Panasonic®

Ultra High-Speed, High-Accuracy Laser Displacement Sensor

HL-C2 Series User's Manual

USB Communication Control

WUME-HLC2USB-12

2019.01 | panasonic.net/id/pidsx/global

Preface

Thank you for purchasing Ultra High-Speed, High-Accuracy Laser Displacement Sensor “HL-C2 Series”.
To fully use this product safely and properly, please read this manual carefully.
See our Website (https://panasonic.net/id/pidsx/global) for the latest information about the product and latest user’s manual.

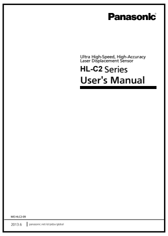
■ Note

1. Please notice that illustrations in this manual might be little different from the actual product.
2. Contents of this manual will be changed without notice due to improvements.
3. This manual and software must not be partially or totally copied or reprinted.
4. If there are any questions, mistakes, paging disorder, or missing pages in this manual, please contact our sales office nearest you.
5. Windows, VisualBasic, and VisualC++ are registered trademarks of Microsoft Corporation in the United States of America and other countries.
6. All other company names and product names in this manual are trademarks or a registered trademarks of their respective companies.
7. We have no responsibility of any results of operations regardless of the above.

Whole USER'S MANUAL Construction

The HL-C2 Series is prepared for the following user's manuals.
Read them as necessary.

HL-C2 Series USER'S MANUAL (PDF)



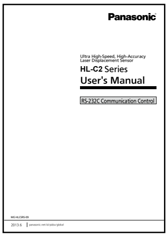
Panasonic

Ultra High-Speed, High-Accuracy Laser Displacement Sensor
HL-C2 Series
User's Manual

ME-HLC2-09
2013.6 | panasonic.net/id/pidsx/global

This manual describes cautions for using HL-C2 Series, and installation method, operation method, function details, specifications, maintenance and inspection method of system components (controller, sensor head compact console).

HL-C2 Series USER'S MANUAL: RS-232C Communication Control (PDF)



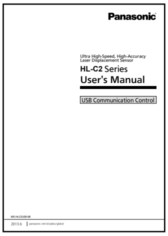
Panasonic

Ultra High-Speed, High-Accuracy Laser Displacement Sensor
HL-C2 Series
User's Manual
RS-232C Communication Control

ME-HLCSRS-09
2013.6 | panasonic.net/id/pidsx/global

The manual describes various commands for controlling the system by PLC or PC using RS-232C communication.

HL-C2 Series USER'S MANUAL: USB Communication Control (PDF)



Panasonic

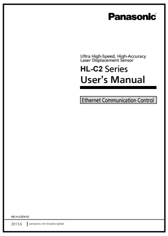
Ultra High-Speed, High-Accuracy Laser Displacement Sensor
HL-C2 Series
User's Manual
USB Communication Control

ME-HLCSUSB-08
2013.6 | panasonic.net/id/pidsx/global

This manual

The manual describes API for controlling the system by PLC or PC using USB communication.

HL-C2 Series USER'S MANUAL: Ethernet Communication Control (PDF)



Panasonic

Ultra High-Speed, High-Accuracy Laser Displacement Sensor

HL-C2 Series

User's Manual

Ethernet Communication Control

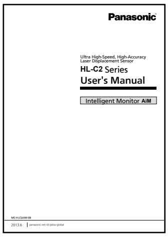
2013.6

This manual explains various settings to acquire measurement information of the HL-C2 system by PLC using Ethernet communication. For detailed explanation concerning the system's functions, precautions for use, etc., refer to the separate "HL-C2 Series USER'S MANUAL".

■ USER'S MANUAL for Intelligent Monitor AiM

The Intelligent Monitor AiM, which contains various useful functions in addition to the compact console, is available when developing PC-based system.

HL-C2 Series USER'S MANUAL: Intelligent Monitor AiM (PDF)



Panasonic

Ultra High-Speed, High-Accuracy Laser Displacement Sensor

HL-C2 Series

User's Manual

Intelligent Monitor AiM

2013.6

This manual is included as a PDF file in the Intelligent Monitor AiM, which can be downloaded on our website. This manual describes installation method, operation method, functional details and error messages of the software.
It also describes an evaluation analysis of HL-C2 Series or use of buffering function and received light intensity waveform display function, which are useful for optimum system setting.

Manual Construction

Contents

Safety Precautions

This product is intended to detect the objects and does not have the control function to ensure safety such as accident prevention.
Do not use the product as a sensing device to protect human body.
Please use the products that comply with local laws and standards for human body protection specified by e.g., OSHA, ANSI and IEC.
Please read this manual carefully before using the product and use it correctly.

■ Symbol Indications

This manual uses symbols to indicate safety precautions, instructions, and reference.
Before reading this manual, fully understand the meanings of these indications.

⚠WARNING	"WARNING" indicates the possibility that death or serious injury could result if a handling error occurs.
⚠CAUTION	"CAUTION" indicates the possibility that the user could be injured or property could be damaged if a handling error occurs.

❗CHECK	"CHECK" indicates any instructions or precautions for using the system.
REFERENCE	"REFERENCE" indicates any hints for operation, detail explanations, or references.
TECHNIQUE	"TECHNIQUE" indicates useful conditions or techniques (know-how) for operation of the system.

⚠WARNING

- Install a fail-safe device when the product is used for the purpose that has a possibility of physical injury or serious extended damage.
- Do not use the product in the atmosphere of flammable gas, to prevent explosion.

⚠CAUTION

- Use the product within specifications.
  Abnormal heat or smoke generation may occur.
- Do not disassemble or remodel the product. Electrical shock or smoke generation may occur.
- Connect the electric wire securely with the terminal screws.
  Imperfect connection may cause abnormal heat or smoke generation.
- Do not touch the terminal during energization of the product, to prevent electrical shock.

For Correct Use

This manual describes API function for controlling the HL-C2 system by USB communication.
For the detailed description of construction and use of the system, refer to "HL-C2 Series USER'S MANUAL" (separate volume).

Correct Handling

For the items listed below, refer to "HL-C2 Series USER'S MANUAL" (separate volume).

・Installation Environment
・Use Environment
・Measures to Noise
・Warming Up Time
・Insulation Resistance and Voltage Resistance
・Power Supply
・Instantaneous Power Failure
・Grounding
・Installation

Cautions on Handling Laser Light

Refer to "HL-C2 Series USER'S MANUAL".

Standards

Refer to "HL-C2 Series USER'S MANUAL".

Use Condition

C-E-20160614

Software License Agreement

Panasonic Industrial Devices SUNX Co., Ltd. (“PIDSX”) grants to you a license to use this Software on condition that you accept this Agreement. You must read this Software License Agreement (this “Agreement”) carefully before using this Software. Only in case that you accept this Agreement, you may start your use of this Software.
Your unsealing the package of this Software, or your downloading, installing or launching this Software or the like shall be deemed as your acceptance of this Agreement.

Article 1 Grant of License

1-1. PIDSX hereby grants to you a non-exclusive license to use this Software only in combination with PIDSX product(s) specified in the manual of this Software (the “Product”) in accordance with the terms of this Agreement.

1-2. For the purpose stipulated in the preceding paragraph, you may, at your own responsibility, modify this Software and/or distribute (including transfer, rent or lease either for fees or for free) this Software to a third party on condition that you procure from such third party a consent to the terms of this Agreement; provided, however, that PIDSX shall not be liable for any defects or malfunctions caused by your modification under any circumstances.

1-3. In no event may you use, modify, or distribute to a third party this Software in connection with products of any third party other than PIDSX.

Article 2 Restrictions

You may not use this Software by methods or for purposes other than those specified in the manual provided by PIDSX.

Article 3 Disclaimer

3-1. PIDSX HEREBY DISCLAIMS ALL OTHER WARRANTIES ON THIS SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MARCHANTABILITY, FITNESS FOR PARTICULAR PURPOSE, AND

NON-INFRINGEMENT OF THIRD PARTY RIGHTS.

3-2. UNDER NO CIRCUMSTANCES SHALL PIDSX BE LIABLE FOR ANY DAMAGES (INCLUDING DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL OR SPECIAL OR WHATSOEVER) ARISING OUT OF THE USE OF THIS SOFTWARE, INABILITY TO USE THIS SOFTWARE, DEFECTS IN THIS SOFTWARE (e.g., BUGS, SECURITY HOLES, AND MALFUNCTION), MODIFICATION OF THIS SOFTWARE, DISTRIBUTION OF THIS SOFTWARE, OR OTHERWISE IN CONNECTION WITH THIS SOFTWARE.

Article 4 Term

4-1. This Agreement shall come into effect upon your unsealing the package of this Software, or your downloading, installing or launching this Software or the like.

4-2. PIDSX may terminate this Agreement immediately, if you breach any of the provisions of this Agreement.

4-3. You shall, at your own costs, return, delete or destroy this Software and any of its copies within four (4) weeks after termination of this Agreement.

Article 5 Export Control

You shall comply with all laws and regulations regarding export control under any competent jurisdiction, including but not limited to the Japanese Foreign Exchange & Foreign Trade Control Law, the export control regulations based on resolutions of the United Nations Security Council, etc. If any license or appropriate approval from a governmental authority is required under the applicable laws, you may not export this Software without such approval to any countries either directly or indirectly. Furthermore, you shall neither use nor sell this Software for military purposes either directly or indirectly.

Article 6 Intellectual Property Rights

All intellectual property rights in this Software, including the copyright, belong to PIDSX and/or the licensors of PIDSX.

Article 7 Upgrade of this Software

7-1. Release of future upgrades or updates of this Software is not guaranteed and left to the sole discretion of PIDSX. Furthermore, PIDSX may charge fees for upgrading or updating of this Software.

7-2. If any upgrades or updates are provided to you either for fees or for free, such upgrades or updates shall be deemed as a part of this Software and shall be governed by this Agreement, unless PIDSX designates otherwise at the time of provision of such upgrades or updates.

Article 8 Limitation on Liability

AGGREGATE LIABILITIES OF PIDSX IN CONNECTION WITH THIS AGREEMENT OR THIS SOFTWARE SHALL IN NO EVENT EXCEED TEN THOUSAND (10,000) YEN.

Article 9 Governing Law and Jurisdiction

9-1. This Agreement shall be governed by the laws of Japan.

9-2. Should any dispute arise from or in connection with this Agreement, Nagoya District Court, Japan shall exclusively have the jurisdiction over such dispute.

1

Prior to Use

This chapter provides the information about the product that users should know prior to use.

1-1 General Description

This manual describes the API (Application Program Interface) used for Ultra High-Speed, High-Accuracy Laser Displacement Sensor "HL-C2 Series"

HL-C2 can be controlled through the external host device (personal computer) through USB interface by using the API function*.
* API function is hereinafter called "API".

USB driver

USB driver should be installed to the external control device (personal computer) in order to control HL-C2 by the device (PC).

If you have already installed the software HL-C2AiM on your PC, also the USB driver has been installed.

Also the USB driver can be downloaded on our website (https://panasonic.net/id/pidsx/global). For the installation method, please refer to ➔ "1-2 Installation of USB Driver"

API file

API (Application Program Interface) is provided to easily control HL-C2 by the external control device.
API is provided in DLL format.

Please feel free to ask our salesman how to obtain a DLL file, or check our website (https://panasonic.net/id/pidsx/global).
Please contact the vendor of development environment for the use of DLL.
Refer to ➔ "Chapter 2 API Function Specifications" in "HL-C2 Series USER'S MANUAL USB Communication Control" for the use of API.

Sample program (Only a Japanese document and commentary.)

Sample program for USB control, which was created using API, is provided.
We offer Sample program, which are for Visual Basic and Visual C++.
Please feel free to ask our salesman how to obtain Sample Program, or check our website (https://panasonic.net/id/pidsx/global).

1-2 Installation of USB Driver

USB driver should be installed to control the HL-C2 by external control device (USB host).

For a PC that has the old USB driver installed already ➔ Follow the procedures in "1-2-1 Uninstallation of Old USB Driver" and then install the new USB driver.
For installation for the first time ➔ Follow the procedures in "1-2-2 Installation of USB Driver" and install the driver.

Supplemental remarks

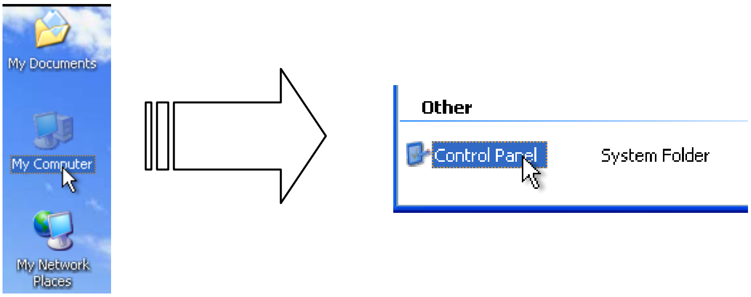
Old USB driver is referring to drivers "FTDI USB Serial Converter Drivers"

* This explanation is based on using a PC with Windows XP as the external control device.
The procedures vary depending on the OS you are using.

1-2-1 Uninstallation of old USB Driver

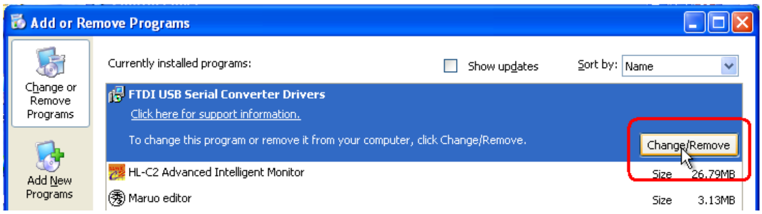
Uninstall following the below procedures.

1 Open Control Panel from My Computer.



2 In Control Panel, open “Add or Remove Programs”

3 Delete “FTDI USB Serial Converter Drivers”.



Currently installed programs: Show updates Sort by: Name

FTDI USB Serial Converter Drivers
Click here for support information.
To change this program or remove it from your computer, click Change/Remove.

HL-C2 Advanced Intelligent Monitor Size 26.79MB

Maruo editor Size 3.13MB

Uninstallation is then complete.
Next, follow the procedures in “1-2-2 Installation of USB Driver”.

1-2-2 Installation of USB Driver

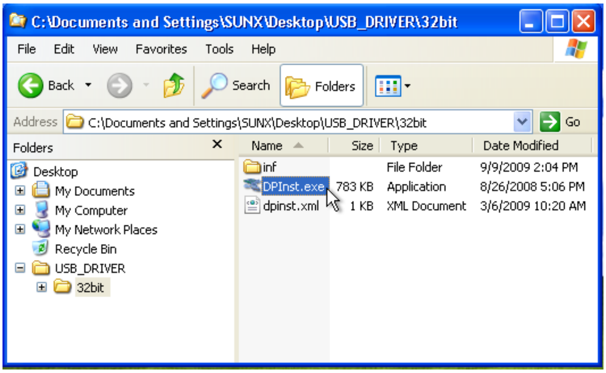
CHECK

For PC that uses the old USB driver, make sure to delete the old driver and then install the new driver. For deleting procedures ➔ Refer to "1-2-1 Uninstallation of Old USB Driver"

In case the control device is not installed yet, execute DPInst.exe following the below procedures, and install the new USB driver.

1 Execute DPInst.exe, which is inside "USB_DRIVER"-> "32bit" folder or "64bit" folder.

This screen example is executing DPInst.exe after having copied USB_DRIVER folder onto the Desktop.



C:\Documents and Settings\SUNX\Desktop\USB_DRIVER\32bit

Name	Size	Type	Date Modified
inf		File Folder	9/9/2009 2:04 PM
DPInst.exe	783 KB	Application	8/26/2008 5:06 PM
dpinst.xml	1 KB	XML Document	3/6/2009 10:20 AM

There is no dialog displayed during installation execution.
After the execution is finished, installation is complete.

MEMO

2 API Function Specifications

This chapter explains specifications of API function that are required to control HL-C2 by external control device (PC) through USB interface.

2-1 Variable Type

These variable type functions are available in the user's program.

BYTE(8bit unsigned data)
typedef unsigned char BYTE;

WORD(16 bit unsigned data)
typedef unsigned short WORD;

DWORD(32 bit unsigned data)
typedef unsigned long DWORD;

PCHAR(8 bit unsigned data pointer)
typedef char * PCHAR;

LPDWORD(32 bit unsigned data pointer)
typedef DWORD *LPDWORD;

HLC2_HANDLE(Handle: handle type which is acquired at device open)
typedef void * HLC2_HANDLE;

HLC2_STATUS(Status: type of Return value for each function)
typedef DWORD HLC2_STATUS;

LPDOUBLE(64 bit floating decimal point data pointer)
typedef double *LPDOUBLE

```
// dwHead selected.
#define   HEADA        0      // Selected HAED-A
#define   HEADB        1      // Selected HEAD-B
// dwOut selected.
#define   OUT1         0      // Selected OUT1.
#define   OUT2         1      // Selected OUT2.
// dwIO selected.
#define   IO_IN        0      // GET
#define   IO_OUT       1      // SET or EXECUTION
```

2-1-1 Data Format Structure

1)HLC2_CONFIG5(For code setting 00000 to 99999)

```
typedef struct
{
        BYTE    Num[5];
} HLC2_CONFIG5;
```

2)HLC2_NUMERIC11(For numerical value setting -999.999999 to +999.99999)

```
typedef struct
{
        BYTE    Sign;           // Sign ("±")
        BYTE    Integer[3];     // 3-digit integer (no zero suppression)
        BYTE    Period;         // Decimal point (".")
        BYTE    Decimal[6];     // 6-digit decimal number
} HLC2_NUMERIC11;
```

3)HLC2_NUMERIC12(For difference value setting -9999.999999 to +9999.999999)

Used for rapid loading of buffering data (RLB command).

```
typedef struct
{
        BYTE    Sign;           // Sign ("±")
        BYTE    Integer[4];     // 4-digit integer (no zero suppression)
        BYTE    Period;         // Decimal point (".")
        BYTE    Decimal[6];     // 6-digit decimal number
} HLC2_NUMERIC12;
```

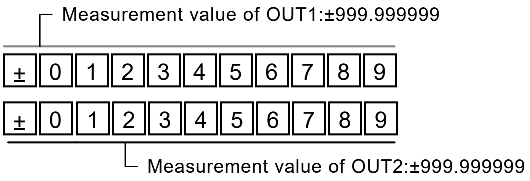
4)HLC2_NUMERIC10(For setting of 10-digit value. Used for the command format 3.)

```
typedef struct
{
        BYTE    Sign;           // Sign ("±")
        BYTE    Integer[10];    // 10-digit integer (no zero suppression)
} HLC2_NUMERIC10;
```

2-1-2　2-Output Measurement Value Readout Structure

1)Data format structure：HLC2_OUTMEASUREVALUE

```
typedef struct
{
        HLC2_NUMERIC11 Numeric[2];      //Measurement value of OUT1/OUT2
} HLC2_OUTMEASUREVALUE;
```


Measurement value of OUT1:±999.999999
± 0 1 2 3 4 5 6 7 8 9
± 0 1 2 3 4 5 6 7 8 9
Measurement value of OUT2:±999.999999

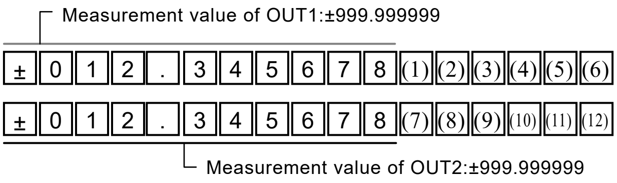
Measurement value: 1-digit sign + 3-digit integer (no zero suppression) + decimal point + 6-digit decimal number

2-1-3 All Output Readout Structure

1) Data format structure: HLC2_OUTALL_DATA

```
typedef struct
{
        HLC2_NUMERIC11 Numeric;   // Measurement value of OUT1/2
        BYTE StrobeOut;           // Strobe output          (0 or 1)
        BYTE HighOut;             // Judgment output HI     (0 or 1)
        BYTE GoOut;               // Judgment output GO     (0 or 1)
        BYTE LowOut;              // Judgment output LO     (0 or 1)
        BYTE ExtOut;              // No used                (0 or 1)
        BYTE AlarmOut;            // Alarm output           (0 or 1)
} HLC2_OUT_DATA;

typedef struct
{
        HLC2_OUT_DATA        OutData[2];
} HLC2_OUTALL_DATA;
```



Measurement value: 1-digit sign + 3-digit integer (no zero suppression) + decimal point + 6-digit decimal number

The following outputs are stored in (1) to (12). (Off: $\boxed{0}$, On: $\boxed{1}$)

No.	Output		
(1)	OUT1	Strobe output	
(2)		Judg. output	HI output
(3)			GO output
(4)			LO output
(5)	Not used		
(6)	Alarm output of Sensor head A		

No.	Output		
(7)	OUT2	Strobe output	
(8)		Judg. output	HI output
(9)			GO output
(10)			LO output
(11)	Not used		
(12)	Alarm output of Sensor head B		

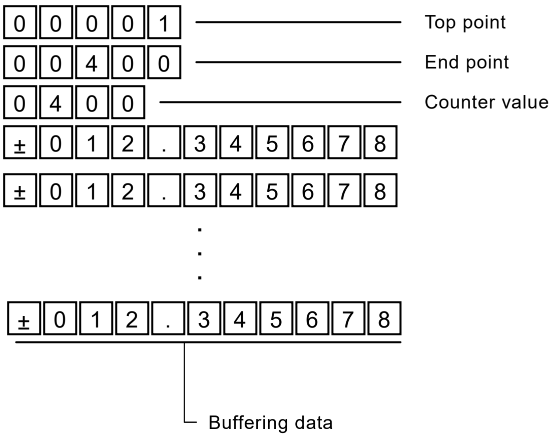
CHECK

(5) and (11) are returned as unfixed data (either $\boxed{0}$ or $\boxed{1}$).

2-1-4　Buffering Data Normal Readout Structure

1) Data format structure: HLC2_BUFFERNORMAL

```
typedef struct
{
    DWORD       TopPoint;                       //Top point (00001 to 99999)
    DWORD       EndPoint;                       //End point (00001 to 99999)
    WORD        dwCount;                        //Counter value (reserved)
    HLC2_NUMERIC11          *pGetData;          //Read data stored point
} HLC2_BUFFERNORMAL;
```



Buffering data from the specified top point to end point is returned.

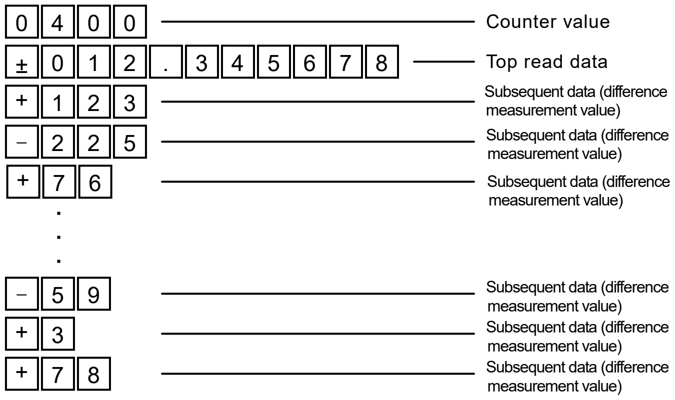
Buffering data: Measurement value: 1-digit sign + 3-digit integer (no zero suppression) + decimal point + 6-digit decimal number

2-1-5 Buffering Data Rapid Readout Structure

1) Data format structure:HLC2_BUFFERRAPID

```
typedef struct
{
        DWORD                   TopPoint;       // Top point (00001～99999)
        DWORD                   EndPoint;       // End point (00001～99999)
        DWORD                   dwCount;        // Counter value ( reserved )
        HLC2_NUMERIC12          *pGetData;      // Read data stored point
} HLC2_BUFFERRAPID
```

Readout data format by buffering data rapid readout command



0 4 0 0
Counter value
± 0 1 2 . 3 4 5 6 7 8
Top read data
+ 1 2 3
Subsequent data (difference measurement value)
− 2 2 5
Subsequent data (difference measurement value)
+ 7 6
Subsequent data (difference measurement value)
− 5 9
Subsequent data (difference measurement value)
+ 3
Subsequent data (difference measurement value)
+ 7 8
Subsequent data (difference measurement value)

<Actual measurement values of the above data example are shown as below>

```
+12.345678, +12.345801, +12.345576, +12.345652,…, +12.345599, +12.34554, +12.345543, +12.345621
              ↑           ↑           ↑                       ↑           ↑           ↑
           (+123)      (-225)       (+76)                   (-59)       (+3)        (+78)
```

The specified top point of data is stored in the below top data format.
Second and later point data are read out as difference measurement value of those previous data (sixth decimal point data).
Top data format: 1-digit sign + 3-digit integer (no zero suppression) + decimal point + 6-digit decimal number

2-1-6 Memory Copy Structure

1)HLC2_MEMCOPY

Structure for setting of copy source memory, copy destination memory, and copy command

```
typedef struct
{
            BYTE    SetMemFrom;     // Source memory #(0～15)
            BYTE    SetMemTo;       // Destination memory #(0～15)
            BYTE    MemCopyCom;     // Command(0:none,1:execution)
} HLC2_MEMCOPY;
```

№	Name	Function	Value	Size(Word)
1	SetMemFrom	Memory No. of copy source	0 to 15	1
2	SetMemTo	Memory No. of copy destination	0 to 15	1
3	MemCopyCom	Copy command	0: No command 1: Execute memory copy	1

2-1-7　Return value

● Status list

Val.	Name	Function
0	HLC2_OK	Normal end
1	HLC2_INVALID_HANDLE	Handle is incorrect.
2	HLC2_DEVICE_NOT_FOUND	Device is not found.
3	HLC2_DEVICE_NOT_OPENED	Device is not opened.
4	HLC2_CONTROLLER_ERROR	Controller error
5	HLC2_INVALID_PARAMETER	Parameter is incorrect.
6	HLC2_ RECEIVE_ERROR	Receive error
7	HLC2_FILE_OPEN_ERROR	File cannot be opened.
8	HLC2_FILE_CREATE_ERROR	File cannot be created.
9	HLC2_SHORTAGE_MEMORY	Available memory is low.

■ Error message

The following shows each error message and those causes/measures.

1 Handle is incorrect.

[Cause]　The handle value which is not recognized as a device is specified.

[Measure]　Specify a handle value which is recognized as a device.

2 Device is not found.

[Cause]　Controller power is off.
Controller is not connected to the appropriate COM/USB port.

[Measure]Turn on the power of controller.
Connect the controller correctly.

3 Device is not opened.

[Cause]　Device is not opened.

[Measure]　Check whether the device is opened or not.

4 Controller error

[Cause]　Abnormal response is sent from the controller.

[Measure]　Check whether the controller operates correctly or not.

5 Parameter is incorrect.

[Cause]　A value out of the parameter range is specified.

[Measure]　Reenter correct parameter value within the parameter range.

6 Receive error

[Cause] Data is destroyed by electrical noise.

[Measure] Remove electrical noise.

7 File cannot be opened.

[Cause] File you specified is being used in other program.
File you specified does not exist.

[Measure] Check whether the file is being used in other program.
Confirm the file name.

8 File cannot be created.

[Cause] The file is write-inhibited.
Insufficient free space

[Measure] Enable write into the file.
Reserve sufficient free space.

9 Available memory is low.

[Cause] Available memory is low.

[Measure] Secure space capacity of the memory.

```
// HLC2_STATUS
#define HLC2_OK                        0
#define HLC2_INVALID_HANDLE            1
#define HLC2_DEVICE_NOT_FOUND          2
#define HLC2_DEVICE_NOT_OPENED         3
#define HLC2_CONTROLLER_ERROR          4
#define HLC2_INVALID_PARAMETER         5
#define HLC2_RECEIVE_ERROR             6
#define HLC2_FILE_OPEN_ERROR           7
#define HLC2_FILE_CREATE_ERROR         8
#define HLC2_SHORTAGE_MEMORY           9
```

2-2 Function

■API function list

Class	No.	API name	Function
USB device control	1)	OpenByIndex	Opens HL-C2 with Device No.
	2)	GetCount	Acquires connection number of HL-C2.
	3)	Init	Initializes specified device.
	4)	Close	Closes specified device.
	5)	GetSerialNumber	Acquires string of serial number.
	6)	Open	Opens HL-C2 with serial number
Head setting (Head A, B) command	1)	HeadSetupMode	IO for installation mode
	2)	HeadFloodLightAjust	IO for emission adjustment
	3)	ExecFloodLight	Executes emitted light intensity search/loads its status.
	4)	HeadAlarmDelayTimes	IO for alarm delay times
	5)	HeadHMeasureMode	IO for Measurement Mode
	6)	HeadMeasureWorkBasis	IO for measurement surface reference
	7)	HeadCalibMeasureValueA	IO for calibration measurement value A
	8)	HeadCalibCorrectValueA	IO for calibration correction value a
	9)	HeadCalibMeasureValueB	IO for calibration measurement value B
	10)	HeadCalibCorrectValueB	IO for calibration correction value b
	11)	ExecCaliburation	Executes calibration/loads its status
	12)	HeadLaserOff	IO for laser control
	13)	Get LightWaveData	Loads received light intensity read data.
	14)	HeadPeakSearchMinLevel	IO for Peak Recognition Sensitivity
	15)	HeadEmissionAdjustment AreaA	IO for emission adjustment area a
	16)	HeadEmissionAdjustment AreaB	IO for emission adjustment area b
	17)	HeadMedianFilter	IO for median filter
	18)	HeadMeasuringRangePointA	IO for measuring range point a
	19)	HeadMeasuringRangePointB	IO for measuring range point b
Out setting (OUT1, 2) command	1)	OutPattern	IO for output selection
	2)	OutMeasureWork	IO for output selection: transparent object
	3)	OutReflectionCalc	IO for output selection: transparent object refractive index calculation
	4)	OutReflectionRate	IO for output selection: transparent object refractive index
	5)	OutZeroSet	IO for zero set
	6)	OutTiming	IO for timing
	7)	OutReset	IO for reset
	8)	OutHold	IO for hold
	9)	OutMeasureMode	IO for analysis mode

Class	No.	API name	Function
Out setting (OUT1, 2) command	10)	OutFilterSelect	IO for Filter Operation
	11)	OutAverageTimes	IO for average moving times
	12)	OutCutOffCycle	IO for Cutoff frequency
	13)	OutSpan	IO for Operation Coefficient
	14)	OutOffsetInput	IO for offset
	15)	OutDecisionMax	IO for judgment output: upper limit value
	16)	OutDecisionMin	IO for judgment output: lower limit value
	17)	OutDecisionHisMax	IO for judgment output: upper limit hysteresis
	18)	OutDecisionHisMin	IO for judgment output: lower limit hysteresis
	19)	OutScalingMeasureValueA	IO for analog scaling measurement value A
	20)	OutScalingVoltageValueA	IO for analog scaling voltage a
	21)	OutScalingMeasureValueB	IO for analog scaling measurement value B
	22)	OutScalingVoltageValueB	IO for analog scaling voltage b
	23)	ExecAnalogScaling	Executes analog scaling.
	24)	OutAnalogOutOnAlarm	IO for analog output at alarm
	25)	OutFixedValueInput	IO for analog output at alarm: fixed value
	26)	OutAnalogOutOnUnfixed	IO for analog output at data unfixed
	27)	OutDegitalOutOnAlarm	IO for digital output at alarm
	28)	OutAlarmDelayChange	IO for output alarm delay
	26)	OutDispDigit	IO for display digit of measurement value
	30)	GetMeasureValue	Loads measurement value.
	31)	GetAlarmState	Loads alarm output status.
	32)	GetStrobeState	Loads strobe status.
	33)	GetHighState	Loads judgment output: HI status.
	34)	GetGoState	Loads judgment output: GO status.
	35)	GetLowState	Loads judgment output: LO status.
Common setting command	1)	CmnSamplingCycle	IO for sampling cycle
	2)	CmnPreventInterference	IO for Interference Prevention
	3)	CmnTerminalInputCtrl	IO for terminal input control
	4)	CmnTerminalInputChattering	IO for chattering prevention of terminal input
	5)	Get2OutMeasureValue	Loads 2 output measurement value readout.
	6)	GetOutAll	Loads all output read.
	7)	CmnOffDaley	IO for Judgment Output Off Delay
System setting command	1)	ExecOutConfigCopy	Copies setting between OUT1 and OUT2.
	2)	SysMemChangePriority	IO for priority setting of memory change
	3)	ExecMemChange	Specifies memory No., then executes memory change or loads its status.
	4)	ExecMemCopy	Specifies copy source and destination, then executes memory copy.

Class	No.	API name	Function
System setting command	5)	ExecMemInitialize	Initializes selected memory/all memory.
	6)	ExecMemSave	Saves all memory.
	7)	SysRs232cBaudrate	IO for RS-232C baud rate
	8)	SysRs232cDataLen	IO for RS-232C data length
	9)	SysRs232cParity	IO for RS-232C parity check
	10)	SysRs232cOutMode	IO for RS-232C Output Mode
	11)	SysRs232cOutType	IO for RS-232C Output Type
	12)	SysMeasureUpdateCycle	IO for display update cycle of measurement value
	13)	SysConsoleStartNo	IO for console start-up screen
	14)	SysConsolePanelLock	IO for console panel lock
Buffering command	1)	BufferingMode	IO for buffering mode
	2)	BufferingType	IO for buffering type
	3)	BufferingRate	IO for buffering rate
	4)	BufferStoreNum	IO for accumulated amount
	5)	BufferSampleTriggerStoreNum	IO for Sample Trigger Accumulation Amount
	6)	BufferTriggerPoint	IO for Trigger Point
	7)	BufferTriggerDelay	IO for Trigger Delay
	8)	BufferEventCondition	IO for Trigger Conditions
	9)	ExecBuffering	Executes buffering operation/loads its status.
	10)	BufferSelfStop	IO for Self-stop
	11)	GetBufferState	Status readout
	12)	GetBufferFinalDataPoint	Loads final data point.
	13)	GetBufferTriggerCount	Trigger counter readout
	14)	GetBufferDataNormal	Data readout (normal)
	15)	GetBufferDataRapid	Data readout (rapid)

* Function is named each API name after HLC2_.

2-2-1 USB Device Control

1)OpenByIndex

Opens HL-C2 by a device No.

```
HLC2_STATUS HLC2_OpenByIndex(DWORD dwIndex,
                             HLC2_HANDLE *hlc2Handle)
```

Argument

DWORD dwIndex	Index of device connection number (Select among 0 to device connection number -1)
HLC2_HANDLE *hlc2Handle	Handle storing variable pointer

Return value

Returns HLC2_OK if normal end.

Summary

Loads a device handle by using an index of acquired device connection number.

2)GetCount

Acquires connection number of HL-C2.

```
HLC2_STATUS HLC2_GetCount(LPDWORD lpCount)
```

Argument

LPDWORD lpCount	Variable pointer for storing connection number of HL-C2 device

Return value

Returns HLC2_OK if normal end.

3)Init

Initializes specified device.

```
HLC2_STATUS HLC2_Init(HLC2_HANDLE hlc2Handle);
```

Argument

HLC2_HANDLE hlc2Handle	Handle

Return value

Returns HLC2_OK if normal end.

4)Close

Closes specified device.

HLC2_STATUS HLC2_Close(HLC2_HANDLE hlc2Handle)

Argument

HLC2_HANDLE hlc2Handle	Handle

Return value

Returns HLC2_OK if normal end.

5)GetSerialNumber

Acquires string of serial No.

HLC2_STATUS　HLC2_GetSerialNumber (DWORD dwIndex, PCHAR szSerialNumberBuffer)

Argument

DWORD dwIndex	Device No.
PCHAR szSerialNumberBuffer	Area pointer for storing serial No. string.

Return value

Returns HLC2_OK if normal end.

Explanation of function

Acquires the connected serial No. information.

6)Open

Opens HL-C2 by a serial No.

HLC2_STATUS　HLC2_Open　(HLC2_HANDLE　*hlc2Handle,　PCHAR szSerialNumberBuffer)

Argument

HLC2_HANDLE *hlc2Handle	Handle storing variable pointer
PCHAR szSerialNumberBuffer	Area pointer for storing serial No. string.

Return value

Returns HLC2_OK if normal end.

2-2-2 Head Setting (Head A/B) Command

1)HeadSetupMode

IO for head setting [installation mode].

HLC2_STATUS HLC2_HeadSetupMode(HLC2_HANDLE hlc2Handle, DWORD dwHead, DWORD dwIO, LPDWORD lpReflect, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwHead	Specifies head. (HEADA or HEADB)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpReflect	Variable pointer for storing installation mode* of load/setting target *Installation mode: *lpReflect = 0:Diffuse 1:Specular
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

2)HeadFloodLightAdjust

IO for head setting [Emission adjustment]

HLC2_STATUS HLC2_HeadFloodLightAjust (HLC2_HANDLE hlc2Handle, DWORD dwHead, DWORD dwIO, LPDWORD lpInfo, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwHead	Specifies head. (HEADA or HEADB)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpInfo	Variable pointer for storing emission adjustment* of load/setting target. *Emission adj.: *lpInfo = 0:AUTO 1:0.04% Fixed 2:0.05% Fixed 3:0.06% Fixed 4:0.08% Fixed 5:0.11% Fixed 6:0.14% Fixed 7:0.18% Fixed 8:0.24% Fixed 9:0.31% Fixed 10:0.40% Fixed 11:0.53% Fixed 12:0.68% Fixed 13:0.89% Fixed 14:1.16% Fixed 15:1.50% Fixed 16:1.95% Fixed 17:2.54% Fixed 18:3.30% Fixed 19:4.29% Fixed 20:5.58% Fixed 21:7.25% Fixed 22:9.43% Fixed 23:12.3% Fixed 24:15.9% Fixed 25:20.7% Fixed 26:26.9% Fixed 27:35.0% Fixed 28:45.5% Fixed 29:59.2% Fixed 30:76.9% Fixed 31:100% Fixed
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

3)ExecFloodLight

Executes head setting [Emitted light intensity search] or loads its status.

HLC2_STATUS HLC2_Head_ExecFloodLight (HLC2_HANDLE hlc2Handle, DWORD dwHead, DWORD dwIO, LPDWORD lpStatus, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwHead	Specifies head. (HEADA or HEADB)
DWORD dwIO	Selects IO. (0: status load, 1: execute)
LPDWORD lpStatus	Variable pointer for storing execution command/status. *Emitted light intensity search command: *lpStatus = 0: No command 1: Execute 2: Searching
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

4)HeadAlarmDelayTimes

IO for head setting [Alarm delay times]

HLC2_STATUS HLC2_HeadAlarmDelayTimes (HLC2_HANDLE hlc2Handle, DWORD dwHead, DWORD dwIO, LPDWORD lpCount, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwHead	Specifies head. (HEADA or HEADB)
DWORD dwIO	Selects IO. (0: load (Input), 1:Setting (Output))
LPDWORD lpCount	Variable pointer for storing alarm delay times* of load/setting target *Alarm delay times: *lpCount = 0: OFF 1 to 65535:Hold previous value
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

5)HeadHMeasureMode

IO for head setting [Measurement Mode]

HLC2_STATUS HLC2_HeadHMeasureMode(HLC2_HANDLE hlc2Handle, DWORD dwHead, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwHead	Specifies head. (HEADA or HEADB)
DWORD dwIO	Selects IO. (0: load (Input), 1:Setting (Output))
LPDWORD lpSelect	Variable pointer for storing Measurement Mode * of load/setting target * Measurement Mode: *lpSelect = 0:Diffuse Reflection[Standard] 1:Specular Reflection[Standard] 2:Metal 1 3:Metal 2 4:Penetration 5:Glass 6:Glass Pattern
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

6)HeadMeasureWorkBasis

IO for head setting [Measurement surface reference]

HLC2_STATUS HLC2_HeadMeasureWorkBasis (HLC2_HANDLE hlc2Handle, DWORD dwHead, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwHead	Specifies head. (HEADA or HEADB)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing measurement surface reference* of load/setting target *Measurement surface reference: *lpSelect = 0:Near 1:Far
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

7)HeadCalibMeasureValueA

IO for head setting [Calibration measurement value A]

HLC2_STATUS HLC2_ HeadCalibMeasureValueA (HLC2_HANDLE hlc2Handle, DWORD dwHead, DWORD dwIO, LPDOUBLE lpdValue, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwHead	Specifies head. (HEADA or HEADB)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDOUBLE lpdValue	Variable pointer for storing calibration measurement value A* of load/setting target *Calibration measurement value A: *lpdValue=-950.000000 to +950.000000[mm]
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

8)HeadCalibCorrectValueA

IO for head setting [Calibration correction value a]

HLC2_STATUS HLC2_HeadCalibCorrectValueA (HLC2_HANDLE hlc2Handle, DWORD dwHead, DWORD dwIO, LPDOUBLE lpdValue, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwHead	Specifies head. (HEADA or HEADB)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDOUBLE lpdValue	Variable pointer for storing calibration correction value a* of load/setting target. * Calibration correction value a: *lpdValue= -950.000000 to +950.000000[mm]
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

9)HeadCalibMeasureValueB

IO for head setting [Calibration measurement value B]

HLC2_STATUS HLC2_HeadCalibMeasureValueB (HLC2_HANDLE hlc2Handle, DWORD dwHead, DWORD dwIO, LPDOUBLE lpdValue, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwHead	Specifies head. (HEADA or HEADB)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDOUBLE lpdValue	Variable pointer for storing calibration measurement value A* of load/setting target *Calibration measurement value B: *lpdValue= -950.000000 to +950.000000[mm]
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

10)HeadCalibCorrectValueB

IO for head setting [Calibration correction value b]

HLC2_STATUS HLC2_HeadCalibCorrectValueB (HLC2_HANDLE hlc2Handle, DWORD dwHead, DWORD dwIO, LPDOUBLE lpdValue, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwHead	Specifies head. (HEADA or HEADB)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDOUBLE lpdValue	Variable pointer for storing calibration correction value b* of load/setting target. * Calibration correction value b: *lpdValue= -950.000000 to +950.000000[mm]
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

11)ExecCaliburation

Executes head setting [Calibration]

HLC2_STATUS HLC2_ExecCaliburation (HLC2_HANDLE hlc2Handle, DWORD dwHead, DWORD dwIO, LPDWORD lpStatus, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwHead	Specifies head. (HEADA or HEADB)
DWORD dwIO	Selects IO. (1: execute)
LPDWORD lpStatus	Variable pointer for storing calibration execution* * Calibration command *lpStatus= 0: No command 1: Execute 2: Cancel
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

12)HeadLaserOff

IO for head setting [Laser control]

HLC2_STATUS HLC2_HeadLaserOff (HLC2_HANDLE hlc2Handle, DWORD dwHead, DWORD dwIO, LPDWORD lpOnOff, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwHead	Specifies head. (HEADA or HEADB)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpOnOff	Variable pointer for storing laser on/off* of load/setting target * Laser control: *lpOnOff =0:Laser on 1:Laser off
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

13)GetLightWaveData

Loads head setting [Received light intensity readout (data)].

HLC2_STATUS HLC2_GetLightWaveData(HLC2_HANDLE hlc2Handle, DWORD dwHEAD, LPDWORD lpValue , BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of HL-C2
DWORD dwHead	Specifies head. (HEADA or HEADB)
LPDWORD lpValue	Variable pointer for storing received light intensity data of Head A and B which is loaded by received light intensity readout command
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

14) HeadPeakSearchMinLevel

IO for head setting [Peak Recognition Sensitivity]

HLC2_STATUS HLC2_HeadPeakSearchMinLevel(HLC2_HANDLE hlc2Handle, DWORD dwHead, DWORD dwIO,LPDWORD lpValue, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of HL-C2
DWORD dwHead	Specifies head. (HEADA or HEADB)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpValue	Variable pointer for storing Peak Recognition Sensitivity* *lpValue = 100 to 400
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

15) HeadEmissionAdjustmentAreaA

IO for head setting [Emission Adjustment Area a]

HLC2_STATUS HLC2_HeadEmissionAdjustmentAreaA(HLC2_HANDLE hlc2Handle, DWORD dwHead, DWORD dwIO, LPDWORD lpValue, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of HL-C2
DWORD dwHead	Specifies head. (HEADA or HEADB)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpValue	Variable pointer for HeadEmissionAdjustmentAreaA* *lpValue = 1 to 512
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

16) HeadEmissionAdjustmentAreaB

IO for head setting [Emission Adjustment Area b]

HLC2_STATUS HLC2_HeadEmissionAdjustmentAreaB(HLC2_HANDLE hlc2Handle, DWORD dwHead, DWORD dwIO, LPDWORD lpValue, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of HL-C2
DWORD dwHead	Specifies head. (HEADA or HEADB)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpValue	Variable pointer for HeadEmissionAdjustmentAreaB* *lpValue = 1 to 512
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

17）HeadMedianFilter

IO for head setting [Median Filter]

HLC2_STATUS HLC2_HeadMedianFilter (HLC2_HANDLE hlc2Handle, DWORD dwHead, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of HL-C2
DWORD dwHead	Specifies head. (HEADA or HEADB)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for HeadMedianFilter* *Median Filter * lpSelect = 0: OFF 1: 7 points 2: 15 points 3: 31 points
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

18）HeadMeasuringRangePointA

IO for head setting [Measuring Range Point a]

HLC2_STATUS HLC2_HeadMeasuringRangePointA (HLC2_HANDLE hlc2Handle, DWORD dwHead, DWORD dwIO, LPDWORD lpValue, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of HL-C2
DWORD dwHead	Specifies head. (HEADA or HEADB)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpValue	Variable pointer for HeadMeasuringRangePointA* *lpValue = 3 to 510
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

19）HeadMeasuringRangePointB

IO for head setting [Measuring Range Point b]

HLC2_STATUS HLC2_HeadMeasuringRangePointB (HLC2_HANDLE hlc2Handle, DWORD dwHead, DWORD dwIO, LPDWORD lpValue, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of HL-C2
DWORD dwHead	Specifies head. (HEADA or HEADB)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpValue	Variable pointer for HeadMeasuringRangePointB* *lpValue = 3 to 510
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

2-2-3 OUT Setting (OUT1/2) Command

1)OutPattern

IO for OUT setting [Output selection]

HLC2_STATUS HLC2_OutPattern(HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO,LPDWORD lpPattern, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpPattern	Variable pointer for storing output selection* of load/setting target *Output selection *lpPattern = 0:[A] 1:[B] 2:[-A] 3:[-B] 4:[A+B] 5:[-(A+B)] 6:[A-B] 7:[B-A] 8:[A Transparent object] 9:[B Transparent object] 10:[-A Transparent object] 11:[-B Transparent object] 12:A1+B1[Transparent object] 13:-(A1+B1)[Transparent object] 14: A1-B1[Transparent object] 15: B1-A1[Transparent object]
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

2)OutMeasureWork

IO for OUT setting [Transparent object]

HLC2_STATUS HLC2_OutMeasureWork (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO,LPDWORD lpWork, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpWork	Variable pointer for storing transparent object* of load/setting target * Transparent object selection *lpWork = 0:[1st surface] 1:[2nd surface] 2:[3rd surface] 3:[4th surface] 4:[Upper limit surface] 5:[1st surface-2nd surface] 6:[1st surface-3rd surface] 7:[1st surface-4th surface] 8:[1st surface-Upper limit surface] 9:[2nd surface-3rd surface] 10:[2nd surface-4th surface] 11:[2nd surface-Upper limit surface] 12:[3rd surface-4th surface] 13:[3rd surface-Upper limit surface] 14:[4th surface-Upper limit surface]
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

3)OutReflectionCalc

IO setting for OUT setting [Refractive index calculation]

HLC2_STATUS HLC2_OutReflectionCalc (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing refractive index calculation* of load/setting target * Refractive calculation: *lpSelect = 0:OFF, 1:ON
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

4)OutReflectionRate

IO for OUT setting [Refractive index]

HLC2_STATUS HLC2_OutReflectionRate (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDOUBLE lpdRate, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDOUBLE lpdRate	Variable pointer for storing refractive index* of load/setting target. * Refractive index: *lpdRate = +000.500000 to +002.000000
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

5)OutZeroSet

IO for OUT setting [Zero set]

HLC2_STATUS HLC2_OutZeroSet (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing zero set* of load/setting target. * Zero set: *lpSelect = 0:OFF, 1:ON
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

6)OutTiming

IO for OUT setting [Timing]

HLC2_STATUS HLC2_OutTiming (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing timing* of load/setting target. * Timing: *lpSelect = 0:OFF, 1:ON
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

7)OutReset

IO for OUT setting [Reset]

HLC2_STATUS HLC2_OutReset (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing reset* of load/setting target * Reset: *lpSelect = 0:OFF, 1:ON
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add))

Return value

Returns HLC2_OK if normal end.

8)OutHold

IO for OUT setting [Hold]

HLC2_STATUS HLC2_OutHold (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing hold* of load/setting target * Hold: *lpSelect = 0:OFF, 1:ON
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

9)OutMeasureMode

IO for OUT setting [Analysis mode]

HLC2_STATUS HLC2_OutMeasureMode (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO,LPDWORD lpMode, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpMode	*Variable pointer for storing analysis mode* of load/setting target * Analysis mode *lpMode =　0: Normal 1: Peak 2: Bottom 3: P-P
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

10) OutFilterSelect

IO for OUT setting [Filter Operation]

HLC2_STATUS HLC2_OutFilterSelect(HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO,LPDWORD lpSelect, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	*Variable pointer for storing Filter Operation * of load/setting target * Filter Operation * lpSelect = 0: Moving average 1: low-pass filter 2: high-pass filter
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

11)OutAverageTimes

IO for OUT setting [Average times]

HLC2_STATUS HLC2_OutAverageTimes (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing average times* of load/setting target * Average times: *lpSelect = 0:1 time 1:2 times 2:4 times 3:8 times 4:16 times 5:32 times 6:64 times 7:128 times 8:256 times 9:512 times 10:1024 times 11:2048 times 12:4096 times 13:8192 times 14:16384 times 15:32768 times 16:65536 times
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

12) OutCutOffCycle

IO for OUT setting [Cutoff frequency]

HLC2_STATUS HLC2_OutCutOffCycle(HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg);;

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing Cutoff frequency* of load/setting target *Cutoff frequency: *lpSelect = 0: 1[Hz] 1: 2[Hz] 2: 4[Hz] 3: 10[Hz] 4: 20[Hz] 5: 40[Hz] 6: 100[Hz] 7: 200[Hz] 8: 400[Hz] 9: 1000[Hz] 10 :2000[Hz]
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

13) OutSpan

IO for OUT setting [Operation Coefficient]

HLC2_STATUS HLC2_OutSpan(HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO,LPDOUBLE lpdValue, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDOUBLE lpdValue	Variable pointer for storing Operation Coefficient* of load/setting target * Operation Coefficient : * lpdValue = 000.100000 ～ 009.999999
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

14) OutOffsetInput

IO for OUT setting [Offset]

HLC2_STATUS HLC2_OutOffsetInput(HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDOUBLE lpdOffset, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDOUBLE lpdOffset	Variable pointer for storing Offset* of load/setting target * Offset : * lpdValue = -950.000000 ～ +950.000000
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

15)OutDecisionMax

IO for OUT setting [Judgment output: Upper limit value]

HLC2_STATUS HLC2_OutDecisionMax (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDOUBLE lpdValue, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDOUBLE lpdValue	Variable pointer for storing judgment output: upper limit value* of load/setting target * Judgment output: Upper limit value *lpdValue = -950.000000 to +950.000000[㎜])
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

16)OutDecisionMin

IO for OUT setting [Judgment output: Lower limit value]

HLC2_STATUS HLC2_OutDecisionMin (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDOUBLE lpdValue, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDOUBLE lpdValue	Variable pointer for storing judgment output: lower limit value* of load/setting target * Judgment output: Lower limit value *lpdValue = -950.000000 to +950.000000[㎜])
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

17)OutDecisionHisMax

IO for OUT setting [Judgment output: Upper limit hysteresis]

HLC2_STATUS HLC2_OutDecisionHisMax (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDOUBLE lpdHis, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDOUBLE lpdHis	Variable pointer for storing judgment output: upper limit hysteresis* of load/setting target * Judgment output: Upper limit hysteresis *lpdHis = +000.000000 to +950.000000[㎜])
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

18)OutDecisionHisMin

IO for OUT setting [Judgment output: Lower limit hysteresis]

HLC2_STATUS HLC2_OutDecisionHisMin (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDOUBLE lpdHis, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDOUBLE lpdHis	Variable pointer for storing judgment output: lower limit hysteresis* of load/setting target * Judgment output: Lower limit hysteresis *lpdHis = +000.000000 to +950.000000[㎜])
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

19)OutScalingMeasureValueA

IO for OUT setting [Analog scaling measurement value A]

HLC2_STATUS HLC2_OutScalingMeasureValueA (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDOUBLE lpdValue, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDOUBLE lpdValue	Variable pointer for storing analog scaling measurement value A* of load/setting target * Analog scaling measurement value A *lpdValue = -950.000000 to +950.000000[㎜])
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

20)OutScalingMeasureValueB

IO for OUT setting [Analog scaling measurement value B]

HLC2_STATUS HLC2_OutScalingMeasureValueB (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDOUBLE lpdValue, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDOUBLE lpdValue	Variable pointer for storing analog scaling measurement value B* of load/setting target * Analog scaling measurement value B *lpdValue = -950.000000 to +950.000000[㎜])
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

21)OutScalingVoltageValueA

IO for OUT setting [Analog scaling voltage a]

HLC2_STATUS HLC2_OutScalingVoltageValueA (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDOUBLE lpd Voltage, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDOUBLE lpd Voltage	Variable pointer for storing analog scaling voltage a* of load/setting target * Analog scaling voltage a: *lpd Voltage = ±010.000000[V] The last three digits are zero-fixed.
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

22)OutScalingVoltageValueB

IO for OUT setting [Analog scaling voltage b]

HLC2_STATUS HLC2_OutScalingVoltageValueB (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDOUBLE lpd Voltage, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDOUBLE lpd Voltage	Variable pointer for storing analog scaling voltage b* of load/setting target * Analog scaling voltage b: *lpd Voltage = ±010.000000[V] The last three digits are zero-fixed.
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

23)ExecAnalogScaling

Executes OUT setting [Analog scaling].

HLC2_STATUS HLC2_ExecAnalogScaling(HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, DWORD dwStatus, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (1: execute)
DWORD dwStatus	Execution command * Analog scaling command: dwStatus= 0: No command 1: Execute 2: Cancel
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

24)OutAnalogOutOnAlarm

IO for OUT setting [Analog output at alarm]

HLC2_STATUS HLC2_OutAnalogOutOnAlarm (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing analog output at alarm* of load/setting target * Analog output at alarm *lpSelect = 0:Hold previous value 1:Fixed value
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

25)OutFixedValueInput

IO for OUT setting [Analog output at alarm: Fixed value]

HLC2_STATUS HLC2_OutFixedValueInput (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDOUBLE lpdValue, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDOUBLE lpdValue	Variable pointer for storing analog output at alarm: Fixed value* of load/setting target * Analog output at alarm: Fixed value *lpdValue = ±010.000000[V] The last three digits are zero-fixed.
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

26)OutAnalogOutOnUnfixed

IO for OUT setting [Analog output at data unfixed]

HLC2_STATUS HLC2_OutAnalogOutOnUnfixed (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDOUBLE lpdValue, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDOUBLE lpdValue	Variable pointer for storing analog output at data unfixed* of load/setting target * Analog output at data unfixed: *lpdValue = ±010.000000[V] The last three digits are zero-fixed.
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

27)OutDegitalOutOnAlarm

IO for OUT setting [Digital output at alarm]

HLC2_STATUS HLC2_OutDegitalOutOnAlarm (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing digital output at alarm* of load/setting target * Digital output at alarm: *lpSelect = 0:Hold previous value 1:Fixed value
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

28)OutAlarmDelayChange

IO for OUT setting [Alarm output delay]

HLC2_STATUS HLC2_OutAlarmDelayChange (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing alarm output delay change selection* of load/setting target * Alarm output delay change selection: *lpSelect = 0:OFF 1:ON
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

29)OutIDispDigit

IO for OUT setting [Digit number of measurement value]

HLC2_STATUS HLC2_OutDispDigit (HLC2_HANDLE hlc2Handle, DWORD dwOut, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing digit number selection of measurement value* of load/setting target * Digit number selection of measurement value: *lpSelect = 0: 6 decimal places 1: 5 decimal places 2: 4 decimal places 3: 3 decimal places 4: 2 decimal places 5: 1 decimal place
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

30)GetMeasureValue

Loads OUT setting [Measurement value].

HLC2_STATUS HLC2_GetMeasureValue(HLC2_HANDLE hlc2Handle, DWORD dwOut, LPDOUBLE lpdValue, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of HL-C2
DWORD dwOut	Specifies output. (OUT1 or OUT2)
LPDOUBLE lpdValue	Variable pointer for storing loaded measurement value* * Readout measurement value: *lpdvalue = -999.999999 to +999.999999[mm])
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

31)GetAlarmState

Loads OUT setting [Alarm output status].

HLC2_STATUS HLC2_Get AlarmState (HLC2_HANDLE hlc2Handle,
DWORD dwOut, LPDWORD lpStatus, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of HL-C2
DWORD dwOut	Specifies output. (OUT1 or OUT2)
LPDWORD lpStatus	Variable pointer for storing loaded alarm output status* * Readout alarm output status: *lpStatus = 0: No alarm output status (OFF) 1: Measurement alarm output status (ON) 6: Export controlled head connected status (ON) 7: Head connection check error (ON)
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

32)GetStrobeState

Loads OUT setting [Strobe] status.

HLC2_STATUS HLC2 HLC2_GetStrobeState(HLC2_HANDLE hlc2Handle,
DWORD dwOut, LPDWORD lpStatus, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of HL-C2
DWORD dwOut	Specifies output. (OUT1 or OUT2)
LPDWORD lpStatus	Variable pointer for storing loaded strobe status* * Readout strobe status: *lpStatus = 0: No Strobe output status(OFF) 1: Strobe output status (ON)
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

33)GetHighState

Loads OUT setting [Judgment output: HI] status.status

HLC2_STATUS HLC2_GetHighState(HLC2_HANDLE hlc2Handle,
DWORD dwOut, LPDWORD lpStatus , BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of HL-C2
DWORD dwOut	Specifies output. (OUT1 or OUT2)
LPDWORD lpStatus	Variable pointer for storing loaded HI status* * Readout HI status: *lpStatus = 0: Judgment output HI no output status (OFF) 1: Judgment output HI output status (ON)
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

34)GetGoState

Loads OUT setting [Judgment output:GO] status. status

HLC2_STATUS HLC2 HLC2_GetGoState(HLC2_HANDLE hlc2Handle,
DWORD dwOut, LPDWORD lpStatus, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of HL-C2
DWORD dwOut	Specifies output. (OUT1 or OUT2)
LPDWORD lpStatus	Variable pointer for storing loaded GO status* * Readout GO status: *lpStatus = 0: Judgment output GO no output status (OFF) 1: Judgment output GO output status (ON)
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

35)GetLowState

Loads OUT setting [Judgment output:LO] status. status

```
HLC2_STATUS HLC2 HLC2_GetLowState(HLC2_HANDLE hlc2Handle,
DWORD dwOut, LPDWORD lpStatus, BYTE bccFlg)
```

Argument

HLC2_HANDLE hlc2Handle	Handle of HL-C2
DWORD dwOut	Specifies output. (OUT1 or OUT2)
LPDWORD lpStatus	Variable pointer for storing loaded LO status* * Readout LO status: *lpStatus = 0: Judgment output LO no output status (OFF) 1: Judgment output LO output status (ON)
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

2-2-4 Common Setting Command

1)CmnSamplingCycle

IO for common setting [Sampling cycle]

HLC2_STATUS HLC2_CmnSamplingCycle(HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpCycle, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpCycle	Variable pointer for storing sampling cycle selection* of load/setting target * Sampling cycle selection: *lpCycle = 0:10[μs], 1:20[μs], 2:40[μs], 3:100[μs], 4:200[μs], 5:400[μs], 6:1[ms], 7:2[ms]
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

CHECK

If this setting is changed between 10μs ↔ 20μs or above, the USB of the controller will be reset. To continuously use the USB device after such changes, reconnect following the USB device Close and Open procedures.

REFERENCE

USB Close Function : HLC2_Close()

USB Open Function : HLC2_OpenByIndex(), HLC2_Open()

2)CmnPreventInterference

IO for common setting [Interference Prevention]

HLC2_STATUS HLC2_CmnPreventInterference(HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing Interference Prevention selection* of load/setting target *Interference Prevention selection: * lpSelect = 0:OFF 1: ON
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

3)CmnTerminalInputCtrl

IO for common setting [Terminal input control]

HLC2_STATUS HLC2_CmnTerminalInputCtrl (HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing terminal input control selection* of load/setting target * Terminal input control selection: *lpSelect = 0: Independent 1: All
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

4)CmnTerminalIInputChattering

IO for common setting [Chattering prevention for terminal input]

HLC2_STATUS HLC2_CmnTerminalInputChattering (HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing selection of chattering prevention for terminal input* of load/setting target * Selection of chattering prevention for terminal input: *lpSelect = 0:OFF 1:ON1 2:ON2
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

5)Get2OutMeasureValue

Loads common setting [2 output measurement value readout].

HLC2_STATUS HLC2_Get2OutMeasureValue(HLC2_HANDLE hlc2Handle, HLC2_OUTMEASUREVALUE *pOutMeasureValue, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of HL-C2
HLC2_OUTMEASUREVALUE *pOutMeasureValue RMA	Pointer of the arrangement to store 2 output measurement values (-999.999999 to +999.999999[mm]) loaded by the command
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

6)GetOutAll

Loads common setting [All output read].

HLC2_STATUS HLC2_GetOutAll(HLC2_HANDLE hlc2Handle,
HLC2_OUTALL_DATA *pOutAllData , BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of HL-C2
HLC2_OUTALL_DATA *pOutAllData	Pointer of stored destination for all output read data structure loaded by RMB command
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

7) CmnOffDaley

IO for common setting [Judgment Output Off Delay].

HLC2_STATUS HLC2_CmnOffDaley(HLC2_HANDLE hlc2Handle,
DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of HL-C2
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing selection of Judgment Output Off Delay * of load/setting target *Judgment Output Off Delay selection: *lpSelect = 0:OFF 1:2ms 2:10ms 3:100ms 4:Hold
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

2-2-5 System Setting Command

1)ExecOutConfigCopy

System setting [OUT setting copy]

Copies settings between OUT1 and OUT2.

HLC2_STATUS HLC2_ExecOutConfigCopy(HLC2_HANDLE hlc2Handle, LPDWORD lpSelect, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
LPDWORD lpSelect	Variable pointer for storing copy command* * Copy command: *lpSelect = 0: No command 1: Copy OUT1 to OUT2. 2: Copy OUT2 to OUT1.
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

2)SysMemChangePriority

IO for system setting [Priority setting of memory change]

HLC2_STATUS HLC2_SysMemChangePriority(HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpPriority, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpPriority	Variable pointer for storing priority selection of memory change* of load/setting target * Priority selection of memory change: *lpPriority = 0: Command 1: Terminal
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

3)ExecMemChange

System setting [memory change]

Specifies a memory No., and then executes [Memory change] / loads its status.

HLC2_STATUS HLC2_ExecMemChange(HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpMemNo, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: status load, 1: execute)
LPDWORD lpMemNo	Variable pointer for storing memory change No.* * Memory change No.: *lpMemNo = 0 to 15
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

CHECK

If a memory switchover which changes the sampling cycle setting to more than 10μs ↔ 20μs is executed, the USB of the controller will be reset. To continuously use the USB device after such a memory switchover, reconnect following the USB device Close and Open procedures.

REFERENCE

USB Close Function : HLC2_Close()

USB Open Function : HLC2_OpenByIndex(), HLC2_Open()

4)ExecMemCopy

System setting [Memory copy]

Specifies copy source and copy destination, and then execute [Memory copy].

HLC2_STATUS HLC2_ExecMemCopy(HLC2_HANDLE hlc2Handle, HLC2_MEMCOPY *pMemCopy, BYTE bccFlg);

Argument

HLC2_HANDLE hlc2Handle	Handle of device
HLC2_MEMCOPY *pMemCopy	Pointer of stored destination for copy source memory, copy destination memory, and copy command setting structure
pMemCopy->SetMemFrom	Copy source memory No.(0 to 15) (in)
pMemCopy->SetMemTo	Copy destination memory No. (0 to 15) (in)
pMemCopy-> MemCopyCom	Memory copy command 0: No command, 1: Execute
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

CHECK

If a memory copy which changes the sampling cycle setting to more than 10μs ↔ 20μs is executed on the currently specified memory, the USB of the controller will be reset. To continuously use the USB device after such a memory copy, reconnect following the USB device Close and Open procedures.

REFERENCE

USB Close Function : HLC2_Close()

USB Open Function : HLC2_OpenByIndex(), HLC2_Open()

5)ExecMemInitialize

Executes system setting [Initialize selected memory] and [Initialize all memory].

HLC2_STATUS HLC2_ExecMemInitialize(HLC2_HANDLE hlc2Handle, LPDWORD lpSelect, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
LPDWORD lpSelect	Variable pointer for storing selected memory initialization command* * Selected memory initialization command: *lpSelect = 0:No command 1: Selected memory 2: All memory
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

CHECK

If the setting initialization is executed while the sampling cycle of the controller is set to 10μs, the USB of the controller will be reset. To continuously use the USB device after the setting initialization, reconnect following the USB device Close and Open procedures.

REFERENCE

USB Close Function : HLC2_Close()

USB Open Function : HLC2_OpenByIndex(), HLC2_Open()

6)ExecMemSave

Executes system setting [Save all memory].

HLC2_STATUS HLC2_ExecMemSave(HLC2_HANDLE hlc2Handle, LPDWORD lpSelect, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
LPDWORD lpSelect	Variable pointer for storing all memory save command* * All memory save command: *lpSelect = 0: No command 1: All memory
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

7)SysRs232cBaudrate

IO for system setting [RS-232C baud rate]

HLC2_STATUS HLC2_SysRs232cBaudrate(HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing RS-232C baud rate* of load/setting target * RS-232C baud rate selection: *lpSelect = 0:9600 1:19200 2:38400 3:115200[bps]
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

8)SysRs232cDataLen

IO for system setting [RS-232C data length]

HLC2_STATUS HLC2_SysRs232cDataLen(HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpLength, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpLength	Variable pointer for storing RS-232C data length* of load/setting target * RS-232C data length selection: *lpLength= 0:7 bit, 1:8 bit
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

9)SysRs232cParity

IO for system setting [RS-232C parity check]

HLC2_STATUS HLC2_SysRs232cParity(HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing RS-232C parity check* of load/setting target * RS-232C parity check selection: *lpSelect = 0: even, 1: odd, 2: none
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

10)SysRs232cOutMode

IO for system setting [RS-232C Output Mode]

HLC2_STATUS HLC2_SysRs232cOutMode (HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing RS-232C Output Mode* of load/setting target *RS-232C Output Mode selection: * lpSelect = 0: Handshake 1: Timing 2: Continuous
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

11)SysRs232cOutType

IO for system setting [RS-232C Output Type]

HLC2_STATUS HLC2_SysRs232cOutType (HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing RS-232C Output Type* of load/setting target * RS-232C Output Type selection: * lpSelect = 0: OUT1&2 1: OUT1 2: OUT2
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

12)SysMeasureUpdateCycle

IO for system setting [Display update cycle of measurement value]

HLC2_STATUS HLC2_SysMeasureUpdateCycle (HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing selection of display update cycle of measurement value* of load/setting target * Selection of display update cycle of measurement value: *lpSelect = 0: Fast 1: Standard 2: Slow 3: Very slow
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

13)SysConsoleStartNo

IO for system setting [Console start-up screen]

HLC2_STATUS HLC2_SysConsoleStartNo (HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing console start-up screen* of load/setting target * Console start-up screen: *lpSelect = 0 to 15: Startup screen No.
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

For the details on start-up screen No. and other screens, refer to the below manual.
➔ "HL-C2 Series USER'S MANUAL" - [4-3-4 System Setting] - [Console Setting]

Return value

Returns HLC2_OK if normal end.

14)SysConsolePanelLock

IO for system setting [Console panel lock]

HLC2_STATUS HLC2_SysConsolePanelLock(HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing console panel lock* of load/setting target * Console panel lock selection: *lpSelect = 0:OFF, 1:ON
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

2-2-6 Buffering Setting Command

1)BufferingMode

IO for [Buffering mode]

HLC2_STATUS HLC2_BufferingMode(HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing buffering mode selection* of load/setting target * Buffering mode selection: *lpSelect = 0: Continuous 1: Trigger 2: Timing 3: Sample Trigger
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

2)BufferingType

IO for [Buffering type]

HLC2_STATUS HLC2_BufferingType(HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing buffering type selection* of load/setting target * Buffering type selection: *lpSelect = 0:OUT1&2 1:OUT1 2:OUT2
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

3)BufferingRate

IO for buffering setting [Buffering rate]

HLC2_STATUS HLC2_BufferingRate(HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpRate, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpRate	Variable pointer for storing buffering rate selection* of load/setting target
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

* Buffering rate selection:

*lpRate =	0:1	8:1/256
	1:1/2	9:1/512
	2:1/4	10:1/1024
	3:1/8	11:1/2048
	4:1/16	12:1/4096
	5:1/32	13:1/8192
	6:1/64	14:1/16384
	7:1/128	15:1/32768

Return value

Returns HLC2_OK if normal end.

4)BufferStoreNum

IO for buffering setting [Accumulated amount]

HLC2_STATUS HLC2_BufferStoreNum(HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpNum, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpNum	Variable pointer for storing accumulated amount* of load/setting target * Accumulated amount: *lpNum = 1 to 65000 (Max. accumulation amount) (AIM_BUFF_DATA_MAX): 20000
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

5)BufferSampleTriggerStoreNum

IO for buffering setting [Sample Trigger Accumulation Amount]

HLC2_STATUS HLC2_BufferSampleTriggerStoreNum (HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpNum, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpNum	Variable pointer for storing Sample Trigger Accumulation Amount * of load/setting target * Sample Trigger Accumulation Amount: *lpNum = 1 to accumulation amount (Be sure to set the sample trigger accumulation amount so that (accumulation amount) ÷ (sample trigger accumulation amount) is an integer value.)
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

6)BufferTriggerPoint

IO for buffering setting [Trigger Point]

HLC2_STATUS HLC2_BufferTriggerPoint(HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpNum, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpNum	Variable pointer for storing Trigger Point* of load/setting target *Trigger Point *lpNum = 1 to accumulation amount
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

7)BufferTriggerDelay

IO for buffering setting [Trigger Delay]

HLC2_STATUS HLC2_BufferTriggerDelay(HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpValue, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpValue	Variable pointer for storing Trigger Delay* of load/setting target * Trigger Delay *lpNum = 0 to 100000000
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

8)BufferEventCondition

IO for buffering setting [Trigger Conditions]

HLC2_STATUS HLC2_BufferEventCondition(HLC2_HANDLE hlc2Handle, DWORD dwOUT, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing Trigger Conditions * of load/setting target * Trigger Conditions * lpSelect = 0: At timing input ON 1: At HI 2: At LO 3: At HIorLO 4: When HI turns to GO 5: When LO turns to GO 6: When HIorLO turns to GO 7: At an alarm occurred 8: At an alarm released
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

9)ExecBuffering

Executes buffering setting [Buffering operation] or status load.

HLC2_STATUS HLC2_ExecBuffering(HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpStatus, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpStatus	Variable pointer for storing accumulation stop status of load/setting target * Buffering operation: *lpStatus = 0: Stop, 1: Start
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

10)BufferSelfStop

IO for buffering setting [Self-stop]

HLC2_STATUS HLC2_BufferSelfStop(HLC2_HANDLE hlc2Handle, DWORD dwIO, LPDWORD lpSelect, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwIO	Selects IO. (0: load (input), 1:setting (output))
LPDWORD lpSelect	Variable pointer for storing Self-stop* of load/setting target * Self-stop * lpSelect = 0: OFF, 1: ON
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

11)GetBufferState

Executes buffering setting [Status readout].

HLC2_STATUS HLC2_GetBufferState(HLC2_HANDLE hlc2Handle, DWORD dwOut, LPDWORD lpStatus , BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
LPDWORD lpStatus	Variable pointer for storing readout status* * Readout status: *lpStatus= 0: Non-buffering 1: Wait for trigger 2: Accumulating 3: Accumulation completed
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

12)GetBufferFinalDataPoint

Executes buffering setting [Final data point load].

HLC2_STATUS HLC2_GetBufferFinalDataPoint(HLC2_HANDLE hlc2Handle, DWORD dwOut, LPDWORD lpNum, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
LPDWORD lpNum	Variable pointer for storing read final data point
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

13)GetBufferTriggerCount

Executes buffering setting [Trigger counter readout].

HLC2_STATUS HLC2_GetBufferTriggerCount(HLC2_HANDLE hlc2Handle, DWORD dwOut, LPDWORD lpCount, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
LPDWORD lpCount	Variable pointer for storing Trigger counter readout
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

14)GetBufferDataNormal

Executes buffering setting [Data readout (normal)].

HLC2_STATUS HLC2_GetBufferDataNormal(HLC2_HANDLE hlc2Handle, DWORD dwOut, HLC2_BUFFERNORMAL *pBufferNormal , BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
HLC2_BUFFERNORMAL *pBufferNormal	Pointer of stored destination for buffering data structure loaded by data readout (normal) command pBufferNormal->TopPoint Top point (00001 to 99999) (in) pBufferNormal->EndPoint End point (00001 to 99999) (in) pBufferNormal->pGetData Pointer of stored area of normal read data (1-digit sign + 3-digit integer (no zero suppression) + decimal point + 6-digit decimal number) (out)
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

Remarks

The size of the memory area where the data loaded by data readout (normal) command are stored is provided in the following calculating formula. Secure or release the storage memory in API invoker side.

(End point -Top point +1)×11*

* Readout data size of each point (1-digit sign + 3-digit integer (no zero suppression) + decimal point + 6-digit decimal number)

CHECK

· For reading buffering code by calculating BCC cord, perform one reading every 400 points.
· In case of omitting BCC code calculation, up to 64000 points can be read.

15)GetBufferDataRapid

Executes buffering setting [Data readout (rapid)].

HLC2_STATUS HLC2_GetBufferDataRapid(HLC2_HANDLE hlc2Handle, DWORD dwOut, HLC2_BUFFERRAPID *pBufferRapid , LPDWORD lpSelect, BYTE bccFlg)

Argument

HLC2_HANDLE hlc2Handle	Handle of device
DWORD dwOut	Specifies output. (OUT1 or OUT2)
HLC2_BUFFERRAPID *pBufferRapid	Pointer of stored destination for buffering data structure loaded by data readout (rapid) command pBufferRapid->TopPoint Top point (00001 to 99999) (in) pBufferRapid->EndPoint End point (00001 to 99999) (in) pBufferRapid->dwCount Data size of data readout (rapid) (out) pBufferRapid->pGetData Head data of data readout (rapid) Pointer of stored area of normal read data (1-digit sign + 3-digit integer (no zero suppression) + decimal point + 6-digit decimal number) (out)
LPDWORD lpSelect	Variable pointer for storing output contents selection of difference area(*) *output contents selection of difference area: *lpSelect= 0: Returns measurement value calculated from difference. 1: Returns data formed into 4-digit integer and 6 decimal places.)
BYTE bccFlg	Selects BCC addition (0:BCC omit, 1:BCC add)

Return value

Returns HLC2_OK if normal end.

Remarks

The size of the memory area where the data loaded by data readout (rapid) command are stored is provided in the following calculating formula. Secure or release the storage memory in API invoker side.

1) lpSelect=0 is specified

(End point -Top point +1)×12[*]

* Readout data size of each point (1-digit sign + 4-digit integer (no zero suppression) + decimal point + 6-digit decimal number)

2) lpSelect=1 is specified

12[*1]+(End point -Top point)×12[*2]

*1 Readout data size of top point (1-digit sign + 4-digit integer (no zero suppression) + decimal point + 6-digit decimal number)

*2 Difference data size to previous data (1-digit sign + 4-digit integer (no zero suppression) + decimal point + 6-digit decimal number)

CHECK

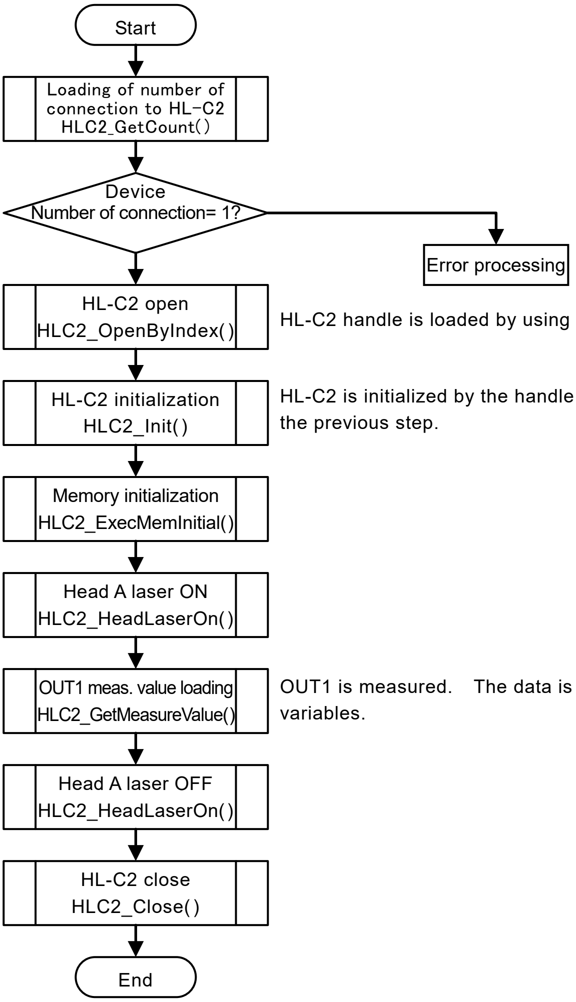
· For reading buffering code by calculating BCC cord, perform one reading every 400 points.
· In case of omitting BCC code calculation, up to 64000 points can be read.
· The readout counter value “dwCount” is not used.

MEMO

3 Control Example

This chapter explains the example of control to load the measurement value from the HL-C2 by using API function.

3-1　HL-C2 Measurement Value Loading (example)



HL-C2 handle is loaded by using device number.

HL-C2 is initialized by the handle that was loaded in the previous step.

OUT1 is measured.　The data is stored into double variables.

■ Example of C code to load HL-C2 measurement value

```
#include “HLC2_DLL.h”

#define HEADA                   0
#define OUT1                    0
#define IO_OUT                  1
#define MEMINI_1                1
#define BCC_OFF                 0
#define LASERON                 0
#define LASEROFF                1

DWORD        dwLaser;
double       dMeasureValue;

BOOL HLC2_Get_Measure_Data(void)
      {
      DWORD              ret;
      DWORD              dwCount;
      HLC2_HANDLE    hlc2Handle;

      // Loading of number of connection to HL-C2
      ret = HLC2_GetCount(&dwCount);
      if (dwCount != 1) {
            // Number of connection to HL-C2≠1error
            return FALSE;
            }

      // HL-C2 open
      ret = HLC2_OpenByIndex(dwCount - 1, &hlc2Handle)
      if ( ret != HLC2_OK) {
            // HL-C2 open failed
            return FALSE;
            }

      // HL-C2 device initialization
      ret = HLC2_Init(hlc2Handle)
      if ( ret != HLC2_OK) {
            // HL-C2 device initialization failed
            return FALSE;
            }
// Memory initialization
ret = HLC2_ExecMemInitialize(hlc2Handle, MEMINI_1, BCC_OFF)
if ( ret != HLC2_OK) {
      // Memory initialization failed
      return FALSE;
      }

// Head A laser ON
dwLaser = LASERON;
ret = HLC2_HeadLaserOff(hlc2Handle, HEADA, IO_OUT, &dwLaser, BCC_OFF)
if ( ret != HLC2_OK) {
      // Head A laser control ON failed
      return FALSE;
      }

// OUT1 measurement value loading */
ret= HLC2_GetMeasureValue(hlc2Handle,OUT1, &MeasureValue, BCC_OFF);
if (ret != HLC2_OK) {
      // OUT1 measurement value loading failed */
      return FALSE;
      }

// Head A laser OFF
dwLaser = LASEROFF;
ret = HLC2_ HeadLaserOff (hlc2Handle, HEADA, IO_OUT, &dwLaser, BCC_OFF)
if ( ret != HLC2_OK) {
      // Head A laser control OFF failed
      return FALSE;
      }

// HLC2 close
HLC2_Close(hlc2Handle);
if (ret !=HLC2_OK) {
      // HLC2 close failed
      return FALSE;
      }

return TRUE;
]
```

Appendix

1 Index

Apx

Revision history

Released date	Revision No.
October 2007	First release
May 2008	Second release
July 2008	Third release
September 2009	Fourth release
June 2010	Fifth release
February 2011	Sixth release
October 2012	Seventh release
June 2013	Eighth release
January 2019	Ninth release

Please contact ....

Panasonic Industrial Devices SUNX Co., Ltd.

■ Overseas Sales Division (Head Office): 2431-1 Ushiyama-cho, Kasugai-shi, Aichi, 486-0901, Japan
■ Telephone: +81-568-33-7861 ■ Facsimile: +81-568-33-8591
panasonic.net/id/pidsx/global

For our sale network, please visit our website.

January, 2019 PRINTED IN JAPAN
WUME-HLC2USB-12