ISC VM SDK V2
ライブラリマニュアル

【ご注意】

1．本マニュアルの内容の一部または全部を無断転載することは禁止されています
2．本マニュアルの内容に関しては将来予告なしに変更することがあります
3．本マニュアルの内容について万全を期して作成しております、万一ご不審な点や誤り、記載漏れなどお気づきのことがございましたら、ご連絡ください
4．運用した結果の影響に関しては、3．項にかかわらず責任を負いかねますのでご了承ください

目次

１．はじめに

本文は、ISC SDK バージョン 2（以下　ISC SDK V2）について説明します。

ISC SDK V2 は、ISC-100VM/PP2(FPGA バージョン 73)以降のカメラに対応する SDK です。バージョン 73 以降のカメラを使用する場合は、本 SDK をお使いください。

旧カメラ・旧 SDK の組合せでお使いのお客様で、新たに追加された API を必要とされずに旧 API を使い続けたい場合は、旧 API と互換性を持つ V1.6.0.0 互換 IF 用の SDK をお使い下さい

※SDK の名称についての補足説明

リリースノートにおける名称	本マニュアルでの名称	内容
上位互換バージョン SDK	V1.6.0.0 互換 IF	旧バージョンと互換性のあるインターフェースを提供します
非互換バージョン SDK	V2	新機能を含み、旧バージョンと互換性のないインターフェースを提供します

2．動作環境

【サポートするカメラ】

本体	ISC-100VM/PP2
FPGA バージョン	73（HEX）以降

【動作確認済み環境】

プラットフォーム	OS
x86　PC　※1	Widnows10　64bit
	Ubuntu 18.04 LTS
Jetson Xavier	Jetpack 4.3
Jetson TX2　※2	Jetpack 4.2 Jetpack 3.1

※1　Intel® Core™ i7-7700 CPU @3.6GHz　16GB RAM

※2　30FPS で動作確認

【Build 環境】

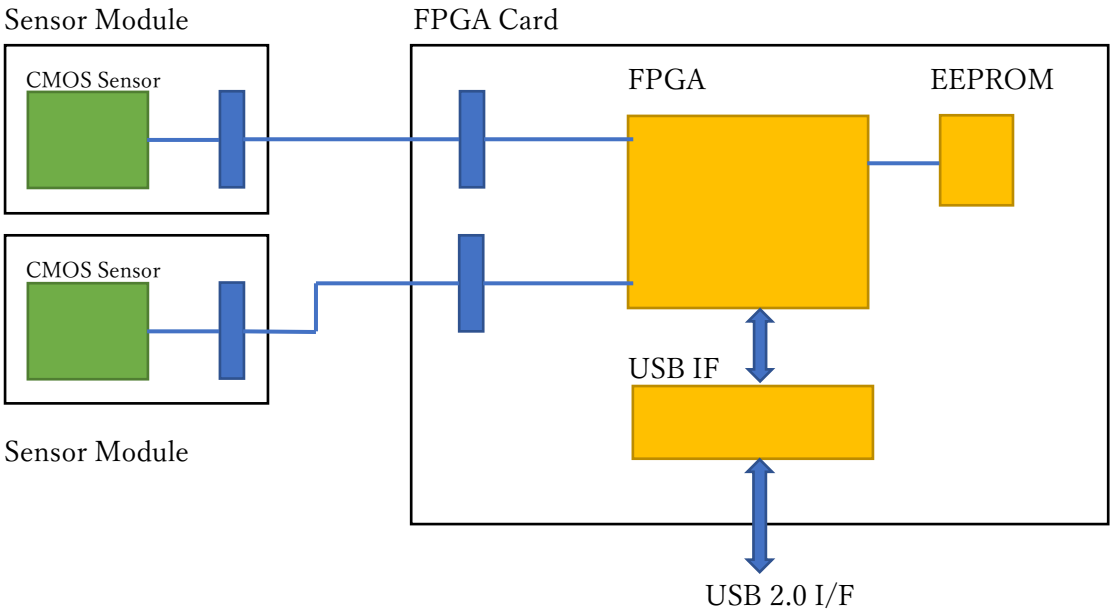
OS	コンパイラ
Widnows10　64bit	Visual Studio 2017 プラットフォームツールセット　v141 SDK バージョン　10.0.17134.0
Ubuntu	GCC 7.3

【使用言語】

OS	言語
Widnows10	C/C++
Ubuntu	C/C++

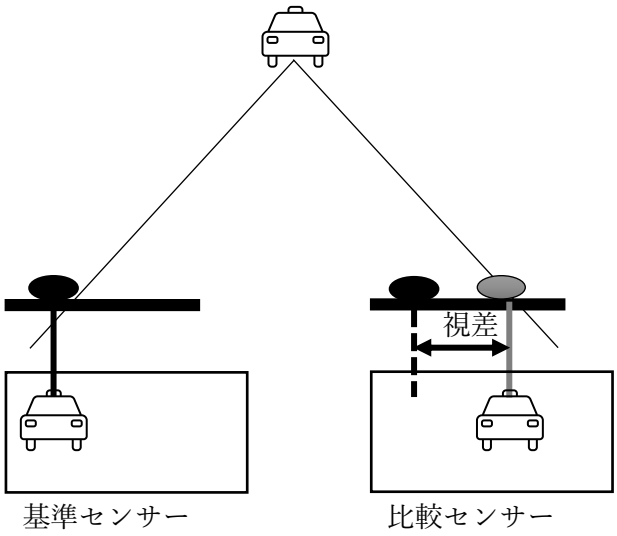
3．ステレオカメラの構成

下図にカメラの簡略なブロック図を示します。
カメラはセンサーを搭載した Sensor Module と、ステレオ処理及び USB インターフェースを行う FPGA Card によって構成されています。



3．1　基準画像カメラと参照画像カメラ

ステレオカメラで対象物までの距離を求めるには，下図に示すように三角測量の原理を利用して算出しており、同じ対象物を 左右のカメラで撮像した際の撮像位置の差分（視差）を求めています。



3．2　画像データフォーマット

（1）センサーサイズ　752（幅）×　480（高）　画素
（2）画像有効サイズ　752（幅）×　480（高）　画素
（3）視差有効サイズ　640（幅）×　480（高）　画素
（4）画像データフォーマット

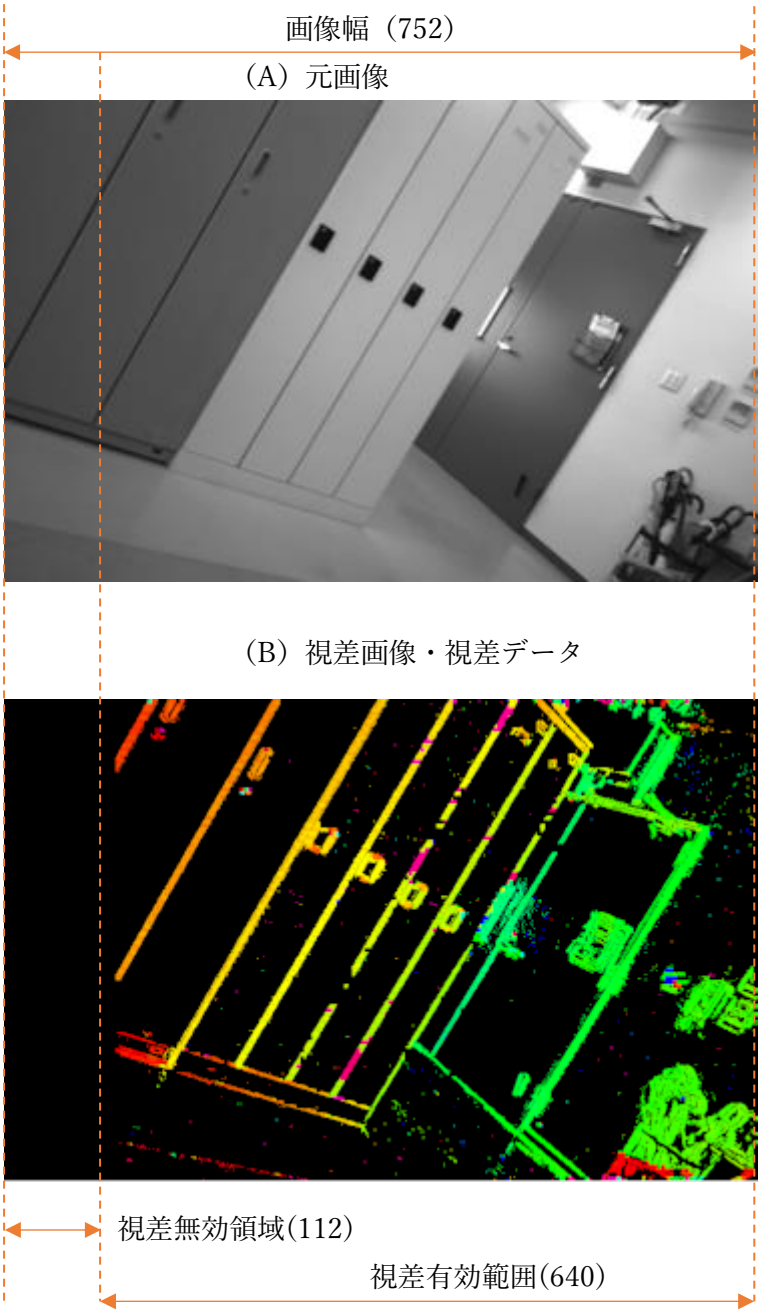
受信データは、画像の左下画素より格納されています。

画像バッファー（752x480 のケース）

0	1				750	751
360208						360959

（５）視差の無効領域

下図のように視差データは無効な領域を含みます。



３．３　EEPROM（不揮発性メモリ）の機能

EEPROM には、下記のデータを保存しています。
ユーザーは、API を使用して一部のデータを取得できますが、直接アクセスすることはできません。

データ名称	備考
補正校正テーブル	
システム設定データ	一部のデータは API（GetCameraParamInfo）を使用して取得できます
Auto Calibration データ	動作パラメータ及びエラーステータスが保存されます

４．露光調整機能

４．１　概要

カメラのシャッター制御を行います。
シャッター制御はマニュアルモード及びシングルシャッターモードとダブルシャッターモードを持ちます。
入力された補正画像の指定されたエリアの輝度ヒストグラムを作成し Exposure （シャッターを開いている時間 = 蓄積時間)を制御し、CMOS センサーに入射する光量を最適化すると共に、CMOS センサーのアンプの Gain(利得)を制御することにより、最適な輝度出力が得られるようにします。

４．２　露光調整測定エリアの指定

シャッターの値を制御するときには、イメージ全体の輝度分を使うわけではなく、指定されたエリアの輝度分布を使用します。
エリアの指定は以下の 2 種類が準備されています。

① 上下左右の 4 か所の座標を指定して長方形のエリアを指定します
② 上下の座標と、上辺の左右、下辺の左右の 6 か所の座標を指定して四角形又は三角形のエリアを指定します

詳細は、API(SetMeasArea)を参照してください。

４．３　シングルシャッターモード（SampleViewer 設定名：Normal On）

1 フレームの入力画像を解析し Exposure と Gain を決定する為、早い応答速度で最適な露光調整値を算出します。
４．４章のダブルシャッターモードと比較し、1/3～同等（数 frame～数 10frame）の応答速度となります。

４．４　ダブルシャッターモード

ダイナミックレンジの広い入力画像に対して、視差情報の欠損を最小限に抑えるよう
シャッターを制御する機能です。
一般的な HDR では長いシャッターと短いシャッターの画像を合成し 1 フレームを生成する為、合成後の画像では輝度情報が圧縮されてしまいます。
輝度情報から視差情報を生成するステレオカメラにとって、輝度情報が圧縮される事は視差情報が減少する事に繋がり、HDR の効果を最大限得ることはできません。
この問題を解消する機能がダブルシャッターになります。

ダブルシャッターモードでは 1 フレーム毎に長いシャッターと短いシャッターの画像が交互に入力されるようにシャッターの値を制御します。
視差情報の算出をそれぞれのフレームで行う事で視差情報の欠損を抑えています。

４．４．１　画像合成有り（SampleViewer 設定名：Double On（Image Fusion））

カメラから交互に出力される長いシャッターと短いシャッターの画像を SDK で合成し表示するモードです。
ダブルシャッターによるシャッター制御を視覚的に理解する為の表示モードです。
またこの画像を使い後段アプリを設計する事も可能です。

４．４．２　画像合成無し（SampleViewer 設定名：Double On）

カメラから交互に出力される長いシャッターと短いシャッターの画像をそのまま表示するモードです。従って、ダイナミックレンジの広いシーンではフレーム毎に明暗の点滅が起きます。
SDK で画像合成を行わない為、マイコンなど限られたリソース環境での開発に適しています。

４．５　マニュアルモード（SampleViewer 設定名：Off）

シャッター制御をユーザーの指定した値とします。
API（SetExposureValue/SetGainValue）を使用して値を指定します。

５．自動調整機能（Auto Calibration）

経年劣化や熱膨張等でカメラが変形した事による視差密度低下を FPGA で検出・改善する機能です。
自動調整には以下２モードがあります。

- 自動調整：カメラ内部で視差密度低下を検出し改善します。
- 強制調整：アプリケーションからのトリガにより視差密度の改善を行います。

６．関数一覧

関数名	概要	備考
OpenISC	ライブラリのオープン	
CloseISC	ライブラリのクローズ	
StartGrab	画像取り込み開始	
StopGrab	画像取り込み停止	
GetImage	画像取得	
GetImageEx	画像取得　同期タイプ	
GetDepthInfo	視差データ取得	
GetCameraParamInfo	カメラ固有パラメータ取得	
GetImageSize	画像サイズの取得	
SetAutoCalibration	自動調整モード指定（自動・強制）	
GetAutoCalibration	自動調整モード取得（自動・強制）	
SetShutterControlMode	露光調整モード指定	
GetShutterControlMode	露光調整モード取得	
SetExposureValue	露光設定値指定	
GetExposureValue	露光設定値取得	
SetGainValue	ゲイン設定値指定	
GetGainValue	ゲイン設定値取得	
SetHDRMode	HDR モード指定	
SetHiResolutionMode	ハイレゾリューションモード指定	
SetNoiseFilter	ノイズフィルター値指定	
GetNoiseFilter	ノイズフィルター値取得	
SetMeasArea	露光調整　測定エリア指定	
GetMeasArea	露光調整　測定エリア取得	
SetCameraFPSMode	カメラのフレームレート指定	
GetCameraFPSMode	カメラのフレームレート取得	

※V1.6.0.0 互換 IF については、１３．アップグレード情報を参照してください。

７．関数説明

実装している関数について説明します。

API は原則として Windows 版と Linux 版で共通ですが、いくつかの違いがあります。
以降は Windows 版について説明しますので、Windows 版と Linux 版の違いについては、１２．Linux 版における変更点　を参照してください。

OpenISC

C++/C	int OpenISC()

Parameter		
Return Value	成功時　0 失敗時　!=0	
Remarks		
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	ライブラリの使用を開始します

	int ret = OpenISC();
	if(ret !=0){
	　// Error 処理
	}

CloseISC

C++/C	int CloseISC ()

Parameter		
Return Value	成功時　0 失敗時　!=0	
Remarks		
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	ライブラリの使用を終了します

	int ret = CloseISC ();
	if(ret !=0){
	　// Error 処理
	}

StartGrab

C++/C	int StartGrab(int nMode);

Parameter	nMode	取り込みモードを指定します 2：視差モード 3：補正後画像モード 4：補正前画像モード
Return Value	成功時　0 失敗時　!=0	
Remarks		
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	画像の取り込みを開始します

	int nMode=2;
	int ret = StartGrab(nMode);
	if(ret !=0){
	// Error 処理
	}

StopGrab

C++/C	int StopGrab();

Parameter		
Return Value	成功時　0 失敗時　!=0	
Remarks		
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	画像取り込みを停止します

	int ret = StopGrab ();
	if(ret !=0){
	　// Error 処理
	}

GetImage

C++/C	int GetImage(unsigned char* pBuffer1, unsigned char* pBuffer2,int nSkip)

Parameter	pBuffer1 pBuffer2	※StartGrab で指定する取り込みモード及び SetShutterControlMode で指定するモードによって内容が変わります
	nSkip	未使用　0 を指定する
Return Value	成功時　0 失敗時　!=0	
Remarks		
See also	失敗時のコードは、エラーコード一覧を参照してください	

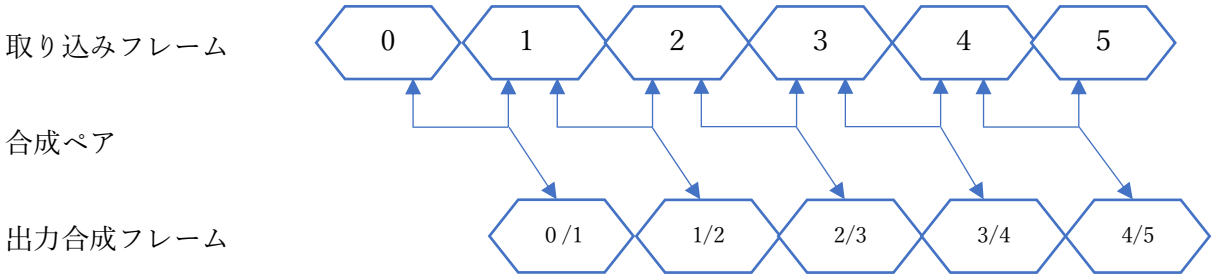
Example	1 Frame の画像を取得する

	unsigned char* pBuffer1 = new unsigned char[width * height];
	unsigned char* pBuffer2 = new unsigned char[width * height];
	Int ret = GetImage(pBuffer1, pBuffer2, 0);
	If(ret!= 0){
	　// Error
	}
	// データ使用
	・・・
	delete[] pBuffer1;
	delete[]pBuffer2;

【データの内容】

		pBuffer1	pBuffer2
露光調整モード	取り込みモード		
マニュアル・ シングルシャッター	視差画像モード	視差画像	基準センサー補正後画像
	補正後画像モード	比較センサー補正後画像	基準センサー補正後画像
	補正前画像モード	比較センサー補正後画像	基準センサー補正前画像
ダブルシャッター （画像合成あり）	視差画像モード	合成視差画像	合成基準センサー補正後画像
	補正後画像モード	比較センサー補正後画像	基準センサー補正後画像
	補正前画像モード	比較センサー補正前画像	基準センサー補正前画像
ダブルシャッター （画像合成なし）	視差画像モード	視差画像	基準センサー補正後画像
	補正後画像モード	比較センサー補正後画像	基準センサー補正後画像
	補正前画像モード	比較センサー補正前画像	基準センサー補正前画像

【合成のデータについて】
合成は、常に手前のフレームと行います。



【エラー処理】
Return value の標準的な処理を以下に示します。

```
int ret = GetImage(xxxx):

if(ret != 0){
  // ERROR

  If(ret == ERR_USB_NO_IMAGE){
    画像の準備ができていないため、新しく表示又は処理するデータがありません
    例えば、カメラの受信周期より速い周期で呼び出した場合に発生します
  }
  else if(ret == FT_IO_ERROR){
    USB の接続エラーです
    例えば、ケーブルが外れたケースなどです
    アプリケーションに合わせて適切に処理してください
  }
  else if(ret == ERR_NO_VALID_IMAGES_CALIBRATING){
    自動調整中なので処理又は表示するためのデータがありません
  }
  else {
    画像に起因するエラーが多いため、確認のために表示することを推奨します
    画像を確認し適切に処理してください
  }
}
```

GetImageEx

C++/C	int GetImageEx(unsigned char* pBuffer1, unsigned char* pBuffer2, int nSkip, int nWaitTime = 100)

Parameter	pBuffer1 pBuffer2	※StartGrab で指定する取り込みモード及び SetShutterControlMode で指定するモードによって内容が変わります
	nSkip	未使用　0 を指定する
	nWaitTime	タイムアウト時間(msec)を指定します
Return Value	成功時　0 失敗時　!=0	
Remarks	SDK バージョン　2.3.0 より使用可能です ブロッキングタイプの関数です。データを受信する又はタイムアウトするまで関数は戻りません 他の機能は、GetImage と同一です	
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	1 Frame の画像を取得する

	unsigned char* pBuffer1 = new unsigned char[width * height];
	unsigned char* pBuffer2 = new unsigned char[width * height];
	Int ret = GetImageEx(pBuffer1, pBuffer2, 0, 10);
	If(ret!= 0){
	　// Error
	}
	// データ使用
	・・・
	delete[] pBuffer1;
	delete[]pBuffer2;

GetDepthInfo

C++/C	int GetDepthInfo(float* pBuffer)

Parameter	pBuffer	視差データ
Return Value	成功時　0 失敗時　!=0	
Remarks	先行して GetImage()を呼びだす必要があります 視差データを距離(m)に変換するための方法は、10.距離変換を参照してください	
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	1 Frame の視差情報を取得します

	float* pBuffer = new float[width * height];
	int ret = GetDepthInfo(pBuffer);
	if(ret != 0){
	　// Error
	}
	//・・・データ使用
	delete[] pBuffer;

【視差データフォーマット】
視差値（浮動小数点）が画像データと同じ順番で入ります。

GetCameraParamInfo

C++/C	int GetCameraParamInfo(CameraParamInfo* pParam)

Parameter	pParam	カメラ固有のパラメータ構造体です
Return Value	成功時　0 失敗時　!=0	
Remarks		
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	パラメータを取得します

	CameraParamInfo parameter = {};
	int ret = GetCameraParamInfo(&parameter);
	if(ret != 0){
	　// Error
	}
	// ・・・データ使用

【CameraParamInfo 構造体】

型	変数名	内容
float	fD_INF	無限遠値
unsigned int	nD_INF	※1
float	fBF	BF 値
float	fBaseLength	基線長
float	dZ	ΔZ 値　※1
float	fViewAngle	視野角
unsigned int	nImageWidth	画像幅
unsigned int	nImageHeight	画像高さ
unsigned int	nProductNumber	製品番号
unsigned int	nProductNumber2	製品番号（上位桁）
char	nSerialNumber	製品シリアル番号文字列 最大　8 文字（半角）
unsigned int	nFPGA_Version_Major	製品 FPGA メジャーバージョン
unsigned int	nFPGA_Version_Minor	製品 FPGA マイナーバージョン
unsigned int	nDistanceHistValue	※1
unsigned int	nParallaxThreshold	※1

※1　未使用です。　将来のために予約されています

GetImageSize

C++/C	GetImageSize(unsigned int* pnWidth, unsigned int* pnHeight)

Parameter	pnWidth	画像幅
	pnHeight	画像高さ
Return Value	成功時　0 失敗時　!=0	
Remarks		
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	画像の幅、高さを取得します

	unsigned int width=0, height=0;
	int ret = GetImageSize(&width, &height);
	if(ret != 0){
	　// Error
	}
	// ・・・データ使用

SetAutoCalibration

C++/C	int SetAutoCalibration(int nMode)

Parameter	nMode	自動調整モードを指定します 0：停止 1：自動調整 On 2：強制調整開始
Return Value	成功時　0 失敗時　!=0	
Remarks		
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	調整モードを設定します

	int mode = 1;
	int ret = SetAutoCalibration(mode);
	if(ret !=0({
	　// Error
	}

GetAutoCalibration

C++/C	int GetAutoCalibration(int* nMode)

Parameter	nMode	現在の自動調整モードを表します bit / 内容 0 / 強制調整動作状態 0 ：待機中 1 ：動作中 1 / 自動調整 On/Off 0 ：Off 1 ：O n 2~15 / 予約
Return Value	成功時　0 失敗時　!=0	
Remarks		
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	調整モードの状態を取得します

	int mode = 0;
	int ret = GetAutoCalibration (&mode);
	if(ret !=0({
	　// Error
	}
	if((mode & 0x00000001) != 0){
	　// 調整中
	}

SetShutterControlMode

C++/C	int SetShutterControlMode(int nMode)

Parameter	nMode	露光調整モードを指定します 0：マニュアルモード 1：シングルシャッターモード 2：ダブルシャッター動作モード 3：ダブルシャッター動作モード （画像合成なし動作）
Return Value	成功時　0 失敗時　!=0	
Remarks	V1.6.0.0 互換 IF と V2 でパラメータの相違があります ダブルシャッター機能の詳細は、４．露光調整の項を参照してください	
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	シングルシャッターモードを指定します

	int mode = 1;
	int ret = SetShutterControlMode (mode);
	if(ret !=0({
	　// Error
	}

GetShutterControlMode

C++/C	int GetShutterControlMode(int* nMode)

Parameter	nMode	現在の露光調整モードを表します 0：マニュアルモード 1：シングルシャッターモード 2：ダブルシャッター動作モード 3：ダブルシャッター動作モード （画像合成なし動作）
Return Value	成功時　0 失敗時　!=0	
Remarks	V1.6.0.0 互換 IF と V2 でパラメータの相違があります	
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	動作モードを取得します

	int mode = 0;
	int ret = GetShutterControlMode (&mode);
	if(ret !=0({
	　// Error
	}

SetExposureValue

C++/C	int SetExposureValue(unsigned int nValue)

Parameter	nValue	露光制御の値を指定します 有効設定範囲：1~480
Return Value	成功時　0 失敗時　!=0	
Remarks	ShutterControlMode でマニュアルモードを設定時のみ有効です	
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	露光値を設定します

	unsigned int value = 120;
	int ret = SetExposureValue (value);
	if(ret !=0({
	　// Error
	}

※設定値は、画像取り込みで反映されます。画像取り込みを停止している場合に設定した値は、次回の画像取り込み時又は、画像増取り込み後に読み込むことができます。
画像取り込み中に設定した値は、常時反映されます。

GetExposureValue

C++/C	int GetExposureValue(unsigned int* pnValue)

Parameter	pnValue	現在の設定値を表します
Return Value	成功時　0 失敗時　!=0	
Remarks		
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	現在の設定値を取得します

	unsigned int value = 0;
	int ret = GetExposureValue (&value);
	if(ret !=0({
	　// Error
	}

SetGainValue

C++/C	int SetGainValue(unsigned int nValue)

Parameter	nValue	ゲインの値を指定します 有効設定範囲：16~64
Return Value	成功時　0 失敗時　!=0	
Remarks	ShutterControlMode でマニュアルモードを設定時のみ有効です	
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	ゲイン値を設定します

	unsigned int value = 20;
	int ret = SetGainValue (value);
	if(ret !=0({
	　// Error
	}

※設定値は、画像取り込みで反映されます。画像取り込みを停止している場合に設定した値は、次回の画像取り込み時又は、画像増取り込み後に読み込むことができます。
画像取り込み中に設定した値は、常時反映されます。

GetGainValue

C++/C	int GetGainValue(unsigned int* pnValue)

Parameter	pnValue	現在の設定値を表します
Return Value	成功時　0 失敗時　!=0	
Remarks		
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	現在の設定値を取得します

	unsigned int value = 0;
	int ret = GetGainValue (&value);
	if(ret !=0({
	　// Error
	}

SetHDRMode

C++/C	int SetHDRMode(int nValue)

Parameter	nValue	HDR の On/Off を指定します 0 ：Off 1 ：On
Return Value	成功時　0 失敗時　!=0	
Remarks	CMOS センサーの機能を使用します	
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	HDR モードを On に設定します

	int value = 1;
	int ret = SetHDRMode (value);
	if(ret !=0({
	　// Error
	}

SetHiResolutionMode

C++/C	int SetHiResolutionMode(int nValue)

Parameter	nValue	High Resolution モードの On/Off を指定します 0 ：Off 1 ：On
Return Value	成功時　0 失敗時　!=0	
Remarks	CMOS センサーの機能を使用します	
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	High Resolution モードを On に設定します

	int value = 1;
	int ret = SetHiResolutionMode (value);
	if(ret !=0({
	　// Error
	}

SetNoiseFilter

C++/C	int SetNoiseFilter(unsigned int nDCDX)

Parameter	nDCDX	ノイズフィルターの値を指定します 有効設定範囲： 0 ～255 値が大きいほどノイズ除去が強くなります
Return Value	成功時　0 失敗時　!=0	
Remarks		
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	フィルター値を設定します

	unsigned int value = 4;
	int ret = SetNoiseFilter (value);
	if(ret !=0({
	　// Error
	}

GetNoiseFilter

C++/C	int GetNoiseFilter(unsigned int* nDCDX)

Parameter	nDCDX	現在の設定値を表します
Return Value	成功時　0 失敗時　!=0	
Remarks		
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	現在の設定値を取得します

	unsigned int value = 0;
	int ret = GetNoiseFilter (&value);
	if(ret !=0({
	　// Error
	}

SetMeasArea

C++/C	int SetMeasArea(int mode, int nTop, int nLeft, int nRight, int nBottom, int nTop_Left, int nTop_Right, int nBottom_Left, int nBottom_Right)

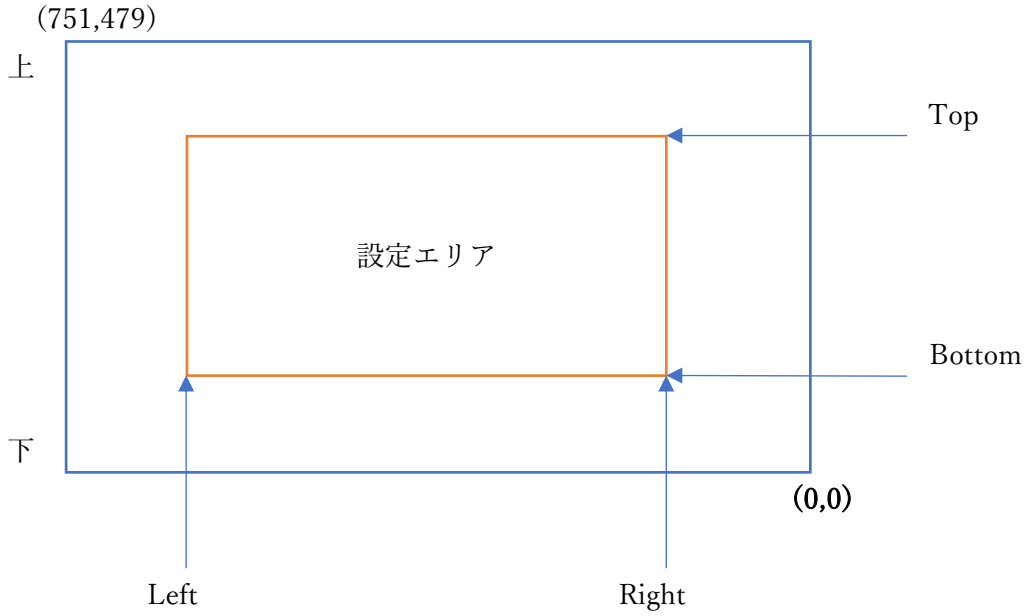
Parameter	mode	露光調整測定エリアの設定方法を指定します 0 ：4 か所の座標を指定 ※以下 mode(0)と書きます 1 ：6 か所の座標を指定 ※以下 mode(1)と書きます
	nTop	mode(0)：上座標 mode(1)：上座標
	nLeft	mode(0)：左座標 mode(1)：(未使用)
	nRight	mode(0)：右座標 mode(1)：(未使用)
	nBottom	mode(0)：下座標 mode(1)：下座標
	nTop_Left	mode(0)：(未使用) mode(1)：上辺の左座標
	nTop_Right	mode(0)：(未使用) mode(1)：上辺の右座標
	nBottom_Left	mode(0)：(未使用) mode(1)：下辺の左座標
	nBottom_Right	mode(0)：(未使用) mode(1)：下辺の右座標
Return Value	成功時　0 失敗時　!=0	
Remarks	V2 のみ提供	
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	mode(0)で 4 点を指定します
	int mode = 0;

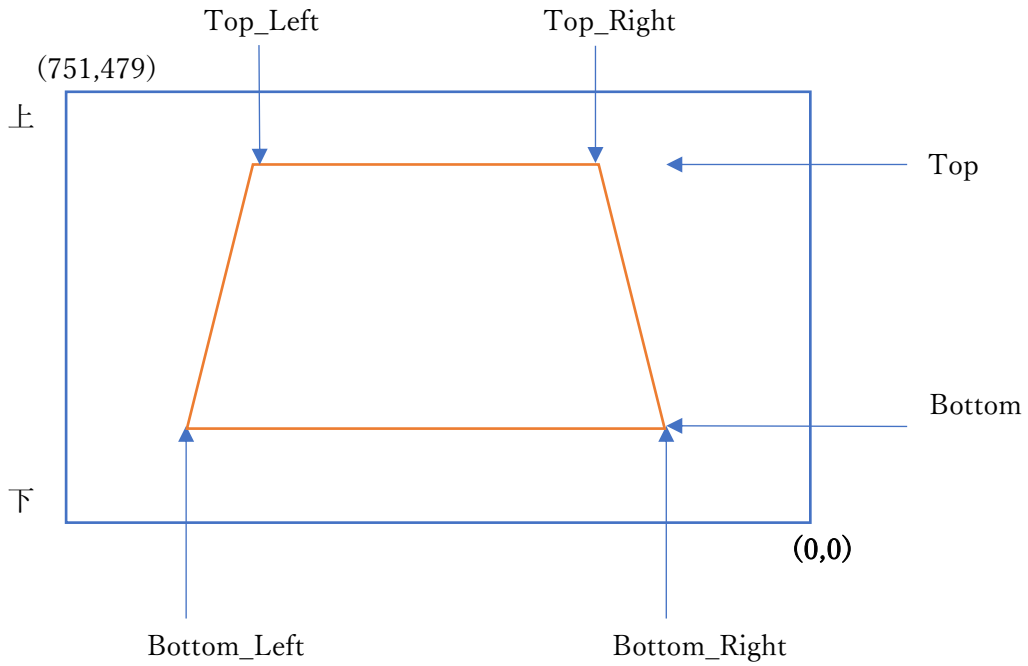
	int nTop = 400;
	int nLeft = 750;
	int nRight = 2;
	int nBottom = 80;
	int nTop_Left = nTop_Right = nBottom_Left = int nBottom_Right = 0;
	
	int ret = SetMeasArea (mode, nTop, nLeft, nRight, nBottom,
	nTop_Left, nTop_Right, nBottom_Left, nBottom_Right);
	if(ret !=0({
	　// Error
	}

【座標系】

・4 か所の座標を指定



・6 か所の座標を指定



※Top_Left=Top_Right または、Bottom_Left=Bottom_Right とすることで、三角形にすることが可能です。

GetMeasArea

C++/C	int GetMeasArea(int* mode, int* nTop, int* nLeft, int* nRight, int* nBottom, int* nTop_Left, int* nTop_Right, int* nBottom_Left, int* nBottom_Right)

Parameter	mode	現在の設定方法を表します 0 ：4 か所の座標を指定 ※以下 moed(0)と書きます 1 ：6 か所の座標を指定 ※以下 moed(1)と書きます
	nTop	moed(0)：上座標 moed(1)：上座標
	nLeft	moed(0)：左座標 moed(1)：(未使用)
	nRight	moed(0)：右座標 moed(1)：(未使用)
	nBottom	moed(0)：下座標 moed(1)：下座標
	nTop_Left	moed(0)：(未使用) moed(1)：上辺左座標
	nTop_Right	moed(0)：(未使用) moed(1)：上辺右座標
	nBottom_Left	moed(0)：(未使用) moed(1)：下辺左座標
	nBottom_Right	moed(0)：(未使用) moed(1)：下辺右座標
Return Value	成功時　0 失敗時　!=0	
Remarks	V2 のみ提供	
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	現在の設定値を取得します
	int mode = 0;

	int nTop = nLeft = nRight = nBottom = 0;
	int nTop_Left = nTop_Right = nBottom_Left = int nBottom_Right = 0;
	
	int ret = GetMeasArea (&mode, &nTop, &nLeft, &nRight, &nBottom,
	&nTop_Left, &nTop_Right, &nBottom_Left, &nBottom_Right);
	if(ret !=0({
	　// Error
	}

SetCameraFPSMode

C++/C	int SetCemraFPSMode(int nMode)

Parameter	nMode	モードを指定します 0：システムデフォルト（60FPS） 1：30FPS モード 2：60FPS モード
Return Value	成功時　0 失敗時　!=0	
Remarks		
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	フレームレートを 30FPS に設定します

	int value = 1;
	int ret = SetCameraFPSMode (value);
	if(ret !=0({
	　// Error
	}

GetCameraFPSMode

C++/C	int GetCemraFPSMode(int* pnMode, int* pnNominalFPS)

Parameter	pnMode	モードを表します 1 ：30FPS 2 ：60FPS
	pnNominalFPS	pnMode に対応する公称 FPS 値を表します 30 又は 60 となります。
Return Value	成功時　0 失敗時　!=0	
Remarks		
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	フレームレートを取得します

	int value = 0;
	int nominalFPS = 0;
	int ret = GetCameraFPSMode (&value, & nominalFPS);
	if(ret !=0({
	　// Error
	}

8．エラーコード一覧

コード	内容	備考
1	無効なハンドル	FTDI エラー
2	デバイスが見つからない	FTDI エラー
3	デバイスがオープンされない	FTDI エラー
4	I/O エラー	FTDI エラー
5	リソースが不足	FTDI エラー
6	無効なパラメータ	FTDI エラー
7	不当なボーレート	FTDI エラー
8	開かれていないデバイスを消そうとしてできなかった	FTDI エラー
9	開かれていないデバイスに書き込もうとしたができなかった	FTDI エラー
10	デバイスへの書き込みに失敗した	FTDI エラー
11	EEPROM の読み込みに失敗した	FTDI エラー
12	EEPROM の書き込みに失敗した	FTDI エラー
13	EEPROM の消去に失敗した	FTDI エラー
14	EEPROM がこのデバイスでは提供されていない	FTDI エラー
15	EEPROM をプログラムされていない	FTDI エラー
16	不当な引数であった	FTDI エラー
17	サポートされていない	FTDI エラー
18	その他のエラー	FTDI エラー
19	デバイスリストが用意されていない	FTDI エラー
-1	読み込みサイズエラー	
-2	書き込みサイズエラー	
-3	データ受信タイムアウトエラー	
-4	ISC object 未作成エラー	
-5	USB Open エラー	
-6	USB コンフィグレーション エラー	
-7	カメラ コンフィグレーション エラー	
-8	レジスタ設定エラー	
-9	受信スレッドエラー	
-10	リセットエラー	
-11	画像取り込みモードエラー	
-12	画像取り込みモード設定エラー	

-13	補正テーブルエラー	
-14	モード設定得エラー	
-15	キャリブレーションエラー	
-16	画像取得エラー	
-17	無効データ	
-18	モードエラー(画像取り込みモード中以外)	
-19	調整動作中のため画像は無効	
-20	要求を受け付けられない	
-100	USB error	
-101	既に Open 済み	
-102	取得イメージ無し	
-201	自動調整失敗　自動調整を連続で失敗	FPGA エラー
-202	自動調整失敗　自動調整連続失敗が繰り返し発生	FPGA エラー
-203	自動調整失敗　自動調整の調整範囲外	FPGA エラー
-204	画像不良　画像が暗すぎるなど	FPGA エラー
-205	（予約）	
-206	自動調整不能エラー	FPGA エラー
-207	（予約）	
-208	（予約）	
-209	（予約）	
-210	（予約）	

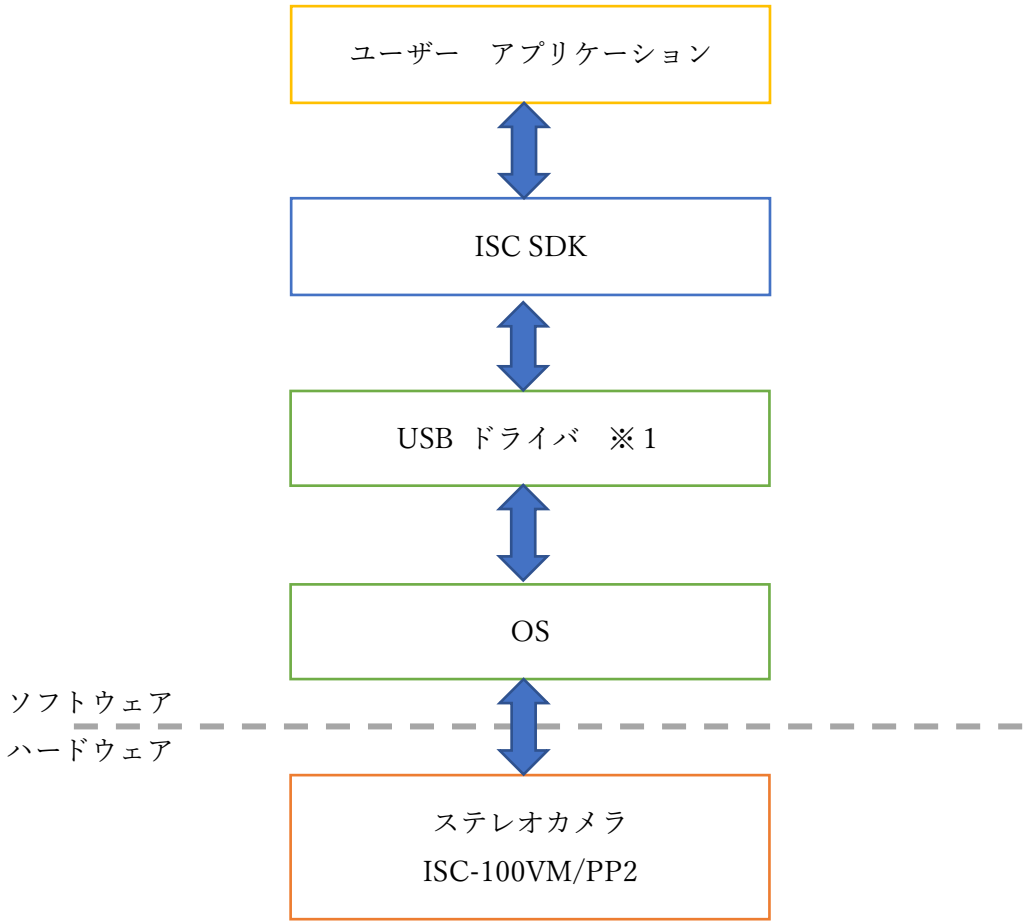
【FPGA エラーとその対応】

コード	エラー名	内容
-201	ERR_AUTOCALIB_GIVEUP_WARN	自動調整失敗－1
-202	ERR_AUTOCALIB_GIVEUP_ERROR	自動調整失敗－2
-203	ERR_AUTOCALIB_OUTRANGE	自動調整失敗－3
-204	ERR_IMAGEINPUT_IMAGEERROR	センサー画像不良
-206	ERR_AUTOCALIB_FAIL_AUTOCALIB	自動調整失敗－4

各エラーコードの詳細は、以下となります。

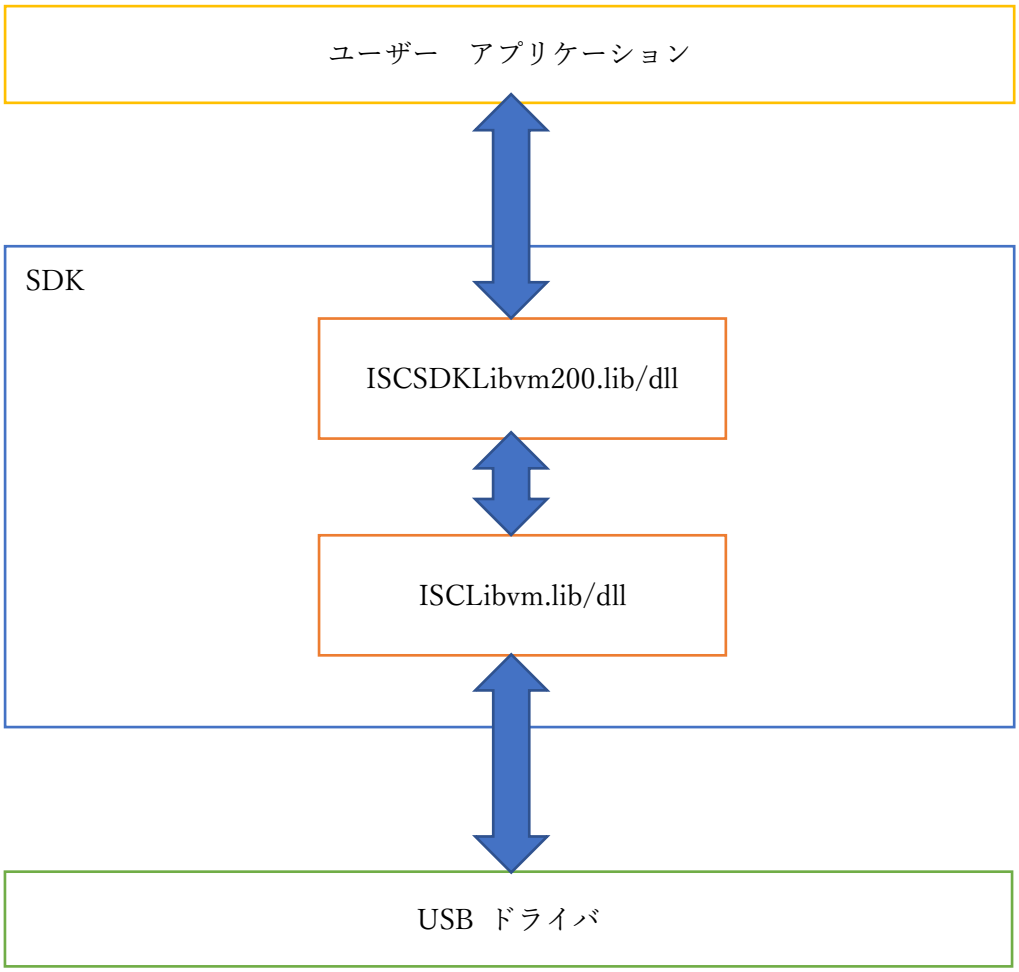
エラー名	種別	原因	対応
自動調整失敗－1	ワーニング	一定回数連続で自動調整が発動し調整に失敗した	・シーン変更 ・電源 OFF/ON
自動調整失敗－2	エラー	自動調整失敗ワーニングが一定回数連続で発生し調整に失敗した	カメラ再調整 （代理店へお問い合わせください。）
自動調整失敗－3	エラー	カメラ歪みが自動調整範囲を超えてしまった	カメラ再調整 （代理店へお問い合わせください。）
センサー画像不良	ワーニング	・レンズに異物が付着している 等	・レンズ異物除去 ・電源 OFF/ON
自動調整失敗－4	エラー	自動調整に失敗した	カメラ再調整 （代理店へお問い合わせください。）

9．SDK　ライブラリアーキテクチャ



※1　Windows 用 USB ドライバ
FTDI 社　提供のドライバを使用します
初めてカメラを接続したときに PC がインターネットに接続されている場合には、自動的にドライバがインストールされます。
それ以外の場合は、FTDI 社のホームページを参照してください。

【SDK ファイル構成】


ユーザー　アプリケーション
SDK
ISCSDKLibvm200.lib/dll
ISCLibvm.lib/dll
USB ドライバ

１０．SDK　フォルダー構成

```
SDK
├──LINUX                          Linux 用 SDK　V2
│    ├──lib                       Linux 用ライブラリ
│    │    ├──aarc64               arm 対応（Jetson Jetpack 4.3 で Build）
│    │    └──x86-64               x86/64bit 対応
│    └──sample
│         └──iscviewer            viewer サンプルコード
│
├──WINDOWS                        Windows 用 SDK　V2
│    ├──lib                       Windows 用ライブラリ
│    │    └──x64                  64bit 対応
│    ├──sample                    サンプルコード
│    │    ├──SampleViewer         Viewre サンプルコード
│    │    └──SDK
│    └──viewer                    Windows 用ビューワー
│
├──XLIBRARY_FOR_BACKWARD_COMPATIBILITY        V1.6.0.0 互換 IF
│    ├──LINUX                     Linux 用 SDK　 V1.6.0.0 互換 IF
│    │
│    └──WINDOWS                   Windows 用 SDK　 V1.6.0.0 互換 IF
│
└──Manual                         マニュアル
```

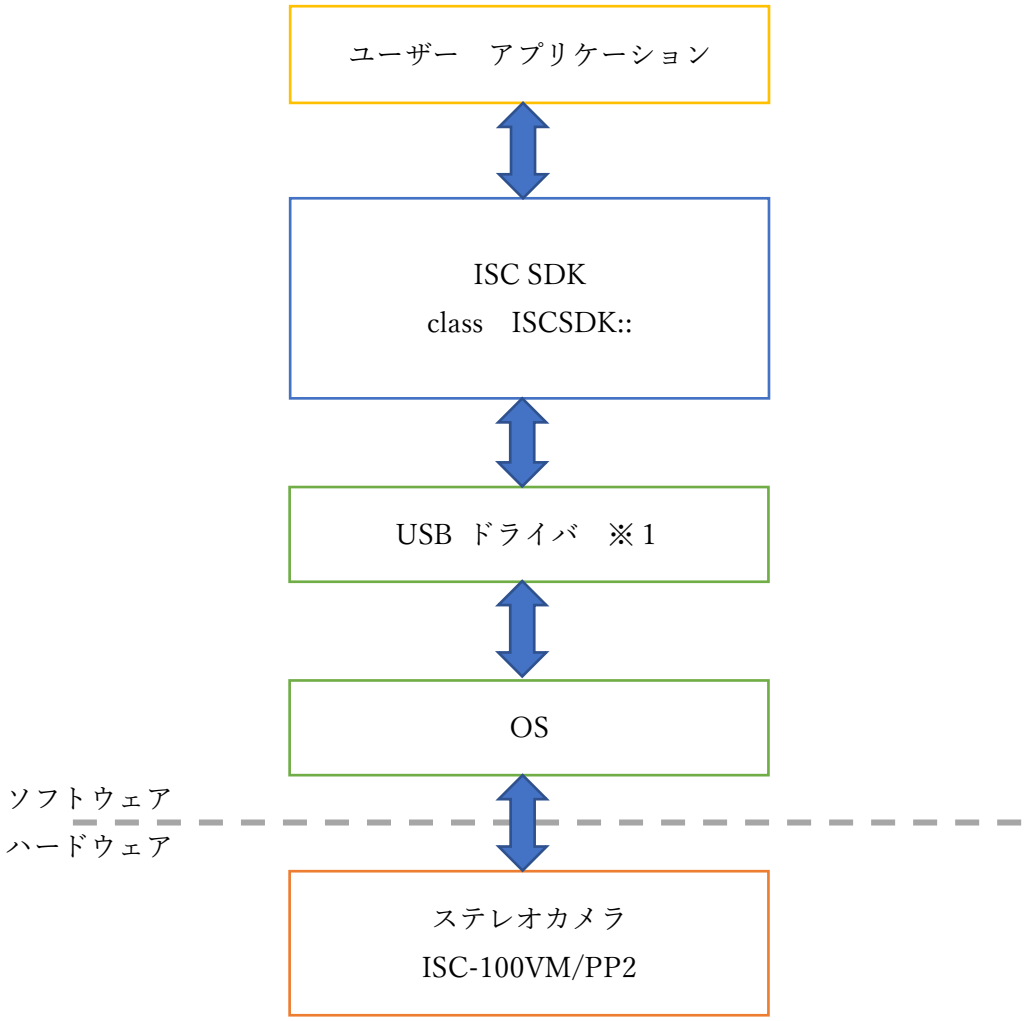
１１．サンプルコードの使い方（Windows 版）

対応環境：Windows10 64bit
開発環境：Visual Studio2017

（１） SampleViewer.sln を起動します
（２） 構成が x64 であることを確認し、Build します
（３） Lib フォルダー内より DLL を実行フォルダーへコピーします
（４） 実行します

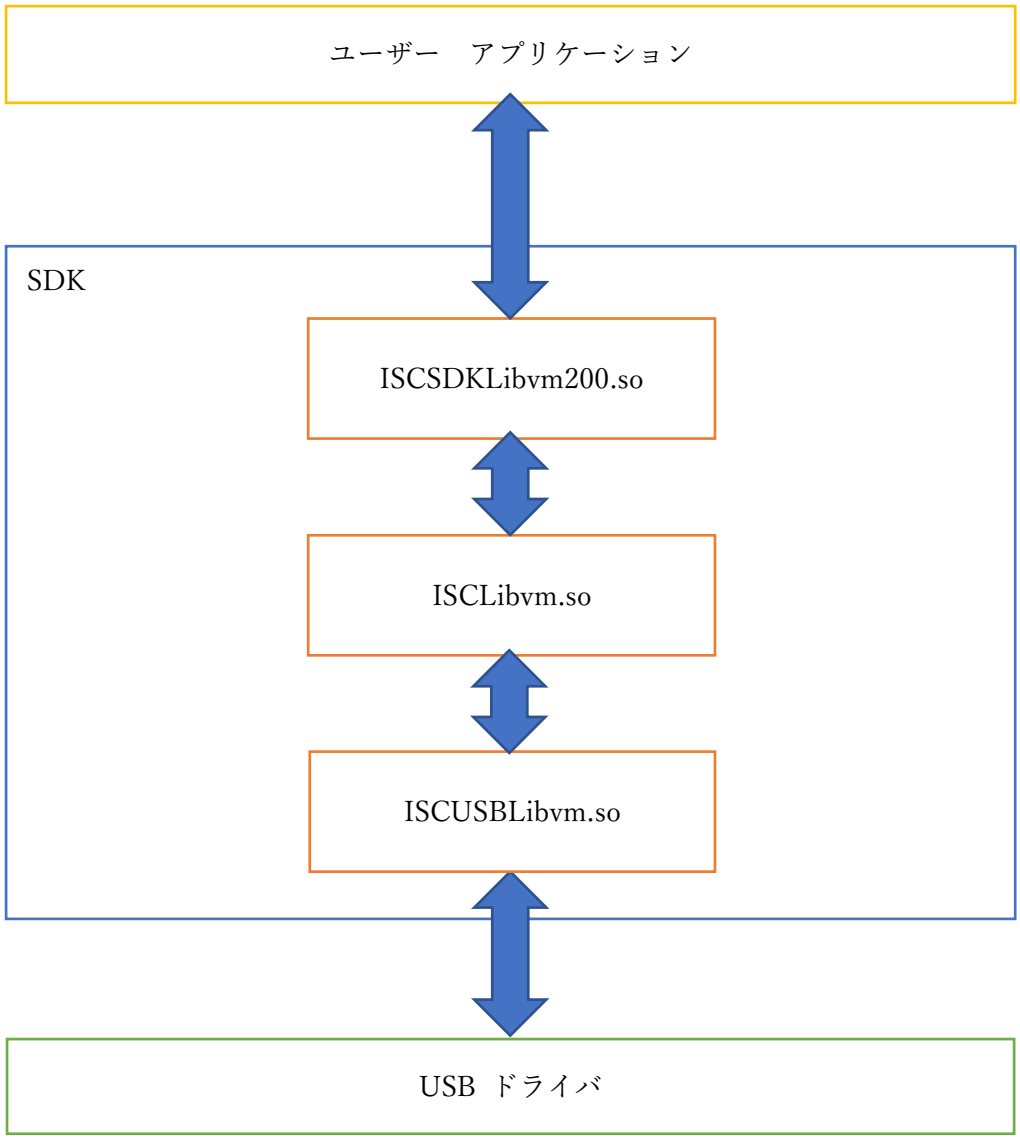
１２．Linux 版における変更点

１２．１　SDK ライブラリアーキテクチャ



※１　Linux 用 USB ドライバ
libFTDI/USB-1.0 を使用します。
インストール方法は、別紙“Linux Installation Manual.pdf”　を参照してください。

【SDK ファイル構成】


ユーザー　アプリケーション
SDK
ISCSDKLibvm200.so
ISCLibvm.so
ISCUSBLibvm.so
USB ドライバ

１３．アップグレード情報

旧バージョン (V1.6.0.0) を使用していたアプリケーションで、FPGA バージョン　73(Hex) 以降のカメラを使用するためのインターフェースが提供されます。
本インターフェースを提供するライブラリを V1.6.0.0 互換 IF と呼びます。

１３．１　V2 との相違点

V1.6.0.0 互換 IF と V2 の API には、以下の違いがあります。

関数	V1.6.0.0 互換 IF	V2
GetImage	画像幅：640	画像幅：752 ※設定により 640 に変更可能
GetDepthInfo	同上	同上
GetCameraParamInfo	V1.6.0.0 と同じパラメータ	追加パラメータ有り
GetImageSize	画像幅：640	画像幅：752
SetShutterControlMode	シングルシャッターモードの On/Off を指定	以下の 4 種類のモードを指定 ・マニュアル調整 ・シングルシャッター ・ダブルシャッター ・ダブルシャッター(合成無し)
GetShutterControlMode	シングルシャッターモードの On/Off を取得	以下の 4 種類のモードを取得 ・マニュアル調整 ・シングルシャッター ・ダブルシャッター ・ダブルシャッター(合成無し)

１３．２　旧バージョン（V1.6.0.0）との相違点

以下の関数は、追加機能のコントロールのために API が追加されます。

関数	内容
SetDoubleShutterControlMode	ダブルシャッターの On/Off を指定
GetDoubleShutterControlMose	ダブルシャッターの On/Off を指定
SetFPSMode	カメラのフレームレートを指定
GetFPSMode	カメラのフレームレートを取得

１３．３　V1.6.0.0 互換 IF では提供されない機能

以下の関数は V2 のみで提供されます。

関数	内容
SetMeasArea	露光調整測定エリアを指定
GetMeasArea	露光調整測定エリアを取得
GetImageEx	画像取得　同期タイプ

１３．４　API

V2 と差分のある関数について説明します。

【関数一覧】

関数名	概要	備考
GetCameraParamInfo	カメラ固有パラメータ取得	パラメータに違いがあります
SetShutterControlMode	露光調整モード指定	パラメータに違いがあります
GetShutterControlMode	露光調整モード取得	パラメータに違いがあります
SetDoubleShutterControlMode	ダブルシャッターモード指定	V1.6.0.0 互換 IF のみで提供します
GetDoubleShutterControlMode	ダブルシャッターモード取得	V1.6.0.0 互換 IF のみで提供します

GetCameraParamInfo

C++/C	int GetCameraParamInfo(CameraParamInfo* pParam)

Parameter	pParam	カメラ固有のパラメータ構造体です
Return Value	成功時　0 失敗時　!=0	
Remarks		
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	パラメータを取得します

	CameraParamInfo parameter = {};
	int ret = GetCameraParamInfo(&parameter);
	if(ret != 0){
	　// Error
	}
	// ・・・データ使用

【CameraParamInfo 構造体】

型	変数名	内容
float	fD_INF	無限遠値
unsigned int	nD_INF	※1
float	fBF	BF 値
float	fBaseLength	基線長
float	fViewAngle	視野角
unsigned int	nImageWidth	画像幅
unsigned int	nImageHeight	画像高さ
unsigned int	nProductNumber	製品番号
unsigned int	nSerialNumber	製品シリアル番号
unsigned int	nFPGA_Version	製品 FPGA バージョン
unsigned int	nDistanceHistValue	※1
unsigned int	nParallaxThreshold	※1

※1　未使用です。　将来のために予約されています

SetShutterControlMode

C++/C	int SetShutterControlMode(bool bUse)

Parameter	nMode	モードを指定します false：マニュアルモード true：シングルシャッターモード
Return Value	成功時　0 失敗時　!=0	
Remarks	モードの詳細は、４．露光調整の項を参照してください	
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	シングルシャッターモードを指定します

	bool mode = true;
	int ret = SetShutterControlMode (mode);
	if(ret !=0({
	　// Error
	}

GetShutterControlMode

C++/C	int GetShutterControlMode(bool* bUse)

Parameter	bUse	現在のモードを表します false：マニュアルモード true：シングルシャッターモード
Return Value	成功時　0 失敗時　!=0	
Remarks	モードの詳細は、４．露光調整の項を参照してください	
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	動作モードを取得します

	bool mode = false;
	int ret = GetShutterControlMode (&mode);
	if(ret !=0({
	　// Error
	}

SetDoubleShutterControlMode

C++/C	int SetDoubleShutterControlMode(bool bUse, bool enabledIndependentImages)

Parameter	nMode	モードを指定します false：マニュアルシャッターモード true：ダブルシャッターモード
	enabledIndependentImages	画像合成の On/Off を指定します false：Off true：On
Return Value	成功時　0 失敗時　!=0	
Remarks	モードの詳細は、４．露光調整の項を参照してください	
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	ダブルシャッターモードを指定します

	bool mode = true;
	bool enabledII = false;
	int ret = SetDoubleShutterControlMode (mode, enabledII);
	if(ret !=0({
	　// Error
	}

GetDoubleShutterControlMode

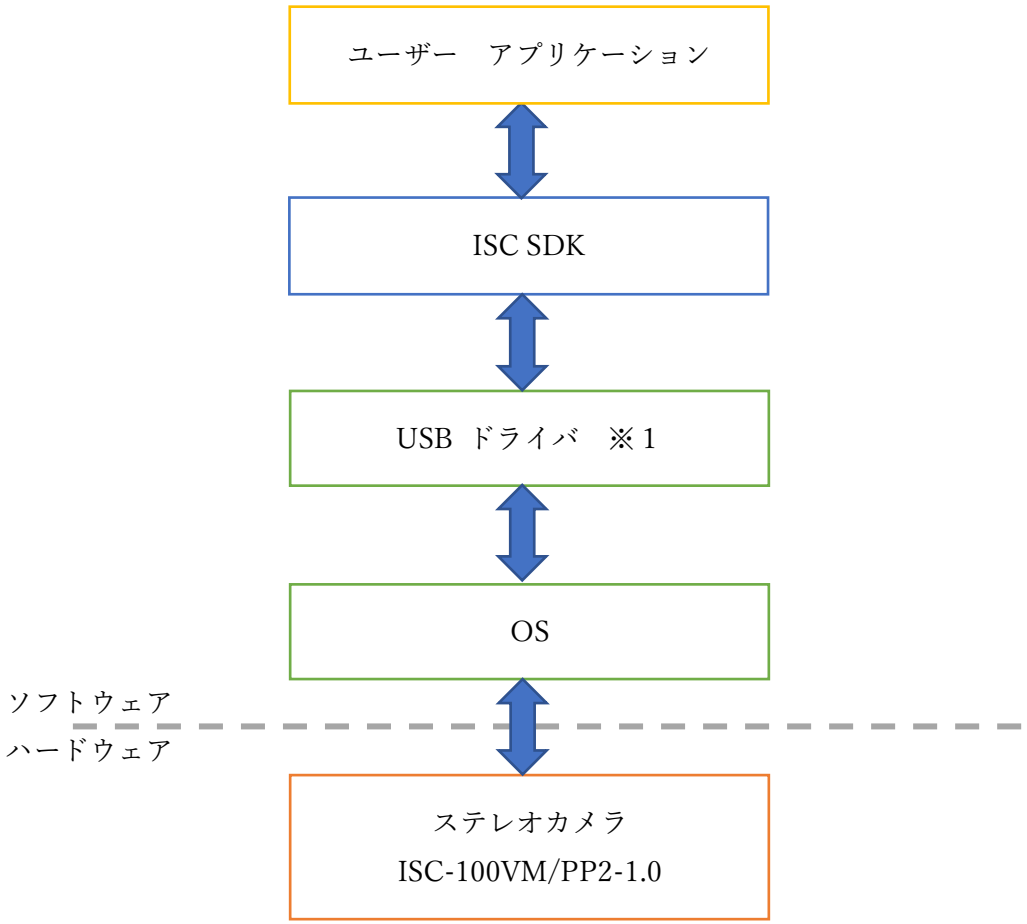
C++/C	int GetDoubleShutterControlMode(bool* bUse, bool* enabledIndependentImages)

Parameter	bUse	現在のモードを表します false：マニュアルモード又は シングルシャッターモード true：ダブルシャッターモード
	enabledIndependentImages	現在の画像合成の On/Off を表します false：Off true：On
Return Value	成功時　0 失敗時　!=0	
Remarks	モードの詳細は、４．露光調整の項を参照してください	
See also	失敗時のコードは、エラーコード一覧を参照してください	

Example	動作モードを取得します

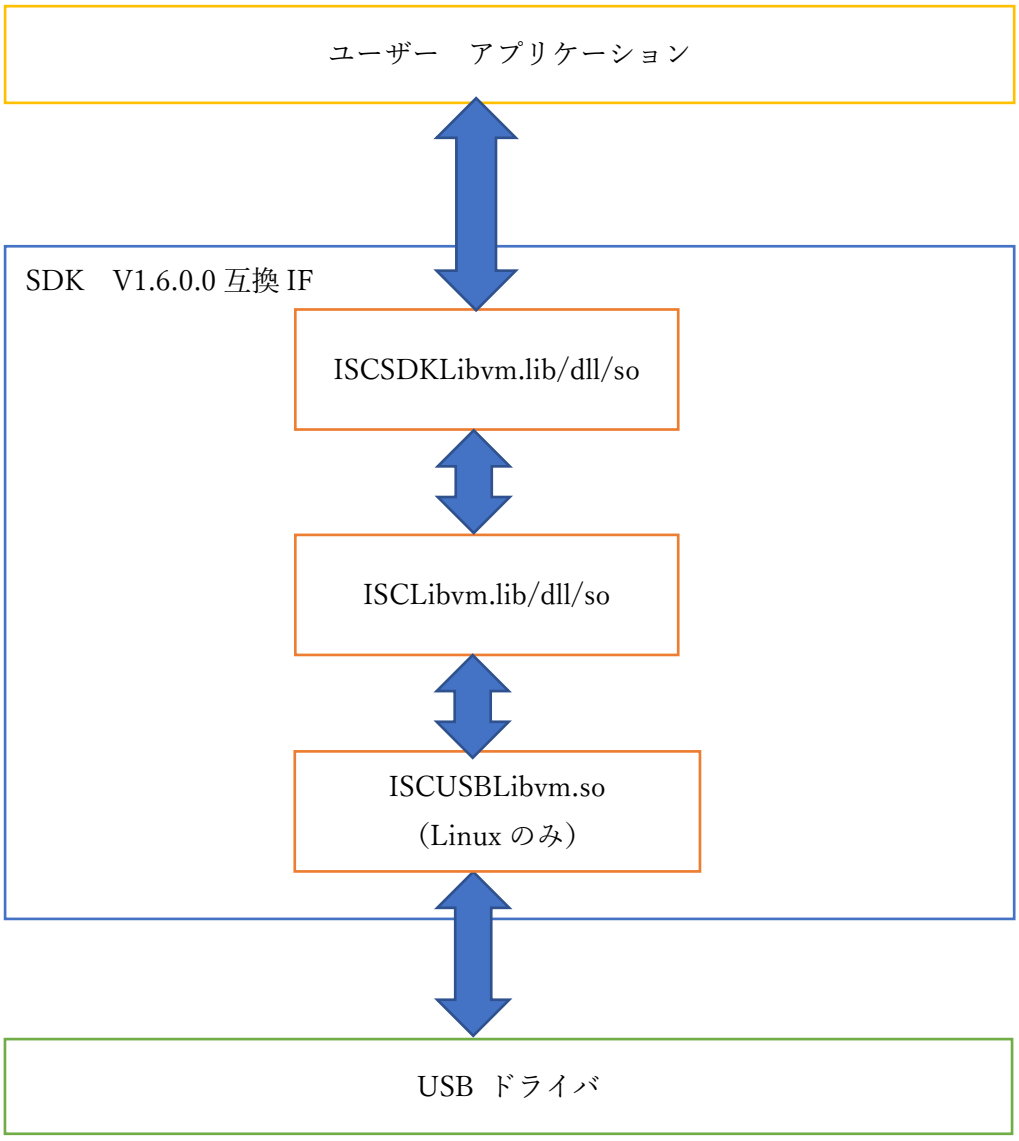
	bool mode = false;
	bool c = false;
	int ret = GetDoubleShutterControlMode (&mode, & enabledII);
	if(ret !=0({
	　// Error
	}

１３．５　SDK ライブラリアーキテクチャ



※１　USB ドライバは、V2 と同一です

【SDK ファイル構成】
V1.6.0.0 互換 IF では、一部で使用する DLL ファイルが異なります。



１３．６　画像データフォーマット

（１）Windows 版

受信データは、画像の左下画素より格納されています。

画像バッファー（640x480 のケース）

0	1				638	639
306560						307199

（２）Linux 版

受信データは、画像の右下画素より格納されています。

画像バッファー（640x480）

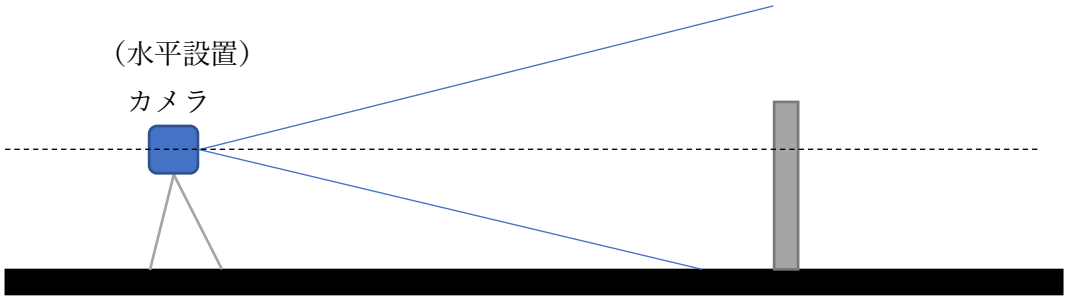
0	1			638	639
306560					307199

Appendix A　視差の使い方

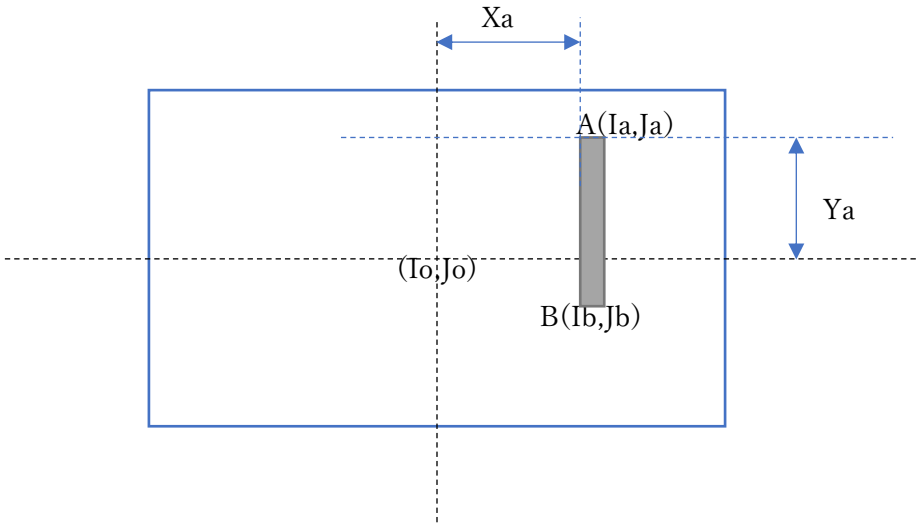
視差データを距離に変換するための方法を説明します。

１０．１　水平設置における距離の取得

カメラ及び対象物の設置が下図のようなケースにおける距離の取得方法を説明します。



カメラでは、下図のようなデータが取得されたとします。



ステレオカメラから取得できるパラメータより以下を使用します。
（GetCameraParamInfo で取得します）

① BF　　：補正校正時に決定されるカメラ固有のパラメータ
② DINF　　：補正校正時に決定されるカメラ固有のパラメータ
③ B　　：基線長

また、画面中心の座標を(Io,Jo)とします。

（画像のサイズが 640×480 であれば、(Io,Jo)=(320,240)となります）

この時 A における視差値を Da とすると、A までの距離 Za(m)は以下の式で求めることができます。

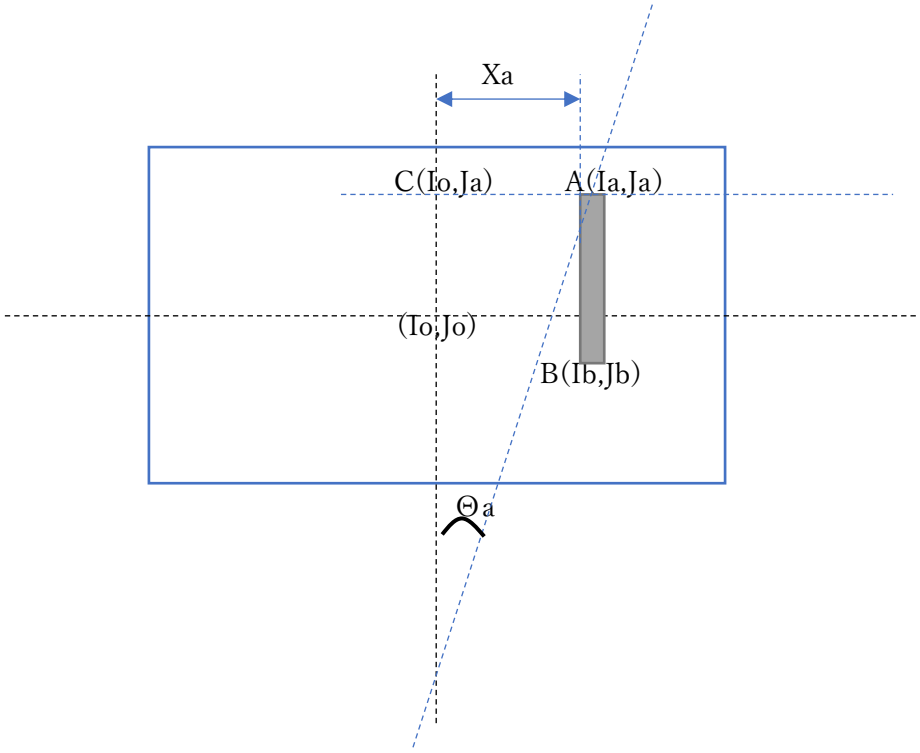
$$Za(m) = \frac{BF}{Da - DINF}$$

中心からの距離、Xa(m)及び Ya(m)は以下となります。

$$Xa(m) = \frac{B \times (Ia - Io)}{Da - DINF}$$

$$Ya(m) = \frac{B \times (Ja - Jo)}{Da - DINF}$$

中心からの角度の求め方を説明します。

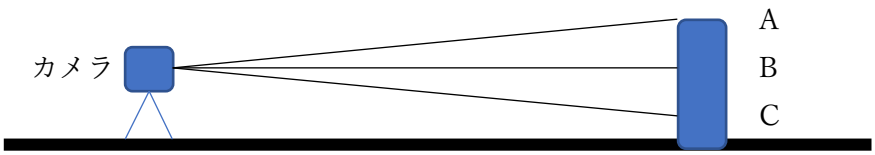


画面中心（C）から A 点までの角度は、以下で求めることができます。

$$\theta a(^\circ) = \tan^{-1}\frac{Xa}{Za}$$

※　ステレオカメラにおける距離
ステレオカメラでは、距離を平面として取得します。

例えば、下図のような構成においては、各 A,B,C までの距離 Za と Zb 及び Zc は等しくなります。(Za＝Zb=Zc)



Appendix B　ライセンス

本ライブラリで使用する外部ライブラリのライセンスを表記します

The C library "libftdi1" is distributed under the
GNU Library General Public License version 2.

A copy of the GNU Library General Public License (LGPL) is included
in this distribution, in the file COPYING.LIB.

The C++ wrapper "ftdipp1" is distributed under the GNU General
Public License version 2 (with a special exception described below).

A copy of the GNU General Public License (GPL) is included
in this distribution, in the file COPYING.GPL.

As a special exception, if other files instantiate templates or use macros
or inline functions from this file, or you compile this file and link it
with other works to produce a work based on this file, this file
does not by itself cause the resulting work to be covered
by the GNU General Public License.

However the source code for this file must still be made available
in accordance with section (3) of the GNU General Public License.

This exception does not invalidate any other reasons why a work based
on this file might be covered by the GNU General Public License.

改版履歴

Rev	Date	Content
1.0.0	2017/09/30	初版
1.1.0	2017/12/07	調整用関数追加 SetAutoCalibration() GetAutoCalibration() エラーコード -18 を追加
1.2.0	2018/11/05	HDR 設定関数追加 SetHDRMode() SetHiResolutionMode()
1.3.0	2019/01/15	カメラ切断検知追加(ErrorCode=-4) 画像取得無し追加(ErrorCode=-102) Calibration 情報保存/設定機能追加
1.3.1β	2019/11/14	ダブルシャッター追記
2.0.0	2020/4/30	製品 Version 1.0 に対応する SDK V2 リリースに合わせ改訂
2.0.1	2020/10/5	誤記訂正
2.1.0	2020/10/30	SDK バージョン　2.3.0 対応 　GetImageEx 関数追加 エラーコード追加 エラー処理の説明を追加
2.1.1	2021/1/12	Exposure 及び Gain の反映に対する注意事項を追加