色彩輝度計 **CS-200**

DLL 利用方法

初版：	2005.1.26	発行。
第二版：	2008.8.4	サンプルコードに、OHCI 準拠の USB ホストコントローラを搭載した PC 環境で測定を行う場合に必要なウェイト処理を追加。
第三版：	2013.8.19	社名変更
第四版：	2016.8.12	誤記修正

KONICA MINOLTA, INC.

CS-200 用 USB ドライバおよび本書を含むマニュアル類に関するご注意

- 色彩輝度計 CS-200 を PC などと接続して使用する場合、CS-200 用 USB ドライバが必要です。
- CS-200 用 USB ドライバおよび本書を含むマニュアル類（以下本ソフトウェアと呼ぶ）の著作権は、コニカミノルタ㈱にあります。
- 本ソフトウェアの一部または全部を無断で使用、複製することはできません。
- 本ソフトウェアは、色彩輝度計 CS-200 をご購入いただいたお客様が使用許諾書に同意いただいたもとでのみ使用することができます。
- 通信仕様書に記載のコマンドを無断で、商用販売目的のソフトウェアに使用することはできません。
- 本書は PC 通信の基本を理解されている方への説明用として準備されたものです。
- 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。
- 本書の内容に関しては、将来予告なしに変更することがあります。
- 本書は内容について万全を期していますが、万一不審な点や誤り、記載もれなどでお気づきの点がございましたら、お問い合わせ窓口までご連絡ください。
- 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。
- 本書に記載の会社名、商品名は各社の商標または登録商標です。

目次

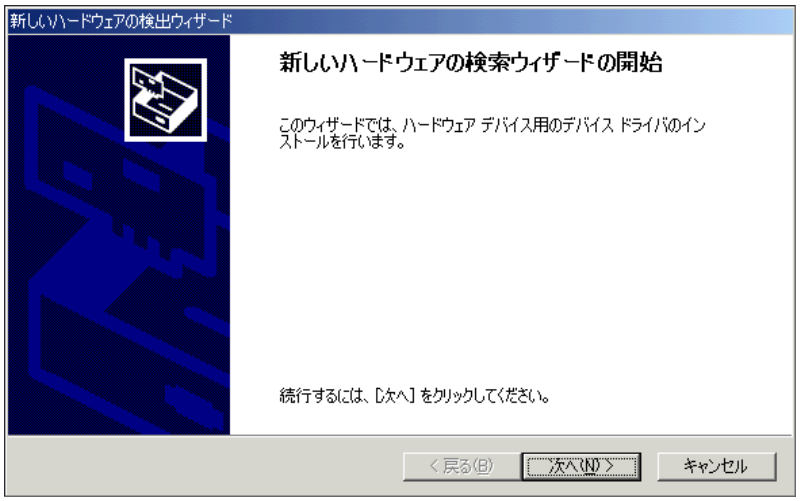
1.　前準備

1.1　USB ドライバのインストール

初めて PC と CS-200 を接続した場合には、ドライバをインストールする必要があります。
手順としては以下となります。

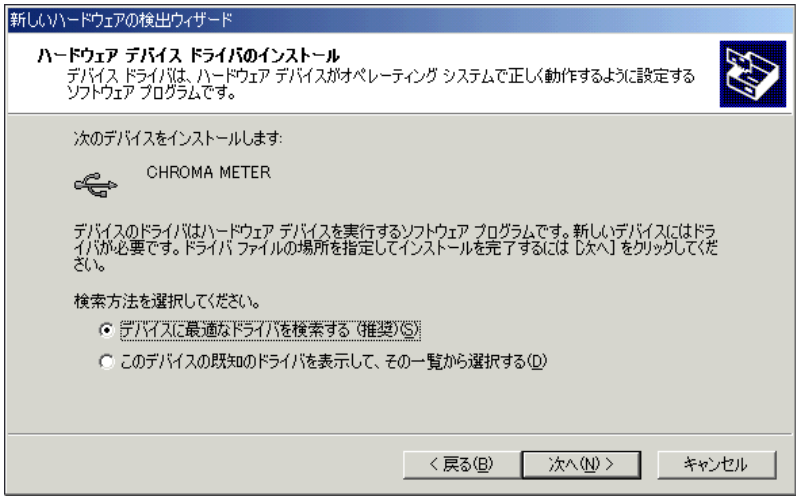
（1）PC と CS-200 を USB ケーブルで接続し、CS-200 の電源を立ち上げると、「新しいハードウェアの検出ウィザード」（画面 1-1-1）が表示されます。この画面では、「次へ」ボタンをクリックします。

≪画面 1-1-1≫



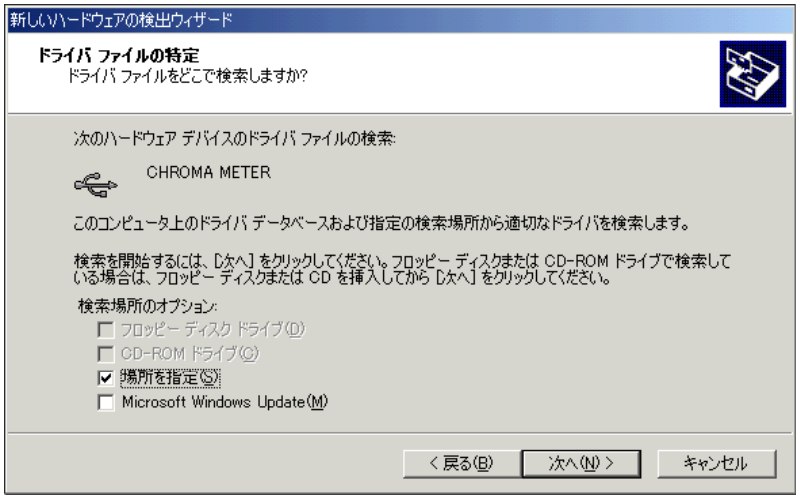
（2）画面 1-1-2 が表示されますので、ドライバの検索方法として「デバイスに最適なドライバを検索する」を選択し、「次へ」ボタンをクリックします。

≪画面 1-1-2≫



（3）画面 1-1-3 が表示されますので、「場所を指定」にチェックを付け、「次へ」ボタンをクリックします。

≪画面 1-1-3≫



（4）画面 1-1-4 が表示されますので、以下のどちらかの場所を参照して下さい。

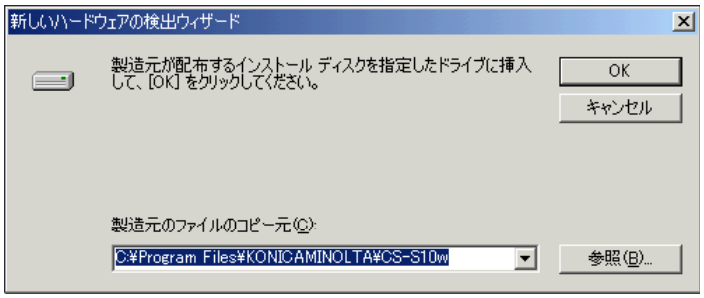
－　CS-S10w がインストールされている場合

⇒　インストール先のフォルダ

－　CS-S10w がインストールされていない場合

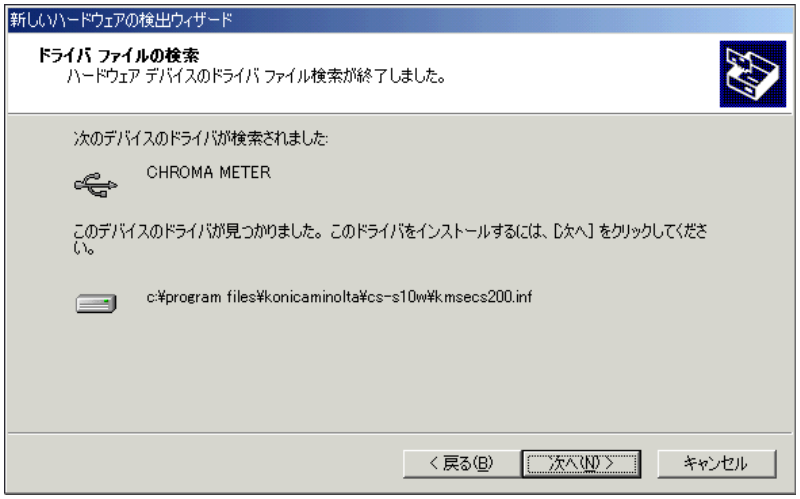
⇒　WEB サイトからダウンロードして、USB ドライバを保存したフォルダ

≪画面 1-1-4≫



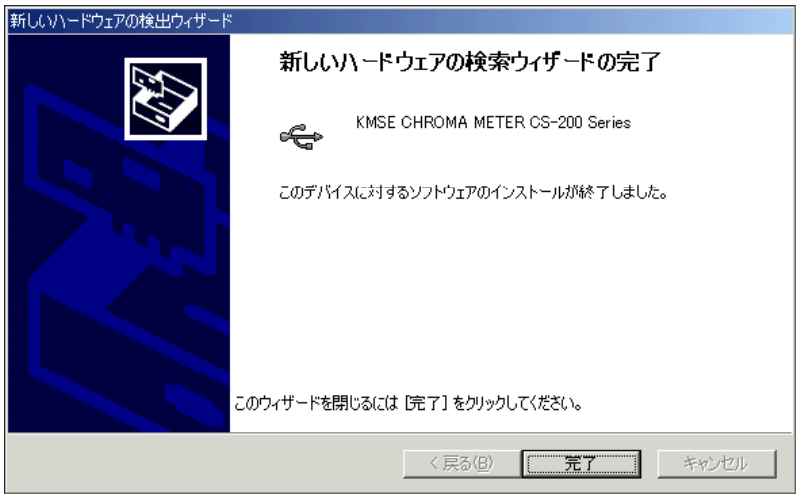
（5）ドライバを検索し、成功すると画面 1-1-5 が表示されますので、「次へ」ボタンをクリックすると、ドライバのインストールが開始されます。

≪画面 1-1-5≫



（6）インストールが完了すると、画面 1-1-6 が表示され、「完了」ボタンをクリックすると、操作完了です。

≪画面 1-1-6≫



2. DLL 利用方法

※64bit 環境で動作しますが、64bit 版のアプリケーションは作成できません

2.1 DLL のインストール

CS-200 で USB 通信を行う場合には、“Kmsecs200.dll”を以下のいずれかのディレクトリにコピーして使用する必要があります。(“Kmsecs200.dll”はコニカミノルタ株式会社の WEB サイトにて、USB ドライバとともにダウンロードしていただけます。)

(1) 実行可能ファイル(EXE)を含むディレクトリ

(2) カレント作業ディレクトリ

※ Visual Basic のカレントディレクトリとは VB32.EXE があるディレクトリを指す

(3) Windows のシステムディレクトリ

ただし、実行可能ファイル (EXE) が完成し、他の PC などにインストールする場合には、実行可能ファイル (EXE) と同じディレクトリにコピーするようにして下さい。

2.2 DLL で使用可能なコマンド

使用可能なコマンドは以下の 5 種類となります。詳細はそれぞれの説明を参照して下さい。

int_usb	指定した測定器 (番号指定) をオープンする
end_usb	指定した測定器 (番号指定) をクローズする
get_num	システムに接続され使用可能な測定器の数を取得する
write64_usb	指定した測定器 (番号指定) へ送信 (DMA) を行う
read64_usb	指定した測定器 (番号指定) から受信 (DMA) を行う

2.2.1 int_usb

コマンド	int int_usb(int Index)	
戻り値	0	正常終了
	-1	Index の値が不正
	-2	デバイス未接続
	-3	送信 (FIFO) パイプオープンに失敗
	-4	受信 (FIFO) パイプオープンに失敗
	-5	送信 (DMA) パイプオープンに失敗
	-6	受信 (DMA) パイプオープンに失敗
引数	Index	測定器 ID (0～126)
説明	Index で指定した測定器とのパイプをオープンする。オープンに失敗した場合には、パイプをすべてのパイプをクローズする。 終了時には end_usb()を呼び出すこと。 〔補足〕 測定器 ID は 0～126 の整数で、測定器を 1 台しか接続していない場合の測定器 ID は 0 となる。複数台の測定器を接続している場合には、測定器が接続されている USB ポートのポート番号が若い順に測定器 ID が 0 から割り当てられる。	

2.2.2 end_usb

コマンド	int end_usb(int Index)	
戻り値	0	正常終了
	-1	Index の値が不正
引数	Index	測定器 ID (0～126)
説明	Index で指定した測定器とのパイプをクローズする。	

2.2.3 get_num

コマンド	int get_num()
戻り値	使用可能な測定器の数
引数	
説明	システムに接続され、使用可能な測定器の数を取得する。

2.2.4 write64_usb

コマンド	int write64_usb(int Index, char* Cmd, int Timeout, int WriteLen)	
戻り値	0 以上	正常終了
	-1	Index の値が不正
	-2	パイプがオープンされていない
	-3	送信エラー
引数	Index	測定器 ID (0～126)
	Cmd	送信文字列 (最大 64 文字)
	Timeout	使用していないが、1 以上の値を入れること。
	WriteLen	送信文字列長
説明	Index で指定した測定器へ指定文字列を送信 (DMA) する。	

2.2.5 read64_usb

コマンド	int read64_usb(int Index, char* Dat, int Timeout, int ReadLen)	
戻り値	0 以上	正常終了
	-1	Index の値が不正
	-2	パイプがオープンされていない
	-3	受信エラー
引数	Index	測定器 ID (0～126)
	Dat	受信文字列
	Timeout	使用していないが、1 以上の値を入れること。
	ReadLen	受信文字列長　※必ず 250 文字とすること。
説明	Index で指定した測定器から文字列を受信 (DMA) する。	

3. サンプルコード

3.1 Visual C++の場合

3.1.1 DLL のリンク

(1) DLL をリンクし、ハンドルを取得する

以下の方法で DLL のハンドルを取得します。このハンドルを用いて (2)・(3) の操作を行います。

```
HINSTANCE m_hDll = LoadLibrary("kmsecs200.dll");
```

(2) DLL 内のエクスポート関数のアドレスを取得する

あらかじめ関数ポインタ用に対応する typedef を作成しておきます。

```
typedef INT (CALLBACK* USB_INI)(INT);
typedef INT (CALLBACK* USB_NUM)(VOID);
typedef INT (CALLBACK* USB_IO)(INT, LPSTR, INT, INT);
```

それぞれの関数に対応したアドレスを取得し、このアドレスを使用して各種操作を行うことができます。

```
USB_INI int_usb = (USB_INI)GetProcAddress(m_hDll, "int_usb");
USB_INI end_usb = (USB_INI)GetProcAddress(m_hDll, "end_usb");
USB_NUM get_num = (USB_NUM)GetProcAddress(m_hDll, "get_num");
USB_IO write64_usb = (USB_IO)GetProcAddress(m_hDll, "write64_usb");
USB_IO read64_usb = (USB_IO)GetProcAddress(m_hDll, "read64_usb");
```

(3) リンクを解放する

```
FreeLibrary(m_hDll);
```

3.1.2 サンプルコード

```
// 型定義
typedef INT (CALLBACK* USB_INI)(INT);
typedef INT (CALLBACK* USB_NUM)(VOID);
typedef INT (CALLBACK* USB_IO)(INT, LPSTR, INT, INT);



//（例）測定する場合

// DLL をリンクする
HINSTANCE hDll = LoadLibrary("kmsecs200.dll");
```

```
// 関数のアドレスを取得する
USB_INI int_usb = (USB_INI)GetProcAddress(hDll, "int_usb");
USB_INI end_usb = (USB_INI)GetProcAddress(hDll, "end_usb");
USB_IO write64_usb = (USB_IO)GetProcAddress(hDll, "write64_usb");
USB_IO read64_usb = (USB_IO)GetProcAddress(hDll, "read64_usb");

// パイプをオープンする
int_usb(0);

char cBuf[250];

// リモート ON にする
char cRemote[]={"RMT,1¥r¥n"};
write64_usb(0, cRemote, 1, 7);
read64_usb(0, cBuf, 1, 250);

// 測定を開始する
char cMes[]={"MES,1¥r¥n"};
write64_usb(0, cMes, 1, 7);
read64_usb(0, cBuf, 1, 250);

// 測定時間取得
CString str(cBuf);
int nTime = atoi(str.Mid(5, 2));

// (測定時間-0.5)秒のウェイト時間を作成する
DWORD dwTime = (DWORD)(nTime*1000 - 500);
Sleep(dwTime);
```

```
// データを受信する
char cMdr[]={"MDR,0¥r¥n"};
while(1){
    write64_usb(0, cMdr, 1, 7);
    read64_usb(0, cBuf, 1, 250);

    CString str(cBuf);
    if (str.Left(4) == _T("ER02")){
        // 測定中
        // データ受信コマンドを再送する前に 0.3 秒ウェイトする
        Sleep(300);
    }else{
        // 正常終了またはエラー
        break;
    }
}

// OK の場合は cBuf にデータが格納されている

// パイプをクローズする
end_usb(0);

// DLL のリンクを解除する
FreeLibrary(hDll);
```

3.2 Visual Basic の場合

3.2.1 DLL のリンク

以下の方法で DLL 関数の定義を行います。あとは、指定した関数名を使用して各種操作を行うことができます

```
Public Declare Function int_usb Lib "kmsecs200.dll" (ByVal Index As Long) As Long
Public Declare Function end_usb Lib "kmsecs200.dll" (ByVal Index As Long) As Long
Public Declare Function get_num Lib "kmsecs200.dll" () As Long
Public Declare Function write64_usb Lib "kmsecs200.dll" (ByVal Index As Long, ByVal
  Cmd As String, ByVal TimeOut As Long, ByVal WriteLen As Long) As Long
Public Declare Function read64_usb Lib "kmsecs200.dll" (ByVal Index As Long, ByVal
  Dat As String, ByVal TimeOut As Long, ByVal ReadLen As Long) As Long
```

3.2.2 サンプルコード

```
’関数定義
Public Declare Function int_usb Lib "kmsecs200.dll" (ByVal Index As Long) As Long
Public Declare Function end_usb Lib "kmsecs200.dll" (ByVal Index As Long) As Long
Public Declare Function write64_usb Lib "kmsecs200.dll" (ByVal Index As Long, ByVal
Cmd As String, ByVal TimeOut As Long, ByVal WriteLen As Long) As Long
Public Declare Function read64_usb Lib "kmsecs200.dll" (ByVal Index As Long, ByVal Dat
As String, ByVal TimeOut As Long, ByVal ReadLen As Long) As Long

’（例）測定する場合

’パイプをオープンする
int_usb 0

’変数を定義する
Dim strSend As String
Dim strRecv As String
Dim sngTime As Single
```

```
’リモートONする
strSend = "RMT,1" + vbCr + vbLf
strRecv = String(250, Chr(0))
write64_usb 0, strSend, 1, Len(strSend)
read64_usb 0, strRecv, 1, 250

’測定を開始する
strSend = "MES,1" + vbCr + vbLf
strRecv = String(250, Chr(0))
write64_usb 0, strSend, 1, Len(strSend)
read64_usb 0, strRecv, 1, 250

If Left(rBuf, 2) = "OK" Then
    ’測定時間取得
    sngTime = CSng(Mid(rBuf, 6, 2))
    ’(測定時間-0.5)秒のウェイト時間を作成する
    WaitSec(sngTime - 0.5)
Else
    ’測定エラー
End If

’データを受信する
strSend = "MDR,0" + vbCr + vbLf
Do
    strRecv = String(250, Chr(0))
    write64_usb 0, strSend, 1, Len(strSend)
    read64_usb 0, strRecv, 1, 250
    If Left(strRecv,4) = “ER02” Then
        ’データ受信コマンドを再送する前に0.3秒ウェイトする
        WaitSec(0.3)
    Else
        Exit Do
    End If
Loop While(1)
```

```
’OK の場合は strRecv にデータが格納されている

’パイプをクローズする
end_usb 0
```

```
’ウェイト関数例
Sub WaitSec(t As Single)
    Dim start_t, end_t As Single
    Dim passed_t As Single

    start_t = Timer
    Do
        end_t = Timer
        passed_t = end_t - start_t
        If passed_t < 0 Then passed_t = passed_t + 24# * 60# * 60#
        DoEvents
    Loop Until passed_t >= t
End Sub
```