

CA-SDK 使用説明書

Ver.4.5



KONICA MINOLTA

2016 年 7 月 6 日
コニカミノルタ株式会社 センシング事業本部
〒590-8551 大阪府堺市堺区大仙西町3-91
<http://www.konicaminolta.jp/instruments/>

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、ご購入の販売店または“サービスのご案内”に記載のお問い合わせ窓口までご連絡ください。
- ・本書に記載のサンプルコードは、お客様のソフトウェア開発の参考にしていただくためのものであり、サンプルコードの正確性を保証するものではありません。
- ・本書の内容を運用した結果につきましては責任を負いかねますので、あらかじめご了承ください。
- ・Windows®は、米国マイクロソフト社の商標です。その他、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。

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
VB, Visual Basic	Microsoft® Visual Basic
Windows	Microsoft® Windows®
Windows 7	Microsoft® Windows® 7 Operating System
Windows 8.1	Microsoft® Windows® 8.1 Operating System
Windows 10	Microsoft® Windows® 10 Operating System

目 次

1. CA-SDK のインストール	2
1-1 CA-SDK のインストール/アンインストール方法	3
1-2 USB ドライバのインストール方法	4
2. VisualC# 2013 でのアプリケーション作成方法	6
2-1 VisualC# 2013 のプロジェクト作成	7
2-2 CA-SDK の参照設定	8
2-3 アプリケーションの GUI/コードの作成	9
3. VisualBasic 2013 でのアプリケーションの作成方法	13
3-1 VisualBasic 2013 のプロジェクト作成	14
3-2 CA-SDK の参照設定	15
3-3 アプリケーションの GUI/コードの作成	16
4. CA-SDK サンプルソフト用コントロールの使用方法	19
4-1 CaControl (CA 測定器セッティング用コントロール)	20
4-2 xyControl (xy 色度図表示コントロール)	27
5. アプリケーション例	32
5-1 ホワイトバランス調整	33
5-2 γ 調整	36

CA-SDK 使用説明書

1. CA-SDK のインストール

1-1 CA-SDK インストール/アンインストール方法

CA-SDK をインストール/アンインストールする場合、管理者権限をもつユーザーで行ってください。
ユーザーの権限について明確でない場合、システムの管理者に確認してください。

SDK 使用上のシステム要件

本 SDK(Software Development Kit)を使用するためには、以下の条件を満たすソフトウェア開発環境が必要です。

- ・ Windows 7 / Windows 8.1/ Windows 10 のいずれかの OS が正常に動作する PC/AT 互換 PC
- ・ Microsoft COM コンポーネントをサポートしている開発ツール(Visual Basic など)がインストールされていること
- ・ USB1.1 準拠の USB ポートまたは RS-232C ポートが装備されていること

1-1-1 インストール

【注記】CA-SDK の旧バージョンがインストールされている場合には、先に旧バージョンのアンインストールが必要です。

- ① CA-SDK セットアップディスクを CD ドライブにセットしてください。
- ② マイコンピューターを開き、CD ドライブ内のフォルダー「x64」(64bit 版)または「x86」(32bit 版)を開いてください。
- ③ フォルダーにある「setup.exe」を実行します

これで、セットアップが開始されます。

以降は、セットアップのメッセージに従って操作してください。

なお、CA-310/CA-210 を接続して動作させるには、「1-2 USBドライバのインストール方法」に従って USB ドライバをインストールする必要があります。

関連事項:

インストール後、サンプルソフトを利用してパターンジェネレータを制御しその設定データを変更する場合、制限つきユーザーで行うと問題が発生する場合があります。この場合、CA-SDK インストールフォルダ/ファイルに対して、使用するユーザーアカウントに適切なアクセス許可の設定をする必要があります。

設定の例:

CA-SDK をインストールした最上位フォルダー

(デフォルトでは、64bitOS、64bitSDK の場合は、通常、「C:\Program Files\KONICA MINOLTA\CA-SDK」フォルダー)に対して、対象ユーザーのアクセス許可を「フルコントロール」にします。

また、サンプルソフト・プロジェクトを開発ツールで開いて作業する場合、制限ユーザーでは、問題が発生する場合があります。

問題が発生した場合、上のアクセス許可の設定作業をした後、次の作業を 2 回行なってください。

・管理者権限でプロジェクトを開き、正常にプロジェクトが開けたことを確認後、プロジェクトを閉じ、PC を再起動する(これを 2 回繰り返す)

1-1-2 アンインストール

CA-SDK をアンインストールするには、通常のアプリケーションの削除方法(「コントロールパネル」から「アプリケーションの追加と削除」を実行)に従って行ってください。

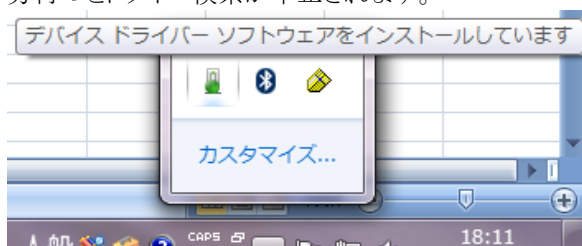
1-2 USBドライバのインストール方法

CA-310/CA-210 を USB 接続するには、USB のデバイスドライバをインストールする必要があります。
手順を記載します。

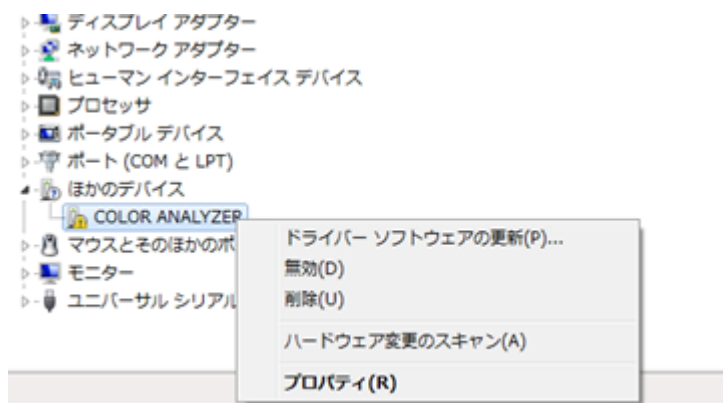
1. CA-SDK の CD-ROM を CD ドライブにセットします。
2. USB ポートに CA-310/CA-210 を接続します。

Windows7 の場合、「新しいハードウェアが見つかりました」等のダイアログが表示されたら、ここではインストールせず、キャンセルして閉じます。

Windows7 では自動的に WindowsUpdate に接続してドライバを検索し始めますので、画面右下タスクバーにある、検索中を表すアイコンを選択し、「WindowsUpdate からのドライバーソフトウェアの取得をスキップする」をクリックしてください。数分待つとドライバ検索が中止されます。



3. デバイスマネージャーを開きます。「COLOR ANALYZER」を右クリック-「ドライバーソフトウェアの更新…」を選択します。

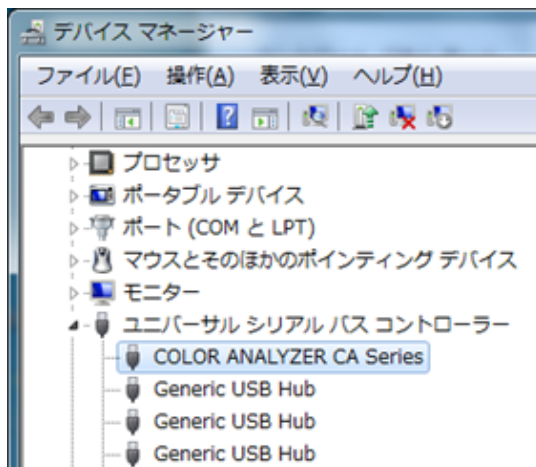


4. 「コンピュータを参照してドライバーソフトウェアを検索します」を選択し、CD-ROM が入っているドライブ内の Driver フォルダーを選択して、「次へ」を押します。
(例)Eドライブの場合、E:\Driver
5. インストールの確認メッセージが表示された場合は、「はい」又は「OK」を選択します。
6. 「閉じる」をクリックしてインストールを終了します。

＜USBドライバがインストールされたことの確認＞

USBドライバがインストールされると、CA-310/CA-210 が USB 機器として認識されます。

CA-310/CA-210 両方とも、デバイスマネージャーで「COLOR ANALYZER CA Series」と表示されます。



CA-SDK 使用説明書

2. VisualC# 2013 での アプリケーション作成方法

Visual Studio 2013 を使用して、CA-SDK のアプリケーションを開発する方法を説明します。(開発環境のバージョンが異なる場合、本説明で示した内容と多少異なる場合があります。)

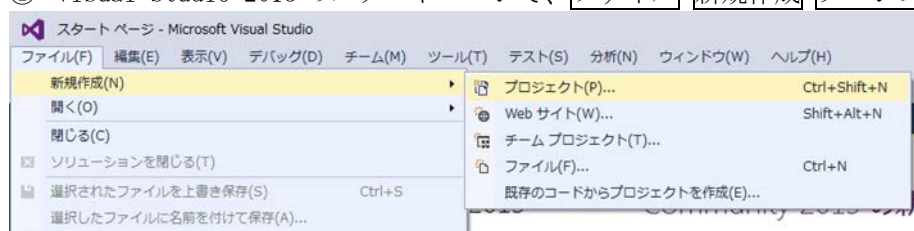
アプリケーションは以下の手順で作成します。

1. CA-SDK をインストールする。
2. VisualC# 2013 のプロジェクトを作成する。
3. CA-SDK の参照設定をする。
4. アプリケーションの GUI/コードを作成する。

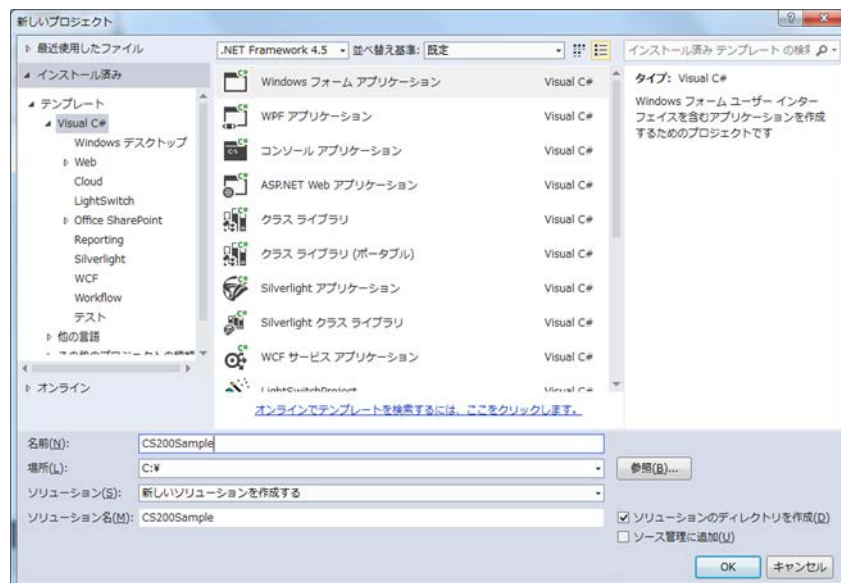
VisualC# 2013 での作業を始める前に、“1. CA-SDK をインストールする。”を行ってください。
以下、上記の“2. VisualC# 2013 のプロジェクトを作成する。”以降の説明をします。

2-1 VisualC# 2013 のプロジェクト作成

① Visual Studio 2013 のスタートページで、**ファイル** **新規作成** **プロジェクト**を選択します。



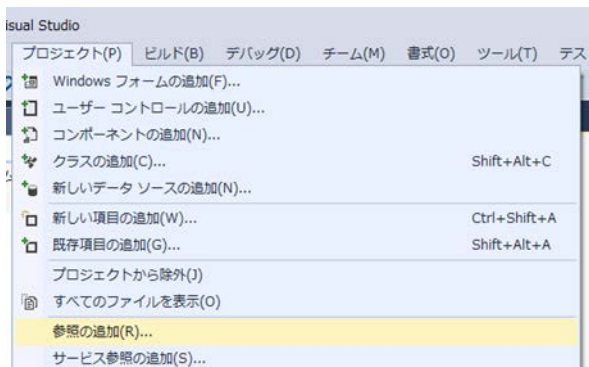
②新しいプロジェクトダイアログで、テンプレートとして、「Windows フォームアプリケーション」を選択して、プロジェクト名として“Ca200Sample”を入力し、OK をクリックします。



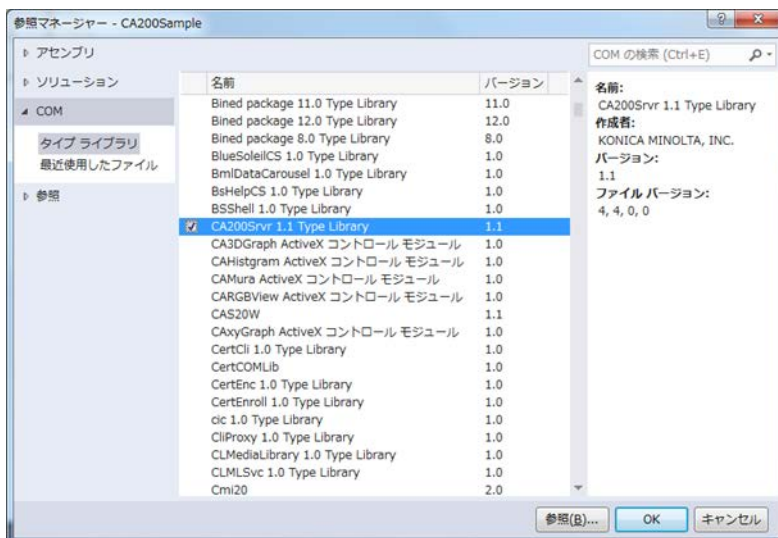
プロジェクトが作成され、フォームのデザイン画面が表示されます。

2-2 CA-SDK の参照設定

- ① メニューの **プロジェクト** - **参照の追加** を選択します。



- ② 参照マネージャー左側で、COM を選択、表示されるリストから CA200Srvr 1.1 Type Library のチェックボックスを ON にして、OK ボタンをクリックします。



2-3 アプリケーションの GUI/コードの作成

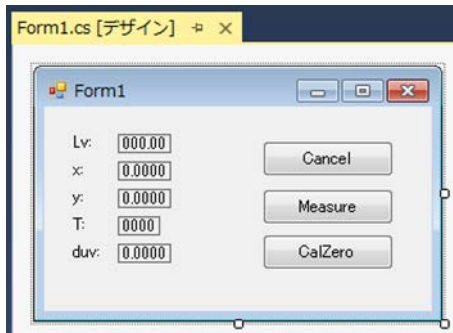
サンプルソフトのGUIを以下のようにします。

ラベルを 10 個（ラベル用・測定値表示用各 5 個）、ボタンを 3 個利用します。

Cal Zeroボタンをクリックすると、CA本体で、ゼロ校正を実行します。

Measureボタンをクリックすると、20回連続測定し、測定結果を測定値表示用ラベルに表示します。

Cancelボタンをクリックすると、測定を途中で中止します。



サンプルコードを以下に示します。

```
namespace Ca200Sample
{
    public partial class Form1 : Form
    {
        private CA200SRVRLib.Ca200 objCa200;
        private CA200SRVRLib.Ca objCa;
        private CA200SRVRLib.Probe objProbe;
        private Boolean isMsr;
        long vbObjectError = -2147221504;

        public Form1()
        {
            InitializeComponent();
            try
            {
                objCa200 = new CA200SRVRLib.Ca200();
                objCa200.AutoConnect();
                objCa = objCa200.SingleCa;
                objProbe = objCa.SingleProbe;

                objCa.ExeCalZero += new
CA200SRVRLib._ICaEvents_ExeCalZeroEventHandler(objCa_ExeCalZero);

                ButtonCancel.Enabled = false;
                ButtonMeasure.Enabled = true;
                ButtonCalZero.Enabled = true;
            }
            catch (Exception er)
            {
                DisplayError(er);
                Application.Exit();
            }
        }
    }
}
```

```

    }
}

private void DisplayError(Exception er)
{
    String msg;
    msg = "Error from" + er.Source + "¥r¥n";
    msg += er.Message + "¥r¥n";
    msg += "HR:" + (er.HResult - vbObjectError).ToString();
    MessageBox.Show(msg);
}

private void ButtonMeasure_Click(object sender, EventArgs e)
{
    int i;
    try
    {
        isMsr = true;

        ButtonCancel.Enabled = true;
        ButtonMeasure.Enabled = false;
        ButtonCalZero.Enabled = false;

        for (i = 0; i < 20; i++)
        {
            objCa.Measure();
            LabelLv.Text = objProbe.Lv.ToString("##0.00");
            Labelx.Text = objProbe.sx.ToString("0.0000");
            Labely.Text = objProbe.sy.ToString("0.0000");
            LabelT.Text = objProbe.T.ToString("####");
            Labelduv.Text = objProbe.duv.ToString("0.0000");
            Application.DoEvents();

            if (isMsr == false)
            {
                break;
            }
        }

        ButtonCancel.Enabled = false;
        ButtonMeasure.Enabled = true;
        ButtonCalZero.Enabled = true;

    }
    catch (Exception er)
    {
        DisplayError(er);
        Application.Exit();
    }
}

private void ButtonCancel_Click(object sender, EventArgs e)
{

```

```

        isMsr = false;
        ButtonCancel.Enabled = false;
        ButtonMeasure.Enabled = false;
        ButtonCalZero.Enabled = false;
    }

    private void ButtonCalZero_Click(object sender, EventArgs e)
    {
        bool calzero_success = false;

        while (calzero_success == false)
        {
            ButtonMeasure.Enabled = false;
            ButtonCalZero.Enabled = false;

            try
            {
                objCa.CalZero();
                calzero_success = true;
            }
            catch (Exception er)
            {
                DisplayError(er);
                if (MessageBox.Show("Zero Cal Error¥r¥nRetry?", "CalZero",
MessageBoxButtons.OKCancel) == DialogResult.Cancel)
                {
                    objCa.RemoteMode = 0;
                    Application.Exit();
                }
            }
        }

        ButtonMeasure.Enabled = true;
        ButtonCalZero.Enabled = true;
    }

    private void objCa_ExeCalZero()
    {
        if (MessageBox.Show("CalZero?", "CalZero", MessageBoxButtons.OKCancel) ==
DialogResult.Cancel)
        {
            return;
        }
        ButtonMeasure.Enabled = false;
        ButtonCalZero.Enabled = false;

        try
        {
            objCa.CalZero();
        }
        catch (Exception er)
        {
            DisplayError(er);
        }
    }

```

```

    }

    ButtonMeasure.Enabled = true;
    ButtonCalZero.Enabled = true;

}

private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    objCa.RemoteMode = 0;
    objCa200 = null;
    objCa = null;
    objProbe = null;
}

}

}

```

CA-SDK 使用説明書

3. Visual Basic 2013 での アプリケーションの作成方法

Visual Studio 2013 を使用して、CA-SDK のアプリケーションを開発する方法を説明します。(開発環境のバージョンが異なる場合、本説明で示した内容と多少異なる場合があります。)

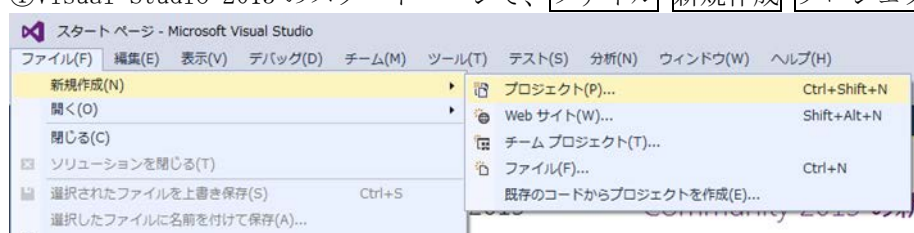
アプリケーションは以下の手順で作成します。

1. CA-SDK をインストールする。
2. VisualBasic 2013 のプロジェクトを作成する。
3. CA-SDK の参照設定をする。
4. アプリケーションの GUI/コードを作成する。

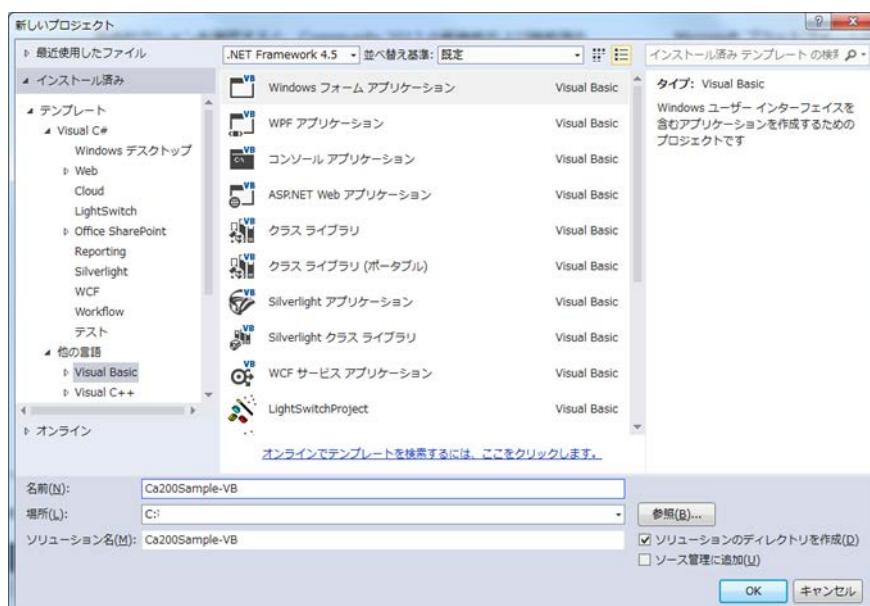
VisualBasic 2013 での作業を始める前に、“1. CA-SDK をインストールする。”を行ってください。
以下、上記の“2. VisualBasic 2013 のプロジェクトを作成する。”以降の説明をします。

3-1 VisualBasic 2013 のプロジェクト作成

①Visual Studio 2013 のスタートページで、**ファイル** **新規作成** **プロジェクト**を選択します。



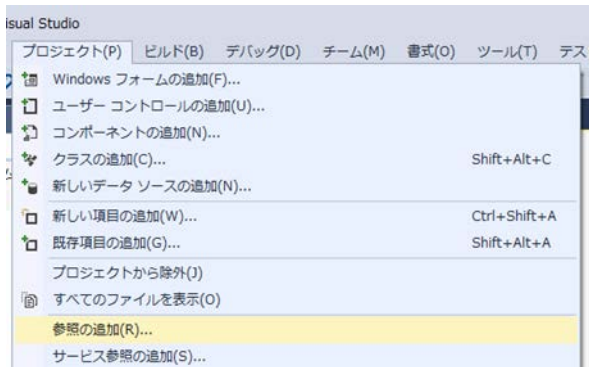
②新しいプロジェクトダイアログで、テンプレートとして、「Windows フォームアプリケーション」を選択して、プロジェクト名として“Ca200Sample-VB”を入力し、OK をクリックします。



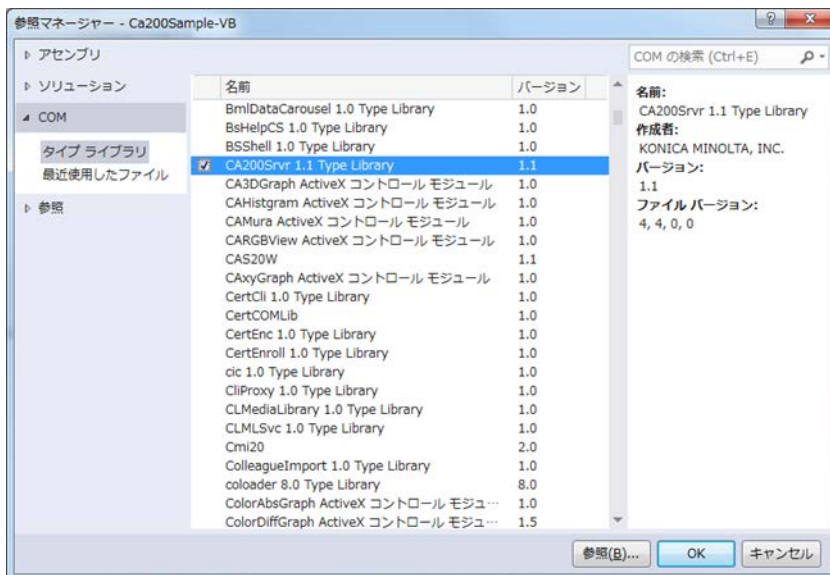
プロジェクトが作成され、フォームのデザイン画面が表示されます。

3-2 CA-SDK の参照設定

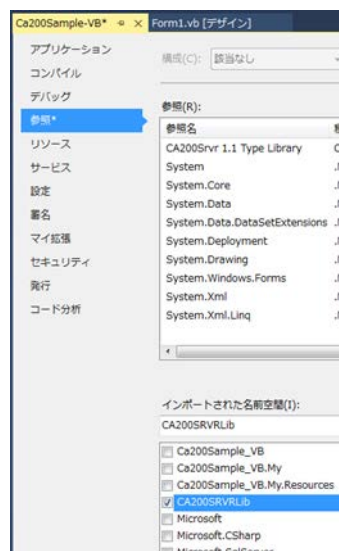
① メニューの **プロジェクト** - **参照の追加** を選択します。



② 参照マネージャー左側で、COM を選択、表示されるリストから CA200Srvr 1.1 Type Library のチェックボックスを ON にして、OK ボタンをクリックします。



③ メニューの **プロジェクト** - **Ca200Sample-VB のプロパティ** を選択し、左側の参照タブをクリックします。画面下側の、「インポートされた名前空間」で、「CA200SRVRLib」をチェック ON します。



3-3 アプリケーションの GUI/コードの作成

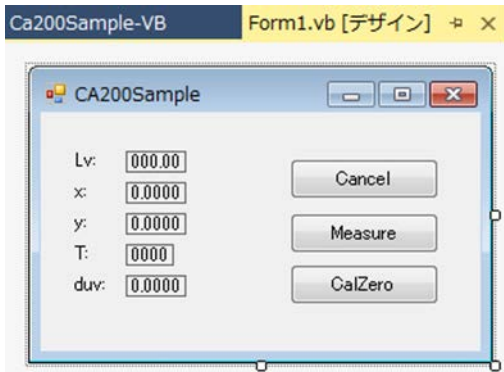
サンプルソフトのGUIを以下のようにします。

ラベルを 10 個（ラベル用・測定値表示用各 5 個）、ボタンを 3 個利用します。

Cal Zeroボタンをクリックすると、CA本体で、ゼロ校正を実行します。

Measureボタンをクリックすると、20回連続測定し、測定結果を測定値表示用ラベルに表示します。

Cancelボタンをクリックすると、測定を途中で中止します。



サンプルコードを以下に示します。

なお、VisualBasic 2013 では、従来の VB と同様の On Error Goto によるエラー処理のほか、Try...catch による構造化例外処理も記述できます。

```
Public Class Form1
```

```
    Private objCa200 As Ca200
```

```
    Private WithEvents objCa As Ca
```

```
    Private objProbe As Probe
```

```
    Private isMsr As Boolean
```

```
    Private Sub Form1_FormClosing(sender As Object, e As FormClosingEventArgs) Handles Me.FormClosing
```

```
        objCa.RemoteMode = 0
```

```
        objCa200 = Nothing
```

```
        objCa = Nothing
```

```
        objProbe = Nothing
```

```
    End Sub
```

```
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles Me.Load
```

```
        On Error GoTo Error1
```

```
        objCa200 = New Ca200
```

```
        objCa200.AutoConnect()
```

```
        objCa = objCa200.SingleCa
```

```
        objProbe = objCa.SingleProbe
```

```
        ButtonCancel.Enabled = False
```

```
        ButtonMeasure.Enabled = True
```

```
        ButtonCalZero.Enabled = True
```

```
    Exit Sub
```

```
Error1:
```

```
    DisplayError()
```

```
End
```

End Sub

Private Sub DisplayError()

```
Dim msg As String
msg = "Error from" + Err.Source + vbCrLf
msg = msg + Err.Description + vbCrLf
msg = msg + "HR:" + (Err.Number - vbObjectError).ToString
MessageBox.Show(msg)
```

End Sub

Private Sub ButtonCancel_Click(sender As Object, e As EventArgs) Handles ButtonCancel.Click

```
isMsr = False
ButtonCancel.Enabled = False
ButtonMeasure.Enabled = True
ButtonCalZero.Enabled = True
```

End Sub

Private Sub ButtonMeasure_Click(sender As Object, e As EventArgs) Handles ButtonMeasure.Click

```
Dim i As Integer

On Error GoTo Error2

isMsr = True
ButtonCancel.Enabled = True
ButtonMeasure.Enabled = False
ButtonCalZero.Enabled = False
For i = 1 To 20
    objCa.Measure()
    LabelLv.Text = objProbe.Lv.ToString("##0.00")
    Labelx.Text = objProbe.sx.ToString("0.0000")
    Labely.Text = objProbe.sy.ToString("0.0000")
    LabelT.Text = objProbe.T.ToString("####")
    Labelduv.Text = objProbe.duv.ToString("0.0000")
    Application.DoEvents()
    If isMsr = False Then
        Exit For
    End If
Next

ButtonCancel.Enabled = False
ButtonMeasure.Enabled = True
ButtonCalZero.Enabled = True
Exit Sub
```

Error2:

```
DisplayError()
End
```

End Sub

Private Sub ButtonCalZero_Click(sender As Object, e As EventArgs) Handles ButtonCalZero.Click

```
Dim calzero_success As Boolean = False
```

```

While calzero_success = False
    ButtonMeasure.Enabled = False
    ButtonCalZero.Enabled = False
    Try
        objCa.CalZero()
        calzero_success = True
    Catch er As Exception
        DisplayError()
        If MessageBox.Show("Zero Cal Error" + vbCrLf + "Retry?", "CalZero",
MessageBoxButtons.OKCancel) = DialogResult.Cancel Then
            objCa.RemoteMode = 0
            End
        End If
    End Try
End While

ButtonMeasure.Enabled = True
ButtonCalZero.Enabled = True

End Sub

Private Sub objCa_ExeCalZero() Handles objCa.ExeCalZero
    If MessageBox.Show("CalZero?", "CalZero", MessageBoxButtons.OKCancel) = DialogResult.Cancel Then
        Exit Sub
    End If
    ButtonMeasure.Enabled = False
    ButtonCalZero.Enabled = False

    Try
        objCa.CalZero()
    Catch er As Exception
        DisplayError()
    End Try

    ButtonMeasure.Enabled = True
    ButtonCalZero.Enabled = True

End Sub
End Class

```

CA-SDK 使用説明書

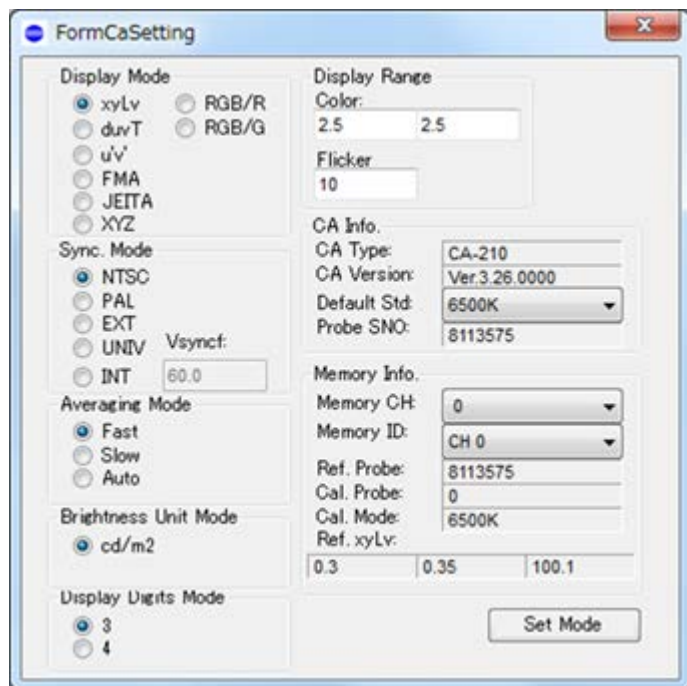
4. CA-SDK サンプルソフト用 コントロールの使用方法

CA-SDK の VB サンプルソフトでは CaControl, xyControl, VGControl の 3 つのコントロールを使用しています。また、VB サンプルソフトはソースコードを全て公開しています。CA-SDK を利用してアプリケーションソフトを作成する際に参考になさってください。VB サンプルソフトは、**スタートメニュー→すべてのプログラム→KONICAMINOLTA-CA-SDK** に各ソフト exe ファイルへのショートカットと、使い方を説明した ReadMe.htm を用意しています。

コントロールはいずれも VB サンプルソフト用に作成されたものですが、他のソフトで利用することもできます。ここでは、そのうちの CA 測定器の各種セッティングを行うコントロール CaControl と、測定結果をxy色度図表示するコントロール xyControl について、使用法を説明します。

4-1 CaControl(CA 測定器セッティング用コントロール)

CaControl の”…”ボタンをクリックすると、下に示す”FormCaSetting”ダイアログが表示されます。このダイアログを使って、CA 測定器の各種設定をすることができます。



4-1-1 CaControl のプロパティ・メソッド

Ca プロパティ

セッティング対象のCaオブジェクトを設定/取得します。
コントロールを使用する前に、このプロパティにより、制御するCaオブジェクトを設定します。

構文:

```
Dim objCaControl as CaControl  
Dim objCa as Ca
```

```
objCaControl.Ca = objCa
```

Probe プロパティ

セッティング対象のProbeオブジェクトを設定/取得します。
コントロールを使用する前に、このプロパティにより、制御するProbeオブジェクトを設定します。

構文:

```
Dim objCaControl as CaControl
```

```
Dim objProbe as Probe
```

```
objCaControl.Probe = objProbe
```

Memory プロパティ

セッティング対象のMemoryオブジェクトを設定/取得します。
コントロールを使用する前に、このメソッドによりコントロールを使用して制御するMemoryオブジェクトを設定します。

構文:

```
Dim objCaControl as CaControl
```

```
Dim objMemory as Memory
```

```
objCaControl.Memory = objMemory
```

UpdateCaInfo メソッド

コントロールに現在のCA測定器の状態(メモリチャンネルを除く)を反映させます。
コントロールに必要なCA-SDKオブジェクトを設定した後、コントロールを使用する前に、一度はこのメソッドを実行してください。

構文:

```
Dim objCaControl as CaControl
```

```
objCaControl.UpdateCaInfo()
```

UpdateMemoryInfo メソッド

コントロールに現在のCA測定器のメモリチャンネルの状態を反映させます。
コントロールに必要なCA-SDKオブジェクトを設定した後、コントロールを使用する前に、一度はこのメソッドを実行してください。

構文:

```
Dim objCaControl as CaControl
```

```
objCaControl.UpdateMemoryInfo()
```

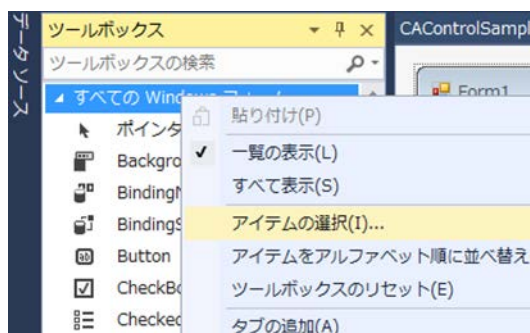
Update イベント

コントロールにより、CA 測定器のセッティングが変更されたとき、発行されます。

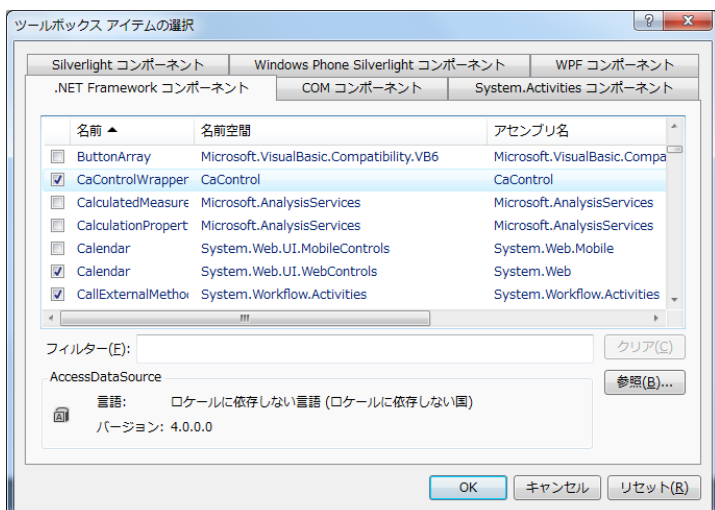
4-1-2 VisualC# 2013 での利用法

以下の手順で、コントロールを VisualC#プロジェクトに追加します。

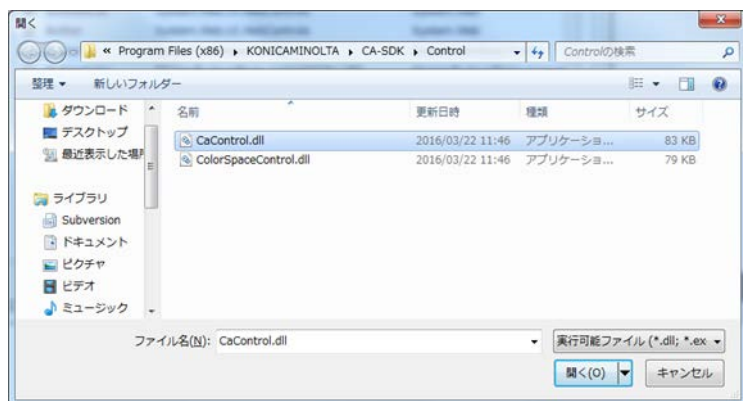
- ①メニューで表示-ツールボックスを選択し、ツールボックスを表示します。
- ②ツールボックス内で右クリック-アイテムの選択...をクリックします。



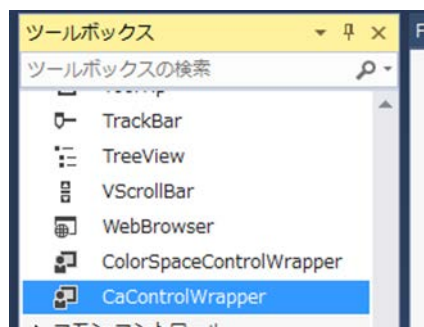
③ .NET Framework コンポーネントから”CaControlWrapper”をチェック ON し、OK をクリックします。



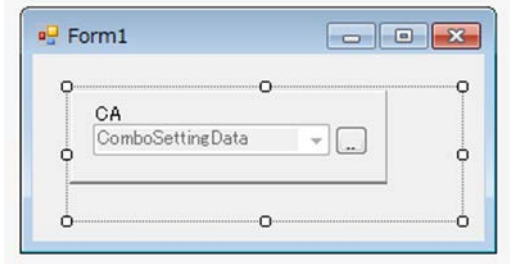
CA-SDK をインストールしているのに CaControlWrapper が表示されない場合は、**参照...**ボタンを押し、CA-SDK がインストールされているフォルダーから CaControl1.dll を選択してください。



コントロールの追加が終わると、フォームエディターのツールボックスに、コントロールアイコンが表示され、編集できる状態になります。



④ CaControlWrapper をフォームに追加します。



以下に、サンプルコードを示します。

SDK のオブジェクトを生成した後、コントロールのプロパティにより、CA 測定器を制御するためのオブジェクトをコントロールに設定します。その後、UpdateCaInfo、UpdateMemoryInfo で、コントロールに現在の CA 測定器の状態により、コントロールを初期化します。

以上の操作で、コントロールが使用できる状態になります。

```
namespace CAControlSample
{
    public partial class Form1 : Form
    {
        private CA200SRVRLib.Ca200 objCa200;
        private CA200SRVRLib.Ca objCa;
        private CA200SRVRLib.Probe objProbe;
        private CA200SRVRLib.Memory objMemory;
        private CA200SRVRLib.IProbeInfo objProbeInfo;

        private CaControl.CaControl objCaControl;

        public Form1()
        {
            InitializeComponent();

            objCa200 = new CA200SRVRLib.Ca200();
            objCa200.AutoConnect();
            objCa = objCa200.SingleCa;
            objProbe = objCa.SingleProbe;
            objMemory = objCa.Memory;
            objProbeInfo = (CA200SRVRLib.IProbeInfo)objProbe;

            objCaControl = caControlWrapper1.cacontrolobj;

            objCaControl.Ca = objCa;
            objCaControl.Probe = objProbe;
            objCaControl.Memory = objMemory;

            objCaControl.UpdateCaInfo();
            objCaControl.UpdateMemoryInfo();
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            objCa.RemoteMode = 0;
        }
    }
}
```

```

objCa200 = null;
objCa = null;
objProbe = null;
objMemory = null;
objProbeInfo = null;

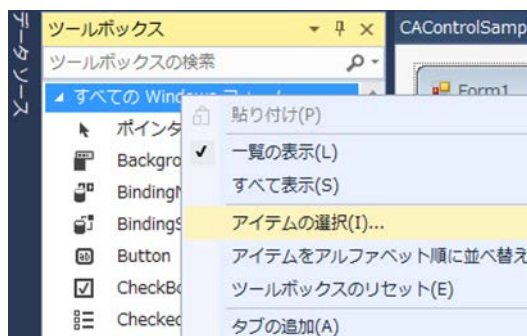
objCaControl = null;
}
}
}

```

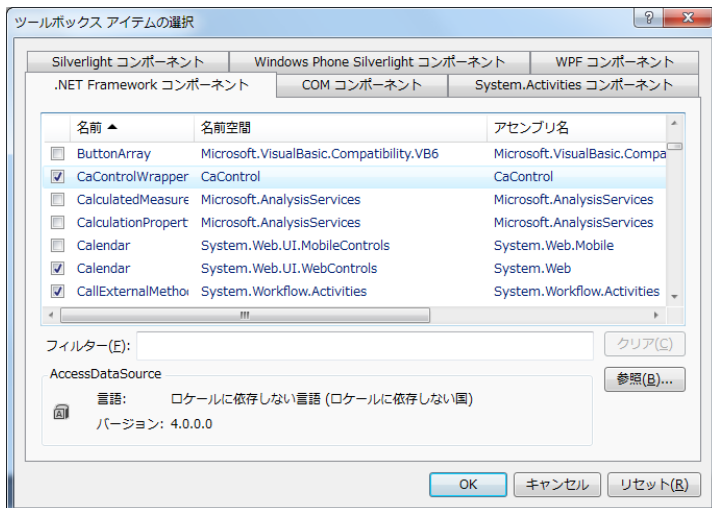
4-1-3 VisualBasic 2013 での利用法

以下の手順で、コントロールを VisualBasic プロジェクトに追加します。

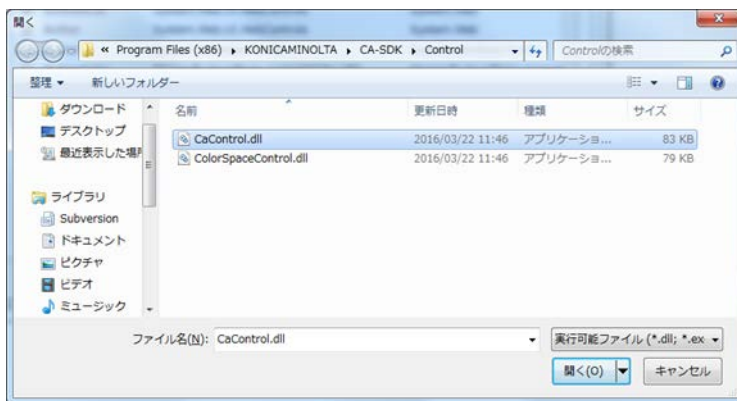
- ①メニューで表示-ツールボックスを選択し、ツールボックスを表示します。
- ②ツールボックス内で右クリック-アイテムの選択...をクリックします。



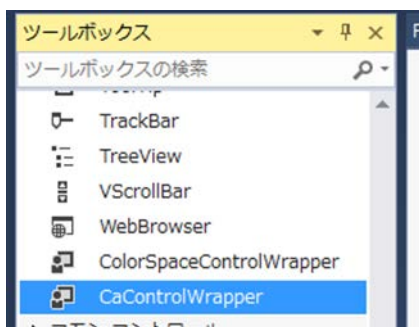
- ③ .NET Framework コンポーネントから”CaControlWrapper”をチェック ON し、OK をクリックします。



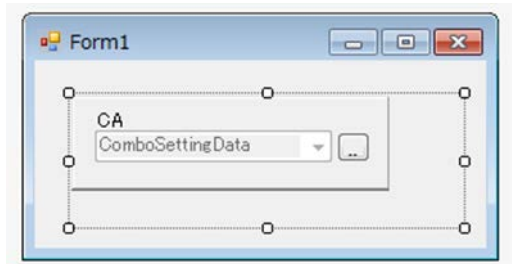
CA-SDK をインストールしているのに CaControlWrapper が表示されない場合は、参照...ボタンを押し、CA-SDK がインストールされているフォルダーから CaControl1.dll を選択してください。



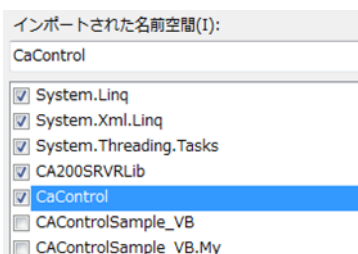
コントロールの追加が終わると、フォームエディターのツールボックスに、コントロールアイコンが表示され、編集できる状態になります。



- ④ CaControlWrapper をフォームに追加します。



- ⑤ メニューでプロジェクト(プロジェクト名)のプロパティを選択し、参照タブをクリックします。インポートされた名前空間の CaControl をチェック ON します。



以下に、サンプルコードを示します。

SDK のオブジェクトを生成した後、コントロールのプロパティにより、CA 測定器を制御するためのオブジェクトをコントロールに設定します。その後、UpdateCaInfo、UpdateMemoryInfo で、コントロールに現在の CA 測定器の状態により、コントロールを初期化します。

以上の操作で、コントロールが使用できる状態になります。

```

Public Class Form1
    ' =====
    ' SDK Object
    ' =====

    Private objCa200 As Ca200
    Private objCa As Ca
    Private objProbe As Probe
    Private objMemory As Memory
    Private objProbeInfo As IProbeInfo

    Private objCaControl As CaControl.CaControl

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ' =====
        ' Create SDK/Application Object
        ' =====
        objCa200 = New Ca200

        ' =====
        ' Set Configuration
        ' =====
        objCa200.AutoConnect()

        ' =====
        ' Initialize SDK Object
        ' =====
        objCa = objCa200.SingleCa
        objProbe = objCa.SingleProbe
        objMemory = objCa.Memory
        objProbeInfo = objProbe

        objCaControl = CaControlWrapper1.cacontrolobj

        objCaControl.Ca = objCa
        objCaControl.Probe = objProbe
        objCaControl.Memory = objMemory

        objCaControl.UpdateCaInfo()
        objCaControl.UpdateMemoryInfo()
    End Sub

    Private Sub Form1_FormClosing(sender As Object, e As FormClosingEventArgs) Handles
MyBase.FormClosing
        objCa.RemoteMode = 0
        objCa200 = Nothing
        objCa = Nothing
        objProbe = Nothing
        objMemory = Nothing
        objProbeInfo = Nothing

        objCaControl = Nothing
    End Sub
End Class

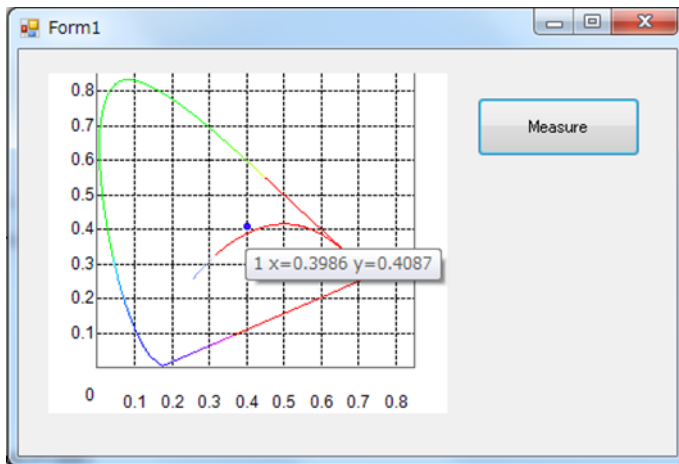
```

4-2 xyControl(xy色度図表示コントロール)

xyControl は CA 測定器の色測定結果をxy色度図にプロットします。xyControl には約 1000 個のデータを登録できます。

自動的にデータ登録/表示する機能のほか、表示するデータを、インデックスで指定する機能(複数表示可)があります。

最大で、400%の拡大表示機能があります(マウス右クリックで倍率メニューが表示されます)。データプロットにマウスを近づけると、データの通し番号(インデックス+1の値)と色度値を表示します。



4-2-1 プロパティ/メソッド

Ca プロパティ

測定結果を表示するCaオブジェクトを設定します。
コントロールを使用する前に、設定します。

構文：

```
Dim objxyControl as xyControl  
Dim objCa as Ca
```

```
objxyControl.Ca = objCa
```

Probe プロパティ

測定結果を表示するProbeオブジェクトを設定します。
コントロールを使用する前に、設定します。

構文：

```
Dim objxyControl as xyControl  
Dim objProbe as Probe
```

```
objxyControl.Probe = objProbe
```

SetXYGraphData メソッド

Probe オブジェクトから現在の x y 測定結果を取得・データ登録し、x y 色度図に表示します。
データのインデックスはインクリメントされます。
それ以前のデータの表示は不可視に設定されます。

構文：

```
Dim objxyControl as xyControl
```

```
objxyControl.SetXYGraphData()
```

AddXYGraphData メソッド

Probe オブジェクトから現在の x y 測定結果を取得、指定したインデックスのデータとして登録し、 x y 色度図に表示します。

インデックスは指定したインデックスに設定されます。

他のインデックスのデータの表示状態を変更することはありません。

構文：

```
Dim objxyControl as xyControl
```

```
Dim lIndex as long
```

```
objxyControl.AddXYGraphData(lIndex)
```

SetXYData メソッド

指定したインデックス、 $x \cdot y$ 色度値でデータを登録し、 x y 色度図に表示します。

インデックスは指定したインデックスに設定されます。

他のインデックスのデータの表示状態を変更することはありません。

構文：

```
Dim objxyControl as xyControl
```

```
Dim lIndex as long
```

```
Dim fx as float
```

```
Dim fy as float
```

```
objxyControl.SetXYData(lIndex, fx, fy)
```

SetVisible メソッド

指定したインデックスのデータ表示を可視に設定します。

構文：

```
Dim objxyControl as xyControl
```

```
Dim lIndex as long
```

```
objxyControl.SetVisible(lIndex)
```

SetVisibleAll メソッド

指定した Boolean の値 True/False にしたがって、全データの表示を可視/不可視に設定します。

構文：

```
Dim objxyControl as xyControl
```

```
Dim lIndex as long
```

```
Dim bVisible as Boolean
```

```
objxyControl.SetVisibleAll(bVisible)
```

ClearData メソッド

登録されている全データの内容を削除し、インデックスを初期化します。

構文：

```
Dim objxyControl as xyControl

objxyControl. ClearData()
```

4-2-2 VisualC# 2013 での利用法

4-1-2で説明した手順で、xyControl コントロールを VisualC#プロジェクトに追加してください。

以下にサンプルコードを示します。

SDK のオブジェクトを生成した後、コントロールのプロパティにより、CA 測定器を制御するためのオブジェクトをコントロールに設定します。その後、ClearData でコントロールのデータを初期化します。

色測定を実行すると測定結果のxy値を色度図にプロットします。これは、SetXYGraphData で実行しています。SetXYGraphData メソッドは最新のデータのみ表示します。

```
namespace xyControlSample
{
    public partial class Form1 : Form
    {
        private CA200SRVRLib.Ca200 objCa200;
        private CA200SRVRLib.Ca objCa;
        private CA200SRVRLib.Probe objProbe;
        private CA200SRVRLib.Memory objMemory;
        private CA200SRVRLib.IProbeInfo objProbeInfo;

        private ColorSpaceControl.xyControl objxyControl;

        public Form1()
        {
            InitializeComponent();

            objCa200 = new CA200SRVRLib.Ca200();
            objCa200.AutoConnect();
            objCa = objCa200.SingleCa;
            objProbe = objCa.SingleProbe;
            objMemory = objCa.Memory;
            objProbeInfo = (CA200SRVRLib.IProbeInfo)objProbe;

            objxyControl = colorSpaceControlWrapper1.xycontrolobj;
            objxyControl.Probe = objProbe;
            objxyControl.Ca = objCa;
            objxyControl.ClearData();
        }

        private void button_Measure_Click(object sender, EventArgs e)
        {
            objCa.Measure();
            objxyControl.SetXYGraphData();
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            objCa.RemoteMode = 0;
        }
    }
}
```

```

        objCa200 = null;
        objCa = null;
        objProbe = null;
        objMemory = null;
        objProbeInfo = null;

        objxyControl = null;
    }
}
}

```

4-2-3 VisualBasic 2013 での利用法

4-1-3で説明した手順で、xyControl コントロールを VisualBasic プロジェクトに追加してください。

以下にサンプルコードを示します。

SDK のオブジェクトを生成した後、コントロールのプロパティにより、CA 測定器を制御するためのオブジェクトをコントロールに設定します。その後、ClearData でコントロールのデータを初期化します。

色測定を実行すると測定結果のxy値を色度図にプロットします。これは、SetXYGraphData で実行しています。SetXYGraphData メソッドは最新のデータのみ表示します。

```

Public Class Form1
    Public objCa200 As Ca200
    Public objCa As Ca
    Public objProbe As Probe
    Public objMemory As Memory
    Public objProbeInfo As IProbeInfo

    Public objxyControl As ColorSpaceControl.xyControl

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ' =====
        ' Create SDK/Application Object
        ' =====
        objCa200 = New Ca200

        ' =====
        ' Set Configuration
        ' =====
        objCa200.AutoConnect()

        ' =====
        ' Initialize SDK Object
        ' =====
        objCa = objCa200.SingleCa
        objProbe = objCa.SingleProbe
        objMemory = objCa.Memory
        objProbeInfo = objProbe

        objxyControl = ColorSpaceControlWrapper1.xycontrolobj
        objxyControl.Probe = objProbe
        objxyControl.Ca = objCa
        objxyControl.ClearData()
    End Sub
End Class

```


End Sub

```
Private Sub Button_Measure_Click(sender As Object, e As EventArgs) Handles Button_Measure.Click
    objCa.Measure()
    objxyControl.SetXYGraphData()
End Sub
```

```
Private Sub Form1_FormClosing(sender As Object, e As FormClosingEventArgs) Handles
MyBase.FormClosing
    objCa.RemoteMode = 0
    objCa200 = Nothing
    objCa = Nothing
    objProbe = Nothing
    objMemory = Nothing
    objProbeInfo = Nothing

    objxyControl = Nothing
End Sub
End Class
```

CA-SDK 使用説明書

5. アプリケーション例

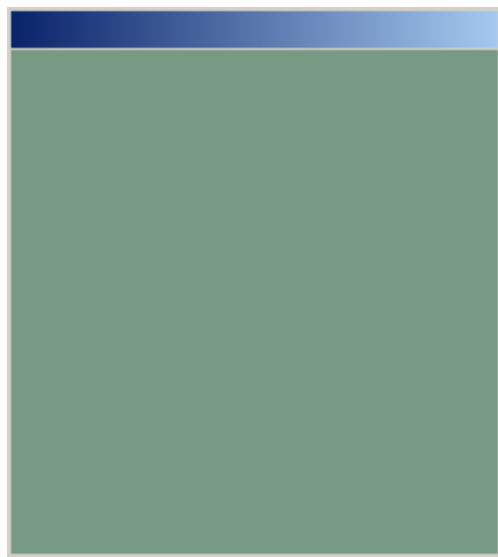
5-1 ホワイトバランス調整

5-1-1 ホワイトバランス調整

複数台の液晶モニターに白色を出した場合、色が違って見える場合があります。
この原因は、液晶モニターを構成する部品（フィルタ、回路等）にバラツキがあり、完全に同じ白色にならないからです。
例えば、部品バラツキにより、 x 、 y の色度が各々0.02異なると、下図のような色となります。



$x=0.310, y=0.332, L_v=172.5$
目標の色度、輝度



$x=0.292, y=0.356, L_v=170.3$
調整されていない状態の色度、輝度

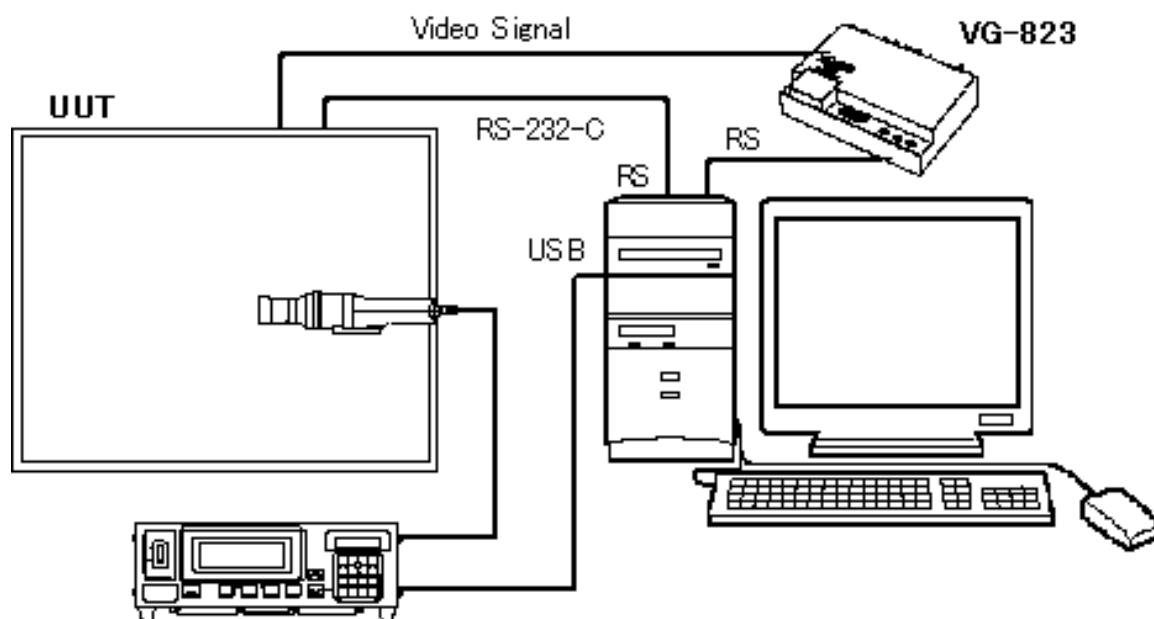
他方、最近、液晶モニターで画像をみるケースが増えてきており、従来以上に色度、輝度に対する要求が厳しくなっています。このバラツキを少なくするために、液晶モニターのホワイトバランス調整をおこなうケースが増えてきています。



$x=0.315, y=0.337, L_v=169.4$
調整後のパターン図

以下に、CA-210を使用した白色の輝度、色度を高速に、精度よく調整、及びコントラストを検査するシステム例を説明します。

5-1-2 システム構成ブロック図



CA-210 + 1 probe 輝度、色度測定用

UUT(Unit Under Test) 調整対象液晶モニター

外部から、校正定数を書き込み可能タイプ 本例ではRS-232-Cを使用

PC 測定器、液晶モニター、パターンジェネレータ制御用パソコン

USB port × 1(for CA-210)

RS-232-C Port × 2 (for LCD Control & VG control)

PentiumⅢ 660MHz 以上

解像度：XGA

VG-823,813,812 被測定モニターへのパターン制御で使用

アストロデザイン社製パターンジェネレータ

注：I²C使用時は、RS-232-CからI²C変換基板が必要。

5-1-3 ホワイトバランス調整ソフトの機能

- ・ 液晶モニターを設定された色度、輝度になるように、液晶モニターのゲインを変更して、調整を行います。最大6種類の設定を、続けて行うことができます。
- ・ コントラスト検査
- ・ 合否判定機能

5-1-4 ホワイトバランス調整時間

調整精度が、

輝度Lv ±10cd/m² 色度±10/1000 以内 なら、5秒以下

輝度Lv ±5cd/m² 色度±5/1000 以内 なら、10秒以下

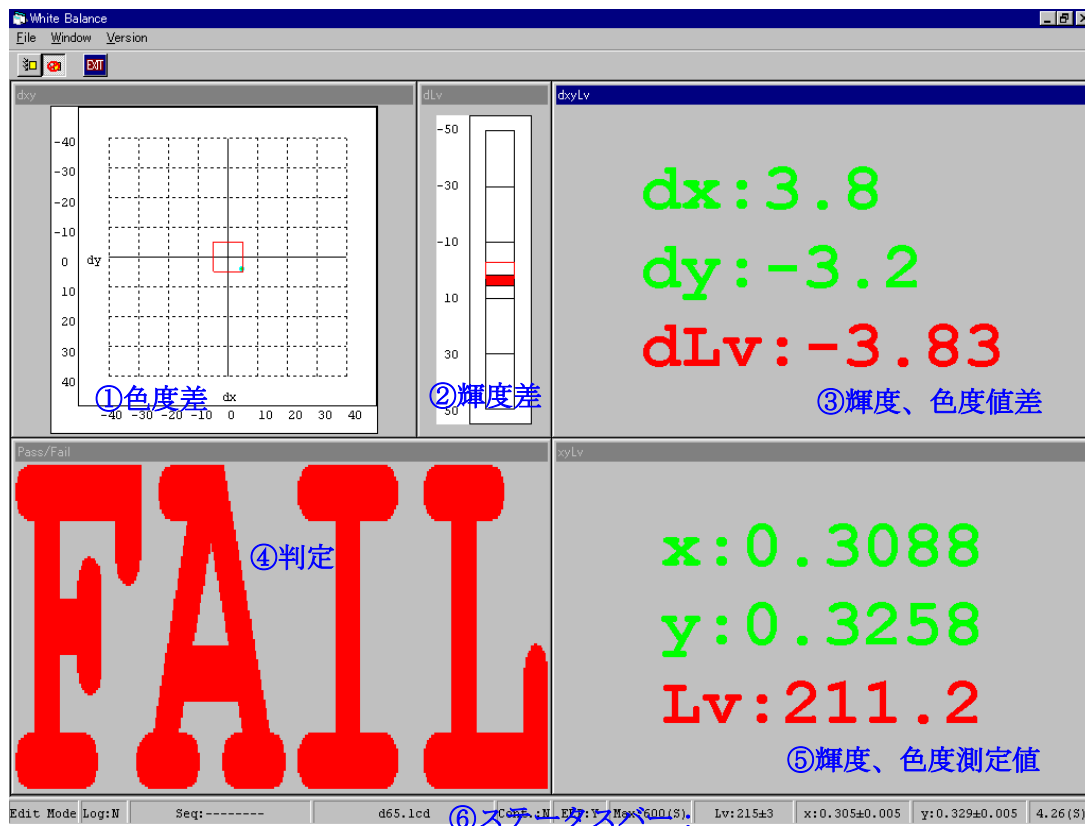
動作環境:富士通FMV-6600MF8/X PⅢ 660MHz メモリー128M時。

注) 液晶モニターにより、調整時間は変化します。

5-1-5 GUI

調整画面

調整の状態を視覚的に表示します。



図の見方

設定されている目標値 $L_v : 215 \pm 3$ 、

$x : 0.305 \pm 0.005$ 、 $y : 0.329 \pm 0.005$

(図番号と対応しています。)

- ①色度差：設定されている目標値が赤い枠で表示されています。その枠内に入るように自動調整を行います。
- ②輝度差：設定されている目標値が赤い枠で表示されています。その枠内に入るように自動調整を行います。
- ③輝度、色度差：設定されている目標値に対しての差を表示します。目標値内時、文字が緑色になります。
- ④判定：輝度、色度ともに目標値内で、Passとなります。Pass時は、文字が緑色になります。
- ⑤輝度、色度測定値：測定値そのものを表示します。目標値内時、文字が緑色になります。
- ⑥ステータスバー：目標輝度、色度などの設定状態、調整時間などの表示を行います。

上記のように、CA-310/210 を用いれば、高速、高精度でのホワイトバランス調整を行うことが可能となります。

5-2 γ 調整

5-2-1 γ 調整

複数台の液晶モニターに同じカラー画像を出した場合、中間階調の色が、違って見える場合があります。この原因は、液晶モニターを構成する部品（フィルタ、回路等）にバラツキがあり、完全に同じ中間階調の色にならないからです。

例えば、部品バラツキにより、R, G, Bの中間調の明るさが違うと下図のような色あいとなります。



原画

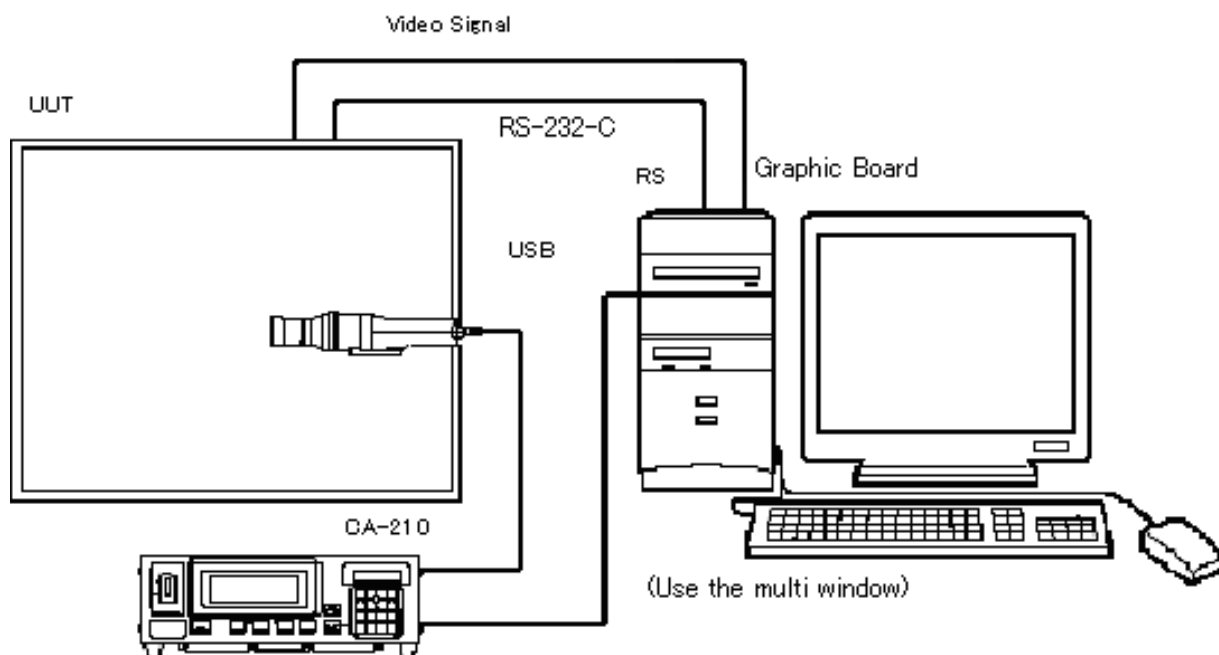


中間調の B l u e が強い状態

他方、最近、液晶モニターで画像をみるケースが増えてきており、従来以上に中間調に対する要求が厳しくなっています。このバラツキを少なくするために、液晶モニターの γ 調整をおこなうケースが出てきています。

以下に、CA-310/210 を使用して、この中間階調の各色のバランスを高速に、精度よく調整するシステム例を説明します。

5-2-2 システム構成ブロック図



CA-210 + 1 probe 輝度、色度測定用
 UUT(Unit Under Test) 調整対象液晶モニター
 外部から、校正定数を書き込み可能タイプ 本例ではRS-232Cを使用
 PC 測定器、液晶モニター制御用パソコン
 USB port × 1(for CA-210)
 RS-232C Port × 1 (for LCD)
 PentiumⅢ 660MHz 以上
 解像度：XGA、True Color 24bit 以上
 Graphic Board True color (24Bit 以上) Graphic Board 型番指定します。
 あるいは、VG-823,813,812 アストロデザイン社製パターンジェネレータ

5-2-3 調整方法

被測定モニターへのパターン表示は、PC用のグラフィックカードから、出力します。
 PCの設定をマルチウィンドウに設定し、PC本体側にソフト表示、グラフィックカードから、被測定パターンを表示します。

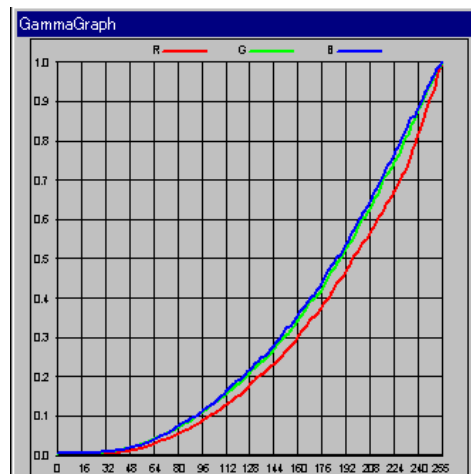
測定の方法は、IEC61966-4に
 規定された3刺激値により計算をしています。

横軸：0-255階調
 縦軸：各色最大値での正規化

① γ 測定

まず、液晶本来の γ 特性を任意の階調で、測定します。
 (階調:8、16、32、64、128、256)

①



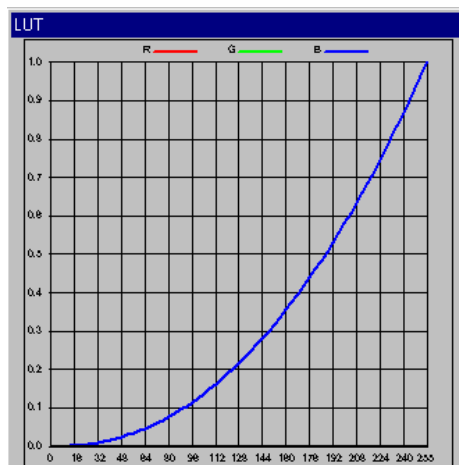
② γ 値設定

γ 特性を設定します。
 右図は、 $\gamma = 2.2$ の例です。
 $\gamma \text{ curve} = \text{step}^{2.2}$ の計算から、設定データを作成します。

①で測定したデータから、右図の設定したデータになるように
 LUT(Look Up Table)を計算し、
 液晶モニターに書き込みます。

LUTの計算方法は、例えば、階調16での
 データの場合、 $(16^{2.2}) / (255^{2.2})$ (: 値 / 最大値) の値が
 測定したデータのどこにあるかを調べて、
 その上下2点のデータから、線形補間して、
 階調16でのLUT値を求めます。

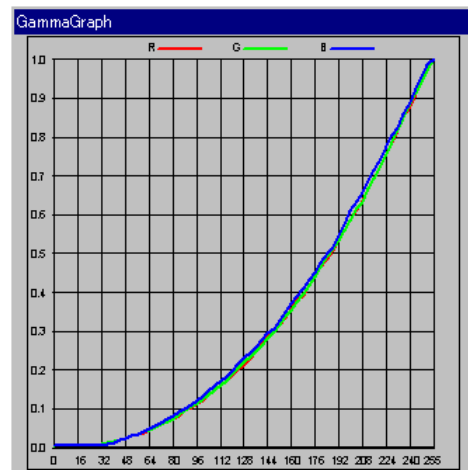
②



③ γ 検査

LUTを書き込んだ液晶をもう一度測定し、
正確に反映されている事を検証します。
右図がその例です。

③



5-2-4 γ 調整時間

CA-310/210 を用いて、R/G/B 3色の γ 調整を実施した場合の調整時間を下表に示します。

注: 表示を切り替えてから、液晶の応答安定に100msのウェイトを入れています。

液晶モニターへのLUT書き込み時間は含まれていません。

動作環境: FMV-6600MF8/X PIII 660MHz メモリー128M

γ 調整時間 [秒]

階調数	8	16	32	64	128	256
調整時間	6	8	12	19	34	64

上記のように、CA-210を用いれば、高輝度から低輝度までの γ 調整を高速に行え、中間色も正確に表現可能になります。

※IEC-61966-4

Method of measurement

The centred colour patches shall be displayed for equally stepped values of input data from

$\frac{1}{m}2^N$ to $M=2^N-1$, where m is the number of data and should be at least 32, and N is the

number of bits per channel. For the red channel measurement, $D_G = D_B = 0$; for the green channel, $D_R = D_B = 0$, and for the blue channel, $D_R = D_G = 0$ shall be kept, respectively.

The readings of the colorimeter for each colour patch on the LCD shall be recorded successively and noted as X^i_c , Y^i_c , Z^i_c where the subscript C shall be replaced by R, G and B, for the red, green, and blue channels, respectively; and the subscript i corresponds to measurement steps, $i = 1, 2, \dots, m$.

The measured tristimulus values shall be normalized by the values corresponding to the maximum excitation for the last step m with input data $M=2^N-1$,

$$X''_{ic} = \frac{X^i_c}{X^m_c}$$

$$Y''_{ic} = \frac{Y^i_c}{Y^m_c}$$

$$Z''_{ic} = \frac{Z^i_c}{Z^m_c}$$

where the subscript C shall be replaced by R, G and B.

X, Y, Z : Measured raw data using spectroradiometers and colorimeters corresponding to tristimulus values.

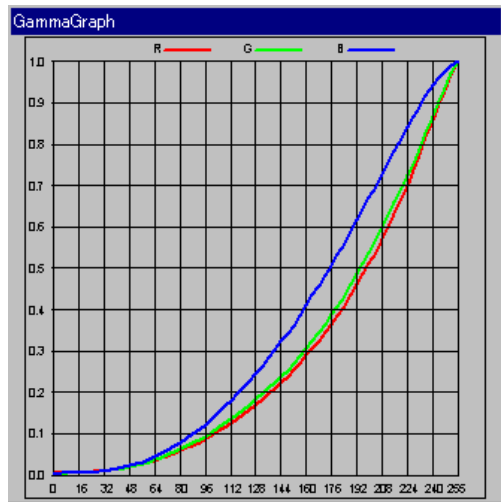
5-2-5 GUI

調整画面例

測定時の状態を視覚的に表示します。

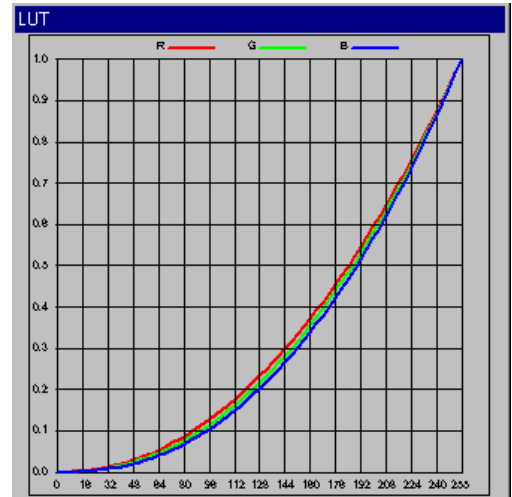
① γ 測定画面

設定された測定階調に応じて、液晶モニターの γ 特性を測定、表示する画面です。



② LUT設定表示画面

液晶モニターへ書き込む設定データを表示します。この場合、R:2.1, G:2.2, B:2.3に設定されています。



③ γ 確認画面

LUTを書き込んだ液晶モニターを測定し、その γ 特性を表示します。

