CL-SDK

Reference Manual

[Version 1.30R2]

●Note this manual abbreviates product names as follows

Name Given in This Manual	Official Name
VC++	Microsoft Visual C++
Windows 10	Microsoft® Windows® 10 Operating System
Windows 11	Microsoft® Windows® 11 Operating System

●Trademark Notices

●Notes of this Manual

Table of Contents

Introduction

The CL-SDK is the software development kit to create a PC application for the Illuminance Spectrophotometer (CL-500A).

This manual explains how to use the CL-SDK. The application developers are assumed to use Microsoft Visual C++ and the examples how to program are provided in Microsoft Visual C++.

1. System Requirements

The CL-SDK requires the following environment.

- OS : Windows 10(x86), Windows 10(x64), Windows 11
- Software development language : All the development environment where ANSI C interface is available
- The computer must be equipped with a USB2.0-compliant USB port

  [Note] The CL-500A ensures a performance by a USB2.0-compliant USB port only. It does not ensure a performance by a USB1.1 or USB3.0 compliant USB port.

  If use a USB hub, the USB hub is recommended a self-power type which is guaranteed 500mA output for each port.

The instrument which can control by the CL-SDK is as below.

Instrument : CL-500A

[Note] The CL-SDK cannot control the CL-200 and the CL-200A.

2. How to Install (or Uninstall) the CL-SDK

2.1 Installing the CL-SDK

The CL-SDK does not need to install. All you need is to place the CL-SDK on any locale on the PC.

The CL-SDK folder includes the following folders.

Folders	Items in each folder
include file	Definition files for use with both 64-bit and 32-bit applications.
Manual	Reference manuals.
SampleCode	Sample code with CL-SDK.
SDK(x64)	CL-SDK library files. To use for 64bit applications.
SDK(x86)	CL-SDK library files. To use for 32bit applications.

The below files are required for the development of an application.

Dynamic Link Library	
libclapi.dll	Interface Library
libclcolor.dll	Color calculating Library
libclcalib.dll	Calibration calculating Library
libclhid.dll	Device communicating Library
Import Library(*)	
libclapi.lib	Interface Library
Source file	
CLAPI.h	The definition file of CL-SDK interface
CLCondition.h	The definition file of the information in using the CL-SDK
CLColorCondition.h	The definition file of the information on calculating color
TypeDefine.h	The type definition file
ErrorDefine.h	The error definition file
Version.h	The definition file of the version information

(*)NOTES: These import library files are created by Microsoft Visual C++. Therefore they are available in Microsoft Visual C++ only, but other development environments like Borland C are not.

2.2 Uninstalling the CL-SDK

Delete the CL-SDK from your PC in uninstalling the CL-SDK.

2.3 Installing the USB Driver

The installation of the device driver will be required when connecting CL-500A via USB for first time. Then the installation will start automatically.

[Note]

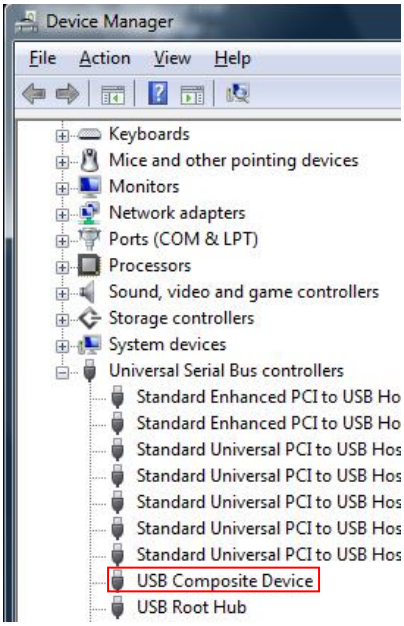
Don't turn power OFF on the instrument or pull out the USB cable from the PC, until completing the install on driver.

If fails to install the driver, please perform the install with refer to the install guide for the

CL-S10w.

2.4 Confirm the installation of the USB driver

After installing the USB driver successfully, CL-500A can be operated as a USB device. Open the Device Manager to confirm the installation. Then new “USB Composite Device” could be found if the installation was successful.



3. Creating of the VC++ application using CL-SDK

The examples of creating VC++ application using the CL-SDK are described below.

3.1 The process for creating the application

The process for creating an application explains according to the following outline.

Step1: Creating the application project

Step2: Setting the references to the CL-SDK

Step3: Creating the Device handle object

Step4: Coding of the application

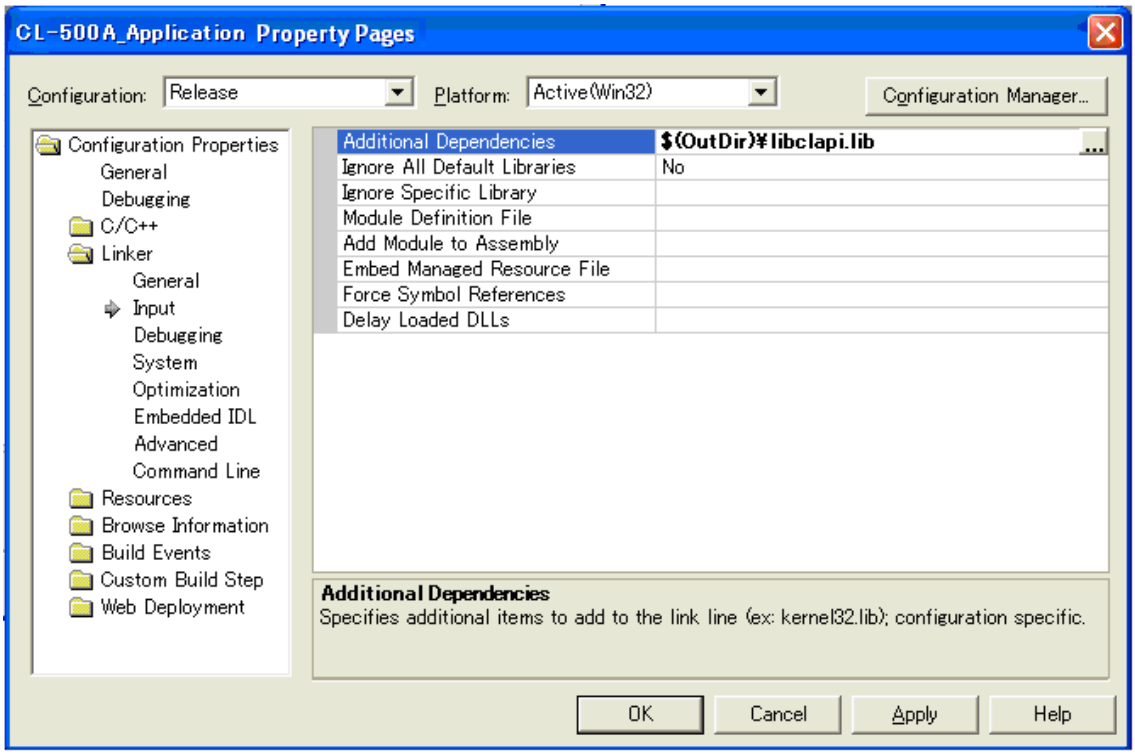
Step5: Compiling and debugging

Step1: Creating the application project

Create project of application according to wizard of Visual Studio.

Step2: Setting the references to the CL-SDK

Reference setting of CL-SDK is performed, and makes sure that CL-SDK will link with application. Click menu [Project]-[Property], the [property pages] dialog is shown as below. Then click [Linker]-[Input] in the dialog, please input the import library files (.lib) to [Additional Dependencies].



Step3: Creating device handle object

The application is required to get device handle to control the instrument which connected on the PC.

To get a device handle, use CLOpenDevice().

And, to control the instrument, the remote mode is required to be ON. After getting the device handle, set a remote mode be ON. Then the instrument can control.

Release the device handle by using CLCloseDevice() when end the control of the instrument.

To finish the control of the instrument, release the device handle by using CLCloseDevice().

```
DEVICE_HANDLE hDevice;                      // Define the device handle of object
CLOpenDevice(&hDevice);                     // Get the device handle
CLSetRemoteMode(hDevice, CL_RMODE_ON)       // Set the remote mode ON
```

[Note]

The application can control the multiple instruments. About how to control the multiple instruments by the application, refer to "§3.3 The control of multiple instrument".

Step4: Coding of the application

Add the source code for controlling the instrument by the application. Refer to "§4 CL-SDK API Reference" about the API to control the instrument.

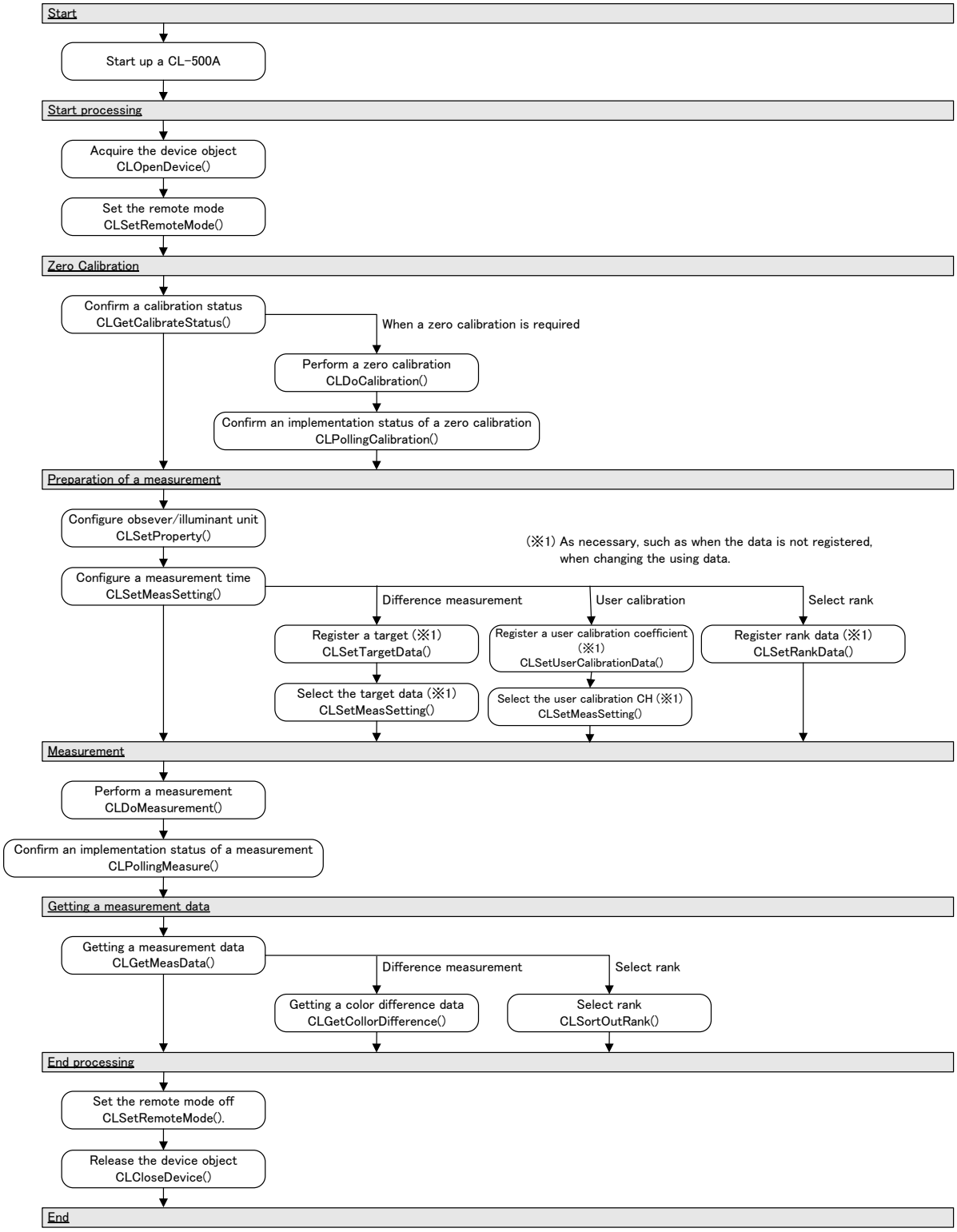
In addition, the CL-SDK package attaches the sample code. Refer in the application development.

The sample codes which provides are as follows. The sample code is created by VC++2013.

Sample code list	
Measurement.vcproj	The operation for irradiance measurement is described
ManualMeasurement.vcproj	The operation for irradiance manual measurement is described
ColorDifference.vcproj	The operation for difference measurement is described
MultipleControl.vcproj	The operation for control of multiple instrument is described
MultipleManualControl.vcproj	The operation for manual measurement control of multiple instrument is described
Rank.vcproj	The operation for selection of rank is described
UserCalibration.vcproj	The operation for user calibration is described
DeviceStoreData.vcproj	The operation for getting the stored data in the instrument is described
DeviceSetting.vcproj	The operation for configuration of measurement condition and system is described

3.2 The basic operation flow

The basic operation flow to control instrument by using the CL-SDK is shown below.



3.2.1 Calculating a measurement data

The measurement data is calculated by the CL-SDK when controlling the instrument using the CL-SDK. Configure the measurement condition (Observer and Illuminance units) of the CL-SDK by CLSetProperty().

The target data in a different measurement, the user calibration coefficient in a user calibration, and the rank data in selecting rank are used the data which is registered in the instrument.
If necessary, please register the data to the instrument and select the data to use, before a measurement.

Difference Measurement

Register the target data to the instrument by CLSetTargetData().
The number which can register as the target data to the instrument is 20 data.
Specify the data No. to be used in a difference measurement by CLSetMeasSetting().
The target data of specified data No. is used in calculating a color difference.

User Calibration

Register the user calibration coefficient to the instrument by CLSetUserCalibrationData().
The number which can register as the user calibration coefficient to the instrument is 10 data.
Specify the user calibration CH to be used in a user calibration by CLSetMeasSetting().
The user calibration coefficient of specified user calibration CH is used in calculating.
If specify the “UC00” as calibration CH, a measurement is performed in accordance with Konica Minolta calibration standards.

Select Rank

Register the rank data to the instrument by CLSetRankData().
The number which can register as the rank data to the instrument is 20 data.

3.3 The control of multiple instrument

The CL-SDK can control the multiple instruments which are connected on the PC.
Please contact about the number of controllable instrument.

The CL-SDK gets the device handle of the instrument to be controlled by CLOpenDevice().The device handle which gets is different by the instrument.
By entering the device handle of the instrument to be controlled into the parameter of the object handle(hDevice) of an each functions, the CL-SDK can control the target instrument.

[Note]
When getting the object handle by CLOpenDevice() in condition that the multiple instruments are connected on the PC, the order for the obtained object handle is random.
Therefore if the object handle is same
Therefore, to make the same of the instrument to be controlled by the obtained object handle, control the instrument with confirming the serial number

3.3.1 Performing a measurement

The CL-SDK provides CLDoMeasurement() and CLDoMeasurementAll() as a measurement function. Use these functions for usage of a measurement. Please use these functions properly by the usage of a measurement.

CLDoMeasurement()
This function performs a measurement of any specified one instrument.
This function cannot a measurement with same timing in case of multiple instruments.

CLDoMeasurementAll()
This function performs a measurement of all instruments which is connected on the PC.
But this function cannot select the instrument to perform a measurement.
This function can perform a measurement with same timing nearly for all instruments.
[Note]
The measurement timing is not same completely.
In addition, the larger the number of instruments which are controlled, the larger the time lag of between the instrument which starts a measurement at first and the instrument which starts a measurement at last

```
/* The example for a measurement with multiple instruments (3 units ) */

// The definition of the object handle
DEVICE_HANDLE hDevice1 = NULL;       // The definition of the object handle for instrument-1
DEVICE_HANDLE hDevice2 = NULL;       // The definition of the object handle for instrument-2
DEVICE_HANDLE hDevice3 = NULL;       // The definition of the object handle for instrument-3

// Get the object handle for each instrument
ER_CODE ret = CLOpenDevice(&hDevice1);
if (ret != SUCCESS) return;

ret = CLOpenDevice(&hDevice2);
if (ret != SUCCESS) return;

ret = CLOpenDevice(&hDevice3);
if (ret != SUCCESS) return;

// Set the remote mode ON for each instrument
ret = CLSetRemoteMode(hDevice1, CL_RMODE_ON);
if (ret != SUCCESS){
    CLCloseDevice(hDevice1);
    return;
}

ret = CLSetRemoteMode(hDevice2, CL_RMODE_ON);
if (ret != SUCCESS){
    CLCloseDevice(hDevice2);
    return false;
}

ret = CLSetRemoteMode(hDevice3, CL_RMODE_ON);
if (ret != SUCCESS){
    CLCloseDevice(hDevice3);
    return;
}
```

```
// Perform a measurement
int meastime;                          // Measurement time
CL_MEASSTATUS status = CL_MEAS_FREE; // Measurement status

// Perform a measurement by instrument-1
ret = CLDoMeasurement(hDevice1, &meastime);
if (ret > WARNING) return;

// Wait until a measurement is complete
do {
    // Confirm the measurement status after waiting for a moment
    Sleep(100);

    ret = CLPollingMeasure(hDevice1, &status);
    if (ret != SUCCESS) return;

} while (status != CL_MEAS_FINISH);

// Perform a measurement by instrument-2
ret = CLDoMeasurement(hDevice2, &meastime);
if (ret > WARNING) return;

// Wait until a measurement is complete
status = CL_MEAS_FREE;
do {
    // Confirm the measurement status after waiting for a moment
    Sleep(100);

    ret = CLPollingMeasure(hDevice2, &status);
    if (ret != SUCCESS) return;

} while (status != CL_MEAS_FINISH);

// Perform a measurement by instrument-3
ret = CLDoMeasurement(hDevice3, &meastime);
```

```
if (ret > WARNING) return;

// Wait until a measurement is complete
status = CL_MEAS_FREE;
do {
    // Confirm the measurement status after waiting for a moment
    Sleep(100);

    ret = CLPollingMeasure(hDevice3, &status);
    if (ret != SUCCESS) return;

} while (status != CL_MEAS_FINISH);

// Get a measurement data
CL_MEASDATA data;

// Get the illuminance spectral data of instrument-1
ret = CLGetMeasData(hDevice1, CL_COLORSPACE_SPC, &data);
if (ret > WARNING) return;
for (int i = 0; i < IRRADIANCE_LEN; i++){
    printf("SpctralData[%d]¥t%.010f¥n", i, data.Spc.Data[i]);
}

// Get the Ev/x/y data of instrument-2
ret = CLGetMeasData(hDevice2, CL_COLORSPACE_EVXY, &data);
if (ret > WARNING) return;
printf("Evxy.Ev¥t%.010f¥nEvxy.x¥t%.010f¥nEvxy.y¥t%.010f¥n", data.Evxy.Ev, data.Evxy.x, data.Evxy.y);

// Get the color rendering index data of instrument-3
ret = CLGetMeasData(hDevice3, CL_COLORSPACE_RENDERING, &data);
if (ret > WARNING) return;
for (int i = 0; i < CL_RENDERING_LEN; i++){
    printf("Rendering[%d]¥t%.010f¥n", i, data.Rendering.Data[i]);
}
```

3.3.2 The Zero Calibration

A zero calibration takes about 27 sec.

If perform a zero calibration for multiple instrument, first execute CLDoCalibration() for all instruments. After that, please confirm the calibration status for each instrument by CLPollingCalibration().

```
/* The example for a zero calibration with multiple instruments (2 units ) */

// The definition of the object handle
DEVICE_HANDLE hDevice1 = NULL;       // The definition of the object handle for instrument-1
DEVICE_HANDLE hDevice2 = NULL;       // The definition of the object handle for instrument-2

// Get the object handle for each instrument
ER_CODE ret = CLOpenDevice(&hDevice1);
if (ret != SUCCESS) return;

ret = CLOpenDevice(&hDevice2);
if (ret != SUCCESS) return;

// Set the remote mode ON for each instrument
ret = CLSetRemoteMode(hDevice1, CL_RMODE_ON);
if (ret != SUCCESS){
          CLCloseDevice(hDevice1);
          return;
}

ret = CLSetRemoteMode(hDevice2, CL_RMODE_ON);
if (ret != SUCCESS){
    CLCloseDevice(hDevice2);
    return;
}

// Perform zero calibration for each instrument
ret = CLDoCalibration(hDevice1);
if(ret != SUCCESS) return;
```

```
ret = CLDoCalibration(hDevice2);
if(ret != SUCCESS) return;

// Confirm the implementation status of zero calibration
CL_CALIBMEASSTATUS calStatus1 = CL_CALIBMEAS_FREE;      // Implementation status for instrument-1
CL_CALIBMEASSTATUS calStatus2 = CL_CALIBMEAS_FREE;      // Implementation status for instrument-2
bool bFinish1 = false;                                  // Completion flag for instrument-1
bool bFinish2 = false;                                  // Completion flag for instrument-2

// Confirm whether all instruments are completed a zero calibration
while(!bFinish1 || !bFinish2){
    // Confirm the measurement status after waiting for a moment
    Sleep(1000);

    ret = CLPollingCalibration(hDevice1, &calStatus1);
    if(ret != SUCCESS) return;
    if (calStatus1 == CL_CALIBMEAS_FINISH) bFinish1 = true;

    ret = CLPollingCalibration(hDevice2, &calStatus2);
    if(ret != SUCCESS) return;
    if (calStatus2 == CL_CALIBMEAS_FINISH) bFinish2 = true;
}

printf("Complete the zero calibration¥n");
```

3.3.2 Performing a manual measurement

The CL-SDK provides CLDoManualMeasurement() and CLDoManualMeasurementAll() as a manual measuremen function.

CLDoManualMeasurement()

This function performs a manual measurement of the instrument that you specified.
This function cannot perform synchronous measurements of multiple instruments.

CLDoManualMeasurementAll()

This function performs a manual measurement of all instruments which are connected to the PC.
This function can perform practically synchronous measurements of multiple instruments.
However, in this function, you cannot specify the order of the measurement of each instrument.

[Note]

The timings of measurements of each instrument are not simultaneous completely.
In addition, the more instruments you operate, the longer time lag between the first measurement and the last measurement will be.

```
/* an example for manual measurement */

// The definition of a object handle
DEVICE_HANDLE hDevice = NULL;        // 測定器のオブジェクトハンドラの定義

// get a object handle
ER_CODE ret = CLOpenDevice(&hDevice);
if (ret != SUCCESS) return;

// Set the remote mode of a instrument ON
ret = CLSetRemoteMode(hDevice, CL_RMODE_ON);
if (ret != SUCCESS){
    CLCloseDevice(hDevice1);
    return;
}
```

```
// Perform a manual measurement
int exposureTime = 400;                         // exposure time
int cumulativeNum = 10;                         // cumulative number

CL_MEASSTATUS status = CL_MEAS_FREE; // measurement status



ret = CLDoManualMeasurement(hDevice1, &exposureTime, &cumulativeNum);
if (ret > WARNING) return;

// Wait until a measurement is complete
do {
    // Confirm the measurement status after waiting for a moment
    Sleep(100);

    ret = CLPollingMeasure(hDevice, &status);
    if (ret != SUCCESS) return;

} while (status != CL_MEAS_FINISH);

// Get a measurement data
CL_MEASDATA data;



// Get the illuminance spectral data
ret = CLGetMeasData(hDevice, CL_COLORSPACE_SPC, &data);
if (ret > WARNING) return;
for (int i = 0; i < IRRADIANCE_LEN; i++){
    printf("SpctralData[%d]¥t%.010f¥n", i, data.Spc.Data[i]);
}

// Get the Ev/x/y data
ret = CLGetMeasData(hDevice, CL_COLORSPACE_EVXY, &data);
if (ret > WARNING) return;
printf("Evxy.Ev¥t%.010f¥nEvxy.x¥t%.010f¥nEvxy.y¥t%.010f¥n", data.Evxy.Ev, data.Evxy.x, data.Evxy.y);
```

```
// Get the color rendering index data
ret = CLGetMeasData(hDevice, CL_COLORSPACE_RENDERING, &data);
if (ret > WARNING) return;
for (int i = 0; i < CL_RENDERING_LEN; i++){
    printf("Rendering[%d]¥t%.010f¥n", i, data.Rendering.Data[i]);
}
```

3.4 The measurement by key trigger of the instrument

The CL-SDK can perform a measurement by a key trigger with the instrument.
Confirm the states of the trigger key (pressed or not pressed) by CLGetButtonStatus().
When the trigger key is pressed, the confirm process When pressing the trigger key, stops the confirmation process, and by executing a CLDoMeasurement(), the CL-SDK can perform a measurement by key trigger.

```
/* The example for a measurement by key trigger of the instrument */

// The definition of the object handle
DEVICE_HANDLE hDevice = NULL;

// Get the object handle
ER_CODE ret = CLOpenDevice(&hDevice);
if (ret != SUCCESS) return;

// Set the remote mode ON
ret = CLSetRemoteMode(hDevice, CL_RMODE_ON);
if (ret != SUCCESS){
    CLCloseDevice(hDevice);
    return;
}

// Wait until being pressed the meas button
CL_KEYINFO key_state;
do {
    // Confirm the key status of the meas button
    ret = CLGetButtonStatus(hDevice, &key_state);
    if (ret != SUCCESS) return;
} while (key_state.MeasKey != CL_KEY_STATE_ON);    // If the meas button is pressed, finish the processing

// Perform a measurement
int meastime;
ret = CLDoMeasurement(hDevice, &meastime);
```

3.5 Getting the stored data in the instrument

The CL-SDK can get the measurement data which is stored in the instrument.
Get the stored data with specifying the colorimetric type to be got.

```
/* The example of getting the stored data in the instrument */

// The definition of the object handle
DEVICE_HANDLE handle = NULL;

// Get the object handle
ER_CODE ret = CLOpenDevice(&handle);
if(ret != SUCCESS) return;

// Set the remote mode ON
ret = CLSetRemoteMode(handle, CL_RMODE_ON);
if(ret != SUCCESS){
    CLCloseDevice(handle);
    return;
}

// Get the number of data which is stored  in the instrument
int32_km num;      // Number of stored data
ret = CLGetDeviceStoredDataNum(handle, &num);
if(ret != SUCCESS) return;

// Get the stored data (Ev/x/y) in the instrument
CL_LISTDATA *pStoredData = new CL_LISTDATA[num];
ret = CLGetDeviceStoredData(handle, pStoredData, CL_COLORSPACE_EVXY, num);
if(ret != SUCCESS){
    delete [] pStoredData;
    return;
}

for(int32_km i = 0; i < num; i++){
    // The stored data
```

```
    printf("[%4d/%2d/%2d %2d:%2d:%2d]¥n",
        pStoredData[i].DateTime.Year, pStoredData[i].DateTime.Month, pStoredData[i].DateTime.Day,
        pStoredData[i].DateTime.Hour, pStoredData[i].DateTime.Minute, pStoredData[i].DateTime.Second
    );
    // The stored data (Ev/x/y)
    printf("data[%d].Ev:%f¥n", i + 1, pStoredData[i].Data.Evxy.Ev);
    printf("data[%d].x:%f¥n", i + 1, pStoredData[i].Data.Evxy.x);
    printf("data[%d].y:%f¥n¥n", i + 1, pStoredData[i].Data.Evxy.y);
}

delete [] pStoredData;
```

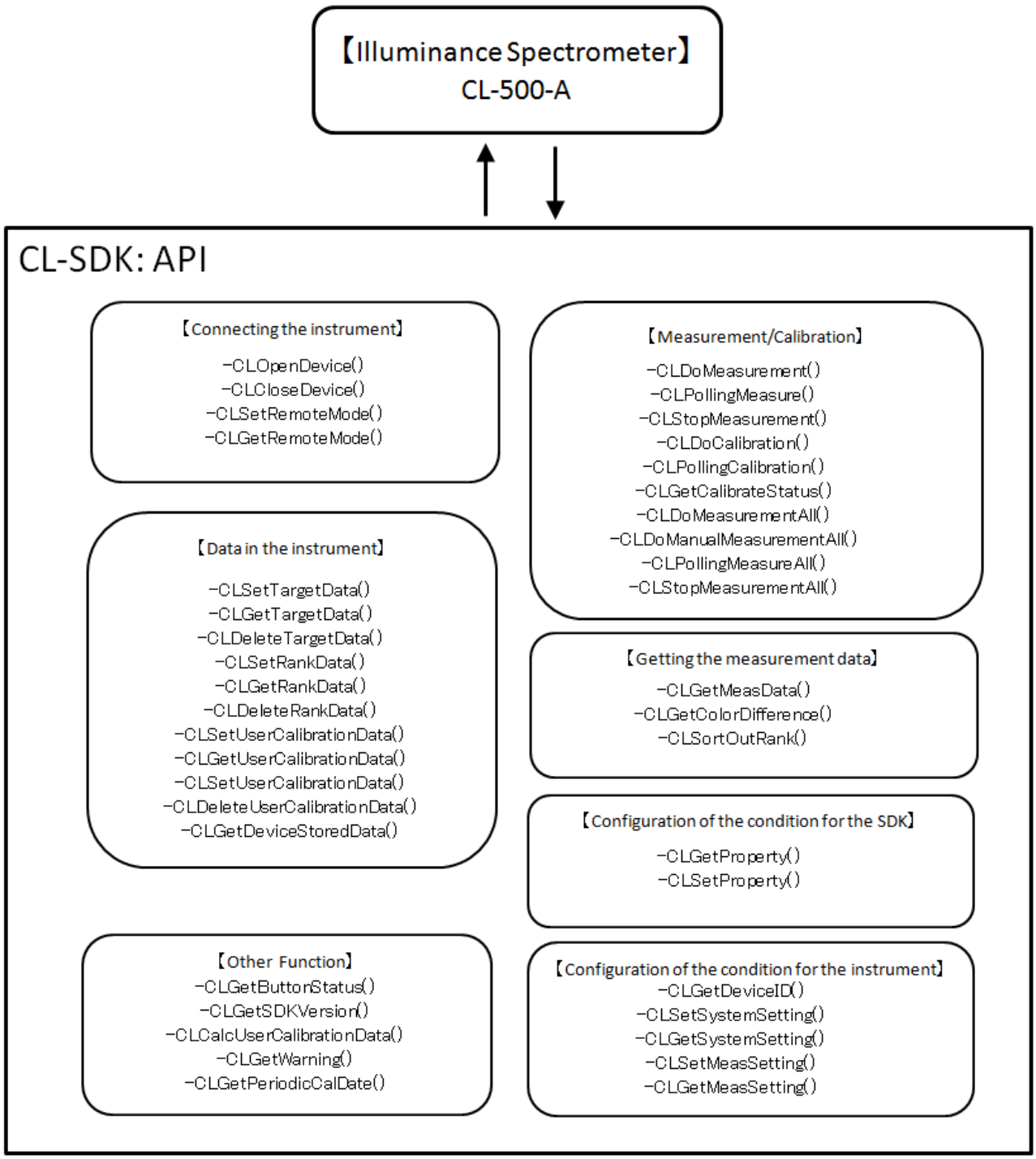
4. CL-SDK API Reference

CL-SDK API List

The all API which provides with the CL-SDK is shown below.

Function	Explanation
CLOpenDevice()	Gets the object handle of the instrument which is connected on the PC
CLCloseDevice()	Releases the object handle of the instrument which is stored in the CL-SDK
CLSetRemoteMode()	Configure a remote mode ON/OFF
CLGetRemoteMode()	Gets the remote mode ON/OFF
CLDoMeasurement()	Performs a measurement
CLDoManualMeasurement()	Performs a manual measurement
CLPollingMeasure()	Confirms the implementation status of measurement
CLStopMeasurement()	Stops a measurement
CLDoMeasurementAll()	Performs a measurement with all instrument
CLDoManualMeasurementAll()	Performs a measurement with all instrument
CLPollingMeasureAll()	Confirms the implementation status of a measurement by CLDoMeasurementAll()
CLStopMeasurementAll()	Stops a measurement by CLDoMeasurementAll()
CLDoCalibration()	Performs a zero calibration
CLPollingCalibration()	Gets the implementation status of a zero calibration
CLGetCalibrateStatus()	Gets a status whether a zero calibration have performed
CLGetMeasData()	Gets the each colorimetric data at the latest measurement
CLGetColorDifference()	Gets each color difference data at the latest measurement
CLSortOutRank()	Selects rank for the measurement data
CLSetRankData()	Registers a rank data which is used when selecting rank
CLGetRankData()	Gets the rank data which is registered in the instrument
CLDeleteRankData()	Deletes the rank data which is registered in the instrument
CLSetTargetData()	Registers a target data to the instrument
CLGetTargetData()	Gets the target data which is registered in the instrument
CLDeleteTargetData()	Deletes the target data which is registered in the instrument
CLGetDeviceStoredDataNum()	Gets the number of stored data in the instrument
CLGetDeviceStoredData()	Gets the number of stored data in the instrument
CLDeleteDeviceStoredData()	Deletes the stored data in the instrument
CLSetUserCalibrationData()	Registers a user calibration coefficient to the instrument
CLGetUserCalibrationData()	Gets the user calibration coefficient which is registered in the instrument
CLDeleteUserCalibrationData()	Deletes a user calibration coefficient which is registered in the instrument
CLCalcUserCalibrationData()	Calculates a user calibration coefficients
CLSetProperty()	Configures the measurement condition when control the instrument by the CL-SDK
CLGetProperty()	Gets the measurement condition when control the instrument by the CL-SDK
CLGetButtonStatus()	Gets the key status for instrument
CLSetMeasSetting()	Configures the measurement condition in standalone measurement
CLGetMeasSetting()	Gets the measurement condition in standalone measurement
CLSetSystemSetting()	Configures the system setting of the instrument

CLGetSystemSetting()	Gets the system setting of the instrument
CLGetDeviceID()	Gets the instrument information
CLGetSDKVersion()	Gets the version information of the CL-SDK
CLGetWarning()	Gets a detail of warning at the latest processing
CLGetPeriodicCalDate()	Gets the information about the periodic calibration date


【Illuminance Spectrometer】
CL-500-A
CL-SDK: API
【Connecting the instrument】
-CLOpenDevice()
-CLCloseDevice()
-CLSetRemoteMode()
-CLGetRemoteMode()
【Data in the instrument】
-CLSetTargetData()
-CLGetTargetData()
-CLDeleteTargetData()
-CLSetRankData()
-CLGetRankData()
-CLDeleteRankData()
-CLSetUserCalibrationData()
-CLGetUserCalibrationData()
-CLSetUserCalibrationData()
-CLDeleteUserCalibrationData()
-CLGetDeviceStoredData()
【Other Function】
-CLGetButtonStatus()
-CLGetSDKVersion()
-CLCalcUserCalibrationData()
-CLGetWarning()
-CLGetPeriodicCalDate()
【Measurement/Calibration】
-CLDoMeasurement()
-CLPollingMeasure()
-CLStopMeasurement()
-CLDoCalibration()
-CLPollingCalibration()
-CLGetCalibrateStatus()
-CLDoMeasurementAll()
-CLDoManualMeasurementAll()
-CLPollingMeasureAll()
-CLStopMeasurementAll()
【Getting the measurement data】
-CLGetMeasData()
-CLGetColorDifference()
-CLSortOutRank()
【Configuration of the condition for the SDK】
-CLGetProperty()
-CLSetProperty()
【Configuration of the condition for the instrument】
-CLGetDeviceID()
-CLSetSystemSetting()
-CLGetSystemSetting()
-CLSetMeasSetting()
-CLGetMeasSetting()

4.1 The definition value

	Value	Explanation
IRRADIANCE_LEN	421	The number of illuminance spectral data
IRRADIANCE_BEGIN	360	The start wavelength for illuminance spectral data
IRRADIANCE_END	780	The end wavelength for illuminance spectral data
IRRADIANCE_PITCH	1	The wavelength pitch for illuminance spectral data
CL_OVER_ERROR_VALUE	-10000	The value to be obtained when exceeds measurement range by the CL-500A
CL_INCOMPUTABLE_VALUE	-11000	The value to be obtained when measure the light source which cannot be calculated the correlated color temperature
CL_USERCALIB_NAMESIZE	13	The maximum character length of user calibration data name
CL_USERCALIB_CH_ALLDEL	-1	The value for deleting the all user calibration coefficient in the instrument
CL_USERCALIB_CH_START	1	The start No. which is available when specifying the user calibration coefficient
CL_USERCALIB_CH_END	10	The last No. which is available when specifying the user calibration coefficient
CL_TARGET_NAMESIZE	13	The maximum character length of target data name
CL_TARGET_NO_ALLDEL	-1	The value for deleting the all target data in the instrument
CL_TARGET_NO_START	1	The start No. which is available when specifying the target data
CL_TARGET_NO_END	20	The last No. which is available when specifying the target data
CL_RANK_POINTNUM	10	The maximum point number which can configure as rank area
CL_RANK_NAMESIZE_SAVE	41	The maximum byte size of rank data name
CL_RANK_NO_ALLDEL	-1	The value for deleting the all rank data in the instrument
CL_RANK_NO_START	1	The start No. which can specify on the rank data
CL_RANK_NO_END	20	The end No. which can specify on the rank data
CL_LIST_NO_ALLDEL	-1	The value for deleting the all stored data in the instrument
CL_LIST_NO_START	1	The start No. which can specify on the stored data in the instrument
CL_LIST_NO_END	100	The end No. which can specify on the stored data in the instrument
CL_RENDERING_LEN	16	The buffer size of color rendering index
CL_USERCUSTOM_LEN	4	The number of data which can specify on custom color mode

4.2 The enumeration

CL_REMOTEMODE

This enumeration is used to set/get the configuration of the remote mode by CLSetRemoteMode() or CLGetRemoteMode().

CL_REMOTEMODE	Value	Explanation
CL_RMODE_OFF	0	A remote mode is OFF
CL_RMODE_ON	1	A remote mode is ON

CL_MEASSTATUS

This enumeration is used to get the implementation status of measurement by CLPollingMeasure().

CL_MEASSTATUS	Value	Explanation
CL_MEAS_FREE	0	A measurement is not performed
CL_MEAS_BUSY	1	A measurement is performing
CL_MEAS_FINISH	2	A measurement is completed

CL_CALIBSTATUS

This enumeration is used to get the status of zero calibration by CLGetCalibrateStatus().

CL_CALIBSTATUS	Value	Explanation
CL_CALIB_NA	0	A zero calibration is not able to perform
CL_CALIB_OK	1	A zero calibration is completed
CL_CALIB_WR	2	A zero calibration is recommended (It have passed a certain time from the last calibration)
CL_CALIB_NG	3	A zero calibration is not performed

(*1) A measurement is possible even if a zero calibration does not perform again.
However, to perform a measurement with accuracy, perform a zero calibration.

CL_CALIBMEASSTATUS

This enumeration is used to get the implementation status of zero calibration by CLPollingCalibration().

CL_CALIBMEASSTATUS	Value	Explanation
CL_CALIBMEAS_FREE	0	A zero calibration is not performed
CL_CALIBMEAS_BUSY	1	A zero calibration is performing
CL_CALIBMEAS_FINISH	2	A zero calibration is completed

CL_RANK_TYPE

This enumeration is used to set/get the configuration type of the rank area by CLSetRankData() or CLGeRankData().

CL_RANK_TYPE	Value	Explanation
CL_RANK_CHROMA	0	The rank area is specified by xy
CL_RANK_COLORTEMP	1	The rank area is specified by Tcp/⊿uv

CL_KEYSTATUS

This enumeration is used to get the key status by CLGetButtonStatus().

CL_KEYSTATUS	Value	Explanation
CL_KEY_STATE_OFF	0	The key is not pressed
CL_KEY_STATE_ON	1	The key is pressed

CL_COLORSPACE

This enumeration is used to specify the colorimetric data to be obtained by CLGetMeasData() or CLGetDeviceStoredData().

CL_MEAS_STATUS	Value	Explanation
CL_COLORSPACE_EVXY	0	Ev, x, y
CL_COLORSPACE_EVUV	1	Ev, u', v'
CL_COLORSPACE_EVTCPJISDUV	2	Ev, Correlated color temperature Tcp(JIS) (*1), ⊿uv
CL_COLORSPACE_EVTCPDUV	3	Ev, Correlated color temperature Tcp(*2), ⊿uv
CL_COLORSPACE_EVDWPE	4	Ev, Dominant wavelength λd, Excitation purity Pe
CL_COLORSPACE_XYZ	5	X, Y, Z
CL_COLORSPACE_RENDERING	6	Color Rendering Index (Ra, R1 to R15)
CL_COLORSPACE_PW	7	Peak wavelength
CL_COLORSPACE_SPC	8	Illuminance spectral data
CL_COLORSPACE_SCOTOPIC	9	Ev, scotopic lux Ev', S/P ratio

(*1) : Color temperature which uses the formula designated by JIS Z 8725

(*2) : Though the color temperature computing algorithm that Konica Minolta adopted is similar to JIS, a high speed computing algorithm is adopted to speed up calculation time.
There might be a slight difference in the values of Tcp between Tcp (JIS) in some cases.
The margin of error of the chromatic range of the color temperature calculable designatedbyJISZ 8725 between the Tcp (JIS) based values and Tcp values are within ±3%.

CL_PROPERTIES

This enumeration is used to set/get the measurement condition for the CL-SDK by CLSetProperty() or CLGetProperty().

CL_PROPERTIES	Value	Explanation
CL_PR_OBSERVER	0	Observer
CL_PR_ILLUNIT	1	Illuminance units

CL_OBSERVER

This enumeration is used to set/get the observer by CLSetProperty(), by CLGetProperty(), CLSetMeasSettings() or CLGetMeasSettings().

CL_OBSERVER	Value	Explanation
CL_OBS_02DEG	0	2° observer
CL_OBS_10DEG	1	10° observer

CL_ILLUMINANT_UNIT

This enumeration is used to set/get the illuminance units by CLSetProperty(), by CLGetProperty(), CLSetMeasSettings() or CLGetMeasSettings().

CL_ILLUMINANT_UNIT	Value	Explanation
CL_ILLUNIT_LX	0	lux
CL_ILLUNIT_FCD	1	foot-candela **[Note]** When use CLGetMeasSetting(), foot-candela cannot be configured, if display language of instrument is Japanese.

CL_MEASSETTYPE

This enumeration is used to specify the measurement condition type by CLSetMeasSetting() or CLGetMeasSetting().

CL_MEASSETTYPE	Value	Explanation
CL_MEASSET_DISPTYPE	0	Display Type
CL_MEASSET_OBS	1	Observer
CL_MEASSET_COLORSPACE	2	Color Space
CL_MEASSET_ILLUNIT	3	Illuminance unit
CL_MEASSET_EXPOSURETIME	4	Measurement Time
CL_MEASSET_USERCALIBCH	5	User calibration CH
CL_MEASSET_TARGETNO	6	Target data No.
CL_MEASSET_USERCUSTOM	7	Custom color mode
CL_MEASSET_MEASMODE	8	Measurement mode
CL_MEASSET_TIMERCONF	9	Timer configuration
CL_MEASSET_USERWAVELENGTH	10	Arbitary wavelengths in custom color mode

CL_DISP_TYPES

This enumeration is used to set/get the configuration of display type by CLSetMeasSetting() or CLGetMeasSetting().

CL_DISP_TYPES	Value	Explanation
CL_DISP_ABS	0	Absolute
CL_DISP_DIFF	1	Difference
CL_DISP_RANK	2	Select Rank

CL_COLOR_MODE

This enumeration is used to set/get the configuration of color space by CLSetMeasSetting() or CLGetMeasSetting().

CL_COLOR_MODE	Value	Explanation
CL_MODE_EVXY	0	Ev, x, y
CL_MODE_EVUV	1	Ev, u', v'
CL_MODE_EVTCPDUV	2	Ev, Correlated color temperature Tcp, ⊿uv
CL_MODE_XYZ	3	X, Y, Z
CL_MODE_EVDWPE	4	Ev, Dominant wavelength λd, Excitation purity Pe
CL_MODE_RENDERING	5	Color rendering index (Ra, R1 to R15)
CL_MODE_SPECTRAL_GRAPH	6	Spectral irradiance graph, Peak wavelength
CL_MODE_CUSTOM	7	Custom

CL_MEASUREMENT_TIME

This enumeration is used to set/get the configuration of measurement time by CLSetMeasSetting() or CLGetMeasSetting().

CL_MEASUREMENT_TIME	Value	Explanation
CL_MEAS_TIME_FAST	0	FAST mode : The measurement time : about 0.5 second
CL_MEAS_TIME_SLOW	1	SLOW mode : The measurement time : about 2.5 second
CL_MEAS_TIME_AUTO	2	AUTO mode (Measures in accordance with the brightness of the light source.). The measurement time : 0.5 to 27 seconds
CL_MEAS_TIME_SUPER_FAST	3	SUPER FAST mode : The measurement time : about 0.2second **[Note]** : The SUPER FAST mode is only available when controlling from the PC (Not available in the standalone measurement).

CL_CUSTOMDATA_ITEM

This enumeration is used to set/get the configuration of display items for custom color mode by CLSetMeasSetting() or CLGetMeasSetting().

CL_CUSTOMDATA_ITEM	Value	Explanation
CL_CUSTOM_NONE	0	No display
CL_CUSTOM_EV_DIFF	1	Ev(Difference with target)
CL_CUSTOM_EV_RATIO	2	Ev(Percentage of target)
CL_CUSTOM_SX	3	x
CL_CUSTOM_SY	4	y
CL_CUSTOM_U	5	u'
CL_CUSTOM_V	6	v'
CL_CUSTOM_TCP	7	Correlated color temperature Tcp
CL_CUSTOM_DUV	8	⊿uv

CL_CUSTOM_LX_DIFF	9	X(Difference with target)
CL_CUSTOM_LX_RATIO	10	X(Percentage of target)
CL_CUSTOM_LY_DIFF	11	Y(Difference with target)
CL_CUSTOM_LY_RATIO	12	Y(Percentage of target)
CL_CUSTOM_LZ_DIFF	13	Z(Difference with target)
CL_CUSTOM_LZ_RATIO	14	Z(Percentage of target)
CL_CUSTOM_DW	15	Dominant wavelength λd
CL_CUSTOM_PE	16	Excitation purity Pe
CL_CUSTOM_RANK	17	Rank
CL_CUSTOM_RA	18	Ra
CL_CUSTOM_R1	19	R1
CL_CUSTOM_R2	20	R2
CL_CUSTOM_R3	21	R3
CL_CUSTOM_R4	22	R4
CL_CUSTOM_R5	23	R5
CL_CUSTOM_R6	24	R6
CL_CUSTOM_R7	25	R7
CL_CUSTOM_R8	26	R8
CL_CUSTOM_R9	27	R9
CL_CUSTOM_R10	28	R10
CL_CUSTOM_R11	29	R11
CL_CUSTOM_R12	30	R12
CL_CUSTOM_R13	31	R13
CL_CUSTOM_R14	32	R14
CL_CUSTOM_R15	33	R15
CL_CUSTOM_EVS_DIFF	34	Scotopic lux Ev' (Difference with target)
CL_CUSTOM_EVS_RATIO	35	Scotopic lux Ev' (Percentage of target)
CL_CUSTOM_SP	36	S/P ratio (Difference with target)
CL_CUSTOM_EE_AB1	37	Ee(λ1)
CL_CUSTOM_EE_AB2	38	Ee(λ2)
CL_CUSTOM_EE_AB3	39	Ee(λ3)
CL_CUSTOM_EE_AB4	40	Ee(λ4)

CL_SYSTEMTYPE

This enumeration is used to specify the system type by CLSetSystemSetting() or CLGetSystemSetting().

CL_SYSTEMTYPE	Value	Explanation
CL_SYSTEM_DATETIME	0	Date and time
CL_SYSTEM_DISPLAY	1	Orientation of the display
CL_SYSTEM_BEEP	2	Buzzer setting
CL_SYSTEM_LANGUAGE	3	Display language
CL_SYSTEM_DATEFORMAT	4	Date display format
CL_SYSTEM_UCAL_LIMIT	5	Zero calibration expiry
CL_SYSTEM_AUTOPOWEROFF	6	Auto power off
CL_SYSTEM_PCALNOTIFY	7	Periodic calibration warning message

CL_DISPLAYTYPE

This enumeration is used to set/get the configuration of orientation of the display in the instrument by CLSetSystemSetting() or CLGetSystemSetting().

CL_DISPLAYTYPE	Value	Explanation

DISPLAY_NORMAL	0	The LCD screen displays normal
DISPLAY_INVERSE	1	The LCD screen displays invert

CL_BEEP

This enumeration is used to set/get the configuration of buzzer in the instrument by CLSetSystemSetting() or CLGetSystemSetting().

CL_BEEP	Value	Explanation
CL_BEEP_OFF	0	The buzzer does not sound
CL_BEEP_ON	1	The buzzer sounds **[Note]** When the measurement time is "SUPER FAST" mode, the buzzer does not sound even if the buzzer setting is On.

CL_LANGMODE

This enumeration is used to set/get the configuration of display language in the instrument by CLSetSystemSetting() or CLGetSystemSetting().

CL_LANGMODE	Value	Explanation
CL_LANG_ENG	0	English
CL_LANG_JAN	1	Japanese
CL_LANG_CHN	2	Chinese

CL_TYPE_DATEFORMAT

This enumeration is used to set/get the configuration of date display format in the instrument by CLSetSystemSetting() or CLGetSystemSetting().

CL_TYPE_DATEFORMAT	Value	Explanation
CL_TYPE_YYMMDD	0	Display the date in year/month/day order
CL_TYPE_MMDDYY	1	Display the date in month/day/year order
CL_TYPE_DDMMYY	2	Display the date in day/month/year order

CL_USERCAL_LIMIT

This enumeration is used to set/get the configuration of zero calibration expiry in the instrument by CLSetSystemSetting() or CLGetSystemSetting().

CL_USERCAL_LIMIT	Value	Explanation
CL_USERCAL_LIMIT_3H	0	The calibration prompt screen is displayed when elapsing 3 hour from last calibration

CL_USERCAL_LIMIT_6H	1	The calibration prompt screen is displayed when elapsing 6 hour from last calibration
CL_USERCAL_LIMIT_12H	2	The calibration prompt screen is displayed when elapsing 12 hour from last calibration
CL_USERCAL_LIMIT_24H	3	The calibration prompt screen is displayed when elapsing 24 hour from last calibration
CL_USERCAL_LIMITLESS	4	The calibration prompt screen is not displayed

CL_AUTOPOWEROFF

This enumeration is used to set/get the configuration of auto power off in the instrument by CLSetSystemSetting() or CLGetSystemSetting().

CL_AUTOPOWEROFF	Value	Explanation
CL_AUTOPOWEROFF_OFF	0	Disable the configuration of the auto power off
CL_AUTOPOWEROFF_ON	1	Enable the configuration of the auto power off

When the configuration of the auto power off is enabled, the instrument turns power OFF automatically, if the instrument is not operated.

When the instrument is connected to the PC, the instrument does not turn power off automatically even if the auto power off setting is ON.

CL_PCALNOTIFY

This enumeration is used to set/get the configuration of periodic calibration in the instrument by CLSetSystemSetting() or CLGetSystemSetting().

CL_PCALNOTIFY	Value	Explanation
CL_PCALNOTIFY_OFF	0	Disables display of the periodic calibration warning message
CL_PCALNOTIFY_ON	1	Enables display of the periodic calibration warning message

CL_PERIODICCAL_TYPE

This enumeration is used to get the periodic calibration date in the instrument by CLGetPeriodicCalDate().

CL_PERIODICAL_TYPE	Value	Explanation
CL_PERIODICCAL_START	0	Start date on the periodic calibration
CL_PERIODICCAL_END	1	Expiration date on the periodic calibration

CL_MEASMODE

This enumeration is used to set/get the configuration of measurement mode in the instrument by CLSetMeasSetting() or CLGetMeasSetting().

CL_MEASMODE	定義値	説明
CL_MEASMODE_SINGLE	0	Single measurement
CL_MEASMODE_AVERAGED	1	Averaged measurement
CL_MEASMODE_CONTINUOUS	2	Continuous Measurement

4.3 The structure / The union

CL_TARGET_DATA

Brief :
The structure of target data

Syntax :

```
typedef struct tCL_TARGET_DATA
{
    real            SPCData[IRRADIANCE_LEN];
    CL_DATETIME     DateTime;
    int8_km         Name[CL_TARGET_NAMESIZE];
}
CL_TARGET_DATA;
```

Parameter :

Parameter	Explanation
SPCData	Illuminance spectral data Wavelength range : 360 to 780nm Wavelength pitch : 1nm Number of data : 421 data
DateTime	The stored date and time Available date : 2011/01/01 0:00:00 to 2100/12/31 23:59:59
Name	Target data name The character length : 13 characters (include NULL character) Available character : refer to §5.2 Character code table

Explanation :
This structure is used by CLSetTargetData() or CLGetTargetData() when treating the target data.

CL_USERCALIB DATA

Brief :
The structure of user calibration coefficient

Syntax :

```
typedef struct tCL_USERCALIB_DATA
{
    real          Coef[IRRADIANCE_LEN];
    CL_DATETIME   DateTime;
    int8_km       Name[CL_USERCALIB_NAMESIZE];
}
CL_USERCALIB_DATA;
```

Parameter :

Parameter	Explanation
Coef	The user calibration coefficient Wavelength range : 360 to 780nm Wavelength pitch : 1nm Number of data : 421 data Setting range : 0.001 to 1000
DateTime	The date information Available date : 2011/01/01 0:00:00 to 2100/12/31 23:59:59
Name	The calibration coefficient name The character length : 13 characters (include NULL character) Available character : refer to §5.2 Character code table

Explanation :
This structure is used by CLSetUserCalibrationData() or CLGetUserCalibrationData() when treating the user calibration coefficient

CL_MEASDATA

Brief :
The union of measurement data

Syntax :

```
typedef union tCL_MEASDATA
{
    CL_EvxyDATA             Evxy;
    CL_EvuvDATA             Evuv;
    CL_EvTduvDATA           EvTduv;
    CL_EvDWPeDATA           EvDWPe;
    CL_XYZDATA              XYZ;
    CL_RenderingDATA        Rendering;
    CL_PWDATA               Pw;
    CL_SPCDATA              Spc;
}
CL_MEASDATA;
```

Variable :

Parameter	Explanation
Evxy	Ev/x/y data
Evuv	Ev/u' /v' data
EvTduv	Ev/Tcp/⊿uv data
EvDWPe	Ev/Dominant wavelength /Excitation purity data
XYZ	X/Y/Z data
Rendering	Color rendering index
Pw	Peak wavelength
Spc	Illuminance spectral data

Explanation :
This union is used by CLGetMeasData() when treating the measurement data.

[Note]
This variable is union. If getting the multiple colorimetric data by CLGetMeasData(), get the next colorimetric data after storing the obtained colorimetric data to other buffer.

CL_SPCDATA

Brief :
The structure of illuminance spectral data

Syntax :

```
typedef struct tCL_SPCDATA
{
    real  Data[IRRADIANCE_LEN];
}
CL_SPCDATA;
```

Variable :

Parameter	Explanation
Data	Illuminance spectral data

Explanation :
This structure is used by CLGetMeasData() when treating the measurement data.

CL_EvxyDATA

Brief :
The structure of Evxy data

Syntax :

```
typedef struct tCL_EvxyDATA
{
    float Ev;
    float x;
    float y;
}
CL_EvxyDATA;
```

Variable :

Parameter	Explanation
Ev	Ev data
x	x data
y	y data

Explanation :
This structure is used by CLGetMeasData() when treating the measurement data.
This structure is stored Ev/x/y data.

CL_EvuvDATA

Brief :
The structure of Ev/u' /v' data

Syntax :

```
typedef struct tCL_EvuvDATA
{
    float Ev;
    float u;
    float v;
}
CL_EvuvDATA;
```

Parameter :

Parameter	Explanation
Ev	Ev data
u	u' data
v	v' data

Explanation :
This structure is used by CLGetMeasData() when treating the measurement data.
This structure is stored Ev/u' /v' data.

CL_EvTduvDATA

Brief :
The structure of Ev/Tcp/⊿uv data

Syntax :

```
typedef struct tCL_EvTduvDATA
{
    float Ev;
    float T;
    float duv;
}
CL_EvTduvDATA;
```

Variable :

Parameter	Explanation
Ev	Ev data
T	Correlated color temperature Tcp data
duv	⊿uv data

Explanation :
This structure is used by CLGetMeasData() when treating the measurement data.
This structure is stored Ev/Tcp/⊿uv data.

CL_EvDWPeDATA

Brief :
The structure of Ev/ Dominant wavelength/Excitation purity data

Syntax :

```
typedef struct tCL_EvDWPeDATA
{
    float Ev;
    float DW;
    float Pe;
}
CL_EvDWPeDATA;
```

Variable :

Parameter	Explanation
Ev	Ev data
DW	Dominant wavelength λd data
Pe	Excitation purity Pe data

Explanation :
This structure is used by CLGetMeasData() when treating the measurement data.
This structure is stored Ev/λd/Pe data.

CL_XYZDATA

Brief :
The XYZ data structure

Syntax :

```
typedef struct tCL_XYZDATA
{
    float X;
    float Y;
    float Z;
}
CL_XYZDATA;
```

Variable :

Parameter	Explanation
X	X data
Y	Y data
Z	Z data

Explanation :
This structure is used by CLGetMeasData() when treating the measurement data.
This structure is stored X/Y/Z data.

CL_RenderingDATA

Brief :
The structure of color rendering index

Syntax :

```
typedef struct tCL_RenderingDATA
{
     float Data[CL_RENDERING_LEN];
}
CL_RenderingDATA;
```

Variable :

Parameter	Explanation
Data	The color rendering index The data is stored in the array in below order

0	1	2	・・・	14	15
Ra	R1	R2	・・・	R14	R15

Explanation :
This structure is used by CLGetMeasData() or CLGetColorDifference() when treating the measurement data.
This structure is stored color rendering index.

CL_PWDATA

Brief :
The structure of Peak wavelength

Syntax :

```
typedef struct tCL_PWDATA
{
    float PeakWave;
}
CL_PWDATA;
```

Variable :

Parameter	Explanation
PeakWave	Peak wavelength data

Explanation :
This structure is used by CLGetMeasData() or CLGetColorDifference() when treating the measurement data.
This structure is stored peak wavelength data.

CL_ScotopicDATA

Brief :
The structure of Ev, Scotopic lux Ev', S/P ratio data

Syntax :

```
typedef struct     tCL_ScotopicDATA
{
    float                  Ev;
    float                  Es;
    float                  SP;
}
CL_ScotopicDATA;
```

Variable :

Parameter	Explanation
Ev	Ev data
Es	Scotopic lux Ev' data
SP	S/P ratio data

CL_DIFFDATA

Brief :
The union of difference data

Syntax :

```
typedef union tCL_DIFFDATA
{
    CL_Evxy_DIFFDATA        Evxy;
    CL_Evuv_DIFFDATA        Evuv;
    CL_EvTduv_DIFFDATA      EvTduv;
    CL_EvDWPe_DIFFDATA      EvDWPe;
    CL_XYZ_DIFFDATA         XYZ;
    CL_RenderingDATA        Rendering;
    CL_PWDATA               Pw;
}
CL_DIFFDATA;
```

Variable :

Parameter	Explanation
Evxy	Color difference data of Ev/x/y
Evuv	Color difference data of Ev/u/v
EvTduv	Color difference data of Ev/Tcp
EvDWPe	Color difference data of Ev/Dominant wavelength/ Excitation purity
XYZ	Color difference data of X/Y/Z
Rendering	Color difference data of color rendering index
Pw	Color difference data of peak wavelength

Explanation :
This union is used by CLGetColorDifference() when treating the color difference data.

[Note]
This variable is union. If getting the multiple difference data by CLGetMeasData(), get the next difference data after storing the obtained difference data to other buffer.

CL_Evxy_DIFFDATA

Brief :
The structure of color difference data for Ev/x/y

Syntax :

```
typedef struct tCL_Evxy_DIFFDATA
{
    float Ev_Abs;
    float Ev_Ratio;
    float x;
    float y;
}
CL_Evxy_DIFFDATA;
```

Variable :

Parameter	Explanation
Ev_Abs	Difference data of Ev (Difference with target)
Ev_Ratio	Difference data of Ev (Percentage of target)
x	Difference data of x
y	Difference data of y

Explanation :
This structure is used by CLGetColorDifference() when treating the difference data.
This structure is stored the color difference data for Ev/x/y.

CL_Evuv_DIFFDATA

Brief :
The structure of color difference data for Ev/u' /v'

Syntax :

```
typedef struct tCL_Evuv_DIFFDATA
{
    float Ev_Abs;
    float Ev_Ratio;
    float u;
    float v;
}
CL_Evuv_DIFFDATA;
```

Variable :

Parameter	Explanation
Ev_Abs	Difference data of Ev (Difference with target)
Ev_Ratio	Difference data of Ev (Percentage of target)
u	Difference data of u'
v	Difference data of v'

Explanation :
This structure is used by CLGetColorDifference() when treating the difference data.
This structure is stored the color difference data for Ev/u' /v' .

CL_EvTduv_DIFFDATA

Brief :
The structure of color difference data for Ev/Tcp

Syntax :

```
typedef struct tCL_EvTduv_DIFFDATA
{
    float Ev_Abs;
    float Ev_Ratio;
    float T;
    float duv;
}
CL_EvTduv_DIFFDATA;
```

Variable :

Parameter	Explanation
Ev_Abs	Difference data of Ev (Difference with target)
Ev_Ratio	Difference data of Ev (Percentage of target)
T	Difference data of Correlated color temperature Tcp
duv	Not used

Explanation :
This structure is used by CLGetColorDifference() when treating the difference data.
This structure is stored color difference data for Ev/Tcp.

[Note]
Since ⊿uv does not exist a difference data, “duv” is not stored a difference data.

CL_EvDWPe_DIFFDATA

Brief :
The structure of color difference data for Ev/λd/Pe

Syntax :

```
typedef struct tCL_EvDWPe_DIFFDATA
{
    float Ev_Abs;
    float Ev_Ratio;
    float DW;
    float Pe;
}
CL_EvDWPe_DIFFDATA;
```

Variable :

Parameter	Explanation
Ev_Abs	Color difference data of Ev (Difference with target)
Ev_Ratio	Color difference data of Ev (Percentage of target)
DW	Color difference data of Dominant wavelength λd
Pe	Color difference data of Excitation purity Pe

Explanation :
This structure is used by CLGetColorDifference() when treating the color difference data.
This structure is stored the color difference data for E/λd/Pe.

(*1)
When the measurement data is a complementary wavelength, the measurement data is minus value.

CL_XYZ_DIFFDATA

Brief :
The structure of color difference data for X/Y/Z

Syntax :

```
typedef struct tCL_XYZ_DIFFDATA
{
    float X_Abs;
    float Y_Abs;
    float Z_Abs;
    float X_Ratio;
    float Y_Ratio;
    float Z_Ratio;
}
CL_XYZ_DIFFDATA;
```

Variable :

Parameter	Explanation
X_Abs	Difference data of X (Difference with target)
Y_Abs	Difference data of Y (Difference with target)
Z_Abs	Difference data of Z (Difference with target)
X_Ratio	Difference data of X (Percentage of target)
Y_Ratio	Difference data of Y(Percentage of target)
Z_Ratio	Difference data of Z (Percentage of target)

Explanation :
This structure is used by CLGetColorDifference() when treating the difference data.
This structure is stored the color difference data for X/Y/Z.

CL_Scotopic_DIFFDATA

概要：

The structure of color difference data for Ev, scotopic lux Ev', S/P ratio

Syntax:

```
typedef struct tCL_Scotopic_DIFFDATA{
    float Ev_Abs;
    float Ev_Ratio;
    float Es_Abs;
    float Es_Ratio;
    float SP;
}
CL_Scotopic_DIFFDATA;
```

Variable：

Parameter	Explanation
Ev_Abs	Difference data of Ev (Difference with target)
Ev_Ratio	Difference data of Ev (Percentage of target)
Es_Abs	Difference data of Ev' (Difference with target)
Es_Ratio	Difference data of Ev' (Percentage of target)
SP	Difference data of S/P ratio (Difference with target)

CL_RANK

Brief :
The structure of rank information which is sorted out

Syntax :

```
typedef struct tCL_RANK
{
    real          Ev;
    int8_km       RankName[CL_RANK_NAMESIZE];
    int32_km      RankNo;
}
CL_RANK;
```

Variable :

Parameter	Explanation
Ev	Ev data in measurement
RankName	Rank name of the corresponding rank
RankNo	The rank No. to be correspond

Explanation :
This structure is used by CLSortOutRank() when sorting out the rank.

CL_RANK_DATA

Brief :
The structure of rank data which is used in sorting out the rank

Syntax :

```
typedef struct tCL_RANK_DATA
{
    CL_RANK_TYPE   Type;
    int8_km        RankName[CL_RANK_NAMESIZE_SAVE];
    CL_xyDATA      xyPoint[CL_RANK_POINTNUM];
    CL_TcpduvDATA  TcpduvPoint[CL_RANK_POINTNUM];
    int32_km       PointNum;
    int32_km       Enable;
}
CL_RANK_DATA;
```

Variable :

Parameter	Explanation
Type	The point type which creates rank area (xy/Tcp・⊿uv) 0 : xy, 1 : Tcp・⊿uv
RankName	The rank name The character length : 41 characters (include NULL character)
xyPoint	The array for configuring the rank area by xy data Setting range 0.00 < x < 1.00, 0.00 < y < 1.00
TcpduvPoint	The array for configuring the rank area by Tcp/⊿uv data Setting range 2000 <= Tcp <= 50000, -0.03 <= ⊿uv <= 0.03
PointNum	The number of data which configures the rank area Setting range : 3 to 10
Enable	The availability of when sorting out the rank 0:not used, 1 : use

Explanation :
This structure is used by CLSetRankData() or CLGetRankData() when treating the rank data.

The sorting out the rank is performed in using only the rank data which a parameter “Enable” is use.

CL_xyDATA

Brief :
The structure of xy data which is used as rank data

Syntax :

```
typedef struct tCL_xyDATA
{
     float x;
     float y;
}
CL_xyDATA;
```

Variable :

Parameter	Explanation
x	x value Setting range : 0.00 < x < 1.00
y	y value Setting range : 0.00 < y < 1.00

Explanation :
This structure is used by CLSetRankData() or CLGetRankData() when treating the rank data.
This structure is stored the chromaticity xy value in each point which configures the rank area in the rank data.

CL_TcpduvDATA

Brief :
The structure of Tcp/⊿uv data which is used as rank data

Syntax :

```
typedef struct tCL_TcpduvDATA
{
    float Tcp;
    float duv;
}
CL_TcpduvDATA;
```

Variable :

Parameter	Explanation
Tcp	Correlated color temperature Tcp value Setting range : 2000 to 50000
duv	⊿uv value Setting range : -0.03 to 0.03

Explanation :
This structure is used by CLSetRankData() or CLGetRankData() when treating the rank data.
This structure is stored the Tcp and ⊿uv in each point which configures the rank area in the rank data.

CL_LISTDATA

Brief :
The structure of the stored data in the instrument

Syntax :

```
typedef struct tCL_LISTDATA
{
    CL_MEASDATA             Data;
    CL_DATETIME             DateTime;
    CL_OBSERVER             Observer;
    int32_km                OpCalibNo;
    int32_km                TargetNo;
    CL_MEASUREMENT_TIME     ExposureTime;
}
CL_LISTDATA;
```

Variable :

Parameter	Explanation
Data	Stored measurement data (*1)
DateTime	The date and time when saved the measurement data
Observer	The observer when saved the measurement data
OpCalibNo	User calibration CH when saved the measurement data
TargetNo	Target No. when saved the measurement data
ExposureTime	The measurement time at measurement

Explanation :
This structure is used by CLGetDeviceStoredData() when treating the stored data in the instrument.

(*1)
A measurement data is stored the colorimetric data which specifies by CLGetDeviceStoredData().

CL_MEASSETTING

Brief :
The structure of the measurement condition in standalone

Syntax :

```
typedef struct tCL_MEASSETTING
{
    CL_DISP_TYPES           DispType;
    CL_OBSERVER             Obs;
    CL_COLOR_MODE           ColorSpace;
    CL_ILLUMINANT_UNIT      IlluminantUnit;
    CL_EXPOSURE_TIME        ExposureTime;
    int32_km                UserCalibCh
    int32_km                TargetNo;
    CL_CUSTOMDATA_ITEM      UserCustom[CL_USERCUSTOM_LEN];
    CL_MEASMODE             MeasMode;
    CL_TIMERSETTINGS        TimerConf;
    CL_USERWAVELENGTH       UserWavelength;
}
CL_MEASSETTING;
```

Variable :

Parameter	Explanation
DispType	Display type 0:Absolute, 1:difference, 2:Selct Rank
Obs	Observer in standalone measurement 0 : 2° observer, 1 : 10° observer
ColorSpace	Color mode in standalone measurement Setting range : 0 to 7 (Refer to CL_COLOR_MODE about details for each value)
IlluminantUnit	Illuminance units in the standalone measurement 0 : lux, 1 : foot-candela
ExposureTime	The measurement time in standalone measurement 0 : FAST, 1 : SLOW, 2 : AUTO, 3 : SUPER FAST
UserCalibCh	The user calibration CH in standalone measurement Setting range : 0 to 10
TargetNo	The target data No. in standalone measurement Setting range : 1 to 20
UserCustom	The items which display in custom color mode Setting range : 0 to 33(Refer to CL_CUSTOMDATA_ITEM about each value)
MeasMode	Measurement mode 0: Single, 1:Mean, 2:Multiple
TimerConf	Delay time in timer configuratoins Setting range: 0 to 999 [sec]
UserWavelength	Arbitary wavelengths in custom color mode Setting range: 360 to 780 [nm]

Explanation :
This structure is used by CLSetMeasSetting() or CLGetMeasSetting() when treating the measurement condition in standalone measurement.

CL_TIMERCONF

Brief :
The structure of timer configuration

Syntax :

```
typedef struct     tCL_TIMERCONF
{
    uint16_km              DelaySec;
    uint16_km              Interval;
    uint16_km              MeasNum;
}
CL_TIMERCONF;
```

Variable :

Parameter	Explanation
DelaySec	Delay time Setting range: 0 to 999 [sec]
~~Interval~~	Disabled (ver.1.2)
~~MeasNum~~	Disabled (ver.1.2)

CL_USERWAVELENGTH

Brief :
The structure of arbitary wavelengths in custom color mode

Syntax :

```
typedef struct     tCL_USERWAVELENGTH
{
    uint16_km              lambda[4];
}
CL_USERWAVELENGTH;
```

Variable :

Variable	Explanation
lambda	The arbitary wavelengths in custom color mode 1~4 [CL_CUSTOM_EE_AB1 to CL_CUSTOM_EE_AB4] Setting range: 360 to 780 [nm] Pitch : 1 [nm]

CL_DATETIME・CL_DEVDATETIME

Brief :
The structure of information on date and time

Syntax :

```
typedef struct tCL_DEVDATETIME
{
```

```
    uint32_km      Year;
    uint32_km      Month;
    uint32_km      Day;
    uint32_km      Hour;
    uint32_km      Minute;
    uint32_km      Second;
}
CL_DEVDATETIME, CL_DATETIME;
```

Variable：

Variable	Explanation
Year	The year value Setting range：2011 to 2100
Month	The month value Setting range：1 to 12
Day	The a day value Setting range：1 to 31 (*1)
Hour	The hour value Setting range：0 to 23
Minute	The minute value Setting range：0 to 59
Second	The seconds value Setting range：0 to 59

Explanation：
This structure is used when treating the date and time in the instrument by CLSetSystemSetting() or CLGetSystemSetting(), or when treating the date and time on the various data (e.g.: target data etc.).

(*1)
If entering the date which is not exist even if the date value is in setting range (e.g. 2013/2/29), each functions returns an error code.

CL_SYSTEMSETTING

Brief :
The structure of system setting for the instrument

Syntax :

```
typedef struct tCL_SYSTEMSETTING
{
    CL_DATETIME             Datetime;
    CL_DISPLAYTYPE          Display;
    CL_BEEP                 Beep;
    CL_LANGCODE             Lang;
    CL_TYPE_DATEFORMAT      DateFormat;
    CL_USERCAL_LIMIT        UserCalLimit;
    CL_AUTOPOWEROFF         AutoPowerOff;
    CL_PCALNOTIFY           PCalNotify;
}
CL_SYSTEMSETTING;
```

Variable :

Parameter	Explanation
Datetime	The date and time setting for the instrument Available date : 2011/01/01 0:00:00 to 2100/12/31 23:59:59
Display	Orientation of the display 0 : Normal, 1 : Invert
Beep	The buzzer setting for the instrument 0 : buzzer OFF, 1 : buzzer ON
Lang	The language setting for the instrument 0:English, 1:Japanese, 2:Chinese
DateFormat	The date and time format for the instrument 0:yyyy/mm/dd, 1:mm/dd/yyyy, 2:dd/mm/yyyy
UserCalLimit	Zero calibration expiry 0:3H, 1:6H, 2:12H, 3:24H, 4:No warning
AutoPowerOff	Auto power off setting 0 : OFF, 1 : ON
PCalNotify	Periodic calibration warning message 0 : Not display, 1 : Display

Explanation :
This structure is used by CLSetSystemSetting() or CLGetSysytemSetting() when treating the system setting for the instrument.

CL_DEVID

Brief :
The structure of the instrument information

Syntax :

```
typedef struct tCL_DEVID
{
    CL_PRODUCT_CODE         Product[4];
    CL_FACTORY_CODE         Factory;
    CL_SERIAL_CODE          Serial;
    VERSION                 Firmware[RPG_FIRM_NO];
    CL_DATETIME             FactoryDate;
    CL_VARIATION_CODE       Variation;
}
CL_DEVID;
```

Variable :

Variable	Explanation
Product	The product code At the CL-500A, “A53C” is stored.
Factory	The factory code At the CL-500A, “A” is stored.
Serial	The serial No. of instrument
Firmware	The information of the firmware version
FactoryDate	The factory calibration date
Variation	The variation code At the CL-500A, “100” is stored

Explanation :
This structure is used when getting the information of the instrument by CLGetDeviceID().

CL_KEYINFO

Brief :
The structure of key status

Syntax :

```
typedef struct tCL_KEYINFO
{
    CL_KEYSTATUS MeasKey;
    CL_KEYSTATUS UpKey;
    CL_KEYSTATUS DownKey;
    CL_KEYSTATUS BackKey;
    CL_KEYSTATUS EnterKey;
}
CL_KEYINFO;
```

Variable :

Parameter	Explanation
MeasKey	The status of the Measuring button
UpKey	The status of the Up key
DownKey	The status of the Down key
BackKey	The status of the Back key
EnterKey	The status of the Enter key

Explanation :
This structure is used when getting key status of the instrument by CLGetButtonStatus().

When the key is pressed, “CL_KEY_STATE_OFF (=1)” is stored in the variable.
When the key is not pressed, “CL_KEY_STATE_OFF (=0)” is stored in the variable.

CL_SDKVERSION

Brief :
The structure of version information

Syntax :

```
typedef struct tCL_SDK_VERSION
{
    VERSION Main;
    VERSION Calib;
    VERSION Com;
    VERSION Color;
}
CL_SDKVERSION;
```

Variable :

Parameter	Explanation
Main	Version information of the interface library
Calib	Version information of the calibration library
Com	Version information of the device communication library
Color	Version information of the color calculating library

Explanation :
This structure is used when getting the version information for each DLL by CLGetSDKVersion().

4.4 The CL-SDK API

The descriptions of the CL-SDK functions are written in the following format.

Function :
Explanation of the processing which performs by function

Syntax :

Parameter :
Explanation of the each parameter for function

Return Value :
Explanation of the return values when using function.
There's 3 types as the return value as below.

Type	Explanation
Normal	Returns when the processing is successful.
Error	Returns when the processing is failed.
Warning	Returns when the information is required for user though the processing is successful.

Explanation :
More detailed explanation and the precaution when using the function

CLOpenDevice()

Function :
Gets the object handle of the instrument which is connected on the PC

Syntax :
ER_CODE KMAPI CLOpenDevice(DEVICE_HANDLE* hDevice)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to get the object handle
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER_OPENDEVICE	Error	13	Failed to get the object handle

Explanation :
This function gets the object handle of the instrument which is connected on the PC.

If multiple instruments are connected, please execute this function by the number of units which are connected.
When the multiple instruments are connected, perform this function by a number of times which is connected on the PC

Refer to "§ 3.3 The control of multiple instrument" about how to control the multiple instrument.

This function gets only the object handle.
To control the instrument from PC, a remote mode is required to be ON.
Please set a remote mode by CLSetRemoteMode().

CLCloseDevice()

Function :
Releases the object handle of the instrument which is stored in the CL-SDK

Syntax :
ER_CODE KMAPI CLCloseDevice(DEVICE_HANDLE hDevice)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle to be released

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to release the object handle
ER_HANDLENULL	Error	10	The object handle is not exist

Explanation :
This function releases the object handle.

The object handle of the CL-500A which stored in the CL-SDK is released. unless executing this function, the instrument can be controlled by the application

[Note]
This function only releases the object handle. To use by standalone, set the remote mode OFF by CLSetRemoteMode() before executing this function.

CLSetRemoteMode()

Function :
Configure a remote mode ON/OFF

Syntax :
ER_CODE KMAPI CLSetRemoteMode(DEVICE_HANDLE hDevice, CL_REMOTEMODE Mode)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
Mode	I	The configuration of remote mode 0 : a remote mode OFF 1 : a remote mode ON

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to configure a remote mode
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER00301	Error	301	Parameter Mode is NULL value
ER00302	Error	302	Configuration value is invalid value
ER00304	Error	304	Failed to set a remote mode
ER00305	Error	305	Failed to get the calibration data (*1)
ER00306	Error	306	Failed to get the rank data (*1)
ER00307	Error	307	Failed to get the measurement time of the instrument
ER00308	Error	308	Failed to set a remote mode OFF
ER00309	Error	309	Failed to get the user calibration coefficient (*1)

Explanation :
This function configures ON/OFF for a remote mode.
If control the instrument by the PC, configure a remote mode ON surely.
When a remote mode is ON, the operation by standalone is not available.
When the remote mode is ON, the LCD display on the instrument shows "Communicating...".

(*1)
When a remote mode is set ON, this function attempts to get a calibration data etc. for a calculation from the instrument.
If the error occurs in getting that data, this function will return "ER00305", "ER00306" or "ER00309", as the return value.

CLGetRemoteMode()

Function :
Gets the remote mode ON/OFF

Syntax :
ER_CODE KMAPI CLGetRemoteMode(DEVICE_HANDLE hDevice, CL_REMOTEMODE* Mode)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
Mode	O	The buffer to store a remote mode setting

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to get the remote mode ON/OFF
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER00301	Error	301	Parameter Mode is NULL value
ER00303	Error	303	Failed to get a remote mode setting

Explanation :
This function gets the remote mode ON/OFF

CLDoMeasurement()

Function :
Performs a measurement

Syntax :
ER_CODE KMAPI CLDoMeasurement(DEVICE_HANDLE hDevice, int32_km *Time)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
Time	O	The buffer to store a measurement time Unit:100μsec ex.) When the value of "Time" is 2500 2500×100 = 250000 μs = 0.25 sec

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to perform a measurement
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER01301	Error	1301	A zero calibration is not performed
ER01302	Error	1302	The remote mode is OFF
ER01303	Error	1303	Parameter pTime is NULL value
ER01304	Error	1304	Failed to get a calibration status
ER01305	Error	1305	Failed to perform a measurement
ER01306	Error	1306	A zero calibration is not performed
ER01307	Error	1307	The error was occurred in pre measurement
WARNING	Warning	1	It has passed a certain time from the last zero calibration. Perform a zero calibration.

Explanation :
This function performs a measurement.
"Time" stores a rough measurement time by instrument. Confirm as the estimated time until completing a measurement.

If measurement time is AUTO, the measurement times is different by the light source (maximum: about 27 second). Therefore the obtained value of "Time" is different by the light source.

If measurement time is FAST, SLOW, and SUPRE_FAST, the measurement time is fixed.
Therefore this function gets measurement time for each as below.
FAST: about 0.5sec, SLOW: about 2.5sec, SUPER_FAST:about 0.2sec

CLDoManualMeasurement()

Function :
Perform a measurement by specifying exposure time and cumulative number.

Syntax :
ER_CODE KMAPI CLDoManualMeasurement(DEVICE_HANDLE hDevice, int32_km eTime, int32_km cNumber)
Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
eTime	I	Exposure time [x 100us]
cNumber	I	Cumulative number

Retrun Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to perform a measurement
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER01321	Error	1321	A zero calibration is not performed
ER01322	Error	1322	The remote mode is OFF
ER01324	Error	1324	Failed to get a calibration status
ER01325	Error	1325	Failed to perform a measurement
	Error	1326	The firmware of the instrument does not support this function (require Ver.1.20)
ER01328	Error	1328	The value of eTime (exposure time) is invalid
ER01329	Error	1329	The value of cNumber (cumulative number) is invalid
ER01330	Error	1330	Combination of eTime and cNumber is invalid
ER01331	Error	1331	Failed to set exposure time and cumulative number
WARNING	Warning	1	It has passed a certain time from the last zero calibration. Perform a zero calibration.

Explanation :
Specify exposure time and cumulative number when you use this function.
Choose combination of exposure time and cumulative number from the below chart.
(The ranges of illuminance are typical values when you use illuminant A)

combination chart of exposure time and cumulative number

*range of illuminance [lx]	exposure time [x 100us]	cumulative number	measurement time [ms]
10000 ~ 100000	40	100	400
1000 ~ 10000	400	10	400
100 ~ 1000	4000	1	400
10 ~ 20	20000	1	2000
under 10	50000	1	5000
	50000	2	10000
	50000	4	20000

CLPollingMeasure()

Function :
Confirms the implementation status of measurement

Syntax :
ER_CODE KMAPI CLPollingMeasure((DEVICE_HANDLE hDevice, CL_MEASSTATUS *pStatus)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
pStatus	O	The buffer to store the measurement status

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to get the measurement status
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER_ALLMEASUREING	Error	16	The CLDoMeasureAll () is performing
ER01201	Error	1201	Parameter “pStatus” is NULL value
ER01202	Error	1202	The remote mode is OFF
ER01203	Error	1203	Failed to a measurement
ER01204	Error	1204	Exceeds measurement range by the CL-500A (*1)
ER01205	Error	1205	Failed to get the implementation of measurement status
ER01206	Error	1206	Failed to a measurement (*2)

Explanation :
This function confirms the implementation status of measurement.
After the obtained value of “status” becomes “CL_MEAS_FINISH”, get the colorimetric data by CLGetMeasData() or select the rank by CLSortOutRank().
This function can get “CL_MEAS_FINISH” only in first time, after completing a measurement. Once this function gets “CL_MEAS_FINISH” as "status" value, the obtained “status” value becomes “CL_MEAS_FREE” after first times.
(*1)
This function returns “ER01204” as return value if the illuminance exceeds the level that can be measured by the CL-500A.
Attempt a measurement again, increase the distance between the CL-500A and the light source being measured, or use a ND filter to reduce the intensity of the light.
(*2)
This function returns “ER01206” as a return value if the brightness of the light source becomes brighter than at starting a measurement.
In measurement, keep the brightness of the light source constant until completing a measurement. Futhermore, if the brightness is over than the maximum limit of CL-500A, this function returns "ER01206". Then, you should increase the distance between the CL-500A and the light source being measured, or use a ND filter to reduce the intensity of the light.
(*2)
If the brightness of the light source is over than the limit of the range you specified in manual measurement, this function returns "ER01206". On that occasion, perform a measurement in shorter exposure time.

CLStopMeasurement()

Function :
Stops a measurement

Syntax :
ER_CODE KMAPI CLStopMeasurement(DEVICE_HANDLE hDevice)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to stop the measurement
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER01401	Error	1401	The remote mode is OFF
ER01402	Error	1402	Failed to stop a measurement

Explanation :
This function stops a measurement.
Use this function if would like to stop the measurement because of the long measurement time such as a measurement time is AUTO mode.

This function returns “SUCCESS” as a return value when a measurement is not performed

CLDoMeasurementAll()

Function :
Performs a measurement with all instrument

Syntax :
ER_CODE KMAPI CLDoMeasurementAll(int32_km *pTime)

Parameter :

Parameter	I/O	Explanation
pTime	O	The buffer to store a measurement time Unit : 100μsec ex.) When the value of "Time" is 2500 2500×100 = 250000 μs = 0.25 sec

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to perform a measurement by all instruments
ER10801	Error	10801	The object handle is not exist
ER10802	Error	10802	The instrument which a zero calibration is not performed exists
ER10803	Error	10803	The instrument which a remote mode is OFF exists
ER10804	Error	10804	Failed to get a calibration status
ER10805	Error	10805	Failed to perform a measurement
ER10806	Error	10806	Failed to get a measurement time
ER10807	Error	10807	The instrument which a remote mode is OFF exists

Explanation :
This function performs for all the instruments which are connected on the PC.
The instrument to be performed a measurement cannot specify.

The time which this function gets is stored the longest measurement time in all the instruments after determining the measurement time of all the instruments.
Set all the instruments a remote mode ON.
If some instrument which is not performed a zero calibration is this function returns an error code.
Perform a zero calibration before performing a measurement.

CLDoManualMeasurementAll()

Function :
Perform a measurement with all instruments by specifying exposure time and cumulative number.

Syntax :
ER_CODE KMAPI CLDoManualMeasurementAll(int32_km eTime, int32_km cNumber)

Parameter :

Parameter	I/O	Explanation
eTime	I	Exposure time [x 100us]
cNumber	I	Cumulative number

Retrun Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to perform a measurement
ER10821	Error	10821	The object handle is not exist
ER10822	Error	10822	The instrument which a zero calibration is not performed exists
ER10823	Error	10823	The instrument which a remote mode is OFF exists
ER10824	Error	10824	Failed to get a calibration status
ER10825	Error	10825	Failed to perform a measurement
ER10826	Error	10826	The firmware of the instrument not supporting this function exists (require Ver.1.20)
ER10827	Error	10827	The instrument which a remote mode is OFF exists
ER10828	Error	10828	The value of eTime (exposure time) is invalid
ER10829	Error	10829	The value of cNumber (cumulative number) is invalid
ER10830	Error	10830	Combination of eTime and cNumber is invalid
ER10831	Error	10831	Failed to set exposure time and cumulative number

Explanation :
This function performs for all the instruments which are connected on the PC.
The instrument to be performed a measurement cannot specify.

The time which this function gets is stored the longest measurement time in all the instruments after determining the measurement time of all the instruments.
Set all the instruments a remote mode ON.
If some instrument which is not performed a zero calibration is this function returns an error code.
Perform a zero calibration before performing a measurement.

Specify exposure time and cumulative number when you use this function.
Choose combination of exposure time and cumulative number from the below chart.
(The ranges of illuminance are typical values when you use illuminant A)

combination chart of exposure time and cumulative number

*range of illuminance [lx]	exposure time [x 100us]	cumulative number	measurement time [ms]
10000 ~ 100000	40	100	400
1000 ~ 10000	400	10	400
100 ~ 1000	4000	1	400
10 ~ 20	20000	1	2000
under 10	50000	1	5000
	50000	2	10000
	50000	4	20000

CLPollingMeasureAll()

Function :
Confirms the implementation status of a measurement by CLDoMeasurementAll()

Syntax :
ER_CODE KMAPI CLPollingMeasureAll(CL_MEASSTATUS *pStatus)

Parameter :

Parameter	I/O	Explanation
pStatus	I	The object handle to be control

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to a measurement status
ER10901	Error	10901	The number of instrument currently is different from the number of the instrument when starting a measurement.
ER10902	Error	10902	The instrument which a remote mode is OFF exists
ER10903	Error	10903	Exceeds measurement range by the CL-500A (*1)
ER10904	Error	10904	Failed to get the status of a measurement

Explanation :
This function confirms the implementation status of measurement by CLDoMeasurementAll().
As long as the instrument which is performing a measurement exists, the status is during the measurement (“CL_MEAS_BUSY”).
The status becomes “CL_MEAS_FINISH” after completing the measurement by all instruments.
After the obtained value of “status” becomes “CL_MEAS_FINISH”, get the colorimetric data by CLGetMeasData() or select the rank by CLSortOutRank().
The implementation status for individual instruments cannot get during executing the CLDoMeasurementAll() (CL_PollingMeasure() is not available).

(*1)
If the brightness of measurement source becomes brighter at the start of measurement, the measurement will be an error. The CL-SDK returns an error code “ER10903” as a return value. In measurement, keep the brightness of the light source constant until completing a measurement. In addition, this function returns “ER10903” as return value if the illuminance exceeds the level that can be measured by the CL-500A.
Attempt a measurement again, increase the distance between the CL-500A and the light source being measured, or use a ND filter to reduce the intensity of the light.
(*2)
If the brightness of the light source is over than the limit of the range you specified in manual measurement, this function returns "ER10903". On that occasion, perform a measurement in shorter exposure time.

CLStopMeasurementAll()

Function :
Stops a measurement by CLDoMeasurementAll()

Syntax :
ER_CODE KMAPI CLStopMeasurementAll()

Parameter :

None

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to stop a measurement by all instruments

Explanation :
This function stops a measurement by CLDoMeasurementAll()
Use this function if would like to stop the measurement because of the long measurement time such as a measurement time is AUTO mode.

Use this function, if would like to stop measurement, in case of the measurement by CLDoMeasurementAll().
CLStopMeasurement() cannot stop a measurement by CLDoMeasurementAll()
CLStopMeasurement() function could not to stop a measurement by CLDoMeasurementAll()

CLDoCalbiration()

Function :
Performs a zero calibration

Syntax :
ER_CODE KMAPI CLDoCalibration(DEVICE_HANDLE hDevice)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to perform a zero calibration
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER11101	Error	11101	Failed to a zero calibration
ER11102	Error	11102	The remote mode is OFF

Explanation :
This function performs a zero calibration.
Perform a zero calibration with equipped the calibration cap to the receptor window of the instrument.
If a zero calibration is failed, please perform a zero calibration again. A zero calibration takes 27 sec.
Confirm by CLPollingCalibration() whether a zero calibration is completed.

Please perform a zero calibration at least one time in a day to keep a measurement value with accuracy.

CLPollingCalibration()

Function :
Gets the implementation status of a zero calibration

Syntax :
ER_CODE KMAPI CLPollingCaibration((DEVICE_HANDLE hDevice, CL_CALIBMEMSSTATUS *pStatus)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
pStatus	O	The buffer to store the status of a zero calibration

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to get a calibration status
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER11201	Error	11201	Parameter pStatus is NULL value
ER11202	Error	11202	Failed to a zero calibration
ER11203	Error	11203	The abnormality of the device is occurred. (*1)
ER11204	Error	11204	The remote mode is OFF

Explanation :
This function gets the implementation status of a zero calibration.
A zero calibration will take about 27 sec.
When the value of “status” is “CL_CALIBMEAS_FINISH”, a zero calibration is completed.
This function can get “CL_CALIBMEAS_FINISH” only in first time, after completing a zero calibration.
Once this function gets “CL_MEAS_FINISH” as ″status″ value, the obtained “status” value becomes “CL_CALIBMEAS_FREE” after first times.

(*1)
So the abnormality of the device is occurred, the repair of the device is required. Please contact the nearest service facility.

CLGetCalibrateStatus()

Function :
Gets a status whether a zero calibration have performed

Syntax :
ER_CODE KMAPI CLGetCalibrateStatus(DEVICE_HANDLE hDevice, CL_CALIBSTATUS* Status)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
Status	O	The buffer to store the calibration status

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to get the calibration status
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER01101	Error	1101	Parameter Status is NULL value
ER01102	Error	1102	The remote mode is OFF
ER01103	Error	1103	Failed to get the calibration status

Explanation :
This function gets a status whether a zero calibration have performed.
If a zero calibration is not performed, the instrument cannot measure. Please perform a zero calibration.

When the instrument is required a zero calibration, the instrument can perform a measurement. Since it have passed a certain time from the last calibration, perform a zero calibration to measurement
A measurement is able to perform even if a zero calibration is recommended.
However, please perform a zero calibration to keep a measurement value with accuracy. Please perform a zero calibration at least one time in a day.

CLGetMeasData()

Function :
Gets the each colorimetric data at the latest measurement

Syntax :
ER_CODE KMAPI CLGetMeasData(DEVICE_HANDLE hDevice, CL_COLORSPACE Type, CL_MEASDATA *pColor)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
Type	I	The colorimetric type to be obtained
pColor	O	The buffer to store a colorimetric data

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to calculate the colorimetric data
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER03702	Error	3702	Failed to calculate colorimetric value
ER03703	Error	3703	A measurement is not performed
ER03704	Error	3704	The remote mode is OFF
ER03705	Error	3705	Parameter pColor is NULL value
WARNING	Warning	1	Illuminance values is low value than the assured value (*1)
WARNING	Warning	1	The item which could not calculated is exist (*2)
WARNING	Warning	1	The brightness is over than the proper range of the exposure time you specified. The measurements are not guranteed. warning code: 6
WARNING	Warning	1	The brightness is under than the proper range of the exposure time you specified. The measurements are not guranteed. warning code: 6

Explanation :
This function gets each colorimetric data at the latest measurement.
The colorimetric data which can get is only the latest measurement data.
If the measurement data at each measurement is required, get the measurement data before performing the next measurement.
The using user calibration coefficient is the calibration coefficient which is configured as the calibration CH in the instrument.

(*1)
The measurement data is less then certain value, this function returns the low illuminance warning.

Though a measurement data is outputted, the measurement value is not ensured as the CL-500A.
The illuminance value to be low illuminance warning is different by measurement time.
The threshold for the low illuminance warning at each measurement time is below.
FAST mode : 50[lx], SLOW mode:10[lx], SUPERFAST mode:100[lx]

(*2)
By measurement source, the measurement data might not be able to calculate Tcp and so on.
In this case, the value “CL_INCOMPUTABLE_VALUE (-11000)” is stored as measurement data.

[Note]
When the measurement data is a complementary wavelength, the measurement value is minus value.

The number of significant figures of measurement data is 4 digits.

CLGetColorDifference()

Function :
Gets each color difference data at the latest measurement

Syntax :
ER_CODE KMAPI CLGetColorDifference(DEVICE_HANDLE hDevice, CL_COLORSPACE Type, CL_DIFFDATA * pDiff)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
Type	I	The color difference type to be obtained
pDiff	O	The buffer to store the color difference

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to calculate color difference
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER04701	Error	4701	Failed to get the target data
ER04702	Error	4702	A measurement is not performed yet
ER04703	Error	4703	Failed to set a target data
ER04704	Error	4704	Failed to set a measurement data
ER04705	Error	4705	Failed to calculate the color difference
ER04706	Error	4706	The remote mode is OFF
ER04707	Error	4707	Parameter pDiff is NULL value

Explanation :
This function gets each color difference data at the latest measurement.
The color difference data which can get is only the latest measurement data.
If the measurement data at each measurement is required, get the measurement data before performing the next measurement.

The target data which is used by this function is the target data which is used in color difference measurement by standalone measurement.
For use this function, register a target data to the device, and configure the target data to be used in advance.

[Note]
The difference of the dominant wavelength is calculated by subtracting target data from measurement data.
When One is the dominant wavelength and the others are the complementary wavelength, the difference is calculated by subtracting target data from measurement data similarly.
ex.) When the measurement data is complementary wavelength (562nm), and the target data is a dominant wavelength (568nm)
⊿Dominant wavelength = -562 - 568 = -1130 [nm]

CLSortOutRank()

Function :
Selects rank for the measurement data

Syntax :
ER_CODE KMAPI CLSortOutRank(DEVICE_HANDLE hDevice, CL_RANK* Rank);

Parameter :

Parameter	I/O	説
hDevice	I	The object handle of the instrument to be controlled
Rank	I	The buffer to store the rank information

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to select rank
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER08101	Error	8101	Failed to get a measurement data
ER08102	Error	8102	Failed to select rank
ER08103	Error	8103	The remote mode is OFF
ER08104	Error	8104	Parameter Rank is NULL value
ER08105	Error	8105	The rank list is not exist
WARNING	Warning	1	The corresponding rank is not exist

Explanation :
This function selects the rank for measurement data.
The rank data which use in selecting the rank by the CL-SDK is the rank data which is registered in the instrument.
Therefore register the rank data by CLSetRankData() before selecting rank.
Please register the rank data by CLSetRankData() before selecting rank.

In addition, this function selects the rank in ascending order.
Therefore if the measurement data is located on the point which the multiple areas overlap, the CL-SDK sorts the measurement data out as the rank data which is registered in the small data number.

When the measurement data is not corresponded to any the rank data, the CL-SDK returns the warning code as a return value.
The value "N/A", which means that is not corresponded to any rank data, is stored in the parameter.

CLSetRankData()

Function :
Registers a rank data which is used when selecting rank

Syntax :
ER_CODE KMAPI CLSetRankData(DEVICE_HANDLE hDevice, int32_km DataNo, const CL_RANK_DATA RankData)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
DataNo	I	The Data No. which register a rank data Setting range : 1 to 20
RankData	I	The rank data to be registered

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to register the rank data
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER06601	Error	6601	Parameter RankData is NULL value
ER06602	Error	6602	The remote mode is OFF
ER06603	Error	6603	The specified data No. is invalid value
ER06604	Error	6604	Failed to register the rank data
ER06605	Error	6605	The rank area to register is not correct (*1)
ER06606	Error	6606	Failed to convert to the xy data

Explanation :
This function registers a rank data which is used when selecting rank to the instrument.
The rank data is registered as the CL_RANK_DATA structure.

The rank area is combined the point in order which is stored in the array, and is checked whether rank area is divided. If the rank area is divided, this function returns an error code by the judgment that the rank area is not proper.
Refer to " §5.1 The rank area setting" about the details.

If the rank data is exist in the specified No., the rank data is registered overwrite.
Execute this function after confirm whether rank data is exists in the specified number.

(*1)
This function will return an error code, when the rank area to be registered is divided.

In case of one-byte character, the rank name is able to register 40 characters in maximum.
In case of multi-byte character, the rank name is able to register 20 characters in maximum.
And the rank name is able to mix one-byte character and multi-byte character.
Since the instrument can display 15 characters with one-byte character in maximum, the instrument displays 15 characters when the length of the rank name is over 15 characters.
In case of the character which the instrument cannot display, the instrument displays "・" as character.

CLGetRankData()

Function :
Gets the rank data which is registered in the instrument

Syntax :
ER_CODE KMAPI CLGetRankData(DEVICE_HANDLE hDevice, int32_km DataNo, CL_RANK_DATA* RankData)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
DataNo	I	The rank No. of the rank data to be obtained Setting range : 1 to 20
RankData	I/O	The buffer to store rank data

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to get the rank data
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER08201	Error	8201	Failed to get the rank data
ER08202	Error	8202	The remote mode is OFF
ER08203	Error	8203	Parameter RankData is NULL value
ER08204	Error	8204	The rank data is not exist
ER08205	Error	8205	The specified data No. is out of range

Explanation :
This function gets the rank data which is registered in the instrument.
This function gets the rank data of the specified No.
To get all the rank data which is registered in the instrument, please get the rank data in specifying the value of from 1 to 20 repeatedly.

CLDeleteRankData()

Function :
Deletes the rank data which is registered in the instrument

Syntax :
ER_CODE KMAPI CLGetRankData(DEVICE_HANDLE hDevice, int32_km DataNo)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
DataNo	I	The rank data No. to deleted -1:Delete all the rank data 1 to 20 : Delete the rank data of the specified No.

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to delete the rank data
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER06701	Error	6701	The specified data No. is out of range
ER06702	Error	6702	Failed to delete the rank data
ER06703	Error	6703	Failed to delete the rank data
ER06704	Error	6704	The remote mode is OFF

Explanation :
This function deletes the rank data in the instrument.

This function deletes all the rank data by specifying “-1” for “DataNo”.

CLSetTargetData()

Function :
Registers a target data to the instrument

Syntax :
ER_CODE KMAPI CLSetTargetData(DEVICE_HANDLE hDevice, int32_km DataNo, const CL_TARGET_DATA* pTarget)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
DataNo	I	The data No. of the target data to be registered Setting range : 1 to 20
pTarget	I	The target data to be registered

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to register the target data
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER05401	Error	5401	Parameter pData is NULL value
ER05402	Error	5402	The specified data No. is out of range
ER05404	Error	5404	The remote mode is OFF
ER05405	Error	5405	The date to be registered is not correct
ER05406	Error	5406	The data name to be registered is not correct
ER05407	Error	5407	Failed to register the target data

Explanation :
This function registers the target data to the instrument.

[Note]
If this function is executed, the target data is registered for the specified data number.
This function does not confirm whether target data already exists in the specified data number. Therefore if the overwrite confirmation is required, confirm on the overwriting by the application.

CLGetTargetData()

Function :
Gets the target data which is registered in the instrument

Syntax :
ER_CODE KMAPI CLGetTargetData(DEVICE_HANDLE hDevice, int32_km DataNo, CL_TARGET_DATA* pTarget)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
DataNo	I	The data No. of the target data to be obtained Setting range : 1 to 20
Data	I/O	The buffer to store the target data

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to get the target data
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER05501	Error	5501	Parameter pStatus is NULL value
ER05502	Error	5502	The specified data No. is out of range
ER05503	Error	5503	The remote mode is OFF
ER05504	Error	5504	Failed to get a target data
ER05505	Error	5505	Failed to get target data information
ER05506	Error	5506	The target data is not exist in the specified No.

Explanation :
This function gets the target data which is registered in the instrument.

CLDeleteTargetData()

Function :
Deletes the target data which is registered in the instrument

Syntax :
ER_CODE KMAPI CLDeleteTargetData(DEVICE_HANDLE hDevice, int32_km DataNo)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
DataNo	I	The target data No. to delete -1 : Delete all the target data 1 to 20 : Delete the target data of the specified No.

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to delete the target data
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER05601	Error	5601	The specified data No. is out of range
ER05602	Error	5602	Failed to delete the target data
ER05603	Error	5603	The remote mode is OFF

Explanation :
This function deletes the target data which is registered in the instrument.

This function deletes all the target data by specifying “-1” for “DataNo”.

If the target data in the specified No. does not exist, this function returns an error code “ER05602” as return value.

CLGetDeviceStoredDataNum()

Function :
Gets the number of stored data in the instrument

Syntax :
ER_CODE KMAPI CLGetDeviceStoredDataNum(DEVICE_HANDLE hDevice, int32_km* Num)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
Num	O	The buffer to store the number of measurement data

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to get the number of the stored data
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER05901	Error	5901	Parameter Num is NULL value
ER05902	Error	5902	The remote mode is OFF
ER05903	Error	5903	Failed to get the number of measurement data

Explanation :
This function gets the number of data which is stored in the instrument.
The instrument can store 100 measurement data in maximum.
If the stored data does not exist in the instrument, this function returns '0' as return value.
Please reserve the buffer for the stored data which is obtained by CLGetDeviceStoredData() based on the data number which is obtained by this function.

CLGetDeviceStoredData()

Function :
Gets all the stored data in the instrument

Syntax :
ER_CODE KMAPI CLGetDeviceStoredData(DEVICE_HANDLE hDevice, CL_LISTDATA *pData, CL_COLORSPACE Type, int32_km Length)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
pData	O	The buffer to store the measurement data in the instrument
Type	I	The colorimetric type to be obtained
Length	I	The buffer size (*1)

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to gets the stored data
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER05801	Error	5801	Parameter pData is NULL value
ER05802	Error	5802	The remote mode is OFF
ER05803	Error	5803	Failed to get the number of stored data
ER05804	Error	5804	The buffer size is not enough
ER05805	Error	5805	Failed to get the stored data in the instrument
ER05806	Error	5806	Failed to get the stored data in the instrument
ER05807	Error	5807	Failed to set measurement data
ER05808	Error	5808	Failed to calculate the colorimetric data
ER05809	Error	5809	The stored data in the instrument is not exist

Explanation :
This function gets all the stored data in the instrument at one time.
To confirm the number of stored data, please get the number of stored data by CLGetDeviceStoredDataNum().

(*1)
Specify the number of data to get as a buffer size.
ex.) If the number of data to get is 25 data, specify 25 to the Length.

(*2)
By measurement source, the measurement data might not be able to calculate Tcp and so on.
In this case, the value "CL_INCOMPUTABLE_VALUE (-11000)" is stored as measurement data.

[Note]
The instrument can store 100 measurement data in maximum.

If the number of data which stores in the instrument is large, a processing time to get the stored data will take a lot. (A processing time will take about 7 sec in case of 100 data)

CLDeleteDeviceStoredData()

Function :
Deletes the stored data in the instrument

Syntax :
ER_CODE KMAPI CLDeleteDeviceStoredData(DEVICE_HANDLE hDevice, int32_km DataNo)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
DataNo	I	The stored data No. to be deleted -1 : Delete all the stored data 1 to 100 : Delete the stored data of the specified No. (*1)

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to delete the stored data
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER08901	Error	8901	The specified data No. is out of range
ER08902	Error	8902	Failed to delete the stored data
ER08903	Error	8903	Failed to delete the stored data
ER08904	Error	8904	The remote mode is OFF
ER08905	Error	8905	Failed to get the number of stored data in the instrument
ER08906	Error	8906	The data is not exist in the specified No.

Explanation :
This function deletes the stored data in the instrument

(*1)
If the stored data in the specified No. does not exist, this function returns an error code "ER08906" as return value.

[Note]
When deleting the stored data, the data No. is changed for causing the data No. below the deleted data to shift upward. Please be careful on specifying the data No. when deleting the stored data continuously.

CLSetUserCalibraitonData()

Function :
Registers a user calibration coefficient to the instrument

Syntax :
ER_CODE KMAPI CLSetUserCalibraitonData(DEVICE_HANDLE hDevice, int32_km DataNo, const CL_USERCALIB_DATA* pCoef)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
DataNo	I	The user calibration CH to be registered Setting range : 1 to 10
pCoef	I/O	The user calibration coefficient to be registered

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to register the user calibration coefficient
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER02101	Error	2101	The specified data No. is out of range
ER02102	Error	2102	Some user calibration coefficient to be registered is not correct.
ER02103	Error	2103	The remote mode is OFF
ER02104	Error	2104	The date to be registered is not correct
ER02105	Error	2105	Parameter pCoef is NULL value
ER02105	Error	2106	The data name to be registered is not correct
ER02107	Error	2107	Failed to register the user calibration coefficient
ER02108	Error	2108	Failed to update the user calibration coefficient

Explanation :
This function registers a user calibration coefficient to the instrument.

For a user calibration CH which is already registered a user calibration data, this function overwrites.

[Note]
This function registers a user calibration data to the specified number.
This function will not confirm whether a user calibration data exist in the specified number.
If require the confirmation of overwrite, please confirm by the application.

CLGetUserCalibraitonData()

Function :
Gets the user calibration coefficient which is registered in the instrument

Syntax :
ER_CODE KMAPI CLGetUserCalibraitonData(DEVICE_HANDLE hDevice, int32_km DataNo, CL_USERCALIB_DATA* pCoef)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
DataNo	I	User calibration CH to be obtained Setting range : 1 to 10
pCoef	O	The buffer to store the user calibration coefficient

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to get the user calibration coefficient
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER02201	Error	2201	The specified data No. is out of range
ER02202	Error	2202	Parameter pCoef is NULL value
ER02203	Error	2203	The remote mode is OFF
ER02204	Error	2204	Failed to get the user calibration coefficient
ER02205	Error	2205	Failed to confirm the data of the specified CH
ER02206	Error	2206	The data is not exist in the specified CH

Explanation :
This function gets the user calibration coefficient which is registered in the instrument.

CLDeleteUserCalibrationData()

Function :
Deletes a user calibration coefficient which is registered in the instrument

Syntax :
ER_CODE KMAPI CLDeleteUserCalibrationData(DEVICE_HANDLE hDevice, int32_km DataNo)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
DataNo	I	User calibration CH to be deleted -1 : Delete all the data 1 to 10 : Delete the data of the specified No.

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to delete a user calibration coefficient
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER02301	Error	2301	The specified data No. is out of range
ER02302	Error	2302	The remote mode is OFF
ER02303	Error	2303	Failed to delete the user calibration coefficient
ER02304	Error	2304	Failed to update the user calibration coefficient

Explanation :
This function deletes a user calibration coefficient which is registered in the instrument.

When specifying “-1” as data No., this function deletes all the user calibration coefficients.

CLCalcUserCalibrationData()

Function :
Calculates a user calibration coefficients

Syntax :
ER_CODE KMAPI CLCalcUserCalibrationData(DEVICE_HANDLE hDevice, CL_SPC_DATA* Output, const SPC_DATA* Target, const CL_SPC_DATA* Sample)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
Output	O	The buffer to store the calculated user calibration coefficient Buffer size : 421 data
Target	I	The target data to calculate a user calibration coefficient Number of data : 421 data
Sample	I	The measurement data to calculate a user calibration coefficient Number of data : 421 data

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to calculate user calibration coefficient
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER02401	Error	2401	The calculated coefficient is out of range
ER02402	Error	2402	Parameter output is NULL value
ER02403	Error	2403	Parameter Target is NULL value
ER02404	Error	2404	Parameter sample is NULL value

Explanation :
This function calculates a user calibration coefficient.
The calculated coefficient is able to register to the instrument.

Please use CLSetUserCalibrationData(), in case of registering the user calibration coefficient.
If the value of the calculated coefficient is less than 0.001, or larger than 1000.000 this function returns an error code “ER02401” as return value.

CLSetProperty()

Function :
Configures the measurement condition when control the instrument by the CL-SDK

Syntax :
ER_CODE KMAPI CLSetProperty(DEVICE_HANDLE hDevice, CL_PROPERTIES Type, int32_km Param)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
Type	I	The property type to be configured
pParam	I	The property value to be configured

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to configure the measurement condition
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER00501	Error	501	The observer value is invalid value
ER00502	Error	502	The illuminance units value is invalid value
ER00503	Error	503	The remote mode is OFF
ER00504	Error	504	The property type is invalid value

Explanation :
This function configures the measurement condition when control the instrument by the CL-SDK.
The measurement condition for standalone measurement cannot be configured.
If the instrument is disconnected, the configuration which stores in internal of the CL-SDK is cleared.
If necessary, store the configurations by the application.

[Note]
The measurement condition which configures by this function reflects from next measurement data.
About the measurement condition for standalone measurement, please configure the measurement condition by CLSetMeasSetting().

CLGetProperty()

Function :
Gets the measurement condition when control the instrument by the CL-SDK

Syntax :
ER_CODE KMAPI CLGetProperty(DEVICE_HANDLE hDevice, CL_PROPERTIES Type, int32_km *Param)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
Type	I	The property type to be obtained
pParam	O	The buffer to store property value

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to get the measurement condition
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER00401	Error	401	Parameter Param is NULL value
ER00402	Error	402	Specified property type is invalid value
ER00403	Error	403	The remote mode is OFF

Explanation :
This function gets the measurement condition when control the instrument by the CL-SDK.
The measurement condition for standalone measurement cannot be obtained.

[Note]
About the measurement condition for standalone measurement, please get the measurement condition by CLGetMeasSetting().

CLGetButtonStatus()

Function :
Gets the key status for instrument

Syntax :
ER_CODE KMAPI CLGetButtonStatus(DEVICE_HANDLE hDevice, CL_KEYINFO *pStatus)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
pStatus	O	The buffer to store the key status

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to get the key status
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER00801	Error	801	pStatus is NULL value
ER00802	Error	802	The remote mode is OFF
ER00803	Error	803	Failed to get the key status

Explanation :
This function gets the key status for instrument.
By getting the key status, the application can perform a measurement by key trigger.

About how to measure by key trigger, refer to “§3.4 The measurement by key trigger ot the instrument”.

CLSetMeasSetting()

Function :
Configures the measurement condition in standalone measurement

Syntax :
ER_CODE KMAPI CLSetMeasSetting(DEVICE_HANDLE hDevice, CL_MEASSETTYPE Type, const CL_MEASSETTING* pSetting)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
Type	I	The measurement condition type to be configured
pSettings	I	The measurement condition to be configured

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to the measurement condition
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER_PARAM_NULL	Error	14	Parameter pSettings is NULL value
ER01001	Error	1001	Parameter pSettings is NULL value
ER01002	Error	1002	The remote mode is OFF
ER01003	Error	1003	The specified type is invalid value
ER07201	Error	7201	Display type is invalid value
ER07203	Error	7203	Failed to configure the display type
ER07301	Error	7301	The observer value to be configured is invalid
ER07303	Error	7303	Failed to configure the observer
ER07401	Error	7401	Color space is invalid value
ER07403	Error	7403	Failed to configure color space
ER07501	Error	7501	Illuminance units is invalid value
ER07503	Error	7503	Failed to configure the illuminance units
ER07601	Error	7601	Measurement time is invalid value
ER07603	Error	7603	Failed to configure the measurement time
ER07701	Error	7701	User calibration CH is invalid value
ER07703	Error	7703	The calibration coefficient is not exist in specified calibration CH
ER07704	Error	7704	Failed to configure user calibration CH
ER07705	Error	7705	Failed to update the user calibration coefficient
ER07801	Error	7801	Target data No. is invalid value
ER07803	Error	7803	Failed to configure target data No.
ER07901	Error	7901	Custom color mode is invalid value
ER07903	Error	7903	Failed to configure the custom color mode
ER15001	Error	15001	Meas mode is invalid value
ER15003	Error	15003	Failed to configure the meas mode value
ER15004	Error	15004	The firmware of the instrument does not

			support meas mode setting (require Ver.1.20)
ER15103	Error	15103	Failed to configure timer setting Range: 0-999[s]
ER15104	Error	15104	The firmware of the instrument does not support timer configure (require Ver.1.20)
ER15201	Error	15201	The delay time is invalid
ER15503	Error	15503	Failed to configure the arbitary wavelengths values
ER15504	Error	15504	The firmware of the instrument does not support arbitary wavelengths setting (require Ver.1.20)

Explanation:
This function configures the measurement condition in standalone measurement.
The measurement condition when control the instrument by the CL-SDK cannot be configured.
Since this function configures for each item, please configure the item with specifying the Type to be set.

If necessary, store the configurations by the application.

[Note]
About the measurement condition to be obtained by CLGetMeasData(), please configure the measurement condition by CLSetProperty().

CLGetMeasSetting()

Function :
Gets the measurement condition in standalone measurement

Syntax :
ER_CODE KMAPI CLGetMeasSetting(DEVICE_HANDLE hDevice, CL_MEASSETTYPE Type, CL_MEASSETTING* pSetting)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
Type	I	The measurement condition type to be obtained
pSettings	O	The buffer to store a measurement condition

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to get the measurement condition
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER01001	Error	1001	Parameter pSettings is NULL value
ER01002	Error	1002	The remote mode is OFF
ER01003	Error	1003	The specified type is invalid value
ER07202	Error	7202	Failed to get the display type
ER07302	Error	7302	Failed to get the observer
ER07402	Error	7402	Failed to get color space
ER07502	Error	7502	Failed to get the illuminance units
ER07602	Error	7602	Failed to get the measurement time condition
ER07702	Error	7702	Failed to get user calibration CH
ER07802	Error	7802	Failed to get target data No.
ER15002	Error	15002	Failed to get the meas mode
ER15004	Error	15004	The firmware of the instrument does not support meas mode setting (require Ver.1.20)
ER15102	Error	15102	Failed to get timer setting
ER15104	Error	15104	The firmware of the instrument does not support timer configure (require Ver.1.20)
ER15503	Error	15502	Failed to get the arbitary wavelengths values
ER15504	Error	15504	The firmware of the instrument does not support arbitary wavelengths setting (require Ver.1.20)

Explanation :
This function gets the measurement condition in standalone measurement.
The measurement condition when control the instrument by the CL-SDK cannot be obtained.

Since this function gets for each item, please get the item with specifying the Type to be obtained.

[Note]
About the measurement condition to be obtained by CLGetMeasData(), please get the measurement condition by CLGetProperty().

CLSetSystemSetting()

Function :
Configures the system setting of the instrument

Syntax :
ER_CODE KMAPI CLSetSystemSetting(DEVICE_HANDLE hDevice, CL_SYSTEMTYPE Type, const CL_SYSTEMSETTING* pSetting)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
Type	I	The system setting type to be configured
pSetting	I/O	The system setting to be configured

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to configure the system setting of instrument
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER_PARAM_NULL	Error	13	Parameter pSetting is NULL value
ER00601	Error	601	Parameter pSetting is NULL value
ER00602	Error	602	Specified system type is invalid value
ER00603	Error	603	The remote mode is OFF
ER09602	Error	9602	Date and time value is invalid value
ER09603	Error	9603	Failed to configure the date time
ER05001	Error	5001	Orientation of the display value is invalid value
ER05003	Error	5003	Failed to configure the orientation of the display
ER01601	Error	1601	Buzzer value is invalid value
ER01603	Error	1603	Failed to configure the buzzer setting
ER02001	Error	2001	Language value is invalid value
ER02003	Error	2003	Failed to configure the language setting
ER04901	Error	4901	Date format value is invalid value
ER04903	Error	4903	Failed to configure date format setting
ER08101	Error	8001	Zero calibration expiry value is invalid value
ER08103	Error	8003	Failed to configure zero calibration expiry setting
ER10502	Error	10502	Auto power off value is invalid value
ER10503	Error	10503	Failed to configure auto power off setting
ER10401	Error	10402	Periodic calibration value is invalid value
ER10403	Error	10403	Failed to configure periodic calibration value

Explanation :
This function configures the system setting of the instrument.

Since this function configures for each item, please configure the item with specifying the Type to be set.

CLGetSystemSetting()

Function :
Gets the system setting of the instrument

Syntax :
ER_CODE KMAPI CLGetSystemSetting(DEVICE_HANDLE hDevice, CL_SYSTEMTYPE Type, CL_SYSTEMSETTING* pSetting)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
Type	I	The system setting type to be obtained
pParam	I/O	The buffer to store the system setting

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to get the system setting of instrument
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER00601	Error	601	Parameter pSetting is NULL value
ER00602	Error	602	Specified system type is invalid value
ER00603	Error	603	The remote mode is OFF
ER01602	Error	1602	Failed to get the buzzer setting
ER02002	Error	2002	Failed to get language setting
ER04902	Error	4902	Failed to get date and time format
ER05002	Error	5002	Failed to get orientation of the display
ER08102	Error	8002	Failed to get zero calibration expiry
ER09601	Error	9601	Failed to get date and time
ER10401	Error	10401	Failed to get periodic calibration
ER10501	Error	10501	Failed to get the auto power off setting

Explanation :
This function gets the system setting of the instrument.
Since this function gets for each item, please get the item with specifying the Type to be get.

CLGetDeviceID()

Function :
Gets the instrument information

Syntax :
ER_CODE KMAPI CLGetDeviceID(DEVICE_HANDLE hDevice, CL_DEVID *pDeviceID)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
pDeviceID	O	The buffer to store the instrument information

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to get the instrument information
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER00901	Error	901	Parameter pDevID is NULL value
ER00902	Error	902	The factory calibration is not completed (*1)
ER00903	Error	903	The remote mode is OFF

Explanation :
This function gets the instrument information.

(*1)
When the return value is “ER00902”, the instrument cannot perform a measurement and a zero calibration.
Please send the instrument to the nearest service center.

CLGetSDKVersion()

Function :
Gets the version information of the CL-SDK

Syntax :
ER_CODE KMAPI CLGetSDKVersion(DEVICE_HANDLE hDevice, CL_SDKVERSION *SDKVersion)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
Version	O	The buffer to store the version information

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to get the version information
ER00101	Error	101	Parameter SDKVersion is NULL value

Explanation :
This function gets the version information of the CL-SDK.
You can get the information of 4 types DLL.

CLGetWarning()

Function :
Gets a detail of warning at the latest processing

Syntax :
ER_CODE KMAPI CLGetWarning(DEVICE_HANDLE hDevice, WR_CODE *wrcode)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
wrcode	O	The buffer to store a warning details

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to get the warning details
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER10001	Error	10001	The remote mode is OFF

Explanation :
This function gets a detail of warning at the latest processing.
Each function returns a common value “WARNING” as a return value.

Each function returns the common code “WARNING” as return value.
If would like to confirm the warning detail in case that the return value is a “WARNING”, please get the warning detail with this function.

CLGetPeriodicCalDate()

Function :
Gets the information about the periodic calibration date

Syntax :
ER_CODE KMAPI CLGetPeriodicCalDate(DEVICE_HANDLE hDevice, CL_PERIODICCAL_TYPE Type, CL_DATETIME *pDateTime)

Parameter :

Parameter	I/O	Explanation
hDevice	I	The object handle of the instrument to be controlled
Type	I	Type of periodic calibration date 0 : Start date of the factory calibration 1 : The expiration date of the factory calibration
pDateTIme	O	The buffer to store the calibration date

Return Value :

Return Value	Type	Value	Explanation
SUCCESS	Normal	0	Succeeded to get the periodic calibration date
ER_HANDLE_NULL	Error	10	The object handle is not exist
ER09301	Error	9301	Failed to get the periodic calibration date
ER09302	Error	9302	The remote mode is OFF
ER09303	Error	9303	Parameter DateTime is NULL value
ER09304	Error	9304	Specified type is invalid value

Explanation :
This function gets the information about the periodic calibration date.

The start date is the day when have performed a periodic calibration by service center.
However, first start date is the day when the power was first turned ON.
The expiration date is the day when have elapsed 11 months from the start date.

4.5 Error Code

If the CL-SDK will occur an error, the CL-SDK returns the error code which occurs in the each API for finding where an error occurs and analyzing easily.

About The detail of the error code which returns each function, refer to the return value of each function.

The following error codes are a common error code used by all function.

Value	Message	Explanation
0	SUCCESS	The process is completed successful
1	WARNING	There is a warning though the process is completed successful.
10	ER_HANDLE_NULL	The device handle is NULL value
13	ER_OPENDEVICE	Failed to get the device handle.
14	ER_PARM_NULL	The entered parameter is NULL value
16	ER_ALLMEASURING	The CLDoMeasureAll () is performing

All functions return "WARNING" as a warning code. The detail of warning is required to get by CLGetWarning().

When the return value from the function is a warning value, get the detail of warning by CLGetWarning(), if required.

Value	Message	Explanation
0	WRNONE	No warning
1	WR001	It has passed a certain time from the last zero calibration. Perform a zero calibration.
2	WR002	Illuminance values is low value than the assured value
3	WR003	The item which could not calculated is exist
4	WR004	The corresponding rank is not exist
6	WR_OVER_EXPOSURE	The brightness is over than the proper range of the exposure time you specified.
7	WR_UNDER_EXPOSURE	The brightness is under than the proper range of the exposure time you specified.

5. Appendix

5.1 The rank area setting

The rank area must be configured as a single polygon.

The rank area is created by combining the point (from element No. 0) in order which is stored in the array (the last point is combined to the start point).

By the order which is stored in the array, the rank area might be divided as shown in below figure.

When configuring the rank area, the CL-SDK checks whether the rank area is not divided.

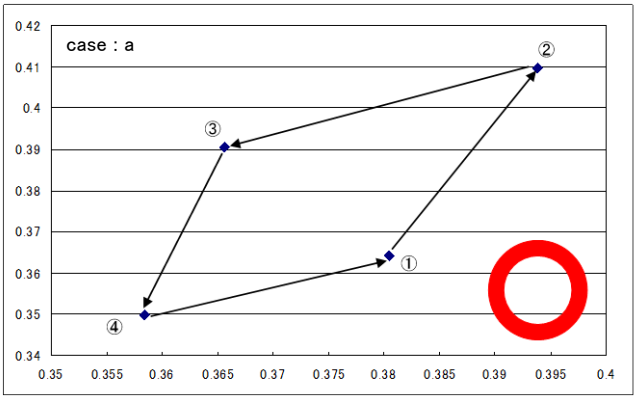
If the rank area is divided, the CL-SDK returns an error code. Be careful the order which stores in the array.

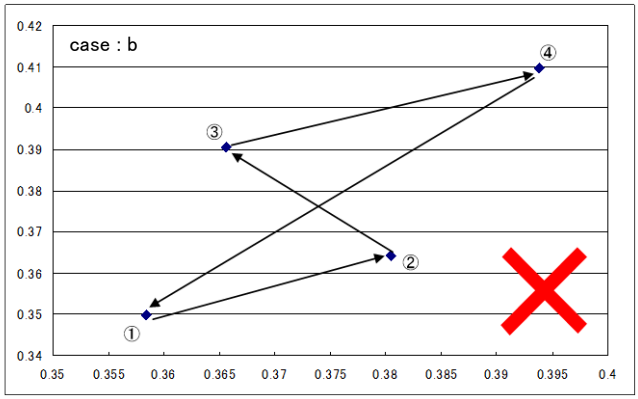
ex.) If configuring the following 4 points as the rank area (The circled number in the figure is the order which is stored in the array)

(x, y) = {0.3584, 0.3499}, {0.3805, 0.3642}, {0.3938, 0.4097}, {0.3656, 0.3905}

a) The array which is stored in order as follows {0.3805, 0.3642}, {0.3938, 0.4097}, {0.3656, 0.3905}, {0.3584, 0.3499}: It is OK as rank area since the area is created by single polygon

b) The array which is stored in order as follows {0.3584, 0.3499}, {0.3805, 0.3642}, {0.3656, 0.3905}, {0.3938, 0.4097} : It is NG as rank area since the area is divided





5.2 Character code table

Available characters for data name of target and user calibration coefficient are as follows.
“SP” means a blank character.

	20	30	40	50	60	70
0	SP	0	@	P	‘	p
1	!	1	A	Q	a	q
2	“	2	B	R	b	r
3	#	3	C	S	c	s
4	$	4	D	T	d	t
5	%	5	E	U	e	u
6	&	6	F	V	f	v
7	’	7	G	W	g	w
8	(	8	H	X	h	x
9	)	9	I	Y	i	y
A	*	:	J	Z	j	z
B	+	;	K	[	k	{
C	,	<	L	¥	l	|
D	-	=	M	]	m	}
E	.	>	N	^	n	
F	/	?	O	_	o	

5.3 Supplementary note

The instruction manual for the CL-500A describes detailed explanation on the correlated color temperature, ⊿uv, dominant wavelength and excitation purity.
Please refer the instruction manual for the CL-500A about these items.


KONICA MINOLTA