CL-SDK

リファレンスマニュアル

[Version 1.30R2]

●本書で使用しているアプリケーション名などの正式名称

本文中の表記	正式名称
VC++	Microsoft Visual C++
Windows 10	Microsoft® Windows® 10 Operating System
Windows 11	Microsoft® Windows® 11 Operating System

●商標について

Microsoft、Windows は、米国 Microsoft Corporation の米国およびその他の国における登録商標です。
その他、本書に記載の会社名、商品名は各社の登録商標または商標です。

●本書に関するご注意

本書の内容の一部または全部を無断で転載することは禁止されています。
本書の内容については、将来予告なしに変更することがあります。
本書の内容を運用した結果につきましては、上記にかかわらず責任を負いかねますので、あらかじめご了承ください。

目次

はじめに

CL-SDK は分光放射照度計(CL-500A)用の PC アプリケーションソフトを開発するためのツールです。
このマニュアルは CL-SDK の使用方法を説明します。アプリケーション開発者は Microsoft Visual C++を使用することを想定しており、プログラミングの方法は Microsoft Visual C++で説明しています。

1. システム環境

CL-SDK は次の環境を必要とします。

- OS： Windows 10(x86)、Windows 10(x64)、Windows 11
- 開発言語：ANSI C インターフェースが利用できる全ての開発環境
- USB2.0 準拠の USB ポートを備えている PC

  [補足] CL-500A は USB2.0 準拠の USB ポートでの動作のみ保証しています。USB1.1、USB3.0 での動作保証はしておりません。また、USB ハブを使用する場合には、セルフパワー型で各ポート 500mA 出力を保証した USB ハブを推奨します。

CL-SDK で制御可能な測定器は以下の測定器です。
測定器：CL-500A
[注意] CL-SDK では CL-200 および CL-200A の制御はできません

2. CL-SDK のインストール・アンインストール方法

2.1 CL-SDK のインストール

CL-SDK はインストールする必要はありません。CL-SDK を PC 上の任意の場所に置いてください。

CL-SDK の解凍フォルダは以下の構成となります。

フォルダ	説明
include file	64bit アプリケーション、32bit アプリケーションで共通に利用する include ファイルのフォルダです
Manual	CL-SDK のリファレンスマニュアルのフォルダです
SampleCode	CL-SDK を利用したサンプルコードのフォルダです
SDK(x64)	CL-SDK のライブラリファイルのフォルダです 64bit アプリケーションはこちらをご利用ください
SDK(x86)	CL-SDK のライブラリファイルのフォルダです 32bit アプリケーションはこちらをご利用ください

以下のファイルがアプリケーションの開発に必要となります。

ダイナミックリンクライブラリ	
libclapi.dll	インターフェースライブラリ
libclcolor.dll	色彩計算ライブラリ
libclcalib.dll	校正演算ライブラリ
libclhid.dll	デバイス通信ライブラリ
インポートライブラリ(*)	
libclapi.lib	インターフェースライブラリ
プログラムソースファイル	
CLAPI.h	CL-SDK の API を定義しているファイル
CLCondition.h	API を使用する際に使う情報を定義しているファイル
CLColorCondition.h	色彩演算に関する情報を定義しているファイル
TypeDefine.h	型定義ファイル
ErrorDefine.h	エラーの定義値ファイル
Version.h	バージョンに関する情報を定義しているファイル

(*)NOTES: 上記のインポートライブラリファイルは Microsoft Visual C++で作成しています。したがって、Microsoft Visual C++でのみ使用可能です。Borland C などのその他の開発環境では使用できません。

2.2 CL-SDK のアンインストール

CL-SDK をアンインストールする場合、PC 上から CL-SDK を削除してください。

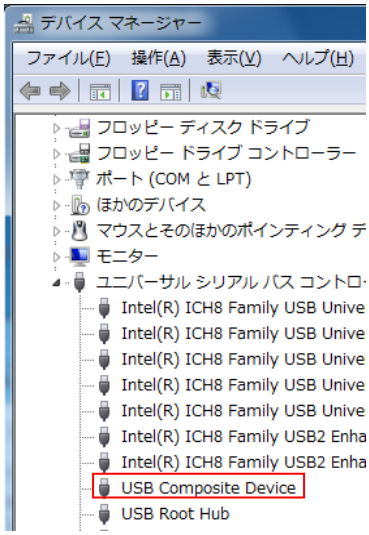
2.3 USB ドライバーのインストール

CL-500A を USB 接続するには、初回接続時に USB のデバイスドライバーをインストールする必要があります。インストールが自動的に開始します。

[注意] ドライバーのインストールが完了するまでは、本体の電源を OFF にしたり、PC から USB ケーブルを抜かないでいでください。
USB ドライバーのインストールに失敗した場合は、CL-S10w のインストールガイドを参考にしてインストールを実行してください。

2.4 USB ドライバーのインストール完了確認

USB ドライバーが正常にインストールされると、CL-500A は USB 機器として認識されます。デバイスマネージャーを開き、新しく“USB Composite Device”もしくは“複合デバイス” と表示されていたら、インストールは正常に完了しています。



3. CL-SDK を使用した VC++アプリケーションの作成

以下に CL-SDK を使用した VC++アプリケーションの作成例を示します。

3.1 アプリケーションの作成手順

以下の概要に従ってアプリケーションの作成手順を説明します。

手順 1: アプリケーションのプロジェクト作成

手順 2: CL-SDK の参照設定

手順 3: デバイスハンドルのオブジェクト作成

手順 4: アプリケーションのコーディング作業

手順 5: コンパイルとデバッグ

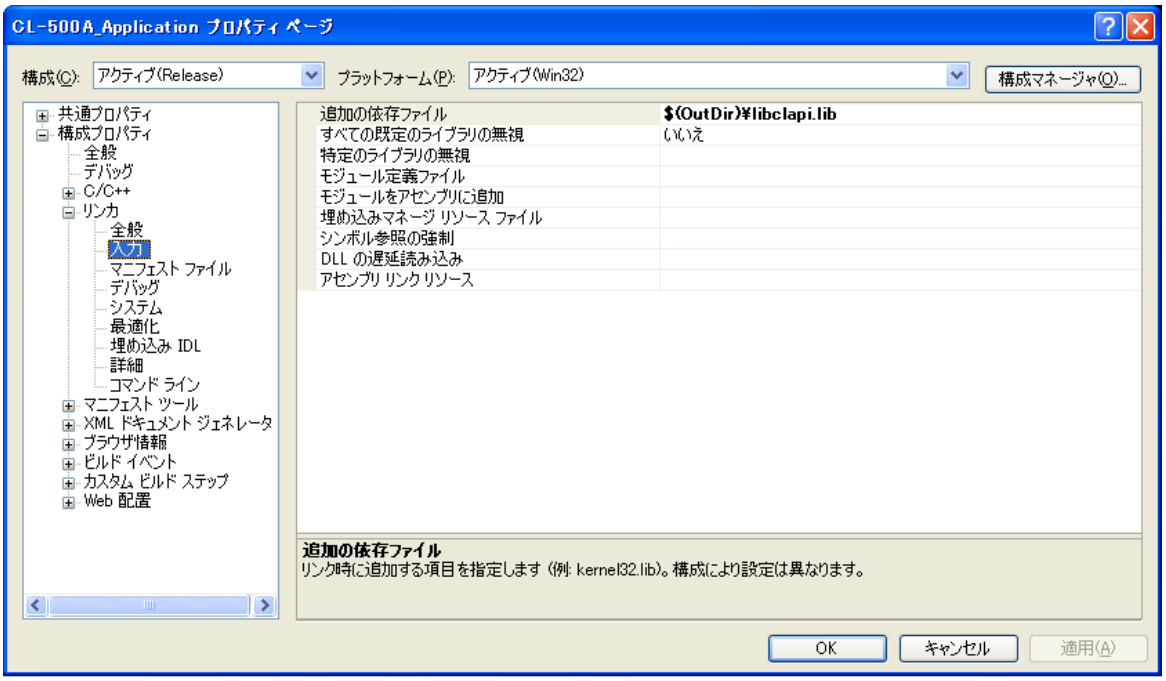
手順 1: VC++アプリケーションの作成

Visual Studio のウィザードに従ってアプリケーションのプロジェクトを作成してください。

手順 2: CL-SDK の参照設定

CL-SDK の参照設定を行って、アプリケーションと CL-SDK がリンクすることを確認します。

[プロジェクト]-[プロパティ]メニューをクリックしてください。以下に示す[プロパティページ]ダイアログが表示されます。ダイアログが表示されたら、ツリーの[リンカ]-[入力]をクリックして、[追加の依存ファイル]にインポートライブラリ(libclapi.lib)を設定してください。



手順 3: デバイスハンドルのオブジェクト作成

アプリケーションは PC に接続されている測定器を制御するためにデバイスハンドルを取得する必要があります。

デバイスハンドルを取得するには、CLOpenDevice()を使用してください。

また測定器を制御する際は、リモートモードを ON にしておく必要があります。したがって、デバイスハンドルを取得した後、CLSetRemoteMode()でリモートモードを ON にしてから、測定器の制御を行ってください。

測定器の制御を終了する際は、CLCloseDevice()を使用して、デバイスハンドルを解放してください。

```
DEVICE_HANDLE hDevice;                  // デバイスハンドルのオブジェクトの定義
CLOpenDevice(&hDevice);                 // デバイスハンドルを取得する
CLSetRemoteMode(hDevice, CL_RMODE_ON)   // リモートモードを ON にする
```

[補足]

CL-SDK ではアプリケーションで複数台の測定器を制御することができます。アプリケーションで複数台の測定器を制御する方法は「§3.3 複数台制御」を参照してください。

手順 4: アプリケーションのコーディング作業

アプリケーションで測定器を制御するためのコードを追加してください。測定器を制御するための API については「§4 CL-SDK API リファレンス」の API リファレンスを参照してください。

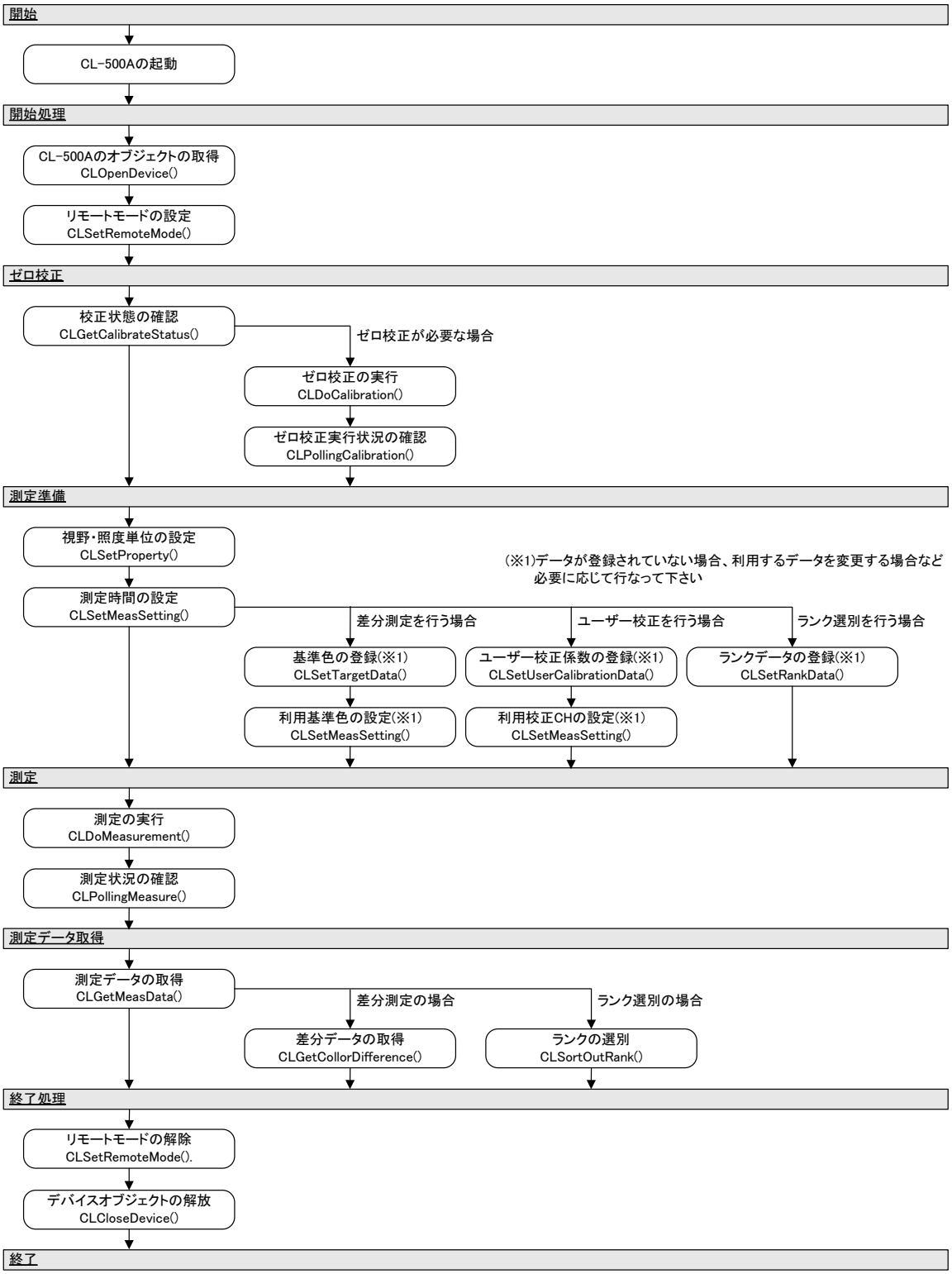
また、CL-SDK のパッケージにはサンプルコードを添付していますので、アプリケーション開発時の参考にしてください。提供しているサンプルコードは以下の通りです。

なお、サンプルコードは VC++2013 で作成しています。

サンプルコード一式	
Measurement.vcproj	照度測定の一連の操作を記載しています
ManualMeasurement.vcproj	マニュアル照度測定の一連の操作を記載しています
ColorDifference.vcproj	差分測定の一連の操作を記載しています
MultipleControl.vcproj	複数台の測定器制御の一連の操作を記載しています
MultipleManualControl.vcproj	複数台のマニュアル測定器制御の一連の操作を記載しています。
Rank.vcproj	ランク選別を行う際の一連の操作を記載しています
UserCalibration.vcproj	ユーザー校正を行う際の一連の操作を記載しています
DeviceStoreData.vcproj	本体内の保存データ取得の際の一連の操作を記載しています
DeviceSetting.vcproj	本体の測定条件設定およびシステム設定一連の操作を記載しています

3.2 基本的な操作フロー

以下に CL-SDK を使用して測定器を制御する場合の基本操作フローを示します。



3.2.1 測定データの演算について

CL-SDK を使用して測定器を制御している際の測定データの演算は CL-SDK で行います。
CL-SDK での測定条件(視野および照度単位)は、CLSetProperty()で設定してください。

また差分測定時の基準色データ、ユーザー校正時の校正係数およびランク選別時のランクデータは本体のデータを使用します。
したがって、測定前に必要に応じてデータの登録や使用データの選択を行っておいてください。

差分測定

CLSetTargetData()を使用して、基準色を本体に登録してください。
登録できる基準色は 20 データです。
CLSetMeasSetting()を使用して、使用する基準色のデータ No.を指定してください。
指定されたデータ No.の基準色を演算時に使用します。

ユーザー校正

CLSetUserCalibrationData()を使用して、校正係数を本体に登録してください。
登録できる校正係数は 10 データです。
CLSetMeasSetting()を使用して、使用する校正 CH を指定してください。
指定された CH の校正係数を演算時に使用します。
なお校正 CH として「UC00」を指定した場合には、コニカミノルタ校正基準による測定を行います。

ランク選別

CLSetRankData()を使用してランクデータを本体に登録してください。
登録できるランクデータは 20 データです。

3.3 複数台制御

CL-SDK では PC に接続されている複数台の測定器を制御することができます(可能な制御台数についてはお問い合わせください)。
CL-SDK で制御する測定器は CLOpenDevice()関数でオブジェクトハンドラを取得します。取得したオブジェクトハンドラは測定器ごとに異なりますので、制御したい測定器のオブジェクトハンドラを各 API のオブジェクトハンドラの引数(hDevice)に入力することで、目的の測定器の制御をすることができます。

[補足]
PC に複数台の測定器が接続された状態で CLOpenDevice()関数でオブジェクトハンドラを取得する場合、取得するオブジェクトハンドラの順番は無作為になっています。
したがって、取得するオブジェクトハンドラで制御する測定器を同じにしたい場合は、シリアル番号を確認して制御して下さい。

3.3.1 測定について

CL-SDK では測定を実行するための関数として、CLDoMeasurement()と CLDoMeasurementAll()を用意しています。測定の用途によって使い分けてください。

CLDoMeasurement()
指定した任意の測定器 1 台の測定を実行します。
複数台制御の場合には、同じタイミングで全ての測定器の測定を実行することができません。

CLDoMeasurementAll()
PC に接続されている全ての測定器で測定を実行します。ただし、測定を実行する測定器は選択することができません。
ほぼ同じタイミングで全ての測定器の測定を実行することができます。
[注意]
完全に同じタイミングにはなりません。また、制御している測定器の数が多いほど、1 番目に測定した測定器と最後に測定をした測定器との測定時間のタイムラグが大きくなります。

```
/* 複数台(測定器:3 台)での測定処理の例 */

// オブジェクトハンドラの定義
DEVICE_HANDLE hDevice1 = NULL;        // 測定器 1 のオブジェクトハンドラの定義
DEVICE_HANDLE hDevice2 = NULL;        // 測定器 2 のオブジェクトハンドラの定義
DEVICE_HANDLE hDevice3 = NULL;        // 測定器 3 のオブジェクトハンドラの定義

// 各測定器のオブジェクトハンドラの取得
ER_CODE ret = CLOpenDevice(&hDevice1);
if (ret != SUCCESS) return;

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

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

// 各測定器のリモートモードを ON にする
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;
}
```

```
// 測定を実行する
int meastime;                          // 測定時間
CL_MEASSTATUS status = CL_MEAS_FREE; // 測定状況

// 測定器 1 での測定
ret = CLDoMeasurement(hDevice1, &meastime);
if (ret > WARNING) return;

// 測定器が完了するまで待つ
do {
    // 少し時間を待ってから、測定状況を確認する
    Sleep(100);

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

} while (status != CL_MEAS_FINISH);

// 測定器 2 での測定
ret = CLDoMeasurement(hDevice2, &meastime);
if (ret > WARNING) return;

// 測定器が完了するまで待つ
status = CL_MEAS_FREE;
do {
    // 少し時間を待ってから、測定状況を確認する
    Sleep(100);

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

} while (status != CL_MEAS_FINISH);

// 測定器 3 での測定
ret = CLDoMeasurement(hDevice3, &meastime);
```

```
if (ret > WARNING) return;

// 測定器が完了するまで待つ
status = CL_MEAS_FREE;
do {
    // 少し時間を待ってから、測定状況を確認する
    Sleep(100);

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

} while (status != CL_MEAS_FINISH);

// 測定データの取得
CL_MEASDATA data;

// 測定器 1 の分光データを取得する
ret = CLGetMeasData(hDevice1, CL_COLORSPACE_SPC, &data);
if (ret > WARNING) return;
for (int i = 0; i < IRRADIANCE_LEN; i++){
    printf("SpctralData[%d]¥t%.01Of¥n", i, data.Spc.Data[i]);
}

// 測定器 2 の Evxy データを取得する
ret = CLGetMeasData(hDevice2, CL_COLORSPACE_EVXY, &data);
if (ret > WARNING) return;
printf("Evxy.Ev¥t%.01Of¥nEvxy.x¥t%.01Of¥nEvxy.y¥t%.01Of¥n", data.Evxy.Ev, data.Evxy.x, data.Evxy.y);

// 測定器 3 の演色評価数データを取得する
ret = CLGetMeasData(hDevice3, CL_COLORSPACE_RENDERING, &data);
if (ret > WARNING) return;
for (int i = 0; i < CL_RENDERING_LEN; i++){
    printf("Rendering[%d]¥t%.01Of¥n", i, data.Rendering.Data[i]);
}
```

3.3.2　ゼロ校正について

測定器のゼロ校正は約 27 秒かかります。

複数台の校正を行う場合には、まず全ての測定器で CLDoCalibration()関数を実行した後に、CLPollingCalibration()で各測定器の校正状況を確認するようにしてください。

```
/* 複数台(測定器:2 台)でのゼロ校正の処理の例 */

// オブジェクトハンドラの定義
DEVICE_HANDLE hDevice1 = NULL;        // 測定器 1 のオブジェクトハンドラの定義
DEVICE_HANDLE hDevice2 = NULL;        // 測定器 2 のオブジェクトハンドラの定義

// 各測定器のオブジェクトハンドラの取得
ER_CODE ret = CLOpenDevice(&hDevice1);
if (ret != SUCCESS) return;

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

// 各測定器のリモートモードを ON にする
ret = CLSetRemoteMode(hDevice1, CL_RMODE_ON);
if (ret != SUCCESS){
          CLCloseDevice(hDevice1);
          return;
}

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

// 各測定器のゼロ校正を実行する
ret = CLDoCalibration(hDevice1);
if(ret != SUCCESS) return;
```

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

// ゼロ校正の実行状況を確認する
CL_CALIBMEASSTATUS calStatus1 = CL_CALIBMEAS_FREE;      // 測定器 1 の実行状況
CL_CALIBMEASSTATUS calStatus2 = CL_CALIBMEAS_FREE;      // 測定器 2 の実行状況
bool bFinish1 = false;                                  // 測定器 1 の完了フラグ
bool bFinish2 = false;                                  // 測定器 2 の完了フラグ

// 全ての測定器のゼロ校正が完了するのを確認する
while(!bFinish1 || !bFinish2){
    // 少し時間を待ってから、測定状況を確認する
    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("ゼロ校正が完了しました¥n");
```

3.3.3 マニュアル測定について

CL-SDK では露光時間と積算回数を指定するマニュアル測定を実行するための関数として、CLDoManualMeasurement()と CLDoManualMeasurementAll()を用意しています。

CLDoManualMeasurement()
指定した任意の測定器 1 台のマニュアル測定を実行します。
複数台制御の場合には、同じタイミングで全ての測定器の測定を実行することができません。

CLDoManualMeasurementAll()
PC に接続されている全ての測定器でマニュアル測定を実行します。ほぼ同じタイミングで全ての測定器の測定を実行することができます。ただし、測定を実行する順番を指定することはできません。

[注意]
完全に同じタイミングにはなりません。また、制御している測定器の数が多いほど、1 番目に測定した測定器と最後に測定をした測定器との測定時間のタイムラグが大きくなります。

```
/*マニュアル測定処理の例 */

// オブジェクトハンドラの定義
DEVICE_HANDLE hDevice = NULL;          // 測定器のオブジェクトハンドラの定義

// 測定器のオブジェクトハンドラの取得
ER_CODE ret = CLOpenDevice(&hDevice);
if (ret != SUCCESS) return;

// 各測定器のリモートモードを ON にする
ret = CLSetRemoteMode(hDevice, CL_RMODE_ON);
if (ret != SUCCESS){
    CLCloseDevice(hDevice1);
    return;
}

// 測定を実行する
int exposureTime = 400;                          // 露光時間
```

```
int cumulativeNum = 10;                          // 積算回数

CL_MEASSTATUS status = CL_MEAS_FREE; // 測定状況

// 測定器 1 での測定
ret = CLDoManualMeasurement(hDevice1, &exposureTime, &cumulativeNum);
if (ret > WARNING) return;

// 測定器が完了するまで待つ
do {
    // 少し時間を待ってから、測定状況を確認する
    Sleep(100);

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

} while (status != CL_MEAS_FINISH);

// 測定データの取得
CL_MEASDATA data;

// 分光データを取得する
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]);
}

// 測定器の Evxy データを取得する
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);

// の演色評価数データを取得する
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　本体キーをトリガーにした測定

CL-SDK では本体キーをトリガーにした測定をすることできます。
GetButtonStatus()関数で、トリガーにするキーの状態(キーが押されたか/押されていないか)を確認します。
キーが押されたら確認処理を抜けて CLDoMeasurement()で実行することで、本体キーをトリガーにした測定が実行できます。

```
/* 本体キーをトリガーにした測定例 */

// オブジェクトハンドラの定義
DEVICE_HANDLE hDevice = NULL;

// オブジェクトハンドラの取得
ER_CODE ret = CLOpenDevice(&hDevice);
if (ret != SUCCESS) return;

// リモートモードを ON にする
ret = CLSetRemoteMode(hDevice, CL_RMODE_ON);
if (ret != SUCCESS){
    CLCloseDevice(hDevice);
    return;
}

// 測定キーが押されるまで待機する
CL_KEYINFO key_state;
do {
    // 測定キーの状態確認
    ret = CLGetButtonStatus(hDevice, &key_state);
    if (ret != SUCCESS) return;
} while (key_state.MeasKey != CL_KEY_STATE_ON);    // 測定キーが押されたら、処理を抜ける

// 測定を実行する
int meastime;
ret = CLDoMeasurement(hDevice, &meastime);
```

3.5　本体内保存データの取得

CL-SDK では本体内の保存データを取得することができます。
取得する色彩データの種類を指定して、保存データを取得して下さい。

```
/* 本体内の保存データの取得処理の例 */

// オブジェクトハンドラの定義
DEVICE_HANDLE handle = NULL;

// オブジェクトハンドラの取得
ER_CODE ret = CLOpenDevice(&handle);
if(ret != SUCCESS) return;

// リモートモードを ON にする
ret = CLSetRemoteMode(handle, CL_RMODE_ON);
if(ret != SUCCESS){
    CLCloseDevice(handle);
    return;
}

// 本体内の保存データ数を取得する
int32_km num;      // 保存データ数
ret = CLGetDeviceStoredDataNum(handle, &num);
if(ret != SUCCESS) return;

// 本体内の保存データ(Ev/x/y)を取得する
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++){
    printf("[%4d/%2d/%2d %2d:%2d:%2d]¥n",
```

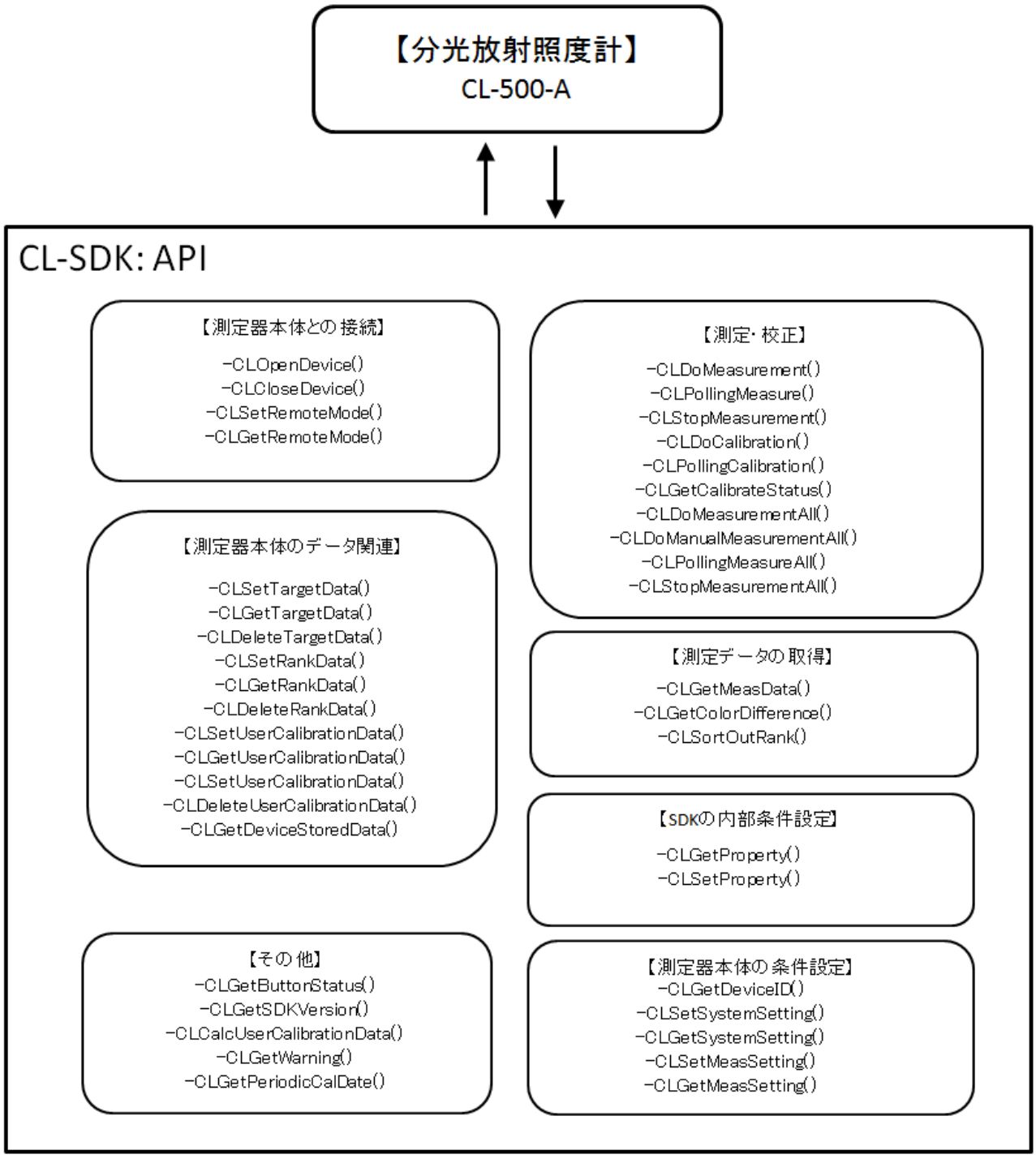
```
        pStoredData[i].DateTime.Year, pStoredData[i].DateTime.Month, pStoredData[i].DateTime.Day,
        pStoredData[i].DateTime.Hour, pStoredData[i].DateTime.Minute, pStoredData[i].DateTime.Second
    );
    printf("data[%d].Ev:%f¥n", i + 1, pStoredData[i].Data.Evxy.Ev);
    printf("data[%d].x:%f¥n", 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 リファレンス

CL-SDK API 一覧

以下に CL-SDK で提供している API の一覧を示します。

関数名	内容
CLOpenDevice()	PC に接続している測定器のオブジェクトハンドラを取得します
CLCloseDevice()	CL-SDK で確保している CL-500A のオブジェクトハンドラを解放します
CLSetRemoteMode()	リモートモードの ON/OFF を設定します
CLGetRemoteMode()	リモートモードの ON/OFF を取得します
CLDoMeasurement()	測定を実行します
CLDoManualMeasurement()	マニュアル測定を実行します
CLPollingMeasure()	測定状況を取得します
CLStopMeasurement()	測定を中断します
CLDoMeasurementAll()	接続されている全ての測定器に対して測定を実行します
CLDoManualMeasurementAll()	接続されている全ての測定器に対してマニュアル測定を実行します
CLPollingMeasureAll()	CLDoMeasurementAll()による測定での測定状況を取得します
CLStopMeasurementAll()	CLDoMeasurementAll()による測定を中断します
CLDoCalibration()	ゼロ校正を実行します
CLPollingCalibration()	ゼロ校正の実行状況を取得します
CLGetCalibrateStatus()	校正状況を取得します
CLGetMeasData()	最新の測定での各種色彩値データを取得します
CLGetColorDifference()	最新の測定での各種差分データを取得します
CLSortOutRank()	測定データのランクを選別します
CLSetRankData()	ランク選別に利用するランクデータを本体に登録します
CLGetRankData()	本体に登録されているランクデータを取得します
CLDeleteRankData()	本体に登録されているランクデータを削除します
CLSetTargetData()	本体に基準色を登録します
CLGetTargetData()	本体に登録されている基準色を取得します
CLDeleteTargetData()	本体に登録されている基準色を削除します
CLGetDeviceStoredDataNum()	本体に保存されている測定データの数を取得します
CLGetDeviceStoredData()	本体に保存されている測定データを取得します
CLDeleteDeviceStoredData()	本体に保存されている測定データを削除します
CLSetUserCalibrationData()	本体にユーザー校正係数を登録します
CLGetUserCalibrationData()	本体に登録されているユーザー校正係数を取得します
CLDeleteUserCalibrationData()	本体に登録されているユーザー校正係数を削除します
CLCalcUserCalibrationData()	ユーザー校正係数を算出します
CLSetProperty()	CL-SDK での測定データの演算時の測定条件を設定します
CLGetProperty()	CL-SDK での測定データの演算時の測定条件を取得します
CLGetButtonStatus()	本体キーの状態を取得します
CLSetMeasSetting()	測定器単体での測定時の測定条件を設定します
CLGetMeasSetting()	測定器単体での測定時の測定条件を取得します
CLSetSystemSetting()	本体のシステム設定を設定します
CLGetSystemSetting()	本体のシステム設定を取得します
CLGetDeviceID()	本体情報を取得します
CLGetSDKVersion()	CL-SDK のバージョン情報を取得します
CLGetWarning()	最新の実行処理での警告内容を取得します
CLGetPeriodicCalDate()	定期校正日時に関する情報を取得します


【分光放射照度計】
CL-500-A
CL-SDK: API
【測定器本体との接続】
-CLOpenDevice()
-CLCloseDevice()
-CLSetRemoteMode()
-CLGetRemoteMode()
【測定器本体のデータ関連】
-CLSetTargetData()
-CLGetTargetData()
-CLDeleteTargetData()
-CLSetRankData()
-CLGetRankData()
-CLDeleteRankData()
-CLSetUserCalibrationData()
-CLGetUserCalibrationData()
-CLSetUserCalibrationData()
-CLDeleteUserCalibrationData()
-CLGetDeviceStoredData()
【その他】
-CLGetButtonStatus()
-CLGetSDKVersion()
-CLCalcUserCalibrationData()
-CLGetWarning()
-CLGetPeriodicCalDate()
【測定・校正】
-CLDoMeasurement()
-CLPollingMeasure()
-CLStopMeasurement()
-CLDoCalibration()
-CLPollingCalibration()
-CLGetCalibrateStatus()
-CLDoMeasurementAll()
-CLDoManualMeasurementAll()
-CLPollingMeasureAll()
-CLStopMeasurementAll()
【測定データの取得】
-CLGetMeasData()
-CLGetColorDifference()
-CLSortOutRank()
【SDKの内部条件設定】
-CLGetProperty()
-CLSetProperty()
【測定器本体の条件設定】
-CLGetDeviceID()
-CLSetSystemSetting()
-CLGetSystemSetting()
-CLSetMeasSetting()
-CLGetMeasSetting()

4.1 定義値一覧

	定義値	説明
IRRADIANCE_LEN	421	分光放射照度のデータ数です。
IRRADIANCE_BEGIN	360	分光放射照度データの開始波長(360nm)です
IRRADIANCE_END	780	分光放射照度データの終端波長(780nm)です
IRRADIANCE_PITCH	1	分光放射照度データの波長間隔(1nm 間隔)です
CL_OVER_ERROR_VALUE	-10000	測定値がオーバーエラーになった際に取得する値です
CL_INCOMPUTABLE_VALUE	-11000	相関色温度の演算などで値を算出できない測定対象物を測定した際に取得する値です。
CL_USERCALIB_NAMESIZE	13	ユーザー校正係数のデータ名の設定可能文字数です
CL_USERCALIB_CH_ALLDEL	-1	ユーザー校正係数の全削除を実行する際に指定します
CL_USERCALIB_CH_START	1	ユーザー校正係数の指定可能な開始 No.です
CL_USERCALIB_CH_END	10	ユーザー校正係数の指定可能な最終 No.です
CL_TARGET_NAMESIZE	13	基準色データの名前の最大文字数です
CL_TARGET_NO_ALLDEL	-1	基準色データの全削除を実行する際に指定します
CL_TARGET_NO_START	1	基準色データの指定可能な開始 No.です
CL_TARGET_NO_END	20	基準色データの指定可能な最終 No.です
CL_RANK_POINTNUM	10	ランク領域の設定可能な最大ポイント数です
CL_RANK_NAMESIZE_SAVE	41	ランクデータのデータ名の設定可能文字バイト数です
CL_RANK_NO_ALLDEL	-1	ランクデータの全削除を実行する際に指定します
CL_RANK_NO_START	1	ランクデータの指定可能な開始 No.です
CL_RANK_NO_END	20	ランクデータの指定可能な最終 No.です
CL_LIST_NO_ALLDEL	-1	本体保存データの全削除を実行する際に指定します
CL_LIST_NO_START	1	本体保存データの指定可能な開始 No.です
CL_LIST_NO_END	100	本体保存データの指定可能な最終 No.です
CL_RENDERING_LEN	16	演色評価数のバッファサイズです
CL_USERCUSTOM_LEN	4	本体のカスタム表色モードの指定データ数です

4.2 列挙体一覧

CL_REMOTEMODE

CLSetRemoteMode()および CLGetRemoteMode()でリモートモード設定の設定・取得する際に利用する列挙体です。

CL_REMOTEMODE	定義値	説明
CL_RMODE_OFF	0	リモートモードを OFF にします
CL_RMODE_ON	1	リモートモードを ON にします

CL_MEASSTATUS

CLPollingMeasure()で測定状況を取得する際に利用する列挙体です。

CL_MEASSTATUS	定義値	説明
CL_MEAS_FREE	0	測定が行われていません
CL_MEAS_BUSY	1	測定を実行中です
CL_MEAS_FINISH	2	測定が完了しました

CL_CALIBSTATUS

CLGetCalibrateStatus()でゼロ校正の実施状態を取得する際に利用する列挙体です。

CL_CALIBSTATUS	定義値	説明
CL_CALIB_NA	0	ゼロ校正ができません
CL_CALIB_OK	1	ゼロ校正が完了しています
CL_CALIB_WR	2	ゼロ校正をすることを推奨します(最終のゼロ校正から一定期間が経過しています)(*1)
CL_CALIB_NG	3	ゼロ校正が行われていません

(*1)ゼロ校正を行わなくても、測定を実行することは可能です。
ただし、精度よく測定を行うためにもゼロ校正を行ってください。

CL_CALIBMEASSTATUS

CLPollingCalibration()でゼロ校正の状況を取得する際に利用する列挙体です。

CL_CALIBMEASSTATUS	定義値	説明
CL_CALIBMEAS_FREE	0	ゼロ校正が行われていません
CL_CALIBMEAS_BUSY	1	ゼロ校正を実行中です
CL_CALIBMEAS_FINISH	2	ゼロ校正が完了しました

CL_RANK_TYPE

CLSetRankData()および CLGeRankData()でランクデータのランク領域の種別の設定・取得する際に利用する列挙体です。

CL_RANK_TYPE	定義値	説明
CL_RANK_CHROMA	0	ランク領域の各ポイントを xy で指定します
CL_RANK_COLORTEMP	1	ランク領域の各ポイントを相関色温度・⊿uv で指定します

CL_KEYSTATUS

CLGetButtonStatus()でボタン状態を取得する際に利用する列挙体です。

CL_KEYSTATUS	定義値	説明
CL_KEY_STATE_OFF	0	キーが押されていません
CL_KEY_STATE_ON	1	キーが押されています

CL_COLORSPACE

CLGetMeasData()やCLGetDeviceStoredData()で取得する色彩値データを指定する際に利用する列挙体です。

CL_MEAS_STATUS	定義値	説明
CL_COLORSPACE_EVXY	0	Ev、x、y
CL_COLORSPACE_EVUV	1	Ev、u’、v’
CL_COLORSPACE_EVTCPJISDUV	2	Ev、相関色温度Tcp(JIS方式)(*1)、⊿uv
CL_COLORSPACE_EVTCPDUV	3	Ev、相関色温度Tcp(*2)、⊿uv
CL_COLORSPACE_EVDWPE	4	Ev、主波長λd、刺激純度Pe
CL_COLORSPACE_XYZ	5	X、Y、Z
CL_COLORSPACE_RENDERING	6	演色評価数(Ra、R1～R15)
CL_COLORSPACE_PW	7	ピーク波長
CL_COLORSPACE_SPC	8	分光放射照度
CL_COLORSPACE_SCOTOPIC	9	Ev, 暗所視照度Ev', S/P比

(*1)：JIS Z 8725で規定されている計算式を用いた色温度

(*2)：従来よりコニカミノルタが採用している固有アルゴリズムによる色温度
色温度の考え方はJISと同じですが、演算時間短縮のための高速計算アルゴリズムを採用しています。

相関色温度(JIS)と相関色温度では、値に若干の差が生じる場合があります。
JIS Z 8725 で規定されている、色温度を算出できる色温度範囲において、相関色温度(JIS)を基準にしたときの相関色温度の誤差量は±3%以内です。

CL_PROPERTIES

CLSetProperty()および CLGetProperty()で設定/取得するプロパティ情報を設定・取得する際に利用する列挙体です。

CL_PROPERTIES	定義値	説明
CL_PR_OBSERVER	0	視野
CL_PR_ILLUNIT	1	照度単位

CL_OBSERVER

CLSetProperty()、CLGetProperty()、CLSetMeasSettings()およびCLGetMeasSettings()で、視野を設定・取得する際に利用する列挙体です。

CL_OBSERVER	定義値	説明
CL_OBS_02DEG	0	2° 視野
CL_OBS_10DEG	1	10° 視野

CL_MEASSETTYPE

CLSetMeasSetting()および CLGetMeasSetting()で設定・取得する測定条件を指定する際に利用する列挙体です。

CL_MEASSETTYPE	定義値	説明
CL_MEASSET_DISPTYPE	0	表示形式
CL_MEASSET_OBS	1	視野
CL_MEASSET_COLORSPACE	2	表色モード
CL_MEASSET_ILLUNIT	3	照度単位
CL_MEASSET_EXPOSURETIME	4	測定時間
CL_MEASSET_USERCALIBCH	5	ユーザー校正 CH
CL_MEASSET_TARGETNO	6	基準色データ No.
CL_MEASSET_USERCUSTOM	7	カスタム表色モード
CL_MEASSET_MEASMODE	8	測定モード
CL_MEASSET_TIMERCONF	9	タイマー
CL_MEASSET_USERWAVELENGTH	10	カスタム表色モード任意波長

CL_DISP_TYPES

CLSetMeasSetting()および CLGetMeasSetting()で本体画面の表示形式設定・取得する際に利用する列挙体です。

CL_DISP_TYPES	定義値	説明
CL_DISP_ABS	0	絶対値表示
CL_DISP_DIFF	1	差分表示
CL_DISP_RANK	2	ランク選別

CL_COLOR_MODE

CLSetMeasSetting()および CLGetMeasSetting()で表示する表色系の設定・取得する際に利用する列挙体です。

CL_COLOR_MODE	定義値	説明
CL_MODE_EVXY	0	Ev、x、y
CL_MODE_EVUV	1	Ev、u’、v’
CL_MODE_EVTCPDUV	2	Ev、相関色温度 Tcp、⊿uv
CL_MODE_XYZ	3	X、Y、Z
CL_MODE_EVDWPE	4	Ev、主波長λd、刺激純度 Pe
CL_MODE_RENDERING	5	演色評価数(Ra、R1～R15)
CL_MODE_SPECTRAL_GRAPH	6	分光放射照度グラフ、ピーク波長
CL_MODE_CUSTOM	7	カスタム表色モード

CL_MEASUREMENT_TIME

CLSetMeasSettings()や CLGetMeasSettings()で測定時間を設定・取得する際に利用する列挙体です。

CL_MEASUREMENT_TIME	定義値	説明
CL_MEAS_TIME_FAST	0	FAST モード(測定時間：約 0.5 秒)
CL_MEAS_TIME_SLOW	1	SLOW モード(測定：約 2.5 秒)
CL_MEAS_TIME_AUTO	2	AUTO モード(測定光源の明るさに応じて適切な露光時間で測定します：最長約 27 秒)
CL_MEAS_TIME_SUPER_FAST	3	SUPER FAST モード(測定時間：約 0.2 秒)

		[注意]：SUPER FAST モードは PC からの制御時のみ利用可能です(本体単独の測定では設定できません)

CL_CUSTOMDATA_ITEM

CLSetMeasSetting()および CLGetMeasSetting()でカスタム表色モードに表示する項目の設定・取得する際に利用する列挙体です。

CL_CUSTOMDATA_ITEM	定義値	説明
CL_CUSTOM_NONE	0	表示なし
CL_CUSTOM_EV_DIFF	1	Ev(基準色との差)
CL_CUSTOM_EV_RATIO	2	Ev(基準色との比率)
CL_CUSTOM_SX	3	x
CL_CUSTOM_SY	4	y
CL_CUSTOM_U	5	u’
CL_CUSTOM_V	6	v’
CL_CUSTOM_TCP	7	相関色温度 Tcp
CL_CUSTOM_DUV	8	⊿uv
CL_CUSTOM_LX_DIFF	9	X(基準色との差)
CL_CUSTOM_LX_RATIO	10	X(基準色との比率)
CL_CUSTOM_LY_DIFF	11	Y(基準色との差)
CL_CUSTOM_LY_RATIO	12	Y(基準色との比率)
CL_CUSTOM_LZ_DIFF	13	Z(基準色との差)
CL_CUSTOM_LZ_RATIO	14	Z(基準色との比率)
CL_CUSTOM_DW	15	主波長 λd
CL_CUSTOM_PE	16	刺激純度 Pe
CL_CUSTOM_RANK	17	ランク
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	暗所視照度 Ev’[差]
CL_CUSTOM_EVS_RATIO	35	暗所視照度 Ev’[%]
CL_CUSTOM_SP	36	S/P 比[差]
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

CLSetSystemSetting()および CLGetSystemSetting()で本体のシステム設定に関する設定・取得する際に利用する列挙体です。

CL_SYSTEMTYPE	定義値	説明
CL_SYSTEM_DATETIME	0	本体の日時設定
CL_SYSTEM_DISPLAY	1	本体画面の表示設定
CL_SYSTEM_BEEP	2	ブザー設定
CL_SYSTEM_LANGUAGE	3	表示言語
CL_SYSTEM_DATEFORMAT	4	表示する日時の表示フォーマット
CL_SYSTEM_UCAL_LIMIT	5	校正警告の表示期間設定
CL_SYSTEM_AUTOPOWEROFF	6	オートパワーオフ設定
CL_SYSTEM_PCALNOTIFY	7	定期校正のメッセージ表示設定

CL_DISPLAYTYPE

CLSetSystemSetting()および CLGetSystemSetting()で本体液晶の表示回転の設定・取得する際に利用する列挙体です。

CL_DISPLAYTYPE	定義値	説明
DISPLAY_NORMAL	0	通常表示
DISPLAY_INVERSE	1	反転表示

CL_BEEP

CLSetSystemSetting()および CLGetSystemSetting()でブザー音設定の設定・取得する際に利用する列挙体です。

CL_BEEP	定義値	説明
CL_BEEP_OFF	0	ブザー音を OFF にします
CL_BEEP_ON	1	ブザー音を ON にします [注意] 測定時間が SUPER FAST の場合は、ブザー音を ON にしてもブザー音は鳴りません

CL_LANGMODE

CLSetSystemSetting()や CLGetSystemSetting()で、本体の表示言語を設定・取得する際に利用する列挙体です。

CL_LANGMODE	定義値	説明
CL_LANG_ENG	0	英語
CL_LANG_JAN	1	日本語
CL_LANG_CHN	2	中国語

CL_TYPE_DATEFORMAT

CLSetSystemSetting()および CLSetSystemSetting()で日付フォーマットの設定・取得する際に利用する列挙体です。

CL_TYPE_DATEFORMAT	定義値	説明
CL_TYPE_YYMMDD	0	年/月/日の順で表示します
CL_TYPE_MMDDYY	1	月/日/年の順で表示します
CL_TYPE_DDMMYY	2	日/月/年の順で表示します

CL_USERCAL_LIMIT

CLSetSystemSetting()や CLGetSystemSetting()で、最終校正からの警告メッセージを表示するまでの時間を設定・取得する際に利用する列挙体です。

CL_USERCAL_LIMIT	定義値	説明
CL_USERCAL_LIMIT_3H	0	最終校正から 3H 経過すると警告メッセージが表示されます
CL_USERCAL_LIMIT_6H	1	最終校正から 6H 経過すると警告メッセージが表示されます
CL_USERCAL_LIMIT_12H	2	最終校正から 12H 経過すると警告メッセージが表示されます
CL_USERCAL_LIMIT_24H	3	最終校正から 24H 経過すると警告メッセージが表示されます
CL_USERCAL_LIMITLESS	4	時間経過による警告メッセージを表示しません

CL_AUTOPOWEROFF

CLSetSystemSetting()および CLGetSystemSetting()でオートパワーオフ設定を設定・取得する際に利用する列挙体です。

CL_AUTOPOWEROFF	定義値	説明
CL_AUTOPOWEROFF_OFF	0	オートパワーオフを OFF にします
CL_AUTOPOWEROFF_ON	1	オートパワーオフを ON にします

オートパワーオフが ON の場合、15 分間操作しないと本体の電源が OFF になります。
ただし本体が PC に接続されている場合には、オートパワーオフが ON の場合でも本体の電源は OFF になりません。

CL_PCALNOTIFY

CLSetSystemSetting()および CLGetSystemSetting()で定期校正メッセージ設定の設定・取得する際に利用する列挙体です。

CL_PCALNOTIFY	定義値	説明
CL_PCALNOTIFY_OFF	0	定期校正推奨メッセージを表示しません
CL_PCALNOTIFY_ON	1	定期校正推奨メッセージを表示します

CL_PERIODICCAL_TYPE

CLGetPeriodicCalDate で定期校正の日時の取得の際に利用する列挙体です。

CL_PERIODICAL_TYPE	定義値	説明
CL_PERIODICCAL_START	0	定期校正の開始日時を取得します
CL_PERIODICCAL_END	1	定期校正の終了日時を取得します

CL_MEASMODE

CLSetMeasSetting()および CLGetMeasSetting()で測定モードを設定・取得する際に利用する列挙体です

CL_MEASMODE	定義値	説明
CL_MEASMODE_SINGLE	0	シングル測定
CL_MEASMODE_AVERAGED	1	平均化測定
CL_MEASMODE_CONTINUOUS	2	連続測定

4.3 構造体・共用体一覧

CL_TARGET_DATA

概要：
基準色データの構造体

形式：

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

変数：

変数	説明
SPCData	分光放射照度のデータを格納します。 波長範囲：360～780nm 波長間隔：1nm 間隔 データ数：421 データ
DateTime	保存日時を格納します 設定可能日時：2011/01/01 0:00:00 ～2100/12/31 23:59:59
Name	基準色名を格納します。 文字数：13 文字(終端文字を含む) 使用可能文字列：§5.2 文字コード表を参照して下さい。

説明：
CLSetTargetData()または CLGetTargetData()で基準色データを扱う際に利用します。

CL_USERCALIB DATA

概要：
ユーザー校正に利用するユーザー校正係数の構造体

形式：

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

変数：

変数	説明
Coef	ユーザー校正係数を格納します 波長範囲：360～780nm 波長間隔：1nm 間隔 データ数：421 データ 設定範囲：0.001～1000
DateTime	日時情報を格納します 設定可能日時：2011/01/01 0:00:00 ～2100/12/31 23:59:59
Name	校正係数の名前を格納します 最大文字数：13 文字(終端文字を含む) 使用可能文字列：§5.2　文字コードを参照して下さい

説明：
CLSetUserCalibrationData()または CLGetUserCalibrationData()でユーザー校正係数を扱う際に利用します。

CL_MEASDATA

概要：
測定データの共用体

形式：

```
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;
```

変数：

引数	説明
Evxy	Evxy データを格納します
Evuv	Evu’v’ データを格納します
EvTduv	Ev 相関色温度⊿uv データを格納します
EvDWPe	Ev/主波長/刺激純度データを格納します
XYZ	XYZ データを格納します
Rendering	演色評価数データを格納します
Pw	ピーク波長データを格納します
Spc	分光放射照度データを格納します

説明：
CLGetMeasData()で測定データを扱う際に利用します。

[注意]
この変数は共用体ですので、CLGetMeasData()で複数の種類の色彩データを取得する場合は取得した測定データを別のバッファに格納するなどしてから、次の色彩データを取得するようにしてください。

CL_SPCDATA

概要：
分光放射照度データの構造体

形式：

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

変数：

引数	説明
Data	分光放射照度データを格納します

説明：
CLGetMeasData()で測定データを扱う際に利用します。

CL_EvxyDATA

概要：
Ev、x、y データの構造体

形式：

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

変数：

引数	説明
Ev	Ev の測定データを格納します
x	x の測定データを格納します
y	y の測定データを格納します

説明：
CLGetMeasData()で測定データを扱う際に利用します。
Ev、x、y データの測定データを格納します。

CL_EvuvDATA

概要：
Ev、u’、v’データの構造体

形式：

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

変数：

引数	説明
Ev	Ev の測定データを格納します
u	u’の測定データを格納します
v	v’の測定データを格納します

説明：
CLGetMeasData()で測定データを扱う際に利用します。
Ev、u’、v’データの測定データを格納します。

CL_EvTduvDATA

概要：
Ev、相関色温度 Tcp、⊿uv の構造体

形式：

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

変数：

引数	説明
Ev	Ev の測定データを格納します
T	相関色温度 Tcp の測定データを格納します
duv	⊿uv の測定データを格納します

説明：
CLGetMeasData()で測定データを扱う際に利用します。
Ev、Tcp、⊿uv の測定データを格納します。

CL_EvDWPeDATA

概要：
Ev、主波長λd、刺激純度 Pe のデータの構造体

形式：

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

変数：

引数	説明
Ev	Ev の測定データを格納します
DW	主波長λd の測定データを格納します(*1)
Pe	刺激純度 Pe の測定データを格納します

説明：
CLGetMeasData()で測定データを扱う際に利用します。
Ev、主波長λd、刺激純度 Pe の測定データを格納します。

(*1)
測定データが補色主波長の場合には、測定データはマイナス値になります。

CL_XYZDATA

概要：
X、Y、Z の構造体

形式：

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

変数：

引数	説明
X	X の測定データを格納します
Y	Y の測定データを格納します
Z	Z の測定データを格納します

説明：
CLGetMeasData()で測定データを扱う際に利用します。
X、Y、Z データの測定データを格納します。

CL_RenderingDATA

概要：
演色評価数の構造体

形式：

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

変数：

引数	説明
Data	演色評価数の測定データを格納します 測定データは以下の順序で配列に格納されます

0	1	2	・・・	14	15
Ra	R1	R2	・・・	R14	R15

説明：
CLGetMeasData()および CLGetColorDifference()で測定データを扱う際に利用します。
演色評価数の測定データを格納します。

CL_PWDATA

概要：
ピーク波長の構造体

形式：

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

変数：

引数	説明
PeakWave	ピーク波長の測定データを格納します

説明：
CLGetMeasData()および CLGetColorDifference()で差分測定での測定データを扱う際に利用します。
ピーク波長の測定データを格納します。

CL_ScotopicDATA

概要：
Ev, 暗所視照度 Es, S/P 値データの構造体

形式：

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

変数：

変数	説明
Ev	Ev の測定データを格納します
Es	暗所視照度 Ev'の測定データを格納します
SP	S/P 比の測定データを格納します

CL_DIFFDATA

概要：
差分データの共用体

形式：

```
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;
```

変数：

引数	説明
Evxy	Ev、x、y の差分データを格納します
Evuv	Ev、u、v の差分データを格納します
EvTduv	Ev、相関色温度の差分データを格納します
EvDWPe	Ev、主波長、刺激純度の差分データを格納します
XYZ	XYZ の差分データを格納します
Rendering	演色評価数の差分データを格納します
Pw	ピーク波長の差分データを格納します

説明：
CLGetColorDifference()で差分測定での測定データを扱う際に利用します。

[注意]
この変数は共用体ですので、CLGetColorDifference()で複数の種類の差分データを取得する場合は取得した差分データを別のバッファに格納するなどしてから、次の差分データを取得するようにしてください。

CL_Evxy_DIFFDATA

概要：
Ev、x、y の差分データの構造体

形式：

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

変数：

引数	説明
Ev_Abs	Ev の差分データ(基準色との差)を格納します
Ev_Ratio	Ev の差分データ(基準色との比率)を格納します
x	x の差分データを格納します
y	y の差分データを格納します

説明：
CLGetColorDifference()で差分データを扱う際に利用します。
Ev、x、y の差分データを格納します。

CL_Evuv_DIFFDATA

概要：
Ev、u’、v’の差分データの構造体

形式：

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

変数：

引数	説明
Ev_Abs	Ev の差分データ(基準色との差)を格納します
Ev_Ratio	Ev の差分データ(基準色との比率)を格納します
u	u’の差分データを格納します
v	v’の差分データを格納します

説明：
CLGetColorDifference()で差分データを扱う際に利用します。
Ev、u’、v’データの差分データを格納します。

CL_EvTduv_DIFFDATA

概要：
Ev、相関色温度 Tcp の差分データの構造体

形式：

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

変数：

引数	説明
Ev_Abs	Ev の差分データ(基準色との差)を格納します
Ev_Ratio	Ev の差分データ(基準色との比率)を格納します
T	相関色温度 Tcp の差分データを格納します
duv	未使用

説明：
CLGetColorDifference()で差分データを扱う際に利用します。
Ev、相関色温度 Tcp の差分データを格納します。

[注意]
⊿uv には差分が存在しないため、duv には差分データが格納されません。

CL_EvDWPe_DIFFDATA

概要：
Ev、主波長λd、刺激純度 Pe の差分データの構造体

形式：

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

変数：

引数	説明
Ev_Abs	Ev の差分(基準色との差)を格納します
Ev_Ratio	Ev の差分(基準色との比率)を格納します
DW	主波長λd の差分データを格納します
Pe	刺激純度 Pe の差分データを格納します

説明：
CLGetColorDifference()で差分データを扱う際に利用します。
Ev、主波長λd、刺激純度 Pe の差分データを格納します。

CL_XYZ_DIFFDATA

概要：
X、Y、Z の差分データの構造体

形式：

```
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;
```

変数：

引数	説明
X_Abs	X の差分データ(基準色との差)を格納します
Y_Abs	Y の差分データ(基準色との差)を格納します
Z_Abs	Z の差分データ(基準色との差)を格納します
X_Ratio	Ev の差分データ(基準色との比率)を格納します
Y_Ratio	Ev の差分データ(基準色との比率)を格納します
Z_Ratio	Ev の差分データ(基準色との比率)を格納します

説明：
CLGetColorDifference()で差分データを扱う際に利用します。
X、Y、Z の差分データを格納します。

CL_Scotopic_DIFFDATA

概要：
Ev, 暗所視照度 Ev', S/P 値の差分データの構造体

形式：

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

変数：

引数	説明
Ev_Abs	Ev の絶対値差の演算結果を格納します
Ev_Ratio	Ev の比率差の演算結果を格納します
Es_Abs	暗所視照度 Ev'の絶対値差の演算結果を格納します
Es_Ratio	暗所視照度 Ev'の比率差の演算結果を格納します
SP	S/P 比の絶対値差の演算結果を格納します

CL_RANK

概要：
ランク選別時のランク情報の構造体

形式：

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

変数：

引数	説明
Ev	Ev の測定データを格納します
RankName	測定データの該当するランク名を格納します
RankNo	該当するランクのデータ No.を格納します

説明：
CLSortOutRank()で測定データのランク選別の結果を扱う際に利用します。

CL_RANK_DATA

概要：
ランク選別に利用するランクデータの構造体

形式：

```
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;
```

変数：

引数	説明
Type	ランク領域のポイントの指定方法(xy/相関色温度・⊿uv)を格納します 0：xy、1：相関色温度 Tcp・⊿uv
RankName	ランク名を格納します。 最大文字数：半角 41 文字(終端文字を含む)
xyPoint	ランク領域を構成する色度点の配列を格納します。 設定範囲 0.00 ＜ x ＜ 1.00、0.00 ＜ y ＜ 1.00
TcpduvPoint	ランク領域を構成する相関色温度・⊿uv の配列を格納します 設定範囲 2000 <= Tcp <= 50000、-0.03 <= ⊿uv <= 0.03
PointNum	ランク領域を構成するデータ数 設定範囲：3～10
Enable	ランク選別時の使用可否 0:使用しない、1：使用する

説明：
CLSetRankData()または CLGetRankData()でランクデータを扱う際に利用します。

ランク選別の際には、Enable で「使用する」と設定されているランクデータのみを使用してランク選別を実行します。

CL_xyDATA

概要：
ランクデータで利用する xy データの構造体

形式：

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

変数：

引数	説明
x	x の値を格納します 設定範囲：0.00 ＜ x ＜ 1.00
y	y の値を格納します 設定範囲：0.00 ＜ y ＜ 1.00

説明：
CLSetRankData()および CLGetRankData()でランクデータを扱う際に利用します。
ランクデータでランク領域を構成する各ポイントでの色度(x、y)の値を格納します。

CL_TcpduvDATA

概要：
ランクデータで利用する相関色温度 Tcp、⊿uv データの構造体

形式：

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

変数：

引数	説明
Tcp	相関色温度の値を格納します 設定範囲：2000～50000
duv	⊿uv の値を格納します 設定範囲：-0.03～0.03

説明：
CLSetRankData()および CLGetRankData()でランクデータを扱う際に利用します。
ランクデータでランク領域を構成する各ポイントでの相関色温度 Tcp と⊿uv の値を格納します。

CL_LISTDATA

概要：
本体の保存データの構造体

形式：

```
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;
```

変数：

引数	説明
Data	測定データを格納します(*1)
DateTime	保存日時を格納します
Observer	保存時の視野を格納します
OpCalibNo	利用しているユーザー校正 CH を格納します
TargetNo	利用している基準色データ No.を格納します
ExposureTime	測定時の測定時間を格納します

説明：
CLGetDeviceStoredData()で本体の保存データを扱う際に利用します。

(*1)
測定データは CLGetDeviceStoredData()の引数で指定した色彩データの種類のデータが格納されます。

CL_MEASSETTING

概要：
本体の測定条件の構造体

形式：

```
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;
```

変数：

引数	説明
DispType	本体画面の表示形式を格納します 0：絶対値表示、1：差分表示、2：ランク選別
Obs	本体での視野を格納します 0：2° 視野、1：10° 視野
ColorSpace	本体画面での表示モードを格納します 設定範囲：0～7(各値の詳細は CL_COLOR_MODE を参照してください)
IlluminantUnit	本体で利用する照度単位を格納します 0：lx
ExposureTime	本体での測定時間を格納します 0：FAST、1：SLOW、2：AUTO、3：SUPER FAST
UserCalibCh	本体のユーザー校正時に利用するユーザー校正係数のデータ No.を格納します 設定範囲：0～10
TargetNo	本体の差分測定で利用する基準色のデータ No.を格納します 設定範囲：1～20
UserCustom	本体のカスタム表色モードで表示する項目を格納します 設定範囲：0～33(各値の詳細は CL_CUSTOMDATA_ITEM を参照してください)
MeasMode	本体の測定モードの種類（シングル測定、平均化測定、連続測定）を設定します
TimerConf	本体のタイマー設定（遅延時間）を格納します 設定範囲：0-999[s]
UserWavelength	カスタム表色系任意波長設定を格納します 設定範囲：360-780[nm]

説明：
CLSetMeasSetting()または CLGetMeasSetting()で本体単体での測定時の測定条件を扱う際に利用します。

CL_TIMERCONF

概要：
タイマー設定データの構造体

形式：

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

変数：

変数	説明
DelaySec	遅延時間設定[s]を格納します 範囲: 0-999[s]
~~Interval~~	ver.1.2 では使用出来ません
~~MeasNum~~	ver.1.2 では使用出来ません

CL_USERWAVELENGTH

概要：
任意波長設定データの構造体

形式：

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

変数：

変数	説明
lambda	任意波長照度 1~4(CL_CUSTOM_EE_AB1~4)の波長 範囲: 360-780nm ピッチ：1nm

CL_DATETIME・CL_DEVDATETIME

概要：
日時情報を扱う構造体

形式：

```
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;
```

変数：

変数	説明
Year	年の値を格納します 設定範囲：2011～2100
Month	月の値を格納します 設定範囲：1～12
Day	日の値を格納します 設定範囲：1～31 (*1)
Hour	時の値を格納します 設定範囲：0～23
Minute	分の値を格納します 設定範囲：0～59
Second	秒の値を格納します 設定範囲：0～59

説明：
CLSetSystemSetting()または CLGetSystemSetting()で本体の日時を扱う際や、各種データ(基準色データ etc.)に付随する日時データを扱う際に利用します。

(*1)
設定範囲内の値でも日付として存在しない値(ex. 2013 年 2 月 29 日 etc.)を入力した場合には、各関数でエラーとなります。

CL_SYSTEMSETTING

概要：
本体のシステム設定を格納する構造体

形式：

```
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;
```

変数：

引数	説明
Datetime	日時設定を格納します 設定可能日時：2011/01/01 0:00:00 ～2100/12/31 23:59:59
Display	表示設定を格納します 0：通常表示、1：反転表示
Beep	ブザー音設定を格納します 0：ブザー音 OFF、1：ブザー音 ON
Lang	言語設定を格納します 0：英語、1：日本語、2：中国語
DateFormat	日時フォーマット設定を格納します 0：yyyy/mm/dd、1：mm/dd/yyyy、2：dd/mm/yyyy
UserCalLimit	ユーザー校正警告設定を格納します 0：3H、1：6H、2：12H、3：24H、4：警告なし
AutoPowerOff	オートパワーオフ設定を格納します 0：OFF、1：ON
PCalNotify	定期校正メッセージの表示設定を格納します 0：表示しない、1：表示する

説明：
CLSetSystemSetting()または CLGetSysytemSetting()で本体のシステム条件を扱う際に利用します。

CL_DEVID

概要：
本体情報の構造体

形式：

```
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;
```

変数：

変数	説明
Product	製品コードを格納します CL-500A では「A53C」が格納されます。
Factory	工場コードを格納します。 CL-500A では「A」が格納されます
Serial	本体のシリアル No.を格納します
Firmware	ファームウェアのバージョン情報を格納します
FactoryDate	工場校正日時を格納します
Variation	バリエーションコードを格納します CL-500A では「100」が格納されます。

説明：
CLGetDeviceID()で本体情報を取得する際に利用します。

CL_KEYINFO

概要：
本体のキー状態の構造体

形式：

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

変数：

引数	説明
MeasKey	測定キーのキー状態を格納します
UpKey	Up キーのキー状態を格納します
DownKey	Down キーのキー状態を格納します
BackKey	Back キーのキー状態を格納します
EnterKey	Enter キーのキー状態を格納します

説明：
CLGetButtonStatus()で本体のキー状態を取得する際に利用します。

キーが押されている場合には「CL_KEY_STATE_OFF(=1)」、キーが押されていない場合には「CL_KEY_STATE_OFF(=0)」が格納されます。

CL_SDKVERSION

概要：
バージョン情報の構造体

形式：

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

変数：

引数	説明
Main	インターフェースライブラリのバージョン情報を格納します
Calib	校正演算ライブラリのバージョン情報を格納します
Com	デバイス通信ライブラリのバージョン情報を格納します
Color	色彩演算ライブラリのバージョン情報を格納します

説明：
CLGetSDKVersion()で CL-SDK の各 DLL のバージョン情報をを取得する際に利用します。

4.4 API 一覧

各 API は以下のフォーマットで記載されています。

概要：
関数で実行できる処理について説明しています

形式：
関数の書式について説明しています

引数：
関数の引数について説明をしています

戻り値：
関数を利用した際に返ってくる戻り値について説明をしています
戻り値の種別として以下の 3 種類があります。

種別	
正常	処理に成功した際に返ってきます
エラー	処理に失敗した際に返ってきます
警告	処理には成功はしたが、ユーザーに対して情報を提供する必要がある際に返ってきます

説明：
関数を利用する際に必要な情報等や注意事項などを説明しています

CLOpenDevice()

概要：
PC に接続している測定器のオブジェクトハンドラを取得します

形式：
ER_CODE KMAPI CLOpenDevice(DEVICE_HANDLE* hDevice)

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	オブジェクトハンドラを取得できました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER_OPENDEVICE	エラー	13	オブジェクトハンドラの取得に失敗しました

説明：
PC に接続している測定器のオブジェクトハンドラを取得します。
複数の測定器が接続されている場合は、接続されている台数分だけ本関数を実行してください。
なお、複数台の測定器の制御方法は「§3.3　複数台制御」を参照してください。
本関数はオブジェクトハンドラを取得するのみです。

PC から測定器の制御を行うには、リモートモードにする必要がありますので、CLSetRemoteMode()関数でリモートモードを ON にしてください。

CLCloseDevice()

概要：
CL-SDK で確保している CL-500A のオブジェクトハンドラを解放します

形式：
ER_CODE KMAPI CLCloseDevice(DEVICE_HANDLE hDevice)

引数：

引数	I/O	説明
hDevice	I	解放するオブジェクトハンドラ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	オブジェクトハンドラを解放しました
ER_HANDLENULL	エラー	10	オブジェクトハンドラが存在しません

説明：
CL-SDK で確保している CL-500A のオブジェクトハンドラを解放します。
本関数を実行しない限り、アプリケーション側での測定器の制御可能な状態になります。

[注意]
本関数はオブジェクトハンドラを解放するのみですので、本体単独での操作を可能にするためには本関数の実行前に CLSetRemoteMode()でリモートモードを OFF にしてください。

CLSetRemoteMode()

概要：
リモートモードの ON/OFF を設定します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
Mode	I	リモートモードの ON/OFF の指定 0：リモートモード OFF 1：リモートモード ON

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	リモートモードを設定しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER00301	エラー	301	引数 Mode が NULL 値になっています
ER00302	エラー	302	設定値が不正な値です
ER00304	エラー	304	リモートモードの設定に失敗しました
ER00305	エラー	305	校正データの取得に失敗しました(*1)
ER00306	エラー	306	ランクデータの取得に失敗しました(*1)
ER00307	エラー	307	本体の測定時間設定の取得に失敗しました
ER00308	エラー	308	リモートモードの OFF に失敗しました
ER00309	エラー	309	ユーザー校正データの取得に失敗しました(*1)

説明：
リモートモードの ON/OFF を設定します。
PC から測定器を制御する場合には、必ずリモートモードを ON にしてください。
リモートモードが ON になると、本体の操作はできません。
また、リモートモードが ON になると本体画面には「通信中」と表示されます。

(*1)
リモートモードが ON になったら、演算時に利用する校正データなどを本体より取得します。そのデータの取得の際にエラーが発生すると、戻り値が「ER00305」、「ER00306」または「ER00309」になります。

CLGetRemoteMode()

概要：
リモートモードの ON/OFF を取得します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
Mode	I	取得するリモートモードの設定値を格納するバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	リモートモードを取得しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER00301	エラー	301	引数 Mode が NULL 値になっています
ER00303	エラー	303	リモートモードの取得に失敗しました

説明：
リモートモードの ON/OFF を取得します。

CLDoMeasurement()

概要：
測定を実行します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
Time	O	取得する測定時間を格納するバッファ 単位：100μ秒 ex.） 取得したTimeの値が2500の場合 2500×100 = 250000 μs = 0.25 秒

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	本体が測定を開始しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER01301	エラー	1301	ゼロ校正がされていません
ER01302	エラー	1302	リモートモードがOFFになっています
ER01303	エラー	1303	引数pTimeがNULL値になっています
ER01304	エラー	1304	校正状況の取得に失敗しました
ER01305	エラー	1305	測定実行に失敗しました
ER01306	エラー	1306	ゼロ校正がされていません
ER01307	エラー	1307	プレ露光でエラーになりました
WARNING	警告	1	最終校正から一定期間が経過しています。ゼロ校正を行ってください

説明：
測定を実行します。
Timeに本体でのおおよその測定時間を格納しますので、測定完了までの目安時間として確認してください。
測定時間がAUTOの場合には測定光源によって測定時間が異なります(最大約27秒)ので、Timeで取得する値は測定光源ごとに異なります。
測定時間がFAST、SLOWおよびSUPER_FASTの場合は露光時間が決まっているので、FASTの場合は約0.5秒、SLOWの場合は約2.5秒、SUPER_FASTの場合は約0.2秒に相当する値を取得します。

CLDoManualMeasurement()

概要：
露光時間と積算回数を任意に指定して測定を実行します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
eTime	I	指定する露光時間パラメータを指定するバッファ [x 100us]
cNumber	I	指定する積算回数パラメータを指定するバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	本体が測定を開始しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER01321	エラー	1321	校正が実行されていません
ER01322	エラー	1322	リモートモードが OFF になっています
ER01324	エラー	1324	校正状態の取得に失敗しました
ER01325	エラー	1325	測定実行に失敗しました
		1326	この機能に対応してない本体が接続されています。 (本体ファーム ver.1.20 以降対応)
ER01328	エラー	1328	露光時間の指定値が不正な値です
ER01329	エラー	1329	積算回数の指定値が不正な値です
ER01330	エラー	1330	露光時間と積算回数の組み合わせが不正です
ER01331	エラー	1331	露光時間と積算回数の指定に失敗しました
WARNING	警告	1	最終校正から一定期間が経過しています。校正を行ってください

説明：
マニュアル測定使用時には、露光時間と積算回数を指定してください。
露光時間と積算回数の組み合わせは以下の中から選んでください。(下表の照度範囲は、A 光源使用時の目安です)

露光時間 x 積算回数組み合わせ表

*照度範囲 [lx]	露光時間[x 100us]	積算回数	測光時間 [ms]
10000 ~ 100000	40	100	400
1000 ~ 10000	400	10	400
100 ~ 1000	4000	1	400
10 ~ 20	20000	1	2000
10 以下	50000	1	5000
	50000	2	10000
	50000	4	20000

CLPollingMeasure()

概要：
測定状況を取得します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
pStatus	O	取得する測定状況を格納するバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	測定状況を取得しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER_ALLMEASUREING	エラー	16	全測定器測定が実行中です
ER01201	エラー	1201	引数 pStatus が NULL 値になっています
ER01202	エラー	1202	リモートモードが OFF になっています
ER01203	エラー	1203	測定に失敗しました
ER01204	エラー	1204	測定可能範囲を上回っています(*1)
ER01205	エラー	1205	測定状況の取得に失敗しました
ER01206	エラー	1206	測定に失敗しました(*2)

説明：
測定状況を確認します。
取得した status の値が「測定完了」となってから、CLGetMeasData()関数や CLSortOutRank()関数で色彩値の取得やランク選別を行ってください。
status が「測定完了」となるのは、測定完了後の最初の status 取得時のみです(一度「測定完了」を取得すると次の status 取得時には「未測定」となります)。

(*1)
CL-500A で測定可能な範囲を超えている場合、CL-SDK からの戻り値は「ER01204」となります。
その場合には、測定している光源からの距離を離すか、ND フィルターを使用して光源の光量を落としてから測定を行って下さい。

(*2)
* 測定中に測定対象物の明るさが測定開始時より明るくなった場合にはエラーとなり、CL-SDK からの戻り値は「ER01206」となります。
測定時は測定完了まで測定対象物の明るさを一定に保つようにしてください。
また、CL-500A で測定可能な範囲を超えている場合にも、CL-SDK からの戻り値は「ER01206」となります。
その場合には、測定している光源からの距離を離すか、ND フィルターを使用して光源の光量を落としてから測定を行って下さい。
(*2)
マニュアル測定時に指定した露光時間で測定可能な範囲を超えている場合にも、CL-SDK からの戻り値は「ER01206」となります。
その場合には、露光時間を短くしてから測定をおこなって下さい。

CLStopMeasurement()

概要：
測定を中断します

形式：
ER_CODE KMAPI CLStopMeasurement(DEVICE_HANDLE hDevice)

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	測定を中断しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER01401	エラー	1401	リモートモードが OFF になっています
ER01402	エラー	1402	測定中断に失敗しました

説明：
測定を中断します。
測定時間が AUTO の場合など、測定時間が長い場合に測定を中断したい場合に利用してください。

なお、測定が実行されていない状態で本関数を実行した場合は、戻り値として「SUCCESS」が戻ってきます。

CLDoMeasurementAll()

概要：
接続されている全ての測定器に対して測定を実行します

形式：
ER_CODE KMAPI CLDoMeasurementAll(int32_km *pTime)

引数：

引数	I/O	説明
pTime	O	取得する測定時間を格納するバッファ 単位：100μ秒 ex.） 取得した Time の値が 2500 の場合 2500×100 = 250000 μs = 0.25 秒

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	測定を開始しました
ER10801	エラー	10801	オブジェクトハンドラが存在しません
ER10802	エラー	10802	ゼロ校正ができていない測定器が存在します
ER10803	エラー	10803	リモートモードが OFF になっている測定器が存在します
ER10804	エラー	10804	校正状態の取得に失敗しました
ER10805	エラー	10805	測定の実行に失敗しました
ER10806	エラー	10806	測定時間の取得に失敗しました
ER10807	エラー	10807	リモートモードが OFF になっている測定器が存在します

説明：
PC に接続されている全ての測定器で測定を実行します(測定を実行する測定器を選択することはできません)。
取得する測定時間は全ての測定器での測定時間が決定した後、全ての測定器の中で最大の測定時間を格納します。
測定器は全てリモートモードを ON にしておいて下さい。また、ゼロ校正が完了していない測定器が 1 台でも存在するとエラーとなるので、実行前に必ずゼロ校正を完了させておいて下さい。

CLDoManualMeasurementAll()

概要：
接続されている全ての測定器に対して露光時間と積算回数を指定して測定を実行します

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

引数：

引数	I/O	説明
eTime	I	露光時間 [x 100us]
cNumber	I	積算回数 [回]

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	本体が測定を開始しました
ER10821	エラー	10821	オブジェクトハンドラが存在しません
ER10822	エラー	10822	校正されていない測定器が存在します
ER10823	エラー	10823	リモートモードが OFF になっている測定器が存在します
ER10824	エラー	10824	校正状態の取得に失敗しました
ER10825	エラー	10825	測定の実行に失敗しました
ER10826	エラー	10826	この機能に対応していない測定器が存在します。 (本体ファーム ver.1.20 以降対応)
ER10827	エラー	10827	リモートモードが OFF になっている測定器が存在します
ER10828	エラー	10828	露光時間の指定値が不正な値です
ER10829	エラー	10829	積算回数の指定値が不正な値です
ER10830	エラー	10830	露光時間と積算回数の組み合わせが不正です
ER10831	エラー	10831	露光時間と積算回数の指定に失敗しました

説明：
PC に接続されている全て測定器で測定を実行します(測定を実行する測定器を選択することはできません)。
測定器は全てリモートモードを ON にしておいて下さい。また、校正が完了していない測定器が 1 台でも存在するとエラーとなるので、実行前に必ず校正を完了させておいて下さい。
マニュアル測定使用時には、露光時間と積算回数を指定してください。
露光時間と積算回数の組み合わせは以下の中から選んでください。(下表の照度範囲は、A 光源使用時の目安です)

露光時間 x 積算回数組み合わせ表

*照度範囲 [lx]	露光時間[x 100us]	積算回数	測光時間 [ms]
10000 ~ 100000	40	100	400
1000 ~ 10000	400	10	400
100 ~ 1000	4000	1	400
10 ~ 20	20000	1	2000
10 以下	50000	1	5000
	50000	2	10000
	50000	4	20000

CLPollingMeasureAll()

概要：
CLDoMeasurementAll()による測定での測定状況を取得します

形式：
ER_CODE KMAPI CLPollingMeasureAll(CL_MEASSTATUS *pStatus)

引数：

引数	I/O	説明
pStatus	I	取得する測定状況を格納するバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	測定状況を取得しました
ER10901	エラー	10901	測定実行時の測定器と接続されている測定器の数が異なります
ER10902	エラー	10902	リモートモードが OFF になっている測定器が存在します
ER10903	エラー	10903	測定可能範囲を上回っています(*1)
ER10904	エラー	10904	測定状況の取得に失敗しました

説明：
CLDoMeasurementAll()による測定での測定状況を確認します。
1 台でも測定中の測定器が存在する限り、status で取得できる値は測定中(CL_MEAS_BUSY)になります。
status が測定完了(CL_MEAS_FINISH)になるのは、全ての測定器での測定が完了した後です。
取得した status の値が測定完了となってから、CLGetMeasData()関数や CLSortOutRank()関数で色彩値の取得やランク選別を行ってください。
全測定器測定中は個々の測定器の測定状況を取得することはできません(CL_PollingMeasure()関数は使用できません)。

(*1)
測定中に測定対象物の明るさが測定開始時より明るくなった場合にはエラーとなり、CL-SDK からの戻り値は「ER10903」となります。
測定時は測定完了まで測定対象物の明るさを一定に保つようにしてください。
また、CL-500A で測定可能な範囲を超えている場合にも、CL-SDK からの戻り値は「ER10903」となります。
その場合には、測定している光源からの距離を離すか、ND フィルターを使用して光源の光量を落としてから測定を行って下さい。
(*2)
マニュアル測定時に指定した露光時間で測定可能な範囲を超えている場合にも、CL-SDK からの戻り値は「ER10903」となります。
その場合には、露光時間を短くしてから測定をおこなって下さい。

CLStopMeasurementAll()

概要：
CLDoMeasurementAll()による測定を中断します

形式：
ER_CODE KMAPI CLStopMeasurementAll()

引数：

なし

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	全測定器の測定を中断しました

説明：
CLDoMeasurementAll()による測定を中断します。
測定時間が AUTO の場合など、測定時間が長い場合に測定を中断したい場合に利用してください。
全測定器測定での測定中断の際は、この関数を利用して下さい。通常の CLStopMeasurement()関数では測定を中断することはできません。

CLDoCalbiration()

概要：
ゼロ校正を実行します

形式：
ER_CODE KMAPI CLDoCalibration(DEVICE_HANDLE hDevice)

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	ゼロ校正を実行しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER11101	エラー	11101	ゼロ校正に失敗しました
ER11102	エラー	11102	リモートモードが OFF になっています

説明：
ゼロ校正を実行します。
ゼロ校正を実行する際は、標準付属品の校正キャップを受光部に装着した状態でゼロ校正を行ってください。
ゼロ校正エラーになった場合は、再度ゼロ校正を実行してください。
また、ゼロ校正には約 27 秒かかります。
ゼロ校正が完了したかどうかは、CLPollingCalibration()で確認して下さい。

なお、ゼロ校正は測定値の精度を保つために 1 日に 1 回は行うようにして下さい。

CLPollingCalibration()

概要:
ゼロ校正の実行状況を取得します

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

引数:

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
pStatus	O	取得する校正状況を格納するバッファ

戻り値:

戻り値	種別	値	説明
SUCCESS	正常	0	ゼロ校正の実行状況を取得しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER11201	エラー	11201	引数 pStatus が NULL 値になっています
ER11202	エラー	11202	ゼロ校正に失敗しました
ER11203	エラー	11203	デバイス異常が発生しています(*1)
ER11204	エラー	11204	リモートモードが OFF になっています

説明:
ゼロ校正の実行状況を確認します。
ゼロ校正には約 27 秒かかります。
取得した status の値が「校正完了」となったら、ゼロ校正が完了しています。
status が「校正完了」となるのは、校正完了後の最初の実行状況取得時のみです(一度「校正完了」を取得すると次の status 取得時には「未校正」となります)。

(*1)
デバイス異常が発生していますので、本体の修理が必要です。お近くのサービスまでお問い合わせください。

CLGetCalibrateStatus()

概要：
校正状況を取得します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
Status	O	取得する校正状況のバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	校正状況を取得しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER01101	エラー	1101	引数 Status が NULL 値になっています
ER01102	エラー	1102	リモートモードが OFF になっています
ER01103	エラー	1103	校正状況の取得に失敗しました

説明：
校正状況を取得します。
未完了の場合は測定が実行できませんので、ゼロ校正を実行してください。
要校正の場合には測定することは実行できますが、最後のゼロ校正から一定時間が経過していますので
測定精度を保証するためにゼロ校正することをお勧めします。
ゼロ校正は１日に１回は行うようにして下さい。

CLGetMeasData()

概要：
最新の測定での各種色彩値データを取得します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
Type	I	取得する色彩値の種類
pColor	O	取得する色彩値データを格納するバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	色彩値を演算しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER03702	エラー	3702	色彩値演算に失敗しました
ER03703	エラー	3703	測定が未実施です
ER03704	エラー	3704	リモートモードが OFF になっています
ER03705	エラー	3705	引数 pColor が NULL 値になっています
WARNING	警告	1	低照度値です(保証外の値です)(*1)
WARNING	警告	1	値を演算できなかった項目が存在します(*2)
WARNING	警告	1	マニュアル測定で指定した露光時間が適正範囲を超えています(保証外の値です) 警告値:6
WARNING	警告	1	マニュアル測定で指定した露光時間が適正範囲を下回っています(保証外の値です) 警告値:7

説明：
最新の測定での各種色彩値データを取得します。
取得できる色彩値データは、最新の測定データに対してのみです。
各測定での測定データが必要な場合は、次の測定前に必ず測定データを取得してください。
なお、利用するユーザー校正係数は本体に設定されている CH のユーザー校正係数です。

(*1)
測定データの照度値がある値以下の場合、低照度値警告となります。演算結果は出力しますが、CL-500A としては保証外ですので、ご注意ください。
低照度警告が出る照度値は測定時間によって異なります。各測定時間での低照度警告がでる閾値は以下の通りです。
FAST モード：50[lx]、SLOW モード：10[lx]、SUPERFAST モード：100[lx]
(*2)
測定光源によっては、相関色温度などの値が演算できない場合があります。その場合は、測定データとして「CL_INCOMPUTABLE_VALUE(-11000)」が格納されます。

[補足]
測定データが補色主波長の場合には、測定データはマイナス値になります。
各測定データの有効桁数は 4 桁となっていますので、ご注意ください。

CLGetColorDifference()

概要：
最新の測定での各種差分データを取得します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
Type	I	取得する差分データの種類
pDiff	O	取得する差分データのバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	差分演算が完了しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER04701	エラー	4701	基準色データの取得に失敗しました
ER04702	エラー	4702	測定が未実施です
ER04703	エラー	4703	基準色データの設定に失敗しました
ER04704	エラー	4704	測定データの設定に失敗しました
ER04705	エラー	4705	差分演算に失敗しました
ER04706	エラー	4706	リモートモードが OFF になっています
ER04707	エラー	4707	引数 pDiff が NULL 値になっています

説明：
最新の測定での各種差分データを取得します。
取得できる差分データは、最新の測定データに対してのみです。
各測定でのデータが必要な場合は、必ず測定前にデータを取得してください。

本関数で利用する基準色データは、制御している CL-500A で差分測定を行う際に利用する基準色データです。
したがって、本関数を利用する前に予め本体に基準色データの登録と、利用する基準色を設定しておいてください。

[補足]
主波長の差分計算は「測定データ － 基準色」で求めています。
したがって、片方が主波長で一方が補色主波長の場合も同様です。
ex.)測定データが補色主波長(562nm)で、基準色が主波長(568nm)の場合
⊿主波長 = -562 - 568 = -1130 [nm]

CLSortOutRank()

概要：
測定データのランクを選別します。

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

引数：

引数	I/O	説
hDevice	I	制御対象の測定器のオブジェクトハンドラ
Rank	I	取得するランク情報の構造体

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	ランクの選別を行いました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER08101	エラー	8101	測定データの取得に失敗しました
ER08102	エラー	8102	ランクの選別に失敗しました
ER08103	エラー	8103	リモートモードが ON ではありません
ER08104	エラー	8104	引数 Rank が NULL 値になっています
ER08105	エラー	8105	ランクリストが存在しません
WARNING	警告	1	どのランクにも該当しませんでした

説明：
測定データのランクを選別します。
CL-SDK でのランク選別では、本体に登録されているランクリストを利用します。
したがって、ランク選別を行う前に CLSetRankData()を利用してランクを登録してください。

また、ランクの選別は登録されているリスト No.の小さいランクから判別します。したがって、測定データの値がランク領域の重なっている箇所のポイントだった場合には、ランクリストのリスト No.の小さい方のランクとして選別されます。

測定データがどのランクにも該当しなかった場合には戻り値として警告値が返り、変数にも該当ランクなしという意味の「N/A」が格納されます。

CLSetRankData()

概要：
ランク選別に利用するランクデータを本体に登録します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
DataNo	I	ランクデータを登録するデータ No. 設定範囲：1～20
RankData	I	設定するランクデータ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	ランクデータを設定しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ERO6601	エラー	6601	引数 RankData が NULL 値になっています
ERO6602	エラー	6602	リモートモードが OFF になっています
ERO6603	エラー	6603	設定するランク No.が範囲外です
ERO6604	エラー	6604	ランクデータの登録に失敗しました
ERO6605	エラー	6605	ランク領域の設定が正しくありません(*1)
ERO6606	エラー	6606	xy への変換に失敗しました

説明：
ランクデータは CL_RANK_DATA 構造体として登録します。
ランク領域は配列に格納した順にポイントを結んでいき、領域として分割されていないかをチェックします。分割されている場合は領域として適切ではないと判断し、エラーとなります。
詳細は「§5.1 ランク領域の設定」をご覧ください。
指定したリスト No.にすでにランクデータが存在する場合、上書き保存されますのでランクが登録されているかどうかを確認の上、本関数を実行してください。

(*1)
登録するランク領域が分割されている場合はエラーとなります。

ランク名は半角文字で最大 40 文字(全角文字なら最大 20 文字)まで設定可能です。また、ランク名には全角文字と半角文字が混在させることができます。
ただし、本体側での表示可能な最大文字数は半角 15 文字ですので、ランク名が 15 文字以上の場合は本体には 15 文字目までが表示されます。また全角文字で本体が表示できない文字の場合には「・」と表示されます。

CLGetRankData()

概要：
本体に登録されているランクデータを取得します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
DataNo	I	取得するランクデータのデータ No. 設定範囲：1～20
RankData	I/O	取得する基準色データを格納するバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	ランクデータを取得しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER08201	エラー	8201	ランクデータの取得に失敗しました
ER08202	エラー	8202	リモートモードが OFF になっています
ER08203	エラー	8203	ランクデータが存在しません
ER08204	エラー	8204	ランクデータが登録されていません
ER08205	エラー	8205	指定したデータ No.が範囲外です

説明：
本体に登録されているランクデータを取得します。
本関数は No.で指定した番号のランクデータを取得します。本体に登録されている全てのランクデータを取得する場合は、No1～20 の値を指定して繰り返してください。

CLDeleteRankData()

概要：
本体に登録されているランクデータを削除します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
DataNo	I	削除するランクデータのデータ No. -1:全データ削除 1～20：指定したデータ No.

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	ランクデータを削除しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER06701	エラー	6701	指定したデータ No.が範囲外です
ER06702	エラー	6702	ランクデータの削除に失敗しました
ER06703	エラー	6703	ランクデータの削除に失敗しました
ER06704	エラー	6704	リモートモードが OFF になっています

説明：
本体に登録されているランクデータを削除します。

DataNo に「-1」を指定すると全てのランクデータを削除します。

CLSetTargetData()

概要：
本体に基準色を登録します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
DataNo	I	基準色を登録するデータ No. 設定範囲：1～20
pTarget	I	設定する基準色

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	基準色を登録しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER05401	エラー	5401	引数 pData が NULL 値になっています
ER05402	エラー	5402	指定したデータ No.が不正な値です
ER05404	エラー	5404	リモートモードが OFF になっています
ER05405	エラー	5405	日時データに誤りがあります
ER05406	エラー	5406	データ名に誤りがあります
ER05407	エラー	5407	基準色の登録に失敗しました

説明：
測定器に基準色を登録します。

[注意]
CL-SDK では本関数を実行すると、指定したデータ No.に対して基準色を書き込みます。
指定したデータ No.に既にデータが存在するかどうかを確認しませんので、上書き確認が必要な場合はアプリケーション側でチェックを行ってください。

CLGetTargetData()

概要：
本体に登録されている基準色を取得します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
DataNo	I	取得する基準色のデータ No. 設定範囲：1～20
Data	I/O	取得する基準色を格納するバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	基準色を取得しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER05501	エラー	5501	引数 pStatus が NULL 値になっています
ER05502	エラー	5502	指定したデータ No.が範囲外です
ER05503	エラー	5503	リモートモードが OFF になっています
ER05504	エラー	5504	基準色の取得に失敗しました
ER05505	エラー	5505	基準色情報の取得に失敗しました
ER05506	エラー	5506	指定したデータ No.には基準色が存在しません

説明：
本体に登録されている基準色を取得します。

CLDeleteTargetData()

概要：
本体に登録されている基準色を削除します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
DataNo	I	削除する基準色のデータ No. -1：全データ削除 1～20：基準色のデータ No.

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	基準色を削除しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER05601	エラー	5601	指定したデータ No.が不正な値です
ER05602	エラー	5602	基準色の削除に失敗しました
ER05603	エラー	5603	リモートモードが OFF になっています

説明：
本体に登録されている基準色を削除します。

DataNo に「-1」を指定すると全ての基準色を削除します。
指定したデータ No.に基準色が存在しない場合、エラー(ER05602)になります。

CLGetDeviceStoredDataNum()

概要：
本体に保存されている測定データの数を取得します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
Num	O	取得する測定データ数を格納するバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	本体の保存データ数を取得しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER05901	エラー	5901	引数 Num が NULL 値になっています
ER05902	エラー	5902	リモートモードが OFF になっています
ER05903	エラー	5903	本体の保存データ数の取得に失敗しました

説明：
本体に保存されている測定データ数を取得します。
本体に保存できるデータ数は最大 100 データです。
本体に測定データが 1 つも保存されていない場合は、0 を取得します。
取得したデータ数を元に CLGetDeviceStoredData()で保存データを取得するためのバッファを確保してください。

CLGetDeviceStoredData()

概要：
本体に保存されている測定データを取得します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
pData	O	取得する保存データを格納するバッファ
Type	I	取得する色彩データの種類
Length	I	バッファのサイズ(*1)

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	保存データを取得しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER05801	エラー	5801	引数 pData が NULL 値になっています
ER05802	エラー	5802	リモートモードが OFF になっています
ER05803	エラー	5803	保存データ数の取得に失敗しました
ER05804	エラー	5804	バッファのサイズが小さいです
ER05805	エラー	5805	保存データの取得に失敗しました
ER05806	エラー	5806	保存データの取得に失敗しました
ER05807	エラー	5807	データの設定に失敗しました
ER05808	エラー	5808	色彩値演算に失敗しました
ER05809	エラー	5809	本体に保存データが 1 つも存在しません

説明：
本体に保存されている測定データは一括で取得します。本体に保存されている測定データの個数を確認したい場合は、別途 CLGetDeviceStoredDataNum()関数で保存データ数を取得してください。

(*1)
バッファのサイズとして、取得するデータ数を指定して下さい。
ex.)取得するデータ数が 25 データなら、Length には「25」を指定して下さい。

(*2)
測定光源によっては、相関色温度などの値が演算できない場合があります。その場合は、測定データとして「CL_INCOMPUTABLE_VALUE(-11000)」が格納されます。

[注意]
本体には最大 100 データを保存することができます。データ数が多いとデータ取得に時間がかかりますので、ご注意ください(100 データの場合で約 7 秒)。

CLDeleteDeviceStoredData()

概要：
本体に保存されている測定データを削除します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
DataNo	I	削除する保存データの No. -1：全データの削除 1～100：指定したデータ No. (*1)

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	測定データを削除しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER08901	エラー	8901	指定したデータ No.が不正な値です
ER08902	エラー	8902	保存データの削除に失敗しました
ER08903	エラー	8903	保存データの削除に失敗しました
ER08904	エラー	8904	リモートモードが OFF になっています
ER08905	エラー	8905	測定データ数の取得に失敗しました
ER08906	エラー	8906	指定したデータ No.に測定データが存在しません

説明：
本体に保存されている測定データを削除します。

(*1)
指定したデータ No.の保存データが存在しない場合は、エラーとなります。

[注意]
本体の保存データを削除すると、削除したデータ以降の保存が繰り上がるため、保存データ No.が変わります。引き続いて保存データを削除する場合は削除するデータ No.の指定にご注意ください。

CLSetUserCalibraitonData()

概要：
本体にユーザー校正係数を登録します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
DataNo	I	ユーザー校正係数を登録するデータ No. 設定範囲：1～10
pCoef	I/O	登録するユーザー校正係数のバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	ユーザー校正係数を登録しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER02101	エラー	2101	指定したデータ No.が不正な値です
ER02102	エラー	2102	登録するユーザー校正係数の値に不正な値があります
ER02103	エラー	2103	リモートモードが OFF になっています
ER02104	エラー	2104	設定する日時に誤りがあります
ER02105	エラー	2105	引数 pCoef が NULL 値になっています
ER02105	エラー	2106	設定するデータ名に誤りがあります
ER02107	エラー	2107	ユーザー校正係数の登録に失敗しました
ER02108	エラー	2108	ユーザー校正係数の更新に失敗しました

説明：
本体で利用するユーザー校正係数を登録します。

すでにユーザー校正係数が登録されているデータ No.に対してユーザー校正係数を登録する場合は上書き登録します。

[注意]
CL-SDK では本関数を実行すると、指定したデータ No.に対してユーザー校正係数を書き込みます。
指定したデータ No.に既にデータが存在するかどうかを確認しませんので、上書き確認が必要な場合はアプリケーション側でチェックを行ってください。

CLGetUserCalibraitonData()

概要：
本体に登録されているユーザー校正係数を取得します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
DataNo	I	取得するユーザー校正係数のデータ No. 設定範囲：1～10
pCoef	O	取得するユーザー校正係数を格納するバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	ユーザー校正係数を取得しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER02201	エラー	2201	指定したデータ No.が不正な値です
ER02202	エラー	2202	引数 pCoef が NULL 値になっています
ER02203	エラー	2203	リモートモードが OFF になっています
ER02204	エラー	2204	ユーザー校正係数の取得に失敗しました
ER02205	エラー	2205	指定した CH のデータ確認に失敗しました
ER02206	エラー	2206	指定した CH にデータが存在しません

説明：
本体に登録されているユーザー校正係数を取得します。

CLDeleteUserCalibrationData()

概要：
本体に登録されているユーザー校正係数を削除します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
DataNo	I	削除するユーザー校正 CH のデータ No. -1：全データ削除 1～10：指定した校正 CH のデータ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	ユーザー校正係数を削除しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER02301	エラー	2301	指定したデータ No.が範囲外です
ER02302	エラー	2302	リモートモードが OFF になっています
ER02303	エラー	2303	ユーザー校正データの削除に失敗しました
ER02304	エラー	2304	ユーザー校正係数の更新に失敗しました

説明：
本体に登録されているユーザー校正係数を削除します。

DataNo に「-1」を指定すると、全ての CH のデータを削除します。

CLCalcUserCalibrationData()

概要：
ユーザー校正係数を算出します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
Output	O	算出したユーザー校正係数を格納するバッファ バッファサイズ：421 データ
Target	I	ユーザー校正係数を算出するために利用する基準色データ データ数：421 データ
Sample	I	ユーザー校正係数を算出するために利用する測定データ データ数：421 データ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	ユーザー校正係数を算出しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER02401	エラー	2401	算出したユーザー校正係数の値が範囲外の値です
ER02402	エラー	2402	引数 output が NULL 値になっています
ER02403	エラー	2403	引数 Target が NULL 値になっています
ER02404	エラー	2404	引数 sample が NULL 値になっています

説明：
ユーザー校正係数を算出します。
算出したユーザー校正係数は、本体に登録することが可能です。
本体にユーザー校正係数を登録する際は、CLSetUserCalibrationData()関数を利用してください。
ユーザー校正係数は算出した値が 0.001～1000.000 でない場合にはエラー(ER02401)となります。

CLSetProperty()

概要：
CL-SDK での測定データの演算時の測定条件を設定します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
Type	I	設定するプロパティの種類
pParam	I	設定するプロパティの設定値

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	設定が完了しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER00501	エラー	501	視野の設定値が不正な値です
ER00502	エラー	502	照度単位の設定値が不正な値です
ER00503	エラー	503	リモートモードが OFF になっています
ER00504	エラー	504	プロパティの種類の値が不正な値です

説明：
CL-SDK で測定器を制御している場合の測定時の測定条件を変更します(制御している測定器単体での測定時の測定条件は変更しません)。
また、測定器を切断すると、CL-SDK 内部で保持している設定情報は失われます。
必要なら作成したアプリケーション側で設定情報を記憶してください。

[注意]
本関数で設定した条件は次の測定データから反映されます。
制御している測定器単体での測定時の測定条件は、CLSetMeasSetting()を利用して設定して下さい。

CLGetProperty()

概要：
CL-SDK での測定データの演算時の測定条件を取得します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
Type	I	取得するプロパティの種類
pParam	O	取得するプロパティ値を格納するバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	設定を取得できました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER00401	エラー	401	引数 Param が NULL 値になっています
ER00402	エラー	402	プロパティの種類の値が不正な値です
ER00403	エラー	403	リモートモードが OFF になっています

説明：
CL-SDK で測定器を制御している場合の測定時の測定条件を取得します(制御している測定器単体での測定時の測定条件は取得しません)。

[注意]
制御している測定器単体での測定時の測定条件は、CLGetMeasSetting()を利用して取得して下さい。

CLGetButtonStatus()

概要：
本体キーの状態を取得します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
pStatus	O	取得するキー状態を格納するバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	キー状態を取得しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER00801	エラー	801	pStatus が NULL 値になっています
ER00802	エラー	802	リモートモードが OFF になっています
ER00803	エラー	803	キー状態の取得に失敗しました

説明：
本体キーの状態を取得します。
本体キーの状態を取得することで、本体キーをトリガーにした測定を実行することが可能です。
本体キーをトリガーにした測定については「§3.4　本体キーをトリガーにした測定」を参照してください。

CLSetMeasSetting()

概要：
測定器単体での測定時の測定条件を設定します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
Type	I	設定する測定条件の種別
pSettings	I	設定する測定条件

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	測定条件を設定しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER_PARAM_NULL	エラー	14	引数 pSettings が NULL 値になっています
ER01001	エラー	1001	引数 pSettings が NULL 値になっています
ER01002	エラー	1002	リモートモードが OFF になっています
ER01003	エラー	1003	測定条件の種別の値が不正な値です
ER07201	エラー	7201	表示形式の設定値が不正な値です
ER07203	エラー	7203	表示形式の設定に失敗しました
ER07301	エラー	7301	視野の設定値が不正な値です
ER07303	エラー	7303	視野の設定に失敗しました
ER07401	エラー	7401	表色モードの設定値が不正な値です
ER07403	エラー	7403	表色モードの設定に失敗しました
ER07501	エラー	7501	照度単位の設定値が不正な値です
ER07503	エラー	7503	照度単位の設定に失敗しました
ER07601	エラー	7601	測定時間の設定値が不正な値です
ER07603	エラー	7603	測定時間の設定に失敗しました
ER07701	エラー	7701	ユーザー校正 CH の設定値が不正な値です
ER07703	エラー	7703	指定したユーザー校正 CH にはデータが存在しません
ER07704	エラー	7704	ユーザー校正 CH の設定に失敗しました
ER07705	エラー	7705	ユーザー校正係数の更新に失敗しました
ER07801	エラー	7801	基準色データ No.の設定値が不正な値です
ER07803	エラー	7803	基準色データ No.の設定に失敗しました
ER07901	エラー	7901	カスタム表色モードの設定値が不正な値です
ER07903	エラー	7903	カスタム表色モードの設定に失敗しました
ER15001	エラー	15001	測定モードの設定値が不正な値です
ER15003	エラー	15003	測定モードの設定に失敗しました
ER15004	エラー	15004	測定モード設定機能に対応してない本体が接続されています。 （本体ファーム ver.1.20 以降対応）
ER15103	エラー	15103	タイマーの設定に失敗しました 範囲:0-999[s]

ER15104	エラー	15104	タイマー設定機能に対応してない本体が接続されています。 (本体ファーム ver.1.20 以降対応)
ER15201	エラー	15201	タイマー(遅延時間)の設定値が不正な値です
ER15503	エラー	15503	任意波長設定の設定に失敗しました
ER15504	エラー	15504	任意波長設定機能に対応してない本体が接続されています。 (本体ファーム ver.1.20 以降対応)

説明：
測定器単体での測定時の測定条件を設定します(CL-SDK で測定器を制御している場合の測定時の測定条件は変更しません)。
項目ごとに設定するので、設定したい項目を「Type」で指定して下さい。

必要なら作成したアプリケーション側で設定情報を記憶してください。

[注意]
CLGetMeasData()で取得する測定データの測定条件は、CLSetProperty()で設定してください。

CLGetMeasSetting()

概要：
測定器単体での測定時の測定条件を取得します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
Type	I	設定する測定条件の種別
pSettings	O	取得する測定条件の値を格納するバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	測定条件を取得しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER01001	エラー	1001	引数 pSettings が NULL 値になっています
ER01002	エラー	1002	リモートモードが OFF になっています
ER01003	エラー	1003	測定条件の種別の値が不正な値です
ER07202	エラー	7202	表示形式設定の取得に失敗しました
ER07302	エラー	7302	視野設定の取得に失敗しました
ER07402	エラー	7402	表色モード設定の取得に失敗しました
ER07502	エラー	7502	照度単位設定の取得に失敗しました
ER07602	エラー	7602	測定時間設定の取得に失敗しました
ER07702	エラー	7702	ユーザー校正 CH 設定の取得に失敗しました
ER07802	エラー	7802	基準色データ No.の取得に失敗しました
ER15002	エラー	15002	測定モードの取得に失敗しました
ER15004	エラー	15004	測定モード設定機能に対応してない本体が接続されています。 (本体ファーム ver.1.20 以降対応)
ER15102	エラー	15102	タイマー設定の取得に失敗しました
ER15104	エラー	15104	タイマー設定機能に対応してない本体が接続されています。 (本体ファーム ver.1.20 以降対応)
ER15502	エラー	15502	任意波長設定の取得に失敗しました
ER15504	エラー	15504	任意波長設定機能に対応してない本体が接続されています。 (本体ファーム ver.1.20 以降対応)

説明：
測定器単体での測定時の測定条件を取得します(CL-SDK で測定器を制御している場合の測定時の測定条件は取得しません)。
項目ごとに取得するので、取得したい項目を「Type」で指定して下さい。

[注意]
CLGetMeasData()で取得する測定データの測定条件は、CLGetProperty()で取得してください。

CLSetSystemSetting()

概要：
本体のシステム設定を設定します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
Type	I	設定するシステム設定の種別
pSetting	I/O	設定するシステム設定の設定値

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	本体のシステム設定を設定しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER_PARAM_NULL	エラー	13	引数 pSetting が NULL 値になっています
ER00601	エラー	601	引数 pSetting が NULL 値になっています
ER00602	エラー	602	指定した種別の値が不正です
ER00603	エラー	603	リモートモードが OFF になっています
ER09602	エラー	9602	日時の設定値に誤りがあります
ER09603	エラー	9603	日時設定に失敗しました
ER05001	エラー	5001	本体表示回転の設定値が不正な値です
ER05003	エラー	5003	本体表示回転設定に失敗しました
ER01601	エラー	1601	ブザー音の設定値が不正な値です
ER01603	エラー	1603	ブザー音設定に失敗しました
ER02001	エラー	2001	言語設定の設定値が不正な値です
ER02003	エラー	2003	言語設定に失敗しました
ER04901	エラー	4901	日時フォーマットの設定値が不正な値です
ER04903	エラー	4903	日時フォーマット設定に失敗しました
ER08101	エラー	8001	校正警告の設定値が不正な値です
ER08103	エラー	8003	校正警告設定に失敗しました
ER10502	エラー	10502	オートパワーオフの設定値が不正な値です
ER10503	エラー	10503	オートパワーオフ設定に失敗しました
ER10401	エラー	10402	工場校正警告の設定値が不正な値です
ER10403	エラー	10403	工場校正警告設定に失敗しました

説明：
本体のシステム設定を設定します。
項目ごとに設定するので、設定したい項目を「Type」で指定して下さい。

CLGetSystemSetting()

概要：
本体のシステム設定を取得します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
Type	I	取得する属性項目を指定
pParam	I/O	取得する属性項目の設定値の格納先

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	本体のシステム設定を取得しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER00601	エラー	601	引数 pSetting が NULL 値になっています
ER00602	エラー	602	指定した属性項目の値が不正な値です
ER00603	エラー	603	リモートモードが OFF になっています
ER01602	エラー	1602	ブザー音設定の取得に失敗しました
ER02002	エラー	2002	言語設定の取得に失敗しました
ER04902	エラー	4902	日時フォーマット設定の取得に失敗しました
ER05002	エラー	5002	本体表示回転設定の取得に失敗しました
ER08102	エラー	8002	校正警告設定の取得に失敗しました
ER09601	エラー	9601	日時の取得に失敗しました
ER10401	エラー	10401	工場校正警告設定の取得に失敗しました
ER10501	エラー	10501	オートパワーオフ設定の取得に失敗しました

説明：
本体のシステム設定を取得します。
項目ごとに取得するので、取得したい項目を「Type」で指定して下さい。

CLGetDeviceID()

概要：
本体情報を取得します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
pDeviceID	O	取得する本体情報を格納するバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	本体情報を取得しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER00901	エラー	901	引数 pDevID が NULL 値になっています
ER00902	エラー	902	工場校正が完了していません(*1)
ER00903	エラー	903	リモートモードが OFF になっています

説明：
本体情報を取得します。

(*1)
戻り値として「ER00902」が戻ってきた場合には、ゼロ校正や測定ができません。すぐに本体を最寄りのサービス拠点に送付して下さい。

CLGetSDKVersion()

概要：
CL-SDK のバージョン情報を取得します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
Version	O	取得するバージョン情報を格納するバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	バージョン情報を取得しました
ER00101	エラー	101	引数 SDKVersion が NULL 値になっています

説明：
CL-SDK のバージョン情報を取得します。
4 つの DLL のバージョン情報を取得できます。

CLGetWarning()

概要：
最新の実行処理での警告内容を取得します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
wrcode	O	取得する警告内容を格納するバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	警告内容を取得しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER10001	エラー	10001	リモートモードが OFF になっています

説明：
警告内容の詳細を取得します。
各 API の戻り値として返ってくる値は「WARNING」で共通です。
API からの戻り値が WARNING で警告の詳細を確認したい場合は、本関数で警告内容の詳細を取得してください。

CLGetPeriodicCalDate()

概要：
定期校正日時に関する情報を取得します

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

引数：

引数	I/O	説明
hDevice	I	制御対象の測定器のオブジェクトハンドラ
Type	I	取得する定期校正日時の種別 0：工場校正の開始日時 1：工場校正日時の満了日
pDateTIme	O	取得する校正日時を格納するバッファ

戻り値：

戻り値	種別	値	説明
SUCCESS	正常	0	定期校正日時情報を取得しました
ER_HANDLE_NULL	エラー	10	オブジェクトハンドラが存在しません
ER09301	エラー	9301	定期校正日時情報の取得に失敗しました
ER09302	エラー	9302	リモートモードが OFF になっています
ER09303	エラー	9303	引数 DateTime が NULL 値になっています
ER09304	エラー	9304	指定した種別が範囲外です

説明：
定期校正日時に関する情報を取得します。
開始日は、サービスによる定期校正を行った日時となります。ただし、最初の定期校正開始日時は本体の初回電源 ON 時の日時となります。
満了日は、定期校正開始日から 11 ヶ月後の日時となります。

4.5 エラーコード

CL-SDK ではエラーが発生した際に、CL-SDK 内のどこでエラーになったかが分かるよう(エラー解析が容易にできるよう)に、各関数で発生したエラーを API の戻り値として返すようにしています。
各 API でのエラー値の詳細については各 API の戻り値の項目を参照してください。
以下に示すエラー値は、全 API 共通で利用するエラー値です。

値	メッセージ	説明
0	SUCCESS	処理が正常に完了しました。
1	WARNING	処理は完了しましたが、警告があります。
10	ER_HANDLE_NULL	デバイスハンドルが NULL 値になっています
13	ER_OPENDEVICE	デバイスハンドルの取得に失敗しました
14	ER_PARM_NULL	代入したパラメータが NULL 値になっています
16	ER_ALLMEASURING	全数測定中です(CLDoMeasureAll()を実行中です)

警告値については全 API 共通で「WARNING」を返します。警告の詳細については、CLGetWarning()関数で警告の詳細を取得する必要があります。
API からの戻り値が警告値だった場合は、必要に応じて CLGetWarning()関数で警告の詳細内容を取得してください。

値	メッセージ	説明
0	WRNONE	警告がありません
1	WR001	最終のゼロ校正から一定時間が経過しています。ゼロ校正を実行して下さい。
2	WR002	照度値が保証範囲より低い値です
3	WR003	色彩演算ができなかった値が存在します
4	WR004	該当するランクがありませんでした
6	WR_OVER_EXPOSURE	マニュアル測定で指定した露光時間が適正範囲を超えています
7	WR_UNDER_EXPOSURE	マニュアル測定で指定した露光時間が適正範囲を下回っています

5. 付録

5.1 ランク領域の設定について

ランク領域は 1 つの多角形として設定する必要があります。

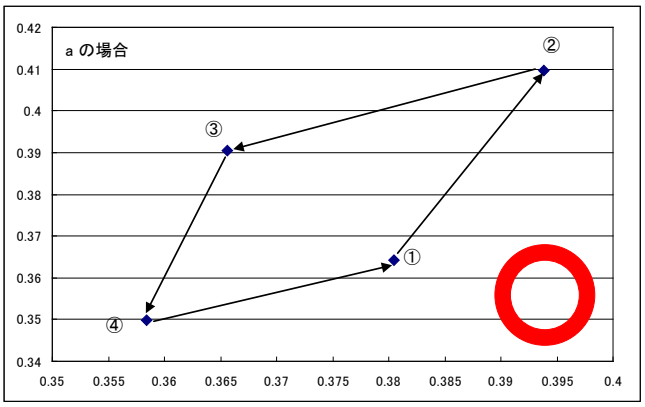
ランク領域は配列に格納された順(要素番号の小さい順=要素番号 0)にポイントを結ぶ(最後のポイントは始点と結びます)形で領域を作成するため、配列に格納する順によっては図に示すように領域が分割される可能性があります。

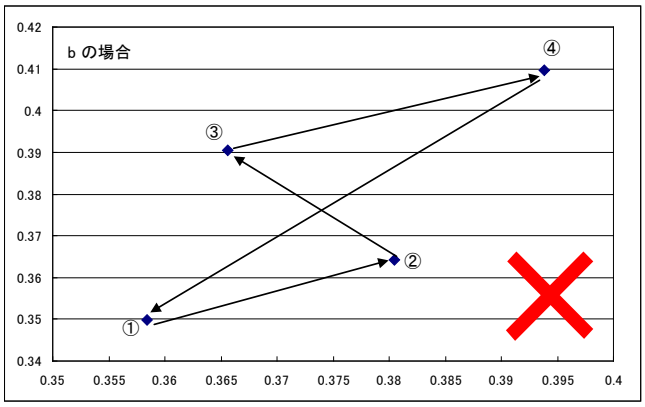
ランク領域の設定時には、設定するランク領域が分割されていないをチェックし、ランク領域が分割されている場合はエラーとなりますので、配列に格納する順序についてはご注意下さい。

ex.)(x, y) = {0.3584, 0.3499}、{0.3805, 0.3642}、{0.3938, 0.4097}、{0.3656, 0.3905}、の 4 点を領域として設定する場合(図中の〇数字は配列の格納順を示します)

a) {0.3805, 0.3642}、{0.3938, 0.4097}、{0.3656, 0.3905}、{0.3584, 0.3499}の順に配列に格納した場合：1 つの領域になっているので、ランク領域として OK となります

b) {0.3584, 0.3499}、{0.3805, 0.3642}、{0.3656, 0.3905}、{0.3938, 0.4097}の順に配列に格納した場合：領域が分割されているので、ランク領域として NG となります





5.2 文字コード表

基準色データ、ユーザー校正係数データのデータ名に利用できる文字は以下の文字です。「SP」は空白を表します。

	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 補足事項

相関色温度、⊿uv、主波長、刺激純度については、本体取説に詳細な説明がありますので、参照して下さい。


KONICA MINOLTA