TOPPAN ToF *senSPure*™ SDK API Reference Manual

TOPPAN 3D ToF Camera

TOPPAN Holdings Inc.

Revision 1.12

September 9, 2025

Contents

1. Overview

1-1. Purpose of this document

This document describes the application program interface (API) specifications for TOPPAN 3D ToF cameras using the TOPPAN ToF *senSPure*™ SDK.
"*senSPure*™ SDK" and "TOPPAN ToF SDK" both refer to the same dedicated SDK.

The following cameras are currently supported for operation.

Table 1. Supported camera model

Model	Product code	Camera firmware version
C11U	TPSC1AS1Z	3.1.0

1-2. Definitions of terms and abbreviations

Table 2. Definitions of terms and abbreviations

Term	Definition
SDK	Software Development Kit
ToF	Time of Flight Determination of the distance between the camera and subject from the Time of Flight (ToF) using light pulse.

1-3. Related document

The following documents should be consulted in combination when referring to this manual.

Table 3. Related document

Related document	Description
TOPPAN ToF *senSPure*™ SDK Development Environment Setup Guide	Building the TOPPAN ToF SDK software development environment.
TOPPAN ToF *senSPure*™ SDK PostFilter Library Reference Guide	Post filter library designed to work in co-operation with the TOPPAN ToF SDK.

1-4. SDK Structure

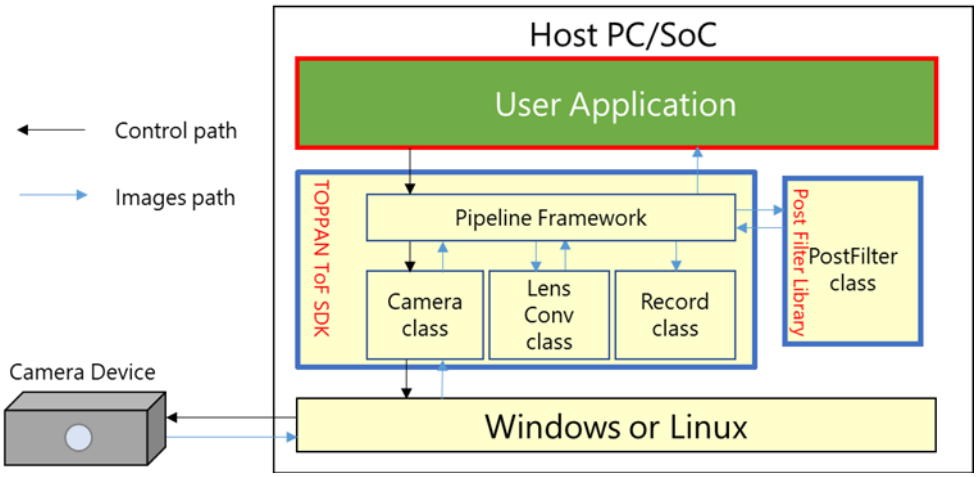


Figure 1. TOPPAN ToF SDK (w/ PostFilter) software structure diagram

Table 4. Description of each block

Block		Contents
User Application		Application that controls this SDK and/or a sample application provided in this SDK
TOPPAN ToF SDK (Library)		Libraries provided as this SDK
	PipelineFramework	A framework that manages the Camera class, Record class, and LensConv class together, and allows you to build a Pipeline process that includes other arbitrary processing (OSS-linked processing or user-specific processing).
	Camera class	C++ class for controlling camera devices and/or playing back stored files
	LensConv class	C++ class for lens-related conversion of ToF camera output images
	Record class	C++ class for saving camera device output images to a file
PostFilter (Library)		Post filter libraries designed to work in co-operation with this SDK. C++ class that provides image filtering function

1-5. Operating Environment

The following describes the operating environment of this SDK.

Table 5. Operating environment

Name		OS type	Version
PC		Windows	Windows 10 64bit Windows 11 64bit
		Linux	Ubuntu 20.04LTS 64bit Ubuntu 22.04LTS 64bit
SoC	NVIDIA Jetson AGX Orin	Linux	JetPack 5.0.1 (Ubuntu 20.04LTS 64bit) JetPack 6.0 (Ubuntu 22.04LTS 64bit)

1-6. Recommended environment for host computer

The recommended environment for the host computer running this SDK is as follows:

Table 6. Recommended environment (Performances)

Name	Recommended performance
CPU	64-bit, 4-core 2GHz or higher
RAM	8GB or more
Interface	USB3.1 (Gen1) port

Note: If the Pipeline Framework requires multi-threaded configuration and high frame rate processing, a host PC with higher specifications in terms of CPU core count and frequency will be needed.

1-6-1. Programming language

This SDK is developed in C++ (C++17 standard). However, MISRA-C++ and CERT-C++ are not supported.

2. TOPPAN ToF SDK API Usage

2-1. Provided files

The following files are stored in the include directory generated after the build. For the configuration of the provided directory, refer to "TOPPAN ToF SDK Development Environment Setup Guide" described in "***1-3. Related document***".

Table 7. Provided files

File type	File name	
Header files	Common definition	TpTofSdkDefine.h
	PlFw class	PlFw.h
	EvtThread class	EvtThread.h
	RecordThread class	RecordThread.h
	LensConvThread class	LensConvThread.h
	PostFilterThread Class	PostFilterThread.h
	Camera class	Camera.h
	Record class	Record.h
	LensConv class	LensConv.h
	PostFilter Class	PostFilter.h
Library file	Windows	TpTofSdk.dll PostFilter.dll
	Linux	libTpTofSdk.so libPostFilter.so

2-2. Related library files

This SDK library refers to the following library files. User programs that use this SDK library need to link to the above library files.

Table 8. Related libraries

OS Type	Related library
Windows	Windows SDK (DirectShow, WinUSB)
Linux	libusb
(Common)	OpenCV, boost

EWCLIB (http://insubaru.g1.xrea.com/ewclib/) is included and used in the Windows version.

3. API provided by TOPPAN ToF SDK

3-1. List of classes

The following is a list of classes provided as APIs by TOPPAN ToF SDK.

Table 9. List of classes

Class name	Description
PipelineFramework	A framework that manages the Camera class, the Record class, and the LensConv class together, and can construct Pipeline processing including arbitrary processing other than these (OSS-linked processing and user-specific processing). For the classes related to PipelineFramework, see "***3-3-3. PipelineFramework internal classes***".
Camera class	C ++ class for controlling camera devices or a C++ class to play a saved file *This class is not allowed when using PipelineFramework.
Record class	C++ class to save an output image from a camera device to a file *This class is not allowed when using PipelineFramework.
LensConv class	C++ class that performs lens-related conversion processing on the output image of the camera device *Use of this class is prohibited when using PipelineFramework.

3-2. Common definition

The definitions commonly used by this SDK, which are described in TpTofSdkDefine.h, are shown below.

3-2-1. Constant definition

3-2-1-1. List of constant definitions

Table 10. Constant definitions

Name	Description
SDK_VERSION	SDK Version
INVALID_DEPTH	Invalid distance image data (far)
SATURATION_DEPTH	Invalid distance image data (near saturation)

3-2-1-2. Constant definition details

3-2-1-2-1. SDK_VERSION

Table 11. SDK_VERSION definition

Definition	const Version SDK_VERSION;
Description	・ Indicates the version information of this SDK.
Reference	3-2-3-2-1. Version

3-2-1-2-2. INVALID_DEPTH

Table 12. INVALID_DEPTH definition

Definition	const uint16_t INVALID_DEPTH = 0xFFFFU;
Description	· Indicates an invalid value (far) for distance image data.
Reference	3-2-2-2-2. ImageKind, 3-2-3-2-9. ImageData

3-2-1-2-3. SATURATION_DEPTH

Table 13. SATURATION_DEPTH definition

Definition	const uint16_t SATURATION_DEPTH = 0x0000U;
Description	· Invalid value (near saturation) for distance image data.
Reference	3-2-2-2-2. ImageKind, 3-2-3-2-9. ImageData

3-2-2. Enumeration definition

3-2-2-1. List of enumeration definitions

Table 14. Enumeration definitions

Name	Description
Result	Return value
ImageKind	Type of image data
PcdKind	Point cloud data type

3-2-2-2. Enumeration definition details

3-2-2-2-1. Result

Table 15. Result definition

Definition	enum Result {
	SUCCESS = 0,
	CANCELED,
	REACH_EOF,
	ERR_INVALID_PTR,
	ERR_OVER_RANGE,
	ERR_BAD_ARG,
	ERR_BAD_STATE,
	ERR_NOT_EXIST,
	ERR_TIMEOUT,
	ERR_EMPTY,
	ERR_FULL,
	ERR_NOT_SUPPORT,
	ERR_SYSTEM
	};		

Description	· Indicates the return value of each API.		
Value	**Name**	**Value**	**Remark**
	SUCCESS	0	Success
	CANCELED	1	Wait status was released.
	REACH_EOF	2	End-of-file reached

	ERR_INVALID_PTR	3	Argument pointer is invalid.
	ERR_OVER_RANGE	4	The set value is outside the specifiable range.
	ERR_BAD_ARG	5	Other invalid argument
	ERR_BAD_STATE	6	Status transition error
	ERR_NOT_EXIST	7	The device file, etc. does not exist.
	ERR_TIMEOUT	8	Timeout occurred while waiting for processing.
	ERR_EMPTY	9	Empty buffer, etc.
	ERR_FULL	10	Not enough space, etc.
	ERR_NOT_SUPPORT	11	Unsupported function
	ERR_SYSTEM	12	System error (such as memory allocation failure)
Reference	All APIs		

3-2-2-2-2. ImageKind

Table 16. ImageKind definition

Definition	enum ImageKind : uint8_t {
	IMG_DEPTH = 0,
	IMG_IR,
	IMG_RAW1,
	IMG_RAW2,
	IMG_RAW3,
	IMG_RAW4,
	IMG_KINDS
	};		

Description	・ Indicates the type of image data.		
Value	**Name**	**Value**	**Remark**
	IMG_DEPTH	0	Distance image data
	IMG_IR	1	IR image data
	IMG_RAW1	2	Sensor RAW G1 image data
	IMG_RAW2	3	Sensor RAW G2 image data
	IMG_RAW3	4	Sensor RAW G3 image data
	IMG_RAW4	5	Sensor RAW G4 image data
	IMG_KINDS	6	Number of image types
Reference	3-2-3-2-4. ImageFormat, 3-2-4-2-2. Images		

3-2-2-2-3. PcdKind

Table 17. PcdKind definition

Definition	enum PcdKind {
	PCD_XYZ = 0,
	PCD_RGBXYZ,
	PCD_IRXYZ,
	};		

Description	・ Indicates the type of point cloud data.		
Value	**Name**	**Value**	**Remark**
	PCD_XYZ	0	No color information Point cloud

	PCD_RGBXYZ	1	RGB Point cloud
	PCD_IRXYZ	2	IR point cloud
Reference	3-2-3-2-10. PcdData		

3-2-3. Structure definition

3-2-3-1. List of structure definitions

Table 18. Structure definitions

Name	Description
Version	Version information
Point2d	Pixel position information on the image plane
Point3d	Coordinate information in three-dimensional space
ImageFormat	Image data format information
Range	Distance range information
FrameError	Received frame error information
ConvState	Received frame conversion status information
FrameInfo	Received frame additional information
ImageData	Image data
PcdData	Point cloud data
Frame	Received frame information

3-2-3-2. Structure definition details

3-2-3-2-1. Version

Table 19. Version definition

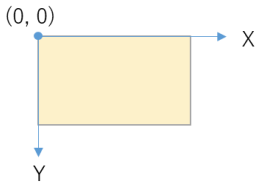
Definition	struct Version {
	uint8_t major;
	uint8_t minor;
	uint16_t rev;
	};		

Description	・ Indicates version information.		
Arguments	**Type**	**Name**	**Remark**
	uint8_t	major	Major version
	uint8_t	minor	Minor version
	uint16_t	rev	Revision
Reference	3-2-3-2-1. Version, 3-4-8-2-2-4. DeviceInfo		

3-2-3-2-2. Point2d

Table 20. Point2d definition

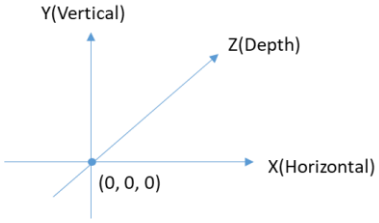
Definition	struct Point2d {
	uint16_t x;
	uint16_t y;
	};

Description	• Indicates the position of a pixel on the image plane. • The pixel position is taken with the upper left as the origin.  		
Arguments	**Type**	**Name**	**Remark**
	uint16_t	x	X coordinate [pixel]
	uint16_t	y	Y coordinate [pixel]
Reference	3-4-8-2-2-6. LensInfo		

3-2-3-2-3. Point3d

Table 21. Point3d definition

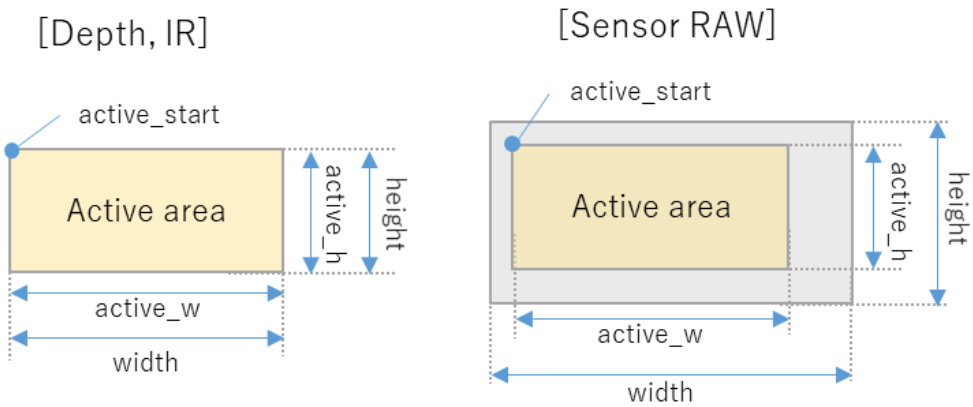
Definition	struct Point3d {
	uint32_t color;
	float x;
	float y;
	float z;
	};		

Description	• Indicates coordinate information in three-dimensional space.  • For RGB or IR point clouds, color indicates color information.		
Arguments	**Type**	**Name**	**Remark**
	uint32_t	color	For UINT32_MAX, it indicates the invalid point. For RGB point clouds, it indicates the value of Blue, Green, and Red for every 1byte from the highest byte. Blue, Green, and Red values. For IR point clouds, it indicates the 16bit IR value.
	float	x	X-axis coordinate [mm]
	float	y	Y-axis coordinate [mm]
	float	z	Z-axis coordinate [mm]
Reference	3-2-3-2-10. PcdData		

3-2-3-2-4. ImageFormat

Table 22. ImageFormat definition

Definition	struct ImageFormat { uint16_t width; uint16_t height; Point2d active_start; uint16_t active_w; uint16_t active_h; uint32_t pixels;

	uint8_t bpp; size_t size; };
Description	・ Indicates the format information of image data. ・ As output from the Camera class, all pixels specified by width and height are output. If the image type is sensor RAW (IMG_RAW1, IMG_RAW2, IMG_RAW3, IMG_RAW4), the area specified by the effective pixel information (active_start, active_w, active_h) is recognized as the effective pixel area. If the image type is not sensor RAW, active_start is always set to (0,0) and active_w and active_h are the same as width and height respectively. In other words, all output pixels are effective pixels. (Reference: 3-4-2-1. Device control functions) 

Arguments	Type	Name	Remark
	uint16_t	width	Image Data Width [pixel]
	uint16_t	height	Image Data Height [pixel]
	Point2d	active_start	Effective pixel start position [pixel]
	uint16_t	active_w	Effective pixel width [pixel]
	uint16_t	active_h	Effective pixel height [pixel]
	uint32_t	pixels	Image Data Pixels (Width × Height) [pixel]
	uint8_t	bpp	Image data 1 pixel size [byte]
	size_t	size	Image data size [byte]
Reference	3-2-4-2-1. ImageFormats		

3-2-3-2-4-1.ImageFormat::set

Table 23. ImageFormat::set method

Function	Set image data format information			
Syntax	inline void set (uint16_t w = 0, uint16_t h = 0, uint8_t b = 0);			
Description	・ Sets the image format information. ・ If this argument is omitted, the format information is initialized.			
Arguments	Type	Name	in/out	Remark
	uint16_t	w	in	Image data width [pixel]

	uint16_t	h	in	Image data height [pixel]
	uint8_t	b	in	Image data 1 pixel size [byte]
Return value	None			
Sync/Async	Synchronous			

3-2-3-2-4-2.ImageFormat::isExist

Table 24. ImageFormat::isExist method

Function	Check for the existence of an image			
Syntax	inline bool isExist (void);			
Description	・ Retrieves whether the image format information exists.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	None			
Return value	true	Exists		
	false	Not exists		
Sync/Async	Synchronous			

3-2-3-2-5. Range

Table 25. Range definition

Definition	struct Range {
	uint16_t min;
	uint16_t max;
	};		

Description	・ Indicates distance range information.		
Arguments	**Type**	**Name**	**Description**
	uint16_t	min	Min. distance [mm]
	uint16_t	max	Max. distance [mm]
Reference	3-4-8-2-2-8. ModeInfo		

3-2-3-2-6. FrameError

Table 26. FrameError definition

Definition	struct FrameError {
	uint16_t drop : 1;
	uint16_t crc : 1;
	uint16_t reserved : 14;
	};			

Description	・ Indicates received frame error information. ・ Non-0 is set in each bit if an error is found.			
Arguments	**Type**		**Name**	**Remark**
	uint16_t	1bit	drop	Frame discontinuity information (0: Continuous, 1: Discontinuous)
	uint16_t	1bit	crc	CRC Error information (0: No error, 1: Error)
	uint16_t	14bit	reserved	reserved for future
Reference	3-2-3-2-8. FrameInfo			

3-2-3-2-7. ConvState

Table 27. ConvState definition

Definition	struct ConvState {
	uint8_t is_crct_dist : 1;
	uint8_t is_filt_med : 1;
	uint8_t is_filt_bil : 1;
	uint8_t is_filt_fly_p : 1;
	uint8_t reserved : 4;
	};			

Description	• Indicates the conversion status of the received frame. • is_crct_dist is 1 for distortion corrected data. • is_crct_dist is 0 except for depth, IR image, and point cloud data. • For data to which the median filter of PostFilter is applied, is_filt_med is 1. • is_filt_med is 0 except for Depth, IR image and point cloud data. • For data to which the bilateral filter of PostFilter is applied, is_filt_bil is 1. • is_filt_bil is 0 except for Depth, IR image and point cloud data. • For data to which the flying pixel filter of PostFilter is applied, is_filt_fly_p is 1. • is_filt_fly_p is 0 except for Depth image and point cloud data.			
Arguments	**Type**		**Name**	**Remark**
	uint8_t	1bit	is_crct_dist	Whether distortion has been corrected
	uint8_t	1bit	is_filt_med	Whether median filter has been applied
	uint8_t	1bit	is_filt_bil	Bilateral filter applied
	uint8_t	1bit	is_filt_fly_p	Flying pixel filter applied
	uint8_t	4bit	reserved	reserved for future
Reference	3-2-3-2-8. FrameInfo			

3-2-3-2-8. FrameInfo

Table 28. FrameInfo definition

Definition	struct FrameInfo {
	uint32_t number;
	timespec time;
	FrameError frm_err;
	int16_t temperature;
	uint32_t light_cnt;
	ConvState conv_stat;
	};		

Description	• Indicates additional information about the received frame. • If the camera device does not add time stamp information, the information of the time when the frame is received by this SDK is added. • If the temperature information in the camera device cannot be acquired, the value of temperature is UINT16_MAX.		
Arguments	**Type**	**Name**	**Remark**
	uint32_t	number	Frame Number
	timespec	time	Timestamp information
	FrameError	frm_err	Frame error information
	uint16_t	temperature	Camera device temperature information (Fixed point: Integer 10bit, Fractional 6bit)

			(UINT16_MAX: Invalid temperature value)
	uint32_t	light_cnt	Number of "light times" value
	ConvState	conv_stat	Conversion status information
Reference	3-2-3-2-6. FrameError, 3-2-3-2-7. ConvState, 3-2-3-2-9. ImageData, 3-2-3-2-10. PcdData		

3-2-3-2-9. ImageData

Table 29. ImageData definition

Definition	struct ImageData {
	FrameInfo info;
	std::vector<uint16_t> data;
	};		

Description	• Indicates image data of the following image types. • Depth(IMG_DEPTH) • IR(IMG_IR) • Sensor RAW (IMG_RAW1, IMG_RAW2, IMG_RAW3, IMG_RAW4) • Image data format information is defined by ImageFormat. • For Depth, invalid pixels are set to INVALID_DEPTH or SATURATION_DEPTH. • For Depth, the value is in mm.		
Arguments	**Type**	**Name**	**Remark**
	FrameInfo	info	Frame additional information
	std::vector<uint16_t>	data	Image data
Reference	3-2-1-2-2. INVALID_DEPTH, 3-2-1-2-3. SATURATION_DEPTH, 3-2-2-2-2. ImageKind, 3-2-3-2-4. ImageFormat, 3-2-3-2-8. FrameInfo, 3-2-4-2-2. Images, 3-6-5-6. LensConv::correctDist, 3-6-5-7. LensConv::convPcdCamera, 3-6-5-8. LensConv::convPcdWorld		

3-2-3-2-9-1.ImageData::resize

Table 30. ImageData::resize method

Function	Resize Image Data			
Syntax	inline void resize (const ImageFormat& format);			
Description	• Resizes the image data based on the image format information.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	const ImageFormat&	format	in	Image format
Return value	None			
Sync/Async	Synchronous			

3-2-3-2-10. PcdData

Table 31. PcdData definition

Definition	struct PcdData {
	FrameInfo info;
	PcdKind kind;
	std::vector<Point3d> data;
	};

Description	· Indicates point cloud data. · The maximum value of the number of point cloud data is the same number of pixels as Depth. However, the actual number of point clouds included in the data should be acquired from data.size().		
Arguments	**Type**	**Name**	**Remark**
	FrameInfo	info	Frame additional information
	PcdKind	kind	Point cloud type
	std::vector<Point3d>	data	Point cloud data
Reference	3-2-3-2-8. FrameInfo, 3-2-2-2-3. PcdKind, 3-2-3-2-3. Point3d, 3-6-5-7. LensConv::convPcdCamera, 3-6-5-8. LensConv::convPcdWorld		

3-2-3-2-10-1.PcdData::resize

Table 32. PcdData::resize method

Function	Resize Point cloud data			
Syntax	inline void resize (const ImageFormat& format);			
Description	· Resize point cloud data using depth image format information.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	const ImageFormat&	format	in	Depth image format
Return value	None			
Sync/Async	Synchronous			

3-2-3-2-11. Frame

Table 33. Frame definition

Definition	struct Frame {
	Images images;
	PcdData pcd;
	};		

Description	· Indicates received frame information (single frame).		
Arguments	**Type**	**Name**	**Remark**
	Images	images	All image data
	PcdData	pcd	Point cloud data
Reference	3-2-4-2-2. Images, 3-2-3-2-10. PcdData, 3-4-5-10. Camera::capture, 3-5-5-6. Record::recFrame		

3-2-4. Type definition

3-2-4-1. List of type definitions

Table 34. Type definitions

Name	Description
ImageFormats	Format information for all image types
Images	Image data of all image types

3-2-4-2. Type definition details

3-2-4-2-1. ImageFormats

Table 35. ImageFormats definition

Definition	using ImageFormats = std::array<ImageFormat, IMG_KINDS>;
Description	・ Indicates the format information for all image types.
Reference	3-2-2-2-2. ImageKind, 3-2-3-2-4. ImageFormat

3-2-4-2-2. Images

Table 36. Images definition

Definition	using Images = std::array<ImageData, IMG_KINDS>;
Description	・ Shows image data for all image types.
Reference	3-2-2-2-2. ImageKind, 3-2-3-2-9. ImageData

3-3. PipelineFramework

3-3-1. Overview

Table 37. PipelineFramework overview

Provided header files	PlFw.h	PipelineFramework class definitions
	FrameData.h	FrameData class definitions
	EvtThread.h	EvtThread class definitions
	RecordThread.h	RecordThread class definitions
	RecordThreadEvent.h	Event type definition for RecordThread class
	LensConvThread.h	RecordThread class definition
	LensConvThreadEvent.h	Event type definition for RecordThread class
Member namespace	krm	
Description	Framework to build Pipeline processing and each thread processed by Pipeline By providing your own class that inherits from EvtThread class, you can incorporate your own processing as one thread in Pipeline processing.	
Thread Safety	The PipelineFramework class is thread-safe. There is no need for exclusive processing on the user program side.	

3-3-2. Provided functions

The following is an overview of the functions provided by PipelineFramework.

Table 38. PipelineFramework internally provided functions

Provided functions	Functional overview
Camera device control	・ Use the Camera class to control camera devices. ・ For details on this function, refer to the chapter below. 3-4-2-1. Device control functions
File playback	・ Performs file playback using the Camera class (PlayBack). ・ For details on this function, refer to the following chapter.

	3-4-2-2. File playback function
File save	・ Saves a file using the Record class. ・ For details about this function, see the following chapter. 3-5-2. Provided function
Distortion correction Point cloud conversion	・ Performs distortion correction and point cloud conversion using the LensConv class. ・ For details on this function, see the following chapter. 3-6-2. Provided function
Pipeline processing	・ This function executes each process on image data received from the camera device as a Pipeline process and outputs the processing result. ・ You can determine the order and configuration of each process to be executed as a process within the Pipeline. However, if there is a dependency on the order of execution, you must consider the order of processing and determine the Pipeline processing configuration in your application.
Function for adding user threads	・ This function allows you to add arbitrary processing threads to Pipeline processing. ・ You can add arbitrary processing classes that inherit from EvtThread to Pipeline processing. ・ For image processing to image data, place them before LensConvThread, and for processing to Point Cloud data, place them after LensConvThread.
Event notification	・ You can notify processing threads added to Pipeline of their own events. (Parameter changes, processing start/stop, etc.) ・ For details about the unique events for each processing thread, see the chapter for each processing thread class.
Status notification	・ This function reports the status of each processing thread, such as the occurrence of an error or completion of processing. ・ For details about the notifications for each processing thread, see the chapter for each processing thread class.

3-3-3. PipelineFramework internal classes

The following is a list of related classes used as PipelineFramework.

Table 39. List of internal classes of PipelineFramework

Class name	Description
PIFw	C++ class that centrally manages classes with a single function, such as Camera class, Record class, and LensConv class, and provides a framework for control sequences and parallel processing by Pipeline. By controlling this class, you can collectively control the functions of each included class.
FrameData	C++ class that manages single-frame images, Point Cloud, and other data
EvtThread	C++ class as a base class for a parallel processing framework using Pipeline
RecordThread	C++ class as a processing thread on Pipeline using the Record class
LensConvThread	C++ class as a processing thread on Pipeline using the LensConv class

3-3-3-1. Class relationships

The relationships among the classes in Pipeline Framework are shown below.

The Pipeline Framework (PlFw) class controls each thread registered as a Pipeline process (LensConvThread, RecordThread, or User Thread inherited from EvtThread, etc.) to perform various processes on the images (FrameData) received from the camera device as a Pipeline.

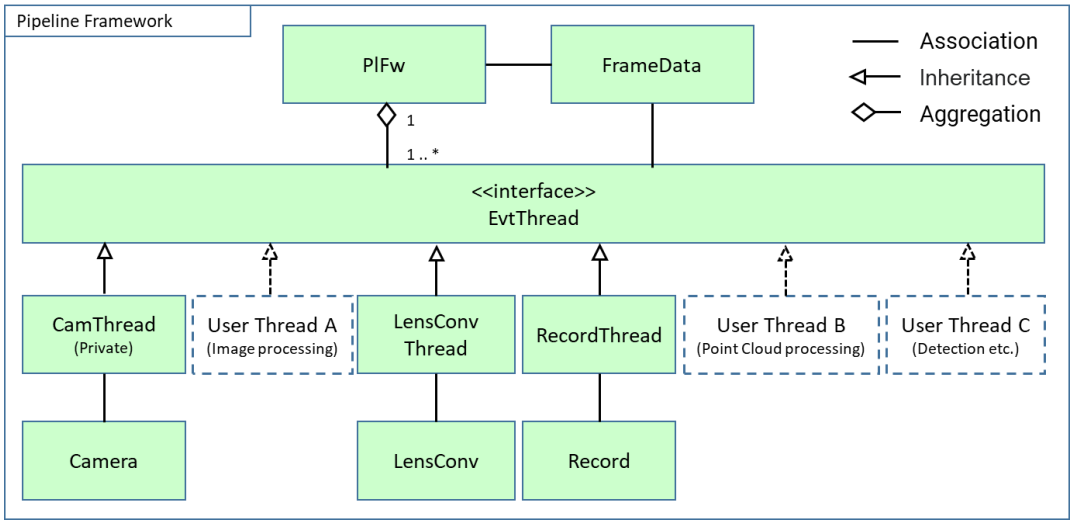


Figure 2. Class relationship diagram

3-3-4. Control sequence

This section describes the camera control sequence using the PipelineFramework class. In the following sequence, the judgment of the abnormal system such as the return value judgment of the function is omitted.

3-3-4-1. Initialization sequence

This section describes the sequence in which each thread is registered as Pipeline processing during initialization. Change the threads to be registered according to the use.

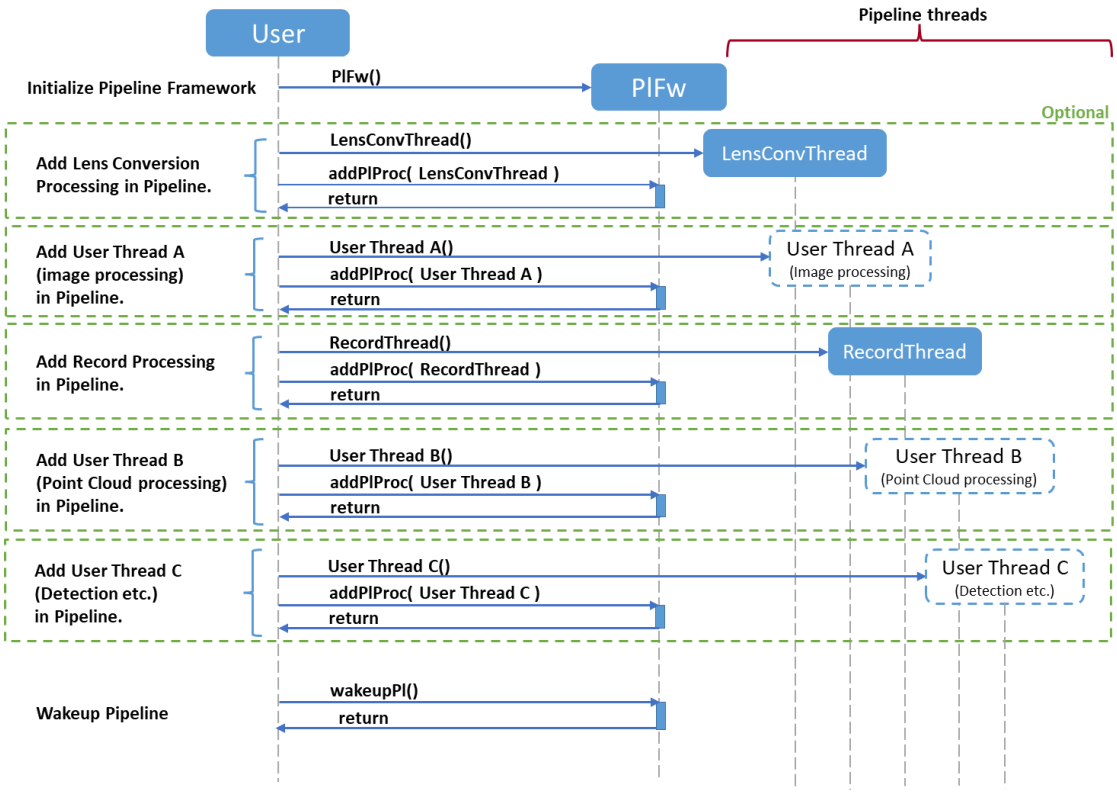


Figure 3. Initialization sequence

3-3-4-2. Information acquisition and operation mode switching sequence

The sequence of information acquisition and operation mode switching after initialization is described below.

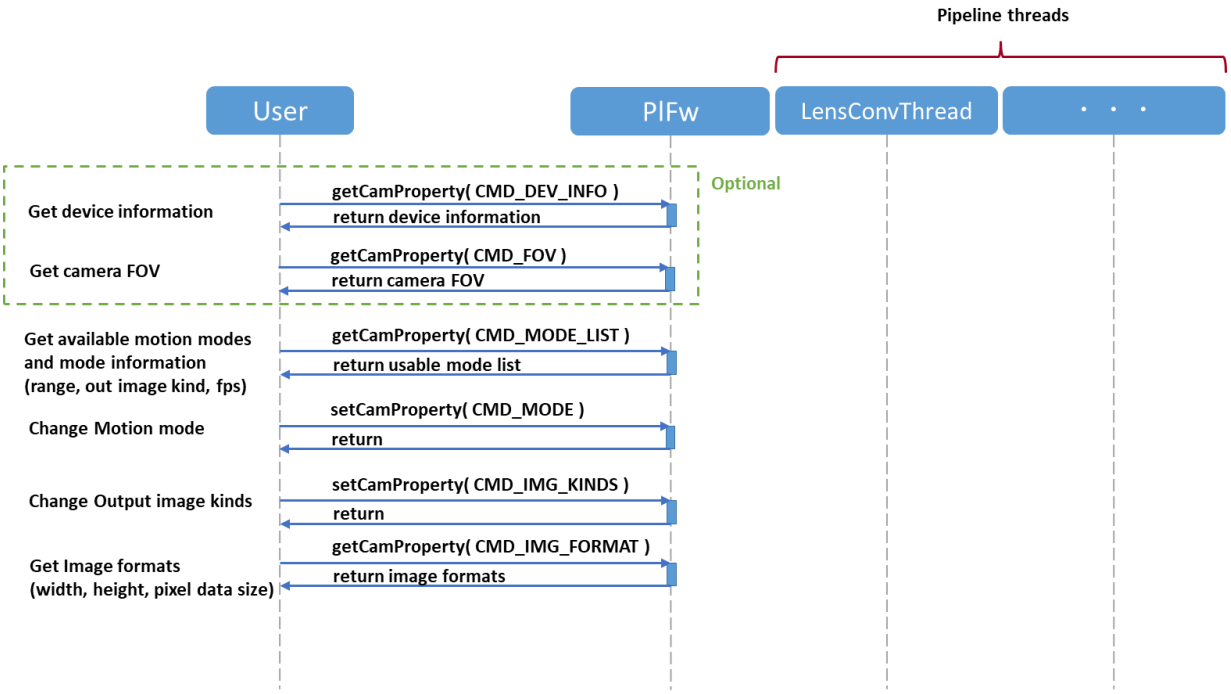


Figure 4. Information acquisition and operation mode switching sequence

3-3-4-3. Image reception sequence (closed_pl = false)

The image reception sequence when the closed_pl argument of PlFw::wakeupPl() is set to false is shown below. In this case, the User Application must receive the image using PlFw::getEvent().

Note that the same method is called for all threads registered as Pipeline, so it is omitted in this sequence.

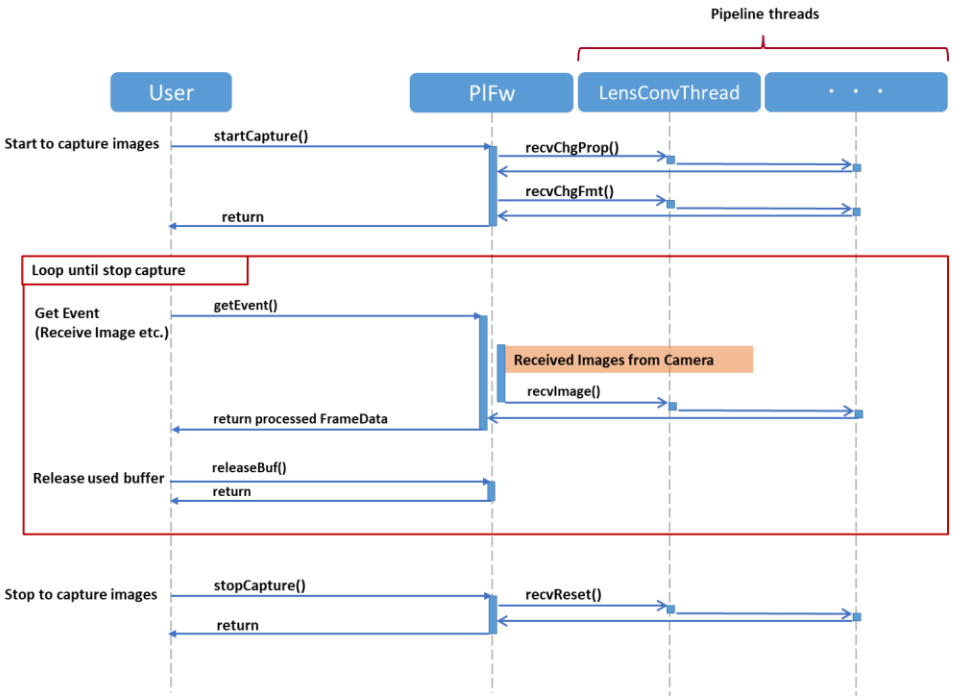


Figure 5. Image reception sequence (closed_pl = false)

3-3-4-4. Image reception sequence (Closed Pipeline)

The image reception sequence when the closed_pl argument of PlFw::wakeupPl() is set to true is shown below. In this case, the User Application cannot receive the image using PlFw::getEvent().

Since the same method is called for all threads registered as Pipeline, it is omitted in this sequence.

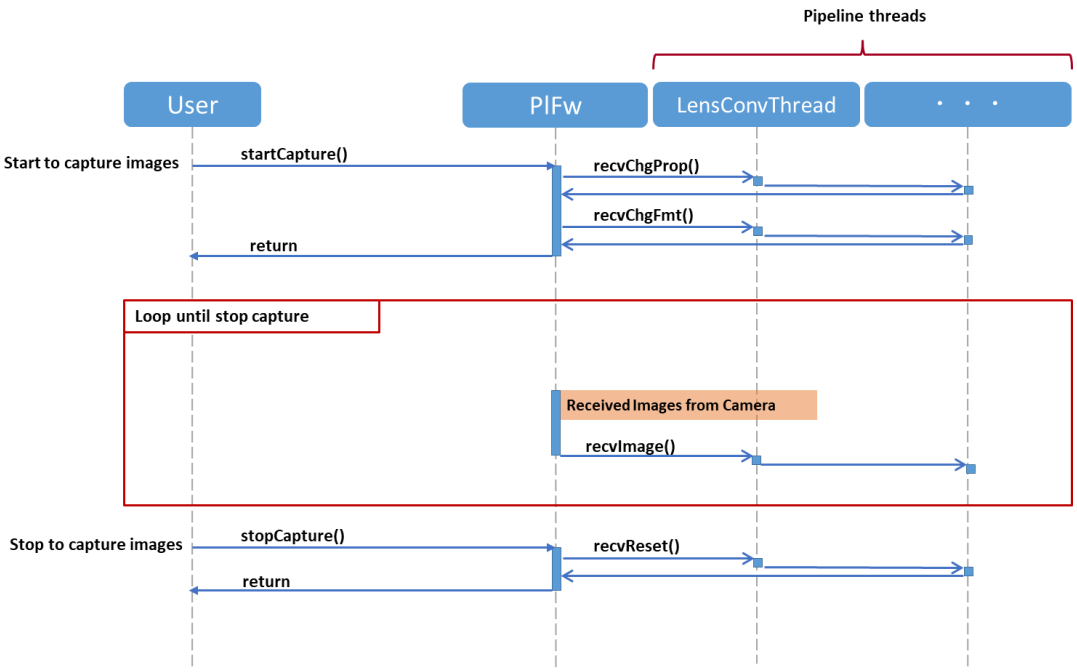


Figure 6. Image reception sequence diagram (Closed Pipeline)

3-3-4-5. Event notification sequence

The event notification sequence using PlFw::notifyEvent() is shown below.
As an example, the notification of save start and save stop events to a RecordThread is shown, but the format is the same for non-RecordThread.

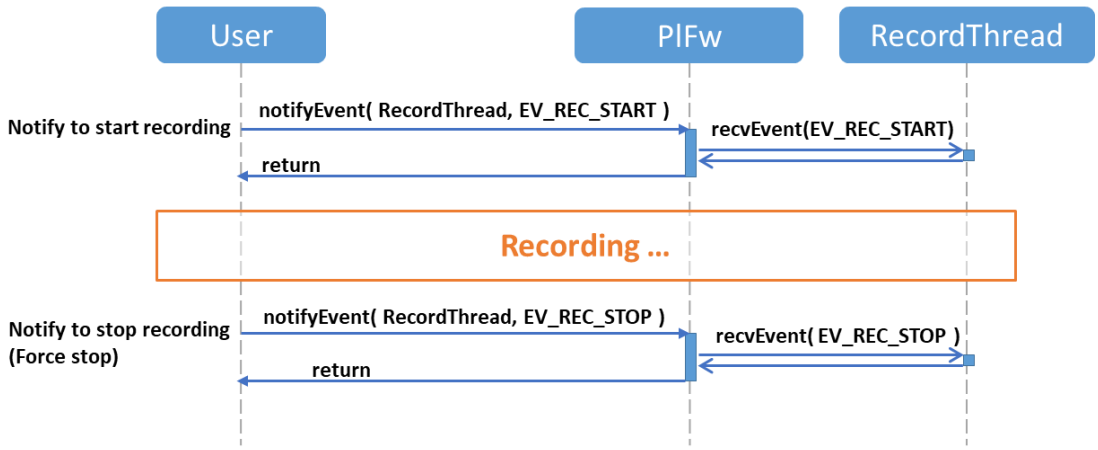


Figure 7. Event notification sequence diagram (Example: RecordThread)

3-3-4-6. Status notification sequence

The status notification sequence from a thread registered for Pipeline processing is shown below.
As an example, the status notification of the completion of a save after the start of a save for a RecordThread is shown here, but the format is similar for other than RecordThread.

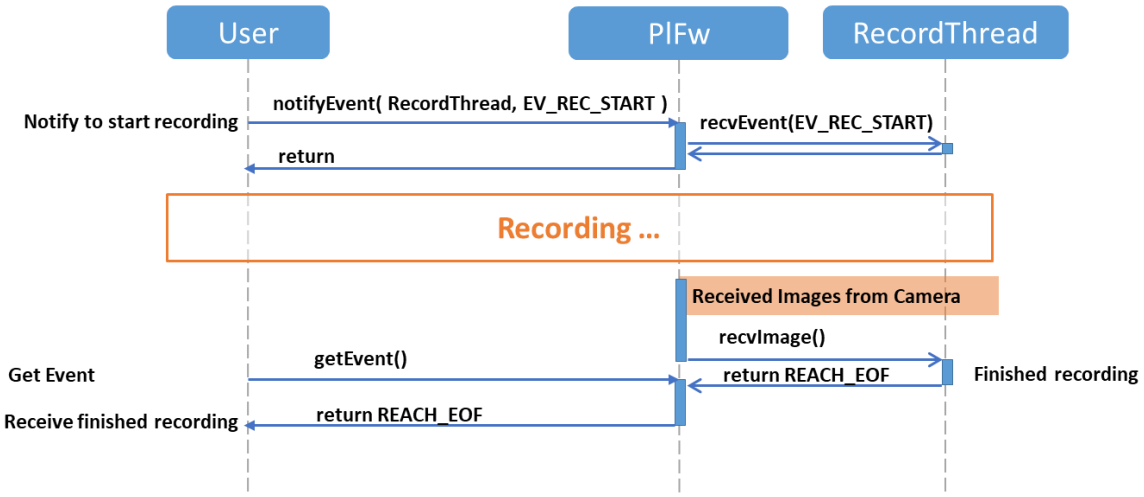


Figure 8. Status notification sequence diagram (Example: RecordThread)

3-3-4-7. End sequence

When finished, the sequence is shown below. Since all threads registered in the pipeline follow the same pattern, they are omitted from this sequence diagram.

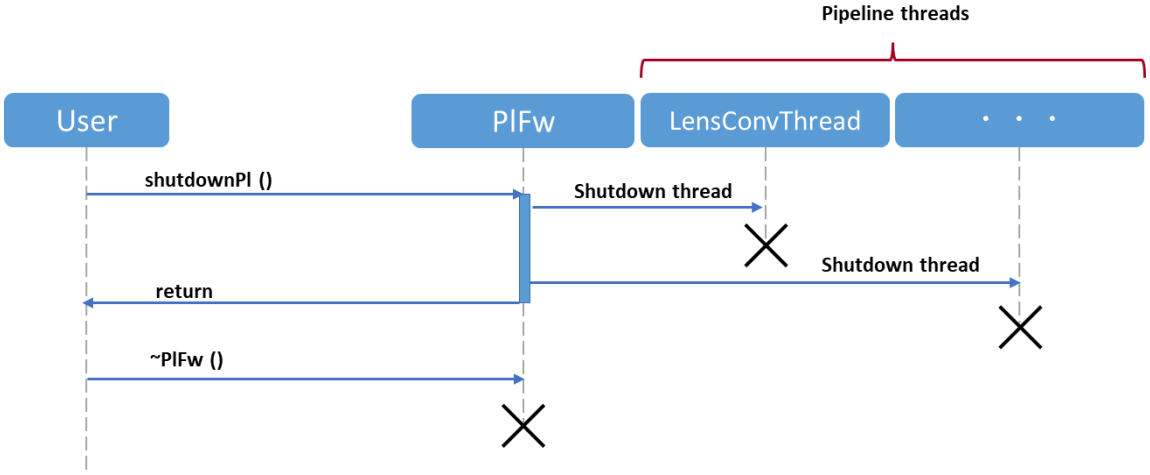


Figure 9. End sequence diagram

3-3-5. Pipeline processing configuration example

Pipeline Framework allows you to construct Pipeline processing by combining RecordThread, LensConvThread, and User Thread that inherit from EvtThread.

3-3-5-1. Pipeline processing configuration example (1): Saving camera output as a file

In this configuration example, only RecordThread is registered as Pipeline processing and only has the function to save the image output from Camera as a file.

By setting closed_pl=true when executing PlFw::wakeupPl(), Application can omit processing of the image data.

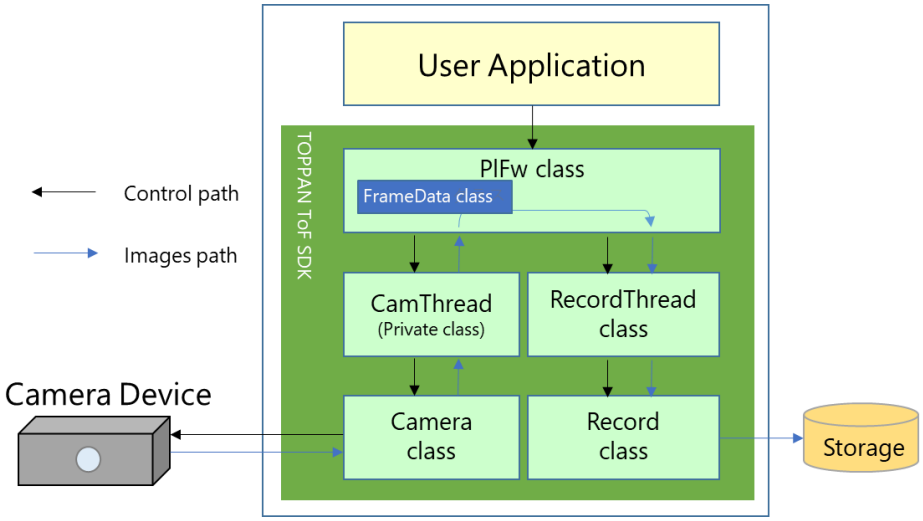


Figure 10. Pipeline processing configuration example (1): Saving camera output as a file

3-3-5-2. Pipeline processing example (2): Image display after image processing

In this example, only LensConvThread is registered as Pipeline processing, and the result of performing distortion correction and PointCloud conversion on the image output from Camera is displayed on the screen.

Setting closed_pl=false when executing PlFw::wakeupPl() enables Application to receive the image data after Pipeline processing.

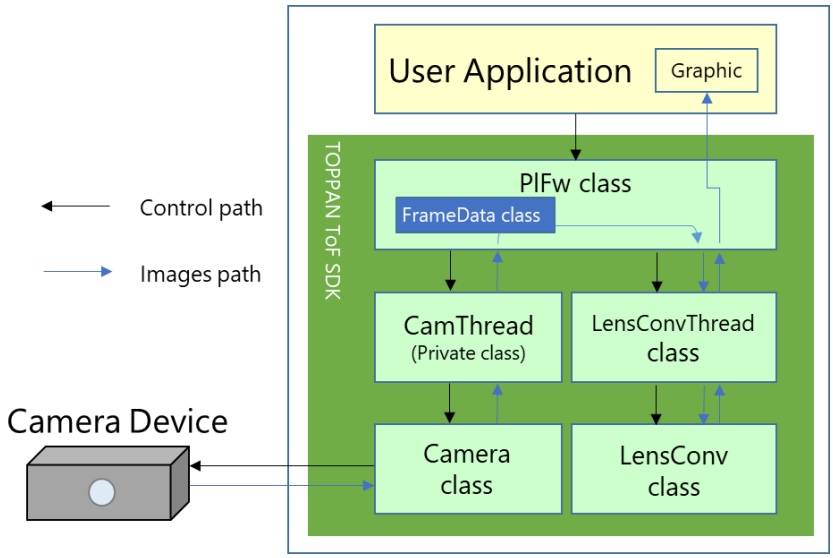


Figure 11. Pipeline processing example (2): Image display after image processing

3-3-5-3. Pipeline processing example (3): Detection processing after image processing

In this example, the original image processing, LensConvThread, and the original detection processing are registered as Pipeline processing, and the detection results are displayed on the screen.

The original image processing that inherits EvtThread is provided as User Thread A (tentative name), and the original detection processing is provided as User Thread B (tentative name). By registering each of these as Pipeline processing, a series of processing can be collectively controlled as a Pipeline.

By specifying an additional buffer size when executing PIFw::addPIProc(), a unique area can be provided in the FrameData class that manages image data, and the detection results can be passed according to the image data.

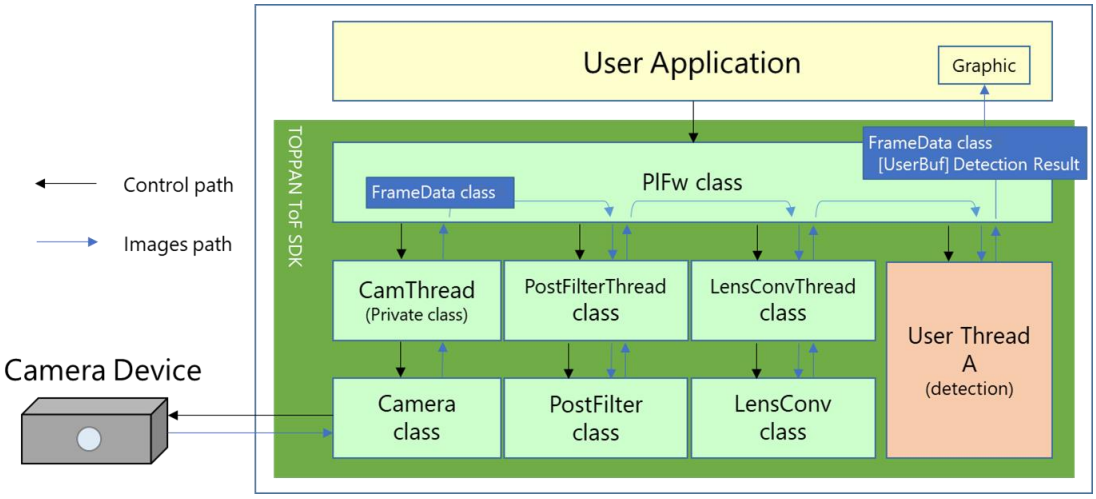


Figure 12. Pipeline processing example (3): Detection processing after image processing

3-3-5-4. Pipeline processing example (4): External output of detection results

This is a configuration example of outputting the detection results to an external device instead of the screen for the configuration of "Pipeline processing example (3): Detection processing after image processing".

By adding processing to send to an external device that inherits EvtThread as User Thread C (tentative name) to the Pipeline processing, you can collectively execute a series of processing up to the output to the external device in the Pipeline processing.

Note that if it is also necessary to receive the control signal from the external device in this configuration, you must perform the reception processing in the User Application.

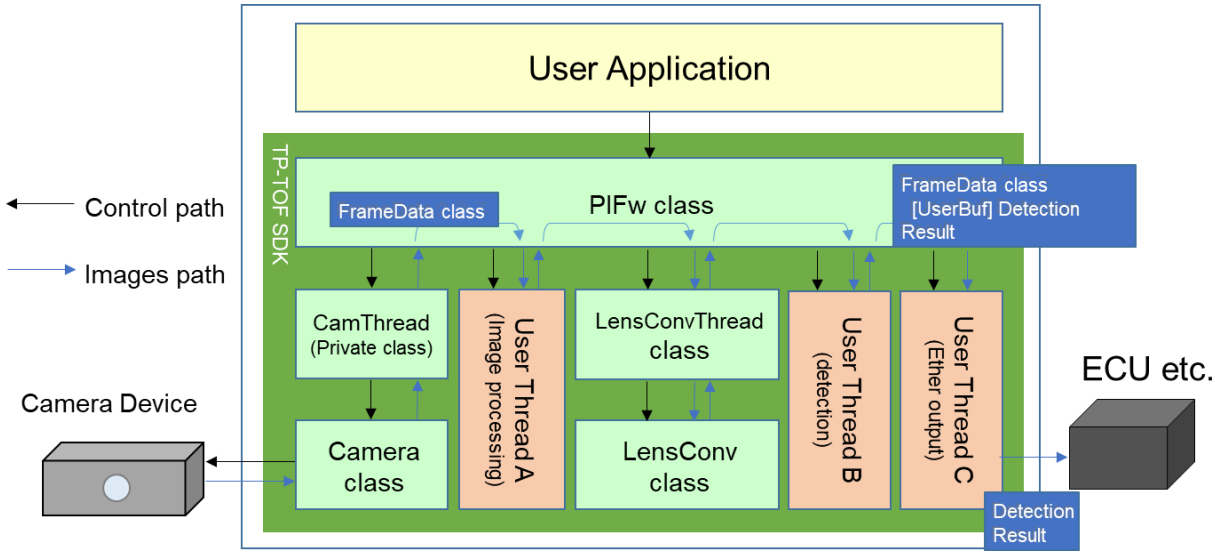


Figure 13. Pipeline processing example (4): External output of detection results

3-3-6. PIFw

3-3-6-1. Overview

Table 40. PIFw class overview

Class name	PIFw
Member Namespace	Krm
Description	PipelineFramework representative class

3-3-6-2. List of methods

Table 41. PipelineFramework class methods

Method name	Description
PIFw::PIFw	Constructor
PIFw::~PIFw	Destructor
PIFw::getCamDeviceList	Obtaining connected camera device list
PIFw::addPIProc	Adding pipeline processing
PIFw::wakeupPI	Starting pipeline processing
PIFw::shutdownPI	Ending pipeline processing
PIFw::getCamProperty	Getting camera device parameters
PIFw::setCamProperty	Setting camera device parameter
PIFw::startCapture	Image output start
PIFw::stopCapture	Image output stop

PlFw::getEvent	Receive output image or status notification after pipeline processing
PlFw::releaseBuf	Release output image
PlFw::notifyEvent	Informing the pipeline process

3-3-6-2-1. State machine

PipelineFramework using the PlFw class has the state machine shown in the following figure.

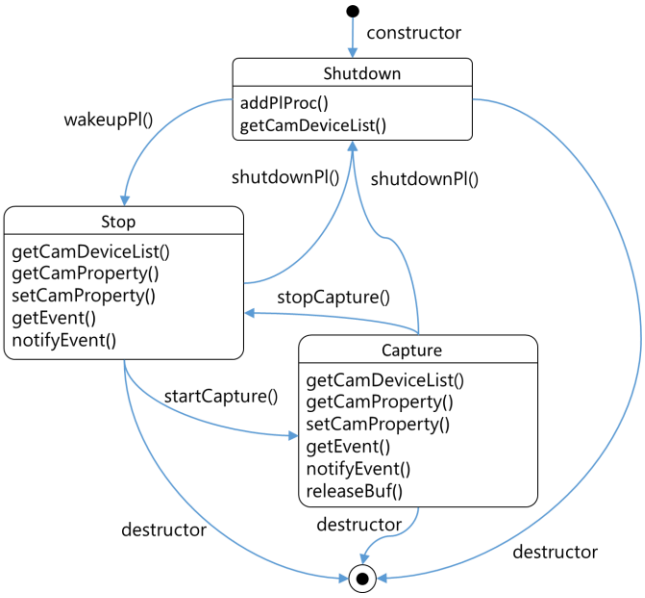


Figure 14. PipelineFramework state machine

3-3-6-3. Method details

3-3-6-3-1. PlFw::PlFw

Table 42. PlFw::PlFw method

Function	Constructor			
Syntax	PlFw (CameraType type, const void* param, Result& res)			
Description	• Initializes the Pipeline framework and camera control. • ERR_OVER_RANGE is returned in the res argument if the type of argument is out of the default. • When using a 3D ToF camera (C11U), specify C11U_USB for the type argument and nullptr for the param argument. • When using the file playback function, specify PLAYBACK for the type argument and PlayBack::ConfigParam* for the param argument. The playback target directory path can also be reset using PlFw::setCamProperty (PlayBack::CMD_PLAY_TARGET).			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	CameraType	type	in	Camera type used
	const void*	param	in	Device parameter
	Result&	res	out	Return value

Return value	SUCCESS	0	Success
	ERR_INVALID_PTR	3	An argument pointer is invalid (when PLAYBACK is specified).
	ERR_OVER_RANGE	4	Incorrect camera type
	ERR_SYSTEM	12	Memory allocation failed
Sync/Async	Synchronous		

3-3-6-3-2. PlFw::~PlFw

Table 43. PlFw::~PlFw method

Function	Destructor			
Syntax	~PlFw (void)			
Description	• Performs initialization of Pipeline framework and termination processing of Camera control. • To release internally allocated resources in this processing, make sure that the destructor is executed when the processing is terminated. • After executing PlFw::wakeupPl(), if PlFw::shutdownPl() is not executed, PlFw::shutdownPl() is executed in this method. • After executing PlFw::startCapture(), if PlFw::stopCapture() is not executed, PlFw::stopCapture() is executed in this method. • If the image buffer received by PlFw::getEvent() is not released by PlFw::releaseBuf(), PlFw::releaseBuf() is executed in this method.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	None			
Return value	None			
Sync/Async	Synchronous			

3-3-6-3-3. PlFw::getCamDeviceList

Table 44. PlFw::getDeviceList method

Function	Acquire connected camera device list			
Syntax	Result getCamDeviceList (std::vector<ConnDevice>& dev_list)			
Description	• Acquires List of connected camera devices. • This method has the same meaning as Camera:: getDeviceList().			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	std::vector<ConnDevice>&	dev_list	out	Connected camera device list
Return value	SUCCESS	0	Success	
	ERR_BAD_STATE	6	Bad state	
	ERR_NOT_EXIST	7	Target device not connected	
	ERR_SYSTEM	12	System error (memory allocation failure, etc.)	
Sync/Async	Synchronous			

3-3-6-3-4. PlFw::addPlProc

Table 45. PlFw::addPlProc method

Function	Add Pipeline processing				
Syntax	Result addPlProc (std::shared_ptr<EvtThread>& proc, size_t ex_buf_size, uint16_t& proc_id)				
Description	• Adds any processing to the Pipeline processing. • Pipeline processing is performed in the order in which it is added by this method. • In the proc argument, specify the object of the processing thread on which the constructor is executed. • In the ex_buf_size argument, specify the size of the buffer to be added as frame information. If no additional buffer is needed, specify 0. Since the additional buffer is allocated as a contiguous address, variable-length data and pointers cannot be used in the additional buffer. • On success, the process ID added to the proc_id argument is returned. This process ID is used to determine the event notification source to be notified by PlFw::getEvent(), to notify the event by PlFw::notifyEvent(), and as an argument to FrameData::getUserBuf(). • The registered processing thread is released by the destructor of this class. • If this method is called after executing PlFw::wakeupPl(), ERR_BAD_STATE is returned. • The maximum number of processes that can be added by this method is 65534. If this limit is exceeded, ERR_FULL is returned.				
Arguments	**Type**	**Name**	**in/out**	**Remark**	
	std:: shared_ptr<EvtThread>&	proc	in	Processing thread to add	
	size_t	ex_buf_size	in	Additional buffer size	
	uint16_t&	proc_id	out	Process ID	
Return value	SUCCESS	0	Success		
	ERR_INVALID_PTR	3	An argument pointer is invalid.		
	ERR_BAD_STATE	6	Status transition error		
	ERR_FULL	10	The maximum registrable limit (65534) has been exceeded.		
	ERR_SYSTEM	12	System error (memory allocation failure, etc.)		
Sync/Async	Synchronous				

3-3-6-3-5. PlFw::wakeupPl

Table 46. PlFw::wakeupPl method

Function	Invoke Pipeline processing
Syntax	Result wakeupPl (uint16_t dev_id, bool closed_pl = false, const PlCbFunc notify_cb = nullptr)
Description	• Starts Pipeline processing and establishes a connection with a camera device. • For the dev_id argument, specify the device ID acquired by PlFw::getCamDeviceList().

	・ When the argument closed_pl is true, Pipeline processing is closed in this class, and PlFw::getEvent() cannot be used. ・ In the argument notify_cb, register a callback function only when closed_pl is true. If an error occurs in the receive status or a status notification occurs from each thread, the return value similar to that of PlFw::getEvent() is notified to the registered callback. ・ When the argument closed_pl is false, Pipeline processing is performed in this class, and the processed data is notified to PlFw::getEvent(). ・ After executing this method, if this method is executed again without PlFw::shutdownPl() being executed, SUCCESS is returned without any processing.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	uint16_t	dev_id	in	Device ID
	bool	closed_pl	in	Closed Pipeline Flag true : closed Pipeline false : enable PlFw::getEvent()
	const PlCbFunc	notify_cb	in	Status notification callback
Return value	SUCCESS	0	Success	
	ERR_NOT_EXIST	7	Target device not connected	
	ERR_BAD_STATE	6	Abnormal state transition	
	ERR_NOT_SUPPORT	11	The firmware version of the target camera is not the target camera.	
	ERR_SYSTEM	12	System error (memory allocation failure, etc.)	
Sync/Async	Synchronous			

3-3-6-3-6. PlFw::shutdownPl

Table 47. PlFw::shutdownPl method

Function	End of Pipeline processing			
Syntax	Result shutdownPl (void)			
Description	・ Disconnects the camera device and terminates Pipeline processing. ・ After executing this method, PlFw::getEvent() returns CANCELED. ・ If this method is executed while PlFw::wakeupPl() has not been executed, it returns SUCCESS without processing anything. ・ After executing PlFw::startCapture(), if PlFw::stopCapture() has not been executed, PlFw::stopCapture() is executed within this method.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	None			
Return value	SUCCESS	0	Success	
	ERR_BAD_STATE	6	Status transition error	
	ERR_SYSTEM	12	System error (memory allocation failure, etc.)	
Sync/Async	Synchronous			

3-3-6-3-7. PlFw::getCamProperty

Table 48. PlFw::getCamProperty method

Function	Aquire camera device parameters
Syntax	Result getCamProperty (uint16_t prop_cmd,

	void* param)
Description	・ Gets the parameters of the camera device. ・ This method has the same contents as Camera::getProperty. For details about the arguments and return values, see the chapter on setProperty() and property commands in the Camera class. ・ ERR_BAD_STATE is returned if this method is called without executing PlFw::wakeupPl().

Arguments	Type	Name	in/out	Remark
	uint16_t	prop_cmd	in	Property commands (for camera devices) Property command (for file playback only)
	void*	param	in,out	Acquisition parameter

Return value			
	SUCCESS	0	Success
	ERR_INVALID_PTR	3	The argument pointer is invalid.
	ERR_OVER_RANGE	4	The set value is outside the specifiable range (prop_cmd)
	ERR_BAD_STATE	6	Status transition error
	ERR_TIMEOUT	8	A timeout occurred during communication between camera devices.
	ERR_NOT_SUPPORT	11	Unsupported functions
	ERR_SYSTEM	12	System error
Sync/Async	Synchronous		

3-3-6-3-8. PlFw::setCamProperty

Table 49. PlFw::setCamProperty method

Function	Set camera device parameters
Syntax	Result setCamProperty (uint16_t prop_cmd, const void* param = nullptr)
Description	・ Set camera device parameters. ・ This method has the same contents as Camera:: setProperty(). For details about the arguments and return values, refer to setProperty() and property commands in the Camera Library API Specification. ・ ERR_BAD_STATE is returned if this method is called without executing PlFw::wakeupPl().

Arguments	Type	Name	in/out	Remark
	uint16_t	prop_cmd	in	Property commands (for camera devices) Property command (for file playback only)
	const void*	param	in,out	Acquisition parameter

Return value			
	SUCCESS	0	Success
	ERR_INVALID_PTR	3	Argument pointer is invalid.
	ERR_OVER_RANGE	4	The set value is outside the specifiable range.
	ERR_BAD_ARG	5	Other, invalid argument
	ERR_BAD_STATE	6	Abnormal state transition
	ERR_NOT_EXIST	7	Target directory file does not exist.

	ERR_TIMEOUT	8	A timeout occurred during communication between camera devices.
	ERR_NOT_SUPPORT	11	Unsupported functions
	ERR_SYSTEM	12	System error
Sync/Async	Synchronous		

3-3-6-3-9. PlFw::startCapture

Table 50. PlFw::startCapture method

Function	Start image output			
Syntax	Result startCapture (void)			
Description	· Starts image output from the camera device. · When the closed_pl argument is set to false in PlFw::wakeupPl(), the image is received by PlFw::getEvent() after image output starts. · ERR_BAD_STATE is returned when this method is called without executing PlFw::wakeupPl(). · If this method is called when image output from the camera device has already been performed, no processing is performed and SUCCESS is returned.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	None			
Return value	SUCCESS	0	Success	
	ERR_BAD_STATE	6	State transition error	
	ERR_NOT_EXIST	7	Target device disconnected (no playback target file)	
	ERR_SYSTEM	12	System error	
Sync/Async	Synchronous			

3-3-6-3-10. PlFw::stopCapture

Table 51. PlFw::stopCapture method

Function	Stop image output			
Syntax	Result stopCapture (void)			
Description	· Stops image output from the camera device. · ERR_BAD_STATE is returned if this method is called without executing PlFw::wakeupPl(). · If this method is called without executing PlFw::startCapture(), nothing is processed and SUCCESS is returned.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	None			
Return value	SUCCESS	0	Success	
	ERR_BAD_STATE	6	Abnormal state transition	
	ERR_SYSTEM	12	System error	
Sync/Async	Synchronous			

3-3-6-3-11. PlFw::getEvent

Table 52. PlFw::capture method

Function	Receiving output images or status notifications after Pipeline processing			
Syntax	Result getEvent (uint16_t& proc_id, FrameData** frame, bool block = true)			
Description	・ Gets an image that has been processed by Pipeline on the output image of the camera device. ・ If the proc_id argument is PROC_PIPELINE, the output image after Pipeline processing is notified, and the output image is stored in the frame argument. Be sure to use PlFw::releaseBuf() to release the output image in the frame argument after using it. ・ If the proc_id argument is not PROC_PIPELINE, the processing ID of the processing thread added by PlFw::addPlProc() is input, and the status notification from the processing thread is returned as the return value. For details about the reported return value, see the chapter for each processing thread. ・ ERR_BAD_STATE is returned when this method is called without executing PlFw::wakeupPl(). ・ ERR_BAD_STATE is returned when the closed_pl argument of PlFw::wakeupPl() is set to true. ・ If this method is not called after image output is started with PlFw::startCapture(), the received frame is discarded. ・ The reception wait behavior varies according to the specification of the block argument as follows. [When the block argument is true] ・ This method waits for reception until an output image or status notification is received. ・ When PlFw::shutdownPl() is called while waiting for reception, the reception is canceled and CANCELED is returned. ・ When the external trigger type is Standalone (EXT_TRG_STANDALONE) or Primary (Master, EXT_TRG_MASTER): If the camera doesn't get an image within about 1 second, it'll try stopping and restarting reception 5 times. If it still can't get an image after those tries, it'll time out and return an ERR_TIMEOUT. ・ When the external trigger type is Secondary (Slave, EXT_TRG_SLAVE): If an image isn't received within roughly 1 second, Camera::capture() will return an ERR_TIMEOUT. The camera tracks consecutive ERR_TIMEOUTs, and if that count hits 86,400 (about 24 hours), it'll ultimately return an ERR_TIMEOUT. This count resets once the camera is no longer waiting for an image. ・ Even if image output has not started by PlFw::startCapture(), reception wait status is set. In this case, reception wait timeout does not occur. [If the block argument is false] ・ If there is no output image or status notification, reception wait status is not set in this method and ERR_NOT_EXIST is returned.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	uint16_t&	proc_id	out	Process ID
	FrameData**	frame	out	Receive frame
	bool	block	in	Receive wait flag

				true : Waits for reception false : Does not wait for reception
Return value	SUCCESS		0	Success
	CANCELED		1	Wait status released.
	REACH_EOF		2	End of file reached (PLAYBACK only)
	ERR_INVALID_PTR		3	Invalid pointer (frame)
	ERR_BAD_STATE		6	Status transition error
	ERR_NOT_EXIST		7	Output image does not exist.
	ERR_TIMEOUT		8	Timeout occurred while waiting for reception.
	ERR_SYSTEM		12	System error
Sync/Async	Synchronous			

3-3-6-3-12. PlFw::releaseBuf

Table 53. PlFw::releaseBuf method

Function	Release output image			
Syntax	Result releaseBuf (FrameData** frame)			
Description	· Releases the output image acquired by PlFw::getEvent(). · The frame argument becomes nullptr after it is released. · ERR_BAD_STATE is returned when this method is called without executing PlFw::wakeupPl(). · If true is specified for the closed_pl argument of PlFw::wakeupPl(), this method returns ERR_BAD_STATE.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	FrameData**	frame	in,out	Incoming Frames
Return value	SUCCESS		0	Success
	ERR_INVALID_PTR		3	Illegal argument
	ERR_BAD_STATE		6	Status transition error
	ERR_SYSTEM		12	System error
Sync/Async	Synchronous			

3-3-6-3-13. PlFw::notifyEvent

Table 54. PlFw::notifyEvent method

Function	Notification of events to processing threads in Pipeline processing
Syntax	Result notifyEvent (uint16_t proc_id, uint8_t event, void* param = nullptr)
Description	· This function notifies the processing thread added by PlFw::addPlProc(). · In the proc_id argument, specify the processing ID returned by PlFw::addPlProc(). · The event and param arguments are unique events for each processing thread added by PlFw::addPlProc(). For details, see the chapter for each processing thread

	class. For user threads created independently by inheriting the EvtThread class , define events in the corresponding user thread. ・ The contents and area of the param argument must be retained until processing of this method is completed. ・ If this method is called without executing PlFw::wakeupPl(), ERR_BAD_STATE is returned. ・ ERR_TIMEOUT is returned if there is no response for 25 seconds after notification of an event due to a processing delay of the processing thread of the notification destination.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	uint16_t	proc_id	in	Process ID
	uint8_t	event	in	Event ID
	void*	param	in,out	Notification parameter
Return value	SUCCESS	0	Success	
	ERR_BAD_ARG	5	The processing ID or event ID is invalid.	
	ERR_BAD_STATE	6	Status transition error	
	ERR_TIMEOUT	8	Timeout occurred while waiting for a response.	
	ERR_SYSTEM	12	System error	
Sync/Async	Synchronous			

3-3-6-4. Definition for PlFw class

The definitions used by the PlFw class described in PlFw.h are shown below. Definitions used only internally by PipelineFramework are omitted.

3-3-6-4-1. Constant definition

3-3-6-4-1-1.List of constant definitions

Table 55. Constant definition

Name	Description
PROC_PIPELINE	Processing ID indicating image reception

3-3-6-4-1-2.Constant definition details

3-3-6-4-1-2-1.PROC_PIPELINE

Table 56. PROC_PIPELINE definition

Definition	static const uint16_t PROC_PIPELINE = 0;
Description	・Processing ID that indicates image reception in PlFw::getEvent()
Reference	3-3-6-3-11. PlFw::getEvent

3-3-7. FrameData class

3-3-7-1. Overview

Table 57. FrameData class overview

Class name	FrameData

Member Namespace	krm
Description	C++ class for managing image frames received with PlFw::getEvent() and EvtThread::recvChgProp()

3-3-7-2. List of methods

Table 58. FrameData class methods

Method name	Description
FrameData::getFrame	Image frame reference
FrameData::getFrameExt	Acquisition of frame additional information
FrameData::getUserBuf	Additional buffer reference

3-3-7-3. Method details

3-3-7-3-1. FrameData::getFrame

Table 59. FrameData::getFrame method

Function	Image frame reference			
Syntax	Frame* getFrame (void)			
Description	・ References an image frame.			
Arguments	Type	Name	in/out	Remark
	None			
Return value	Frame*	Image frame		
Sync/Async	Synchronous			

3-3-7-3-2. FrameData::getFrameExt

Table 60. FrameData::getFrameExt method

Function	Acquisition of frame additional information			
Syntax	FrameExtInfo* getFrameExt(void)			
Description	・ As frame additional information, the received frame rate (measured value) and the current playback position information at the time of PlayBack are referenced. ・ FrameExtInfo play_time is valid only when PLAYBACK is specified by PlFw::PlFw().			
Arguments	Type	Name	in/out	Remark
	None			
Return value	FrameExtInfo*	Frame additional information		
Sync/Async	Synchronous			

3-3-7-3-3. FrameData::getUserBuf

Table 61. FrameData::getUserBuf method

Function	Additional buffer reference
Syntax	void* getUserBuf (uint16_t proc_id)
Description	・ Refers to the buffer added during PlFw::addPlProc(). ・ The proc_id argument specifies the processing ID returned by PlFw::addPlProc(). ・ If no buffer was added during PlFw::addPlProc(), nullptr is returned.

	・ If the proc_id argument is invalid, nullptr is returned. ・ Cast the return value to any type.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	uint16_t	proc_id	in	Process ID
Return value	void*	First pointer of the additional buffer		
Sync/Async	Synchronous			

3-3-7-4. Definition for FrameData class

The following shows the definition used by the FrameData class described in FrameData.h.
Note that the definition used only within PipelineFramework is omitted.

3-3-7-4-1. Structure definition

3-3-7-4-1-1.List of structure definitions

Table 62. Structure definition

Name	Description
FrameExtInfo	Additional information for PipelineFramework frames

3-3-7-4-1-2.Structure definition details

3-3-7-4-1-2-1.FrameExtInfo

Table 63. FrameData::FrameExtInfo definition

Definition	struct FrameExtInfo {
	uint16_t rcv_fps;
	PlayBack::PlayTime play_time;
	};		

Description	・ Indicates additional frame information calculated or acquired in PipelineFramework.		
Arguments	**Type**	**Name**	**Remark**
	uint16_t	rcv_fps	Receive Frame Rate [fps × 100]
	PlayBack::CMD_PLAY_TIME	play_time	Current playback position information (only during PlayBack operation)
Reference	3-3-7-3-2. FrameData::getFrameExt		

3-3-8. EvtThread class

3-3-8-1. Overview

Table 64. EvtThread class overview

Class name	EvtThread
Member Namespace	krm
Description	Pipeline processing base class The processing thread to be added as Pipeline processing must inherit from this class.

3-3-8-2. List of methods

Table 65. EvtThread class methods

Method name	Description
EvtThread::EvtThread	Constructor
EvtThread::~EvtThread	Destructor
EvtThread::getKind	Acquisition of processing type
EvtThread::enableFrameDrop	Permit to discard frame data during processing stagnation
EvtThread::recvChgProp	Reception of camera device information change notification
EvtThread::recvChgFmt	Reception of image format change notification
EvtThread::recvImage	Image Frame Notification Receive
EvtThread::recvReset	Reset Notification Receive
EvtThread::recvUserEvent	Event Receive

3-3-8-3. Method details

3-3-8-3-1. EvtThread::EvtThread

Table 66. EvtThread::EvtThread method

Function	Constructor			
Syntax	EvtThread (void)			
Description	・ Initializes the processing thread class.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	None			
Return value	None			
Sync/Async	Synchronous			

3-3-8-3-2. EvtThread::~EvtThread

Table 67. EvtThread::~EvtThread method

Function	Destructor			
Syntax	~ EvtThread (void)			
Description	・ Destroys the processing thread class.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	None			
Return value	None			
Sync/Async	Synchronous			

3-3-8-3-3. EvtThread::getKind

Table 68. EvtThread::getKind method

Function	Acquisition of processing type			
Syntax	ProcKind getKind (void)			
Description	・ Returns the processing type. ・ If the EvtThread class is inherited, use this method without overriding it.			
Arguments	**Type**	**Name**	**in/out**	**Remark**

	None				
Return value	ProcKind	Processing type			
Sync/Async	Synchronous				

3-3-8-3-4. EvtThread::enableFrameDrop

Table 69. EvtThread::enableFrameDrop method

Function	Permitted to discard frame data when processing stagnates.			
Syntax	bool enableFrameDrop (void)			
Description	・ Returns whether to discard image frames if Pipeline processing is stalled during image reception. ・ If discarding of image frames is not allowed when EvtThread class is inherited, override and return false. However, if discarding of image frames is not allowed, image reception from the camera device is stopped when the image frame buffer required for image reception from the camera device is exhausted. ・ If discarding of image frames is allowed, image frames are discarded when the maximum number of accumulated frames returned by EvtThread::getMaxQueuingFrames() is exceeded when processing is stalled within a thread. When discarding an image frame, a frame discontinuity flag is set for the next image frame. (FrameInfo rm_err.drop =1) ・ This method is referenced in PlFw::addPlProc().			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	None			
Return value	true	Allow image frame discard when processing is stalled		
	false	Prohibit image frame discard when processing is stalled		
Sync/Async	Synchronous			

3-3-8-3-5. EvtThread::getMaxQueuingFrames

Table 70. EvtThread::getMaxQueuingFrames method

Function	Acquire maximum number of accumulated frames			
Syntax	uint8_t getMaxQueuingFrames (void)			
Description	・ Gets the maximum number of image frames that can be accumulated when the Pipeline processing is stalled during image reception. ・ If EvtThread::enableFrameDrop() returns true and the value returned by this method is exceeded, the overflowing image frames are discarded. ・ The initial value is 3 frames. If you want to set a value other than the initial value, override it when the EvtThread class is inherited and return the maximum number of frames to accumulate. ・ This method is referenced in PlFw::addPlProc().			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	None			
Return value	1～255	Maximum number of frames to store		
	0	The initial value (3) is used as the maximum number of frames to store.		
Sync/Async	Synchronous			

3-3-8-3-6. EvtThread::recvChgProp

Table 71. EvtThread::recvChgProp method

Function	Receive camera device information			
Syntax	Result recvChgProp (const CameraProperty& property)			
Description	· Called when PlFw::startCapture() is executed. · Override this method to use the camera device information (Ranging range, FPS, etc.) when the EvtThread class is inherited. · If this method returns a value other than SUCCESS, it will be reflected as the return value of PlFw::startCapture(). · The arguments of this method are valid only during execution of this method. Do not store them as pointers, etc.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	const CameraProperty&	property	in	Camera Device Information
Return value	SUCCESS	0	Success	
	Other	–	Defined by inherited class	
Sync/Async	Synchronous			

3-3-8-3-7. EvtThread::recvChgFmt

Table 72. EvtThread::recvChgFmt method

Function	Reception of image format information			
Syntax	Result recvChgFmt (const ImageFormats& formats)			
Description	· Called when an image frame is received after executing PlFw::startCapture(). · Override this method to use an image frame when you inherit from the EvtThread class. · If you overwrite the data in the image frame, the overwritten status is notified to the next Pipeline process. · The frame argument is passed with ownership. It is not necessary to release ownership in this method. · This method should be completed within an interval of one frame. · If this method returns a value other than SUCCESS, the return value is returned to the callback function registered with PlFw::getEvent() or PlFw::wakeupPl(). · The arguments of this method are valid only during execution of this method. Do not store them as pointers, etc.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	const ImageFormats&	formats	in	Image format
Return value	SUCCESS	0	Success	
	Other	–	Defined in inherited classes	
Sync/Async	Synchronous			

3-3-8-3-8. EvtThread::recvImage

Table 73. EvtThread::recvImage method

Function	Receive image frame notification

Syntax	Result recvImage (std::shared_ptr<FrameData>& frame)			
Description	· Called when an image frame is received after executing PlFw::startCapture(). · Override this method to use an image frame when you inherit from the EvtThread class. · If you overwrite the data in the image frame, the overwritten status is notified to the next Pipeline process. · The frame argument is passed with ownership. It is not necessary to release ownership in this method. · This method should be completed within an interval of one frame. · If this method returns a value other than SUCCESS, the return value is returned to the callback function registered with PlFw::getEvent() or PlFw::wakeupPl(). · The arguments of this method are valid only during execution of this method. Do not store them as pointers, etc.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	std::shared_ptr<FrameData>&	frame	in,out	Image frame
Return value	SUCCESS	0	Success	
	Other	–	Defined in inherited classes	
Sync/Async	Synchronous			

3-3-8-3-9. EvtThread::recvReset

Table 74. EvtThread::recvReset method

Function	Receive reset notification			
Syntax	void recvReset (void)			
Description	· Called when PlFw::stopCapture() is executed. · When the EvtThread class is inherited, override this method if initialization processing for stop processing is required.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	None			
Return value	None			
Sync/Async	Synchronous			

3-3-8-3-10. EvtThread::recvUserEvent

Table 75. EvtThread::recvUserEvent method

Function	Event reception
Syntax	Result recvUserEvent (uint8_t event, void* param)
Description	· Called when an event notification has occurred via PlFw::notifyEvent(). · Override this method if custom event processing is required when the EvtThread class is inherited. · The contents of the event and param arguments must be specified in the inherited class. · The return value returned by this method is returned as the return value of PlFw::notifyEvent().

	・ If the contents of the param argument are changed, the param argument of the side that called PlFw::notifyEvent() is also updated. ・ The arguments of this method are valid only during execution of this method. Do not store them in a pointer, etc.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	uint8_t	event	in	Event ID
	void*	param	in,out	Notification parameter
Return value	SUCCESS	0	Success	
	Other	–	Defined by inherited class	
Sync/Async	Synchronous			

3-3-8-4. List of variables in class

The following variables can be referenced by a class that inherits the EvtThread class:

Table 76. EvtThread class variable

Variable type	Variable name	Description
uint16_t	proc_id_	Process ID Stores the ID when it was added to Pipeline processing by PlFw::addPlProc(). Use this to access the additional buffer by FrameData::getUserBuf().

3-3-8-5. Definition for EvtThread class

The definitions used in EvtThread class described in EvtThread.h are shown below.
The definitions used only in PipelineFramework are omitted.

3-3-8-5-1. Enumeration definition

3-3-8-5-1-1.List of enumeration definitions

Table 77. Enumeration definition

Name	Description
ProcKind	Processing type

3-3-8-5-1-2.Enumeration definition details

3-3-8-5-1-2-1.ProcKind

Table 78. ProcKind definition

Definition	enum ProcKind : uint16_t {
	PROC_CAMERA = 0,
	PROC_RECORD,
	PROC_LENSCONV,
	PROC_POSTFILT,
	PROC_USER
	};		

Description	・ Indicates the processing type added to the Pipeline processing.		
Value	**Name**	**Value**	**Remark**

	PROC_CAMERA	0	Camera Receive Processing
	PROC_RECORD	1	File saving processing
	PROC_LENSCONV	2	Lens system conversion processing
	PROC_POSTFILT	3	PostFilter processing
	PROC_USER	4	User Additional processing
Reference	3-3-8-3-3. EvtThread::getKind		

3-3-8-5-2. Structure definition

3-3-8-5-2-1.List of structure definitions

Table 79. Structure definition

Name	Description
CameraProperty	Camera-specific information

3-3-8-5-2-2.Structure definition details

3-3-8-5-2-2-1.CameraProperty

Table 80. CameraProperty definition

Definition	struct CameraPropety { ModeInfo mode_info; LensInfo lens_info; PostFiltInfo post_filt_info; CamFov fov; } ;		
Description	・ Indicates camera-specific information (Operation mode information, Lens system conversion processing parameters, PostFilter processing information, FOV).		
Arguments	**Type**	**Name**	**Remark**
	ModeInfo	mode_info	Operation mode information
	LensInfo	lens_info	Lens type conversion processing parameters
	PostFiltInfo	post_filt_info	PostFilter processing information
	CamFov	fov	Camera Field of View
Reference	3-3-8-3-6. EvtThread::recvChgProp		

3-3-8-6. Type definition

3-3-8-6-1. List of type definitions

Table 81. Type definition

Name	Description
PlCbFunc	Callback function for state change notification

3-3-8-6-2. Type definition details

3-3-8-6-2-1.PlCbFunc

Table 82. PlCbFunc definition

Definition	using PlCbFunc = std::function<void(uint16_t, Result)>;
Description	・ Indicates a callback function for status change notification.

	・ In the argument, the processing ID of the processing thread added by PlFw::addPlProc() and the status notification from the corresponding processing thread indicated by the processing ID are returned as values. For details about the status notification that is reported, see the chapter for each processing thread.	
Arguments	**Type**	**Remark**
	uint16_t	Process ID
	Result	Status notification value
Return value	None	
Reference	3-3-6-3-5. PlFw::wakeupPl	

3-3-9. RecordThread class

3-3-9-1. Overview

Table 83. RecordThread class overview

Class name	RecordThread
Member Namespace	krm
Description	Thread that performs file storage processing using the Record class For the methods that inherit from EvtThread, see the EvtThread class contents.
Additional buffer	Not used

3-3-9-2. List of events

The following events can be used with PlFw::notifyEvent().

Table 84. RecordThread class events

Event	Description
EV_REC_START	Start saving file
EV_REC_STOP	File save stop

3-3-9-2-1. EV_REC_START

Table 85. EV_REC_START overview

Information type	File save start
Argument type	RecordParam
Description	・ File save starts. ・ The return value of Record:: openRec() is returned.

3-3-9-2-2. EV_REC_STOP

Table 86. EV_REC_STOP overview

Information type	Stop saving file
Argument type	None
Description	・ Stops saving the file. ・ The return value of Record:: closeRec() is returned.

3-3-9-3. Status notification

When a status notification occurs during RecordThread processing, the following values are notified to the PlFw::getEvent() or PlFw::addPlProc() callback function.

Table 87. RecordThread class status notifications

Notified value	Description
REACH_EOF	Completed saving for the specified number of frames
ERR_BAD_STATE	Failed to save due to abnormal state transition
ERR_EMPTY	Saving failure due to empty buffer
ERR_FULL	Saving failure due to insufficient space
ERR_SYSTEM	Saving failure due to a system error (memory allocation failure, etc.)

3-3-9-4. Definition for RecordThread class

The definitions used for event notification for the RecordThread class described in RecordThreadEvent.h are as follows:

3-3-9-4-1. Enumeration definition

3-3-9-4-1-1.List of enumeration definitions

Table 88. Enumeration definition

Name	Description
RecEvent	Event ID of event notification for the RecordThread class

3-3-9-4-1-2.Enumeration definition details

3-3-9-4-1-2-1. **RecEvent**

Table 89. RecEvent definition

Definition	enum RecEvent : uint8_t {
	EV_REC_START,
	EV_REC_STOP,
	};		

Description	・ Indicates the event ID of the event notification for the RecordThread class.		
Value	**Name**	**Value**	**Remark**
	EV_REC_START	0	Start saving file
	EV_REC_STOP	1	Stop saving file
Reference	3-3-6-3-13. PlFw::notifyEvent, 3-3-8-3-10. EvtThread::recvUserEvent, 3-3-9-2-1. EV_REC_START, 3-3-9-2-2. EV_REC_STOP		

3-3-9-4-2. Structure definition

3-3-9-4-2-1.List of structure definitions

Table 90. Structure definition

Name	Description
RecordParam	Saved setting information

3-3-9-4-2-2. Structure definition details

3-3-9-4-2-2-1.RecordParam

Table 91. RecordParam definition

Definition	struct RecordParam {
	std::filesystem::path path;
	uint32_t save_frames;
	uint16_t packing_frames;
	bool is_filt_med;
	bool is_filt_bil;
	bool is_filt_fly_p;
	bool is_crct_dist;
	};		

Description	・ Indicates the settings to be saved.		
Arguments	**Type**	**Name**	**Remark**
	std::filesystem::path	path	Destination directory
	uint32_t	save_frames	Number of frames to be saved
	uint16_t	packing_frames	Number of frames to be included in one file
	bool	is_filt_med	Median filter applied
	bool	is_filt_bil	Bilateral filter applied
	bool	is_filt_fly_p	Flying pixel filter applied
	bool	is_crct_dist	Whether distortion has been corrected
Reference	3-3-9-2-1. EV_REC_START		

3-3-10. LensConvThread class

3-3-10-1. Overview

Table 92. LensConvThread class overview

Class name	LensConvThread
Member Namespace	krm
Description	Thread that performs Lens type conversion processing For the methods that inherit from EvtThread, see the EvtThread class contents.
Additional buffer	Not used

3-3-10-2. List of methods

Table 93. LensConvThread method

Method name	Description
LensConvThread::LensConvThread	Constructor

3-3-10-3. Method details

3-3-10-3-1. LensConvThread::LensConvThread

Table 94. LensConvThread::LensConvThread method

Function	Constructor

Syntax	LensConvThread (bool enable_pcd = true, bool enable_distortion = true,)			
Description	・ Initializes the thread that performs lens-type conversion processing. ・ When the enable_pcd argument is enabled, point cloud data (pcd) is output to the image frame output by FrameData::getFrame(). PlFw::notifyEvent() cannot be used to switch point cloud output. ・ When the enable_distortion argument is enabled, depth and IR images in the image frame output by FrameData::getFrame() are output with distortion corrected. PlFw::notifyEvent(EV_LENS_DIST) can also be used to switch. ・ Even when distortion correction is disabled, distortion is corrected for depth images used for point cloud conversion. ・ If the input depth image is already distorted, distortion correction is disabled regardless of the enable_distortion setting.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	bool	enable_pcd	in	Enable Point Cloud Conversion true : Enabled (initial value) false : Disabled
	bool	enable_distortion	in	Enable distortion compensation true : Enabled (initial value) false : Disabled
Return value	None			
Sync/Async	Synchronous			

3-3-10-4. List of events

The following events can be used with PlFw::notifyEvent().

Table 95. LensConvThread class events

Event name	Description
EV_LENS_PSBL_DIST	Notification of whether distortion correction is possible
EV_LENS_DIST	ON/OFF of distortion correction
EV_LENS_PCD_KIND	Coordinate system switching of point group transformation
EV_LENS_PCD_ORG_POS	Notification of origin position and rotation information of world coordinate transformation
EV_LENS_PCD_COLOR	Settings for color information after point group transformation

3-3-10-4-1. EV_LENS_PSBL_DIST

Table 96. EV_LENS_PSBL_DIST overview

Information type	Acquisition of information about whether distortion correction is possible
Argument type	bool
Description	・ Acquisition of information about whether distortion correction is possible. ・ This is false if distortion correction is performed in the camera device.

3-3-10-4-2. EV_LENS_DIST

Table 97. EV_LENS_DIST overview

Information type	ON/OFF of distortion correction
Argument type	bool
Description	· If the argument is true, distortion correction is turned ON. If false, distortion correction is turned OFF. · The initial state is the value of enable_ distortion argument of LensConvThread::LensConvThread.

3-3-10-4-3. EV_LENS_PCD_KIND

Table 98. EV_LENS_PCD_KIND overview

Information type	Point cloud transformation coordinate system switching
Argument type	bool
Description	· If the argument is true, the point cloud transformation is performed in the world coordinate system. If false, the point cloud transformation is performed in the camera coordinate system. · The initial state is the point cloud transformation in the camera coordinate system.

3-3-10-4-4. EV_LENS_PCD_ORG_POS

Table 99. EV_LENS_PCD_ORG_POS overview

Information type	Notification of world coordinate transformation origin position and rotation information
Argument type	PosOrgRotation
Description	· Notification of origin position and rotation information during world coordinate system point group conversion. · The initial state is x = 0, y = 0, z = 0 for origin position and pitch = 0, roll = 0, yaw = 0 for rotation information.

3-3-10-4-5. EV_LENS_PCD_COLOR

Table 100. EV_LENS_PCD_COLOR overview

Information type	Settings for color information after point cloud conversion
Argument type	PcdColorKind
Description	· Sets the color information after point cloud conversion. · If PCD_COLOR_NONE is specified, no color information is set. · If PCD_COLOR_IR is specified, the color information of the point cloud data except for invalid points is replaced with the point cloud data including IR (.kind= PCD_IRXYZ). · The initial state is PCD_COLOR_NONE.

3-3-10-5. Status notification

When a status notification occurs during processing by LensConvThread, the following values are notified to the callback function of PlFw::getEvent() or PlFw::addPlProc().

Table 101. LensConvThread class status notifications

Notified value	Description
ERR_INVALID_PTR	Conversion processing failure due to invalid pointer
ERR_BAD_ARG	Conversion processing failure due to invalid image format
ERR_BAD_STATE	Conversion processing failure due to status transition error
ERR_SYSTEM	Conversion processing failure due to system error

3-3-10-6. Definition for LensConvThread class

The following shows the definitions used as event notifications for the LensConvThread classdescribed in LensConvThreadEvent.h.

3-3-10-6-1. Enumeration definition

3-3-10-6-1-1.List of enumeration definitions

Table 102. Enumeration definitions

Name	Description
LensConvEvent	Event ID of event notification for the LensConvThread class
PcdColorKind	Color information after point cloud conversion Type

3-3-10-6-1-2.Enumeration definition details

3-3-10-6-1-2-1.LensConvEvent

Table 103. LensConvEvent definition

Definition	enum LensConvEvent: uint8_t {
	EV_LENS_PSBL_DIST,
	EV_LENS_DIST,
	EV_LENS_PCD_KIND,
	EV_LENS_PCD_ORG_POS,
	EV_LENS_PCD_COLOR
	};		

Description	・ Indicates the event ID of the event notification for the LensConvThread class.		
Value	**Name**	**Value**	**Remark**
	EV_LENS_PSBL_DIST	0	Notification of whether distortion correction is possible
	EV_LENS_DIST	1	ON/OFF of distortion correction
	EV_LENS_PCD_KIND	2	Coordinate system switching of point group transformation
	EV_LENS_PCD_ORG_POS	3	Notification of origin position and rotation information of world coordinate transformation
	EV_LENS_PCD_COLOR	4	Settings for color information after point cloud conversion
Reference	3-3-6-3-13. PlFw::notifyEvent ()		

3-3-10-6-1-2-2.PcdColorKind

Table 104. PcdColorKind definition

Definition	enum PcdColorKind : uint8_t {

	PCD_COLOR_NONE, PCD_COLOR_IR };		
Description	・ Indicates the color information setting type after point cloud conversion.		
Value	**Name**	**Value**	**Remark**
	PCD_COLOR_NONE	0	Do not set color info
	PCD_COLOR_IR	1	Set IR in color info
Reference	3-3-6-3-13. PlFw::notifyEvent, 3-3-10-4-5. EV_LENS_PCD_COLOR		

3-4. Camera class

3-4-1. Overview

Table 105. Camera class overview

Header file	Camera class definition	Camera class definition
	Camera class common type definition	Camera class common type definition
	PlayBack function-only type definition	PlayBack function-only type definition
Member Namespace	krm	
Description	camera device control or a C++ class for playing a saved file	
Thread-safe	The Camera class is thread-safe. There is no need for exclusive processing on the user program side.	

3-4-2. Provided function

A summary of the functions provided by the Camera class is shown below.

3-4-2-1. Device control functions

A summary of the functions provided as controls for camera devices is shown below.

Table 106. Device control functions (Camera class)

Provided function	Functional overview
Device parameter acquisition	The following parameter information stored in the camera device is acquired. ・ Camera device information (Name, serial number, adjustment number) ・ Firmware version information in the device ・ FOV information of the camera device ・ External trigger type and signal offset information ・ PostFilter information of the camera device (for PostFilter processing) ・ Lens information of the camera device (for Lens-based conversion processing) ・ Available operation mode information ・ Ranging range, FPS, and receivable image type information for each operation mode ・ Output image format information

	· Light emission frequency information · AE function status and control interval information · RAW saturation threshold and IR disable threshold information · Interference prevention function information · Register information in the camera device · Operation mode information · Camera calibration Operation mode Image information · USB peripheral controller access key
Device parameter setting	Set the following parameters for the camera device. · External trigger type, signal offset information * · Operation mode switching * · Output image switching * · Number of flashes setting · AE function status and control interval information * · RAW saturation threshold and IR invalid threshold information · Interference prevention function information · Register settings in camera device * · Register write status reset setting * · Operation mode information * · Camera calibration Operation mode Image information * · USB peripheral controller access key · USB peripheral controller update mode setting *These parameter settings cannot be set during image output from the camera device.
Image Receive	The following images are received from the camera device. · Depth Image · IR Image · Sensor RAW image The type of image to receive is determined by the operation mode switch and output image switch. In the Camera class, only effective pixels are output for the depth image and IR image output from the camera module, and all pixels are output for the sensor RAW image. In addition, additional frame information is output for each image.

3-4-2-2. File playback function

The following shows an overview of the functions provided as extended functions for file playback.

Table 107. File playback function (Camera class)

Provided function	Functional overview
File playback	· This function uses a file saved by the Record class as input, and outputs the depth image, IR image, sensor RAW image, and additional frame information in the saved file. · The type of image that is output depends on the saved file.
Pause	· This function pauses or cancels the pause during file playback. · When the pause is canceled, playback returns to the playback state regardless of the previous state. · Frames are not output during pause.
Double Speed / Slow Playback	· This function allows you to play back a file at double speed (2x, 3x, 4x) or slow speed (1/2x, 1/3x, 1/4x).

	・ The maximum speed is 120fps, and the minimum speed is 10fps.
Forward/Rewind	・ This function forwards or rewinds a specified number of frames during file playback. ・ If the end frame is reached as a result of the forward operation, file playback stops. ・ If the end frame is reached as a result of the rewind operation, playback starts from the beginning.

3-4-2-2-1. Format version supported for file playback

The file playback function can be played only when the RecInfo.json format version saved by the Record clas is >Ver3.0.4. Files saved by this version or earlier cannot be played.

3-4-3. List of methods

Table 108. Camera class methods

Method name	Description
Camera	Constructor
~Camera	Destructor
getDeviceList	Obtaining a list of connected camera devices
openDevice	Device Open Processing
closeDevice	Device Close Processing
getProperty	Device Parameter Acquisition
setProperty	Device Parameter Setting
startCapture	Image Output Start
stopCapture	Image Output Stop
capture	Acquisition of an output image
cancel	Release of waiting for receiving an output image

3-4-3-1. State machine

The Camera class has the following state machine.

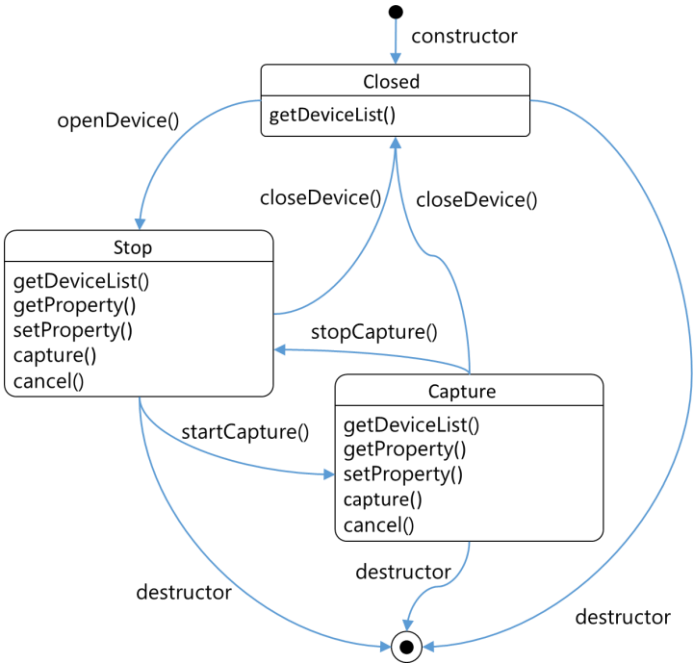


Figure 15. Camera class state machine

3-4-3-2. State machine (File Playback)

When a property command for file playback is used in the Camera class file playback, the following state transitions occur from the Capture state onward. Transitions from the Pause, Fast, and Slow states to the Stop, Closed, and End states are the same as the Capture state.

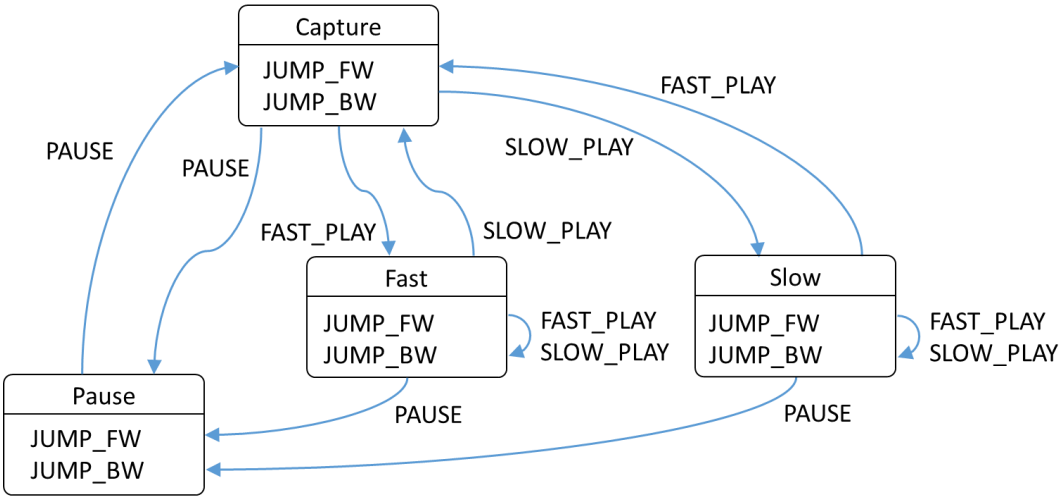


Figure 16. Camera class (file playback) state machine

This section describes the basic camera control sequence using the Camera class. For the following sequences, judgment of abnormal systems such as function return value judgment is omitted.

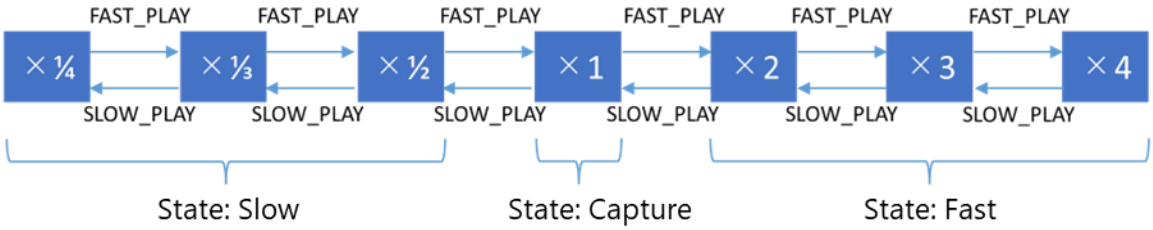


Figure 17. Playback speed transitions during file playback

3-4-4. Control sequence

This section describes the basic camera control sequence using the Camera class. For the following sequences, judgment of abnormal systems such as function return value judgment is omitted.

3-4-4-1. Initialization sequence

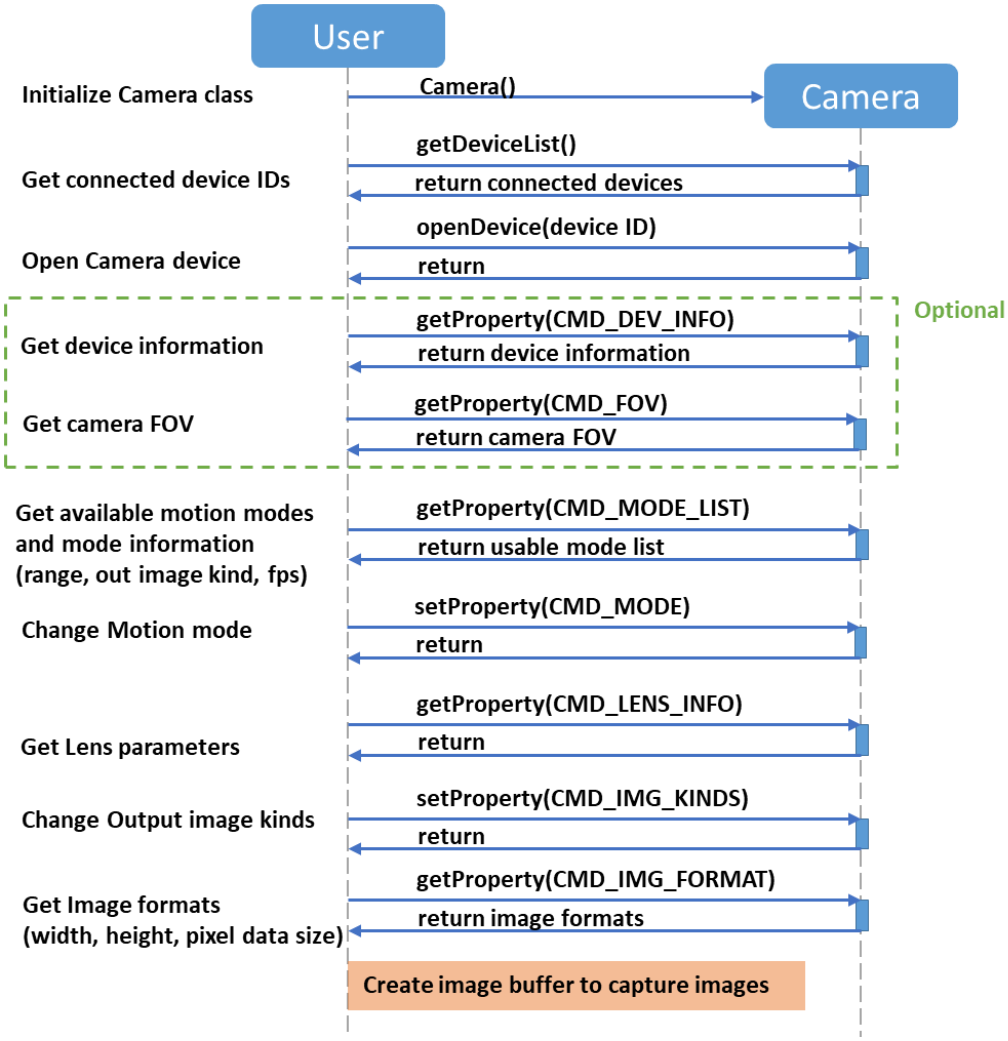


Figure 18. Initialization sequence diagram

3-4-4-2. Image reception sequence

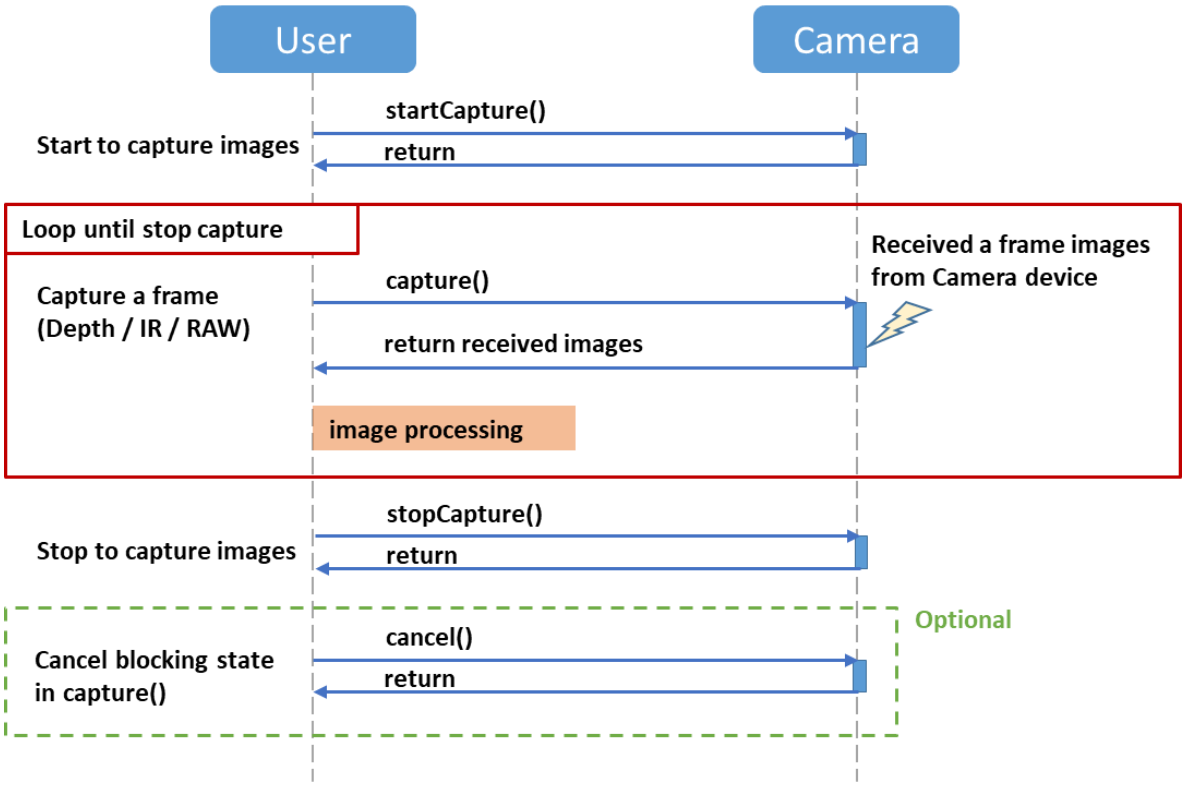


Figure 19. Image reception sequence diagram

3-4-4-3. End sequence

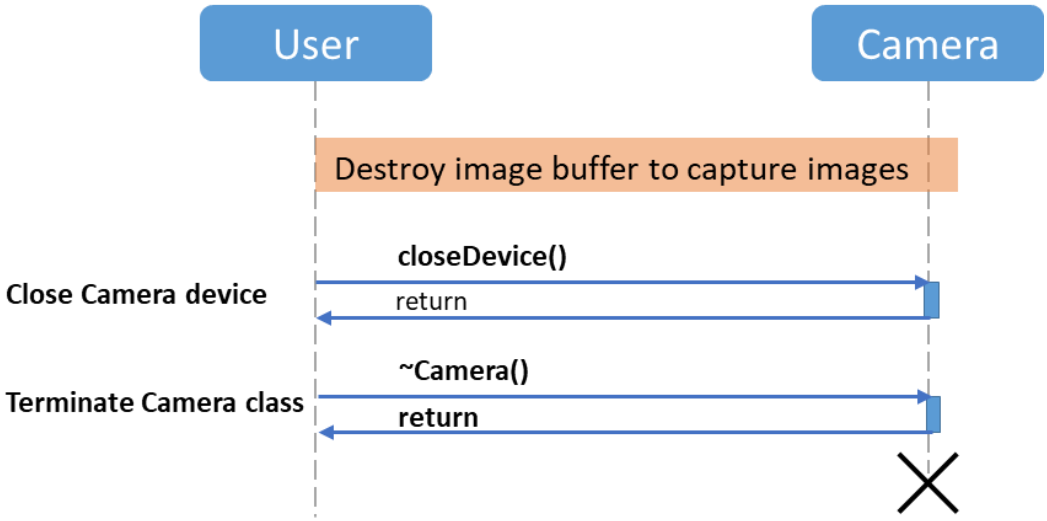


Figure 20. End sequence diagram

3-4-5. Method details

3-4-5-1. Camera

Table 109. Camera::Camera method

Function	Constructor			
Syntax	Camera (CameraType type, const void* param, Result& res)			
Description	・ Initializes camera control. ・ RET_OVER_RANGE is returned in the res argument if the type argument is out of the default. ・ When using a 3D ToF Camera (C11U), specify C11U_USB for the type argument and nullptr or OpInfo* for the param argument. If nullptr is specified, the normal operation mode (OP_NORMAL) is used. ・ When using the file playback function, specify PLAYBACK for the type argument and PlayBack::ConfigParam* for the param argument. You can also reset the playback target directory path by using setProperty (PlayBack::CMD_PLAY_TARGET).			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	CameraType	type	in	Camera type to be used
	const void*	param	in	Device parameter
	Result&	res	out	Return value
Return value	SUCCESS	0	Success	
	ERR_INVALID_PTR	3	Argument pointer is invalid (when PLAYBACK is specified)	
	ERR_OVER_RANGE	4	Camera type is invalid	
	ERR_BAD_STATE	6	Abnormal state transition	
	ERR_SYSTEM	12	Memory allocation failed	
Sync/Async	Synchronous			

3-4-5-2. ~Camera

Table 110. Camera::~Camera method

Function	Destructor			
Syntax	~Camera (void)			
Description	・ Performs camera control termination processing. ・ To release internally allocated resources during this processing, make sure that the destructor is executed when the camera is terminated. ・ After executing openDevice(), if closeDevice() is not executed, closeDevice() is executed within this method. ・ After executing startCapture(), if stopCapture() is not executed, stopCapture() is executed within this method.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	None			
Return value	None			

Sync/Async	Synchronous

3-4-5-3. getDeviceList

Table 111. Camera::getDeviceList method

Function	Acquire List of Connected Camera Devices				
Syntax	Result getDeviceList (std::vector<ConnDevice>& dev_list)				
Description	・ Acquires List of Connected Camera Devices.				
Arguments	Type		Name	in/out	Remark
	std::vector<ConnDevice>&		dev_list	out	Connected Camera Device List
Return value	SUCCESS	0	Success		
	ERR_NOT_EXIST	7	Target device is not connected		
	ERR_SYSTEM	12	System error (such as memory allocation failure)		
Sync/Async	Synchronous				

3-4-5-4. openDevice

Table 112. Camera::openDevice method

Function	Device Open Processing				
Syntax	Result openDevice (uint16_t dev_id)				
Description	・ Performs Open processing for the specified camera device. ・ In the dev_id argument, specify the device ID acquired from getDeviceList(). ・ If this method is called when the camera device has already been opened, ERR_BAD_STATE is returned. ・ If this method is called when the operation mode that cannot be set in the current camera device state was specified in the constructor, ERR_BAD_STATE is returned.				
Arguments	Type		Name	in/out	Remark
	uint16_t		dev_id	in	Device ID
Return value	SUCCESS	0	Success		
	ERR_BAD_STATE	6	State transition error		
	ERR_NOT_EXIST	7	Target device not connected		
	ERR_NOT_SUPPORT	11	The firmware version of the target camera is not the target camera.		
	ERR_SYSTEM	12	System error (memory allocation failure, etc.)		
Sync/Async	Synchronous				

3-4-5-5. closeDevice

Table 113. Camera::closeDevice method

Function	Device Close Processing
Syntax	Result closeDevice (void)

Description	• Close processing of the camera device opened by openDevice(). • If this method is called when openDevice() has not been executed, no processing is performed and SUCCESS is returned. • After startCapture() is executed, if stopCapture() has not been executed, stopCapture() is executed within this method.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	None			
Return value	SUCCESS	0	Success	
	ERR_SYSTEM	12	System error	
Sync/Async	Synchronous			

3-4-5-6. getProperty

Table 114. Camera::getProperty method

Function	Acquire Device Parameter			
Syntax	Result getProperty (uint16_t prop_cmd, void* param)			
Description	• This method gets information about a camera device opened by openDevice(). • The content of the param argument depends on the prop_cmd argument. • For details about the types and contents of the prop_cmd and param arguments, see Property commands (for camera devices) and Property command (for file playback only). • ERR_BAD_STATE is returned when this method is called without opening a camera device by openDevice().			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	uint16_t	prop_cmd	in	Property commands (for camera devices) Property command (for file playback only)
	void*	param	in,out	Acquisition parameter
Return value	SUCCESS	0	Success	
	ERR_INVALID_PTR	3	The argument pointer is invalid.	
	ERR_OVER_RANGE	4	The specified value is outside the specifiable range (prop_cmd)	
	ERR_BAD_STATE	6	State transition error	
	ERR_TIMEOUT	8	A timeout occurred during communication between camera devices.	
	ERR_NOT_SUPPORT	11	Unsupported function	
	ERR_SYSTEM	12	System error	
Sync/Async	Synchronous			

3-4-5-7. setProperty

Table 115. Camera::setProperty method

Function	Device Parameter Setting
Syntax	Result setProperty (

	uint16_t prop_cmd, const void* param = nullptr)
Description	・ Sets the camera device opened by openDevice(). ・ The contents of the param argument depend on the prop_cmd argument. ・ For details about the types and contents of the prop_cmd and param arguments, see Property commands (for camera devices) and Property command (for file playback only). ・ ERR_BAD_STATE is returned if this method is called when the camera device is not opened by openDevice(). ・ If this method is called when an image is being output from the camera device by startCapture(), check the contents of each property command.

Arguments	**Type**	**Name**	**in/out**	**Remark**
	uint16_t	prop_cmd	in	Property commands (for camera devices) Property command (for file playback only)
	const void*	param	in	Configuration parameter

Return value			
	SUCCESS	0	Success
	ERR_INVALID_PTR	3	Invalid argument pointer
	ERR_OVER_RANGE	4	Set value is outside the specifiable range.
	ERR_BAD_ARG	5	Other or invalid argument
	ERR_BAD_STATE	6	Abnormal state transition
	ERR_NOT_EXIST	7	Target directory file does not exist.
	ERR_TIMEOUT	8	A timeout occurred during communication between camera devices.
	ERR_NOT_SUPPORT	11	Unsupported function
	ERR_SYSTEM	12	System error
Sync/Async	Synchronous		

3-4-5-8. startCapture

Table 116. Camera::startCapture method

Function	Start image output
Syntax	Result startCapture (void)
Description	・ Starts image output of the camera device opened by openDevice(). ・ After image output starts, the image is received by capture(). ・ If this method is called when openDevice() has not been executed, ERR_BAD_STATE is returned. ・ If this method is called when image output from the camera device has already been executed, no processing is performed and SUCCESS is returned.

Arguments	**Type**	**Name**	**in/out**	**Remark**
	None			

Return value			
	SUCCESS	0	Success
	ERR_BAD_STATE	6	Status transition error
	ERR_NOT_EXIST	7	Target device disconnected (no playback target file)
	ERR_SYSTEM	12	System error

Sync/Async	Synchronous

3-4-5-9. stopCapture

Table 117. Camera::stopCapture method

Function	Stop Image Output			
Syntax	Result stopCapture (void)			
Description	・ Stops image output for a camera device opened by openDevice(). ・ If this method is called when openDevice() has not been executed, ERR_BAD_STATE is returned. ・ If this method is called when startCapture() has not been executed, no processing is performed and SUCCESS is returned.			
Arguments	Type	Name	in/out	Remark
	None			
Return value	SUCCESS	0	Success	
	ERR_BAD_STATE	6	Status transition error	
	ERR_SYSTEM	12	System error	
Sync/Async	Synchronous			

3-4-5-10. capture

Table 118. Camera::capture method

Function	Acquire output image
Syntax	Result capture (Frame& frame, bool block = true)
Description	・ Gets the output image of the camera device opened by openDevice(). ・ If this method is called when the camera device has not been opened by openDevice(), ERR_BAD_STATE is returned. ・ For the receive data buffer (ImageData.data) of each image type in the frame argument, the caller must allocate an area of the size that can be acquired by getProperty(CMD_IMG_FORMAT). If no receive buffer is allocated, ERR_EMPTY is returned. ・ If this method is not called after image output is started by startCapture(), the oldest frame is discarded. ・ The receive wait operation varies depending on the block argument as follows. [When the block argument is true] ・ This method waits for reception until an output image is received. ・ When cancel() is called while waiting for reception, the method cancels the wait and returns CANCELED. ・ If an image is not received within one second while waiting for reception, reception wait times out and ERR_TIMEOUT is returned. ・ The method waits for reception even if image output has not started by startCapture(). In this case, reception wait times out do not occur. [If the block argument is false]

	・ If there is no output image, the method does not wait for reception and ERR_NOT_EXIST is returned.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	Frame&	frame	in,out	Incoming Frames
	bool	block	in	Receive-wait flag true : Waits for reception false : Does not wait for reception
Return value	SUCCESS	0	Success	
	CANCELED	1	Wait status is released.	
	REACH_EOF	2	End of file reached (PLAYBACK only)	
	ERR_BAD_STATE	6	Status transition error	
	ERR_NOT_EXIST	7	There is no output image.	
	ERR_TIMEOUT	8	Timeout occurred while waiting for reception.	
	ERR_EMPTY	9	Empty buffer, etc.	
	ERR_SYSTEM	12	System error	
Sync/Async	Synchronous			

3-4-5-11. cancel

Table 119. Camera::cancel method

Function	Release waiting for output image			
Syntax	Result cancel (void)			
Description	・ Cancels the capture() waiting state. ・ When the wait state is released, CANCELED is returned from capture(). ・ ERR_BAD_STATE is returned if this method is called when the camera device has not been opened by openDevice().			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	None			
Return value	SUCCESS	0	Success	
	ERR_BAD_STATE	6	State transition error	
	ERR_SYSTEM	12	System error	
Sync/Async	Synchronous			

3-4-6. Property commands (for camera devices)

3-4-6-1. List of property commands

The following shows the property commands used in Camera:: getProperty() and Camera:: setProperty() of the Camera class, and their availability in Camera:: getProperty() and Camera:: setProperty() in each operation mode.

Table 120. Property commands (for camera device control)

Command name	Information	Normal operation mode		Camera calibration operation mode	
		get	set	get	set
CMD_DEV_INFO	Device	✓	N/A	N/A	N/A
CMD_FOV	FOV	✓	N/A	N/A	N/A
CMD_EXT_TRG_TYPE	External trigger type	✓	✓	N/A	N/A
CMD_EXT_TRG_OFFSET	External trigger signal offset	✓	✓	N/A	N/A
CMD_MODE_LIST	Operation mode list	✓	N/A	N/A	N/A
CMD_MODE	Operating mode	✓	✓	✓	N/A
CMD_IMG_KINDS	Output image type	✓	✓	N/A	N/A
CMD_IMG_FORMAT	Image format	✓	N/A	✓	N/A
CMD_POSTFILT_INFO	PostFilter processing	✓	N/A	N/A	N/A
CMD_LENS_INFO	Lens conversion processing parameter	✓	N/A	N/A	N/A
CMD_LIGHT_TIMES	light emission count	✓	✓	N/A	N/A
CMD_AE_STATE	AE function status	✓	✓	N/A	N/A
CMD_AE_INTERVAL	AE function control interval	✓	✓	N/A	N/A
CMD_RAW_SAT_TH	RAW saturation threshold	✓	✓	N/A	N/A
CMD_IR_DARK_TH	IR invalid threshold	✓	✓	N/A	N/A
CMD_INT_SUPP_INFO	Interference cancellation function	✓	✓	N/A	N/A

3-4-6-1-1. CMD_DEV_INFO

Table 121. CMD_DEV_INFO overview

Information type	Device information
GET / SET	GET
Argument type	DeviceInfo
Description	・ The device information of the camera device opened by openDevice() is acquired.

3-4-6-1-2. CMD_FOV

Table 122. CMD_FOV overview

Information type	FOV information
GET / SET	GET
Argument type	CamFov
Description	・ The FOV information of the camera device opened by openDevice() is acquired.

3-4-6-1-3. CMD_EXT_TRG_TYPE

Table 123. CMD_EXT_TRG_TYPE overview

Information type	External trigger type information
GET / SET	GET/SET
Argument type	ExtTriggerType

Description	• Gets or sets external trigger type information of the camera device opened by openDevice(). • ERR_BAD_STATE is returned if the external trigger type is set when image output from the camera device is executed by startCapture(). • ERR_BAD_STATE is returned if the current external trigger type is set to Slave (EXT_TRG_SLAVE).

3-4-6-1-4. CMD_EXT_TRG_OFFSET

Table 124. CMD_EXT_TRG_OFFSET overview

Information type	External trigger signal offset information
GET / SET	GET/SET
Argument type	uint8_t
Description	• Gets or sets external trigger signal offset information of the camera device opened by openDevice(). • ERR_BAD_STATE is returned if the external trigger signal offset information is set by startCapture().

3-4-6-1-5. CMD_MODE_LIST

Table 125. CMD_MODE_LIST overview

Information type	Operation mode list information
GET / SET	GET/SET
Argument type	ModeList
Description	• Gets list information of the operation modes available for the camera device opened by openDevice().

3-4-6-1-6. CMD_MODE

Table 126. CMD_MODE overview

Information type	Operation mode information
GET / SET	GET/SET
Argument type	uint8_t
Description	• Gets or sets the current operation mode ID for the camera device opened by openDevice(). • For setting, specify the operation mode ID acquired from CMD_MODE_LIST. • The initial operation mode after executing openDevice() is the first operation mode ID acquired from CMD_MODE_LIST. • ERR_BAD_STATE is returned when set by startCapture() while image output from the camera device is being performed.

3-4-6-1-7. CMD_IMG_KINDS

Table 127. CMD_IMG_KINDS overview

Information type	Output image type information
GET / SET	GET/SET
Argument type	ImgOutKind
Description	• This command gets or sets the current acquired image type for the camera device opened by openDevice() • For setting, specify the received image type from the image type information that can be acquired by CMD_MODE_LIST.

	• The initial operation mode after executing openDevice() is all image types that can be acquired by CMD_MODE_LIST. • The initial output image type after executing openDevice() and setProperty(CMD_MODE) is determined by the camera device. • ERR_BAD_STATE is returned if this is set while image output from the camera device is being performed by startCapture().

3-4-6-1-8. CMD_IMG_FORMAT

Table 128. CMD_IMG_FORMAT overview

Information type	Image format information
GET / SET	GET
Argument type	ImageFormats
Description	• Gets the image format information of the currently acquired image type for the camera device opened by openDevice(). • You can acquire the image format information of the image type that can be acquired by getProperty(CMD_IMG_KINDS).

3-4-6-1-9. CMD_POSTFILT_INFO

Table 129. CMD_POSTFILT_INFO overview

Information type	PostFilter processing information
GET / SET	GET
Argument type	PostFiltInfo
Description	• Gets PostFilter processing information for the camera device that is opened by openDevice().

3-4-6-1-10. CMD_LENS_INFO

Table 130. CMD_LENS_INFO overview

Information type	Lens conversion processing parameter
GET / SET	GET
Argument type	LensInfo
Description	• Gets the parameters for Lens-based conversion of the camera device opened by openDevice().

3-4-6-1-11. CMD_LIGHT_TIMES

Table 131. CMD_LIGHT_TIMES overview

Information type	Light emission count information
GET / SET	GET/SET
Argument type	LightTimesInfo
Description	• Gets or sets the number of flashes for the camera device opened by openDevice(). • The minimum number of flashes (min), maximum number of flashes (max), and current number of flashes (count) that can be set can be acquired. • The minimum number of flashes (min) and maximum number of flashes (max) values are not referred to when setting. Only the number of flashes (count) value can be set. • When the operation mode is changed with setProperty(CMD_MODE), the

	number of flashes is reset to the default value for each operation mode.

3-4-6-1-12. CMD_AE_STATE

Table 132. CMD_AE_STATE overview

Information type	AE function status information
GET / SET	GET/SET
Argument type	bool
Description	• Gets or sets the AE function enable/disable information in the camera device opened by openDevice(). • The AE function is always disabled when the minimum number of flashes (min) and maximum number of flashes (max) in the flash count information (LightTimesInfo) are the same. • ERR_BAD_STATE is returned when the image output from the camera device is set by startCapture().

3-4-6-1-13. CMD_AE_INTERVAL

Table 133. CMD_AE_INTERVAL overview

Information type	AE function control interval information
GET / SET	GET/SET
Argument type	AEIntervalInfo
Description	• Gets or sets the control interval information when the AE function is enabled for the camera device opened by openDevice(). • The minimum AE control interval that can be set (min), the maximum AE control interval (max), and the current AE control interval (interval) can be acquired. • The minimum AE control interval (min) and maximum AE control interval (max) values are not referenced when setting. Only the AE control interval (interval) value can be set. • When the operation mode is switched with setProperty(CMD_MODE), the AE control interval before the operation mode was switched is inherited. • ERR_BAD_STATE is returned when the image output from the camera device is set with startCapture().

3-4-6-1-14. CMD_RAW_SAT_TH

Table 134. CMD_RAW_SAT_TH overview

Information type	RAW saturation threshold information
GET / SET	GET/SET
Argument type	SignalThresholdInfo
Description	• Gets or sets external trigger type information of the camera device opened by openDevice(). • At the time of acquisition, the minimum saturation threshold (min), maximum saturation threshold (max) and current saturation threshold (threshold) that can be set can be acquired. • At the time of setting, the minimum saturation threshold (min) and maximum saturation threshold (max) values are not referenced. Only the saturation threshold value is set.

	・ When the operation mode is switched using setProperty(CMD_MODE), the saturation threshold value is reset to the initial value of each operation mode.

3-4-6-1-15. CMD_IR_DARK_TH

Table 135. CMD_IR_DARK_TH overview

Information type	IR invalid threshold information
GET / SET	GET/SET
Argument type	SignalThresholdInfo
Description	・ Gets or sets external trigger type information of the camera device opened by openDevice(). ・ At acquisition, the minimum disabled threshold (min), maximum disabled threshold (max) and the current disabled threshold (threshold) can be acquired. ・ The minimum invalid threshold (min) and maximum invalid threshold (max) values are not referenced when setting. Only the invalid threshold value is set. ・ When the operation mode is changed by setProperty(CMD_MODE), the invalid threshold value is reset to the initial value of each operation mode.

3-4-6-1-16. CMD_INT_SUPP_INFO

Table 136. CMD_INT_SUPP_INFO overview

Information type	Interference prevention function information
GET / SET	GET/SET
Argument type	IntSuppInfo
Description	・ Gets or sets the anti-interference function information (Operating mode, manual mode parameters and automatic parameters 1 ~ 3) of the camera device opened by openDevice(). ・ At acquisition, the minimum value (min), maximum value (max) and current parameter value (value) that can be set for a numeric parameter can be acquired. ・ The minimum value (min) and maximum value (max) values are not referred when setting. Only the value of the parameter value is set.

3-4-7. Property command (for file playback only)

3-4-7-1. List of property commands

The following shows the list of property commands that can be used with the file playback function, and whether they can be used with Camera:: getProperty() and Camera:: setProperty().

Table 137. Property commands (for file playback only)

Command name	Information	get	set
CMD_DEV_INFO	Device	✓	N/A
CMD_FOV	FOV	✓	N/A
CMD_EXT_TRG_TYPE	External trigger type	N/A	N/A
CMD_EXT_TRG_OFFSET	External trigger signal offset	N/A	N/A
CMD_MODE_LIST	Operation mode list	✓	N/A

CMD_MODE	Operating mode	✓	✓
CMD_IMG_KINDS	Output image type	✓	N/A
CMD_IMG_FORMAT	Image format	✓	N/A
CMD_POSTFILT_INFO	PostFilter processing	✓	N/A
CMD_LENS_INFO	Lens conversion processing parameter	✓	N/A
CMD_LIGHT_TIMES	Light emission count	N/A	N/A
CMD_AE_STATE	AE function status	N/A	N/A
CMD_AE_INTERVAL	AE function control interval	N/A	N/A
CMD_RAW_SAT_TH	RAW saturation threshold	N/A	N/A
CMD_IR_DARK_TH	IR invalid threshold	N/A	N/A
CMD_INT_SUPP_INFO	Interference prevention function	N/A	N/A
PlayBack::CMD_PLAY_TARGET	File target directory information	✓	✓
PlayBack::CMD_PLAY_TIME	Playback time information	✓	✓
PlayBack::CMD_PLAY_STATUS	Playback status information	✓	N/A
PlayBack::CMD_PAUSE	Playback pause control	N/A	✓
PlayBack::CMD_FAST_PLAY	Double speed playback control	N/A	✓
PlayBack::CMD_SLOW_PLAY	Slow playback control	N/A	✓
PlayBack::CMD_JUMP_FW	Forward control	N/A	✓
PlayBack::CMD_JUMP_BW	Rewind control	N/A	✓

The following describes the details of the property commands that can be used only with the file playback function.

3-4-7-1-1. PlayBack::CMD_PLAY_TARGET

Table 138. PlayBack::CMD_PLAY_TARGET overview

Information type	Directory information for file playback
GET / SET	GET/SET
Argument type	std::filesystem::path
Description	· Gets or sets the directory to be played back. · ERR_NOT_EXIST is returned if there is no directory file that can be played back in the directory at the time of setting. · ERR_NOT_SUPPORT is returned if the "Format version supported for file playback" is not a supported version. · ERR_BAD_STATE is returned when this is set with startCapture() in a state where file playback is being performed.

3-4-7-1-2. PlayBack::CMD_PLAY_TIME

Table 139. PlayBack::CMD_PLAY_TIME overview

Information type	Playback time information
GET / SET	GET/SET
Argument type	PlayBack::PlayTime
Description	· Gets or sets the current playback time. · When set, only current in the argument is referenced, not the total value. · When set in the pause state, a frame of the specified time is output by capture(), and then the pause state is set.

	· If no playable directory file is set, ERR_NOT_EXIST is returned. · If the specified time exceeds the total playback time of the target, ERR_OVER_RANGE is returned. · If set in the stop state, ERR_BAD_STATE is returned.

3-4-7-1-3. PlayBack::CMD_PLAY_STATUS

Table 140. PlayBack::CMD_PLAY_STATUS overview

Information type	Playback state information
GET / SET	GET
Argument type	PlayBack::PlayStatus
Description	· Gets the file playback status. · During double-speed or slow playback, the received frame rates can be acquired with getProperty(CMD_MODE_LIST). During double-speed or slow playback, the received frame rates are 2x, 3x, and 4x for double-speed playback, and 1/2x, 1/3x, and 1/4x for slow playback. Therefore, use this information to determine the received frame rate during double-speed or slow playback. · If the playback status is other than double-speed or slow playback, the received frame rate will be the same as that acquired with getProperty(CMD_MODE_LIST).

3-4-7-1-4. PlayBack::CMD_PAUSE

Table 141. PlayBack::CMD_PAUSE overview

Information type	Playback pause control
GET / SET	SET
Argument type	None (nullptr)
Description	· When the playback state, double-speed playback state, or slow playback state is set, the playback mode changes to the pause state. · If set in the pause state, it returns to the play state. · If set in the stop state, ERR_BAD_STATE is returned.

3-4-7-1-5. PlayBack::CMD_FAST_PLAY

Table 142. PlayBack::CMD_FAST_PLAY overview

Information type	Double speed playback control
GET / SET	SET
Argument type	None (nullptr)
Description	· When set in the playback state, the system switches to the double speed playback state (2x speed). · When set in the double speed playback state, the system switches to 3x speed when set in 2x speed, and to 4x speed when set in 3x speed. · ERR_OVER_RANGE is returned when 4x speed or the received frame rate exceeds 120fps. · When set in the slow playback state, the system switches to 1/3x when set in 1/4x speed, 1/2x when set in 1/3x speed, and to play when set in 1/2x speed. · ERR_BAD_STATE is returned when set in the stop or pause state.

3-4-7-1-6. PlayBack::CMD_SLOW_PLAY

Table 143. PlayBack::CMD_SLOW_PLAY overview

Information type	Slow playback control
GET / SET	SET
Argument type	None (nullptr)
Description	• When set in the playback state, it transitions to the slow playback state (1/2x). • When set in the slow playback state, it transitions to 1/3x when set in 1/2x, and to 1/4x when set in 1/3x. • ERR_OVER_RANGE is returned when set in 1/4x or when the received frame rate is less than 10fps. • When set in the double playback state, it transitions to 3x when set in 4x, 2x when set in 3x, and 2x when set in 2x. • ERR_BAD_STATE is returned when set in the stop or pause state.

3-4-7-1-7. PlayBack::CMD_JUMP_FW

Table 144. PlayBack::CMD_JUMP_FW overview

Information type	Forward control
GET / SET	SET
Argument type	uint32_t
Description	• Executes forward by the frames specified by the argument. • When set in the pause state, the frame after the forward processing is output by capture() and then the pause state is set. • When the end frame is exceeded as a result of the forward processing, the stop state is changed and REACH_EOF is returned by capture(). • When set in the stop state, ERR_BAD_STATE is returned.

3-4-7-1-8. PlayBack::CMD_JUMP_BW

Table 145. PlayBack::CMD_JUMP_BW overview

Information type	Rewind control
GET / SET	Setting only
Argument type	uint32_t
Description	• Rewinds the frame specified by the argument. • When set in the pause state, the rewound frame is output by capture() and the pause state is reached. • If the rewound frame is before the first frame, the playback starts from the first frame. • When set in the stop state, ERR_BAD_STATE is returned.

3-4-8. Definition for camera class

The following definitions are used by the Camera class, as described in CameraType.h:

3-4-8-1. Enumeration definition

3-4-8-1-1. List of enumeration definitions

Table 146. Enumeration definitions

Name	Description
CameraType	Camera device type
PropCmd	List of property commands
OperationMode	Camera device operation mode
ExtTriggerType	External trigger type
ImgOutKind	Output image type combination information
CamPrmKind	Camera parameter type
RegDevType	Register control device type
IntSuppModeType	Interference prevention function mode

3-4-8-1-2. Enumeration definition details

3-4-8-1-2-1.CameraType

Table 147. CameraType definition

Definition	enum CameraType {
	C11U_USB = 0,
	PLAYBACK,
	};		

Description	・ Indicates the type of camera device to be used.		
Value	**Name**	**Value**	**Remark**
	C11U_USB	0	3D ToF Camera (C11U)
	PLAYBACK	1	For file playback
Reference	3-4-5-1. Camera		

3-4-8-1-2-2.PropCmd

Table 148. PropCmd definition

Definition	enum PropCmd : uint16_t { CMD_DEV_INFO = 0, CMD_FOV, CMD_EXT_TRG_TYPE, CMD_EXT_TRG_OFFSET, CMD_MODE_LIST, CMD_MODE, CMD_IMG_KINDS, CMD_IMG_FORMAT, CMD_POSTFILT_INFO, CMD_LENS_INFO, CMD_LIGHT_TIMES, CMD_AE_STATE, CMD_AE_INTERVAL, CMD_RAW_SAT_TH, CMD_IR_DARK_TH, CMD_INT_SUPP_INFO,

	CMD_CAM_PRM, CMD_REG_DEVS, CMD_REG, CMD_REG_LIST, CMD_REG_REFRESH, CMD_OP_MODE, CMD_CALB_OP_INFO, CMD_USB_PC_ACC_KEY, CMD_USB_PC_UPDATE, CMD_MAX };
Description	・ Indicates the command used as a List of property commands

Value

Name	Value	Remark
CMD_DEV_INFO	0	Device
CMD_FOV	1	FOV
CMD_EXT_TRG_TYPE	2	External trigger type
CMD_EXT_TRG_OFFSET	3	External trigger signal offset
CMD_MODE_LIST	4	Operation mode list
CMD_MODE	5	Operating mode
CMD_IMG_KINDS	6	Output image type
CMD_IMG_FORMAT	7	Image format
CMD_POSTFILT_INFO	8	PostFilter processing
CMD_LENS_INFO	9	Lens conversion processing parameter
CMD_LIGHT_TIMES	10	light emission count
CMD_AE_STATE	11	AE function status
CMD_AE_INTERVAL	12	AE function control interval
CMD_RAW_SAT_TH	13	RAW saturation threshold
CMD_IR_DARK_TH	14	IR invalid threshold
CMD_INT_SUPP_INFO	15	Interference cancellation function
CMD_CAM_PRM	16	Camera parameter
CMD_REG_DEVS	17	Register controllable device
CMD_REG	18	Register
CMD_REG_LIST	19	Register list
CMD_REG_REFRESH	20	Refresh register write status
CMD_OP_MODE	21	Operating mode
CMD_CALB_OP_INFO	22	Image information in camera calibration operation mode
CMD_USB_PC_ACC_KEY	23	USB peripheral controller access key
CMD_USB_PC_UPDATE	24	USB peripheral controller update mode
CMD_MAX	25	Common command value upper limit

Reference	3-4-5-6. getProperty, 3-4-5-7. setProperty, 3-4-6. Property commands (for camera devices), 3-4-7. Property command (for file playback only)

3-4-8-1-2-3.OperationMode

Table 149. OperationMode definition

Definition	enum OperationMode {
	OP_NORMAL = 0,
	OP_CALIB,
	};		

Description	· Indicates the operating mode of the camera device.		
Value	**Name**	**Value**	**Remark**
	OP_NORMAL	0	Normal operation mode
	OP_CALIB	1	Camera calibration operation mode
Reference	3-4-8-2-2-3. OpInfo		

3-4-8-1-2-4.ExtTriggerType

Table 150. ExtTriggerType definition

Definition	enum ExtTriggerType : uint8_t {
	EXT_TRG_STANDALONE = 1U,
	EXT_TRG_SLAVE,
	EXT_TRG_MASTER,
	};		

Description	· Indicates the external trigger type of the camera device.		
Value	**Name**	**Value**	**Remark**
	EXT_TRG_STANDALONE	1	Standalone
	EXT_TRG_SLAVE	2	Secondary (Slave)
	EXT_TRG_MASTER	3	Primary (Mater)
Reference	3-4-6-1-3. CMD_EXT_TRG_TYPE		

3-4-8-1-2-5.ImgOutKind

Table 151. ImgOutKind definition

Definition	enum ImgOutKind {
	OUT_IMG_DEPTH = 0,
	OUT_IMG_IR,
	OUT_IMG_DEPTH_IR,
	OUT_IMG_DEPTH_IR_RAW,
	OUT_IMG_RAW
	};		

Description	· Indicates image type combination information output from the camera device.		
Value	**Name**	**Value**	**Remark**
	OUT_IMG_DEPTH	0	Depth Image Output
	OUT_IMG_IR	1	IR Image Output
	OUT_IMG_DEPTH_IR	2	Depth image + IR image output
	OUT_IMG_DEPTH_IR_RAW	3	Depth image + IR image + Sensor Gate RAW image output
	OUT_IMG_RAW	4	Sensor RAW image output
Reference	3-4-8-2-2-8. ModeInfo, 3-4-6-1-7. CMD_IMG_KINDS, 3-4-8-2-2-2. CalbOpInfo		

3-4-8-1-2-6.CamPrmKind

Table 152. CamPrmKind definition

Definition	enum CamPrmKind : uint8_t {
	CAM_PRM_ALL,
	CAM_PRM_PART,
	CAM_PRM_ARTIX_CONF,
	CAM_PRM_FIRM_2
	};		

Description	· Indicates the type of internal camera parameter.		
Value	**Name**	**Value**	**Remark**
	CAM_PRM_ALL	0	All parameters
	CAM_PRM_PART	1	Partial parameters
	CAM_PRM_ARTIX_CONF	2	Flash ROM area 1
	CAM_PRM_FIRM_2	3	Flash ROM area 2
Reference	3-4-8-2-2-12. CamPrmRequest		

3-4-8-1-2-7.RegDevType

Table 153. RegDevType definition

Definition	enum RegDevType {
	REG_IMG_SENSOR = 0,
	REG_IMG_PROCESSOR,
	};		

Description	· Registers Indicates the type of device to be controlled.		
Value	**Name**	**Value**	**Remark**
	REG_IMG_SENSOR	0	Image sensor
	REG_IMG_PROCESSOR	1	Image processing device
Reference	3-4-8-2-2-13. RegDevInfo, 3-4-8-2-2-14. RegInfo, 3-4-8-2-2-15. RegList		

3-4-8-1-2-8.IntSuppModeType

Table 154. IntSuppModeType definition

Definition	enum IntSuppModeType : uint8_t {
	INT_SUPP_MODE_OFF = 0, */
	INT_SUPP_MODE_MANUAL,
	INT_SUPP_MODE_AUTO,
	};		

Description	· Indicates the type of interference prevention mode. · Automatic mode is an experimental function and is not recommended for use.		
Value	**Name**	**Value**	**Remark**
	INT_SUPP_MODE_OFF	0	Off
	INT_SUPP_MODE_MANUAL	1	Manual mode
	INT_SUPP_MODE_AUTO	2	Automatic mode (Experimental function)
Reference	3-4-8-2-2-17. IntSuppInfo		

3-4-8-2. Structure definition

3-4-8-2-1. List of structure definitions

Table 155. Structure definitions

Name	Description
ConnDevice	Connected device information
CalbOpInfo	Receive setting information in camera calibration operation mode
OpInfo	Camera device operation mode
DeviceInfo	Camera device information
PostFiltInfo	PostFilter processing information
LensInfo	Parameter information used in Lens-based conversion processing
CamFov	FOV information of camera device
ModeInfo	Operation mode information
LightTimesInfo	Number of times of light emission information
AEIntervalInfo	Control interval information when the AE function is enabled
SignalThresholdInfo	RAW saturation threshold information, IR disable threshold information
CamPrmRequest	Parameter information in the camera
RegDevInfo	Camera device register information
RegInfo	Register information
RegList	Register information (consecutive addresses)
IntSuppParamInfo	Anti-interference mode parameter information
IntSuppInfo	Anti-interference function information
UsbPCAccKey	USB peripheral controller access key information

3-4-8-2-2. Structure definition details

3-4-8-2-2-1.ConnDevice

Table 156. ConnDevice definition

Definition	struct ConnDevice {
	uint16_t id;
	std::string name;
	};		

Description	・ Indicates information about the connected device.		
Arguments	**Type**	**Name**	**Remark**
	uint16_t	id	Device ID
	std::string	name	Device Name
Reference	3-4-5-3. getDeviceList		

3-4-8-2-2-2.CalbOpInfo

Table 157. CalbOpInfo definition

Definition	struct CalbOpInfo { ImgOutKind kind; uint16_t image_w; uint16_t image_h;

	uint8_t num_raw; uint16_t fps; };		
Description	・ Indicates reception setting information in camera calibration operation mode.		
Arguments	**Type**	**Name**	**Remark**
	ImgOutKind	kind	Received image type
	uint16_t	image_w	Sensor output image width [pixel]
	uint16_t	image_h	Sensor output image height [pixel]
	uint8_t	num_raw	Number of RAW images
	uint16_t	fps	Received frame rate [fps × 100]
Reference	3-4-8-1-2-5. ImgOutKind, 3-4-8-2-2-3. OpInfo		

3-4-8-2-2-3.OpInfo

Table 158. OpInfo definition

Definition	struct OpInfo {
	OperationMode mode;
	CalbOpInfo calib_prm;
	};		

Description	・ Indicates operation mode information for the camera device. ・ calib_prm is valid only when mode is OP_CALB.		
Arguments	**Type**	**Name**	**Remark**
	OperationMode	mode	Operation Mode
	CalbOpInfo	calib_prm	Parameters for camera calibration operation mode
Reference	3-4-5-1. Camera, 3-4-8-2-2-2. CalbOpInfo		

3-4-8-2-2-4.DeviceInfo

Table 159. DeviceInfo definition

Definition	struct DeviceInfo {
	uint32_t hw_kind;
	uint32_t serial_no;
	Version map_ver;
	uint32_t adjust_no;
	Version firm_ver;
	uint16_t ld_wave;
	uint16_t ld_enable;
	uint16_t correct_calib;
	};		

Description	・ Indicates device information for the camera device.		
Arguments	**Type**	**Name**	**Remark**
	uint32_t	hw_kind	HW Model Number Upper 16 bits: Sensor Model Number Lower 16 bits: Lens Model Number
	uint32_t	serial_no	Device Identification Number
	Version	map_ver	In-camera setting MAP version
	uint32_t	adjust_no	Adjustment number

	Version	firm_ver	Camera Firmware version
	uint16_t	ld_wave	Light source wavelength [nm]
	uint16_t	ld_enable	Effective information of each light source (number of lights information) Effective information of each light source is input in order from the least significant bit. (0b: Disabled, 1b: Enabled) [0] LD1, [1] LD2, [2] LD3 ...
	uint16_t	correct_calib	Compensation Calibration Revision
Reference	3-2-3-2-1. Version, 3-4-6-1-1. CMD_DEV_INFO		

3-4-8-2-2-5.PostFiltInfo

Table 160. LensInfo definition

Definition	struct PostFiltInfo {
	bool cam_med_filt;
	bool cam_bil_filt;
	bool cam_fly_p_filt;
	};		

Description	・ Indicates PostFilter processing information.		
Arguments	**Type**	**Name**	**Remark**
	bool	cam_med_filt	Median filter implemented in camera device
	bool	cam_bil_filt	Bilateral filter implemented in camera device
	bool	cam_fly_p_filt	Flying pixel filter implemented in camera device
Reference	3-4-6-1-9. CMD_POSTFILT_INFO		

3-4-8-2-2-6.LensInfo

Table 161. LensInfo definition

Definition	struct LensInfo {
	uint16_t sens_w;
	uint16_t sens_h;
	uint32_t focal_len;
	uint8_t thin_w;
	uint8_t thin_h;
	Point2d crop;
	bool cam_dist;
	uint64_t dist[9];
	uint16_t lens_calib;
	};		

Description	・ Indicates parameter information used in lens-based conversion processing. ・ The contents of thin_w, thin_h, and crop are the same as those in ModeInfo. ・ If cam_dist is true, the contents of dist are invalid.		
Arguments	**Type**	**Name**	**Remark**
	uint16_t	sens_w	Sensor width [pixel]
	uint16_t	sens_h	Sensor height [pixel]
	uint32_t	focal_len	Focal length (Fixed point: Integer 12bit, Fractional 20bit)
	uint8_t	thin_w	Horizontal thinning argument (1/thin_w)

	uint8_t	thin_h	Vertical thinning argument (1/thin_h)
	Point2d	crop	Image clipping position from sensor pixel
	uint64_t	dist[9]	Distortion correction parameters [fx, fy, cx, cy, k1, k2, p1, p2, k3] (Fixed-point: 1bit sign, 16bit integer, 47bit fraction)
	bool	cam_dist	Perform distortion correction in the camera device
	uint16_t	lens_calib	Lens Calibration Revision
Reference	3-2-3-2-2. Point2d, 3-4-6-1-10. CMD_LENS_INFO		

3-4-8-2-2-7.CamFov

Table 162. CamFov definition

Definition	struct CamFov {
	uint16_t horz;
	uint16_t vert;
	};		

Description	・ Indicates FOV information for the camera device.		
Arguments	**Type**	**Name**	**Remark**
	uint16_t	horz	Horizontal viewing angle [degree × 100]
	uint16_t	vert	Vertical field of view [degree × 100]
Reference	3-4-6-1-2. CMD_FOV		

3-4-8-2-2-8.ModeInfo

Table 163. ModeInfo definition

Definition	struct ModeInfo {
	uint8_t id;
	std::string description;
	std::vector<ImgOutKind> img_out;
	Range dist_range;
	uint16_t fps;
	uint8_t thin_w;
	uint8_t thin_h;
	Point2d crop;
	bool light_times;
	uint16_t range_calib;
	};		

Description	・ Indicates the operation mode information.		
Arguments	**Type**	**Name**	**Remark**
	uint8_t	id	Operation Mode ID
	std::string	description	Operation Mode Description
	std::vector<ImgOutKind>	img_out	Receivable image type
	Range	dist_range	Range [mm]
	uint16_t	fps	Received frame rate [fps × 100]
	uint8_t	thin_w	Horizontal thinning argument (1/thin_w)
	uint8_t	thin_h	Vertical thinning argument (1/thin_h)
	Point2d	crop	Image clipping position from sensor pixel

	bool	light_times	Can the number of flashes be changed? (true: Can be changed, false: Cannot be changed)
	uint16_t	range_calib	Calibration Revision
Reference	3-2-3-2-5. Range, 3-4-8-1-2-5. ImgOutKind, 3-4-8-3-2-1. ModeList		

3-4-8-2-2-9.LightTimesInfo

Table 164. LightTimesInfo definition

Definition	struct LightTimesInfo {
	uint32_t min;
	uint32_t max;
	uint32_t count;
	};		

Description	· Indicates the number of "Light times (light emission)".		
Arguments	**Type**	**Name**	**Remark**
	uint32_t	min	Number of emissions, lower limit value
	uint32_t	max	Number of emissions, upper limit value
	uint32_t	count	Number of emissions, current value
Reference	3-4-6-1-11. CMD_LIGHT_TIMES		

3-4-8-2-2-10.AEIntervalInfo

Table 165. AEIntervalInfo definition

Definition	struct AEIntervalInfo {
	uint8_t min;
	uint8_t max;
	uint8_t interval;
	};		

Description	· Indicates control interval information when the AE function is enabled.		
Arguments	**Type**	**Name**	**Remark**
	uint8_t	min	AE interval, lower limit [frame]
	uint8_t	max	AE interval, upper limit [frame]
	uint8_t	interval	AE Interval, Current value [frame]
Reference	3-4-6-1-13. CMD_AE_INTERVAL		

3-4-8-2-2-11.SignalThresholdInfo

Table 166. SignalThresholdInfo definition

Definition	struct SignalThresholdInfo {
	uint16_t min;
	uint16_t max;
	uint16_t threshold;
	};		

Description	· Indicates RAW saturation threshold or IR invalid threshold information.		
Arguments	**Type**	**Name**	**Remark**
	uint16_t	min	Lower threshold value of RAW amplitude data

	uint16_t	max	Upper threshold value of IR amplitude data
	uint16_t	threshold	Current threshold value
Reference	3-4-6-1-14. CMD_RAW_SAT_TH, 3-4-6-1-15. CMD_IR_DARK_TH		

3-4-8-2-2-12.CamPrmRequest

Table 167. CamPrmRequest definition

Definition	struct CamPrmRequest {
	CamPrmKind kind;
	uint16_t id;
	std::vector<uint8_t> data;
	};		

Description	· Indicates camera parameter information. · Parameter ID is valid only when the parameter type is CAM_PRM_PART.		
Arguments	**Type**	**Name**	**Remark**
	CamPrmKind	kind	Camera parameter type
	uint16_t	id	Camera parameter ID
	std::vector<uint8_t>	data	Camera parameter data
Reference	3-4-8-1-2-6. CamPrmKind		

3-4-8-2-2-13.RegDevInfo

Table 168. RegDevInfo definition

Definition	struct RegDevInfo {
	RegDevType target;
	uint8_t addr_len;
	uint8_t val_len;
	uint8_t list_len;
	};		

Description	· Indicates information about the camera device registers.		
Arguments	**Type**	**Name**	**Remark**
	RegDevType	target	Target device to be controlled
	uint8_t	addr_len	Size of register address [byte]
	uint8_t	val_len	Size of register value [byte]
	uint8_t	list_len	Maximum number of registers during continuous R/W
Reference	3-4-8-1-2-7. RegDevType, 3-4-8-3-2-2. RegDevs		

3-4-8-2-2-14.RegInfo

Table 169. RegInfo definition

Definition	struct RegInfo {
	RegDevType target;
	uint16_t addr;
	uint16_t value;
	};		

Description	· Indicates the register information of the camera device.		
Arguments	**Type**	**Name**	**Remark**

	RegDevType	target	Target register to be controlled
	uint16_t	addr	Register address
	uint16_t	value	Register value
Reference	3-4-8-1-2-7. RegDevType		

3-4-8-2-2-15.RegList

Table 170. RegList definition

Definition	struct RegList {
	RegDevType target;
	uint16_t addr;
	std::vector<uint16_t> values;
	};		

Description	· Indicates the list information of the register that is the consecutive address of the camera device.		
Arguments	**Type**	**Name**	**Remark**
	RegDevType	target	Target register to be controlled
	uint16_t	addr	Top register address
	std::vector<uint16_t>	values	Register value list
Reference	3-4-8-1-2-7. RegDevType		

3-4-8-2-2-16.IntSuppParamInfo

Table 171. IntSuppParamInfo definition

Definition	struct IntSuppParamInfo {
	uint8_t min;
	uint8_t max;
	uint8_t value;
	};		

Description	· Indicates the parameters of the interference prevention function.		
Arguments	**Type**	**Name**	**Remark**
	uint8_t	min	Minimum allowed value of the parameter
	uint8_t	max	Maximum allowed value of the parameter
	uint8_t	values	Parameter value
Reference	3-4-8-2-2-17. IntSuppInfo		

3-4-8-2-2-17.IntSuppInfo

Table 172. IntSuppInfo definition

Definition	struct IntSuppInfo {
	IntSuppModeType mode;
	IntSuppParamInfoprm_m;
	IntSuppParamInfoprm_a1;
	IntSuppParamInfoprm_a2;
	IntSuppParamInfoprm_a3;
	};

Description	· Indicates information about the interference prevention function.

		Name	Remark
	IntSuppModeType	mode	Interference prevention function mode
	IntSuppParamInfo	prm_m	Manual mode parameter
	IntSuppParamInfo	prm_a1	Automatic mode parameter 1 Value:31
	IntSuppParamInfo	prm_a2	Automatic mode parameter 2 Value:65
	IntSuppParamInfo	prm_a3	Automatic mode parameter 3 Value:4
Reference	3-4-8-1-2-8. IntSuppModeType, 3-4-8-2-2-16. IntSuppParamInfo, 3-4-6-1-16. CMD_INT_SUPP_INFO		

3-4-8-2-2-18.UsbPCAccKey

Table 173. UsbPCAccKey definition

Definition	struct UsbPCAccKey {
	uint16_t key;
	};		

Description	・ Indicates the access key for the USB peripheral controller.		
Arguments	**Type**	**Name**	**Remark**
	uint16_t	key	16 bit access key
Reference	-		

3-4-8-3. Type definition

3-4-8-3-1. List of type definitions

Table 174. Type definitions

Name	Description
ModeList	List of operation modes
RegDevs	Register controllable device information list

3-4-8-3-2. Type definition details

3-4-8-3-2-1.ModeList

Table 175. ModeList definition

Definition	using ModeList = std::vector<ModeInfo>;
Description	・ Indicates list information for the operation mode.
Reference	3-4-8-2-2-8. ModeInfo, 3-4-6-1-5. CMD_MODE_LIST

3-4-8-3-2-2.RegDevs

Table 176. RegDevs definition

Definition	using RegDevs = std::vector<RegDevInfo>;
Description	・ Indicates register controllable device information
Reference	3-4-8-2-2-13. RegDevInfo

3-4-9. Definition for the file playback function

The definition used only for the file playback function of the Camera class described in PlayBackType.h is shown below.

3-4-9-1. Enumeration definition

3-4-9-1-1. List of enumeration definitions

Table 177. Enumeration definitions

Name	Description
PlayBack::PlayBackCmd	Property commands available only with the file playback function
PlayBack::PlayState	File playback status

3-4-9-1-2. Enumeration definition details

***3-4-9-1-2-1.*PlayBack::PlayBackCmd**

Table 178. PlayBack::PlayBackCmd definition

Definition	enum PlayBackCmd : uint16_t {
	CMD_PLAY_TARGET = CMD_MAX,
	CMD_PLAY_TIME,
	CMD_PLAY_STATUS,
	CMD_PAUSE,
	CMD_FAST_PLAY,
	CMD_SLOW_PLAY,
	CMD_JUMP_FW,
	CMD_JUMP_BW,
	};		

Description	・ Indicates the property command that can be used only with the file playback function.		
Value	**Name**	**Value**	**Remark**
	CMD_PLAY_TARGET	12	File Replay target directory information
	CMD_PLAY_TIME	13	Playback time information
	CMD_PLAY_STATUS	14	Playback status information
	CMD_PAUSE	15	Playback pause control
	CMD_FAST_PLAY	16	Double speed playback control
	CMD_SLOW_PLAY	17	Slow playback control
	CMD_JUMP_FW	18	Forward control
	CMD_JUMP_BW	19	Rewind control
Reference	3-4-5-6. getProperty, 3-4-5-7. setProperty, 3-4-7. Property command (for file playback only)		

3-4-9-1-2-2.PlayBack::PlayState

Table 179. PlayBack:: PlayState definition

Definition	enum PlayState { STOPPED, PLAYING, PAUSE,

	FAST, SLOW };
Description	· Indicates the file playback status.

Value	Name	Value	Remark
	STOPPED	0	Stop status
	PLAYING	1	Playback status (constant speed playback)
	PAUSE	2	Pause status
	FAST	3	Double speed playback status
	SLOW	4	Slow playback status

Reference	3-4-9-2-2-3. PlayBack::PlayStatus

3-4-9-2. Structure definition

3-4-9-2-1. List of structure definitions

Table 180. Structure definitions

Name	Description
PlayBack::ConfigParam	Initial parameters for file playback
PlayBack::PlayTime	Playback time information
PlayBack::PlayStatus	Status of the file playback function

3-4-9-2-2. Structure definition details

3-4-9-2-2-1.PlayBack::ConfigParam

Table 181. PlayBack::ConfigParam definition

Definition	struct ConfigParam {
	std::filesystem::path path;
	};

Description	· Indicates the initial parameter for file playback.

Arguments	Type	Name	Remark
	std::filesystem::path	path	Target directory for Playback

Reference	3-4-5-1. Camera

3-4-9-2-2-2.PlayBack::PlayTime

Table 182. PlayBack::PlayTime definition

Definition	struct PlayTime {
	uint32_t total;
	uint32_t current;
	};

Description	· Indicates playback time information. · To convert frame numbers to playback time, use the received frame rate information (fps) that can be acquired by getProperty(CMD_MODE_LIST). (Playback time [sec] = frame number ÷ fps)

Arguments	Type	Name	Remark
	uint32_t	total	Total frames

	uint32_t	current	Current frame number (from 0-th frame)
Reference	3-4-7-1-2. PlayBack::CMD_PLAY_TIME		

3-4-9-2-2-3.PlayBack::PlayStatus

Table 183. PlayBack::PlayStatus definition

Definition	struct PlayStatus {
	PlayState state;
	uint16_t playing_fps;
	};		

Description	・ Indicates the status information of the file playback function. ・ playing_fps indicates the frame rate during playback. If state is the playback state (constant speed playback), the actual operation will also be the original frame rate. This is the same as the received frame rate that can be acquired by getProperty(CMD_MODE_LIST). In the double speed or slow playback state, the frame rate will be the frame rate after doubling or slowing the original frame rate. (Example: Original at 30fps, 60fps at 2x playback speed, 15fps at 1/2x slow playback speed)		
Arguments	**Type**	**Name**	**Remark**
	PlayBack::PlayState	state	File Playback status
	uint16_t	playing_fps	Frame rate during playback [fps × 100]
Reference	3-4-9-1-2-2. PlayBack::PlayState, 3-4-7-1-3. PlayBack::CMD_PLAY_STATUS		

3-5. Record class

3-5-1. Overview

Table 184. Record class overview

Provided header files	Record.h	Record class definition
	RecordType.h	Record class type definition
Member Namespace	krm::Record	
Description	C++ class for saving camera device output images to a file	
Thread Safety	The Record class is thread-safe. There is no need for exclusive processing in the user program being used.	

3-5-2. Provided function

3-5-2-1. Functional overview

The following provides an overview of the functions provided by the Record class.

Table 185. Functional overview of Record class

Provided function	Functional overview
File save function	This function saves image data as a file. The depth image, IR image, sensor RAW image and additional frame information received as input are output as a file.

3-5-2-1-1. Data structure to be saved by Record function

The following shows the directory structure after saving a file by using the Record class.

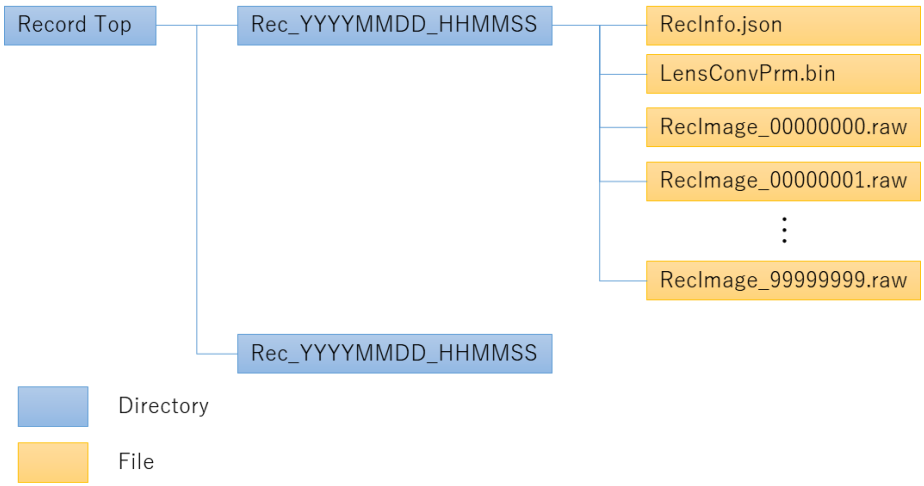


Figure 21. Data structure to be saved by Record function

3-5-2-1-1-1.Rec_YYYYMMDD_HHMMSS Directory

During initialization of the destination directory by Record:: openRec(), the directory (YYYYMMDD_HHMMSS = YYYY: year, MM: month, DD: day, HH: hour, MM: minute, SS: second) is created with the date and time of the start of saving.

3-5-2-1-1-2.RecInfo.json

It is generated based on the information set during initialization of the destination directory by Record:: openRec(). The structure of this file is as follows.

The current format version recorded in RecInfo.json is Ver3.0.4.

Table 186. RecInfo.json information

Parameter name			Size (byte)	Information
Version			-	Recinfo format version
	major		1	Major version
	minor		1	Minor version
	revision		2	Revision
Record			-	Record information
	rec_frames		4	Number of frames saved
	packing_frames		2	Number of frames in one file
Device			-	Device information
	hw_kind		4	HW model number Upper 16 bits : Sensor model number Lower 16 bits : Lens model number
	serial_no		4	Device identification number
	map_ver		-	In-camera settings MAP version
		major	1	Major Version
		minor	1	Minor Version
		revision	2	Revision
	adjust_no		4	Adjustment Number
	firm_ver		-	Camera Firmware Version
		major	1	Major Version
		minor	1	Minor Version
		revision	2	Revision
	ld_wave		2	Light source wavelength [nm]
	ld_enable		2	Valid information for each light source (light number information)
	correct_calib		2	Compensation Calibration Revision
PostFilt			-	PostFilter information
	cam_med_filt		1	Median filter applied at the input of the save function
	cam_bil_filt		1	Bilateral filter applied at the input of the save function
	cam_fly_p_filt		1	Flying pixel filter applied at the time of input of storage function
Lens			-	Parameter information for lens system conversion
	sens_w		2	Sensor width [pixel]
	sens_h		2	Sensor height [pixel]
	focal_len		4	Focal length (Fixed point: Integer 12bit, Fractional 20bit)
	thin_w		1	Horizontal weeding argument (1/thin_w)

	thin_h	1	Vertical thinning argument (1/thin_h)
	crop_x	2	Image clipping from sensor pixel X-coordinate position [pixel]
	crop_y	2	Image clipping from sensor pixel Y-coordinate position [pixel]
	dist[]	8×9	Parameters for distortion correction [fx, fy, cx, cy, k1, k2, p1, p2, k3] (Fixed-point: 1bit sign, 16bit integer, 47bit fraction)
	cam_dist	1	Distortion correction status at the time of input of the save function
	lens_calib	2	Lens Calibration Revision
Fov		-	FOV information
	horz	2	Horizontal viewing angle [degree × 100]
	vert	2	Vertical viewing angle [degree × 100]
Mode		-	Operating mode information
	id	1	Operation Mode ID
	description	32	Operation Mode Description
	Images []	-	Image type information (Array data of image type included in RecImage.raw)
	kind	2~6	Image type (string) “Depth,” “IR”, “RAW G1,” “RAW G2,” “RAW G3,” “RAW G4”
	width	2	Image width [pixel]
	height	2	Image height [pixel]
	active_start []	2×2	Effective pixel start position [X coordinate position, Y coordinate position]
	active_w	2	Effective pixel width [pixel]
	active_h	2	Effective pixel height [pixel]
	bpp	1	Size of one pixel [byte]
	range	-	Ranging range information
	min	2	Closest distance [mm]
	max	2	Farthest Distance [mm]
	fps	2	Received Frame Rate [fps × 100]
	range_calib	2	Ranging Calibration Revision

3-5-2-1-1-3.RecImage.raw

The saved image data is saved in binary format (little endian) in RecImage_XXXXXXXX.raw.The file name XXXXXXXX becomes a 10 decimal value (00,000,000 ~99999999) starting from 0. Each time the number of frames corresponding to the number of frames (packing_frames) in one RecInfo.json file is entered, it is divided into another file and the file name XXXXXXXX is incremented.

The contents of RecImage_XXXXXXXX.raw are in the form in which image data for multiple frames (1 to packing_frames) are included in one file based on the information set when the storage directory is initialized by Record:: openRec() as shown below.One frame includes image data according to the order and contents of Mode – Images in RecInfo.json for each image type.

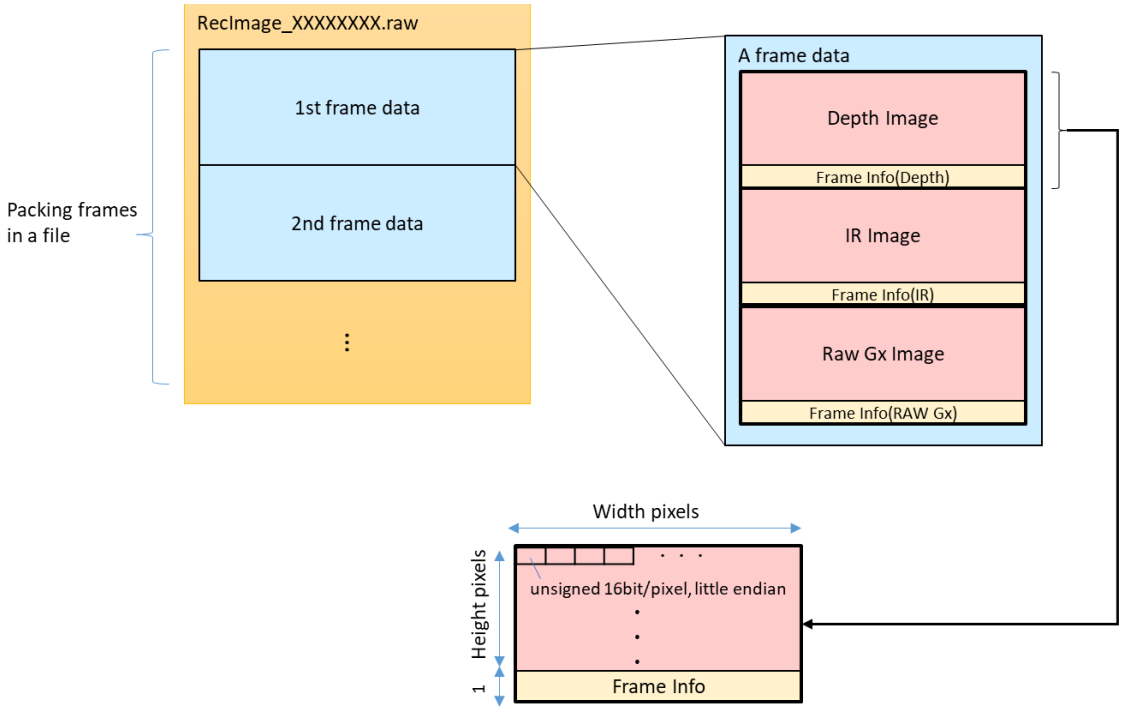


Figure 22. RecImage.raw data structure

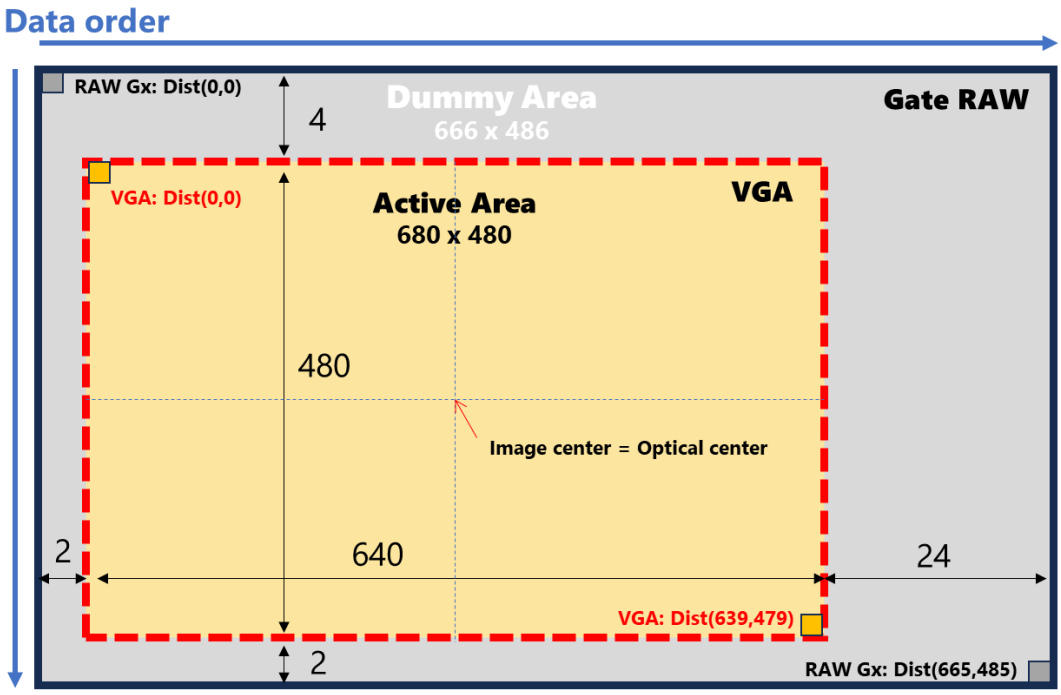


Figure 23. VGA output data array information

Note: IR/Depth data and Gate RAW data (RAW G1, G2, G3, G4) have different output data sizes.
Note: IR/Depth data is output only for Active Area.
Note: Gate RAW (RAW G1, G2, G3, G4) data output is a debugging function.

3-5-2-1-1-3-1.RecImage.raw – Frame Info

In the position of Frame Info in RecImage.raw, the following information is stored from the top in an area for 1 line of each image.

Table 187. FrameInfo data structure

Parameter name		Size	
Frame Number		4byte	
Time Stamp	second	8byte	
	nanosecond	8byte	
Frame Error	dropped	1bit	2byte
	crc error	1bit	
	reserved	14bit	
Temperature (Integer: 10bit, Decimal: 6bit)		2byte	
Light times		4byte	
Conversion State	distortion corrected	1bit	1byte
	median filter applied	1bit	
	bilateral filter applied	1bit	
	flying pixel filter applied	1bit	
	reserved	4bit	
reserved		Remaining for 1Line (2 bytes ×width – 29 bytes)	

3-5-3. List of methods

Table 188. Record class methods

Method name	Description
Record	Constructors
~Record	Destructors
openRec	Record destination directory initialization processing
closeRec	Record destination directory termination processing
recFrame	Frame data storage processing

3-5-3-1. Record class state machine

The status transitions of the Record class are as follows.

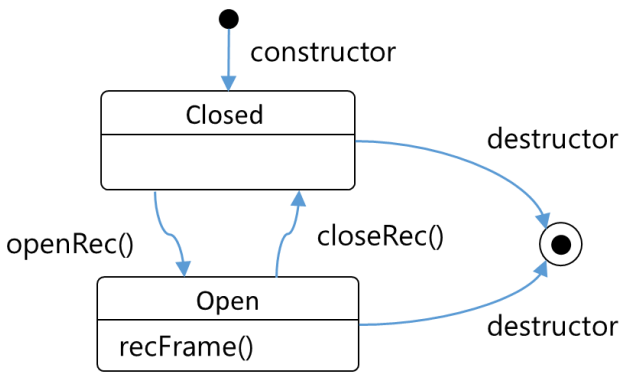


Figure 24. Record class state machine

3-5-4. Control sequence

This section describes a record control sequence using the Record class. For the following sequences, judgment of abnormal systems such as function return value judgment is omitted.

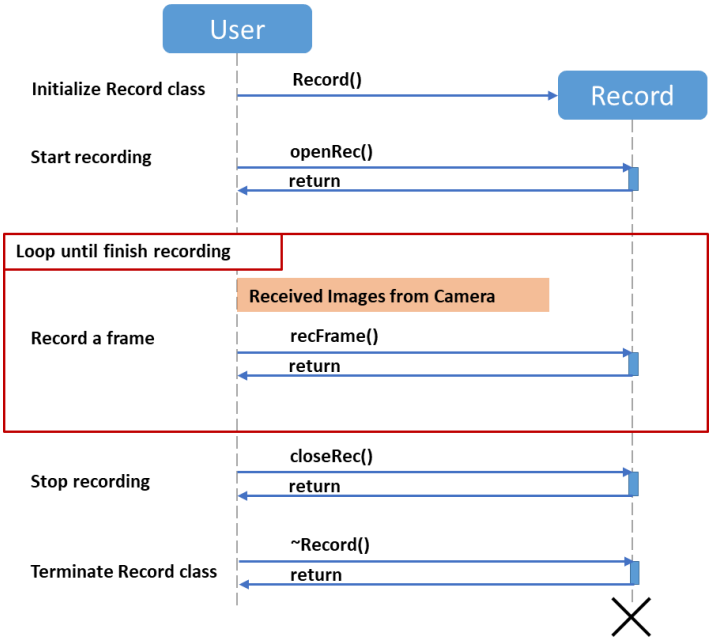


Figure 25. Data record control sequence

3-5-5. Method details

3-5-5-1. Record

Table 189. Record::Record method

Function	Constructor			
Syntax	Record (void)			
Description	・ Initializes the file record function.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	None			
Return value	None			
Sync/Async	Synchronous			

3-5-5-2. ~Record

Table 190. Record::~Record method

Function	Destructor			
Syntax	~Record (void)			
Description	・ Terminates the file saving function. ・ After executing openRec(), if closeRec() has not been executed, closeRec() is executed in this method.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	None			

Return value	None
Sync/Async	Synchronous

3-5-5-3. openRec (1)

Table 191. Record::openRec method

Function	Initialization of the Record destination directory.				
Syntax	Result openRec (const ConfigParam& config)				
Description	・ Performs initialization of the Record destination directory. ・ ERR_BAD_STATE is returned if this method is called when the destination directory has already been opened. ・ All image types output from the Camera will be saved. Enter image data when recFrame() is used as the image type to be saved. ・ The save file will be synchronized when the end of file is reached. However, if param.packing_frames is set to 10 or more, the save file will be synchronized every 10 frames. ・ The param.save_frames and param.packing_frames arguments must be set to 1 or greater. If either of these values is 0, ERR_BAD_ARG is returned. ・ ERR_FULL is returned if the remaining space after saving the number of save frames (param.save_frames) is expected to be less than 1GB.				
Arguments	**Type**		**Name**	**in/out**	**Remark**
	const Record::ConfigParam&		config	in	Save settings
Return value	SUCCESS	0	Success		
	ERR_BAD_ARG	5	The config.save_frames or config.packing_frames argument is 0.		
	ERR_BAD_STATE	6	Status transition error		
	ERR_NOT_EXIST	7	The specified directory does not exist.		
	ERR_FULL	10	Insufficient remaining space.		
	ERR_SYSTEM	12	System error (such as memory allocation failure)		
Sync/Async	Synchronous				

3-5-5-4. openRec (2)

Table 192. Record::openRec method

Function	Initialization of the Record destination directory.
Syntax	Result openRec (const RecInfoParam& param)
Description	・ Performs initialization of the destination directory. ・ ERR_BAD_STATE is returned if this method is called when the destination directory has already been opened. ・ All image types output from the Camera will be saved. Enter image data when recFrame() is used as the image type to be saved.

	・ The save file will be synchronized when the end of file is reached. However, if param.packing_frames is set to 10 or more, the save file will be synchronized every 10 frames. ・ The param.save_frames and param.packing_frames arguments must be set to 1 or greater. If either of these values is 0, ERR_BAD_ARG is returned. ・ ERR_FULL is returned if the remaining space after saving the number of save frames (param.save_frames) is expected to be less than 1GB.				
Arguments	**Type**		**Name**	**in/out**	**Remark**
	const Record::RecInfoParam&		param	in	Save settings
Return value	SUCCESS	0	Success		
	ERR_BAD_ARG	5	The config.save_frames or config.packing_frames argument is 0.		
	ERR_BAD_STATE	6	Status transition error		
	ERR_NOT_EXIST	7	The specified directory does not exist.		
	ERR_FULL	10	Insufficient remaining space.		
	ERR_SYSTEM	12	System error (such as memory allocation failure)		
Sync/Async	Synchronous				

3-5-5-5. closeRec

Table 193. Record::closeRec method

Function	Close the Record destination directory			
Syntax	Result closeRec (void)			
Description	・ Closes the Record destination directory. ・ If this method is called when openRec() has not been executed, no processing is performed and SUCCESS is returned.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	None			
Return value	SUCCESS	0	Success	
	ERR_SYSTEM	12	System error (memory allocation failure, etc.)	
Sync/Async	Synchronous			

3-5-5-6. recFrame

Table 194. Record::recFrame method

Function	Frame data storage processing
Syntax	Result recFrame (const Frame& frame)
Description	・ Saves data for one frame. ・ REACH_EOF is returned when the number of saved frames specified by openRec() has been reached. After that, save processing cannot be performed until closeRec() and openRec() are executed again, and ERR_BAD_STATE is returned. ・ ERR_EMPTY is returned when the image data of the image type output from the Camera does not exist or is of a different size at the time of openRec().

	・ ERR_BAD_STATE is returned if this method is called when openRec() has not been executed.				
Arguments	**Type**		**Name**	**in/out**	**Remark**
	const Frame&		frame	in	Frame data to be saved
Return value	SUCCESS	0	Success		
	REACH_EOF	2	Storing of the specified number of frames completed		
	ERR_BAD_STATE	6	Abnormal state transition		
	ERR_EMPTY	9	Empty buffer, etc.		
	ERR_FULL	10	A space shortage occurred.		
	ERR_SYSTEM	12	System error (memory allocation failure, etc.)		
Sync/Async	Synchronous				

3-5-6. Definition for record class

The following definitions are used by the Record class, which is described in RecordType.h.

3-5-6-1. Structure definition

3-5-6-1-1. List of structure definitions

Table 195. Structure definitions

Name	Description
Record::ConfigParam	Configuration information to be saved
Record::RecInfoParam	Configuration information to be saved (without Camera object)

3-5-6-1-2. Structure definition details

3-5-6-1-2-1.Record::ConfigParam

Table 196. Record::ConfigParam definitions

Definition	struct ConfigParam {		
	std::filesystem::path	path;	
	uint32_t	save_frames;	
	uint16_t	packing_frames;	
	Camera*	cam_obj;	
	bool	is_crct_dist;	
	bool	is_filt_med;	
	bool	is_filt_bil;	
	bool	is_filt_fly_p;	
	};		
Description	・ Indicates the settings to be saved.		
Arguments	**Type**	**Name**	**Remark**
	std::filesystem::path	path	Destination Directory
	uint32_t	save_frames	Number of frames saved
	uint16_t	packing_frames	Number of frames included in one file
	Camera*	cam_obj	Camera object
	bool	is_crct_dist	Whether distortion has been corrected

	bool	is_filt_med	Whether median filter has been applied
	bool	is_bil_med	Whether bilateral filter has been applied
	bool	is_filt_fly_p	Whether Flying pixel filter applied
Reference	3-4-5-1. Camera		

3-5-6-1-2-2.Record::RecInfoParam

Table 197. Record::RecInfoParam definition

Definition

```
struct RecInfoParam {
        std::filesystem::path       path;
        uint32_t                    save_frames;
        uint16_t                    packing_frames;
        DeviceInfo                  device_info;
        LensInfo                    lens_info;
        CamFov                      fov;
        ModeList                    mode_list;
        ImageFormats                image_formats;
        uint8_t                     mode;
        bool                        is_crct_dist;
        bool                        is_filt_med;
        bool                        is_filt_bil;
        bool                        is_filt_fly_p;
};
```

Description

- Indicates the settings to be saved.

Arguments

Type	Name	Remark
std::filesystem::path	path	Destination directory
uint32_t	save_frames	Number of frames to be saved
uint16_t	packing_frames	Number of frames to be included in one file
DeviceInfo	device_info	Camera device information
LensInfo	lens_info	Lens system conversion processing information
CamFov	fov	Camera device FOV information
ModeList	mode_list	Operation mode list information
ImageFormats	image_formats	Format information for all image types
uint8_t	mode	Operation mode ID
bool	is_crct_dist	Whether distortion has been corrected
bool	is_filt_med	Whether median filter has been applied
bool	is_filt_bil	Whether bilateral filter has been applied
bool	is_filt_fly_p	Whether Flying pixel filter applied

Reference

3-4-8-2-2-4. DeviceInfo, 3-4-8-2-2-6. LensInfo, 3-4-8-2-2-7. CamFov, 3-4-8-3-2-1. ModeList, 3-2-4-2-1. ImageFormats

3-6. LensConv class

3-6-1. Overview

Table 198. LensConv class overview

Provided header files	LensConv.h	LensConv class definition
	LensConvType.h	LensConv class type definition
Member Namespace	krm	
Description	C++ class that performs lens-related conversion processing on the camera device's output image	
Thread safety	The LensConv class is not thread-safe. Therefore, the user program to be used must perform exclusive processing as necessary.	

3-6-2. Provided function

3-6-2-1. Functional overview

The following shows an overview of the functionality provided by the LensConv class.

Table 199. Functional overview of LensConv class

Provided function	Functional overview
Distortion correction	This function corrects an image that is distorted by the lens curve. It corrects the distortion of the Depth image or IR image received as input, and outputs the corrected image. Distortion correction is performed using the Lens conversion processing parameters of the Camera class.
Point cloud conversion	This function receives the corrected Depth image as input, and outputs the point cloud data in 3-dimensional space. Point Cloud Conversion is performed using the Lens conversion processing parameters of the Camera class.

3-6-2-1-1. Distortion correction function

Images output from the Camera class are distorted by lens distortion. The distortion correction function corrects distortion caused by distortion as shown below.

Figure 26. Distortion correction

3-6-2-1-2. Point cloud conversion function

This function converts each pixel of the distortion corrected Depth image to a point cloud in the camera coordinate system or the world coordinate system.

3-6-2-1-2-1.Camera coordinates

Point Cloud Conversion of camera coordinates converts the Depth image value D to point P(Px, Py, Pz) in the 3-dimensional coordinates of the lens center.

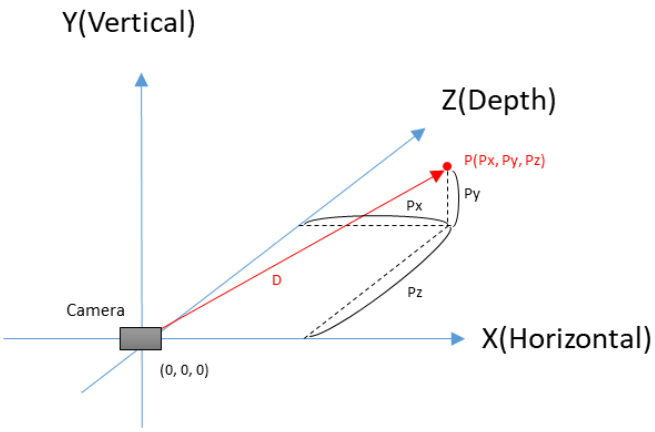


Figure 27. Camera coordinates Point cloud conversion

3-6-2-1-2-1-1.World coordinates

World Coordinate Point Cloud Transformation converts the Depth image value D to a 3D coordinate point P'(Px', Py', Pz') whose origin is different from the camera coordinate center O(Ox, Oy, Oz).

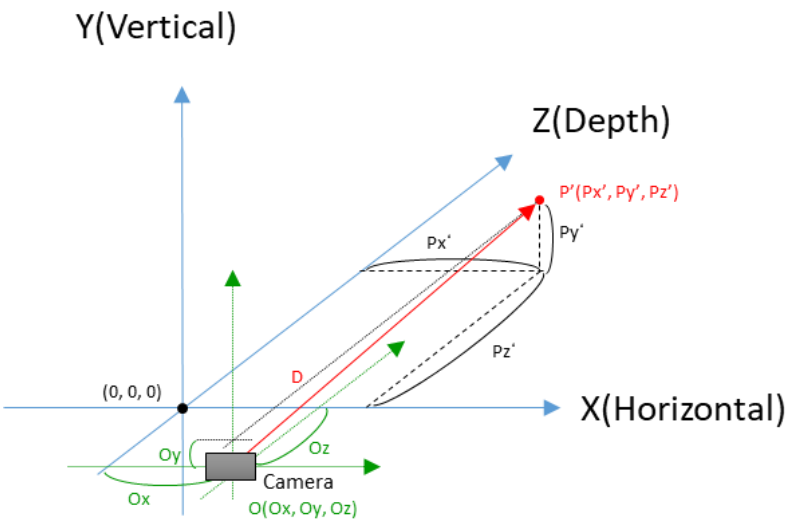


Figure 28. World coordinate Point cloud transformation

When transforming the world coordinate point cloud, the point cloud can be rotated according to the installation conditions of the camera device.

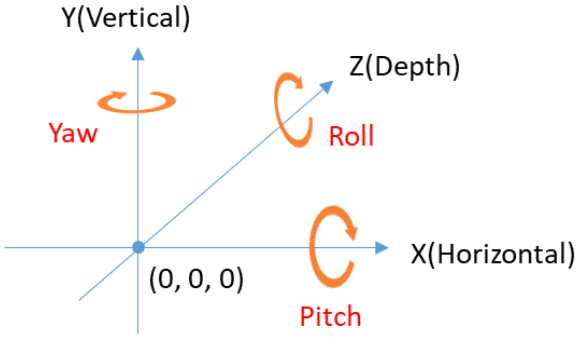


Figure 29. Point cloud rotation direction

3-6-3. List of methods

Table 200. LensConv class methods

Method name	Description
LensConv	Constructor
~LensConv	Destructor
setLensPrm	Lens system conversion processing parameter setting
setFormat	Image format information setting
setPosOrgRotation	World coordinate origin and point group rotation information setting
correctDist	Execution of distortion correction
convPcdCamera	Camera coordinate point group conversion execution
convPcdWorld	World coordinate point group conversion execution

3-6-3-1. State machine

The state machine of the LensConv class are as follows.

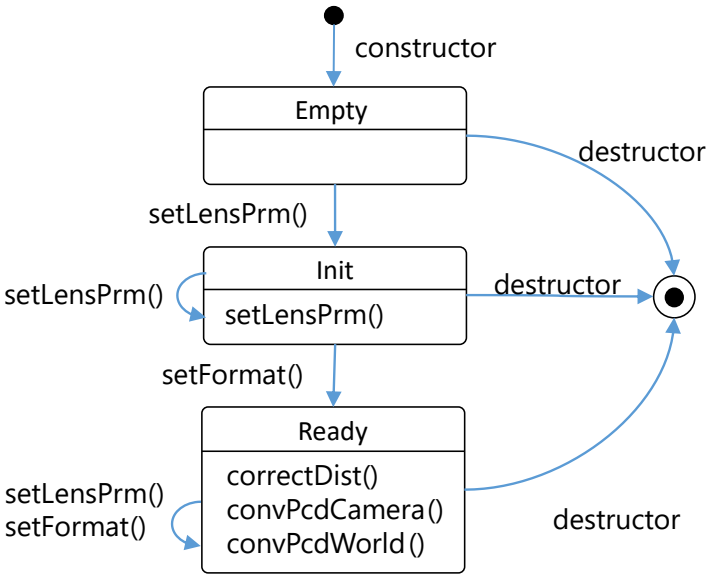


Figure 30. LensConv class state machine

3-6-4. Control sequence

The image conversion sequence using the LensConv class is described below. In the following sequence, judgment of abnormal systems such as function return value judgment is omitted.

Setting parameters for lens system conversion processing (setLensPrm(), setFormat()) must be performed before image reception.

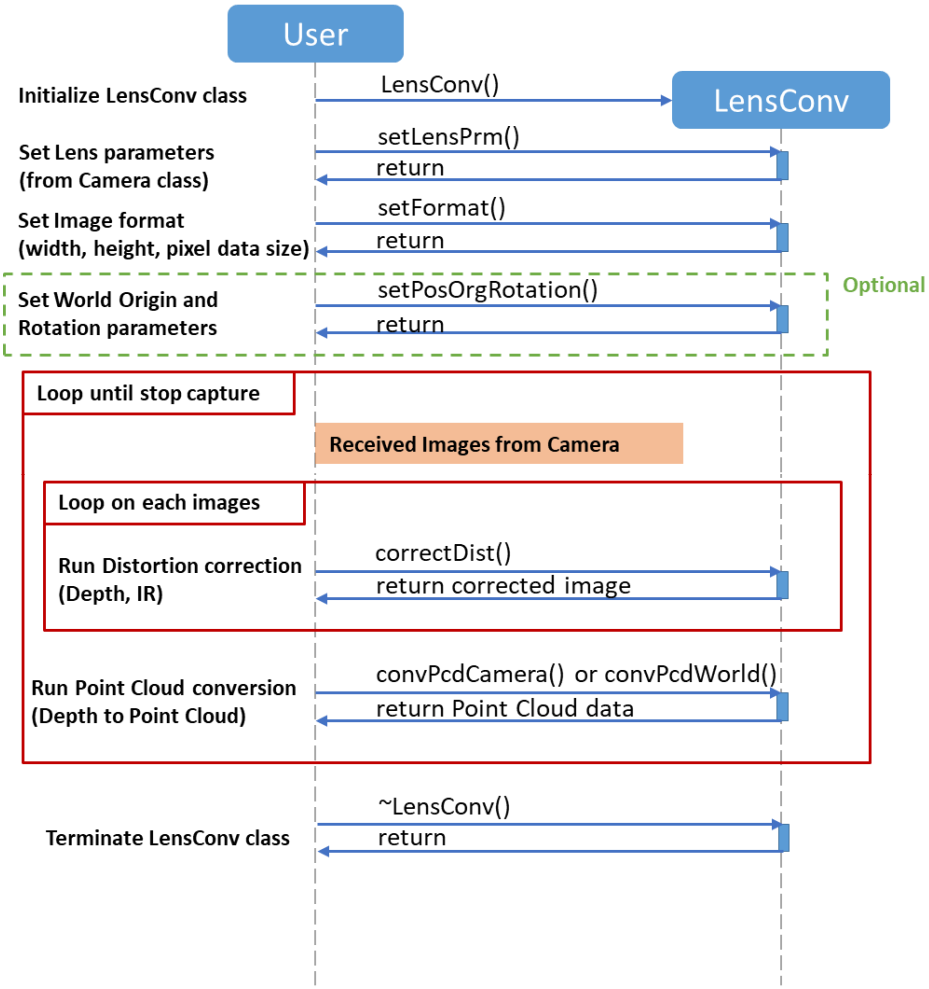


Figure 31. Image conversion sequence

3-6-5. Method details

3-6-5-1. LensConv

Table 201. LensConv::LensConv method

Function	Constructor			
Syntax	LensConv (void)			
Description	・ Initializes Lens-based conversion processing.			
Arguments	Type	Name	in/out	Remark
	None			
Return value	None			
Sync/Async	Synchronous			

3-6-5-2. ~LensConv

Table 202. LensConv::~LensConv method

Function	Destructor			
Syntax	~LensConv (void)			
Description	· Terminates Lens type conversion processing.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	None			
Return value	None			
Sync/Async	Synchronous			

3-6-5-3. setLensPrm

Table 203. LensConv::setLensPrm method

Function	Setting parameters for lens system conversion processing			
Syntax	Result setLensPrm (const LensInfo& lens_info)			
Description	· Sets parameter information for using the distortion correction function and point cloud conversion function.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	const LensInfo&	lens_info	in	Lens type conversion processing parameter
Return value	SUCCESS	0	Success	
	ERR_BAD_ARG	5	Other invalid argument	
Sync/Async	Synchronous			

3-6-5-4. setFormat

Table 204. LensConv::setFormat method

Function	Image format setting information			
Syntax	Result setFormat (const ImageFormat& format)			
Description	· Specifies format information for images handled by the LensConv class. · To use the point cloud conversion function, set the image format of the Depth image. · To apply the distortion correction function to multiple types of images, they must have the same image format (Image Width, Image Height). · ERR_BAD_STATE is returned if this method is called when the parameters for lens system conversion processing have not been set by setLensPrm().			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	const ImageFormat&	format	in	Image format
Return value	SUCCESS	0	Success	
	ERR_BAD_ARG	5	Image format information is empty.	

	ERR_BAD_STATE	6	Status transition error
Sync/Async	Synchronous		

3-6-5-5. setPosOrgRotation

Table 205. LensConv:: setPosOrgRotation method

Function	World coordinate origin and point cloud rotation setting			
Syntax	void setPosOrgRotation (const PosOrgRotation& pos_rot)			
Description	・ Sets the origin position and point cloud rotation information to be converted by the point cloud conversion function of the world coordinate system. ・ If this method is not called, the origin position is the same coordinates as the camera coordinate system, and the point cloud rotation angle is all 0.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	const PosOrgRotation&	pos_rot	in	Origin position and point cloud rotation information
Return value	None			
Sync/Async	Synchronous			

3-6-5-6. correctDist

Table 206. LensConv::correctDist method

Function	Distortion correction processing			
Syntax	Result correctDist (const ImageData& org_img, bool is_depth, ImageData& aft_img)			
Description	・ Performs distortion correction on the input image. ・ Allocate an area with the same number of pixels as org_img in the aft_img argument. ・ For the is_depth argument, set true if org_img is a depth image and false if it is an IR image. ・ ERR_BAD_STATE is returned when this method is called without setLensPrm() or setFormat() having been executed.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	const ImageData&	org_img	in	Input image
	bool	is_depth	in	Is the input image a depth image? true : Depth image false : IR image
	ImageData&	aft_img	out	Image after distortion correction
Return value	SUCCESS	0	Success	
	ERR_BAD_ARG	5	In the org_img and aft_img arguments, An image format different from setFormat() is specified.	
	ERR_BAD_STATE	6	State transition error	

Sync/Async	Synchronous

3-6-5-7. convPcdCamera

Table 207. LensConv:: convPcdCamera method

Function	Camera coordinate point cloud conversion processing			
Syntax	Result convPcdCamera (const ImageData& depth, PcdData& pcd)			
Description	・ Performs point cloud conversion to the camera coordinate system. ・ The pcd argument must have the same number of pixels as the depth argument. ・ ERR_BAD_STATE is returned if this method is called without setLensPrm() or setFormat() having been executed.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	const ImageData&	depth	in	Depth Image
	PcdData&	pcd	out	Point cloud data
Return value	SUCCESS	0	Success	
	ERR_BAD_ARG	5	In the depth and pcd arguments, An image format different from setFormat() is specified.	
	ERR_BAD_STATE	6	Status transition error	
Sync/Async	Synchronous			

3-6-5-8. convPcdWorld

Table 208. LensConv:: convPcdWorld method

Function	World Coordinate Point Cloud Conversion processing			
Syntax	Result convPcdWorld (const ImageData& depth, PcdData& pcd)			
Description	・ Converts the point cloud to the world coordinate system. ・ The pcd argument must have the same number of pixels as the depth argument. ・ ERR_BAD_STATE is returned if this method is called without setLensPrm() or setFormat() having been executed.			
Arguments	**Type**	**Name**	**in/out**	**Remark**
	const ImageData&	depth	in	Depth Image
	PcdData&	pcd	out	Point cloud data
Return value	SUCCESS	0	Success	
	ERR_BAD_ARG	5	In the depth and pcd arguments, An image format different from setFormat() is specified.	
	ERR_BAD_STATE	6	Status transition error	
Sync/Async	Synchronous			

3-6-6. Definition for LensConv class

The following shows the definition used by the LensConv class, which is described in LensConvType.h.

3-6-6-1. Structure definition

3-6-6-1-1. List of structure definitions

Table 209. Structure definition

Name	Description
PosOrgRotation	Origin position and rotation information

3-6-6-1-2. Structure definition details

3-6-6-1-2-1.PosOrgRotation

Table 210. PosOrgRotation definition

Definition	struct PosOrgRotation {
	struct {
	int16_t x;
	int16_t y;
	int16_t z;
	} offset;
	struct {
	float pitch;
	float yaw;
	float roll;
	} rotation;
	};		

Description	・ Indicates origin position and rotation information for point cloud transformation to the world coordinate system.		
Arguments	**Type**	**Name**	**Remark**
	int16_t	x	Origin position: Offset along X axis [mm]
	int16_t	y	Origin position: Offset along Y axis [mm]
	int16_t	z	Origin position: Offset along Z axis [mm]
	float	pitch	Pitch rotation angle [degree]
	float	yaw	Yaw rotation angle [degree]
	float	roll	Roll rotation angle [degree]
Reference	3-6-5-5. setPosOrgRotation		

4. Usage Restrictions

4-1. Data output restriction

When the C11U camera captures images with significant intervals between captures, frame data containing a high level of dark current noise [Note 1] may be output from the camera. To remove frame data affected by dark current, the first two frames immediately after frame output starts are discarded internally by the Camera class. However, this does not apply when the external trigger type is Secondary (Slave, EXT_TRG_SLAVE) and there are long intervals between external triggers, in which case frames affected by dark current may appear.

Depending on the operating system and environment of the host computer, frame acquisition may not be stable. Therefore, the first acquired frame after data output begins may not always be the third frame.

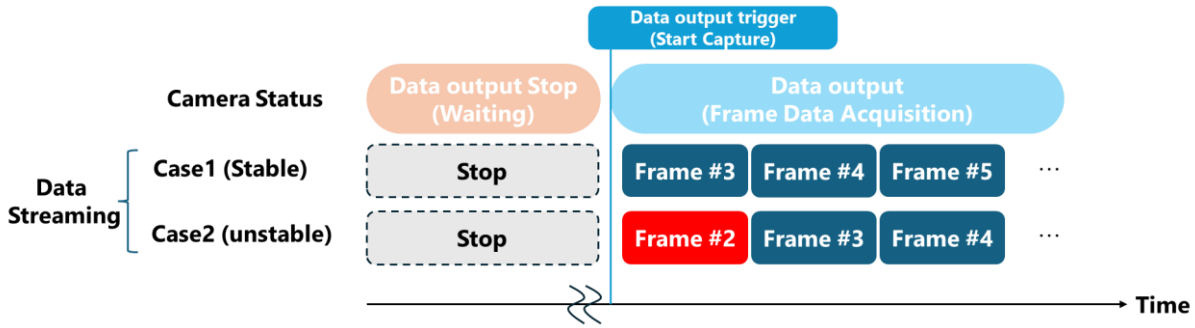


Figure 32. Data output restriction

Note[1]:
Dark current is a type of noise that occurs even when a light-receiving element (pixel) is not exposed to light. This dark current noise mixes with the optical signal that represents the original distance information, reducing the accuracy of distance measurement. Particularly, if pixel operation is stopped for an extended period, the effect of dark current noise may become prominent in the frame data after resumption.

4-2. Restrictions for External Trigger Type is Secondary

When the external trigger type is Secondary (Slave, EXT_TRG_SLAVE), leaving the system in a reception waiting state for a long time without receiving any images may put a load on the host PC. Extended high-load operation can lead to malfunctions or failures, so it is recommended to cancel the reception waiting state if it will be prolonged.

5. Terms of Use and Disclaimer

Please refer to the "C11U User's Guide", "TOPPAN ToF *senSPure*™ SDK API Reference Manual", and other related documents for the terms and conditions of use for products from TOPPAN Holdings Inc. and TOPPAN Inc. (hereinafter referred to as "the Company").

- Unauthorized copying, reproduction, or reprinting of any part or all of this document is strictly prohibited.
- The contents of this document are subject to change without notice.
- While the Company takes every measure to provide accurate information, it does not assume responsibility for errors or omissions. Furthermore, the Company is not liable for any damages resulting from the use of the information contained in this document.
- The Company shall not be liable for any damages, including but not limited to data loss, loss of opportunity, loss of profit, and other incidental, indirect, or consequential damages, arising from the use of this product.
- Product names and company names, and other proper nouns mentioned in this document and related documents, belong to their respective companies or individuals. In this document, the TM (™) and R (®) marks may be omitted. These names are used for identification and explanation purposes within this document only. The Company has no intention to infringe on any of these rights.

Document history

Date	Version	Comment
2024/09/05	1.00	Initial release
2025/03/19	1.10	Initial release for C11U ES Version
2025/06/24	1.11	Support SDK Ver.3.0.4 ・1.5 Operating environment; updated (Added compatibility with Windows 11. JetPack version listed) ・3.3.6.3.11 PlFw::getEvent; updated (Added waiting time when operating as a slave) ・4. Added Usage restrictions 4-1. Restrictions on data output 4-2. Restrictions when external trigger type is Secondary (Slave, EXT_TRG_SLAVE) ・Minor typos
2025/09/09	1.12	Support SDK Ver.3.0.6 1.5 Operating environment; updated (Added compatibility with Ubuntu22.04LTS. JetPack 6.0 listed)

TOPPAN

TOPPAN ホールディングス株式会社 TOF 事業推進センター
TOF Business Development Center, TOPPAN Holdings Inc.

TOPPAN 株式会社 エレクトロニクス事業本部
Electronics Division, TOPPAN Inc.

Location
(日本語) 〒108-8539 東京都港区芝浦 3-19-26 トッパン芝浦ビル
(English) 3-19-26, Shibaura, Minato-ku, Tokyo, 108-8539
E-mail electronics@toppan.co.jp
Website https://www.toppan.com/ja/electronics/device/tof/ (TOPPAN Inc.)

ToF camera product support center
For support related to ToF camera products, please contact the designated support center.
E-mail btop_support@toppan.co.jp