Detection Technology

X-LIB SOFTWARE LIBRARY

Programmer's Manual

Document ID: DS0000087

Revision: 1.7

Detection Technology

Table of Contents

Warning

Detection Technology, Plc. Assumes no liability for damages consequent to the use of the product presented in this manual. The product presented is not designed with components of a level of reliability suitable for the use in life support, medical or other critical applications.

Disclaimer

All information presented in this manual is believed to be accurate and reliable. However, Detection Technology, Plc. Assumes no responsibility for the use of such information or for any infringements of patents or other rights of third parties that may result from its use. Due to continuous product development, the information in this document is subject to change without notice.

Copyright

Contact address

Detection Technology Oyj

Elektroniikkatie 10,

FI-90590 Oulu, Finland

Tel +358 20 766 9700

Fax +358 20 766 9709

http://www.deetee.com/

contact@deetee.com

Change record

Rev	Date issued	Description of change
1.0	2015-08-31	Proofreading and structural modifications. Add XpiecewiseCorrect, XreferenceCorrect and XOffHLCorrect classes.
1.1	2015-12-29	Add figure 2-1, 2-4, 2-7 and 2-8 and relative illustrations.
1.2	2016-04-18	Add function XAcquisition::GetGrabTimeDiff(); Add Async trigger stamp mode.
1.3	2016-08-31	Correct figure 2-11; Revise "[OA]" command; Add parameter "XPARA_TRIGGER_STAMP_PARITY"; Revise table 6-3.
1.4	2016-12-10	Add XTifFormat class; Add commands 0x8C/PT, 0x8E/WT, 0x8F/OD, 0x90/OE, 0x91/CF and 0x92/BR.
1.5	2017-05-18	Add chapter 8.

1.6	2017-08-21	Revise XDevice::GetDMPixelNumber(); Add description of method XHealthParaW XCommandW.GetHeath(); revise the description of non-continuous integration time; cancel command 0x7F/TH.
1.7	2018-08-09	Support pixel depth of 20 bits; Add “XPARA_POWR_UP_TIME” and “XPARA_GCU_WORK_TIME” to XCommand::GetPara(); Support the new generation of x-ray detector; support visual studio 2012/2013/2015; Add command 0x80/RL, 0x94/RA, 0x95/RC, 0x96/AV, 0x97/PC, 0x98/AS;

1. INTRODUCTION

1.1. Overview

The X-LIB software library is designed to work with X-GCU products. It is built on the application layer of Ethernet technology using a UDP IPv4 Ethernet protocol as the transport layer protocol.

The X-LIB software library achieves the following goals:

- Supports rapid, safe and robust data transmission between an X-GCU product and a computer.
- Has distinctive steps for IP configuration and for device enumeration.
- Keeps the command and image data in different channels logically
- Is based on existing IP protocols to minimize development efforts of an application.

1.2. Scope

The X-LIB programmer's manual covers the following topics:

- Protocol of the X-GCU
- Getting started
- Basic class reference
- Command list

1.3. Requirements

The X-LIB software supports Windows7/8.1/10 and Ubuntu 14.04/16.04 both 32-bits and 64-bits systems. The following compilers are supported:

- Microsoft Visual C++ 2008/2010/2012/2013/2015
- Microsoft Visual C# 2010/2012/2013/2015
- GCC 4.6.3

1.4. Terms, definitions and abbreviations

Abbreviation	Definition
ACK	The acknowledgement sent from an X-GCU after getting a command from the computer.
Aurora B	16-bit basic X-ray detector platform.
Aurora GT	20-bit X-ray detector platform.
CAT-5e/6	Category 5/6 cable.

Abbreviation	Definition
CMD	Command code between an X-GCU and a computer.
CRC	Cyclic Redundancy Check.
Digital X-Card	Dual-energy or Single-energy X-ray detector board.
DM	The front-end Detector Module which is composed of X-DFE and X-Ray cards.
DMID	The index of DMs; the first DM of the first X-GCU channel is marked as DM1, the following DMs are marked by the order.
EPCS	Electronic Prescriptions for Controlled Substances.
ERR_ID	Error ID sent from an X-GCU.
GCC	GNU Compiler Collection.
HEX	Hexadecimal.
Hi/Lo trigger	The high energy and low energy external line trigger generated by dual- energy LCS system.
IP	Internet Protocol address.
LED	Light-Emitting Diode.
LVDS	Low-Voltage Differential Signaling.
MAC	Media Access Control address.
MTU	Maximum Transmission Unit of TCP/IP protocol.
OPE	Operation mode of command between X-GCU and computer.
PDC	Pixel Discontinuity Correction.
UDP	User Datagram Protocol.
X-DEF	The front end control unit of the detector which reads the signal from analog front-end, digitizes the signals and sends back the image data to the X-GCU.
X-GCU	Gigabit control unit board of the detector which reads the image data from numbers of X-DFEs and sends the data to the system host computer.
X-GCU_CL	Gigabit Control Unit Board, Camera Link version.
X-GCU_EX	Gigabit Control Unit Board, Extra version.
X-GCU_STD	Gigabit Control Unit Board, Standard version.
X-LIB	Software library to support the X-GCU.

2. PROTOCOL OF X-GCU

2.1. Overview

The X-GCU device adopts the Gigabit Ethernet technology to transmit command and image data. It allows for easy interfacing between an X-GCU product and a computer using a standard CAT-5e/6 cable.

All the X-GCU protocols use the UDP IPv4 Ethernet protocol as the transport layer protocol. There are three main adopted protocol formats: broadcast, command and image packets.

The X-GCU device adopts 3 UDP ports to build broadcasting channel, command channel and image channel, which are used for configuring the Ethernet parameter, controlling the device and sending the image data.

The CRC algorithm adopted is CRC-32/MPEG-2. CRC polynomial is

$$x32+x26+x23+x22+x16+x12+x11+x10+x8+x7+x5+x4+x2+x+1$$

2.2. IP configuration

IP configuring negotiation covers the sequence of events for X-GCU device to get a valid IP address using standard IP protocols and for a control application to enumerate devices on the network. Once the negotiation is completed, an application is ready to send control requests to an X-GCU device.

Figure 2-1 shows the configuring process.

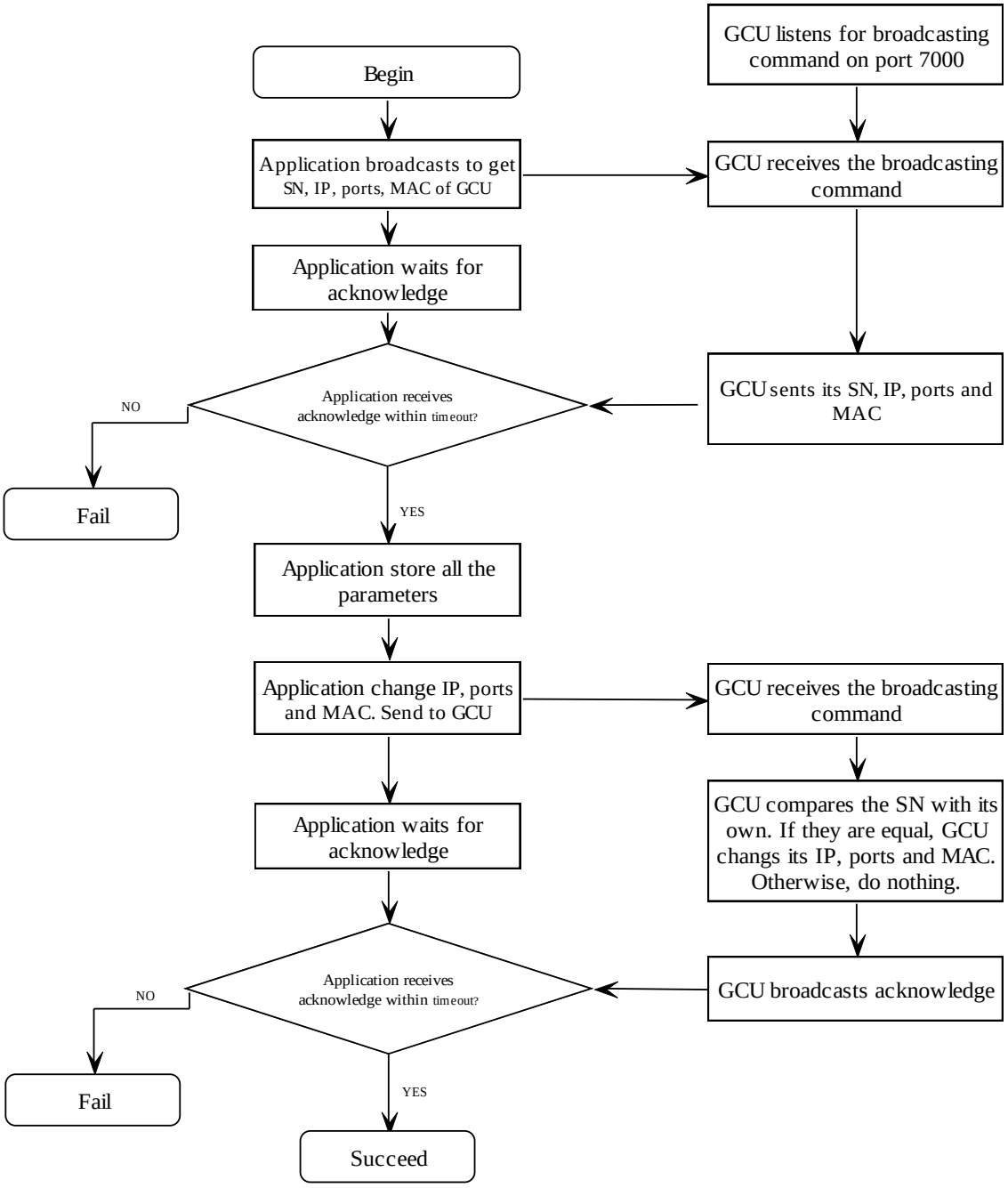


Figure 2-1. IP configuration flowchart

The broadcasting command of application is based on the UDP broadcasting protocol, as is the acknowledge (ACK) of the X-GCU. The default broadcasting port for the X-GCU is 7000. The default Ethernet parameters of X-GCU are IP 192.168.1.2, command port 3000 and image port 4001.

The byte order must be big-endian (network byte order).

Figure 2-2 shows the broadcasting command format:

Start code	0xBCBC

End code	0xFCFC
CMD	Specific command code
OPE	Operation mode: none/0x00, write/0x01, read/0x02, save/0x03 or load/0x04
DM ID	DM ID number or address: none/0x00
SIZE	Size of DATA segment
DATA	Data value: IP/4bytes, command port/2bytes, image port/2bytes and MAC/6bytes
CRC	CRC of CMD,OPE, DM ID, SIZE and DATA sections

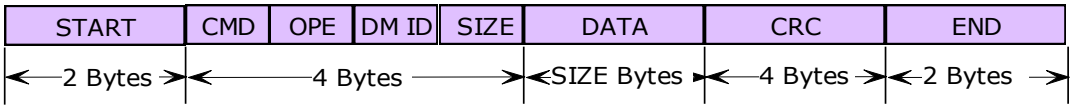


Figure 2-2. Broadcasting command format

Figure 2-3 shows the broadcasting ACK format of the X-GCU:

Start code	0xBCBC
End code	0xFCFC
CMD	The same with the command code of the application: 0x01
ERR ID	Error ID: success/0x00 or a specific meaningful error ID (defined by firmware)
DM ID	DM ID number or address: none/0x00
SIZE	Size of DATA segment
DATA	Data value: IP/4bytes, command port/2bytes, image port/2bytes and MAC/6bytes
CRC	CRC of CMD, OPE, DM ID, SIZE and DATA sections

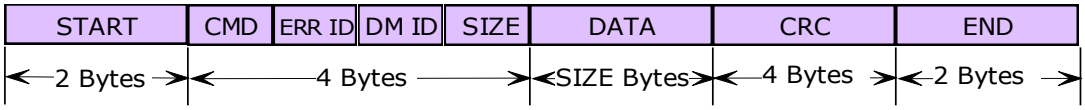


Figure 2-3. Broadcasting ACK format

Table 2-1. Broadcasting command list

Application broadcasting command					**Description**	**X-GCU ACK**				
CMD	**OPE**	**DM ID**	**SIZE**	**DATA**		**CMD**	**ERRID**	**DM ID**	**SIZE**	**DATA**

0x01	0x01	0x00	0x2E	SN: 32bytes; X-GCU's IP: 4bytes; X-GCU's MAC: 6bytes; Cmd port: 2bytes; Img port: 2bytes	Set X-GCU's IP, MAC and port. X-GCU needs to check the SN before setting	0x01	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Read X-GCU's SN, IP, MAC and port		0x00: Success; Other: Fail	0x00	0x2E	SN: 32bytes; X-GCU's IP: 4bytes; X-GCU's MAC: 6bytes; Cmd port: 2bytes; Img port: 2bytes
	0x03	0x00	0x00	None	Save the above settings into flash		0x00: Success; Other: Fail	0x00	0x00	None
	0x04	0x00	0x00	None	Load the above settings from flash		0x00: Success; Other: Fail	0x00	0x00	None
	0x05	0x00	0x00	None	Recover the default value		0x00: Success; Other: Fail	0x00	0x00	None

2.3. Command channel protocol

After the IP configuration negotiation succeeds, the connection between X-GCU and application is established. Command requests are always initiated by application, and X-GCU gives the corresponding ACK after executing the specific command.

Command and acknowledge messages must each be contained in a single packet. The Application must wait for the acknowledge message (when requested) before sending the next command message. If the application doesn't get ACK message within the timeout, it should report the timeout error to user.

Below is the flowchart of the interaction of command messages.

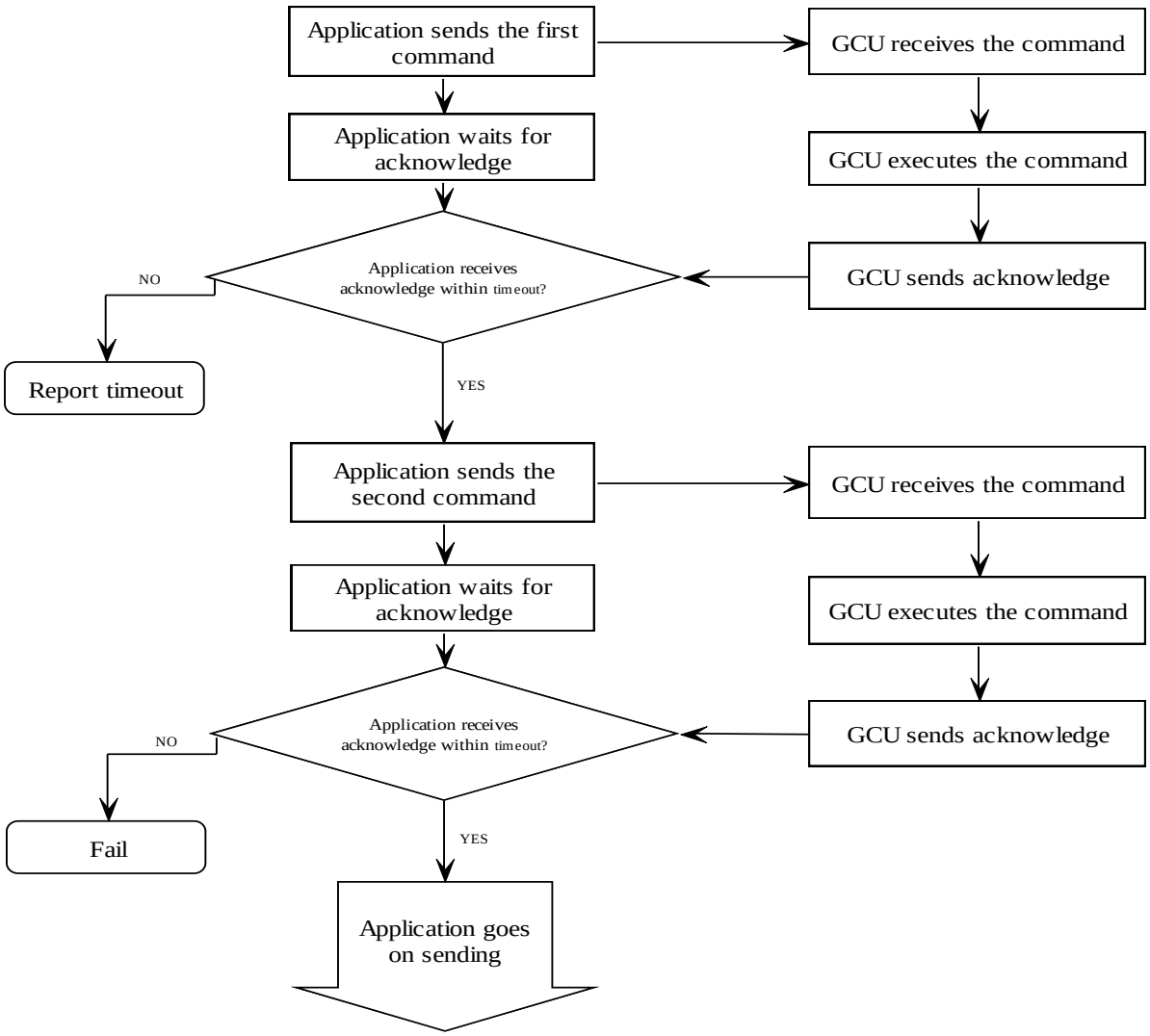


Figure 2-4. Command channel flowchart

The X-GCU command protocol uses the UDP with IPv4 as the transport protocol. The command channel port number of the application is dynamically allocated or designated and that of the X-GCU device is designated by the negotiation process. The default command port number of X-GCU is 3000.

The command and ACK messages must be contained in a single UDP packet. All the command and ACK messages are transported by the command port of the X-GCU and the application.

The byte order must be big-endian (network byte order).

Figure 2-5 shows the command packet format:

Start code	0xBCBC
End code	0xFCFC
CMD	Specific command code
OPE	Operation mode: none/0x00, write/0x01, read/0x02, save/0x03 or

	load/0x04
DM ID	DM ID number or address: none/0x00, all/0xFF or a specific ID number
SIZE	Size of DATA segment
DATA	Data value: none or specific data
CRC	CRC of CMD, OPE, DM ID, SIZE and DATA sections

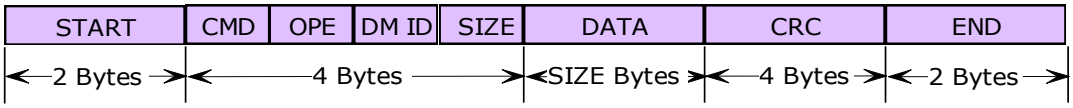


Figure 2-5. Command packet format

Figure 2-6 shows the ACK packet format of the GCU:

Start code	0xBCBC
End code	0xFCFC
CMD	The same with the command code of the application
ERR ID	Error ID: success/0x00 or a specific meaningful error ID (defined by firmware)
DM ID	DM ID number or address: none/0x00, all/0xFF or specific ID number
SIZE	Size of DATA segment
DATA	Data value: none or specific data
CRC	CRC of CMD, OPE, DM ID, SIZE and DATA sections

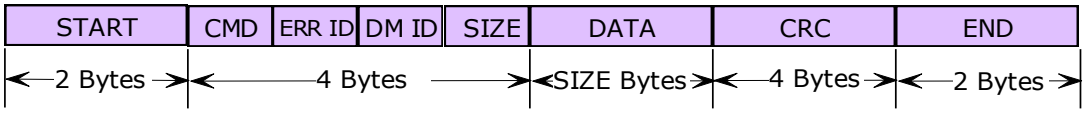


Figure 2-6. ACK packet format

Please refer to section 6.1 for the hex command list.

UDP is a connectionless protocol. As such, it does not inherently support a scheme to guarantee the remote device is still connected. X-GCU device supports a heartbeat counter and keeps sending heartbeat packets periodically.

If no heartbeat packet comes within the timeout, the application should report the error to user. Following figure is the heartbeat flowchart.

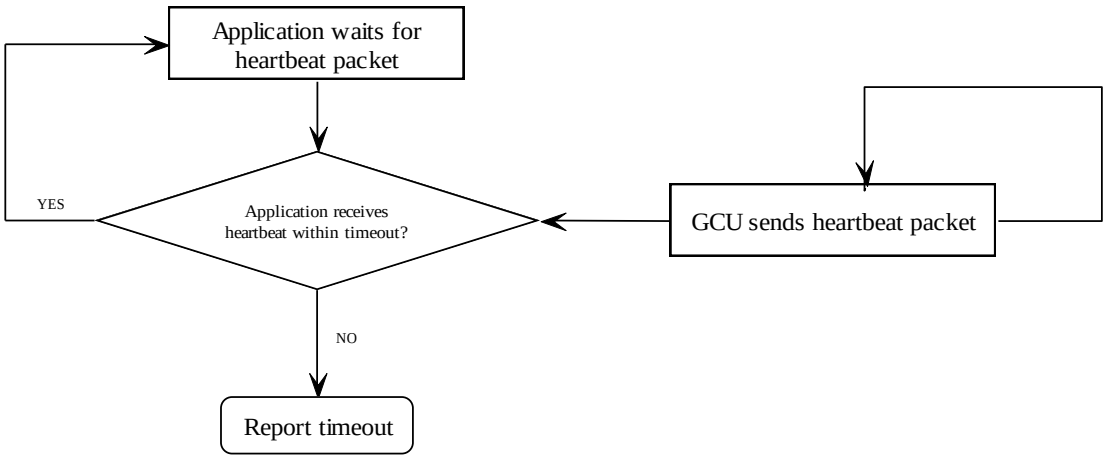


Figure 2-7. Heartbeat Flowchart

The byte order must be big-endian (network byte order).

In table 2-2, the calculations of X-GCU voltage, temperature and humidity are as follows:

Voltage value = (voltage high byte*256 + voltage low byte)*2.048/2047 (V)

v1 = voltage value*24/1.5(V)

v2 = voltage value*2(V)

v3 = voltage value*2(V)

v4 = voltage value(V)

Temperature value = (temperature high byte*256 + temperature low byte)*0.125(°C)

Humidity value = (humidity high byte*256 + humidity low byte)*125/65536 – 6(%)

The correct value range should be as follows:

v1: 24V±10% (power 24V) , 12V±10% (power 12V)

v2: 3.3V±5%

v3: 2.5V±5%

v4: 1.1V±5%

The relative humidity is not available with the X-GCU_STD.

Table 2-2. Heartbeat packet

Description	**X-GCU ACK**				
	CMD	**ERRID**	**DM ID**	**SIZE**	**DATA**
Heartbeat	0xFF	0x00	0x00	0x0C	X-GCU's voltage: 2bytes * 4 Temperature: 2bytes

					Humidity: 2 bytes

2.4. Image channel protocol

During the IP negotiation process, the application gets the image channel port of X-GCU, it should bind the same port to receive image data. The default image port number of X-GCU is 4001. After receiving the starting scan command, X-GCU device keeps sending image data packet to the image port of application until receiving the stopping scan command.

The image data transmission must be initiated by application. Application doesn't give any ACK to X-GCU, so the image channel works in single direction mode. Application should start a timeout counter after starting scan. If no image data packet is received within the timeout period, application should report the timeout error to user.

Figure 2-8 shows the flowchart of image channel.

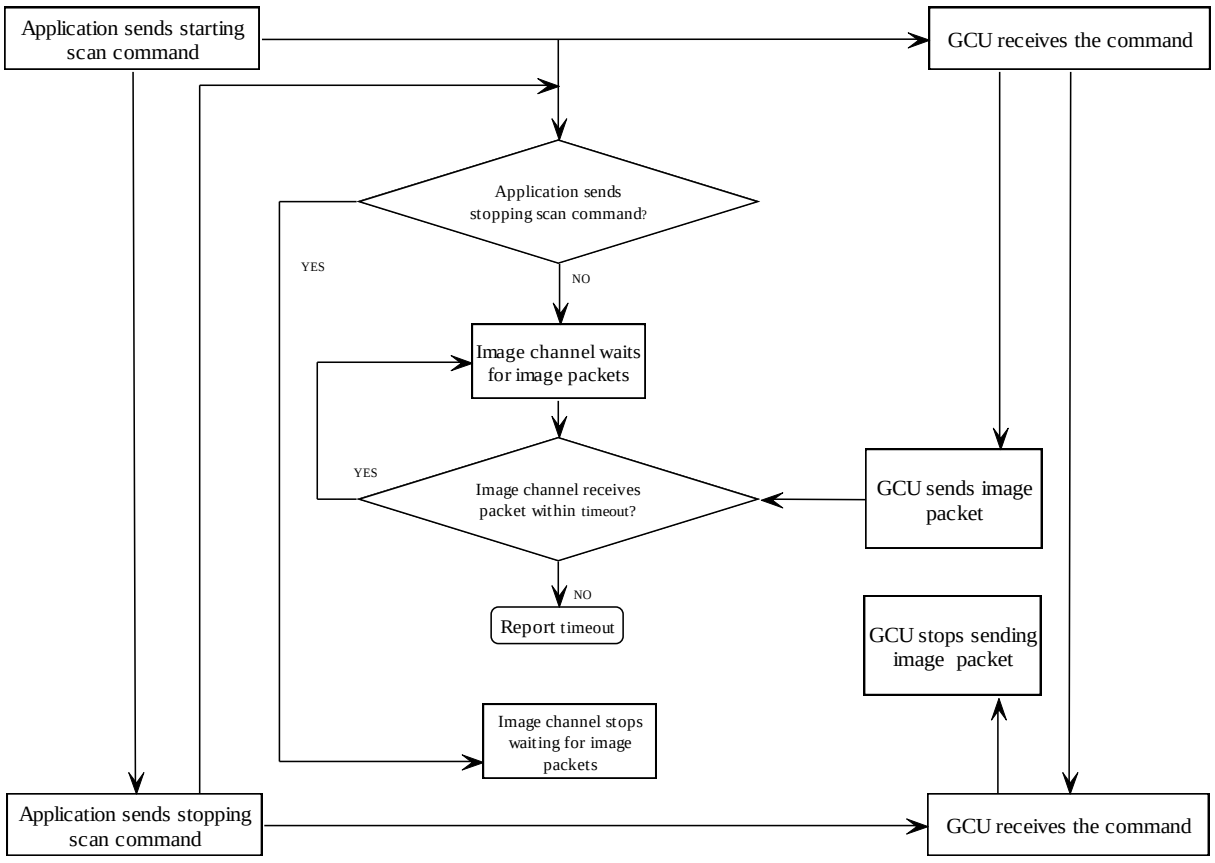


Figure 2-8. Image Channel Flowchart

The X-GCU image protocol uses the UDP with IPv4 as the transport protocol. The basic unit of an image is a pixel line composed by pixels, which is also the unit for transmission. An additional leader packet should be sent before the line payload packets. The leader packet provides specific information about the pixel line.

There should be two different packet types in the image channel:

- Line leader packet.

- Line payload packet.

Figure 2-9 illustrates the sequence of packets in the image channel.

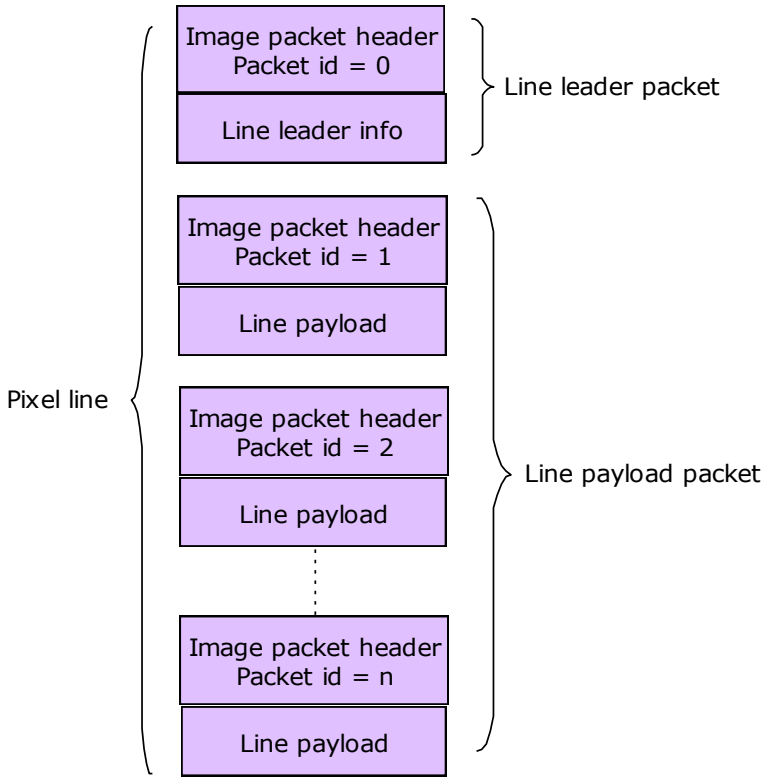


Figure 2-9. Pixel line packets

The byte order must be big-endian (network byte order).

Figure 2-10 shows the line leader packet format:

Start code	0xBCBC
CMD	Normal data/0xE0; X-DEF test mode/0xE1; X-GCU test mode/0xE3
LINE ID	Line index number
PACKET ID	Packet index number within one line: leader/0x00
PAYLOAD SIZE	Payload size of the packet
LINE STAMP	Time stamp or encoder count
LINE SIZE	Size of the whole pixel line by bytes
PIXEL SIZE	Size of the pixel multiply 10
ENERGY FLAG	High or low energy flag: low/0x00, high/0x01
COMPRESSION FLAG	Compression flag: none/0x00, others/specific compression
DM PACKET NUM	Total DM packet number

DM INFO	Each DM's info covers 8 bytes: CRC err flag: no err/0x00; Temperature; Voltage err flag: no err/0x00; Humidity; HE gain and LE gain
CRC	CRC of CMD, LINE ID, PACKET ID, PAYLOADSIZE and payload data sections

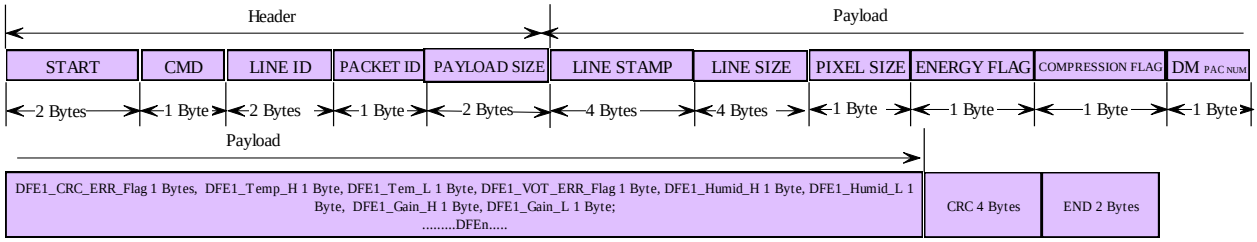


Figure 2-10. Line leader packet format

The calculations of X-DFE temperature and humidity are as follows:

Temperature value = (temperature high byte*256 + temperature low byte)*0.125(°C)

Humidity value = (humidity high byte*256 + humidity low byte)*125/65536 – 6(%)

Figure 2-11 shows the line payload packet format:

Start code	0xBCBC
CMD	Normal data/0xE0; X-DEF test mode/0xE1; X-GCU test mode/0xE3
LINE ID	Line index number
PACKET ID	Packet index number within on line: above 0x00
PAYLOAD SIZE	Payload size of the packet
PIXEL PAYLOAD	Pixel data in big-endian byte order
CRC	CRC of the CMD, LINE ID, PACKET ID, PAYLOADSIZE and payload data sections

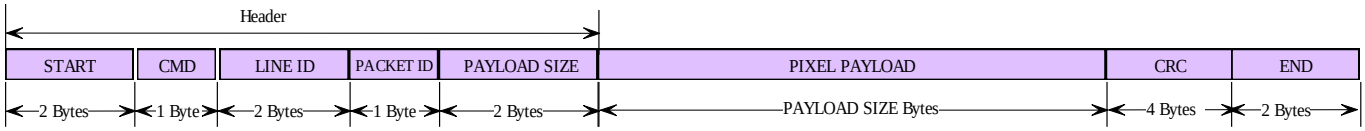


Figure 2-11. Line payload packet format

3. CONFIGURATION

3.1. Library files

File name	Description	Location
*.h	Header files	x-lib sdk\include
XLibDll.lib	32-bits and 64-bits .lib files (Win C++)	x-lib sdk\lib\x86, \lib\x64
XLibDll.dll	32-bits and 64-bits .dll files (Win C++)	x-lib sdk\lib\x86, \lib\x64
XLibWrapper.dll	32-bits and 64-bits .dll files (.NET)	x-lib sdk\NET\x86, \NET\x64
XLib.so	32-bits and 64-bits so files (Ubuntu C++)	x-lib sdk\lib
Demo	Sample files	x-lib sdk\demo

3.2. Compiler configuration

Visual Studio 2008\2010\2012\2013\2015 C++

- Copy \include and \lib to project folder
- Add .\include in "Project | Properties | C/C++ | General | Additional Include Directories"
- Add .\lib\x86 or .\lib\x64 in "Project | Properties | Linker | General | Additional Library Dependencies"
- In "Project | Properties |C/C++ | Code Generation | Runtime Library", choose "Multi-threaded DLL"

Visual Studio 2010\2012\2013\2015 C#

- Copy \NET to the project folder
- Add XLibWrapper.dll to "References"
- Add "using XLibWrapper" to the source code

GCC 4.6.3

- Copy \include and \lib to the project folder
- Compile the source code as follows:
    - g++ -rdynamic –I..\Include .\lib\xlib.so main.cpp –o main

3.3. Demos

Please refer to the \demo folder for the specific examples. For Visual Studio 2008 demo, you should copy the "stdint.h" file from demo project to "C:\Program Files \Microsoft Visual Studio 9.0\VC\include".

4. C++ CLASS REFERENCES

4.1. Class table

Class	Description
IXCmdSink	The user defines derived class from this abstract class to handle events of the XCommand object.
IXImgSink	The user defines derived class from this abstract class to handle events of the XAcquisition object.
XAcquisition	This class is in charge of getting image data from the X-GCU.
XAnalyze	Provides the basic analysis functions.
XCommand	This class is in charge of command controlling of the X-GCU.
XDevice	Wraps the basic parameters of the X-GCU which is used as argument by other classes.
XDisplay	Is in charge of displaying frames. Only available for the Windows system.
XDualEngCorrect	Reconstructs the dual energy image of the interlaced dual-energy raw data.
XDualLineCorrect	Reconstructs the line sequence of the dual-line detector sub-system.
XFrameTransfer	Gets line data from the XAcquisition object and combines line data into frame.
XGigFactory	Is in charge of generating objects of network.
XImage	Wraps frame data.
XMultiTransfer	When connecting with the multi-detector synchronizes each XFrameTranser object and generates the whole frame.
XOffCorrect	Provides off-board gain and offset correcting functions for all operational modes except non-continuous hi/lo trigger mode.
XOffHLCorrect	Provides off-board gain and offset correcting functions only for non-continuous hi/lo trigger mode.
XPiecewiseCorrect	Provides off-board three points piecewise correction functions for all operation modes except non-continuous hi/lo trigger mode.
XPixelCorrect	Provides functions to correct bad pixels.
XReferenceCorrect	Provides functions to correct signal variations caused by LINAC X-ray source.

XSystem	This class communicates with the X-GCU by broadcast, which reads and sets the Ethernet configurations of the X-GCU.
XTifFormat	Provides manipulating ways of TIFF format image.

4.2. Class references

IXCmdSink class

Description:

Abstract class. The user defines derived class from it to handle the events of the XCommand and XSystem objects.

#include "ixcmd_sink.h"

Functions:

virtual void OnXError(uint32_t err_id, const char* err_msg_) = 0

Parameters:

err_id Error ID.

err_msg_ Error description.

Remark:

The error callback of the XCommand and XSystem objects. Please refer to the section 4.3 for the specific error ID.

virtual void OnXEvent(uint32_t event_id, float data) = 0

Parameters:

event_id Event ID.

data Event data.

Remark:

The event callback of the XCommand and XSystem objects. Please refer to the section 4.4 for the specific event ID. This callback responds to the event IDs from 56 to 57.

IXImgSink class

Description:

Abstract class. The user defines derived class from this abstract class to handle events of the XAcquisition and XFrameTransfer objects.

#include "iximg_sink.h"

Functions:

virtual void OnXError(uint32_t err_id, const char* err_msg_) = 0

Parameters:

err_id Error ID.

err_msg_ Error description.

Remark:

The error callback of the XAcquisition and XFrameTransfer objects. Please refer to the section 4.3 for the specific error ID.

virtual void OnXEvent(uint32_t event_id, uint32_t data) = 0

Parameters:

event_id Event ID.

data Event data.

Remark:

The event callback of the XAcquisition and XFrameTransfer objects. Please refer to the section 4.4 for the specific event ID. This callback responds to the event IDs from 50 to 55.

virtual void OnFrameReady(XImage* image_) = 0

Parameters:

image_ XImage object pointer.

Remark:

When the frame is completed, the user can get the frame data in this callback function. Please note, if this function takes too much time to return, the frame buffer could overflow.

XAcquisition class

Description:

The XAcquisition class provides functions to manipulate image data acquisition. It gets data packets from the X-GCU and parses packets into line data and passes line data to the XFrameTransfer object.

#include "xacquisition.h"

Constructors:

XAcquisition::XAcquisition(IXFactory* factory_, uint32_t timeout = XIMG_TIMEOUT);

XAcquisition::XAcquisition()

Parameters:

factory_ IXFactory object pointer. See XGigFactory class.

timeout Waiting time while grabbing image. It will show an error if no image data arrives within the period. The timeout unit is ms with the default value 20000. The range is from 0 to 2000000. If it is 0, the grabbing thread will block waiting for data without reporting any error.

Remark:

For the default constructor with on parameters, the user must set the factory pointer by the function SetFactory(IXFactory*) before opening.

Functions:

void XAcquisition::Close()

Remark:

Releases all the resources.

void XAcquisition::EnableLineInfo(bool enable)

Parameters:

enable Line info mode, 1 enable, 0 disable.

Remark:

When enabled, there will be an info header added on the front of each line data. The content of the header is line id (2 bytes), line stamp(4 bytes), energy flag(1 byte) and DFE number flag(1 byte). It is disabled by default.

If the user is utilizing the class XMuliTransfer to synchronize the multi-X-GCU, the line info mode must be enabled.

If it is enabled, the positions of lost lines will be filled with zero values, or else new line data will substitute the lost lines.

uint32_t XAcquisition::GetGrabTimeDiff()

Return:

When start grabbing image data, it will do some extra initializing works first such as initializing thread and cleaning buffers. So it will take longer time for getting the first frame while calling grab() function. For snap() function, it will do the extra work for each frame. This function returns the time difference between the point of calling grab()/snap() and the point that actual data scanning happens. It could help to correct the object position in the first frame. The unit of returned time is ms for Windows system and us for Linux system.

bool XAcquisition::GetIsGrabbing()

Return:

Returns the working state, 1 grabbing, 0 stopping.

bool XAcquisition::GetIsOpen()

Return:

Returns the state, 1 opening, 0 closing.

uint32_t XAcquisition::GetLastError()

Return:

Returns the error ID if any exception occurs. Please refer to the section 4.3 for the explanation of the specific error.

bool XAcquisition::Grab(uint32_t frame_num)

Parameters:

frame_num	Number of frame to grab. If it is 0, it will keep grabbing until the Stop() is called. Otherwise, it will stop automatically after getting the number of frames.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Starts grabbing image data from the X-GCU. If it returns 0, the user can utilize the GetLastError() to trace the error.

bool XAcquisition::Open(XDevice* dev_, XCommand* cmd_handle_)

Parameters:

dev_	XDevice object pointer which provides the detector's info. See XDevice class.
cmd_handle	XCommand object pointer. See XCommand class.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Allocates the resources needed. If it returns 0, the user can utilize the GetLastError() to trace the error.

void XAcquisition::RegisterEventSink(IXImgSink* img_sink)

Parameters:

img_sink_	IXImgSink object pointer which is defined by the user.

Remark:

The IXImgSink object handles the error and event callbacks. This function should be called before opening.

void XAcquisition::RegisterFrameTransfer(IXTransfer* transfer_)

Parameters:

transfer_ IXTransfer object pointer. See XFrameTransfer class.

Remark:

Must be called before opening.

void XAcquisition::SetFactory(IXFactory* factory_)

Parameters:

factory_ IXFactory object pointer. See XGigFactory class.

Remark:

If the user is utilizing a default constructor, it should be called before opening.

void XAcquisition::SetTimeout(uint32_t time)

Parameters:

time Timeout value.

Remark:

If no image data comes during the timeout period while grabbing, it reports an error and stop scanning. The timeout unit is ms with the default value 20000. The range is from 0 to 2000000. If it is 0, the grabbing thread will block waiting for data without reporting any error. It should be set before opening.

bool XAcquisition::Snap()

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Gets one frame from the X-GCU and is blocked until one frame is received. If it returns 0, the user can utilize the GetLastError() to trace the error.

bool XAcquisition::Stop()

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Stops grabbing frame which should be used with the function Grab(0). If it returns 0, the user can utilize the GetLastError() to trace the error.

XAnalyze class

Description:

Provides functions to calculate average and noise.

#include “xanalyze.h”

Constructors:

XAnalyze::XAnalyze()

Remark:

Default constructor.

Functions:

bool XAnalyze::DoAnalyze(XImage* image_, uint32_t type)

Parameters:

image_	XImage object pointer. See XImage class.
type	Defines what analysis to apply.
	0: Calculates row average, min and max values;
	1: Calculates row noise;
	2: Calculates column average, min and max values;
	3: Calculates column noise.

Return:

Returns 0 if the parameter arrays fail to allocate.

Remark:

If you want to get noise, the averaging analysis must be done first. The user can get the public parameter arrays directly after doing the analysis.

Public attributes:

uint32_t* _col_avg_

Remark:

The column average array which can be received after the averaging analysis. The array size is the image width. The default is NULL.

uint32_t* _col_max_

Remark:

The column max array which can be received after the max analysis. The array size is the image width. The default is NULL.

uint32_t* _col_min_

Remark:

The column min array which can be received after the min analysis. The array size is the image width. The default is NULL.

float* _col_noise_

Remark:

The column noise array which can be received after noise analysis. The array size is the image width. The default is NULL.

uint32_t* _row_avg_

Remark:

The row average array which can be received after the averaging analysis. The array size is the image height. The default is NULL.

uint32_t* _row_max_

Remark:

The row max array which can be received after the max analysis. The array size is the image height. The default is NULL.

uint32_t* _row_min_

Remark:

The row min array which can be received after the min analysis. The array size is the image height. The default is NULL.

float* _row_noise_

Remark:

The row noise array which can be received after the noise analysis. The array size is the image height. The default is NULL.

XCommand class

Description:

The XCommand class manages the basic communication with the X-GCU. It adopts question and answer mode the X-GCU gives the ACK for each command. The user can send an ASCII or a HEX command directly to the X-GCU or use the wrapping functions instead.

#include”xcommand.h”

Constructors:

XCommand::XCommand(IXFactory* factory_, uint32_t timeout = XCMD_TIMEOUT)

XCommand::XCommand()

Parameters:

factory_ Pointer to factory object.

timeout Waiting time after sending a command. It will show an error if the ACK does not arrive within the period. The unit is ms. The range is 12000 to 2000000. The default value is 20000.

Remark:

For the default constructor with no parameters, the user must set the factory pointer by the function SetFactory(IXFactory*) before opening.

Functions:

void XCommand::Close()

Remark:

Releases all the resources.

int32_t XCommand::ExecutePara(uint32_t para, uint64_t data=0)

Parameters:

para Execution type as defined in the following table.

Data Some of the executions need this argument. See the following table.

Para	Data	Description
XPARA_INIT_PARA	None	Initializes the X-GCU with settings saved in the user's flash.
XPARA_INIT1_PARA	None	Initializes the X-GCU with default settings.
XPARA_SAVE_PARA	None	Saves all current settings into the user's flash.
XPARA_LOAD_GAIN	Range 0 – 3	Loads channel gain from the specific section of flash to the FPGA. There are 4 groups of parameters.
XPARA_LOAD_OFFSET	None	Loads channel offset from the flash to the FPGA.
XPARA_RESET_GAIN	None	Rests channel gain to 1.0.
XPARA_RESET_OFFSET	None	Rests channel offset to 0.
XPARA_SIMULATE_FRAME_TRIGGER	None	Triggers one frame in the external frame trigger mode. Generates a soft frame trigger.

Return:

Returns 1 if successful, 0 no such execution, -1 error.

Remark:

The user calls this function to force the X-GCU carry out the specific execution. It sends the specific command to the X-GCU. The X-GCU does the corresponding execution then returns the ACK. If an error occurs, the user can call the GetLastError() to trace the error.

bool XCommand::GetIsOpen()

Return:

Returns the state, 1 opening, 0 closing.

uint32_t XCommand::GetLastError()

Return:

Returns the error ID if any exception occurs. Please refer to the error list for the explanation of the specific error.

int32_t XCommand::GetPara(uint32_t para, uint64_t& data, uint8_t dm_id=0)

Parameters:

para Parameter type as defined in the following table.

Data Returned value of the specific parameter.

Dm_id DM module index which starts from 1. Some of the commands need this argument. If it is 0xFF, the X-GCU will apply the command to all DMs. Otherwise, the X-GCU only operates the specified DM module. The DM ID 1 represents the first DM module in the first X-GCU channel.

Return:

Returns 1 if successful, 0 no such parameter, -1 error.

Remark:

The user calls this function to get the specific parameter from the X-GCU. If any error occurs, the user can call the GetLastError() to trace the error.

Para	Data	DM_ID	Description
XPARA_INT_TIME	Returned value	None	Integration time, 4-byte unsigned value. The unit is us. The range is defined by max and min integration time. Default is 3000.
XPARA_NON_INTTIM	Returned value	None	Non-continuous integration time, 2-byte unsigned value.

Para	Data	DM_ID	Description
E			The unit is us. Default is 580.
XPARA_OPE_MODE	Returned value	None	Operation mode, 1-byte unsigned value. 0: Continuous mode; 1: Non-continuous mode 2: Constant integration time mode; 3: Non-continuous with Hi/Lo trigger. Default is 0.
XPARA_DM_GAIN	Returned value	DM index	DM module gain value, 2-byte unsigned value. The high 8 bits are high energy gain, the low 8 bits are low energy gain. Default is 0x0606. Valid range: LCS2: 1~63; X-Card 0.2/0.4/0.8: 1~2; X-Card 1.5/1.6/2.5: 1~12; X-Card2 0.2/0.4: 1~2; X-Card2 0.8: 1~8; Aurora GT/B: 1~127. The DM index cannot be 0xFF. It is not available while scanning.
XPARA_HILO_MODE	Returned value	None	Defines which DM module gain to launch in single energy mode. 1-byte unsigned value. 0: Low-energy gain; 1: High-energy gain; Default is 0. It is only available while XPARA_OPE_MODE is 3.
XPARA_CH_NUM	Returned value	None	DM module number in each X-GCU's channel. It is a five-byte value. The first channel number is the highest byte and the fifth channel number is the

Para	Data	DM_ID	Description
			lowest byte.
XPARA_EN_SCAN	Returned value	None	Scanning state, 1-byte unsigned value. 0: Stopping; 1: Scanning.
XPARA_EN_GAIN_CORRECT	Returned value	None	Gain correcting state, 1-byte unsigned value. Enables or disables channel gain correction. 0: Disable; 1: Enable. Default is 0.
XPARA_EN_OFFSET_CORRECT	Returned value	None	Offset correcting state, 1-byte unsigned value. Enables or disables channel offset correction. 0: Disable; 1: Enable. Default is 0.
XPARA_EN_BASELINE_CORRECT	Returned value	None	Baseline correcting state, 1-byte unsigned value. Enables or disables baseline correction. 0: Disable; 1: Enable. Default is 0.
XPARA_BASE_LINE	Returned value	None	Base line value, 2-byte unsigned value. Range 0 – 1000. Default is 0.
XPARA_BINNING_MODE	Returned value	None	Pixel binning mode, 1-byte unsigned value. 0: Original pixel; 1: Averaging every 2 pixels; 2: Summing every 2 pixels;

Para	Data	DM_ID	Description
			3: Averaging every 4 pixels; 4: Summing every 4 pixels
XPARA_AVERAGE_MODE	Returned value	None	Line averaging filter mode, 1-byte unsigned value. 0: None; 1: 2 lines; 2: 4 lines; 3: 8 lines; 4: 16 lines; 5: 32 lines; 6: 64 lines; 7: 128 lines; 8: 256 lines. Default is 0.
XPARA_SUMMING_MODE	Returned value	None	Line summing filter mode, 1-byte unsigned value. 0: None; 1: 2 lines; 2: 4 lines; 3: 8 lines; 4: 16 lines; 5: 32 lines; 6: 64 lines; 7: 128 lines; 8: 256 lines. Default is 0.
XPARA_OUTPUT_SCALE	Returned value	None	Output scale, 1-byte unsigned value. The output pixel depth will be the original bit subtracting the scale value. 0: Original;

Para	Data	DM_ID	Description
			n: Original – n. Default is 0. E.g., the original bit is 16, the scale value is 8, the output pixel bits will be 8.
XPARA_OFFSET_AVERAGE	Returned value	None	Offset averaging filter, 1-byte unsigned value. 0: None; 1: 2 lines; 2: 4 lines; 3: 8 lines. Default is 0. It is only available while XPARA_OPE_MODE is 3.
XPARA_LINE_TRIGGER_MODE	Returned value	None	External line trigger mode, 1-byte unsigned value. 0: Rising edge; 1: Falling edge; 2: Sync trigger stamp mode; 3: Async trigger stamp mode. Default is 0.
XPARA_EN_LINE_TRIGGER	Returned value	None	Enables external line trigger function, 1-byte unsigned value. 0: Disable; 1: Enable. Default is 0.
XPARA_LINETRG_FINE_DELAY	Returned value	None	External line trigger fine delay, 1-byte unsigned value. The unit is 0.125us. Default is 0.
XPARA_LINETRG_RAW_DELAY	Returned value	None	External line trigger raw delay, 1-byte unsigned value. The unit is 64us. Default is 0.

Para	Data	DM_ID	Description
XPARA_FRAME_TRIGGER_MODE	Returned value	None	External frame trigger mode, 1-byte unsigned value. 0: Rising edge; 1: Falling edge. Default is 0.
XPARA_EN_FRAME_TRIGGER	Returned value	None	Enables external frame trigger function, 2-byte unsigned value. 0: Disable; n: Enable, and output 32*n lines by each trigger. Default is 0.
XPARA_FRAME_TRIGGER_DELAY	Returned value	None	External frame trigger raw delay, 1-byte unsigned value. The unit is 32 lines. Default is 0.
XPARA_HILO_EDGE	Returned value	None	Hi/Lo line trigger with hi/lo edge level detection, 1-byte unsigned value. 0: Low level; 1: High level. Default is 0.
XPARA_HILO_SAMPLE_POS	Returned value	None	Hi/Lo line trigger sampling position, 1-byte unsigned value. The unit is 1us. Range is 0-255, default is 0.
XPARA_PIXEL_NUMBER	Returned value	None	Total pixel number of all DMs, 2-byte unsigned value.
XPARA_PIXEL_SIZE	Returned value	None	Pixel size, 1-byte unsigned value. The returned value is the real pixel size * 10. The unit of the real pixel size is mm.
XPARA_PIXEL_DEPTH	Returned value	None	Pixel depth, 1-byte unsigned value.

Para	Data	DM_ID	Description
			16: 16 bits; 18: 18 bits; 20: 20 bits. Default is 16.
XPARA_MAXMIN_INT TIME	Returned value	None	Max and min integration time, 8-byte unsigned value. The high four bytes are the max value, the low four bytes are the min value.
XPARA_GCU_FIRM_V ER	Returned value	None	X-GCU's firmware version, 2-byte unsigned value.
XPARA_DM_FIRM_VE R	Returned value	DM index	DM's firmware version, 2-byte unsigned value. The DM index cannot be 0xFF. It is not available while scanning.
XPARA_GCU_TEST_M ODE	Returned value	None	X-GCU's test pattern mode, 1-byte unsigned value. 0: Disable; 1: Normal test pattern; 2: Slope test pattern; Default is 0.
XPARA_DM_TEST_M ODE	Returned value	DM index	DM's test pattern mode, 1-byte unsigned value. 0: Normal image data mode; 1: Enable DM board fixed value (AAAAA[Hex]) test mode.; 2: Enable I'm here function, Light the LED of the specific DM. Make sure the LED must enabled. Default is 0. The DM index cannot be 0xFF. It is not available while scanning.
XPARA_DM_PIXEL_N UM	Returned value	None	DM's pixel number per energy mode, 1-byte unsigned value. 5: 32;

Para	Data	DM_ID	Description
			6: 64; 7: 128; 8: 256; 9: 512.
XPARA_DFE_CARD_NUM	Returned value	None	Card number per X-DFE, 1-byte unsigned value. Range is 1~2, default is 1.
XPARA_CARD_TYPE	Returned value	None	Card type, 1-byte unsigned value. 0: LCS2 DM; 1: X-Card 0.2/0.4/0.8; 2: X-Card 1.5/1.6/2.5; 3: X-Card2 0.2/0.4/0.8; 4: Dual-row LCS2; 5: Aurora GT; 6: X-ICT; 7: Aurora B.
XPARA_EN_LED	Returned value	None	LED state, 1-byte unsigned value. 0: Disable; 1: Enable. Default is 1.
XPARA_TERMINAL_RESISTOR	Returned value	None	Terminal resistor state, 1-byte unsigned value. 0: Disable; 1: Enable. Default is 0.
XPARA_GCU_TYPE	Returned value	None	X-GCU type. 0: Ethernet; 1: Camera Link. Default is 0.

Para	Data	DM_ID	Description
XPARA_ENERGY_MODE	Returned value	None	Energy mode. 0: Single energy; 1: Dual energy stack; 2: Dual energy parallel.
XPARA_TRIGGER_STAMP_PARITY	Returned value	None	Trigger stamp parity check mode. 0: Disable parity check; 1: Use "odd" parity check; 2: Use "even" parity check
XPARA_POWER_UP_TIME	Returned value	None	System working time counter, which is increased by one step per 2 hours.
XPARA_GCU_WORK_TIME	Returned value	None	X-GCU's working time after each powering up.

int32_t XCommand::GetPara(uint32_t para, std::string &data, uint8_t dm_id=0)

int32_t XCommand::GetPara(uint32_t para, char* data_, uint8_t dm_id=0)

Parameters:

para Parameter type as defined in the following table.

data Returned value of specific parameter.

data_ Returned value of specific parameter.

dm_id DM module index which starts from 1. Some of the commands need this argument. If it is 0xFF, the X-GCU will apply the command to all DMs. Otherwise, the X-GCU only operates the specified DM module. The DM ID 1 represents the first DM module in the first X-GCU channel.

Return:

Returns 1 if successful, 0 no such parameter, -1 error.

Remark:

The user calls this function to get the specific parameter from the X-GCU. If any error occurs, the user can call the GetLastError() to trace the error.

Para	Data	DM_ID	Description
XPARA_GCU_SERIAL	Returned value	None	X-GCU's serial number, 32-byte ASCII code.
XPARA_DM_SERIAL	Returned value	DM index	DM's serial number, 32-byte ASCII code. The DM index cannot be 0xFF. It is not available while scanning. It is not supported by Aurora B.
XPARA_PRODUCT_INFO	Returned value	None	Product info, 128-byte ASCII code.

int32_t XCommand::GetPara(uint32_t para, float &temperature, float &humidity, float &v1, float &v2, float &v3, float &v4, float &v5, float &v6, float &v7, float &high_temp, uint8_t dm_id=0)

Parameters:

para — Parameter type as defined in the following table.

temperature — Returned temperature value of the X-GCU or DM.

humidity — Returned humidity value of the X-GCU or DM.

v1~v7 — Returned voltage values of the X-GCU or DM.

high_temp — High resolution temperature.

dm_id — DM module index which starts from 1. Some of the commands need this argument. If it is 0xFF, the X-GCU will apply the command to all DMs. Otherwise, the X-GCU only operates the specified DM module. The DM ID 1 represents the first DM module in the first X-GCU channel.

Return:

Returns 1 if successful, 0 no such parameter, -1 error.

Remark:

The user calls this function to get the specific parameter from the X-GCU. If any error occurs, the user can call the GetLastError() to trace the error.

Para	Returns	DM_ID	Description
XPARA_GCU_HEALTH	Temperature; Humidity; v1~v4.	None	Gets X-GCU's temperature (°C), relative humidity (%) and 4 voltage values. The correct voltage values should be as follows:

Para	Returns	DM_ID	Description
			v1: 24V±10% (power 24V) 12V±10% (power 12V); v2: 3.3V±5%; v3: 2.5V±5%; v4: 1.1V±5%. The relative humidity is not available with the X-GCU_STD.
XPARA_DM_HEALTH	Temperature; Humidity; v1~v7(v1~v4 for Aurora GT and no voltage is available for Aurora B); High resolution temperature.	DM index	Gets DM's temperature (°C), relative humidity (%) and 7 voltage values. The correct voltage values should be as follows: v1: 2.5V±5%; v2: 24V±10% (power 24V) 12V±10% (power 12V); v3: 5.2V±5%; v4: 1.2V±5%; v5: 5V±5%; v6: 3.3V±5%; v7: 4.096V±5%. For Aurora GT: v1: 24V±10% (power 24V) 12V±10% (power 12V); v2: 2.5V±5%; v3: 5V±5%; v4: 3.3V±5%. The high resolution temperature is only available with the DM which has high resolution sensor installed (high tier Digital X-Cards and LCS2 DM), otherwise the return is always 511.9. The DM index cannot be 0xFF. It is not available while scanning. This function is not supported by Aurora B.

bool XCommand::Open(XDevice* dev_)

Parameters:

dev_ XDevice object pointer which provides detector's info. See the XDevice class.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Allocates the resources needed. If it returns 0, the user can utilize the GetLastError() to trace the error.

void XCommand::RegisterEventSink(IXCmdSink* cmd_sink_)

Parameters:

cmd_sink_ IXCmdSink object pointer which is defined by the user.

Remark:

The IXCmdSink object handles error events. This function should be called before opening.

int32_t XCommand::SendAscCmd(std::string asc_send, std::string& asc_recv)

int32_t XCommand::SendAscCmd(char* asc_send_, char* asc_recv_)

Parameters:

asc_send_

asc_send Command string. Format: "[CODE,OPERATION,DM_ID,DATA]". Example: "[ST,W,0,3E8]", set integration time to be 1000; "[ST,R,0]", get integration time.

asc_recv_

asc_recv Returned ACK string. Format: "[FLAG, DATA]". Example: "[4]", error; "[0]", successful with no data; "[0, DATA]" successful with data.

Return:

Returns 1 if successful, -1 otherwise.

Remark:

Accepts an ASCII format command and translates it to hex format, then sends the hex command to the X-GCU. Please refer to the chapter 6 for the ASCII command list.

Wrapped by SetPara(), GetPara() and ExcutePare() functions. The user can utilize the three wrappers or call this function directly to send the command to the X-GCU.

int32_t XCommand::SendCommand(uint8_t cmd_code, uint8_t operation, uint8_t dm_id, uint16_t data_size, uint8_t* send_data_, uint8_t* recv_data_)

Parameters:

cmd_code	Command code.
operation	Operating code: none/0x00, write/0x01, read/0x02, save/0x03 or load/0x04.
dm_id	DM index: none/0x00, all/0xFF or specific ID number. Some of the commands need this argument. If it is 0xFF, the X-GCU will apply the command to all DMs. Otherwise, the X-GCU only operates the specified DM module. The DM ID 0x01 represents the first DM module in the first X-GCU channel.
data_size	Size of data section.
send_data_	Sending data buffer. The byte order is big-endian.
recv_data_	Receiving data buffer. The byte order is big-endian.

Return:

Returns -1 if an error occurs, else returns the size of receiving data.

Remark:

Accepts a hex format command. Please refer to the chapter 6 for the hex command list.

Example:

```
uint8_t send_data[128];

uint8_t recv_data[128];

//Set integration time to be 0x03E8 (1000us)

send_data[0] = 0x03;

send_data[1] = 0xE8;

command.SendCommand(0x20, 0x01, 0x00, 0x02, send_data, recv_data);
```

void XCommand::SetFactory(IXFactory* factory_)

Parameters:

factory_	IXFactory object pointer. See the XGigFactory class.

Remark:

If you are using a default constructor, it should be called before opening.

int32_t XCommand::SetPara(uint32_t para, uint64_t data, uint8_t dm_id=0)

Parameters:

para Parameter type as defined in the following table.

data Value to set

dm_id DM module index which starts from 1. Some of the commands need this argument. If it is 255, the X-GCU will apply the command to all DMs. Otherwise, the X-GCU only operates the specified DM module. The DM ID 1 represents the first DM module in the first X-GCU channel.

Return:

Returns 1 if successful, 0 no such parameter, -1 error.

Remark:

The user calls this function to set the specific parameter of the X-GCU. If any error occurs, the user can call the GetLastError() to trace the error.

Para	Data	DM_ID	Description
XPARA_INT_TIME	Integration time value	None	Integration time, 4-byte unsigned value. The unit is us. The range is defined by max and min integration time. Default is 3000.
XPARA_NON_INTTIME	Integration time in Non-continuous mode and in Constant mode	None	Integration time in Non-continuous mode and in Constant mode, 2-byte unsigned value. The unit is us. Default is 580.
XPARA_OPE_MODE	Operation mode value	None	Operation mode, 1-byte unsigned value. 0: Continuous mode; 1: Non-continuous mode 2: Constant integration time mode; 3: Non-continuous with Hi/Lo trigger.

Para	Data	DM_ID	Description
			Default is 0.
XPARA_DM_GAIN	Gain value	DM index. 0xFF: All DMs; others above 0: specific DM.	DM module gain value, 2-byte unsigned value. The high 8 bits are high energy gain, the low 8 bits are low energy gain. Default is 0x0606. LCS2: 1~63; X-Card 0.2/0.4/0.8: 1~2; X-Card 1.5/1.6/2.5: 1~12; X-Card2 0.2/0.4: 1~2; X-Card2 0.8: 1~8; Aurora GT/B: 1~127. It is not available while scanning.
XPARA_HILO_MODE	HI/LO mode value	None	Defines which DM module gain to launch in single energy mode. 1-byte unsigned value. 0: Low-energy gain; 1: High-energy gain. Default is 0. It is only available while XPARA_OPE_MODE is 3.
XPARA_CH_NUM	Channel card number	None	DM module number in each X-GCU's channel. It is a five-byte value. The first channel number is the highest byte, and the fifth channel number is the lowest byte.
XPARA_EN_SCAN	Scanning state value	None	Scanning state, 1-byte unsigned value. 0: Stopping; 1: Scanning.
XPARA_EN_GAIN_CORRECT	Gain correcting state value	None	Gain correcting state, 1-byte unsigned value. Enables or disables channel

Para	Data	DM_ID	Description
			gain correction. 0: Disable; 1: Enable. Default is 0.
XPARA_EN_OFFSET_CORRECT	Offset correcting state value	None	Offset correcting state, 1-byte unsigned value. Enables or disables channel offset correction. 0: Disable; 1: Enable. Default is 0.
XPARA_EN_BASELINE_CORRECT	Base line correcting state value	None	Base line correcting state, 1-byte unsigned value. 0: Disable; 1: Enable. Default is 0.
XPARA_BASE_LINE	Baseline value	None	Base line value, 2-byte unsigned value. Range 0 – 1000. Default is 0.
XPARA_BINNING_MODE	Binning mode value	None	Pixel binning mode, 1-byte unsigned value. 0: Original pixel; 1: Averaging every 2 pixels; 2: Summing every 2 pixels; 3: Averaging every 4 pixels; 4: Summing every 4 pixels. Default is 0.
XPARA_AVERAGE_MODE	Line averaging filter mode value	None	Line averaging filter mode, 1-byte unsigned value. 0: None;

Para	Data	DM_ID	Description
			1: 2 lines; 2: 4 lines; 3: 8 lines; 4: 16 lines; 5: 32 lines; 6: 64 lines; 7: 128 lines; 8: 256 lines. Default is 0.
XPARA_SUMMING_MODE	Line summing filter mode value	None	Line summing filter mode, 1-byte unsigned value. 0: None; 1: 2 lines; 2: 4 lines; 3: 8 lines; 4: 16 lines; 5: 32 lines; 6: 64 lines; 7: 128 lines; 8: 256 lines. Default is 0.
XPARA_OUTPUT_SCALE	Output scale value	None	Output scale, 1-byte unsigned value. The output pixel depth will be original bit subtracting the scale value. 0: Original; n: Original – n. Default is 0. E.g., the original bit is 16, the scale value is 8, the output pixel bits will be 8.

Para	Data	DM_ID	Description
XPARA_OFFSET_AVERAGE	Offset averaging filter value	None	Offset averaging filter, 1-byte unsigned value. 0: None; 1: 2 lines; 2: 4 lines; 3: 8 lines. Default is 0. It is only available while XPARA_OPE_MODE is 3.
XPARA_LINE_TRIGGER_MODE	External line trigger mode value	None	External line trigger mode, 1-byte unsigned value. 0: Rising edge; 1: Falling edge; 2: Sync trigger stamp mode; 3: Async trigger stamp mode. Default is 0.
XPARA_EN_LINE_TRIGGER	External line trigger state value	None	Enables external line trigger function, 1-byte unsigned value. 0: Disable; 1: Enable. Default is 0.
XPARA_LINETRG_FINE_DELAY	External line trigger fine delay value	None	External line trigger fine delay, 1-byte unsigned value. The unit is 0.125us. Default is 0.
XPARA_LINETRG_RAW_DELAY	External line trigger raw delay value	None	External line trigger raw delay, 1-byte unsigned value. The unit is 64us. Default is 0.
XPARA_FRAME_TRIGGER_MODE	External frame trigger mode value	None	External frame trigger mode, 1-byte unsigned

Para	Data	DM_ID	Description
			value. 0: Rising edge; 1: Falling edge. Default is 0.
XPARA_EN_FRAME_TRIGGER	External frame trigger state value	None	Enables external frame trigger function, 2-byte unsigned value. 0: Disable; n: Enable, and output 32*n lines on each trigger. Default is 0.
XPARA_FRAME_TRIGGER_DELAY	External frame trigger delay value	None	External frame trigger raw delay, 1-byte unsigned value. The unit is 32 lines. Default is 0.
XPARA_HILO_EDGE	Hi/Lo trigger edge mode value	None	Hi/Lo line trigger with hi/lo edge level detection, 1-byte unsigned value. 0: Low level; 1: High level. Default is 0.
XPARA_HILO_SAMPLE_POS	Sampling position value	None	Hi/Lo line trigger sampling position, 1-byte unsigned value. The unit is 1us. Range is 0-255, default is 0.
XPARA_MAXMIN_INTTIME	Max/min integration time value	None	Max and min integration time, 8-byte unsigned value. The high four bytes are the max value, the low four bytes are the min value.
XPARA_GCU_TEST_MODE	GCU's test pattern mode value	None	X-GCU's test pattern mode, 1-byte unsigned value. 0: Disable;

Para	Data	DM_ID	Description
			1: Normal test pattern; 2: Slope test pattern. Default is 0.
XPARA_DM_TEST_MODE	DM's test pattern mode value	DM index. 0xFF: All DMs; others above 0: specific DM.	DM's test pattern mode, 1-byte unsigned value. 0: Normal image data mode; 1: Enable DM board fixed value (AAAAA[Hex]) test mode.; 2: Enable I'm here function, Light the LED of the specific DM. Make sure the LED must enabled. Default is 0. It is not available while scanning.
XPARA_EN_LED	LED state value	None	LED state, 1-byte unsigned value. 0: Disable; 1: Enable. Default is 1.
XPARA_TERMINAL_RESISTOR	Terminal resistor state value	None	Terminal resistor state, 1-byte unsigned value. 0: Disable; 1: Enable. Default is 0.
XPARA_TRIGGER_STAMP_PARITY	Trigger stamp parity mode value	None	Trigger stamp parity check mode, 1-byte unsigned value. 0: Disable parity check; 1: Use "odd" parity check; 2: Use "even" parity check

int32_t XCommand::SetPara(uint32_t para, std::string data, uint8_t dm_id=0)

Parameters:

para Parameter type as defined in the following table.

data Value to set.

dm_id DM module index which starts from 1. Some of the commands need this argument. If it is 255, the X-GCU will apply the command to all DMs. Otherwise, the X-GCU only operates the specified DM module. The DM ID 1 represents the first DM module in the first X-GCU channel.

Return:

Returns 1 if successful, 0 no such parameter, -1 error.

Remark:

The user calls this function to set the specific parameter of the X-GCU. If any error occurs, the user can call the GetLastError() to trace the error.

Para	Data	DM_ID	Description
XPARA_GCU_SERIAL	Serial number	None	X-GCU's serial number, 32-byte ASCII code. It is not available for the user.
XPARA_DM_SERIAL	Serial number	DM index	DM's serial number, 32-byte ASCII code. The DM index cannot be 0xFF. It is not available for the user.
XPARA_PRODUCT_INFO	Product info	None	Product info, 128-byte ASCII code. It is not available for the user.

void XCommand::SetTimeout(uint32_t time)

Parameters:

time Timeout value.

Remark:

If the ACK does not arrive during the timeout period after sending a command, it will report an error. The timeout unit is ms, the default value is 20000, and the range is from 12000 to 2000000.

bool XCommand::StartHeartbeat()

Return:

Returns 1 if successful, 0 otherwise.

Remark:

If it starts correctly, the X-GCU will keep sending the heartbeat packet by period of 1s. If it cannot get the heartbeat 10 times in a row, it will report the error XERROR_CMD_HEARTBEAT_FAIL. The user should check the NET cable or X-GCU in that case.

bool XCommand::StopHeartbeat()

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Stops the X-GCU sending the heartbeat packet. If it forces the thread exiting it will report the error XERROR_CMD_HEARTBEAT_STOP_ABNORMAL. In that case, it cannot start again.

XDevice class

Description:

The XDevice class wraps the basic parameters of the detector. It is used as an argument in the initializing process of the XCommand and XAcquisition objects.

Functions:

uint32_t XDevice::GetCardNumber()

Return:

Returns the total DM number.

uint32_t XDevice::GetCardType()

Return:

Returns the type number of the X-Card.

0: LCS2 DM;

1: X-Card 0.2/0.4 and 0.8;

2: X-Card 1.5/1.6 and 2.5;

3: X-Card2 0.2/0.4 and 0.8;

4: Dual-row LCS2 DM;

5: Aurora GT;

6: X-ICT;

7: Aurora B.

uint16_t XDevice::GetCmdPort()

Return:

Returns the command channel port number of the X-GCU.

uint32_t XDevice::GetDMPixelNumber()

Return:

Returns the pixel number of the DM for one energy level.

5: 32;

6: 64;

7: 128;

8: 256;

9: 512.

uint16_t XDevice::GetImgPort()

Return:

Returns the image channel port number of the X-GCU.

const char* XDevice::GetIP()

Return:

Returns the IP address string of the X-GCU.

uint8_t* XDevice::GetMAC()

Return:

Returns the 6-byte MAC address array of the X-GCU.

uint32_t XDevice::GetOPMode()

Return:

Returns the operational mode.

0: Continuous mode;

1: Non-continuous mode;

2: Constant integration time mode;

3: Non-continuous with Hi/Lo trigger.

uint32_t XDevice::GetPixelDepth()

Return:

Returns the pixel depth of the X-GCU.

uint32_t XDevice::GetPixelNumber()

Return:

Returns the total pixel number of the X-GCU.

uint64_t XDevice::GetSerialNum()

Return:

Returns the serial number of the X-GCU.

XSystem* XDevice::GetSystem()

Return:

Returns the system object pointer which creates the XDevice object.

void XDevice::SetMAC(uint8_t* mac_)

Parameters:

mac_ 6-byte array.

Remark:

Sets the MAC address of the X-GCU.

void XDevice::SetCmdPort(uint16_t port)

Parameters:

port Command channel port number.

Remark:

Sets the command channel port number of the X-GCU.

void XDevice::SetImgPort(uint16_t port)

Parameters:

port Image channel port number.

Remark:

Sets the image channel port number of the X-GCU.

void XDevice::SetIP(const char* ip)

Parameters:

ip IP address string.

Remark:

Sets the IP address of the X-GCU.

XDisplay class

Description:

The XDisplay class is in charge of displaying raw image data, which is only available in the Windows system.

#include “xdisplay.h”

Constructor:

XDisplay::XDisplay()

Remark:

Default constructor.

Functions:

void XDisplay::Close()

Remark:

Releases all resources.

void XDisplay::Display(XImage* image_)

Parameters:

image_ XImage object pointer. See the class XImage.

Remark:

Displays the raw image data.

float XDisplay::GetGama()

Return:

Returns the gama value.

bool XDisplay::GetIsDDrawEnable()

Return:

Returns 1 if direct draw enabled, 0 otherwise.

Remark:

Used to check whether the direct draw is enabled after opening. If the direct draw is not enabled, it will use the GDI to display the image data.

uint32_t XDisplay::GetLastError()

Return:

Returns the error ID if any exception occurs. Please refer to the error list for the explanation of the specific error.

bool XDisplay::GetIsOpen()

Return:

Returns the state, 1 opening, 0 closing.

bool XDisplay::Open(uint32_t width, uint32_t height, uint32_t pixel_depth, HWND hwnd, uint32_t color_mode=0)

bool XDisplay::Open(XDevice* dev_, uint32_t line_num, HWND hwnd, uint32_t color_mode=0)

Parameters:

width	Image width.
height	Image height.
pixel_depth	Pixel depth.
hwnd	Window handler.
color_mode	Pseudo color. 0: Gray; 1: Sin pseudo color; 2: Cos pseudo color; 3: Hot pseudo color; 4: Jet pseudo color.
dev_	XDevice object pointer.
line_num	The same with the image height.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Allocates the needed resources. If it returns 0, the user can utilize the GetLastError() to trace the error.

void XDisplay::SetGama(float gama_val)

Parameters:

gama_val	Gama value, range 1.0 – 4.0.

Remark:

Sets the gama value.

XDualEngCorrect class

Description:

The dual energy line data of the interlaced dual-energy raw data sub-system comes according to the sequence of the external line trigger. For most applications, the Hi/Lo trigger comes alternatively, so the Hi/Lo line data comes in pairs. In that case, the XDualEngCorrect object combines two lines into one line where the Lo data is on the left, the Hi data is on the right.

Before using this correction, the line info mode of the XAcquisition object must be enabled.

#include "xdual_eng_correct.h"

Constructor:

XDualEngCorrect()

Remark:

Default constructor.

Functions:

bool XDualEngCorrect::DoCorrect(XImage* image_)

Parameters:

image_ XImage object pointer.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

If successful, the user should call the GetImage() to get the combined image. If the allocation of the combined image fails, it returns 0.

XImage* XDualEngCorrect::GetImage()

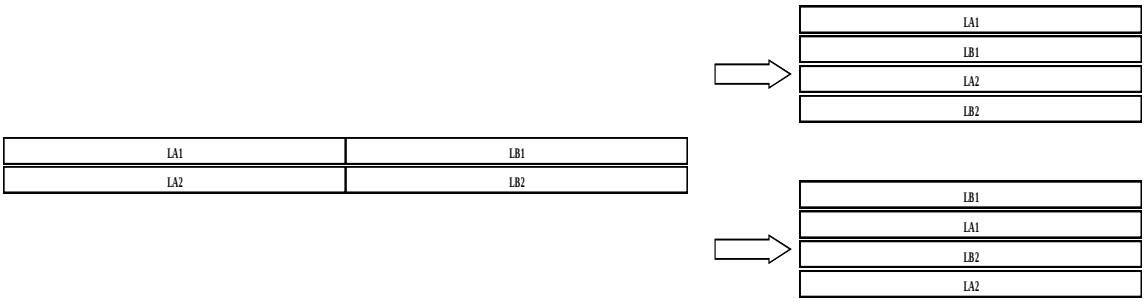
Return:

Returns the combined image.

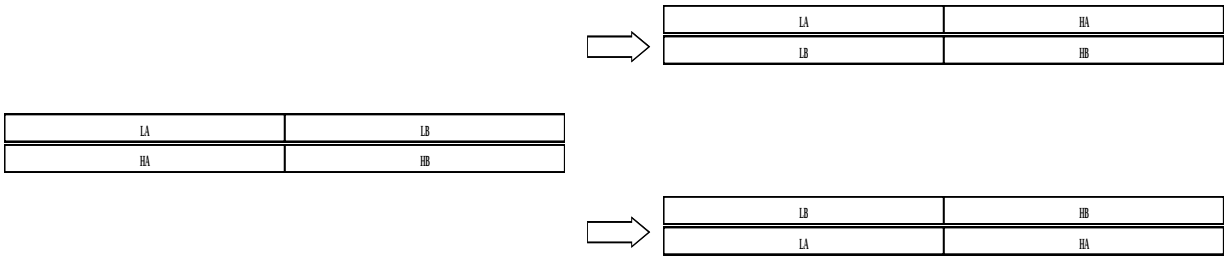
XDualLineCorrect class

Description:

For the dual-line detector sub-system, two lines come together. The XDualLineCorrect object can re-arrange the sequence and the position of the two line data. In single energy mode, the two lines can be arranged into two cases according to the direction flag, as in the following figure.



In dual energy mode, if Hi/Lo line data comes in pairs, the two energy data of two lines are arranged as in the following figure. .



Before using this correction, the line info mode of the XAcquisition object must be enabled.

#include "xdual_line_correct.h"

Constructor:

XDualLineCorrect()

Remark:

Default constructor.

Functions:

bool XDualLineCorrect::DoCorrect(XImage* image_)

Parameters:

image_ XImage object pointer.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

If successful, the user should call the GetImage() to get the combined image. If the allocation of the combined image fails, it returns 0.

XImage* XDualLineCorrect::GetImage()

Return:

Returns the combined image.

void XDualLineCorrect::SetDirect(bool dir)

Parameters:

dir Direction flag.

Remark:

Sets the sequence of the arrangement of the two lines.

void XDualLineCorrect::SetDualEnergy(bool eng)

Parameters:

eng Energy flag.

Remark:

Sets the energy mode of the coming data to deal with.

XFrameTransfer class

Description:

The XFrameTransfer class receives raw line data from the XAcquisition class and combines lines into a frame. The user can get the frame data from its callback function.

#include "xframe_transfer.h"

Constructor:

XFrameTransfer::XFrameTransfer(uint32_t line_num)

XFrameTransfer::XFrameTransfer()

Parameters:

line_num Line number of frame.

Remark:

For the default constructor, the default line number is 1024.

Functions:

bool XFrameTransfer::GetIsRunning()

Return:

Returns the working state, 1 grabbing, 0 stopping.

uint32_t XFrameTransfer::GetLastError()

Return:

Returns the error ID if any exception occurs. Please refer to the error list for the explanation of the specific error.

uint32_t XFrameTransfer::GetLineNum()

Return:

Returns the line number of frame.

void XFrameTransfer::RegisterEventSink(IXImageSink* img_sink_)

Parameters:

img_sink_ IXImgSink object pointer which is defined by the user.

Remark:

The IXImgSink object handles the error event and frame ready event. This function should be called before opening the XAcquisition object.

void XFrameTransfer::SetLineNum(uint32_t line_num)

Parameters:

line_num Line number of frame.

Remark:

Changes the line number of the frame after the construction, which should be called before opening the XAcquisition object.

XGigFactory

Description:

The XGigFactory class is responsible for allocating all resources. Then the user has to define the factory object and pass it to the other objects needing it.

#include"xgig_factory.h"

Constructor:

XGigFactory::XGigFactory()

Remark:

Default constructor.

XImage class

Description:

The XImage class wraps the frame data and the information of the frame.

Functions:

Uint8_t* XImage::GetLineAddr(uint32_t line_num)

Parameters:

line_num Line index.

Return:

Returns the address of the specified line.

Remark:

If the line index argument is out of range, it returns NULL.

uint32_t XImage::GetPixelVal(uint32_t row, uint32_t col)

Parameters:

row Vertical coordinate.

col Horizontal coordinate.

Return:

Returns the pixel value.

bool XImage::Save(const char* file_name)

Parameters:

file_name_ ".txt" file.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Saves the image data as a ".txt" file, supports only the ".txt" file.

void XImage::SetPixelVal(uint32_t row, uint32_t col, uint32_t pixel_value)

Parameters:

row	Vertical coordinate.
col	Horizontal coordinate.
pixel_value	Pixel value to set.

Remark:

Sets the pixel value.

Public attributes:

Uint8_t* _data_

Remark:

Frame data pointer.

uint32_t _data_offset

Remark:

The position of the first pixel value is 0 by default. If using the line info mode, the line information is attached on the front of each line. Please see the XAcquisition:: EnableLineInfo(bool enable) function.

uint32_t _height

Remark:

The line number of the frame.

uint32_t _pixel_depth

Remark:

The pixel depth of the frame.

uint32_t _size

Remark:

The size of the frame, byte is the unit.

uint32_t _width

Remark:

The column number of the frame.

XMultiTransfer class

Description:

The XMultiTransfer class synchronizes multi-FrameTransfer objects while connecting with more than one X-GCUs. It combines the sub-frames into one frame. The XAcquisition object should enable the line info mode while working with the XMultiTransfer object.

#include "xmulti_transfer.h"

Constructor:

XMultiTransfer::XMultiTransfer()

Remark:

Default constructor.

Functions:

bool XMultiTransfer::AddTransfer(IXTransfer* transfer_)

Parameters:

transfer_ IXTransfer object pointer.

Return:

Returns 1 if successful, 0 if total attached transfer object number is above 8 or the line number of each transfer object is not equal.

Remark:

Attaches the transfer objects into the list. The line number of each transfer object must be 2^n (n < 15). This function should be called before opening.

void XMultiTransfer::Close()

Remark:

Releases all resources.

uint32_t XMultiTransfer::GetLastError()

Return:

Returns the error ID if any exception occurs. Please refer to the error list for the explanation of the specific error.

bool XMultiTransfer::Open()

Return:

Returns 1 if successful, else return 0.

Remark:

To use the XMultiTransfer class, the XAcquisition must enable line info mode.

void XMultiTransfer::RegisterEventSink(IXImageSink* img_sink_)

Parameters:

img_sink_ IXImgSink object pointer which is defined by the user.

Remark:

The IXImgSink object handles the error event and frame ready event. This function should be called before opening.

XOffCorrect class

Description:

The XOffCorrect class is in charge of the off-board or on-board correction functions. It manages the calculation process of gain and offset and saves the result to a disk or the flash of X-GCU board. If you are using non-continuous hi/lo trigger mode, please refer to XOffHLCorrect class.

For the workflow of correction, please refer to chapter 8.

#include "xoff_correct.h"

Constructor:

XOffCorrect::XOffCorrect()

Remark:

Default constructor.

Functions:

bool XOffCorrect::CalculatePara(uint32_t type, XAcquisition* acq_, XFrameTransfer* transfer_ , uint32_t start, uint32_t end, uint32_t target)

Parameters:

type Define calculating which correction, 0 gain, 1 offset.

acq_ XAcquistion object pointer.

transfer_ XFrameTransfer object pointer.

start	Starting channel of correcting area.
end	Ending channel of correcting area.
target	0: Uses min channel value as the target; 1: Uses mean channel value as the target; 2: Uses max channel value as the target; Other: Uses the specific value as the target.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Calculates the gain or offset parameters.

bool XOffCorrect::CalculatePara(uint32_t type, XAcquisition* acq_, XFrameTransfer* transfer_, uint32_t target)

Remark:

Calculates all the channels (Please refer to the previous function).

void XOffCorrect::Close()

Remark:

Releases all the allocated resources.

bool XOffCorrect::DoCorrect(XImage* image_)

Parameters:

image_	XImage object pointer.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Dose off-board correction. User can use it to correct image data in frame-ready callback.

void XOffCorrect::LoadFile(const char* file_)

Parameters:

file_	File name which saves gain and offset parameters.

Remark:

Loads the gain and offset parameters from the file.

bool XOffCorrect::LoadFlash(XCommand* cmd_, uint8_t index)

Parameters:

cmd_ XCommand object pointer.

index Gain index, rang 0-3, default 0.

Return:

Returns 1 if successful, otherwise 0.

Remark:

Loads the gain and offset parameters from the flash of X-GCU board.

bool XOffCorrect::Open(XDevice* dev_)

Parameters:

dev_ XDevice object pointer.

Return:

Returns 1 if successful, otherwise 0.

Remark:

Allocates the needed resources.

void XOffCorrect::Reset()

Remark:

Sets the gain to 1.0, offset to 0.

void XOffCorrect::SaveFile(const char* file_)

Parameters:

file_ File name which saves gain and offset parameters.

Remark:

Saves the gain and offset parameters to the file.

bool XOffCorrect::SaveFlash(XCommand* cmd_, uint8_t index)

Parameters:

cmd_ XCommand object pointer.

index Gain index, rang 0-3, default 0.

Return:

Returns 1 if successful, otherwise 0.

Remark:

Saves the gain and offset parameters to the flash of X-GCU board. It can save 4 groups of gain by using different gain index.

XOffHLCorrect class

Description:

The XOffHLCorrect class is in charge of the off-board correction functions only in non-continuous hi/lo trigger mode.

For the workflow of correction, please refer to chapter 8.

#include "xoff_hilo_correct.h"

Constructor:

XOffHLCorrect::XOffHLCorrect()

Remark:

Default constructor.

Functions:

bool XOffHLCorrect::CalculatePara(uint32_t type, XAcquisition* acq_, XFrameTransfer* transfer_ , uint32_t start, uint32_t end, uint32_t target)

Parameters:

type	Define calculating which correction, 0 gain, 1 offset.
acq_	XAcquistion object pointer.
transfer_	XFrameTransfer object pointer.
start	Starting channel of correcting area.
end	Ending channel of correcting area.
target	The target value to be corrected.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Calculates the gain or offset parameters.

bool XOffHLCorrect::CalculatePara(uint32_t type, XAcquisition* acq_, XFrameTransfer* transfer_, uint32_t target)

Remark:

Calculates all the channels (Please refer to the previous function).

void XOffHLCorrect::Close()

Remark:

Releases all the allocated resources.

bool XOffHLCorrect::DoCorrect(XImage* image_)

Parameters:

image_ XImage object pointer.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Corrects the gain and offset.

void XOffHLCorrect::LoadFile(const char* file_)

Parameters:

file_ File name which saves gain and offset parameters.

Remark:

Loads the gain and offset parameters from the file.

bool XOffHLCorrect::Open(XDevice* dev_)

Parameters:

dev_ XDevice object pointer.

Return:

Returns 1 if successful, otherwise 0.

Remark:

Allocates the needed resources.

void XOffHLCorrect::Reset()

Remark:

Sets the gain to 1.0, offset to 0.

void XOffHLCorrect::SaveFile(const char* file_)

Parameters:

file_ File name which saves gain and offset parameters.

Remark:

Saves the gain and offset parameters to the file.

XPiecewiseCorrect class

Description:

The XPiecewiseCorrect object takes three air images corresponding to low, middle and high X-ray to build a more accurate piecewise correcting function. It only supports the off-board correction.

For the workflow of correction, please refer to chapter 8.

#include "xpiecewise_correct.h"

Constructor:

XPiecewiseCorrect::XPiecewiseCorrect()

Remark:

Default constructor.

Functions:

bool XPiecewiseCorrect::CalculateGain1(XAcquisition* acq_, XFrameTransfer* transfer_, uint32_t target)

Parameters:

acq_ XAcquisition object pointer.

transfer_ XFrameTransfer object pointer.

target The target value for low X-ray.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Calculates the gain parameters of low X-ray.

bool XPiecewiseCorrect::CalculateOffset1(XAcquisition* acq_, XFrameTransfer* transfer_)

Parameters:

acq_ XAcquisition object pointer.

transfer_ XFrameTransfer object pointer.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Calculates the offset parameters of low X-ray.

bool XPiecewiseCorrect::CalculatePara2(XAcquisition* acq_, XFrameTransfer* transfer_, uint32_t target)

Parameters:

acq_ XAcquisition object pointer.

transfer_ XFrameTransfer object pointer.

target The target value for middle X-ray.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Calculates the gain and offset parameters of middle X-ray.

bool XPiecewiseCorrect::CalculatePara3(XAcquisition* acq_, XFrameTransfer* transfer_, uint32_t target)

Parameters:

acq_ XAcquisition object pointer.

transfer_ XFrameTransfer object pointer.

target The target value for high X-ray.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Calculates the gain and offset parameters of high X-ray.

void XPiecewiseCorrect::Close()

Remark:

Releases all the allocated resources.

bool XPiecewiseCorrect::DoCorrect(XImage* image_)

Parameters:

image_ XImage object pointer.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Corrects the image data with piecewise parameters.

void XPiecewiseCorrect::LoadFile(const char* file_)

Parameters:

file_ File name which saves gain and offset parameters.

Remark:

Loads the gain and offset parameters from the file.

bool XPiecewiseCorrect::Open(XDevice* dev_)

Parameters:

dev_ XDevice object pointer.

Return:

Returns 1 if successful, otherwise 0.

Remark:

Allocates the needed resources.

void XPiecewiseCorrect::SaveFile(const char* file_)

Parameters:

file_ File name which saves gain and offset parameters.

Remark:

Saves the gain and offset parameters to the file.

XPixelCorrect class

Description:

The XPixelCorrect object corrects the bad pixels.

#include "xpixel_correct.h"

Constructor:

XPixelCorrect::XPixelCorrect()

Remark:

Default constructor.

Functions:

bool XPixelCorrect::DoCorrect(XImage* image_)

Parameters:

image_ XImage object pointer.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Corrects the bad pixels.

void XPixelCorrect::PushPixel(uint32_t pixel_pos)

Parameters:

pixel_pos Bad pixel position.

Remark:

Places the bad pixel position into a list.

void XPixelCorrect::Refresh()

Remark:

Cleans the bad pixel list.

XReferenceCorrect class

Description:

The XReferenceCorrect object provides function to correct the signal variations caused by LINAC X-ray source.

#include "xreference_correct.h"

Constructor:

XReferenceCorrect::XReferenceCorrect()

Remark:

Default constructor.

Functions:

bool XReferenceCorrect::CalculatePara(XAcquisitioin* acq_, XFrameTransfer* transfer_)

Parameters:

acq_ XAcquisition object pointer.

transfer_ XFrameTransfer object pointer.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Calculates the correcting parameters.

void XReferenceCorrect::Close()

Remark:

Releases all the allocated resources.

bool XReferenceCorrect::DoCorrect(XImage* image_)

Parameters:

image_ XImage object pointer.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Corrects the image with signal variations.

bool XReferenceCorrect::Open(XDevice* dev_, uint32_t ref_num)

Parameters:

dev_ XDevice object pointer.

Ref_num Referenced pixel number.

Return:

Returns 1 if successful, otherwise 0.

Remark:

Allocates the needed resources.

XSystem class

Description:

The XSystem object communicates with the X-GCU by broadcast. It enumerates the IP, MAC and port info of the X-GCU and also configures the network settings.

#include "xsystem.h"

Constructor:

XSystem::XSystem(const char* local_ip_, uint16_t local_port = 0, uint32_t timeout = XSYS_TIMEOUT)

XSystem::XSystem()

Parameters:

local_ip_ Local IP address to band.

loca_port Local port number to bind, default is random.

timeout Time out value, default is 2000ms.

Remark:

The XSystem object must bind with the fixed IP address. If using the default constructer, the SetLocalIP() function must be called before opening.

Functions:

void XSystem::Close()

Remark:

Releases all the resources.

int32_t XSystem::ConfigureDevice(const XDevice* dev_)

Parametares:

dev_ XDevice object pointer.

Return:

Returns -1 if an error occurs, 1 successful.

Remark:

Configures the IP, MAC and port of X-GCU.

void XSystem::EnableLog(bool enable)

Parameters:

enable Flag to enable log function: 0 disable, 1 enable.

Remark:

Enables or disables the log function; the default state is enabled. If it disables the log file, the log file should be called before opening.

int32_t XSystem::FindDevice()

Return:

Returns -1 if an error occurs, else device number found.

Remark:

Enumerates the X-GCU devices which connect with the bound network adapter.

XDevice* XSystem::GetDevice(uint32_t device_index)

Parameters:

device_index The device number index, start from 0.

Return:

Returns the XDevice object pointer. If the device index is out of range, returns NULL.

size_t GetDeviceNum()

Return:

Returns the device number found.

bool XSystem::GetIsOpen()

Return:

Returns the state, 1 opening, 0 closing.

uint32_t XSystem::GetLastError()

Return:

Returns the error ID if any exception occurs. Please refer to the error list for the explanation of the specific error.

bool XSystem::Open()

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Allocates the needed resources. If it returns 0, the user can utilize the GetLastError() to trace the error.

int32_t XSystem::RecoverDevice()

Return:

Returns -1 if an error occurs, 1 successful.

Remark:

Recovers IP, MAC and port configurations to the default value. IP: 192.168.1.2, Cmd Port: 3000, Img Port: 4001.

Void XSystem::RegisterEventSink(IXCmdSink* cmd_sink)

Parameters:

cmd_sink_ IXCmdSink object pointer which is defined by the user.

Remark:

The IXCmdSink object handles the error event. This function should be called before opening.

Void XSystem::SetLocalIP(const char* ip_addr_)

Parameters:

ip_addr_ Local IP address string.

Remark:

If using the default constructor, the local IP address must be set before opening.

Void XSystem::SetTimeout(uint32_t time)

Parameters:

time Timeout value.

Remark:

The timeout unit is ms with the default value 2000. The range is from 0 to 2000000. It should be called before opening.

XTifFormat class

Description:

The XTifFormat object could save some properties of detector into a tif format image, and also could load and parse the image.

#include "xtif_format.h"

Constructor:

XTifFormat::XTifFormat(XImage* image_, XDevice* dev_)

XTifFormat::XTifFormat()

Parameters:

image_ Pointer of XImage object.

dev_ Pointer of XDevice object.

Remark:

The XTifFormat object can get image data and detector properties such as image width, image height, SN, number of X-DFE and so on by the constructor with parameters.

If using the default constructor, it needs to set image data and detector properties by function SetPara() before saving image.

If using XTifForamt object to load a tif file, you have to use the default constructor.

Functions:

bool XTifFormat::GetPara(uint32_t para, uint32_t &data)

Parameters:

para Parameter type as defined in the following table.

data Returned value of the specific parameter.

Return:

Returns 1 if successful, 0 no such parameter.

Remark:

The user calls this function to get the specific parameter from the object.

Para	Data	Description
XTIF_PARA_WIDTH	Returned value	Image width.
XTIF_PARA_HEIGHT	Returned value	Image height.

Para	Data	Description
XTIF_PARA_PIXEL_DEPTH	Returned value	Image pixel depth.
XTIF_PARA_CARD_NUM	Returned value	The number of X-Cards.
XTIF_PARA_CARD_TYPE	Returned value	Card type, 0: LCS2 DM; 1: X-Card 0.2/0.4/0.8; 2: X-Card 1.5/1.6/2.5; 3: X-Card2 0.2/0.4/0.8; 4: Dual-row LCS2; 5: Aurora GT; 6: X-ICT; 7: Aurora B.
XTIF_PARA_DM_PIXEL	Returned value	DM's pixel number per energy mode. 5: 32; 6: 64; 7: 128; 8: 256; 9: 512.
XTIF_PARA_OP_MODE	Returned value	Operation mode. 0: Continuous mode; 1: Non-continuous mode 2: Constant integration time mode; 3: Non-continuous with Hi/Lo trigger.
XTIF_PARA_INT_TIME	Returned value	Integration time.
XTIF_PARA_ENERGY_MODE	Returned value	Energy mode. 0: Single energy; 1: Dual energy stack; 2: Dual energy parallel.

Para	Data	Description
XTIF_PARA_BIN_MODE	Returned value	Binning mode. 0: Original pixel; 1: Averaging every 2 pixels; 2: Summing every 2 pixels; 3: Averaging every 4 pixels; 4: Summing every 4 pixels

bool XTifFormat::GetPara(uint32_t para, float &data)

Parameters:

para Parameter type as defined in the following table.

data Returned value of the specific parameter.

Return:

Returns 1 if successful, 0 no such parameter.

Remark:

The user calls this function to get the specific parameter from the object.

Para	Data	Description
XTIF_PARA_TEMPER	Returned value	The temperature of X-GCU
XTIF_PARA_HUMIDITY	Returned value	The humidity of X-GCU.

bool XTifFormat::GetPara(uint32_t para, uint8_t** data_)

Parameters:

para Parameter type as defined in the following table.

data_ Returned value of the specific parameter.

Return:

Returns 1 if successful, 0 no such parameter.

Remark:

The user calls this function to get the specific parameter from the object.

Para	Data	Description
XTIF_PARA_DATA	Returned value	The image data pointer.
XTIF_PARA_SN	Returned value	The serial number string pointer.

Para	Data	Description
		The string length is less than 32 bytes.
XTIF_PARA_DATE_TIME	Returned value	The date time string pointer. The string length is less than 32 bytes.

bool XTifFormat::Load(const char* file_)

Parameters:

file_ ".tif" image file.

Return:

Returns 1 if successful, 0 failed.

Remark:

Reads and parses tif format image file.

bool XTifFormat::Save(const char* file_);

Parameters:

file_ ".tif" image file.

Return:

Returns 1 if successful, 0 failed.

Remark:

Saves data as tif format image file.

bool XTifFormat::SetPara(uint32_t para, uint32_t data)

Parameters:

para Parameter type as defined in the following table.

data Value to set.

Return:

Returns 1 if successful, 0 no such parameter.

Remark:

The user calls this function to set the specific parameter of the object.

Para	Data	Description
XTIF_PARA_WIDTH	Image width	Image width.
XTIF_PARA_HEIGHT	Image height	Image height.

Para	Data	Description
XTIF_PARA_PIXEL_DEPTH	Pixel depth	Image pixel depth.
XTIF_PARA_CARD_NUM	Card number	The number of X-Cards.
XTIF_PARA_CARD_TYPE	Card type	Card type, 0: LCS2 DM; 1: X-Card 0.2/0.4/0.8; 2: X-Card 1.5/1.6/2.5; 3: X-Card2 0.2/0.4/0.8; 4: Dual-row LCS2; 5: Aurora GT; 6: X-ICT; 7: Aurora B.
XTIF_PARA_DM_PIXEL	DM pixel number	DM's pixel number per energy mode. 5: 32; 6: 64; 7: 128; 8: 256; 9: 512.
XTIF_PARA_OP_MODE	Operation mode	Operation mode. 0: Continuous mode; 1: Non-continuous mode 2: Constant integration time mode; 3: Non-continuous with Hi/Lo trigger.
XTIF_PARA_INT_TIME	Integration time	Integration time.
XTIF_PARA_ENERGY_MODE	Energy mode	Energy mode. 0: Single energy; 1: Dual energy stack; 2: Dual energy parallel.

Para	Data	Description
XTIF_PARA_BIN_MODE	Binning mode	Binning mode. 0: Original pixel; 1: Averaging every 2 pixels; 2: Summing every 2 pixels; 3: Averaging every 4 pixels; 4: Summing every 4 pixels

bool XTifFormat::SetPara(uint32_t para, uint8_t* data_)

Parameters:

para Parameter type as defined in the following table.

data_ Value to set.

Return:

Returns 1 if successful, 0 no such parameter.

Remark:

The user calls this function to set the specific parameter of the object.

Para	Data	Description
XTIF_PARA_DATA	Image data pointer	The image data pointer.
XTIF_PARA_SN	Serial number string pointer	The serial number string pointer. The string length is less than 32 bytes.
XTIF_PARA_DATE_TIME	Date time string pointer	The date time string pointer. The string length is less than 32 bytes.

bool XTifFormat::SetPara(uint32_t para, float data)

Parameters:

para Parameter type as defined in the following table.

data Value to set.

Return:

Returns 1 if successful, 0 no such parameter.

Remark:

The user calls this function to set the specific parameter of the object.

Para	Data	Description
XTIF_PARA_TEMPER	Temperature	The temperature of X-GCU.
XTIF_PARA_HUMIDITY	Humidity	The humidity of X-GCU.

4.3. Error list

Error ID	Macro	Description
1	XERROR_SYS_SOCK_OPEN_FAIL	XSystem object's socket fails to open.
2	XERROR_SYS_SOCK_BIND_FAIL	XSystem object's socket fails to bind IP and port.
3	XERROR_SYS_SOCK_SEND_FAIL	XSystem object's socket fails to send command.
4	XERROR_SYS_SOCK_RECV_TIMEOUT	XSystem object's socket receives timeout.
5	XERROR_SYS_ENGINE_RECV_ERRCMD	XSystem object receives error ACK.
6	XERROR_SYS_ENGINE_RECV_ERRCODE	XSystem object receives error code.
7	XERROR_SYS_ENGINE_RECV_ERRCRC	XSystem object receives error CRC.
8	XERROR_SYS_ENGINE_NOT_OPEN	XSystem object is not opening.
9	XERROR_SYS_ALLOCATE_FAIL	XSystem object fails to allocate resource.
10	XERROR_ASCPAS_FORMAT_ERR	XSystem or Xcommand object gets error format ASCII command.
11	XERROR_ASCPAS_NONE_CMD	XSystem or Xcommand object gets non-defined ASCII command.
12	XERROR_CMD_SOCK_OPEN_FAIL	Xcommand object's socket

Error ID	Macro	Description
		fails to open.
13	XERROR_CMD_SOCK_BIND_FAIL	Xcommand object's socket fails to bind IP and port
14	XERROR_CMD_SOCK_SEND_FAIL	Xcommand object's socket fails to send command.
15	XERROR_CMD_SOCK_RECV_TIMEOUT	Xcommand object's socket receives timeout.
16	XERROR_CMD_ENGINE_RECV_ERRCMD	Xcommand object receives error ACK.
17	XERROR_CMD_ENGINE_RECV_ERRCODE	Xcommand object receives error code.
18	XERROR_CMD_ENGINE_RECV_ERRCRC	Xcommand object receives error CRC.
19	XERROR_CMD_ENGINE_NOT_OPEN	Xcommand object is not opening.
20	XERROR_CMD_ALLOCATE_FAIL	Xcommand object fails to allocate resource.
21	XERROR_IMG_SOCK_OPEN_FAIL	XAcquisition object's socket fails to open.
22	XERROR_IMG_SOCK_BIND_FAIL	XAcquisition object's socket fails to bind IP and port.
23	XERROR_IMG_SOCK_RECV_TIMEOUT	XAcquisition object's socket receives timeout.
24	XERROR_IMG_ALLOCATE_FAIL	XAcquisition object fails to allocate resource.
25	XERROR_IMG_ENGINE_NOT_OPEN	XAcquisition grabbing engine is not opening.
26	XERROR_IMG_ENGINE_START_FAIL	XAcquisition grabbing engine fails to start grabbing.
27	XERROR_IMG_ENGINE_STOP_ABNORMAL	XAcquisition grabbing engine stops abnormally.
28	XERROR_IMG_PARSE_OPEN_FAIL	XAcquisition parsing engine fails to open.
29	XERROR_IMG_PARSE_NOT_OPEN	XAcquisition parsing engine is not open.

Error ID	Macro	Description
30	XERROR_IMG_PARSE_START_FAIL	XAcquisition parsing engine fails to start.
31	XERROR_IMG_PARSE_STOP_ABNORMAL	XAcquisition parsing engine stops abnormally.
32	XERROR_IMG_TRANSFER_NOT_OPEN	XframeTransfer object is not open.
33	XERROR_IMG_TRANSFER_START_FAIL	XframeTransfer object fails to start.
34	XERROR_IMG_TRANSFER_STOP_ABNORMAL	XframeTransfer object stops abnormally.
35	XERROR_IMG_PACKET_POOL_OPEN_FAIL	Packet pool fails to open.
36	XERROR_MULTI_TRANS_ALLOCATE_FAIL	XmultiTransfer object fails to allocate resource.
37	XERROR_MULTI_TRANS_NOT_OPEN	XmultiTransfer object is not opening.
38	XERROR_DISP_ALLOCATE_FAIL	Xdisplay object fails to allocate resource.
39	XERROR_CMD_HEARTBEAT_FAIL	Xcommand fails to get heartbeat packet.
40	XERROR_CMD_HEARTBEAT_START_FAIL	Xcommand fails to start heartbeat thread.
41	XERROR_CMD_HEARTBEAT_STOP_ABNORMAL	Xcommand stops heartbeat thread abnormally.
42	XERROR_CMD_HEARTBEAT_VOL_ERR	X-GCU's voltage is out of range.
43	XERROR_IMG_ENGINE_GRAB_ABNORMAL	XAcquisition object grabbing engine works abnormally.
44	XERROR_FILE_OPERATE_ERROR	XTifFormat object fails to operate tif file.
45	XERROR_FILE_TIF_TAG_ENTRY_ABNORMAL	XTifFormat object fails to parse tag entry of tif file.
46	XERROR_FILE_TIF_COMPRESSED	XTifFormat object doesn't support compressed tif file.
47	XERROR_FILE_TIF_ALLOC_FAIL	XTifFormat object fails to allocate data buffer.

Error ID	Macro	Description
48	XERROR_MULTI_TRANS_START_FAIL	XMultiTransfer object fails to start thread.
49	XERROR_MULTI_TRANS_STOP_ABNORMAL	XMultiTransfer object fails to stop thread.

4.4. Event list

Event ID	Macro	Data	Description
50	XEVENT_IMG_PARSE_DATA_LOST	Lost line number	X-GCU line data lost.
51	XEVENT_IMG_TRANSFER_BUF_FULL	Buffer size	XFrameTransfer buffer is full.
52	XEVENT_IMG_PARSE_DM_DROP	DM packet number received	DM packet drops during LVDS transmission.
53	XEVENT_IMG_PARSE_PAC_LOST	Lost packet number	X-GCU packet drops during Ethernet transmission.
54	XEVENT_IMG_PARSE_CRC_ERR	Error DM ID	DM packet CRC error.
55	XEVENT_IMG_PARSE_VOL_ERR	Error DM ID	DM voltage error.
56	XEVENT_CMD_HEARTBEAT_TEMPRA	Temperature value	Reports X-GCU's temperature.
57	XEVENT_CMD_HEARTBEAT_HUMIDITY	Humidity value	Reports X-GCU's humidity.

5. .NET CLASS REFERENCES

5.1. Class table

Class	Description
XAcquisitionW	This class is in charge of getting image data from the X-GCU.
XAnalyzeW	Provides the basic analysis functions.
XCommandW	This class is in charge of command controlling of the X-GCU.
XDeviceW	Wraps the basic parameters of the X-GCU which is used as argument by the other classes.
XDisplayW	It is in charge of displaying frames.
XDualEngCorrectW	Reconstructs the dual energy image of the interlaced dual-energy raw data.
XDualLineCorrectW	Reconstructs the line sequence of the dual-line detector sub-system.
XFrameTransferW	Gets line data from the XAcquisition object and combines line data into frame.
XGigFactoryW	It is in charge of generating objects of network.
XImageW	Wraps frame data.
XMultiTransferW	When connecting with the multi-detector, it synchronizes each XFrameTranser object and generates the whole frame.
XOffCorrectW	Provides off-board gain and offset correcting functions for all operational modes except non-continuous hi/lo trigger mode.
XOffHLCorrectW	Provides off-board gain and offset correcting functions only for non-continuous hi/lo trigger mode.
XPiecewiseCorrectW	Provides off-board three points piecewise correction functions for all operation modes except non-continuous hi/lo trigger mode.
XPixelCorrectW	Provides functions to correct bad pixels.
XReferenceCorrect	Provide functions to correct signal variations caused by LINAC X-ray source.
XSystemW	This class communicates with the X-GCU by broadcast which reads and sets the Ethernet configurations of the X-GCU.
XTifFormatW	Provides manipulating ways of TIFF format image.

5.2. Class references

XAcquisitionW class

Description:

The XAcquisitionW class provides functions to manipulate image data acquisition. It gets data packets from the X-GCU and parses packets into line data and passes line data to the XFrameTransferW object.

Methods & properties:

void XAcquisitionW.Close() Method

Remark:

Releases all the resources.

bool XAcquisitionW.EnableLineInfo Property (write only)

Remark:

Enables or disables the line info mode; 1 is enable, 0 is disable. If it is enabled, there will be an info header added in the front of each line data. The content of the header is line id (2 bytes), line stamp(4 bytes), energy flag(1 byte) and DFE number flag(1 byte). It is disabled by default.

If you are using the class XMuliTransfer to synchronize the multi-X-GCU, the line info mode must be enabled.

unsigned long XAcquisitionW.GrabTimeDiff() property (read only)

Remark:

When start grabbing image data, it will do some extra initializing works first such as initializing thread and cleaning buffers. So it will take longer time for getting the first frame while calling grab() function. For snap() function, it will do the extra work for each frame. This function returns the time difference between the point of calling grab()/snap() and the point that actual data scanning happens. It could help to correct the object position in the first frame. The unit of returned time is ms for Windows system and us for Linux system.

bool XAcquisitionW.IsGrabbing Property (read only)

Remark:

Gets the working state; 1 grabbing, 0 stopping.

bool XAcquisitionW.IsOpen Property(read only)

Remark:

Gets the state; 1 opening, 0 closing.

unsigned long XAcquisitionW.LastError Property (read only)

Remark:

Gets the error ID if any exception occurs. Please refer to the section 5.3 for the explanation of the specific error.

bool XAcquisitionW.Grab(unsigned long frame_num) Method

Parameters:

frame_num Number of frame to grab. If it is 0, it will keep grabbing until the Stop() is called. Otherwise, it will stop automatically after getting the number of frames.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Starts grabbing the image data from the X-GCU. If it returns 0, the user can utilize the XAcquisition.LastError to trace the error.

bool XAcquisitionW.Open(XDeviceW dev, XCommandW cmd_handle) Method

Parameters:

dev XDeviceW object which provides the detector's info. See XDeviceW class.

cmd_handle XCommandW object. See XCommandW class.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Allocates the needed resources. If it returns 0, the user can utilize the XAcquisition.LastError to trace the error.

XFrameTransferW XAcquisitionW.Transfer Property (write only)

Remark:

XFrameTransferW object handle. Must be set before opening.

XGigFactoryW XAcquisitionW.Factory Property (write only)

Remark:

XGigFactoryW object handle. Must be set before opening.

unsigned long XAcquisitionW.Timeout Property (write only)

Remark:

If no image data comes during the timeout period while grabbing, it reports an error and stop scanning. The timeout unit is ms with the default value 20000. The range is from 0 to 2000000. If it is 0, the grabbing thread will

block waiting for data without reporting any error. It should be set before opening.

bool XAcquisitionW.Snap() Method

Remark:

Gets one frame from the X-GCU and will be blocked until the one frame is received. If it returns 0, the user can utilize the XAcquisitionW.LastError to trace the error.

bool XAcquisitionW.Stop() Method

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Stops grabbing the frame which should be used with the function Grab(0). If it returns 0, the user can utilize the XAcquisitionW.LastError to trace the error.

Delegations:

void XAcquisitionW.DelOnXError(int err_id, string err_msg)

Parameters:

err_id Error ID.

err_msg_ Error description.

Remark:

The error callback of the XAcquisitionW object. Please refer to the section 5.3 for the specific error ID.

void XAcquisitionW.DelOnXEvent(int event_id, int data)

Parameters:

event_id Event ID.

data Event data.

Remark:

The event callback of the XAcquisitionW object. Please refer to the section 5.4 for the specific event ID. This callback responds to the event IDs from 50 to 55.

XAnalyzeW class

Description:

Provides functions to calculate average and noise.

Methods & properties:

bool XAnalyzeW.DoAnalyze(XImageW image, unsigned long type) Method

Parameters:

Image XImageW object.

type Defines what analysis to apply.

0: Calculates row average, min and max values;

1: Calculates row noise;

2: Calculates column average, min and max values;

3: Calculates column noise.

Return:

Returns 0 if the parameter arrays fail to allocate.

Remark:

If the user wants to get noise, the averaging analysis must be done first. The user can get the public parameter arrays directly after doing the analysis.

IntPtr _col_avg_ Property (read only)

Remark:

The column average array which can be received after the averaging analysis. The array size is the image width, the type is unsigned long*.

IntPtr _col_max_ Property (read only)

Remark:

The column max array which can be received after the max analysis. The array size is the image width, the type is unsigned long*.

IntPtr _col_min_ Property (read only)

Remark:

The column min array which can be received after the min analysis. The array size is the image width, the type is unsigned long*.

IntPtr _col_noise_ Property (read only)

Remark:

The column noise array which can be received after the noise analysis. The array size is the image width, the type is float*.

IntPtr _row_avg_ Property (read only)

Remark:

The row average array which can be received after the averaging analysis. The array size is the image height, the type is unsigned long*.

IntPtr _row_max_ Property (read only)

Remark:

The row max array which can be received after the max analysis. The array size is the image height, the type is unsigned long*.

IntPtr _row_min_ Property (read only)

Remark:

The row min array which can be received after the min analysis. The array size is the image height, the type is unsigned long*.

IntPtr _row_noise_ Property (read only)

Remark:

The row noise array which can be received after the noise analysis. The array size is the image height, the type is float*.

XCommandW class

Description:

The XCommandW class manages the basic communication with the X-GCU. It adopts question and answer mode the X-GCU gives the ACK for each command. The user can send an ASCII command directly to the X-GCU or utilize the wrapping functions instead.

Methods & properties:

void XCommandW.Close() Method

Remark:

Releases all resources.

long XCommandW.ExecutePara(unsigned long para, unsigned long long data) Method

Parameters:

para Execution type as defined in the following table.

Data Some of the executions need this argument. See the following table.

Para	Data	Description
1	None	Initializes the X-GCU with settings saved in the user's flash.

56	None	Initializes the X-GCU with default settings.
2	None	Saves all current settings into the user's flash.
48	Range 0 – 3	Loads channel gain from specific section of flash to FPGA. There are 4 groups of parameters.
50	None	Loads channel offset from flash to FPGA.
18	None	Rests channel gain to 1.0.
19	None	Rests channel offset to 0.
32	None	Triggers one frame in external frame trigger mode. Generates a soft frame trigger.

Return:

Returns 1 if successful, 0 no such execution, -1 error.

Remark:

The user calls this function to force the X-GCU to carry out the specific execution. It sends the specific command to the X-GCU. The X-GCU does the corresponding execution, then returns the ACK. If an error occurs, the user can call the XCommandW.LastError to trace the error.

bool XCommandW.IsOpen Property (read only)

Remark:

Gets the state, 1 opening, 0 closing.

long XCommandW.LastError Property (read only)

Remark:

Gets the error ID if any exception occurs. Please refer to the error list for the explanation of the specific error.

long XCommandW.GetPara(unsigned long para, unsigned long long &data, unsigned long dm_id) Method

Parameters:

para Parameter type defined as following table.

data Returned value of specific parameter.

dm_id DM module index which starts from 1. Some of the commands need this argument. If it is 0xFF, the X-GCU will apply the command to all DMs. Otherwise, the X-GCU only operates the specified DM module. The DM ID 1 represents the first DM module in the first X-GCU channel.

Return:

Returns 1 if successful, 0 no such execution, -1 error.

Remark:

The user calls this function to get the specific parameter from the X-GCU. If any error occurs, the user can call the XCommandW.LastError to trace the error.

Para	Data	DM_ID	Description
3	Returned value.	None	Integration time, 4-byte unsigned value. The unit is us. The range is defined by max and min integration time. Default is 3000.
4	Returned value.	None	Non-continuous integration time, 2-byte unsigned value. Unit is us. Default is 580.
5	Returned value.	None	Operation mode, 1-byte unsigned value. 0: Continuous mode; 1: Non-continuous mode 2: Constant integration time mode; 3: Non-continuous with Hi/Lo trigger. Default is 0.
6	Returned value	DM index	DM module gain value, 2-byte unsigned value. High 8 bits are high energy gain; low 8 bits are low energy gain. Default is 0x0606. Valid range: LCS2: 1~63; X-Card 0.2/0.4/0.8: 1~2; X-Card 1.5/1.6/2.5: 1~12; X-Card2 0.2/0.4: 1~2; X-Card2 0.8: 1~8; Aurora GT/B: 1~127. The DM index can't be 0xFF. It is not available while scanning.
7	Returned value	None	Defines which DM module gain to launch in single energy mode. 1-byte unsigned value. 0: Low-energy gain; 1: High-energy gain. Default is 0, It is only available while

Para	Data	DM_ID	Description
			XPARA_OPE_MODE is 3.
8	Returned value	None	DM module number in each X-GCU's channel. It is a five-byte value. The first channel number is the highest byte, and the fifth channel number is the lowest byte.
10	Returned value	None	Scanning state, 1-byte unsigned value. 0: Stopping; 1: Scanning.
11	Returned value	None	Gain correcting state, 1-byte unsigned value. Enables or disables channel gain correction. 0: Disable; 1: Enable. Default is 0.
12	Returned value	None	Offset correcting state, 1-byte unsigned value. Enables or disables channel offset correction. 0: Disable; 1: Enable. Default is 0.
13	Returned value	None	Base line correcting state, 1-byte unsigned value. Enables or disables baseline correction. 0: Disable; 1: Enable. Default is 0.
16	Returned value	None	Base line value, 2-byte unsigned value. Range 0 – 1000. Default is 0.
20	Returned value	None	Pixel binning mode, 1-byte unsigned value. 0: Original pixel; 1: Averaging every 2 pixels; 2: Summing every 2 pixels; 3: Averaging every 4 pixels; 4: Summing every 4 pixels

Para	Data	DM_ID	Description
21	Returned value	None	Line averaging filter mode, 1-byte unsigned value. 0: None; 1: 2 lines; 2: 4 lines; 3: 8 lines; 4: 16 lines; 5: 32 lines; 6: 64 lines; 7: 128 lines; 8: 256 lines. Default is 0.
22	Returned value	None	Line summing filter mode, 1-byte unsigned value. 0: None; 1: 2 lines; 2: 4 lines; 3: 8 lines; 4: 16 lines; 5: 32 lines; 6: 64 lines; 7: 128 lines; 8: 256 lines. Default is 0.
23	Returned value	None	Output scale, 1-byte unsigned value. The output pixel depth will be original bit subtracting the scale value. 0: Original; n: Original – n. Default is 0. E.g., Original bit is 16, scale value is 8, the output pixel bits will be 8.

Para	Data	DM_ID	Description
24	Returned value	None	Offset averaging filter, 1-byte unsigned value. 0: None; 1: 2 lines; 2: 4 lines; 3: 8 lines. Default is 0. It is only available while XPARA_OPE_MODE is 3.
25	Returned value	None	External line trigger mode, 1-byte unsigned value. 0: Rising edge; 1: Falling edge; 2: Sync trigger stamp mode; 3: Async trigger stamp mode. Default is 0.
26	Returned value	None	Enables external line trigger function, 1-byte unsigned value. 0: Disable; 1: Enable. Default is 0.
27	Returned value	None	External line trigger fine delay, 1-byte unsigned value. Unit is 0.125us. Default is 0.
28	Returned value	None	External line trigger raw delay, 1-byte unsigned value. Unit is 64us. Default is 0.
29	Returned value	None	External frame trigger mode, 1-byte unsigned value. 0: Rising edge effect; 1: Falling edge effect. Default is 0.
30	Returned value	None	Enables external frame trigger function, 2-byte unsigned value.

Para	Data	DM_ID	Description
			0: Disable; n: Enable, and output 32*n lines by each trigger. Default is 0.
31	Returned value	None	External frame trigger raw delay, 1-byte unsigned value. The unit is 32 lines. Default is 0.
33	Returned value	None	Hi/Lo line trigger with hi/lo edge level detection, 1-byte unsigned value. 0: Low level; 1: High level. Default is 0.
34	Returned value	None	Hi/Lo line trigger sampling position, 1-byte unsigned value. The unit is 1us. Range is 0-255, default is 0.
35	Returned value	None	Total pixel number of all DMs, 2-byte unsigned value.
36	Returned value	None	Pixel size, 1-byte unsigned value. The returned value is real pixel size * 10. The unit of the real pixel size is mm.
37	Returned value	None	Pixel depth, 1-byte unsigned value. 16: 16 bits; 18: 18 bits; 20: 20 bits. Default is 16.
38	Returned value	None	Max and min integration time, 8-byte unsigned value. The high four bytes are the max value, the low four bytes are the min value.
39	Returned value	None	X-GCU's firmware version, 2-byte unsigned value.
40	Returned value	DM index	DM's firmware version, 2-byte unsigned value. The DM index cannot be 0xFF. It is not available while scanning.

Para	Data	DM_ID	Description
41	Returned value	None	X-GCU's test pattern mode, 1-byte unsigned value. 0: Disable; 1: Normal test pattern; 2: Slope test pattern; Default is 0.
42	Returned value	DM index	DM's test pattern mode, 1-byte unsigned value. 0: Normal image data mode; 1: Enable DM board fixed value (AAAAA[Hex]) test mode.; 2: Enable I'm here function, Light the LED of the specific DM. Make sure the LED must enabled. Default is 0. The DM index cannot be 0xFF. It is not available while scanning.
43	Returned value	None	DM's pixel number per energy mode, 1-byte unsigned value. 5: 32; 6: 64; 7: 128; 8: 256; 9: 512.
44	Returned value	None	Card number per DFE, 1-byte unsigned value. Range is 1~2, default is 1.
45	Returned value	None	Card type, 1-byte unsigned value. 0: LCS2 DM; 1: X-Card 0.2/0.4/0.8; 2: X-Card 1.5/1.6/2.5; 3: X-Card2 0.2/0.4/0.8; 4: Dual-row LCS2 DM; 5: Aurora GT;

Para	Data	DM_ID	Description
			6: X-ICT; 7: Aurora B.
57	Returned value	None	LED state, 1-byte unsigned value. 0: Disable; 1: Enable. Default is 1.
58	Returned value	None	Terminal resistor state, 1-byte unsigned value. 0: Disable; 1: Enable. Default is 0.
61	Returned value	None	X-GCU type. 0: Ethernet; 1: Camera Link. Default is 0.
62	Returned value	None	Energy mode. 0: Single energy; 1: Dual energy stack; 2: Dual energy parallel.
66	Returned value	None	Trigger stamp parity check mode. 0: Disable parity check; 1: Use "odd" parity check; 2: Use "even" parity check
67	Returned value	None	System working time counter, which is increased by one step per 2 hours.
68	Returned value	None	X-GCU's working time after each powering up.

string XCommandW.GetPara(unsigned long para, unsigned long dm_id)
Method

Parameters:

para Parameter type defined as in the following table.

dm_id DM module index which starts from 1. Some of the commands need this argument. If it is 0xFF, the X-GCU will apply the command to all DMs. Otherwise, the X-GCU only operates the specified DM module. The DM ID 1 represents the first DM module in the first X-GCU channel.

Return:

Returns specified parameter.

Remark:

The user calls this function to get the specific parameter from the X-GCU. If any error occurs, the user can call the XCommandW.LastError to trace the error.

Para	Return	DM_ID	Description
51	Returned value	None	X-GCU's serial number, 32-byte ASCII code.
52	Returned value	DM index	DM's serial number, 32-byte ASCII code. The DM index cannot be 0xFF. It is not available while scanning. It is not available for Aurora B.
64	Returned value	None	Product info, 128-byte ASCII code.

long XCommandW.GetPara(unsigned long para, float &temperature, float &humidity, float &v1, float &v2, float &v3, float &v4, float &v5, float &v6, float &v7, float &high_temp, unsigned long dm_id) Method

Parameters:

para Parameter type defined as in the following table.

temperature Returned temperature value of X-GCU or DM.

humidity Returned humidity value of X-GCU or DM.

v1~v7 Returned voltage values of X-GCU or DM.

high_temp High resolution temperature.

dm_id DM module index which starts from 1. Some of the commands need this argument. If it is 0xFF, the X-GCU will apply the command to all DMs. Otherwise, the X-GCU only operates the specified DM module. The DM ID 1 represents the first DM module in the first X-GCU channel.

Return:

Returns 1 if successful, 0 no such parameter, -1 error.

Remark:

The user calls this function to get the specific parameter from the X-GCU. If any error occurs, the user can call the GetLastError() to trace the error.

Para	Returns	DM_ID	Description
59	Temperature; Humidity; v1~v4.	None	Gets X-GCU's temperature (°C), relative humidity (%) and 4 voltage values. The correct voltage values should be as follows: v1: 24V±10% (power 24V) 12V±10% (power 12V); v2: 3.3V±5%; v3: 2.5V±5%; v4: 1.1V±5%. The relative humidity is not available with the X-GCU_STD.
60	Temperature; Humidity; v1~v7(v1~v4 for Aurora GT and no voltage is available for Aurora B); High resolution temperature.	DM index	Gets DM's temperature (°C), relative humidity (%) and 7 voltage values. The correct voltage values should be as follows: v1: 2.5V±5%; v2: 24V±10% (power 24V) 12V±10% (power 12V); v3: 5.2V±5%; v4: 1.2V±5%; v5: 5V±5%; v6: 3.3V±5%; v7: 4.096V±5%. For Aurora GT: v1: 24V±10% (power 24V) 12V±10% (power 12V); v2: 2.5V±5%; v3: 5V±5%; v4: 3.3V±5%.

			For Aurora B: No voltage is available. The high resolution temperature is only available with the DM which has high resolution sensor installed (high tier Digital X-Cards and LCS2 DM), otherwise the return is always 511.9. The DM index cannot be 0xFF. It is not available while scanning. Aurora B does not support this function.

XHealthParaW XCommandW.GetHeath(unsigned para, unsigned long dm_id)

Method

Parameter:

para Parameter type defined as in the following table.

dm_id DM module index which starts from 1. Some of the commands need this argument. If it is 0xFF, the X-GCU will apply the command to all DMs. Otherwise, the X-GCU only operates the specified DM module. The DM ID 1 represents the first DM module in the first X-GCU channel.

Return:

XHealthParaW is a struct that wraps temperature, humidity and voltage values.

Remark:

The user calls this function to get the specific parameter from the X-GCU. If any error occurs, the user can call the GetLastError() to trace the error.

Para	Returns	DM_ID	Description
59	X-GCU's heath parameters.	None	Get X-GCU's temperature (°C), relative humidity (%) and 4 voltage values. The correct voltage values should be as follows: v1: 24V±10% (power 24V) 12V±10% (power 12V); v2: 3.3V±5%; v3: 2.5V±5%; v4: 1.1V±5%.

			The relative humidity is not available with X-GCU_STD.
60	DM's heath parameters.	DM index	Get DM's temperature (°C), relative humidity (%) and 7 voltage values. The correct voltage values should be as follows: v1: 2.5V±5%; v2: 24V±10% (power 24V) 12V±10% (power 12V); v3: 5.2V±5%; v4: 1.2V±5%; v5: 5V±5%; v6: 3.3V±5%; v7: 4.096V±5%. The DM index cannot be 0xFF. It is not available while scanning.

bool XCommandW.Open(XDeviceW dev) Method

Parameters:

dev XDeviceW object which provides detector's info.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Allocates the resources needed. If it returns 0, the user can utilize the XCommandW.LastError to trace the error.

string XCommandW.SendAscCmd(string command) Method

Parameters:

command Command string. Format: "[CODE,OPERATION,DM_ID,DATA]". Example: "[ST,W,0,3E8]", set integration time to be 100; "[ST,R,0]", get integration time.

Return:

Returns the ACK string. Format: "[FLAG, DATA]". Example: "[4]", error; "[0]", successful with no data; "[0, DATA]" successful with data.

Remark:

Accepts an ASCII format command and translates it to a hex format, then sends the hex command to the X-GCU. Please refer to the chapter 6 for the ASCII command list.

Wrapped by the SetPara(), GetPara() and ExcutePare() functions. The user can utilize the three wrappers or call this function directly to send the command to the X-GCU.

XGigFactory XCommandW.Factory Property (write only)

Remark:

XGigFactoryW object handle, should be set before opening.

long XCommandW.SetPara(unsigned long para, unsigned long long data, unsigned long dm_id) Method

Parameters:

para Parameter type defined in the following table.

data Value to set

dm_id DM module index which starts from 1. Some of the commands need this argument. If it is 255, the X-GCU will apply the command to all DMs. Otherwise, the X-GCU only operates the specified DM module. The DM ID 1 represents the first DM module in the first X-GCU .

Return:

Returns 1 if successful, -1 error.

Remark:

The user calls this function to set the specific parameter of the X-GCU. If any error occurs, the user can call the XCommandW.LastError to trace the error.

Para	Data	DM_ID	Description
3	Integration time value	None	Integration time, 4-byte unsigned value. The unit is us. The range is defined by max and min integration time. Default is 3000.
4	Integration time in Non-continuous mode and in Constant mode	None	Integration time in Non-continuous mode and in Constant mode, 2-byte unsigned value. The unit is us. Default is 580.

Para	Data	DM_ID	Description
5	Operation mode value	None	Operation mode, 1-byte unsigned value. 0: Continuous mode; 1: Non-continuous mode 2: Constant integration time mode; 3: Non-continuous with Hi/Lo trigger. Default is 0.
6	Gain value	DM index. 0xFF: All DMs; others above 0: specific DM.	DM module gain value, 2-byte unsigned value. The high 8 bits are high energy gain, the low 8 bits are low energy gain. Default is 0x0606. Valid range: LCS2: 1~63; X-Card 0.2/0.4/0.8: 1~2; X-Card 1.5/1.6/2.5: 1~12; X-Card2 0.2/0.4: 1~2; X-Card2 0.8: 1~8; Aurora GT/B: 1~127. It is not available while scanning.
7	HI/LO mode value	None	Defines which DM-module gain to launch in single energy mode. 1-byte unsigned value. 0: Low-energy gain; 1: High-energy gain. Default is 0, It is only available while XPARA_OPE_MODE is 3.
8	Channel card number	None	DM module number in each X-GCU's channel. It is a five-byte value. The first channel number is the highest byte and the fifth channel number is the lowest byte.
10	Scanning state value	None	Scanning state, 1-byte unsigned value. 0: Stopping;

Para	Data	DM_ID	Description
			1: Scanning.
11	Gain correcting state value	None	Gain correcting state, 1-byte unsigned value. Enables or disables channel gain correction. 0: Disable; 1: Enable. Default is 0.
12	Offset correcting state value	None	Offset correcting state, 1-byte unsigned value. Enables or disables channel offset correction. 0: Disable; 1: Enable. Default is 0.
13	Base line correcting state value	None	Base line correcting state, 1-byte unsigned value. 0: Disable; 1: Enable. Default is 0.
16	Baseline value	None	Base line value, 2-byte unsigned value. Range 0 – 1000. Default is 0.
20	Binning mode value	None	Pixel binning mode, 1-byte unsigned value. 0: Original pixel; 1: Averaging every 2 pixels; 2: Summing every 2 pixels; 3: Averaging every 4 pixels; 4: Summing every 4 pixels. Default is 0.
21	Line averaging filter mode value	None	Line averaging filter mode, 1-byte unsigned value. 0: None;

Para	Data	DM_ID	Description
			1: 2 lines; 2: 4 lines; 3: 8 lines; 4: 16 lines; 5: 32 lines; 6: 64 lines; 7: 128 lines; 8: 256 lines. Default is 0.
22	Line summing filter mode value	None	Line summing filter mode, 1-byte unsigned value. 0: None; 1: 2 lines; 2: 4 lines; 3: 8 lines; 4: 16 lines; 5: 32 lines; 6: 64 lines; 7: 128 lines; 8: 256 lines. Default is 0.
23	Output scale value	None	Output scale, 1-byte unsigned value. The output pixel depth will be the original bit subtracting the scale value. 0: Original; n: Original – n. Default is 0. E.g., the original bit is 16, the scale value is 8, the output pixel bits will be 8.
24	Offset averaging filter value	None	Offset averaging filter, 1-byte unsigned value.

Para	Data	DM_ID	Description
			0: None; 1: 2 lines; 2: 4 lines; 3: 8 lines. Default is 0. It is only available while XPARA_OPE_MODE is 3.
25	External line trigger mode value	None	External line trigger mode, 1-byte unsigned value. 0: Rising edge; 1: Falling edge; 2: Sync trigger stamp mode; 3: Async trigger stamp mode. Default is 0.
26	External line trigger state value	None	Enabling external line trigger function, 1-byte unsigned value. 0: Disable; 1: Enable. Default is 0.
27	External line trigger fine delay value	None	External line trigger fine delay, 1-byte unsigned value. The unit is 0.125us. Default is 0.
28	External line trigger raw delay value	None	External line trigger raw delay, 1-byte unsigned value. The unit is 64us. Default is 0.
29	External frame trigger mode value	None	External frame trigger mode, 1-byte unsigned value. 0: Rising edge; 1: Falling edge. Default is 0.
30	External frame trigger state value	None	Enabling external frame trigger function, 2-byte unsigned value.

Para	Data	DM_ID	Description
			0: Disable; n: Enable, and output 32*n lines on each trigger. Default is 0.
31	External frame trigger delay value	None	External frame trigger raw delay, 1-byte unsigned value. The unit is 32 lines. Default is 0.
33	Hi/Lo trigger edge mode value	None	Hi/Lo line trigger with hi/lo edge level detection, 1-byte unsigned value. 0: Low level; 1: High level. Default is 0.
34	Hi/Lo trigger sampling position value	None	Hi/Lo line trigger sampling position, 1-byte unsigned value. The unit is 1us. Range is 0-255, default is 0.
38	Max/min integration time value	None	Max and min integration time, 8-byte unsigned value. The high four bytes are the max value, the low four bytes are the min value.
41	GCU's test pattern mode value	None	GCU's test pattern mode, 1-byte unsigned value. 0: Disable; 1: Normal test pattern; 2: Slope test pattern; Default is 0.
42	DM's test pattern mode value	DM index. 0xFF: All DMs; others above 0: specific	DM's test pattern mode, 1-byte unsigned value. 0: Normal image data mode; 1: Enable DM board fixed value (AAAAA[Hex]) test mode.; 2: Enable I'm here function, Light the LED of the specific DM. Make

Para	Data	DM_ID	Description
		DM.	sure the LED must enabled. Default is 0. It is not available while scanning.
57	LED state value	None	LED state, 1-byte unsigned value. 0: Disable; 1: Enable. Default is 1.
58	Terminal resistor state value	None	Terminal resistor state, 1-byte unsigned value. 0: Disable; 1: Enable. Default is 0.
66	Trigger stamp parity mode value	None	Trigger stamp parity check mode, 1-byte unsigned value. 0: Disable parity check; 1: Use "odd" parity check; 2: Use "even" parity check

long XCommandW.SetPara(unsigned long para, string data, unsigned long dm_id)

Method

Parameters:

para Parameter type defined in the following table.

data Value to set

dm_id DM module index which starts from 1. Some of the commands need this argument. If it is 255, the X-GCU will apply the command to all DMs. Otherwise, the X-GCU only operates the specified DM module. The DM ID 1 represents the first DM module in the first X-GCU channel.

Return:

Returns 1 if successful, -1 error.

Remark:

The user calls this function to set the specific parameter of the X-GCU. If any error occurs, the user can call the XCommandW.LastError to trace the error.

Para	Data	DM_ID	Description
51	Serial number	None	X-GCU's serial number, 32-byte ASCII code. It is not available for the user.
52	Serial number	DM index	DM's serial number, 32-byte ASCII code. The DM index cannot be 0xFF. It is not available for the user.
64	Product info	None	Product info, 128-byte ASCII code. It is not available for the user.

unsigned long XCommandW.Timeout Property (write only)

Remark:

If the ACK does not arrive during the timeout period after sending a command, it will report an error. The timeout unit is ms with the default value 20000. The range is from 12000 to 2000000.

bool XCommandW.StartHeartbeat() Method

Return:

Returns 1 if successful, 0 otherwise.

Remark:

If it starts correctly, the X-GCU will keep sending the heartbeat packet by a period of 1s. If it cannot get the heartbeat after 10 times in a row it will report the error XERROR_CMD_HEARTBEAT_FAIL. The user should check the NET cable or the X-GCU in that case.

bool XCommandW.StopHeartbeat() Method

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Stops the X-GCU sending the heartbeat packet. If it forces the thread exit, it will report the error XERROR_CMD_HEARTBEAT_STOP_ABNORMAL. In that case, it cannot start again.

Delegations:

void XCommandW.DelOnXError(int err_id, string err_msg)

Parameters:

err_id Error ID.

err_msg_ Error description.

Remark:

The error callback of the XCommandW object. Please refer to the section 5.3 for the specific error ID.

void XCommandW.DelOnXEvent(int event_id, float data)

Parameters:

event_id Event ID.

data Event data.

Remark:

The event callback of the XCommandW object. Please refer to the section 5.4 for the specific event ID. This callback responds to the event IDs from 56 to 57.

XDeviceW class

Description:

The XDeviceW class wraps the basic parameters of the detector. It is used as an argument in the initializing process of the XCommandW and XAcquisitionW objects.

Methods & properties:

unsigned long XDeviceW.CardNumber Property (read only)

Remark:

Gets total DM number.

unsigned long XDeviceW.CardType Property (read only)

Remark:

Gets the X-Card type number.

0: LCS2 DM;

1: X-Card 0.2/0.4/0.8;

2: X-Card 1.5/1.6/2.5;

3: X-Card2 0.2/0.4/0.8;

4: Dual-row LCS2 DM;

5: Aurora GT;

6: X-ICT;

7: Aurora B.

unsigned short XDeviceW.CmdPort Property (read /write)

Remark:

Sets or gets the command channel port number of the X-GCU.

unsigned long XDeviceW.DMPixelNumber Property (read only)

Remark:

Gets the pixel number of the DM for one energy level.

5: 32;

6: 64;

7: 128;

8: 256;

9: 512.

unsigned short XDeviceW.ImgPort Property (read/write)

Remark:

Sets or gets the image channel port number of the X-GCU.

string XDeviceW.IP Property (read/write)

Remark:

Sets or gets the IP address string of the X-GCU.

IntPtr XDeviceW.MAC Property (read/write)

Remark:

Sets or gets the 6-byte MAC address array of the X-GCU. The type is char*.

unsigned long XDeviceW.OPMode Property (read only)

Remark:

Gets operational mode.

0: Continuous mode;

1: Non-continuous mode;

2: Constant integration time mode;

3: Non-continuous with Hi/Lo trigger.

unsigned long XDeviceW.PixelDepth Property(read only)

Remark:

Gets the pixel depth of the X-GCU.

unsigned long XDeviceW.PixelNumber Property(read only)

Remark:

Gets the total pixel number of the X-GCU.

unsigned long long XDeviceW.SerialNum Property(read only)

Remark:

Gets the serial number of the X-GCU.

XDisplayW class

Description:

The XDisplayW class is in charge of displaying the raw image data.

Methods & properties:

void XDisplayW.Close() Method

Remark:

Releases all resources.

void XDisplayW.Display(XImageW image) Method

Parameters:

Image XImageW object. See class XImageW.

Remark:

Displays raw image data.

float XDisplayW.Gama Property (read\write)

Remark:

Sets or gets the gama value.

bool XDisplayW.IsDDrawEnable Property (read only)

Remark:

Returns 1 if the direct draw is enabled, 0 otherwise. It is used to check whether the direct draw is enabled after opening. If the direct draw is not enabled, it will use the GDI to display the image data.

unsigned long XDisplayW.LastError Property (read only)

Remark:

Returns the error ID if any exception occurs. Please refer to the error list for the explanation of the specific error.

bool XDisplayW.IsOpen Property (read only)

Remark:

Gets the state, 1 opening, 0 closing.

bool XDisplayW.Open(unsigned long width, unsigned long height, unsigned long pixel_depth, IntPtr hwnd, unsigned long color_mode) Method

bool XDisplayW.Open(XDeviceW dev, uint32_t height, IntPtr hwnd, unsigned long color_mode) Method

Parameters:

width	Image width.
height	Image height.
pixel_depth	Pixel depth.
hwnd	Window handler.
color_mode	Pseudo color. 0: Gray; 1: Sin pseudo color; 2: Cos pseudo color; 3: Hot pseudo color; 4: Jet pseudo color.
dev	XDeviceW object pointer.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Allocates the needed resources. If it returns 0, the user can utilize the XDispalyW.LastError to trace the error.

XDualEngCorrectW class

Description:

The dual energy line data of the interlaced dual-energy raw data sub-system comes according to the sequence of the external line trigger. For most applications, the Hi/Lo trigger comes alternatively, so the Hi/Lo line data comes in pairs. In that case, the XDualEngCorrectW object combines two lines into one line where Lo data is on the left, Hi data is on the right.

Before using this correction, the line info mode of the XAcquisitionW object must be enabled.

Methods & properties:

bool XDualEngCorrectW.DoCorrect(XImageW image) Method

Parameters:

image	XImageW object.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

If successful, the user should call the XDualEngCorrectW.GetImage() to get the combined image. If the allocation of combined image fails, it returns 0.

XImageW XDualEngCorrectW.GetImage() Method

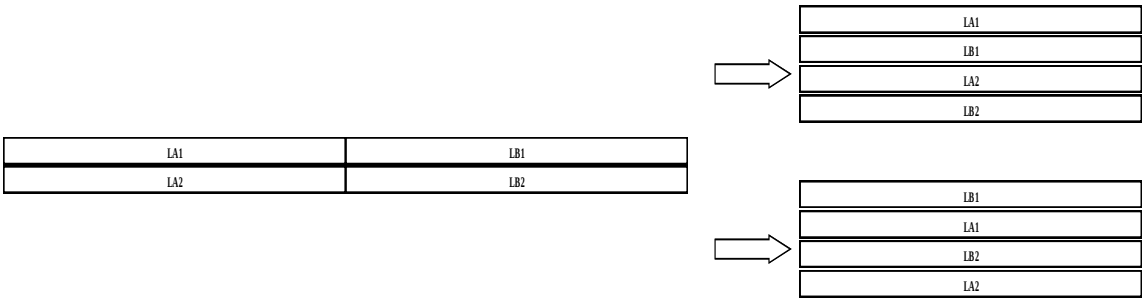
Return:

Returns the combined image.

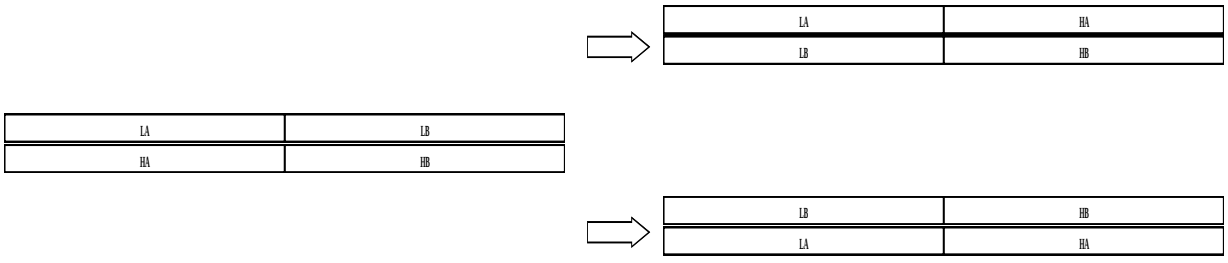
XDualLineCorrectW class

Description:

For the dual-line detector sub-system, two lines come together. The XDualLineCorrectW object can rearrange the sequence and the position of the two line data. In the single energy mode, the two lines can be arranged into two cases according to the direction flag as in the following figure.



In the dual energy mode, if the Hi/Lo line data comes in pairs, the two energy data of the two lines is arranged as follows.



Before using this correction, the line info mode of the XAcquisitionW object must be enabled.

Methods & properties:

bool XDualLineCorrectW.DoCorrect(XImageW image) Method

Parameters:

image XImageW object.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

If successful, the user should call the GetImage() to get the combined image. If the allocation of combined image fails, it returns 0.

XImageW XDualLineCorrectW.GetImage() Method

Return:

Returns the combined image.

void XDualLineCorrectW.SetDirect(bool dir) Method

Parameters:

dir Direction flag.

Remark:

Sets the sequence of the arrangement of the two lines.

void XDualLineCorrectW.SetDualEnergy(bool eng) Method

Parameters:

eng Energy flag.

Remark:

Sets the energy mode of the coming data to deal with.

XFrameTransferW class

Description:

The XFrameTransferW class receives the raw line data from the XAcquisitionW class and combines lines into the frame. The user can get the frame data from its callback function.

Methods & properties:

bool XFrameTransferW.IsRunning Property (read only)

Remark:

Returns the working state, 1 grabbing, 0 stopping.

unsigned long XFrameTransferW.LastError Property (read only)

Remark:

Returns the error ID if any exception occurs. Please refer to the error list for the explanation of the specific error.

unsigned long XFrameTransferW.LineNum Property (read\write)

Remark:

Sets or gets the line number of a frame. It should be set before opening the XAcquisitionW object.

Delegations:

void XFrameTransferW.DelOnXError(int err_id, string err_msg)

Parameters:

err_id Error ID.

err_msg_ Error description.

Remark:

The error callback of the XFrameTransferW object. Please refer to the section 5.3 for the specific error ID.

void XFrameTransferW.DelOnXEvent(int event_id, int data)

Parameters:

event_id Event ID.

data Event data.

Remark:

The event callback of the XFrameTransferW object. Please refer to the section 5.4 for the specific event ID. This callback responds to the event IDs from 50 to 55.

void XFrameTransferW.DelOnFrameReady(XImageW image)

Parameters:

image XImageW object.

Remark:

When the frame is completed, the user can get the frame data in this callback function. Please note, if this function takes too much time to return, the frame buffer could overflow.

XGigFactoryW

Description:

The XGigFactoryW class is responsible for allocating all resources. The user has to define the factory object and pass it to the other objects needing it.

XImageW class

Description:

The XImageW class wraps the frame data and the information of a frame.

Methods & properties:

unsigned long XImageW.GetPixelVal(unsigned long row, unsigned long col) Method

Parameters:

row Vertical coordinate.

col Horizontal coordinate.

Return:

Returns the pixel value.

bool XImageW.Save(string file_name) Method

Parameters:

file_name ".txt" file.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Saves image data as ".txt" file, only supports ".txt" file.

void XImageW.SetPixelVal(unsigned long row, unsigned long col, unsigned long pixel_value) Method

Parameters:

row Vertical coordinate.

col Horizontal coordinate.

pixel_value Pixel value to set.

Remark:

Sets pixel value.

IntPtr XImageW.DataAddr Property (read only)

Remark:

Frame data pointer.

unsigned long XImageW.DataOffset Property (read only)

Remark:

The position of the first pixel value default is 0. If using the line info mode, the line information is attached on the front of each line. Please see the XAcquisitionW. EnableLineInfo.

unsigned long XImageW.Height Property (read only)

Remark:

Line number of frame.

unsigned long XImageW.PixelDepth Property (read only)

Remark:

Pixel depth of frame.

unsigned long XImageW.Size Property (read only)

Remark:

Size of frame, byte is the unit.

unsigned long XImageW.Width Property (read only)

Remark:

Column number of frame.

XMultiTransferW class

Description:

The XMultiTransferW class synchronizes multi-FrameTransfer objects while connecting with more than one X-GCU. It combines the sub-frames into one frame. The XAcquisitionW object should enable the line info mode while working with the XMultiTransferW object.

Methods & properties:

bool XMultiTransferW.AddTransfer(XFrameTransferW transfer) Method

Parameters:

transfer XFrameTransferW object pointer.

Return:

Returns 1 if successful, 0 if the total attached transfer object number is above 8 or the line number of each transfer object is not equal.

Remark:

Attaches transfer objects into the list. The line number of each transfer object must be 2^n (n < 15). This function should be called before opening.

void XMultiTransferW.Close() Method

Remark:

Releases all resources.

unsigned long XMultiTransferW.LastError Property (read only)

Remark:

Returns the error ID if any exception occurs Please refer to the error list for the explanation of the specific error.

bool XMultiTransferW.Open() Method

Return:

Returns 1 if successful, else return 0.

Remark:

To use the XMultiTransferW object, the XAcquisitionW object must enable the line info mode.

Delegations:

void XMultiTransferW.DelOnXError(int err_id, string err_msg)

Parameters:

err_id	Error ID.
err_msg_	Error description.

Remark:

The error callback of the XMultiTransferW object. Please refer to the section 5.3 for the specific error ID.

void XMultiTransferW.DelOnXEvent(int event_id, int data)

Parameters:

event_id	Event ID.
data	Event data.

Remark:

The event callback of the XMultiTransferW object. Please refer to the section 5.4 for the specific event ID. This callback responds to the event IDs from 50 to 55.

void XMultiTransferW.DelOnFrameReady(XImageW image)

Parameters:

image	XImageW object.

Remark:

When the frame is completed, the user can get the frame data in this callback function. Please note, if this function takes too much time to return, the frame buffer could overflow.

XOffCorrectW class

Description:

The XOffCorrectW class is in charge of the off-board or on-board correction functions. It manages the calculation process of gain and offset and saves the result to a disk or the flash of X-GCU board. If you are using non-continuous hi/lo trigger mode, please refer to XOffHLCorrectW class.

For the workflow of correction, please refer to chapter 8.

Methods & properties:

bool XOffCorrectW.CalculatePara(unsigned long type, XAcquisitionW acq, XFrameTransferW transfer , unsigned long start, unsigned long end, unsigned long target) Method

Parameters:

type	Define calculating which correction, 0 gain, 1 offset.
acq	XAcquistion object pointer.
transfer	XFrameTransfer object pointer.
start	Starting channel of correcting area.
end	Ending channel of correcting area.
target	0: Uses min channel value as the target; 1: Uses mean channel value as the target; 2: Uses max channel value as the target; Other: Uses the specific value as the target.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Calculates the gain or offset parameters.

bool XOffCorrectW.CalculatePara(unsigned long type, XAcquisitionW acq, XFrameTransferW transfer, uint32_t target) Method

Remark:

Calculates all the channels. (Please refer to the previous function).

void XOffCorrectW.Close() Method

Remark:

Releases all the allocated resources.

bool XOffCorrectW.DoCorrect(XImageW image) Method

Parameters:

Image	XImageW object.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Does the gain and offset correction.

void XOffCorrectW.LoadFile(string file) Method

Parameters:

file File name which saves gain and offset parameters.

Remark:

Loads the gain and offset parameters from the file.

bool XOffCorrectW.LoadFlash(XCommandW cmd, unsigned char index) Method

Parameters:

cmd XCommandW object.

index Gain index, rang 0-3.

Return:

Returns 1 if successful, otherwise 0.

Remark:

Loads the gain and offset parameters from the flash.

bool XOffCorrectW.Open(XDeviceW dev) Method

Parameters:

dev XDeviceW object.

Return:

Returns 1 if successful, otherwise 0.

Remark:

Allocates the needed resources.

void XOffCorrectW.Reset() Method

Remark:

Sets the gain to 1.0, offset to 0.

void XOffCorrectW.SaveFile(string file) Method

Parameters:

file File name which saves gain and offset parameters.

Remark:

Saves the gain and offset parameters to the file.

bool XOffCorrectW.SaveFlash(XCommandW cmd, unsigned char index) Method

Parameters:

cmd XCommandW object.

index Gain index, rang 0-3.

Return:

Returns 1 if successful, otherwise 0.

Remark:

Saves the gain and offset parameters to the flash. It can save 4 groups of gain by using different gain index.

XOffHLCorrectW class

Description:

The XOffHLCorrectW class is in charge of the off-board correction functions only in non-continuous hi/lo trigger mode.

For the workflow of correction, please refer to chapter 8.

Methods & properties:

bool XOffHLCorrectW.CalculatePara(unsigned long type, XAcquisitionW acq, XFrameTransferW transfer , unsigned long start, unsigned long end, unsigned long target) Method

Parameters:

type Define calculating which correction, 0 gain, 1 offset.

acq XAcquistion object pointer.

transfer XFrameTransfer object pointer.

start Starting channel of correcting area.

end Ending channel of correcting area.

target The target value to be corrected.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Calculates the gain or offset parameters.

void XOffHLCorrectW.Close() Method

Remark:

Releases all the allocated resources.

bool XOffHLCorrectW.DoCorrect(XImageW image) Method

Parameters:

Image XImageW object.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Does the gain and offset correction.

void XOffHLCorrectW.LoadFile(string file) Method

Parameters:

file File name which saves gain and offset parameters.

Remark:

Loads the gain and offset parameters from the file.

bool XOffHLCorrectW.Open(XDeviceW dev) Method

Parameters:

dev XDeviceW object.

Return:

Returns 1 if successful, otherwise 0.

Remark:

Allocates the needed resources.

void XOffHLCorrectW.Reset() Method

Remark:

Sets the gain to 1.0, offset to 0.

void XOffHLCorrectW.SaveFile(string file) Method

Parameters:

file File name which saves gain and offset parameters.

Remark:

Saves the gain and offset parameters to the file.

XPiecewiseCorrectW class

Description:

The XPiecewiseCorrectW object takes three air images corresponding to low, middle and high X-ray to build a more accurate piecewise correcting function. It only supports the off-board correction.

For the workflow of correction, please refer to chapter 8.

Methods & properties:

bool XPiecewiseCorrectW.CalculateGain1(XAcquisitionW acq, XFrameTransferW transfer, uint32_t target) Method

Parameters:

acq XAcquisitionW object.

transfer XFrameTransferW object.

target The target value for low X-ray.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Calculates the gain parameters of low X-ray.

bool XPiecewiseCorrectW.CalculateOffset1(XAcquisitionW acq, XFrameTransferW transfer) Method

Parameters:

acq XAcquisitionW object.

transfer XFrameTransferW object.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Calculates the offset parameters of low X-ray.

bool XPiecewiseCorrectW.CalculatePara2(XAcquisitionW acq, XFrameTransferW transfer, uint32_t target) Method

Parameters:

acq XAcquisitionW object.

transfer XFrameTransferW object.

target The target value for middle X-ray.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Calculates the gain and offset parameters of middle X-ray.

bool XPiecewiseCorrectW.CalculatePara3(XAcquisitionW acq, XFrameTransferW transfer, uint32_t target) Method

Parameters:

acq XAcquisitionW object.

transfer XFrameTransferW object.

target The target value for high X-ray.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Calculates the gain and offset parameters of high X-ray.

void XPiecewiseCorrectW.Close() Method

Remark:

Releases all the allocated resources.

bool XPiecewiseCorrect.DoCorrect(XImageW image) Method

Parameters:

image XImageW object.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Corrects the image data with piecewise parameters.

void XPiecewiseCorrectW.LoadFile(string file) Method

Parameters:

file File name which saves gain and offset parameters.

Remark:

Loads the gain and offset parameters from the file.

bool XPiecewiseCorrectW.Open(XDeviceW dev) Method

Parameters:

dev XDeviceW object.

Return:

Returns 1 if successful, otherwise 0.

Remark:

Allocates the needed resources.

void XPiecewiseCorrectW.SaveFile(string file) Method

Parameters:

file File name which saves gain and offset parameters.

Remark:

Saves the gain and offset parameters to the file.

XPixelCorrectW class

Description:

The XPixelCorrectW object corrects the bad pixels.

Methods & properties:

bool XPixelCorrectW.DoCorrect(XImageW image) Method

Parameters:

Image XImageW object.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Corrects the bad pixels.

void XPixelCorrectW.PushPixel(unsigned long pixel_pos) Method

Parameters:

pixel_pos Bad pixel position.

Remark:

Places the bad pixel position into a list.

void XPixelCorrectW.Refresh() Method

Remark:

Cleans the bad pixel list.

XReferenceCorrectW class

Description:

The XReferenceCorrectW object provides function to correct the signal variations caused by LINAC X-ray source.

Methods & properties:

bool XReferenceCorrectW.CalculatePara(XAcquisitioinW acq, XFrameTransferW transfer) Method

Parameters:

acq XAcquisitionW object.

transfer XFrameTransferW object.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Calculates the correcting parameters.

void XReferenceCorrectW.Close() Method

Remark:

Releases all the allocated resources.

bool XReferenceCorrectW.DoCorrect(XImageW image) Method

Parameters:

Image XImageW object.

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Corrects the image with signal variations.

bool XReferenceCorrectW.Open(XDeviceW dev, uint32_t ref_num) Method

Parameters:

dev XDeviceW object.

Ref_num Referenced pixel number.

Return:

Returns 1 if successful, otherwise 0.

Remark:

Allocates the needed resources.

XSystemW class

Description:

The XSystemW object communicates with the X-GCU by broadcast. It enumerates the IP, MAC and port info of the X-GCU and also configures the network settings.

Methods & properties:

void XSystemW.Close() Method

Remark:

Releases all the resources.

long XSystemW.ConfigureDevice(XDeviceW dev) Method

Parametares:

dev XDeviceW object.

Return:

Returns -1 if an error occurs, 1 successful.

Remark:

Configures the IP, MAC and the X-GCU port.

void XSystemW.EnableLog(BOOL enable) Method

Parameters:

enable Flag to enable log function: 0 disable, 1 enable.

Remark:

Enables or disables the log function; the default state is enabled. If the state is disabled, it should be called before opening.

long XSystemW.FindDevice() Method

Return:

Returns -1 if an error occurs, the device number otherwise.

Remark:

Enumerates the X-GCU devices which connect with the bound network adapter.

XDeviceW XSystem.GetDevice(unsigned long device_index) Method

Parameters:

device_index The device number index, starts from 0.

Return:

Returns the XDeviceW object pointer. If the device index is out of range, returns NULL.

bool XSystemW.IsOpen Property (read only)

Remark:

Returns the state, 1 opening, 0 closing.

unsigned long XSystemW.LastError Property (read only)

Return:

Returns the error ID if any exception occurs. Please refer to the error list for the explanation of the specific error.

bool XSystemW.Open() Method

Return:

Returns 1 if successful, 0 otherwise.

Remark:

Allocates the needed resources. If it returns 0, the user can utilize the XSystemW.LastError to trace the error.

string XSystemW.LocalIP Property (write only)

Remark:

Local IP address must be set before opening.

long XSystemW.RecoverDevice() Method

Return:

Returns -1 if an error occurs, 1 successful.

Remark:

Recovers the IP, MAC and port configurations to the default value. IP: 192.168.1.2, Cmd Port: 3000, Img Port: 4001.

unsigned long XSystemW.Timeout Property (write only)

Remark:

If the ACK does not arrive during the timeout period after sending a command, it will report an error. The timeout unit is ms with the default value 2000. The range is 0 to 2000000.

Delegations:

void XSystemW.DelOnXError(int err_id, string err_msg)

Parameters:

err_id Error ID.

err_msg_ Error description.

Remark:

The error callback of the XSystemW object. Please refer to the section 5.3 for the specific error ID.

XTifFormatW class

Description:

The XTifFormatW object could save some properties of detector into a tif format image, and also could load and parse the image.

Methods & properties:

IntPtr XTifFormatW.DataAddr Property (read\write)

Remark:

Image data pointer.

long XTifFormatW.GetPara(unsigned long para, unsigned long &data) Method

Parameters:

para Parameter type as defined in the following table.

data Returned value of the specific parameter.

Return:

Returns 1 if successful, 0 no such parameter.

Remark:

The user calls this function to get the specific parameter from the object.

Para	Data	Description
2	Returned value	Image width.
3	Returned value	Image height.
5	Returned value	Image pixel depth.
6	Returned value	The number of X-Cards.
7	Returned value	Card type, 0: LCS2 DM; 1: X-Card 0.2/0.4/0.8; 2: X-Card 1.5/1.6/2.5; 3: X-Card2 0.2/0.4/0.8; 4: Dual-row LCS2; 5: Aurora GT; 6: X-ICT; 7: Aurora B.
10	Returned value	DM's pixel number per energy mode. 5: 32; 6: 64; 7: 128;

Para	Data	Description
		8: 256; 9: 512.
11	Returned value	Operation mode. 0: Continuous mode; 1: Non-continuous mode 2: Constant integration time mode; 3: Non-continuous with Hi/Lo trigger.
12	Returned value	Integration time.
14	Returned value	Energy mode. 0: Single energy; 1: Dual energy stack; 2: Dual energy parallel.
15	Returned value	Binning mode. 0: Original pixel; 1: Averaging every 2 pixels; 2: Summing every 2 pixels; 3: Averaging every 4 pixels; 4: Summing every 4 pixels

long XTifFormatW.GetPara(unsigned long para, float &data) Method

Parameters:

para Parameter type as defined in the following table.

data Returned value of the specific parameter.

Return:

Returns 1 if successful, 0 no such parameter.

Remark:

The user calls this function to get the specific parameter from the object.

Para	Data	Description

Para	Data	Description
8	Returned value	The temperature of X-GCU
9	Returned value	The humidity of X-GCU.

string XTifFormatW.GetPara(unsigned long para) Method

Parameters:

para Parameter type as defined in the following table.

Return:

Returns specified parameter.

Remark:

The user calls this function to get the specific parameter from the object.

Para	Data	Description
4	Returned value	The serial number string pointer. The string length is less than 32 bytes.
13	Returned value	The date time string pointer. The string length is less than 32 bytes.

long XTifFormatW.Load(string file) Method

Parameters:

file ".tif" image file.

Return:

Returns 1 if successful, 0 failed.

Remark:

Reads and parses tif format image file.

long XTifFormatW.Save(string file) Method

Parameters:

file ".tif" image file.

Return:

Returns 1 if successful, 0 failed.

Remark:

Saves data as tif format image file.

long XTifFormatW.SetPara(unsigned long para, unsigned long data)
Method

Parameters:

para Parameter type as defined in the following table.

data Value to set.

Return:

Returns 1 if successful, 0 no such parameter.

Remark:

The user calls this function to set the specific parameter of the object.

Para	Data	Description
2	Image width	Image width.
3	Image height	Image height.
5	Pixel depth	Image pixel depth.
6	Card number	The number of X-Cards.
7	Card type	Card type, 0: LCS2 DM; 1: X-Card 0.2/0.4/0.8; 2: X-Card 1.5/1.6/2.5; 3: X-Card2 0.2/0.4/0.8; 4: Dual-row LCS2; 5: Aurora GT; 6: X-ICT; 7: Aurora B.
10	Energy mode	DM's pixel number per energy mode. 5: 32; 6: 64; 7: 128; 8: 256; 9: 512.

Para	Data	Description
11	Operation mode	Operation mode. 0: Continuous mode; 1: Non-continuous mode 2: Constant integration time mode; 3: Non-continuous with Hi/Lo trigger.
12	Integration time	Integration time.
14	Energy mode	Energy mode. 0: Single energy; 1: Dual energy stack; 2: Dual energy parallel.
15	Binning mode	Binning mode. 0: Original pixel; 1: Averaging every 2 pixels; 2: Summing every 2 pixels; 3: Averaging every 4 pixels; 4: Summing every 4 pixels

long XTifFormatW.SetPara(unsigned long para, unsigned long data) Method

Parameters:

para Parameter type as defined in the following table.

data Value to set.

Return:

Returns 1 if successful, 0 no such parameter.

Remark:

The user calls this function to set the specific parameter of the object.

Para	Data	Description
4	Serial number	The serial number string pointer. The string length is less than 32

Para	Data	Description
	string pointer	bytes.
13	Date time string pointer	The date time string pointer. The string length is less than 32 bytes.

long XTIfFormatW.SetPara(unsigned long para, float data) Method

Parameters:

para Parameter type as defined in the following table.

data Value to set.

Return:

Returns 1 if successful, 0 no such parameter.

Remark:

The user calls this function to set the specific parameter of the object.

Para	Data	Description
8	Temperature	The temperature of X-GCU.
9	Humidity	The humidity of X-GCU.

5.3. Error list

Error ID	Macro	Description
1	XERROR_SYS_SOCK_OPEN_FAIL	XSystemW object's socket fails to open.
2	XERROR_SYS_SOCK_BIND_FAIL	XSystemW object's socket fails to bind IP and port.
3	XERROR_SYS_SOCK_SEND_FAIL	XSystemW object's socket fails to send command.
4	XERROR_SYS_SOCK_RECV_TIMEOUT	XSystemW object's socket receives timeout.
5	XERROR_SYS_ENGINE_RECV_ERRCMD	XSystemW object receives error ACK.
6	XERROR_SYS_ENGINE_RECV_ERRCODE	XSystemW object receives error code.

Error ID	Macro	Description
7	XERROR_SYS_ENGINE_RECV_ERRCRC	XSystemW object receives error CRC.
8	XERROR_SYS_ENGINE_NOT_OPEN	XSystemW object is not opening.
9	XERROR_SYS_ALLOCATE_FAIL	XSystemW object fails to allocate resource.
10	XERROR_ASCPAS_FORMAT_ERR	XSystemW or XCommandW object gets error format ASCII command.
11	XERROR_ASCPAS_NONE_CMD	XSystemW or XCommandW object gets non-defined ASCII command.
12	XERROR_CMD_SOCK_OPEN_FAIL	XCommandW object's socket fails to open.
13	XERROR_CMD_SOCK_BIND_FAIL	XCommandW object's socket fails to bind IP and port
14	XERROR_CMD_SOCK_SEND_FAIL	XCommandW object's socket fails to send command.
15	XERROR_CMD_SOCK_RECV_TIMEOUT	XCommandW object's socket receives timeout.
16	XERROR_CMD_ENGINE_RECV_ERRCMD	XCommandW object receives error ACK.
17	XERROR_CMD_ENGINE_RECV_ERRCODE	XCommandW object receives error code.
18	XERROR_CMD_ENGINE_RECV_ERRCRC	XCommandW object receives error CRC.
19	XERROR_CMD_ENGINE_NOT_OPEN	XCommandW object is not opening.
20	XERROR_CMD_ALLOCATE_FAIL	XCommandW object fails to allocate resource.
21	XERROR_IMG_SOCK_OPEN_FAIL	XAcquisitionW object's socket fails to open.
22	XERROR_IMG_SOCK_BIND_FAIL	XAcquisitionW object's socket fails to bind IP and port.
23	XERROR_IMG_SOCK_RECV_TIMEOUT	XAcquisitionW object's socket receives timeout.
24	XERROR_IMG_ALLOCATE_FAIL	XAcquisitionW object fails to allocate resource.

Error ID	Macro	Description
25	XERROR_IMG_ENGINE_NOT_OPEN	XAcquisitionW grabbing engine is not opening.
26	XERROR_IMG_ENGINE_START_FAIL	XAcquisitionW grabbing engine fails to start grabbing.
27	XERROR_IMG_ENGINE_STOP_ABNORMAL	XAcquisitionW grabbing engine stops abnormally.
28	XERROR_IMG_PARSE_OPEN_FAIL	XAcquisitionW parsing engine fails to open.
29	XERROR_IMG_PARSE_NOT_OPEN	XAcquisitionW parsing engine is not open.
30	XERROR_IMG_PARSE_START_FAIL	XAcquisitionW parsing engine fails to start.
31	XERROR_IMG_PARSE_STOP_ABNORMAL	XAcquisitionW parsing engine stops abnormally.
32	XERROR_IMG_TRANSFER_NOT_OPEN	XFrameTransferW object is not open.
33	XERROR_IMG_TRANSFER_START_FAIL	XFrameTransferW object fails to start.
34	XERROR_IMG_TRANSFER_STOP_ABNORMAL	XFrameTransferW object stops abnormally.
35	XERROR_IMG_PACKET_POOL_OPEN_FAIL	Packet pool fails to open.
36	XERROR_MULTI_TRANS_ALLOCATE_FAIL	XMultiTransferW object fails to allocate resource.
37	XERROR_MULTI_TRANS_NOT_OPEN	XMultiTransferW object is not opening.
38	XERROR_DISP_ALLOCATE_FAIL	XDisplayW object fails to allocate resource.
39	XERROR_CMD_HEARTBEAT_FAIL	XCommandW fails to get heartbeat packet.
40	XERROR_CMD_HEARTBEAT_START_FAIL	XCommandW fails to start heartbeat thread.
41	XERROR_CMD_HEARTBEAT_STOP_ABNORMAL	XCommandW stops heartbeat thread abnormally.
42	XERROR_CMD_HEARTBEAT_VOL_ERR	X-GCU’s voltage is out of range.

Error ID	Macro	Description
43	XERROR_IMG_ENGINE_GRAB_ABNORMAL	XAcquisitionW object grabbing engine works abnormally.
44	XERROR_FILE_OPERATE_ERROR	XTifFormat object fails to operate tif file.
45	XERROR_FILE_TIF_TAG_ENTRY_ABNORMAL	XTifFormat object fails to parse tag entry of tif file.
46	XERROR_FILE_TIF_COMPRESSED	XTifFormat object doesn't support compressed tif file.
47	XERROR_FILE_TIF_ALLOC_FAIL	XTifFormat object fails to allocate data buffer.
48	XERROR_MULTI_TRANS_START_FAIL	XMultiTransferW object fails to start thread.
49	XERROR_MULTI_TRANS_STOP_ABNORMAL	XMultiTransferW object fails to stop thread.

5.4. Event list

Event ID	Macro	Data	Description
50	XEVENT_IMG_PARSE_DATA_LOST	Lost line number	X-GCU line data lost.
51	XEVENT_IMG_TRANSFER_BUF_FULL	Buffer size	XFrameTransfer buffer is full.
52	XEVENT_IMG_PARSE_DM_DROP	DM packet number received	DM packet drops during LVDS transmission.
53	XEVENT_IMG_PARSE_PAC_LOST	Lost packet number	X-GCU packet drops during Ethernet transmission.
54	XEVENT_IMG_PARSE_CRC_ERR	Error DM ID	DM packet CRC error.
55	XEVENT_IMG_PARSE_VOL_ERR	Error DM ID	DM voltage error.
56	XEVENT_CMD_HEARTBEAT_TEMPRA	Temperature value	Reports X-GCU's temperature.
57	XEVENT_CMD_HEARTBEAT_HUMIDITY	Humidity value	Reports X-GCU's humidity.

6. COMMAND LIST

6.1. HEX command

Thwe X-GCU command protocol is described in the section 2.3. According to the protocol, the API function XCommand::SendCommand() provides a low level mechanism to communicate with the X-GCU. The specific HEX commands are listed as a table 6-1. All the listed commands can be used as parameters of the XCommand::SendCommand().

CMD	Specific command code
OPE	Operation mode: none/0x00, write/0x01, read/0x02, save/0x03 or load/0x04
DM ID	DM ID number or address: none/0x00, all/0xFF or specific ID number
SIZE	Size of DATA segment
DATA	Data value: none or specific data
ERR ID	Success/0x00, EPCS(flash read/write) error/0x01, LVDS receive CRC error/0x02, calibration data read/write error/0x03, X-GCU gets undefined command/0x04, X-GCU SN not match while setting MAC/IP/0x05, timeout or no reply from X-DFE to X-GCU/0x06, X-GCU receives CRC error from PC/0x07, X-GCU receives parameter out of range/0x08

Table 6-1. HEX command list

Application command					**Description**	**X-GCU ACK**				
CMD	**OPE**	**DM ID**	**SIZE**	**DATA**		**CMD**	**ERRID**	**DM ID**	**SIZE**	**DATA**
0x10	0x04	0x00	0x00	None	Initializes X-GCU device from the user flash	0x10	0x00: Success; Other: Fail	0x00	0x00	None
	0x03	0x00	0x00	None	Saves the settings to the user flash		0x00: Success; Other: Fail	0x00	0x00	None
0x11	0x04	0x00	0x00	None	Initializes with default parameters from the main flash	0x11	0x00: Success; Other: Fail	0x00	0x00	None
	0x03	0x00	0x00	None	Saves default parameters to the main and backup flash		0x00: Success; Other: Fail	0x00	0x00	None

Application command					Description	X-GCU ACK				
0x20	0x01	0x00	0x04	Integration time value	Sets integration time, the unit is us	0x20	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads integration time		0x00: Success; Other: Fail	0x00	0x04	Integration time value
0x21	0x01	0x00	0x02	Non-continuous integration time value	Sets non-continuous integration time, the unit is us	0x21	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads non-continuous integration time		0x00: Success; Other: Fail	0x00	0x02	Integration time value
0x22	0x01	0x00	0x01	0x00: Continuous mode; 0x01: Non-continuous mode; 0x02: Constant integration time mode; 0x03: Non-continuous with Hi/Lo trigger	Sets operation mode	0x22	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads operation mode		0x00: Success; Other: Fail	0x00	0x01	0x00: Continuous mode; 0x01: Non-continuous mode; 0x02: Constant integration time mode; 0x03: Non-continuous with Hi/Lo trigger
0x23	0x01	0xFF: All DMs; Other: Specif	0x02	High gain 1byte, low gain 1byte	Sets DM gain value. It is not available while scanning	0x23	0x00: Success; Other: Fail	0xFF: All DMs; Other: Specific DM	0x00	None

Application command					Description	X-GCU ACK				
		ic DM								
	0x02	0xFF: All DMs; Others: Specific DM	0x00	None	Reads DM gain value. It is not available while scanning		0x00: Success; Other: Fail	Specific DM	0xXX	High gain 1byte, low gain 1byte
0x24	0x01	0x00	0x01	0x00: Low gain; 0x01: High gain;	Sets high or low energy operation mode in single energy mode	0x24	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads high or low energy operation mode in single energy mode		0x00: Success; Other: Fail	0x00	0x01	0x00: Low gain; 0x01: High gain;
0x25	0x01	0x00	0x05	First channel's DM number: 1byte; Second channel's DM number: 1byte; Third channel's DM number: 1byte; Forth channel's DM number: 1byte; Fifth channel's DM number: 1byte	Sets DM number of each channel	0x25	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads DM number of each channel		0x00: Success; Other: Fail	0x00	0x05	First channel's DM number: 1byte; Second channel's

Application command					Description	X-GCU ACK				
										DM number: 1byte; Third channel's DM number: 1byte; Forth channel's DM number: 1byte; Fifth channel's DM number: 1byte
0x27	0x01	0x00	0x01	0x00: Stop scanning; 0x01: Start scanning	Starts/stops scanning	0x27	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads scanning status		0x00: Success; Other: Fail	0x00	0x01	0x00: Stop scanning; 0x01: Start scanning
0x30	0x01	0x00	0x01	0x00: Disable gain correction; 0x01: Enable gain correction	Enables gain correction	0x30	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads gain correction mode		0x00: Success; Other: Fail	0x00	0x01	0x00: Disable gain correction; 0x01: Enable gain correction
0x31	0x01	0x00	0x01	0x00: Disable offset correction; 0x01: Enable offset correction	Enables offset correction	0x31	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads offset correction mode		0x00: Success; Other: Fail	0x00	0x01	0x00: Disable offset correction; 0x01: Enable offset

Application command					Description	X-GCU ACK				
										correction
0x32	0x01	0x00	0x01	0x00: Disable base line correction; 0x01: Enable base line correction	Enables base line correction	0x32	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads base line correction mode		0x00: Success; Other: Fail	0x00	0x01	0x00: Disable base line correction; 0x01: Enable base line correction
0x33	0x03	0x00	0xXX	Flash index: 1 byte, 0-3; Payload index: 1 byte; Payload: 1024 bytes, each gain account for 4 bytes, float type	Saves gain to the flash	0x33	0x00: Success; Other: Fail	0x00	0x00	None
	0x04	0x00	0x01	Flash index to load gain, range is 0-3	Loads gain from the flash		0x00: Success; Other: Fail	0x00	0x00	None
0x34	0x03	0x00	0xXX	Payload index: 1 byte; Payload: 1024 bytes, each offset account for 4 bytes	Saves offset to the flash	0x34	0x00: Success; Other: Fail	0x00	0x00	None
	0x04	0x00	0x00	None	Loads offset from the flash		0x00: Success; Other: Fail	0x00	0x00	None
0x35	0x01	0x00	0x02	Base line value	Sets base line value	0x35	0x00: Success; Other: Fail	0x00	0x00	None

Application command					Description	X-GCU ACK				
	0x02	0x00	0x00	None	Reads base line value		0x00: Success; Other: Fail	0x00	0x02	Base line value
0x37	0x00	0x00	0x00	None	Resets gain to 1.0	0x37	0x00: Success; Other: Fail	0x00	0x00	None
0x38	0x00	0x00	0x00	None	Resets offset to 0	0x38	0x00: Success; Other: Fail	0x00	0x00	None
0x39	0x03	0x00	0xXX	Payload index: 1 byte; Payload: 1024 bytes, each data account for 4 bytes	Saves PDC position data to the flash	0x39	0x00: Success; Other: Fail	0x00	0x00	None
	0x04	0x00	0x00	None	Loads PDC position data from flash		0x00: Success; Other: Fail	0x00	0x00	None
0x3A	0x03	0x00	0xXX	Payload index: 1 byte; Payload: 1024 bytes, each data account for 4 bytes, float type	Saves PDC coefficient data to the flash	0x3A	0x00: Success; Other: Fail	0x00	0x00	None
	0x04	0x00	0x00	None	Loads PDC coefficient data from the flash		0x00: Success; Other: Fail	0x00	0x00	None
0x3B	0x01	0x00	0x01	0x00: Disable PDC correction; 0x01: Enable PDC correction	Enables PDC correction	0x3B	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads PDC correction mode		0x00: Success; Other: Fail	0x00	0x01	0x00: Disable PDC correction; 0x01: Enable PDC correction

Application command					Description	X-GCU ACK				
0x3C	0x01	0x00	0x01	Pixel position: 0x00 – 0xFF	Sets PDC correction pixel position	0x3C	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads PDC correction pixel position		0x00: Success; Other: Fail	0x00	0x01	Pixel position: 0x00 – 0xFF
0x40	0x01	0x00	0x01	0x00: Original pixel; 0x01: Averaging every 2 pixels; 0x02: Summing every 2 pixels; 0x03: Averaging every 4 pixels; 0x04: Summing every 4 pixels	Pixelbinning	0x40	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads pixel binning status		0x00: Success; Other: Fail	0x00	0x01	0x00: Original pixel; 0x01: Averaging every 2 pixels; 0x02: Summing every 2 pixels; 0x03: Averaging every 4 pixels; 0x04: Summing every 4 pixels
0x41	0x01	0x00	0x01	Filter value n: 0x00-0x08	Line averaging, the filter size is 2^n	0x41	0x00: Success; Other: Fail	0x00	0x00	None

Application command					Description	X-GCU ACK				
	0x02	0x00	0x00	None	Reads line averaging file size		0x00: Success; Other: Fail	0x00	0x01	Filter value n: 0x00-0x08
0x42	0x01	0x00	0x01	Filter vale n: 0x00-0x08	Line summing, the filter size is 2^n	0x42	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads line summing the filter size		0x00: Success; Other: Fail	0x00	0x01	Filter vale n: 0x00-0x08
0x43	0x01	0x00	0x01	0x00:original 0x01:original -1 0x02:original -2 0x03:original -3 0x04:original -4 0x05:original -5 0x06:original -6 0x07:original -7 0x08:original -8 …	Sets output scale	0x43	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads output scale		0x00: Success; Other: Fail	0x00	0x01	0x00:original 0x01:original -1 0x02:original -2 0x03:original -3 0x04:original -4 0x05:original -5 0x06:original -6 0x07:original -7 0x08:original -8 …
0x44	0x01	0x00	0x01	0x00: 1 line; 0x01: 2 lines; 0x02: 4 lines; 0x03: 8 lines	Sets offset averaging filter	0x44	0x00: Success; Other: Fail	0x00	0x00	None

Application command					Description	X-GCU ACK				
	0x02	0x00	0x00	None	Reads offset averaging filter status		0x00: Success; Other: Fail	0x00	0x01	0x00: 1 line; 0x01: 2 lines; 0x02: 4 lines; 0x03: 8 lines
0x50	0x01	0x00	0x01	0x00: Rising edge mode; 0x01: Falling edge mode; 0x02: Sync trigger stamp mode; 0x03: Async trigger stamp mode	Sets external line trigger mode	0x50	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads external line trigger mode		0x00: Success; Other: Fail	0x00	0x01	0x00: Rising edge mode; 0x01: Falling edge mode; 0x02: Sync trigger stamp mode; 0x03: Async trigger stamp mode
0x51	0x01	0x00	0x01	0x00: Disable external line trigger; 0x01: Enable normal external line trigger;	Enables external line trigger	0x51	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads external line trigger status		0x00: Success; Other: Fail	0x00	0x01	0x00: Disable external line trigger;

Application command					Description	X-GCU ACK				
										0x01: Enable external line trigger;
0x52	0x01	0x00	0x01	Unit is 0.12us	Sets line trigger fine delay	0x52	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads line trigger fine delay		0x00: Success; Other: Fail	0x00	0x01	Unit is 0.12us
0x53	0x01	0x00	0x01	Unit is 64us	Sets line trigger raw delay	0x53	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads line trigger raw delay		0x00: Success; Other: Fail	0x00	0x01	Unit is 64us
0x54	0x01	0x00	0x01	0x00: Rising edge mode; 0x01: Falling edge mode	Sets external frame trigger mode	0x54	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads external frame trigger mode		0x00: Success; Other: Fail	0x00	0x01	0x00: Rising edge mode; 0x01: Falling edge mode
0x55	0x01	0x00	0x02	0x0000: Disable external frame trigger; n: Enable external frame trigger, the frame height is n*32	Enables external frame trigger	0x55	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads external frame trigger status		0x00: Success; Other: Fail	0x00	0x02	0x00: Disable external frame trigger; n: Enable external frame

Application command					Description	X-GCU ACK				
										trigger, the frame height is n*32
0x56	0x01	0x00	0x01	Unit is 32 lines	Sets frame trigger raw delay	0x56	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads frame trigger raw delay		0x00: Success; Other: Fail	0x00	0x01	Unit is 32 lines
0x57	0x00	0x00	0x00	None	Sends a simulated external frame trigger	0x57	0x00: Success; Other: Fail	0x00	0x00	None
0x58	0x01	0x00	0x01	0x00: lo edge; 0x01: hi edge	Sets Hi/Lo line trigger with hi/lo edge level detection	0x58	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads Hi/Lo line trigger with hi/lo edge level detection		0x00: Success; Other: Fail	0x00	0x01	0x00: lo edge; 0x01: hi edge
0x59	0x01	0x00	0x02	Sampling position, unit is 1us	Sets Hi/Lo line trigger sampling position	0x59	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads Hi/Lo line trigger sampling position		0x00: Success; Other: Fail	0x00	0x02	Sampling position, unit is 1us
0x5A	0x01	0x00	0x01	0x00: Disable parity check; 0x01: Use "odd" parity check; 0x02: Use "even" parity check	Sets trigger stamp parity check mode	0x5A	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads trigger stamp parity check mode		0x00: Success; Other: Fail	0x00	0x01	0x00: Disable parity check; 0x01: Use "odd" parity check;

Application command					Description	X-GCU ACK				
										0x02: Use "even" parity check
0x60	0x01	0x00	0x01	0x00: Stop Other: Period of heartbeat packets, unit is 1s	Starts/stops heartbeat packets	0x60	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads heart status		0x00: Success; Other: Fail	0x00	0x01	0x00: Stop Other: Period of heartbeat packets, unit is 1s
0x61	0x01	0x00	0x01	0x00: Ethernet; 0x01: Camlink; Other : Reserved	Sets X-GCU type	0x61	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads X-GCU type		0x00: Success; Other: Fail	0x00	0x01	0x00: Ethernet; 0x01: Camlink; Other : Reserved
0x62	0x01	0x00	0x20	X-GCU's S/N 32bytes	Sets X-GCU S/N	0x62	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads X-GCU S/N		0x00: Success; Other: Fail	0x00	0x20	X-GCU's S/N 32bytes
0x63	0x01	Specif ic DM	0x20	DM's S/N 32bytes	Sets DM S/N. It is not available while scanning	0x63	0x00: Success; Other: Fail	Specific DM	0x00	None
	0x02	Specif ic DM	0x00	None	Reads DM S/N. It is not available while scanning		0x00: Success; Other: Fail	Specific DM	0x20:	DM's S/N 32bytes
0x64	0x02	0x00	0x00	None	Reads pixel number	0x64	0x00: Success; Other: Fail	0x00	0x02	Pixel number

Application command					Description	X-GCU ACK				
0x65	0x01	0x00	0x01	Real pixel size * 10	Sets pixel size	0x65	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads pixel size		0x00: Success; Other: Fail	0x00	0x01	Real pixel size * 10
0x66	0x01	0x00	0x01	0x10: 16bits; 0x12: 18bits; 0x14: 20bits; 0x18: 24bits	Sets pixel depth.	0x66	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads pixel depth.		0x00: Success; Other: Fail	0x00	0x01	0x10: 16bits; 0x12: 18bits; 0x14: 20bits; 0x18: 24bits
0x67	0x01	0x00	0x08	Min integration time: 4bytes; Max integration time: 4bytes	Sets min/max integration time	0x67	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads min/max integration time		0x00: Success; Other: Fail	0x00	0x08	Min integration time: 4bytes; Max integration time: 4bytes
0x68	0x02	0x00	0x00	None	Reads X-GCU firmware version	0x68	0x00: Success; Other: Fail	0x00	0x02	X-GCU firmware version
0x69	0x02	Specific DM	0x00	None	Reads DM firmware version. It is not available while scanning	0x69	0x00: Success; Other: Fail	Specific DM	0x02	DM firmware version

Application command					Description	X-GCU ACK				
0x6A	0x01	0x00	0x01	0x00: Disable; 0x01: Normal test pattern; 0x02: Slope test pattern	Sets test pattern status	0x6A	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads test pattern status		0x00: Success; Other: Fail	0x00	0x01	0x00: Disable; 0x01: Normal test pattern; 0x02: Slope test pattern
0x6B	0x01	0xFF: All DMs; Other: Specific DM	0x01	0x00: Normal image data mode; 0x01: Enable DM board fixed value (AAAAA[Hex]) test mode.; 0x02: Enable I'm here function	Sets DM's test mode. It is not available while scanning	0x6B	0x00: Success; Other: Fail	0xFF: All DMs; Other: Specific DM	0x00	None
	0x02	0xFF: All DMs; Other: Specific DM	0x00	None	Reads DM's test mode. It is not available while scanning		0x00: Success; Other: Fail	Specific DM	0xXX	0x00: Normal image data mode; 0x01: Enable DM board fixed value (AAAAA[Hex]) test mode.; 0x02: Enable I'm here function;
0x6C	0x01	0x00	0x01	0x05: 32; 0x06: 64; 0x07: 128; 0x08: 256;	Sets pixel number per DM	0x6C	0x00: Success; Other: Fail	0x00	0x00	None

Application command					Description	X-GCU ACK				
				0x09: 512						
	0x02	0x00	0x00	None	Reads pixel number per DM		0x00: Success; Other: Fail	0x00	0x01	0x05: 32; 0x06: 64; 0x07: 128; 0x08: 256; 0x09: 512
0x6D	0x01	0x00	0x01	Card number	Sets card number per DFE	0x6D	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00`	0x00	None	Reads card number per DFE		0x00: Success; Other: Fail	0x00	0x01	Card number
0x6E	0x01	0x00	0x01	0x00: LCS2 DM; 0x01: X-Card 0.2/0.4/0.8; 0x02: X-Card 1.5/1.6/2.5; 0x03: X-Card2 0.2/0.4/0.8; 0x04: Dual-row LCS2 DM; 0x05: Aurora GT; 0x06: X-ICT; 0x07: Aurora B	Sets card type	0x6E	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads card type		0x00: Success; Other: Fail	0x00	0x01	0x00: LCS2 DM; 0x01: X-Card 0.2/0.4/0.8; 0x02: X-Card 1.5/1.6/2.5;

Application command					Description	X-GCU ACK				
										0x03: X-Card2 0.2/0.4/0.8; 0x04: Dual-row LCS2 DM; 0x05: Aurora GT; 0x06: X-ICT; 0x07: Aurora B
0x72	0x02	0x00	0x00	None	Reads X-GCU's temperature and voltage. Voltage value = (voltage high byte*256 + voltage low byte)*2.048/2047 (V) v1 = voltage value*24/1.5(V) v2 = voltage value*2(V) v3 = voltage value*2(V) v4 = voltage value(V) Temperature value = (temperature high byte*256 + temperature low byte)*0.125(°C) Humidity value = (humidity high byte*256 + humidity low byte)*125/65536 – 6(%)	0x72	0x00: Success; Other: Fail	0x00	0x0C	X-GCU's voltage: 2bytes * 4 Temperature: 2bytes; Humidity: 2bytes
0x73	0x02	Specific DM	0x00	None	Reads DM's voltage, temperature and humidity. It is not available while scanning Voltage value = (voltage high byte*256 + voltage low byte)*2.048/20	0x73	0x00: Success; Other: Fail	0x00	0x14	DM's voltage: 2bytes*7; Temperature:2bytes; Humidity: 2bytes; High resolution

Application command					Description	X-GCU ACK				
					47 (V) v1 = voltage value*2(V) v2 = voltage value*24/1.5(V) v3 = voltage value*5.2/1.66(V) v4 = voltage value(V) v5 = voltage value*5.0/1.6(V) v6 = voltage value*2(V) v7 = voltage value*4.096/1.3(V) (for Aurora GT only v1~v4 are available: v1 = voltage value*13.1(V) v2 = voltage value*2(V) v3 = voltage value*3(V) v4 = voltage value*2(V) for Aurora B, no voltage is available) Temperature value = (temperature high byte*256 + temperature low byte)*0.125(°C) Humidity value = (humidity high byte*256 + humidity low byte)*125/65536 – 6(%) High resolution temperature = (highbyte*256+ lowbyte)/128					temperature :2bytes The high resolution temperature is only available with the DM which has high resolution sensor installed (high tier Digital X-Cards and LCS2 DM)

Application command					Description	X-GCU ACK				
0x75	0x01	0x00	0x01	0x00: Disable; 0x01: Enable	Enables LED light	0x75	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00`	0x00	None	Reads LED light state		0x00: Success; Other: Fail	0x00	0x01	0x00: Disable; 0x01: Enable
0x76	0x01	0x00	0x01	0x00: Disable; 0x01: Enable	Enables terminal resistor	0x76	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00`	0x00	None	Reads terminal resistor state		0x00: Success; Other: Fail	0x00	0x01	0x00: Disable; 0x01: Enable
0x77	0x00	Channel ID: 0x01 ~0x05	0x01	Test data	LVDS loopback	0x77	0x00: Success; Other: Fail	Channel ID	0x01	Test data
0x78	0x02	0x00	0x00	None	Reads hidden X-GCU firmware version	0x78	0x00: Success; Other: Fail	0x00	0x01	Hidden X-GCU firmware version
0x79	0x02	Specific DM	0x00	None	Reads hidden DM firmware version. It is not available while scanning	0x79	0x00: Success; Other: Fail	Specific DM	0x01	Hidden DM firmware version
0x7A	0x01	0x00	0x01	0x00: Low from first, high from first; 0x01: Low from last, high from first; 0x02: Low from first, high from last; 0x03: Low from last, high from last	Sets pixel order mode of card	0x7A	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads pixel order mode of card		0x00: Success; Other: Fail	0x00	0x01	0x00: Low from first, high from first;

Application command					Description	X-GCU ACK				
										0x01: Low from last, high from first; 0x02: Low from first, high from last; 0x03: Low from last, high from last
0x7B	0x01	0x00	0x01	0x00: SE; 0x01: DE stacked; 0x02: DE parallel	Sets energy mode	0x7B	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads energy mode		0x00: Success; Other: Fail	0x00	0x01	0x00: SE; 0x01: DE stacked; 0x02: DE parallel
0x7C	0x01	0x00	0x01	0x01~0x05	Sets gain table ID	0x7C	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Read gain table ID		0x00: Success; Other: Fail	0x00	0x01	0x01~0x05
0x7D	0x01	0x00	0x80	Product info ASCII code, 128bytes	Sets Product Info	0x7D	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads Product Info		0x00: Success; Other: Fail	0x00	0x80	Product info ASCII code, 128bytes
0x7E	0x01	0x00	0x01	0x00: 1500 bytes; 0x01: 8192 bytes	Sets MTU size	0x7E	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads MTU size		0x00: Success; Other: Fail	0x00	0x01	0x00:1500bytes 0x01:8192bytes

Application command					Description	X-GCU ACK				
0x80	0x01	0x00	0x01	0x00: Disable; 0x01: Enable	Enables Ethernet LED	0x80	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads Ethernet LED state		0x00: Success; Other: Fail	0x00	0x01	0x00: Disable; 0x01: Enable
0x8A	0x01	0x00	0x01	0x00: Not swap; 0x01: Swap	Sets swap mode of low energy and high energy part	0x8A	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads swap mode of low energy and high energy part		0x00: Success; Other: Fail	0x00	0x01	0x00: Not swap; 0x01: Swap
0x8B	0x01	0x00	0x01	0x00~0xFF	Sets Z gap value, step is 0.1mm	0x8B	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads Z gap value		0x00: Success; Other: Fail	0x00	0x01	0x00~0xFF
0x8C	0x01	0x00	0x04	Powering up time	Sets X-GCU powering up time	0x8C	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads X-GCU powering up time		0x00: Success; Other: Fail	0x00	0x04	Powering up time
0x8E	0x02	0x00	0x00	None	Reads X-GCU working time	0x8E	0x00: Success; Other: Fail	0x00	0x02	GCU working time
0x8F	0x01	0x00	0x02	Offset delay value unit : us	Sets offset delay value	0x8F	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads offset delay value		0x00: Success; Other: Fail	0x00	0x02	Delay value
0x90	0x01	0x00	0x01	0x00: Disable offset delay time setting 0x01: enable Off	Enable/disable offset delay	0x90	0x00: Success; Other: Fail	0x00	0x00	None

Application command					Description	X-GCU ACK				
				set delay setting						
	0x02	0x00	0x00	none	Reads offset delay setting		0x00: Success; Other: Fail	0x00	0x01	Offset delay setting
0x91	0x01	0x00	0x01	0x00: Normal mode(low energy part preceding with high energy part); 0x01: Interweave mode(low energy data interweaved with high energy data)	Sets camera link data format in dual energy mode	0x91	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads camera link data format in dual energy mode		0x00: Success; Other: Fail	0x00	0x01	Camera link data format in dual energy mode
0x92	0x01	0x00	0x01	0x00: 115200 0x01: 9600	Sets serial port communication baud rate	0x92	0x00: Success; Other: Fail	0x00	0x00	None
	0x02	0x00	0x00	None	Reads baud rate		0x00: Success; Other: Fail	0x00	0x01	Baud rate
0x94	0x00	0xFF: All DMs; Other: Specific DM	0x00	None	Resets AD chip. Only applicable to Aurora GT	0x94	0x00: Success; Other: Fail	0xFF: All DMs; Other: Specific DM	0x00	None
0x95	0x00	0x00: GCU 0xFF: All DMs; Other: Specif	0x00	None	Reconfigures firmware. Only applicable to Aurora GT	0x95	0x00: Success; Other: Fail	0x00: GCU 0xFF: All DMs; Other: Specific DM	0x00	None

Application command					Description	X-GCU ACK				
		ic DM								
0x96	0x01	0xFF: All DMs; Other: Specific DM	0x01	0x00: Disable; 0x01: Enable	Enable/disable AD Vref. Only applicable to Aurora GT	0x96	0x00: Success; Other: Fail	0xFF: All DMs; Other: Specific DM	0x00	None
	0x02	0xFF: All DMs; Other: Specific DM	0x00	None	Read AD Vref status. Only applicable to Aurora GT		0x00: Success; Other: Fail	0xFF: All DMs; Other: Specific DM	0x01	0x00: Disable; 0x01: Enable
0x97	0x01	0xFF: All DMs; Other: Specific DM	0x01	0x00: Disable; 0x01: Enable	Enable/disable PD connection. Only applicable to Aurora GT	0x97	0x00: Success; Other: Fail	0xFF: All DMs; Other: Specific DM	0x00	None
	0x02	0xFF: All DMs; Other: Specific DM	0x00	None	Reads PD connection status. Only applicable to Aurora GT		0x00: Success; Other: Fail	0xFF: All DMs; Other: Specific DM	0x01	0x00: Disable; 0x01: Enable
0x98	0x01	0xFF: All DMs; Other: Specific DM	0x01	0x00: Low speed; 0x01: High speed	Sets AD speed mode. Only applicable to Aurora GT	0x98	0x00: Success; Other: Fail	0xFF: All DMs; Other: Specific DM	0x00	None
	0x02	0xFF: All DMs; Other: Specific DM	0x00	None	Reads AD speed mode. Only applicable to Aurora GT		0x00: Success; Other: Fail	0xFF: All DMs; Other: Specific DM	0x01	0x00: Low speed; 0x01: High speed
0xXX	0xXX	0xXX	0xXX	0xXX	No such command	0x00	0x00	0x00	0x00	0x00

6.2. HEX command frame

To accelerate the command transmission, several commands are combined into a single command frame. These frames are "Operation Frame", "Diagnostic Frame" and "Configuration Frame". Please see the following table.

Table 6-2 HEX command frame list

Application command					Description	GCU ACK				
CMD	OPE	DM ID	SIZE	DATA		CMD	ERRID	DM ID	SIZE	DATA
0xF1	0x01	0x00	0xXX	0~3: integration time; 4~5: non-continuous integration time; 6: operation mode; 7: X-DFE gain mode; 8~12: X-DFE number of each segment; 13: RESEVE; 14: enable/disable gain correction; 15: enable/disable offset correction; 16: enable/disable base value correction; 17~18: base line value; 19: enable/disable pixel binning; 20: enable/disable average line filter; 21: enable/disable sum line filter; 22: set output scale bit; 23: set offset averaging filter; 24: set line trigger mode; 25: enable/disable external line trigger; 26: line trigger fine delay; 27: line trigger raw delay; 28: frame trigger mode; 29: line number for one frame; 30: frame trigger raw delay; 31: Set Hi/Lo line trigger with	Operation frame	0xF1	0x00: Success; Others: Fail	0x00	0x00	None

Application command					Description	GCU ACK				
				hi/lo edge level detection; 32~33: Set Hi/Lo line trigger sampling position; 34: set heartbeat; 35: set X-GCU type; 36: pixel size; 37: pixel depth; 38~45: integration time range; 46: X-GCU test mode; 47: pixel number per card; 48: card number per DFE; 49: card type; 50~99: reserved; From the 100 byte: DM1: LE Gain: 1byte; DM1: HE Gain: 1byte; DM2: LE Gain: 1byte; DM2: HE Gain:1 byte; … DMn: LE Gain: 1byte; DMn: HE Gain: 1byte Fixed size is 516, unused bytes filled by 0xFF						
0xF2	0x01	0x00	0xXX	0~3: integration time; 4~5: non-continuous integration time; 6: operation mode; 7: X-DFE gain mode; 8~12: X-DFE number of each segment; 13: RESERVE; 14: enable/disable gain correction; 15: enable/disable offset correction; 16: enable/disable base value	Diagnostic frame to enable diagnostic mode	0xF2	0x00: Success; Others: Fail	0x00	0x00	None

Application command					Description	GCU ACK				
				correction; 17~18: base line value; 19: enable/disable pixel binning; 20: enable/disable average line filter; 21: enable/disable sum line filter; 22: set output scale bit; 23: set offset averaging filter; 24: set line trigger mode; 25: enable/disable external line trigger; 26: line trigger fine delay; 27: line trigger raw delay; 28: frame trigger mode; 29: line number for one frame; 30: frame trigger raw delay; 31: Set Hi/Lo line trigger with hi/lo level detection; 32~33: Set Hi/Lo line trigger sampling position; 34: set heartbeat; 35: set X-GCU type; 36: pixel size; 37: pixel depth; 38~45: integration time range; 46: X-GCU test mode; 47: pixel number per card; 48: card number per X-DFE; 49: card type; 50~99: reserved; From the 100 byte: DM1 Diag.Mode: 1byte; DM2 Diag.Mode: 1byte; …						

Application command					Description	GCU ACK				
				DMn Diag.Mode:1byte Fixed size is 516, unused bytes filled by 0xFF						
0xF3	0x01	0xFF:all DMs; Others: specific DM 0x00:GCU	0xXX	Payload index: 4byte; Conf Data payload: 512 bytes at most Fixed size is 516, unused bytes filled by 0xFF	Configuration frame to program	0xF3	0x00: Success; Others: Fail	0xFF:all DMs; Others: specific DM; 0x00:GCU	0x00	None

6.3. ASCII command

To make the communication with the X-GCU more efficient, the X-LIB provides a method to warp the HEX command as an ASCII command. It is more convenient for the user to send the command as ASCII string format. The API function XCommand::SendAscCmd() provides the mechanism.

For the ASCII command format, all commands obey the following rules: The first character of the command body is '[', the last character is ']'; The first section is the key word written in capital letters; The second section is the operation mode, which is 'W'/write, 'R'/read, 'E'/execute, 'S'/save or 'L'/load; The third section is the DM index, which is '0'/None, 'FF'/all DMs or the specific number of the HEX format; The last section is data, which is in the HEX format. All sections are separated by the character ','.

The return command is also in the ASCII format. Normally, "[0]" or "[0, data]" is the successful return. Otherwise, some errors may occur.

For example, the command "[ST,W,0,3E8]" is to set the integration time to be 1000us, the successful return should be "[0]"; the command "[ST,R,0]" is to get the integration time, the successful return should be "[0,3E8]"

The following table shows the ASCII command list.

Table 6-3. ASCII command list

HEX	ASCII	Return	Description
0x10 0x03 0x00 0x00 none	[IN,S,0]	[0],[0,data]: success; [1]: EPCS(flash read/write) error; [2]: LVDS receive CRC error;	Saves current settings to the user flash.
0x10 0x04 0x00 0x00 none	[IN,L,0]		Loads settings from the user flash.
0x11 0x03 0x00 0x00 none	[IN1,S,0]		Savesdefault settings to the main and backup flash (not

HEX	ASCII	Return	Description
		[3]: Calibration data read/write error; [4]: X-GCU gets undefined command; [5]: X-GCU SN not match while setting MAC/IP; [6]: Timeout or no reply from X-DFE to X-GCU; [7]: X-GCU receives CRC error from PC; [8]: X-GCU receives parameter out of range; [9]: Reply timeout or no replay from X-GCU to PC; [10]: PC receives CRC error; [11]: PC received command code doesn't match the one sent	for the customer).
0x11 0x04 0x00 0x00 none	[IN1,L,0]		Loads default setting from the main flash.
0x20 0x01 0x00 0x04 data	[ST,W,0,data]		Sets integration time. The integration time is a 4-byte unsigned value.The unit is us. The range is defined by max and min integration times. Default is 0xBB8.
0x20 0x02 0x00 0x00 none	[ST,R,0]		Reads integration time.
0x21 0x01 0x00 0x02 data	[NT,W,0,data]		Sets Integration time in Non-continuous mode and in Constant mode. The non-continuous integration time is a 2-byte unsigned value. The unit is us. Default is 0x244.
0x21 0x02 0x00 0x00 none	[NT,R,0]		Reads Integration time in Non-continuous mode and in Constant mode.
0x22 0x01 0x00 0x01 data	[OM,W,0,data]		Sets operational mode. The operation mode is a 1-byte unsigned value. 0: Continuous mode; 1: Non-continuous mode 2: Constant integration mode; 3: Non-continuous with Hi/Lo trigger. Default is 0.
0x22 0x02 0x00 0x00 none	[OM,R,0]		Reads operational mode.
0x23 0x01 dmid 0x02 data	[SG,W,dmid,data]		Sets DM gain. The DM module gain value is a 2-byte unsigned value. The

HEX	ASCII	Return	Description
			high 8 bits are high energy gain, the low 8 bits are low energy gain. It is not available while scanning. Default is 0606. Valid range: LCS2: 1~3F; X-Card 0.2/0.4/0.8: 1~2; X-Card 1.5/1.6/2.5: 1~C; X-Card2 0.2/0.4: 1~2; X-Card2 0.8: 0~7; Aurora GT/B: 1~7F.
0x23 0x02 dmid 0x00 none	[SG,R,dmid]		Reads DM gain. The DM index cannot be 0xFF. It is not available while scanning.
0x24 0x01 0x00 0x01 data	[DC,W,0,data]		Sets hi/lo mode in single energy mode. In the single energy mode defines which DM module gain to launch. 1-byte unsigned value. 0: Low-energy gain; 1: High-energy gain. Default is 0. It is only used by LCS system.
0x24 0x02 0x00 0x00 none	[DC,R,0]		Reads hi/lo mode in single energy mode.
0x25 0x01 0x00 0x05 data	[CN,W,0,data]		Sets DM module number in each X-GCU's channel. It is a five-byte value. The first channel number is the highest byte and the fifth channel number is the lowest byte.
0x25 0x02 0x00 0x00 none	[CN,R,0]		Reads DM number of each channel.
0x27 0x01 0x00 0x01 data	[SF,W,0,data]		Starts/Stosp scanning. The scanning status is a 1-byte unsigned value. 0: Stopping; 1: Scanning.

HEX	ASCII	Return	Description
0x27 0x02 0x00 0x00 none	[SF,R,0]		Reads scanning status.
0x30 0x01 0x00 0x01 data	[EG,W,0,data]		Enables/disables gain correction. Gain correcting state, 1-byte unsigned value. Enables or disables channel gain correction. 0: Disable; 1: Enable. Default is 0.
0x30 0x02 0x00 0x00 none	[EG,R,0]		Read gain correction status.
0x31 0x01 0x00 0x01 data	[EO,W,0,data]		Enables/disables offset correction. Offset correcting state, 1-byte unsigned value. Enables or disables channel offset correction. 0: Disable; 1: Enable. Default is 0.
0x31 0x02 0x00 0x00 none	[EO,R,0]		Reads offset correction status.
0x32 0x01 0x00 0x01 data	[EB,W,0,data]		Enables/disables baseline correction. Baseline correcting state, 1-byte unsigned value. Enables or disables baseline correction. 0: Disable; 1: Enable. Default is 0.
0x32 0x02 0x00 0x00 none	[EB,R,0]		Reads baseline correction status.
0x33 0x04 0x00 0x01 data	[GC,L,0,data]		Loads channel gain from the specific section of the flash to FPGA. There are 4 groups of parameters.

HEX	ASCII	Return	Description
			Range is 0~3.
0x34 0x04 0x00 0x00 none	[OC,L,0]		Loads channel offset from flash to FPGA.
0x35 0x01 0x00 0x02 data	[BS,W,0,data]		Sest base line value. Base line value, 2-byte unsigned value. Default is 0.
0x35 0x02 0x00 0x00 none	[BS,R,0]		Reads base line value.
0x37 0x00 0x00 0x00 none	[RG,E,0]		Rests channel gain to 1.0.
0x38 0x00 0x00 0x00 none	[RO,E,0]		Resets channel offset to 0.
0x39 0x04 0x00 0x00 none	[PP,L,0]		Loads PDC position data.
0x3A 0x04 0x00 0x00 none	[PC,L,0]		Loads PDC coefficient data.
0x3B 0x01 0x00 0x01 data	[EP,W,0,data]		Enables/disables PDC correction. 0: Disable; 1: Enable.
0x3B 0x02 0x00 0x00 none	[EP,R,0]		Reads PDC correction status.
0x3C 0x01 0x00 0x01 data	[CP,W,0,data]		Sets correction pixel position.
0x3C 0x02 0x00 0x00 none	[CP,R,0]		Reads correction pixel position.
0x40 0x01 0x00 0x01 data	[DA,W,0,data]		Sets pixel binning mode. Pixel binning mode, 1-byte unsigned value. 0: Original pixel; 1: Averaging every 2 pixels; 2: Summing every 2 pixels; 3: Averaging every 4 pixels; 4: Summing every 4 pixels.

HEX	ASCII	Return	Description
0x40 0x02 0x00 0x00 none	[DA,R,0]		Reads pixel binning status.
0x41 0x01 0x00 0x01 data	[AF,W,0,data]		Sets averaging filter mode. Line averaging filter mode, 1-byte unsigned value. 0: None; 1: 2 lines; 2: 4 lines; 3: 8 lines; 4: 16 lines; 5: 32 lines; 6: 64 lines; 7: 128 lines; 8: 256 lines. Default is 0.
0x41 0x02 0x00 0x00 none	[AF,R,0]		Reads average filter status.
0x42 0x01 0x00 0x01 data	[SU,W,0,data]		Sets summing filter mode. Line summing filter mode, 1-byte unsigned value. 0: None; 1: 2 lines; 2: 4 lines; 3: 8 lines; 4: 16 lines; 5: 32 lines; 6: 64 lines; 7: 128 lines; 8: 256 lines. Default is 0.
0x42 0x02 0x00 0x00 none	[SU,R,0]		Reads summing filter status.

HEX	ASCII	Return	Description
0x43 0x01 0x00 0x01 data	[SB,W,0,data]		Sets output scale. Output scale, 1-byte unsigned value. The output pixel depth will be the original bit subtracting the scale value. 0: Original; n: Original – n. Default is 0. E.g.,the original bit is 16, the scale value is 8, the output pixel bits will be 8.
0x43 0x02 0x00 0x00 none	[SB,R,0]		Reads output scale.
0x44 0x01 0x00 0x01 data	[OA,W,0,data]		Sets offset averaging filter. Offset averaging filter, 1-byte unsigned value. 0: 1 line; 1: 2 lines; 2: 4 lines; 3: 8 lines. Default is 0. It's only available for LCS2 system.
0x44 0x02 0x00 0x00 none	[OA,R,0]		Reads offset averaging filter.
0x50 0x01 0x00 0x01 data	[LM,W,0,data]		Sets external line trigger mode. External line trigger mode, 1-byte unsigned value. 0: Rising edge effect; 1: Falling edge effect; 2: Sync trigger stamp mode; 3: Async trigger stamp mode. Default is 0.
0x50 0x02 0x00 0x00	[LM,R,0]		Reads external line trigger

HEX	ASCII	Return	Description
none			mode.
0x51 0x01 0x00 0x01 data	[EL,W,0,data]		Enables/disables external line trigger. 0: Disable; 1: Enable. Default is 0.
0x51 0x02 0x00 0x00 none	[EL,R,0]		Reads external line trigger enabled.
0x52 0x01 0x00 0x01 data	[TD,W,0,data]		Sets line trigger fine delay. External line trigger fine delay, 1-byte unsigned value. The unit is 0.125us. Default is 0.
0x52 0x02 0x00 0x00 none	[TD,R,0]		Reads line trigger fine delay.
0x53 0x01 0x00 0x01 data	[RD,W,0,data]		Sets line trigger raw delay. External line trigger raw delay, 1-byte unsigned value. The unit is 64us.
0x53 0x02 0x00 0x00 none	[RD,R,0]		Reads line trigger raw delay
0x54 0x01 0x00 0x01 data	[FM,W,0,data]		Sets external frame trigger mode. 0: Rising edge; 1: Falling edge. Default is 0.
0x54 0x02 0x00 0x00 none	[FM,R,0]		Reads external frame trigger mode.
0x55 0x01 0x00 0x02 data	[EF,W,0,data]		Enables/disables external frame trigger. Enables external frame trigger, 2-byte unsigned value. 0: Disable; n: Enable, and output 32*n lines by each trigger. Default is 0.

HEX	ASCII	Return	Description
0x55 0x02 0x00 0x00 none	[EF,R,0]		Reads external frame trigger enabled.
0x56 0x01 0x00 0x01 data	[FD,W,0,data]		Sets external frame trigger delay. External frame trigger raw delay, 1-byte unsigned value. The unit is 32 lines. Default is 0.
0x56 0x02 0x00 0x00 none	[FD,R,0]		Reads external frame trigger delay.
0x57 0x00 0x00 0x00 none	[SE,E,0]		Sends a simulated external frame trigger.
0x58 0x01 0x00 0x01 data	[LE,W,0,data]		Sets Hi/Lo line trigger with rising/falling edge detection. Hi/Lo line trigger with rising/falling edge detection, 1-byte unsigned value. 0: Rising edge; 1: Falling edge. Default is 0.
0x58 0x02 0x00 0x00 none	[LE,R,0]		Reads Hi/Lo line trigger with hi/lo edge level detection.
0x59 0x01 0x00 0x02 data	[SP,W,0,data]		Sets Hi/Lo line trigger sampling position. Hi/Lo line trigger sampling position, 1-byte unsigned value. The unit is 1us. Range is 0-255, default is 0.
0x59 0x02 0x00 0x00 none	[SP,R,0]		Reads Hi/Lo line trigger sampling position.
0x5A 0x01 0x00 0x01 data	[TC,W,0,data]		Sets trigger stamp parity check mode. 0: Disable parity check; 1: Use “odd” parity check;

HEX	ASCII	Return	Description
			2: Use “even” parity check
0x5A 0x02 0x00 0x00 none	[TC,R,0]		Reads trigger stamp parity check mode.
0x60 0x01 0x00 0x01 data	[TP,W,0,data]		Sets heartbeat period. 1-byte unsigned value. The unit is s.
0x60 0x02 0x00 0x00 none	[TP,R,0]		Reads heartbeat period.
0x61 0x01 0x00 0x01 data	[GT,W,0,data]		Sets X-GCU type. 1-byte unsigned value. 0: Gigabit Ethernet; 1: Camera Link. Default is 0.
0x61 0x02 0x00 0x00 none	[GT,R,0]		Reads X-GCU type.
0x62 0x02 0x00 0x00 none	[GS,R,0]		Reads X-GCU serial number.
0x63 0x02 dmid 0x00 none	[DS,R,dmid]		Reads DM serial number. It is not available while scanning.
0x64 0x02 0x00 0x00 none	[PN,R,0]		Reads pixel number. Total pixel number of all DMs, 2-byte unsigned value.
0x65 0x02 0x00 0x00 none	[PS,R,0]		Reads pixel size.
0x66 0x02 0x00 0x00 none	[PD,R,0]		Reads pixel depth.
0x67 0x01 0x00 0x08 data	[IR,W,0,data]		Sets integration time range. Max and min integration time, 8-byte unsigned value. The high four bytes are the max value, the low four bytes are the min value.
0x67 0x02 0x00 0x00 none	[IR,R,0]		Reads integration time range.
0x68 0x02 0x00 0x00 none	[GF,R,0]		Reads X-GCU firmware version.

HEX	ASCII	Return	Description
0x69 0x02 dmid 0x00 none	[DF,R,dmid]		Reads DM firmware version. It is not available while scanning.
0x6A 0x01 0x00 0x01 data	[ED,W,0,data]		Sets test pattern. X-GCU's test pattern mode, 1-byte unsigned value. 0: Disable; 1: Normal test pattern; 2: Slope test pattern; Default is 0.
0x6A 0x02 0x00 0x00 none	[ED,R,0]		Reads test pattern status.
0x6B 0x01 dmid 0x01 data	[DT,W,dmid,data]		Sets DM test mode. It is not available while scanning. DM's test pattern mode, 1-byte unsigned value. 0: Normal image data mode; 1: Enable DM board fixed value (AAAAA[Hex]) test mode; 2: Enable I'm here function. Light the LED of the specific DM. Make sure the LED must enabled by [LC] command. Default is 0.
0x6B 0x02 dmid 0x00 none	[DT,R,dmid]		Reads DM test mode status. It is not available while scanning.
0x6C 0x02 0x00 0x00 none	[DP,R,0]		Reads pixel number per DM.
0x6D 0x02 0x00 0x00 none	[DN,R,0]		Reads card number per X-DFE.
0x6E 0x02 0x00 0x00 none	[CT,R,0]		Reads card type.
0x72 0x02 0x00 0x00	[GI,R,0]		Reads X-GCU's voltage,

HEX	ASCII	Return	Description
none			temperature and humidity.
0x73 0x02 dmid 0x00 none	[DI,R,dmid]		Reads DM's voltage, temperature and humidity. It is not available while scanning.
0x75 0x01 0x00 0x01 data	[LC,W,0,data]		Enables/disables LED. LED state, 1-byte unsigned value. 0: Disable; 1: Enable. Default is 1.
0x75 0x02 0x00 0x00 none	[LC,R,0]		Reads LED state.
0x76 0x01 0x00 0x01 data	[TR,W,0,data]		X-GCU frame and line trigger RS485 bus terminator resistor, used only in multi-X-GCU application. 0: Disable; 1: Enable. Default is 0.
0x76 0x02 0x00 0x00 none	[TR,R,0]		Reads terminal resistor state.
0x7B 0x02 0x00 0x00 none	[EM,R,0]		Reads energy mode.
0x7E 0x01 0x00 0x01 data	[MT,W,0,data]		Sets MTU size. The size of max UDP packet. If to use the big size, please set the Ethernet adapter jumbo packet as 9014 bytes. 0:1500 bytes; 1:8192 bytes. Default is 0.
0x7E 0x02 0x00 0x00 none	[MT,R,0]		Reads MTU size.
0x80 0x01 0x00 0x01 data	[RL,W,0,data]		Enable/disable Ethernet LED. Ethernet LED state, 1-byte unsigned value.

HEX	ASCII	Return	Description
			0: Disable; 1: Enable. Default is 1. Only applicable to Aurora GT/B.
0x80 0x02 0x00 0x00 none	[RL,R,0]		Reads Ethernet LED state. Only applicable to Aurora GT/B.
0x8A 0x01 0x00 0x01 data	[SW,W,0,data]		Sets swap mode of low and high part. 0: Not swap; 1: Swap.
0x8A 0x02 0x00 0x00 none	[SW,R,0]		Reads swap mode of low and high part.
0x8B 0x01 0x00 0x01 data	[ZG,W,0,data]		Sets Z gap, range is 0~0xFF, step is 0.1mm.
0x8B 0x02 0x00 0x00 none	[ZG,R,0]		Reads Z gap.
0x8C 0x01 0x00 0x04 data	[PT,W,0,data]		Sets system working time counter.
0x8C 0x02 0x00 0x00 none	[PT,R,0]		Reads system working time counter, which is increased by one step per 2 hours.
0x8E 0x02 0x00 0x00 none	[WT,R,0]		Reads X-GCU's working time after each powering up.
0x8F 0x01 0x00 0x02 data	[OD,W,0,data]		Sets offset delay value. Unit is us.
0x8F 0x02 0x00 0x00 none	[OD,R,0]		Reads offset delay.
0x90 0x01 0x00 0x01 data	[OE,W,0,data]		Enable/disable offset delay setting. 0: Disable; 1: Enable.
0x90 0x02 0x00 0x00 none	[OE,R,0]		Reads offset delay setting.
0x91 0x01 0x00 0x01 data	[CF,W,0,data]		Sets camera link data format in dual energy

HEX	ASCII	Return	Description
			mode. 0: normal mode(low energy part preceding with high energy part); 1: interweave mode(low energy data interweaved with high energy data)
0x91 0x02 0x00 0x00 none	[CF,R,0]		Reads camera link data format in dual energy mode.
0x92 0x01 0x00 0x01 data	[BR,W,0,data]		Sets serial port communication baud rate. 0: 115200; 1: 9600.
0x92 0x02 0x00 0x00 none	[BR,R,0]		Reads serial port baud rate.
0x94 0x00 dmid 0x00 none	[RA,E,dmid]		Resets AD. Only applicable to Aurora GT.
0x95 0x00 dmid 0x00 none	[RC,E,dmid]		Reconfigures firmware of X-GCU or X-DFE. Only applicable to Aurora GT.
0x96 0x01 dmid 0x01 data	[AV,W,dmid,data]		Sets AD Vref. 0: Disable; 1: Enable. Only applicable to Aurora GT.
0x96 0x02 dmid 0x00 none	[AV,R,dmid]		Reads AD Vref. Only applicable to Aurora GT.
0x97 0x01 dmid 0x01 data	[PC,W,dmid,data]		Sets PD connection. 0: Disable; 1: Enable. Only applicable to Aurora GT.
0x97 0x02 dmid 0x00 none	[PC,R,dmid]		Reads PD connection status. Only applicable to Aurora GT.

HEX	ASCII	Return	Description
0x98 0x01 dmid 0x01 data	[AS,W,dmid,data]		Sets AD speed mode. 0: Low speed; 1: High speed. Only applicable to Aurora GT.
0x98 0x02 dmid 0x00 none	[AS,R,dmid]		Reads AD speed mode. Only applicable to Aurora GT.

7. CAMERA LINK SUPPORT

The X-GCU also supports the Camera Link transmission protocol. In that case, the command channel adopts the RS232 as a transmission protocol and the image channel adopts the Camera Link protocol. The frame grabber card "X64 Xcelera-CL PX4 Dual" is adopted to cooperate with the X-GCU, which provides two independent interfaces. The COM port is mapped by the frame grabber card, so a serial port cable is not needed (Please, refer to the page 22, X64 Xcelera-CL PX4 User's Manual).

7.1. Command and image format

The RS232 command protocol adopts the same communication mechanism and packet format as the UDP command protocol does. All the commands listed in the HEX command table of the chapter 6 are supported.

The Camera Link Image channel adopts the communication mechanism as the Figure 2-8 shows. The format of the line data should be as follows:

Start code	0xBCBCBCBC
LINE ID	Line index number
LINE STAMP	Time stamp or encoder count
ENERGY FLAG	High or low energy flag: low/0x00, high/0x01 (For LCS2)
DM PACKET NUM	Total DM packet number

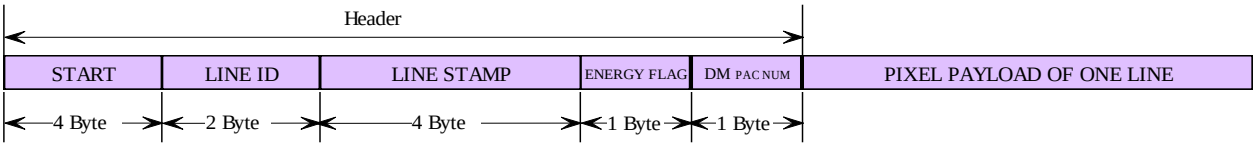


Figure 7-1. Line data format of camera link

7.2. Using X-LIB to control camera link

The user can utilize the Spapera LT SDK (C++ or .NET) to develop the Camera Link application or use the X-LIB (C++) as middleware. The X-LIB (C++) leaves some interfaces for the Camera Link. The user can follow the interfaces to wrap the Sapera LT SDK and insert the wrapper into the X-LIB and then use the API of the X-LIB to control the camera link just as gigabit Ethernet.

The X-LIB C++ demo code gives a completed Camera Link wrapper; the user can utilize it directly or modify it for special use.

For more details of how to operate the Camera Link frame grabber, please refer to the Sapera LT User's Manual and the X64 Xcelera-CL PX4 User's Manual.

8. CORRECTION PROCESS

8.1. On-board Correction Process

Steps	Description	Functions	Commands
1	Disables on-board correction.	XCommand::SetPara(XPARA_EN_GAIN_CORRECT, 0); XCommand::SetPara(XPARA_EN_OFFSET_CORRECT, 0); XCommand::SetPara(XPARA_EN_BASELINE_CORRECT, 0);	0x30/[EG,W,0,0] 0x31/[EO,W,0,0] 0x32/[EB,W,0,0]
2	Resets gain and offset table.	XOffCorrect::Reset(); XCommand::ExecutePara(XPARA_RESET_GAIN); XCommand::ExecutePara(XPARA_RESET_OFFSET);	0x37/[RG,E,0] 0x38/[RO,E,0]
3	Powers off x-ray.	N/A	N/A
4	Calculates offset values.	XOffCorrect::Calculate(1, …);	N/A
5	Powers on x-ray.	N/A	N/A
6	Calculates gain values.	XOffCorrect::Calculate(0, …);	N/A
7	Saves gain table and offset table into X-GCU board.	XOffCorrect::SaveFlash(…);	0x33 0x34
8	Enables gain and offset correction.	XCommand::SetPara(XPARA_EN_GAIN_CORRECT, 1); XCommand::SetPara(XPARA_EN_OFFSET_CORRECT, 1);	0x30/[EG,W,0,1] 0x31/[EO,W,0,1]
9	Sets and enables baseline correction.	XCommand::SetPara(XPARA_BASE_LINE, …); XCommand::SetPara(XPARA_EN_BASE_LINE_CORRECT, 1);	0x35/[BS,W,0,…] 0x32/[EB,W,0,1]
10	Saves current state into X-GCU board.	XCommand::ExecutePara(XPARA_SAVE_PARA);	0x10/[IN,S,0]
11	After finishing the process, loads gain and offset table while restarting the system. X-GCU board applies the correcting tables saved before to do correction in real time.	XCommand::ExecutePara(XPARA_LOAD_GAIN, …); XCommand::ExecutePara(XPARA_LOAD_OFFSET);	0x33/[GC,L,0,…] 0x34/[OC,L,0]

8.2. Off-board Correction Process

Steps	Description	Functions	Commands
1	Disables off-board correction.	Does not call XOffCorrect::DoCorrect(…) function in frame ready call back.	N/A
2	Resets gain and offset table.	XOffCorrect::Reset();	N/A
3	Powers off x-ray.	N/A	N/A
4	Calculates offset values.	XOffCorrect::Calculate(1, …);	N/A
5	Powers on x-ray.	N/A	N/A
6	Calculates gain values.	XOffCorrect::Calculate(0, …);	N/A
7	Saves gain table and offset table into a file.	XOffCorrect::SaveFile(…);	N/A
8	Enables off-board gain and offset correction.	Calls XOffCorrect::DoCorrect(…) function in frame ready call back.	N/A
9	After finishing the process, loads gain and offset table from the saved file while restarting the system.	XOffCorrect::LoadFile(…); Calls XOffCorrect::DoCorrect(…) function in frame ready call back.	N/A

8.3. Off-board Correction Process in Non-continuous hi/lo Trigger Mode

Steps	Description	Functions	Commands
1	Disables off-board correction.	Does not call XOffHLCorrect::DoCorrect(…) function in frame ready call back.	N/A
2	Resets gain table.	XOffHLCorrect::Reset();	N/A
3	Enable on-line offset averaging filter.	XCommand::SetPara(XPARA_OFFSET_AVERAGE, …);	0x41/[AF,W,0,…]
4	Powers on x-ray.	N/A	N/A
5	Calculates gain values.	XOffHLCorrect::Calculate(0, …);	N/A
6	Saves gain table into a file.	XOffHLCorrect::SaveFile(…);	N/A
7	Enables off-board gain correction.	Calls XOffHLCorrect::DoCorrect(…) function in frame ready call back.	N/A
8	After finishing the process, loads gain from the saved file while restarting the system. Enables on-line offset	XOffHLCorrect::LoadFile(…); XCommand::SetPara(XPARA_OFFSET_AVERAGE, …); Calls XOffHLCorrect::DoCorrect(…)	0x41/[AF,W,0,…]

Steps	Description	Functions	Commands
	averaging filter	function in frame ready call back.	

8.4. Off-board Piecewise Correction Process

Steps	Description	Functions	Commands
1	Disables off-board piecewise correction.	Does not call XPiecewiseCorrect::DoCorrect(…) function in frame ready call back.	N/A
2	Powers off x-ray.	N/A	N/A
3	Calculates offset values.	XPiecewiseCorrect::CalculateOffset1(…);	N/A
4	Powers on x-ray to the low intensity.	N/A	N/A
5	Calculates the first group of gain values.	XPiecewiseCorrect::CalculateGain1(…);	N/A
6	Adjust x-ray to the middle intensity.	N/A	N/A
7	Calculate the second group of gain values.	XPiecewiseCorrect::CalculatePara2(…);	N/A
8	Adjust x-ray to the high intensity.	N/A	N/A
9	Calculate the third group of gain values.	XPiecewiseCorrect::CalculatePara3(…);	N/A
10	Saves gain tables and offset table into a file.	XPiecewiseCorrect::SaveFile(…);	N/A
11	Enables piecewise correction.	Calls XPiecewiseCorrect::DoCorrect(…) function in frame ready call back.	N/A
12	After finishing the process, loads gain and offset table from the saved file while restarting the system.	XPiecewiseCorrect::LoadFile(…); Calls XPiecewiseCorrect::DoCorrect(…) function in frame ready call back.	N/A