Sentech GigE Vision カメラ

StGigE SDK

サンプルガイド

センサーテクノロジー株式会社

目次

1. サンプルプログラム

1.1 eBUS Driver Installation API

このサンプルは、ネットワーク構成についての情報を表示し、PC 上の LAN カードに eBUS ドライバ(Universal または Optimal)をインストールする方法を記述しています。

1) Introduction

StGigE SDK の使い方及び、サンプルの見方
* PC にインストールされたネットワークアダプターとドライバーの情報の表示。
* 特定のアダプター(LAN カード)へのドライバーのインストールおよび、アンインストール。
* ドライバーインストール時に、いつ PC を再起動したらよいかわかります。

2) 前提条件

このサンプルは StGigE SDK を使用して eBus ドライバーをソフトウェアに含める場合に参考にしてください。
使用前に SDK に含まれている Custom Install Guide を読んでください。

3) ファイルの説明

3.1 main.cpp
EbNetworkAdapter、EbDriver と EbInstaller クラスの使用方法を記述しています。

1.2 GEVPlayerSample

このサンプルは、StGigE Player アプリケーションのソースコードです。
StGigE Player アプリケーションはカメラの検出、接続、構成に用いられます。

1) Introduction

このサンプルコードは、StGigE SDK の使用方法を深く理解できます。

2) 前提条件

このサンプルを使用する前に StGigE Player アプリケーションを使用してアプリケーション動作を確認してください。このサンプルを使用するには C++とマイクロソフトの MFC を良く理解している必要があります。

3) ファイルの説明

3.1 AboutBox.cpp
StGigE Player の About dialog のソースコードです。
メニューの Help ＞ About StGigE Player..からアクセスします。

3.2 BitmapButton.cpp
StGigE Player のユーザーインターフェースの Play ボタンと Stop ボタンを使用しているクラス。

3.3 BufferOptionsDlg.cpp
Buffer Options dialog のソースコードです。
メニューの Tools > Buffer Options...からアクセスします。

3.4 ConnectionThread.cpp
接続を実行したり、メッセージを ProgressDlg に渡したりするクラスを作成するスレッド（ProgressDlg.cpp 参照）。

3.5 EventMonitorDlg.cpp
Event Monitor dialog のソースコードです。
メニューの Tools > Event Monitor...からアクセスします。

3.6 FilteringDlg.cpp
Image Filtering dialog のソースコードです。
メニューの Tools > Image Filtering...からアクセスします。

3.7 GEVPlayer.cpp
InitInstance で GEVPlayer dialog を実行する CWinApp クラスです。

3.8 GEVPlayer.cpp
StGigE Player を実行した時に、表示される UI など、ほとんどのソースコードが含まれています。

3.9 HTMLDialog.cpp
HTML ウインドウを表示させるための、親クラス。

3.10 ImageSaveDlg.cpp
Image Save dialog のソースコードです。
メニューの Tools > Save Images...からアクセスします。

3.11 LoadingThread.cpp
pvcfg ファイルからロードして接続を実行したり、メッセージを ProgressDlg に渡したりするクラスを作成するスレッド（ProgressDlg.cpp 参照）。

3.12 LogBuffer.cpp
Event Monitor ダイアログで、イベントログに使用（EventMonitorDlg.cpp 参照）。

3.13 ParameterInfo.cpp
GenICam events の為に、event monitor dialog によって使用。

3.14 ProgressDlg.cpp
カメラとの接続中や、構成ファイルを読込や保存している時のダイアログを表示。

3.15 RegisterInterfaceDlg.cpp
Ctrl+Alt+R を押した時、Register interface dialog が表示されます。

3.16 SavingThread.cpp
ProgressDlg でメッセージ表示中に、pvcfg file を保存するクラスを作成するスレッド(ProgressDlg.cpp 参照)。

3.17 SetupDlg.cpp
Setup dialog のソースコードです。
メニューの Tools > Setup...からアクセスします。

3.18 SplashPage.cpp
StGigE Player を起動時(メインダイアログが表示されるまでの間)に、表示されるダイアログ。

3.19 Thread.cpp
ThreadDisplay によって使用されるクラス。

3.20 ThreadDisplay.cpp
映像取り込み、表示するクラスを作成するスレッド。

3.21 stdafx.cpp
標準 MFC アプリケーションの為のソースファイル。

1.3 PvBufferWriterSample

PvBufferWriter と PvBuffer クラスの使用方法を記述しているサンプルコードです。

1) Introduction

PvBuffer クラスの使い方及び、サンプルの見方
＊ バッファの作成、割り当て、画像データ取得。

PvBufferWriter クラスの使い方及び、サンプルの見方
＊ バッファデータを bitmap に保存。

2) 前提条件

このサンプルを動作させるには:
＊ Universal か Optimal がインストールされている LAN カードとカメラを LAN ケーブルで接続してください。

3) ファイルの説明

3.1 PvBufferWriterSample.cpp
PvBufferWriter と PvBuffer クラスを使用する方法を記述しています。これらのクラスが当サンプルでどのように使用しているかは、ソースコードのコメントを参照してください。

1.4 PvCamHeadSerialComLogSample

カメラのデバイス制御とストリーミング機能を使用しているアプリケーションの作成方法を記述しています。
カメラ内部での IP Engine と FPGA 間でのシリアル通信をモニターすることが可能です。

1) Introduction

このアプリケーションは、PvSimpleUISample アプリケーションに基づきます。
MFC ダイアログアプリケーションとして作成されたサンプルです。

映像ストリーム/表示のスレッドコードを除いて、ほとんどの機能は
PvCamHeadSerialComLogSampleDlg に書かれています。

2) 前提条件

このサンプルを動作させるには:
* Universal か Optimal がインストールされている LAN カードとカメラを LAN ケーブルで接続してください。

3) ファイルの説明

3.1 PvCamHeadSerialComLogSampleDlg.cpp
メインのアプリケーションダイアログ。ほとんどのイベントはこのダイアログにあります。

3.2 PvCamHeadSerialComLogSample.cpp
メインのアプリケーション。メインのアプリケーションダイアログを表示させます。

3.3 Thread.cpp
PvPipeline で取得した画像データを表示するクラスを作成するスレッド。

1.5 PvConfigurationReaderSample

PvConfigurationReader と PvConfigurationWriter クラスの使用方法を記述しているサンプルコードです。

1) Introduction

PvConfigurationWriter クラスの使い方及び、サンプルの見方
* カメラ情報を pvxml ファイルへ３種類の構成で保存します。(カメラ、ストリーム、文字列の３種類)
* カメラ構成を指定の名前で保存します。
* ストリーム構成を指定の名前で保存します。
* 文字列情報を指定の名前で保存します。

PvConfigurationReader クラスの使い方及び、サンプルの見方
* カメラとストリーム構成をリストアします。
* 文字列情報をリストアします。

2) 前提条件

このサンプルを動作させるには:
* Universal か Optimal がインストールされている LAN カードとカメラを LAN ケーブルで接続してください。

3) ファイルの説明

3.1 PvConfigurationReaderSample.cpp
PvConfigurationReader と PvConfigurationWriter クラスを使用する方法を記述しています。これらのクラスが当サンプルでどのように使用しているかは、ソースコードのコメントを参照してください。

1.6 PvCustomPipelineSample

PvPipeline の作成方法を記述しているサンプルコードです。

1) Introduction

PvPipeline は画像取込の標準的な動作をします。
アプリケーションを作成する際、標準的ではない PvPipeline 動作又は詳細な PvPipeline 処理が必要となる場合、このソースコードから改良して開発を始めることができます。

2) 前提条件

このサンプルを動作させるには:
＊ Universal か Optimal がインストールされている LAN カードとカメラを LAN ケーブルで接続してください。

3) ファイルの説明

3.1 CustomPipeline.cpp, CustomPipeline.h
PvPipeline のソースコードです。

3.2 PvCustomPipeline.cpp
PvPipeline を改良して使用しているサンプルコードです。これらのクラスが当サンプルでどのように使用しているかは、ソースコードのコメントを参照してください。

1.7 PvDeviceFindingSample

PvSystem, PvInterface, PvDeviceInfo と PvDeviceFinderWnd クラスの使用方法を記述しているサンプルコードです。

1) Introduction

PvSystem と PvInterface クラスの使い方及び、サンプルの見方:
＊ ネットワークに接続されているカメラを見つけます。

PvDeviceFinderWnd クラスの使い方及び、サンプルの見方:
＊ GUI を使用して、ネットワークに接続されているカメラを見つけます。

2) 前提条件

このサンプルを動作させるには:
＊ Universal か Optimal がインストールされている LAN カードとカメラを LAN ケーブルで接続してください。

3) ファイルの説明

3.1 PvDeviceFindingSample.cpp
上記で説明しているサンプルコードです。これらのクラスが当サンプルでどのように使用しているかは、ソースコードのコメントを参照してください。

1.8 PvGenParameterArraySample

PvGenParameterArray を使用してプログラム上でカメラの項目を制御する方法を記述しているサンプルコードです。

1) Introduction

PvGenParameterArray クラスの使い方及び、サンプルの見方:
* PC のコミュニケーション関連の設定を取得。
* カメラの設定を取得。
* イメージストリームコントロールの設定を取得。

PvGenParameter クラスの使い方及び、サンプルの見方:
* PvGenParameter 値の設定と取得。
* PvGenParameter の各項目名を取得
* PvGenParameterArray のすべてのパラメーターを取得
* パラメーター型をチェック
* PvGenParameter をサブクラス型にキャスト

2) 前提条件

このサンプルを動作させるには:
* Universal か Optimal がインストールされている LAN カードとカメラを LAN ケーブルで接続してください。

3) ファイルの説明

3.1 PvGenParameterArraySample.cpp
PvGenParameterArray と PvGenParameter クラスを使用する方法を記述しています。これらのクラスが当サンプルでどのように使用しているかは、ソースコードのコメントを参照してください。

1.9 PvMulticastMasterSample

PvDevice を使用してマルチキャストマスターを制御する方法を記述しています。
１台のカメラをマスターとして接続し、Multicasting Slave Sample (PvMulticastSlaveSample) でスレーブとして接続して使用することができます。
IGMP (Internet Group Management Protocol)をサポートするハブ(IP アドレス、ポート)に接続しなければなりません。

1) Introduction

PvDevice の使い方:
* カメラ接続。
* マルチキャストを行う為のカメラ構成。
* ストリーミングスタート。
* ストリーミングストップ。

2) 前提条件

このサンプルを動作させるには:
* Universal か Optimal がインストールされている LAN カードとカメラを LAN ケーブルで接続してください。

3) ファイルの説明

3.1 PvMulticastMaster.cpp
マルチキャストマスターを制御する為の PvDevice クラスを使用する方法を記述しています。これらのクラスが当サンプルでどのように使用しているかは、ソースコードのコメントを参照してください。

1.10 PvMulticastSlaveSample

マルチキャストスレーブとして PvPipeline クラスの使用方法を記述しているサンプルコードです。
Multicasting master sample (PvMulticastMasterSample)を使用して１台のカメラをマスターとして接続してから、このサンプルを動作させてください。
IGMP (Internet Group Management Protocol)をサポートするハブ(IP アドレス、ポート)に接続しなければなりません。

1) Introduction

PvPipeline (PvStream 含む)の使い方:
* マルチキャストグループとしてストリームをオープン。
* 画像を取得。
* スレーブで接続した場合、カメラの制御はできません。

2) 前提条件

このサンプルを動作させるには:
* Universal か Optimal がインストールされている LAN カードとカメラを LAN ケーブルで接続してください。

3) ファイルの説明

3.1 PvMulticastSlave.cpp
マルチキャストスレーブとして画像を受け取る為の PvPipeline(PvStream 含む)クラスを使用する方法を記述しています。これらのクラスが当サンプルでどのように使用しているかは、ソースコードのコメントを参照してください。

1.11 PvPipelineSample

画像獲得のための PvPipeline クラスの使用方法を記述しているサンプルコードです。

1) Introduction

PvPipeline クラスの使い方及び、サンプルの見方:
* カメラの接続。
* 画像獲得のためのストリーム構成セットアップ。
* カメラから画像を取得。
* 画像取得データの統計。
* ストリームストップ。

2) 前提条件

このサンプルを動作させるには:
* Universal か Optimal がインストールされている LAN カードとカメラを LAN ケーブルで接続してください。

3) ファイルの説明

3.1 PvPipelineSample.cpp
画像を取得する為の PvPipeline クラスの使用方法を記述しています。これらのクラスが当サンプルでどのように使用しているかは、ソースコードのコメントを参照してください。

1.12 PvPlcAndGevEvents

StGigE SDK を使用していくつかの動作をおこなう方法の概要を説明しているサンプルコードです。
動作は以下のボタンにまとめられています。

1) "Select/Connect"ボタン。Gev Device 選択画面で、カメラに接続する方法を記述しています

2) "Acquisition Start" ("Acquisition Stop") ボタン。画像取込、表示のスタート(ストップ)の方法を記述しています。

3) "Setup PLC" ボタン。カメラ PLC のセットアップ方法を記述しています。
3.1) Signal Routing Block のセットアップ:

```
#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 のセットアップ:

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

3.3) PLC_Interrupt_FIFO0_Q7 を有効にして、PvGenEventSink に PLC_Interrupt_FIFI0_IRQ_mask パラメータを登録してください。
PLC 割り込み時に、このパラメータは更新されます。
そして PvGenEventSink：OnParameterUpdate()コールバック機能が呼ばれます。

3.4) PLC_Q17 信号の立ち上がりエッジをカウントし、PLC_Q3 信号によってカウントをクリアさせるようにセットアップします。

3.5) PLC_ctrl1 を false から true にするとカウンタがクリアーされます。

4) "Generate 1 pulse" ボタン。

USE_FVAL が定義(#define USE_FVAL)されていない場合に、ボタンが有効になります
このボタンを押すと、PLC_ctrl0 が true にセットされ、PLC_Q17 と PLC_Q7 の両方に１パルス入力されます。
その結果、カウンター値が１足され、OnParameterUpdate()に１回コールバックされます。結果値が表示されます。

5) "Device control" ボタン。「Gev Device Control」ダイアログを表示させる方法を記述しています。

PLC_ctrl0 を true にセットすると、"Generate 1 pulse"ボタンを押すことと同じ動作を実現できます。

Note: a). デフォルトでは USE_FVAL は定義されていません。
b). カメラの GenICam 項目の定義によっては、すべての PLC が動作するわけではありません。

注)SentechGigE カメラにはコード内で記述されている以下の機能が無い為、以下の機能をコメントアウトした上で使用してください。
（CounterSelector、CounterEventSource、CounterResetSource、CounterResetActivation、CounterValue）

1.13 PvPlcDelayerSample

このサンプルでは、信号を遅延させる方法の概要を説明しているサンプルコードです。このサンプルでは、約３秒信号を遅延させています。

1) Introduction

PvPlcDelayerSample クラスの使い方及び、サンプルの見方:
＊ 約３秒 TTL_OUT[0]への信号を遅延させる為、Timer1 (Pulse Generator 0) と Delayer を構成します。
注)このサンプルには、TTL_OUT[0]信号の出力波形をモニターする為の部分が含まれていますが、SentechGigE カメラにおいては、この部分は動作しません。

2) 前提条件

このサンプルを動作させるには:
＊ Universal か Optimal がインストールされている LAN カードとカメラを LAN ケーブルで接続してください。

3) ファイルの説明

3.1 PvPlcDelayerSample.cpp
Timer1 (Pulse Generator 0) を使用し 23Hz のパルスを出力させます。Delayer は入力信号を遅延させます。これらのクラスが当サンプルでどのように使用しているかは、ソースコードのコメントを参照してください。

NOTE:このサンプルは、Programmable Logic Controller Reference Guide に記載している『Demonstrating the Rescaler』チュートリアルに基づきます。

1.14 PvPlcRescalerSample

このサンプルは Rescaler、およびいくつかの PLC 機能の使用方法を説明したサンプルコードです。

1) PLC_rsl0 は 36Hz の信号から 3.6Hz の信号にスケールを改めます。
2) Timer1 (Pulse generator 0)は 36Hz の信号を作成します。
3) PLC_ctrl0 は rescaler をオン／オフさせます。
4) LUT ルートで rescaler を Q0 (TTL_OUT[0])に出力させます。

注)このサンプルには、TTL_OUT[0]信号の出力波形をモニターする為の部分が含まれていますが、SentechGigE カメラにおいては、この部分は動作しません。

NOTE:このサンプルは、Programmable Logic Controller Reference Guide に記載している『Demonstrating the Rescaler』チュートリアルに基づきます。

1.15 PvRecoverySample

PvDeviceEventSink を使用した自動復旧をサポートしている、画像取込を行う簡単なコマンドラインアプリケーションの作り方を記述しているサンプルコードです。

1) Introduction

PvDevice と PvStream クラスの使い方及び、サンプルの見方:
* PvDeviceFinderWnd クラスを使用したカメラの選択。
* カメラ接続。
* ストリームオープン。
* カメラからの画像取込スタート。
* PvStream クラスを使用した画像受け取り。
* PvDeviceEventSink クラスからリカバリイベント/コールバックの受け取り。
* リカバリイベントの操作。

2) 前提条件

このサンプルを動作させるには:
* Universal か Optimal がインストールされている LAN カードとカメラを LAN ケーブルで接続してください。

3) ファイルの説明

3.1 PvRecoverySample.cpp
画像取込の単純なコマンドラインアプリケーション作成方法を記述しています。これらのクラスが当サンプルでどのように使用しているかは、ソースコードのコメントを参照してください。

注) SentechGigE カメラにはコード内で記述されている DeviceReset 機能はありません。

1.16 PvRGBFilterSample

PvBufferConverter、PvFilterRGB、PvBufferWriter クラスを使用して、画像の変換および保存方法を説明しているサンプルコードです。

1) Introduction

各クラスの使い方及び、サンプルの見方:
* カメラから PC へ画像を送るために必要なステップ。
* PvBufferConverter クラスを使用して、画像データを RGB32Bit に変える方法。
* PvFilterRGB クラスを使用して、RGB フィルタとホワイトバランスを使用する方法。
* PvBufferWriter クラスを使用して、画像をファイル保存する方法。

2) 前提条件

このサンプルを動作させるには:
* Universal か Optimal がインストールされている LAN カードとカメラを LAN ケーブルで接続してください。

3) ファイルの説明

3.1 PvRGBFilterSample.cpp

画像変換する為の PvBufferConverter、PvFilterRGB クラスおよび画像保存する為の PvBufferWriter クラスの使用方法を記述しています。

1.17 PvSimpleUISample

カメラの接続、画像の取得、表示を行っているアプリケーションの作成方法を説明しているサンプルコードです。

1) Introduction

各クラスの使い方及び、サンプルの見方:
* PvDeviceFinderWnd クラスを使用してカメラを選択。
* カメラ接続。
* ストリームオープン。
* カメラへ画像取得命令のスタート。
* PvStream クラスを使用して画像取得。
* PvDisplayWnd クラスを使用し、画像表示。
* PvGenParameterArray クラスでカメラ機能を取得し、PvGenBrowserWnd クラスを使用して閲覧。

2) 前提条件

このサンプルを動作させるには:
* Universal か Optimal がインストールされている LAN カードとカメラを LAN ケーブルで接続してください。

3) ファイルの説明

3.1 PvSimpleUISample.cpp
UI アプリケーションを作成する方法を記述しています。

1.18 PvStreamSample

カメラの接続、画像の取得、表示を行っているアプリケーションの作成方法を説明しているサンプルコードです。

1) Introduction

PvStream クラスの使い方及び、サンプルの見方:
* カメラ接続。
* 画像取得のためのストリーム設定をセットアップ。
* カメラから画像を取得。
* データの表示。
* 画像取得をストップ。

2) 前提条件

このサンプルを動作させるには:
* Universal か Optimal がインストールされている LAN カードとカメラを LAN ケーブルで接続してください。

3) ファイルの説明

3.1 PvStreamSample.cpp
UI アプリケーションを作成する方法を記述しています。

1.19 NetCommandSample

このサンプルは NetCommand アプリケーションのソースコードです。
NetCommand アプリケーションは複数カメラの検出、接続、構成に用いられます。

1) Introduction

このサンプルコードは、StGigE SDK の使用方法を深く理解できます。

2) 前提条件

このサンプルを理解するには下記が必要:
* Visual Studio 2008 (VS9)を使用してください。
* ビルドしてアプリケーションを作成し、動作を良く理解してください。
* C++とマイクロソフトの MFC を良く理解していること。

1.20 StGEViewerSample

主にカメラとの通信をおこなう方法を説明しているサンプルコードです。

1) Introduction

各クラスの使い方及び、サンプルの見方:
* GenICam コマンドを用いたカメラへの設定。
* PvSerialPortIPEngine クラスを使用したカメラへの設定。

2) 前提条件

このサンプルを理解するには:
* Visual Studio 2005 (VS8) C++を使用して作成したサンプルコードです。
* ビルドしてアプリケーションを作成し、動作を良く理解してください。
* C++とマイクロソフトの MFC を良く理解していることが前提です。

1.21 StGEMultiCameraSample

主に複数カメラとの接続、映像表示をおこなう方法を説明しているサンプルコードです。

1) Introduction

各クラスの使い方及び、サンプルの見方:
* カメラ検索、および接続前 IP アドレスの変更。(IPConfiguration.cpp)
* 複数カメラの自動接続。
* 複数カメラの映像表示。

2) 前提条件

このサンプルを理解するには:
* Visual Studio 2005 (VS8) C++を使用して作成したサンプルコードです。
* ビルドしてアプリケーションを作成し、動作を良く理解してください。
* C++とマイクロソフトの MFC を良く理解していることが前提です。

1.22 StGEOpenCVFacesDetectSample

OpenCV の顔検出を使用したサンプルコードです。
OpenCV をダウンロードし、ライブラリを作成して使用してください。

1) Introduction

各クラスの使い方及び、サンプルの見方:
* カメラの接続。
* 映像データの取得。
* OpenCV を用いた顔検出。
* OpenCV を用いた映像表示。

2) 前提条件

このサンプルを理解するには:
* OpenCV2.1.0 を使用して作成したサンプルコードです。
* Visual Studio 2005 (VS8) C++を使用して作成したサンプルコードです。
* ビルドしてアプリケーションを作成し、動作を良く理解してください。
* C++とマイクロソフトの MFC を良く理解していることが前提です。

1.23 StGEOpenCVSharpSample

OpenCV のアンシャープマスクを使用したサンプルコードです。
OpenCV をダウンロードし、ライブラリを作成して使用してください。

1) Introduction

各クラスの使い方及び、サンプルの見方:
* カメラの接続。
* 映像データの取得。
* OpenCV を用いたアンシャープマスク。
* OpenCV を用いた映像表示。

2) 前提条件

このサンプルを理解するには:
* OpenCV2.1.0 を使用して作成したサンプルコードです。
* Visual Studio 2005 (VS8) C++を使用して作成したサンプルコードです。
* ビルドしてアプリケーションを作成し、動作を良く理解してください。
* C++とマイクロソフトの MFC を良く理解していることが前提です。

2. サンプル例

2.1 ソフトウェアトリガー

2.1.1 PLC を用いたソフトウェアトリガー

PLC_ctrl0 から I5 を通って Q9 に信号を入力させ、入力した立ち上がり信号で PulseGenerator0 で作成した信号を pg0_out(Timer1Out)へ出力させます。pg0_out から I4 を通って Q4 に信号を出力させ、Q４からカメラ FPGA にトリガーパルスを出力させます。
※PLC の詳細については、iPORT.Reference.Programmable_Logic_Controller.pdf を参照。

①カメラを接続し、PvDevice の取得、PvStream,PvPipeline の設定を行います。
GEVPlayerSample、PvSimpleUISample、PvStreamSample 等のサンプルプログラムを参考にしてコードを作成します。

②PLC のルート設定を行います。

```
//ソフトトリガー用の PLC ルート設定を行う関数
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;
}

//LookupTable 部分の設定を行う関数
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;

}

//SignalRoutingBlock 部分の設定を行う関数
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;
}
```

③パルスジェネレータの設定を行います。

```
//パルスジェネレータの設定を行う関数
//-------------------------------------------------------------
//durationOfHigh:パルスの High 時間
//durationOfLow:パルスの Low 時間
//bTrgMode    :True:連続     False:単一パルス
//pulseScale:パルスの単位 30nSec 単位の場合は 1 を設定、30uSec 単位の場合は 1000 を設定
//-------------------------------------------------------------
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;
}
```

④カメラをトリガーモードにします。

```
//トリガーモード設定
//bTriggerMode:  TRUE：トリガーモード  FALSE:フリーランモード
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();
}
```

⑤カメラの露光時間モードを設定します。

```
//露光時間モード設定
//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();
}
```

⑥トリガーソースをソフトトリガーに設定

```
//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();
}
```

⑦ソフトトリガーソースを PLC に設定

```
//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();
}
```

⑧トリガーを入力します。

```
//PLC_ctrl0 に 1mSec のパルスを入力
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;
}

//PLC_ctrl の ON、OFF 関数
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 GenICam コマンドを用いたソフトウェアトリガー

①カメラを接続し、PvDevice の取得、PvStream,PvPipeline の設定を行います。
GEVPlayerSample、PvSimpleUISample、PvStreamSample 等のサンプルプログラムを参考にしてコードを作成します。

②カメラをトリガーモードにします。

```
//トリガーモード設定
```

```
//bTriggerMode:  TRUE：トリガーモード  FALSE:フリーランモード
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();
}
```

③カメラの露光時間モードを設定します。

```
//露光時間モード設定（Timedに設定）
//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();
}
```

④カメラの露光時間を設定します。

```
//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();
}
```

⑤トリガーソースをソフトトリガーに設定

```
//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();
}
```

⑥ソフトトリガーソースを Command に設定

```
//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();
}
```

⑦トリガーを入力します。

```
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 NIC 情報、カメラ情報の取得

■PvDeviceFinderWnd クラスを用いずに NIC(Network Interface Card)情報、カメラ情報を取得する方法。
サンプルプログラムではカメラ接続時に主に PvDeviceFinderWnd クラスを用いて、カメラ選択、接続を行っていますが、下記コードを参考にして、カメラの PvDeviceFinderWnd のダイアログを表示させずに、カメラ選択、接続をプログラミングすることもできます。サンプルプログラムの PvDeviceFindingSample も参考にしてください。

```
//各 NIC、各カメラの MAC アドレス、IP アドレス、サブネットマスクを表示します。
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();
}

//カメラを接続する前に、マックアドレスを指定して、カメラの IP アドレス　サブネットマスク、デフォルトゲートウェイを変えることができます
//lIPAddress = "192.168.2.45";       (指定した IP アドレスに変更)
//lSubnetmask= "255.255.255.0";  (指定したサブネットマスクに変更)
//lGateway = "0.0.0.0";                (指定したデフォルトゲートウェイに変更)
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 IP アドレス設定方法

■IP アドレス、サブネットマスク、デフォルトゲートウェイの設定方法
カメラの出荷時の IP アドレスの設定は、動的に IP アドレスを取得する設定になっています。カメラ電源立ち上げ時に固定の IP アドレス、サブネットマスク、デフォルトゲートウェイになるように、設定することができます。
下記コードは、カメラ接続後に現在の IP アドレス、サブネットマスク、デフォルトゲートウェイを保存する方法を記述しています。

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

    PvResult lResult;
    //GevCurrentIPConfigurationPersistentIP
    PvGenBoolean* lPersistentIPBoolean=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;
    lResult = lFlushGatewayParam->SetValue( lGateway );
    if( lResult.IsFailure() ) return FALSE;

    return TRUE;

}
```

2.4 コマンド情報の取得

■カメラのコマンド情報を取得する方法

```
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 シリアル通信

カメラ内部のシリアル通信を PC から行うことができます。
ポートの番号を PvIPEngineSerial0 に設定した場合、カメラ内部の FPGA と通信します。
PvIPEngineSerial1 は SentechGigE カメラでは使用しておりません。

以下のコードは 6BYTE データを送信し、4BYTE のデータを受信するサンプル例です。

```
//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;
  }
```

〒243-0018
神奈川県厚木市中町 4-9-17（原田センタービル７Ｆ）

センサーテクノロジー株式会社

TEL 046(295)7061 FAX 046(295)7066
URL http://www.sentech.co.jp/