IPVDevice

ラインセンサシステム開発キット
IP-View用ライブラリ

Ver. 1.0

IGUNOSS, Inc.
有限会社イグノス

＊目　次＊

1.概要 1
2.リファレンス 2
2-1.プログラムへの組込み手順 2
2-1-1.ライブラリの準備 2
2-1-2.画像の撮影手順 2
ステップ 0　－　プロジェクトの作成 3
ステップ 1　－　デバイスオープン 5
ステップ 2　－　ステージ・カメラ設定の確認 6
ステップ 3　－　画像の撮影 8
ステップ 4　－　画像の表示・ファイル保存 10
ステップ 5　－　デバイスクローズ 10
2-1-3.画像の表示 11
2-1-4.撮影した画像のファイル保存 13
2-1-5.カメラの設定 15
2-1-6.撮影した画像の加工 18
2-1-7.撮影タイミングをソフトで制御する場合 19
2-1-8.各設定項目の範囲 21
2-2.関数一覧 22
2-3.関数詳細 23
2-3-1.デバイスオープン（igOpenDevice） 23
2-3-2.デバイスクローズ（igCloseDevice） 24
2-3-3.撮影画素数の取得（igGetPixcelLine） 25
2-3-4.撮影画素数の変更（igSetPixcelLine） 26
2-3-5.ステージ駆動パラメタの取得（igGetStageParam） 27
2-3-6.ステージ駆動パラメタの変更（igSetStageParam） 28
2-3-7.カメラ設定パラメタの取得（igGetCameraParam） 29
2-3-8.カメラ設定パラメタの変更（igSetCameraParam） 30
2-3-9.画像サイズを縦横 1：1 比率に設定（igSetEquallyImageRate） 31
2-3-10. ステージの原点復帰（igResetStagePos） 32
2-3-11. ステージの現ポジション取得（igGetCurrentPos） 33
2-3-12. ステージの現ポジションリセット（igResetCurrentPos） 34
2-3-13. ステージの移動（igMoveStagePos） 35
2-3-14. 画像の撮影（igShotImage） 36
2-3-15. ラインセンサ画像撮影（igCaptureImage） 37
2-3-16. 画像データポインタ取得（igGetImageAddress） 38
2-3-17. 画像データの描画（igDrawImage） 40
2-3-18. 画像のファイル保存（igSaveImageFile） 41
2-3-19. 画像ファイルの読み込み（igLoadImageFile） 42
2-4.戻り値一覧 43

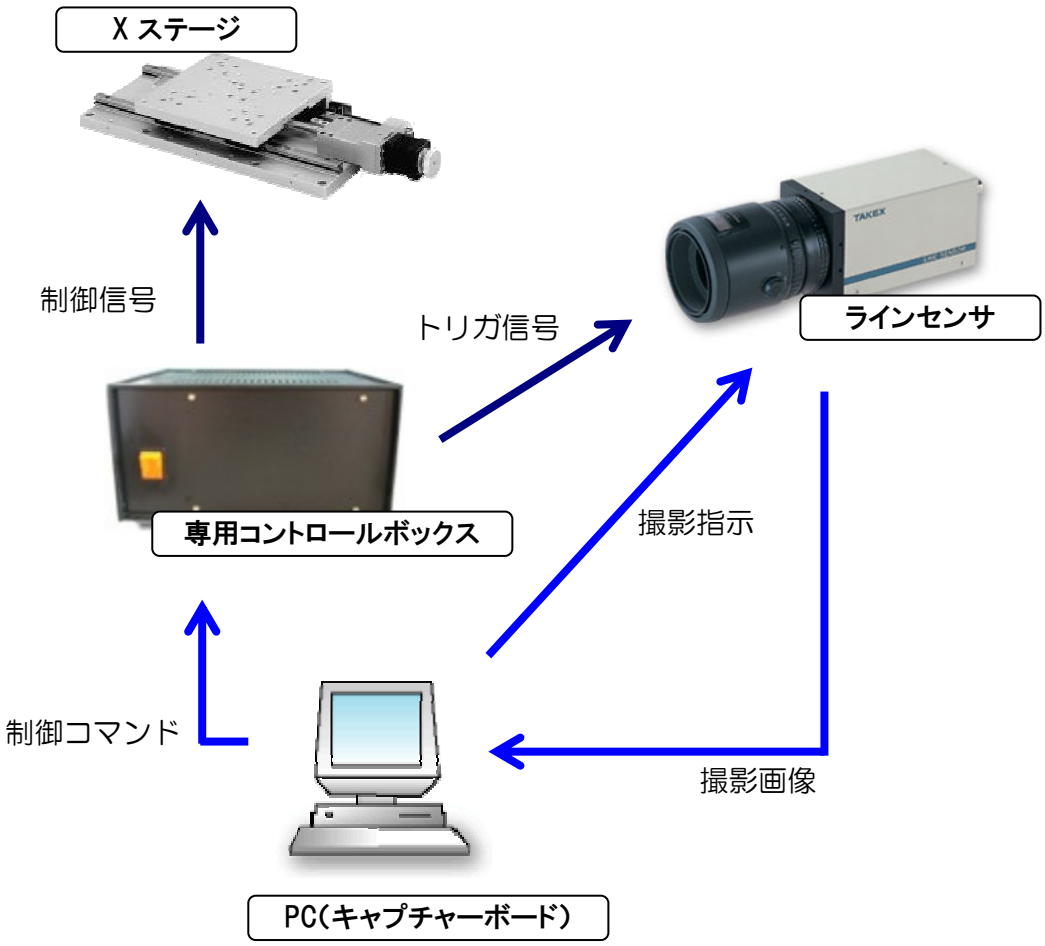
1.概要

本ライブラリは、X ステージ（中央精機製 ALS-220-C2P）の制御を行うと同時に、ラインセンサ（竹中システム機器製 TL-7450UFD）の制御も行い、ラインセンサでの撮影が容易に行えるライブラリです。

開発環境は、Microsoft Visual C++6.0、対応 OS は Windows2000 Professional、および WindowsXP HomeEdition／Professional です。

本ライブラリには、以下の特徴があります。

・デバイスをオープンすると、デフォルト設定で即撮影が可能です。
・撮影した画像は、画面への表示やファイルへの保存が簡単に実行できます。
・メモリ内の画像アドレス（DIB 形式）を取得することで、画像処理が自由に行えます。



2.リファレンス

2-1.プログラムへの組込み手順

ここでは、本ライブラリを呼び出して画像を撮影するサンプルプログラム（IPVSample）の作成手順を説明します。サンプルプログラムでは、画像の撮影だけでなく、ステージのスピードやカメラの設定処理等も行いますので、プログラミングの参考にしてください。

また、サンプルプログラムは初期状態で PC にインストールされていますので、プログラムメニューから起動して、ステージやラインセンサの動作の確認にもご利用いただけます。

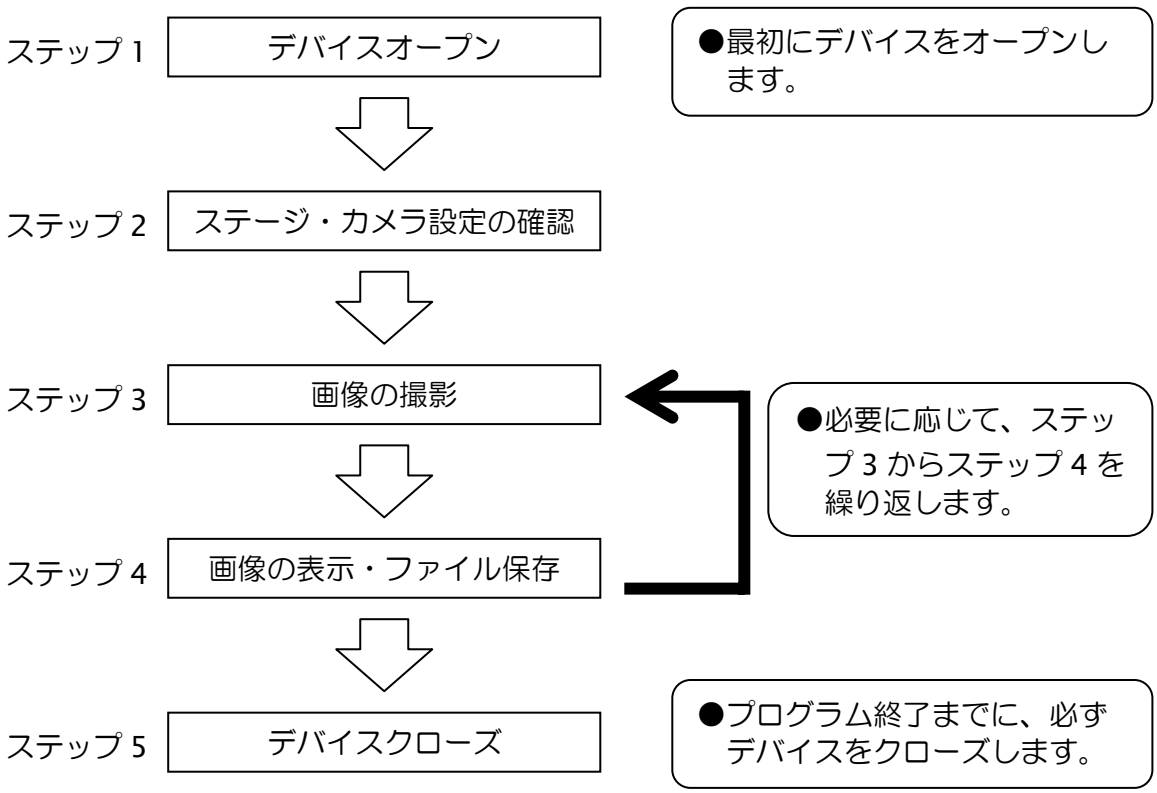
2-1-1.ライブラリの準備

本ライブラリのインストール先（c:￥Program Files￥igunoss￥IP-View）フォルダ内に、LIB ファイル・ヘッダファイル・サンプルプログラム等が入っています。また、DLL ファイルは Windows の System32 フォルダに入っています。

LIB ファイルおよびヘッダファイルは、プロジェクトのビルド時に必要になりますので、プロジェクト作成後にプロジェクトのフォルダにコピーするか、VisualC++6.0 のオプションでパスを設定する等の操作が必要です。

2-1-2.画像の撮影手順

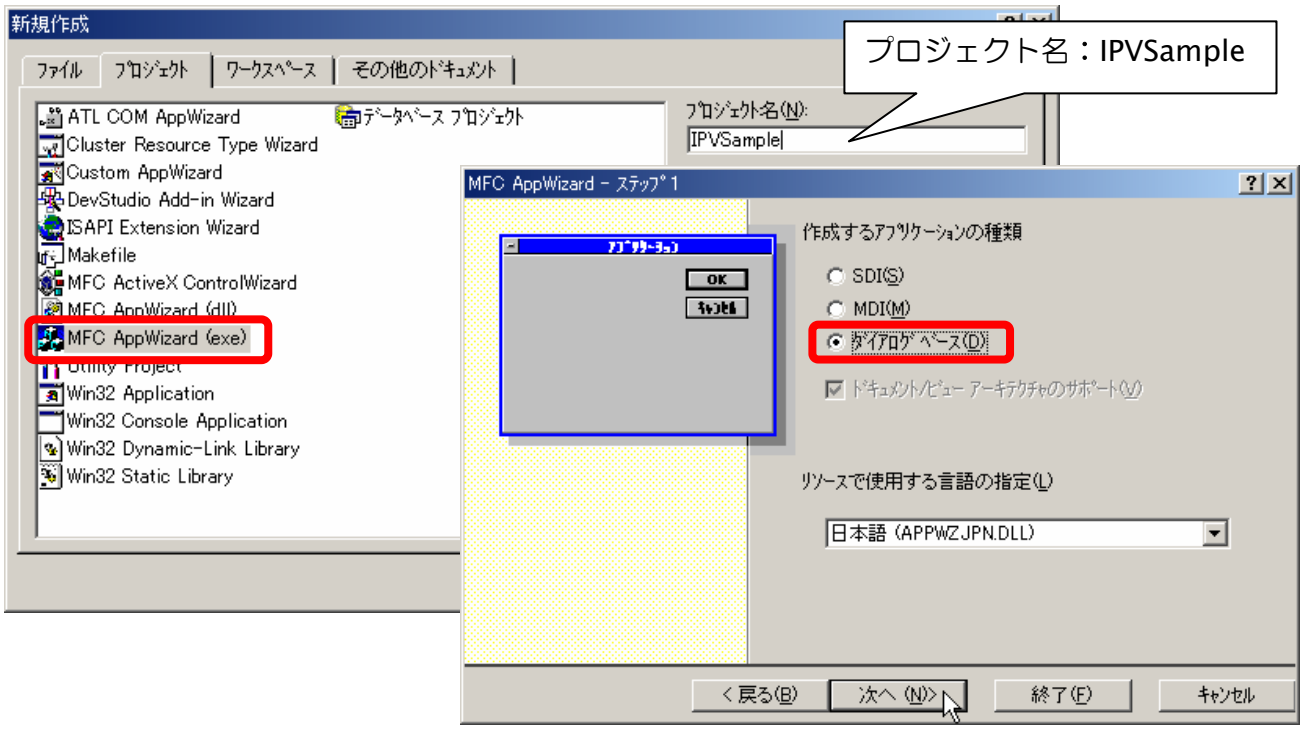
本ライブラリを呼び出して、ラインセンサで画像撮影を行うには、基本的には以下のステップを実行します。



ここからは、VisualC++6.0 のダイアログベースで作成したプロジェクトでの手順を説明します。VisualC++6.0 の操作方法等の詳細については、VisualC++6.0 のヘルプやマニュアルを参照してください。

ステップ 0　－　プロジェクトの作成

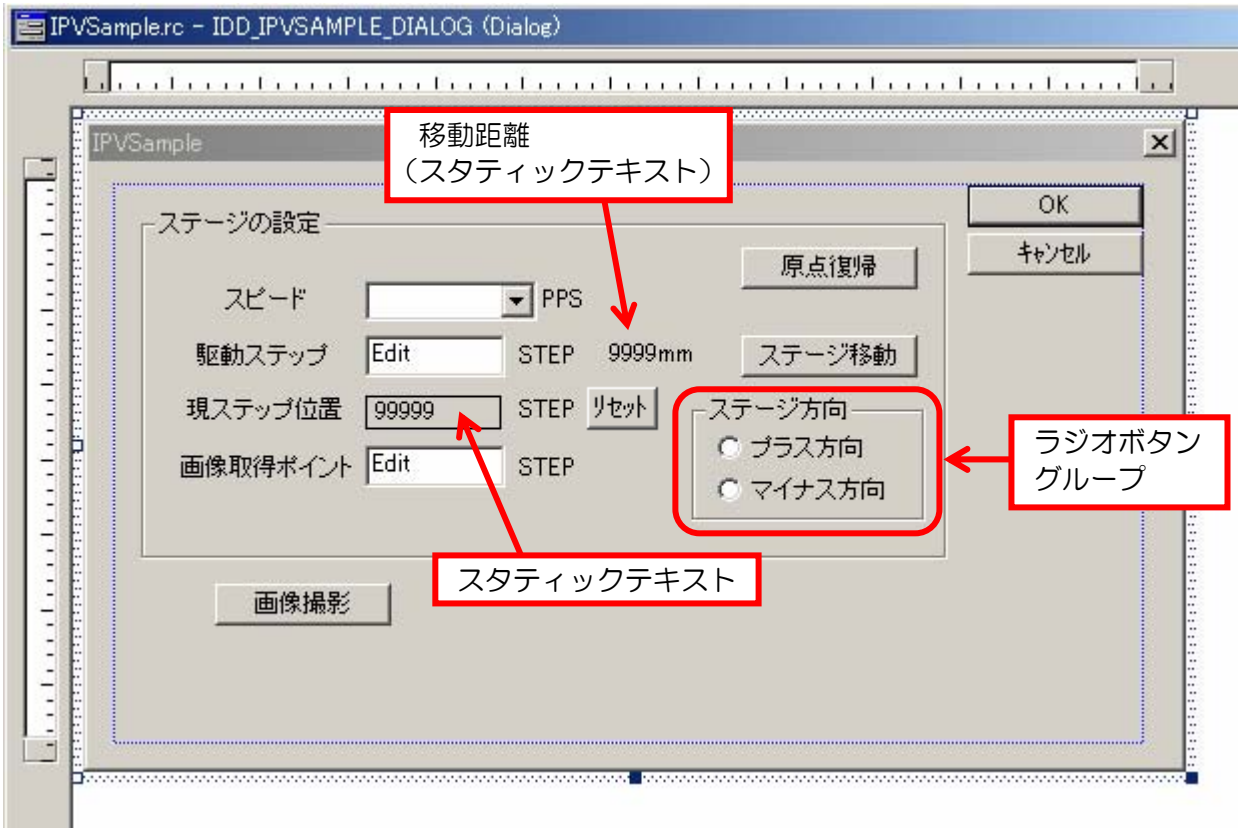
VisualC++6.0 の「ファイル」メニューから「新規作成」を選択し、新規プロジェクト（MFC APP Wizard／ダイアログベース）を作成します。



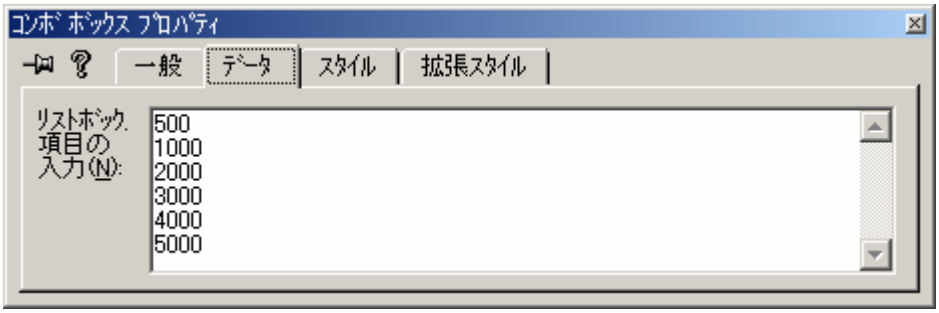
本ライブラリを呼び出すために、LIB ファイル「IPVDevice.lib」およびヘッダファイル「IPVDevice.h」を、作成したプロジェクトのフォルダにコピーしておきます。

メインのダイアログボックス（IDD_IPVSAMPLE_DIALOG）には、以下のコントロールを作成し、必要なメンバ変数を登録します。

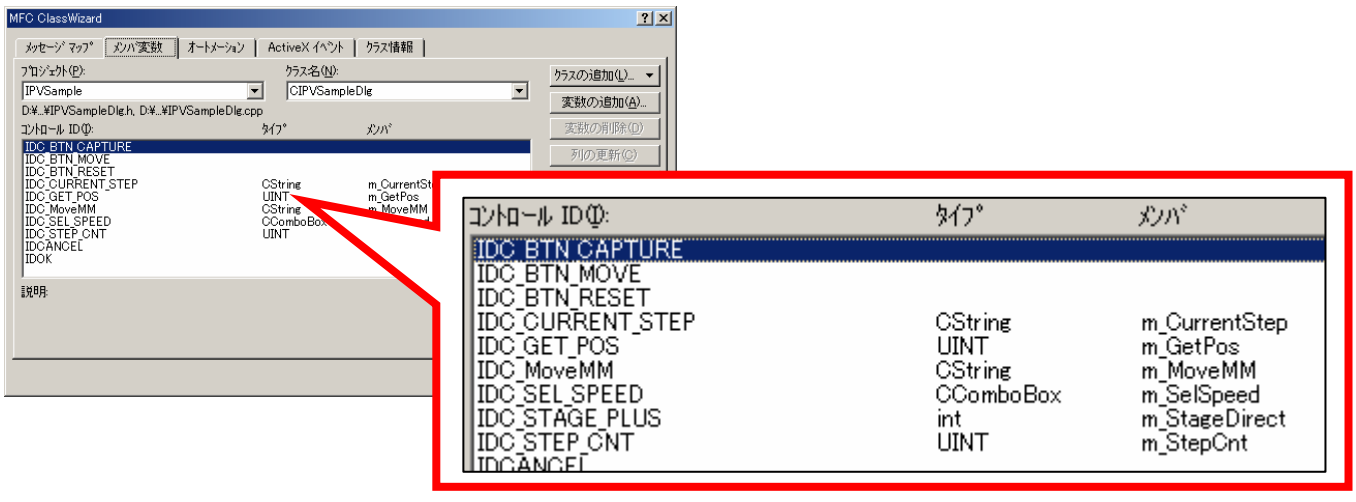
コントロールの ID	コントロールの種類	キャプション	説明
IDC_BTN_RESET	ボタン	原点復帰	
IDC_BTN_MOVE	ボタン	ステージ移動	
IDC_BTN_CAPTURE	ボタン	画像撮影	
IDC_BTN_RESET_STEP	ボタン	リセット	ステップ数リセット
IDC_SEL_SPEED	コンボボックス		ステージのスピード
IDC_STEP_CNT	エディットボックス		駆動ステップ数
IDC_MoveMM	スタティックテキスト		移動距離
IDC_CURRENT_STEP	スタティックテキスト		現在のステップ位置
IDC_STAGE_PLUS	ラジオボタン	プラス方向	プラスにステージ移動
IDC_STAGE_MINUS	ラジオボタン	マイナス方向	マイナスにステージ移動
IDC_STATIC	グループボックス	ステージ方向	ラジオボタングループ
IDC_GET_POS	エディットボックス		画像取得ポイント



スピードを選択するコンボボックスは、プロパティの「データ」タブで、500～5000 の値を入力し、「スタイル」タブで「ソート」のチェックをはずします。



ここではわかりやすいように、現ステップ位置のスタティックテキストに境界線を表示し、それぞれの項目に単位を表示しています。また、ステージ方向のラジオボタンは、タブオーダーとグループを設定して、メンバ変数を登録します。他にも以下のメンバ変数を登録します。



ステップ１　－　デバイスオープン

画像を撮影する前に、まずはデバイスをオープンします。

```
private:
	HANDLE DevHandle;		//デバイスハンドル
```

上記の変数を、クラスの private 変数として定義します。

```
#include "IPVDevice.h"

//  省略  //

BOOL CIPVSampleDlg::OnInitDialog()
{
//  省略  //

	// TODO: 特別な初期化を行う時はこの場所に追加してください。

	ULONG Ret = igOpenDevice(&DevHandle, NULL, NULL, true);
	if(Ret != IG_NO_ERROR){
		AfxMessageBox("デバイスオープンエラー");
	}
```

IPVSampleDlg.cpp を開き、IPVDevice.h をインクルードします。ここでは、CIPVSampleDlg クラスの WM_INITDIALOG メッセージの処理（OnInitDialog）でデバイスオープンします。

デバイスをオープンするには、igOpenDevice 関数を呼び出します。igOpenDevice 関数の定義は以下のとおりです。

```
ULONG   igOpenDevice(
            HANDLE* pigHandle,     //デバイスハンドル格納域アドレス
            LPTSTR igConfigName,   //カメラの設定ファイルパス名
            LPTSTR igPortName,     //COM ポート名
            BOOL igTrigerSignal)   //トリガ信号の使用有無
```

pigHandle には、クラスの private 変数として定義したデバイスハンドルの格納域アドレスを指定します。igConfigName には、カメラ設定ファイルのフルパス名を指定しますが、通常は NULL を指定してください（システム初期設定時のカメラ設定ファイルを参照します）。
igPortName にはステージを接続した COM ポート名を指定します。igPortName に NULL を指定すると、「COM1」を使用します。igTrigerSignal には、コントロールボックスからのトリガ信号を使用するなら TRUE を、使用せずにプログラムから撮影タイミングを指示する場合は FALSE を指定してください。ここではトリガ信号を使用するので、TRUE を指定します。

igOpenDevice 関数の戻り値が IG_NO_ERROR 以外はエラーのため、メッセージを表示しています。本ライブラリの関数の戻り値は、基本的に IG_NO_ERROR は正常終了、それ以外はエラーコードとなっています。また、サンプルプログラムでは詳しいエラー処理は省略していますが、必要に応じてエラー処理を行ってください。各関数でのエラーコードは「2-3.関数詳細」を、エラーコードの一覧は、「2-4.戻り値一覧」を参照してください。

ステップ 2　－　ステージ・カメラ設定の確認

デバイスオープンができたら、次にステージとカメラのデフォルト設定値を取り出します。デフォルト値を取得しなくてもそのまま撮影は可能ですが、ここでは確認（表示）のために取得します。

```
UINT ScanPixel;     //画像のピクセル数
UINT ScanLines;     //画像のライン数
UINT STDIS;         //カメラの STDIS
UINT STENB;         //カメラの STENB
```

上記の変数を、クラスの private 変数として追加します。これらは、カメラの設定ファイルに登録されている値で、メインダイアログボックスには表示せず、変数に格納しておきます。

OnInitDialog で、デバイスオープンに続けて初期設定値を取得します。

```
WORD SelSpeed;      //スピード
//デバイスオープン
ULONG Ret = igOpenDevice(&DevHandle, NULL, NULL, true);
if(Ret != IG_NO_ERROR){
      AfxMessageBox("デバイスオープンエラー");
}else{
      igGetPixcelLine(DevHandle, &ScanPixel, &ScanLines);
      igGetCameraParam(DevHandle, &STDIS, &STENB);
      igGetStageParam(DevHandle, &SelSpeed, &m_StepCnt, &m_GetPos);
}
//設定値を表示
m_SelSpeed.SetCurSel(SelSpeed);
m_MoveMM.Format(" [ %.3fmm ] ", m_StepCnt * IG_MM_PER_STEP);      //移動距離
DispCurrentPos();          //現ポジションの表示
m_StageDirect = 0;         //ステージ方向プラス
UpdateData(false);
```

各設定値は、3 つの関数で取り出します。

撮影する画像のピクセル数およびライン数は、igGetPixcelLine 関数で取り出します。カメラのパラメタ（STDIS および STENB）は、igGetCameraParam 関数で取り出します。そして、ステージのスピード、駆動ステップ数および画像取得ポイントは、igGetStageParam 関数で取り出します。

各関数の定義は以下のとおりです。

```
ULONG   igGetPixcelLine (
        HANDLE igHandle,     //デバイスハンドル
        UINT* pigPixel,      //主走査（ピクセル数）格納域アドレス
        UINT* pigLine)       //副走査（ライン数）格納域アドレス
```

```
ULONG   igGetCameraParam (
        HANDLE igHandle,     //デバイスハンドル
        UINT* pigSTDIS,      //STDIS 格納域アドレス
        UINT* pigSTENB)      //STENB 格納域アドレス
```

```
ULONG   igGetStageParam (
        HANDLE igHandle,            //デバイスハンドル
        WORD* pigStageSpeed,        //ステージのスピード格納域アドレス
        UINT * pigMoveStep,         //駆動ステップ数格納域アドレス
        UINT* pigStartCameraPos)    //画像取得ポイント格納域アドレス
```

各関数の igHandle には、igOpenDevice で取得したハンドルを指定します。続けて、それぞれの設定値を格納する変数のアドレスを指定します。

これ以降はメンバ変数で処理することにし、駆動ステップ数と画像取得ポイントは直接メンバ変数に格納しています。スピードの値のみ一度ローカル変数に格納して、コンボボックスに値をセットしています。

ステージの移動距離は、「移動したステップ数×1 ステップあたりの距離」で計算でき、1 ステップあたりの距離は、「IG_MM_PER_STEP」として IPVDevice.h に定義されています。

現ポジションの位置はステージを移動するたびに再表示するので、以下の private 関数を作成して呼び出します。

```
void CIPVSampleDlg::DispCurrentPos(){
        UINT CPos;
        igGetCurrentPos(DevHandle, &CPos);      //現ポジション取り出し
        m_CurrentStep.Format(" %d", CPos);
        UpdateData(false);
}
```

現ポジションは、igGetCurrentPos 関数で取り出します。igGetCurrentPos 関数の定義は以下のとおりです。

```
ULONG   igGetCurrentPos (
        HANDLE igHandle,    //デバイスハンドル
        UINT* pigCurPos)    //現ポジション格納域アドレス
```

igHandle には、igOpenDevice で取得したハンドルを指定します。pigCurPos には、現ポジションを格納する変数のアドレスを指定します。

この他、「リセット」ボタンのクリックイベントで、現ポジションを 0 にリセットする処理を行います。ステージのポジションは起動時の位置が 0 になるため、ステージを移動後に 0 にリセットできるようにこのボタンを作成してあります。

```
void CIPVSampleDlg::OnBtnResetStep()
{
        UpdateData(true);
        igResetCurrentPos(DevHandle);    //現ポジションリセット
        DispCurrentPos();                //現ポジションの表示
}
```

現ポジションのリセットは、igResetCurrentPos 関数で設定します。igResetCurrentPos 関数の定義は以下のとおりです。igHandle には、igOpenDevice で取得したハンドルを指定します。

```
ULONG   igResetCurrentPos (
        HANDLE igHandle)        //デバイスハンドル
```

ステップ 3　ー　画像の撮影

「画像撮影」ボタンのクリックイベントで撮影の処理を行います。

```
void CIPVSampleDlg::OnBtnCapture()
{
	UpdateData(true);
	if(igSetStageParam(DevHandle, m_SelSpeed.GetCurSel(), m_StepCnt, m_GetPos)
                                                                != IG_NO_ERROR)
		AfxMessageBox("ステージ移動パラメタ設定エラー");
	else if(igCaptureImage(DevHandle, NULL) != IG_NO_ERROR)
		AfxMessageBox("画像取得エラー発生");
	}
	DispCurrentPos();
}
```

ステージの設定値を、igSetStageParam 関数で設定します。igSetStageParam 関数の定義は以下のとおりです。

```
ULONG   igSetStageParam (
        HANDLE igHandle,        //デバイスハンドル
        WORD igStageSpeed,      //ステージのスピード
        UINT igMoveStep,        //駆動ステップ数
        UINT igStartCameraPos)  //画像取得ポイント
```

igHandle には、igOpenDevice で取得したハンドルを指定します。その後に、ステージのスピード、駆動ステップ数、画像取得ポイントを指定します。ステージのスピードには、以下の値（0～5）を指定してください。ここでは、コンボボックスの選択インデックス値が 0～5 になるため、そのまま使用しています。以下の値が IPVDevice.h で定義されています。

定　義	値
IG_SPEED_500	0
IG_SPEED_1000	1
IG_SPEED_2000	2
IG_SPEED_3000	3
IG_SPEED_4000	4
IG_SPEED_5000	5

ステージの設定ができたら、いよいよ画像の撮影です。画像の撮影は igCaptureImage 関数で行います。igCaptureImage 関数の定義は以下のとおりです。

```
ULONG   igCaptureImage (
        HANDLE igHandle,    //デバイスハンドル
        BOOL* pigCancel)    //キャンセルフラグのアドレス
```

igCaptureImage 関数では、ステージの移動、ラインセンサでの撮影から BMP 形式データ作成までを行います。

igHandle には、igOpenDevice 関数で取得したハンドルを指定します。pigCancel には、キャンセルフラグのアドレスを指定します。撮影の途中でこの変数の値が true になると、撮影が中止されます。必要な場合のみ指定してください。NULL を指定するとキャンセルはできません。

画像の撮影に伴って、ステージの移動と原点復帰の処理が必要になるため、それらの処理を記述します。

「ステージ移動」ボタンのクリックイベントに以下の処理を追加します。

```
void CIPVSampleDlg::OnBtnMove()
{
	UpdateData(true);
	int StepCnt = (m_StageDirect == 0 ? (int)m_StepCnt : -1 * (int)m_StepCnt);

	if(igSetStageParam(DevHandle, m_SelSpeed.GetCurSel(), m_StepCnt, m_GetPos) !=
IG_NO_ERROR)
		AfxMessageBox("ステージ移動パラメタ設定エラー");
	else if(igMoveStagePos(DevHandle, StepCnt) != IG_NO_ERROR)
			AfxMessageBox("ステージ移動エラー発生");

	DispCurrentPos();
}
```

ステージの移動は igMoveStagePos 関数で行います。igMoveStagePos 関数の定義は以下のとおりです。

```
ULONG   igMoveStagePos (
        HANDLE igHandle,     //デバイスハンドル
        long igStepCount)    //移動ステップ数
```

igHandle には、igOpenDevice 関数で取得したハンドルを指定します。igStepCount には、移動するステップ数を指定します。マイナス方向に移動する場合は、マイナス値を指定してください。ここでは、ラジオボタンの値が 0 の場合はプラス方向、1 の場合はマイナス方向に移動するので、マイナス方向を指定された場合は、ステップ数をマイナス値に変更しています。

「原点復帰」ボタンのクリックイベントに以下の処理を追加します。

```
void CIPVSampleDlg::OnBtnReset()
{	if(igResetStagePos(DevHandle) != IG_NO_ERROR)
		AfxMessageBox("原点復帰エラー発生");
	DispCurrentPos();
}
```

ステージの原点復帰は igResetStagePos 関数で行います。関数の定義は以下のとおりです。

```
ULONG   igResetStagePos (
      HANDLE igHandle)  //デバイスハンドル
```

igHandle には、igOpenDevice で取得したハンドルを指定します。

ステップ 4　－　画像の表示・ファイル保存

撮影した画像の表示とファイル保存については、次項以降で説明します。

ステップ 5　－　デバイスクローズ

デバイスの使用が終了したら、必ずデバイスのクローズを行ってください。本ライブラリでは、デバイスクローズするまでは、他のプロセスでのデバイスの使用ができません。デバイスクローズせずにアプリケーションを終了した場合、それ以降のデバイス動作は保証しません。

ここでは、CIPVSampleDlg クラスの WM_DESTROY メッセージの処理（OnDestroy）でデバイスのクローズを行います。

```
void CIPVSampleDlg::OnDestroy()
{
	CDialog::OnDestroy();

	igCloseDevice(DevHandle);		//デバイスクローズ
}
```

画像の撮影は igCloseDevice 関数で行います。igCloseDevice 関数の定義は以下のとおりです。

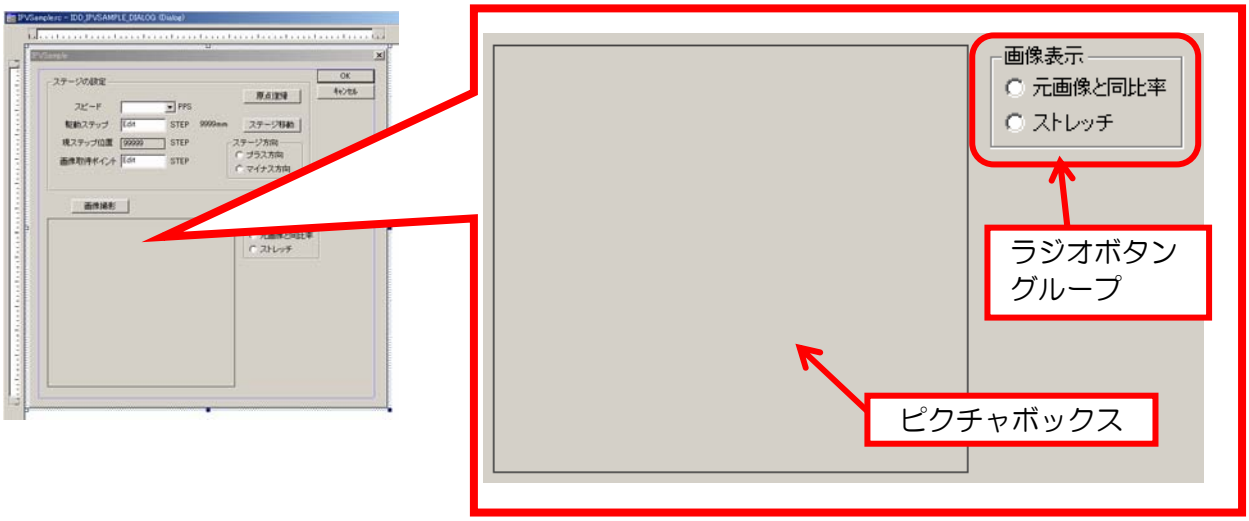
```
ULONG   igCloseDevice(
            HANDLE igHandle)     //デバイスハンドル
```

igHandle には、igOpenDevice で取得したハンドルを指定します。

2-1-3.画像の表示

次に、撮影した画像をピクチャボックスに表示してみましょう。
メインのダイアログ（IDD_IPVSAMPLE_DIALOG）に以下のコントロールを追加します。

コントロールの ID	コントロールの種類	キャプション	説明
IDC_CAP_IMAGE	ピクチャボックス		撮影画像の描画エリア
IDC_DRAW_MODE1	ラジオボタン	元画像と同比率	
IDC_DRAW_MODE2	ラジオボタン	ストレッチ	
IDC_STATIC	グループボックス	画像表示	ラジオボタングループ



ここでは、CIPVSampleDlg クラスの WM_PAINT メッセージの処理（OnPaint）で画像を描画します。

```
	BOOL CaptureFlag;
```

上記の変数を、クラスの private 変数として追加します。これは、描画するときに、撮影画像があるかどうかをチェックするフラグです。画像の撮影前に false をセットして、撮影後に true をセットします。また、OnInitDialog 内で、false をセットしておきます。

OnPaint で、CaptureFlag 変数が true の場合に描画処理を行います。ここでは DrawPicture という private 関数を作成して、それを呼び出すことにします。

```
void CIPVSampleDlg::OnPaint()
{
	if (IsIconic())

//  省略  //

	else
	{
		CDialog::OnPaint();
		DrawPicture();		//ピクチャボックスに描画
	}
}
```

```
//ピクチャボックスに描画
void CIPVSampleDlg::DrawPicture(){
	if(!CaptureFlag)	//撮影画像なし
		return;
	UpdateData(true);
	BeginWaitCursor();		//マウスポインタを砂時計に
	igDrawImage(DevHandle, GetDlgItem(IDC_CAP_IMAGE), m_DrawMode);	//描画
	EndWaitCursor();		//マウスポインタを元に戻す
}
```

ピクチャボックスへの描画は igDrawImage 関数で行います。igDrawImage 関数の定義は以下のとおりです。

```
ULONG   igDrawImage (
        HANDLE igHandle,    //デバイスハンドル
        CWnd* pWnd,         //ピクチャボックスコントロールのハンドル
        int igDrawMode)     //描画モード
```

igHandle には、igOpenDevice で取得したハンドルを指定します。pWnd には、ピクチャーボックスのウィンドウハンドルを指定します。igDrawMode には、描画モードを指定します。以下の値が IPVDevice.h で定義されていますので、ビットごとの OR（|）演算子で組み合わせて指定します。

定義	値	説明
IG_DRAW_SAMERATE	0	元画像と同比率で表示する
IG_DRAW_STRETCH	1	ピクチャボックスのサイズに合わせてストレッチ表示する
IG_DRAW_REFRESH	0	描画前に背景をクリアする
IG_DRAW_NOREFRESH	2	描画前に背景をクリアしない

さらに、他の描画が必要な個所で、同様に DrawPicture 関数を呼び出します。ここでは、画像撮影後や、描画モードのラジオボタンのクリックイベントに処理を追加します。

```
void CIPVSampleDlg::OnBtnCapture()
{
//  省略  //
	else if(igCaptureImage(DevHandle, NULL) != IG_NO_ERROR)
		AfxMessageBox("画像取得エラー発生");
	else{
		CaptureFlag = true;
		DrawPicture();		//ピクチャボックスに描画
	}
//  省略  //
}
```

2-1-4.撮影した画像のファイル保存

ここまでで、画像を撮影して表示するまでの部分を作成しました。次に、撮影した画像をファイルに保存します。保存できるのは、BMP 形式のみです。

メインのダイアログ（IDD_IPVSAMPLE_DIALOG）に以下のコントロールを追加します。

コントロールの ID	コントロールの種類	キャプション	説明
IDC_BTN_SAVE	ボタン	画像保存	

「画像保存」ボタンのクリックイベントに、以下の処理を追加します。

```
void CIPVSampleDlg::OnBtnSave()
{
	CFileDialog sfdlg(false, "bmp", "*.bmp",
		OFN_FILEMUSTEXIST | OFN_OVERWRITEPROMPT | OFN_NOCHANGEDIR,
		"画像データ(*.bmp)¥0*.bmp¥0¥0");
	if(sfdlg.DoModal() != IDOK)
		return;

	if(igSaveImageFile(DevHandle, sfdlg.GetPathName()) != IG_NO_ERROR)
		AfxMessageBox("画像ファイル（BMP）保存エラー発生");
}
```

保存ファイル名を決定する方法として、ここではファイル保存ダイアログボックスを表示していますが、もちろんこの方法でなくてもかまいません。

撮影した画像のファイル保存は、igSaveImageFile 関数で行います。igSaveImageFile 関数の定義は以下のとおりです。

```
ULONG   igSaveImageFile (
          HANDLE igHandle,     //デバイスハンドル
          CString FileName)    //保存する BMP ファイル名
```

igHandle には、igOpenDevice で取得したハンドルを指定します。FileName には、保存する BMP 形式のファイルパス名を指定します。拡張子は「.bmp」でなければなりません。

ピクチャボックスへ表示する画像も、ファイルに保存する画像も、同じ内部メモリで管理しています。一度保存したファイルを読み込むと、同様に内部のメモリに読み込むため、そのままピクチャボックスへの描画も可能です。

ファイルの保存の確認の意味も含めて、画像データを読み込んでみましょう。メインのダイアログ（IDD_IPVSAMPLE_DIALOG）に以下のコントロールを追加します。

コントロールの ID	コントロールの種類	キャプション	説明
IDC_BTN_LOAD	ボタン	画像読込み	

```
        BOOL ReadFlag;
```

上記の変数を、クラスの private 変数として追加します。これは、画像を読み込んだかどうかをチェックするフラグです。画像の撮影前と OnInitDialog 内で false をセットして、画像の読込み後に true をセットします。

「画像読込み」ボタンのクリックイベントに、以下の処理を追加します。

```
void CIPVSampleDlg::OnBtnLoad()
{
        CFileDialog rfdlg(true, "bmp", "*.bmp", OFN_PATHMUSTEXIST | OFN_NOCHANGEDIR,
                "画像データ(*.bmp)¥0*.bmp¥0¥0");
        if(rfdlg.DoModal() != IDOK)
                return;

        if(igLoadImageFile(DevHandle, rfdlg.GetPathName()) != IG_NO_ERROR)
                AfxMessageBox("画像ファイル（BMP）読込みエラー発生");
        else{
                ReadFlag = true;
                DrawPicture();      //ピクチャボックスに描画
        }
}
```

画像の読込みは、igLoadImageFile 関数で行います。igLoadImageFile 関数の定義は以下のとおりです。

```
ULONG   igLoadImageFile (
            HANDLE igHandle,     //デバイスハンドル
            CString FileName)    //読み込む BMP ファイル名
```

igHandle には、igOpenDevice で取得したハンドルを指定します。FileName には、保存する BMP 形式のファイルパス名を指定します。拡張子は「.bmp」でなければなりません。

フラグを追加したので、DrawPicture 関数内で、ReadFlag が true の場合にも描画するように変更しておきます。

これで、一度保存したデータをファイルから読み込んだ後にも、ピクチャボックスに表示されます。

2-1-5.カメラの設定

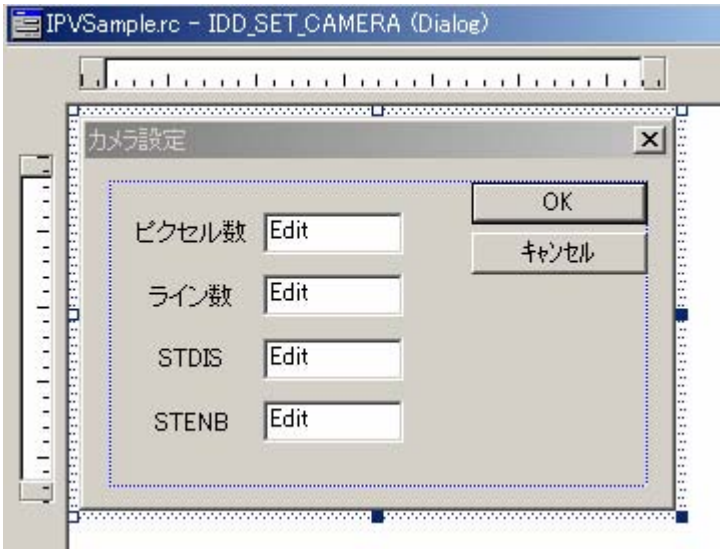
カメラの初期設定値での撮影・描画・保存まではできましたが、画像サイズ等の設定を変更したい場合もあるでしょう。ここでは、設定値を変更するダイアログボックスを作成してカメラの設定値を変更する方法を説明します。

メインのダイアログ（IDD_IPVSAMPLE_DIALOG）に以下のコントロールを追加します。

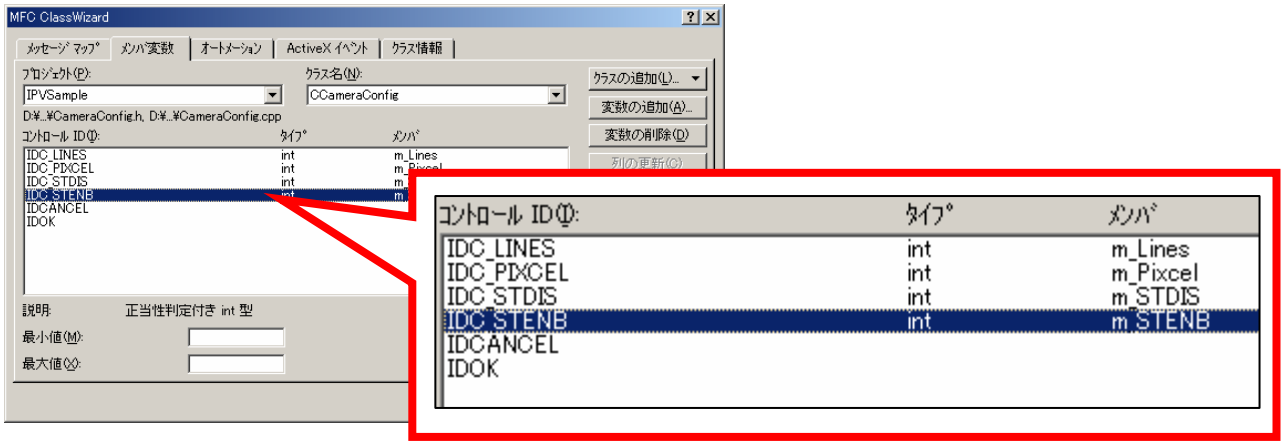
コントロールの ID	コントロールの種類	キャプション	説明
IDC_BTN_SETCONFIG	ボタン	カメラ設定	

新規にダイアログ（IDD_SET_CAMERA）を作成し、以下のコントロールを作成します。

コントロールの ID	コントロールの種類	キャプション	説明
IDC_PIXCEL	エディットボックス	ピクセル数	X 方向のサイズ
IDC_LINES	エディットボックス	ライン数	Y 方向のサイズ
IDC_STDIS	エディットボックス	STDIS	
IDC_STENB	エディットボックス	STENB	



クラス（CCameraConfig）を作成して、それぞれのエディットボックスにメンバ変数を割り当てます。



次に、IPVSampleDlg.cpp で CcameraConfig クラスのヘッダファイル（CameraConfig.h）をインクルードして、「カメラ設定」ボタンのクリックイベントに、以下の処理を追加します。

```
void CIPVSampleDlg::OnBtnSetconfig()
{
	CCameraConfig Dlg;		//カメラ設定ダイアログボックス

	Dlg.m_Pixcel = ScanPixel;
	Dlg.m_Lines = ScanLines;
	Dlg.m_STDIS = STDIS;
	Dlg.m_STENB = STENB;
	if(Dlg.DoModal() == IDOK){
		ScanPixel = Dlg.m_Pixcel;
		ScanLines = Dlg.m_Lines;
		STDIS = Dlg.m_STDIS;
		STENB = Dlg.m_STENB;
		if(igSetPixcelLine(DevHandle, ScanPixel, ScanLines) != IG_NO_ERROR)
			AfxMessageBox("画像サイズの変更エラー");
		else if(igSetCameraParam(DevHandle, STDIS, STENB) != IG_NO_ERROR)
			AfxMessageBox("カメラパラメタの変更エラー");
	}
}
```

private 変数として定義し、OnInitDialog 関数で初期設定値を格納しておいた 4 つの変数を、ダイアログボックスのメンバ変数にコピーします。ダイアログボックスの DoModal 関数を呼び出し、「OK」ボタンがクリックされたら、設定値をメンバ変数からコピーします。

画像サイズの設定は igSetPixcelLine 関数で行います。igSetPixcelLine 関数の定義は以下のとおりです。

```
ULONG	igSetPixcelLine (
		HANDLE igHandle,	//デバイスハンドル
		UINT igPixel,		//主走査（ピクセル数）
		UINT igLine)		//副走査（ライン数）
```

igHandle には、igOpenDevice で取得したハンドルを指定します。igPixel には画像のピクセル数、igLine には撮影するライン数を指定します。

カメラのパラメタ（STDIS および STENB）の設定は igSetCameraParam 関数で行います。igSetCameraParam 関数の定義は以下のとおりです。

```
ULONG	igSetCameraParam (
		HANDLE igHandle,	//デバイスハンドル
		UINT igSTDIS,		//STDIS
		UINT igSTENB)		//STENB
```

igHandle には、igOpenDevice 関数で取得したハンドルを指定します。igSTDIS および igSTENB にカメラのパラメタを指定します。STDIS と STENB は走査周期を表し、一般的には同じ値を設定することで、安定した撮影条件が得られます。

ラインセンサは、ステージを移動しながら走査するため、移動と走査のタイミングがずれると撮影した画像の縦横の比率が 1 対 1 ではなくなってしまいます。本ライブラリでは、設定したステージのスピードに合わせて、1 対 1 の画像が撮影できるように、カメラパラメタ値を調整する関数を用意しています。

メインのダイアログ（IDD_IPVSAMPLE_DIALOG）に以下のボタンを追加します。

IDC_BTN_SET_RATE	ボタン	1 対 1 比率設定

「1 対 1 比率設定」ボタンのクリックイベントに、以下の処理を追加します。

```
void CIPVSampleDlg::OnBtnSetRate()
{
	if(igSetEquallyImageRate(DevHandle) != IG_NO_ERROR)
		AfxMessageBox("カメラパラメタ設定エラー");
	else
		igGetCameraParam(DevHandle, &STDIS, &STENB);
}
```

カメラパラメタ値の調整は igSetEquallyImageRate 関数で行います。igSetEquallyImageRate 関数の定義は以下のとおりです。

```
ULONG   igSetEquallyImageRate (
          HANDLE igHandle)      //デバイスハンドル
```

igHandle に、igOpenDevice 関数で取得したハンドルを指定します。

本システムでの、STDIS・STENB とスピードとの関係は、以下のようになっています。

スピード	STDIS および STENB
5000pps	9250
4000pps	11563
3000pps	15417
2000pps	23125

ただし、2000PPS 以下の値は設定不可能（STDIS および STENB の設定可能範囲外）であるため、1 対 1 の撮影の設定ができるのは、5000pps から 3000pps までとなります。

2-1-6.撮影した画像の加工

igCaptureImage 関数で撮影した画像を加工する場合は、画像データのポインタを取得します。

画像データのポインタは igGetImageAddress 関数で取得します。igGetImageAddress 関数の定義は以下のとおりです。

```
ULONG   igGetImageAddress (
        HANDLE igHandle,      //デバイスハンドル
        LPVOID *pigPtr,       //画像データポインタ格納域アドレス
        int igPtrKind,        //画像データポインタの種別
        int igLineNo)         //画像データのライン番号
```

igHandle に、igOpenDevice 関数で取得したハンドルを指定します。pigPtr には、取得した画像データポインタ格納域のアドレスを指定します。igPtrKind には、取得するポインタの種別を指定します。以下の値が IPVDevice.h で定義されています。

定　義	説　明
IG_BMPFILEHEADER_PTR	BMPFILEHEADER のポインタ
IG_BMPINFOHEADER_PTR	BMPINFOHEADER のポインタ
IG_IMAGE_PTR	BMP 画像データへのポインタ

igLineNo には、画像データ上側からのライン番号（0～）を指定します。メモリ内の画像データは、一般的な DIB 形式であるため、画像の左下のデータから格納されています。画像データのメモリ内での先頭位置アドレスを取得する場合には、IG_BMP_START（IPVDevice.h で定義されています）を指定してください。

取得したメモリは、本ライブラリ内で管理しています。メモリサイズの変更等が必要な場合は、別に領域を用意して、画像データをコピーしてから処理するようにしてください。

2-1-7.撮影タイミングをソフトで制御する場合

ここまで説明した、igCaptureImage 関数を呼び出して撮影する方法は、ステージ制御も同時に行われるため、撮影が容易に実行できます。しかし、ラインセンサの特長を生かして連続撮影する場合など、撮影するタイミングをソフトから制御する必要が発生するかもしれません。その場合は、トリガ信号を発生しない設定でデバイスをオープンして（igOpenDevice 関数の第 4 引数に false を指定する）、撮影は igShotImage 関数を呼び出します。igShotImage 関数の定義は以下のとおりです。

```
ULONG   igShotImage (
        HANDLE igHandle,    //デバイスハンドル
        BOOL* pigCancel,    //キャンセルフラグのアドレス
        LPVOID pigPtr)      //撮影データ格納域アドレス
```

igShotImage 関数では、撮影のみ行います。ステージ制御は行いませんので、必要であれば igMoveStagePos 関数でステージ制御をしてください。また、pigPtr には、撮影した画像データを格納する領域のアドレスを指定します。事前に撮影する画像サイズ分のメモリを確保して、そのアドレスを指定してください。

以下に、撮影タイミングをソフトで制御するサンプルプログラム（SoftTrigSample）から、連続撮影する部分について解説します。

```
LPVOID plmgBuf = new char[ScanPixel * ScanLines];    //メモリ確保
for(Cnt = 0; ; Cnt++){
        //メッセージチェック
        while(::PeekMessage(&msg,NULL,0,0,PM_REMOVE)){
                ::TranslateMessage(&msg);
                ::DispatchMessage(&msg);
        }
        if(!CaptureFlag)    //終了
                break;
        //撮影のみ実行
        Ret = igShotImage(DevHandle, NULL, plmgBuf);
        if(Ret != IG_NO_ERROR){
                WStr.Format("撮影エラー(%d)", Cnt + 1);
                AfxMessageBox(WStr);
                break;
        }
        //撮影画像をメモリに格納
        for(Cnt2 = 0; Cnt2 < (MAX_BUFFER_LINE - 1); Cnt2++)
                memcpy(plmageMemory + (ScanPixel * (Cnt2 + 1)),
                        plmageMemory + (ScanPixel * Cnt2), ScanPixel);
        memcpy(plmageMemory, plmgBuf, ScanPixel);
        //描画
        DrawBMPImage();
}
delete[] plmgBuf;   //メモリ解放
```

撮影中フラグ（CaptureFlag）が true の間撮影を繰り返します。撮影した画像の先頭 1 ラインを、画像データ域（pImageMemory）にコピーします。画像データ域の最大ライン数は MAX_BUFFER_LINE で定義してあり、そのライン数分を 1 ラインずつずらしてコピーし、一番古いデータを捨てています。これを撮影が停止されるまで繰り返します。

また、撮影した画像データは、そのままでは表示や保存ができません。本ライブラリでは、撮影した画像データから BMP 形式のデータを作成する関数も用意しています。ここでは DrawBMPImage という private 関数を作成して、それを呼び出すことにします。

```
void CSoftTrigSampleDlg::DrawBMPImage(){
        BeginWaitCursor();
        igMakeBMPData(DevHandle, pImageMemory, ScanPixel,
                                                ScanLines * MAX_BUFFER_LINE);
        igDrawImage(DevHandle, GetDlgItem(IDC_IMAGE_BOX),
                                    IG_DRAW_SAMERATE | IG_DRAW_NOREFRESH);
        EndWaitCursor();
}
```

撮影した画像データからの BMP 形式のデータ作成は、igMakeBMPData 関数で行います。igMakeBMPData 関数の定義は以下のとおりです。

```
ULONG   igMakeBMPData(
        HANDLE igHandle,     //デバイスハンドル
        LPVOID pigImgData,   //撮影した画像データ格納域アドレス
        UINT igXpixel,       //画像データピクセル数（主走査）
        UINT igYLines)       //画像データライン数（副走査）
```

igHandle に、igOpenDevice 関数で取得したハンドルを指定します。pigImgData には撮影した画像データ格納域アドレスを指定します。ここでは（ScanPixel×MAX_BUFFER_LINE） のサイズで確保してあるデータ域アドレスを指定しますが、pigImgData に NULL を指定すると、最後に撮影した画像を参照し、画像サイズも撮影したサイズになります。pigImgData を指定する場合は、igXpixel にピクセル数、igYLines にライン数も指定してください。

igMakeBMPData 関数で BMP 形式のデータを作成すると、その後 igDrawImage 関数を呼び出してピクチャボックスに描画できます。ピクチャボックスへの描画手順は、2-1-3 を参照してください。

また、igSaveImageFile 関数によりファイルの保存もできます。ファイル保存手順は 2-1-4 を参照してください。

●撮影タイミングをソフトで制御するサンプルプログラム（SoftTrigSample）では、ライブラリの使用手順の説明に重点を置くために、それ以外の部分は極力簡単な記述にしています。そのため一般的なプログラミング方法とは異なる部分があるかもしれません。VC++での一般的なプログラミング方法については、オンラインヘルプや市販の書籍等を参照してください。

2-1-8.各設定項目の範囲

本ライブラリで設定する各項目の指定可能範囲は、以下のとおりです。この範囲を超えた値を指定するとエラーになります。サンプルプログラムでは、メインダイアログの処理ボタンクリック時に範囲チェックを行い、カメラ設定ダイアログボックスの各コントロールのメンバ変数に対しては入力範囲を指定しています。

設定項目	最小値	最大値
画像ピクセル数（X 方向のサイズ）	1	7450
撮影ライン数（Y 方向のサイズ）	1	16200
STDIS	4000	16200
STENB	4000	16200
ステージ移動ステップ数(＊)	-50000	50000

＊：画像撮影時は、1 以上（プラス値）を指定しなければなりません。また、0 は指定できません。

2-2.関数一覧

No.	関　数　名	機　　能	参照
1	igOpenDevice	デバイスをオープンし、ステージおよびラインセンサの初期化を行います。	P.5 P.23
2	igCloseDevice	デバイスをクローズし、各リソースを解放します。	P.10 P.24
3	igGetPixcelLine	ラインセンサの、現在設定されている撮影画素数（主走査/副走査）を取得します。	P.6 P.25
4	igSetPixcelLine	ラインセンサの撮影画素数（主走査/副走査）を変更します。カメラの設定ファイルの内容を更新します。	P.16 P.26
5	igGetStageParam	ステージの現在の設定値（ステージスピード/駆動ステップ数/画像取得ポイント）を取得します。	P.6, 7 P.27
6	igSetStageParam	ステージの設定値（ステージスピード/駆動ステップ数/画像取得ポイント）を変更します。	P.8 P.28
7	igGetCameraParam	カメラ設定パラメタ（STDIS・STEMB）を取得します。	P.6 P.29
8	igSetCameraParam	カメラ設定パラメタ（STDIS・STEMB）を変更します。カメラの設定ファイルの内容を更新します。	P.16 P.30
9	igSetEquallyImageRate	撮影画像の縦横の比率が 1：1 になるように、カメラ設定パラメタを調整します。	P.17 P.31
10	igResetStagePos	ステージの原点復帰を行います。	P.9 P.32
11	igGetCurrentPos	起動時の位置を 0 とした、ステージの現ポジションを取得します。	P.7 P.33
12	igResetCurrentPos	ステージの現ポジションをリセットします。	P.7 P.34
13	igMoveStagePos	ステージのポジションを移動します。	P.9 P.35
14	igShotImage	カメラの設定により、画像の撮影のみ行います。	P.19 P.36
15	igCaptureImage	ステージとカメラの設定により、ステージの移動および画像の撮影を行い、BMP 形式データを作成します。	P.8 P.37
16	igGetImageAddress	撮影した画像のメモリアドレスを取得します。	P.18 P.38
17	igMakeBMPData	撮影した画像から、BMP 形式のデータを作成します。	P.20 P.39
18	igDrawImage	メモリ上の画像データをピクチャボックスに描画します。	P.12, 20 P.40
19	igSaveImageFile	メモリ上の画像データを BMP 形式でファイルに保存します。	P.13 P.41
20	igLoadImageFile	BMP 形式のファイルから画像データをメモリ上に読み込みます。	P.14 P.42

2-3.関数詳細

2-3-1.デバイスオープン（igOpenDevice）

関数名	igOpenDevice
機　能	デバイスをオープンし、ステージおよびラインセンサの初期化を行います。
書　式	ULONG　igOpenDevice(HANDLE* pigHandle,　//デバイスハンドル格納域アドレス LPTSTR ConfigName,　//カメラの設定ファイル名 LPTSTR PortName,　//COM ポート名 BOOL TrigerSignal)　//トリガ信号の使用有無
引　数	pigHandle　本ライブラリで使用するデバイスハンドルの格納バッファアドレスを指定します。 ConfigName　カメラの設定ファイル名を、フルパスで指定します。NULL を指定するとシステム初期設定時のカメラ設定ファイルを使用します。通常は NULL を指定してください。 PortName　ステージを接続している COM ポートの名称を指定します。NULL を指定すると、"COM1"を使用します。 TrigerSignal　コントロールボックスからのトリガ信号を使用するなら TRUE を、使用せずにプログラムから撮影タイミングを指示する場合は FALSE を指定してください。ステージと連動させて撮影する場合は、TRUE を指定してください。
戻り値	正常にオープン・初期化を行った場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_ALREADY_USED　他のプロセスがデバイスを使用中 IGERR_CONFIG_FILE　カメラの設定ファイル名の指定が正しくない IGERR_ALLOC_MEM　メモリが確保できない IGERR_IPM_DEVICE　デバイスの未接続やリソースエラー IGERR_IPM_IPCONTROL　デバイスとの通信エラー IGERR_IPM_OPENDEVICE　デバイスドライバがオープンできない IGERR_IPM_INITDEVICE　デバイスの初期化エラー IGERR_OPEN_COMPORT　COM ポートがオープンできない IGERR_NOT_SUPPORT　サポートしていない環境で実行している IGERR_SYSTEM　その他システムエラー発生
備　考	igOpenDevice を呼び出すと、ステージとカメラの初期設定値が設定されます。設定値は、igGetPixcelLine、igGetCameraParam、igGetStageParam を呼び出して確認してください。 ＊使用例参照ページ　P.5

2-3-2.デバイスクローズ（igCloseDevice）

関数名	igCloseDevice
機　能	デバイスをクローズし、リソースを解放します。
書　式	ULONG　igCloseDevice(　　HANDLE igHandle)　　//デバイスハンドル
引　数	igHandle　　igOpenDevice で取得したデバイスハンドルを指定します。
戻り値	正常にデバイスの終了処理を行った場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　　デバイスの未オープン IGERR_INVALID_HANDLE　無効なデバイスハンドル IGERR_IPM_IPCONTROL　デバイスとの通信エラー IGERR_NOT_SUPPORT　　サポートしていない環境で実行している IGERR_SYSTEM　　その他システムエラー発生
備　考	プログラム終了までに、必ず igCloseDevice を呼び出してください。呼び出さずに終了すると、他のプロセスでデバイスが使用できなくなります。 ＊使用例参照ページ　P.10

2-3-3.撮影画素数の取得（igGetPixcelLine）

関数名	igGetPixcelLine
機　能	ラインセンサの撮影画素数を取得します。
書　式	ULONG　igGetPixcelLine (HANDLE igHandle,　　//デバイスハンドル UINT* pigPixel,　　//主走査（ピクセル数）格納域アドレス UINT* pigLine)　　//副走査（ライン数）格納域アドレス
引　数	igHandle　　igOpenDevice で取得したデバイスハンドルを指定します。 pigPixel　　ラインセンサの主走査（ピクセル数）を格納するバッファアドレスを指定します。 pigLine　　ラインセンサの副走査（ライン数）を格納するバッファアドレスを指定します。
戻り値	正常に撮影画素数を取得した場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　　デバイスの未オープン IGERR_INVALID_HANDLE　無効なデバイスハンドル
備　考	カメラ設定ファイルに登録されている設定値を取得します。 ＊使用例参照ページ　P.6

2-3-4.撮影画素数の変更（igSetPixcelLine）

関数名	igSetPixcelLine
機　能	ラインセンサの撮影画素数を変更します。
書　式	ULONG　igSetPixcelLine (HANDLE igHandle,　//デバイスハンドル UINT igPixel,　//主走査（ピクセル数） UINT igLine)　//副走査（ライン数）
引　数	igHandle　igOpenDevice で取得したデバイスハンドルを指定します。 igPixel　ラインセンサに設定する主走査（ピクセル数）を指定します。 igLine　ラインセンサに設定する副走査（ライン数）を指定します。
戻り値	正常に撮影画素数を変更した場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　デバイスの未オープン IGERR_INVALID_HANDLE　無効なデバイスハンドル IGERR_PARAM_PIXCEL　ピクセル数が有効範囲外 IGERR_PARAM_LINE　ライン数が有効範囲外 IGERR_ALLOC_MEM　メモリが確保できない IGERR_FREE_MEM　メモリの解放エラー IGERR_CONFIG_FILE　カメラの設定ファイルのアクセスエラー IGERR_IPM_DEVICE　デバイスエラー（リソースエラー等） IGERR_IPM_IPCONTROL　デバイスとの通信エラー IGERR_IPM_MEMORYTBL　メモリテーブルエラー IGERR_NOT_SUPPORT　サポートしていない環境で実行している IGERR_SYSTEM　その他システムエラー発生
備　考	撮影画像のピクセル数を変更すると、カメラの設定ファイルを変更し、画像格納メモリを一度解放して、新たに確保します。 ピクセル数の有効範囲は 1～7450、ライン数の有効範囲は 1～16200 です。 ＊使用例参照ページ　P.16

2-3-5.ステージ駆動パラメタの取得（igGetStageParam）

関数名	igGetStageParam
機　能	ステージ駆動パラメタを取得します。
書　式	ULONG　igGetStageParam (HANDLE igHandle,　　//デバイスハンドル WORD* pigStageSpeed,　　//ステージのスピード格納域アドレス UINT * pigMoveStep,　　//駆動ステップ数格納域アドレス UINT* pigStartCameraPos)　//画像取得ポイント格納域アドレス
引　数	igHandle　igOpenDevice で取得したデバイスハンドルを指定します。 pigStageSpeed　ステージのスピード（0～5）を格納するバッファアドレスを指定します。備考を参照してください。 pigMoveStep　駆動ステップ数を格納するバッファアドレスを指定します。 pigStartCameraPos　画像取得ポイントを格納するバッファアドレスを指定します。
戻り値	正常にステージ駆動パラメタを取得した場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　デバイスの未オープン IGERR_INVALID_HANDLE　無効なデバイスハンドル
備　考	ステージのスピードには、以下の値を使用します。 定　義 / 値 / 説　明 IG_SPEED_500 / 0 / 500PPS の場合 IG_SPEED_1000 / 1 / 1000PPS の場合 IG_SPEED_2000 / 2 / 2000PPS の場合 IG_SPEED_3000 / 3 / 3000PPS の場合 IG_SPEED_4000 / 4 / 4000PPS の場合 IG_SPEED_5000 / 5 / 5000PPS の場合 ＊使用例参照ページ　P.6～P.7

2-3-6.ステージ駆動パラメタの変更（igSetStageParam）

関数名	igSetStageParam
機　能	画像撮影時のステージ駆動パラメタを変更します。
書　式	ULONG　igSetStageParam (HANDLE igHandle,　　//デバイスハンドル WORD igStageSpeed,　　//ステージのスピード UINT igMoveStep,　　//駆動ステップ数 UINT igStartCameraPos)　　//画像取得ポイント
引　数	igHandle　　igOpenDevice で取得したデバイスハンドルを指定します。 igStageSpeed　　ステージのスピードを指定します。スピードの設定については、**前項**を参照してください。 igMoveStep　　画像撮影時の駆動ステップ数を指定します。0 以下の値は指定できません。 igStartCameraPos　　画像取得ポイントを指定します。0 以下の値は指定できません。
戻り値	正常にステージ駆動パラメタを変更した場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　　デバイスの未オープン IGERR_INVALID_HANDLE　　無効なデバイスハンドル IGERR_PARAM_SPEED　　ステージのスピードが有効範囲外 IGERR_PARAM_STEP　　駆動ステップが有効範囲外、または、スピードの設定が 1000PPS 以上で駆動ステップが 1000 ステップ未満 IGERR_PARAM_CAPTPOS　　画像取得ポイントが有効範囲外 IGERR_IO_COMPORT　　COM ポート通信エラー
備　考	変更する必要がない項目は、igGetStageParam で取得した設定値をそのまま指定してください。 駆動ステップ数と画像取得ポイントの有効範囲は 1～50000 です。 ＊使用例参照ページ　P.8

2-3-7.カメラ設定パラメタの取得（igGetCameraParam）

関数名	igGetCameraParam
機　能	カメラ設定パラメタを取得します。
書　式	ULONG　igGetCameraParam (HANDLE igHandle,　　//デバイスハンドル UINT* pigSTDIS,　　//STDIS 格納域アドレス UINT* pigSTENB)　　//STENB 格納域アドレス
引　数	igHandle　　igOpenDevice で取得したデバイスハンドルを指定します。 pigSTDIS　　STDIS を格納するバッファアドレスを指定します。 pigSTENB　　STENB を格納するバッファアドレスを指定します。
戻り値	正常にカメラ設定パラメタを取得した場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　　デバイスの未オープン IGERR_INVALID_HANDLE　無効なデバイスハンドル
備　考	カメラ設定ファイルに登録されている設定値を取得します。 ＊使用例参照ページ　P.6

2-3-8.カメラ設定パラメタの変更（igSetCameraParam）

関数名	igSetCameraParam
機　能	カメラ設定パラメタを変更します。
書　式	ULONG　igSetCameraParam (HANDLE igHandle,　//デバイスハンドル UINT igSTDIS,　//STDIS UINT igSTENB)　//STENB
引　数	igHandle　igOpenDevice で取得したデバイスハンドルを指定します。 igSTDIS　カメラに設定する STDIS を指定します。 igSTENB　カメラに設定する STENB を指定します。
戻り値	正常にカメラ設定パラメタを変更した場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　デバイスの未オープン IGERR_INVALID_HANDLE　無効なデバイスハンドル IGERR_PARAM_STDIS　STDIS が有効範囲外 IGERR_PARAM_STENB　STENB が有効範囲外 IGERR_CONFIG_FILE　カメラの設定ファイルのアクセスエラー IGERR_IPM_IPCONTROL　デバイスとの通信エラー IGERR_SYSTEM　その他システムエラー発生
備　考	設定を変更すると、カメラ設定ファイルを更新します。また、STDIS および STENB の有効範囲は 4000～16200 です。 ＊使用例参照ページ　P.16

2-3-9.画像サイズを縦横 1：1 比率に設定（igSetEquallyImageRate）

関数名	igSetEquallyImageRate
機　能	撮影画像の縦横の比率が 1 対 1 になるように、カメラパラメタを変更します。
書　式	ULONG　igSetEquallyImageRate (HANDLE igHandle)　　//デバイスハンドル
引　数	igHandle　　igOpenDevice で取得したデバイスハンドルを指定します。
戻り値	正常にカメラ設定パラメタを変更した場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　　デバイスの未オープン IGERR_INVALID_HANDLE　無効なデバイスハンドル IGERR_CONFIG_FILE　　カメラの設定ファイルのアクセスエラー IGERR_IPM_IPCONTROL　デバイスとの通信エラー IGERR_SYSTEM　　その他システムエラー発生
備　考	この縦横比率 1:1 設定は、システム標準レンズ最大倍率（×1）・標準カメラ（4.7μm／pixel）が前提となります。レンズの倍率を変えたり、標準レンズ・標準カメラ以外を使用して撮影した画像の比率は保証しません。その場合は、igSetCameraParam 関数を呼び出して、カメラ設定パラメタを変更してください。 ＊使用例参照ページ　P.17

2-3-10. ステージの原点復帰（igResetStagePos）

関数名	igResetStagePos
機　能	ステージの原点復帰（原点位置に移動）を行います。
書　式	ULONG　igResetStagePos (HANDLE igHandle)　　//デバイスハンドル
引　数	igHandle　　igOpenDevice で取得したデバイスハンドルを指定します。
戻り値	正常に原点復帰を行った場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　　デバイスの未オープン IGERR_INVALID_HANDLE　無効なデバイスハンドル IGERR_IO_COMPORT　　COM ポート通信エラー
備　考	原点位置にステージを移動するとともに、ステージの現ポジションをゼロにリセットします。原点復帰は、画像取得時のパラメタとして設定したステージスピードにかかわらず、最速スピードでステージ移動します。 ＊使用例参照ページ　P.9

2-3-11. ステージの現ポジション取得（igGetCurrentPos）

関数名	igGetCurrentPos
機　能	ステージの現ポジション（ステップ）を取得します。
書　式	ULONG　igGetCurrentPos (HANDLE igHandle,　　//デバイスハンドル UINT* pigCurPos)　　//現ポジション格納域アドレス
引　数	igHandle　　igOpenDevice で取得したデバイスハンドルを指定します。 pigCurPos　　ステージの現ポジションを格納するバッファアドレスを指定します。
戻り値	正常に現ポジションを取得した場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　　デバイスの未オープン IGERR_INVALID_HANDLE　無効なデバイスハンドル
備　考	ステージの現ポジション値は、プログラム起動時のステージの位置を 0 とし、そこから移動したステップ数を示しています。原点復帰を行うとゼロにリセットします。その他 igResetCurrentPos を呼び出してゼロにリセットできます。 ＊使用例参照ページ　P.7

2-3-12. ステージの現ポジションリセット（igResetCurrentPos）

関数名	igResetCurrentPos
機　能	ステージの現ポジション値をゼロにリセットします。
書　式	ULONG　igResetCurrentPos (HANDLE igHandle)　　//デバイスハンドル
引　数	igHandle　　igOpenDevice で取得したデバイスハンドルを指定します。
戻り値	正常に現ポジションをリセットした場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　　デバイスの未オープン IGERR_INVALID_HANDLE　無効なデバイスハンドル
備　考	ステージの現ポジション値は、プログラム起動時のステージの位置を 0 とし、そこから移動したステップ数を示しています。ステージを移動した後に、ポジション値を 0 にリセットする場合にこの関数を呼び出します。 ＊使用例参照ページ　P.7

2-3-13. ステージの移動（igMoveStagePos）

関数名	igMoveStagePos
機　能	ステージを移動します。
書　式	ULONG　igMoveStagePos (HANDLE igHandle,　　//デバイスハンドル long igStepCount)　　//移動ステップ数
引　数	igHandle　　igOpenDevice で取得したデバイスハンドルを指定します。 igStepCount　　ステージを移動するステップ数を指定します。マイナス方向に移動する場合は、マイナス値を指定します。
戻り値	正常に原点復帰を行った場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　　デバイスの未オープン IGERR_INVALID_HANDLE　無効なデバイスハンドル IGERR_PARAM_STEP　　移動ステップ数が有効範囲外 IGERR_IO_COMPORT　　COM ポート通信エラー
備　考	ステージを移動すると現ポジションを更新します。ここで指定したステップ数は、画像取得時のパラメタにある駆動ステップ数には影響しません。 移動ステップ数の有効範囲は-50000～50000 です。ただし、実際のステージの位置によっては、指定したステップ数分移動するとは限りません。 ＊使用例参照ページ　P.9

2-3-14. 画像の撮影（igShotImage）

関数名	igShotImage
機　能	ラインセンサで撮影を行います。
書　式	ULONG　igShotImage (HANDLE igHandle,　//デバイスハンドル BOOL* pSetCancel,　//キャンセルフラグのアドレス LPVOID pigPtr)　//撮影データ格納域アドレス
引　数	igHandle　igOpenDevice で取得したデバイスハンドルを指定します。 pSetCancel　キャンセルフラグのアドレスを指定します。画像の撮影中に処理を中止する場合に指定します。NULL を指定するとキャンセル処理は行いません。また、撮影をキャンセルしても、ステージは停止しません。本関数呼出し時には、変数の初期値として false をセットしてください。 pigPtr　撮影した画像データの格納域アドレスを指定します。事前に、撮影するピクセル数×ライン数分のメモリを確保してください。
戻り値	正常に画像を撮影した場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　デバイスの未オープン IGERR_INVALID_HANDLE　無効なデバイスハンドル IGERR_IO_COMPORT　COM ポート通信エラー IGERR_CANCEL　処理中止（pSetCancel が TRUE になった） IGERR_IPM_DEVICE　デバイスエラー（リソースエラー等） IGERR_IPM_IPCONTROL　デバイスとの通信エラー
備　考	ラインセンサでの撮影のみ行います。ステージの制御が必要な場合は、igMoveStagePos 関数で移動を行ってください。撮影画像の格納域は、撮影するピクセル数×ライン数以上のサイズが必要です。 また、撮影画像から igMakeBMPData 関数で BMP 形式の画像を作成できます。 ＊使用例参照ページ　P.19

2-3-15. ラインセンサ画像撮影（igCaptureImage）

関数名	igCaptureImage
機　能	ステージ制御とラインセンサでの撮影を行い、BMP 形式の画像を作成します。
書　式	ULONG　igCaptureImage (HANDLE igHandle,　//デバイスハンドル BOOL* pigCancel)　//キャンセルフラグのアドレス
引　数	igHandle　igOpenDevice で取得したデバイスハンドルを指定します。 pigCancel　キャンセルフラグのアドレスを指定します。画像の撮影中に処理を中止する場合に指定します。NULL を指定するとキャンセル処理は行いません。また、撮影をキャンセルしても、ステージは停止しません。本関数呼出し時には、変数の初期値として false をセットしてください。
戻り値	正常に画像を撮影した場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　デバイスの未オープン IGERR_INVALID_HANDLE　無効なデバイスハンドル IGERR_IO_COMPORT　COM ポート通信エラー IGERR_CANCEL　処理中止（pSetCancel が TRUE になった） IGERR_IPM_DEVICE　デバイスエラー（リソースエラー等） IGERR_IPM_IPCONTROL　デバイスとの通信エラー IGERR_ALLOC_MEM　メモリが確保できない
備　考	ステージ移動後にステージの現ポジションを更新します。 ＊使用例参照ページ　P.8

2-3-16. 画像データポインタ取得（igGetImageAddress）

関数名	igGetImageAddress
機　能	BMP 形式画像データの各ポインタを取得します。
書　式	ULONG　igGetImageAddress (HANDLE igHandle,　//デバイスハンドル LPVOID *pigPtr,　//画像データポインタ格納域アドレス int igPtrKind,　//画像データポインタの種別 int igLineNo)　//画像データのライン番号
引　数	igHandle　igOpenDevice で取得したデバイスハンドルを指定します。 pigPtr　画像データポインタを格納する変数のポインタを指定します。 igPtrKind　以下の、画像データポインタの種別を指定します。 IG_BMPFILEHEADER_PTR　BMPFILEHEADER のポインタ IG_BMPINFOHEADER_PTR　BMPINFOHEADER のポインタ IG_IMAGE_PTR　BMP 画像データへのポインタ igLineNo　画像データ上側からのライン番号（0～）を指定します。 IG_BMP_START（-1）を指定すると、メモリ上の先頭位置（画像の左下）のアドレスを返します。この引数は PtrKind が IG_IMAGE_PTR の場合のみ有効です。
戻り値	正常にポインタを格納した場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　デバイスの未オープン IGERR_INVALID_HANDLE　無効なデバイスハンドル IGERR_BMP_NODATA　BMP 画像はまだ作成していない IGERR_PARAM_LINE　ライン番号の指定が違う IGERR_PARAM　画像データポインタ種別の指定が違う
備　考	メモリ上には、画像の左下のデータから格納されています。画像の上側から処理する場合は、igLineNo に 0 から（画像のライン数-1）のライン番号を指定して、1 行ずつ読み込んでください。 一般的な DIB フォーマットのデータとして処理する場合は、igLineNo に IG_BMP_START を指定してください。画像の左下のポインタを返します。 ＊使用例参照ページ　P.18

2-3-17. BMP 形式データの作成（igMakeBMPData）

関数名	igMakeBMPData
機　能	撮影した画像データから、BMP 形式のデータを作成します。
書　式	ULONG　igMakeBMPData(HANDLE igHandle,　//デバイスハンドル LPVOID pigImgData,　//撮影した画像データ格納域アドレス UINT igXpixel,　//画像データピクセル数（主走査） UINT igYLines)　//画像データライン数（副走査）
引　数	igHandle　igOpenDevice で取得したデバイスハンドルを指定します。 pigImgData　撮影した画像データの格納域アドレスを指定します。NULL を指定すると、最後に撮影した画像データから作成します。その場合は、igXpixel および igYLines は無効になります。 igXpixel　pImgData で指定した画像データのピクセル数（主走査）を指定します。 igYLines　pImgData で指定した画像データのライン数（副走査）を指定します。
戻り値	正常に描画した場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　デバイスの未オープン IGERR_INVALID_HANDLE　無効なデバイスハンドル IGERR_NOT_CAPTURE　画像を撮影していない IGERR_ALLOC_MEM　メモリが確保できない
備　考	igShotImage 関数で撮影したデータを BMP 形式の画像に変換する場合などに呼び出します。 igMakeBMPData 関数で作成した BMP 形式データは、igDrawImage 関数や igSaveImageFile 関数で、描画や保存ができます。 ＊使用例参照ページ　P.20

2-3-18. 画像データの描画（igDrawImage）

関数名	igDrawImage
機　能	BMP 形式の画像データをピクチャボックスに描画します。
書　式	ULONG　igDrawImage (HANDLE igHandle,　//デバイスハンドル CWnd* pigWnd,　//ピクチャボックスコントロールのハンドル int igDrawMode)　//描画モード
引　数	igHandle　igOpenDevice で取得したデバイスハンドルを指定します。 pigWnd　ピクチャボックスコントロールのウィンドウハンドルを指定します。 igDrawMode　以下の、画像モードを、ビットごとの OR（|）演算子で組み合わせて指定します。 IG_DRAW_SAMERATE　原画像と同比率で描画する IG_DRAW_STRETCH　ピクチャボックスのサイズに合わせて伸縮する IG_DRAW_REFRESH　描画前に背景をクリアする IG_DRAW_NOREFRESH　描画前に背景をクリアしない
戻り値	正常に描画した場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　デバイスの未オープン IGERR_INVALID_HANDLE　無効なデバイスハンドル IGERR_BMP_NODATA　BMP 画像はまだ作成していない IGERR_PARAM　ピクチャボックスコントロールの指定エラー IGERR_DRAW_BMP　BMP 描画エラー
備　考	撮影した画像のほかに、igLoadImageFile 関数を呼び出して読み込んだ画像も描画できます。 igDrawMode は、IG_DRAW_SAMERATE より IG_DRAW_STRETCH の指定が、IG_DRAW_REFRESH より IG_DRAW_NOREFRESH の指定が優先されます。 ＊使用例参照ページ　P.12　P.20

2-3-19. 画像のファイル保存（igSaveImageFile）

関数名	igSaveImageFile
機　能	BMP 形式の画像データをファイルに保存します。
書　式	ULONG　igSaveImageFile (HANDLE igHandle,　　//デバイスハンドル CString igFileName)　//保存する BMP ファイル名
引　数	igHandle　　igOpenDevice で取得したデバイスハンドルを指定します。 igFileName　　画像を保存する、BMP ファイル名をフルパスで指定します。 拡張子は「.bmp」でなければなりません。
戻り値	正常に保存した場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　　デバイスの未オープン IGERR_INVALID_HANDLE　無効なデバイスハンドル IGERR_BMP_NODATA　　BMP 画像はまだ作成していない IGERR_PARAM　　保存ファイル名の指定エラー（拡張子が「.bmp」でない） IGERR_BMP_FILE　　BMP ファイルオープンエラー
備　考	ファイルの格納先フォルダが存在するかどうかはチェックしません。存在しない場合は IGERR_BMP_FILE を返します。 ＊使用例参照ページ　P.13

2-3-20. 画像ファイルの読み込み（igLoadImageFile）

関数名	igLoadImageFile
機　能	BMP 形式の画像データをファイルから読み込みます。
書　式	ULONG　igLoadImageFile (HANDLE igHandle,　　//デバイスハンドル CString igFileName)　//読み込む BMP ファイル名
引　数	igHandle　　igOpenDevice で取得したデバイスハンドルを指定します。 igFileName　画像を保存する、BMP ファイル名をフルパスで指定します。 拡張子は「.bmp」でなければなりません。
戻り値	正常に保存した場合は、IG_NO_ERROR（0）を返します。 エラーが発生した場合は、以下のエラーコードを返します。 IGERR_NOT_OPEN　　デバイスの未オープン IGERR_INVALID_HANDLE　無効なデバイスハンドル IGERR_PARAM　　読み込みファイル名の指定エラー（拡張子が「.bmp」でない） IGERR_BMP_FILE　　BMP ファイルアクセスエラー IGERR_ALLOC_MEM　　メモリが確保できない
備　考	読み込んだ画像は、igDrawImage 関数でピクチャボックスに描画できます。 ＊使用例参照ページ　P.14

2-4.戻り値一覧

戻り値定義名	値	エラー内容
IG_NO_ERROR	0	正常終了
IGERR_SYSTEM	(0xE0000001)	システムエラー（ライブラリの内部エラー等）
IGERR_ALLOC_MEM	(0xE0000002)	メモリが確保できない
IGERR_FREE_MEM	(0xE0000003)	メモリの解放エラー
IGERR_CANCEL	(0xE0000004)	処理中止
IGERR_NOT_CAPTURE	(0xE0000005)	画像を撮影していない
IGERR_DRAW_BMP	(0xE0000006)	BMP 描画エラー
IGERR_BMP_FILE	(0xE0000007)	BMP ファイルエラー
IGERR_BMP_NODATA	(0xE0000008)	画像を撮影または読込みしていない
IGERR_CONFIG_FILE	(0xE0000009)	カメラ設定ファイルのアクセスエラー
IGERR_NOT_SUPPORT	(0xE001000A)	サポート対象外の OS
IGERR_ALREADY_USED	(0xE0000010)	デバイス使用中（別プロセスがオープン中）
IGERR_NOT_OPEN	(0xE0000011)	デバイスをオープンしていない
IGERR_INVALID_HANDLE	(0xE0000012)	デバイスハンドルが無効
IGERR_PARAM	(0xE0000013)	関数のパラメタの指定が間違っている
IGERR_PARAM_PIXCEL	(0xE0000014)	ピクセル数が有効範囲外
IGERR_PARAM_LINE	(0xE0000015)	ライン数が有効範囲外
IGERR_PARAM_SPEED	(0xE0000016)	ステージのスピードが有効範囲外
IGERR_PARAM_STEP	(0xE0000017)	駆動ステップ数が有効範囲外
IGERR_PARAM_CAPTPOS	(0xE0000018)	画像取得ポイント（ステップ数）が有効範囲外
IGERR_PARAM_STDIS	(0xE0000019)	STDIS が有効範囲外
IGERR_PARAM_STENB	(0xE000001A)	STENB が有効範囲外
IGERR_IPM_DEVICE	(0xE0010001)	デバイスエラー（未接続・リソースエラー等）
IGERR_IPM_INITDEVICE	(0xE0010002)	デバイスの初期化エラー
IGERR_IPM_IPCONTROL	(0xE0010003)	デバイスとの通信エラー
IGERR_IPM_OPENDEVICE	(0xE0010004)	デバイスドライバのオープンエラー
IGERR_IPM_MEMORYTBL	(0xE0010005)	メモリテーブルエラー
IGERR_IPM_MEMORY	(0xE0010006)	画像メモリエラー（未使用）
IGERR_IPM_TIMEOUT	(0xE0011001)	タイムアウト発生
IGERR_IPM_FULLFIFO	(0xE0011002)	FIFO が FULL（未使用）
IGERR_IPM_OVERRUN	(0xE0011003)	オーバーランエラー発生
IGERR_IPM_STOPGRABBER	(0xE0011004)	画像処理装置にエラーが発生
IGERR_OPEN_COMPORT	(0xE0020001)	ステージ制御用 COM ポートのオープンエラー
IGERR_IO_COMPORT	(0xE0020002)	COM ポート通信エラー

IGUNOSS, Inc.
有限会社イグノス

IPVDevice（ラインセンサシステム開発キット IP-View 用ライブラリ）
ver.1.0　仕様説明書

2006 年 6 月 21 日　第 1 版発行

有限会社イグノス
岩手県北上市相去町山田 2-18 北上オフィスプラザ 202 号
TEL　0197-67-6396　　FAX　050-7509-7281
URL　http://www.igunoss.co.jp　　E-Mail　igunoss@igunoss.co.jp

IPVDevice
ラインセンサシステム開発キット IP-View 用ライブラリ　ver.1.0