Sentech GigE Vision Camera

StGigE SDK

Sample Guide

Sensor Technology Co., Ltd.

Table of Contents

1. Descriptions of the sample programs

1.1 eBUS Driver Installation API

This sample shows how to display the information about the network configuration (NICs) and apply eBUS drivers (Universal Pro or Optimal driver) to the NICs on the PC.

1) Introduction

This sample shows how to:

Displays the information about the NICs and the drivers installed on the PC.
Installs and uninstall the drivers for the NIC.
To understand the timing of reboot the PC when install and uninstall the drivers.

2) Prerequisite

Please refer this for include the eBUS driver with the StGigE SDK to the software.
Please read the "Custom install Guide" before use the StGigE SDK.

3) Descriptions of the files

3.1 main.cpp

This source code file shows how to use EbNetworkAdapter, EbDriver and EbInstaller classes.

1.2 GEVPlayerSample

This sample is the source code for the StGigE Player application.
StGigE Player application is used to connect and configure the GigE Vision camera.

1) Introduction

This sample help to understand how to use StGigE SDK.

2) Prerequisite

Please check the "StGigE Player" application before use this sample code.
It is necessary to understand and have the knowledge about C++ and Microsoft MFC foundation class.

3) Descriptions of the files

3.1) AboutBox.cpp

The source code file for the about dialog in the StGigE Player.
This is accessible from "About StGigE Player" under "Help" in the menu.

3.2) BitmapButton.cpp

The source code file for the "Play" and the "Stop" button class of the user inter face in the StGigE Player.

3.3) BufferOptionsDlg.cppT

The source code file for the Buffer Options dialog in the StGigE Player.
This is accessible from " Buffer Options..." under "Tools" in the menu.

3.4) ConnectionThread.cpp

The source code file for the thread, which is created classes of the asynchronous connection procedure and pass the message to the ProgressDlg. (Please check ProgressDlg.cpp)

3.5) EventMonitorDlg.cpp
The source code file for the Event Monitor dialog in the StGigE Player.
It is accessible from "Event Monitor" under "Tools" in the menu.

3.6) FilteringDlg.cpp
The source code file for the Image Filtering dialog in the StGigE Player.
It is accessible from "Image Filtering" under "Tools" in the menu.

3.7) GEVPlayer.cpp
The source code file for the CWinApp class that brings up the StGigE Player dialog on InitInstance.

3.8) GEVPlayerDlg.cpp
The source code file for the StGigE Player dialog class.
The most of the source code of the StGigE Player including the user interface elements when use the StGigE Player.

3.9) HTMLDialog.cpp
The source code file for the parent class to display the HTML window.

3.10) ImageSaveDlg.cpp
The source code file for the Image Save dialog in the StGigE Player.
It is accessible from " Save Images..." under "Tools" in the menu.

3.11) LoadingThread.cpp
The source code file for the thread, which is created the class of the asynchronous load from the pvcfg file and the connection procedure, and pass the message to the ProgressDlg. (Please check ProgressDlg.cpp)

3.12) LogBuffer.cpp
The source code file for use the log events for Event Monitor dialog. (Please check EventMonitorDlg.cpp)

3.13) ParameterInfo.cpp
The source code file for use by the Event Monitor dialog to make the GenICam events log.

3.14) ProgressDlg.cpp
The source code file for the progress dialog, which is appeared when connects the camera, or while loading or saving the configuration file.

3.15) RegisterInterfaceDlg.cpp
The source code file for the register interface dialog, which is appeared when selects "Ctrl + Alt + R".

3.16) SavingThread.cpp
The source code file for the thread, which is created the class of the asynchronous save the pvcfg file while passing the message to the ProgressDlg. (Please check ProgressDlg.cpp)

3.17) SetupDlg.cpp
The source code file for the Setup dialog.
It is accessible from " Setup..." under "Tools" in the menu.

3.18) SplashPage.cpp
The source code file for the dialog, which is appeared for a moment while StGigE Player is starting.
This dialog appears until the main dialog appears.

3.19) Thread.cpp
The source code file for the class, which is used by ThreadDisplay.

3.20) ThreadDisplay.cpp
The source code file for the thread, which is created the class of obtains and displays the image.

3.21) stdafx.cpp
The source code file, which is generated by the standard MFC application.

1.3 PvBufferWriterSample

This sample shows how to use the PvBuffer and the PvBufferWriter classes.

1) Introduction
This sample shows how to:
Creates and allocates the buffer, and obtains the image with the PvBuffer class.
Saves the buffer data to the bitmap format file with the PvBufferWriter class.

2) Prerequisite
When use this sample:
The camera has to connect to the NIC, which is installed either the eBUS Universal Pro or the Optimal driver, before use this sample.

3) Descriptions of the files
3.1 PvBufferWriterSample.cpp
The source code file for the sample program of how to use the PvBuffer and the PvBufferWriter classes.
Please check the comments in this source code for the information, which is how to use these classes.

1.4 PvCamHeadSerialComLogSample

This sample shows how to create the simple UI application for the camera control and the communication monitoring.

This sample allows to monitoring the serial communication between the camera register and the IP Engine on the camera.

1) Introduction

This sample is based on the PvSimpleUISample application.

This sample is developed as a simple MFC single dialog application.

The most of functions excluding the image streaming and display thread, are in the PvCamHeadSerialComLogSampleDlg.

2) Prerequisite

When use this sample:

The camera has to connect to the NIC, which is installed either the eBUS Universal Pro or the Optimal driver, before use this sample.

3) Descriptions of the files

3.1 PvCamHeadSerialComLogSampleDlg.cpp

The source code file for the main application dialog.

The most of the events in this file.

3.2 PvCamHeadSerialComLogSample.cpp

The source code file for the main application.

This application displays the main application dialog.

3.3 Thread.cpp

The source code file for the thread, which is created the class of display the image from the PvPipeline on the PvDisplay.

1.5 PvConfigurationReaderSample

This sample shows how to use the PvConfigurationReader and PvConfigurationWriter classes.

1) Introduction

This sample shows how to:

Creates the list of the camera and the stream configuration with the PvConfigurationReader class.
Creates the string information with the PvConfigurationReader class.
Saves three different camera information (The camera, the stream and the string) to the pvxml file with the PvConfigurationWriter class.
Saves the camera configuration with the specific name with the PvConfigurationWriter class.
Saves the stream configuration with the specific name with the PvConfigurationWriter class.
Saves the string information with the specific name with the PvConfigurationWriter class.

2) Prerequisite

When use this sample:

The camera has to connect to the NIC, which is installed either the eBUS Universal Pro or the Optimal driver, before use this sample.

3) Descriptions of the files

3.1 PvConfigurationReaderSample.cpp

The source code file for the sample program of how to use the PvConfigurationReader and the PvConfigurationWriter classes.
Please check the comments in this source code for the information, which is how to use these classes.

1.6 PvCustomPipelineSample

This sample shows how to use the PvPipeline class.

1) Introduction

This sample help to understand how to use the PvPipeline class.
The PvPipeline has standard image acquire operation.
Some application might require the lower latency or different operation from the defalt PvPipeline operation.
In this case, please starts with this sample to make application.

2) Prerequisite

When use this sample:

The camera has to connect to the NIC, which is installed either the eBUS Universal Pro or the Optimal driver, before use this sample.

3) Descriptions of the files

3.1 CustomPipeline.cpp, CustomPipeline.h

The source code file for the pipeline.

3.2 PvCustomPipeline.cpp

The source code file for the sample program of how to use the PvPipeline class.
Please check the comments in this source code for the information, which is how to use this class.

1.7 PvDeviceFindingSample

This sample shows how to use the PvSystem, PvInterface, PvDeviceInfo and PvDeviceFinderWnd classes.

1) Introduction

This sample shows how to:

Finds out the cameras on the network with the PvSystem and the PvInterface classes.
Finds out the cameras on the network with the GUI, with the PvDeviceFinderWnd class.

2) Prerequisite

When use this sample:

The camera has to connect to the NIC, which is installed either the eBUS Universal Pro or the Optimal driver, before use this sample.

3) Descriptions of the files

3.1 PvDeviceFindingSample.cpp

The source code file for the sample program of how to use the PvSystem, the PvInterface, the PvDeviceInfo and the PvDeviceFinderWnd classes.
Please check the comments in this source code for the information, which is how to use these classes.

1.8 PvGenParameterArraySample

This sample shows how to use the PvGenParameterArray class, to control the settings of the camera.

1) Introduction

This sample shows how to:

Obtains the PC communication related settings with the PvGenParameterArray class.
Obtains the settings of the camera with the PvGenParameterArray class.
Obtains the image stream control settings with the PvGenParameterArray class.
Obtains and sets the PvGenParameter value with the PvGenParameter class.
Obtains the PvGenParameter name with the PvGenParameter class.
Obtains all parameters of the PvGenParameterArray with the PvGenParameter class.
Check the parameters type with the PvGenParameter class.
Casts the PvGenParameter in the sublcass type with the PvGenParameter class.

2) Prerequisite

When use this sample:

The camera has to connect to the NIC, which is installed either the eBUS Universal Pro or the Optimal driver, before use this sample.

3) Descriptions of the files

3.1 PvGenParameterArraySample.cpp

The source code file for the sample program of how to use the PvGenParameterArray and the PvGenParameter classes.
Please check the comments in this source code for the information, which is how to use these classes.

1.9 PvMulticastMasterSample

This sample shows how to use the PvDevice class, to control the malticast master.
Once the camera is connected to the PC as the master, use the other PCs as slave PC with Multicasting Slave Sample (PvMulticastSlaveSample).
IGMP (Internet Group Management Protocol) supported Ethernet switch is required for the multicast system.

1) Introduction

This sample shows how to:

- Connects the camera with the PvDevice class.
- Configures the camera for the multicasting with the PvDevice class.
- Starts streaming with the PvDevice class.
- Stop streaming with the PvDevice class.

2) Prerequisite

When use this sample:

- The camera has to connect to the NIC, which is installed either the eBUS Universal Pro or the Optimal driver, before use this sample.

3) Descriptions of the files

3.1 PvMulticastMaster.cpp

- The source code file for the sample program of how to use the PvDevice class.
- Please check the comments in this source code for the information, which is how to use this class.

1.10 PvMulticastSlaveSample

This sample shows how to use the PvPipeline class, to use the multicast slave.
Please connects the camera as the multicast master with the multicasting master sample (PvMulticastSample) before use this sample.
IGMP (Internet Group Management Protocol) supported Ethernet switch is required for the multicast system.

1) Introduction

This sample shows how to:

- Opens the stream to the multicast group.
- Obtains the image.
- CANNOT control the camera when connect as the multicast slave.

2) Prerequisite

When use this sample:

- The camera has to connect to the NIC, which is installed either the eBUS Universal Pro or the Optimal driver, before use this sample.

3) Descriptions of the files

3.1 PvMulticastSlave.cpp

- The source code file, which shows how to use the PvPipeline and the PvStream classes.
- Please check the comments in this source code for the information, which is how to use these classes.

1.11 PvPipelineSample

This sample shows how to use the PvPipeline class, to obtain the image.

1) Introduction

This sample shows how to:

- Connects the camera.
- Setups the stream configuration for obtain the image.
- Obtains the image from the camera.
- Statistics the data of obtain the image.
- Stops the streaming.

2) Prerequisite

When use this sample:

The camera has to connect to the NIC, which is installed either the eBUS Universal Pro or the Optimal driver, before use this sample.

3) Descriptions of the files

3.1 PvPipelineSample.cpp

The source code file for the sample program of how to use the PvPipeline class.
Please check the comments in this source code for the information, which is how to use this class.

1.12 PvPlcAndGevEvents

This sample shows the outlines for how to use the few tasks using the StGigE SDK.
StGigE SDK does NOT have CounterSelector, CounterEventSource, CounterResetSource, CounterResetActivation and CounterValue functions in this sample. Please comments out these function in the sample code before use this.
The tasks are in the following buttons:

1) "Select/Connect" button
This shows how to connect to the camera on the selection window.

2) "Acquisition Start" ("Acquisition Stop") button
This shows how to start and stop the image acquisition and display.

3) "Setup PLC" button
This shows how to setup the camera's PLC.

3.1) Signal Routing Block setup:

```
#ifdef USE_FVAL
    PLC_I0 =PLC_A4     //PLC_A4 is the FVAL signal from camera
#else
    PLC_I0 =PLC_ctrl0
#endif
    PLC_I4 =PLC_ctrl1
```

3.2) PLC LUT setup:

```
PLC_Q7     = I0    // PLC_Interrupt
PLC_Q17    = I0    // Counter increament signal
PLC_Q3     = I4    // Counter reset
```

3.3) Enable PLC_Interrupt_FIFO0_Q7 and register PLC_Interrupt_FIFI0_IRQ_mask parameter to the PvGenEventSink.
Updates this parameter when the PLC interrupt happens.
After update, the PvGenEventSink::OnParameterUpdate() callback function call.

3.4) Setup as the rising edge of PLC_Q17 signal counts and the counter clears by PLC_Q3 signal.

3.5) When change PLC_ctrl1 from false to true, the counter clears.

4) "Generate 1 pulse" button
This is only enable when USE_FVAL is NOT defined.
This shows how to generate one pulse.
When select this button, PLC_ctrl0 change from false to true then one pulse generates to PLC_Q17 and PLC_Q7.
The counter value increases one and OnParameterUpdate() called once. The result shows up on the pop up message.

5) "Device control" button
This shows how to display "Gev Device Control" window.
PLC_ctrl0 change from false to true, is the same effect as select "Generate 1 pulse" button.

Note: a) USE_FVAL is not defined as the default.
b) Based on the GenICam XML implementation in the camera, it may not have all PLC features.

1.13 PvPlcDelayerSample

This sample shows how to make the delay for the signal.
About 3 secounds delay for the signal in this sample.

1) Introduction

This sample help to understand how to use the PvPlcDelayerSample class.
This PvPlcDelayerSample class configures the Timer1 (Pulse Generator 0) and the Delayer to delay the signal output to the TTL_OUT[0] about 3 seconds.
This sample has the monitoring feature for the output signal wave of TTL_OUT[0], but this feature does NOT work for Sentech GigE cameras.

2) Prerequisite

When use this sample:
The camera has to connect to the NIC, which is installed either the eBUS Universal Pro or the Optimal driver, before use this sample.

3) Descriptions of the files

3.1 PvPlcDelayerSample.cpp
The source code file for emits 23 Hz signal wave pulse with Timer (Pulse Generator 0).
Delayer makes the delay for the input signal.
Please check the comments in this source code for the information, which is how to use this class.

NOTE: This sample is based on the Demonstrating the Rescaler tutorial in the Programmable Logic Controller Reference Guide.

1.14 PvPlcRescalerSample

This sample shows how to use the rescaler and other PLC functions.
This sample has the monitoring feature for the output signal wave of TTL_OUT[0], but this feature does NOT work for Sentech GigE cameras.

1) Introduction

This sample help to understand how to use the rescaler and other PLC functions.
PLC_rsl0 is rescale 36Hz signal into 3.6Hz signal.
Timer1 (Pulse Generator 0) generates 36Hz signal.
PLC_ctrl0 is turn on/off the rescaler.
LUT routes rescaler output signal to Q0 (TTL_OUT[0]).

NOTE: This sample is based on the Demonstrating the Rescaler tutorial in the Programmable Logic Controller Reference Guide.

1.15 PvRecoverySample

This sample shows how to create the simple command line application for obtains the image with automatic recovery support by the PvDeviceEventSink.

1) Introduction

This sample shows how to:

- Selects the camera with the PvDeviceFinderWnd class.
- Connects the camera.
- Opens the stream.
- Obtains the image with the PvStream class.
- Obtains the recovery events and callback from the PvDeviceEventSink class.
- Handles the recovery events.

2) Prerequisite

When use this sample:

The camera has to connect to the NIC, which is installed either the eBUS Universal Pro or the Optimal driver, before use this sample.

3) Descriptions of the files

3.1 PvRecoverySample.cpp

The source code file, which is simple command line application that obtains the image.
Please check the comments in this source code for the information, which is how to use these classes.
Sentech GigE cameras do NOT have the DeviceReset function in this.

1.16 PvRGBFilterSample

This sample shows how to use the PVBufferConverter and the PvFilterRGB, to convert the image.
This sample shows how to use the PvBufferWrite class, to save the image into bitmap file.

1) Introduction

This sample shows how to:

- Converts the image to RGB32Bit with the PvBufferConverter class.
- Applies the RGB filter and the white balance with the PvFilterRGB class.
- Saves the RGB32 image into bitmap file with the PvBufferWriter class.

2) Prerequisite

When use this sample:

The camera has to connect to the NIC, which is installed either the eBUS Universal Pro or the Optimal driver, before use this sample.

3) Descriptions of the files

3.1 PvRGBFilterSample.cpp

The source code file for the sample program of how to use the PvBufferConverter, the PvFilterRGB and the PvBufferWriter classes.

1.17 PvSimpleUISample

This sample shows how to create the simple UI application, which is connects to the camera, obtains and displays the image.

1) Introduction

This sample shows how to:

- Selects the camera with the PvDeviceFinderWnd class.
- Connects the camera.
- Opens the stream.
- Starts obtain the image.
- Obtains the image with the PvStream class.
- Displays the image with the PvDisplayWnd.
- Obtains the camera functions with the PvGenParameterArray.
- Displays the camera functions with the PvGenBrowserWnd.

2) Prerequisite

When use this sample:

The camera has to connect to the NIC, which is installed either the eBUS Universal Pro or the Optimal driver, before use this sample.

3) Descriptions of the files

3.1 PvSimpleUISample.cpp

The source code file for the create UI application.

1.18 PvStreamSample

This sample shows how to connects the camera, obtains and displays the image with the PvStream class.

1) Introduction

This sample shows how to:

- Connects the camera.
- Setup the stream setting for obtains the image.
- Obtains the image.
- Stops obtains the image.

2) Prerequisite

When use this sample:

The camera has to connect to the NIC, which is installed either the eBUS Universal Pro or the Optimal driver, before use this sample.

3) Descriptions of the files

3.1 PvStreamSample.cpp

The source code file for the sample program of how to use the PvStream.

1.19 NetCommandSample

This sample shows how to use the NetCommand application.
NetCommand applications use for the multiple cameras discovery, the connection and the configuration.

1) Introduction

This sample help to understand how to use the StGigE SDK.

2) Prerequisite

When use this sample:

Please use the Visual Studio 2008 (VS9).
Required to understand the C++ and the Microsoft MFC.
Please build to make the application to understand the operation.

1.20 StGEViewerSample

This sample shows how to communicate with the camera.

1) Introduction

This sample help to understand how to use each class.
This sample help to understand how to read the sample.
This sample help to understand how to setup the camera with the GenICam command.
This sample help to understand how to setup the camera with the PVSerialPortIPEngine class.

2) Prerequisite

When use this sample:

Please use the Visual Studio 2008 (VS9).
Required to understand the C++ and the Microsoft MFC.
Please build to make the application to understand the operation.

1.21 StGEMultiCameraSample

This sample shows how to connect and display the image with the multiple cameras.

1) Introduction

This sample help to understand how to search the camera and the IP address change before connect. (IPConfiguration.cpp)
This sample help to understand how to connect the multiple cameras automatically.
This sample help to understand how to display the image with the multiple cameras.

2) Prerequisite

When use this sample:

Please use the Visual Studio 2005 (VS8) C++.
Required to understand the C++ and the Microsoft MFC.
Please build to make the application to understand the operation.

1.22 StGEOpenCVFacesDetectSample

This sample shows how to make the face detect sample program with OpenCV.
Please download OpenCV than make the library to use it.

1) Introduction

This sample help to understand how to connects the camera and obtains the image.
This sample help to understand how to make the face detect program with OpenCV.
This sample help to understand how to display the image with OpenCV.

2) Prerequisite

When use this sample:
Please use the Visual Studio 2005 (VS8) C++.
Required to understand the C++ and the Microsoft MFC.
Please build to make the application to understand the operation.
This sample made with OpenCV2.1.0.

1.23 StGEOpenCVSharpSample

This sample shows how to make the un-sharp mask sample program with OpenCV.
Please download OpenCV than make the library to use it.

1) Introduction

This sample help to understand how to connects the camera and obtains the image.
This sample help to understand how to make the un-sharp mask program with OpenCV.
This sample help to understand how to make display the image with OpenCV.

2) Prerequisite

When use this sample:
Please use the Visual Studio 2005 (VS8) C++.
Required to understand the C++ and the Microsoft MFC.
Please build to make the application to understand the operation.
This sample made with OpenCV2.1.0.

2. The sample programs for the specific usages

2.1 Software trigger

2.1.1 Software trigger with PLC function

This sample shows how to generate the software trigger signal with the PLC function.

To generate the software trigger signal:

Input the signal from the PLC_ctrl0 to Q9 through I5.

Output the signal, which is generated by the Pulse Generator 0 at the rising edge of the input signal, to the PG0_out(Timer 1 Out).

The trigger signal from the pg0_out through I4 through Q4 to the camera control FPGA.

Please check with iPORT.Reference.Programmable_Logic_Controller.pdf for more details of the PLC.

A. Connects the camera, obtains the PvDevice then sets PvStream and PvPipeline.

Please check the sample program for GEVPlayerSample, PvSimpleUISample or PvstreamSample to make the applications.

B. Sets the route for the PLC.

```
// Function for set up the PLC route for the software trigger.
BOOL SetupPLCRoute( PvDevice *pDevice )
{
    PvGenBoolean*   lPvGenBoolean;
    BOOL bReval;

    //Q4 = I7
    bReval = SetupLUT( pDevice, 4, "PLC_I7","Or","Zero","Or","Zero","Or","Zero" );
    if( !bReval ) return FALSE;
    //Q9 = I5
    bReval = SetupLUT( pDevice, 9, "PLC_I5","Or","Zero","Or","Zero","Or","Zero" );
    if( !bReval ) return FALSE;

    bReval = SetupSRB( pDevice, 5, "PLC_ctrl0" );
    if( !bReval ) return FALSE;
    bReval = SetupSRB( pDevice, 7, "Timer1Out" );
    if( !bReval ) return FALSE;

    return TRUE;
}

// Function for sets the LookupTable
BOOL SetupLUT( PvDevice *pDevice, int iQ,
      PCHAR pText1,PCHAR pText2,PCHAR pText3,PCHAR pText4,PCHAR pText5,PCHAR pText6,PCHAR pText7
      )
{
    // ********************************************** //
    // Route PLC_Ixx to Qxx                           //
    // ********************************************** //
```

```
    PvGenEnum*      lPvGenEnum;
    char PLCText[64];
    PvResult lResult;

    sprintf( PLCText, "PLC_Q%d_Variable0", iQ );
    lPvGenEnum = dynamic_cast<PvGenEnum *>(pDevice->GetGenParameters()->Get( PLCText ) );
    if(lPvGenEnum==NULL) return FALSE;
    lResult = lPvGenEnum->SetValue(pText1);
    if( !lResult.IsOK() ) return FALSE;

    sprintf( PLCText, "PLC_Q%d_Operator0", iQ );
    lPvGenEnum = dynamic_cast<PvGenEnum *>(pDevice->GetGenParameters()->Get( PLCText ) );
    if(lPvGenEnum==NULL) return FALSE;
    lResult = lPvGenEnum->SetValue(pText2);
    if( !lResult.IsOK() ) return FALSE;

    sprintf( PLCText, "PLC_Q%d_Variable1", iQ );
    lPvGenEnum = dynamic_cast<PvGenEnum *>(pDevice->GetGenParameters()->Get( PLCText ) );
    if(lPvGenEnum==NULL) return FALSE;
    lResult = lPvGenEnum->SetValue(pText3);
    if( !lResult.IsOK() ) return FALSE;

    sprintf( PLCText, "PLC_Q%d_Operator1", iQ );
    lPvGenEnum = dynamic_cast<PvGenEnum *>(pDevice->GetGenParameters()->Get( PLCText ) );
    if(lPvGenEnum==NULL) return FALSE;
    lResult = lPvGenEnum->SetValue(pText4);
    if( !lResult.IsOK() ) return FALSE;

    sprintf( PLCText, "PLC_Q%d_Variable2", iQ );
    lPvGenEnum = dynamic_cast<PvGenEnum *>(pDevice->GetGenParameters()->Get( PLCText ) );
    if(lPvGenEnum==NULL) return FALSE;
    lResult = lPvGenEnum->SetValue(pText5);
    if( !lResult.IsOK() ) return FALSE;

    sprintf( PLCText, "PLC_Q%d_Operator2", iQ );
    lPvGenEnum = dynamic_cast<PvGenEnum *>(pDevice->GetGenParameters()->Get( PLCText ) );
    if(lPvGenEnum==NULL) return FALSE;
    lResult = lPvGenEnum->SetValue(pText6);
    if( !lResult.IsOK() ) return FALSE;

    sprintf( PLCText, "PLC_Q%d_Variable3", iQ );
    lPvGenEnum = dynamic_cast<PvGenEnum *>(pDevice->GetGenParameters()->Get( PLCText ) );
    if(lPvGenEnum==NULL) return FALSE;
    lResult = lPvGenEnum->SetValue(pText7);
    if( !lResult.IsOK() ) return FALSE;

    return TRUE;

}

// Function for sets the SignalRoutingBlock
```

```
BOOL SetupSRB( PvDevice *pDevice, int iI, PCHAR pText )
{
    // ******************************************** //
    // Setup PLC_Ix as PLC_??
    // ******************************************** //
    BOOL bReval=TRUE;

    PvResult lResult;
    char PLCText[64];
    sprintf( PLCText, "PLC_I%d", iI );
    PvGenEnum* lPvGenEnum = dynamic_cast<PvGenEnum *>(pDevice->GetGenParameters()->Get( PLCText ) );
    if(lPvGenEnum !=NULL)
    {
        lResult = lPvGenEnum->SetValue( pText );
        if( !lResult.IsOK() )
           bReval=FALSE;
    }
    else
    {
        bReval=FALSE;
    }

    return bReval;
}
```

C. Sets the pulse generator.

```
// Function for set up the pulse generator.
//-----------------------------------------------------------
//durationOfHigh: Time for the High status of the pulse
//durationOfLow: Time for the Low status of the pulse
//bTrgMode  : Trigger mode, True: Continuously False: Single pulse
//pulseScale: Unit for the pulse. Sets the 1 for 30 nseconds, sets for 10,000 for 30 useconds.
//-----------------------------------------------------------
BOOL SetPalseGenerator( int index, PvDevice *pDevice, int durationOfHigh, int durationOfLow, BOOL
bTrgMode, int pulseScale )
{
    char PLCText[64];
    PvGenEnum* lPvGenEnum;
    PvGenInteger* lPvGenInteger;
    PvResult lResult;

    sprintf( PLCText, "TimerSelector" );
    lPvGenEnum = dynamic_cast<PvGenEnum *>(pDevice->GetGenParameters()->Get( PLCText) );
    if(lPvGenEnum==NULL) return FALSE;
    lResult = lPvGenEnum->SetValue(index);
    if( !lResult.IsOK() ) return FALSE;

    sprintf( PLCText, "TimerDurationRaw" );
    lPvGenInteger = dynamic_cast<PvGenInteger *>(pDevice->GetGenParameters()->Get( PLCText ) );
    if(lPvGenInteger==NULL) return FALSE;
    lResult = lPvGenInteger->SetValue( durationOfHigh );
```

```
    if( !lResult.IsOK() ) return FALSE;

    sprintf( PLCText, "TimerDelayRaw" );
    lPvGenInteger = dynamic_cast<PvGenInteger *>(pDevice->GetGenParameters()->Get( PLCText ) );
    if(lPvGenInteger==NULL) return FALSE;
    lResult = lPvGenInteger->SetValue( durationOfLow );
    if( !lResult.IsOK() ) return FALSE;

    sprintf( PLCText, "TimerGranularityFactor" );
    lPvGenInteger = dynamic_cast<PvGenInteger *>(pDevice->GetGenParameters()->Get( PLCText ) );
    if(lPvGenInteger==NULL) return FALSE;
    lResult = lPvGenInteger->SetValue( pulseScale-1 );
    if( !lResult.IsOK() ) return FALSE;

    sprintf( PLCText, "TimerTriggerSource" );
    lPvGenEnum = dynamic_cast<PvGenEnum *>(pDevice->GetGenParameters()->Get( PLCText) );
    if(lPvGenEnum==NULL) return FALSE;
    lResult = lPvGenEnum->SetValue( bTrgMode );
    if( !lResult.IsOK() ) return FALSE;

    return TRUE;
}
```

D. Sets the operation mode to the trigger mode.

```
// Sets the operation mode.
//bTriggerMode:  Operation mode, True: Trigger mode, False: Free-run mode
BOOL SetTriggerMode( PvDevice *pDevice, BOOL bTriggerMode )
{
    PvResult lResult;
    PvGenEnum* PvGenEnum=dynamic_cast<PvGenEnum*>(pDevice->GetGenParameters()->Get("TriggerMode"));
    if( lPvGenEnum==NULL ) return FALSE;
    if( bTriggerMode ){
        lResult = lPvGenEnum->SetValue("On");
    }
    else
    {
        lResult = lPvGenEnum->SetValue("Off");
    }
    return (BOOL)lResult.IsOK();
}
```

E. Sets the exposure mode.

```
// Sets the exposure mode
//iExposureMode :  0:Timed  1:TriggerWidth
BOOL SetExposureMode( PvDevice *pDevice, int iExposureMode )
{
  PvGenEnum* lPvGenEnum=dynamic_cast<PvGenEnum*>(pDevice->GetGenParameters()->Get("ExposureMode"));
  if( lPvGenEnum==NULL ) return FALSE;
  PvResult lResult = lPvGenEnum->SetValue(iExposureMode);

  return (BOOL)lResult.IsOK();
```

```
}
```

F. Sets the "Software trigger" for the trigger source.

```
//iTriggerSource :  0:Software  1:Hardware
BOOL SetTriggerSource( PvDevice *pDevice, int iTriggerSource )
{
   PvGenEnum* lPvGenEnum=dynamic_cast<PvGenEnum*>(pDevice->GetGenParameters()->Get("TriggerSource"));
   if( lPvGenEnum==NULL ) return FALSE;
   PvResult lResult = lPvGenEnum->SetValue(iTriggerSource);

   return (BOOL)lResult.IsOK();
}
```

G. Sets the "PLC" for the software trigger source.

```
//iTriggerSoftwareSource :  0:PLC  1:UserFPGA 2:Command
BOOL SetTriggerSoftwareSource( PvDevice *pDevice, int iTriggerSoftwareSource )
{
   PvGenEnum* lPvGenEnum=dynamic_cast<PvGenEnum*>(pDevice->GetGenParameters()->Get("TriggerSoftwareSource"));
   if( lPvGenEnum==NULL ) return FALSE;
   PvResult lResult = lPvGenEnum->SetValue(iTriggerSoftwareSource);

   return (BOOL)lResult.IsOK();
}
```

H. Input the trigger.

```
// 1mseconds pulse input to the PLC_ctrl0
BOOL OneShotTrigger( PvDevice *pDeivce )
{
    BOOL bResult;
    bResult = SetControlBit( pDeivce, 0, true );
    if( !bResult ) return FALSE;
    Sleep(1); //1mS
    bResult = SetControlBit( pDeivce, 0, false );
    if( !bResult ) return FALSE;
    return TRUE;
}

// Function for PLC_ctrl (ON/OFF)
BOOL  SetControlBit( PvDevice *pDevice, int index, bool bFlg  )
{
    char PLCText[64];
    sprintf( PLCText, "PLC_ctrl%d", index);
    PvGenBoolean* lGenBoolean;
    lGenBoolean = dynamic_cast<PvGenBoolean *>(pDevice->GetGenParameters()->Get( PLCText ) );
    if(  lGenBoolean==NULL ) return FALSE;
    PvResult lResult = lGenBoolean->SetValue(bFlg);
    return (BOOL)lResult.IsOK();
}
```

2.1.2 Software trigger with GenICam command

This sample shows how to generate the software trigger signal with the GenICam command.

A. Connects the camera, obtains the PvDevice then sets PvStream and PvPipeline.
Please check the sample program for GEVPlayerSample, PvSimpleUISample or PvstreamSample to make the applications.

B. Sets the operation mode to the trigger mode.

```
// Sets the operation mode.
//bTriggerMode:  Operation mode, True: Trigger mode, False: Free-run mode
BOOL SetTriggerMode( PvDevice *pDevice, BOOL bTriggerMode )
{
    PvResult lResult;
    PvGenEnum* lPvGenEnum=dynamic_cast<PvGenEnum *>(pDevice->GetGenParameters()->Get("TriggerMode"));
    if( lPvGenEnum==NULL ) return FALSE;
    if( bTriggerMode ){
        lResult = lPvGenEnum->SetValue("On");
    }
    else
    {
        lResult = lPvGenEnum->SetValue("Off");
    }
    return (BOOL)lResult.IsOK();
}
```

C. Sets the exposure mode.

```
// Sets the exposure mode
//iExposureMode :  0:Timed  1:TriggerWidth
BOOL SetExposureMode( PvDevice *pDevice, int iExposureMode )
{
   PvGenEnum* lPvGenEnum=dynamic_cast<PvGenEnum *>(pDevice->GetGenParameters()->Get("ExposureMode"));
   if( lPvGenEnum==NULL ) return FALSE;
   PvResult lResult = lPvGenEnum->SetValue(iExposureMode);

   return (BOOL)lResult.IsOK();
}
```

D. Sets the exposure time.

```
//iExposureTime :  Min:0  Max:16777215
BOOL SetExposureMode( PvDevice *pDevice, int iExposureTime )
{
  PvGenInteger* lPvGenInteger=dynamic_cast<PvGenInteger*>(pDevice->GetGenParameters()->Get("ExposureTimeRaw"));
  if( lPvGenInteger==NULL ) return FALSE;
  PvResult lResult = lPvGenInteger->SetValue(iExposureTime);

  return (BOOL)lResult.IsOK();
}
```

E. Sets the "Software trigger" for the trigger source.

```
//iTriggerSource :  0:Software  1:Hardware
```

```
BOOL SetTriggerSource( PvDevice *pDevice, int iTriggerSource )
{
  PvGenEnum* lPvGenEnum=dynamic_cast<PvGenEnum *>(pDevice->GetGenParameters()->Get("TriggerSource"));
  if( lPvGenEnum==NULL ) return FALSE;
  PvResult lResult = lPvGenEnum->SetValue(iTriggerSource);

  return (BOOL)lResult.IsOK();
}
```

F. Sets the "Command" for the software trigger source.

```
//iTriggerSoftwareSource :  0:PLC  1:UserFPGA 2:Command
BOOL SetTriggerSoftwareSource( PvDevice *pDevice, int iTriggerSoftwareSource )
{
  PvGenEnum* lPvGenEnum=dynamic_cast<PvGenEnum*>(pDevice->GetGenParameters()->Get("TriggerSoftwareSource"));
  if( lPvGenEnum==NULL ) return FALSE;
  PvResult lResult = lPvGenEnum->SetValue(iTriggerSoftwareSource);

  return (BOOL)lResult.IsOK();
}
```

G. Input the trigger.

```
BOOL SetTriggerSoftware( PvDevice *pDevice )
{
  PvGenCommand* lPvGenCommand=dynamic_cast<PvGenCommand*>(pDevice->GetGenParameters()->Get("TriggerSoftware"));
  if( lPvGenCommand==NULL ) return FALSE;
  PvResult lResult = lPvGenCommand->Execute();

  return (BOOL)lResult.IsOK();
}
```

2.2 Obtain the NIC and the camera information

This sample shows how to obtain the NIC and the camera information without the PvDeviceFinderWnd class

The most of the sample program use the PvDeviceFinderWnd class to selects and connects the camera.
The following code of the program can selects and connects the camera without the PvDeviceFinderWnd dialog.
Also, please check the PvDeviceFindingSample sample.

```
// Displays the MAC Address, IP Address and Subnet Mask information for each NIC and each camera.
BOOL PrintNICandDeviceInformation(void)
{
    PvSystem lSystem;
    PvResult lResult;
    PvDeviceInfo *lDeviceInfo = 0;
    lSystem.SetDetectionTimeout( 2000 );
    lResult= lSystem.Find();
    if( !lResult.IsOK() )
    {
        printf( "PvSystem::Find Error: %s¥n", lResult.GetCodeString() );
    }else{
        PvUInt32 lInterfaceCount = lSystem.GetInterfaceCount();

        for( PvUInt32 x = 0; x < lInterfaceCount; x++ )
        {
            // get pointer to each of interface
            PvInterface * lInterface = lSystem.GetInterface( x );

            printf( "Interface %i MAC Address: %s IP Address: %s Subnet Mask: %s¥n",
                      x,
                      lInterface->GetMACAddress().GetUnicode(),
                      lInterface->GetIPAddress().GetUnicode(),
                      lInterface->GetSubnetMask().GetUnicode() );

           // Get the number of GEV devices that were found using GetDeviceCount.
           PvUInt32 lDeviceCount = lInterface->GetDeviceCount();

           for( PvUInt32 y = 0; y < lDeviceCount ; y++ )
           {
               lDeviceInfo = lInterface->GetDeviceInfo( y );
               printf( "Device %i MAC Address: %s IP Address: %s ¥n",
                            y,
                            lDeviceInfo->GetMACAddress().GetUnicode(),
                            lDeviceInfo->GetIPAddress().GetUnicode()
                       );
           }
       }
    }
    return (BOOL)lResult.IsOK();
}

//IP Address, Subnet Mask and Default gateway can be changeable with the MAC Address information before
```

connects the camera.

```
//lIPAddress = "192.168.2.45"; (Changes to the specified IP Address)
//lSubnetmask= "255.255.255.0"; (Changes to the specified Subnet Mask)
//lGateway = "0.0.0.0"; (Changes to the specified Default Gateway Address)
BOOL ChangeDeviceInformation(PvDeviceInfo *lDeviceInfo, PvString lIPAddress, PvString lSubnetmask, PvString lGateway)
{
  PvDevice lDevice;
  PvResult lResult = lDevice.SetIPConfiguration(lDeviceInfo->GetMACAddress(), lIPAddress , lSubnetmask , lGateway );

  return (BOOL)lResult.IsOK();
}
```

2.3 Sets the IP Address

This sample shows how to set the IP Address, the subnet mask and the default gateway.

The default setting of the IP Address is sets automatically.
The certain IP Address, the subnet mask and the default gateway can sets for the default camera settings.
The following sample program shows how to save the current IP Address, the subnet mask and default gateway to the camera.

```
BOOL SetDefaultIPAddress( PvDevice *lDevice )
{
    PvGenParameterArray *lGenDevice = lDevice->GetGenParameters();
    if( !lGenDevice ) return FALSE;

    PvResult lResult;
    //GevCurrentIPConfigurationPersistentIP
    PvGenBoolean* PersistentIPBoolean=dynamic_cast<PvGenBoolean*>(lGenDevice->Get("GevCurrentIPConfigurationPersistentIP"));
    if( !lPersistentIPBoolean ) return FALSE;
    lResult = lPersistentIPBoolean->SetValue(true);
    if( lResult.IsFailure() )  return FALSE;

    //Read Infomation
    PvInt64 lIPAddress = 0;
    PvInt64 lSubnetMask = 0;
    PvInt64 lGateway = 0;
    PvGenInteger *lIPAddressParam = dynamic_cast<PvGenInteger *>(lGenDevice->Get("GevCurrentIPAddress"));
    if( !lPersistentIPBoolean ) return FALSE;
    PvGenInteger *lSubnetMaskParam = dynamic_cast<PvGenInteger *>(lGenDevice->Get("GevCurrentSubnetMask"));
    if( !lSubnetMaskParam ) return FALSE;
    PvGenInteger *lGatewayParam = dynamic_cast<PvGenInteger *>(lGenDevice->Get("GevCurrentDefaultGateway"));
    if( !lSubnetMaskParam ) return FALSE;
    lResult = lIPAddressParam->GetValue( lIPAddress );
    if( lResult.IsFailure() ) return FALSE;
    lResult = lSubnetMaskParam->GetValue( lSubnetMask );
    if( lResult.IsFailure() ) return FALSE;
    lResult = lGatewayParam->GetValue( lGateway );
    if( lResult.IsFailure() ) return FALSE

    //Set Infomation
    PvGenInteger *lFlushIPAddressParam=dynamic_cast<PvGenInteger *>(lGenDevice->Get("GevPersistentIPAddress"));
    if( !lFlushIPAddressParam ) return FALSE;
    lResult = lFlushIPAddressParam->SetValue( lIPAddress );
    if( lResult.IsFailure() ) return FALSE;

    PvGenInteger *lFlushSubnetMaskParam=dynamic_cast<PvGenInteger *>(lGenDevice->Get("GevPersistentSubnetMask"));
    if( !lFlushSubnetMaskParam ) return FALSE;
    lResult = lFlushSubnetMaskParam->SetValue( lSubnetMask );
    if( lResult.IsFailure() ) return FALSE;

    PvGenInteger *lFlushGatewayParam=dynamic_cast<PvGenInteger *>(lGenDevice->Get("GevPersistentDefaultGateway"));
    if( !lFlushGatewayParam ) return FALSE;
```

```
        IResult = IFlushGatewayParam->SetValue( IGateway );
        if( IResult.IsFailure() ) return FALSE;

        return TRUE;

    }
```

2.4 Obtains the command information

This sample shows how to obtain the command information.

```
BOOL PrintPvGenParameters( PvDevice *lDevice )
{
    PvGenParameterArray *lGenDevice = lDevice->GetGenParameters();
    if( !lGenDevice ) return FALSE;

    PvInt64 lCount = lGenDevice->GetCount();
    PvString aName, strValue, lStr;
    PvInt64 iValue, iMax, iMin;
    bool bValue;
    double dblValue, dblMax, dblMin;
    PvResult lResult;
    PvGenVisibility lVisibility;

    printf("lCount=%d ¥n", lCount);
    for( int i=0; i<lCount; i++ ){
        PvGenParameter *p_pvPara = lGenDevice->Get( i );

        //Name
        p_pvPara->GetName( aName );
        printf( "[%d]ParameterName=%s ¥n", i, aName.GetAscii());

        //Description
        p_pvPara->GetDescription( lStr );
        printf("   Description=%s ¥n", lStr.GetAscii());

        //Category
        p_pvPara->GetCategory( lStr );
        printf("   Category=%s ¥n", lStr.GetAscii());

        //Visibility
        p_pvPara->GetVisibility( lVisibility );
        if( lVisibility==0 )   lStr="Beginner";
        else if( lVisibility==1 )  lStr="Expert";
        else if( lVisibility==2 )  lStr="Guru";
        else       lStr="Invisible";
        printf("   Visibility=%s ¥n", lStr.GetAscii());

        //Type
        PvGenType aType;
        p_pvPara->GetType( aType );
        if( aType==PvGenTypeInteger )  lStr="PvGenTypeInteger"; //0
        else if( aType==PvGenTypeEnum )  lStr="PvGenTypeEnum";  //1
        else if( aType==PvGenTypeBoolean ) lStr="PvGenTypeBoolean"; //2
        else if( aType==PvGenTypeString ) lStr="PvGenTypeString";  //3
        else if( aType==PvGenTypeCommand ) lStr="PvGenTypeCommand"; //4
        else if( aType==PvGenTypeFloat ) lStr="PvGenTypeFloat";  //5
        else        lStr="PvGenTypeUndefined";
```

```
    printf("   Type=%s¥n", lStr.GetAscii());

    if( aType==PvGenTypeInteger ){ //0
        PvGenInteger *lGenInteger = dynamic_cast<PvGenInteger *>( p_pvPara );
        lResult = lGenInteger->GetValue(iValue);
        if( lResult.IsOK() ) printf("   Value=%d¥n", iValue );
        lResult = lGenInteger->GetMax(iMax);
        if( lResult.IsOK() ) printf("   Max=%d¥n", iMax );
        lResult = lGenInteger->GetMin(iMin);
        if( lResult.IsOK() ) printf("   Min=%d¥n", iMin );
    }
    else if( aType==PvGenTypeEnum ){ //1
        PvGenEnum *lGenEnum = dynamic_cast<PvGenEnum *>( p_pvPara );
        lResult = lGenEnum->GetValue(strValue);
        if( lResult.IsOK() ) lResult = lGenEnum->GetValue(iValue);
        if( lResult.IsOK() ) printf("   Value=%s(%d)¥n", strValue.GetAscii(),iValue );

        PvInt64 entryCount;
        lGenEnum->GetEntriesCount( entryCount );
        for( PvUInt32 j=0; j<entryCount; j++ )
        {
            const PvGenEnumEntry *lEntry = NULL;
            lGenEnum->GetEntryByIndex( j, &lEntry );
            lEntry->GetName( strValue );
            printf( "    Entry[%d] = %s¥n", j,strValue.GetAscii() );
        }
    }
    else if( aType==PvGenTypeBoolean ){ //2
        PvGenBoolean *lGenBoolean = dynamic_cast<PvGenBoolean *>( p_pvPara );
        lResult = lGenBoolean->GetValue(bValue);
        if( lResult.IsOK() ) printf("   Value=%d¥n", bValue );
    }
    else if( aType==PvGenTypeString ){ //3
        PvGenString *lGenString = dynamic_cast<PvGenString *>( p_pvPara );
        lResult = lGenString->GetValue(strValue);
        if( lResult.IsOK() ) printf("   Value=%s¥n", strValue.GetAscii() );
    }
    else if( aType==PvGenTypeCommand ){ //4
    }
    else if( aType==PvGenTypeFloat ){ //5
        PvGenFloat *lGenFloat = dynamic_cast<PvGenFloat *>( p_pvPara );
        lResult = lGenFloat->GetValue(dblValue);
        if( lResult.IsOK() ) lResult = lGenFloat->GetUnit(strValue);
        if( lResult.IsOK() ) printf("   Value=%f[%s]¥n", dblValue, strValue.GetAscii() );
        lResult = lGenFloat->GetMax(dblMax);
        if( lResult.IsOK() ) printf("   Max=%f¥n", dblMax );
        lResult = lGenFloat->GetMin(dblMin);
        if( lResult.IsOK() ) printf("   Min=%f¥n", dblMin );
    }
    else{
    }
```

```
            }
            return TRUE;
        }
```

2.5 Uses the serial communication

This sample shows how to use the serial communication between the camera and the PC.

Communicates to the main FPGA of the camera when selects "PvIPEngineSerial0" for the port number.
Sentech GigE cameras do NOT use "PvIPEngineSerial1".
The following sample program shows how to sends 6Byte data then receives 4Byte data.

```
//SerialPort Open
BOOL SerialPortOpen( PvDevice *pDevice, PvSerialPortIPEngine *pPort, INT iPortNo )
{
  PvResult lResult;
  if( iPortNo==0 ) lResult = pPort->Open( pDevice, PvIPEngineSerial0 );
  else    lResult = pPort->Open( pDevice, PvIPEngineSerial1 );
  return lResult.IsOK();
}

//SerialPort Close
BOOL SerialPortClose( PvSerialPortIPEngine *pPort )
{
  BOOL bReval=TRUE;
  if( pPort->IsOpened() ){
    PvResult lResult = pPort->Close();
    bReval = lResult.IsOK();
  }
  return bReval;
}

//Data Send and Recieve
BOOL SerialDataSendRev( PvSerialPortIPEngine *pPort, PBYTE pbyteSend, WORD wSendsize, PBYTE pbyteRev,
                   WORD wRevsize, PWORD pReadsize  )
{
  PvUInt32 aBytesWritten;
  PvResult lResult = pPort->Write( pbyteSend, wSendsize, aBytesWritten );
  if( lResult.IsOK() ){
    PvUInt32 readBytes;
    DWORD dwTimes = 100;
    do{
      lResult = pPort->GetRxBytesReady(readBytes);
      if( lResult.IsOK() && readBytes>=wRevsize )
        break;
      Sleep(10);
    }while(--dwTimes);

    PvUInt32 aBytesRead;
    PvUInt32 aTimeout=1000;
    lResult = pPort->Read( pbyteRev, wRevsize, aBytesRead, aTimeout );
    if( lResult.IsOK() ) *pReadsize = aBytesRead;
  }
  return lResult.IsOK();
}
```

```
BOOL GetSerialData(PvDevice *pDevice)
{
  PvSerialPortIPEngine lSerialPort;
  BOOL bReval;
  bReval = SerialPortOpen( pDevice, &lSerialPort, 0 );
  if( !bReval ) return FALSE;

  BYTE byteSend[] = {0x02,0x00,0x30,0x01,0x00,0x03};

  printf( "SendData " );
  for( int i=0; i<sizeof(byteSend); i++ )
    printf( "%02X ",byteSend[i] );
  printf( "¥n¥n" );

  BYTE byteRcv[4];
  WORD wReadSize=0;
  bReval=SerialDataSendRev(&lSerialPort,&byteSend[0],sizeof(byteSend),&byteRcv[0],sizeof(byteRcv),
                           &wReadSize );
  if( bReval ){
    printf( "RcvData " );
    for( int i=0; i<wReadSize; i++ )
      printf( "%02X ",byteRcv[i] );
    printf( "¥n¥n" );
  }
  SerialPortClose( &lSerialPort );

  return bReval;
}
```

Revision History

Rev	Date	Changes	Note
2.00	2012/11/05	New document	

Sensor Technology Co., Ltd

7F, Harada center building
9-17, Naka cho 4 chome
Atsugi-city, Kanagawa
243-0018 Japan
TEL 81-46-295-7061 FAX 81-46-295-7066
URL http://www.sentech.co.jp/