	reserved
		[15..0]	Number of CategoryBlocks (=N)
+0x00C	-	reserved	
+0x010	OffsetForXmlManifestTable	Quadlet offset for XML manifest table	
+0x014~ 0x01C	-	reserved	
+0x020	OffsetForCategoryBlock0	Quadlet offset for CategoryBlock0	
+0x024	OffsetForCategoryBlock1	Quadlet offset for CategoryBlock1	
+0x028	OffsetForCategoryBlock2	Quadlet offset for CategoryBlock2	
+0x02C	OffsetForCategoryBlock3	Quadlet offset for CategoryBlock3	
+0x030	OffsetForCategoryBlock4	Quadlet offset for CategoryBlock4	
+0x034	OffsetForCategoryBlock5	Quadlet offset for CategoryBlock5	
+0x038	OffsetForCategoryBlock6	Quadlet offset for CategoryBlock6	
+0x03C	OffsetForCategoryBlock7	Quadlet offset for CategoryBlock7	
+0x040	OffsetForCategoryBlock8	Quadlet offset for CategoryBlock8	
+0x044	OffsetForCategoryBlock9	Quadlet offset for CategoryBlock9	
+0x048	OffsetForCategoryBlock10	Quadlet offset for CategoryBlock10	
+0x04C~ +0x098	-	reserved	
+0x09C	OffsetForCategoryBlock31	Quadlet offset for CategoryBlock31	
+0x0A0	OffsetForCategoryBlock32	Quadlet offset for CategoryBlock32 (ExpandedCategoryBlock)	
+0x0A4	OffsetForCategoryBlock33	Quadlet offset for CategoryBlock33	
...	...	...	
+(0x004* N +0x01C)	OffsetForCategoryBlock(N-1)	Quadlet offset for CategoryBlock(N-1)	

Table 1 –IIDC2Entry

3.3.1 Keyword

Contains the string “IIDC” in ASCII.

3.3.2 Version

Indicates the version of IIDC specification. This field contains the quadlet 0x0001 0000

3.3.3 NumberOfCategoryBlocks

Indicates the number of **CategoryBlock** a device has. It includes reserved space.

3.3.4 OffsetForXmlManifestTable

Indicates offset address of XML Manifest table which is shown at section 3.6 XmlManifestTable. Offset address is the relative address from the top of **IIDC2Entry**.

XML Manifest table in IIDC2 is used for only interfaces which have no XML Manifest in transport layer (e.g. IEEE1394). If the interface uses an XML Manifest in the transport layer (e.g. CoaXPress), this field SHALL be not-used.

3.3.5 OffsetForCategocyBlock0 ~ (N-1)

Indicates the offset addresses of each CategoryBlock. Offset address is the relative address from the top of **IIDC2Entry**.

3.4 Structure of CategoryBlock

3.4.1 BasicCategoryBlock

The standard features (e.g. Exposure time, Gain and etc...) are categorized and placed in **BasicCategoryBlock**. Relationship between index of **CategoryBlock** and category name is described in the following table.

Name	Description
CategoryBlock0	DeviceControl
CategoryBlock1	TransportLayerControl
CategoryBlock2	ImageFormatControl
CategoryBlock3	AcquisitionControl
CategoryBlock4	LuminanceControl
CategoryBlock5	ChromaControl
CategoryBlock6	LUTControl
CategoryBlock7	TriggerControl
CategoryBlock8	UserSetControl
CategoryBlock9	DigitalIOControl
CategoryBlock10	CounterAndTimerControl
CategoryBlock31	VendorUniqueControl

Table 2 – BasicCategoryBlock list

BasicCategoryBlock contains only **BasicCSRs**. All **BasicCSRs** are located to fixed addressing which is defined by **IIDC2**.

Offset	Description
+0x000	Header
+0x020	BasicCSR[0]
+0x040	BasicCSR[1]
+0x060	BasicCSR[2]
...	...

Table 3 – Structure of BasicCategoryBlock

3.4.2 ExpandedCategoryBlock

ExpandedCategoryBlock contains the **ExpandedCSRs**. Devices MAY implement **ExpandedCSRs** to handle its unique features. Address of **ExpandedCSR** is pointed to by the **OffsetForExpanded** in the **BasicCSR** which is described in section 3.2 General structure.

ExpandedCSRs do not have constant address offsets because the length of the **Value** register space varies according to the **ValueType**.

Offset	Description
+0x000	Header
+0x020	ExpandedCSR[0]
+0xXXX	ExpandedCSR[1]
+0xYYY	ExpandedCSR[2]
...	...

Table 4 – Structure of ExpandedCategoryBlock

3.4.3 Header of CategoryBlock

Offset	Name	Bit	Description
+0x000	Header	[31..24]	CategoryBlockNumber
		[23..0]	SizeOfCategoryBlock (by quadlet)
+0x004~ 0x01C		reserved	

Table 5 – Header of CategoryBlock

CategoryBlockNumber is the index of CategoryBlock.

SizeOfCategoryBlock indicates the size of CategoryBlock (including header area) in quadlets.

3.5 Standard structure of FeatureCSR

Offset	Name	31-24	23-16	15-8	7-0
+0x00	Inquiry	Implemented Active UserSetLoadable Writable Readable FeatureInq	ValueType	ControlInq	DefaultInq AutoOnceInq AutoInq ManualInq NoSpecifyInq
+0x04	OffsetFor Expanded	CategoryBlockNumber	OffsetAddress		
+0x08	Control				Control
+0x0C~	Value Register Area	Value			

Table 6 – Structure of FeatureCSR

3.5.1 Inquiry

The first quadlet represents basic information about the feature. All bits here are read-only.

The upper 8-bits are **FeatureInq**. It describes information of this **FeatureCSR**. The middle 8-bits are **ValueType**. It indicates the type of **ValueRegisterArea**. The lower 16-bits are **ControlInq**. It indicates the available controls in this **FeatureCSR**.

Bit	Field	Description
[31]	Implemented	**1**: Implemented **0**: not Implemented This field represents whether device has this feature. This field never changes.
[30]	Active	**1**: Active **0**: inactive This field represents whether this feature is active currently.

Table 7 – Inquiry

Bit	Field	Description
[26]	UserSetLoadable	**1**: **Value** and **Control** are updated when **UserSetControl** = **Load**. **0**: Not to be updated This field represents whether this feature setting is to be loaded or not when **UserSetControl** = **Load**. Device MAY change this field according the current device status.
[25]	Writable	**1** : **Value** is writable **0** : **Value** is not writable This field represents whether **Value** is writable or not.
[24]	Readable	**1** : **Value** is readable **0** : **Value** is not readable This field represents whether **Value** is readable or not.
[23..16]	ValueType	Described in the section 3.5.4 ValueRegisterArea (Common register type) This field represents Value type.
[4]	DefaultInq	**1** : **Default** is available **0** : **Default** is not available This field represents whether **Default** control is available or not.
[3]	AutoOnceInq	**1** : **AutoOnce** is available **0** : **AutoOnce** is not available This field represents whether **AutoOnce** control is available or not.
[2]	AutoInq	**1** : **Auto** is available **0** : **Auto** is not available This field represents whether **Auto** control is available or not.
[1]	ManualInq	**1** : **Manual** is available **0** : **Manual** is not available This field represents whether **Manual** control is available or not.
[0]	NoSpecifyInq	**1** : **NoSpecify** is available **0** : **NoSpecify** is not available This field represents whether **NoSpecify** mode is available or not.

Table 7 – Inquiry (Contd.)

Implemented	Active	Writable	Readable	Value	Description
0	not-used	not-used	not-used	not-used	Device doesn't have any internal functions related to this **FeatureCSR**. Or device has an internal function which is not provided any control.
1	0	X	X	X	Device has an internal function related to this **FeatureCSR**, but it is not active temporarily because of other **FeatureCSRs** setting. (e.g. **Hue** is not active when **PixelCoding** = **Mono**) Host MAY turn it active with writing other **FeatureCSRs**.
1	1	0	X	last update / last write	**Value** is locked temporarily because of other **FeatureCSRs** setting. (e.g. **ImageSize** is locked when **AcquisitionCommand** = **Continuous**) Host MAY unlock **Value** by appropriately change the **FeatureCSRs**. Or **Value** is write-disable (read-only) permanently.
1	1	X	0	not used	**Value** is invalid number because host wrote to chaining **ExpandedCSR,** (e.g. In case that device which has chaining **ExpandedCSR** whose dynamic range is wider than **BasicCSR**, **Value** of **BasicCSR** is invalid when host wrote external value to **ExpandedCSR**) or host wrote to **FeatureCSR** which handles the same internal function in the device. (e.g. In case that device which has **Gamma** and **LUT FeatureCSRs** using unity internal function, **Value** of Gamma is invalid when host wrote to LUT **FeatureCSR** at the last). Or device is not able to indicate actual number when **Control** is **NoSpecify**, **Auto** or **AutoOnce**.

Table 8 – Truth table of Inquiry

3.5.2 OffsetForExpanded

This field is optional. When this field is **0x000 00000**, then this feature has no **ExpandedCSRs**.

Bit	Field	Description
[31..24]	CategoryBlockNumber	Indicates the index of **CategoryBlock** which exist in chained **ExpandedCSR**. If this field is zero, chaining **ExpandedCSR** is in the same **CategoryBlock**.
[23..0]	Offset	Indicates the offset quadlets of chaining **ExpandedCSR** from the top of **CategoryBlock**.

Table 9 – OffsetForExpanded

3.5.3 Control

The **Control** field sets the mode which determines how to the **Value** field is handled in the **ValueRegisterArea**. If the **Control** field is set to an unavailable control number which is specified in the **ControlInq**, device SHALL discard this writing number and keep previous number.

Bit	Field	Description
[3..0]	Control	Mode control **0** : NoSpecify **1** : Manual **2** : Auto **3** : AutoOnce **4** : Default Set the Mode number among available modes.

Table 10 – Control

3.5.3.1 **NoSpecify**

Device MAY set **Value** to device-dependent value. Device SHOULD update **Value** to actual value, otherwise device SHALL turn **Readable** to **0**.

3.5.3.2 **Manual**

Host MAY write **Value**. Device SHOULD NOT change **Value** except in the following case:

- Corrects rounding issue
- In case of using state control,. internal state was changed.
- Follows actual value updated by chaining **FeatureCSR** (please see section 5.1.1 Value).
- Follows actual value updated by **FeatureCSR** which uses same internal function.
- The range of **Value** (**Min**, **Max** and **Inc**) was changed and **Value** is out of range.

Because of these cases, host SHOULD read back **Value** after writing.

When host writes **Value**, device SHALL change **Control** to **Manual**.

3.5.3.3 **Auto**

Device adjusts **Value** automatically by itself. Device SHOULD update **Value** whenever actual value was changed, otherwise device SHALL turn **Readable** to **0**.

Host MAY change **Control** while adjustment is running. If host changes **Control** to **Manual**, device SHALL set **Value** to the last adjusted value.

3.5.3.4 **AutoOnce**

Device adjusts **Value** by itself only once. After adjustment, Device SHALL turn **Control** to **Manual** and SHOULD update **Value** to actual value. If updating **Value** is impossible, device SHALL turn **Readable** to **0**.

Host MAY change **Control** while adjustment is running. If host changes **Control** to **Manual**, device SHALL set **Value** to previous value (before set to **AutoOnce**).

3.5.3.5 **Default**

Device changes **Value** and **Control** to default values which are device dependent.

If device default is **Auto**, the device sets **Control** to **Auto** after it receives **Default** control setting.

3.5.4 ValueRegisterArea (Common register type)

All **FeatureCSRs** SHALL have **ValueType** described in this section (common register type). **ValueType** in **BasicCSRs** is fixed in the **IIDC2** specification. Device SHALL NOT change **ValueType** in **BasicCSRs**.

3.5.4.1 **Integer**

Offset	Definitions
+0x00	Mult[31..0] (read-only)
+0x04	Div[31..0] (read-only)
+0x08	Min[31..0] (read-only)
+0x0C	Max[31..0] (read-only)
+0x10	Value[31..0]

Table 11 – Integer32 (ValueType = 0x30)

Big Endian

Offset	Definitions
+0x00	Mult[63..32] (read-only)
+0x04	Mult[31..0] (read-only)
+0x08	Div[63..32] (read-only)
+0x0C	Div[31..0] (read-only)
+0x10	Min[63..32] (read-only)
+0x14	Min[31..0] (read-only)
+0x18	Max[63..32] (read-only)
+0x1C	Max[31..0] (read-only)
+0x20	Value[63..32]
+0x24	Value[31..0]

Little Endian

Offset	Definitions
+0x00	Mult[31..0] (read-only)
+0x04	Mult[63..32] (read-only)
+0x08	Div[31..0] (read-only)
+0x0C	Div[63..32] (read-only)
+0x10	Min[31..0] (read-only)
+0x14	Min[63..32] (read-only)
+0x18	Max[31..0] (read-only)
+0x1C	Max[63..32] (read-only)
+0x20	Value[31..0]
+0x24	Value[63..32]

Table 12 – Integer64 (ValueType = 0x40)

Value is signed integer. **Value** SHALL support integers ranging from **Min** to **Max**. If both **Mult** and **Div** are valid values, then **Value** represents absolute value with unit (unit is specified in 4 Device Control Register) and both **Mult** and **Div** are also signed integer. Then absolute value is calculated by the equation below.

$$\text{Absolute Value} = \textbf{Value} * (\textbf{Mult} / \textbf{Div}) \qquad (\textbf{Mult} \text{ <> } \textbf{Div} \text{ or } \textbf{Mult} = \textbf{div} = \textbf{1})$$

If **Mult** = **Div** and they are not equal to **0** or **1**, these indicate increment value (**Inc**). In this case, **Value** is relative value without unit.

If either **Mult**=**0** or **Div**=**0**, then **Value** is relative value and **Inc**=**1**.

$$\textbf{Value} = n * \textbf{Inc} \qquad (n \text{ is integer})$$

$$\textbf{Inc} = \begin{cases} \textbf{Mult} = \textbf{Div} & (\textbf{Mult} = \textbf{Div} \text{ <> } \textbf{0} \text{ and } \textbf{Mult} = \textbf{Div} \text{ <> } \textbf{1}) \\ 1 & (\textbf{Mult} = 0 \text{ or } \textbf{Div} = \textbf{0} \text{ or } \textbf{Mult} = \textbf{Div} = \textbf{1}) \end{cases}$$

If host writes the value that is out of range from **Min** to **Max** or not multiples of **Inc**, device SHALL discard this writing value and keep previous value.

3.5.4.2 **PlainInteger**

Big Endian

Offset	Definitions	
+0x00	Value[7..0]	reserved

Little Endian

Offset	Definitions	
+0x00	reserved	Value[7..0]

Table 13 – PlainInteger8 (ValueType = 0x11)

Offset	Definitions
+0x00	Value[31..0]

Table 14 – PlainInteger32 (ValueType = 0x31)

Big Endian

Offset	Definitions
+0x00	Value[63..32]
+0x04	Value[31..0]

Little Endian

Offset	Definitions
+0x00	Value[31..0]
+0x04	Value[63..32]

Table 15 – PlainInteger64 (ValueType = 0x41)

Value is unsigned integer, and it has no information (**Min**, **Max**, **Mult** or **Div**). **Value** MAY take full range of the bit width.

3.5.4.3 **Float**

Offset	Definitions
+0x00	Min[31..0] (read-only)
+0x04	Max[31..0] (read-only)
+0x08	Value[31..0]

Table 16 – Float32 (ValueType = 0x32)

Big Endian

Offset	Definitions
+0x00	Min[63..32] (read-only)
+0x04	Min[31..0] (read-only)
+0x08	Max[63..32] (read-only)
+0x0C	Max[31..0] (read-only)
+0x10	Value[63..32]
+0x14	Value[31..0]

Little Endian

Offset	Definitions
+0x00	Min[31..0] (read-only)
+0x04	Min[63..32] (read-only)
+0x08	Max[31..0] (read-only)
+0x0C	Max[63..32] (read-only)
+0x10	Value[31..0]
+0x14	Value[63..32]

Table 17 – Float64 (ValueType = 0x42)

Value is floating point value which is defined by IEEE 754-2008.

If host writes a value which is outside the range of **Min** to **Max**, device SHALL discard this writing value and keep previous value.

If there is difference between writing value and applying value attributed to rounding error, device SHALL update **Value** to the actual value (or the nearest value).

3.5.4.4 **Enumeration**

Big Endian

Offset	Definitions
+0x00	ListOfElements[127..96] (read-only)
+0x04	ListOfElements[95..64] (read-only)
+0x08	ListOfElements[63..32] (read-only)
+0x0C	ListOfElements[31..0] (read-only)
+0x10	Value[31..0]

Little Endian

Offset	Definitions
+0x00	ListOfElements[31..0] (read-only)
+0x04	ListOfElements[63..32] (read-only)
+0x08	ListOfElements[95..64] (read-only)
+0x0C	ListOfElements[127..96] (read-only)
+0x10	Value[31..0]

Table 18 – Enumeration (ValueType = 0x03)

This type is used to select one option from the list. Then number is set to **Value** register. (**Value** MAY take from **0** to **127**.) Each bit in **ListOfElements** represents availability of its value (**1** : available, **0** : not available)

If host writes a value that is not available in **ListOfElements** device SHALL discard this writing value and keep previous value.

3.5.4.5 **BulkBoolean**

Offset	Definitions
+0x00	BitWritable[31..0] (read-only)
+0x04	Value[31..0]

Table 19 – BulkBoolean32 (ValueType = 0x34)

Big Endian

Offset	Definitions
+0x00	BitWritable[63..32] (read-only)
+0x04	BitWritable[31..0] (read-only)
+0x08	Value[63..32]
+0x0C	Value[31..0]

Little Endian

Offset	Definitions
+0x00	BitWritable[31..0] (read-only)
+0x04	BitWritable[63..32] (read-only)
+0x08	Value[31..0]
+0x0C	Value[63..32]

Table 20 – BulkBoolean64 (ValueType = 0x44)

This type is used to select one or more options from the list. One register has up to 32 (**BulkBoolean32**) / 64 (**BulkBoolean64**) options. Each bit has writable flag separately, it is **BitWritable**. **0** is intended read-only (not writable), **1** is writable. Device SHALL apply writing action to writable bit fields in **Value**.

3.5.4.6 **Rectangle**

Offset	Definitions
+0x00	MinOffsetX[31..0] (read-only)
+0x04	IncOffsetX[31..0] (read-only)
+0x08	MinWidth[31..0] (read-only)
+0x0C	IncWidth[31..0] (read-only)
+0x10	TotalSizeX[31..0] (read-only)
+0x14	MinOffsetY[31..0] (read-only)
+0x18	IncOffsetY[31..0] (read-only)
+0x1C	MinHeight[31..0] (read-only)
+0x20	IncHeight[31..0] (read-only)
+0x24	TotalSizeY[31..0] (read-only)
+0x28	OffsetX[31..0]
+0x2C	Width[31..0]
+0x30	OffsetY[31..0]
+0x34	Height[31..0]

Table 21 – Rectangle32 (ValueType = 0x35)

This type is used for defining rectangle area.

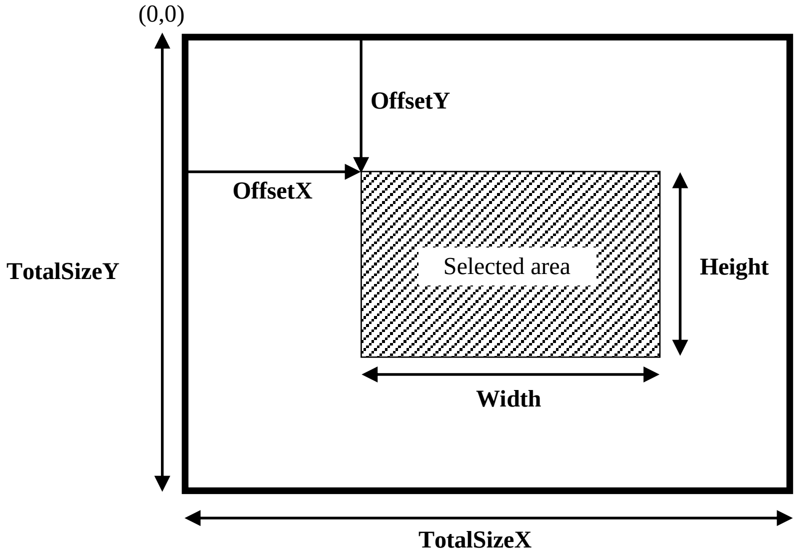

Figure 7 – Elements of Rectangle type

All values (**OffsetX**, **Width**, **OffsetY** and **Height**) are signed integer 32. Their relation SHALL be as follows.

OffsetX = **IncOffsetX** * n1 + **MinOffsetX**

Width = **IncWidth** * n2 + **MinWidth**

OffsetX + **Width** <= **TotalSizeX**

OffsetY = **IncOffsetY** * m1 + **MinOffsetY**

Height = **IncHeight** * m2 + **MinHeight**

OffsetY + **Height**<= **TotalSizeY** (n1, n2, m1, m2 are integer)

3.5.4.7 **Array of Register**

All register types MAY be expanded to array version of that register type. **0x80** is added to the **ValueType** of standard (not arrayed) register type. For example, **ValueType** of **Integer32** is **0x30**, then **ValueType** of **ArrayOfInteger32** is **0x30** + **0x80** = **0xB0**.

In array of register, **ArrayInformationArea** is inserted on the top of **ValueRegisterArea**. And **ExtendedValueArea** is inserted under the bottom of **ValueRegisterArea**.

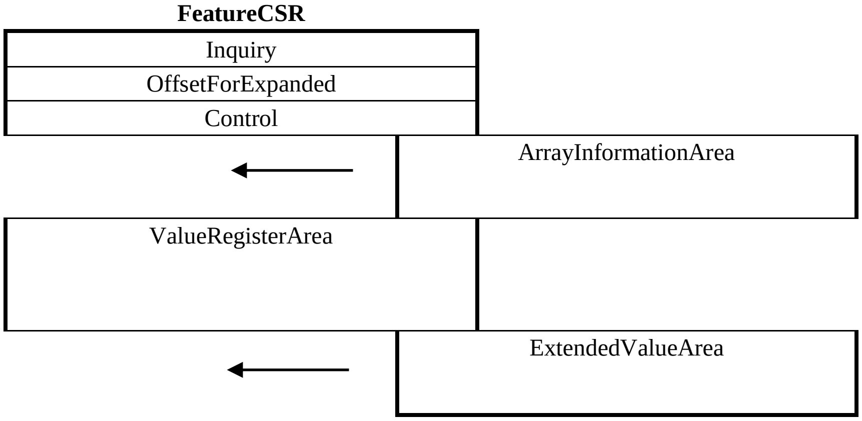

Figure 8 – Array of register

ArrayInformationArea contains one or more quadlets as described below.

Offset	31-24		23-16	15-8	7-0
+0x00	MoreDimension	NumberOfElements			

Table 22 – Quadlet of ArrayInformationArea

MoreDimension indicates whether next quadlet is **ArrayInformationArea** or not. If **MoreDimension** = **1**, **Value** has more dimension and next quadlet indicates information of next dimension. If **MoreDimension** = **0**, this quadlet is end of **ArrayInformationArea**.

NumberOfElements defines the number of elements in this dimension..

Offset	Definitions	
+0x00	MoreDimension=1	NumberOfElements = 3
+0x04	MoreDimension=0	NumberOfElements = 2
+0x08	Mult[31..0] (read-only)	
+0x0C	Div[31..0] (read-only)	
+0x10	Min[31..0] (read-only)	
+0x14	Max[31..0] (read-only)	
+0x18	Value[0][0][31..0]	
+0x1C	Value[0][1][31..0]	
+0x20	Value[1][0][31..0]	
+0x24	Value[1][1][31..0]	
+0x28	Value[2][0][31..0]	
+0x2C	Value[2][1][31..0]	

Table 23 – Example of ArrayOfInteger32 with 2 dimension (ValueType = 0xB0)

3.5.4.8 **String**

String is substituted by **ArrayOfPlainInteger8**.

Big Endian

Offset	Definitions			
+0x00	MoreDimension=0	NumberOfElements = 16		
+0x04	char[0]	char[1]	char[2]	char[3]
+0x08	char[4]	char[5]	char[6]	char[7]
+0x0C	char[8]	char[9]	char[10]	char[11]
+0x10	char[12]	char[13]	char[14]	char[15]

Little Endian

Offset	Definitions			
+0x00	MoreDimension=0	NumberOfElements = 16		
+0x04	char[3]	char[2]	char[1]	char[0]
+0x08	char[7]	char[6]	char[5]	char[4]
+0x0C	char[11]	char[10]	char[9]	char[8]
+0x10	char[15]	char[14]	char[13]	char[12]

Table 24 – Example of String with 16 characters (ValueType = 0x91)

3.5.5 Multi-Byte accessing

The data size of writing action is dependent on the interface. Then there is the situation that data size of **Value** is larger than writing data size. So host needs to use several writing actions for updating **Value**.

In this case, device SHALL wait to apply the writing value until host writes to “field of Largest address number” in **Value**.

3.6 XmlManifestTable

XmlManifestTable is used to connect GenICam from IIDC2. It is used for only interfaces which have no XML Manifest in transport layer (e.g. IEEE1394). If the interface has it in transport layer (e.g. CoaXPress), this field SHALL be not-used.

Offset	Name	Description	
+0x000	XmlManifestSize		
+0x004	XmlManifestSelector		
+0x008	XmlVersion	[31..24]	reserved
		[23..16]	XmlMajorVersion
		[15..8]	XmlMinorVersion
		[7..0]	XmlSubMinorVersion
+0x00C	XmlSchemaVersion	[31..24]	reserved
		[23..16]	SchemaMajorVersion
		[15..8]	SchemaMinorVersion
		[7..0]	ScemaSubMinorVersion
+0x010~	XmlUrlAddress		

Table 25 – XmlManifestTable

3.6.1 XmlManifestSize

Provides the number of XML manifests available.

3.6.2 XmlManifestSelector

Selects the required XML manifest. It SHALL hold a number between 0 and **XmlManifestSize**-1.

3.6.3 XmlVersion

Provides the version number for the XML file given in the manifest referenced by **XmlManifestSelector**.

3.6.4 XmlSchemaVersion

Provides the GenICam schema version for the XML file given in the manifest referenced by **XmlManifestSelector**.

3.6.5 XmlUrlAddress

Provides the address of the start of the URL string referenced by **XmlManifestSelector**. This string SHALL be in the format defined as follows.

3.6.5.1 **URL Format - Non-Volatile Memory**

If the XML files is stored in non-volatile memory in the device, the URL SHALL be of the form:

“Local:<Filename>.<Extension>;<Address>;<Length>” as defined in Table 26.

Field	Description
Local	Indicates the XML file is stored in non-volatile memory in the device.
<Filename>	the name of the XML file. It SHOULD includes the vendor name, model name and device revision.
<Extension>	“xml” indicates a text XML file (i.e. uncompressed). “zip” indicates a ZIP format compressed file.
<Address>	The absolute address of the XML file. It is given in hexadecimal form without a leading “0x”.
<Length>	The length of the XML file in bytes, given in hexadecimal without a leading “0x”.

Table 26 – URL format – Non-volatile memory

Example: “Local:MyCompany_MyProdult_Rev1.zip;B8000;33A” is a ZIP file starting at address 0xB8000 in the device with a length of 0x33A bytes. The XML file is for revision 1 of a device called “MyProduct” made by “MyCompany”.

3.6.5.2 URL Format – Vendor website

If the XML file is stored on the vendor's website, the URL SHALL be of the form:

"Web:<WebURL>/<Filename>.<Extension>" as defined in Table 27.

Field	Description
Web	Indicates the XML file is stored on the vendor's website
<WebURL>	A full web URL, form the scheme name (e.g. http) to the path.
<Filename>	The name of the XML file. It SHOULD includes the vendor name, model name and device revision.
<Extension>	"xml" indicates a text XML file (i.e. uncompressed). "zip" indicates a ZIP format compressed file.

Table 27 – URL format – Vendor website

Example: "Web:http://www.mycompany.com/xml/MyCompany_MyProduct_Rev1.xml" is a text XML file found at http://www.mycompany.com/xml. The XML file is for revision 1 of a device called "MyProduct" made by "MyCompany"

None of the fields are case sensitive.

4 Device Control Register

All tables of **CategoryBlocks** are defined as follows.

□ Device dependent : Device MAY use this field.
🟧 Feature dependent : Device and host SHALL keep the constant value described in this specification.
▩ reserved / not-used : Device SHALL keep zeros. Host SHALL NOT set ones.

If **FeatureCSR** has a **Control** and/or **Integer32** register with absolute value, device SHOULD comply with following table.

Unit of absolute value		describes unit of the absolute value
Reference point		describes reference point of the absolute value
Recommended Value at	AutoOnce / Auto	describe recommended behavior of **Value** in each **Controls**
	NoSpecify	

If **Implemented** is **0**, device MAY set all fields including Feature dependent field to zeros in its **FeatureCSR**.

4.1 CategoryBlock0 (DeviceControl)

Offset	Name	Field	Bit	Description
+0x000	Header	CategoryBlockNumber	[31..24]	= 0
		SizeOfCategoryBlock	[23..0]	= 0x000050
+0x004~ 0x01C		-	-	reserved
+0x020	Device Reset	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		ControlInq	[15..0]	not-used
+0x024		OffsetForExpanded	[31..0]	
+0x028		Control	[31..0]	not-used
+0x02C~ 0x038		ListOfElements	[127..2]	not-used
			[1]	Reset
			[0]	Off
+0x03C		Value	[31..0]	
+0x040	Device Power	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		ControlInq	[15..0]	not-used
+0x044		OffsetForExpanded	[31..0]	
+0x048		Control	[31..0]	not-used
+0x04C~ 0x058		ListOfElements	[127..2]	not-used
			[1]	On
			[0]	LowPower
+0x05C		Value	[31..0]	

Table 28 –CategoryBlock0 (DeviceControl)

Offset	Name	Field	Bit	Description
+0x060	Device Vendor Name	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	= 0 (Read-only)
		Readable	[24]	
		ValueType	[23..16]	= 0x91 (ArrayOfPlainInteger8)
		ControlInq	[15..0]	not-used
+0x064		OffsetForExpanded	[31..0]	
+0x068		Control	[31..0]	not-used
+0x06C		MoreDimension	[31]	= 0
		NumberOfElements	[30..0]	= 16
+0x070~ 0x07C		Value[0]~Value[15]		
+0x080~ 0x09C	Device Model Name	Same Structure as DeviceVendorName		
+0x0A0~ 0x0BC	Device Manufacturer Info	Same Structure as DeviceVendorName		
+0x0C0~ 0x0DC	Device Version	Same Structure as DeviceVendorName		
+0x0E0~ 0x0FC	Device Firmware Version	Same Structure as DeviceVendorName		
+0x100~ 0x11C	DeviceID	Same Structure as DeviceVendorName		
+0x120	Devicez UserID	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x91 (ArrayOfPlainInteger8)
		ControlInq	[15..0]	not-used
+0x124		OffsetForExpanded	[31..0]	
+0x128		Control	[31..0]	not-used
+0x12C		MoreDimension	[31]	= 0
		NumberOfElements	[30..0]	= 16
+0x130~ 0x13C		Value[0]~Value[15]		

Table 28 –CategoryBlock0 (DeviceControl) (Contd.)

DeviceControl handles condition of device and indicates information of device. If the transport layer has same registers (e.g. CoaXPress has **DeviceVendorName**, **DeviceModelName**, **DeviceManufacturerInfo**, **DeviceVersion**, **DeviceFirmwareVersion**, **DeviceID** and **DeviceUserID** registers), device SHOULD NOT have these **FeatureCSRs**.

The information **FeatureCSRs** (from **DeviceVendorName** to **DeviceUserID**) have **Values** as ASCII strings, they are up to 16 characters. Device MAY have **Value** beyond 16 characters by using the chaining **ExpandedCSR** (in this case, **Value** of **BasicCSR** SHALL be filled by NULL string).

4.1.1 DeviceReset

When this feature is executed, the device behavior SHALL be the same a transition to the power on state.

Reset: Reset the device. After reset sequence, device SHALL turn to **Off** automatically.
Off: Normal operation

4.1.2 DevicePower

Controls device’s Power.

On: Power on the device.
LowPower : Power off the device without interface block.

4.1.3 DeviceVendorName

Indicates the manufacturer’s name of the device.

4.1.4 DeviceModelName

Indicates the model name of the device.

4.1.5 DeviceManufacturerInfo

Indicates the extended information of manufacturer.

4.1.6 DeviceVersion

Indicates the version of the device.

4.1.7 DeviceFirmwareVersion

Indicates the version of the firmware in the device.

4.1.8 DeviceID

Indicates the device identifier. It SHOULD be the same as the serial number.

4.1.9 DeviceUserID

Handles the user-programmable identifier. The device user MAY set and read the original ID to device. Device SHALL store it to non-volatile memory in the device when this **FeatureCSR** was written.

4.2 CategoryBlock1 (TransportLayerControl)

This **CategoryBlock** is different for each interfaces.

Please see section 6 Transport layer of IEEE1394 (for IEEE1394) or each specification of transport layer.

4.3 CategoryBlock2 (ImageFormatControl)

Offset	Name	Field	Bit	Description
+0x000	Header	CategoryBlockNumber	[31..24]	= 2
		SizeOfCategoryBlock	[23..0]	= 0x000048
+0x004~ 0x01C		-	-	reserved
+0x020	Image Format Selector	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	not-used
		AutoInq	[2]	not-used
		ManualInq	[1]	
		NoSpecifyInq	[0]	not-used
+0x024		OffsetForExpanded	[31..0]	
+0x028		-	[31..4]	reserved
		Control	[3..0]	
+0x02C~ 0x038		ListOfElements	[127..32]	not-used
			[31]	Format31
			...	...
			[1]	Format1
			[0]	Format0
+0x03C		Value	[31..0]	
+0x040	Apply Image Format	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		ControlInq	[15..0]	not-used
+0x044		OffsetForExpanded	[31..0]	
+0x048		Control	[31..0]	not-used

Table 29 – CategoryBlock2 (ImageFormatControl)

Offset	Name	Field	Bit	Description
+0x04C~0x058	Apply Image Format	ListOfElements	[127..17]	not-used
			[16]	ImageFormatError
			[15..9]	not-used
			[8]	Changed
			[7..2]	not-used
			[1]	Apply
			[0]	Done
+0x05C		Value	[31..0]	
+0x060	ImageSize	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x35 (Rectangle32)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	not-used
		AutoInq	[2]	not-used
		ManualInq	[1]	
		NoSpecifyInq	[0]	not-used
+0x064		OffsetForExpanded	[31..0]	
+0x068		-	[31..4]	reserved
		Control	[3..0]	
+0x06C		MinOffsetX	[31..0]	
+0x070		IncOffsetX	[31..0]	
+0x074		MinWidth	[31..0]	
+0x078		IncWidth	[31..0]	
+0x07C		TotalSizeX	[31..0]	
+0x080		MinOffsetY	[31..0]	
+0x084		IncOffsetY	[31..0]	
+0x088		MinHeight	[31..0]	
+0x08C		IncHeight	[31..0]	
+0x090		TotalSizeY	[31..0]	
+0x094		OffsetX	[31..0]	
+0x098		Width	[31..0]	
+0x09C		OffsetY	[31..0]	
+0x0A0		Height	[31..0]	
+0x0A4~0x0BC		-	-	reserved

Table 29 – CategoryBlock2 (ImageFormatControl) (Contd.)

Offset	Name	Field	Bit	Description
+0x0C0	PixelCoding	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	not-used
		AutoInq	[2]	not-used
		ManualInq	[1]	
		NoSpecifyInq	[0]	not-used
+0x0C4		OffsetForExpanded	[31..0]	
+0x0C8		-	[31..4]	reserved
		Control	[3..0]	
+0x0CC~ 0x0D8		ListOfElements	[127..108]	not-used
			[107]	BayerBGPacked
			[106]	not-used
			[105]	BayerBG
			[104]	BayerGBPacked
			[103]	not-used
			[102]	BayerGB
			[101]	BayerRGPacked
			[100]	not-used
			[99]	BayerRG
			[98]	BayerGRPacked
			[97]	not-used
			[96]	BayerGR
			[95..83]	not-used
			[82]	YUV444Packed
			[81..75]	not-used
			[74]	YUV422Packed
			[73..67]	not-used
			[66]	YUV411Packed
			[65..35]	not-used
			[34]	RGBPacked
			[33]	RGBSigned
			[32]	RGB
			[31..3]	not-used
			[2]	MonoPacked
			[1]	MonoSigned
			[0]	Mono
+0x0DC		Value	[31..0]	

Table 29 – CategoryBlock2 (ImageFormatControl) (Contd.)

Offset	Name	Field	Bit	Description
+0x0E0	PixelSize	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	not-used
		AutoInq	[2]	not-used
		ManualInq	[1]	
		NoSpecifyInq	[0]	not-used
+0x0E4		OffsetForExpanded	[31..0]	
+0x0E8		-	[31..4]	reserved
		Control	[3..0]	
+0x0EC~ 0x0F8		ListOfElements	[127..49]	not-used
			[48]	Bpp48
			[47..25]	not-used
			[24]	Bpp24
			[23..17]	not-used
			[16]	Bpp16
			[15]	not-used
			[14]	Bpp14
			[13]	not-used
			[12]	Bpp12
			[11]	not-used
			[10]	Bpp10
			[9]	not-used
			[8]	Bpp8
			[7..0]	not-used
+0x0FC		Value	[31..0]	

Table 29 – CategoryBlock2 (ImageFormatControl) (Contd.)

Offset	Name	Field	Bit	Description
+0x100	PixelEndian	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	not-used
		AutoInq	[2]	not-used
		ManualInq	[1]	
		NoSpecifyInq	[0]	not-used
+0x104		OffsetForExpanded	[31..0]	
+0x108		-	[31..4]	reserved
		Control	[3..0]	
+0x10C~ 0x118		ListOfElements	[127..2]	not-used
			[1]	LittleEndian
			[0]	BigEndian
+0x11C		Value	[31..0]	

Table 29 – CategoryBlock2 (ImageFormatControl) (Contd.)

4.3.1 ImageFormatSelector

Device MAY have several image formats. **ImageFormatSelector** handles which format will be active. If **ImageFormatSelector** is changed, device SHALL switch attached **FeatureCSRs** (**ImageSize**, **PixelCoding PixelSize** and **PixelEndian**) for active format.

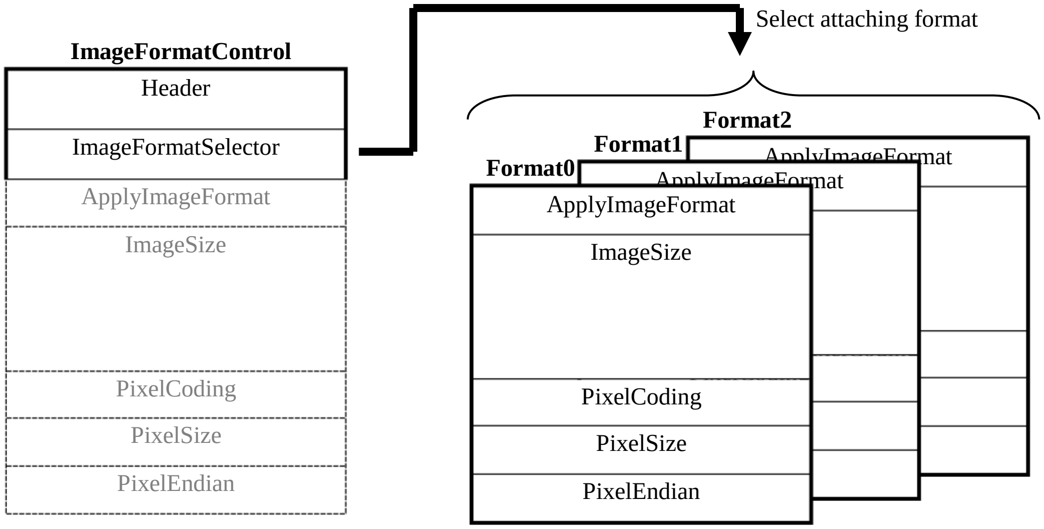

Figure 9 – ImageFormatSelector

4.3.2 ApplyImageFormat

Purpose of this **FeatureCSR** is for applying the image format setting (**ImageSize**, **PixelCoding** and **PixelSize FeatureCSRs**) to **TransportLayerControl**.

Done : Indicates image format setting is already applied. Host SHALL NOT set this **Control**.
Apply :Applies image format setting. Host MAY set this **Control**. If applying sequence is finished, device SHALL change the **Control** to **Done** or **ImageFormatError**.
Changed : Indicates image format setting was changed since last **Apply** state was executed. If image format setting is changed, device SHALL move to this **Control**. Host SHALL NOT set this **Control**.
ImageFormatError : Indicates the image format setting has an error. In applying sequence, the device SHALL move to this **Control** if there is an incorrect value in the image format setting. And the device SHALL move **Changed** if image format setting is changed. Host SHALL NOT set this **Control**.

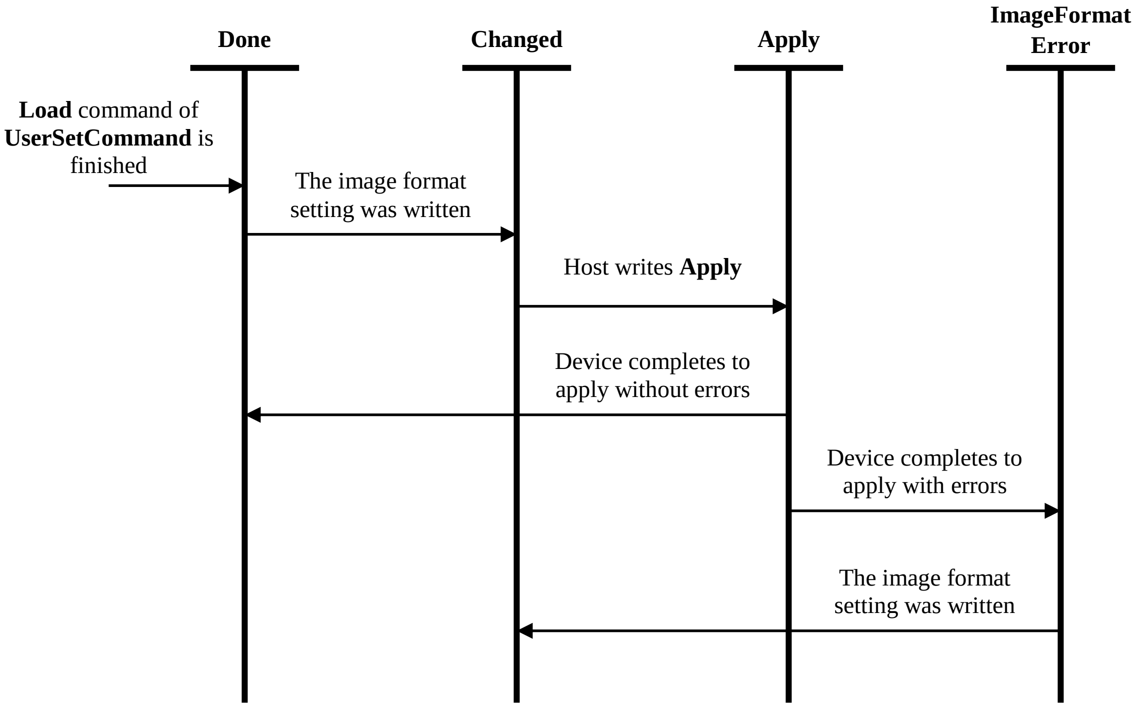
Done
Changed
Apply
ImageFormat Error
Load command of UserSetCommand is finished
The image format setting was written
Host writes Apply
Device completes to apply without errors
Device completes to apply with errors
The image format setting was written

Figure 10 – State machine of ApplyImageFormat

If device supports immediate applying, device MAY implement **Done** and **ImageFormatError** command only as read-only register (**Writable** is set to zero).

4.3.3 ImageSize

Specifies the Image size.

4.3.4 PixelCoding / PixelSize

These features describe available pixel format capability of the device. It is called **PixelFormat** virtually. Available combination of these and corresponding **PixelFormat** are as followed.

PixelCoding	PixelSize	PixelFormat
Mono (= **0**)	Bpp8 (= **8**)	Mono8
MonoPacked (= **2**)	BPP12 (= **12**)	Mono12Packed
Mono (= **0**)	Bpp16 (= **16**)	Mono16
MonoSigned (= **1**)	Bpp16 (= **16**)	MonoSigned16
YUV411Packed (= **66**)	BPP12 (= **12**)	YUV411Packed
YUV422Packed (= **74**)	Bpp16 (= **16**)	YUV422Packed
YUV444Packed (= **82**)	Bpp24 (= **24**)	YUV444Packed
RGBPacked (= **34**)	Bpp24 (= **24**)	RGB8Packed
RGB (= **32**)	Bpp48 (= **48**)	RGB16
RGBSigned (= **33**)	Bpp48 (= **48**)	RGBSigned16
BayerGR (= **96**)	Bpp8 (= **8**)	BayerGR8
BayerGRPacked (= **98**)	BPP12 (= **12**)	BayerGR12Packed
BayerGR (= **96**)	Bpp16 (= **16**)	BayerGR16
BayerRG (= **99**)	Bpp8 (= **8**)	BayerRG8
BayerRGPacked (= **101**)	BPP12 (= **12**)	BayerRG12Packed
BayerRG (= **99**)	Bpp16 (= **16**)	BayerRG16
BayerGB (= **102**)	Bpp8 (= **8**)	BayerGB8
BayerGBPacked (= **104**)	BPP12 (= **12**)	BayerGB12Packed
BayerGB (= **102**)	Bpp16 (= **16**)	BayerGB16
BayerBG (= **105**)	Bpp8 (= **8**)	BayerBG8
BayerBGPacked (= **107**)	BPP12 (= **12**)	BayerBG12Packed
BayerBG (= **105**)	Bpp16 (= **16**)	BayerBG16

Table 30 – Pixel Coding, PixelSize and corresponding PixelFormat

PixelCoding is primary setting value, and **PixelSize** is secondary. Range of **PixelSize** is limited by **PixelCoding**. Host SHALL set **PixelCoding** first when it wants to change **PixelFormat**.

4.3.5 PixelEndian

Selects Endian of pixel data.

4.4 CategoryBlock3 (AcquisitionControl)

Offset	Name	Field	Bit	Description
+0x000	Header	CategoryBlockNumber	[31..24]	= 3
		SizeOfCategoryBlock	[23..0]	= 0x000038
+0x004~ 0x01C		-	-	reserved
+0x020	Acquisition Command	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		ControlInq	[15..0]	not-used
+0x024		OffsetForExpanded	[31..0]	
+0x028		Control	[31..0]	not-used
+0x02C~ 0x038		ListOfElements	[127..17]	not-used
			[16]	Retransmit
			[15..11]	not-used
			[10]	ImageBufferRead
			[9]	MultiFrame
			[8]	Continuous
			[7..2]	not-used
			[1]	Stop
			[0]	Abort
+0x03C		Value	[31..0]	
+0x040	Acquisition FrameCount	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x30 (Integer32)
		ControlInq	[15..0]	not-used
+0x044		OffsetForExpanded	[31..0]	
+0x048		Control	[31..0]	not-used
+0x04C		Mult	[31..0]	not-used
+0x050		Div	[31..0]	not-used
+0x054		Min	[31..0]	
+0x058		Max	[31..0]	
+0x05C		Value	[31..0]	

Table 31 – CategoryBlock3 (AcquisitionControl)

Offset	Name	Field	Bit	Description
+0x060	ImageBuffer Mode	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		ControlInq	[15..0]	not-used
+0x064		OffsetForExpanded	[31..0]	
+0x068		Control	[31..0]	not-used
+0x06C~ 0x078		ListOfElements	[127..2]	not-used
			[1]	On
			[0]	Off
+0x07C		Value	[31..0]	
+0x080	ImageBuffer FrameCount	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	= 0 (read-only)
		ValueType	[23..16]	= 0x30 (Integer32)
		ControlInq	[15..0]	not-used
+0x084		OffsetForExpanded	[31..0]	
+0x088		Control	[31..0]	not-used
+0x08C		Mult	[31..0]	not-used
+0x090		Div	[31..0]	not-used
+0x094		Min	[31..0]	= 0 (Always 0)
+0x098		Max	[31..0]	
+0x09C		Value	[31..0]	
+0x0A0	Acquisition FrameRate	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x30 (Integer32)
		ControlInq	[15..5]	not-used
		DefaultInq	[4]	
		AutoOnceInq	[3]	not-used
		AutoInq	[2]	not-used
		ManualInq	[1]	
		NoSpecifyInq	[0]	

Table 31 – CategoryBlock3 (AcquisitionControl) (Contd.)

Offset	Name	Field	Bit	Description
+0x0A4	Acquisition FrameRate	OffsetForExpanded	[31..0]	
+0x0A8		-	[31..4]	reserved
		Control	[3..0]	
+0x0AC		Mult	[31..0]	
+0x0B0		Div	[31..0]	
+0x0B4		Min	[31..0]	
+0x0B8		Max	[31..0]	
+0x0BC		Value	[31..0]	
+0x0C0~ 0x0DC	Acquisition Frame Interval	Same Structure as AcquisitionFrameRate		

Table 31 – CategoryBlock3 (AcquisitionControl) (Contd.)

4.4.1 AcquisitionCommand

Handles to capture image and transfer image data. Image Buffer feature MAY be also controlled.

Stop : Stop capturing and transferring data after current transmitting frame is finished.
Abort: Abort capturing and transferring data immediately.
Continuous : Start capturing and transferring data continuously.
MultiFrame : Start capturing and transferring multiple frames. The number of capturing / transferring frames are defined by **AcquisitionFrameCount**. After transmission is finished, this field turns to **Stop** or **Abort** automatically.
ImageBufferRead: Transfer image data from Image Buffer. It is available until **ImageBufferMode** = On. The number of transferring frames are defined by **AcquisitionFrameCount**. After transmission is finished, this field turns to **Stop** or **Abort** automatically.
Retransmit : Retransmit image data transferred last. After transmission is finished, this field turns to **Stop** or **Abort** automatically.

4.4.2 AcquisitionFrameCount

Sets number of capturing / transferring frames.

4.4.3 ImageBufferMode

Handles Image Buffer. This feature is related to **AcquisitionCommand**.

Off : Disable Image Buffer feature.
On : Capture and store image data into Image Buffer.

The block diagram of **AcquisitionCommand** and **ImageBufferMode** is shown as following figure.

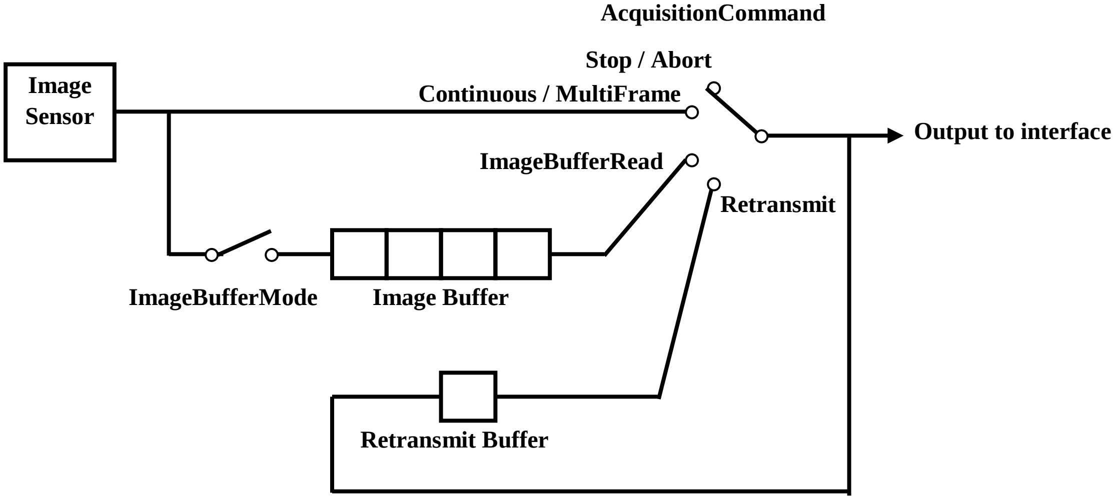

Figure 11 – Block diagram of AcquisitionCommand and ImageBufferMode

In the example shown in the figure below, point **A** is a write request to **ImageBufferMode** enabling the image buffer function. Subsequently two images are acquired and stored in the device. Point **B** is a write request to the **ImageBufferRead** command with **AcquisitionFrameCount** set to two. The device then transfers the two images on the bus from the Image Buffer. Point **C** is a write request to the **Retransmit** command. The device then retransmits the last image.

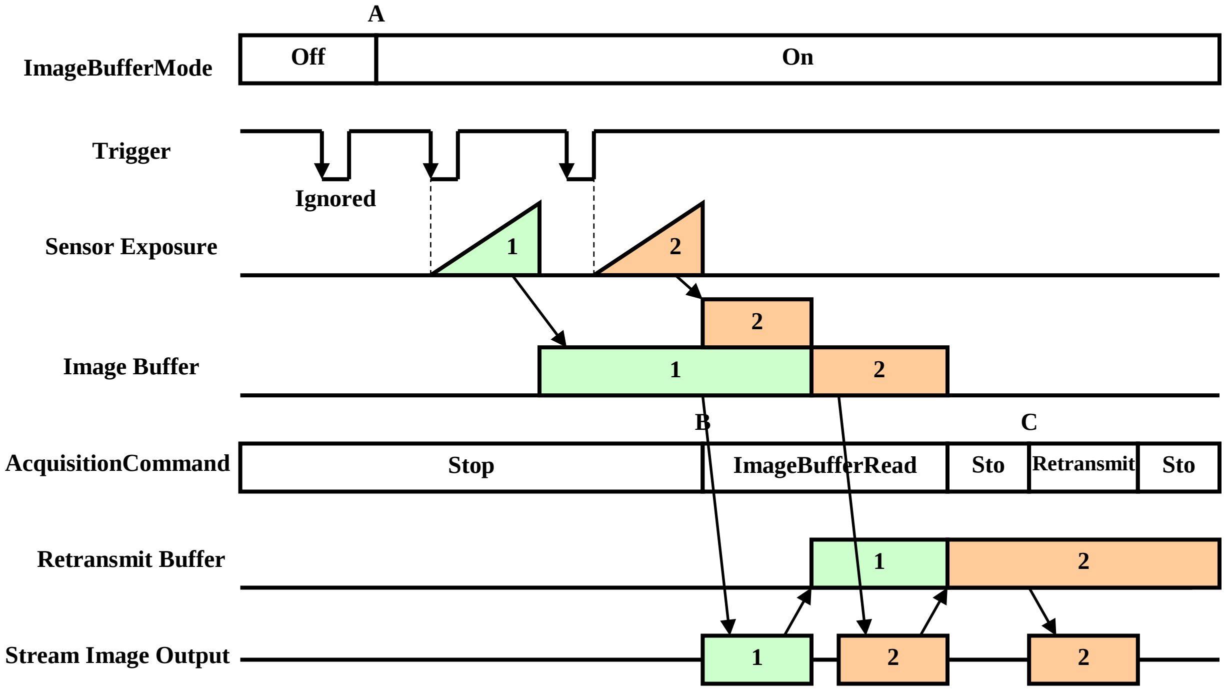

Figure 12 – Example timing of image buffer control

4.4.4 ImageBufferFrameCount

Indicates information about the image buffer. **Max** is the maximum frame number to be stored in Image Buffer. **Value** represents the number of frames in the Image Buffer now.

4.4.5 AcquisitionFrameRate

Handles frame rate of acquisition and transfer.
When **Control** is **Manual**, device SHALL limit **Max** in **ExposureTime** depending on **Value** in this **FeatureCSR**. Host SHOULD check **Max** in **ExposureTime** when this **FeatureCSR** is changed. If **Value** in **ExposureTime** is out of range, device MAY change it to **Max** automatically.
Device SHALL update **Max** and **Min** if **FeatureCSRs** in **CategoryBlock2** (**ImageFormatControl**) are changed. Host SHOULD check **Max** and **Min** after these **FeatureCSRs** are changed.
When **Control** is **NoSpecify**, **Value** SHOULD indicate available frame rate which depends on other **FeatureCSRs** including **ExposureTime**. If it is impossible, device SHALL turn **Readable** to **0**.

Unit of absolute value		fps (Frame Per Second)
Reference point		-
Recommended value at	NoSpecify	**Max** (**ExposureTime** is not limited.)

4.4.6 AcquisitionFrameInterval

It is defined as the reciprocal of **AcquisitionFrameRate**.

Unit of absolute value		second
Reference point		-
Recommended value at	NoSpecify	**Min** (**ExposureTime** is not limited.)

4.5 CategoryBlock4 (LuminanceControl)

Offset	Name	Field	Bit	Description
+0x000	Header	CategoryBlockNumber	[31..24]	= 4
		SizeOfCategoryBlock	[23..0]	= 0x00038
+0x004~ 0x01C		-	-	reserved
+0x020	Exposure Time	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	=0x30 (Integer32)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	
		AutoInq	[2]	
		ManualInq	[1]	
		NoSpecifyInq	[0]	
+0x024		OffsetForExpanded	[31..0]	
+0x028		-	[31..4]	reserved
		Control	[3..0]	
+0x02C		Mult	[31..0]	
+0x030		Div	[31..0]	
+0x034		Min	[31..0]	
+0x038		Max	[31..0]	
+0x03C		Value	[31..0]	
+0x040~ 0x05C	BlackLevel	Same Structure as ExposureTime		
+0x060~ 0x07C	Gain	Same Structure as ExposureTime		
+0x080~ 0x09C	Gamma	Same Structure as ExposureTime		
+0x0A0~ 0x0BC	Sharpness	Same Structure as ExposureTime		
+0x0C0~ 0x0DC	ALCLevel	Same Structure as ExposureTime		

Table 32 – CategoryBlock4 (LuminanceControl)

4.5.1 ExposureTime

Integration time of the incoming light.

Unit of absolute value		second
Reference point		-
Behavior of Control	AutoOnce / Auto	Proper value device calculated with **ALCLevel**
	NoSpecify	Maximum value to keep frame rate

4.5.2 BlackLevel

Black level adjustment of the image.

Unit of absolute value		%
Reference point		0
Recommended value at	AutoOnce / Auto	Proper value device calculated with **ALCLevel**
	NoSpecify	Factory setting value

4.5.3 Gain

Gain control for image.

Unit of absolute value		dB
Reference point		0
Recommended value at	AutoOnce / Auto	Proper value device calculated with **ALCLevel**
	NoSpecify	Factory setting value

4.5.4 Gamma

Define the function between incoming light level and output picture level.

$Y = X^{Gamma}$ Y : output picture level, X : incoming light level

Unit of absolute value		power-law
Reference point		1.0
Recommended value at	AutoOnce / Auto	Proper value device calculated
	NoSpecify	Factory setting value

4.5.5 Sharpness

Sharpness of the image.

Unit of absolute value		-
Reference point		0 (as disable)
Recommended value at	AutoOnce / Auto	Proper value device calculated
	NoSpecify	Factory setting value

4.5.6 ALCLevel

Target level for Auto Luminance Control.
When either **ExposureTime**, **BlackLevel** or **Gain** is set to **Auto** or **AutoOnce**, device handles these features automatically in order to get appropriate level of image.

Unit of absolute value		EV (Exposure Value)
Reference point		0
Recommended value at	AutoOnce / Auto	Proper value device calculated
	NoSpecify	Factory setting value

4.6 CategoryBlock5 (ChromaControl)

Offset	Name	Field	Bit	Description
+0x000	Header	CategoryBlockNumber	[31..24]	= 5
		SizeOfCategoryBlock	[23..0]	= 0x000040
+0x004~ 0x01C		-	-	reserved
+0x020	Hue	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x30 (Integer32)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	
		AutoInq	[2]	
		ManualInq	[1]	
		NoSpecifyInq	[0]	
+0x024		OffsetForExpanded	[31..0]	
+0x028		-	[31..4]	reserved
		Control	[3..0]	
+0x02C		Mult	[31..0]	
+0x030		Div	[31..0]	
+0x034		Min	[31..0]	
+0x038		Max	[31..0]	
+0x03C		Value	[31..0]	
+0x040~ 0x05C	Saturation	Same Structure as Hue		
+0x060~ 0x07C	WhiteBalance R	Same Structure as Hue		
+0x080~ 0x09C	WhiteBalance B	Same Structure as Hue		
+0x0A0~ 0x0BC	WhiteBalance U	Same Structure as Hue		
+0x0C0~ 0x0DC	WhiteBalance V	Same Structure as Hue		
+0x0E0~ 0x0FC	Color Temperature	Same Structure as Hue		

Table 33 – CategoryBlock5 (ChromaControl)

4.6.1 Hue

Color phase of the image.

Unit of absolute value		degree
Reference point		0
Recommended value at	AutoOnce / Auto	Proper value device calculated
	NoSpecify	Factory setting value

4.6.2 Saturation

Color saturation of the image.

Unit of absolute value		%
Reference point		0
Recommended value at	AutoOnce / Auto	Proper value device calculated
	NoSpecify	Factory setting value

4.6.3 WhiteBalanceR / WhiteBalanceB

Adjustment of the white color of the picture. **WhiteBalanceR** handles red plane, and **WhiteBalanceB** is Blue. **Value** is relative setting from green plane.
If the device cannot be set to **Auto** / **AutoOnce** separately, the device SHALL refer **Control** fields to each other.

Unit of absolute value		dB
Reference point		0
Recommended value at	AutoOnce / Auto	Proper value device calculated
	NoSpecify	Factory setting value

4.6.4 WhiteBalanceU / WhiteBalanceV

Adjustment of the white color of the picture. **WhiteBalanceU** handles chrominance red, and **WhiteBalanceV** is chrominance Blue. **Value** is offset level.

If the device cannot be set to **Auto** / **AutoOnce** separately, the device SHALL refer **Control** fields to each other.

Unit of absolute value		%
Reference point		0
Recommended value at	AutoOnce / Auto	Proper value device calculated
	NoSpecify	Factory setting value

4.6.5 ColorTemperature

Adjustment of the color temperature of the picture.

Unit of absolute value		Kelvin
Reference point		0
Recommended value at	AutoOnce / Auto	Proper value device calculated
	NoSpecify	Factory setting value

4.7 CategoryBlock6 (LUTControl)

Offset	Name	Field	Bit	Description
+0x000	Header	CategoryBlockNumber	[31..24]	= 6
		SizeOfCategoryBlock	[23..0]	=0x000020
+0x004~ 0x01C		-	-	reserved
+0x020	LUT Enable	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	not-used
		AutoInq	[2]	not-used
		ManualInq	[1]	
		NoSpecifyInq	[0]	not-used
+0x024		OffsetForExpanded	[31..0]	
+0x028		-	[31..4]	reserved
		Control	[3..0]	
+0x02C~ 0x038		ListOfElements	[127..2]	not-used
			[1]	On
			[0]	Off
+0x03C		Value	[31..0]	
+0x040	LUT Bank Selector	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	not-used
		AutoInq	[2]	not-used
		ManualInq	[1]	
		NoSpecifyInq	[0]	not-used
+0x044		OffsetForExpanded	[31..0]	
+0x048		-	[31..4]	reserved
		Control	[3..0]	

Table 34 – CategoryBlock6 (LUTControl)

Offset	Name	Field	Bit	Description
+0x04C~ 0x058	LUT Bank Selector	ListOfElements	[127..32]	not-used
			[31]	Bank31
			...	...
			[1]	Bank1
			[0]	Bank0
+0x05C		Value	[31..0]	
+0x060	LUTValue All	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		ControlInq	[15..0]	not-used
+0x064		OffsetForExpanded	[31..0]	
+0x068		Control	[31..0]	not-used
+0x06C~ 0x078		ListOfElements	[127..32]	not-used
			[31]	Preset31
			...	...
			[1]	Preset1
			[0]	Preset0
+0x07C		Value	[31..0]	

Table 34 – CategoryBlock6 (LUTControl) (Contd.)

4.7.1 LUTEnable

Handles LUT function.

On : Enable LUT function. LUT number is selected by **LUTBankSelector** .
Off : Disable LUT function.

4.7.2 LUTBankSelector

Selects the bank of LUT.

4.7.3 LUTValueAll

This feature SHALL be implemented as **ExpandedCSR** pointed by **OffsetForExpanded** when LUT value is either readable or writable.
It is defined by **ArrayOfInteger32** with 2-dimension.

Offset	Name	Field	Bit	Description
+0x000	LUTValue All	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0xB0 (ArrayOfInteger32)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	
		AutoInq	[2]	
		ManualInq	[1]	
		NoSpecifyInq	[0]	
+0x004		OffsetForExpanded	[31..0]	
+0x008		-	[31..4]	reserved
		Control	[3..0]	
+0x00C		MoreDimension	[31]	= 1
		NumberOfChannels	[30..0]	Number of channels (Cn)
+0x010		MoreDimension	[31]	= 0
		NumberOfElements	[30..0]	Number of elements per a bank (En)
+0x014		Mult	[31..0]	
+0x018		Div	[31..0]	
+0x01C		Min	[31..0]	
+0x020		Max	[31..0]	
+0x024		Value[0][0]	[31..0]	
+0x028		Value[0][1]	[31..0]	
...		....	[31..0]	
+0xXXX		Value[0][En-1]	[31..0]	
+0xXXX +4		Value[1][0]	[31..0]	
...		...		
+0xYYY		Value[Cn-1][En-1]	[31..0]	

Table 35 – LUTValueAll in ExpandedCategoryBlock (chaining CSR)

4.8 CategoryBlock7 (TriggerControl)

Offset	Name	Field	Bit	Description
+0x000	Header	CategoryBlockNumber	[31..24]	= 7
		SizeOfCategoryBlock	[23..0]	= 0x000038
+0x004~ 0x01C		-	-	reserved
+0x020	Trigger Mode	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	not-used
		AutoInq	[2]	not-used
		ManualInq	[1]	
		NoSpecifyInq	[0]	not-used
+0x024		OffsetForExpanded	[31..0]	
+0x028		-	[31..4]	reserved
		Control	[3..0]	
+0x02C~ 0x038		ListOfElements	[127..2]	not-used
			[1]	On
			[0]	Off
+0x03C		Value	[31..0]	
+0x040	Trigger Sequence	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	not-used
		AutoInq	[2]	not-used
		ManualInq	[1]	
		NoSpecifyInq	[0]	not-used
+0x044		OffsetForExpanded	[31..0]	
+0x048		-	[31..4]	reserved
		Control	[3..0]	

Table 36 – CategoryBlock7 (TriggerControl)

Offset	Name	Field	Bit	Description
+0x04C~ 0x058	Trigger Sequence	ListOfElements	[127..6]	not-used
			[5]	TriggerSequence5
			[4]	TriggerSequence4
			[3]	TriggerSequence3
			[2]	TriggerSequence2
			[1]	TriggerSequence1
			[0]	TriggerSequence0
+0x05C		Value	[31..0]	
+0x060	Trigger Source	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	not-used
		AutoInq	[2]	not-used
		ManualInq	[1]	
		NoSpecifyInq	[0]	not-used
+0x064		OffsetForExpanded	[31..0]	
+0x068		-	[31..4]	reserved
		Control	[3..0]	
+0x06C~ 0x078		ListOfElements	[127..65]	not-used
			[64]	SoftwareTrigger
			[63..32]	not-used
			[31]	IOLine31
			...	...
			[1]	IOLine1
			[0]	IOLine0
+0x07C		Value	[31..0]	

Table 36 – CategoryBlock7 (TriggerControl) (Contd.)

Offset	Name	Field	Bit	Description
+0x080	Trigger Additional Parameter	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x30 (Integer32)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	not-used
		AutoInq	[2]	not-used
		ManualInq	[1]	
		NoSpecifyInq	[0]	not-used
+0x084		OffsetForExpanded	[31..0]	
+0x088		-	[31..4]	reserved
		Control	[3..0]	
+0x08C		Mult	[31..0]	
+0x090		Div	[31..0]	
+0x094		Min	[31..0]	
+0x098		Max	[31..0]	
+0x09C		Value	[31..0]	
+0x0A0	Trigger Delay	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x30 (Integer32)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	
		AutoInq	[2]	
		ManualInq	[1]	
		NoSpecifyInq	[0]	
+0x0A4		OffsetForExpanded	[31..0]	
+0x0A8		-	[31..4]	reserved
		Control	[3..0]	
+0x0AC		Mult	[31..0]	
+0x0B0		Div	[31..0]	
+0x0B4		Min	[31..0]	
+0x0B8		Max	[31..0]	
+0x0BC		Value	[31..0]	

Table 36 – CategoryBlock7 (TriggerControl) (Contd.)

Offset	Name	Field	Bit	Description
+0x0C0	Software Trigger	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	not-used
		AutoInq	[2]	not-used
		ManualInq	[1]	
		NoSpecifyInq	[0]	not-used
+0x0C4		OffsetForExpanded	[31..0]	
+0x0C8		-	[31..4]	reserved
		Control	[3..0]	
+0x0CC~ 0x0D8		ListOfElements	[127..9]	not-used
			[8]	Impulse
			[7..2]	not-used
			[1]	Active
			[0]	Inactive
+0x0DC		Value	[31..0]	

Table 36 – CategoryBlock7 (TriggerControl) (Contd.)

4.8.1 TriggerMode

Selects the trigger mode for acquiring image.

Off : Acquiring image by normal operation.
On : Acquiring image by external trigger mode.

4.8.2 TriggerSequence

Selects the trigger sequence.

Sequence0 : External Edge mode

Device starts integration of the incoming light from external trigger input falling edge. Integration time is described in **ExposureTime**. **TriggerAdditionalParameter** is not used.

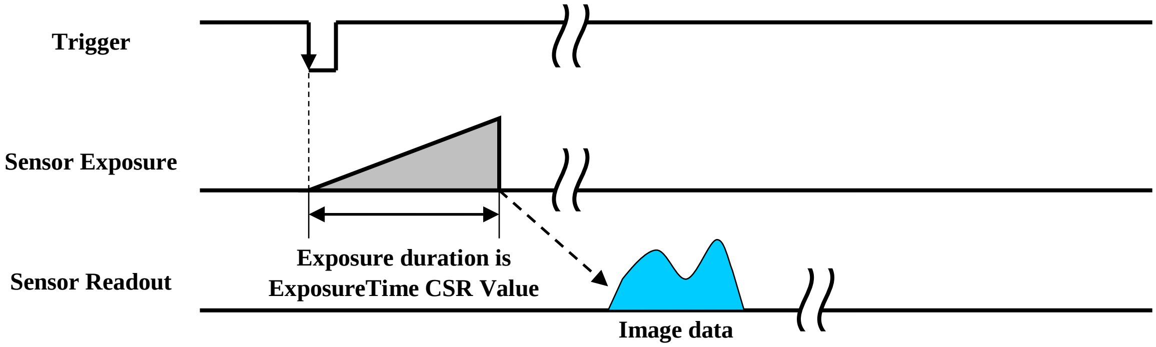

Figure 13 – Trigger Sequence0

Sequence1 : External level mode

Device starts integration of the incoming light from external trigger input falling edge. Integration time is equal to low state time of the external trigger input. **ExposureTime** and **TriggerAdditionalParameter** is not used.

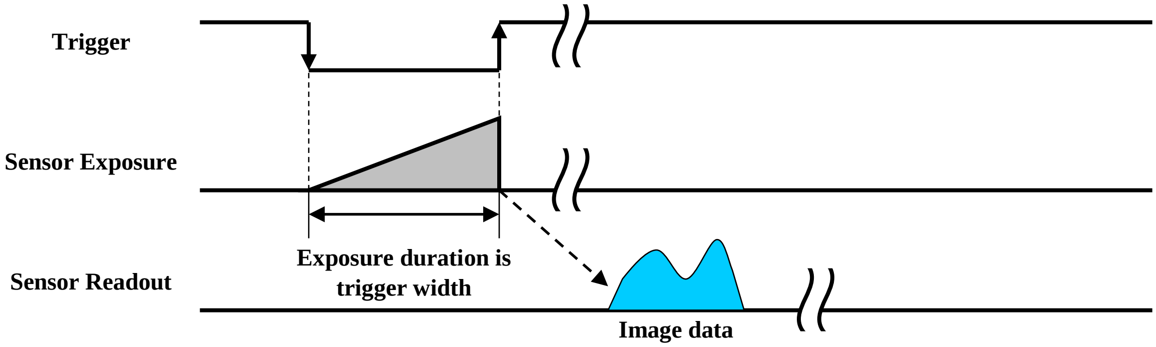

Figure 14 – Trigger Sequence1

Sequence2 : External event mode

Device starts integration of incoming light from first external trigger input falling edge. At the N-th (define by **TriggerAdditionalParameter**) external trigger input falling edge, integration will be stopped. **TriggerAdditionalParameter** is required and SHALL be two or more. (N >= 2)

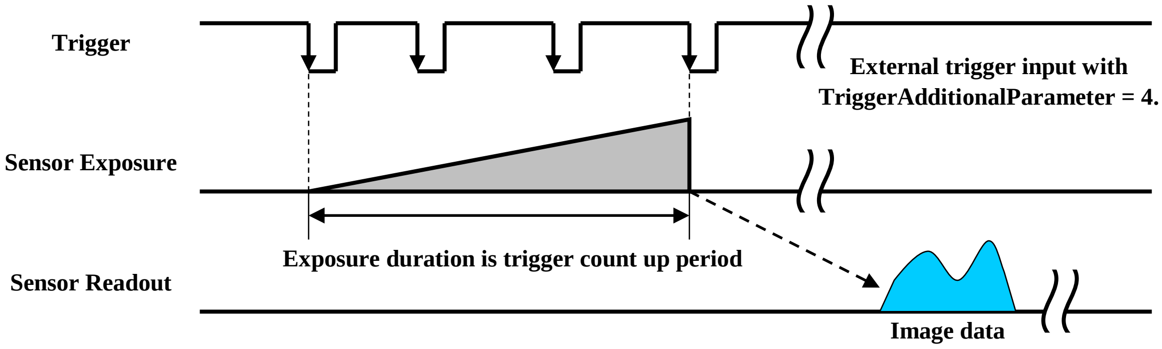
Trigger
Sensor Exposure
Sensor Readout
External trigger input with
TriggerAdditionalParameter = 4.
Exposure duration is trigger count up period
Image data

Figure 15 – Trigger Sequence2

Sequence3 : Frame Interval mode

This is an internal trigger sequence. Device will issue trigger internally and internal trigger cycle time is **TriggerAdditionalParameter** times of the minimum frame interval. Integration time of incoming light is described in **ExposureTime**. **TriggerAdditionalParameter** is required and SHALL be one or more. (N >= 1)

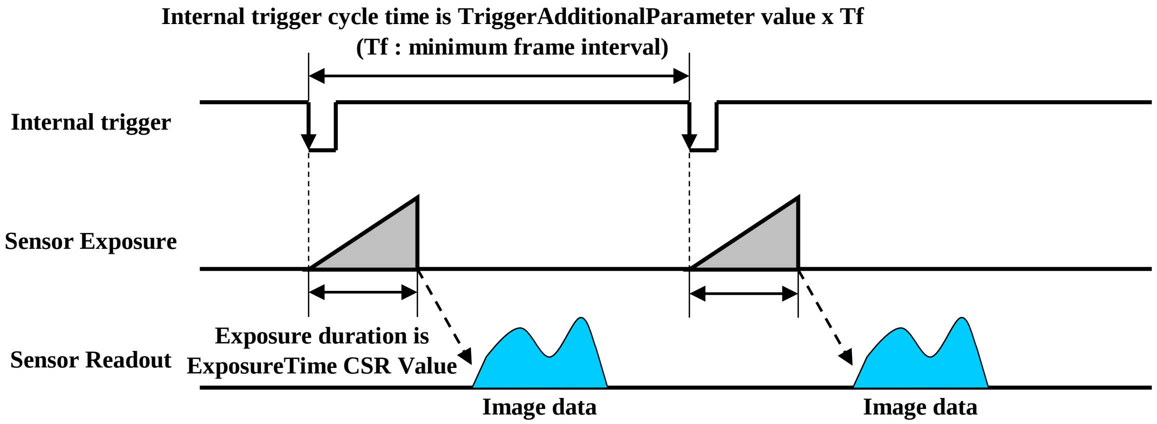

Figure 16 – Trigger Sequence3

Sequence4 : Multiple Shutter Preset mode

Device starts integration of incoming light from first external trigger input falling edge and exposes incoming light at **ExposureTime**. Repeat this sequence the N-th (define by **TriggerAdditionalParameter**) external trigger input falling edge then finish integration. **TriggerAdditionalParameter** is required and SHALL be one or more. (N >= 1)

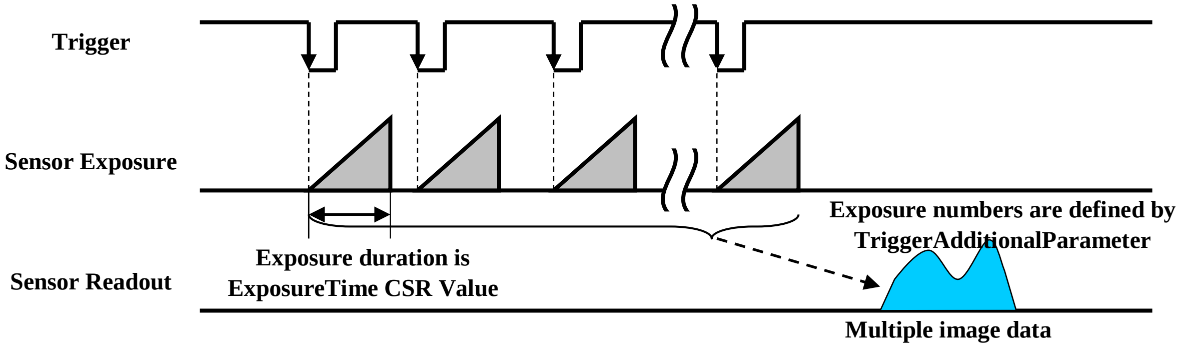
Trigger
Sensor Exposure
Sensor Readout
Exposure numbers are defined by TriggerAdditionalParameter
Exposure duration is ExposureTime CSR Value
Multiple image data

Figure 17 – Trigger Sequence4

Sequence5 : Multiple Shutter Pulse Width mode

Device starts integration of incoming light from first external trigger input falling edge and exposes incoming light until trigger is inactive. Repeat this sequence the N-th (defined by **TriggerAdditionalParameter**) external trigger input falling edge then finish integration. **TriggerAdditionalParameter** is required and SHALL be one or more. (N >= 1)

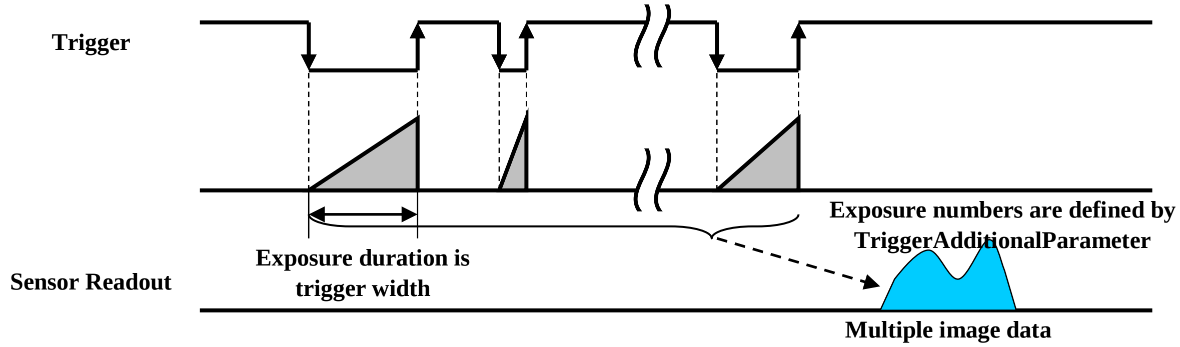
Trigger
Sensor Readout
Exposure numbers are defined by TriggerAdditionalParameter
Exposure duration is trigger width
Multiple image data

Figure 18 – Trigger Sequence5

4.8.3 TriggerSource

Selects trigger source at the external trigger mode. External trigger is always active-lo. Device and host MAY use invert signal with handling **IOLineInverterAll** in **CategoryBlock9** (**DigitalIOControl**).

4.8.4 TriggerAdditionalParameter

Trigger parameter at the external trigger mode if needed (It depends on **TriggerSequence**).

Unit of absolute value		-
Reference point		-
Recommended value at	AutoOnce / Auto	Proper value device calculated
	NoSpecify	Factory setting value

4.8.5 TriggerDelay

Add internal delay of trigger signal.

Unit of absolute value		second
Reference point		0
Recommended value at	AutoOnce / Auto	Proper value device calculated
	NoSpecify	Factory setting value

4.8.6 SoftwareTrigger

Trigger input at the software trigger mode.

Inactive : Set software trigger to inactive
Active : Set software trigger to active.
Impulse: Input the impulse for software trigger. After the impulse is received, device SHALL turn to **Inactive** automatically.

4.9 CategoryBlock8 (UserSetControl)

Offset	Name	Field	Bit	Description
+0x000	Header	CategoryBlockNumber	[31..24]	= 8
		SizeOfCategoryBlock	[23..0]	= 0x000028
+0x004~ 0x01C		-	-	reserved
+0x020	Standard Format	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		ControlInq	[15..0]	not-used
+0x024		OffsetForExpanded	[31..0]	
+0x028		Control	[31..0]	not-used
+0x02C~ 0x038		ListOfElements	[127..43]	not-used
			[42]	HD1080p_YUV422Packed
			[41]	HD1080p_RGB8Packed
			[40]	HD1080p_Mono8
			[39..35]	not-used
			[34]	UXGA_YUV422Packed
			[33]	UXGA_RGB8Packed
			[32]	UXGA_Mono8
			[31..27]	not-used
			[26]	SXGA_YUV422Packed
			[25]	SXGA_RGB8Packed
			[24]	SXGA_Mono8
			[23..19]	not-used
			[18]	HD720p_YUV422Packed
			[17]	HD720p_RGB8Packed
			[16]	HD720p_Mono8
			[15..11]	not-used
			[10]	XGA_YUV422Packed
			[9]	XGA_RGB8Packed
			[8]	XGA_Mono8
			[7..3]	not-used
			[2]	VGA_YUV422Packed
			[1]	VGA_RGB8Packed
			[0]	VGA_Mono8
+0x03C		Value	[31..0]	

Table 37 – CategoryBlock8 (UserSetControl)

Offset	Name	Field	Bit	Description
+0x040	Standard FrameRate	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		ControlInq	[15..0]	not-used
+0x044		OffsetForExpanded	[31..0]	
+0x048		Control	[31..0]	not-used
+0x04C~ 0x058		ListOfElements	[127..8]	not-used
			[7]	240 fps
			[6]	120 fps
			[5]	60 fps
			[4]	30 fps
			[3]	15 fps
			[2]	7.5 fps
			[1]	3.75 fps
			[0]	1.875 fps
+0x05C		Value	[31..0]	
+0x060	UserSet Selector	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		ControlInq	[15..0]	not-used
+0x064		OffsetForExpanded	[31..0]	
+0x068		Control	[31..0]	not-used
+0x06C~ 0x078		ListOfElements	[127..32]	not-used
			[31]	UserSet31
			...	...
			[1]	UserSet0
			[0]	Default
+0x07C		Value	[31..0]	
+0x080	UserSet Command	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		ControlInq	[15..0]	not-used

Table 37 – CategoryBlock8 (UserSetControl) (Contd.)

Offset	Name	Field	Bit	Description

+0x084	UserSet Command	OffsetForExpanded	[31..0]	
+0x088		Control	[31..0]	not-used
+0x08C~ 0x098		ListOfElements	[127..10]	not-used
			[9]	Save
			[8]	Load
			[7..1]	not-used
			[0]	Done
+0x09C		Value	[31..0]	

Table 37 – CategoryBlock8 (UserSetControl) (Contd.)

4.9.1 StandardFormat / StandardFrameRate

Support settings for well-known image formats. Host MAY receive image format with same setting. The device MAY limit **StandardFrameRate** by **StandardFormat**.

Device supporting these **FeatureCSRs** has the table of register set for each image format. If host writes these **FeatureCSRs**, device SHALL apply register set to corresponding **FeatureCSRs** (**ImageSize**, **PixelCoding**, **PixelSize**, **AcquisitionFrameRate**, **AcquisitionFrameInterval** and **FeatureCSRs** in **TransportLayerControl**) immediately.

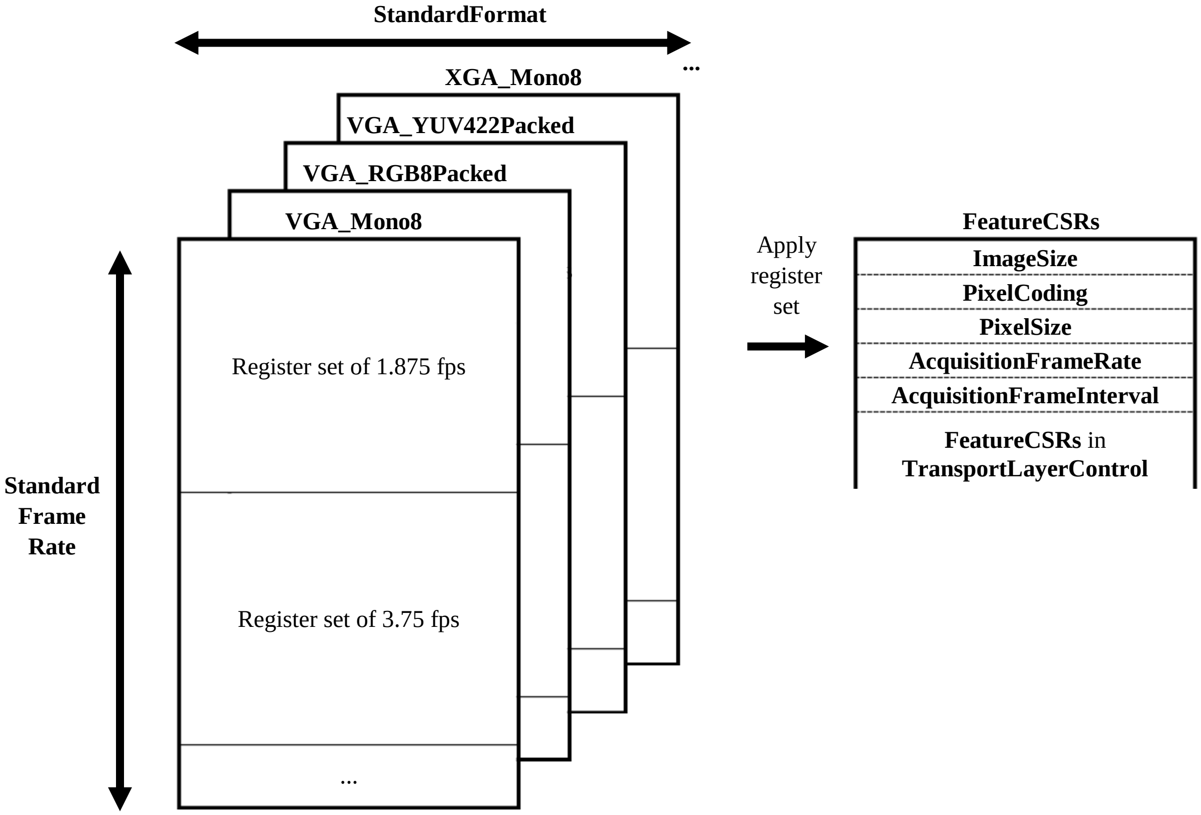

Figure 19 – StandardFormat / StandardFrameRate

Relationship between **ImageSize** and type of **StandardFormat** is as follows.

Type of StandardFormat	ImageSize	
	Width	height
VGA	640	480
XGA	1024	768
HD720p	1280	720
SXGA	1280	960
UXGA	1600	1200
HD1080p	1920	1080

Table 38 –Relationship between ImageSize and type of StandardFormat

Actual values of **AcquisitionFrameRate** and **AcquisitionFrameInterval** depend on the device (Absolute values are same as **StandardFrameRate**).
About **FeatureCSRs** in **TransportLayerControl**, please see section 6 Transport layer of IEEE1394 (for IEEE1394) or each specification of transport layer.

4.9.2 UserSetSelector / UserSetCommand

Handle non-volatile memory in the device. **UserSetSelector** selects the target page, and **UserSetCommand** handles execution of memory command.
Default is the factory setting page. **UserSet1**, **UserSet2**... are user setting pages.

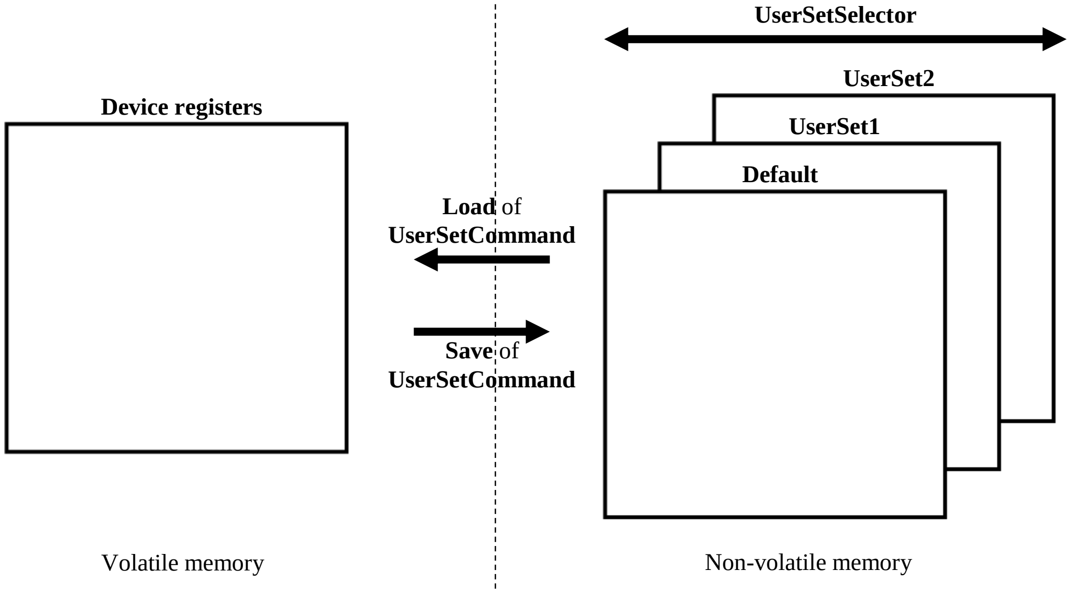

Figure 20 – UserSetSelector / UserSetCommand

4.10 CategoryBlock9 (DigitalIOControl)

Offset	Name	Field	Bit	Description
+0x000	Header	CategoryBlockNumber	[31..24]	= 9
		SizeOfCategoryBlock	[23..0]	= 0x000038
+0x004~ 0x01C		-	-	reserved
+0x020	IOLine ModeAll	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x34 (BulkBoolean32)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	not-used
		AutoInq	[2]	not-used
		ManualInq	[1]	
		NoSpecifyInq	[0]	not-used
+0x024		OffsetForExpanded	[31..0]	
+0x028		-	[31..4]	reserved
		Control	[3..0]	
+0x02C		BitWritable	[31..0]	
+0x030		Value	[31..0]	
+0x034~ 0x03C		-	-	Reserved
+0x040	IOLine InverterAll	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x34 (BulkBoolean32)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	not-used
		AutoInq	[2]	not-used
		ManualInq	[1]	
		NoSpecifyInq	[0]	not-used
+0x044		OffsetForExpanded	[31..0]	
+0x048		-	[31..4]	reserved
		Control	[3..0]	

Table 39 – CategoryBlock9 (DigitalIOControl)

Offset	Name	Field	Bit	Description
+0x04C	IOLine InverterAll	BitWritable	[31..0]	
+0x050		Value	[31..0]	
+0x054~ 0x05C		-	-	Reserved
+0x060	IOLine StatusAll	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	= 0 (read-only)
		Readable	[24]	
		ValueType	[23..16]	= 0x34 (BulkBoolean32)
		ControlInq	[15..0]	not-used
+0x064		OffsetForExpanded	[31..0]	
+0x068		Control	[31..0]	not-used
+0x06C		BitWritable	[31..0]	not-used
+0x070		Value	[31..0]	
+0x074~ 0x07C		-	-	Reserved
+0x080	UserOutput ValueAll	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x34 (BulkBoolean32)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	not-used
		AutoInq	[2]	not-used
		ManualInq	[1]	
		NoSpecifyInq	[0]	not-used
+0x084		OffsetForExpanded	[31..0]	
+0x088		-	[31..4]	reserved
		Control	[3..0]	
+0x08C		BitWritable	[31..0]	
+0x090		Value	[31..0]	
+0x094~ 0x09C		-	-	Reserved

Table 39 – CategoryBlock9 (DigitalIOControl) (Contd.)

Offset	Name	Field	Bit	Description
+0x0A0	IOLine Selector	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		ControlInq	[15..0]	not-used
+0x0A4		OffsetForExpanded	[31..0]	
+0x0A8		Control	[31..4]	not-used
+0x0AC~ 0x0B8		ListOfElements	[127..32]	not-used
			[31]	IOLine31
			...	...
			[1]	IOLine1
			[0]	IOLine0
+0x0BC		Value	[31..0]	
+0x0C0	IOLine Source	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	not-used
		AutoInq	[2]	not-used
		ManualInq	[1]	
		NoSpecifyInq	[0]	not-used
+0x0C4		OffsetForExpanded	[31..0]	
+0x0C8		-	[31..4]	reserved
		Control	[3..0]	
+0x0CC~ 0x0D8		ListOfElements	[127..96]	not-used
			[95]	Timer31Active
			...	...
			[65]	Timer1Active
			[64]	Timer0Active
			[63..33]	not-used
			[32]	UserOutput
			[31..1]	not-used
			[0]	Off
+0x0DC		Value	[31..0]	

Table 39 – CategoryBlock9 (DigitalIOControl) (Contd.)

DigitalIOControl handles all I/O lines which the device has (including Trigger inputs). The following image is a block diagram of **DigitalIOControl**. Devices have this logic for each I/O lines.

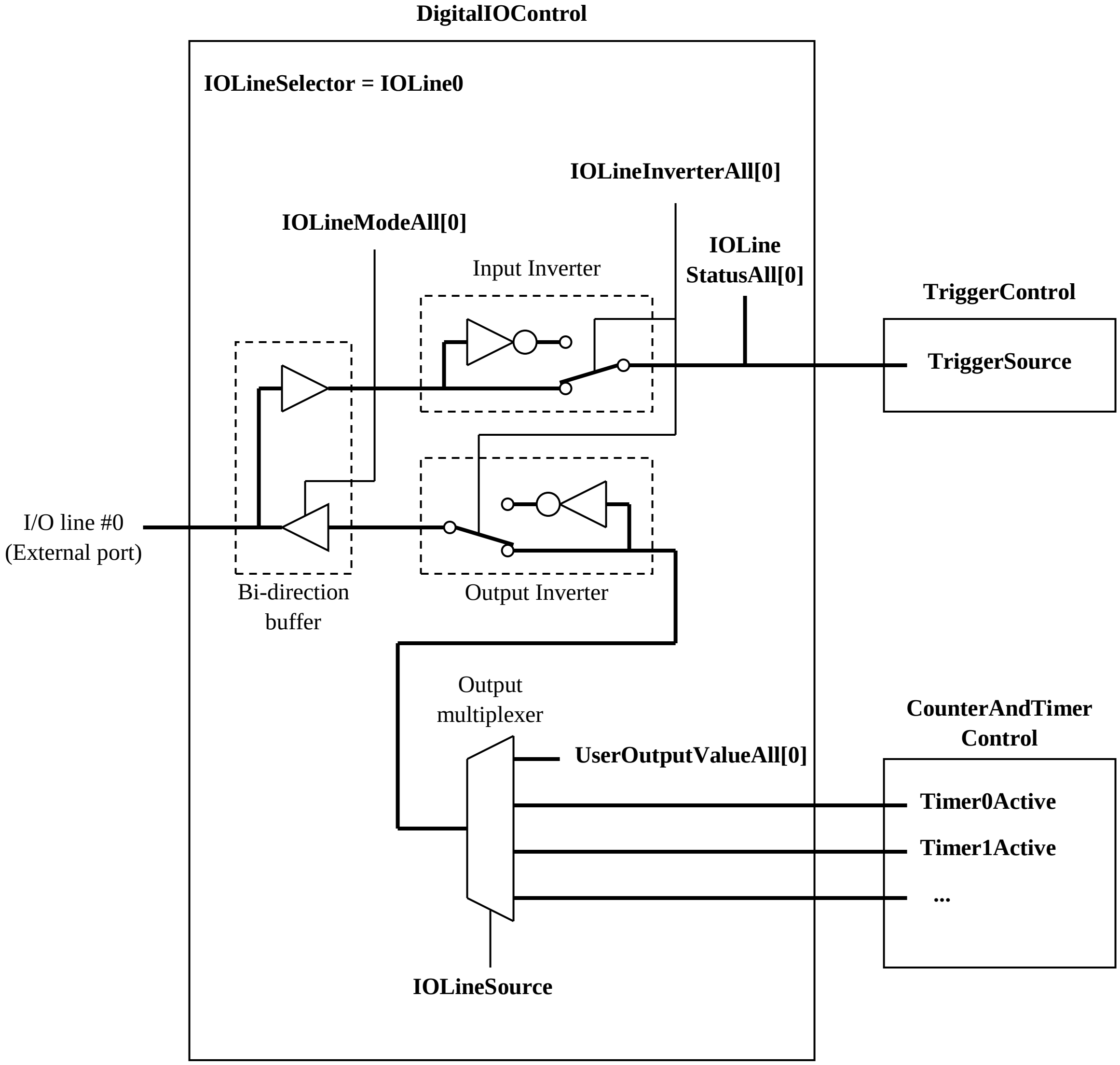

Figure 21 – Block diagram of DigitalIOControl

4.10.1 IOLineModeAll

Handles direction of I/O lines.

0 : Input
1 : Output

If device has fixed direction of I/O line, device SHALL turn **BitWritable** to **0** and fix Value in each corresponding bit location.

4.10.2 IOLineInverterAll

Handles the inversion of I/O lines. Device SHALL reflect it to both input and output buffers.

0 : Not inverted
1 : Inverted

4.10.3 IOLineStatusAll

Indicates the current status of all I/O lines.

0 :Low
1 : High

4.10.4 UserOutputValueAll

Sets the internal register of all user outputs.

0 : Low
1 : High

4.10.5 IOLineSelector

Selects the I/O line to handle. It is reflected to **IOLineSource**.

4.10.6 IOLineSource

Selects which source signals to connect I/O line. The number of I/O line is selected by **IOLineSelector**. Host SHALL set bits corresponding I/O lines which are required to output in **IOLineModeAll** to **1** (Output).

Off : Output is disabled. Device SHALL set I/O line to high-impedance.
UserOutput : Current user output value which is handled by **UserOutputValueAll** (bit position is same as **IOLine** number).
Timer0Active, **Timer1Active1**,...:Current **TimerValue** which is handled by **FeatureCSRs** in **CounterAndTimerControl**

4.11 CategoryBlock10 (CounterAndTimerControl)

Offset	Name	Field	Bit	Description
0x000	Header	CategoryBlockNumber	[31..24]	= 10
		SizeOfCategoryBlock	[23..0]	= 0x000020
+0x004~ 0x01C		-	-	reserved
+0x020	Timer Selector	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		ControlInq	[15..0]	not-used
+0x024		OffsetForExpanded	[31..0]	
+0x028		Control	[31..0]	not-used
+0x02C~ 0x038		ListOfElements	[127..32]	not-used
			[31]	Timer31
			...	...
			[1]	Timer1
			[0]	Timer0
+0x03C		Value	[31..0]	
+0x040	Timer Delay	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x30 (Integer32)
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	
		AutoInq	[2]	
		ManualInq	[1]	
		NoSpecifyInq	[0]	
+0x044		OffsetForExpanded	[31..0]	
+0x048		-	[31..4]	reserved
		Control	[3..0]	
+0x04C		Mult	[31..0]	
+0x050		Div	[31..0]	
+0x054		Min	[31..0]	
+0x058		Max	[31..0]	
+0x05C		Value	[31..0]	

Table 40 – CategoryBlock10 (CounterAndTimerControl)

Offset	Name	Field	Bit	Description
+0x060~ 0x07C	Timer Duration	Same structure as TimerDelay		

Table 40 – CategoryBlock10 (CounterAndTimerControl) (Contd.)

4.11.1 TimerSelector

Selects the Timer to handle. It is reflected to **TimerDelay** and **TimerDuration**.

4.11.2 TimerDelay / TimerDuration

Handles waveform of **TimerActive** signals. Device MAY connect it to output multiplexer of **DigitalIOControl**. For example, it is used for strobe control. Timer is started from exposure start.

Unit of absolute value		second
Reference point		-
Recommended value at	AutoOnce / Auto	Proper value device calculated
	NoSpecify	Factory setting value

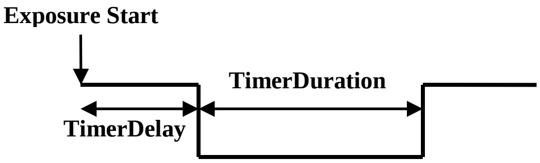

Figure 22 – Definition of TimerActive signal

4.12 CategoryBlock31 (VendorUniqueControl)

Offset	Name	Field	Bit	Description
+0x000	Header	CategoryBlockNumber	[31..24]	= 31
		SizeOfCategoryBlock	[23..0]	
+0x004~ 0x01C		-	-	reserved
+0x020	Feature CSR0	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	
		AutoInq	[2]	
		ManualInq	[1]	
		NoSpecifyInq	[0]	
+0x024		OffsetForExpanded	[31..0]	
+0x028		-	[31..4]	reserved
		Control	[3..0]	
+0x02C~ 0x03C		ValueRegisterArea		
+0x040~ 0x05C	Feature CSR1			
...	...	...		

Table 41 – CategoryBlock31 (VendorUniqueControl)

VendorUniqueControl provides vendor unique control that is not related to other **BasicCSRs**. Each **FeatureCSR** has the following constraints:

- Length of **FeatureCSR** SHALL be in multiples of 8 quadlets. If the actual length is less than 8 quadlets, the remaining bytes SHALL be reserved. If length is longer than 8 quadlets, the bits corresponding to **Implemented** of each **ValueRegisterArea** after 2nd 8 quadlets SHALL be kept to zeros.

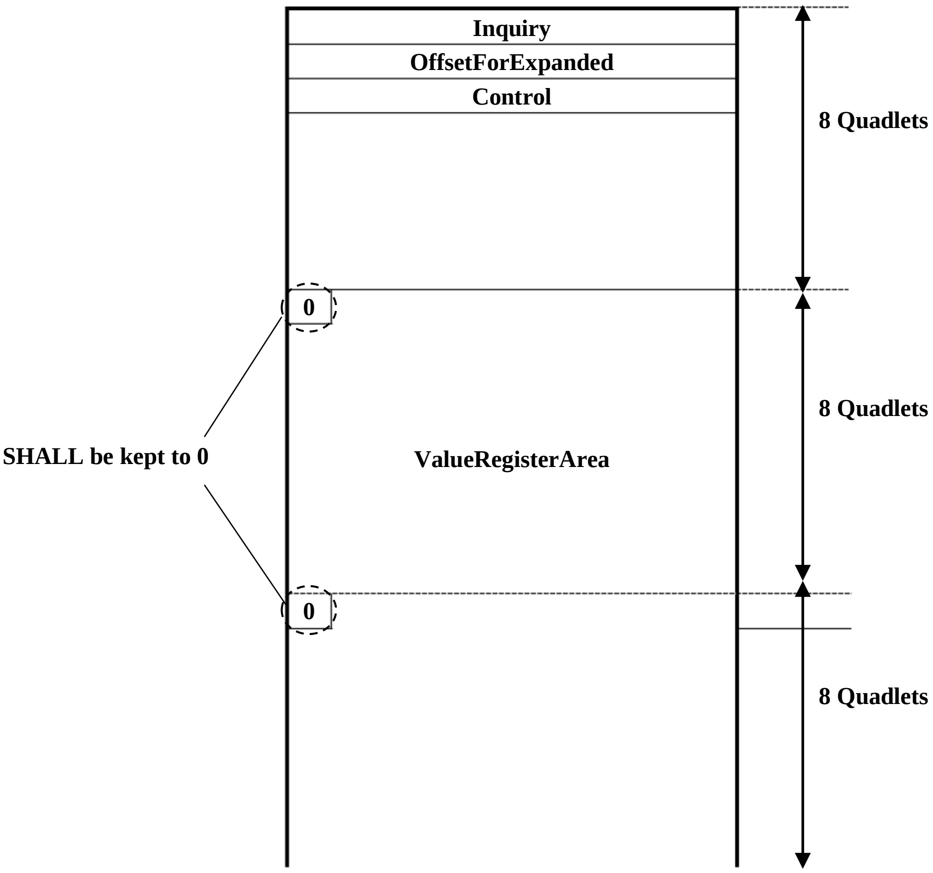

Figure 23 – FeatureCSR longer than 8 quadlets

- **ValueType** SHALL be **Integer32**, **PlainInteger8**, **PlainInteger32**, **PlainInteger64**, **Float32**, **Enumeration**, **BulkBoolean32** , **BulkBoolean64**, **Rectangle32**, or **String** (**ArrayOfPlainInteger8**).

- These **FeatureCSRs** SHALL NOT be chained from other **FeatureCSRs**. These **FeatureCSRs** MAY be chained to **ExpandedCSRs**.

4.13 ExpandedCategoryBlock

Offset	Name	Field	Bit	Description
0x000	Header	CategoryBlockNumber.	[31..24]	32 or more
		SizeOfCategoryBlock.	[23..0]	
+0x004~ 0x01C		-	-	reserved
+0x020	Feature CSR0	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	
		-	[15..5]	reserved
		DefaultInq	[4]	
		AutoOnceInq	[3]	
		AutoInq	[2]	
		ManualInq	[1]	
		NoSpecifyInq	[0]	
+0x024		OffsetForExpanded	[31..0]	
+0x028		-	[31..4]	reserved
		Control	[3..0]	
+0x02C~		ValueRegisterArea		
+0xXXX~	Feature CSR1			
...	...	...		

Table 42 – ExpandedCategoryBlock

ExpandedCategoryBlock is used for expanding **BasicCategoryBlocks**. Each **FeatureCSR** has the following constraints:

- Length of **FeatureCSR** is not fixed. It is up to (2^{24} – 8 (as header)) Quadlets.
- **ValueType** MAY use all types of register including array.
- These **FeatureCSRs** SHALL be chained from other **FeatureCSRs** (**BasicCSRs** or **ExpandedCSRs**). **FeatureCSRs** MAY be chained to **ExpandedCSRs** (SHALL NOT be chained to **BasicCSRs**).

5 FeatureCSR Chaining

5.1 Use case of common logic

FeatureCSR chaining MAY be used for the single hardware logic which has several input ports. The following figure is an example of a device that has two **FeatureCSRs** which handle common gain amplifier hardware logic.

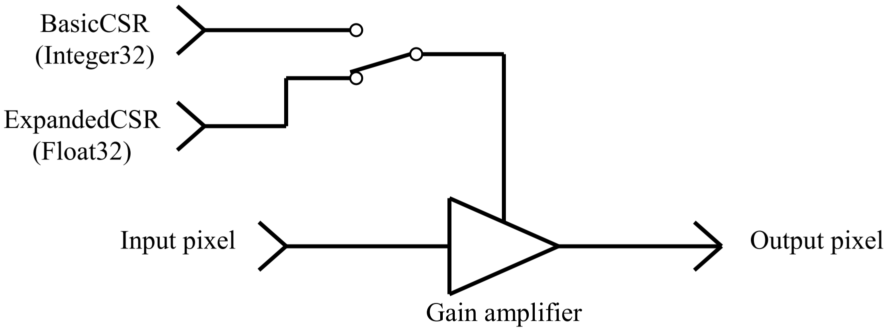

Figure 24 – Gain amplifier which has Integer32 / Float32 register

5.1.1 Value

If either one's **Value** is written, device SHOULD reflect **Value** to the other's, otherwise device SHALL change **Readable** to **0** in the **FeatureCSR** whose **Value** is not correct.

5.1.2 Control

If either one' is **Control** written, device SHALL reflect **Control** to the other's.

5.2 Use case of independent logics

FeatureCSR chaining MAY be used for the several hardware logics. The following figure is an example of a device that has two separate gain amplifier hardware logics.

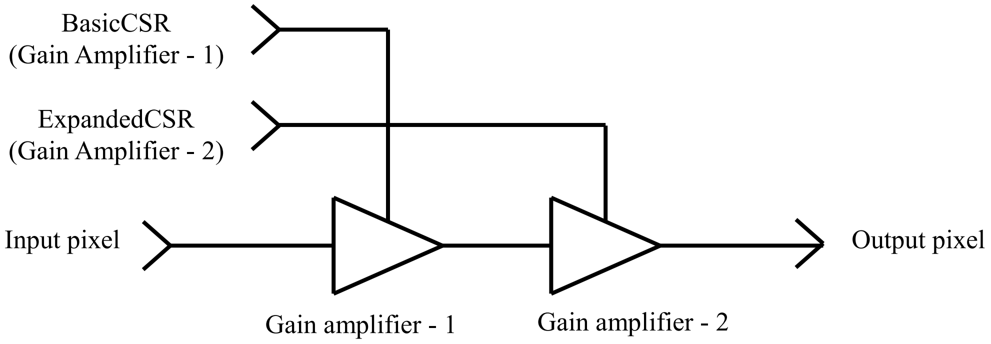

Figure 25 –2 gain amplifier logics

5.2.1 Value

These are independent.

5.2.2 Control

These are independent.

6 Transport layer of IEEE1394

This section is for IEEE1394 only. If an interface other than IEEE1394 is used, please see the transport layer specification for that specific technology.

6.1 Require about bus management

6.1.1 For the devices

The device is neither Isochronous manager capable nor full bus manager capable. The device is also not cycle master capable. The contents of the self_ID packet generated by the device, and the contents of device configuration ROM SHALL accurately reflect this level of capability.

6.1.2 For the Hosts

The host SHALL execute for the following activities related to device operation:

- Force a cycle master capable node to be the root
- Start cycle master operation
- Initialize the **IIDC2 FeatureCSRs** in the device for a desired video mode, frame rate, etc.
- Allocate Isochronous resources needed by the device (Isochronous channel number and bandwidth, as needed for the selected video mode)
- Program the Isochronous channel number and transmit speed into the FeatureCSRs in the device
- Instruct the device to start sourcing image stream data

6.2 Packet transmission format

6.2.1 Packet for accessing IIDC2 register area

For accessing **IIDC2** register area, Asynchronous packets are used - there are Write request for data quadlet, Write request for data block, Read request for data quadlet, Read request for data block, Read response for data quadlet, Read response for data block, Lock request and Lock response packets. The device and host SHALL support these packet transactions.

The device SHALL indicate the maximum payload size to “max_rec” field of the Configuration ROM.

6.2.2 Packet for image stream

For transmitting image stream data, Isochronous data block packet is used. The following table shows the format of the first quadlet in the data field of each Isochronous data block.

<table>
<tr><td colspan="2">31-24</td><td colspan="2">23-16</td><td colspan="2">15-8</td><td colspan="2">7-0</td></tr>
<tr><td colspan="4">data_length</td><td>tg</td><td>channel</td><td>tCode</td><td>sy</td></tr>
<tr><td colspan="8">header_CRC</td></tr>
<tr><td colspan="8">Image stream data payload</td></tr>
<tr><td colspan="8">data_CRC</td></tr>
</table>

Figure 26 – Isochronous data block packet Format

Where the following fields are defined in the IEEE 1394 standard:

data_length : number of bytes in the data field
tg : (tag field) SHALL be set to zero
channel : isochronous channel number, as assigned in the **FwIsochronousChannel**
tCode : (transaction code) SHALL be set to the isochronous data block packet tCode
sy : (synchronization value) SHALL be set to 0x1 on the first isochronous data block of a frame, and SHALL be set to zero on all other isochronous data blocks

6.3 Discovery of device

6.3.1 IEEE 1394 Specific Address Space

The device SHALL be compliant with the IEEE 1394 and IEEE 1212 standards.

The following sections define all CSR and ROM locations that the device SHALL implement. All information in these sections is intended to comply with the IEEE 1394 standard. Where discrepancies arise, the IEEE 1394 standard SHALL prevail.

All address-offset locations in these sections are with respect to a base address of:

0xFFFF F000 0000

6.3.2 Implemented CSR's

The device implements the following core CSR's, as required by the IEEE 1394 standard:

Offset	31-24	23-16	15-8	7-0
+0x0000	STATE_CLEAR			
+0x0004	STATE_SET			
+0x0008	NODE_IDS			
+0x000C	RESET_START			
+0x0010				
+0x0014				
+0x0018	SPLIT_TIMEOUT_HI			
+0x001C	SPLIT_TIMEOUT_LO			

Table 43 – Core CSR's

The device implements the following IEEE 1394 Serial Bus dependent CSR's:

Offset	31-24	23-16	15-8	7-0
+0x0200	CYCLE_TIME			
+0x0204				
+0x0208				
+0x020C				
+0x0210	BUSY_TIMEOUT			

Table 44 – Serial Bus Dependent CSR's

6.3.3 Configuration ROM

IEEE 1394 devices SHALL implement a Configuration ROM as defined in IEEE standard 1212-1991, IEEE standard 1394-2008. These assignments are only an example. The Key codes describe the role of the ROM entry thereby making the order of ROM entries implementation dependent. Please see IEEE std 1212 specification in detail.

unit_sw_version = 0x00 0110 (for IIDC2 Digital Camera Control Specification)

	Offset	31-24	23-16	15-8	7-0
Bus Info Block	+0x400	0x04	crc_length	rom_crc_value	
	+0x404	0x31	0x33	0x39	0x34
	+0x408	0 0 1 0 rsv	0xFF	max_rec rsv mx rom	gen r lk_spd
	+0x40C	node_vendor_id			chip_id_hi
	+0x410	chip_id_lo			
Root Directory	+0x414	0x0003		CRC	
	+0x418	0x03	module_vendor_ID		
	+0x41C	0x0C	rsv	1 0 0 0 0 0 1 1	1 1 0 0 0 0 0 0
	+0x420	0xD1	unit_directory offset		

Table 45 – Root Directory

	Offset	31-24	23-16	15-8	7-0
Unit Directory	0000h	0x0003		CRC	
	0004h	0x12	unit_spec_ID (= 0x00 A02D)		
	0008h	0x13	unit_sw_version (= 0x00 0110)		
	000Ch	0xD4	unit_dependent_directory offset		

Table 46 – Unit Directory

	Offset	31-24	23-16	15-8	7-0
Unit Dependent Info	+0x0000	0x004		CRC	
	+0x0004	0x40	IIDC2Entry		
	+0x0008	0x81	vendor_name_leaf		
	+0x000C	0x82	model_name_leaf		
	+0x0010	0x38	unit_sub_sw_version		

Table 47 – Unit Dependent Directory

Where:

IIDC2Entry is the quadlet offset of **IIDC2Entry** defined in section 3 Digital device control register of this standard from the base address.

vendor_name_leaf specifies the number of quadlets from the address of the **vendor_name_leaf** entry to the address of the **vendor_name** leaf containing an ASCII representation of the vendor name of this node.

model_name_leaf specifies the number of quadlets from the address of the **model_name_leaf** entry to the address of the **model_name** leaf containing an ASCII representation of the model name of this node.

unit_sub_sw_version specifies the version of IIDC2. It is same value of **Version** in **IIDC2Entry** (0x01 0000).

If the device supports both IIDC2 and IIDC 1.32 (or earlier), The Root Directory is as shown in Table 48.

	Offset	31-24	23-16	15-8	7-0
Bus Info Block	+0x400	0x04	crc_length	rom_crc_value	
	+0x404	0x31	0x33	0x39	0x34
	+0x408	0 0 1 0 rsv	0xFF	max_rec rsv mx rom	gen r lk_spd
	+0x40C	node_vendor_id			chip_id_hi
	+0x410	chip_id_lo			
Root Directory	+0x414	0x0004		CRC	
	+0x418	0x03	module_vendor_ID		
	+0x41C	0x0C	rsv	1 0 0 0 0 0 1 1	1 1 0 0 0 0 0 0
	+0x420	0xD1	unit_directory offset (for IIDC 1.32)		
	+0x0424	0xD1	unit_directory_offset (for IIDC2)		

Table 48 – Root Directory in case of dual protocol

For details about IIDC 1.32 Unit or other directories please see IIDC 1394-based Digital Camera Specification Ver.1.32.

6.3.4 Format of Vendor Name and Model Name leaves

The unit dependent directory MAY contain pointers to information leaves that contain the ASCII name of the vendor and model name for this node. The format of these leaves is shown in the following table:

	Offset	31-24	23-16	15-8	7-0
Name Leaf	+0x0000	leaf_length		CRC	
	+0x0004	0x00	0x00 0000		
	+0x0008	0x0000 0000			
	+0x000C	char_0	char_1	char_2	char_3
	+0x0010	char_4	char_5	char_6	char_7
	+0x0014	char_8	...		
	+(n+0x6)	...			char_n-3
	+(n+0xA)	char_n-2	Char_n-1	NUL	NUL

Table 49 – Vendor Name / Model Name leaves

6.4 Condition of IIDC2

6.4.1 Endianness

IEEE1394 uses Big Endian.

6.4.2 Supporting GenICam

GenICam is Supported through **XmlManifestTable** in **IIDC2**.

6.5 CategoryBlock1 (TransportLayerControl) for IEEE1394

Offset	Name	Field	Bit	Description
+0x000	Header	CategoryBlockNumber	[31..24]	= 1
		SizeOfCategoryBlock	[23..0]	= 0x000028
+0x004~ 0x01C		-	-	reserved
+0x020	Fw Isochronous Speed	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x03 (Enumeration)
		ControlInq	[15..0]	not-used
+0x024		OffsetForExpanded	[31..0]	
+0x028		Control	[31..0]	not-used
+0x02C~ 0x038		ListOfElements	[127..6]	not-used
			[5]	S3200
			[4]	S1600
			[3]	S800
			[2]	S400
			[1]	S200
			[0]	S100
+0x03C		Value	[31..0]	
+0x040	Fw Isochronous PacketSize	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x30 (Integer32)
		ControlInq	[15..0]	not-used
+0x044		OffsetForExpanded	[31..0]	
+0x048		Control	[31..0]	not-used
+0x04C		Mult	[31..0]	Inc PacketSize (These field SHALL be used as Inc only)
+0x050		Div	[31..0]	
+0x054		Min	[31..0]	Minimum PacketSize
+0x058		Max	[31..0]	Maximum PacketSize
+0x05C		Value	[31..0]	

Table 50 – TransportLayerControl

Offset	Name	Field	Bit	Description
+0x060	Fw Isochronous Channel	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	
		Readable	[24]	
		ValueType	[23..16]	= 0x30 (Integer32)
		ControlInq	[15..0]	not-used
+0x064		OffsetForExpanded	[31..0]	
+0x068		Control	[31..0]	not-used
+0x06C		Mult	[31..0]	not-used
+0x070		Div	[31..0]	not-used
+0x074		Min	[31..0]	= 0
+0x078		Max	[31..0]	= 63
+0x07C		Value	[31..0]	
+0x080	Fw Isochronous TotalBytes	Implemented	[31]	
		Active	[30]	
		-	[29..27]	reserved
		UserSetLoadable	[26]	
		Writable	[25]	= 0 (read-only)
		Readable	[24]	
		ValueType	[23..16]	= 0x41 (PlainInteger64)
		ControlInq	[15..0]	not-used
+0x084		OffsetForExpanded	[31..0]	
+0x088		Control	[31..0]	not-used
+0x08C		Value	[63..0]	
+0x090				
+0x094~ 0x09C		-	-	reserved

Table 50 – TransportLayerControl (Contd.)

CategoryBlock1 (**TransportLayerControl**) of IEEE1394 provides the setting of image stream data. If device is a camera it SHOULD support the **FeatureCSRs** specified in this section.

6.5.1 FwIsochronousSpeed

Selects speed for isochronous transfer.

6.5.2 FwIsochronousPacketSize

Sets isochronous packet size. Before setting this feature, host SHOULD set **ImageSize**, **PixelCoding** and **PixelSize**. Device SHALL update **Max** when **FwIsochronousSpeed** is changed.

6.5.3 FwIsochronousChannel

Selects isochronous channel number.

6.5.4 FwIsochronousTotalBytes

Indicates total bytes of image data per frame.

6.6 Image Stream Data

6.6.1 Image Stream Data payload structure

Pn : Pixel number / packet

K : $Pn \times n$ (n = 0..N-1)

($Pn \times N$ = Total pixel number / frame.)

6.6.2 Image stream payload format

6.6.2.1 **Mono8 format**

Each component has 8-bit data.

Y[K+0]	Y[K+1]	Y[K+2]	Y[K+3]
Y[K+4]	Y[K+5]	Y[K+6]	Y[K+7]
Y[K+Pn-8]	Y[K+Pn-7]	Y[K+Pn-6]	Y[K+Pn-5]
Y[K+Pn-4]	Y[K+Pn-3]	Y[K+Pn-2]	Y[K+Pn-1]

Figure 27 – Mono8 format

6.6.2.2 **Mono12Packed format**

Each Y component has 12-bit data.

<table>
<tr><td>High byte</td><td>Low nibble</td></tr>
</table>

<table>
<tr><td colspan="2">Y[K+0]
High byte</td><td>Y[K+1]
Low nibble</td><td>Y[K+0]
Low nibble</td><td colspan="2">Y[K+1]
High byte</td><td colspan="2">Y[K+2]
High byte</td></tr>
<tr><td>Y[K+3]
Low nibble</td><td>Y[K+2]
Low nibble</td><td colspan="2">Y[K+3]
High byte</td><td colspan="2">Y[K+4]
High byte</td><td>Y[K+5]
Low nibble</td><td>Y[K+4]
Low nibble</td></tr>
<tr><td colspan="2">Y[K+5]
High byte</td><td colspan="2">Y[K+6]
High byte</td><td>Y[K+7]
Low nibble</td><td>Y[K+6]
Low nibble</td><td colspan="2">Y[K+7]
High Byte</td></tr>
<tr><td colspan="8"></td></tr>
<tr><td>Y[K+Pn-5]
Low nibble</td><td>Y[K+Pn-6]
Low nibble</td><td colspan="2">Y[K+Pn-5]
High byte</td><td colspan="2">Y[K+Pn-4]
High byte</td><td>Y[K+Pn-3]
Low nibble</td><td>Y[K+Pn-4]
Low nibble</td></tr>
<tr><td colspan="2">Y[K+Pn-3]
High byte</td><td colspan="2">Y[K+Pn-2]
High byte</td><td>Y[K+Pn-1]
Low nibble</td><td>Y[K+Pn-2]
Low nibble</td><td colspan="2">Y[K+Pn-1]
High byte</td></tr>
</table>

Figure 28 – Mono12Packed format

6.6.2.3 **Mono16 format**

Each component has 16-bit data.

High byte	Low byte

Y[K+0]	Y[K+1]
Y[K+2]	Y[K+3]
Y[K+Pn-4]	Y[K+Pn-3]
Y[K+Pn-2]	Y[K+Pn-1]

Figure 29 – Mono16 Format

6.6.2.4 MonoSigined16 format

Each component has 16-bit signed integer data.

High byte	Low byte

Y[K+0]	Y[K+1]
Y[K+2]	Y[K+3]
Y[K+Pn-4]	Y[K+Pn-3]
Y[K+Pn-2]	Y[K+Pn-1]

Figure 30 – MonoSigned16 Format

6.6.2.5 YUV411Packed format

Each component has 8-bit data.

U[K+0]	Y[K+0]	Y[K+1]	V[K+0]
Y[K+2]	Y[K+3]	U[K+4]	Y[K+4]
Y[K+5]	V[K+4]	Y[K+6]	Y[K+7]
U[K+Pn-8]	Y[K+Pn-8]	Y[K+Pn-7]	V[K+Pn-8]
Y[K+Pn-6]	Y[K+Pn-5]	U[K+Pn-4]	Y[K+Pn-4]
Y[K+Pn-3]	V[K+Pn-4]	Y[K+Pn-2]	Y[K+Pn-1]

Figure 31 – YUV411Packed Format

6.6.2.6 **YUV422Packed format**

Each component has 8-bit data.

U[K+0]	Y[K+0]	V[K+0]	Y[K+1]
U[K+2]	Y[K+2]	V[K+2]	Y[K+3]
U[K+4]	Y[K+4]	V[K+4]	Y[K+5]
U[K+Pn-6]	Y[K+Pn-6]	V[K+Pn-6]	Y[K+Pn-5]
U[K+Pn-4]	Y[K+Pn-4]	V[K+Pn-4]	Y[K+Pn-3]
U[K+Pn-2]	Y[K+Pn-2]	V[K+Pn-2]	Y[K+Pn-1]

Figure 32 –YUV422Packed Format

6.6.2.7 **YUV444Packed format**

Each component has 8-bit data.

U[K+0]	Y[K+0]	V[K+0]	U[K+1]
Y[K+1]	V[K+1]	U[K+2]	Y[K+2]
V[K+2]	U[K+3]	Y[K+3]	V[K+3]
U[K+Pn-4]	Y[K+Pn-4]	V[K+Pn-4]	U[K+Pn-3]
Y[K+Pn-3]	V[K+Pn-3]	U[K+Pn-2]	Y[K+Pn-2]
V[K+Pn-2]	U[K+Pn-1]	Y[K+Pn-1]	V[K+Pn-1]

Figure 33 –YUV444Packed Format

6.6.2.8 **RGB8Packed format**

Each component has 8-bit data.

R[K+0]	G[K+0]	B[K+0]	R[K+1]
G[K+1]	B[K+1]	R[K+2]	G[K+2]
B[K+2]	R[K+3]	G[K+3]	B[K+3]
R[K+Pn-4]	G[K+Pn-4]	B[K+Pn-4]	R[K+Pn-3]
G[K+Pn-3]	B[K+Pn-3]	R[K+Pn-2]	G[K+Pn-2]
B[K+Pn-2]	R[K+Pn-1]	G[K+Pn-1]	B[K+Pn-1]

Figure 34 –RGB8Packed Format

6.6.2.9 **RGB16 format**

Each component has 16-bit data.

High byte	Low byte

R[K+0]	G[K+0]
B[K+0]	R[K+1]
G[K+1]	B[K+1]
B[K+Pn-2]	R[K+Pn-1]
G[K+Pn-1]	B[K+Pn-1]

Figure 35 –RGB16 Format

6.6.2.10 **RGBsigned16 format**

Each component has 16-bit signed integer data.

High byte	Low byte

R[K+0]	G[K+0]
B[K+0]	R[K+1]
G[K+1]	B[K+1]
B[K+Pn-2]	R[K+Pn-1]
G[K+Pn-1]	B[K+Pn-1]

Figure 36 –RGBSigned16 Format

6.6.2.11 **BayerGR8**

Each component has 8-bit data.

Even line

Gr[K+0]	R[K+1]	Gr[K+2]	R[K+3]
Gr[K+4]	R[K+5]	Gr[K+6]	R[K+7]
Gr[K+W-8]	R[K+W-7]	Gr[K+W-6]	R[K+W-5]
Gr[K+W-4]	R[K+W-3]	Gr[K+W-2]	R[K+W-1]

Odd line

B[K+0]	Gb[K+1]	B[K+2]	Gb[K+3]
B[K+4]	Gb[K+5]	B[K+6]	Gb[K+7]
B[K+W-8]	Gb[K+W-7]	B[K+W-6]	Gb[K+W-5]
B[K+W-4]	Gb[K+W-3]	B[K+W-2]	Gb[K+W-1]

Figure 37 –BayerGR8 Format

6.6.2.12 **BayerGR12Packed**

Each component has 12-bit data.

High byte	Low nibble

Even line

<table>
<tr><td colspan="3">Gr[K+0]
High byte</td><td>R[K+1]
Low nibble</td><td>Gr[K+0]
Low nibble</td><td colspan="2">R[K+1]
High byte</td><td colspan="2">Gr[K+2]
High byte</td></tr>
<tr><td colspan="2">R[K+3]
Low nibble</td><td>Gr[K+2]
Low nibble</td><td colspan="2">R[K+3]
High byte</td><td colspan="2">Gr[K+4]
High byte</td><td>R[K+5]
Low nibble</td><td>Gr[K+4]
Low nibble</td></tr>
<tr><td colspan="9"></td></tr>
<tr><td>R[K+Pn-5]
Low nibble</td><td colspan="2">Gr[K+Pn-6]
Low nibble</td><td colspan="2">R[K+Pn-5]
High byte</td><td colspan="2">Gr[K+Pn-4]
Low nibble</td><td>R[K+Pn-3]
Low nibble</td><td>Gr[K+Pn4]
Low nibble</td></tr>
<tr><td colspan="3">R[K+Pn-3]
High byte</td><td colspan="2">Gr[K+Pn-2]
Low nibble</td><td>R[K+6]
Low nibble</td><td>Gr[K+Pn-2]
Low nibble</td><td colspan="2">R[K+Pn-1]
Low nibble</td></tr>
</table>

Odd line

<table>
<tr><td colspan="3">B[K+0]
High byte</td><td>Gb[K+1]
Low nibble</td><td>B[K+0]
Low nibble</td><td colspan="2">Gb[K+1]
High byte</td><td colspan="2">B[K+2]
High byte</td></tr>
<tr><td colspan="2">Gb[K+3]
Low nibble</td><td>B[K+2]
Low nibble</td><td colspan="2">Gb[K+3]
High byte</td><td colspan="2">B[K+4]
High byte</td><td>Gb[K+5]
Low nibble</td><td>B[K+4]
Low nibble</td></tr>
<tr><td colspan="9"></td></tr>
<tr><td>Gb[K+Pn-5]
Low nibble</td><td colspan="2">B[K+Pn-6]
Low nibble</td><td colspan="2">Gb[K+Pn-5]
High byte</td><td colspan="2">B[K+Pn-4]
Low nibble</td><td>Gb[K+Pn-3]
Low nibble</td><td>B[K+Pn4]
Low nibble</td></tr>
<tr><td colspan="3">Gb[K+Pn-3]
High byte</td><td colspan="2">B[K+Pn-2]
Low nibble</td><td>Gb[K+6]
Low nibble</td><td>B[K+Pn-2]
Low nibble</td><td colspan="2">Gb[K+Pn-1]
Low nibble</td></tr>
</table>

Figure 38 –BayerGR12Packef Format

6.6.2.13 **BayerGR16**

Each component has 16-bit signed integer data.

High byte	Low byte

Even line

Gr[K+0]	R[K+1]
Gr [K+2]	R[K+3]
Gr [K+W-4]	R[K+W-3]
Gr [K+W-2]	R[K+W-1]

Odd line

B[K+0]	Gb[K+1]
B[K+2]	Gb[K+3]
B[K+W-4]	Gb[K+W-3]
B[K+W-2]	Gb[K+W-1]

Figure 39 –BayerGR16 Format

6.6.2.14 **BayerRG8**

Each component has 8-bit data.

Even line

R[K+0]	Gr[K+1]	R[K+2]	Gr[K+3]
R[K+4]	Gr[K+5]	R[K+6]	Gr[K+7]
R[K+W-8]	Gr[K+W-7]	R[K+W-6]	Gr[K+W-5]
R[K+W-4]	Gr[K+W-3]	R[K+W-2]	Gr[K+W-1]

Odd line

Gb[K+0]	B[K+1]	Gb[K+2]	B[K+3]
Gb[K+4]	B[K+5]	Gb[K+6]	B[K+7]
Gb[K+W-8]	B[K+W-7]	Gb[K+W-6]	B[K+W-5]
Gb[K+W-4]	B[K+W-3]	Gb[K+W-2]	B[K+W-1]

Figure 40 –BayerRG8 Format

6.6.2.15 **BayerRG12Packed**

Each component has 12-bit data.

<table>
<tr><td>High byte</td><td>Low nibble</td></tr>
</table>

Even line

<table>
<tr><td colspan="2">R[K+0]
High byte</td><td>Gr[K+1]
Low nibble</td><td>R[K+0]
Low nibble</td><td colspan="2">Gr[K+1]
High byte</td><td colspan="2">R[K+2]
High byte</td></tr>
<tr><td>Gr[K+3]
Low nibble</td><td>R[K+2]
Low nibble</td><td colspan="2">Gr[K+3]
High byte</td><td colspan="2">R[K+4]
High byte</td><td>Gr[K+5]
Low nibble</td><td>R[K+4]
Low nibble</td></tr>
<tr><td colspan="8"></td></tr>
<tr><td>Gr[K+Pn-5]
Low nibble</td><td>R[K+Pn-6]
Low nibble</td><td colspan="2">Gr[K+Pn-5]
High byte</td><td colspan="2">R[K+Pn-4]
Low nibble</td><td>Gr[K+Pn-3]
Low nibble</td><td>R[K+Pn4]
Low nibble</td></tr>
<tr><td colspan="2">Gr[K+Pn-3]
High byte</td><td colspan="2">R[K+Pn-2]
Low nibble</td><td>Gr[K+6]
Low nibble</td><td>R[K+Pn-2]
Low nibble</td><td colspan="2">Gr[K+Pn-1]
Low nibble</td></tr>
</table>

Odd line

<table>
<tr><td colspan="2">Gb[K+0]
High byte</td><td>B[K+1]
Low nibble</td><td>Gb[K+0]
Low nibble</td><td colspan="2">B[K+1]
High byte</td><td colspan="2">Gb[K+2]
High byte</td></tr>
<tr><td>B[K+3]
Low nibble</td><td>Gb[K+2]
Low nibble</td><td colspan="2">B[K+3]
High byte</td><td colspan="2">Gb[K+4]
High byte</td><td>B[K+5]
Low nibble</td><td>Gb[K+4]
Low nibble</td></tr>
<tr><td colspan="8"></td></tr>
<tr><td>Br[K+Pn-5]
Low nibble</td><td>Gb[K+Pn-6]
Low nibble</td><td colspan="2">B[K+Pn-5]
High byte</td><td colspan="2">Gb[K+Pn-4]
Low nibble</td><td>B[K+Pn-3]
Low nibble</td><td>Gb[K+Pn4]
Low nibble</td></tr>
<tr><td colspan="2">B[K+Pn-3]
High byte</td><td colspan="2">Gb[K+Pn-2]
Low nibble</td><td>B[K+6]
Low nibble</td><td>Gb[K+Pn-2]
Low nibble</td><td colspan="2">B[K+Pn-1]
Low nibble</td></tr>
</table>

Figure 41 –BayerRG12Packed Format

6.6.2.16 **BayerRG16**

Each component has 16-bit signed integer data.

High byte	Low byte

Even line

R[K+0]	Gr[K+1]
R[K+2]	Gr[K+3]
R[K+W-4]	Gr[K+W-3]
R[K+W-2]	Gr[K+W-1]

Odd line

Gb[K+0]	B[K+1]
Gb[K+2]	B[K+3]
Gb[K+W-4]	B[K+W-3]
Gb[K+W-2]	B[K+W-1]

Figure 42 –BayerRG16 Format

6.6.2.17 **BayerGB8**

Each component has 8-bit data.

Even line

Gb[K+0]	B[K+1]	Gb[K+2]	B[K+3]
Gb[K+4]	B[K+5]	Gb[K+6]	B[K+7]
Gb[K+W-8]	B[K+W-7]	Gb[K+W-6]	B[K+W-5]
Gb[K+W-4]	B[K+W-3]	Gb[K+W-2]	B[K+W-1]

Odd line

R[K+0]	Gr[K+1]	R[K+2]	Gr[K+3]
R[K+4]	Gr[K+5]	R[K+6]	Gr[K+7]
R[K+W-8]	Gr[K+W-7]	R[K+W-6]	Gr[K+W-5]
R[K+W-4]	Gr[K+W-3]	R[K+W-2]	Gr[K+W-1]

Figure 43 –BayerGB8 Format

6.6.2.18 **BayerGB12Packed**

Each component has 12-bit data.

High byte	Low nibble

Even line

<table>
<tr><td colspan="3">Gb[K+0]
High byte</td><td>B[K+1]
Low nibble</td><td>Gb[K+0]
Low nibble</td><td colspan="2">B[K+1]
High byte</td><td colspan="2">Gb[K+2]
High byte</td></tr>
<tr><td colspan="2">B[K+3]
Low nibble</td><td>Gb[K+2]
Low nibble</td><td colspan="2">B[K+3]
High byte</td><td colspan="2">Gb[K+4]
High byte</td><td>B[K+5]
Low nibble</td><td>Gb[K+4]
Low nibble</td></tr>
<tr><td colspan="9"></td></tr>
<tr><td>B[K+Pn-5]
Low nibble</td><td colspan="2">Gb[K+Pn-6]
Low nibble</td><td colspan="2">B[K+Pn-5]
High byte</td><td colspan="2">Gb[K+Pn-4]
Low nibble</td><td>B[K+Pn-3]
Low nibble</td><td>Gb[K+Pn4]
Low nibble</td></tr>
<tr><td colspan="3">B[K+Pn-3]
High byte</td><td colspan="2">Gb[K+Pn-2]
Low nibble</td><td>B[K+6]
Low nibble</td><td>Gb[K+Pn-2]
Low nibble</td><td colspan="2">B[K+Pn-1]
Low nibble</td></tr>
</table>

Odd line

<table>
<tr><td colspan="3">R[K+0]
High byte</td><td>Gr[K+1]
Low nibble</td><td>R[K+0]
Low nibble</td><td colspan="2">Gr[K+1]
High byte</td><td colspan="2">R[K+2]
High byte</td></tr>
<tr><td colspan="2">Gr[K+3]
Low nibble</td><td>R[K+2]
Low nibble</td><td colspan="2">Gr[K+3]
High byte</td><td colspan="2">R[K+4]
High byte</td><td>Gr[K+5]
Low nibble</td><td>R[K+4]
Low nibble</td></tr>
<tr><td colspan="9"></td></tr>
<tr><td>Gr[K+Pn-5]
Low nibble</td><td colspan="2">R[K+Pn-6]
Low nibble</td><td colspan="2">Gr[K+Pn-5]
High byte</td><td colspan="2">R[K+Pn-4]
Low nibble</td><td>Gr[K+Pn-3]
Low nibble</td><td>R[K+Pn4]
Low nibble</td></tr>
<tr><td colspan="3">Gr[K+Pn-3]
High byte</td><td colspan="2">R[K+Pn-2]
Low nibble</td><td>Gr[K+6]
Low nibble</td><td>R[K+Pn-2]
Low nibble</td><td colspan="2">Gr[K+Pn-1]
Low nibble</td></tr>
</table>

Figure 44 –BayerGB12Packed Format

6.6.2.19 **BayerGB16**

Each component has 16-bit signed integer data.

High byte	Low byte

Even line

G[K+0]	B[K+1]
G[K+2]	R[K+3]
G[K+W-4]	B[K+W-3]
G[K+W-2]	R[K+W-1]

Odd line

G[K+0]	R[K+1]
G[K+2]	B[K+3]
G[K+W-4]	R[K+W-3]
G[K+W-2]	B[K+W-1]

Figure 45 –BayerGB16 Format

6.6.2.20 **BayerBG8**

Each component has 8-bit data.

Even line

B[K+0]	Gb[K+1]	B[K+2]	Gb[K+3]
B[K+4]	Gb[K+5]	B[K+6]	Gb[K+7]
B[K+W-8]	Gb[K+W-7]	B[K+W-6]	Gb[K+W-5]
B[K+W-4]	Gb[K+W-3]	B[K+W-2]	Gb[K+W-1]

Odd line

Gr[K+0]	R[K+1]	Gr[K+2]	R[K+3]
Gr[K+4]	R[K+5]	Gr[K+6]	R[K+7]
Gr[K+W-8]	R[K+W-7]	Gr[K+W-6]	R[K+W-5]
Gr[K+W-4]	R[K+W-3]	Gr[K+W-2]	R[K+W-1]

Figure 46 –BayerBG8 Format

6.6.2.21 **BayerBG12Packed**

Each component has 12-bit data.

High byte	Low nibble

Even line

<table>
<tr><td colspan="3">B[K+0]
High byte</td><td>Gb[K+1]
Low nibble</td><td>B[K+0]
Low nibble</td><td colspan="2">Gb[K+1]
High byte</td><td colspan="2">B[K+2]
High byte</td></tr>
<tr><td colspan="2">Gb[K+3]
Low nibble</td><td>B[K+2]
Low nibble</td><td colspan="2">Gb[K+3]
High byte</td><td colspan="2">B[K+4]
High byte</td><td>Gb[K+5]
Low nibble</td><td>B[K+4]
Low nibble</td></tr>
<tr><td colspan="9"></td></tr>
<tr><td>Gb[K+Pn-5]
Low nibble</td><td colspan="2">B[K+Pn-6]
Low nibble</td><td colspan="2">Gb[K+Pn-5]
High byte</td><td colspan="2">B[K+Pn-4]
Low nibble</td><td>Gb[K+Pn-3]
Low nibble</td><td>B[K+Pn4]
Low nibble</td></tr>
<tr><td colspan="3">Gb[K+Pn-3]
High byte</td><td colspan="2">B[K+Pn-2]
Low nibble</td><td>Gb[K+6]
Low nibble</td><td>B[K+Pn-2]
Low nibble</td><td colspan="2">Gb[K+Pn-1]
Low nibble</td></tr>
</table>

Odd line

<table>
<tr><td colspan="3">Gr[K+0]
High byte</td><td>R[K+1]
Low nibble</td><td>Gr[K+0]
Low nibble</td><td colspan="2">R[K+1]
High byte</td><td colspan="2">Gr[K+2]
High byte</td></tr>
<tr><td colspan="2">R[K+3]
Low nibble</td><td>Gr[K+2]
Low nibble</td><td colspan="2">R[K+3]
High byte</td><td colspan="2">Gr[K+4]
High byte</td><td>R[K+5]
Low nibble</td><td>Gr[K+4]
Low nibble</td></tr>
<tr><td colspan="9"></td></tr>
<tr><td>R[K+Pn-5]
Low nibble</td><td colspan="2">Gr[K+Pn-6]
Low nibble</td><td colspan="2">R[K+Pn-5]
High byte</td><td colspan="2">Gr[K+Pn-4]
Low nibble</td><td>R[K+Pn-3]
Low nibble</td><td>Gr[K+Pn4]
Low nibble</td></tr>
<tr><td colspan="3">R[K+Pn-3]
High byte</td><td colspan="2">Gr[K+Pn-2]
Low nibble</td><td>R[K+6]
Low nibble</td><td>Gr[K+Pn-2]
Low nibble</td><td colspan="2">R[K+Pn-1]
Low nibble</td></tr>
</table>

Figure 47 –BayerBG12Packed Format

6.6.2.22 **BayerBG16**

Each component has 16-bit signed integer data.

High byte	Low byte

Even line

B[K+0]	Gb[K+1]
B[K+2]	Gb[K+3]
B[K+W-4]	Gb[K+W-3]
B[K+W-2]	Gb[K+W-1]

Odd line

Gr[K+0]	R[K+1]
Gr [K+2]	R[K+3]
Gr [K+W-4]	R[K+W-3]
Gr [K+W-2]	R[K+W-1]

Figure 48 –BayerBG16 Format

6.6.2.23 **Little Endian Mode**

If **PixelEndian** = **LittleEndian**, the order of the high and low bytes for <**Mono16, RGB16, Monosigned16, RGBsigned16,BayerXX16** formats SHALL be reversed as shown below.

Low byte	High byte

All other formats are not affected.

6.6.3 Data structure

6.6.3.1 Mono8, RGB8Packed, BayerXX8

Each component (Y, R, G, B) has 8-bit data. The data type is “Unsigned Char”.

Y,R,G,B	Signal level (Decimal)	Data (Hexadecimal)
Highest	255	0xFF
	254	0xFE
	:	:
	1	0x01
Lowest	0	0x00

6.6.3.2 YUV411Packed, YUV422Packed, YUV444Packed

Each component (Y, U, V) has 8-bit data. The Y component is the same as in the above table. The data type is "Straight Binary" for U and V data.

U, V	Signal level (Decimal)	Data (Hexadecimal)
Highest (+)	127	0xFF
	126	0xFE
	:	:
	1	0x81
Lowest	0	0x80
	-1	0x7F
	:	:
	-127	0x01
Highest (-)	-128	0x00

6.6.3.3 **Mono16, RGB16, BayerXX16**

Each component (Y,R,G,B) has 16-bit data. The data type is “Unsigned Short (big-endian)”.

Y,R,G,B	Signal level (Decimal)	Data (Hexadecimal)
Highest	65535	0xFFFF
	65534	0xFFFE
	:	:
	1	0x0001
Lowest	0	0x0000

6.6.3.4 **MonoSigned16, RGBSigned16**

Each component (Y,R,G,B) has signed 16-bit data. The data type is “Signed Short (big-endian)”.

Y,R,G,B	Signal level (Decimal)	Data (Hexadecimal)
Highest	32767	0x7FFF
	32766	0x7FFE
	:	:
	1	0x0001
	0	0x0000
	-1	0xFFFF
	:	:
	-32767	0x8001
Lowest	-32768	0x8000

6.6.3.5 **Mono12Packed, BayerXX12Packed**

Each component (Y) has 12-bit data. The data type is “Unsigned Short (big-endian)”.

.

Y,R,G,B	Signal level (Decimal)	Data (Hexadecimal)
Highest	4095	0xFFF
	4094	0xFFE
	:	:
	1	0x001
Lowest	0	0x000

6.7 StandardFormat and StandardFrameRate

StandardFormat and **StandardFrameRate** refer to **FwIsochronousPacketSize**. Relationship of there is as follows.

StandardFormat	StandardFrameRate							
	240 fps	120 fps	60 fps	30 fps	15 fps	7.5 fps	3.75 fps	1.875 fps
VGA_Mono8	🟪 10240	🟧 5120	2560	1280	640	320	160	80
VGA_RGB8Packed	🟥 30720	🟪 15360	🟧 7680	3840	1920	960	480	240
VGA_YUV422Packed	🟥 20480	🟪 10240	🟧 5120	2560	1280	640	320	160
XGA_Mono8	🟥 24576	🟪 12288	🟧 6144	3072	1536	768	384	192
XGA_RGB8Packed	-	-	🟥 18432	🟪 9216	🟧 4608	2304	1152	576
XGA_YUV422Packed	-	🟥 24576	🟪 12288	🟧 6144	3072	1536	768	384
HD720p_Mono8	🟥 30720	🟪 15360	🟧 7680	3840	1920	960	480	240
HD720p_RGB8Packed	-	-	🟥 23040	🟪 11520	🟧 5760	2880	1440	720
HD720p_YUV422Packed	-	🟥 30720	🟪 15360	🟧 7680	3840	1920	960	480
SXGA_Mono8	🟥 20480	🟪 10240	🟧 5120	2560	1280	640	320	160
SXGA_RGB8Packed	🟥 30720	🟪 15360	🟧 7680	3840	1920	960	480	240
SXGA_YUV422Packed	-	🟥 20480	🟪 10240	🟧 5120	2560	1280	640	320
UXGA_Mono8	🟥 32000	🟪 16000	🟧 8000	4000	2000	1000	500	250
UXGA_RGB8Packed	-	-	🟥 24000	🟪 12000	🟧 6000	3000	1500	750
UXGA_YUV422Packed	-	🟥 32000	🟪 16000	🟧 8000	4000	2000	1000	500
HD1080p_Mono8	-	-	🟥 19200	🟪 9600	🟧 4800	2400	1200	600
HD1080p_RGB8Packed	-	-	-	🟥 28800	🟪 14400	🟧 7200	3600	1800
HD1080p_YUV422Packed	-	-	-	🟥 19200	🟪 9600	🟧 4800	2400	1200

🟧 :required S800 data rate, 🟪 : required S1600 data rate, 🟥 : required S3200 data rate

Table 51 – Standard FwIsochronousPacketSize

6.8 Bibliography for IEEE1394

[B1] IEEE Std 1212-1991, Standard for a Control and Status Registers (CSR) Architecture for microcomputer buses

[B2] IEEE Std 1394-2008, Standard for a High Performance Serial Bus

[B3] IIDC 1394-based Digital Camera Specification Ver.1.32

[B4] ISO/IEC 9899:1990, Programming Languages—C

Annex A

Recommended FeatureCSRs of camera devices

The following table describes the recommended FeatureCSRs of minimum capability camera.

FeatureCSR	Description
CategoryBlock2 (ImageFormatControl)	
ImageFormatSelector	Camera SHOULD have at least one image format.
ApplyImageFormat	Camera SHOULD support Done status.
ImageSize	Camera SHOULD have at least one pattern of image size.
PixelCoding / PixelSize	Camera SHOULD support at least one pair of PixelCoding and PixelSize.
CategoryBlock3 (AcquisitionControl)	
Acquisition Command	Camera SHOULD support either Stop or Abort command, and support any one of Continuous, MultiFrame or ImageBufferRead at least.

Table 52 – Recommended FeatureCSRs

Annex B

Supporting multiple-ROI

IIDC2 supports multiple-ROI handling. This section describes recommended implementing method.

B.1 Overview

An example of multiple-ROI is as following figure.

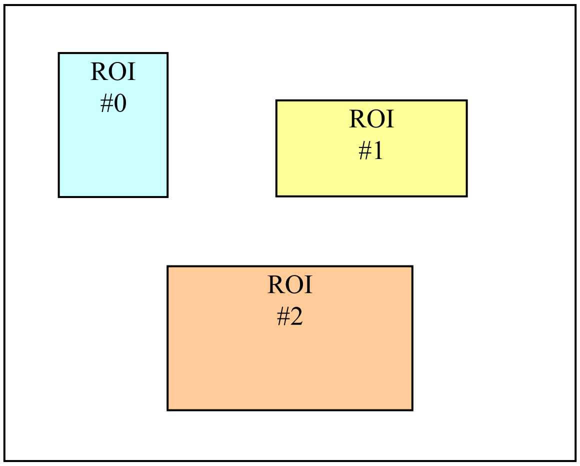

Figure 49 – Example of multiple-ROI

The condition of multiple-ROI is as follows.

- All regions are shaped rectangle
- There is flexibility of using number of ROIs.
- Streaming data format is defined on transport layer (there is no information in this section)

Multiple-ROI is implemented as **ExpandedCSR** of **ImageSize**. The block diagram is as following figure.

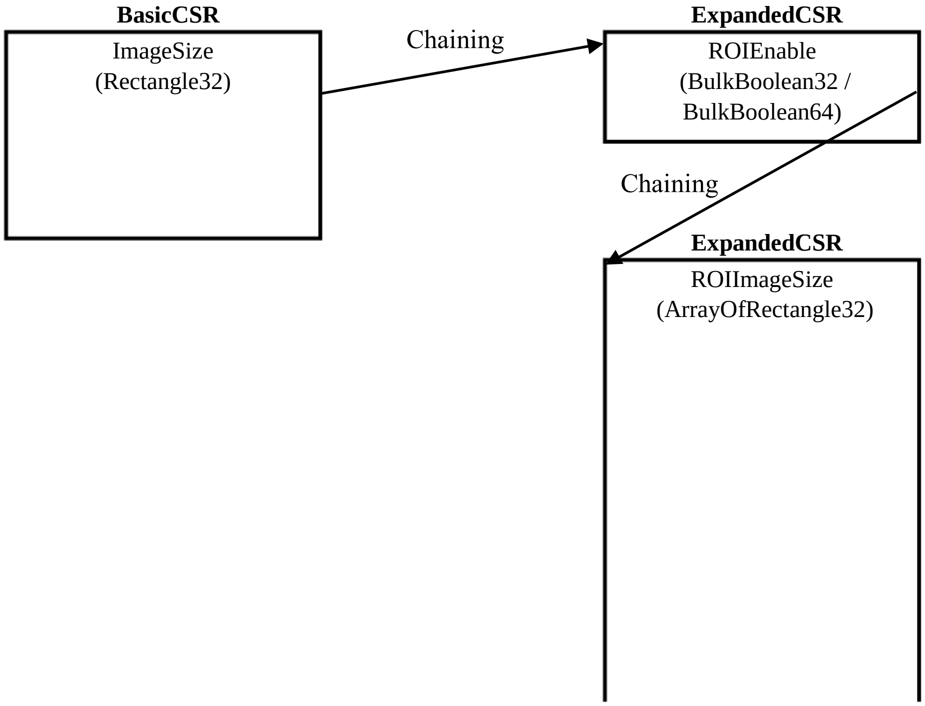

Figure 50 – Structure of multiple-ROI

B.2 BasicCSR

For backward compatibility, device SHOULD support **BasicCSR** handling (1 ROI mode).

B.3 ROIEnable

Handles which ROIs are used. **1** is enable, **0** is disable. **ValueType** is **BulkBoolean32** (up to 32 ROIs) or **BulkBoolean64** (up to 64 ROIs). A bit position indicates ROI number.

B.4 ROIImageSize

Sets image size and position of each ROIs. An index of array indicates ROI number.