OMRON

Displacement Sensor
ZW-8000/7000/5000 series
Confocal Fiber Type
Displacement Sensor

Communication Library
Reference Manual

ZW-8000□/7000□/5000□

Technology Introduction Guide

SYSMAC always in control

Z364-E1-04

Table of Contents

1. Revision History

Revision Symbol	Revision Date	Reason for Revision and Revised Page
01	April 1, 2016	First edition
03	July 9, 2018	Corresponding to MacOS. Adding some libraries. Support ZW-8000.
04	April 1, 2025	Updated operating environment.

2. Software License Agreement (translation)

This Software License Agreement (“Agreement”) is a binding agreement between you (“User”) and OMRON Corporation (“OMRON”) on the terms and conditions of the license of this Software.

1. The term “Software” used in this Agreement means the computer programs and related documentations identified below. All title, ownership rights and intellectual property rights in and to the Software and any copies thereof remain the sole property of OMRON or its third party suppliers and shall not be assigned to the User under this Agreement.

The Software: ZW-8000/7000/5000 series Communication Library

2. OMRON grants to the User a non-exclusive, non-transferable and limited license as follows:
(1)to copy the Software solely for manufacturing User’s products associated with OMRON’s product ("User’s Products”)
(2)to use the Software integrated into the User’s Products
(3)to distribute the Software integrated into the User’s Products to customers of User

3. Except as specified in article 2, the User shall not sub-license, assign, rent nor lease the Software to any third party without prior written consent of OMRON.

4. The User may not decompile, disassemble and reverse engineer nor otherwise attempt to derive source code nor other confidential information from the Software.

5. The User shall treat any information contained in the Software as confidential and shall not disclose it

to any third party. This obligation shall survive the termination of this Agreement.

6. OMRON warrants the Software will perform substantially in accordance with the specifications.

7. OMRON provides the Software for the use as is and OMRON DOES NOT PROVIDE ANY WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTY OF MERCHANTABILITY OR FITNESS FOR PARTICULAR PURPOSE.
IN NO EVENT, OMRON WILL BE LIABLE FOR ANY LOST PROFITS OR OTHER INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS AGREEMENT OR USE OF THE SOFTWARE.

8. OMRON shall have no liability for any claim of patent, trade secret or copyright infringement based on the use of the Software.

9. If the User breaches this Agreement, OMRON may terminate this Agreement upon notice to the User. In that event, the User shall return the Software.

10. The formation, validity, construction and performance of this Agreement shall be governed and interpreted by and in accordance with the laws of Japan.

11. Any and all dispute, controversy or difference which may arise between the parties hereto out of or in relation to or in connection with this Agreement shall be finally settled by arbitration in Tokyo in accordance with the Arbitration Rules of the Japan Commercial Arbitration Association. The award rendered by arbitrator(s) shall be final binding upon the parties hereto.

This is a translation of the original agreement in Japanese. In the event of any discrepancy, the original agreement in Japanese shall prevail.

3. Introduction

3.1. Introduction

Thank you for purchasing ZW-8000/7000/5000 Series product.
The ZW-8000/7000/5000 series communication library provides the communication interface for controlling the ZW-8000/7000/5000 series from a user application (32-bit/64-bit DLL). For more specific usage, refer to the sample programs.
This manual provides information regarding functions, performance and operating methods that are required

for using ZW-8000/7000/5000 Series product. When using ZW-8000/7000/5000 Series product, be sure to observe the following:

· ZW-8000/7000/5000 Series product must be operated by personnel knowledgeable in electrical engineering.
· To ensure correct use, please read this manual thoroughly to deepen your understanding of the product.
· Please keep this manual in a safe place so that it can be referred to whenever necessary.

The contents of this manual, including product specifications, are subject to change based on improvements of the product without prior notice. Your understanding is appreciated
We are committed to providing precise information. Should you have any questions or con-cerns regarding the contents of this document, please do not hesitate to contact us. When you contact us, please be sure to provide us with the Catalog number printed on the back cover.

3.2. Terms and Conditions Agreement

3.2.1. Warranty, Limitations of Liability

■ Warranties

● Exclusive Warranty

Omron's exclusive warranty is that the Products will be free from defects in materials and workmanship for a period of twelve months from the date of sale by Omron (or such other period expressed in writing by Omron). Omron disclaims all other warranties, express or implied.

● Limitations

OMRON MAKES NO WARRANTY OR REPRESENTATION, EXPRESS OR IMPLIED, ABOUT NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OF THE PRODUCTS. BUYER ACKNOWLEDGES THAT IT ALONE HAS DETERMINED THAT THE PRODUCTS WILL SUITABLY MEET THE REQUIREMENTS OF THEIR INTENDED USE.

Omron further disclaims all warranties and responsibility of any type for claims or expenses based on infringement by the Products or otherwise of any intellectual property right.

● Buyer Remedy

Omron's sole obligation hereunder shall be, at Omron's election, to (i) replace (in the form originally shipped with Buyer responsible for labor charges for removal or replacement thereof) the non-complying Product, (ii) repair the non-complying Product, or (iii) repay or credit Buyer an amount equal to the purchase price of the non-complying Product; provided that in no event shall

Omron be responsible for warranty, repair, indemnity or any other claims or expenses regarding the Products unless Omron's analysis confirms that the Products were properly handled, stored, installed and maintained and not subject to contamination, abuse, misuse or inappropriate modification. Return of any Products by Buyer must be approved in writing by Omron before shipment. Omron Companies shall not be liable for the suitability or unsuitability or the results from the use of Products in combination with any electrical or electronic components, circuits, system assemblies or any other materials or substances or environments. Any advice, recommendations or information given orally or in writing, are not to be construed as an amendment or addition to the above warranty.

See http://www.omron.com/global/ or contact your Omron representative for published information.

■ Limitation on Liability; Etc

OMRON COMPANIES SHALL NOT BE LIABLE FOR SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, LOSS OF PROFITS OR PRODUCTION OR COMMERCIAL LOSS IN ANY WAY CONNECTED WITH THE PRODUCTS, WHETHER SUCH CLAIM IS BASED IN CONTRACT, WARRANTY, NEGLIGENCE OR STRICT LIABILITY.
Further, in no event shall liability of Omron Companies exceed the individual price of the Product on which liability is asserted.

3.2.2.Application Considerations

■ Suitability of Use

Omron Companies shall not be responsible for conformity with any standards, codes or regulations which apply to the combination of the Product in the Buyer's application or use of the Product. At Buyer's request, Omron will provide applicable third party certification documents identifying ratings and limitations of use which apply to the Product. This information by itself is not sufficient for a complete determination of the suitability of the Product in combination with the end product, machine, system, or other application or use. Buyer shall be solely responsible for determining appropriateness of the particular Product with respect to Buyer's application, product or system. Buyer shall take application responsibility in all cases.
NEVER USE THE PRODUCT FOR AN APPLICATION INVOLVING SERIOUS RISK TO LIFE OR PROPERTY WITHOUT ENSURING THAT THE SYSTEM AS A WHOLE HAS BEEN DESIGNED TO ADDRESS THE RISKS, AND THAT THE OMRON PRODUCT(S) IS PROPERLY RATED AND INSTALLED FOR THE INTENDED USE WITHIN THE OVERALL EQUIPMENT OR SYSTEM.

■ Programmable Products

Omron Companies shall not be responsible for the user’s programming of a programmable Product, or any consequence thereof.

3.2.3. Disclaimers

■ Performance Data

Data presented in Omron Company websites, catalogs and other materials is provided as a guide for the user in determining suitability and does not constitute a warranty. It may represent the result of Omron’s test conditions, and the user must correlate it to actual application requirements. Actual performance is subject to the Omron’s Warranty and Limitations of Liability.

■ Change in Specifications

Product specifications and accessories may be changed at any time based on improvements and other reasons. It is our practice to change part numbers when published ratings or features are changed, or when significant construction changes are made. However, some specifications of the Product may be changed without any notice. When in doubt, special part numbers may be assigned to fix or establish key specifications for your application. Please consult with your Omron’s representative at any time to confirm actual specifications of purchased Product.

■ Error and Omissions

Information presented by Omron Companies has been checked and is believed to be accurate; however, no responsibility is assumed for clerical, typographical or proofreading errors or omissions.

3.3. Precautions on Safety

For details on the precautions on safety, refer to the following manual:
"Precautions on Safety" described in Displacement Sensor ZW-8000/7000/5000 series Confocal Fiber Type
Displacement Sensor User's Manual (Z362-E1-01)

3.4. Precautions for Safe Use

For details on the precautions for safe use, refer to the following manual:

"Precautions for Safe Use" described in Displacement Sensor ZW-8000/7000/5000 series Confocal Fiber Type Displacement Sensor User's Manual (Z362-E1-01)

3.5. Precautions for Correct Use

For details on the precautions for correct use, refer to the following manual:
"Precautions for Correct Use" described in Displacement Sensor ZW-8000/7000/5000 series Confocal Fiber Type Displacement Sensor User's Manual (Z362-E1-01)

3.6. Regulations and Standards

For details on the regulations and standards, refer to the following manual:
"Regulations and Standards" described in Displacement Sensor ZW-8000/7000/5000 series Confocal Fiber Type Displacement Sensor User's Manual (Z362-E1-01)

3.7. Copyrights and Trademarks

3.8. Related Manuals

The following manual is related to Controllers. Use this manual for reference.

Cat. No.	Manual name	Description	Application
W504	Sysmac Studio Version 1 Operation Manual	Describes the operating procedures of the Sysmac Studio.	Learning about the operating procedures and functions of the Sysmac Studio.

Z362	Confocal Fiber Type Displacement Sensor ZW-8000/7000/5000 series User's Manual	Describes how to set-up of Confocal Fiber Type Displacement Sensor of ZW-7000/5000 series.	To learn how to set-up of Confocal Fiber Type Displacement Sensor of ZW-8000/7000/5000 series.
Z363	Confocal Fiber Type Displacement Sensor ZW-8000/7000/5000 series User's Manual for Communication Setting	Describes how to use communication settings of Confocal Fiber Type Displacement Sensor of ZW-7000/5000 series.	To learn how to use communication settings of Confocal Fiber Type Displacement Sensor of ZW-8000/7000/5000 series.

4. Operating Environment

4.1. Windows

DLL Software Ver.4.000:

Operating system (OS)	**Windows 10 (32bit/64bit edition)/ Windows 11 (64bit edition)**
CPU	**Windows personal computer with an Intel® Core™ i5-3320M (2.60GHz) CPU or better. Intel® Core™ Ultra 5 135U (1.6GHz) or faster is recommended.**
Main memory	**4GB or more 16GB or more is recommended.**
Hard disk	**Free disk space of 1.6GB or more**
Communication port	**Ethernet port**
Supported languages	**Japanese, English**

DLL Software Ver.3.001 (*1):

Operating system (OS)	**Windows 7 (32bit/64bit edition)/ Windows 8 (32bit/64bit edition)/ Windows 8.1 (32bit/64bit edition)/ Windows Embedded Standard 7 (32bit/64bit edition)/ Windows Embedded 8 Standard (32bit/64bit edition)**
CPU	**Windows personal computer with an Intel® Celeron® 540 (1.8GHz) CPU or better. Intel® Core™ i5 M520 (2.4GHz) or faster is recommended.**

Main memory	**2GB or more** **4GB or more is recommended.**
Hard disk	**Free disk space of 1.6GB or more**
Communication port	**Ethernet port**
Supported languages	**Japanese, English**

*1: DLL Software Ver.3.001 operates identically to DLL Software Ver.3.000.

4.1.1. Runtime Environment

Here is the environment that is necessary to run an application that makes use of the ZW-8000/7000/5000 series communication library.

4.1.2. Microsoft .NET Framework 4 Client Profile

This is the runtime that is required for the operation of DLL.

Operation has been verified using Microsoft .NET Framework 4.6.2 and 4.8 Advanced Services.

Execute dotNetFx40_Client_x86_x64.exe, and then install the software.

4.2. MacOS

DLL Software Ver.3.001 (*1):

Operating system (OS)	OS X 12（64bit edition）
CPU	MacOS personal computer with an Intel® Core™ i5 M520 (2.0GHz) or faster is recommended.
Main memory	8GB or more is recommended.
Hard disk	Free disk space of 1.6GB or more
Communication port	Ethernet port
Supported languages	Japanese, English

*1: DLL Software Ver.3.001 operates identically to DLL Software Ver.3.000.

5. File Composition

5.1. Windows

DSComm.dll	DLL body
Source	Source is a folder of sample source by C#.
Sample	Sample is a folder of sample software(.exe).
Document	Document is a folder. Documents related sample program cleated by C# is stored.

5.2. MacOS

DSComm.dll	DLL body
Source	Source is a folder of sample source by Swift.
Sample	Sample is a folder of sample software(.exe).
Document	Document is a folder. Documents related sample program cleated by Swift is stored.

6. Embedding Method

6.1. File Composition

Here is the file necessary for execution.

Place the following file in the same folder as that of an executable file.

6.1.1. C#

- DSComm.dll

6.1.2. Swift

- DSComm.framework

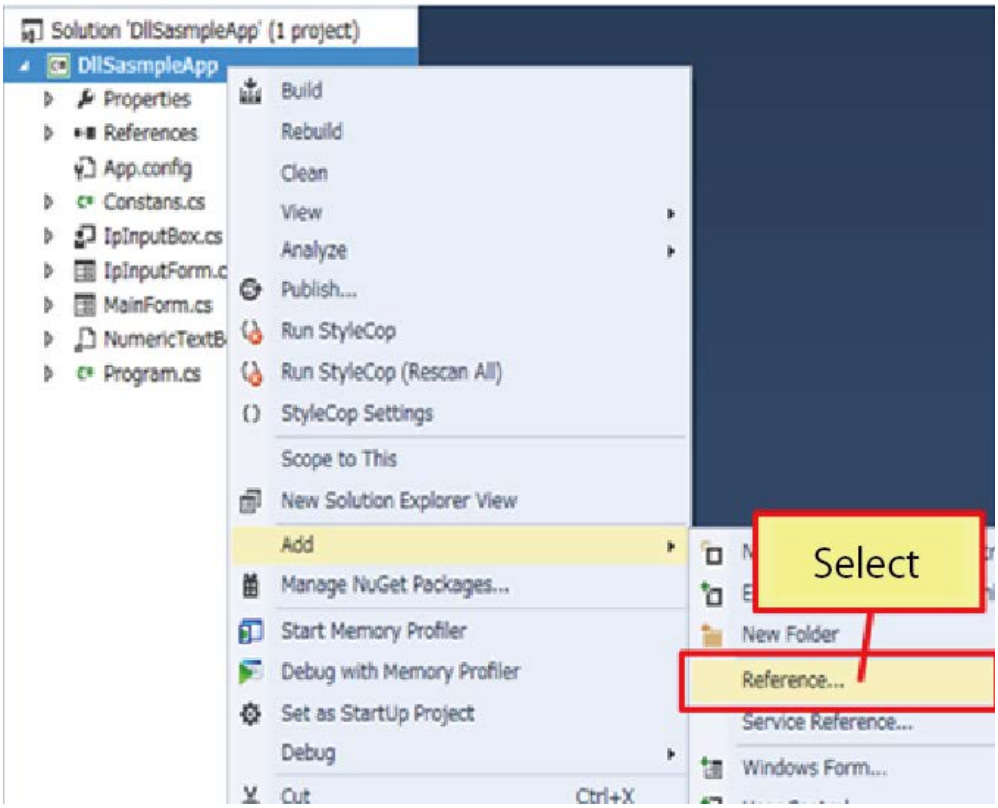
6.2. Link

6.2.1.C#

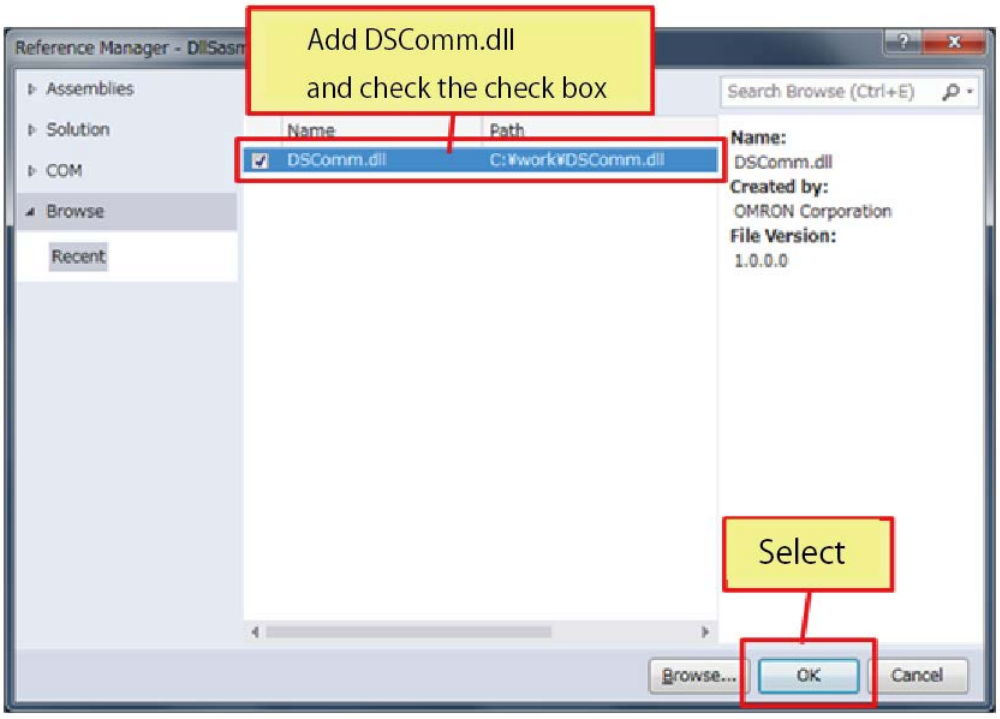
6.2.1.1. Reference

In the reference settings on the project, select "DisplacementSensorSDK(DSComm.dll)."

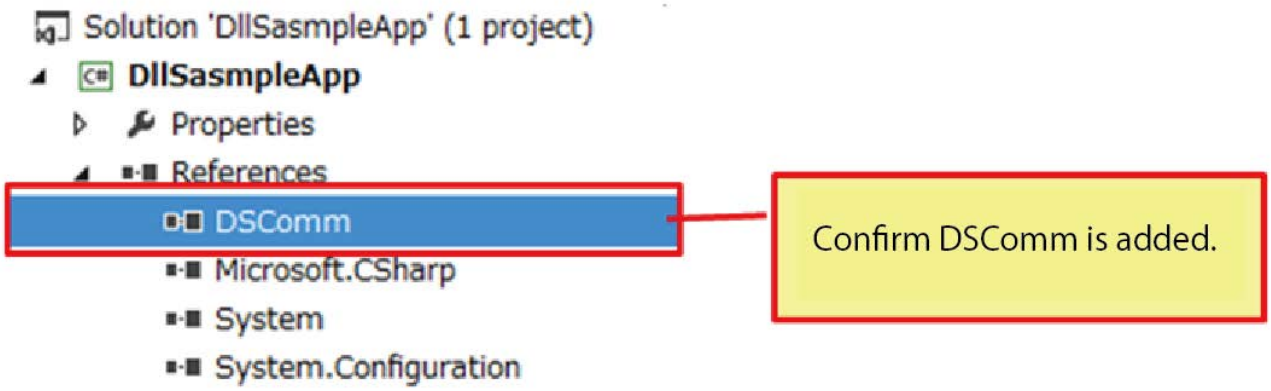
Step1



Step2



Step3



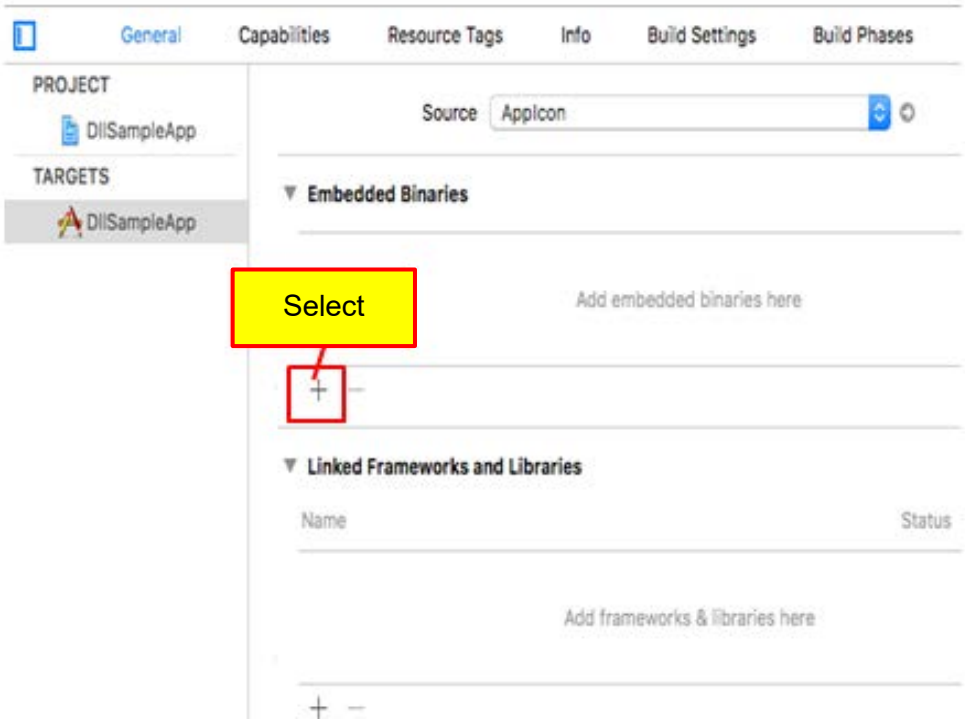
6.2.2.Swift

6.2.2.1. Reference

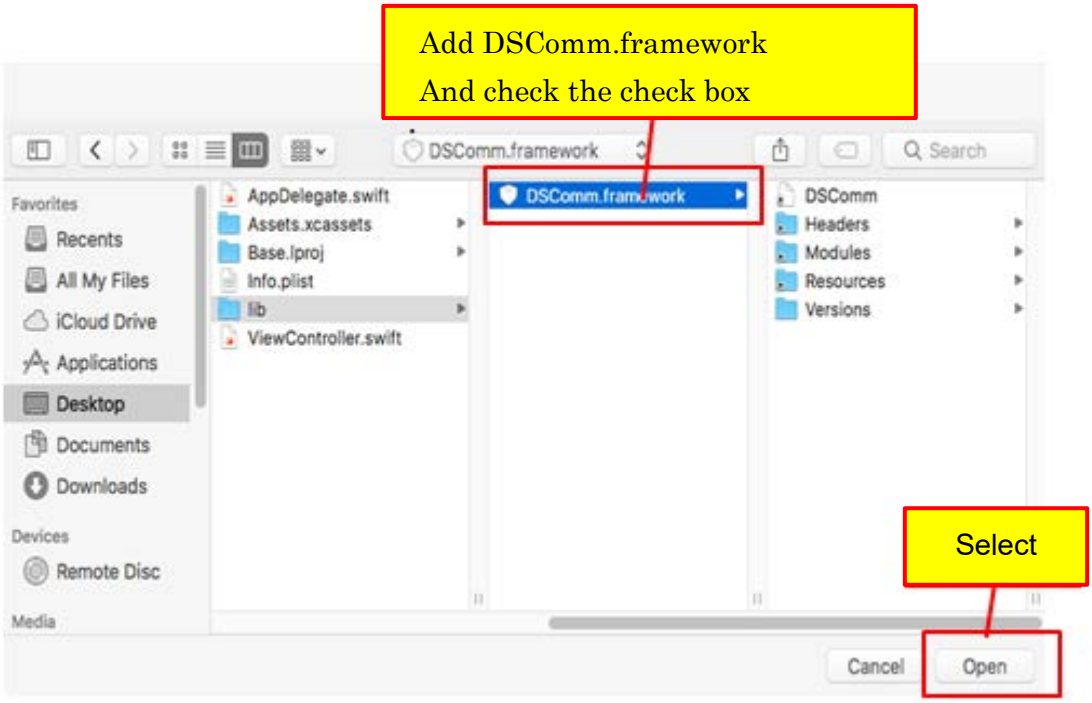
In the reference settings on the project, select "DisplacementSensorSDK(DSComm.dll)."

Select “General” – “Embedded Binaries”, and add “DSComm.framework”.

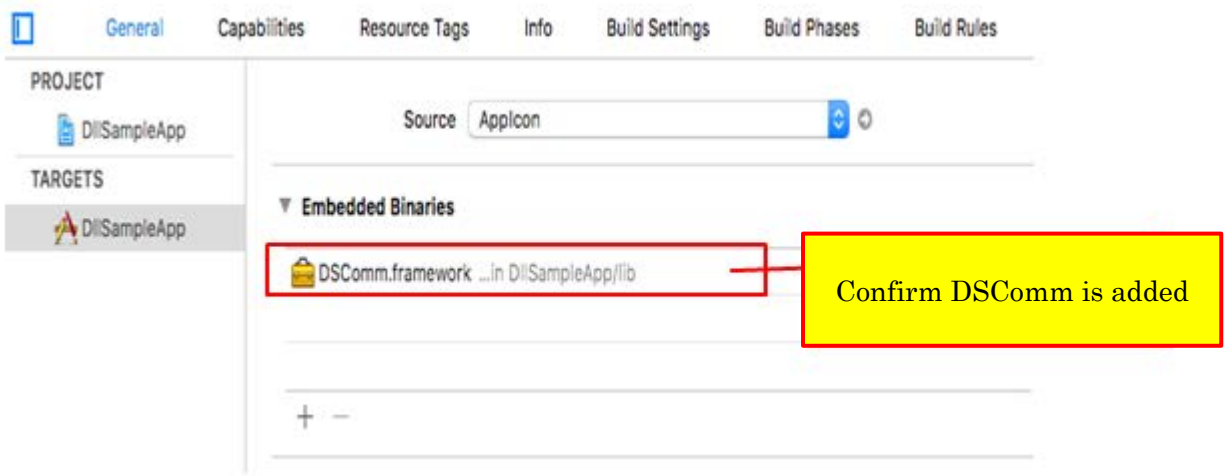
Step.1



Step.2


Add DSComm.framework
And check the check box
DSComm.framework
Search
Favorites
Recents
All My Files
iCloud Drive
Applications
Desktop
Documents
Downloads
Devices
Remote Disc
Media
AppDelegate.swift
Assets.xcassets
Base.lproj
Info.plist
lib
ViewController.swift
DSComm.framework
DSComm
Headers
Modules
Resources
Versions
Select
Cancel
Open

Step.3


General
Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules
PROJECT
DllSampleApp
TARGETS
DllSampleApp
Source
AppIcon
Embedded Binaries
DSComm.framework ...in DllSampleApp/lib
Confirm DSComm is added

7. Datatype

This document is based on the assumption that the datatype of the variables are defined as follows:

7.1. Windows

bool	**Boolean value (true or false)**
byte	**Unsigned 8bit integer**
short	**Signed 16bit integer**
ushort	**Unsigned 16bit integer**
int	**Signed 32bit integer**
uint	**Unsigned 32bit integer**
string	**Unicode character sequence**

7.2. MacOS

bool	Boolean value (true or false)
UInt8	Unsigned 8bit integer
Int16	Signed 16bit integer
UInt16	Unsigned 16bit integer
Int32	Signed 32bit integer
UInt32	Unsigned 32bit integer
String	Unicode character sequence

8. Structure Definitions of Constants and Data Classes

8.1. Constant Definitions

Name	Version number specification

Definition (Windows)	enum Version {
	ZW2,
	};

Definition (MacOS)	enum Version: Int32 {
	case ZW2 // ZW2
	}

Description	Used for specifying the corresponding version in creating an instance of DSComm.
Remark	—

Name	Task number specification

Definition (Windows)	enum Task {
	T1 = 0, // TASK 1
	T2 = 1, // TASK 2
	T3 = 2, // TASK 3
	T4 = 3, // TASK 4
	ALL = 4, // TASK 1 to 4
	};

Definition (MacOS)	enum Task: Int32 {
	case T1 = 0, // TASK 1
	case T2 = 1, // TASK 2
	case T3 = 2, // TASK 3
	case T4 = 3, // TASK 4
	case ALL = 4, // TASK 1～4
	}

Description	Used for specifying the target task in a method for control.
Remark	—

Name	Bank number specification

Definition (Windows)	enum Bank {
	B1 = 0, // BANK 1
	B2 = 1, // BANK 2
	B3 = 2, // BANK 3
	B4 = 3, // BANK 4

	B5 = 4, // BANK 5
	B6 = 5, // BANK 6
	B7 = 6, // BANK 7
	B8 = 7, // BANK 8
	B9 = 8, // BANK 9
	B10 = 9, // BANK 10
	B11 = 10, // BANK 11
	B12 = 11, // BANK 12
	B13 = 12, // BANK 13
	B14 = 13, // BANK 14
	B15 = 14, // BANK 15
	B16 = 15, // BANK 16
	B17 = 16, // BANK 17
	B18 = 17, // BANK 18
	B19 = 18, // BANK 19
	B20 = 19, // BANK 20
	B21 = 20, // BANK 21
	B22 = 21, // BANK 22
	B23 = 22, // BANK 23
	B24 = 23, // BANK 24
	B25 = 24, // BANK 25
	B26 = 25, // BANK 26
	B27 = 26, // BANK 27
	B28 = 27, // BANK 28
	B29 = 28, // BANK 29
	B30 = 29, // BANK 30
	B31 = 30, // BANK 31
	B32 = 31, // BANK 32
	};

Definition (MacOS)	enum Bank: Int32 { case B1 = 0, // BANK 1 case B2 = 1, // BANK 2 case B3 = 2, // BANK 3 case B4 = 3, // BANK 4 case B5 = 4, // BANK 5 case B6 = 5, // BANK 6 case B7 = 6, // BANK 7 case B8 = 7, // BANK 8

	case B9 = 8, // BANK 9 case B10 = 9, // BANK 10 case B11 = 10, // BANK 11 case B12 = 11, // BANK 12 case B13 = 12, // BANK 13 case B14 = 13, // BANK 14 case B15 = 14, // BANK 15 case B16 = 15, // BANK 16 case B17 = 16, // BANK 17 case B18 = 17, // BANK 18 case B19 = 18, // BANK 19 case B20 = 19, // BANK 20 case B21 = 20, // BANK 21 case B22 = 21, // BANK 22 case B23 = 22, // BANK 23 case B24 = 23, // BANK 24 case B25 = 24, // BANK 25 case B26 = 25, // BANK 26 case B27 = 26, // BANK 27 case B28 = 27, // BANK 28 case B29 = 28, // BANK 29 case B30 = 29, // BANK 30 case B31 = 30, // BANK 31 case B32 = 31, // BANK 32 }
Description	Used for specifying the target bank in handling a bank.
Remark	—

Name	Flag specification

Definition (Windows)	enum Flag {
	OFF = 0, // OFF
	ON = 1, // ON
	};

Definition (MacOS)	enum Flag: Int32 {
	case OFF = 0,// OFF
	case ON = 1, // ON
	}

Description	Used for control by ON/OFF.
Remark	—

Name	Area specification

Definition (Windows)	enum Area {
	A1 = 0, // Area 1
	A2 = 1, // Area 2
	};

Definition (MacOS)	enum Area: Int32 {
	case A1 = 0, // Area 1
	case A2 = 1, // Area 2
	}

Description	Used for specifying the target area for obtaining waveform data.
Remark	Area 2 permits the data acquisition only with the area mode set to "2 area mode."

Name	Output data specification

Definition (Windows)	enum Out {
	O1 = 0, // OUT 1
	O2 = 1, // OUT 2
	O3 = 2, // OUT 3
	O4 = 3, // OUT 4
	};

Definition (MacOS)	enum Out: Int32 {
	case O1 = 0, // OUT 1
	case O2 = 1, // OUT 2
	case O3 = 2, // OUT 3
	case O4 = 3, // OUT 4
	}

Description	Used for the specifying the output data number for obtained internal logging data.
Remark	—

8.2. Structure Definitions of Data Classes

Name	Measured waveform information

Definition (Windows)	class MeasureWaveData {
	ushort BankNo; // Bank number
	byte AreaMode; // 2 area mode
	ushort AreaNo; // Area number
	int RecivedLight1; // Amount of received light (1st surface in Area1)
	int RecivedLight2; // Amount of received light (2nd surface in Area1)
	int RecivedLight3; // Amount of received light (3rd surface in Area1)
	int RecivedLight4; // Amount of received light (4th surface in Area1)
	ushort MeasurementValuePIX1; // Measurement value (1st surface in Area1) (PIX)
	ushort MeasurementValuePIX2; // Measurement value (2nd surface in Area1) (PIX)
	ushort MeasurementValuePIX3; // Measurement value (3rd surface in Area1) (PIX)
	ushort MeasurementValuePIX4; // Measurement value (4th surface in Area1) (PIX)
	ushort AreaStartPos; // Specify area：Start coordinate
	ushort AreaEndPos; // Specify area：End coordinate
	ushort MaskAreaStartPos; // Specify area：Mask area (start)
	ushort MaskAreaEndPos; // Specify area：Mask area (end)
	ushort FlagAxisPos1; // Graph axis coordinate 1(pix)
	ushort FlagAxisPos2; // Graph axis coordinate 2 (pix)
	ushort FlagAxisPos3; // Graph axis coordinate 3 (pix)
	ushort FlagAxisPos4; // Graph axis coordinate 4 (pix)
	ushort FlagAxisPos5; // Graph axis coordinate 5 (pix)
	uint MeasureRange; // Measurement range (nm)
	ushort MeasurementPeriod; // Measurement cycle
	ushort LightPower; // Amount of emitted light
	ushort RecivedLightAdjust; // Amount of received light
	ushort CurrentOrVoltageValue; // Current / voltage DAC value
	byte CurrentOrVoltageValueState; // Current / voltage status

	int AbsoluteDistance; // Distance
	int Task1Result; // Measurement result of TASK1 (nm)
	int Task2Result; // Measurement result of TASK2 (nm)
	int Task3Result; // Measurement result of TASK3 (nm)
	int Task4Result; // Measurement result of TASK4 (nm)
	int Task1Resolution; // Resolution of TASK1
	int Task2Resolution; // Resolution of TASK2
	int Task3Resolution; // Resolution of TASK3
	int Task4Resolution; // Resolution of TASK4
	int Task1UpperLimitValue; // Upper limit of TASK1
	int Task2UpperLimitValue; // Upper limit of TASK2
	int Task3UpperLimitValue; // Upper limit of TASK3
	int Task4UpperLimitValue; // Upper limit of TASK4
	int Task1LowerLimitValue; // Lower limit of TASK1
	int Task2LowerLimitValue; // Lower limit of TASK2
	int Task3LowerLimitValue; // Lower limit of TASK3
	int Task4LowerLimitValue; // Lower limit of TASK4
	byte ErrorNo; // Error information
	int[] WaveDatas; // Line bright data};

Definition (MacOS)	class MeasureWaveData { var BankNo:UInt16 // Bank number AreaMode:UInt8 // 2 area mode AreaNo:UInt16 // Area number RecivedLight1:Int32 // Amount of received light (1st surface in Area1) RecivedLight2:Int32 // Amount of received light (2nd surface in Area1) RecivedLight3:Int32 // Amount of received light (3rd surface in Area1) RecivedLight4:Int32 // Amount of received light (4th surface in Area1) MeasurementValuePIX1:Uint16 // Measurement value (1st surface in Area1) (PIX) MeasurementValuePIX2:Uint16 // Measurement value (2nd surface in Area1) (PIX) MeasurementValuePIX3:Uint16 // Measurement value (3rd surface in Area1) (PIX) MeasurementValuePIX4:UInt16 // Measurement value

	(4th surface in Area1) (PIX) AreaStartPos:UInt16 // Specify area：Start coordinate AreaEndPos:UInt16 // Specify area：End coordinate MaskAreaStartPos:UInt16 // Specify area：Mask area (start) MaskAreaEndPos:UInt16 // Specify area：Mask area (end) FlagAxisPos1:UInt16 // Graph axis coordinate 1(pix) FlagAxisPos2:UInt16 // Graph axis coordinate 2 (pix) FlagAxisPos3:UInt16 // Graph axis coordinate 3 (pix) FlagAxisPos4:UInt16 // Graph axis coordinate 4 (pix) FlagAxisPos5:UInt16 // Graph axis coordinate 5 (pix) MeasureRange:Uint32 // Measurement range (nm) MeasurementPeriod:UInt16 // Measurement cycle LightPower:UInt16 // Amount of emitted light RecivedLightAdjust:UInt16 // Amount of received light CurrentOrVoltageValue:Uint16 // Current / voltage DAC value CurrentOrVoltageValueState:UInt8 // Current / voltage status AbsoluteDistance:Int32 // Distance Task1Result:Int32 // Measurement result of TASK1 (nm) Task2Result:Int32 // Measurement result of TASK2 (nm) Task3Result:Int32 // Measurement result of TASK3 (nm) Task4Result:Int32 // Measurement result of TASK4 (nm) Task1Resolution:Int32 // Resolution of TASK1 Task2Resolution:Int32 // Resolution of TASK2 Task3Resolution:Int32 // Resolution of TASK3 Task4Resolution:Int32 // Resolution of TASK4 Task1UpperLimitValue:Int32 // Upper limit of TASK1 Task2UpperLimitValue:Int32 // Upper limit of TASK2 Task3UpperLimitValue:Int32 // Upper limit of TASK3 Task4UpperLimitValue:Int32 // Upper limit of TASK4 Task1LowerLimitValue:Int32 // Lower limit of TASK1 Task2LowerLimitValue:Int32 // Lower limit of TASK2 Task3LowerLimitValue:Int32 // Lower limit of TASK3 Task4LowerLimitValue:Int32 // Lower limit of TASK4 ErrorNo:Uint8 // Error information WaveDatas:[Int32] // Line bright data};
Description	Information relating to the measured waveform.
Remark	For ZW-7000/5000 Error information:

	bit[0]-[2] : Error received light quantity (0:Stability light quantity, 1:Adjusting light quantity, 2:Light quantity upper limit exceeded, 3:Light quantity lower limit not reached, 4:LIGHT OFF, 5: Mutual interference prevention OFF) bit[3]:System Error bit[4]:reserved bit[5]: Error the number of edge bit[6]:STAB State bit[7]:Error sensor calibration Line bright data : The size of array is 256.

Name	Measured waveform information 2

Definition (Windows)	class MeasureWaveData2 {
	ushort BankNo; // Bank number
	byte AreaMode; // 2 area mode
	ushort AreaNo; // Area number
	int RecivedLight1; // Amount of received light (1st surface in Area1)
	int RecivedLight2; // Amount of received light (2nd surface in Area1)
	int RecivedLight3; // Amount of received light (3rd surface in Area1)
	int RecivedLight4; // Amount of received light (4th surface in Area1)
	ushort MeasurementValuePIX1; // Measurement value (1st surface in Area1) (PIX)
	ushort MeasurementValuePIX2; // Measurement value (2nd surface in Area1) (PIX)
	ushort MeasurementValuePIX3; // Measurement value (3rd surface in Area1) (PIX)
	ushort MeasurementValuePIX4; // Measurement value (4th surface in Area1) (PIX)
	ushort AreaStartPos; // Specify area：Start coordinate
	ushort AreaEndPos; // Specify area：End coordinate
	ushort MaskAreaStartPos; // Specify area：Mask area (start)
	ushort MaskAreaEndPos; // Specify area：Mask area (end)
	ushort FlagAxisPos1; // Graph axis coordinate 1(pix)
	ushort FlagAxisPos2; // Graph axis coordinate 2 (pix)

	ushort FlagAxisPos3; // Graph axis coordinate 3 (pix)
	ushort FlagAxisPos4; // Graph axis coordinate 4 (pix)
	ushort FlagAxisPos5; // Graph axis coordinate 5 (pix)
	uint MeasureRange; // Measurement range (nm)
	ushort MeasurementPeriod; // Measurement cycle
	ushort LightPower; // Amount of emitted light
	ushort RecivedLightAdjust; // Amount of received light
	ushort CurrentOrVoltageValue; // Current / voltage DAC value
	byte CurrentOrVoltageValueState; // Current / voltage status
	int AbsoluteDistance; // Distance
	int Task1Result; // Measurement result of TASK1 (nm)
	int Task2Result; // Measurement result of TASK2 (nm)
	int Task3Result; // Measurement result of TASK3 (nm)
	int Task4Result; // Measurement result of TASK4 (nm)
	int Task1Resolution; // Resolution of TASK1
	int Task2Resolution; // Resolution of TASK2
	int Task3Resolution; // Resolution of TASK3
	int Task4Resolution; // Resolution of TASK4
	int Task1UpperLimitValue; // Upper limit of TASK1
	int Task2UpperLimitValue; // Upper limit of TASK2
	int Task3UpperLimitValue; // Upper limit of TASK3
	int Task4UpperLimitValue; // Upper limit of TASK4
	int Task1LowerLimitValue; // Lower limit of TASK1
	int Task2LowerLimitValue; // Lower limit of TASK2
	int Task3LowerLimitValue; // Lower limit of TASK3
	int Task4LowerLimitValue; // Lower limit of TASK4
	byte ErrorNo; // Error information
	ushort CenterPosition1; // Center position of edge 1 track
	ushort CenterPosition2; // Center position of edge 1 track
	ushort CenterPosition3; // Center position of edge 1 track
	ushort CenterPosition4; // Center position of edge 1 track
	int[] WaveDatas; // Line bright data};

Definition (MacOS)	class MeasureWaveData { var BankNo:UInt16 // Bank number AreaMode:UInt8 // 2 area mode AreaNo:UInt16 // Area number RecivedLight1:Int32 // Amount of received light (1st surface in Area1)

	RecivedLight2:Int32 // Amount of received light (2nd surface in Area1) RecivedLight3:Int32 // Amount of received light (3rd surface in Area1) RecivedLight4:Int32 // Amount of received light (4th surface in Area1) MeasurementValuePIX1:Uint16 // Measurement value (1st surface in Area1) (PIX) MeasurementValuePIX2:Uint16 // Measurement value (2nd surface in Area1) (PIX) MeasurementValuePIX3:Uint16 // Measurement value (3rd surface in Area1) (PIX) MeasurementValuePIX4:UInt16 // Measurement value (4th surface in Area1) (PIX) AreaStartPos:UInt16 // Specify area：Start coordinate AreaEndPos:UInt16 // Specify area：End coordinate MaskAreaStartPos:UInt16 // Specify area：Mask area (start) MaskAreaEndPos:UInt16 // Specify area：Mask area (end) FlagAxisPos1:UInt16 // Graph axis coordinate 1(pix) FlagAxisPos2:UInt16 // Graph axis coordinate 2 (pix) FlagAxisPos3:UInt16 // Graph axis coordinate 3 (pix) FlagAxisPos4:UInt16 // Graph axis coordinate 4 (pix) FlagAxisPos5:UInt16 // Graph axis coordinate 5 (pix) MeasureRange:Uint32 // Measurement range (nm) MeasurementPeriod:UInt16 // Measurement cycle LightPower:UInt16 // Amount of emitted light RecivedLightAdjust:UInt16 // Amount of received light CurrentOrVoltageValue:Uint16 // Current / voltage DAC value CurrentOrVoltageValueState:UInt8 // Current / voltage status AbsoluteDistance:Int32 // Distance Task1Result:Int32 // Measurement result of TASK1 (nm) Task2Result:Int32 // Measurement result of TASK2 (nm) Task3Result:Int32 // Measurement result of TASK3 (nm) Task4Result:Int32 // Measurement result of TASK4 (nm) Task1Resolution:Int32 // Resolution of TASK1 Task2Resolution:Int32 // Resolution of TASK2 Task3Resolution:Int32 // Resolution of TASK3 Task4Resolution:Int32 // Resolution of TASK4

	Task1UpperLimitValue:Int32 // Upper limit of TASK1 Task2UpperLimitValue:Int32 // Upper limit of TASK2 Task3UpperLimitValue:Int32 // Upper limit of TASK3 Task4UpperLimitValue:Int32 // Upper limit of TASK4 Task1LowerLimitValue:Int32 // Lower limit of TASK1 Task2LowerLimitValue:Int32 // Lower limit of TASK2 Task3LowerLimitValue:Int32 // Lower limit of TASK3 Task4LowerLimitValue:Int32 // Lower limit of TASK4 ErrorNo:Uint8 // Error information CenterPosition1:Uint16 // Center position of edge 1 track CenterPosition2:Uint16 // Center position of edge 1 track CenterPosition3:Uint16 // Center position of edge 1 track CenterPosition4:Uint16 // Center position of edge 1 track WaveDatas:[Int32] // Line bright data};
Description	Information relating to the measured waveform.
Remark	For ZW-8000 Error information: bit[0]-[2] : Error received light quantity (0 : Stability light quantity, 1 : Adjusting light quantity, 2 : Light quantity upper limit exceeded, 3 : Light quantity lower limit not reached, 4 : LIGHT OFF, 5 : Mutual interference prevention OFF) bit[3] : System Error bit[4] : reserved bit[5] : Error the number of edge bit[6] : STAB State bit[7] : Error sensor calibration bit[8] : Eror edge 1 track bit[9] : Eror edge 2 track bit[10] : Eror edge 3 track bit[11] : Eror edge 4 track bit[12] : Eror edge 1 track received light quantity bit[13] : Eror edge 2 track received light quantity bit[14] : Eror edge 3 track received light quantity bit[15] : Eror edge 4 track received light quantity WaveDatas : The size of array is 1024.

Name	Flow data

Definition (Windows)	class FlowData {
	uint OutNo; // OUT number
	bool Timing; // Parallel input：TIMING
	bool Reset; // Parallel input：RESET
	bool LEDOff; // Parallel input：LEDOFF
	bool Zero; // Parallel input：ZERO
	bool Logging; // Parallel input：LOGGING
	bool Sync; // Parallel input：SYNC
	bool Busy; // Parallel output：Busy
	bool Enable; // Parallel output：Enable
	bool Low; // Parallel output：Low
	bool Pass; // Parallel output：Pass
	bool High; // Parallel output：High
	bool TaskStat; // Parallel output：TASKSTAT
	bool LogStat; // Parallel output：LOGSTAT
	bool LogErr; // Parallel output：LOGERR
	bool SyncFlg; // Parallel output：SYNCFLG
	bool Stability; // Parallel output：STABILITY
	bool BufferErr; // Overflow the high-speed data communication bit
	bool FlowStop; // Stop the high-speed data communication
	int MeasureData; // Measurement data
	};

Definition (MacOS)	class FlowData { OutNo:Int32 // OUT number Timing:Bool // Parallel input：TIMING Reset:Bool // Parallel input：RESET LEDOff:Bool // Parallel input：LEDOFF Zero:Bool // Parallel input：ZERO Logging:Bool // Parallel input：LOGGING Sync:Bool // Parallel input：SYNC Busy:Bool // Parallel output：Busy Enable:Bool // Parallel output：Enable Low:Bool // Parallel output：Low Pass:Bool // Parallel output：Pass High:Bool // Parallel output：High TaskStat:Bool // Parallel output：TASKSTAT LogStat:Bool // Parallel output：LOGSTAT

	LogErr:Bool // Parallel output：LOGERR SyncFlg:Bool // Parallel output：SYNCFLG Stability:Bool // Parallel output：STABILITY BufferErr:Bool // Overflow the high-speed data communication bit FlowStop:Bool // Stop the high-speed data communication MeasureData:Int32 // Measurement data };
Description	Information relating to flow data.
Remark	**Measurement data** The unit of measurement data depends on the value of the decimal point information (DecimalInfo). false: nm (nanometer) true: μm (micrometer) Unit of the measurement data differ depending on the information value of a decimal point position (DecimalInfo). false：nm（nanometer） true：μm（micrometer）

8.3. Interface of the Delegate Method

Format(Windows)	void DisConnectDelegate()

Format(MacOS)	protocol DisConnectDelegate {
	func DisConnectDelegate ()
	}

Parameters	None
Return values	—
Description	Method to be called when the communication to the Sensor Controller is disconnected.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

Format(Windows)	void LoggingDataDelegate(List<FlowData> flowDataList)

Format(MacOS)	protocol LoggingDataDelegate {
	func LoggingDataDelegate (flowDataList[FlowData])
	}

Parameters	flowDataList Flow data for each task
Return values	—
Description	Method to be called when periodically output measured values are received.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

8.4. Propaty

Name	HardwareType
Description	This value used to detect the connected controller's hard type(ZW-8000 or others). Offline : ""(blank) Online : The Model of the controller is connected(ex. "ZW-8000")
Supported version	It's impossible that is set this value in the instance.

9. Functions

9.1. List of Methods

9.1.1. Methods Relating to Class

Even if the Sensor Controller is in the system error state, the processing is performed normally.

Method name	General description
DSComm	Constructor
Dispose	Destruction of an object

9.1.2. Establishment and Disconnection of Communication Path to the Controller

Even if the Sensor Controller is in the system error state, the processing is performed normally.

Method name	General description
Open	To establish a connection via Ethernet
Close	To disconnect the connection

9.1.3. System Control

Even if the Sensor Controller is in the system error state, except for "ReturnToFactorySetting," the processing is performed normally.

In the system error state, "ReturnToFactorySetting" can fail (for example, when the head is not connected).

Method name	General description
RebootController	To re-launch the Sensor Controller.
ReturnToFactorySetting	To return to the factory default settings of Sensor Controller.
GetSoftwareVersion	To obtain the version of the Sensor Controller
GetSensorSerialNumber	To obtain the head serial information of Sensor Controller
GetSensorName	To obtain the Sensor Controller name.
SetSensorName	To set the Sensor Controller name
GetError	To obtain the system error number of the Sensor Controller

9.1.4.Measurement Control

If the Sensor Controller is in the system error state, the processing fails.

Method name	General description
ZeroReset	To issue the zero reset
Timing	To issue the timing
Reset	To issue the reset
ClearMemory	To initialize the internal memory
TurnLight	To turn off or light up the measurement light
CalibrationSensor	To perform the calibration of the sensor head

9.1.5.Related to Setting Change and Read Processing

If the Sensor Controller is in the system error state, the processing fails.

Method name	General description
GetSystemData	To obtain the system data of the Sensor Controller
SetSystemData	To send setting values to the system data of the Sensor Controller
GetBankData	To obtain the bank data of the Sensor Controller
SetBankData	To send setting values to the bank data of the Sensor Controller
GetBackupData	To get all bank data and system data at once.
SetBackupData	To set all bank data and system data at once.
InitializeSetting	To initialize the set values of the Sensor Controller
InitializeCurrentBankSetting	To initialize the set values of the current bank
SaveSettings	To reflect the contents of the setting write area to the area for in-operation setting and the area for save.
CopyBank	To copy the current bank
GetActiveBank	To obtain active banks
ChangeActiveBank	To switch active banks

9.1.6.Acquisition of Measurement Results

If the Sensor Controller is in the system error state, the processing fails.

Method name	General description
GetMeasurementValue	To obtain the measured value
GetJudgementValue	To obtain the judgement result
GetMeasureWaveData	To obtain the measured waveform
GetRawImageData	To obtain the received light waveform

9.1.7.Related to Internal Logging Function

If the Sensor Controller is in the system error state, the processing fails.

Method name	General description
StartStorage	To start Internal logging
StopStorage	To stop Internal logging
GetStorageStatus	To obtain the status of Internal logging
GetStorageData	To obtain the measured value after Internal logging

9.1.8.Related to High-Speed Data Communication

If the Sensor Controller is in the system error state, the processing fails.

Method name	General description
PreStartHighSpeedDataCommunication	To prepare for starting the high-speed data communication
StartHighSpeedDataCommunication	To start the high-speed data communication
StopHighSpeedDataCommunication	To stop the high-speed data communication
SingleHighSpeedDataCommunication	To start the high-speed data communication (single)

9.2. Method Reference

9.2.1. Handling Relating to Class

All the return values of the functions in which an error can occur are of the integer type.

In a normal state, 0 (OK) is returned. The return code is represented as a common error code.

For the return codes common to functions, refer to Section 9.1 Common Error Codes.

■ Constructor

Format(Windows)	DSComm(Version version)
Format(MacOS)	init(version: Version)
Parameters	version (in) Version corresponding to the displacement sensor connected
Return values	Instance of DSComm
Description	Constructor
Supported version	All

■ Destruction of an object

Format(Windows)	void Dispose()
Format(MacOS)	func Dispose()
Parameters	—
Return values	—
Description	Destruction of an object Releases only unmanaged resources.
Supported version	All

9.2.2. Establishment and Disconnection of Communication Path to the Sensor Controller

■

■ **Ethernet communication**

Format(Windows)	int Open(byte[] ipAddress, DisConnectDelegate method)
Format(MacOS)	func Open(ipAddress: [UInt8]) -> Int32
Parameters	ipAddress (in) Specify the IP address of the destination. Set it on each octet. <Windows> Ex.) 192.168.250.50 ⇒ new byte[]{0xC0, 0xA8, 0xFA, 0x32}; <MacOS> Ex.) 192.168.250.50 ⇒ [UInt8] = [0xC0, 0xA8, 0xFA, 0x32] method (in) Method to be called when the communication is disconnected. <Windows> delegate void DisConnectDelegate(); <MacOS> func DisConnectDelegate()
Return values	OK ERR_CONNECT ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_APPLICATION
Description	Establishes a connection so as to enable the communication to the displacement sensor connected via Ethernet. If it takes more than 200ms to process the measured waveform acquisition method (GetMeasureWaveData), the effects of the TCP delayed acknowledgement may be the cause. If so, changing the receive buffer size of the socket allows the effects of the TCP delayed acknowledgement to be avoided. To change the receive buffer size, create a DSComm.ini file in the same folder as DSComm.dll to write the setting value. Ex.)rcvBuffSize_WaveData=1024 Predefined value: 512bytes(ZW-7000/5000) , 752byte(ZW-8000)
Supported	ZW-8000/7000/5000 series and ver2.00, or later

version	

■ Disconnection of communication path

Format (Windows)	int Close()
Format (MacOS)	func Close() -> Int32
Parameters	—
Return values	OK ERR_APPLICATION
Description	Disconnects the connection of Ethernet. A call with no connection established does not result in an error.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

9.2.3. System Control

■ Sensor Controller reboot

Format (Windows)	int RebootController()
Format (MacOS)	func RebootController() -> Int32
Parameters	—
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_APPLICATION
Description	Reboots the Sensor Controller.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Return to factory default

Format (Windows)	int RetrunToFactorySetting()
Format (MacOS)	func ReturnToFactorySetting() -> Int32
Parameters	—
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_APPLICATION
Description	Returns all the settings of the Sensor Controller to the factory default.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Acquisition of version

Format (Windows)	int GetSoftwareVersion(out string version)
Format (MacOS)	func GetSoftwareVersion(version: inout String) -> Int32
Parameters	version (out) Version information of the Sensor Controller (8bytes) “ 1 . 0 0 0 ”
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_APPLICATION
Description	Obtains the full name of the version.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Acquisition of head serial information

Format (Windows)	int GetSensorSerialNumber(out string serialNo)
Format (MacOS)	func GetSensorSerialNumber(serialNo: inout String) -> Int32
Parameters	serialNo (out) Sensor header information (8bytes) “ 0 1 2 3 4 5 6 ”
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_APPLICATION
Description	Obtains the head serial information.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Acquisition of Sensor Controller name

Format (Windows)	int GetSensorName(out string sensorName)
Format (MacOS)	func GetSensorName(sensorName: inout String) -> Int32
Parameters	sensorName (out) Name of the displacement sensor (up to 32bytes) “ Z W - 7 0 0 0 ”
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_APPLICATION
Description	Obtains the Sensor Controller name.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Sensor Controller name setting

Format (Windows)	int SetSensorName(string sensorName)
Format (MacOS)	func GetSensorName(sensorName: inout String) -> Int32
Parameters	sensorName (in) Name of the displacement sensor (character strings of up to 32bytes)
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_APPLICATION
Description	Sets the Sensor Controller name.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Acquisition of system error number

Format (Windows)	int GetError(out ushort systemErrorNum)
Format (MacOS)	func GetError(systemErrorNum: inout UInt16) -> Int32
Parameters	systemErrorNum (out) System error number
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Obtains the system error number of the Sensor Controller.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

9.2.4. Measurement Control

■ Zero reset issue

Format (Windows)	int ZeroReset(Flag flag, Task task)
Format (MacOS)	func ZeroReset(flag: Flag, task: Task) -> Int32
Parameters	flag (in) ON: Zero reset request, OFF: Clear request of zero reset task (in) Task to be processed. Valid values: T1 to T4, ALL
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Issues a zero reset request. If the task to be processed is set not to be measured, an error does not occur.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Timing issue

Format (Windows)	int Timing(Flag flag)
Format (MacOS)	func Timing(flag: Flag) -> Int32
Parameters	flag (in) ON: Timing ON request, OFF: OFF request
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Issues a timing request.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Reset issue

Format (Windows)	int Reset(Flag flag)
Format (MacOS)	func Reset(flag: Flag) -> Int32
Parameters	flag (in) ON: Reset ON request, OFF: OFF request
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Issues a reset request.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Internal memory clear

Format (Windows)	int ClearMemory()
Format (MacOS)	func ClearMemory() -> Int32
Parameters	—
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Initializes the internal memory.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Measurement light illumination

Format (Windows)	int TurnLight(Flag flag)
Format (MacOS)	func TurnLight(flag: Flag) -> Int32
Parameters	flag (in) ON: Request to light up the measurement light, OFF: Request to turn off the light
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Lights up or turns off the LED that emits the measurement light.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Sensor head calibration

Format (Windows)	int CalibrateSensor()
Format (MacOS)	func CalibrateSensor() -> Int32
Parameters	—
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Calibrates the sensor head by measurement sensing.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

9.2.5. Related to Setting Change and Read Processing

■ Acquisition of system data setting value

Format (Windows)	int GetSystemData(int dataNo, out int value)
Format (MacOS)	func GetSystemData(dataNo: Int32, value: inout Int32) -> Int32
Parameters	dataNo (in) Data number value (out) The obtained value is returned.
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Obtains the specified item of the system data from the Sensor Controller.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Transmission of system data setting value

Format (Windows)	int SetSystemData(int dataNo, int value)
Format (MacOS)	func SetSystemData(dataNo: Int32, value: Int32) -> Int32
Parameters	dataNo (in) Data number value (in) Value to be reflected
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Sends the setting value to the specified item of the system data. Power shutdown causes the set values not to be saved, so the set data needs to be saved to the internal memory of the controller. (Apply the "SaveSettings.")
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Acquisition of bank data setting value

Format (Windows)	int GetBankData(int unitNo, int dataNo, out int value)
Format (MacOS)	func GetBankData(unitNo: Int32, dataNo: Int32, value: inout Int32) -> Int32
Parameters	unitNo (in) Unit number dataNo (in) Data number value (out) The obtained value is returned.
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION

Description	Obtains the specified item of bank data from the Sensor Controller.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Transmission of bank data setting value

Format (Windows)	int SetBankData(int unitNo, int dataNo, int value)
Format (MacOS)	func SetBankData(unitNo: Int32, dataNo: Int32, value: Int32) -> Int32
Parameters	unitNo (in) Unit number dataNo (in) Data number Note: For details of the unit number and data number, refer to Section 10.2 Data List of Processing Items. value (in) Value to be reflected
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Sends the setting value to the specified item of the bank data. Power shutdown causes the set values not to be saved, so the set data needs to be saved to the internal memory of the Sensor Controller. (Apply the "SaveSettings.")
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Acquisition of all bank data and system data

Format (Windows)	int GetBackupData(out byte[] binaryData)
Format (MacOS)	func GetBackupData(binaryData: inout [UInt8]) -> Int32
Parameters	binaryData (out) Returns the binary data of the acquired setting data.
Return values	OK ERR_COMMUNICATION

	ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	To get all bank data and system data at once.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Transmission of all bank data and system data

Format (Windows)	int SetBackupData(byte[] binaryData)
Format (MacOS)	func SetBackupData(binaryData: [UInt8]) -> Int32
Parameters	binaryData (in) Binary data of setting data to be transmitted
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	To set all bank data and system data at once. Since the set value is not retained when the power is turned off, it is necessary to save the setting data in the controller's internal memory. (Please use SaveSettings) If the version of the setting data is newer than the version of the controller, or if the setting data type is different from that of the controller, it becomes an error of the setting parameter (ERR_PARAM).
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Set value initialization

Format (Windows)	int InitializeSetting()
Format (MacOS)	func InitializeSetting() -> Int32
Parameters	—
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Initializes all the settings.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Current bank initialization

Format (Windows)	int InitializeCurrentBankSetting()
Format (MacOS)	func InitializeCurrentBankSetting() -> Int32
Parameters	—
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Initializes the current bank data.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Set value save

Format (Windows)	int SaveSettings()
Format (MacOS)	func SaveSettings() -> Int32
Parameters	—
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Saves all the settings to the internal memory.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Current bank copy

Format (Windows)	int CopyBank(Bank srcBankNo, Bank dstBankNo)
Format (MacOS)	func CopyBank(srcBankNo: Bank, dstBankNo: Bank) -> Int32
Parameters	srcBankNo (in) Bank number of the copy source Valid values: B1 to B32 dstBankNo (in) Bank number of the copy destination Valid values: B1 to B32
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Overwrites the copy destination with the bank setting value of the copy source.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Active bank acquisition

Format (Windows)	int GetActiveBank(out Bank bankNo)
Format (MacOS)	func GetActiveBank(srcBankNo: inout Bank) -> Int32
Parameters	bankNo (out) Valid bank number
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Obtains the valid bank number.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Active bank switching

Format (Windows)	int ChangeActiveBank(Bank bankNo)
Format (MacOS)	func ChangeActiveBank(bankNo: Bank) -> Int32
Parameters	bankNo (in) Bank number after switching Valid values: B1 to B32
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Switches the current bank number to a specified bank number. If the "bankNo" is specified to be the same as the active bank number, the active number does not change.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

9.2.6. Acquisition of Measurement Results

■ Acquisition of measurement values

<table>
<tr><td>Format (Windows)</td><td>int GetMeasurementValue(Task task, out int[] measureValue)</td></tr>
<tr><td>Format (MacOS)</td><td>func GetMeasurementValue(task: Task, measureValue: inout [Int32]) -> Int32</td></tr>
<tr><td>Parameters</td><td>task (in)
Task number for the measurement results to be obtained
Valid values: T1 to T4, ALL
measureValue (out)
Including unmeasured tasks, the data of four tasks is stored.
For an unmeasured task, 0 is stored.
For a measurement-impossible task, Int32.MaxValue is stored.
<table>
<tr><td>Indexes of the array</td><td>0</td><td>1</td><td>2</td><td>3</td></tr>
<tr><td>Task corresponding to measurement results</td><td>Task 1</td><td>Task 2</td><td>Task 3</td><td>Task 4</td></tr>
</table></td></tr>
<tr><td>Return values</td><td>OK
ERR_COMMUNICATION
ERR_PARAM
ERR_TIME_OUT
ERR_RUN_MODE
ERR_APPLICATION</td></tr>
<tr><td>Description</td><td>Obtains the latest measurement results (measurement values).</td></tr>
<tr><td>Supported version</td><td>ZW-8000/7000/5000 series and ver2.00, or later</td></tr>
</table>

Acquisition of measured waveform

Format (Windows)	int GetMeasureWaveData(Area area, out MeasureWaveData waveData)
Format (MacOS)	func GetMeasureWaveData(area: Area, waveData: inout MeasureWaveData) -> Int32
Parameters	area (in) Area where the acquisition is performed waveData (out) Data of the measured waveform
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Obtains the latest measured waveform (after processing). For ZW-7000/5000 When called this function under connected the ZW-8000, it will be returned “ERR_NONCOMPLIANT_METHODS”
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

Format (Windows)	int GetMeasureWaveData(Area area, out MeasureWaveData2 waveData)
Format (MacOS)	func GetMeasureWaveData(area: Area, waveData: inout MeasureWaveData2) -> Int32
Parameters	area (in) Area where the acquisition is performed waveData (out) Data of the measured waveform
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Obtains the latest measured waveform (after processing).

	For ZW-8000 When called this function under connected the ZW-7000/5000, it will be returned "ERR_NONCOMPLIANT_METHODS"
Supported version	

■ Acquisition of judgement results

Format (Windows)	int GetJudgementValue(Task task, out int[] judgementValue)
Format (MacOS)	func GetJudgementValue(task: Task, judgementValue: inout [Int32]) -> Int32
Parameters	task (in) Task number for the judgement results to be obtained Valid values: T1 to T4, ALL judgementValue (out) Including non-judgement tasks, the data of four tasks is stored. For a non-judgement task, 0 (PASS) is stored. <Judgement results> PASS: 0 HIGH: 1 LOW: 2 ERROR: 3

Indexes of the array	0	1	2	3
Task corresponding to measurement results	Task 1	Task 2	Task 3	Task 4

Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Obtains the latest measurement results (judgement values).
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Acquisition of received light waveform

Format (Windows)	int GetRawImageData(Area area, out MeasureWaveData waveData)
Format (MacOS)	func GetMeasureWaveData(area: Area, waveData: inout MeasureWaveData) -> Int32
Parameters	area (in) Area where the acquisition is performed waveData (out) Data of the received light waveform
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Obtains the latest received light waveform (unprocessed). For ZW-7000/5000 When called this function under connected the ZW-8000, it will be returned “ERR_NONCOMPLIANT_METHODS”
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

Format (Windows)	int GetRawImageData(Area area, out MeasureWaveData2 waveData)
Format (MacOS)	func GetMeasureWaveData(area: Area, waveData: inout MeasureWaveData2) -> Int32
Parameters	area (in) Area where the acquisition is performed waveData (out) Data of the received light waveform
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Obtains the latest received light waveform (unprocessed). For ZW-8000 When called this function under connected the ZW-7000/5000, it will be returned “ERR_NONCOMPLIANT_METHODS”
Supported version	

9.2.7. Related to Internal Logging

■ Start internal logging

Format (Windows)	int StartStorage(int cycle, int count)
Format (MacOS)	func StartStorage(cycle: Int32, count: Int32, clear: Bool = true) -> Int32
Parameters	cycle (in) Period in which logging data is saved Valid values: 1 to 1000 count (in) Maximum number of logging Valid values: 1 to 2000000
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Starts internal logging
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Acquisition of internal logging

Format (Windows)	int GetStorageStatus(out int status, out int count)
Format (MacOS)	func GetStorageStatus(status: inout Int32, count: inout Int32) -> Int32
Parameters	status (out) The operating status of logging (0: Stop, 1: In operation) is returned. count (out) The number of saved logging data is returned.
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION

Description	Obtains the internal logging information. The operating status and the number of saved logging data are obtained.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

Format (Windows)	int GetStorageStatus(out int status, out int count, out int labelCount)
Format (MacOS)	func GetStorageStatus(status: inout Int32, count: inout Int32, labelCount: inout Int32) -> Int32
Parameters	status (out) The operating status of logging (0: Stop, 1: In operation) is returned. count (out) The number of saved logging data is returned. labelCount(out) The number of label is returned.
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Obtains the internal logging information. The operating status and the number of saved logging data and label are obtained.
Supported version	ZW-8000/7000/5000 series and ver2.10, or later

Format (Windows)	int GetStorageStatus(int labelNo, out int status, out int count)
Format (MacOS)	func GetStorageStatus(labelNo: Int32, status: inout Int32, count: inout Int32) -> Int32
Parameters	labelNo (in) The number of get logging data label. Valid values: 1 to 16777215 status (out) The operating status of logging (0: Stop, 1: In operation) is returned. count (out) The number of saved logging data is returned. labelCount(out)

	The number of label is returned.
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Obtains the internal logging information. The operating status and the number of saved logging data and label are obtained.
Supported version	ZW-8000/7000/5000 series and ver2.10, or later

■ Storage stop

Format (Windows)	int StopStorage()
Format (MacOS)	func StopStorage() -> Int32
Parameters	—
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Stops internal logging.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ Acquisition of internal logging

Format (Windows)	int GetStorageData(Out outNo, out int[] data)
Format (MacOS)	func GetStorageData(outNo: Out, data: inout [Int32]) -> Int32
Parameters	outNo (in) Output data number for the internal logging data to be obtained Valid values: O1 to O4 data (out) The internal logging data corresponding to an output data number specified at "outNo" is returned. The array size will be the maximum number that is set in " StartStorage "; If " StopStorage " is performed during a logging process, it will be the number of saved data that is already logged (can be checked from " GetStorageStatus ").
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Obtains the internal logging data.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

Format (Windows)	int GetStorageData(Out outNo, out int[][] data)
Format (MacOS)	func GetStorageData(outNo: Out, data: inout [[Int32]]) -> Int32
Parameters	outNo (in) Output data number for the internal logging data to be obtained Valid values: O1 to O4 data (out) The internal logging data of all label corresponding to an output data number specified at "outNo" is returned. The array size will be the maximum number that is set in " StartStorage "; If " StopStorage " is performed during a logging process, it will be the number of saved data that is already logged (can be checked from " GetStorageStatus ").
Return values	OK

	ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Obtains the internal logging data of all label.
Supported version	ZW-8000/7000/5000 series and ver2.10, or later

Format (Windows)	int GetStorageData(Out outNo, int labelNo, out int[] data)
Format (MacOS)	func GetStorageData(outNo: Out, labelNo: Int32, data: inout [Int32]) -> Int32
Parameters	outNo (in) Output data number for the internal logging data to be obtained Valid values: O1 to O4 data (out) The internal logging data of all label corresponding to an output data number specified at "outNo" is returned. The array size will be the maximum number that is set in " StartStorage "; If " StopStorage " is performed during a logging process, it will be the number of saved data that is already logged (can be checked from " GetStorageStatus ").
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Obtains the internal logging data of specified label.
Supported version	ZW-8000/7000/5000 series and ver2.10, or later

9.2.8. Related to High-Speed Data Communication

■ Preparation for the start of the high-speed data communication

<table>
<tr><td>Format (Windows)</td><td>int PreStartHighSpeedDataCommunication(bool[] logCtrlFlag, int thinningNum, int saveNum)</td></tr>
<tr><td>Format (MacOS)</td><td>func PreStartHighSpeedDataCommunication(logCtrlFlag: inout [Bool], thinningNum: Int32, saveNum: Int32) -> Int32</td></tr>
<tr><td>Parameters</td><td>logCtrlFlag (in)
True: Target of high-speed output
false: Extension of high-speed output
Array size: 4
<table>
<tr><td>Indexes of the array</td><td>0</td><td>1</td><td>2</td><td>3</td></tr>
<tr><td>Task to be set</td><td>OUT 1</td><td>OUT 2</td><td>OUT 3</td><td>OUT 4</td></tr>
</table>
thinningNum (in)
The number of decimation
Valid values: 0 to 65535
saveNum (in)
The number of saves
Valid values: 0 to 128</td></tr>
<tr><td>Return values</td><td>OK
ERR_COMMUNICATION
ERR_PARAM
ERR_TIME_OUT
ERR_RUN_MODE
ERR_APPLICATION</td></tr>
<tr><td>Description</td><td>Configures the settings for the high-speed data communication.</td></tr>
<tr><td>Supported version</td><td>ZW-8000/7000/5000 series and ver2.00, or later</td></tr>
</table>

■ High-speed data communication start

Format (Windows)	int StartHighSpeedDataCommunication(LoggingDataDelegate method)
Format (MacOS)	func StartHighSpeedDataCommunication(method: LoggingDataDelegate) -> Int32
Parameters	method Flow data acquisition delegate method delegate void LoggingDataDelegate(List<FlowData> flowDataList) flowDataList: Flow data for each task
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Starts the high-speed data communication. Measurement is performed for the set number of sampling and repeated.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ High-speed data communication stop

Format (Windows)	int StopHighSpeedDataCommunication()
Format (MacOS)	func StopHighSpeedDataCommunication() -> Int32
Parameters	—
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Stops the high-speed data communication.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

■ High-speed data communication start (single)

Format	int SingleHighSpeedDataCommunication(out List<FlowData> flowDataList)
Parameters	flowDataList (out) Flow data for each task
Return values	OK ERR_COMMUNICATION ERR_PARAM ERR_TIME_OUT ERR_RUN_MODE ERR_APPLICATION
Description	Starts the high-speed data communication (single). Upon completion of the measurement for the set number of sampling, the communication stops.
Supported version	ZW-8000/7000/5000 series and ver2.00, or later

10. Common Codes

10.1. Common Error Codes

The return values of the IF functions described in "Chapter 8 Functions" are defined as follows:

Definition name	Data	Cause
OK	0x00000000	Successful completion
ERR_PARAM	0x01080610	An error in the set parameters
ERR_RUN_MODE	0x01FF2204	An error in the operation mode
ERR_COMMUNICATION	0x02110100	An error in the reception and transmission
ERR_TIME_OUT	0x02110101	A timeout occurred in the reception
ERR_CONNECT	0x02110104	Connection failed
ERR_APPLICATION	0x12160802	Application error

11. Appendices

11.1. List of System Data

Data	Minimum	Maximum	Unit	Note
Bank number	1	32	-	
2 area mode	0	1	-	0: 1 area mode 1: 2 area mode
Area number	0	1	-	0: waveform (area0) 1: waveform (area1)
Amount of received light (1st surface in Area1)	0	4096	Gradation	
Amount of received light (2nd surface in Area)	0	4096	Gradation	
Amount of received light (3rd surface in Area)	0	4096	Gradation	
Amount of received light (4th surface in Area)	0	4096	Gradation	
Measurement value of 1st surface	0	255*256	pixel	（1/256）[pixel/div]
Measurement value of 2nd surface	0	255*256	pixel	（1/256）[pixel/div]
Measurement value of 3rd surface	0	255*256	pixel	（1/256）[pixel/div]
Measurement value of 4th surface	0	255*256	pixel	（1/256）[pixel/div]
Area setting: Start coordinate	0	255	pixel	
Area setting: End coordinate	0	255	pixel	
Area setting: Mask area (start)	0	255	pixel	
Area setting:	0	255	pixel	

Mask area (end)				
Graph axis of coordinate 1	0	255*256	pixel	（1/256）[pixel /div]
Graph axis of coordinate 2	0	255*256	pixel	（1/256）[pixel /div]
Graph axis of coordinate 3	0	255*256	pixel	（1/256）[pixel/div]
Graph axis of coordinate 4	0	255*256	pixel	（1/256）[pixel /div]
Graph axis of coordinate 5	0	255*256	pixel	（1/256）[pixel /div]
Measuring range (nm)	0	999999999	nm	
Measurement cycle	0	10000	μs	0.1[μs/div]
Amount of emitted light	0	10000	%	0.01[%/div]
Amount of received light	0	65535	Gradation	
Current / voltage DAC value	0	65535	-	Calculates as the followings: Voltage value = (20-4) / (59069-5069)×(DAC value - 5069)＋4 Current value = (10-(-10)) / (50969 -5069) - 10
Current / voltage state	0	1	-	0: Voltage output 1: Current output
Distance	-999999999	999999999	nm	
Measurement result of TASK1 (nm)	-999999999	999999999	nm	
Measurement result of TASK2 (nm)	-999999999	999999999	nm	
Measurement result of TASK3 (nm)	-999999999	999999999	nm	
Measurement result of TASK4 (nm)	-999999999	999999999	nm	
Resolution of TASK1	-999999999	999999999	nm	
Resolution of TASK2	-999999999	999999999	nm	
Resolution of TASK3	-999999999	999999999	nm	
Resolution of TASK4	-999999999	999999999	nm	
Upper limit of TASK1	-999999999	999999999	nm	

Upper limit of TASK2	-999999999	999999999	nm	
Upper limit of TASK3	-999999999	999999999	nm	
Upper limit of TASK4	-999999999	999999999	nm	
Lower limit of TASK1	-999999999	999999999	nm	
Lower limit of TASK2	-999999999	999999999	nm	
Lower limit of TASK3	-999999999	999999999	nm	
Lower limit of TASK4	-999999999	999999999	nm	
Error Information	0	255	-	Attach the following information to each bits: b 0 to 2: Amount of received light (0: Stable 1: Adjust 3: Saturation 4: LIGHT OFF 5: Mutual interference preventing) b 3: System error b 4: Short- circuit of load (0 fixed) b 5: area error b 6: STAB status b 7: Sensor Head verification error
Line bright data	0	4095	nm	4 Byte × 256 pixels

11.2. Flow Data

Data	Minimum	Maximum	Unit	Note
OUT number	0	3	-	0: OUT1 information 1: OUT2 information 2: OUT3 information 3: OUT4 information
TIMING input	0	1	-	0: TIMING input OFF 1: TIMING input ON
RESET input	0	1	-	0: RESET input OFF 1: RESET input ON
LIGHTOFF input	0	1	-	0: LIGHT input OFF 1: LIGHT input ON
ZERO input	0	1	-	0: ZERO input OFF 1: ZERO input ON
LOGGING input	0	1	-	0: LOGGING input OFF 1: LOGGING input ON
SYNC input	0	1	-	0: SYNC input OFF 1: SYNC input ON
BUSY output	0	1	-	0: BUSY output OFF 1: BUSY output ON
ENABLE output	0	1	-	0: ENABLE output OFF 1: ENABLE output ON
LOW output	0	1	-	0: LOW output OFF 1: LOW output ON
PASS output	0	1	-	0: PASS output OFF 1: PASS output ON
HIGH output	0	1	-	0: HIGH output OFF 1: HIGH output ON
TASKSTART output	0	1	-	0: TASKSTAT output OFF 1: TASKSTAT output ON
LOGSTART output	0	1	-	0: LOGSTART output OFF 1: LOGSTART output ON
LOGERR output	0	1	-	0: LOGERR output OFF 1: LOGERR output ON
SYNCFLG output	0	1	-	0: SYNCFLG output OFF 1: SYNCFLG output ON
STABLITY output	0	1	-	0: STABLTY output ON

				1: STABLTY output ON
Overflow the high-speed data communication bit（BUFFER_ERR）	0	1	-	0: No data communication 1: data communication <Note> When the data communication occurs, number of saved data at the preparation of high-seed data communication may be overflowed.
Stop the high-speed data communication	0	1	-	0: ENABLES output OF 1: ENABLES output ON
Measurement data	-999999999	999999999	-	

12. Sample Program

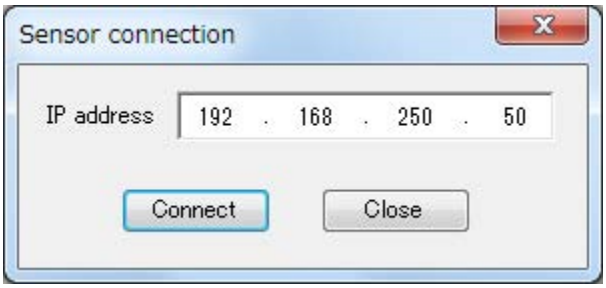
Describes the sample program which attached an example of application creation using communication library.

12.1. User Interface Specification

12.1.1. Window to Enter the IP Address

Enter the IP address of ZW-8000/7000/5000 series Sensor Controller to communicate.

・Pane Layout



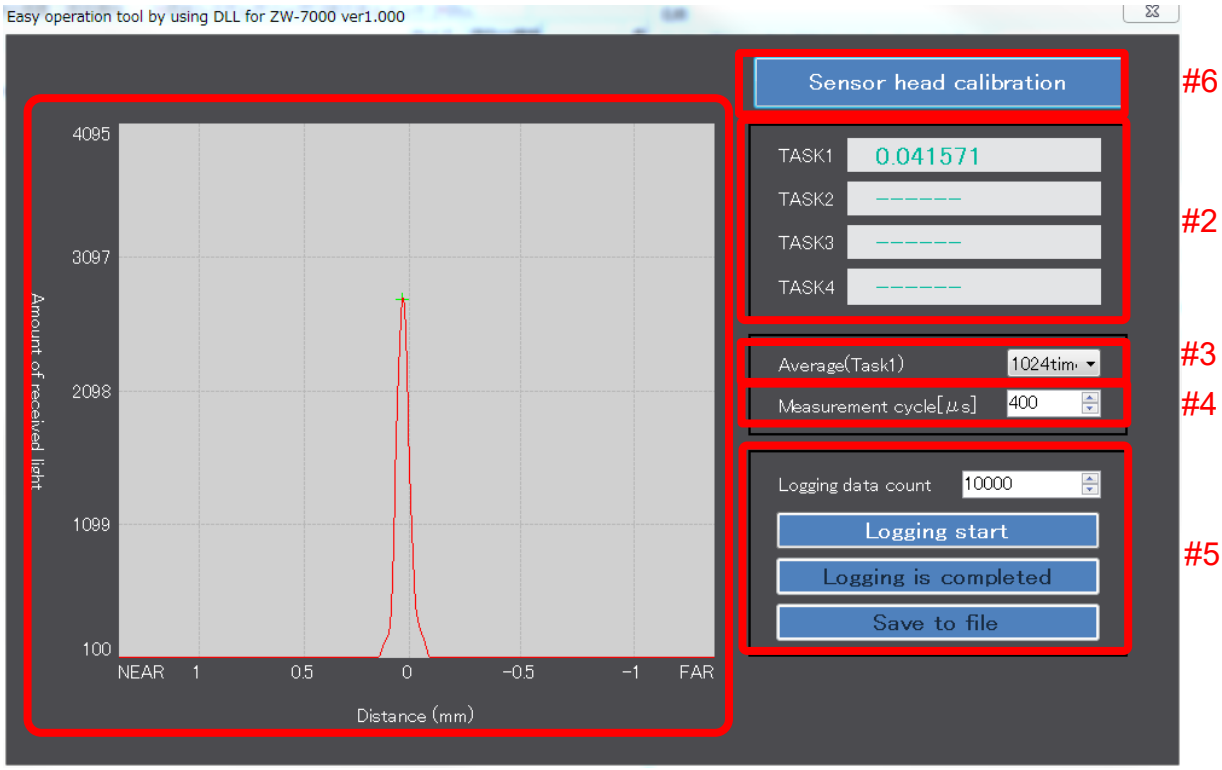
・Function

#	Item	Describe
1	IP address	Enter IP address.
2	Communication	Connect to the ZW-8000/7000/5000 series Sensor Controller, and then displays the main pane.
3	Close	End the application.

・Message Display

#	Display Condition	Message
1	When the communication is failed.	Fail to the communication.

12.1.2. Main Pane



・Function

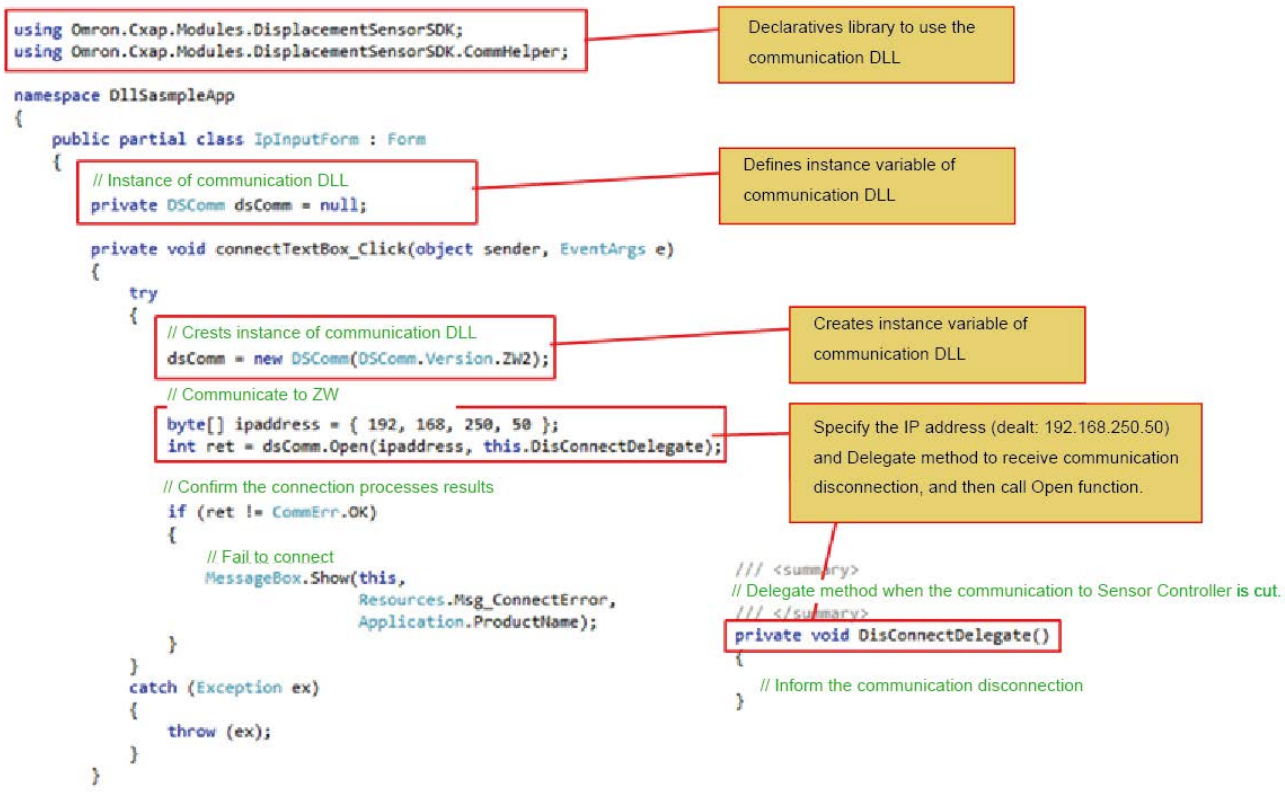
#	Item	Description
1	Monitor of Received light waveform	Displays the received light waveform.
2	Measurement monitor	Displays the selected measurement data.
3	Average count (TASK1)	Set the average count of TASK1.
4	Measurement cycle [μs]	Displays the Measurement cycle of setting item.
5	Control of Internal logging	Control the internal logging.
6	Execute the calibration of Sensor Head	Executes the calibration of Sensor Head.

・Message Display

#	Display Condition	Message
1	When start the internal logging	Internal logging is started.
2	When end the internal logging	Internal logging was completed.

12.2. Sample Source

12.2.1. Communication Establishment



12.2.2. Acquisition of Measurement value

```
MeasureWaveData waveData;
if (beforeWaveRadio.Checked == true)
{
    // Acquires the received light waveform
    retApi = this.dsComm.GetRawImageData(DSComm.Area.A1, out waveData);
    if (retApi != CommErr.OK)
    {
        return;
    }
}
else
{
    // Acquires the measured wave form
    retApi = this.dsComm.GetMeasureWaveData(DSComm.Area.A1, out waveData);
    if (retApi != CommErr.OK)
    {
        return;
    }
}
// Acquires the measurement value
int[] measureData;
retApi = this.dsComm.GetMeasurementValue(DSComm.Task.ALL, out measureData);
if (retApi != CommErr.OK)
{
    return;
}
```

Acquires the received light waveform of specified task.

Acquires the measured waveform data of specified task.

Acquires the measurement value of specified task.

12.2.3. Acquisition and Setting of Bank Data

```
// Acquire the average count from Sensor
retApi = this.dsComm.GetBankData(Constans.UNIT_NO_AVERAGE,
                                 Constans.DATA_NO_AVERAGE,
                                 out value);
if (retApi != CommErr.OK)
{
    MessageBox.Show(this, GetErrMessage(retApi), Application.ProductName);
    return;
}
// Set the average count to Sensor
retApi = this.dsComm.SetBankData(Constans.UNIT_NO_AVERAGE,
                                 Constans.DATA_NO_AVERAGE,
                                 value);
if (retApi != CommErr.OK)
{
    MessageBox.Show(this, GetErrMessage(retApi), Application.ProductName);
    return;
}
```

Acquire the bank data
Specifies the unit number and data number to acquire.

Acquire the bank data
Specifies the unit number and data number to acquire.

OMRON Corporation **Industrial Automation Company**

Kyoto, JAPAN **Contact : www.ia.omron.com**

Regional Headquarters

OMRON EUROPE B.V.
Wegalaan 67-69, 2132 JD Hoofddorp
The Netherlands
Tel: (31) 2356-81-300 Fax: (31) 2356-81-388

OMRON ELECTRONICS LLC
2895 Greenspoint Parkway, Suite 200
Hoffman Estates, IL 60169 U.S.A.
Tel: (1) 847-843-7900 Fax: (1) 847-843-7787

OMRON ASIA PACIFIC PTE. LTD.
438B Alexandra Road, #08-01/02 Alexandra
Technopark, Singapore 119968
Tel: (65) 6835-3011 Fax: (65) 6835-3011

OMRON (CHINA) CO., LTD.
Room 2211, Bank of China Tower,
200 Yin Cheng Zhong Road,
PuDong New Area, Shanghai, 200120, China
Tel: (86) 21-6023-0333 Fax: (86) 21-5037-2388

Authorized Distributor:

Cat. No. Z364-E1-04 0325