

CA-SDK
Instruction Manual
Ver.4.5



KONICA MINOLTA

Trademarks:

- Windows, Visual Studio, Visual Basic, and Visual C# are trademarks or registered trademarks of Microsoft Corp.
- Company names and product names mentioned in this document are trademarks or registered trademarks of their respective companies.

•Note that this manual abbreviates product names as follows.

<i>Name Given in This Manual</i>	<i>Official Name</i>
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

Notice Regarding Component Names:

- In this document, the component names continue to be listed as "Minolta" in some cases.

Notice:

- Copy or reproduction of all or any part of the contents of this manual without Konica Minolta's permission is strictly prohibited.
- The contents of this manual are subjects to change without prior notice.
- Every effort has been made in the preparation of this manual to ensure the accuracy of its contents. However, should you have any questions or find any errors, please contact a Konica Minolta authorized service facility.
- The sample code described in this document is for you to reference your software development. But Konica Minolta will not guarantee the accuracy of the sample code.
- Konica Minolta will not accept any responsibility for consequences arising from the use of the software.

Table of Contents

1. Install Guide of CA-SDK-----	2
1-1 How to Install (Or Uninstall) CA-SDK-----	3
1-2 Installing the USB Driver-----	4
2. Creating a VisualC# 2013 application using CA-SDK-----	6
2-1 Creating the VisualC# 2013 project-----	7
2-2 Setting references to CA-SDK-----	8
2-3 Creation of Application GUI/Code-----	9
3. Creating a VisualBasic 2013 application using CA-SDK-----	13
3-1 Creating the VisualBasic 2013 project-----	14
3-2 Setting references to CA-SDK-----	15
3-3 Creation of Application GUI/Code-----	16
4. How to use CA-SDK Sample Software Controls-----	19
4-1 CaControl (for Settings of CA-Series Instruments)-----	20
4-2 xyControl (for Displaying Data in xy Color Space)-----	28
5. Application Examples-----	33
5-1 White Balance Adjustment System-----	34
5-2 Gamma Adjustment System-----	37

CA-SDK

Instruction Manual

1. Install Guide of CA-SDK

1-1 How to Install (or Uninstall) CA-SDK

System Requirements

This SDK requires the following environment.

- DOS/V computer running the Windows 7, Windows 8.1, or Windows 10 operating system.
- COM-compliant development tools (VB, etc.) installed on the computer.
- The computer must be equipped with an RS-232C port or a USB1.1-compliant USB port.

1-1-1 Installing CA-SDK

【Notice】 If an older version of CA-SDK has been installed, you need to uninstall it before installing this version.

- ① Insert the SDK's Setup disk into the computer's CD-ROM drive.
- ② Open (My) Computer and open the folder ("x64" folder for 64-bit version or "x86" folder for 32-bit version) of CD-ROM drive.
- ③ Double-click "setup.exe" in the folder.

Note: If you are connecting to the CA units by USB, you must also install the CA-200 USB driver. For information, refer to Section 1-2, "Installing the USB Driver."

Additional information on installing/uninstalling the SDK:

When installing or uninstalling the SDK, be sure that you are logged in with Administrator rights.
If you are not sure of the current access rights, check with your system administrator.

Related points:

After installation, problems may occur if a user with limited rights attempts to change the pattern-generator settings when using the sample software to control a pattern generator. In this case, it is necessary to give the user suitable access permission for the SDK installation folder/files.

Setting example 1: The subject user should be given "Full control" access to the top-level SDK installation folder (the default top-level installation folder is normally when using 64bit environment "C:\Program Files\KONICA MINOLTA\CA-SDK").

In addition, problems may occur if a restricted user attempts to use a development tool to open a sample software project and perform work.

If a problem does occur, after setting the above access permissions, perform the following sequence of works twice.

Sequence of works: Check that the project can be opened normally when using Administrator rights. If it can, close the project and restart the computer.
(Repeat this sequence twice.)

1-1-2 Uninstalling CA-SDK

In the case of uninstalling CA-SDK from the computer

If you wish to uninstall CA-SDK, use the general method for uninstalling a program (the Add/Remove applet in the Control Panel).

1-2 Installing the USB Driver

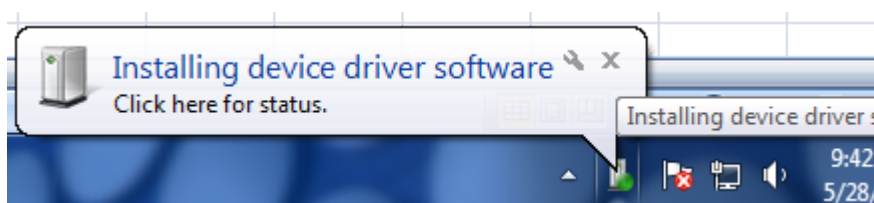
The installation of the device driver is required when connecting CA-310/CA-210 via USB.
Please follow the steps below.

1. Insert the CA-SDK installation CD-ROM in the PC.
2. Connect CA-310/CA-210 to the USB port in the PC

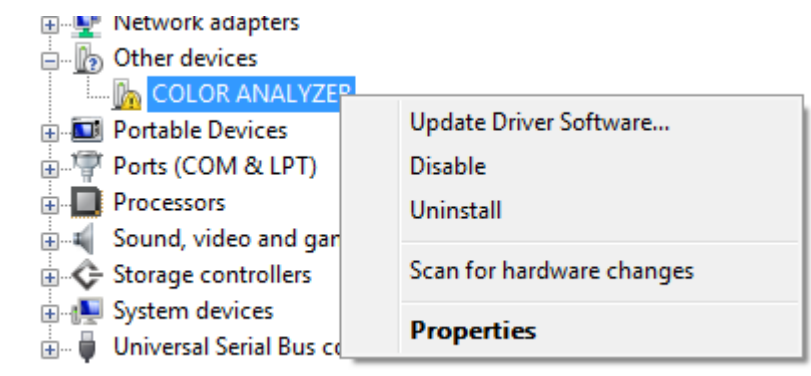
[Notice]

When you use **Windows 7**, if the message such as “Found New Hardware”, do not install now. Click “Cancel” to close the dialog.

When using Windows 7, the installation of the device driver should start automatically, so select the icon “showing Installing device driver software” from the taskbar. Click on ‘Skip obtaining driver software from Windows Update’. And then Click “Yes” in the Confirmation window. Please wait for a while as it may take several minutes to cancel searching the device driver.



3. The screen below will appear after clicking ‘Device Manager’ on the left of the Control Panel. Right click on ‘COLOR ANALYZER’ and select ‘Update Driver Software’.

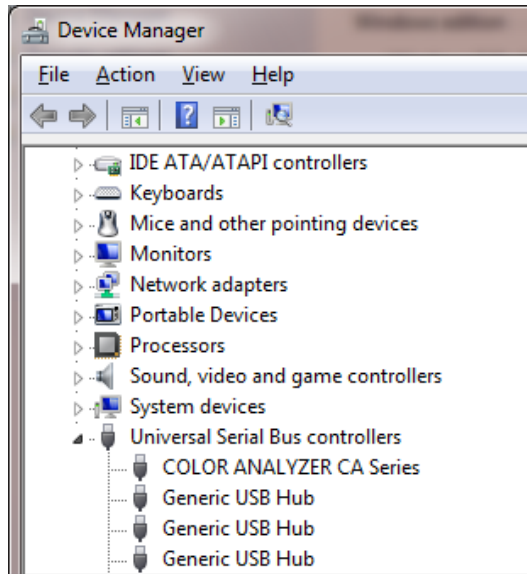


4. Click on ‘Browse my computer for driver software’. Designate the driver folder in the CA-SDK CD-ROM and click “Next”.
(Example) In the case of CD-ROM in E drive.
E:\Driver
5. If some confirmation message about the driver installation is displayed, select “Yes” or “OK”.
6. Click “Close” to complete the installation.

<Confirm the installation of the USB driver>

After installing the USB driver, CA-310/210 can be operated as a USB device.

“COLOR ANALYZER CA Series” will be displayed in the device manager if you use CA-310 or CA-210..



CA-SDK

Instruction Manual

2. **Creating a VisualC# 2013 application
using CA-SDK**

In this section, we will explain how to create a CA-SDK application program using Visual Studio 2013. (In this explanation, we will use Visual Studio 2013. Other versions of Visual Studio may have screens somewhat different than those used here.)

Creation of an application consists of the following processes:

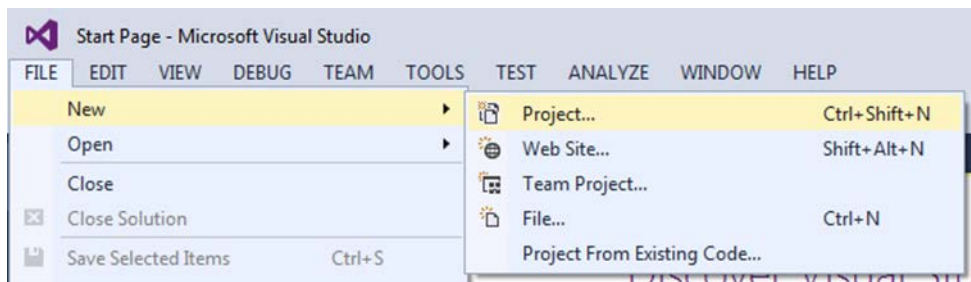
- 1 Installing CA-SDK.
- 2 Creating a new VisualC# 2013 project.
- 3 Setting the references to CA-SDK.
- 4 Creating the application GUI and code.

Before beginning to work in VisualC# 2013, perform step 1: Installing CA-SDK.

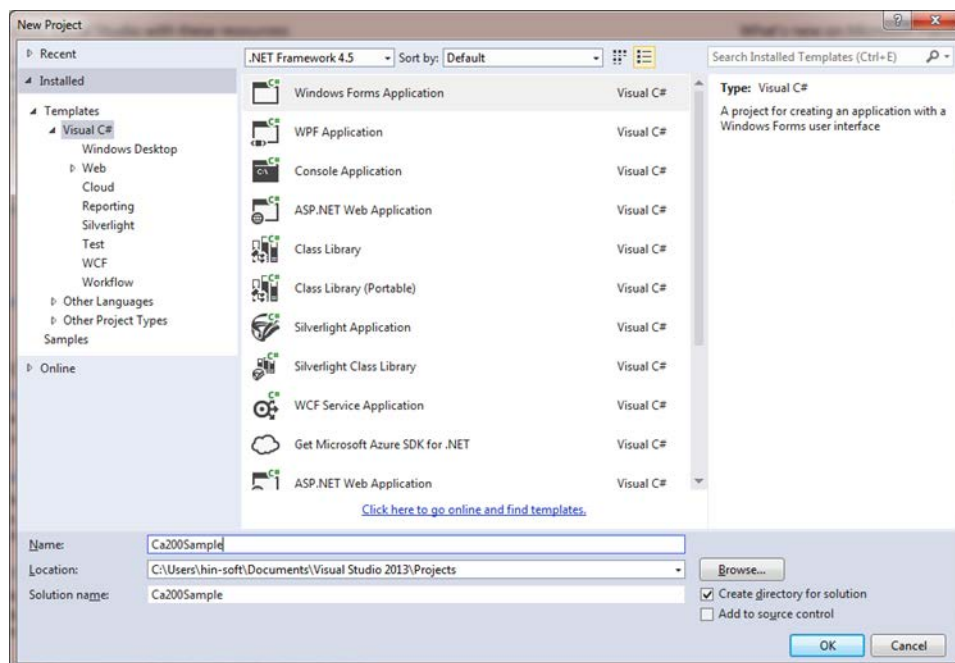
The explanation that follows is for "Step 2: Creating a new VisualC# 2013 project ", and the remaining steps.

2-1 Creating the VisualC# 2013 project

1. Select "FILE"->"New"->"Project..." in the StartPage of Visual Studio 2013.



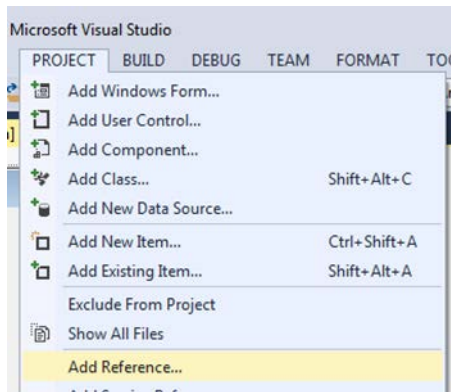
2. Select "Windows Form Application VisualC#", and input "Ca200Sample" as the new project name. Then click "OK" button.



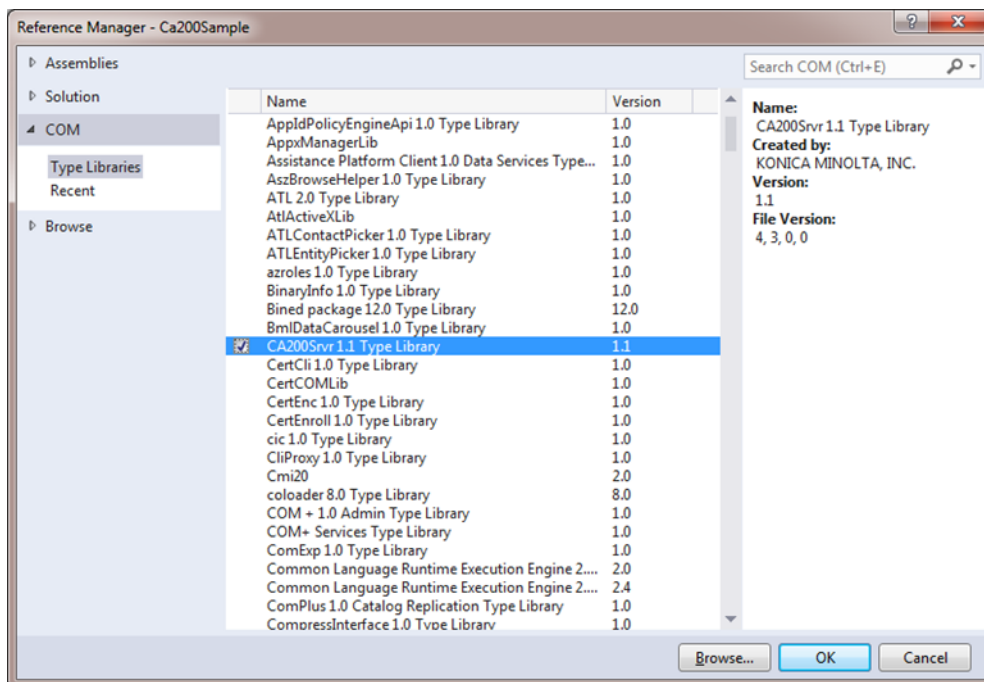
The new project is created and a design view of form will be shown.

2-2 Setting references to CA-SDK

1. Select "PROJECT"->"Add Reference..." from menu.



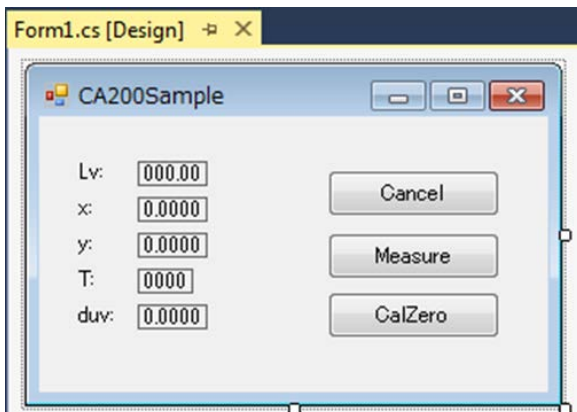
2. Select "COM" tab in the Reference Manager Dialog, and select "CA200Srvr 1.1 Type Library" from the list. Then click "OK" button.



2-3 Creation of Application GUI/Code

The UI (user interface) will be as shown below. There are a total of 10 static text controls (5 as labels and 5 for displaying measurement data) and 3 push buttons.

- When the Zero Cal button is pressed, zero calibration is performed by the CA instrument.
- When the Measure button is pressed, a series of 20 measurements are taken and the measurement results are shown in the main window after each measurement.
- When the Cancel button is pressed, the series of measurements is interrupted and canceled.



The sample software code is shown below.

```
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 { }
        }
    }
}
```

```

    }
    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

Instruction Manual

3. Creating a VisualBasic 2013 application using CA-SDK

In this section, we will explain how to create a CA-SDK application program using Visual Studio 2013. (In this explanation, we will use Visual Studio 2013. Other versions of Visual Studio may have screens somewhat different than those used here.)

Creation of an application consists of the following processes:

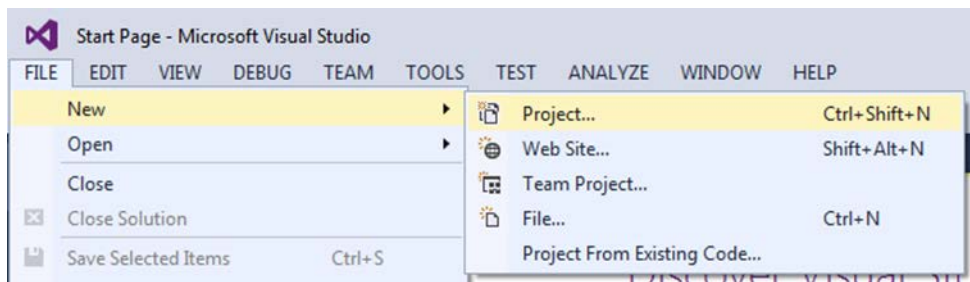
- 1 Installing CA-SDK.
- 2 Creating a new VisualBasic 2013 project.
- 3 Setting the references to CA-SDK.
- 4 Creating the application GUI and code.

Before beginning to work in VisualBasic 2013, perform step 1: Installing CA-SDK.

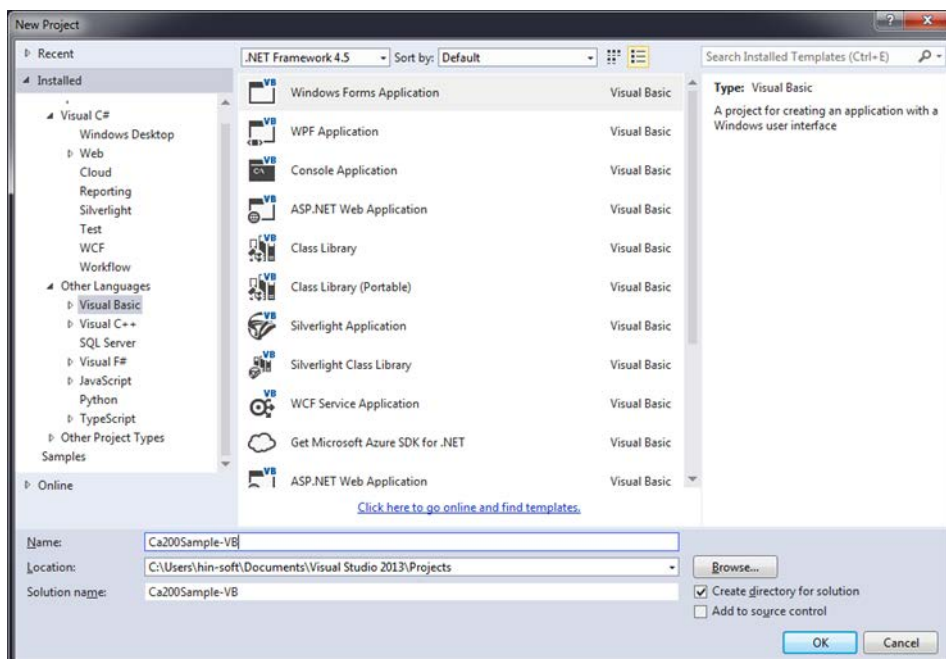
The explanation that follows is for "Step 2: Creating a new VisualBasic 2013 project ", and the remaining steps.

3-1 Creating the VisualBasic 2013 project

1. Select "FILE"->"New"->"Project..." in the StartPage of Visual Studio 2013.



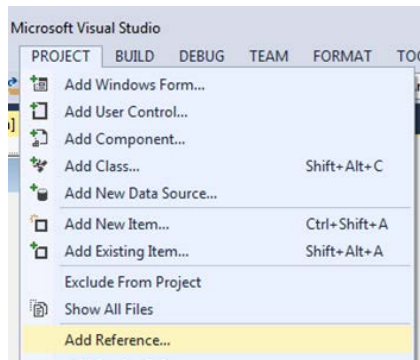
2. Select "Windows Form Application VisualBasic", and input "Ca200Sample-VB" as the new project name. Then click "OK" button.



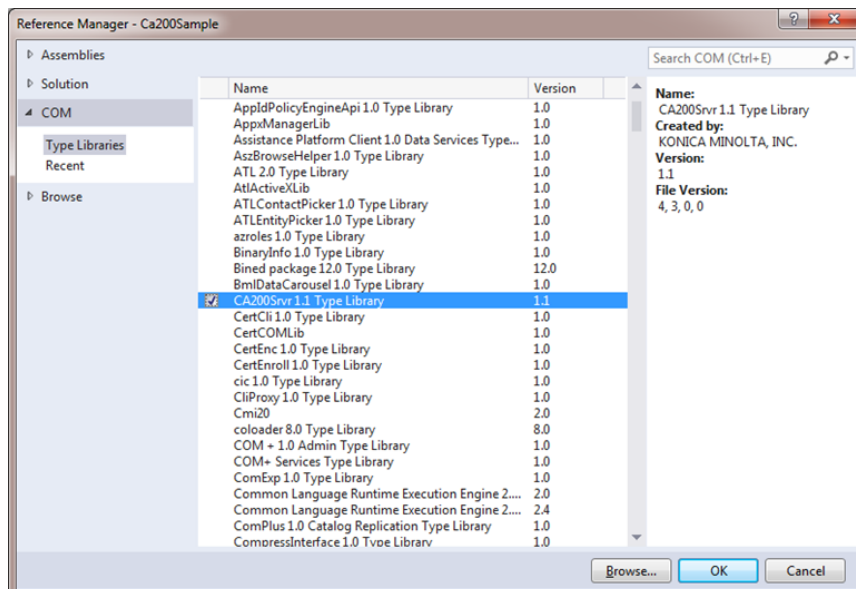
The new project is created and a design view of form will be shown.

3-2 Setting references to CA-SDK

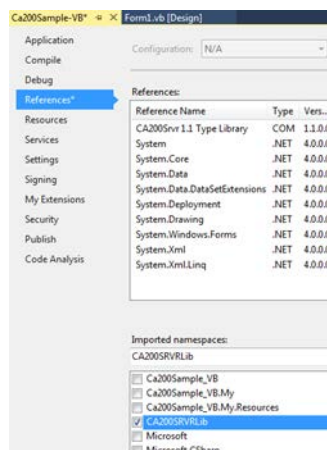
1. Select "PROJECT"->"Add Reference..." from menu.



2. Select "COM" tab in the Reference Manager Dialog, and select "CA200Srvr 1.1 Type Library" from the list. Then click "OK" button.



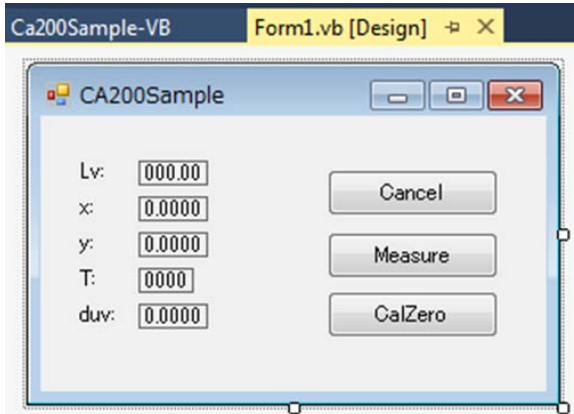
3. Select "Project"->"Ca200Sample-VB Properties..." from menu, and select "References" Tab on the left side. In "Imported namespaces", select "CA200SRVRLib" and set the checkbox "on".



3-3 Creation of Application GUI/Code

The UI (user interface) will be as shown below. There are a total of 10 static text controls (5 as labels and 5 for displaying measurement data) and 3 push buttons.

- When the Zero Cal button is pressed, zero calibration is performed by the CA instrument.
- When the Measure button is pressed, a series of 20 measurements are taken and the measurement results are shown in the main window after each measurement.
- When the Cancel button is pressed, the series of measurements is interrupted and canceled.



The sample software code is shown below.

In VisualBasic 2013, error handling code can be written in the same way as for the previous version of VB. In addition, structured exception handling can also be written.

```
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:
```

```

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
End 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

Instruction Manual

4. How to use CA-SDK Sample Software Controls

Three controls (CaControl, xyControl, VGControl) are used in the CA-SDK VB sample applications.

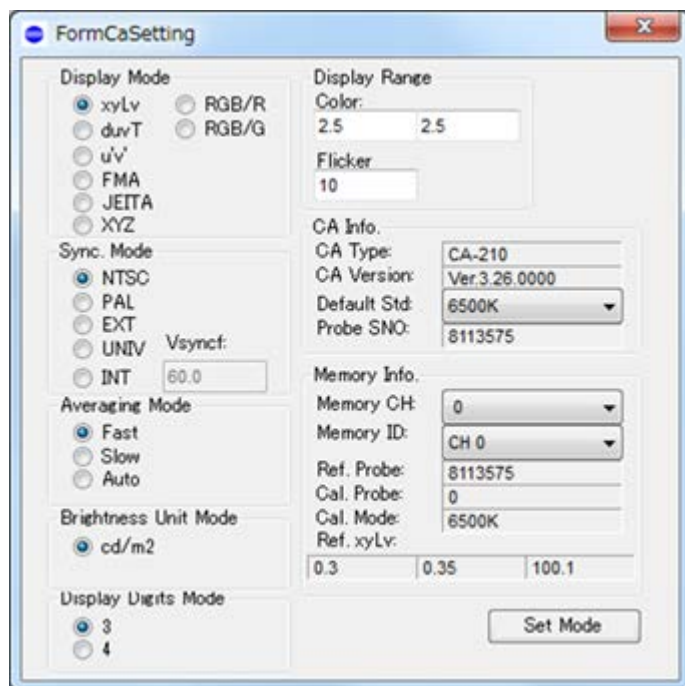
Also, VB sample applications are along with the entire source codes. Please make yourself those sample codes as a reference when developing your application. You can use the VB sample applications and read “ReadMe.htm” that is explained about how to use them by Windows shortcut menu, “startmenu-All Programs-KONICAMINOLTA-CA-SDK”.

Although three controls were originally developed for use with the sample software, any of them can also be used by other software.

In this document, how to use the CaControl (a control for performing various settings on CA-series instruments) and the xyControl (a control for displaying measurement results on a graph of the xy color space) will be explained.

4-1 CaControl (for Settings of CA-Series Instruments)

Clicking the [. . .] button of the CaControl opens the FormCaSetting dialog shown below. Using this dialog, the various setting sof the CA unit can be performed.



4-1-1 CaControl Properties/Methods

Ca property

Sets and retrieves the Ca object to which settings will be applied. Before doing anything else with the control, use this property to set the Ca object which will be controlled.

Syntax:

```
Dim objCaControl as CaControl
```

```
Dim ObjCa as Ca
```

```
ObjCaControl.Ca = objCa
```

Probe property

Sets and retrieves the Probe object to which settings will be applied. Before doing anything else with the control, use this property to set the Probe object which will be controlled.

Syntax:

```
Dim objCaControl as CaControl
Dim objProbe as Probe
objCaControl.Probe = objProbe
```

Memory property

Sets and retrieves the memory object to which settings will be applied. Before doing anything else with the control, use this property to set the Memory object which will be controlled.

Syntax:

```
Dim objCaControl as CaControl
Dim objMemory as Memory
ObjCaControl.Memory = objMemory
```

UpdateCaInfo method

Updates the information (except the memory channel) displayed in the Ca control to reflect the current status of the CA unit.

After setting the required CA-SDK object, this method should be used once before doing anything else with the control.

Syntax:

```
Dim objCaControl as CaControl

ObjCaControl.UpdateCaInfo()
```

UpdateMemoryInfo method

Updates the memory channel displayed in the Ca control to reflect the currently set memory channel on the CA unit.

After setting the required CA-SDK object, this method should be used once before doing anything else with the control.

Syntax:

```
Dim objCaControl as CaControl

ObjCaControl.UpdateMemoryInfo()
```

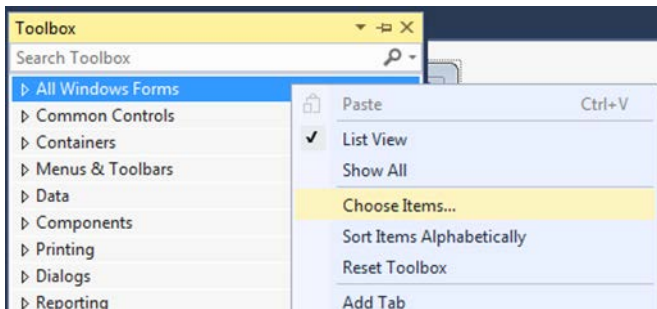
Update event

This event is fired when the control has changed a setting for the CA unit.

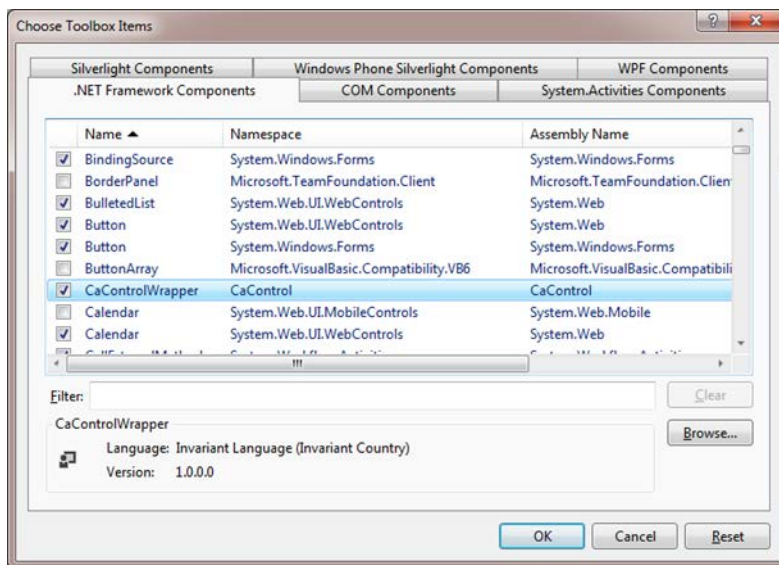
4-1-2 CaControl: Using in VisualC# 2013

Add the control to the VisualC# project in the following procedure.

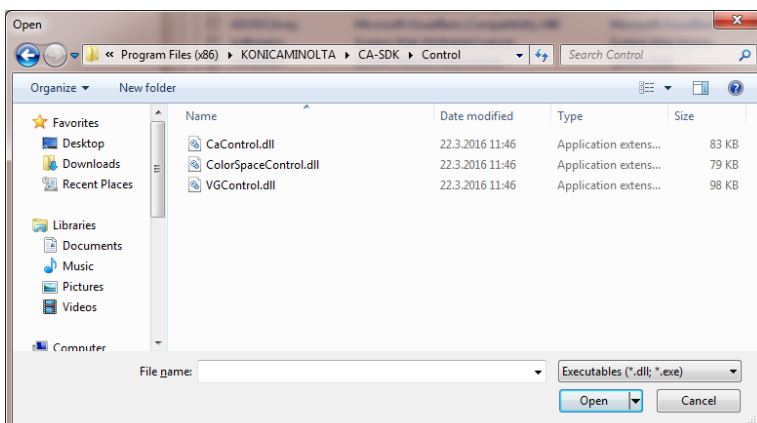
1. Select “View”-“Toolbox” from the menu bar, then Toolbox will be shown.
2. Right-click in the Toolbox, and select “Choose Items...”.



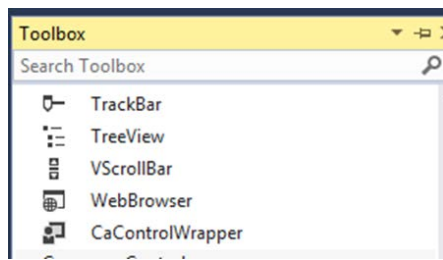
3. Select “CaControlWrapper” and check the checkbox “on”. Then click “OK” button.



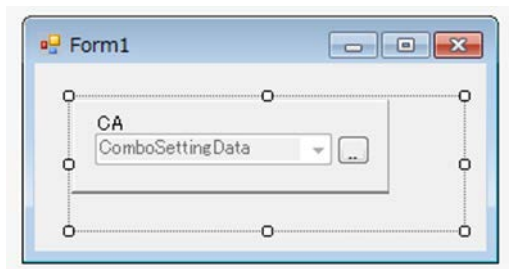
If “CaControlWrapper” is not shown although CA-SDK has been installed, click “Browse...” button, and select CAControl.dll from the folder that CA-SDK is installed.



When addition of the control is completed, the new control icon (CaControlWrapper) will be shown in the Toolbox, and then can be used.



4. Add "CaControlWrapper" to the form.



The sample source code is shown below.

After creating the SDK object, the objects for controlling the CA unit are set on the control using the control's properties. Then, the current status of the CA unit is obtained and reflected in the control using UpdateCaInfo and UpdateMemoryInfo to initialize the control.

After performing the above procedure, the control is ready to be used.

```
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;

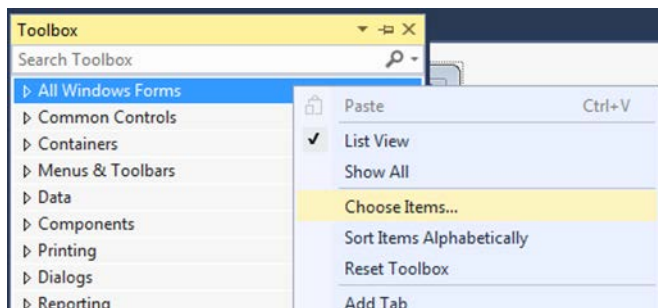
    objCaControl = null;
}
}
}

```

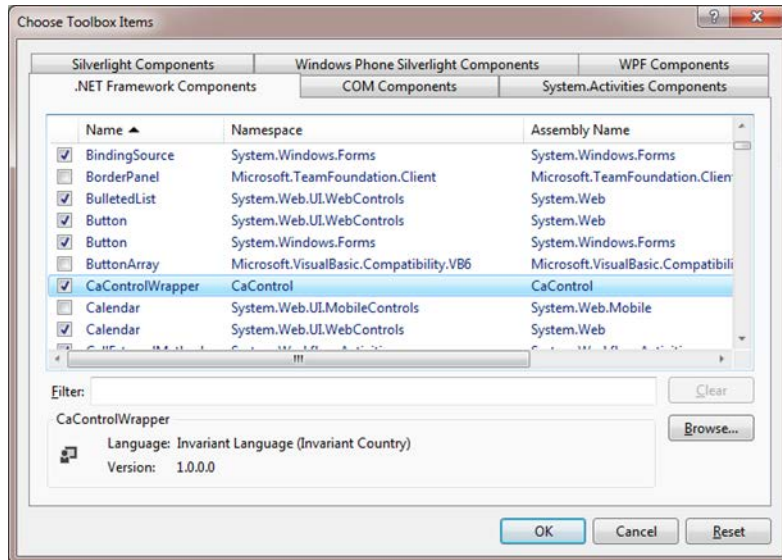
4-1-3 CaControl: Using in VisualBasic 2013

Add the control to the VisualC# project in the following procedure.

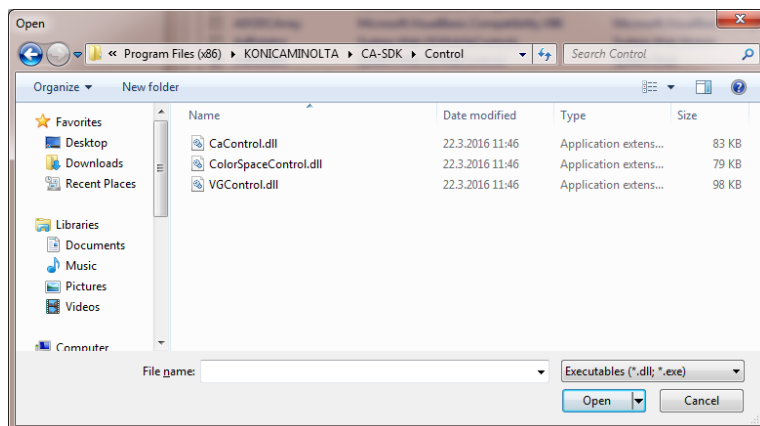
1. Select "View"->"Toolbox" from the menu bar, then Toolbox will be shown.
2. Right-click in the Toolbox, and select "Choose Items..."



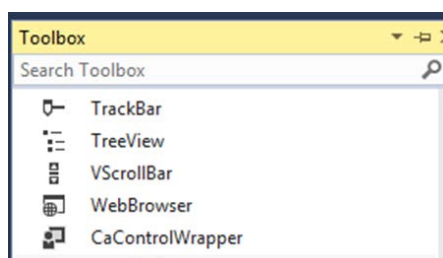
3. Select "CaControlWrapper" and check the checkbox "on". Then click "OK" button.



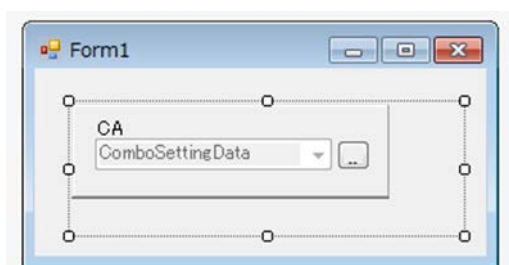
If "CaControlWrapper" is not shown although CA-SDK has been installed, click "Browse..." button, and select CAControl.dll from the folder that CA-SDK is installed.



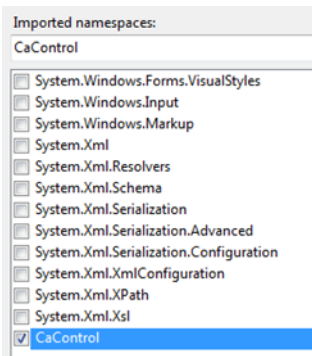
When addition of the control is completed, the new control icon (CaControlWrapper) will be shown in the Toolbox, and then can be used.



4. Add "CaControlWrapper" to the form.



5. Select "Project"-"(Project name) Properties..."from the menu bar, and select "References"Tab on the left side. In "Imported namespaces", select "CaControl" and check the checkbox "on".



The sample source code is shown below.

After creating the SDK object, the objects for controlling the CA unit are set on the control using the control's properties. Then, the current status of the CA unit is obtained and reflected in the control using UpdateCaInfo and UpdateMemoryInfo to initialize the control.

After performing the above procedure, the control is ready to be used.

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

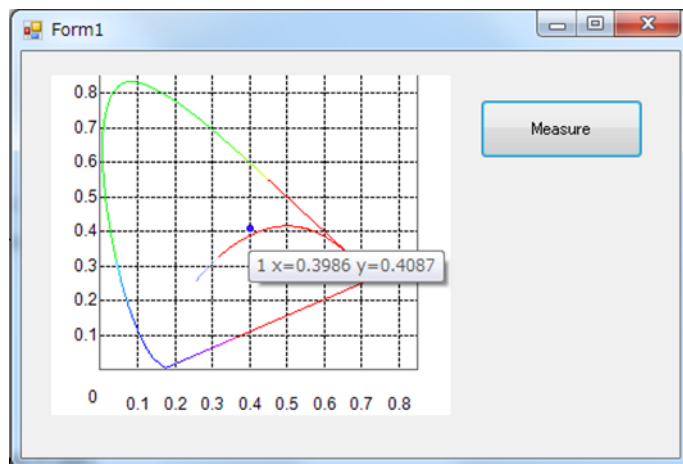
```

4-2 xyControl (for Displaying Data in xy Color Space)

The xyControl plots the color measurement data of the CA unit on an xy chromaticity diagram. Up to approx. 1000 data can be registered on the xyControl.

In addition to a function for automatically registering and displaying data, a function for specifying data with an index (multiple displays are possible) is also provided.

A maximum 400% display magnification function is also provided; right-clicking using a mouse brings up a magnification menu. Bringing the mouse close to a data point causes the data ID number (equal to data index + 1) and the chromaticity values to be displayed.



4-2-1 xyControl Properties/Methods

Ca property

Sets the Ca object for which measurement results will be displayed.

Before using the control for anything else, this property should be set.

Syntax:

```
Dim objxyControl as xyControl
```

```
Dim objCa as Ca
```

```
ObjxyControl.Ca = objCa
```

Probe property

Sets the Probe object for which measurement results will be displayed.

Before using the control for anything else, this property should be set.

Syntax:

```
Dim objxyControl as xyControl
```

```
Dim objProbe as Probe
```

```
ObjxyControl.Probe = objProbe
```

SetXYGraphData method

Obtains the current xy data from the Probe object, registers the data, and displays the data on the xy chromaticity diagram.

Increments the data index.

Sets previous data to invisible.

Syntax:

```
Dim objxyControl as xyControl
ObjxyControl.SetXYGraphData()
```

AddXYGraphData method

Obtains the current xy data from the Probe object, registers the data at the specified index, and displays the data on the xy chromaticity diagram.

The index is set to the specified index.

The display statuses of data at other index values are left unchanged.

Syntax:

```
Dim objxyControl as xyControl
Dim lIndex as long
ObjxyControl.AddXYGraphData(lIndex)
```

SetXYData method

Registers the specified x, y chromaticity data at the specified index, and displays the data on the xy chromaticity diagram.

The index is set to the specified index.

The display statuses of data at other index values are left unchanged.

Syntax:

```
Dim objxyControl as xyControl
Dim lIndex as long
Dim fx as float
Dim fy as float
ObjxyControl.SetXYData(lIndex, fx, fy)
```

SetVisible method

Sets the display status of the data at the specified index to visible.

Syntax:

```
Dim objxyControl as xyControl
Dim lIndex as long
```

```
ObjxyControl.SetVisible(!Index)
```

SetVisibleAll method

Sets the display statuses of all data to visible or invisible according to the specified boolean value (True or False).

Syntax:

```
Dim objxyControl as xyControl
Dim bVisible as Boolean
ObjxyControl.SetVisibleAll(bVisible)
```

ClearData method

Clears all data registered on the control and initializes the index.

Syntax:

```
Dim objxyControl as xyControl
ObjxyControl.ClearData()
```

4-2-2 xyControl: Using in VisualC# 2013

Add the CA-SDK component xyControl to the VB project as described in section 4-1-2: CaControl: Using in VisualC# 2013.

The sample source code is shown below.

After creating the SDK object, the objects for controlling the CA unit are set on the control using the control's properties. Then, the control is initialized using ClearData.

When a measurement is taken the measurement results are plotted on the xy chromaticity diagram. This is performed using SetXYGraphData. SetXYGraphData sets only the latest data to visible; previous measurement data are not displayed.

```
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 xyControl: Using in VisualBasic 2013

Add the CA-SDK component xyControl to the VB project as described in section 4-1-3: CaControl: Using in VisualBasic 2013.

The sample source code is shown below.

After creating the SDK object, the objects for controlling the CA unit are set on the control using the control's properties. Then, the control is initialized using ClearData.

When a measurement is taken the measurement results are plotted on the xy chromaticity diagram. This is performed using SetXYGraphData. SetXYGraphData sets only the latest data to visible; previous measurement data are not displayed.

```

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

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
Instruction Manual
5. Application Examples

5-1 White Balance Adjustment System

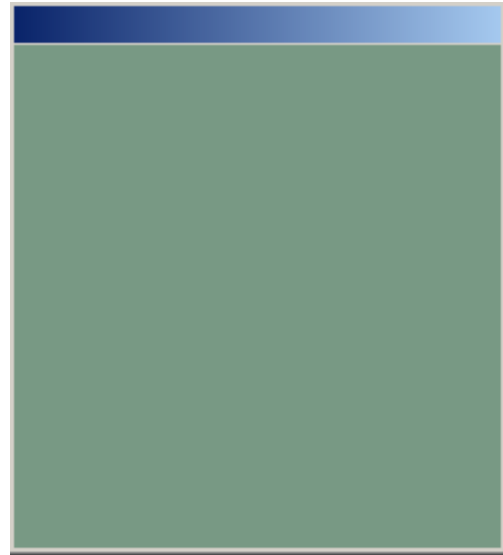
5-1-1 White Balance Adjustment System

When white is displayed on several units of LCD monitors, the colors shown on the displays may appear to be different. This is due to variations in the components (filters, circuits, etc.) which causes the white to not appear to be exactly the same color.

For example, if there was a 0.02 difference in the x, y chromaticity due to component variations, the colors would be as shown below.



$x=0.310, y=0.332, L_v=172.5$
Target chromaticity and luminance



$x=0.292, y=0.356, L_v=170.3$
Unadjusted chromaticity and luminance

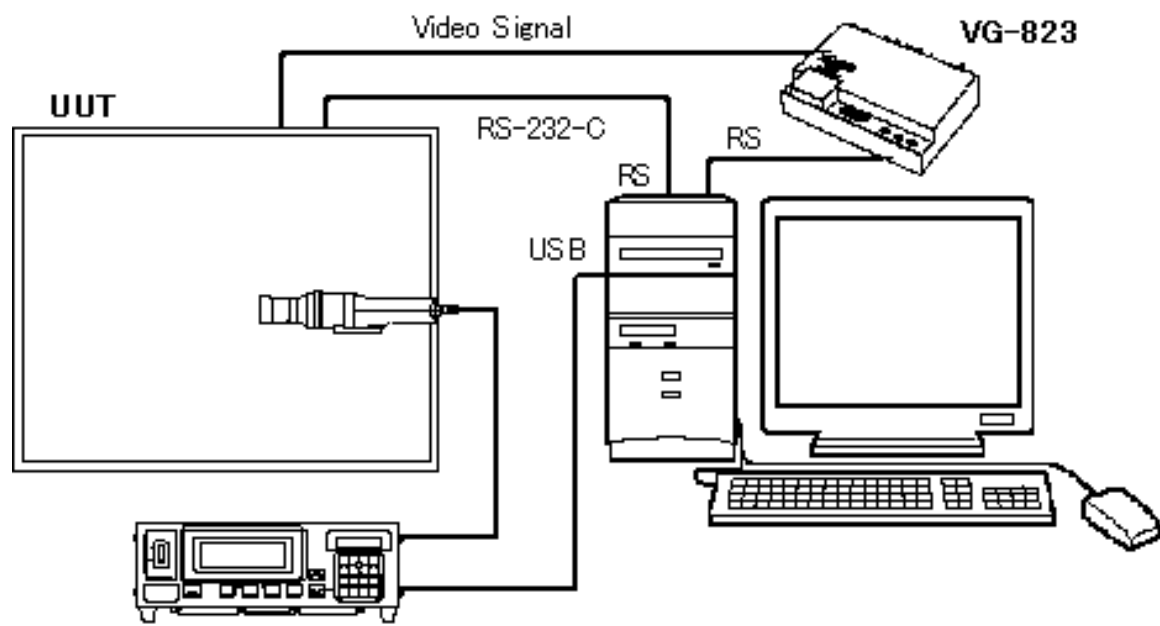
On the other hand, recently the use of LCD monitors for viewing images has increased, and demands for more accurate chromaticity and luminance reproduction is also increasing. In order to reduce variations in the displayed color, cases where white balance adjustment of LCD monitors is performed have been increasing.



$x=0.315, y=0.337, L_v=169.4$
Chromaticity and luminance after adjustment

An example of a system using the CA-210 for high-speed, high-accuracy adjustment of the luminance and chromaticity of white and inspection of contrast will be explained in the following sections.

5-1-2 System Block Diagram



CA-210 + 1 probe	For measuring luminance and chromaticity
UUT (Unit Under Test)	The LCD monitor to be adjusted. The type to which calibration data can be written from an external device. In this example, communication is performed using RS-232C.
PC	Computer to control the measuring instrument, LCD monitor, and pattern generator USB port x 1 (for CA-210) RS-232C port x 2 (1 for controlling LCD and 1 for controlling VG) Pentium III 660MHz or faster Resolution: XGA
VG-823, 813, 812	Used for controlling the pattern displayed on the UUT. Pattern generator manufactured by Astrodesign Inc.

Note: If I2C is used, an RS-232C to I2C converter is necessary.

5-1-3 White Balance Adjustment Software Functions

Varies the gain of the LCD monitor to adjust it so that the color displayed on the monitor matches the specified chromaticity and luminance. Up to 6 types of settings can be performed in sequence.

- Inspects the contrast.
- Has pass/fail judgment function.

5-1-4 White Balance Adjustment Time

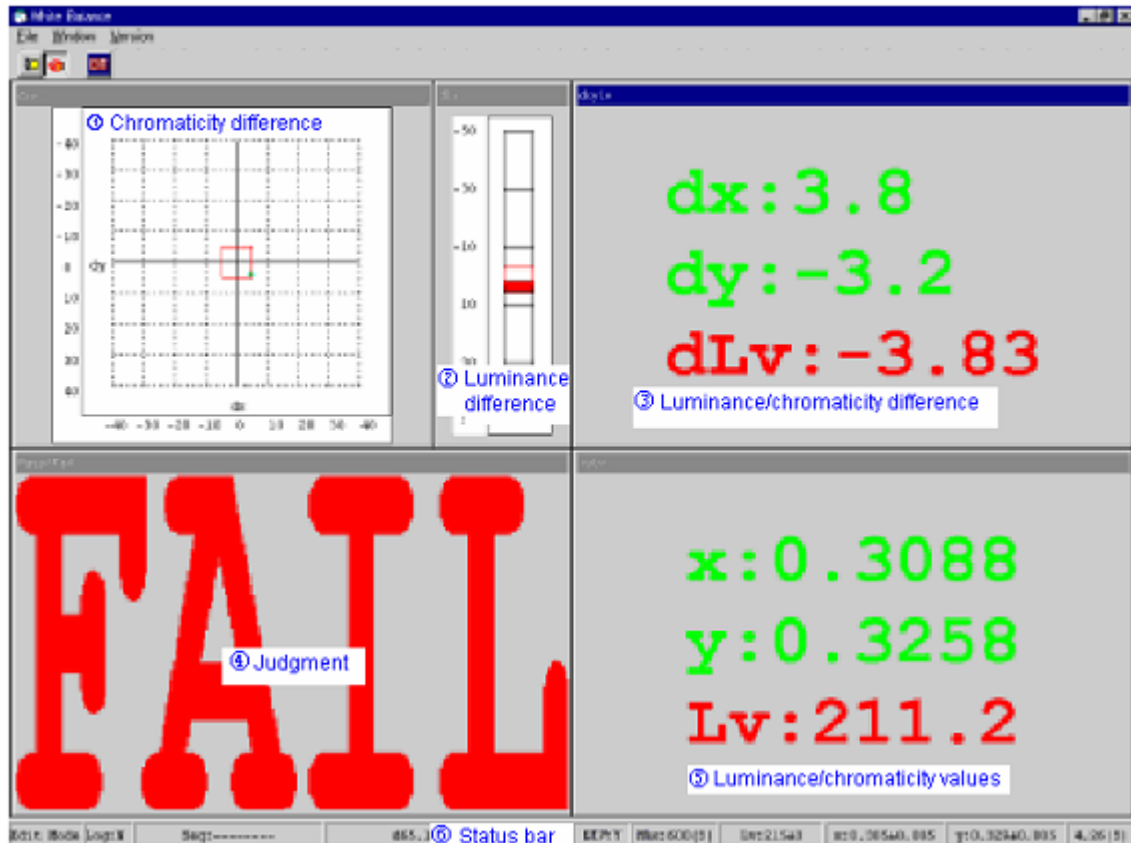
The white balance adjustment time varies depending on the required adjustment accuracy.		
For luminance accuracy of:	And chromaticity accuracy of:	Adjustment time would be
$\pm 10\text{cd/m}^2$	$\pm 0.01\text{cd/m}^2$	Within 5 sec.
$\pm 5\text{cd/m}^2$	$\pm 0.005\text{cd/m}^2$	Within 10 sec.

- Operating environment: Fujitsu FMV-6600MF8/X PIII 660MHz; memory: 128MB

Notice: Adjusting time depends on LCD monitors.

5-1-5 GUI

The adjustment screen shows the adjustment status clearly so it can be easily viewed at a glance.



In the above screen, the target values are set at:

Lv: 215 ± 3
 x: 0.305 ± 0.005
 y: 0.329 ± 0.005

- ①Chromaticity difference: The specified tolerance range from the target is indicated by the red frame. Automatic adjustment is performed to bring the displayed color within that frame.
- ②Luminance difference: The specified tolerance range from the target is indicated by the red frame. Automatic adjustment is performed to bring the displayed luminance within that frame.
- ③Luminance/chromaticity difference: The numerical differences from the target values are shown. If the measured values are within the specified tolerance range, the difference values will be shown in green; if they are outside the tolerance range, the values are shown in red.
- ④Judgment: If the luminance and chromaticity values are within the specified tolerance range, "Pass" will be shown in green letters. If the luminance or either chromaticity value is outside the specified tolerance range, "Fail" will be shown in red letters.
- ⑤Luminance/chromaticity measurement values: The numerical measurement values are shown. If the measurement values are within the specified tolerance range, the values will be shown in green; if they are

outside the tolerance range, the values are shown in red.

⑥ Status bar: Shows various settings including the target luminance and chromaticity values and the adjustment time.

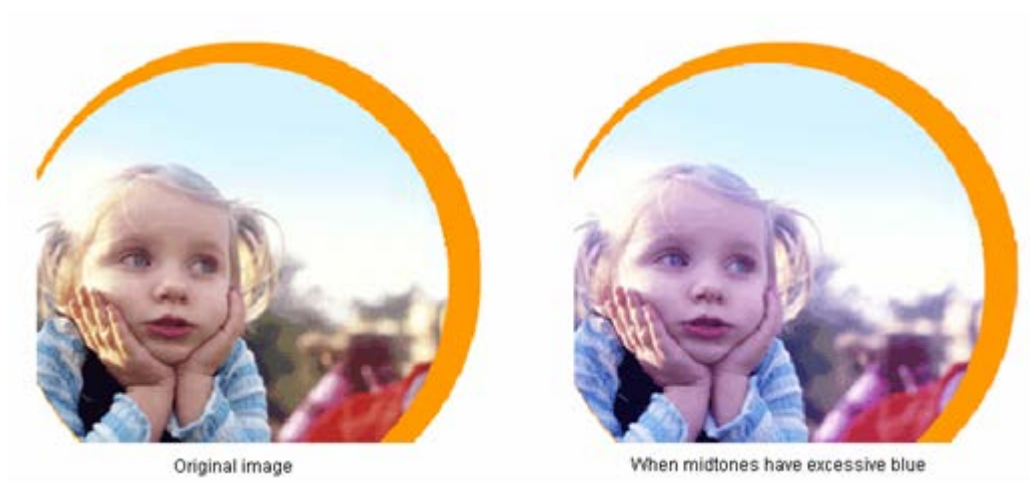
When the CA-310/210 is used as described above, high-speed, high-accuracy white balance adjustment can be performed.

5-2 Gamma Adjustment System

5-2-1 Gamma Adjustment

When multiple LCD monitors display the same image, the midtones may appear to be different. This is due to variations in the components (filters, circuits, etc.) which causes the midtones to not appear to be exactly the same colors.

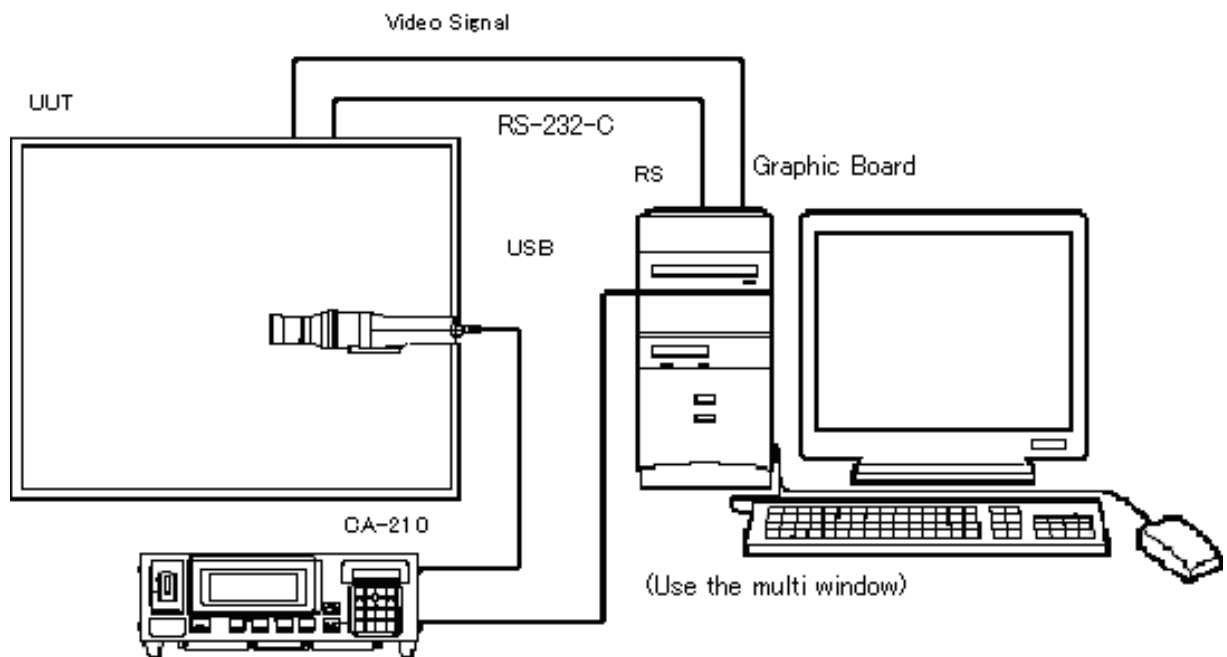
For example, if the brightness of the midtones varied due to component variations, the images on the LCD monitors might be as shown below.



On the other hand, recently the use of LCD monitors for viewing images has increased, and demands for more accurate midtone reproduction is also increasing. In order to reduce variations in the displayed color, cases where adjustment of LCD monitors is performed have been increasing.

An example of a system using the CA-210 for high-speed, high-accuracy adjustment of midtone balance for each primary color will be explained in the following sections.

5-2-2 System Block Diagram



CA-210 + 1 probe	For measuring luminance and chromaticity
UUT (Unit Under Test)	The LCD monitor to be adjusted. The type to which calibration data can be written from an external device. In this example, communication is performed using RS-232C.
PC	Computer to control the measuring instrument, and LCD monitor USB port x 1 (for CA-210) RS-232C port x 1 (for controlling LCD) Pentium III 660MHz or faster Resolution: XGA, True Color (24-bit or higher)
Graphic board	Specified graphic board capable of displaying True Color (24-bit or higher)

Note: If I2C is used, an RS-232C to I2C converter is necessary.

5-2-3 Gamma Adjustment Process

The patterns displayed on the unit under test are output by the PC's graphics board.

Set the PC to display multiple windows, and let the PC display the software and also have the graphic card display the test pattern.

The method for measuring characteristics is as specified in IEC61966-4, where gamma is calculated from tristimulus values.

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.

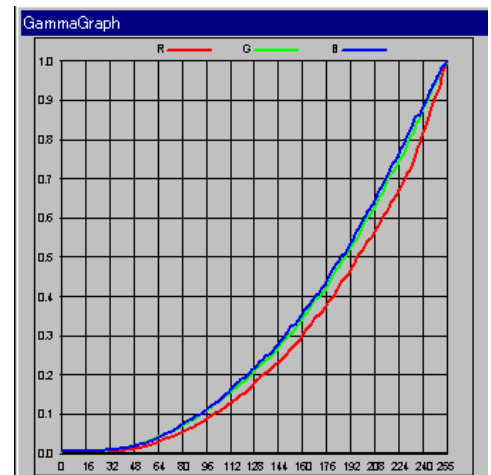
Horizontal axis: Steps 0 to 255

Vertical axis: Measured values for each color normalized using the maximum values for each color.

① γ measurement

First, the current characteristics of the LCD are measured at several steps. (number of steps: 8, 16, 32, 64, 128, and 256)

An example of the results is shown at right.



② Setting of γ value

The γ characteristics are set.

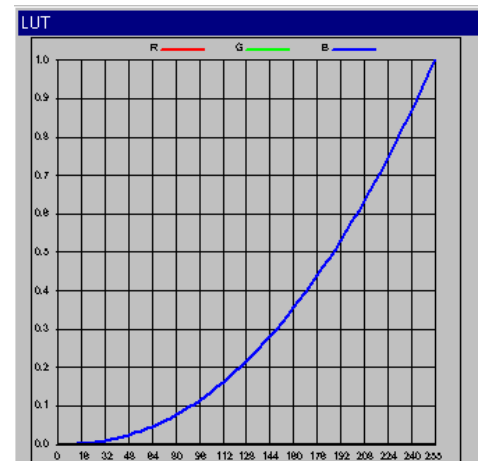
In the example at right, is set to 2.2.

The setting data are calculated for each step using the equation y curve = (step) \times 2.2

An LUT (Look Up Table) of the setting data used in the example at right is calculated from the data measured in ①, and this LUT is written to the LCD monitor.

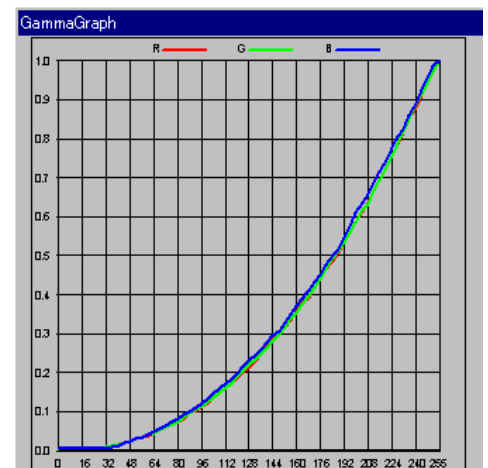
The calculation method for the LUT is as follows:

For example, for step 16, where the data calculated as $(16^{2.2}) / (255^{2.2})$ (= Value/Max. value) is located in the measurement data would be determined, and the LUT value would then be determined by linear interpolation using the point above and the point below this point.



③ γ inspection

The LCD to which the LUT has been written is then measured again to verify that the LUT data has been accurately reflected in the display. An example is shown at right.



5-2-4 Gamma Adjustment Time

The time for adjusting the γ characteristics of the 3 colors R, G, and B is shown in Table 1 below.

Note: A 100ms wait is provided to allow the LCD response to stabilize after the displayed pattern has been changed. The times listed do not include the time to write the LUT to the LCD monitor.

Operating environment: Fujitsu FMV-6600MF8/X PIII 660MHz; Memory: 128MB

γ adjustment time(sec.)

Number of steps	8	16	32	64	128	256
Adjusting Time	6	8	12	19	34	64

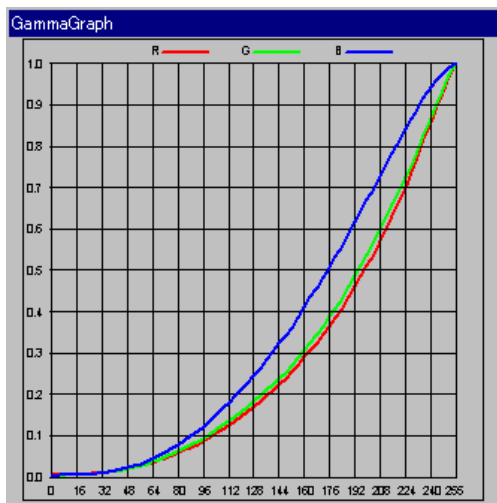
5-2-5 GUI

Adjustment display screen examples

The focus is on displaying the condition at the time of measurement.

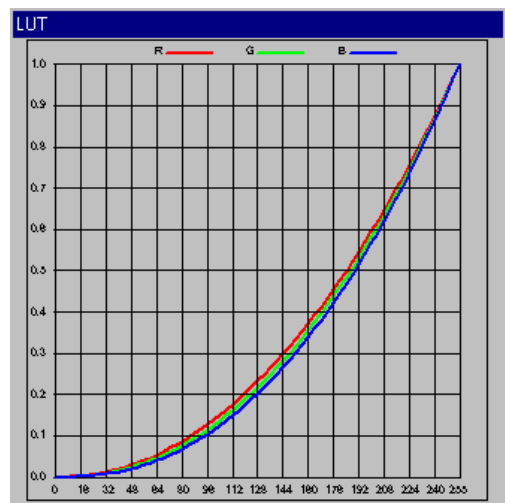
① γ measurement screen

The LCD monitor's characteristics are measured at the specified measurement steps and the results are displayed.



② LUT setting display screen

The setting data which will be written to the LCD monitor are displayed. In this example, the values are set to 2.1 for R, 2.2 for G, and 2.3 for B.



③ γ verification screen

The LCD monitor to which the LUT was written is measured again, and the resulting γ characteristics are displayed.

