

画像認識ユニット

NVP-Ax230SDK

Software Development Kit

Fine Vision Processor

コマンドリファレンス

maxell

マクセルシステムテック株式会社

はじめに

このたびは、「画像認識ユニット：NVP-Ax230CL/235CL」および「画像処理ソフトウェア開発キット：NVP-Ax230SDK」をお買い上げいただきまして、誠にありがとうございます。

本マニュアルは、NVP-Ax230SDKの「コマンドリファレンス」について記載しております。

ご注意

- システムの構築やプログラム作成などの操作を行う前に、本マニュアルの記載内容をよく読み、書かれている指示や注意を十分理解して下さい。誤った操作によりシステムの故障が発生することがあります。
- 本マニュアルの記載内容について理解できない内容、疑問点または不明点がございましたら、弊社営業窓口までお知らせ下さい。
また、弊社ホームページのお問い合わせのページからも受け付けておりますのでご利用ください。
<http://www.systemtech.maxell.co.jp/solution/vp/>
- お客様の誤った操作に起因する、事故発生や損害につきましては、弊社は責任を負いかねますのでご了承ください。
- 弊社提供のハードウェアおよびソフトウェアを無断で改造しないでください。この場合の品質および安全につきましては、弊社は責任を負いかねますのでご了承ください。

画像処理コマンドリファレンス

●説明形式

本章で使用する説明形式について説明します。

本文中にボード／ボード番号の表現がありますが、それぞれユニット／ユニット番号と読み替えてください。

- (1) C言語API
C言語のAPI呼び出し形式を表します。
- (2) パラメータ
APIのパラメータ（引数）を表します。
- (3) 対応画面タイプ／画面データタイプ
APIで使用する画面の属性とデータタイプを表します。

RGB画面を処理画面とする場合、R、G、Bの3面を1組にして処理するコマンドと1画面だけを処理するコマンドがあります。表中のRGBは3面を1組とした場合、RGB. R、RGB. G、RGB. Bはそれぞれの画面を示します。

ただし、RGB画面のタイプはRGB. R画面タイプとなります。

画面タイプ	内容
Y	モノクロ画面（Y画面）
RGB	R G B画面（3面1組）
RGB. R	R G BのR成分の画面
RGB. G	R G BのG成分の画面
RGB. B	R G BのB成分の画面
RGB16	R G B画面 16ビット構成

画面データタイプ	内容
U	符号なし8ビット濃淡画面
S	符号付8ビット濃淡画面
B	2値画面

判定	内容
○	対応タイプ
△Y	対応タイプ（Y画面とみなして処理）
△R	対応タイプ（R画面だけを処理）
△U	対応タイプ（符号なし8ビット濃淡画面とみなして処理）
△S	対応タイプ（符号付8ビット濃淡画面とみなして処理）
△B1	対応タイプ（2値画面とみなして処理） 符号ビット（MSB）を演算に使用
△B2	対応タイプ（2値画面とみなして処理） 符号拡張したデータを演算に使用
×	非対応タイプ（エラーチェック有り）
◆	データタイプが上書きされて常にこのタイプになる
◇1／◇2／ ◇S／◇	データタイプが上書きされてこのタイプになる場合がある （ソース1依存／ソース2依存／システムデータタイプ依存／ ソース1画面データタイプが2値ならばソース2画面データタイプ。 他はソース1画面データタイプ）
□	デスティネーションのデータタイプは変更なし
—	対象外

- (4) リターンパラメータ
A P I の戻り値及び値を格納したパラメータを表します。
- (5) エラーコード
A P I でエラーが発生した時にエラー情報テーブルに設定される値を表します。
エラー情報の読み出しは『ReadIPErrorTable()』コマンドで行います。
 - ・エラーコード説明の用語の補足を下記に示します。
 - I M P : 画像処理モジュール
 - V I N : 映像入力モジュール
 - D U : 映像出力モジュール
- (6) 機能
A P I の機能を表します。
- (7) 制限事項
使用上の制限について記載します。
- (8) サンプルコード
画像処理コマンドの使用例を表します。

●注意事項

画像処理ライブラリのA P I を使用するにあたり、注意事項を説明します。

- (1) 画面番号
A P I のパラメータに設定するソース画面番号とデスティネーション画面番号には、同一の画面番号を設定しないで下さい。同一の画面番号を指定した場合、動作は保証しません。
- (2) 画面データタイプ
画面データタイプが想定外（対応画面タイプ／画面データタイプが○以外）の場合、エラーにはなりません、期待する処理結果を得られない場合があります。
- (3) ウィンドウ
ウィンドウ無効（『DisableIPWindow()』コマンド実行）の場合、全てのウィンドウが現在のビデオ画面サイズに設定されます。ビデオ画面サイズは『InitIP()』コマンドにより初期値512×480に設定されます。ビデオ画面サイズの変更は『SetVideoFrame()』コマンドで行います。

μITRONは、“Micro Industrial TRON”の略称です。
TRONは、“The Real time Operating system Nucleus”の略称です。

Microsoft, Windows, WindowsXP, VisualC+++は、米国Microsoft Corporationの米国およびその他の国における登録商標です。
その他、本マニュアルに記載されている会社名および製品名は、各社の商標または登録商標です。

画像処理コマンド一覧

次に画像処理コマンドの一覧を記載します。

デバイスID指定について、説明します。

- ・デバイスID指定が×のコマンドは、デバイスID (DEVID devID)は指定できません。
- ・デバイスID指定が○のコマンドは、デバイスID (DEVID devID)を指定することができます。
デバイスIDを指定する場合は、引数の先頭に番号を指定してください。

記述例：

デバイスID指定なし ret = SetTriggerMode(enum TrigerMode mode , int speed);

デバイスID指定あり ret = SetTriggerMode(DEVID devID , enum TrigerMode mode , int speed);

なお、プログラムには、以下の条件がありますので注意してください。

” ボード番号を指定できないコマンド” を除き、” ボード番号を指定できるコマンド” の” ボード番号指定あり” と、” ボード指定なし” のコマンドの混在はできません。

以下は不可のプログラム例：

```
ret = SelectCamera( enum CameraID id , enum CameraType type );    //デバイスID指定なし
ret = SetTriggerMode( enum TrigerMode mode , int speed );        //デバイスID指定なし
ret = GetCamera( DEVID devID , int ImgID );                      //デバイスID指定あり
```

1. 初期化コマンド

1.1 デバイスID無しリモートコマンドのAPI初期化

NO	コマンド名	機能	デバイスID指定
1	StartIP	デバイスID無しリモートコマンドの使用開始	×
2	StopIP	デバイスID無しリモートコマンドの使用停止	×
3	ActiveIP	ボード番号の切替え	×

1.2 デバイスID付きリモートコマンドのAPI初期化

NO	コマンド名	機能	デバイスID指定
1	OpenIPDev	デバイスID付きリモートコマンドの使用開始	×
2	CloseIPDev	デバイスID無しリモートコマンドの使用停止	○

1.3 API初期化サポートコマンド

NO	コマンド名	機能	デバイスID指定
1	EnumAttachIPDev	装着ボードの列挙	×
2	ResetIP	画像処理ボードのリセット	×
3	GetIPLastError	エラーコード取得	×

2. 画像処理コマンド

2.1 システム制御コマンド

NO	コマンド名	機能	デバイスID 指定
1	InitIP	画像処理システムの初期化	○
2	InitIPExt	画像処理システムの初期化（電源ON時）	○
3	EnableIPErrorMessage	エラーメッセージの出力の許可	×
4	DisableIPErrorMessage	エラーメッセージの出力の禁止	×
5	ReadIPErrorTable	コマンドエラー情報の取得	×
6	ClearIPError	エラー情報クリア	×
7	SetIPDataType	画像処理動作データタイプ設定	○
8	GetIPVersionInfo	バージョン情報の取得	○

2.2 画像メモリ領域管理コマンド

NO	コマンド名	機能	デバイスID 指定
1	AllocImg	画像メモリ領域確保	○
2	AllocComponentImg	同時取込み用画像メモリ領域確保	○
3	FreeImg	画像メモリ領域解放（指定画面）	○
4	FreeAllImg	画像メモリ領域解放（全画面）	○
5	ReadImgTable	画像メモリ管理テーブル読出し	○
6	GetElementImgID	コンポーネント画面番号取出し	○
7	ChangeImgDataType	画面データタイプ変更	○
8	SetWindow	ウィンドウ設定	○
9	SetAllWindows	ウィンドウ設定（全種）	○
10	ResetAllWindows	ウィンドウリセット	○
11	EnableIPWindow	ウィンドウ有効化	○
12	DisableIPWindow	ウィンドウ無効化	○
13	ReadWindow	ウィンドウ設定座標読出し	○
14	AllocRGBImg	画像メモリ領域確保（コンポーネントRGB画面）	○
15	AllocRGB16Img	画像メモリ領域確保（16bit RGB画面）	○
16	GetGImgID	G画面番号抽出	○
17	GetBImgID	B画面番号抽出	○

2.3 映像入力コマンド

映像入力手順(単一・連続)

NO	コマンド名	機能	デバイスID 指定
1	ActiveVideoPort	ビデオポート選択	○
2	SetVideoFrame	映像入力画面設定	○
3	SelectCamera	カメラ番号とカメラタイプの選択	○
4	GetCamera	カメラ映像入力	○
5	SetVFDelay	映像入力画面の遅延サイズ設定	○
6	SetTriggerMode	カメラ映像入力トリガモードの設定	○
7	SetStrobeMode	カメラ映像入力ストロボの設定	○
8	SetPartialMode	カメラ映像入力パーシャルモードの設定	○
9	SetVideoLUTMode	ビデオ入力LUTの設定	○
10	GetSceneCounter	連続入力シーン番号カウンタ取得	○
11	SetSceneCounter	連続入力シーン番号カウンタ設定	○
12	SetCameraData	カメラデータの設定	○
13	GetCameraData	カメラデータの取得	○
14	LoadCameraFile	カメラデータファイルのロード	○
15	CaptureContinuous	連続映像入力モードの設定と起動	○
16	StopCaptureContinuous	連続キャプチャ停止	○
17	GetCameraReqScene	連続入力シーン番号指定付きカメラ映像入力	○
18	GetCameraResScene	連続入力シーン番号出力付きカメラ映像入力	○

2.4 映像表示コマンド

NO	コマンド名	機能	デバイスID 指定
1	SetConfigDisp	表示画面の構成制御設定	○
2	SetDispPalette	表示パレットの設定	○
3	SelectDisp	映像出力形式の選択	○
4	DispCamera	カメラ映像表示	○
5	DisableDisp	カメラ映像表示終了	○
6	DispImg	画像メモリ表示	○
7	NoDisp	表示終了	○
8	DispOverlap	画像メモリのオーバーレイ表示	○
9	DisableOverlap	画像メモリのオーバーレイ表示終了	○
10	CombineRGB	コンポーネントRGB画面から16bitRGB画面の合成	○
11	CombineRGBEx	コンポーネントRGB画面から16bitRGB画面の合成 (拡張)	○

2.5 画像クリアコマンド

NO	コマンド名	機能	デバイスID 指定
1	IP_ClearAllImg	画像メモリクリア (全チャネル)	○
2	IP_ClearImg	画像メモリクリア (指定画面)	○
3	IP_Const	定数発生	○

2.6 画像転送・アフィン変換コマンド

NO	コマンド名	機能	デバイスID 指定
1	IP_Copy	転送	○
2	IP_Zoom	ズーム	○
3	IP_ZoomExt	ズーム（縦横任意倍率）	○
4	IP_ZoomOut	ズームアウト	○
5	IP_ZoomOutExt	ズームアウト（縦横任意倍率）	○
6	IP_ZoomIn	ズームイン	○
7	IP_ZoomInExt	ズームイン（縦横任意倍率）	○
8	IP_ZoomS	シフト付きズーム	○
9	IP_Shift	シフト	○
10	IP_Rotate	回転	○

2.7 2値化コマンド

NO	コマンド名	機能	デバイスID 指定
1	IP_Binarize	2 値化	○
2	IP_BinarizeExt	範囲・反転付き 2 値化	○

2.8 画素変換コマンド

NO	コマンド名	機能	デバイスID 指定
1	IP_Invert	論理反転	○
2	IP_Minus	符号反転	○
3	IP_Abs	絶対値	○
4	IP_AddConst	定数加算	○
5	IP_SubConst	定数減算	○
6	IP_SubConstAbs	定数減算絶対値	○
7	IP_MultConst	定数乗算	○
8	IP_MinConst	定数比較 M i n	○
9	IP_MaxConst	定数比較 M a x	○
10	WriteConvertLUT	濃度変換データ設定	○
11	IP_ConvertLUT	濃度変換	○
12	IP_ShiftDown	定数シフトダウン	○
13	IP_ShiftUp	定数シフトアップ	○

2.9 画像間算術演算コマンド

NO	コマンド名	機能	デバイスID 指定
1	IP_Add	加算	○
2	IP_Sub	減算	○
3	IP_SubAbs	減算絶対値	○
4	IP_Comb	係数付加算	○
5	IP_CombAbs	係数付加算絶対値	○
6	IP_Mult	乗算	○
7	IP_Average	平均	○
8	IP_Min	比較 M i n	○
9	IP_Max	比較 M a x	○
10	IP_SubConstAbsAdd	定数減算絶対値和	○
11	IP_SubConstMultAdd	定数減算自乗和	○
12	IP_SubConstMult	定数減算積	○
13	IP_CombDrop	係数付加算（切り捨て）	○

2.10 画像間論理演算コマンド

NO	コマンド名	機能	デバイスID 指定
1	IP_And	論理積	○
2	IP_Or	論理和	○
3	IP_Xor	排他的論理和	○
4	IP_InvertAnd	否定論理積	○
5	IP_InvertOr	否定論理和	○
6	IP_Xnor	排他的否定論理和	○

2.11 2値画像形状変換コマンド

NO	コマンド名	機能	デバイスID 指定
1	IP_PickNoise4	2 値画像ノイズ除去（4 連結）	○
2	IP_PickNoise8	2 値画像ノイズ除去（8 連結）	○
3	IP_Outline4	2 値画像輪郭抽出（4 連結）	○
4	IP_Outline8	2 値画像輪郭抽出（8 連結）	○
5	IP_Dilation4	2 値画像膨張（4 連結）	○
6	IP_Dilation8	2 値画像膨張（8 連結）	○
7	IP_Erosion4	2 値画像収縮（4 連結）	○
8	IP_Erosion8	2 値画像収縮（8 連結）	○
9	IP_Thin4	2 値画像細線化（4 連結）	○
10	IP_Thin8	2 値画像細線化（8 連結）	○
11	IP_Shrink4	2 値画像縮退化（4 連結）	○
12	IP_Shrink8	2 値画像縮退化（8 連結）	○

2.12 コンボリレーションコマンド

N0	コマンド名	機能	デバイスID 指定
1	IP_SmoothFLT	平滑化	○
2	IP_EdgeFLT	濃淡画像輪郭強調	○
3	IP_EdgeFLTAbs	濃淡画像輪郭強調（絶対値）	○
4	IP_Lapl4FLT	ラプラシアン（4連結）	○
5	IP_Lapl8FLT	ラプラシアン（8連結）	○
6	IP_Lapl4FLTAbs	ラプラシアン（4連結・絶対値）	○
7	IP_Lapl8FLTAbs	ラプラシアン（8連結・絶対値）	○
8	IP_LineFLT	ラインフィルタ	○
9	IP_LineFLTAbs	ラインフィルタ（絶対値）	○

2.13 ミニ／マックスフィルタコマンド

N0	コマンド名	機能	デバイスID 指定
1	IP_MinFLT	局所最小値フィルタ	○
2	IP_MinFLT4	局所最小値フィルタ（4連結）	○
3	IP_MinFLT8	局所最小値フィルタ（8連結）	○
4	IP_MaxFLT	局所最大値フィルタ	○
5	IP_MaxFLT4	局所最大値フィルタ（4連結）	○
6	IP_MaxFLT8	局所最大値フィルタ（8連結）	○
7	IP_LineMinFLT	ライン局所最小値フィルタ	○
8	IP_LineMaxFLT	ライン局所最大値フィルタ	○

2.14 ランクフィルタコマンド

N0	コマンド名	機能	デバイスID 指定
1	IP_RankFLT	局所ランクフィルタ	○
2	IP_Rank4FLT	局所ランクフィルタ（4連結）	○
3	IP_Rank8FLT	局所ランクフィルタ（8連結）	○
4	IP_MedFLT	局所メディアンフィルタ	○
5	IP_Med4FLT	局所メディアンフィルタ（4連結）	○
6	IP_Med8FLT	局所メディアンフィルタ（8連結）	○

2.15 ラベリングコマンド

NO	コマンド名	機能	デバイスID 指定
1	IP_Label4	ラベリング（4連結）	○
2	IP_Label8	ラベリング（8連結）	○
3	IP_Label4withAreaFLT	面積フィルタ付ラベリング（4連結）	○
4	IP_Label8withAreaFLT	面積フィルタ付ラベリング（8連結）	○
5	IP_Label4withAreaFLTSort	面積フィルタ付ラベリング（面積ソート4連結）	○
6	IP_Label8withAreaFLTSort	面積フィルタ付ラベリング（面積ソート8連結）	○
7	IP_ExtractL0RegionX	ラベル毎領域X座標抽出（Min/Max X座標）	○
8	IP_ExtractL0RegionY	ラベル毎領域Y座標抽出（Min/Max Y座標）	○
9	IP_ExtractL0Area IP_ExtractL0AreaExt	ラベル毎面積抽出	○
10	IP_ExtractL0Gravity	ラベル毎重心座標抽出	○

2.16 濃淡画像特徴量抽出コマンド

NO	コマンド名	機能	デバイスID 指定
1	IP_ExtractG0Features	濃淡画像基本特徴量抽出	○
2	IP_Histogram	濃度ヒストグラム（32ビット）	○
3	IP_HistogramShort	濃度ヒストグラム（16ビット）	○
4	IP_HistogramFeatures	濃度ヒストグラム特徴量抽出	○
5	IP_ProjectG0	X/Y軸への濃度累積値投影	○
6	IP_ProjectG0onX	X軸への濃淡累積値投影	○
7	IP_ProjectG0onY	Y軸への濃淡累積値投影	○
8	IP_ProjectG0MaxValue	X/Y軸への最大濃度投影	○
9	IP_ProjectG0MinValue	X/Y軸への最小濃度投影	○
10	IP_ProjectBlockG0	領域毎濃淡画像濃度累積	○
11	IP_ProjectBlockG0MinMaxValue	領域毎濃淡画像最大/最小濃度抽出	○
12	IP_ProjectLabelG0	ラベル毎濃淡画像濃度累積	○
13	IP_ProjectLabelG0MinMaxValue	ラベル毎濃淡画像最小/最大濃度値抽出	○

2.17 2値画像特徴量抽出コマンド

NO	コマンド名	機能	デバイスID 指定
1	IP_ExtractB0Area	2値画像累積値抽出	○
2	IP_ExtractB0Features	2値画像基本特徴量抽出	○
3	IP_ProjectB0	X/Y軸への投影	○
4	IP_ProjectB0RegionX	領域X座標抽出（Y軸へのMin&Max X投影）	○
5	IP_ProjectB0RegionY	領域Y座標抽出（X軸へのMin&Max Y投影）	○
6	IP_ProjectBlockB0	領域毎2値画像濃度累積	○

2.18 画像メモリアクセスコマンド

NO	コマンド名	機能	デバイスID 指定
1	OpenImg	画像メモリアクセス可能	○
2	OpenImgExt	画像メモリアクセス可能（指定画面）	○
3	CloseImg	画像メモリアクセス禁止	○
4	ReadImg	画像メモリ読出し（矩形領域）	○
5	WriteImg	画像メモリ書込み（矩形領域）	○
6	SetPixelPointer	画像メモリポインタ設定	○
7	ReadPixel	画像メモリ1画素読出し	○
8	WritePixel	画像メモリ1画素書込み	○
9	ReadPixelContinue	画像メモリ連続1画素読出し	○
10	WritePixelContinue	画像メモリ連続1画素書込み	○
11	RefreshImg	画像メモリのリフレッシュ	○
12	OpenImgDirect	画像メモリ直接アクセス可能	○
13	CloseImgDirect	画像メモリ直接アクセス不可	○

2.19 2値パイプラインフィルタコマンド

NO	コマンド名	機能	デバイスID 指定
1	SetTrsPipelineFLTMode	パイプラインフィルタのモード設定	○
2	IP_TrspipelineFLT	2値パイプラインフィルタの実行	○

2.20 パイプライン制御コマンド

NO	コマンド名	機能	デバイスID 指定
1	EnablePipeline	パイプラインモードの有効化	○
2	DisablePipeline	パイプラインモードの無効化	○

2.21 2値マッチングフィルタコマンド

NO	コマンド名	機能	デバイスID 指定
1	SetBinMatchTemplate	2値画像マッチングフィルタテンプレート登録	○
2	IP_BinMatchFLT	2値画像マッチングフィルタ実行	○

2.22 正規化関連コマンド

正規化関連コマンド使用手順

NO	コマンド名	機能	デバイスID 指定
1	SetCorrMode	相関サーチモード設定	○
2	DisableCorrMask	テンプレートマスク無効	○
3	EnableCorrMask	テンプレートマスク有効	○
4	SetCorrBreakThr	打切り閾値設定	○
5	DisableCorrBreak	相関演算途中打切りの無効化	○
6	EnableCorrBreak	相関演算途中打切りの有効化	○
7	SetCorrControl	正規化相関制御	○
8	SetCorrTemplate	正規化相関テンプレート登録	○
9	SetCorrTemplateExt	正規化相関テンプレート登録（間引き付き）	○
10	IP_Corr	正規化相関	○
11	IP_CorrPecise	正規化相関（高精度位置決め）	○

2.23 グラフィックスコマンド

グラフィックスコマンド使用手順

NO	コマンド名	機能	デバイスID 指定
1	SetDrawMode	描画面面設定	○
2	SetStringAttributes	文字の属性設定	○
3	RefreshGraphics	パターン作成終了	○
4	DrawString	文字列描画	○
5	DrawLine	直線描画	○
6	DrawSegments	直線描画（複数）	○
7	DrawLines	折れ線描画	○
8	DrawRectangle	矩形描画	○
9	DrawPolygon	多角形描画	○
10	DrawArc	円弧描画	○
11	FillRectangle	矩形描画（塗りつぶし）	○
12	FillPolygon	多角形描画（塗りつぶし）	○

2.24 画像ファイリングコマンド

NO	コマンド名	機能	デバイスID 指定
1	LoadBMPFile	BMP ファイル画像データ読出し	○
2	SaveBMPFile	BMP ファイル画像データ書込み	○

2.25 構成制御コマンド

NO	コマンド名	機能	デバイスID 指定
1	SetConfigCamera	映像入力構成制御設定	○
2	ExitIPCcamera	カメラ映像入力プログラム終了	○
3	SetConfigView	画像メモリ表示構成制御設定	○
4	ExitIPView	画像メモリ表示プログラム終了	○

2.26 ランレングス・ラベリングコマンド

NO	コマンド名	機能	デバイスID 指定
1	IP_Label14byRL	ランレングス・ラベリング処理（4 連結）	○
2	IP_Label18byRL	ランレングス・ラベリング処理（8 連結）	○
3	IP_Label14byRLwithAreaFLT	面積フィルタ付きランレングス・ラベリング処理（4 連結）	○
4	IP_Label18byRLwithAreaFLT	面積フィルタ付きランレングス・ラベリング処理（8 連結）	○
5	IP_Label14byRLwithAreaFLTSort	面積フィルタ付きランレングス・ラベリング処理 （ソート・4 連結）	○
6	IP_Label18byRLwithAreaFLTSort	面積フィルタ付きランレングス・ラベリング処理 （ソート・8 連結）	○
7	IP_Label14byRLExt	ランレングス・ラベリング処理（4 連結・拡張）	○
8	IP_Label18byRLExt	ランレングス・ラベリング処理（8 連結・拡張）	○
9	IP_Label14byRLwithAreaFLTExt	面積フィルタ付きランレングス・ラベリング処理 （4 連結・拡張）	○
10	IP_Label18byRLwithAreaFLTExt	面積フィルタ付きランレングス・ラベリング処理 （8 連結・拡張）	○
11	IP_Label14byRLwithAreaFLTSortExt	面積フィルタ付きランレングス・ラベリング処理 （ソート・4 連結・拡張）	○
12	IP_Label18byRLwithAreaFLTSortExt	面積フィルタ付きランレングス・ラベリング処理 （ソート・8 連結・拡張）	○
13	IP_LabelCombine	統合ラベリング	○

2.27 2値化閾値算出支援コマンド

N0	コマンド名	機能	デバイスID 指定
1	HistAnalyze	判別分析法	○

2.28 拡張コンボリユーションコマンド

N0	コマンド名	機能	デバイスID 指定
1	IP_SmoothFLT5x5	平滑化 (5 × 5)	○
2	IP_SmoothFLT7x7	平滑化 (7 × 7)	○
3	IP_EdgeFLT5x5	濃淡画像輪郭強調 (5 × 5)	○
4	IP_EdgeFLT7x7	濃淡画像輪郭強調 (7 × 7)	○
5	IP_MinFLT5x5	局所最小値フィルタ (5 × 5)	○
6	IP_MaxFLT5x5	局所最大値フィルタ (5 × 5)	○
7	IP_SmoothFLTExt	平滑化(拡張版)	○
8	IP_EdgeFLTAbsExt	輪郭強調フィルタ・絶対値(拡張版)	○
9	IP_SmoothFLT5x5Ext	拡張平滑化 (5 × 5)	○
10	IP_SmoothFLT7x7Ext	拡張平滑化 (7 × 7)	○
11	IP_EdgeFLT5x5Ext	画像輪郭強調 (5 × 5)	○
12	IP_EdgeFLT7x7Ext	画像輪郭強調 (7 × 7)	○

2.29 拡張画像処理コマンド

N0	コマンド名	機能	デバイスID 指定
1	IP_Sobel	エッジソーベル	○
2	IP_SobelBinarize	エッジソーベル (2 値化)	○
3	IP_RegisterLUT	濃度変換テーブルの登録	○
4	IP_Prewitt	輪郭拡張 (プレビット)	○
5	IP_PrewittBinarize	輪郭拡張 2 値化 (プレビット 2 値化)	○

2.30 線分化コマンド

N0	コマンド名	機能	デバイスID 指定
1	ExtractPolyline	線分列抽出	○
2	PolyArea	線分列から面積抽出	○
3	PolyPerim	線分列からの周囲長抽出	○
4	PolyGrav	線分列から重心抽出	○
5	PolyFeatures	線分列から形状特徴量抽出	○

2.31 2値画像の穴埋めコマンド

NO	コマンド名	機能	デバイスID 指定
1	IP_FillHole	2 値画像穴埋め	○
2	IP_FillHoleExt	2 値画像穴埋め(拡張)	○

2.32 RGLUT変換コマンド

NO	コマンド名	機能	デバイスID 指定
1	CreateRGLUT	RGLUTオブジェクトの生成	○
2	DeleteRGLUT	RGLUTオブジェクトの削除	○
3	OpenRGLUT	RGLUTオブジェクトの使用開始	○
4	CloseRGLUT	RGLUTオブジェクト使用終了	○
5	IP_ConvertRGLUT	RGLUT変換の実行	○
6	IP_ConvertRGLUTEx	RGLUT変換の実行(拡張)	○

2.33 RAW/RGB変換コマンド

NO	コマンド名	機能	デバイスID 指定
1	CreateConvertRAWData	RAW/RGB変換データ生成	○
2	DeleteConvertRAWData	RAW/RGB変換データ削除	○
3	SetRAWtoRGBGain	RAW/RGB変換ゲイン設定	○
4	IP_ConvertRAWtoRGB	RAWデータからRGBデータへの変換	○
5	IP_ConvertRAWtoRGBEx	RAWデータからRGBデータへの変換(拡張)	○

2.34 IMR(歪み補正)コマンド

NO	コマンド名	機能	デバイスID 指定
1	imrcmd_CreateObj	歪み補正オブジェクト作成 (メモリ確保)	○
2	imrcmd_DeleteObj	歪み補正オブジェクト削除 (メモリ解放)	○
3	imrcmd_SetObjMode	歪み補正モード設定	○
4	imrcmd_SetGrid	歪み補正モデルの生成 (補正情報の設定)	○
5	imrcmd_ImrExec	歪み補正実行	○

3. VP互換コマンド

3.1 正規化関連コマンド(VP互換コマンド)

N0	コマンド名	機能	デバイスID 指定
1	vpxAllocCorrTemplate	テンプレート特徴量データ領域確保	○
2	vpxFreeCorrTemplate	テンプレート特徴量データ領域解放	○
3	vpxReadCorrTemplate	テンプレート特徴量データ読出し	○
4	vpxWriteCorrTemplate	テンプレート特徴量データ書き込み	○
5	vpxEnableCorrMask	テンプレートマスクの有効化	○
6	vpxDisableCorrMask	テンプレートマスクの無効化	○
7	vpxSetCorrTemplate	トレーニング	○
8	vpxSetCorrTemplateExt	トレーニング	○
9	vpxIP_CorrStep	ラスタサーチ	○
10	vpxIP_CorrPoint	近傍サーチ	○
11	vpxIP_CorrPrecise	サブピクセルサーチ	○
12	vpxEnableCorrBreak	相関演算途中打ち切りの有効化	○
13	vpxDisableCorrBreak	相関演算途中打ち切りの無効化	○
14	vpxSetCorrBreakThr	相関演算途中打ち切りの閾値設定	○
15	vpxSetSearchDistance	サーチ除外領域の設定	○
16	vpxSetCorrMode	相関サーチモード設定	○
17	vpxSetCorrPrecise	サブピクセル算出桁数設定	○

3.2 直線抽出コマンド(VP互換コマンド)

N0	コマンド名	機能	デバイスID 指定
1	GetHoughLine	離散データ座標からの直線抽出	○
2	GetHoughLineRow	離散データ座標からの2 直線抽出	○
3	GetHoughLineRowExt	離散データ座標からの2 直線抽出 (拡張)	○
4	GetSideHoughLine	離散データ座標からの直線抽出 (角度固定)	○
5	GetCrossPoint	ハフ変換直線の交点座標算出	○
6	GetRectPoint	ハフ変換直線の矩形交点座標の算出	○
7	GetRectCenter	矩形頂点中心座標の算出	○
8	GetAnglePoint4	矩形頂点間中心座標 4 ポイントからの角度算出	○
9	GetAnglePoint2	矩形頂点間中心座標 2 ポイントからの角度算出	○

3.3 イメージキャリパコマンド(VP互換コマンド)

N0	コマンド名	機能	デバイスID 指定
1	ProjectLine	傾斜矩形領域の投影	○
2	LineEdgeFilter	エッジ抽出	○
3	LineEdgeFilterExt	フィルター付きエッジ抽出	○
4	LineCaliper	ラインキャリパ	○
5	CaliperLPtoSP	ライン座標から平面座標への変換	○
6	GetCaliperScore	エッジ判定閾値の算出	○
7	SetCaliperWidth	投影幅の算出	○
8	LineEdgeFinder	ラインエッジファインダ	○
9	EdgeFinderLPtoSP	ライン座標から平面座標への変換	○

3.4 図形描画コマンド(VP互換コマンド)

N0	コマンド名	機能	デバイスID 指定
1	PutLine	線・矩形描画	○
2	PutCross	十字カーソル描画	○
3	PutLineWindow	ラインウィンドウ描画	○
4	PutPolygon	多角形描画	○
5	PutCircle	円描画	○
6	PutArc	円弧描画	○
7	PutRectangle	矩形描画	○
8	PutTriangle	三角形描画	○
9	PutDiamond	菱形描画	○
10	PutCrossRect	十字形描画	○
11	PutTwinRectangle	2連矩形描画	○
12	PutArcExt	円弧描画（拡張）	○
13	PutEllipse	楕円描画処理	○
14	PutString	文字列描画（半角、全角、漢字）	○

4. ダウンロードモジュール制御

4.1 モジュールサポート

NO	コマンド名	機能	デバイスID 指定
1	LoadIPDLM	ダウンロードモジュールのロード	○
2	UnloadIPDLM	ダウンロードモジュールのアンロード	○
3	InitIPParamTable	モジュールのパラメータテーブルの初期化	○
4	SetIntegerParam	整数型パラメータのセット	○
5	SetFloatParam	実数型パラメータのセット	○
6	SetParamTable	配列データ（入力）パラメータのセット	○
7	AllocParamTable	配列データ（出力）パラメータ領域の確保	○
8	GetParamTable	配列データ（出力）パラメータの読み出し	○
9	SetParamTableExt	配列データ（入力）パラメータのセット	○
10	AllocParamTableExt	配列データ（出力）パラメータ領域の確保	○
11	ReadIPMemData	ボード側システムメモリデータリード	○
12	WriteIPMemData	ボード側システムメモリデータライト	○

4.2 インテリジェントモジュール

NO	コマンド名	機能	デバイスID 指定
1	RegistInteliIPModule	インテリジェントモジュールの登録	○
2	ExecuteInteliIPModule	インテリジェントモジュールの実行	○

4.3 PIO割込オブジェクト制御コマンド(PC側)

NO	コマンド名	機能	デバイスID 指定
1	CreateInterruptLink	割り込みリンクオブジェクトの生成	○
2	DeleteInterruptLink	割り込みリンクオブジェクトの削除	○
3	EnableInterruptObject	割り込み動作の有効化	○
4	DisableInterruptObject	割り込みオブジェクトの無効化	○

4.4 タスク制御

NO	コマンド名	機能	デバイスID 指定
1	CreateIPTask	画像処理タスクの生成	○
2	RegistIPTask	割込モジュールの登録	○
3	StartIPTaskwithParam	割込モジュールのサービス開始	○
4	WakeupIPTaskwithParam	割込モジュールの実行	○
5	WaitforIPTaskSignal	割込モジュールからの終了シグナルウェイト	○
6	ReadyforWaitIPTaskSignal	割込モジュールからの終了シグナルウェイト準備	○
7	WakeupIPTaskExt	割り込みモジュールの実行 2	○
8	CancelIPTaskWait	終了シグナルウェイトの解除	○
9	StartIPTask	割込モジュールのサービス開始	○
10	WakeupIPTask	割込モジュールの実行	○
11	TerminateIPTask	割込モジュールのサービス終了	○
12	ResumeIPTask	割込モジュールサービスの再開	○
13	SuspendIPTask	割込モジュールサービスの一時停止	○
14	DeleteIPTask	割込モジュールの削除	○

4.5 PCとの同期

NO	コマンド名	機能	デバイスID 指定
1	SendSignaltoPC	画像処理タスクからのシグナル送信（オンボードCPUのみ）	×

5. ボード制御

5.1 I/O 入出力

NO	コマンド名	機能	デバイスID 指定
1	ipport_Read	I/Oポート入力	○
2	ipport_Write	I/Oポート出力	○
3	ipport_SetDIOMode	I/Oポートモード設定	○

5.2 SCIコントロール

NO	コマンド名	機能	デバイスID 指定
1	scicom_Setup	SCI セットアップ	○
2	scicom_Purge	SCI バッファのパージ	○
3	scicom_Recv	SCI データ受信	○
4	scicom_tRecv	SCI データ受信(タイムアウト付き)	○
5	scicom_Send	SCI データ送信	○
6	scicom_GetRecvStatus	受信ステータスの取得	○
7	scicom_GetError	SCI の通信エラー取得	○

5.3 タイマコマンド

NO	コマンド名	機能	デバイスID 指定
1	SetIPTimeMode	時間計測モード設定	○
2	startGetIPTime	タイマスタート	○
3	timeGetIPTime	タイマ時間読出し	○
4	stopGetIPTime	タイマストップ	○
5	GetPerformanceFrequency	パフォーマンスカウンタの周波数取得	○
6	GetPerformanceCounter	パフォーマンスカウンタの現在値取得	○
7	IPSleep	タイマウェイト	○
8	SetIPTime	時刻の登録	○
9	GetIPTime	時刻の取得	○

5.4 USB-HID 制御コマンド

NO	コマンド名	機能	デバイスID 指定
1	USB_HID_Start	USB_HID 制御の開始	○
2	USB_HID_Terminate	USB_HID 制御の終了	○
3	USB_HID_GetStatus	USB_HID デバイスからの情報取得	○
4	USB_HID_SetReport	USB_HID デバイスへの情報設定	○

5.5 WDTコマンド

NO	コマンド名	機能	デバイスID 指定
1	WDT_Start	WDTの起動	○
2	WDT_Stop	WDTの停止	○
3	WDT_Reset	WDTカウンタのリセット	○
4	WDT_GetStatus	WDTステータスの取得	○
5	WDT_ClearStatus	WDTステータスのクリア	○

5.6 CPUエラーコマンド

NO	コマンド名	機能	デバイスID 指定
1	CPUERR_Set	CPUエラー出力	○
2	CPUERR_Clear	CPUエラークリア	○

6. ファイルアクセスコマンド

NO	コマンド名	機能	デバイスID 指定
1	fmOpen	ファイルオープン	○
2	fmClose	ファイルクローズ	○
3	fmRead	ファイル読み込み	○
4	fmWrite	ファイル書き込み	○
5	fmGetFileSize	ファイルサイズ取得	○
6	fmDelete	ファイル削除	○
7	fmRename	ファイル名変更	○
8	fmFindFile	ファイル検索	○
9	fmChgAttr	ファイルの属性変更	○
10	fmFileCopy	P C - オンボードディスク間のファイルコピー	○
11	fmGetError	エラーコード取得	○
12	fmSeek	ファイルポインタの移動	○
13	fmCreateDirectory	ディレクトリの作成	○
14	fmRemoveDirectory	ディレクトリの削除	○
15	fmFindFirstFile	ファイル検索と検索ハンドルのオープン	○
16	fmfindNextFile	ファイル検索	○
17	fmFindClose	検索ハンドルのクローズ	○
18	fmRefAttr	ファイルの属性参照（ファイル名指定）	○
19	fmDriveFormat	ディスクドライブのフォーマット	○
20	fmDriveSpace	ディスクサイズと空きエリア容量の取得	○
21	fmCopyFile	オンボードディスク間のファイルコピー	○
22	FileTimeToIPTime	システム時刻形式変換	○

7. μ ITRONサービスコール

NO	サービスコール	機能	デバイスID 指定
1	acre_tsk	タスクの生成	○
2	del_tsk	タスクの削除	○
3	act_tsk	タスクの起動	○
4	ext_tsk/exd_tsk	自タスクの終了	○
5	ter_tsk	タスクの強制終了	○
6	chg_pri	タスク優先度の変更	○
7	ref_tsk	タスクの状態参照	○
8	slp_tsk/tslp_tsk	起床待ち	○
9	wup_tsk	タスクの起床	○
10	can_wup	タスク起床要求のキャンセル	○
11	sus_tsk	強制待ち状態への移行	○
12	rsm_tsk	強制待ち状態からの再開	○
13	dly_tsk	タスク遅延	○
14	acre_sem	セマフォの生成	○
15	del_sem	セマフォの削除	○
16	sig_sem	セマフォ資源の返却	○
17	wai_sem/pol_sem/twai_sem	セマフォ資源の獲得	○
18	ref_sem	セマフォの状態参照	○
19	acre_flg	イベントフラグの生成	○
20	del_flg	イベントフラグの削除	○
21	set_flg	イベントフラグのセット	○
22	clr_flg	イベントフラグのクリア	○
23	wai_flg/pol_flg/twai_flg	イベントフラグ待ち	○
24	ref_flg	イベントフラグの状態参照	○

8. Windowsアプリケーションサポートコマンド

NO	コマンド名	機能	デバイスID 指定
1	ReadBMPImg	IMデータをBMPに変換し転送	○
2	WriteBMPImg	BMPデータをIMに転送	○
3	CreateBMPImgObject	画像メモリビットマップオブジェクトの生成	○
4	DeleteBMPImgObject	画像メモリビットマップオブジェクトの削除	×
5	GetBMPImgInfoEntry	ビットマップ情報ヘッダー先頭アドレスの取得	×
6	GetBMPImgBitsEntry	ビットマップデータ先頭アドレスの取得	×

9. イーサネット通信コマンド

NO	コマンド名	機能	デバイスID 指定
1	socket	ソケット生成	○
2	bind	通信アドレス情報の指定	○
3	listen	受動モード設定	○
4	accept	ソケットに対するコネクションの受入れ	○
5	connect	ソケットの接続の開始	○
6	getpeername	相手側通信アドレス情報取得	○
7	getsockname	自通信アドレス情報取得	○
8	recv	ソケットからの受信データ取得	○
9	recvfrom	受信データと送信者アドレス取得	○
10	send	ソケットへの送信データ設定	○
11	sendto	送信データと送信先アドレス設定	○
12	closesocket	ソケットのクローズ	○
13	shutdown	コネクション閉鎖	○
14	getsockopt	プロトコルのオプション取得	○
15	setsockopt	プロトコルのオプション変更	○
16	selectsocket	ソケットの利用可能待ち	○
17	set_blocking_socket	ブロッキングモード設定	○
18	get_errno	エラーコード取得	○
19	get_thread_errno	スレッドエラーコード取得	○
20	set_sock_timewait	TIMEWAIT 時間の変更	○
21	get_sock_recvlen	受信データ長の取得	○
22	set_sock_keepalive	キープアライブ制御情報の変更	○

第1章 初期化コマンド

1.1 デバイスID無しリモートコマンドのAPI初期化

1.1.1 デバイスID無しリモートコマンドの使用開始（ StartIP ）

□ C言語API

```
int ret = StartIP( int BoardNo, int Opt );
```

□ パラメータ

int	BoardNo	ボード番号（0～15）
int	Opt	オプション（0を指定して下さい）

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

異常終了した場合、GetIPLastErrorでエラー情報を取得できます。

0x00020101	パラメータエラー
0x00020102	ボード未装着又はドライバがインストールされていないか動作していない
0x00020103	ドライバ情報が取得できなかった
0x00020104	ボード枚数 読出しエラー
0x00020105	ボード番号 読出しエラー
0x00020106	ボード番号 2重設定エラー
0x00020107	ドライバを動作させるリソース不足
0x00020108	オープンコマンドがドライバ内で異常終了した
0x00020109	オンボードシステムでエラーが発生した
0x0002010A	ファイルオープンエラー
0x0002010B	ファイルアクセスエラー
0x0002010C	ファイルタイプエラー
0x0002010D	メモリ確保エラー
0x0002010E	オンボードシステムを起動できなかった
0x0002010F	ターゲットボード不一致

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

指定した番号のボードの初期化を行います。本コマンドはデバイスID無しのAPIを使用する場合に行ってください。

画像処理ボードのボード番号は、レジストリ設定(VPSETREG.EXE)で設定したIPアドレス、ポート番号のボード番号を指定して下さい（1台のみの使用の場合は「0」以外の設定にしないで下さい）。

1.1.2 デバイスID無しリモートコマンドの使用停止（ StopIP ）

□ C言語API

```
int ret = StopIP( int BoardNo );
```

□ パラメータ

int BoardNo ボード番号（ 0 ～ 1 5 ）

□ リターンパラメータ

int ret 正常終了（ 0 ）またはエラーコード

□ エラーコード

- 1 7 パラメータエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

指定したボード番号のボードのクローズ処理を行います。本コマンドは、StartIPコマンドで初期化したアプリケーションのクリーンアップに使用してください。

1.1.3 ボード番号の切り替え (ActiveIP)

□ C言語API

```
int ret = ActiveIP( int BoardNo );
```

□ パラメータ

int BoardNo ボード番号 (0 ~ 15)

□ リターンパラメータ

int ret 正常終了 (0) またはエラーコード

□ エラーコード

- 17 パラメータエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

デバイスID無しのAPIを使用している場合、指定したボード番号のボードに処理を切り換えます。
ボード番号については、StartIPコマンドを参照してください。

1.2 デバイスID付きリモートコマンドのAPI初期化

1.2.1 デバイスID付きリモートコマンドの使用開始 (OpenIPDev)

□ C言語API

```
DEVID devID = OpenIPDev( int BoardNo , int opt );
```

□ パラメータ

int	BoardNo	ボード番号 (0 ~ 15)
int	opt	オプション。通常は0を設定して下さい。
	OPENIP_OPT_RESET_AUTO	(0x0000) ブートしていない場合リセットします
	OPENIP_OPT_RESET_MANUAL	(0x0001) マニュアルリセットします
	OPENIP_OPT_RESET_POWERON	(0x0002) パワーオンリセットします
	OPENIP_OPT_RESET_DISABLE	(0x0008) リセットしない
	OPENIP_OPT_BOOT_AUTO	(0x0000) ブートしていない場合、システムをダウンロードし ブートします
	OPENIP_OPT_BOOT_DISABLE	(0x0080) ブートしない

□ リターンパラメータ

DEVID devID > 0 : デバイス I D
ISPX_NULL : 異常終了

□ エラーコード

異常終了した場合、GetIPLastErrorでエラー情報を取得できます。

0x00020101	パラメータエラー
0x00020102	ボード未装着又はドライバがインストールされていないか動作していない
0x00020103	ドライバ情報が取得できなかった
0x00020104	ボード枚数 読出しエラー
0x00020105	ボード番号 読出しエラー
0x00020106	ボード番号 2重設定エラー
0x00020107	ドライバを動作させるリソース不足
0x00020108	オープンコマンドがドライバ内で異常終了した
0x00020109	オンボードシステムでエラーが発生した
0x0002010A	ファイルオープンエラー
0x0002010B	ファイルアクセスエラー
0x0002010C	ファイルタイプエラー
0x0002010D	メモリ確保エラー
0x0002010E	オンボードシステムを起動できなかった
0x0002010F	ターゲットボード不一致

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

指定した番号のボードの初期化を行いボードに対する制御情報を識別するためのデバイス I D (devID) を取得します。

画像処理ボードのボード番号は、レジストリ設定 (VPSETREG.EXE) で設定した I P アドレス、ポート番号のボード番号を指定して下さい (1 台のみの使用の場合は「0」以外の設定にしないで下さい) 。

1.2.2 デバイスID付きリモートコマンドの使用停止（CloseIPDev）

□ C言語API

```
int ret = CloseIPDev( DEVID devID );
```

□ パラメータ

DEVID	devID	デバイス I D

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

異常終了した場合、GetIPLastErrorでエラー情報を取得できます。

0x00020101	パラメータエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

指定したデバイス I Dのボードのクローズ処理を行います。本コマンドは、OpenIPDevコマンドで初期化したアプリケーションのクリーンアップに使用してください。

1.3.2 画像処理ボードのリセット (ResetIP)

□ C言語API

```
int ret = ResetIP( int BoardNo, int Mode );
```

□ パラメータ

int	BoardNo	ボード番号 (0 ~ 1 5)
int	Mode	モード (0 または 4 を指定して下さい) (0) P C 側リセット、ターゲット C P U 側処理なし (4) P C 側リセット、ターゲット C P U 側リセット

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

異常終了した場合、GetIPLastErrorでエラー情報を取得できます。

0x00020101	パラメータエラー
0x00020102	ボード未装着又はドライバがインストールされていないか動作していない
0x00020103	ドライバ情報が取得できなかった
0x00020104	ボード枚数 読出しエラー
0x00020105	ボード番号 読出しエラー
0x00020106	ボード番号 2 重設定エラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

指定したボード番号のボードをリセットし、ROMのイニシャライズプログラムを起動します。本コマンドは、デバイス I D 無しの A P I でもデバイス I D 付きの A P I でも同じインタフェースで使用できます。また、StartIPやOpenIPDevコマンド発行前でも使用可能です。

画像処理ボードのボード番号は、レジストリ設定 (VPSETREG.EXE) で設定した I P アドレス、ポート番号のボード番号を指定して下さい (1 台のみの使用の場合は「0」以外の設定にしないで下さい) 。

1.3.3 エラーコード取得 (GetIPLastError)

□ C言語API

```
int Error = GetIPLastError( void );
```

□ パラメータ

なし

□ リターンパラメータ

int	Error	エラー無し (0) それ以外はエラーコード

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

A P I 初期化コマンドで発生したエラー情報を読み出します。

第2章 画像処理コマンド

2.1 システム制御

2.1.1 画像処理システムの初期化（InitIP）

□ C言語API

```
int ret = InitIP();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	0 : 正常終了
		- 2 : OSシステムエラー
		- 1 3 1 : V I Nドライバでのエラー
		- 1 3 2 : D Uドライバでのエラー
		- 1 3 6 : システム管理データが不正
		- 1 3 8 : 待機状態で実行
		- 1 4 0 : メモリ管理ドライバでのエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ エラーコード

なし

□ 機能

画像処理システムのハードウェア、システム管理テーブルをイニシャライズし、全画像メモリを解放後、下記内容に設定します。

『InitIP()』コマンドまたは『InitIPExt()』コマンドは画像認識コマンドAPIを使用する前にアプリケーション初期化時に必ず実行して下さい。

設定項目	設定内容
システムデータタイプ	符号なし8ビット
ウィンドウ（全種）	有効（512（X）×480（Y））
文字描画属性（サイズ）	標準サイズ（16×16）
文字描画属性（文字間隔）	0ドット
文字描画属性（行間隔）	0ドット
正規化関連マスク	無効
正規化関連途中演算打ち切り	無効
正規化関連打ち切りしきい値	0
パイプラインモード	解除

2.1.2 画像処理システムの初期化（電源ON時）（InitIPExt）

□ C言語API

```
int ret = InitIPExt();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	0 : 正常終了
		- 2 : OSシステムエラー
		- 1 3 1 : V I Nドライバでのエラー
		- 1 3 2 : D Uドライバでのエラー
		- 1 3 6 : システム管理データが不正
		- 1 3 8 : 待機状態で実行
		- 1 4 0 : メモリ管理ドライバでのエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ エラーコード

なし

□ 機能

本コマンドは、『InitIP()』コマンドと同機能となります。
 詳細については『InitIP()』コマンドを参照して下さい。

2.1.3 エラーメッセージの出力許可 (EnableIPErrorMessage)

☐ C言語API

```
int ret = EnableIPErrorMessage();
```

☐ パラメータ

なし

☐ 対応画面タイプ／画面データタイプ

なし

☐ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

☐ エラーコード

なし

☐ 機能

本コマンドを実行すると、コマンドエラー発生時にエラーメッセージが出力されます。(リモートコマンドでは出力許可がデフォルト、オンボード側では出力禁止状態がデフォルト設定です)

本コマンドは、デバイスID付きAPIにおいてもパラメーターにデバイスIDは不要です。リモートコマンドではスレッド毎、オンボードのコマンドではタスク毎に管理されているためスレッド毎、タスク毎に設定が必要です。

2.1.4 エラーメッセージの出力禁止（DisableIPErrorMessage）

☐ C言語API

```
int ret = DisableIPErrorMessage();
```

☐ パラメータ

なし

☐ 対応画面タイプ／画面データタイプ

なし

☐ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

☐ エラーコード

なし

☐ 機能

本コマンドを実行すると、コマンドエラー発生時のエラーメッセージが出力されなくなります。本コマンドは、デバイスID付きAPIにおいてもパラメーターにデバイスIDは不要です。リモートコマンドではスレッド毎、オンボードのコマンドではタスク毎に管理されているためスレッド毎、タスク毎に設定が必要です。

2.1.5 エラー情報読み出し／クリア (ReadIPErrorTable)

□ C言語API

```
int ret = ReadIPErrorTable(IPErrorTbl *Tbl);
```

□ パラメータ

IPErrorTbl	*Tbl	エラー情報テーブルを返すアドレス

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	0 : 正常終了 -1 : エラー情報テーブルを返すアドレスが 不正、またはドライバエラー
IPErrorTbl	*Tbl	エラー情報テーブルを格納したアドレス

□ パケットの構造

```
typedef struct {
    int      ErrorCode;           エラーコード
    char      ErrorRoutin[64];    エラー発生コマンド名
    int      ErrorINFO;          リザーブ
} IPErrorTbl;
```

□ エラーコード

なし

□ 機能

エラー情報テーブルを読み出し、ユーザテーブルに格納します。また、コマンドエラーをクリアします。コマンドエラーから復帰するには、本コマンドもしくは『ClearIPError()』コマンドを実行してコマンドエラーのクリアを行う必要があります。なお、復帰のためにボードリセットが必要な場合があります。付録A「エラーコード一覧」を参照して下さい。

エラー未発生時やコマンドエラーのクリア後に呼び出した場合は、エラー情報テーブルの各メンバーに“0”が設定され、正常終了となります。

本コマンドは、デバイスID付きAPIにおいてもパラメーターにデバイスIDは不要です。リモートコマンドではスレッド毎、オンボードのコマンドではタスク毎に管理されているためスレッド毎、タスク毎に本コマンドの発行が必要です。

2.1.6 エラー情報クリア (ClearIPError)

□ C言語API

```
int ret = ClearIPError();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	0 : 正常終了
		-1 : ドライバエラー

□ エラーコード

なし

□ 機能

コマンドエラーをクリアします。コマンドエラーから復帰するには、本コマンドもしくは『ReadIPErrorTable()』コマンドを実行してコマンドエラーのクリアを行う必要があります。なお、復帰のためにボードリセットが必要な場合があります。付録A「エラーコード一覧」を参照して下さい。

本コマンドは、デバイスID付きAPIにおいてもパラメーターにデバイスIDは不要です。リモートコマンドではスレッド毎、オンボードのコマンドではタスク毎に管理されているためスレッド毎、タスク毎に本コマンドの発行が必要です。

2.1.7 画像処理動作データタイプ設定 (SetIPDataType)

□ C言語API

```
int ret = SetIPDataType(enum DataType type);
```

□ パラメータ

enum DataType	type	システムデータタイプ
	SIGN8_DATA	(0) 符号付8ビット (−128～127)
	UNSIGN8_DATA	(1) 符号なし8ビット (0～255)
	BINARY_DATA	(2) 2値 (0または255)

※本コマンドでは、BINARY_DATAは使用できません

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (−1)

□ エラーコード

20	システムデータタイプ設定範囲外
−	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

システムが管理しているテーブルにシステムデータタイプを設定します。システムデータタイプとは、画像処理、カメラ入力で扱う画像のデータタイプのことを示し、符号付8ビット (−128～127) と符号なし8ビット (0～255) の2種類が設定できます。『InitIP()』コマンド発行後は、符号なし8ビットとなります。

指定されたデータタイプにより、画像データが次のような値をとると見なして処理します。

● 符号なし8ビットの場合

- ・画像処理のデータ入出力は、0～255の数値をとります。
- ・2値化データの背景画像は0、物体画像は255の値をとります。

● 符号付8ビットの場合

- ・画像処理のデータ入出力は、−128～127の数値をとります。
- ・2値化データの背景画像は0、物体画像は−1の値をとります。

設定は本コマンドの実行後に発行されたコマンドに対して有効です。先行するコマンドの処理が終了していない場合 (画像処理中又はビデオ取り込み中、取り込み待ち中) でも、これらの処理には影響を与えません。

ただし、パイプラインモード時は、パラメータチェックとヒストグラム処理にのみ反映されます。この為、パイプラインモード時に本コマンドを使用する場合、パイプライン無効時の処理結果と相違が生じる場合がありますので注意して下さい。

2.1.8 バージョン情報の取得（GetIPVersionInfo）

□ C言語API

```
int ret = GetIPVersionInfo( VERSIONINFO *VersionInfo );
```

□ パラメータ

VERSIONINFO *VersionInfo バージョン情報格納領域の先頭アドレス。
あらかじめ、sizeof(VERSIONINFO)の領域を確保して下さい。

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

— エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

PCドライバ、画像処理コマンド、ボード、ROM等のバージョンを読み出します。

VERSIONINFOのフォーマットを、下記に示します。

```
typedef struct{
    unsigned long   IPCMD_Env;
    unsigned long   BOARD_Info;
    unsigned long   IPCMD_Version;
    unsigned long   IPCMD_Update;
    unsigned long   ROM_Version;
    unsigned long   ROM_Update;
    unsigned long   PCDRV_Version;
    unsigned long   PCDRV_Update;
    unsigned long   DEVDRV_Version;
    unsigned long   DEVDRV_OS;
    unsigned long   Reserve[6];
} VERSIONINFO;
```

以下にVERSIONINFO構造体のメンバ変数の詳細を示します。

メンバ名	内容						
IPCMD_Env	画像処理コマンドの動作環境						
	対応数	内容					
	0	ボードが動作していない					
	1	ROM上のIPLが動作している					
	2	RAMにダウンロードされたオンボードシステムが動作している					
BOARD_Info	画像処理ボードの情報						
	31 _____ 12 11 _____ 0						
	品種	ボード情報					
品種 0x20047 : VP-Ax230CL 0x20041 : VP-Ax235CL 0x20147 : VP-Ax230CL (Linux版) 0x20141 : VP-Ax235CL (Linux版)							
IPCMD_Version	画像処理コマンド (ボードCPU側) のバージョン情報						
	31 _____ 24 23 _____ 16 15 _____ 8 7 _____ 0						
<table><tr><td>Major Version</td><td>Minor Version</td><td>Opt</td><td>Build</td></tr></table>				Major Version	Minor Version	Opt	Build
Major Version	Minor Version	Opt	Build				
IPCMD_Update	画像処理コマンド (ボードCPU側) の最終更新日						
	31 _____ 24 23 _____ 16 15 _____ 8 7 _____ 0						
<table><tr><td>Year</td><td>Month</td><td>Day</td><td>Opt</td></tr></table>				Year	Month	Day	Opt
Year	Month	Day	Opt				
ROM_Version	ROMのバージョン情報						
	31 _____ 24 23 _____ 16 15 _____ 8 7 _____ 0						
<table><tr><td>Major Version</td><td>Minor Version</td><td>Opt</td><td>Build</td></tr></table>				Major Version	Minor Version	Opt	Build
Major Version	Minor Version	Opt	Build				
ROM_Update	ROMの最終更新日						
	31 _____ 24 23 _____ 16 15 _____ 8 7 _____ 0						
<table><tr><td>Year</td><td>Month</td><td>Day</td><td>Opt</td></tr></table>				Year	Month	Day	Opt
Year	Month	Day	Opt				
PCDRV_Version	PCドライバ (DLL) のバージョン情報						
	31 _____ 24 23 _____ 16 15 _____ 8 7 _____ 0						
<table><tr><td>Major Version</td><td>Minor Version</td><td>Opt</td><td>Build</td></tr></table>				Major Version	Minor Version	Opt	Build
Major Version	Minor Version	Opt	Build				
PCDRV_Update	PCドライバ (DLL) の最終更新日						
	31 _____ 24 23 _____ 16 15 _____ 8 7 _____ 0						
<table><tr><td>Year</td><td>Month</td><td>Day</td><td>Opt</td></tr></table>				Year	Month	Day	Opt
Year	Month	Day	Opt				
DEVDRV_Version	デバイスドライバのバージョン情報						
	31 _____ 24 23 _____ 16 15 _____ 8 7 _____ 0						
<table><tr><td>Major Version</td><td>Minor Version</td><td>Opt</td><td>Build</td></tr></table>				Major Version	Minor Version	Opt	Build
Major Version	Minor Version	Opt	Build				
DEVDRV_OS	デバイスドライバの種類						
	対応数	内容					
	3	Windows WDM (デバイスドライバ)					
	4	TCP DLL (LAN通信インタフェース)					
Reserve	予約 (24バイト)						

2.2 画像メモリ領域管理

2.2.1 画像メモリ領域確保（AllocImg）

□ C言語API

```
IMGID ImgID = AllocImg(enum ImageFrameSize size);
```

□ パラメータ

enum ImageFrameSize size 画面サイズ

画面サイズ	対応定数	X方向	Y方向
IMG_FS_256H_256V	0x01000100	256	256
IMG_FS_256H_512V	0x02000100	256	512
IMG_FS_320H_256V	0x01000140	320	256
IMG_FS_320H_512V	0x02000140	320	512
IMG_FS_512H_256V	0x01000200	512	256
IMG_FS_512H_512V	0x02000200	512	512
IMG_FS_640H_256V	0x01000280	640	256
IMG_FS_640H_512V	0x02000280	640	512
IMG_FS_768H_512V	0x02000300	768	512
IMG_FS_768H_576V	0x02400300	768	576
IMG_FS_1024H_768V	0x03000400	1024	768
IMG_FS_1024H_1024V	0x04000400	1024	1024
IMG_FS_1280H_1024V	0x04000500	1280	1024
IMG_FS_1344H_1024V	0x04000540	1344	1024
IMG_FS_1600H_1024V	0x04000640	1600	1024
IMG_FS_2560H_1920V	0x07800A00	2560	1920
IMG_FS_2560H_2048V	0x08000A00	2560	2048
IMG_FS_2688H_2048V	0x08000A80	2688	2048
IMG_FS_2688H_2176V	0x08800A80	2688	2176

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	—	—	—	—	—	◆	—	—

□ リターンパラメータ

IMGID ImgID 画面番号（1～CFG_MAXIMGID）または異常終了（-1）
※CFG_MAXIMGIDはシステムで定義しています。

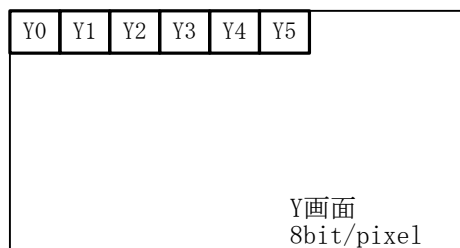
□ エラーコード

2 0 画面サイズ設定値範囲外
4 6 画面ID空きなし
4 8 動的メモリ確保エラー
— エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

画像メモリにパラメータで指定されたサイズのY画面領域を確保し、その画面番号を返します。本コマンドで確保した画像メモリ領域は、濃淡及び2値の画像処理、モノクロ映像入力表示に使用できます。また、画像メモリのデータ属性は符号なし8ビットになります。
RGB画面の確保は、AllocRGBImg()を使用します。



確保した画像メモリを映像入力に使用する場合、以下の制限事項があります。

- ・ 使用するカメラの設定が2TAP (Dual)、4TAP、または、8TAPの場合、X方向サイズをかならず128の倍数にする必要がございます。映像入力に使用する画像メモリにはX方向サイズが、320、1344、1600の画像メモリサイズを利用しないでください。

2.2.2 同時取込み用画像メモリ領域確保 (AllocComponentImg)

□ C言語API

```
IMGID ImgID = AllocComponentImg( enum ImgFrameSize ImgSize , int comp-CH );
```

□ パラメータ

enum ImgFrameSize

ImgSize

画面サイズ

画面サイズ	対応定数	X方向	Y方向
IMG_FS_256H_256V	0x01000100	256	256
IMG_FS_256H_512V	0x02000100	256	512
IMG_FS_320H_256V	0x01000140	320	256
IMG_FS_320H_512V	0x02000140	320	512
IMG_FS_512H_256V	0x01000200	512	256
IMG_FS_512H_512V	0x02000200	512	512
IMG_FS_640H_256V	0x01000280	640	256
IMG_FS_640H_512V	0x02000280	640	512
IMG_FS_768H_512V	0x02000300	768	512
IMG_FS_768H_576V	0x02400300	768	576
IMG_FS_1024H_768V	0x03000400	1024	768
IMG_FS_1024H_1024V	0x04000400	1024	1024
IMG_FS_1280H_1024V	0x04000500	1280	1024
IMG_FS_1344H_1024V	0x04000540	1344	1024
IMG_FS_1600H_1024V	0x04000640	1600	1024
IMG_FS_2560H_1920V	0x07800A00	2560	1920
IMG_FS_2560H_2048V	0x08000A00	2560	2048
IMG_FS_2688H_2048V	0x08000A80	2688	2048
IMG_FS_2688H_2176V	0x08800A80	2688	2176

int

comp-CH

画面チャネル数(2~4)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	—	—	—	—	—	◆	—	—

□ リターンパラメータ

IMGID ImgID ≥1 : 画面番号 または、異常終了 (-1)

□ エラーコード

2 0	画面サイズ設定値範囲外
2 1	画面チャンネル数設定値範囲外
4 6	画面領域空きなし
4 8	動的メモリ確保エラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

画像メモリに、パラメータで指定されたサイズのモノクロ同時入力映像を格納するための画面領域(コンポーネント画面)を確保し、その画面番号を返します。

モノクロ同時映像入力用にはモノクロ画面(Y画面)をcomp-CHで指定した画面数確保しますが、本コマンドでは1画面分の番号だけ返して、その他の画面はシステムで管理します。

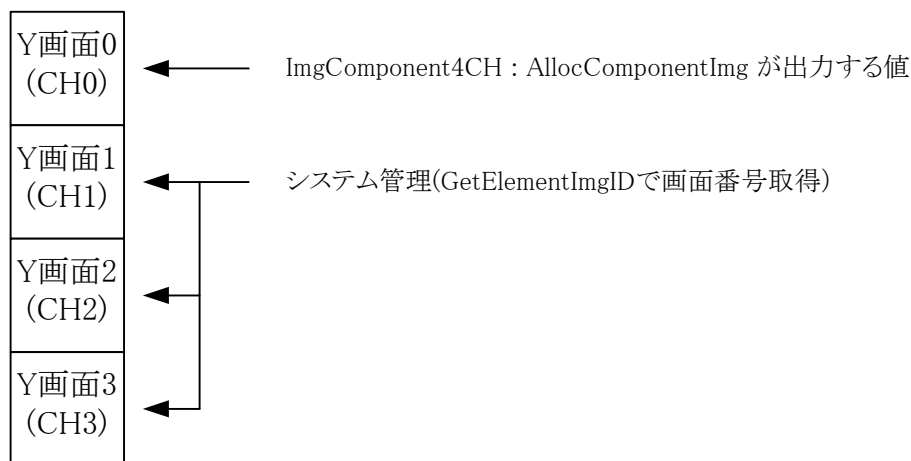
他の画面番号はGetElementImgIDで取得してください。

なお、RGBカメラの同時入力はできませんので、RGB画面の確保はできません。

4カメラ同時入力用コンポーネント画面の確保は下図のようになります。

```
int ImgComponent4CH;
```

```
ImgComponent4CH = AllocComponentImg( IMG_FS_512H_512V, 4 )
```



確保した画像メモリを映像入力に使用する場合、以下の制限事項があります。

- ・ 使用するカメラの設定が2TAP (Dual1)、4TAP、または、8TAPの場合、X方向サイズをかならず128の倍数にする必要があります。映像入力に使用する画像メモリにはX方向サイズが、320、1344、1600の画像メモリサイズを利用しないでください。

2.2.3 画像メモリ領域解放（指定画面）（FreeImg）

□ C言語API

```
int ret = FreeImg(IMGID ImgID);
```

□ パラメータ

IMGID ImgID 画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	○	—	—	—	○	○	○	○

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2 OSシステムエラー
 1 0 画面番号が有効範囲外（不当画面番号エラー）
 1 3 1 VINドライバでのエラー
 1 3 2 DUドライバでのエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

パラメータで指定された画面番号の画像メモリ領域を解放します。画面番号に対応する画面が未確保の場合は、処理をせずに正常終了します。

また、連続映像入力キャプチャ画面の場合は、連続映像入力を停止してから解放します。画像メモリ表示画面の場合は、画像メモリ表示を停止してから解放します。

2.2.4 画像メモリ領域解放（全画面）（FreeAllImg）

□ C言語API

```
int ret = FreeAllImg();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
1 3 1	V I Nドライバでのエラー
1 3 2	D Uドライバでのエラー
-	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

画像確保した全画像メモリ領域を解放します。

連続映像入力時は連続映像入力を停止してから解放します。画像メモリ表示時は、画像メモリ表示を停止してから解放します。

2.2.5 画像メモリ管理テーブル読み出し (ReadImgTable)

□ C言語API

```
int ret = ReadImgTable(IMGID ImgID, IMGTBL *Tbl);
```

□ パラメータ

IMGID ImgID 画面番号

IMGTBL *Tbl 画像メモリ管理テーブルを返すアドレス

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	○	○	○	○	○	○	○	○

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

IMGTBL *Tbl 画像メモリ管理テーブルを格納したアドレス

□ パケットの構造

```
typedef struct {
    int      mbmp_id;      メモリビットマップID
    int      ch;           画像メモリチャンネル番号 (内部管理)
    int      sx;           画像処理領域の開始X座標 (画面原点相対)
    int      sy;           画像処理領域の開始Y座標 (画面原点相対)
    int      size;         画面サイズ番号 (ImageFrameSize列挙型に対応)
    int      xlng;         画面X方向サイズ
    int      ylng;         画面Y方向サイズ
    int      dtyp;         画面データタイプ
                                0 : 符号付8ビット (-128～127)
                                1 : 符号なし8ビット (0～255)
                                2 : 2値 (0または255)
                                -1 : 不定画面
    int      lock;         画面ロック制御フラグ (内部管理)
    int      open;         オープン制御フラグ (内部管理)
    int      dirty;        画面更新制御フラグ (内部管理)
    int      color;        画面属性管理情報
                                0 : モノクロ画面
                                4 : RGB-R画面
                                5 : RGB-G画面
                                6 : RGB-B画面
                                7 : RGB-16画面
    int      arrange;      画面配置管理情報 (内部管理)
    int      next_id;      システム管理画面番号 (コンポーネント画面 (最後尾コンポーネント画面を除く) 以外の場合は0)
    char     *top;         IMPに設定する画像メモリ先頭アドレス
    int      cache;        キャッシュ制御フラグ
                                0 : キャッシュページ要求クリア
                                1 : デスティネーション画面となる場合のみ
                                   キャッシュページ要求
                                2 : キャッシュページ要求
                                3 : キャッシュページ要求
    int      work_id;      システム管理 (ワーク) 画面番号 (内部管理)
    int      change;       画像メモリ管理テーブル情報入れ替え制御フラグ (内部管理)
    int      type;         画像メモリ領域管理フラグ (内部管理)
    char     *vtop;        CPUからアクセスする画像メモリ先頭アドレス
    int      width;        画像処理領域のX方向の長さ (内部管理)
    int      height;       画像処理領域のY方向の長さ (内部管理)
    char     *heap;        画像メモリアドレス (内部管理)
    int      bpp;          1画素あたりのバイト数 (内部管理)
    int      comp-ch;      コンポーネントチャンネル
} IMGTBL;
```

□ エラーコード

1 0	番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
2 1	画像メモリ管理テーブルのアドレス不正
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

パラメータで指定された画面番号の画像メモリ管理テーブルを読み出し、ユーザテーブルに格納します。

- 『ch』, 『lock』, 『open』, 『dirty』, 『arrange』, 『work_id』, 『change』, 『type』, 『width』, 『height』, 『heap』, 『bpp』は、本ライブラリの内部で使用する一時的なデータ、または未使用データです。
- 『mbmp_id』はメモリ管理ドライバ関数で確保されたメモリ領域のビットマップIDです。0以上の値をとります。
- 『sx』, 『sy』は画像処理起動前に設定されます。画像処理コマンドのソース0画面、ソース1画面、デスティネーション画面のいずれかに同一の画面番号を指定した場合、『sx』, 『sy』は上書きされ、1画面分の情報しか保持されません。
- 『next_id』はコンポーネント画面（最後尾画面を除く）以外の場合、0となります。
- 『cache』は画像メモリアクセスコマンドやグラフィックスコマンド等のSH-CPUによるメモリアクセスを行った場合に設定され、キャッシュパーージ後に0に設定されます。『cache』の3は『cache』の1と『cache』の2のORの状態で、2と3の動作は同じになります。
- 『comp-CH』は、同時取込み用画面のチャンネル数です。

2.2.6 コンポーネント画面番号取出し (GetElementImgID)

□ C言語API

```
IMGID ImgID = GetElementImgID ( IMGID ImgComponent , int ElementNo );
```

□ パラメータ

IMGID	ImgComponent	コンポーネント画面の画面番号
int	ElementNo	エレメント番号(0~3)

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgComponent	○	×	×	×	×	×	○	○	○
ImgID	○	—	—	—	—	—	○	—	—

ただし、ImgIDはAllocComponentImg() コマンド実行後、画面データタイプの変更をしていないものとします。

□ リターンパラメータ

IMGID	ImgID	コンポーネント画面の画面番号(>0) または、異常終了 (≤ 0)

□ エラーコード

1 0	指定画面空き領域、または範囲外 (不当画面番号エラー)
2 1	エレメント番号設定値範囲外
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

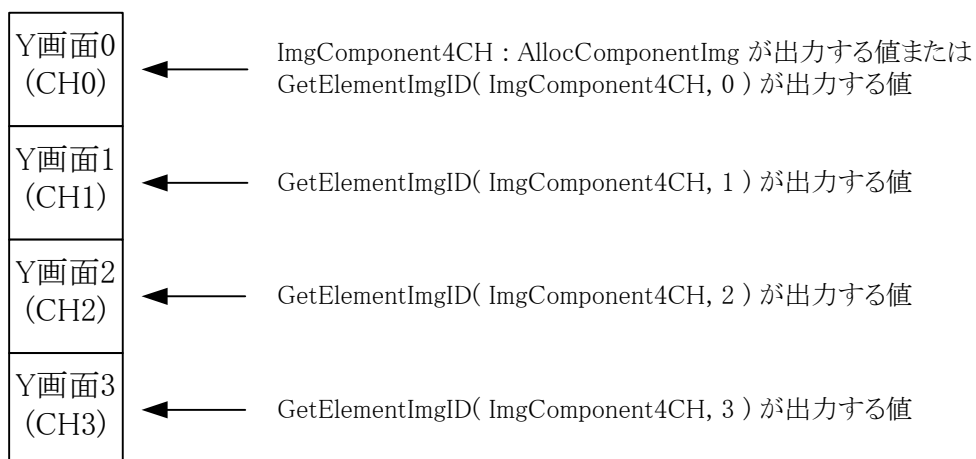
パラメータで指定されたコンポーネント画面(ComponentImgID)に対するシステムが管理している画面から
 エlement番号(ElementNo)で指定されたモノクロ(Y)画面の画面番号を返します。
 なお、RGBカメラの同時入力はできませんので、RGB画面の確保はできません。

CH1～CH3の画面の処理が必要な場合、本コマンドで画面番号を取得して処理を行います。

4カメラ同時入力用のコンポーネント画面の場合は下図のようになります。

```
int ImgComponent4CH;  

ImgComponent4CH = AllocComponentImg( IMG_FS_512H_512V, 4 )
```



2.2.7 画面データタイプ変更 (ChangeImgDataType)

□ C言語API

```
int ret = ChangeImgDataType(IMGID ImgID, enum DataType type);
```

□ パラメータ

IMGID	ImgID	画面番号
enum DataType	type	画面データタイプ
	SIGN8_DATA	(0) 符号付8ビット (−128〜127)
	UNSIGN8_DATA	(1) 符号なし8ビット (0〜255)
	BINARY_DATA	(2) 2値 (0または255)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	—	○	○	○	×	○	○	○

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (−1)

□ エラーコード

1 0	番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 1	画面タイプ設定範囲外
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

各画面の持つ画面データタイプを変換します。

設定は本コマンドの実行後に発行されたコマンドに対して有効です。先行するコマンドの処理が終了していない場合 (画像処理中又はビデオ取り込み中、取り込み待ち中) でも、これらの処理には影響を与えません。

ただし、パイプラインモード時は、パラメータチェックとヒストグラム処理にのみ反映されます。この為、パイプラインモード時に本コマンドを使用する場合、パイプライン無効時の処理結果と相違が生じる場合がありますので注意して下さい。

2.2.8 ウィンドウ設定（種類指定）（SetWindow）

□ C言語API

```
int ret = SetWindow(enum WindowType type, int sx, int sy, int ex, int ey);
```

□ パラメータ

enum WindowType	type	ウィンドウ種類
	SRC0_WIN	(0) 画像処理のソース画面1のウィンドウ
	SRC1_WIN	(1) 画像処理のソース画面2のウィンドウ
	SRC2_WIN	(2) 画像処理のソース拡張画面のウィンドウ（未使用）
	DST_WIN	(3) 画像処理のデスティネーション画面のウィンドウ
	SYS_WIN	(4) 画像メモリ制御コマンドのウィンドウ
	DST_EXT_WIN	(6) 画像処理のデスティネーション拡張画面のウィンドウ（未使用）

int	sx	始点X座標
int	sy	始点Y座標
int	ex	終点X座標
int	ey	終点Y座標

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2 0	ウィンドウ種類設定値範囲外
2 1	始点X座標範囲外
2 2	始点Y座標範囲外
2 3	終点X座標範囲外、始点X座標＞終点X座標と設定
2 4	終点Y座標範囲外、始点Y座標＞終点Y座標と設定
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

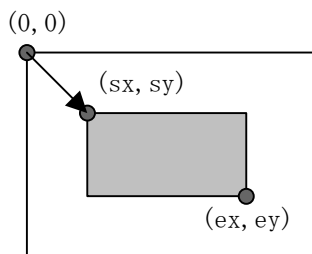
□ 機能

システムが管理しているテーブルにウィンドウの種類ごとに座標を設定します。ウィンドウは画面上で画像処理の対象となる領域を指定するために使用します。本ライブラリでは各画像処理コマンドのソース画面1、ソース画面2、デスティネーション画面のそれぞれに対応するウィンドウと、画像メモリアクセスコマンドで使用するウィンドウの合計4種類のウィンドウを使用可能です。なお、これらのウィンドウはウィンドウ有効時に、ウィンドウを使用可能な画像処理コマンドと画像メモリアクセスコマンドの全てに対して共通に適用されます。

ウィンドウの幅と高さの最小値はそれぞれ2となります。2未満の場合には正しい画像処理結果を得られない場合があります。また、最大値はチェックしていません。

ウィンドウの座標は画面原点相対で指定して下さい。

また、パイプラインモード中のウィンドウの設定変更は禁止します。



2.2.9 ウィンドウ設定（全種）（SetAllWindows）

□ C言語API

```
int ret = SetAllWindows(int sx, int sy, int ex, int ey);
```

□ パラメータ

int	sx	始点X座標
int	sy	始点Y座標
int	ex	終点X座標
int	ey	終点Y座標

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2 0	始点X座標範囲外
2 1	始点Y座標範囲外
2 2	終点X座標範囲外、始点X座標＞終点X座標と設定
2 3	終点Y座標範囲外、始点Y座標＞終点Y座標と設定
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

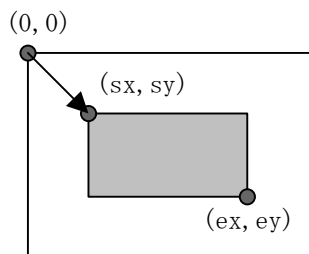
□ 機能

システムが管理しているテーブルに全種ウィンドウの座標を設定します。ウィンドウは画面上で画像処理の対象となる領域を指定するために使用します。本ライブラリでは各画像処理コマンドのソース画面1、ソース画面2、デスティネーション画面のそれぞれに対応するウィンドウと、画像メモリアクセスコマンドで使用するウィンドウの合計4種類のウィンドウを使用可能です。なお、これらのウィンドウはウィンドウ有効時に、ウィンドウを使用可能な画像処理コマンドと画像メモリアクセスコマンドの全てに対して共通に適用されます。

ウィンドウの幅と高さの最小値はそれぞれ2となります。2未満の場合には正しい画像処理結果を得られない場合があります。また、最大値はチェックしていません。

ウィンドウの座標は画面原点相対で指定して下さい。

また、パイプラインモード中のウィンドウの設定変更は禁止します。



2.2.10 ウィンドウリセット (ResetAllWindows)

□ C言語API

```
int ret = ResetAllWindows();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OS システムエラー
1 3 3	アクティブビデオポートが未設定
1 3 6	システム管理データが不正
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

システムが管理しているテーブルに全種ウィンドウの座標を設定します。全てのウィンドウ設定を解除し、現在のビデオフレームサイズに従いウィンドウを再設定します。
 ウィンドウ種類のSRC2_WINとDST_EXT_WINも設定されますが、これらは現在未使用となっています。
 パイプラインモード中のウィンドウの設定変更は禁止します。

2.2.11 ウィンドウ有効 (EnableIPWindow)

□ C言語API

```
int ret = EnableIPWindow();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

全ウィンドウを有効にします。このコマンド発行後の処理（映像入力/表示系を除く）は、ウィンドウサイズで実行されます。

パイプラインモード中のウィンドウの設定変更は禁止します。

2.2.12 ウィンドウ無効 (DisableIPWindow)

□ C言語API

```
int ret = DisableIPWindow();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
1 3 3	アクティブビデオポートが未設定
1 3 6	システム管理データが不正
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

全ウィンドウを無効にします。このコマンド発行後の処理（映像入力/表示系を除く）は、確保したビデオフレームサイズで実行されます。
パイプラインモード中のウィンドウの設定変更は禁止します。

2.2.13 ウィンドウ設定座標読み出し (ReadWindow)

□ C言語API

```
int ret = ReadWindow(enum WindowType type, int *sx, int *sy, int *ex, int *ey);
```

□ パラメータ

enum WindowType	type	ウィンドウ種類
	SRC0_WIN	(0) 画像処理のソース画面 1 のウィンドウ
	SRC1_WIN	(1) 画像処理のソース画面 2 のウィンドウ
	SRC2_WIN	(2) 画像処理のソース拡張画面のウィンドウ (未使用)
	DST_WIN	(3) 画像処理のデスティネーション画面のウィンドウ
	SYS_WIN	(4) 画像メモリ制御コマンドのウィンドウ
	DST_EXT_WIN	(6) 画像処理のデスティネーション拡張画面のウィンドウ (未使用)

int	*sx	始点X座標を返すアドレス

int	*sy	始点Y座標を返すアドレス

int	*ex	終点X座標を返すアドレス

int	*ey	終点Y座標を返すアドレス

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

int	*sx	始点X座標を格納したアドレス

int	*sy	始点Y座標を格納したアドレス

int	*ex	終点X座標を格納したアドレス

int	*ey	終点Y座標を格納したアドレス

□ エラーコード

2 0	ウィンドウ種類設定値範囲外
2 1	始点X座標を返すアドレスが不正
2 2	始点Y座標を返すアドレスが不正
2 3	終点X座標を返すアドレスが不正
2 4	終点Y座標を返すアドレスが不正
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

パラメータで指定されたウィンドウ種類のウィンドウサイズを読み出します。
ウィンドウ種類のSRC2_WINとDST_EXT_WINは、現在未使用となっています。

2.2.14 画像メモリ領域確保(コンポーネントRGB画面) (AllocRGBImg)

□ C言語API

```
IMGID ImgID = AllocRGBImg(enum ImageFrameSize size);
```

□ パラメータ

enum ImageFrameSize size 画像メモリ画面サイズ

画面サイズ	対応定数	X方向	Y方向
IMG_FS_256H_256V	0x01000100	256	256
IMG_FS_256H_512V	0x02000100	256	512
IMG_FS_320H_256V	0x01000140	320	256
IMG_FS_320H_512V	0x02000140	320	512
IMG_FS_512H_256V	0x01000200	512	256
IMG_FS_512H_512V	0x02000200	512	512
IMG_FS_640H_256V	0x01000280	640	256
IMG_FS_640H_512V	0x02000280	640	512
IMG_FS_768H_512V	0x02000300	768	512
IMG_FS_768H_576V	0x02400300	768	576
IMG_FS_1024H_768V	0x03000400	1024	768
IMG_FS_1024H_1024V	0x04000400	1024	1024
IMG_FS_1280H_1024V	0x04000500	1280	1024
IMG_FS_1344H_1024V	0x04000540	1344	1024
IMG_FS_1600H_1024V	0x04000640	1600	1024
IMG_FS_2560H_1920V	0x07800A00	2560	1920
IMG_FS_2560H_2048V	0x08000A00	2560	2048
IMG_FS_2688H_2048V	0x08000A80	2688	2048
IMG_FS_2688H_2176V	0x08800A80	2688	2176

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	—	○	—	—	—	—	◆	—	—

□ リターンパラメータ

IMGID ImgID 確保した画面の画面番号(≧1) または、異常終了 (≦0)

□ エラーコード

2 0 画面サイズ設定値範囲外
4 6 画面領域空きなし
— エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

画像メモリに、パラメータで指定されたサイズのRGB画面領域を確保します。

本コマンドで画像領域を確保した場合の画像設定状態を、下記に示します。

ワーク画面などシステムで使用する空きエリアが必要となるので、そのことを考慮して5画面分空きエリアとすることを推奨します。

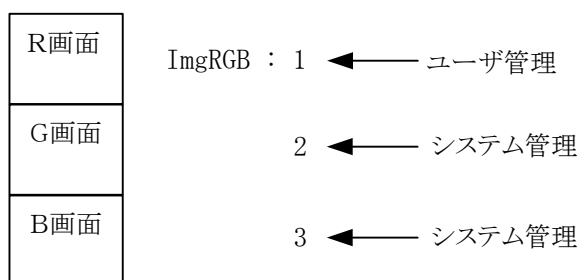
項 目	設 定 値
チャンネル	不定
座標	不定
領域サイズ	パラメータで指定
データ属性	システムデータタイプによる
システムプロセッサからの画像メモリアクセス	アクセス不可

RGBカラー処理用画像メモリの確保は下図のようになります。

[RGBカラー処理用画像メモリ確保例]

```
int  ImgRGB;
```

```
ImgRGB= AllocRGBImg( devID, IMG_FS_512H_512V );
```



RGB画面では色情報は、3画面で1セットとなっています。

確保した画像メモリを映像入力に使用する場合、以下の制限事項があります。

- 使用するカメラの設定が2TAP (Dual)、4TAP、または、8TAPの場合、X方向サイズをかならず128の倍数にする必要がございます。映像入力に使用する画像メモリにはX方向サイズが、320、1344、1600の画像メモリサイズを利用しないでください。

2.2.15 画像メモリ領域確保(16bitRGB画面) (AllocRGB16Img)

□ C言語API

```
IMGID ImgID = AllocRGB16Img(enum ImageFrameSize ImgSize);
```

□ パラメータ

enum ImageFrameSize ImgSize 画面サイズ

画面サイズ	対応定数	X方向	Y方向
IMG_FS_256H_256V	0x01000100	256	256
IMG_FS_256H_512V	0x02000100	256	512
IMG_FS_320H_256V	0x01000140	320	256
IMG_FS_320H_512V	0x02000140	320	512
IMG_FS_512H_256V	0x01000200	512	256
IMG_FS_512H_512V	0x02000200	512	512
IMG_FS_640H_256V	0x01000280	640	256
IMG_FS_640H_512V	0x02000280	640	512
IMG_FS_768H_512V	0x02000300	768	512
IMG_FS_768H_576V	0x02400300	768	576
IMG_FS_1024H_768V	0x03000400	1024	768
IMG_FS_1024H_1024V	0x04000400	1024	1024
IMG_FS_1280H_1024V	0x04000500	1280	1024
IMG_FS_1344H_1024V	0x04000540	1344	1024
IMG_FS_1600H_1024V	0x04000640	1600	1024
IMG_FS_2560H_1920V	0x07800A00	2560	1920
IMG_FS_2560H_2048V	0x08000A00	2560	2048
IMG_FS_2688H_2048V	0x08000A80	2688	2048
IMG_FS_2688H_2176V	0x08800A80	2688	2176

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	—	—	—	—	—	○	◆	—	—

□ リターンパラメータ

IMGID ImgID 確保した画面の画面番号(≧1) または、異常終了 (≦0)

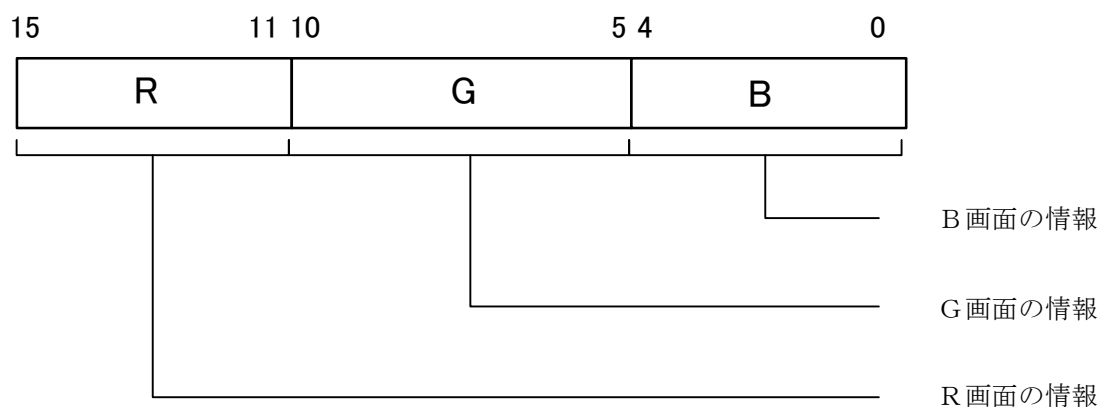
□ エラーコード

2 0 画面サイズ設定値範囲外
 4 6 画面領域空きなし
 — エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

AllocRGB16Imgは1画素あたりの情報量が16bitになります。通常は8bit。
上位から5、6、5bitがR、G、Bの情報になります。



確保した画像メモリを映像入力に使用する場合、以下の制限事項があります。

- 使用するカメラの設定が2TAP (Dual)、4TAP、または、8TAPの場合、X方向サイズをかならず128の倍数にする必要があります。映像入力に使用する画像メモリにはX方向サイズが、320、1344、1600の画像メモリサイズを利用しないでください。

2.2.16 G画面番号抽出 (GetGImgID)

□ C言語API

```
IMGID  GImgID = GetGImgID(IMGID RGBImgID);
```

□ パラメータ

IMGID RGBImgID 対象画面の R G B 画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
RGBImgID	×	○	×	×	×	×	○	○	○
GImgID	—	—	—	○	—	—	○	—	—

ただし、GImgIDはAllocRGBImg() コマンド実行後、画面タイプや画面データタイプの変更をしていないものとします。

□ リターンパラメータ

IMGID GImgID G画面の画面ID (≥1) または、異常終了 (≤0)

□ エラーコード

1 0 指定画面空き領域、または範囲外 (不当画面番号エラー)
— エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

R G B画面の画面番号からG画面の画面番号を読み出します。
カラー画面の処理が必要な場合、またはG画面のみ処理を行いたい場合は、G画面番号をGetGImgID() で取得しR画面とは別に処理を行います。

2.2.17 B画面番号抽出 (GetBImgID)

□ C言語API

```
IMGID BImgID = GetBImgID(IMGID RGBImgID);
```

□ パラメータ

IMGID RGBImgID 対象画面のRGB画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
RGBImgID	—	○	—	—	—	—	○	○	○
BImgID	—	—	—	—	○	—	○	—	—

ただし、BImgIDはAllocRGBImg() コマンド実行後、画面タイプや画面データタイプの変更をしていないものとします。

□ リターンパラメータ

IMGID BImgID B画面の画面ID (≥1) または、異常終了 (≤0)

□ エラーコード

1 0 指定画面空き領域、または範囲外 (不当画面番号エラー)
— エラーリセットコマンド (ClearIPError) 未発行

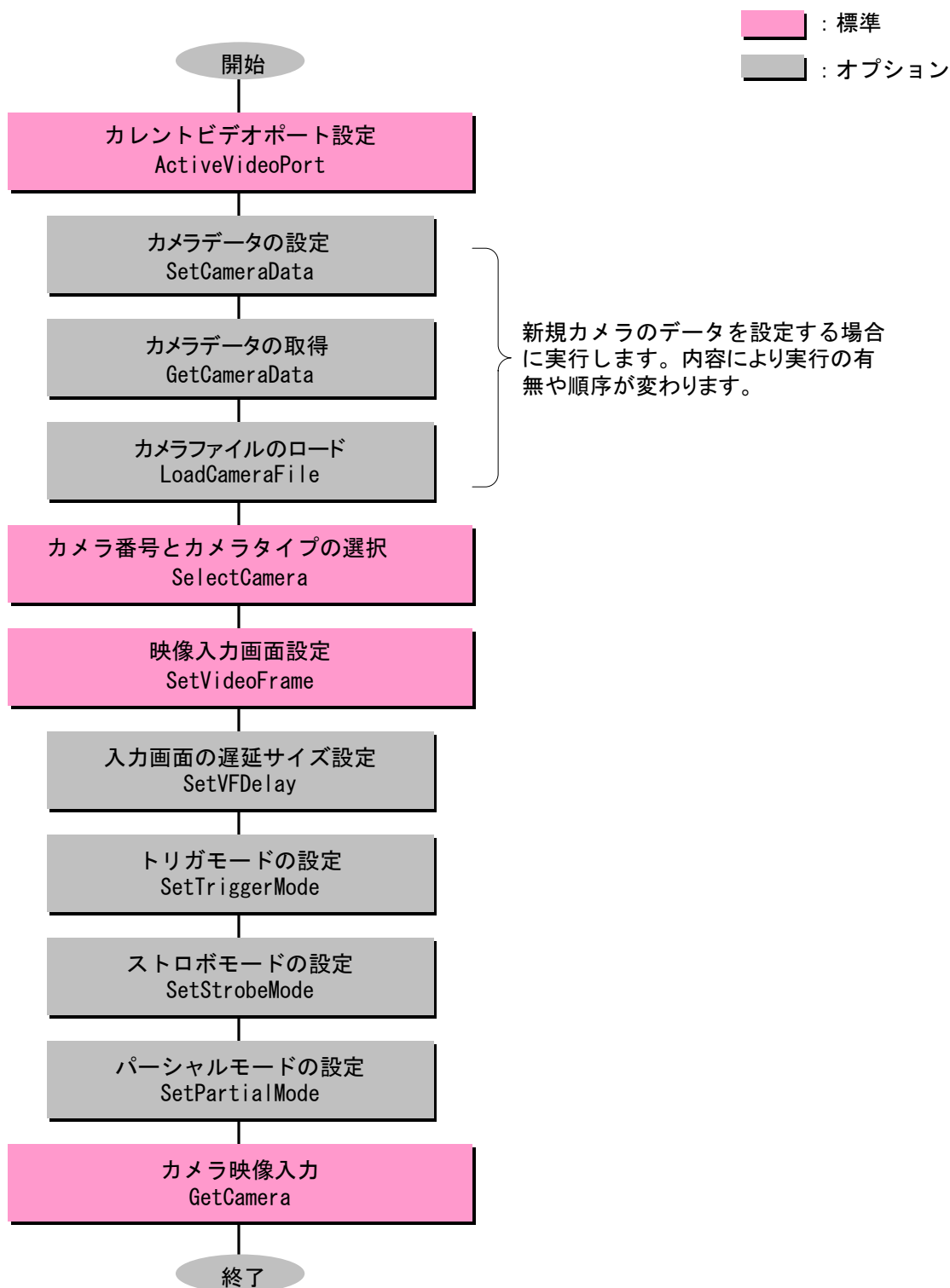
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

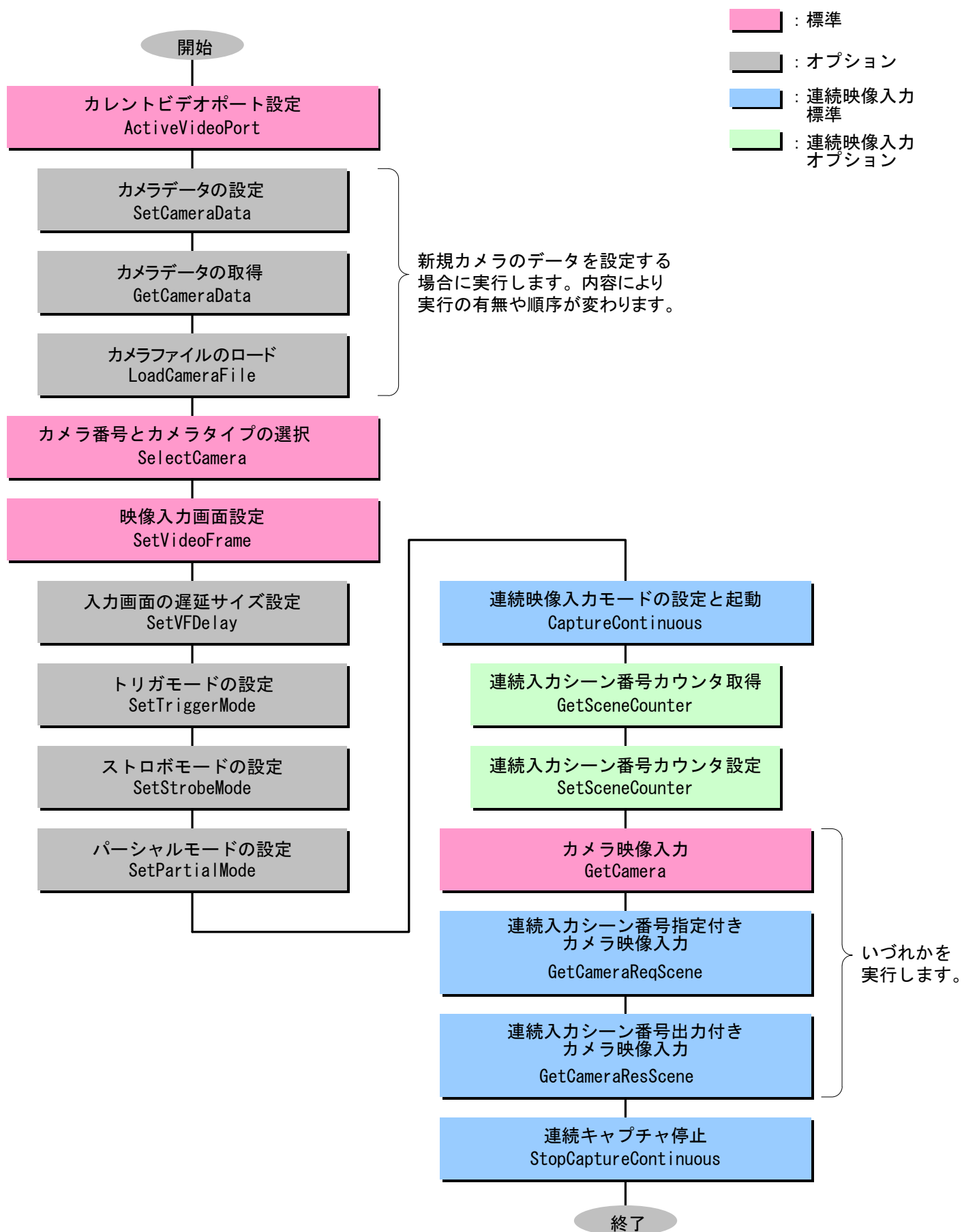
RGB画面の画面番号からB画面の画面番号を読み出します。
カラー画面の処理が必要な場合、またはB画面のみ処理を行いたい場合は、B画面番号をGetBImgID() で取得しR画面とは別に処理を行います。

2.3 映像入力

<単一映像入力手順>



<連続映像入力手順>



2.3.1 ビデオポート選択（ActiveVideoPort）

□ C言語API

```
int ret = ActiveVideoPort(enum VideoPortID vpid);
```

□ パラメータ

enum VideoPortID vpid ビデオポート番号

ビデオポートNo.	対応定数	内容
VIDEO_PORT0	0	ビデオポート#0
VIDEO_PORT1	1	ビデオポート#1
VIDEO_PORT2	2	ビデオポート#2
VIDEO_PORT3	3	ビデオポート#3

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2 OSシステムエラー
 20 パラメータエラー
 131 V I Nドライバでのエラー
 - エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

カレントのビデオポート（ビデオポート#0～ビデオポート#3）を設定します。
 映像入力コマンドを使用する場合、必ず本コマンドでビデオポートの指定を行って下さい。
 カメラチャネルCH1～CH4がそれぞれビデオポート#0～#3に対応します。

カメラチャネルCH1から映像入力する場合、ビデオポート#0を指定してください。同様に、
 カメラチャネルCH2、CH3、CH4はビデオボード#1、#2、#3に対応します。

2.3.2 映像入力画面設定 (SetVideoFrame)

□ C言語API

```
int ret = SetVideoFrame(enum Interlace mode, enum VideoFrameSize size);
```

□ パラメータ

enum Interlace mode 走査方式

走査方式	対応定数	内容
NONINTERLACE	0	ノンインタレース

enum VideoFrameSize size 映像入力サイズ (ビデオフレームサイズ)

映像入力サイズ	対応定数	映像入力サイズ		備考
		H	V	
VIDEO_FS_256H_220V	0x00DC0100	256	220	
VIDEO_FS_512H_220V	0x00DC0200	512	220	
VIDEO_FS_256H_440V	0x01B80100	256	440	
VIDEO_FS_512H_440V	0x01B80200	512	440	
VIDEO_FS_256H_240V	0x00F00100	256	240	
VIDEO_FS_320H_240V	0x00F00140	320	240	
VIDEO_FS_512H_240V	0x00F00200	512	240	
VIDEO_FS_640H_240V	0x00F00280	640	240	
VIDEO_FS_256H_480V	0x01E00100	256	480	
VIDEO_FS_320H_480V	0x01E00140	320	480	
VIDEO_FS_512H_480V	0x01E00200	512	480	デフォルト
VIDEO_FS_640H_480V	0x01E00280	640	480	
VIDEO_FS_1024H_768V	0x03000400	1024	768	
VIDEO_FS_768H_512V	0x02000300	768	512	
VIDEO_FS_768H_576V	0x02400300	768	576	
VIDEO_FS_2432H_2048V	0x08000980	2432	2048	

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
2 0	走査方式設定値範囲外
2 1	映像入力サイズ設定値範囲外
1 3 1	V I Nドライバでのエラー
1 3 2	D Uドライバでのエラー
1 3 3	アクティブビデオポートまたは、連続映像入力時(CaptureContinuous)、未設定
1 3 8	カメラ選択未実施
—	エラーリセットコマンド（ClearIPErrors）未発行

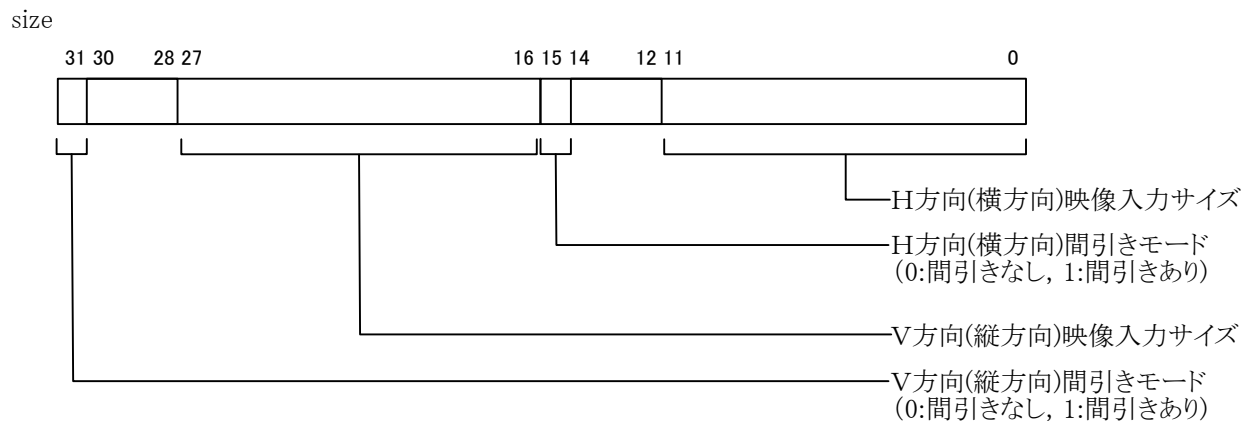
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

本コマンドは、SelectCameraの実行後に実行してください。

カメラ、表示映像での走査方式、映像入力サイズ(ビデオフレームサイズ)を、パラメータに従い設定します。なお、連続取り込みが起動されている場合はエラーになります。

間引きは無効です。間引きなしに設定して下さい。



H方向(横方向)映像入力サイズは、32バイトの倍数に設定する必要があります。32バイトの倍数でない場合、32バイトの倍数に切り上げて設定されます。

V方向(縦方向)映像入力サイズは、TAP数の倍数(*1)に設定する必要があります。TAP数の倍数でない場合、TAP数の倍数に切り上げて設定されます。

映像入力サイズの設定値がカメラのフレームサイズを超えないように設定してください。

*1 1TAP/2TAP(Singleモード)の場合、1の倍数

2TAP(Dualモード)の場合、2の倍数

4TAP/8TAPの場合、4の倍数

● 走査方式について

走査方式はNONINTERLACEで使用して下さい。(カメラリンク対応カメラ)
カメラリンクのTAPおよびTAPモードによって画像メモリサイズ、ビデオフレームサイズに制限があります。

● 画像メモリサイズ、ビデオフレームサイズ制限

	カメラリンクTAP、 TAPモード	映像入力をする 画像メモリサイズ制限	ビデオフレームサイズ制限
1	1TAP 2TAP(Singleモード) RGBカラー	特になし	Hサイズが32byteアライン 2ライン以上、最大2048ライン
3	2TAP(Dualモード)	Hサイズは128byteアライン 8bit/画素の場合16画素単位	Hサイズが32byteアライン 4ライン以上、2ライン単位、最大4096ライン
4	4TAP/8TAP	Hサイズは128byteアライン 8bit/画素の場合16画素単位	Hサイズが32byteアライン 8ライン以上、8ライン単位、最大8192ライン

※その他の画像メモリサイズ制限は、その他画像メモリ確保コマンドを参照して下さい。

指定したビデオフレームサイズに対し、Hサイズが32byteアラインでない場合、32byteアラインに調整されたサイズで映像入力されますのでご注意ください。(ex. Hサイズを1020とした場合1024がビデオフレームサイズとなります。)また、ビデオフレームサイズがカメラの有効画素を超えないように設定して下さい。カメラの有効画素は、『SetCameraData』『LoadCameraFile』で指定します。

指定するビデオフレームサイズは、SelectCamera、SetCameraData、または、LoadCameraFileにて設定される最大フレームサイズに対して、

- ・ 垂直方向

1TAP、2TAP(single)の場合： 最大フレームサイズ(垂直方向) $\geq$ ビデオフレーム垂直方向サイズ+1
2TAP(dual)の場合 : 最大フレームサイズ(垂直方向) $\geq$ ビデオフレーム垂直方向サイズ+2
4TAP/8TAPの場合 : 最大フレームサイズ(垂直方向) $\geq$ ビデオフレーム垂直方向サイズ+4

- ・ 水平方向

最大フレームサイズ(水平方向) $\geq$ ビデオフレーム水平方向サイズ

となるように設定して下さい。

2.3.3 カメラ番号とカメラタイプの選択 (SelectCamera)

□ C言語API

```
int ret = SelectCamera(enum CameraID id, enum CameraType type);
```

□ パラメータ

enum CameraID id カメラポート番号

カメラポートNo.	対応定数	内容
CAMERA_PORT0	0	カメラポート#0

カメラ同時入力情報	対応定数	内容
CAMERA_COMPONENT1	0x00010000	2カメラ同時入力 (CH1&CH2またはCH3&CH4)
CAMERA_COMPONENT2	0x00020000	2カメラ同時入力 (CH1&CH3またはCH2&CH4)
CAMERA_COMPONENT3	0x00030000	3カメラ同時入力 (CH1&CH2&CH3)
CAMERA_COMPONENT4	0x00040000	4カメラ同時入力 (CH1&CH2&CH3&CH4)

enum CameraType type カメラタイプ

カメラタイプNo.	対応定数	内容	備考
SONY_XCL_C500	0	モノクロカメラ	デフォルト(2TAP, 2448x2048)
SONY_XCL_C130	1	モノクロカメラ	デフォルト(2TAP, 1280x960)
–	2～127	リザーブ	拡張用
–	128～131 (0x80～0x83)	ユーザ定義カメラ	

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (–1)

□ エラーコード

2 OSシステムエラー
7 指定したユーザー定義カメラのカメラデータのカメラ属性 (CamAtr) が範囲外
2 0 カメラポート番号設定値範囲外
2 1 カメラタイプ設定値範囲外
1 3 1 V I Nドライバでのエラー
1 3 2 D Uドライバでのエラー
1 3 3 アクティブビデオポートが未設定
1 3 8 連続映像入力動作時 (CaptureContinuous)
2 0 0 PLL設定エラー
– エラーリセットコマンド (ClearIPErrors) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

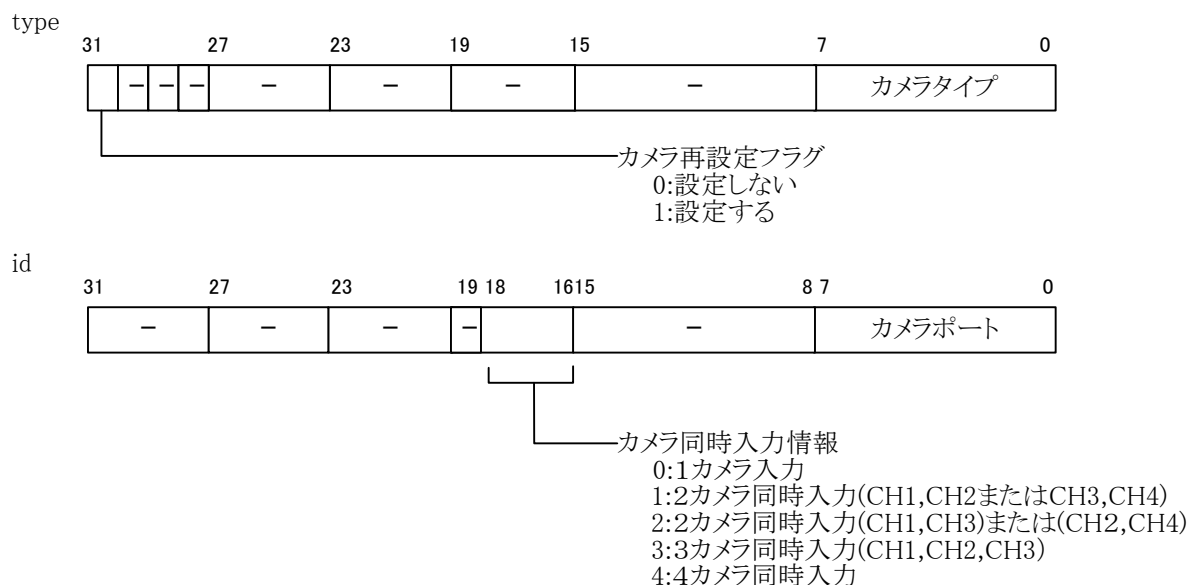
□ 機能

設定されたカメラポート番号にカメラ入力を切り替えます。

本コマンドは、必ず、ActiveVideoPortの実行後に実行してください。

パラメータidは、カメラ番号以外にbit16～bit18にカメラの同時入力情報を設定することができます。カメラ同時入力を行う場合は、カメラポート情報と同時入力情報のORをとり、idに設定してください。

カメラ再設定フラグ機能は無効です。本ビットに関係なく、本コマンドが呼ばれた場合、引数に従い画像処理ボードの設定を行います。



カメラチャネル(CH)とビデオポートおよびカメラポートの関係を次の表に示します。

カメラチャネル	ビデオポート	カメラポート
CH1	VIDEO_PORT0	CAMERA_PORT0
CH2	VIDEO_PORT1	
CH3	VIDEO_PORT2	
CH4	VIDEO_PORT3	

カメラリンクのTAPおよびVINモードによる同時入力可否を表に示します。

	カメラリンクTAP	1カメラ	2カメラ同時入力		3カメラ 同時入力 CH1～CH3	4カメラ 同時入力 CH1～CH4
			CH1&CH2 CH3&CH4	CH1&CH3 CH2&CH4		
1	1TAP	○※1	○	○	○	○
2	2TAP(Singleモード)	○※1	○	○	○	○
3	2TAP(Dualモード) ※2	○	○※3			
4	4TAP/8TAP ※4					
5	RGBカラー	○※5				

※1 各チャネルで使用が可能です。

※2 CH1およびCH2で使用が可能です。CH1で使用する場合、CH3での映像入力はできません。CH2で使用する場合、CH4での映像入力はできません。

※3 CH1&CH2の同時入力のみ可能です。

※4 未サポートです。

※5 CH1でのみ使用が可能です。その際CH2とCH3での映像入力はできません。

同時入力設定をする場合、同時入力したいCHのもっとも若い番号に対し設定を行ってください。
(ex.CH2とCH4の同時入力を行う場合、CH2に対し『SelectCamera』を実行します。)

SelectCameraコマンド設定例を示します。

例1) 1カメラ映像入力 (SONY_XCL_C130 (2TAP-Single), ノンインタレースモード, CH1, 映像サイズ512×480)

```
ImgID = AllocImg( IMG_FS_512H_512V );  
ActiveVideoPort( VIDEO_PORT0 );  
SelectCamera( CAMERA_PORT0, SONY_XCL_C130 );  
    < カメラ切り替えのためのウェイト >  
SetVideoFrame( NONINTERLACE, VIDEO_FS_512H_480V );  
GetCamera( ImgID );
```

2.3.4 カメラ映像入力 (GetCamera)

□ C言語API

```
int ret = GetCamera(IMGID ImgID);
```

□ パラメータ

IMGID ImgID カメラ映像を入力する画面の画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	×	×	×	×	×	◇S	◇S	—
	×	○	×	×	×	×	◆	—	—

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

2	OS システムエラー
8	V I N 入力タイムアウト
1 0	番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー) 、
1 1 3	SetConfigCamera で CFG_VFW_CAMERA と CFG_AVI_FILE の場合、 『VFWCam2. exe』 ツール通信エラー
1 3 1	V I N ドライバでのエラー
1 3 2	D U ドライバでのエラー
1 3 3	アクティブビデオポートが未設定
1 3 5	画面サイズが不正、またはカメラタイプと画面タイプの対応が不正
1 3 6	システム管理データが不正
1 3 7	画面管理データが不正
1 3 8	映像入力画面設定コマンド (SetVideoFrame) 未発行 SetConfigCamera で CFG_VFW_CAMERA と CFG_AVI_FILE の場合、 『VFWCam2. exe』 ツール未起動
1 3 9	映像入力タイムアウト時間設定値 (システムで定義している CFG_VINTMO の値) が 0 未満
—	エラーリセットコマンド (ClearIPErrors) 未発行

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

カメラから画像メモリに、映像を取り込みます。
2 カメラ以上の同時入力の場合は、GetElementImgID () で画面番号を取得してください。

モノクロカメラの映像入力で入力画面に Y 画面以外を指定した場合はエラー出力します。
R G B カラーカメラの映像入力で入力画面に RGB 画面以外を指定した場合はエラー出力します。

入力画面 (ImgID) のサイズと映像入力サイズ (ビデオフレームサイズ) は次の条件で設定してください。これ以外の場合はエラーとなり、本処理は行われません。

入力画面サイズ ≥ 映像入力サイズ (ビデオフレームサイズ)

2TAP(dual), 4TAP/8TAP モードの場合、入力画面 X サイズは 128 の倍数として下さい。

本ライブラリの映像入力コマンドでは、単一キャプチャモードと連続キャプチャモードをサポートしています。

● 単一キャプチャモード：

映像入力起動時に1回だけ画像メモリに映像を入力します。
カメラ毎のフレームレートで映像入力します。

● 連続キャプチャモード：

『CaptureContinuous()』コマンドで指定した画像メモリ（3画面）に常に映像入力が行われており、『GetCamera()』コマンド発行時に取込が終了した最新のフレームの映像が取り込まれます。常に映像を先行入力しているため、『GetCamera()』コマンド発行から映像入力取得までの時間は0ms～カメラ毎のフレームレートで映像入力します。連続で映像入力を行った場合はカメラ毎のフレームレートで映像入力します。

また、連続キャプチャモードを使用し、プリフェッチ映像入力が可能です。

詳細については、『CaptureContinuous()』コマンドを参照して下さい。

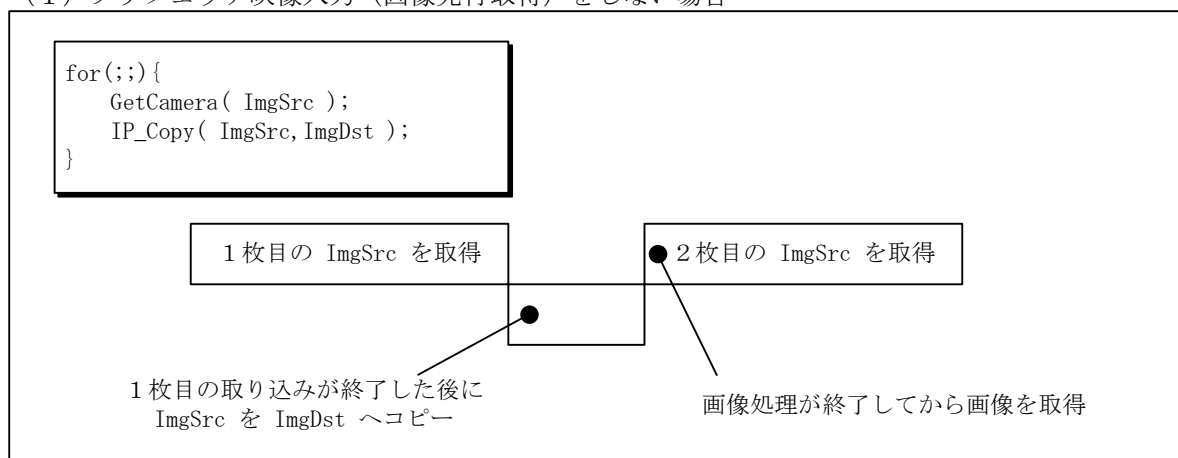
<プリフェッチ映像入力について>

通常の画像処理アプリケーションでは、映像入力を行いその入力した映像に対し画像処理を実行するため、映像入力と画像処理を順番に行っています。ビデオレートで画像処理アプリケーションを実行させるには、映像入力と画像処理をパイプラインで実行する必要があります。

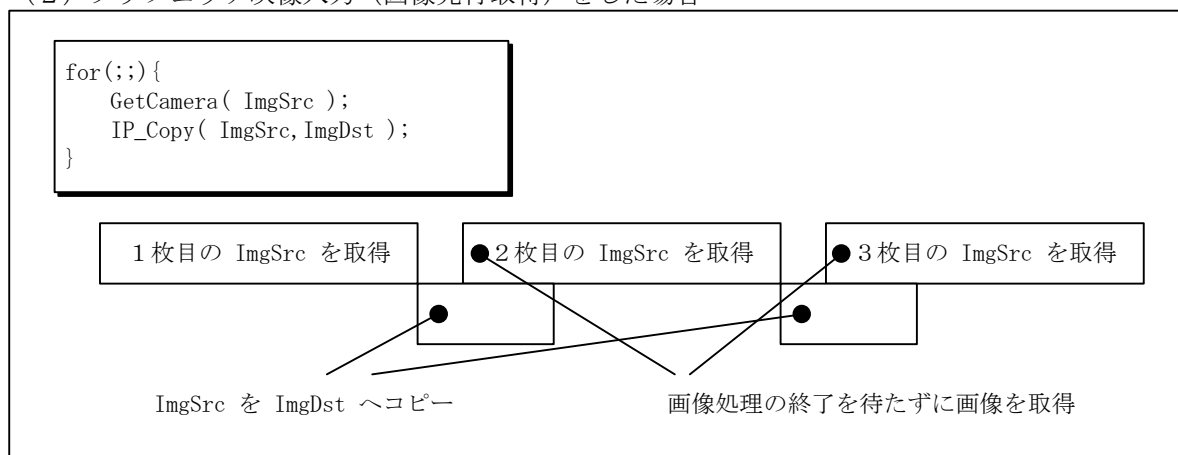
プリフェッチ映像入力は、画像先行取得を行うことで映像入力と画像処理をパイプライン処理し、全体的な処理速度の向上を図るものです。

以下にプリフェッチ映像入力による画像処理の例を示します。

(1) プリフェッチ映像入力（画像先行取得）をしない場合



(2) プリフェッチ映像入力（画像先行取得）をした場合



2.3.5 カメラ入力画面の遅延サイズ設定（SetVFDelay）

□ C言語API

```
int ret = SetVFDelay(int HOffset , int VOffset );
```

□ パラメータ

int HOffset

遅延横サイズ(有効画像領域左上端からの水平画素数)からのオフセット

int VOffset

遅延縦サイズ(有効画像領域左上端からの垂直ライン数)からのオフセット

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
2 0	走査方式設定値範囲外
2 1	映像入力サイズ設定値範囲外
1 3 3	アクティブビデオポートが未設定
1 3 6	システム管理データが不正
1 3 8	映像入力画面設定コマンド（SetVideoFrame）未発行 または、連続映像入力動作時（CaptureContinuous）
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

映像画面の遅延サイズを設定します。

カメラ入力画面の遅延サイズ(hdly,vdly)は、カメラとビデオフレームサイズにより決定されます。

SelectCamera()とSetVideoFrame()を実行することで標準値が設定されますが、さらにカメラからの映像入力画面の調整が必要な場合には、本コマンドで標準遅延サイズからのオフセット値を設定して調整します。

遅延サイズは、カメラリンクTAPモードによって制限があるため調整されます。下表「遅延サイズ制限」の項を参照してください。

● 標準値について

標準値 = (カメラの有効画素サイズ - ビデオフレームサイズ) / 2

※カメラの有効画素(最大フレームサイズ)は、『SetCameraData』『LoadCameraFile』で設定した値となります。割り切れない場合小数部は切り捨てられます。

● 遅延サイズ制限

	カメラリンクTAP、 TAPモード	遅延サイズの制限
1	1TAP、2TAP (Singleモード) RGBカラー	特になし ※1
2	2TAP (Dualモード)	2ライン単位 ※1
3	4TAP/8TAP	4ライン単位

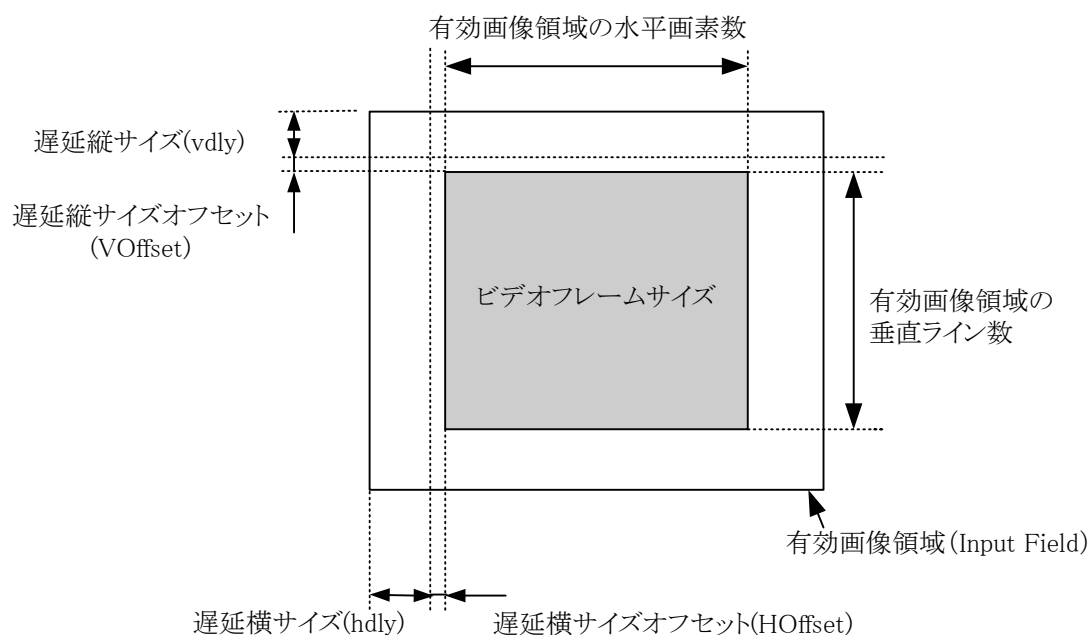
ex. カメラの有効画素サイズが1282x962、ビデオフレームサイズが640x480で2TAP (Dualモード) の場合標準値は、

遅延横サイズ(xdly) = (1282 - 640) / 2 = 321

遅延縦サイズ(vdly) = (962 - 480) / 2 = 241→240(2ライン単位に調整されます。)

となります。また、例えばこの標準値に対し VOffset = 1とした場合、240のままとなります。Voffset = 2とした場合242となります。

※1 TAPフォーマットが「2TAP-2X1Y-BLK」「2TAP-2X1Y-BLK-MR」の場合は、HOffsetは、2画素単位として下さい。



vdly,hdly : ビデオコーダからの有効画像領域(Input Field)の左上を基準として、カメラからの映像入力時の有効映像領域の開始点を遅延サイズとします。SelectCamera()とSetVideoFrame()を実行することで標準値が設定されます。

2.3.6 カメラ映像入力トリガモードの設定 (SetTriggerMode)

□ C言語API

```
int ret = SetTriggerMode( enum TrigerMode mode , int speed );
```

□ パラメータ

enum TrigerMode mode トリガモード

トリガモード	対応定数	説明
NOTRIGER	0	トリガ無し（スルー映像出力）
TRIGER	2	トリガ内部起動
DI_POSI	0x10	トリガD I 入力起動（正極性）
DI_NEGA	0x11	トリガD I 入力起動（負極性）
DI_THROUGH	0x100000	D I スルー出力※

※DIスルー出力は、他の設定とOR指定が可能です。

int speed シャッタスピード(μ S)(10～640000)

※シャッタスピードは10 μ S単位で設定されます(10 μ S未満切り捨て)。
例えば、speed = 85 μ Sとした場合シャッタスピードは80 μ Sになります。

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
2 0	トリガモード設定値範囲外
2 1	シャッタスピード設定値範囲外
1 3 1	V I Nドライバでのエラー
1 3 2	D Uドライバでのエラー
1 3 3	アクティブビデオポートが未設定
1 3 8	映像入力画面設定コマンド (SetVideoFrame) 未発行 または、映像入力画面未設定または連続映像入力動作時 (CaptureContinuous)
—	エラーリセットコマンド (ClearIPErrors) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

フレームシャッターカメラのトリガモードとカメラのシャッタースピードを設定します。トリガモードで映像を入力する場合は、SelectCamera()、SetVideoFrame()後に本コマンドで TRIGGERモードに設定してから GetCamera() コマンドで画像入力を行います。

SelectCamera() コマンドを発行するとTRIGGERモードが解除されますので、再設定して下さい。

(1) トリガモード

カメラに出力するトリガ信号を指定します。トリガ信号は、カメラのCC1に出力されます。NOTRIGGER(トリガ無し)の場合出力されません。また、DI_THROUGH(DIスルー出力)を有効にした場合、CC2にDIスルー信号が出力されます。

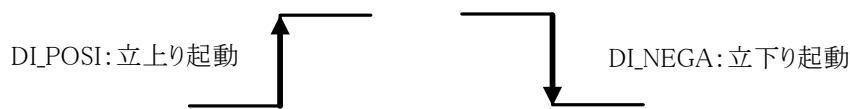
・TRIGGER(トリガ内部起動)

トリガ起動は、コマンドで生成されます。『GetCameraContinuous』による連続取り込みと、『DispCamera』のカメラ表示においては、『SetCameraData』または『LoadCameraFile』で設定する周期でトリガを起動します。シングル取り込みの場合は『GetCamera』毎に単発起動されます。

・DI_POSI/DI_NEGA(トリガDI入力起動)

トリガ起動は、DIの立ち上がりまたは立下りによって起動されます。トリガDI入力起動時の対応するDI信号は以下の対応となります。

CH1 - DI0、CH2 - DI1、CH3 - DI2、CH4 - DI3



・DI_THROUGH(DIスルー出力)

DIに入力された信号をカメラにそのまま出力します。DIスルー信号によってトリガ信号の起動は行いません。また、本設定はほかのトリガモードとOR指定が可能です。

DIスルー出力時のカメラに出力するDI信号は以下の対応となります。

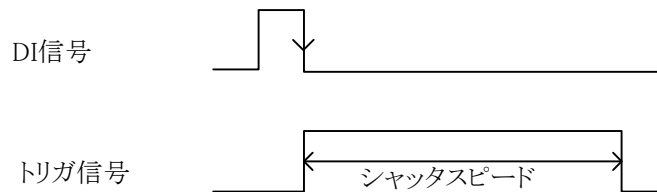
CH1 - DI4、CH2 - DI5、CH3 - DI6、CH4 - DI7

DI信号で映像入力を行う場合、映像取り込みコマンドを発行後一定期間内にトリガを起動しないとタイムアウトエラーとなります。適切なタイミングで映像入力を行ってください。

I/O入力による映像入力起動を解除する場合にはSelectCamera()コマンドを実行するか、SetTriggerMode()でTRIGGERに再設定して下さい。

(2) シャッタースピード

トリガ有効時(トリガDIスルーを除く)に、トリガ起動パルスによって出力するトリガ信号のパルス幅を指定します。



トリガDI立下りでの例

同時取り込みの場合、アクティブポートに対し本コマンドを発行して下さい。それにより同時取り込みを行うすべてのカメラに対し同じ設定がされます。DIスルー出力については、同時入力の場合でも各CHに対応するDI信号が出力されます。

2.3.7 カメラ映像入力ストロボモードの設定 (SetStrobeMode)

□ C言語API

```
int ret = SetStrobeMode( int ch , enum StrobeMode mode , STRB_INFO *strbinfo );
```

□ パラメータ

Int ch ストロボチャンネル(0:CH0, 1:CH1, 2:CH2, 3:CH3)

enum StrobeMode mode ストロボモード

ストロボモード	対応定数	説明
NOSTROBE	0	ストロボ無効
STROBE	1	ストロボ有効
MODE_ON	2	ストロボオン
MODE_OFF	3	ストロボオフ
MODE_STROB_FIRST	0x00010000	ストロボ先行モード(OR指定可能)

※ ストロボ先行モードは他のモードとOR指定となります。指定しない場合(デフォルト)はトリガ先行モードとなります。

STRB_INFO *strbinfo ストロボ情報テーブルを格納したアドレス

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ パケットの構造

```
typedef struct {
    int      VideoPort;      ビデオポート番号 (0:ビデオポート#0～3:ビデオポート#3)
    int      Delay;          出力ディレイ (μ S) (0～640000)
    int      Length;         ストロボ出力長 (μ S) (10～640000)
    int      Pole;           ストロボ極性 (0:正極性, 1:負極性)
} STRB_INFO;
```

※ ストロボ出力ディレイおよびストロボ出力長は10 μ S単位で設定されます(10 μ S未満切り捨て)。例えば、Length = 85 μ Sとした場合ストロボ出力長は80 μ Sになります。

□ エラーコード

2 0 ストロボチャンネル設定値範囲外
 2 1 ストロボモード設定値範囲外
 2 2 ストロボ情報テーブル設定値範囲外
 - エラーリセットコマンド (ClearIPErr) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

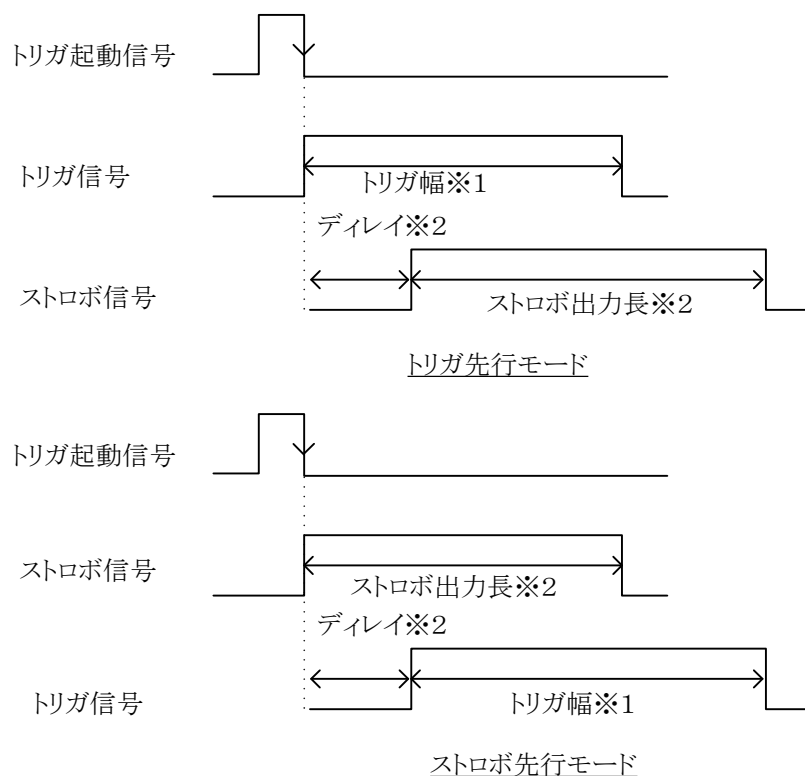
ストロボモードで映像入力する場合、SetTriggerMode()でトリガモードを設定し、本コマンドでストロボ情報を設定してから映像入力を行います。SelectCamera()コマンドを発行するとトリガモードが解除されますので、注意して下さい。

ストロボは4チャンネルあり、それぞれ独立にストロボ情報を設定することができます。

CH2,CH3のストロボ信号を使用する場合、SetDioModeでのDIO端子設定をあわせて行なってください。

同時取り込みの場合、アクティブビデオポートに設定されたストロボ信号を出力します。但し、出力するチャンネル毎に設定が必要です。その際STRB_INFOのビデオポート番号は、アクティブビデオポートを指定して下さい。その他の設定は同じにしてください。ストロボ無効に設定する場合も、ストロボ有効に設定した場合と同じ値をストロボ情報テーブルに設定してください。

また、modeにMODE_ONあるいはMODE_OFFを指定した場合は、直接ストロボをオン／オフします。ストロボチャンネル毎個別に設定する場合は、ストロボ制御テーブルにはNULLを設定するか、ビデオポート番号に自チャンネル番号を指定して下さい。



※1 トリガ幅は、SetTriggerMode()で設定します。

※2 ディレイ、ストロボ出力長は本関数のSTRB_INFO構造体で設定します。

2.3.8 カメラ映像入力パーシャルモードの設定（SetPartialMode）

□ C言語API

```
int ret = SetPartialMode( enum PartialMode mode , void *pinfo , int len);
```

□ パラメータ

enum PartialMode mode パーシャルモード

パーシャルモード	対応定数	説明
FIX_LINE_MODE	1	固定ラインモード

void *pinfo パーシャル情報テーブルを格納したアドレス

int len パーシャル情報テーブルサイズ

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
2 0	パーシャルモード設定値範囲外
2 1	パーシャル情報テーブル設定値範囲外
2 2	パーシャル情報テーブルサイズ範囲外
1 3 1	V I Nドライバでのエラー
1 3 2	D Uドライバでのエラー
1 3 3	アクティブビデオポートが未設定
1 3 6	システム管理データが不正
1 3 8	映像入力画面設定コマンド（SetVideoFrame）未発行 または、連続映像入力動作時（CaptureContinuous）
—	エラーリセットコマンド（ClearIPErrors）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

固定シャッターモードの場合のパーシャル情報テーブルのフォーマットを下記に示します。

```
typedef struct {  
    int      Delay;   映像の縦方向遅延画素数(0～1023)  
    int      Length;  映像の縦サイズ(0～4095)  
} FIX_LINE_INFO;
```

□ 制限事項

本コマンドはサポートされません。

2.3.9 ビデオ入力LUTの設定（SetVideoLUTMode）

□ C言語API

```
int ret = SetVideoLUTMode( enum CameraLutID id , int mode , int *lut , int len , int opt );
```

□ パラメータ

enum CameraLutID id

LUT番号

カメラLUTNo.	対応定数	内容
CAMERA_LUT0	0	カメラLUT#0
CAMERA_LUT1	1	カメラLUT#1
CAMERA_LUT2	2	カメラLUT#2
CAMERA_LUT3	3	カメラLUT#3

int mode

LUTモード(0:無効、1:有効)

int *lut

LUTへ書き込むデータテーブルを格納したアドレス
sizeof(int)*len分の領域を確保し、データを設定して下さい

int len

書き込みデータ数。カメラの出力ビット長に合わせて設定して下さい
8bit:len = 256
10bit:len = 1024
12bit:len = 4096
但し10/12bitは未サポート

int opt

将来拡張用。0を指定してください

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int

ret

正常終了（0）または異常終了（-1）

□ エラーコード

2 0 LUT番号設定値範囲外
 2 1 LUTモード設定値範囲外
 2 3 書き込みデータ数設定値範囲外
 2 4 オプション設定値範囲外
 - エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ビデオ入力濃度変換メモリへのデータ書き込みを行います。
カメラLUT#0～#3は、カメラチャネルCH1～CH4に対応しています。カメラCH1からの映像を濃度変換する場合はカメラLUT#0、カメラCH2からの映像を濃度変換する場合はカメラLUT#1のようにそれぞれ設定してください。

カメラ接続チャネルおよびTAP別にLUT設定が必要なチャネルを表に示します。

	カメラ設定	カメラ 接続先	設定が必要なLUT番号			
			LUT0	LUT1	LUT2	LUT3
1	1TAP, 2TAP(Single)	CH1	○			
		CH2		○		
		CH3			○	
		CH4				○
2	2TAP(Dual)	CH1	○		○	
		CH2		○		○
3	RGBカラー	CH1	○	○	○	
4	4TAP/8TAP	CH1	○	○	○	○

※1 2TAP(Dual),4TAP/8TAPの場合、設定が必要なLUT番号に対し同じ設定をしてください。

※2 RGBカラーの場合、LUT0～LUT2がR,G,Bのどのデータを返還対象にするかは以下の通りです。

CameraLink PortA : LUT0

CameraLink PortC : LUT1

CameraLink PortB : LUT2

例えばPortAにRが出力されるカメラの場合、LUT0の設定がRデータに対し有効となります。

R,G,Bがどのポートに出力されるかは、カメラの仕様をご確認ください。

2.3.10 連続入力シーン番号カウンタ取得 (GetSceneCounter)

□ C言語API

```
int ret = GetSceneCounter( unsigned long *SceneCnt );
```

□ パラメータ

unsigned long *SceneCnt シーン番号のカウンタ値を格納したアドレス

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)
unsigned long	*SceneCnt	シーン番号のカウンタ値を格納したアドレス

□ エラーコード

2	OSシステムエラー
2 0	シーン番号のカウンタ値格納アドレス範囲外
1 3 1	VINドライバでのエラー
1 3 3	アクティブビデオポートが未設定
—	エラーリセットコマンド (ClearIPErrors) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

連続映像入力モードでは、映像入力のシーン番号を管理しています。
本コマンドは、コマンド実行時の連続入力シーン番号のカウンタ値を取得します。

2.3.11 連続入力シーン番号カウンタ設定 (SetSceneCounter)

□ C言語API

```
int ret = SetSceneCounter( unsigned long SceneCnt );
```

□ パラメータ

unsigned long SceneCnt	シーン番号のカウンタ値

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

2	OSシステムエラー
1 3 1	VINドライバでのエラー
1 3 3	アクティブビデオポートが未設定
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

連続映像入力モードでは、映像入力のシーン番号を管理しています。
本コマンドは、連続入力シーン番号のカウンタ値を任意の値に設定することができます。

2.3.12 カメラデータの設定 (SetCameraData)

□ C言語API

```
int ret = SetCameraData( int type , CAMERA_DATA_TBL *pDataTbl );
```

□ パラメータ

int	type	カメラタイプ
CAMERA_DATA_TBL	*pDataTbl	映像入力用カメラデータテーブルのポインタを格納したアドレス

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ パケットの構造

```
typedef struct {
    VIDEO_CAM_FRM_INFO    CamFrmInfo;    カメラデータフォーマット情報
    VIDEO_FRM_TBL         Frmtbl;         映像入力画面サイズテーブル
    VIDEO_CAM_TBL         Camtbl;         カメラ情報テーブル
    VIDEO_CLINK_MODE_TBL  ClnkModeTbl;    カメラリンク設定テーブル
    VIDEO_VPP_DATA_TBL    VppDataTbl;    VPP設定テーブル
} CAMERA_DATA_TBL, *PCAMERA_DATA_TBL;
```

□ エラーコード

2 0	カメラタイプ設定値範囲外
2 1	映像入力用カメラデータテーブルが不正

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

カメラタイプと映像入力用カメラデータの設定を行います。

本コマンドは、サポートカメラのデフォルトデータを変更する場合や新しいカメラデータを設定する場合に用います。カメラタイプtypeはSelectCameraで設定可能な値を設定してください。新しいカメラデータは4つまで設定することができます。カメラタイプ(type)は0x80～0x83を使用します。

本コマンドを使用する場合には、以下のファイルをインクルードしてください。

```
#include "video.h"
```

(1) カメラデータフォーマット情報

```
typedef struct {
    unsigned long    inf1;    /* カメラフォーマット情報1 */
    unsigned long    inf2;    /* カメラフォーマット情報2 */
} VIDEO_CAM_FRM_INFO, *PVIDEO_CAM_FRM_INFO;
```

- ・カメラフォーマット情報1／カメラフォーマット情報2 は、以下の固定値を必ず設定して下さい。

```
inf1 = 0x00000002;
inf2 = 0x00000000;
```

(2) 映像入力画面サイズテーブル

```
typedef struct {
    VIDEO_FRM_SIZE    pluslng; /* 水平垂直周期(画素数) */
    VIDEO_FRM_SIZE    maxfs;   /* 最大フレームサイズ */
} VIDEO_FRM_TBL, *PVIDEO_FRM_TBL;

typedef struct {
    unsigned short    Vsize;    /* 垂直方向サイズ */
    unsigned short    Hsize;    /* 水平方向サイズ */
} VIDEO_FRM_SIZE, *PVIDEO_FRM_SIZE;
```

- ・水平垂直周期(pluslng)
未使用です。Vsize,Hsize共に0固定として下さい。

- ・最大フレームサイズ
カメラの有効画素を示します。カメラの設定と同サイズを設定して下さい。カメラの設定値または設定方法は、カメラのマニュアルを参照して下さい。**設定が異なる場合、映像を正しく取得できませんのでご注意ください。**

(3) カメラ情報テーブル

```
typedef struct {
    char                name[64]; /* カメラ名称 */
    unsigned long        CamAtr; /* カメラ属性 */
    unsigned long        Hcycle; /* 水平周波数(ns) */
    unsigned long        OptInfo; /* オプション */
    unsigned long        Reserve; /* リザーブ */
} VIDEO_CAM_TBL, *PVIDEO_CAM_TBL;
```

- ・カメラ名称(name)
設定するカメラデータの識別名称を設定して下さい。
- ・カメラ属性(CamAtr)
カメラリンクモノクロ、RAWカメラの場合 CAMATR_CLINK (0x07)を指定して下さい。
カメラリンクRGBカメラの場合 CAMATR_CLINKRGB (0x08)を指定して下さい。
※その他は未サポートです。
- ・水平周波数(Hcycle)
カメラのLVALまたはDVAL周期をns単位で設定して下さい。(カメラのマニュアルを参照下さい。)
- ・オプション
0を指定して下さい。
- ・リザーブ
0を指定して下さい。

(4) カメラリンク設定テーブル

```

typedef struct {
    unsigned char    camtype; /* 0:PoCL/CameraLink 1:PoCL-Lite */
    unsigned char    scantype; /* 0:Area Scan 1: Line Scan */
    unsigned char    tapform; /* TAPフォーマット */
    /* 0x14 : 1TAP */
    /* 0x21 : 2TAP-2X1Y-BLK */
    /* 0x22 : 2TAP-2X1Y-BLK-MR */
    /* 0x23 : 2TAP-2X1Y-PXL */
    /* 0x24 : 2TAP-1X2Y-LINE */
    /* 0x25 : 2TAP-1X2Y-BLK */
    /* 0x41 : 4TAP-4X1Y-BLK */
    /* 0x43 : 4TAP-4X1Y-PXL */
    /* 0x44 : 4TAP-1X4Y-LINE */
    /* 0x45 : 4TAP-1X4Y-BLK */
    /* 0x81 : 8TAP-8X1Y-BLK */
    /* 0x83 : 8TAP-8X1Y-PXL */
    /* 0x84 : 8TAP-1X8Y-LINE */
    /* 0x85 : 8TAP-1X8Y-BLK */
    unsigned char    pxlform; /* 0:8bpp 1:10bpp 2:12bpp 3:24bpp(RGB) */
    unsigned char    pxlconv; /* 0:no convert 1:convert 8bit */
    unsigned char    seldval; /* 0:DVAL 1:LVAL */
    unsigned char    vinmode; /* 0:1TAP/2TAP(single), 1:2TAP(dual),2:RGB,3:4TAP/8TAP */
    unsigned char    reserve[13];
    /* reserve[0] : RGBカメラフォーマット */
    /* 0x00 : PortA:R, PortB:G, PortC:B */
    /* 0x01 : PortA:R, PortB:B, PortC:G */
    /* 0x02 : PortA:G, PortB:R, PortC:B */
    /* 0x03 : PortA:G, PortB:B, PortC:R */
    /* 0x04 : PortA:B, PortB:R, PortC:G */
    /* 0x05 : PortA:B, PortB:G, PortC:R */
    /* reserve[1-12] : 0固定 reserve */
} VIDEO_CLINK_MODE_TBL, *PVIDEO_CLINK_MODE_TBL;

```

- カメラリンクタイプ (camtype)

PoCL-Lite型カメラリンクカメラの場合1を指定して下さい。
それ以外のカメラリンクカメラは0を指定して下さい。
但しPoCL-Lite(1)は未サポートです。

- スキャンタイプ (scantype)

エリアスキャンタイプの場合0を指定して下さい。
ラインスキャンの場合1を指定して下さい。**但し本設定は未サポートです。**

- TAPフォーマット (tapform)

カメラリンクカメラのTAPフォーマットを指定して下さい。カメラに合わせ設定して下さい。
後述のTAPフォーマット詳細を参照して下さい。
※24bpp(RGB)カメラの場合1TAPを指定してください。

- 画素フォーマット (pxlform)

カメラの画素フォーマットを指定します。
8bpp : 0、 10bpp : 1、 12bpp : 2、 24bpp(RGB) : 3
但し10bpp/12bppは未サポートです。

- 画素変換 (pxlconv)

画素フォーマットを10bit/12bitを指定した場合に、本設定を有効にすると8bitに正規化され映像入力されます。
8bpp変換しない : 0、 8bpp変換する : 1

※画素フォーマットを8bpp/24bpp(RGB)で指定した場合は、0を指定して下さい。
但し本設定は未サポートです。0固定として下さい。

- 水平／垂直同期信号指定 (seldval)

通常は、0(DVAL)を指定して下さい。
DVAL信号を出力しないカメラの場合1(LVAL/FVAL)を指定します。カメラのマニュアルを参照して設定して下さい。

- VINモード (vinmode)

カメラ設定に合わせ以下を指定して下さい。

1TAP、2TAP(single) : 0、 2TAP(dual) : 1、 24bpp(RGBカメラ) : 2、 4TAP/8TAP : 3

※2TAPカメラの場合にカメラ周波数が50MHz以下の場合2TAP(single)モードでの映像入力が可能です。

2TAP(dual)は、カメラ周波数が85MHz以下の場合指定可能です。各モードでの違いは『SetVideoFrame』『SelectCamera』およびユーザズマニュアルを参照して下さい。

但し2:4TAP/8TAPは未サポートです。

- RGBフォーマット(reserve[0])

RGB出力フォーマットを指定してください。出力フォーマットは、各カメラのマニュアルを参照してください。

0x00 :PortA=R, PortB=G, PortC=B

0x01 :PortA=R, PortB=B, PortC=G

0x02 :PortA=G, PortB=R, PortC=B

0x03 :PortA=G, PortB=B, PortC=R

0x04 :PortA=B, PortB=R, PortC=G

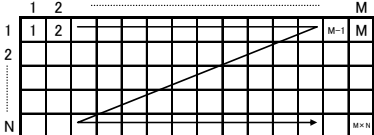
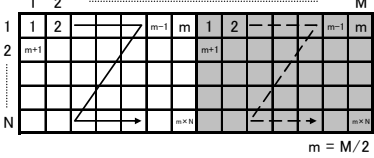
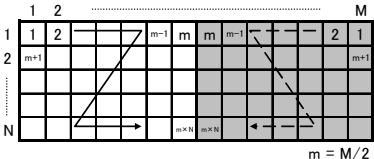
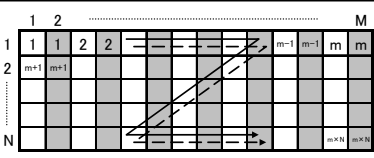
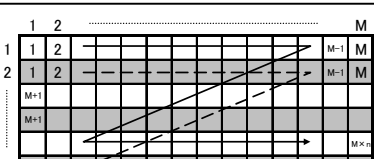
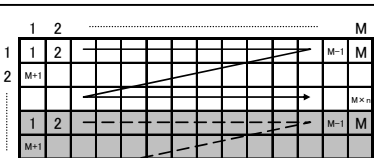
0x05 :PortA=B, PortB=G, PortC=R

ex. カメラリンクのPortAにR, PortBにB, PortCにGデータが出力されるRGBカメラの場合は、(0x00)を指定します。

- リザーブ (reserve[1~12])

0固定として下さい。

TAPフォーマット詳細

TAPフォーマット	対応定数	内容(カメラ画像データ転送フォーマット)
1TAP	0x14	 左上から右下の順に転送します。1TAPまたは、24bpp(RGB)カメラの場合本フォーマットを指定して下さい。
2TAP-2X1Y-BLK	0x21	 横方向をTAP数で分割して転送するフォーマットです。左右2ブロックに分けて転送します。転送する順序が1TAP目と2TAP目で同じです。(左上から右下)
2TAP-2X1Y-BLK-MR	0x22	 2TAP-2X1Y-BLKに対し2TAP目の転送順序を左右反転したフォーマットです。
2TAP-2X1Y-PXL	0x23	 横方向をTAP数で分割して転送するフォーマットです。奇数画素を1TAP目、偶数画素を2TAP目で転送します。
2TAP-1X2Y-LINE	0x24 ※	 縦方向をTAP数で分割して転送するフォーマットです。奇数ラインを1TAP目、偶数ラインを2TAP目で転送します。
2TAP-1X2Y-BLK	0x25 ※	 縦方向をTAP数で分割して転送するフォーマットです。上下2ブロックに分け上段を1TAP目、下段を2TAP目で転送します。
4TAP-4X1Y-BLK	0x41 ※	横方向を4ブロックに分割して転送するフォーマットです。2TAP-2X1Y-BLK(2ブロック分割)の図を参考にしてください。
4TAP-4X1Y-PXL	0x43 ※	4画素単位で転送するフォーマットです。2TAP-2X1Y-PXL(2画素単位)の図を参考にしてください。
4TAP-1X4Y-LINE	0x44 ※	4ライン単位で転送するフォーマットです。2TAP-1X2Y-LINE(2ライン単位)の図を参考にしてください。
4TAP-1X4Y-BLK	0x45 ※	縦方向を4ブロックに分割して転送するフォーマットです。2TAP-1X2Y-BLK(2ブロック分割)の図を参考にしてください。
8TAP-8X1Y-BLK	0x81 ※	横方向を8ブロックに分割して転送するフォーマットです。2TAP-2X1Y-BLK(2ブロック分割)の図を参考にしてください。
8TAP-8X1Y-PXL	0x83 ※	8画素単位で転送するフォーマットです。2TAP-2X1Y-PXL(2画素単位)の図を参考にしてください。
8TAP-1X8Y-LINE	0x84 ※	8ライン単位で転送するフォーマットです。2TAP-1X2Y-LINE(2ライン単位)の図を参考にしてください。
8TAP-1X8Y-BLK	0x85	縦方向を8ブロックに分割して転送するフォーマットです。2TAP-1X2Y-BLK(2ブロック分割)の図を参考にしてください。

※ 4TAP/8TAPおよび2TAP-1X2Y-LINE、2TAP-1X2Y-BLKは、未サポートです。

(5) VPP設定テーブル

```
typedef struct {
    unsigned long    data[20];
        /* data[0] : tcycle    トリガ周期[uS単位]                */
        /* data[1] : trig_out_cc トリガ出力先CC                  */
        /* data[2] : di_out_cc   DIスルー信号出力先CC            */
        /* data[3] : mode_sig    cc極性 0:正極, 1:負極            */
        /* data[4] : pxl_offset 有効画素オフセット                */
        /* data[5-19] : 0固定 reserve                            */
} VIDEO_VPP_DATA_TBL, *PVIDEO_VPP_DATA_TBL;
```

•data[0]

トリガ周期(tcycle)をuS単位設定します。内部トリガモード設定時の連続取り込みおよび、DispCamera(カメラスルー表示)をした場合、本設定周期に従いカメラに対しトリガ信号を出力します。

10us～640mSの範囲で設定可能です。10us単位未満は切り捨てられます。

※トリガモードについては『SetTriggerMode』を参照して下さい。

•data[1]

トリガ出力先CC(trig_out_cc)を設定します。

カメラリンクカメラのどのCC番号(1～4)にトリガ信号を出力するか指定します。

1(デフォルト)の場合CC1に出力します。

1～4を設定した場合は、それぞれCC1～4へ出力します。

※トリガ出力をしない(トリガ無効)場合は、CCへは出力されません。

※トリガモードについては『SetTriggerMode』を参照して下さい。

•data[2]

DIスルー信号出力先CC(di_out_cc)を設定します。

カメラリンクカメラのどのCC番号(1～4)にトリガ信号を出力するか指定します。

2(デフォルト)の場合CC2に出力します。

1～4を設定した場合は、それぞれCC1～4へ出力します。

※DIスルー出力をしない場合は、CCへは出力されません。また、トリガ出力先CCと同じCCに設定しないでください。

※DIスルーについては『SetTriggerMode』を参照して下さい。

•data[3]

カメラのCCへの出力の極性(mode_sig)を設定します。

bit0～4がそれぞれCC1～4に対応します。

正極：0、負極：1

を設定して下さい。

•data[4]

有効画素オフセットを設定します。

bit0～15：Hoffset、bit16～31：Voffset

•data[5～19]

リザーブ。0固定として下さい。

2.3.13 カメラデータの取得（ GetCameraData ）

□ C言語API

```
int ret = GetCameraData( int type , CAMERA_DATA_TBL *pDataTbl );
```

□ パラメータ

int	type	カメラタイプ
CAMERA_DATA_TBL *	pDataTbl	映像入力用カメラデータテーブルを格納したアドレス

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）
CAMERA_DATA_TBL *	pDataTbl	映像入力用カメラデータテーブルを格納したアドレス

□ パケットの構造

```
typedef struct {
    VIDEO_CAM_FRM_INFO    CamFrmInfo;    カメラデータフォーマット情報
    VIDEO_FRM_TBL         Frmtbl;         映像入力画面サイズテーブル
    VIDEO_CAM_TBL         Camtbl;         カメラ情報テーブル
    VIDEO_CLINK_MODE_TBL  ClnkModeTbl;    カメラリンク設定テーブル
    VIDEO_VPP_DATA_TBL    VppDataTbl;    VPP設定テーブル
} CAMERA_DATA_TBL, *PCAMERA_DATA_TBL;
```

□ エラーコード

2 0	カメラタイプ設定値範囲外
2 1	映像入力用カメラデータテーブルがNULL

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

カメラタイプと映像入力用カメラデータの読み出しを行います。
カメラタイプtypeはSelectCameraで設定可能な値を設定してください。

本コマンドを使用する場合には、以下のファイルをインクルードしてください。

```
#include "video.h"
```

テーブルのフォーマットの詳細は、「SetCameraData」を参照して下さい。

2.3.14 カメラデータファイルのロード (LoadCameraFile)

□ C言語API

```
int ret = LoadCameraFile( int type , char *filename );
```

□ パラメータ

int	type	カメラタイプ
char	*filename	映像入力用カメラデータファイル名を格納したアドレス (カメラデータファイルは、弊社提供(拡張子cam)のテキストファイルです)

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

2 0	カメラタイプ設定値範囲外
2 1	ファイル名不正
4 8	動的メモリ確保エラー
1 0 0	カメラデータファイルオープンエラー
1 0 1	不当カメラデータファイルエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

カメラデータファイルは4つまでロードすることができます。カメラタイプ(type)は0x80～0x83を使用します。

カメラデータファイルは、弊社が提供する拡張子をcamとするテキストファイルです。拡張子がcam以外の場合はエラーになります。

カメラデータファイルのフォーマットを以下に示します。カメラデータは[]で示されるタグでデータ項目を指定し、改行してデータを記述します。ファイル中には、"//"で始まる行をコメント行として記述することができますが、データ行にコメントを記述することはできません。

<カメラファイル形式>

```
[Camera Data Version]
Version=2.0

[Camera Info]
name=SONY_XCL_C130_1TAP
attr=CAMATR_CLINK
hcycle=33200
option=0

[Video Frame]
plusing_vsize=0
plusing_hsize=0
maxfm_vsize=964
maxfm_hsize=1280

[Camera Link]
camera_type=CAMERA_LINK
scan_type=AERA_SCAN
tap_format=1TAP
pixel_format=8BPP
pixel_convert=NON
dval=DVAL
vin_mode=SINGLE
rgb_format=NON

[Trigger]
cycle=40000
out_cc=CC1
diout_cc=CC2
cc1_mode=NORMAL
cc2_mode=NORMAL
cc3_mode=NORMAL
cc4_mode=NORMAL
voffset=0
hoffset=0
```

本コマンドでは弊社が提供するカメラデータファイル以外を使用しないでください。それ以外を使用した場合には、動作の保証はできません。

2.3.15 連続映像入力モードの設定と起動 (CaptureContinuous)

□ C言語API

```
int ret = CaptureContinuous( CapImgTbl *capimg );
```

□ パラメータ

CapImgTbl *capimg キャプチャ先画面IDテーブルを格納したアドレス

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID[]	○	○	×	×	×	×	○	○	○

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

2	OS システムエラー
1 0	番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー) 、
2 0	キャプチャ先画面 ID テーブル設定値範囲外
1 3 1	V I N ドライバでのエラー
1 3 2	D U ドライバでのエラー
1 3 3	アクティブビデオポートが未設定
1 3 5	画面サイズまたは画面タイプが不正
1 3 6	システム管理データが不正
1 3 7	画面管理データが不正
1 3 8	映像入力画面未設定
—	エラーリセットコマンド (ClearIPErrors) 未発行

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

キャプチャ先画面IDテーブルのフォーマットを、下記に示します。

typedef struct {		
int	Num;	画面番号設定数 “3”を設定して下さい
int	ImgID[3];	画面番号テーブル 画像メモリ領域確保コマンドで確保済みの3面分の画面番号を、テーブルの[0]～[2]へ設定して下さい
} CapImgTbl;		

連続映像入力モードの設定と起動を行います。

連続映像入力は、あらかじめ、画像メモリ3画面分のバッファが必要です。本コマンドでは、capimgで指定された画面をVINのメモリスレジスタに設定し、連続映像入力を起動します。なお、指定する3面分の画面は、モノクロでかつ、全て同じサイズ、同じ属性の画面を指定して下さい。
画面番号テーブルに同一の画面番号を指定した場合、動作の保証はいたしません。

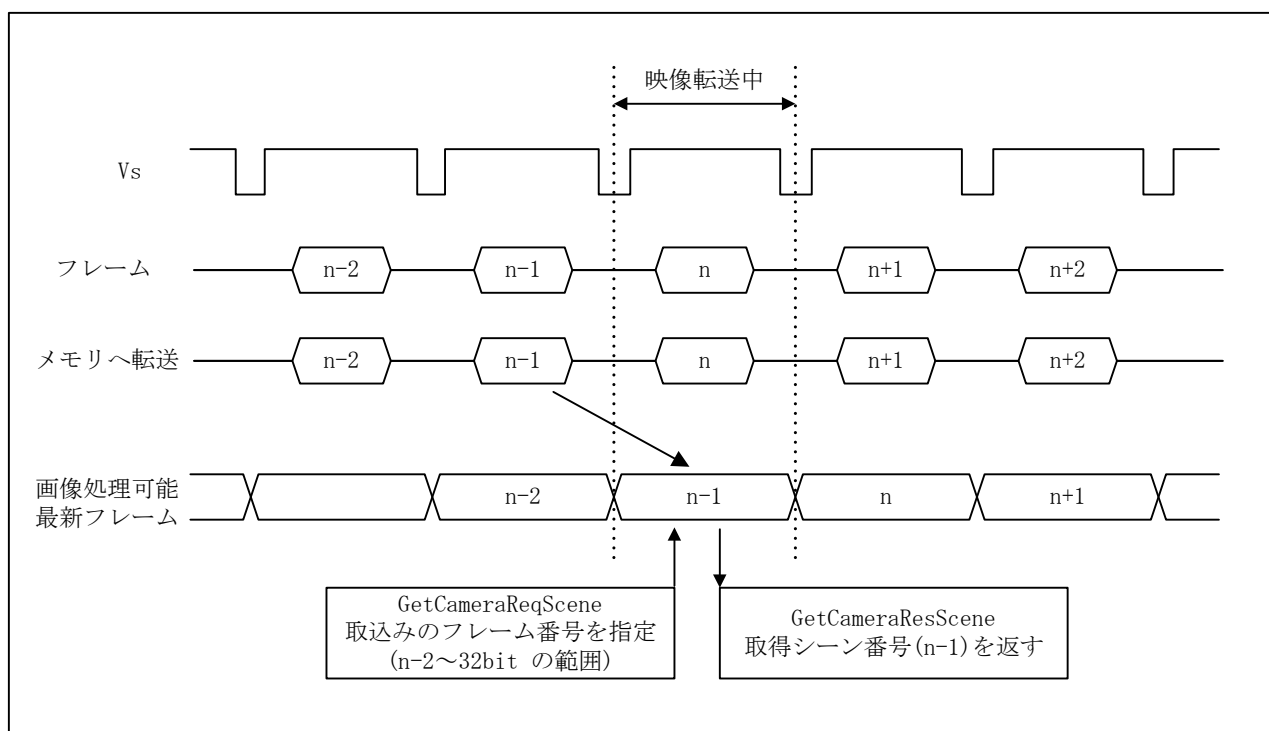
本コマンドの2重起動を行った場合は、パラメータチェックだけ行い、何もせずに正常終了します。
また、画面番号テーブルに指定した3面分の画面番号は本コマンドのワーク画面扱いとなりますので、これらの画面番号は画像処理に使用しないで下さい。

モノクロカメラの連続映像入力で入力画面にY画面以外を指定した場合はエラー出力します。
RGBカラーカメラの連続映像入力で入力画面にRGB画面以外を指定した場合はエラー出力します。

連続映像入力モードでは、常に映像のシーン番号を管理しています。そのため映像のシーン番号を指定した映像入力や映像入力した映像のシーン番号の取得ができます。連続映像入力は以下のコマンドで行います。

No.	設定項目	設定内容
1	GetCamera	最新のフレームを入力します。コマンド実行時、映像入力が終了している最新のフレームを入力します。
2	GetCameraReqScene	指定されたシーン番号のフレームを入力します。コマンド実行時、指定されたシーン番号のフレームの映像入力が終了していない場合、指定されたシーン番号のフレームの映像入力が終了するまでウェイトします。
3	GetCameraResScene	最新のフレームを入力し、そのフレームのシーン番号を返します。コマンド実行時、映像入力が終了している最新のフレームを入力します。

連続映像入力での映像フレームの画像メモリへの転送とそれに対する画像処理可能フレームのタイムチャートを下図に示します。連続映像入力では、常に3画面の画像メモリバッファで交代しながら映像入力しているため、常に映像転送中の画面の前2画面の映像を保持しています。映像フレームがシーン番号 n のタイミングで映像入力コマンド『GetCameraResScene()』を実行すると取得シーン番号は $(n-1)$ を返します。このとき取込みのフレーム番号指定の映像入力コマンド『GetCameraReqScene()』は $(n-2) \sim 32 \text{ bit}$ の範囲で有効になります。



●複数カメラ同時入力時の連続映像入力

複数カメラ同時入力時の連続入力では、映像入力画面のコンポーネント画面番号を取得することができます。以下のようにコンポーネント画面番号を取得してください。

```

CaptureContinuous( &capimg );           //連続映像入力開始

ImgID2 = GetElementImgID( ImgID1, 1 );    //コンポーネント画面番号の取得
for(;;){
    GetCamera( ImgID1 );                  // 2 カメラ同時入力
    DispImg(ImgID1);                      //カメラ 1 の映像表示
    DispImg(ImgID2);                      //カメラ 2 の映像表示
}

StopCaptureContinuous();                  //連続キャプチャ停止

```

2.3.16 連続キャプチャ停止 (StopCaptureContinuous)

□ C言語API

```
int ret = StopCaptureContinuous();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
1 3 1	V I Nドライバでのエラー
1 3 3	アクティブビデオポートが未設定
—	エラーリセットコマンド (ClearIPErrors) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

連続キャプチャを停止し、単一キャプチャモードに設定します。再度、連続映像入力を実行する場合には、『CaptureContinuous()』コマンドを実行して下さい。

2.3.17 連続入力シーン番号指定付きカメラ映像入力 (GetCameraReqScene)

□ C言語API

```
int ret = GetCameraReqScene(IMGID ImgID, unsigned long ReqSceneNo,
                           unsigned long *ResSceneNo, long tmout);
```

□ パラメータ

IMGID	ImgID	映像取り込み画面番号
unsigned long	ReqSceneNo	要求シーン番号
unsigned long	*ResSceneNo	応答シーン番号を格納するアドレス
long	tmout	映像入力タイムアウト(ms)

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	×	×	×	×	×	◇S	◇S	-
	×	○	×	×	×	×	◆	-	-

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)
unsigned long	*ResSceneNo	シーン番号を格納するアドレス

□ エラーコード

2	OSシステムエラー
8	V I N入力タイムアウト
1 0	番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 2	シーン番号を返すアドレスが不正
2 3	映像入力タイムアウト設定値範囲外 (0 未満)
1 3 1	V I Nドライバでのエラー
1 3 2	D Uドライバでのエラー
1 3 3	アクティブビデオポートが未設定
1 3 5	画面サイズが不正、またはカメラタイプと画面タイプの対応が不正
1 3 6	システム管理データが不正
1 3 7	画面管理データが不正
1 3 8	映像入力画面未設定 または、単一映像入力動作時 SetConfigCameraでCFG_VFW_CAMERAとCFG_AVI_FILEの場合、 『VFWCam2.exe』ツール未起動
-	エラーリセットコマンド (ClearIPErrors) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

要求シーン番号を指定してカメラから画像メモリに映像入力を行います。要求シーン番号に0以外を指定した場合、指定シーン番号の取り込みの終了を待ちます。すでに指定シーン番号が過ぎてしまった場合又は、要求シーン番号に0を指定した場合は、最新のシーンの映像を取り込みます。取り込みが成功すると、応答シーン番号格納領域に取り込んだシーン番号を返します。映像入力タイムアウトは、要求シーン番号に合わせて指定して下さい。映像入力タイムアウトで指定した時間内に取り込みが終了しない場合は、エラーとなります。

本コマンドの実行は『CaptureContinuous()』コマンドにより連続取り込みモードに設定しておく必要があります。このコマンドで映像を入力する際には、入力画面サイズと映像入力サイズ (ビデオフレームサイズ) を (画面サイズ ≥ 映像入力サイズ) かつ 『CaptureContinuous()』コマンドで指定した画面と同じサイズに設定して下さい。上記の条件以外の場合はエラーとなり、この処理は行われません。

2.3.18 連続入力シーン番号出力付きカメラ映像入力 (GetCameraResScene)

□ C言語API

```
int ret = GetCameraResScene( IMGID ImgID, unsigned long *ResSceneNo );
```

□ パラメータ

IMGID	ImgID	映像の取り込み画面番号
unsigned long	*ResSceneNo	応答シーン番号を格納するアドレス

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	×	×	×	×	×	◇S	◇S	-
	×	○	×	×	×	×	◆	-	-

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)
unsigned long	*ResSceneNo	シーン番号を格納するアドレス

□ エラーコード

2	OS システムエラー
8	V I N 入力タイムアウト
1 0	番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 1	シーン番号を返すアドレスが不正
1 3 1	V I N ドライバでのエラー
1 3 2	D U ドライバでのエラー
1 3 3	アクティブビデオポートが未設定
1 3 5	画面サイズが不正、または画面タイプが不正
1 3 6	システム管理データが不正
1 3 7	画面管理データが不正
1 3 8	映像入力画面未設定 または、単一映像入力動作時 SetConfigCameraでCFG_VFW_CAMERAとCFG_AVI_FILEの場合、 『VFWCam2.exe』ツール未起動
1 3 9	映像入力タイムアウト時間設定値 (システムで定義しているCFG_VINTMOの値) が 0 未満
-	エラーリセットコマンド (ClearIPErrors) 未発行

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

カメラから画像メモリに映像入力を行いその入力シーン番号を返します。常に最新のシーンの映像を取り込みます。取り込みが成功すると、応答シーン番号格納領域に取り込んだシーン番号を返します。

本コマンドの実行は『CaptureContinuous()』コマンドにより連続取り込みモードに設定しておく必要があります。このコマンドで映像を入力する際には、入力画面サイズと映像入力サイズ (ビデオフレームサイズ) を (画面サイズ ≥ 映像入力サイズ) かつ 『CaptureContinuous()』コマンドで指定した画面と同じサイズに設定して下さい。上記の条件以外の場合はエラーとなり、この処理は行われません。

2.4 映像表示

2.4.1 表示画面の構成制御設定（SetConfigDisp）

□ C言語API

```
int ret = SetConfigDisp(ConfigDispPara *para);
```

□ パラメータ

ConfigDispPara *para 表示画面の構成制御テーブルを格納したアドレス

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ パケットの構造

```
typedef struct {
    int    xsize;           X方向の表示サイズ
    int    ysize;           Y方向の表示サイズ
    int    dpx;             X方向の表示開始位置
    int    dpy;             Y方向の表示開始位置
    int    disp_plane;      映像表示プレーン番号
    int    overlay_plane;   オーバレイ表示プレーン番号
    int    overlay_pallet;  オーバレイ表示パレット番号
} ConfigDispPara;
```

□ エラーコード

2	OSシステムエラー
2 0	表示画面の構成制御テーブル設定値範囲外
1 3 1	V I Nドライバでのエラー
1 3 2	D Uドライバでのエラー
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

画像認識ライブラリでサポートするライブ表示及び画像メモリ表示に使用する表示画面の構成情報を設定します。『DispCamera()』コマンドや『DispImg()』コマンド、『DispOverlap』コマンドで映像表示中に本コマンドを発行した場合、全ての表示が中止されます。本ライブラリの映像表示コマンドを使用する場合は、適切なパラメータを設定し本コマンドを必ず発行して下さい。プレーン表示優先順位は、デフォルトで以下に設定されています。

```
DISP_PLANE_1 > DISP_PLANE_2 > DISP_PLANE_3 >
DISP_PLANE_4 > DISP_PLANE_5 > DISP_PLANE_6 >
DISP_PLANE_7 > DISP_PLANE_8
```

オーバレイ表示のプレーン番号は、映像表示のプレーン番号よりも小さい番号を指定してください。プレーン番号の設定は、ビデオポート0/1、映像表示/オーバレイ表示の4種類ありますが、これらに同じプレーン番号を指定した場合（オーバレイ表示プレーン番号の“0”は除く）はエラーとなります。

以下に表示画面の構成制御パラメータ詳細を示します。

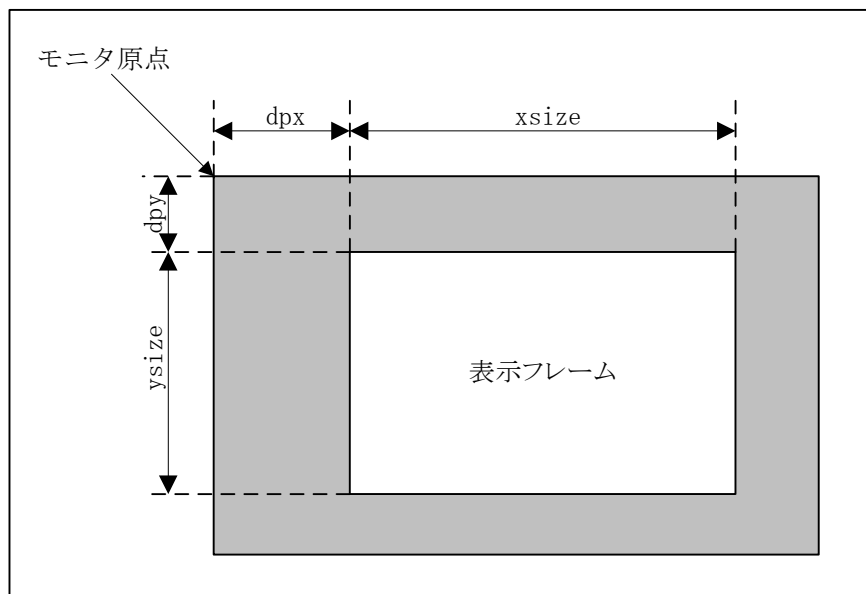
- xsize / ysize : 表示フレームサイズ

設定項目	動作	設定範囲	備考
xsize	X方向の表示サイズ	1～2046 ※	
ysize	Y方向の表示サイズ	1～1023 ※	

- dpx / dpy : 表示位置

設定項目	動作	設定範囲	備考
dpx	X方向の表示開始位置	0～2046 ※	
dpy	Y方向の表示開始位置	0～1023 ※	

※ 対象システムが実際に表示可能な範囲内で設定してください。また、X方向の表示サイズはハード制約により、偶数に設定しないと設定サイズ分表示されません。



- disp_plane : 映像表示プレーン番号

ライブ表示／画像メモリ表示に使用するプレーン番号を指定します。本パラメータはRGB表示の場合のみ有効です。

DISP_PLANE_1	(1)	プレーンNo. 1
DISP_PLANE_2	(2)	プレーンNo. 2
DISP_PLANE_3	(3)	プレーンNo. 3
DISP_PLANE_4	(4)	プレーンNo. 4
DISP_PLANE_5	(5)	プレーンNo. 5
DISP_PLANE_6	(6)	プレーンNo. 6
DISP_PLANE_7	(7)	プレーンNo. 7
DISP_PLANE_8	(8)	プレーンNo. 8

- overlay_plane : オーバレイ表示プレーン番号

オーバレイ表示に使用するプレーン番号を指定します。プレーンNo. 1～No. 8を指定して下さい。“0”を指定するとオーバレイ表示を行いません。本パラメータはRGB表示の場合のみ有効です。

- overlay_pallet : オーバレイ表示パレット番号

オーバレイ表示に使用するカラーパレット番号を指定します。使用するパレットは、『SetDispPalette()』コマンドであらかじめ設定して下さい。“0”を指定するとオーバレイ表示を行いません。本パラメータはRGB表示の場合のみ有効です。

DISP_PALLET_1	(1)	パレットNo. 1
DISP_PALLET_2	(2)	パレットNo. 2
DISP_PALLET_3	(3)	パレットNo. 3
DISP_PALLET_4	(4)	パレットNo. 4
DISP_PALLET_5	(5)	パレットNo. 5
DISP_PALLET_6	(6)	パレットNo. 6

2.4.2 表示パレットの設定 (SetDispPalette)

□ C言語API

```
int ret = SetDispPalette(int PaletteNo, DispPaletteTbl *PalTbl);
```

□ パラメータ

int	PaletteNo	パレット番号
DispPaletteTbl	*PalTbl	パレット情報を格納したアドレス

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ パケットの構造

```
typedef struct {
    unsigned char Alpha;      ブレンド値 (無効)
    unsigned char Red;        赤 (下位2bitは無効)
    unsigned char Green;      緑 (下位2bitは無効)
    unsigned char Blue;       青 (下位2bitは無効)
} DURGB_Color;

typedef struct {
    DURGB_Color Color[256];   カラー情報テーブル
} DispPaletteTbl;
```

□ エラーコード

2 0	パレット番号範囲外
2 1	パレット情報を格納したアドレスが不正
1 3 2	D Uドライバでのエラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

オーバレイ表示と擬似カラー表示の際の表示パレットを設定します。

- PaletteNo : パレット番号
設定するカラーパレット番号を指定します。

DISP_PALLET_1	(1)	パレットNo. 1
DISP_PALLET_2	(2)	パレットNo. 2
DISP_PALLET_3	(3)	パレットNo. 3
DISP_PALLET_4	(4)	パレットNo. 4

2.4.3 映像表示選択 (SelectDisp)

□ C言語API

```
int ret = SelectDisp(enum DispType type);
```

□ パラメータ

enum DispType	type	映像出力形式
		BW_DISPLAY (0) モノクロ表示
		PAL1_DISPLAY (1) 擬似カラー表示：パレット1
		PAL2_DISPLAY (2) 擬似カラー表示：パレット2
		PAL3_DISPLAY (3) 擬似カラー表示：パレット3
		PAL4_DISPLAY (4) 擬似カラー表示：パレット4

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
2 0	映像出力形式値範囲外
1 3 2	DUドライバでのエラー
1 3 8	表示画面の構成制御設定コマンド (SetConfigDisp) 未発行
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

『DispImg()』コマンド及び『DispCamera()』コマンドでの映像表示の形式を変更します。非表示状態で本コマンドを発行すると、次の『DispImg()』コマンド及び『DispCamera()』コマンドで指定した表示形式で表示されます。また、『DispImg()』コマンド及び『DispCamera()』コマンドで表示中に発行すると、即時に指定した表示形式で表示されます。変更された表示形式は、再度『DispImg()』コマンド及び『DispCamera()』コマンドを発行した後も、本コマンドの発行なしに継続されます。

BW_DISPLAY	カラー映像をモノクロ表示します
PAL1_DISPLAY	モノクロカメラ映像及びY画面をパレット1で擬似カラー表示します
PAL2_DISPLAY	モノクロカメラ映像及びY画面をパレット2で擬似カラー表示します
PAL3_DISPLAY	モノクロカメラ映像及びY画面をパレット3で擬似カラー表示します
PAL4_DISPLAY	モノクロカメラ映像及びY画面をパレット4で擬似カラー表示します

映像出力形式	DispImg	DispCamera
BW_DISPLAY	※	※
PAL1_DISPLAY	○	○
PAL2_DISPLAY	○	○
PAL3_DISPLAY	○	○
PAL4_DISPLAY	○	○

○：指定可能

※：モノクロで表示

本コマンド発行前に『SetConfigDisp()』コマンドを行って下さい。なお、擬似カラー表示は、RGB表示の場合のみ有効です。

2.4.4 カメラ映像表示 (DispCamera)

□ C言語API

```
int ret = DispCamera();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OS システムエラー
4 6	画面 ID 空きなし
4 8	動的メモリ確保エラー
1 3 1	V I N ドライバでのエラー
1 3 2	D U ドライバでのエラー
1 3 3	同時入力の設定されている (SelectCamera)
1 3 6	システム管理データが不正
1 3 8	連続映像入力動作時 (CaptureContinuous)、表示画面の構成制御設定 コマンド (SetConfigDisp) 未発行、または、映像入力画面設定コマンド (SetVideoFrame) 未発行
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

『SetConfigDisp()』コマンドで指定した映像表示プレーン:disp_planeに表示開始位置:dpx/dpyから表示サイズ:xsize/ysizeでカメラ映像をライブ表示します。

『CaptureContinuous()』コマンドで連続映像入力動作時に本コマンドを実行することはできません。『StopCaptureContinuous()』コマンドで連続映像入力を停止してから本コマンドを実行してください。また、本コマンドの2重起動を行った場合、パラメータチェックだけ行い、何もせずに正常終了します。

本コマンドを使用する前に、『ActiveVideoPort()』コマンド、『SelectCamera()』コマンド、『SetVideoFrame()』コマンドを実行し、あらかじめカメラ映像の入力準備を行って下さい。

『DispCamera()』コマンドのカメラ映像表示をビデオポートで切り替えるには『DisableDisp()』コマンドで一旦カメラ映像表示を停止してから『ActiveVideoPort()』コマンドでビデオポートを切り替え、再度『DispCamera()』コマンドを実行してください。

また、『SelectDisp()』コマンドを使用することでモノクロ映像の擬似カラー表示が可能です。

RGBカメラでは、映像表示はできません。また、擬似カラー表示もできません。

本コマンドは内部でワーク画面を3枚使用します。画像メモリの空がなく、ワーク画面が確保できない場合は、エラー (エラーコード4 8) となります。この場合は、『FreeImg()』コマンドで不要な画面を解放して、空き画面を確保して下さい。

2.4.5 カメラ映像表示終了（DisableDisp）

□ C言語API

```
int ret = DisableDisp();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
1 3 1	V I Nドライバでのエラー
1 3 2	D Uドライバでのエラー
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

カメラ映像のライブ表示を終了します。

2.4.6 画像メモリ表示 (DispImg)

□ C言語API

```
int ret = DispImg(IMGID ImgID);
```

□ パラメータ

IMGID ImgID 画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	○	○	○	○	○	○	○	○

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

2	OSシステムエラー
1 0	番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)、
1 1 3	SetConfigViewでCFG_DISP_VIEWの場合、 『IPView2. exe』 ツール通信エラー
1 3 1	V I Nドライバでのエラー
1 3 2	D Uドライバでのエラー
1 3 6	システム管理データが不正
1 3 8	表示画面の構成制御設定コマンド (SetConfigDisp) 未発行 SetConfigViewでCFG_DISP_VIEWの場合、 『IPView2. exe』 ツール未起動
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

『SetConfigDisp()』コマンドで指定した映像表示プレーン:disp_planeに表示開始位置:dpx/dpy、表示サイズ:xsize/ysizeのパラメータで指定された画面を表示します。

指定画面がモノクロ画面およびRGB画面の場合、モノクロ映像が表示されます。
RGB16画面を表示した場合、カラー映像が表示されます。RGB画面をカラー表示する場合にはCombineRGB() コマンドでRGB16画面に変換してから表示してしてください。

2.4.7 表示中止 (NoDisp)

□ C言語API

```
int ret = NoDisp();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OS システムエラー
1 3 1	V I Nドライバでのエラー
1 3 2	D Uドライバでのエラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

表示を中止します。画像メモリ表示とライブカメラ表示、画像メモリオーバーレイ表示の全てが中止され、表示画面には何も表示されない状態になります。このコマンドで表示中止状態に設定すると、表示処理のオーバーヘッドが削減され、画像処理が高速になる場合があります。処理の高速化を求める場合は、本コマンドを実行して表示中止状態にして下さい。

2.4.8 画像メモリのオーバーレイ表示（DispOverlap）

□ C言語API

```
int ret = DispOverlap(IMGID ImgID, enum OverlapMode mode);
```

□ パラメータ

IMGID	ImgID	画面番号
enum OverlapMode	mode	重ね合せ表示属性（本ライブラリでは無視）
	OVERLAP_MAX（0）	カメラ映像と画像メモリデータを比較し、大きいデータを出力（無効）
	OVERLAP_ADD（4）	カメラ映像と画像メモリデータを加算して出力（無効）

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	×	×	×	×	○	○	○	○

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
1 0	番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
2 1	重ね合せ表示属性設定値範囲外
1 3 2	D Uドライバでのエラー
1 3 5	画面タイプが不正
1 3 8	表示画面の構成制御設定コマンド（SetConfigDisp）未発行、またはオーバーレイプレーン番号、オーバーレイパレット番号に0を指定
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

パラメータで指定された画面と『DispCamera()』コマンド及び『DispImg()』コマンドで表示する映像を合成して表示します。本ライブラリでは、重ね合せ表示属性『OVERLAP_MAX』、『OVERLAP_ADD』は無視され、透過色（0）による透過色処理を行いカラーパレットで指定した色でオーバーレイ表示します。

本コマンドを使用する場合、表示画面制御設定『SetConfigDisp()』コマンドでオーバーレイプレーン番号、及びオーバーレイパレット番号を指定して下さい。

2.4.9 画像メモリのオーバーレイ表示終了（DisableOverlap）

□ C言語API

```
int ret = DisableOverlap();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
1 3 2	DUドライバでのエラー
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

オーバーレイ表示を終了します。

2.4.10 コンポーネントRGB画面から16bitRGB画面の合成 (CombineRGB)

□ C言語API

```
int ret = CombineRGB( IMGID ImgSrc, IMGID ImgDst );
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号 (コンポーネントRGB画面)
IMGID	ImgDst	デスティネーション画面番号 (16bitRGB画面)

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	×	○	×	×	×	×	○	○	○
ImgDst	×	×	×	×	×	○	◆	—	—

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理実行タイムアウト
8	VIN入力タイムアウト
10	ImgSrcの不当画面番号エラー
13	ImgDstの不当画面番号エラー
15	カラー画面空き領域、または、範囲外
49	不定画面エラー
51	ワーク画面確保エラー
138	表示画面の構成制御設定コマンド (SetConfigDisp) 未発行
—	エラーリセットコマンド (ClearIPErrors) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

コンポーネントRGBのソース画面を16bitRGBに合成し、デスティネーション画面に出力します。ソース画面には、AllocRGBImg() コマンドで確保したコンポーネントRGB画面、デスティネーション画面にはAllocRGB16Img() コマンドで確保したRGB16画面を指定してください。処理領域は、SetConfigDisp() で設定した表示サイズになります。

2.4.11 コンポーネントRGB画面から16bitRGB画面の合成(拡張) (CombineRGBEx)

□ C言語API

```
int ret = CombineRGBEx( IMGID ImgSrc, IMGID ImgDst , int sx , int sy , int dsx , int dsy ,
                        int xlng , int ylng );
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号 (コンポーネントRGB画面)
IMGID	ImgDst	デスティネーション画面番号(16bitRGB画面)
int	sx	ソース画面転送開始点X座標
int	sy	ソース画面転送開始点Y座標
int	dsx	デスティネーション画面書込み開始点X座標
int	dsy	デスティネーション画面書込み開始点Y座標
int	xlng	デスティネーション画面X方向転送幅
int	ylng	デスティネーション画面Y方向転送幅

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	×	○	×	×	×	×	○	○	○
ImgDst	×	×	×	×	×	○	◆	—	—

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理実行タイムアウト
8	VIN入力タイムアウト
10	ImgSrcの不当画面番号エラー
13	ImgDstの不当画面番号エラー
15	カラー画面空き領域、または、範囲外
22	第3引数の設定エラー
23	第4引数の設定エラー
24	第5引数の設定エラー
25	第6引数の設定エラー
26	第7引数の設定エラー
27	第8引数の設定エラー
49	不定画面エラー
51	ワーク画面確保エラー
—	エラーリセットコマンド (ClearIPErr) 未発行

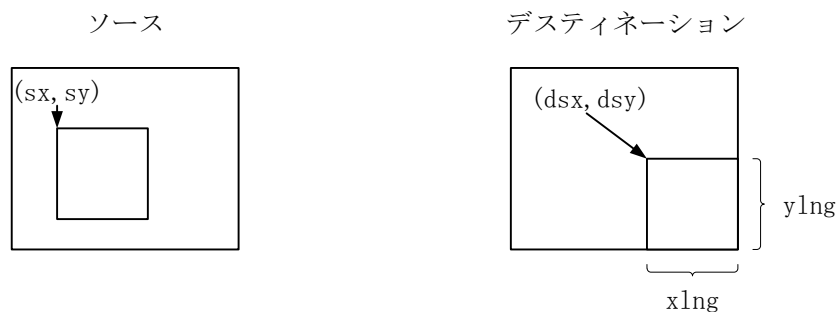
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

コンポーネントRGBのソース画面を16bitRGBに合成し、デスティネーション画面に出力します。

処理領域は、ソース画面の(sx 、 sy)を開始座標とし、デスティネーション画面の(dsx 、 dsy)を開始点とするX方向 $xlng$ 、Y方向 $ylng$ の領域に書き込みます。

ソース画面には、AllocRGBImg() コマンドで確保したコンポーネントRGB画面、デスティネーション画面にはAllocRGB16Img() コマンドで確保したRGB16画面を指定してください。



2.5 画像クリア

2.5.1 画像メモリクリア（全画面）（IP_ClearAllImg）

□ C言語API

```
int ret = IP_ClearAllImg();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
-	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

コマンド発行時に有効な画面に割り当てられている全画像メモリを0クリアします。処理範囲は画面全領域で、ウィンドウに依存しません。

2.5.2 画像メモリクリア（指定画面）（IP_ClearImg）

□ C言語API

```
int ret = IP_ClearImg(IMGID ImgDst);
```

□ パラメータ

IMGID	ImgDst	デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgDst	○	○	—	○	○	○	◇S	◇S	—

□ リターンパラメータ

```
int      ret      正常終了 (0) または異常終了 (-1)
```

❏ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

機能

パラメータで指定された画面の画像メモリを0クリアします。処理範囲は画面全領域で、ウィンドウに依存しません。

2.5.3 定数発生 (IP_Const)

□ C言語API

```
int ret = IP_Const(IMGID ImgDst, int constant);
```

□ パラメータ

IMGID	ImgDst	デスティネーション画面番号
int	constant	定数

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgDst	○	△R	○	○	○	○	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 1	定数値設定範囲外
3 3	デスティネーション画面ウィンドウ設定エラー
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

パラメータで指定された画面に定数を書き込みます。処理範囲はデスティネーション画面ウィンドウ (DST_WIN) となります。

constの範囲はシステムデータタイプが符号付8ビットの場合は-128～127、符号なし8ビットの場合は0～255です。

パラメータで指定された画面が16bitRGB画面の場合、constの範囲はシステムデータタイプが符号付8ビットの場合は-32768～32767、符号なし8ビットの場合は0～65535です。またRGB16画面の場合は処理範囲のX方向幅が8より小さい場合はエラーになります。

2.6 転送／アフィン変換

2.6.1 転送（IP_Copy）

□ C言語API

```
int ret = IP_Copy(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	○	—	○	○	○	○	○	○
ImgDst	○	○	—	○	○	○	◇1	◇1	◇1

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

画面をコピーします。

ソースとデスティネーションの組合せ

Y→Y	Y画面コピー
Y→RGB	Y画面をRGB. R画面にコピー
Y→RGB16	エラー(13)
RGB→Y	RGB. R画面をY画面へコピー
RGB→RGB	RGB画面へコピー
RGB→RGB16	エラー(13)
RGB16→Y	エラー(13)
RGB16→RGB	エラー(13)
RGB16→RGB16	RGB16画面へコピー

2.6.2 拡大／縮小 (IP_Zoom)

□ C言語API

```
int ret = IP_Zoom(IMGID ImgSrc, IMGID ImgDst, float mag);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
float	mag	変換倍率 (0.04 ~ 128)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	○	—	○	○	○	○	○	○
ImgDst	○	○	—	○	○	○	◇1	◇1	◇1

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
22	変換倍率範囲外
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

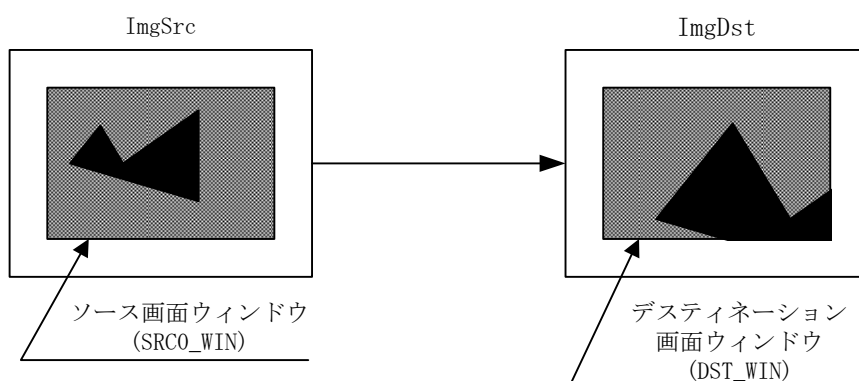
パラメータで指定の変換倍率に従い、拡大／縮小をします。倍率は実数です。

本機能はソフトウェアにより処理しています。実際の処理は、最初に処理変換倍率の逆数を65536倍後小数点以下を切り捨て「1画素間の移動量×65536」を算出します。この値を使用して算出した「転送元座標×65536」を、16ビットシフトダウンにより65536で除算した整数の座標値にし、ソース画面データをデスティネーション画面に書き込みます。

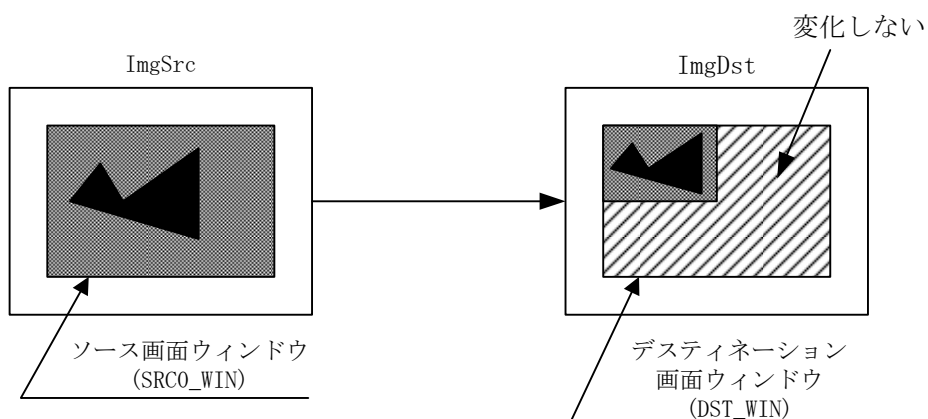
なお、このコマンドを実行したときの出力画面サイズは、デスティネーション画面サイズ、またはデスティネーション画面ウィンドウサイズ (DST_WIN) となります。拡大／縮小後の画面サイズがDST_WINサイズより大きい場合、DST_WINでクリッピングされます。拡大／縮小後の画面サイズがDST_WINサイズより小さい場合、転送対象以外の部分は変化しません。

変換倍率が128に近い拡大率で、幅又は高さ512画素を超える領域に転送する場合には、拡大率を調整しても期待通りの幅又は高さにならない場合があります。

[実行例 1] IP_Zoom(ImgSrc, ImgDst, 2.0);



[実行例 2] IP_Zoom(ImgSrc, ImgDst, 0.5);



2.6.3 拡大／縮小（縦横任意倍率）（IP_ZoomExt）

□ C言語API

```
int ret = IP_ZoomExt(IMGID ImgSrc, IMGID ImgDst, float xmag, float ymag);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
float	xmag	X方向の変換倍率（0.04～128）
float	ymag	Y方向の変換倍率（0.04～128）

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	○	—	○	○	○	○	○	○
ImgDst	○	○	—	○	○	○	◇1	◇1	◇1

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
22	X方向の変換倍率範囲外
23	Y方向の変換倍率範囲外
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

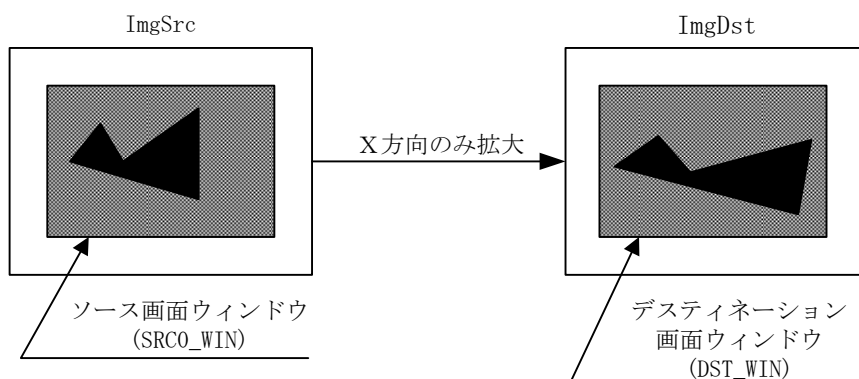
パラメータで指定のX方向とY方向の変換倍率に従い、拡大／縮小をします。倍率は実数です。

本機能はソフトウェアにより処理しています。実際の処理は、最初に処理変換倍率の逆数を65536倍後小数点以下を切り捨て「1画素間の移動量×65536」を算出します。この値を使用して算出した「転送元座標×65536」を、16ビットシフトダウンにより65536で除算した整数の座標値にし、ソース画面データをデスティネーション画面に書き込みます。

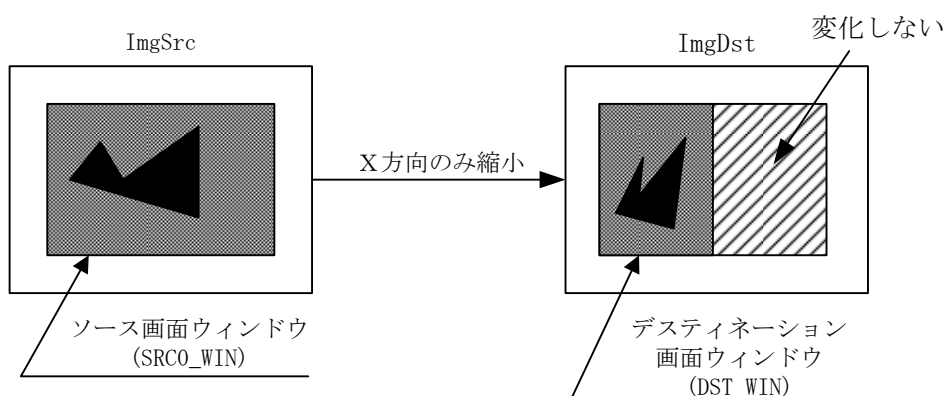
なお、このコマンドを実行したときの出力画面サイズは、デスティネーション画面サイズ、またはデスティネーション画面ウィンドウサイズ（DST_WIN）となります。拡大／縮小後の画面サイズがDST_WINサイズより大きい場合、DST_WINでクリッピングされます。拡大／縮小後の画面サイズがDST_WINサイズより小さい場合、転送対象以外の部分は変化しません。

変換倍率が128に近い拡大率で、幅又は高さ512画素を超える領域に転送する場合には、拡大率を調整しても期待通りの幅又は高さにならない場合があります。

[実行例 1] IP_ZoomExt (ImgSrc, ImgDst, 2.0, 1.0);



[実行例 2] IP_ZoomExt (ImgSrc, ImgDst, 0.5, 1.0);



2.6.4 縮小 (IP_ZoomOut)

□ C言語API

```
int ret = IP_ZoomOut(IMGID ImgSrc, IMGID ImgDst, int mag);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	mag	変換倍率 (1 ~ 8)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇1	◇1	◇1

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
22	変換倍率範囲外
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

パラメータで指定した変換倍率に従い、縮小します。倍率は整数です。

このコマンドはハードウェアで行い、コマンドを実行したときの出力画面サイズは、デスティネーション画面サイズ、またはデスティネーション画面ウィンドウサイズ (DST_WIN) となります。縮小後の画面サイズがDST_WINサイズより大きい場合、DST_WINでクリッピングされます。縮小後の画面サイズがDST_WINサイズより小さい場合、転送対象外の部分は変化しません。

ソースとデスティネーションの組合せ

Y→Y	Y画面へ拡大/縮小
Y→RGB	Y画面をRGB. R画面へ拡大/縮小
Y→RGB16	エラー (13)
RGB→Y	RGB. R画面をY画面へ拡大/縮小
RGB→RGB	RGB画面へ拡大/縮小
RGB→RGB16	エラー (13)
RGB16→Y	エラー (10)
RGB16→RGB	エラー (10)
RGB16→RGB16	エラー (10)

2.6.5 縮小（縦横任意倍率）（IP_ZoomOutExt）

□ C言語API

```
int ret = IP_ZoomOutExt(IMGID ImgSrc, IMGID ImgDst, int xmag, int ymag);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	xmag	X方向の変換倍率（1～8）
int	ymag	Y方向の変換倍率（1～8）

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇1	◇1	◇1

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
2 2	X方向の変換倍率範囲外
2 3	Y方向の変換倍率範囲外
3 0	ソース画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 9	ソース画面が不定画面
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

パラメータで指定した変換倍率に従い、縮小します。倍率は整数です。

このコマンドはハードウェアで行い、コマンドを実行したときの出力画面サイズは、デスティネーション画面サイズ、またはデスティネーション画面ウィンドウサイズ（DST_WIN）となります。縮小後の画面サイズがDST_WINサイズより大きい場合、DST_WINでクリッピングされます。縮小後の画面サイズがDST_WINサイズより小さい場合、転送対象外の部分は変化しません。

ソースとデスティネーションの組合せ

Y→Y	Y画面へ拡大/縮小
Y→RGB	Y画面をRGB. R画面へ拡大/縮小
Y→RGB16	エラー(13)
RGB→Y	RGB. R画面をY画面へ拡大/縮小
RGB→RGB	RGB画面へ拡大/縮小
RGB→RGB16	エラー(13)
RGB16→Y	エラー(10)
RGB16→RGB	エラー(10)
RGB16→RGB16	エラー(10)

2.6.6 ズームイン (IP_ZoomIn)

□ C言語API

```
int ret = IP_ZoomIn( IMGID ImgSrc , IMGID ImgDst , int mag);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	mag	変換倍率

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇1	◇1	◇1

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

1 0, 1 3	画面空き領域、または範囲外 (不当画面番号エラー)
2 2	変換倍率範囲外
4 9	アクセス画面が不定画面
5 0	画像メモリチャネル競合
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

パラメータで指定した変換倍率に従い、拡大します。

このコマンドはハードウェアで行い、ソース画面の画像をパラメータで指定した変換倍率に従い拡大し、その結果をデスティネーション画面へ転送します。

拡大の画像サイズがDST_WINサイズより大きい場合DST_WINでクリッピングされます。

拡大の画像サイズがDST_WINサイズより小さい場合転送対象外の部分は変化しません。

ソースとデスティネーションの組合せ

Y→Y	Y画面へ拡大/縮小
Y→RGB	Y画面をRGB. R画面へ拡大/縮小
Y→RGB16	エラー (13)
RGB→Y	RGB. R画面をY画面へ拡大/縮小
RGB→RGB	RGB画面へ拡大/縮小
RGB→RGB16	エラー (13)
RGB16→Y	エラー (10)
RGB16→RGB	エラー (10)
RGB16→RGB16	エラー (10)

2.6.7 ズームイン(縦横任意倍率) (IP_ZoomInExt)

□ C言語API

```
int ret = IP_ZoomInExt( IMGID ImgSrc , IMGID ImgDst , int xmag , int ymag);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	xmag	X方向への変換倍率 (1 ~ 8)
int	ymag	Y方向への変換倍率 (1 ~ 8)

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇1	◇1	◇1

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

1 0 , 1 3	画面空き領域、または範囲外 (不当画面番号エラー)
2 2	X方向の変換倍率範囲外
2 3	Y方向の変換倍率範囲外
4 9	アクセス画面が不定画面
5 0	画像メモリチャネル競合
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

パラメータで指定した変換倍率に従い、拡大します。
 なお、本コマンドを実行したときの出力画面サイズは、デスティネーション画面サイズ、またはデスティネーション・ウィンドウサイズ (DST_WIN) となります。

ソースとデスティネーションの組合せ

Y→Y	Y画面へ拡大/縮小
Y→RGB	Y画面をRGB. R画面へ拡大/縮小
Y→RGB16	エラー (13)
RGB→Y	RGB. R画面をY画面へ拡大/縮小
RGB→RGB	RGB画面へ拡大/縮小
RGB→RGB16	エラー (13)
RGB16→Y	エラー (10)
RGB16→RGB	エラー (10)
RGB16→RGB16	エラー (10)

ソース画面の画像をパラメータで指定した変換倍率に従い拡大し、その結果をデスティネーション画面へ転送します。

拡大の画像サイズがDST_WINサイズより大きい場合DST_WINでクリッピングされます。

拡大の画像サイズがDST_WINサイズより小さい場合転送対象外の部分は変化しません。

[実行例]

IP_ZoomInExt (ImgSrc , ImgDst , 2.0 , 1.0)



拡大後のサイズがDST_WINサイズより小さい場合、転送対象以外の部分は変化されません。

[実行例]

IP_Zoom (ImgSrc , ImgDst , 0.5 , 1.0)



2.6.8 拡大／縮小（シフト）（IP_ZoomS）

□ C言語API

```
int ret = IP_ZoomS(IMGID ImgSrc, IMGID ImgDst, float mag, int dx, int dy, enum SPACE_OPT opt);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
float	mag	変換倍率（0.04～128）
int	dx	X方向へのシフト量
int	dy	Y方向へのシフト量
enum SPACE_OPT	opt	周辺処理オプション
	SPACE_0CLEAR	(0) 周辺を0クリアする
	SPACE_NOTOUCH	(1) 周辺は変化させない
	SPACE_128CLEAR	(2) 周辺を128クリアする

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	○	○	○	○	○	○	○	○
ImgDst	○	○	○	○	○	○	◇1	◇1	◇1

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
22	変換倍率設定範囲外
23	X方向へのシフト量設定範囲外
24	Y方向へのシフト量設定範囲外
25	オプション設定値範囲外
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面のウィンドウ内画像をパラメータで指定の変換倍率に従い拡大／縮小を行い、デスティネーション画面のウィンドウ内にパラメータ指定分シフトさせて出力します。

本機能はソフトウェアにより処理し、『IP_Rotate()』コマンドの回転がない場合と同様の動作をします。

なお、ソース画面からデスティネーション画面への転送画素数は、条件が同じでも処理が異なるため『IP_Zoom()』、『IP_ZoomExt()』コマンドとは異なる場合があります。

シフト量設定範囲外エラーの条件は、以下の通りです。

シフト量 ≥ 0 : シフト量の絶対値 $\geq$ デスティネーション画面のウィンドウ長

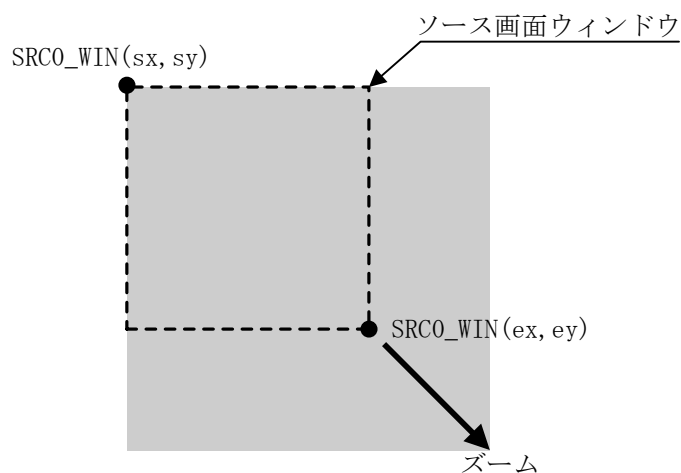
シフト量 < 0 : シフト量の絶対値 $\geq$ ソース画面のウィンドウ長 $\times$ 変換倍率

「ソース画面のウィンドウ長 $\times$ 変換倍率」は0.4を加算して切り捨てします。例えば、「ソース画面のウィンドウ長 $\times$ 変換倍率」が10.24の場合、エラーになるのはシフト量 ≤ -10 の時です。

周辺処理オプションにSPACE_128CLEARを指定した場合、システムデータタイプが符号付8ビットのとき周辺部は-128になり、符号なし8ビットのとき周辺部は128になります。

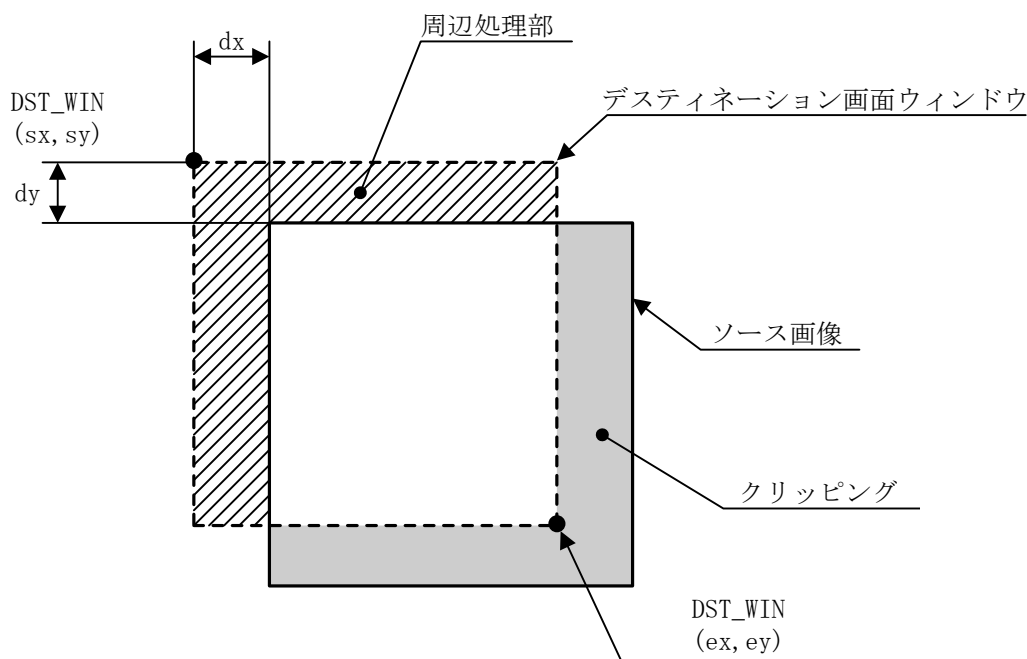
RGB16でopt=SPACE_128CLEARを選択した場合0x8410でクリアします。

以下に、本コマンドの動作手順を説明します。
ソース画面ウィンドウ (SRC0_WIN) の領域を拡大／縮小します。



デスティネーション画面ウィンドウ (DST_WIN) の始点座標 (sx, sy) から (dx, dy) 画素だけシフトした位置に拡大／縮小画像を転送します。

拡大／縮小およびシフトした画面サイズがデスティネーションウィンドウ画面サイズ (DST_WIN) より大きい場合、DST_WINでクリッピングされます。拡大／縮小およびシフトした画面サイズがデスティネーションウィンドウ画面サイズ (DST_WIN) より小さい場合、周辺部はオプション指定に従い、処理されます。



2.6.9 シフト (IP_Shift)

□ C言語API

```
int ret = IP_Shift(IMGID ImgSrc, IMGID ImgDst, int dx, int dy, enum SPACE_OPT opt);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	dx	X方向へのシフト量
int	dy	Y方向へのシフト量
enum SPACE_OPT	opt	周辺処理オプション
	SPACE_OCLEAR	(0) 周辺を0クリアする
	SPACE_NOTOUCH	(1) 周辺は変化させない
	SPACE_128CLEAR	(2) 周辺を128クリアする

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇1	◇1	◇1

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
22	X方向へのシフト量設定範囲外
23	Y方向へのシフト量設定範囲外
24	オプション設定値範囲外
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

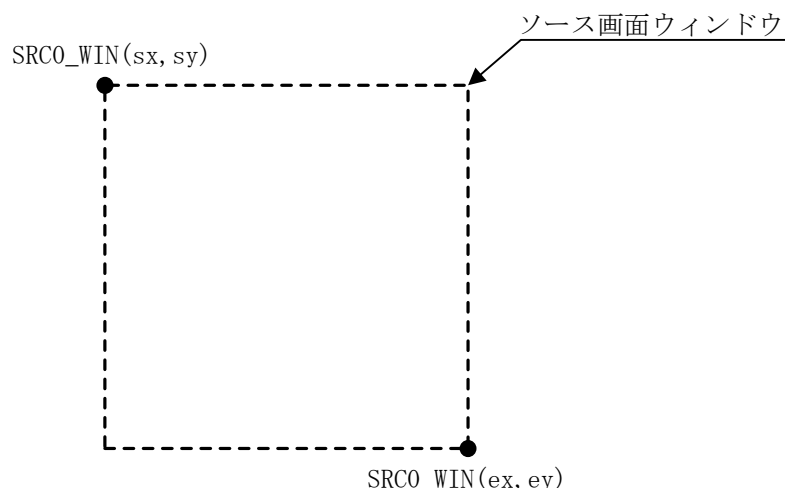
□ 機能

ソース画面のウインドウ内画像を、デスティネーション画面のウインドウ内にパラメータ指定分シフトさせて出力します。

画面サイズもしくはデスティネーションのウインドウ画面サイズ以上のシフト量を設定した場合にはエラーとなります。

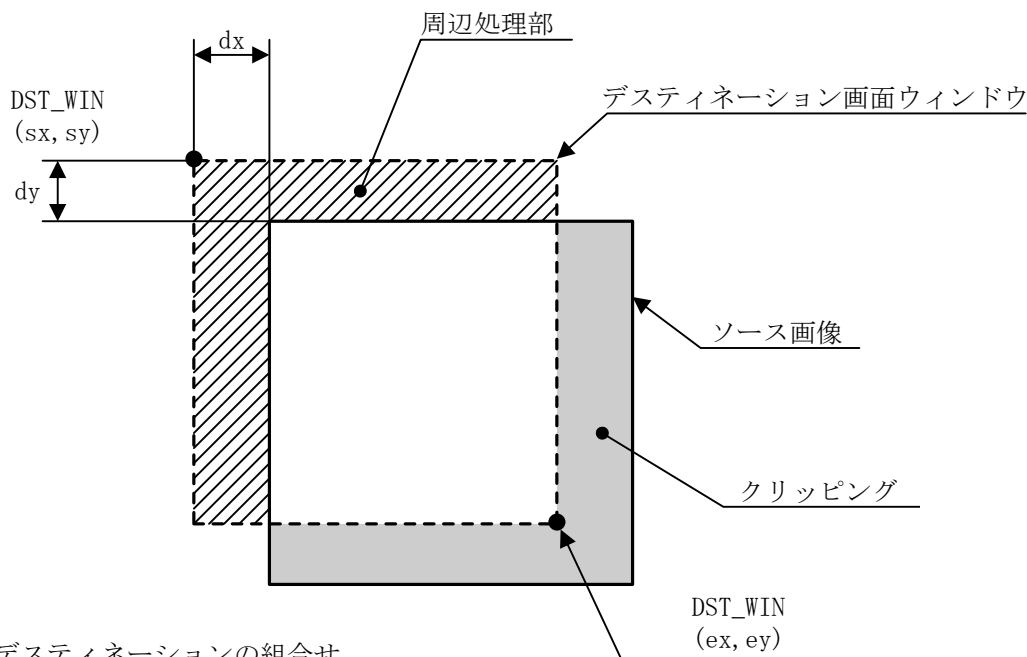
周辺処理オプションにSPACE_128CLEARを指定した場合、システムデータタイプが符号付8ビットのとき周辺部は-128になり、符号なし8ビットのとき周辺部は128になります。

以下に、本コマンドの動作手順を説明します。
 ソース画面ウィンドウ (SRC0_WIN) の領域がシフト対象です。



デスティネーション画面ウィンドウ (DST_WIN) の始点座標 (sx, sy) から (dx, dy) 画素だけ移動した位置にソース画像を転送します。

シフトした画面サイズがデスティネーションウィンドウ画面サイズ (DST_WIN) より大きい場合、DST_WINでクリッピングされます。シフトした画面サイズがデスティネーションウィンドウ画面サイズ (DST_WIN) より小さい場合、周辺部はオプション指定に従い、処理されます。



ソースとデスティネーションの組合せ

Y→Y	Y画面へシフト
Y→RGB	Y画面をRGB. R画面へシフト
Y→RGB16	エラー(13)
RGB→Y	RGB. R画面をY画面へシフト
RGB→RGB	RGB画面へシフト
RGB→RGB16	エラー(13)
RGB16→Y	エラー(10)
RGB16→RGB	エラー(10)
RGB16→RGB16	エラー(10)

2.6.10 回転 (IP_Rotate)

□ C言語API

```
int ret = IP_Rotate(IMGID ImgSrc, IMGID ImgDst, float mag, float theta, int xc, int yc,
                    int dx, int dy, enum SPACE_OPT opt);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
float	mag	変換倍率 (0.04 ~ 128)
float	theta	回転角度 (−360° ~ 360°)
int	xc	回転中心X座標
int	yc	回転中心Y座標
int	dx	X方向へのシフト量
int	dy	Y方向へのシフト量
enum SPACE_OPT	opt	周辺処理オプション
	SPACE_OCLEAR	(0) 周辺を0クリアする
	SPACE_NOTOUCH	(1) 周辺は変化させない
	SPACE_128CLEAR	(2) 周辺を128クリアする

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	○	○	○	○	○	○	○	○
ImgDst	○	○	○	○	○	○	◇1	◇1	◇1

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (−1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
22	変換倍率設定範囲外
24	回転中心X座標設定範囲外
25	回転中心Y座標設定範囲外
26	X方向へのシフト量設定範囲外
27	Y方向へのシフト量設定範囲外
28	オプション設定値範囲外
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面のウィンドウ内画像を回転中心座標に対してパラメータで指定の変換倍率に従い拡大／縮小後、右回りに回転させ、デスティネーション画面のウィンドウ内にパラメータ指定分シフトさせて出力します。

本機能はソフトウェアにより処理します。実際にはデスティネーション画面ウィンドウ範囲内の座標についてソース画面の座標を算出し、算出座標がソース画面ウィンドウ範囲内であればソース画面データをデスティネーション画面に書き込みます。ソース画面座標の小数点以下は整数型への型変換による切り捨て処理を行います。

なお、ソース画面からデスティネーション画面への転送画素数は、条件が同じでも処理が異なるため『IP_Zoom()』、『IP_ZoomExt()』コマンドとは異なる場合があります。

シフト量設定範囲外エラーの条件は、以下の通りです。回転角度には影響されません。

シフト量 ≥ 0 : シフト量の絶対値 $\geq$ デスティネーション画面のウィンドウ長

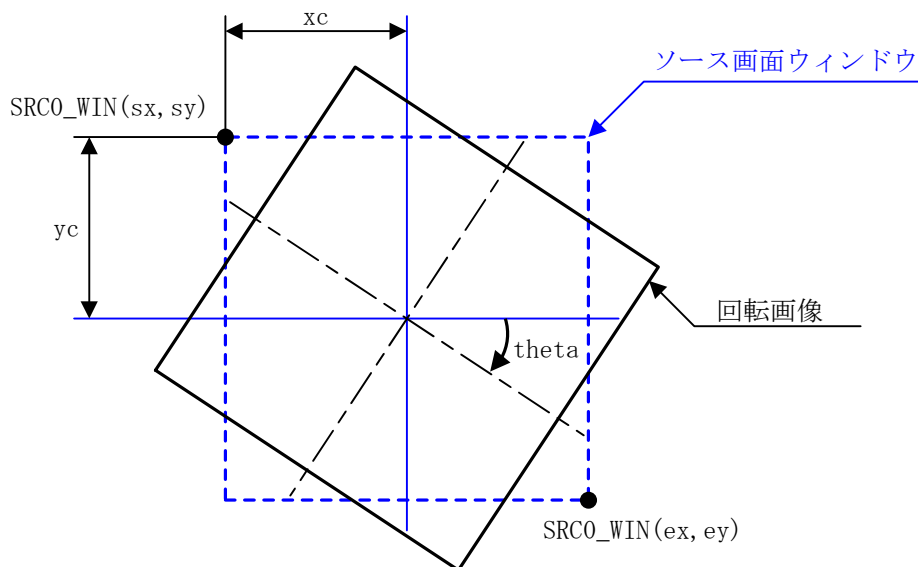
シフト量 < 0 : シフト量の絶対値 $\geq$ ソース画面のウィンドウ長 $\times$ 変換倍率

「ソース画面のウィンドウ長 $\times$ 変換倍率」は0.4を加算して切り捨てします。例えば、「ソース画面のウィンドウ長 $\times$ 変換倍率」が10.24の場合、エラーになるのはシフト量 ≤ -10 の時です。

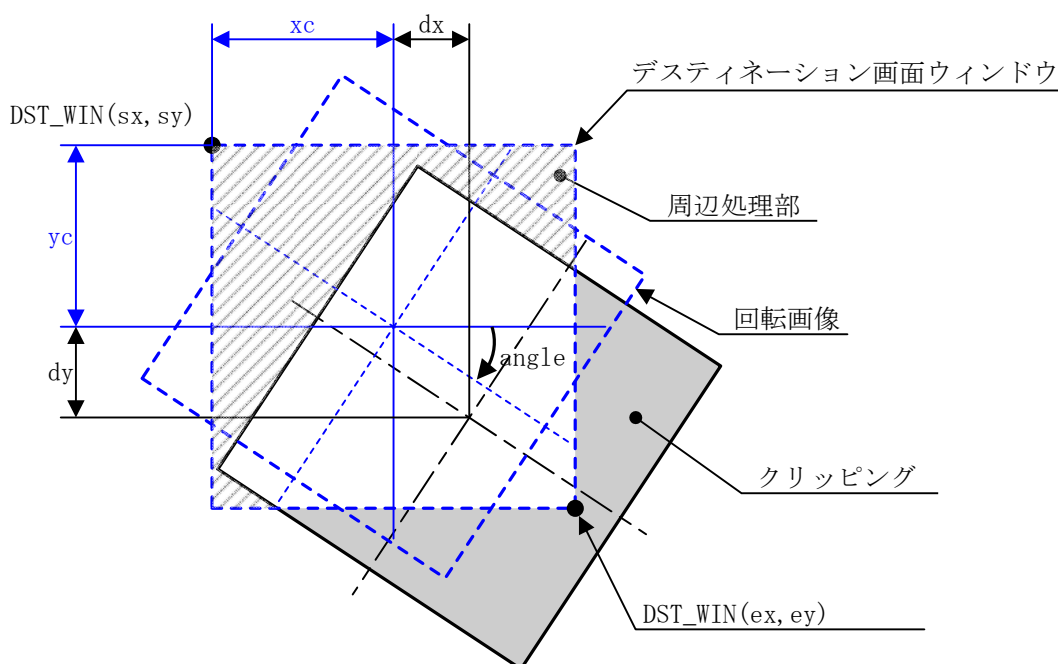
周辺処理オプションにSPACE_128CLEARを指定した場合、ソース画像が符号付8ビットのとき周辺部は-128になり、ソース画像が符号なし8ビットおよび2値のとき周辺部は128になります。

以下に、本コマンドの動作手順を説明します。

ソース画面ウィンドウ (SRC0_WIN) で指定された範囲を(sx , sy)からの相対位置(xc , yc)を中心として拡大／縮小後、 $theta$ 角度回転させます。



そして、DST_WINで指定された(sx , sy)から(dx , dy)画素だけシフトした位置に回転したソース画像を転送します。回転画像がDST_WINの範囲を超えた部分はクリッピングされ、周辺部はオプション指定に従い、処理されます。



2.7 2値化

2.7.1 2値化（IP_Binarize）

□ C言語API

```
int ret = IP_Binarize(IMGID ImgSrc, IMGID ImgDst, int thr);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	thr	しきい値

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	△S
ImgDst	○	△R	○	○	○	×	-	-	◆

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
22	しきい値設定範囲外
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド（ClearIPError）未発行

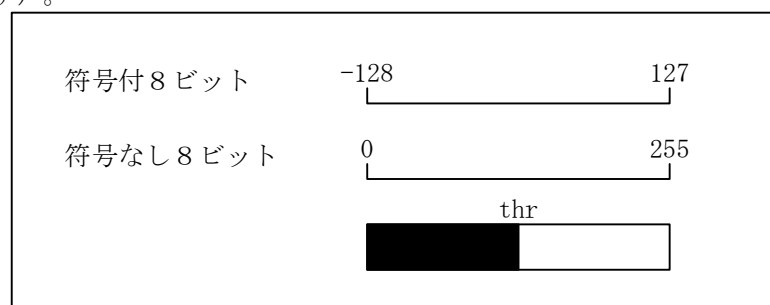
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

濃淡画像を、パラメータ指定のしきい値で2値化します。パラメータ thr で指定するしきい値以上の値を白、それより小さい値を黒で2値化します。

設定可能なしきい値はソース画面データタイプにより異なります。ソース画面データタイプが符号なし8ビットの場合のしきい値は0～255、それ以外は-128～127になります。

ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します（255は-1として処理します）。



2.7.2 範囲・反転付き2値化 (IP_BinarizeExt)

□ C言語API

```
int ret = IP_BinarizeExt(IMGID ImgSrc, IMGID ImgDst, int thrmin, int thrmax, int opt);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	thrmin	しきい値最小値
int	thrmax	しきい値最大値
int	opt	オプション設定 0 : 2 値化、1 : 反転2 値化

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	△S
ImgDst	○	△R	○	○	○	×	-	-	◆

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 2	しきい値最小値範囲設定エラー
2 3	しきい値最大値範囲設定エラー
2 4	オプション設定値範囲外
3 0	ソース画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 9	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

濃淡画像を、パラメータ指定のしきい値及びオプション設定の内容で2値化します。

設定可能なしきい値はソース画面データタイプにより異なります。ソース画面データタイプが符号なし8ビットの場合のしきい値は0～255、それ以外は-128～127になります。

ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します (255は-1として処理します)。

オプション	処理結果	しきい値の色
	符号付8ビット -128 127  符号なし8ビット 0 255 	
0	thrmin thrmax 	白
1	thrmin thrmax 	黒

2.8 画素変換

2.8.1 論理反転（IP_Invert）

□ C言語API

```
int ret = IP_Invert(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	◇1

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面の論理反転を行い、結果をデスティネーション画面に出力します。

演算式	$F = \text{NOT} (f)$

F：デスティネーション画面
f：ソース画面

2.8.2 符号反転 (IP_Minus)

□ C言語API

```
int ret = IP_Minus(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
3 0	ソース画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 9	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面の符号反転を行い、結果をデスティネーション画面に出力します。

ソース画面データタイプとシステムデータタイプが符号付8ビットの場合、-128はオーバーフローとなり127になります。

ソース画面データタイプとシステムデータタイプが符号なし8ビットの場合、全ての値がアンダーフローとなり0、ソース画面データタイプが符号なし8ビットでシステムデータタイプが符号付8ビットの場合、128～255はアンダーフローとなり-128になります。

ソース画面データタイプが2値の場合は、画素データの最上位ビット (1または0) を演算に使用します。

演算式	$F = 0 - f$

F: デスティネーション画面

f: ソース画面

2.8.3 絶対値 (IP_Abs)

□ C言語API

```
int ret = IP_Abs(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
3 0	ソース画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 9	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面の絶対値化を行い、結果をデスティネーション画面に出力します。

ソース画面データタイプとシステムデータタイプが符号付8ビットの場合、-128はオーバーフローとなり127になります。また、ソース画面データタイプが符号なし8ビットでシステムデータタイプが符号付8ビットの場合、128～255はオーバーフローとなり127になります。

ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します (255は-1として処理します)。

演算式	$F = f $

F: デスティネーション画面

f: ソース画面

2.8.4 定数加算 (IP_AddConst)

□ C言語API

```
int ret = IP_AddConst(IMGID ImgSrc, IMGID ImgDst, int constant);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

int constant 定数 (- 2 5 6 ~ 2 5 5)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

2 OSシステムエラー

3 IMP処理タイムアウト

8 VIN入力タイムアウト

1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

2 2 定数設定値範囲外

3 0 ソース画面ウィンドウ設定エラー

3 3 デスティネーション画面ウィンドウ設定エラー

4 9 ソース画面が不定画面

- エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面に定数を加算し、結果をデスティネーション画面に出力します。
結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付8ビットの場合は1 2 7
／ - 1 2 8、符号なし8ビットの場合は2 5 5 / 0を出力します。
ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します (2 5 5は-1と
して処理します)。

演算式	$F = f + \text{constant}$

F : デスティネーション画面
f : ソース画面

2.8.5 定数減算 (IP_SubConst)

□ C言語API

```
int ret = IP_SubConst(IMGID ImgSrc, IMGID ImgDst, int constant);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

int constant 定数 (−256〜255)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	−

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (−1)

□ エラーコード

2 OSシステムエラー

3 IMP処理タイムアウト

8 VIN入力タイムアウト

1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

2 2 定数設定値範囲外

3 0 ソース画面ウィンドウ設定エラー

3 3 デスティネーション画面ウィンドウ設定エラー

4 9 ソース画面が不定画面

− エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面から定数を減算し、結果をデスティネーション画面に出力します。
結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付8ビットの場合は127／−128、符号なし8ビットの場合は255／0を出力します。
ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します (255は−1として処理します)。

演算式	$F = f - \text{constant}$

F : デスティネーション画面
f : ソース画面

2.8.6 定数減算絶対値 (IP_SubConstAbs)

□ C言語API

```
int ret = IP_SubConstAbs(IMGID ImgSrc, IMGID ImgDst, int constant);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

int constant 定数 (- 2 5 6 ~ 2 5 5)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

2 OS システムエラー

3 IMP 処理タイムアウト

8 VIN 入力タイムアウト

1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

2 2 定数設定値範囲外

3 0 ソース画面ウィンドウ設定エラー

3 3 デスティネーション画面ウィンドウ設定エラー

4 9 ソース画面が不定画面

- エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面から定数を減算し、絶対値をデスティネーション画面に出力します。
結果がオーバーフローしたとき、システムデータタイプが符号付8ビットの場合は127、符号なし8ビットの場合は255を出力します。
ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します (255は-1として処理します) 。

演算式	$F = f - \text{constant} $

F : デスティネーション画面
f : ソース画面

2.8.7 定数乗算 (IP_MultConst)

□ C言語API

```
int ret = IP_MultConst(IMGID ImgSrc, IMGID ImgDst, int constant, int scale);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	constant	定数 (−256〜255)
int	scale	ダウンシフト量 (0〜15)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	−

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (−1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
22	定数設定値範囲外
23	ダウンシフト量設定値範囲外
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
−	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面と定数を乗算し、結果をダウンシフトしてデスティネーション画面に出力します。

ソース画面データタイプが符号付8ビットの場合は、ダウンシフトは算術シフトになります。

ソース画面データタイプが2値の場合は、画素データの最上位ビット（1または0）を演算に使用します。

また、ダウンシフトは四捨五入を行い、ソース画面データタイプが符号付8ビットの場合-0.5は0になります。ダウンシフトした結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付8ビットの場合は127／-128、符号なし8ビットの場合は255／0を出力します。

演算式	$F = f \times \text{constant}$

F：デスティネーション画面

f：ソース画面

ダウンシフト量	設定内容
0	演算結果ダウンシフトなし
1	演算結果1／2
2	演算結果1／4
3	演算結果1／8
⋮	⋮
15	演算結果1／32768

2.8.8 定数比較 (Min) (IP_MinConst)

□ C言語API

```
int ret = IP_MinConst(IMGID ImgSrc, IMGID ImgDst, int constant);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

int constant 定数 (- 2 5 6 ~ 2 5 5)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

2 OS システムエラー

3 IMP 処理タイムアウト

8 VIN 入力タイムアウト

1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

2 2 定数設定値範囲外

3 0 ソース画面ウィンドウ設定エラー

3 3 デスティネーション画面ウィンドウ設定エラー

4 9 ソース画面が不定画面

- エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面の個々の画素値と定数を比較し、値の小さい方をデスティネーション画面に出力します。

結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付8ビットの場合は1 2 7 / - 1 2 8、符号なし8ビットの場合は2 5 5 / 0 を出力します。

ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します (2 5 5 は - 1 として処理します) 。

定数には - 2 5 6 ~ 2 5 5 の値が設定できますが、定数値が - 2 5 6 ~ - 1 2 9 の範囲では、システムデータタイプが符号付8ビットの場合は、結果は全て - 1 2 8 になり、符号なし8ビットの場合は、結果は全て0になります。

演算式	$F = \text{MIN} (f , \text{constant})$

F : デスティネーション画面
f : ソース画面

2.8.9 定数比較 (Max) (IP_MaxConst)

□ C言語API

```
int ret = IP_MaxConst(IMGID ImgSrc, IMGID ImgDst, int constant);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

int constant 定数 (- 2 5 6 ~ 2 5 5)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

2 OSシステムエラー

3 IMP処理タイムアウト

8 VIN入力タイムアウト

1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

2 2 定数設定値範囲外

3 0 ソース画面ウィンドウ設定エラー

3 3 デスティネーション画面ウィンドウ設定エラー

4 9 ソース画面が不定画面

- エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面の個々の画素値と定数を比較し、値の大きい方をデスティネーション画面に出力します。

結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付8ビットの場合は1 2 7 / - 1 2 8、符号なし8ビットの場合は2 5 5 / 0を出力します。

ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します (2 5 5 は - 1 として処理します)。

定数には - 2 5 6 ~ 2 5 5 の値が設定できますが、定数値が - 2 5 6 ~ - 1 2 9 の範囲では、結果はソース画面になります。

演算式	$F = \text{MAX} (f , \text{constant})$

F : デスティネーション画面
f : ソース画面

2.8.10 濃度変換テーブル書き込み (WriteConvertLUT)

□ C言語API

```
int ret = WriteConvertLUT(CNVLUT *lut);
```

□ パラメータ

CNVLUT *lut 濃度変換テーブルへ書き込むデータテーブルのアドレス
 テーブルはint型で256個の容量を確保して下さい

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ パケットの構造

```
typedef int CNVLUT
```

❏ エラーコード

2 OSシステムエラー
3 IMP処理タイムアウト
8 VIN入力タイムアウト
20 濃度変換テーブルへ書き込むデータテーブルのアドレスが不正
— エラーリセットコマンド (ClearIPError) 未発行

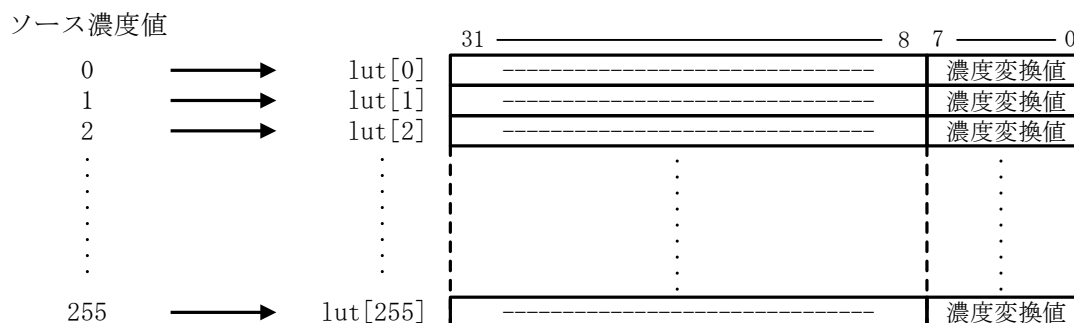
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

機能

濃度変換テーブルへ濃度変換値を書き込みます。濃度変換は『IP_ConvertLUT()』コマンドで行います。本コマンドで書き込まれた濃度変換テーブルの設定値は書き換えられるまで有効です。濃度変換テーブルの書き換えを行うコマンドを以下に示します。

分類	コマンド名称
画素変換	WriteConvertLUT
ラベリング	IP_Label4 / IP_Label8 IP_Label4withAreaFLT / IP_Label8withAreaFLT IP_Label4withAreaFLTSort / IP_Label8withAreaFLTSort

lut テーブルのフォーマットを以下に示します。lut テーブルの濃度変換値はint型の下位 8 bitのみ有効で、上位 24 bitは無効となります。



2.8.11 濃度変換 (IP_ConvertLUT)

□ C言語API

```
int ret = IP_ConvertLUT(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇1	◇1	◇1

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
3 0	ソース画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 9	ソース画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

濃度変換テーブルの内容で、画面データを書き換えます。

このコマンドを発行する前に、必ず『WriteConvertLUT()』コマンドで濃度変換テーブルへの書き込みをしておいて下さい。『WriteConvertLUT()』コマンドで書き込まれた濃度変換テーブルの設定値は書き換えられるまで有効です。

濃度変換テーブルの書き換えを行うコマンドを以下に示します。

分類	コマンド名称
画素変換	WriteConvertLUT
ラベリング	IP_Label4 / IP_Label8 IP_Label4withAreaFLT / IP_Label8withAreaFLT IP_Label4withAreaFLTSort / IP_Label8withAreaFLTSort

2.8.12 画像シフトダウン演算 (IP_ShiftDown)

□ C言語API

```
int ret = IP_ShiftDown(IMGID ImgSrc, IMGID ImgDst, int scale);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	scale	シフトダウン量 (0～8)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇1	◇1	◇1

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
22	シフトダウン量設定値範囲外
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

パラメータで指定されたビット数だけ画像データをシフトダウンします。ソース画面データタイプが符号付8ビットの場合は算術シフトになります。2値の場合は、符号付8ビットに変換して算術シフトします (255は-1として処理します)。そのため、白データ (255) はシフトダウン量にかかわらず白データ (255) になります。

また、IP_ShiftDownは、画像データのシフトダウンされたデータに対して四捨五入処理を行いません。つまり、シフトダウン後のデータの有効な8ビットのデータを、そのままデスティネーション画面に格納します。

例) 3ビットシフトダウンした場合

ソースデータ (符号なし8ビット)		デスティネーションデータ
128	10000000	16 00010000
	論理シフト	
ソースデータ (符号付8ビット)		デスティネーションデータ
-128	10000000	-16 11110000
	算術シフト	

2.8.13 画像シフトアップ演算 (IP_ShiftUp)

□ C言語API

```
int ret = IP_ShiftUp(IMGID ImgSrc, IMGID ImgDst, int scale);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	scale	シフトアップ量 (0～8)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇1	◇1	◇1

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 2	シフトアップ量設定値範囲外
3 0	ソース画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 9	ソース画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

パラメータで指定されたビット数だけ画像データをシフトアップします。ソース画面データタイプが符号付8ビットの場合も論理シフトになります。2値の場合は画素データの最上位ビットを演算に使用し、1または0の論理シフトを行います。

また、画像データのシフトアップされたデータに対して、クリッピング処理を行いません。つまり、シフトアップ後のデータが255を越える場合であっても、下位8ビットのデータをそのままデスティネーション画面に格納します。

例) 4ビットシフトアップした場合

ソースデータ (符号なし8ビット)		デスティネーションデータ
2 4 0	1 1 1 1 0 0 0 0	0 0 0 0 0 0 0 0 0
	論理シフト	

ソースデータ (符号付8ビット)		デスティネーションデータ
- 1 6	1 1 1 1 0 0 0 0	0 0 0 0 0 0 0 0 0
	論理シフト	

2.9 画像間算術演算

2.9.1 加算 (IP_Add)

□ C言語API

```
int ret = IP_Add(IMGID ImgSrc0, IMGID ImgSrc1, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc0 ソース 0 画面番号

IMGID ImgSrc1 ソース 1 画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

2 OSシステムエラー

3 IMP処理タイムアウト

8 VIN入力タイムアウト

1 0 ソース 0 画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 1 ソース 1 画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

3 0 ソース 0 画面ウィンドウ設定エラー

3 1 ソース 1 画面ウィンドウ設定エラー

3 3 デスティネーション画面ウィンドウ設定エラー

4 9 ソース 0 画面またはソース 1 画面が不定画面

- エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

ソース 0 画面とソース 1 画面を加算し、結果をデスティネーション画面に出力します。
画面データタイプが 2 値の場合は、画素データの最上位ビット (1 または 0) を演算に使用します。
結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付 8 ビットの場合は 1 2 7
／-1 2 8、符号なし 8 ビットの場合は 2 5 5 / 0 を出力します。

演算式	$F = f + g$

F : デスティネーション画面
f : ソース 0 画面
g : ソース 1 画面

2.9.2 減算 (IP_Sub)

□ C言語API

```
int ret = IP_Sub(IMGID ImgSrc0, IMGID ImgSrc1 IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc0 ソース 0 画面番号

IMGID ImgSrc1 ソース 1 画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

2 OSシステムエラー

3 IMP処理タイムアウト

8 VIN入力タイムアウト

1 0 ソース 0 画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 1 ソース 1 画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

3 0 ソース 0 画面ウィンドウ設定エラー

3 1 ソース 1 画面ウィンドウ設定エラー

3 3 デスティネーション画面ウィンドウ設定エラー

4 9 ソース 0 画面またはソース 1 画面が不定画面

- エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース 0 画面からソース 1 画面を減算し、結果をデスティネーション画面に出力します。
画面データタイプが 2 値の場合は、画素データの最上位ビット (1 または 0) を演算に使用します。
結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付 8 ビットの場合は 1 2 7
／- 1 2 8、符号なし 8 ビットの場合は 2 5 5 / 0 を出力します。

演算式	$F = f - g$

F : デスティネーション画面
f : ソース 0 画面
g : ソース 1 画面

2.9.3 減算絶対値 (IP_SubAbs)

□ C言語API

```
int ret = IP_SubAbs(IMGID ImgSrc0, IMGID ImgSrc1, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc0 ソース 0 画面番号

IMGID ImgSrc1 ソース 1 画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

2 OSシステムエラー

3 IMP処理タイムアウト

8 VIN入力タイムアウト

1 0 ソース 0 画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 1 ソース 1 画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

3 0 ソース 0 画面ウィンドウ設定エラー

3 1 ソース 1 画面ウィンドウ設定エラー

3 3 デスティネーション画面ウィンドウ設定エラー

4 9 ソース 0 画面またはソース 1 画面が不定画面

- エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース 0 画面からソース 1 画面を減算し、絶対値をデスティネーション画面に出力します。
画面データタイプが 2 値の場合は、画素データの最上位ビット (1 または 0) を演算に使用します。
結果がオーバーフローしたとき、システムデータタイプが符号付 8 ビットの場合は 1 2 7、符号なし 8 ビットの場合は 2 5 5 を出力します。

演算式	$F = f - g $

F : デスティネーション画面
f : ソース 0 画面
g : ソース 1 画面

2.9.4 係数付加算 (IP_Comb)

□ C言語API

```
int ret = IP_Comb(IMGID ImgSrc0, IMGID ImgSrc1,
                  IMGID ImgDst, int scale, int a, int b);
```

□ パラメータ

IMGID	ImgSrc0	ソース 0 画面番号
IMGID	ImgSrc1	ソース 1 画面番号
IMGID	ImgDst	デスティネーション画面番号
int	scale	ダウンシフト量 (0 ~ 1 5)
int	a	係数 a (有効範囲 - 2 5 6 ~ 2 5 5)
int	b	係数 b (有効範囲 - 2 5 6 ~ 2 5 5)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

2	OS システムエラー
3	IMP 処理タイムアウト
8	VIN 入力タイムアウト
1 0	ソース 0 画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 1	ソース 1 画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 3	ダウンシフト量設定値範囲外
2 4	係数 a 設定値範囲外
2 5	係数 b 設定値範囲外
3 0	ソース 0 画面ウィンドウ設定エラー
3 1	ソース 1 画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 9	ソース 0 画面またはソース 1 画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

ソース0画面と係数aを乗算したものと、ソース1画面と係数bを乗算したものを加算し、結果をダウンシフトしてデスティネーション画面に出力します。

画面データタイプが符号付8ビットの場合は、ダウンシフトは算術シフトになります。

画面データタイプが2値の場合は、画素データの最上位ビット（1または0）を演算に使用します。

また、ダウンシフトは四捨五入を行い、ソース画面データタイプが符号付8ビットの場合-0.5は0になります。ダウンシフトした結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付8ビットの場合は127／-128、符号なし8ビットの場合は255／0を出力します。

演算式	$F = f \times a + g \times b$

F：デスティネーション画面

f：ソース0画面

g：ソース1画面

ダウンシフト量	処理内容
0	演算結果ダウンシフトなし
1	演算結果1／2
2	演算結果1／4
3	演算結果1／8
⋮	⋮
15	演算結果1／32768

2.9.5 係数付加算絶対値 (IP_CombAbs)

□ C言語API

```
int ret = IP_CombAbs(IMGID ImgSrc0, IMGID ImgSrc1, IMGID ImgDst, int scale, int a, int b);
```

□ パラメータ

IMGID	ImgSrc0	ソース 0 画面番号
IMGID	ImgSrc1	ソース 1 画面番号
IMGID	ImgDst	デスティネーション画面番号
int	scale	ダウンシフト量 (0 ~ 15)
int	a	係数 a (有効範囲 -256 ~ 255)
int	b	係数 b (有効範囲 -256 ~ 255)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース0画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
11	ソース1画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
23	ダウンシフト量設定値範囲外
24	係数a設定値範囲外
25	係数b設定値範囲外
30	ソース0画面ウィンドウ設定エラー
31	ソース1画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース0画面またはソース1画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース0画面と係数aを乗算したものと、ソース1画面と係数bを乗算したものを加算し、絶対値化した結果をダウンシフトしてデスティネーション画面に出力します。

画面データタイプが符号付8ビットの場合は、ダウンシフトは算術シフトになります。

画面データタイプが2値の場合は、画素データの最上位ビット（1または0）を演算に使用します。

また、ダウンシフトは四捨五入を行います。ダウンシフトした結果がオーバーフローしたとき、システムデータタイプが符号付8ビットの場合は127、符号なし8ビットの場合は255を出力します。

演算式	$F = f \times a + g \times b $

F：デスティネーション画面

f：ソース0画面

g：ソース1画面

ダウンシフト量	処理内容
0	演算結果ダウンシフトなし
1	演算結果1／2
2	演算結果1／4
3	演算結果1／8
⋮	⋮
15	演算結果1／32768

2.9.6 乗算 (IP_Mult)

□ C言語API

```
int ret = IP_Mult(IMGID ImgSrc0, IMGID ImgSrc1, IMGID ImgDst, int scale);
```

□ パラメータ

IMGID	ImgSrc0	ソース 0 画面番号
IMGID	ImgSrc1	ソース 1 画面番号
IMGID	ImgDst	デスティネーション画面番号
int	scale	ダウンシフト量 (0 ~ 15)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	—

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OS システムエラー
3	IMP 処理タイムアウト
8	VIN 入力タイムアウト
1 0	ソース 0 画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 1	ソース 1 画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 3	ダウンシフト量設定値範囲外
3 0	ソース 0 画面ウィンドウ設定エラー
3 1	ソース 1 画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 9	ソース 0 画面またはソース 1 画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

ソース0画面とソース1画面を乗算し、結果をダウンシフトしてデスティネーション画面に出力します。

画面データタイプが符号付8ビットの場合は、ダウンシフトは算術シフトになります。

画面データタイプが2値の場合は、画素データの最上位ビット（1または0）を演算に使用します。

また、ダウンシフトは四捨五入を行い、ソース画面データタイプが符号付8ビットの場合-0.5は0になります。ダウンシフトした結果がオーバー/アンダーフローしたとき、システムデータタイプが符号付8ビットの場合は127/ー128、符号なし8ビットの場合は255/0を出力します。

演算式	$F = f \times g$

F：デスティネーション画面

f：ソース0画面

g：ソース1画面

ダウンシフト量	処理内容
0	演算結果ダウンシフトなし
1	演算結果1／2
2	演算結果1／4
3	演算結果1／8
⋮	⋮
15	演算結果1／32768

2.9.7 平均 (IP_Average)

□ C言語API

```
int ret = IP_Average(IMGID ImgSrc0, IMGID ImgSrc1, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc0 ソース 0 画面番号

IMGID ImgSrc1 ソース 1 画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

2 OS システムエラー

3 IMP 処理タイムアウト

8 VIN 入力タイムアウト

1 0 ソース 0 画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 1 ソース 1 画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

3 0 ソース 0 画面ウィンドウ設定エラー

3 1 ソース 1 画面ウィンドウ設定エラー

3 3 デスティネーション画面ウィンドウ設定エラー

4 9 ソース 0 画面またはソース 1 画面が不定画面

- エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

ソース 0 画面とソース 1 画面の平均を、デスティネーション画面に出力します。

画面データタイプが 2 値の場合は、画素データの最上位ビット (1 または 0) を演算に使用します。

処理結果は四捨五入を行い、デスティネーション画面データタイプが符号付 8 ビットの場合 - 0. 5 は 0 になります。

結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付 8 ビットの場合は 1 2 7 / - 1 2 8、符号なし 8 ビットの場合は 2 5 5 / 0 を出力します。

演算式	$F = (f + g) / 2$

F : デスティネーション画面
f : ソース 0 画面
g : ソース 1 画面

2.9.8 比較 (Min) (IP_Min)

□ C言語API

```
int ret = IP_Min(IMGID ImgSrc0, IMGID ImgSrc1, IMGID ImgDst);
```

□ パラメータ

IMGID	ImgSrc0	ソース 0 画面番号
IMGID	ImgSrc1	ソース 1 画面番号
IMGID	ImgDst	デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	ソース 0 画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 1	ソース 1 画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
3 0	ソース 0 画面ウィンドウ設定エラー
3 1	ソース 1 画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 9	ソース 0 画面またはソース 1 画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

ソース 0 画面とソース 1 画面の個々の画素値を比較し、値の小さい方をデスティネーション画面に出力します。
画面データタイプが 2 値の場合は、画素データの最上位ビット (1 または 0) を演算に使用します。
結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付 8 ビットの場合は 1 2 7 / - 1 2 8、符号なし 8 ビットの場合は 2 5 5 / 0 を出力します。

演算式	$F = \text{MIN} (f , g)$
	F : デスティネーション画面 f : ソース 0 画面 g : ソース 1 画面

2.9.9 比較 (Max) (IP_Max)

□ C言語API

```
int ret = IP_Max(IMGID ImgSrc0, IMGID ImgSrc1, IMGID ImgDst);
```

□ パラメータ

IMGID	ImgSrc0	ソース 0 画面番号
IMGID	ImgSrc1	ソース 1 画面番号
IMGID	ImgDst	デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

2	OS システムエラー
3	IMP 処理タイムアウト
8	VIN 入力タイムアウト
1 0	ソース 0 画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 1	ソース 1 画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
3 0	ソース 0 画面ウィンドウ設定エラー
3 1	ソース 1 画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 9	ソース 0 画面またはソース 1 画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

ソース 0 画面とソース 1 画面の個々の画素値を比較し、値の大きい方をデスティネーション画面に出力します。

画面データタイプが 2 値の場合は、画素データの最上位ビット (1 または 0) を演算に使用します。

結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付 8 ビットの場合は 1 2 7 / - 1 2 8、符号なし 8 ビットの場合は 2 5 5 / 0 を出力します。

演算式	$F = \text{MAX} (f , g)$

F : デスティネーション画面

f : ソース 0 画面

g : ソース 1 画面

2.9.10 定数減算絶対値和 (IP_SubConstAbsAdd)

□ C言語API

```
int ret = IP_SubConstAbsAdd(IMGID ImgSrc0, IMGID ImgSrc1, IMGID ImgDst,
                             int scale, int a, int b);
```

□ パラメータ

IMGID	ImgSrc0	ソース 0 画面番号
IMGID	ImgSrc1	ソース 1 画面番号
IMGID	ImgDst	デスティネーション画面番号
int	scale	ダウンシフト量 (0 ~ 15)
int	a	係数 a (有効範囲 -256 ~ 255)
int	b	係数 b (有効範囲 -256 ~ 255)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース0画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
11	ソース1画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
23	ダウンシフト量設定値範囲外
24	係数 a 設定値範囲外
25	係数 b 設定値範囲外
30	ソース0画面ウィンドウ設定エラー
31	ソース1画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース0画面またはソース1画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース0画面から定数aを引いたものの絶対値と、ソース1画面から定数bを引いたものの絶対値とを加算した結果をダウンシフトしてデスティネーション画面に出力します。

画面データタイプが符号付8ビットの場合は、ダウンシフトは算術シフトになります。

画面データタイプが2値の場合は、画素データの最上位ビット（1または0）を演算に使用します。

また、ダウンシフトは四捨五入を行います。ダウンシフトした結果がオーバーフローしたとき、システムデータタイプが符号付8ビットの場合は127、符号なし8ビットの場合は255を出力します。

演算式	$F = f - a + g - b $

F: デスティネーション画面

f: ソース0画面

g: ソース1画面

ダウンシフト量	処理内容
0	演算結果ダウンシフトなし
1	演算結果1／2
2	演算結果1／4
3	演算結果1／8
⋮	⋮
15	演算結果1／32768

2.9.11 定数減算自乗和 (IP_SubConstMultAdd)

□ C言語API

```
int ret = IP_SubConstMultAdd(IMGID ImgSrc0, IMGID ImgSrc1, IMGID ImgDst,
                             int scale, int a, int b);
```

□ パラメータ

IMGID	ImgSrc0	ソース 0 画面番号
IMGID	ImgSrc1	ソース 1 画面番号
IMGID	ImgDst	デスティネーション画面番号
int	scale	ダウンシフト量 (0 ~ 15)
int	a	係数 a (有効範囲 -256 ~ 255)
int	b	係数 b (有効範囲 -256 ~ 255)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース0画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
11	ソース1画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
23	ダウンシフト量設定値範囲外
24	係数 a 設定値範囲外
25	係数 b 設定値範囲外
30	ソース0画面ウィンドウ設定エラー
31	ソース1画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース0画面またはソース1画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース0画面から定数aを引いたものの自乗と、ソース1画面から定数bを引いたものの自乗とを加算した結果をダウンシフトしてデスティネーション画面に出力します。

画面データタイプが符号付8ビットの場合は、ダウンシフトは算術シフトになります。

画面データタイプが2値の場合は、画素データの最上位ビット（1または0）を演算に使用します。

また、ダウンシフトは四捨五入を行います。ダウンシフトした結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付8ビットの場合は127／－128、符号なし8ビットの場合は255／0を出力します。

演算式	$F = (f - a) \times (f - a) + (g - b) \times (g - b)$

F：デスティネーション画面

f：ソース0画面

g：ソース1画面

ダウンシフト量	処理内容
0	演算結果ダウンシフトなし
1	演算結果1／2
2	演算結果1／4
3	演算結果1／8
⋮	⋮
15	演算結果1／32768

2.9.12 定数減算積 (IP_SubConstMult)

□ C言語API

```
int ret = IP_SubConstMult(IMGID ImgSrc0, IMGID ImgSrc1, IMGID ImgDst, int scale, int a, int b);
```

□ パラメータ

IMGID	ImgSrc0	ソース 0 画面番号
IMGID	ImgSrc1	ソース 1 画面番号
IMGID	ImgDst	デスティネーション画面番号
int	scale	ダウンシフト量 (0 ~ 15)
int	a	係数 a (有効範囲 -256 ~ 255)
int	b	係数 b (有効範囲 -256 ~ 255)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース0画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
11	ソース1画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
23	ダウンシフト量設定値範囲外
24	係数 a 設定値範囲外
25	係数 b 設定値範囲外
30	ソース0画面ウィンドウ設定エラー
31	ソース1画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース0画面またはソース1画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース0画面から定数aを引いたものと、ソース1画面から定数bを引いたものとを乗算した結果をダウンスフトしてデスティネーション画面に出力します。

画面データタイプが符号付8ビットの場合は、ダウンスフトは算術シフトになります。

画面データタイプが2値の場合は、画素データの最上位ビット（1または0）を演算に使用します。

また、ダウンスフトは四捨五入を行い、ソース画面データタイプが符号付8ビットの場合-0.5は0になります。ダウンスフトした結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付8ビットの場合は127／-128、符号なし8ビットの場合は255／0を出力します。

演算式	$F = (f - a) \times (g - b)$

F：デスティネーション画面

f：ソース0画面

g：ソース1画面

ダウンスフト量	処理内容
0	演算結果ダウンスフトなし
1	演算結果1／2
2	演算結果1／4
3	演算結果1／8
⋮	⋮
15	演算結果1／32768

2.9.13 係数付加算（切り捨て）（IP_CombDrop）

□ C言語API

```
int ret = IP_CombDrop(IMGID ImgSrc0, IMGID ImgSrc1, IMGID ImgDst, int scale, int a, int b);
```

□ パラメータ

IMGID	ImgSrc0	ソース 0 画面番号
IMGID	ImgSrc1	ソース 1 画面番号
IMGID	ImgDst	デスティネーション画面番号
int	scale	ダウンシフト量（0～15）
int	a	係数 a（有効範囲 -256～255）
int	b	係数 b（有効範囲 -256～255）

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース0画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
11	ソース1画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
23	ダウンシフト量設定値範囲外
24	係数 a 設定値範囲外
25	係数 b 設定値範囲外
30	ソース0画面ウィンドウ設定エラー
31	ソース1画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース0画面またはソース1画面が不定画面
-	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース0画面と係数aを乗算したものと、ソース1画面と係数bを乗算したものを加算し、結果をダウンシフトしてデスティネーション画面に出力します。

画面データタイプが符号付8ビットの場合は、ダウンシフトは算術シフトになります。

画面データタイプが2値の場合は、画素データの最上位ビット（1または0）を演算に使用します。

本コマンドは、『IP_Comb()』コマンド（係数付加算）の四捨五入を行わない切り捨てバージョンです。ダウンシフトした結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付8ビットの場合は1 2 7／－1 2 8、符号なし8ビットの場合は2 5 5／0を出力します。

演算式	$F = f \times a + g \times b$

F：デスティネーション画面

f：ソース0画面

g：ソース1画面

ダウンシフト量	処理内容
0	演算結果ダウンシフトなし
1	演算結果1／2
2	演算結果1／4
3	演算結果1／8
⋮	⋮
15	演算結果1／3 2 7 6 8

2.10 画像間論理演算

2.10.1 論理積 (IP_And)

□ C言語API

```
int ret = IP_And(IMGID ImgSrc0, IMGID ImgSrc1, IMGID ImgDst);
```

□ パラメータ

IMGID	ImgSrc0	ソース 0 画面番号
IMGID	ImgSrc1	ソース 1 画面番号
IMGID	ImgDst	デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇	◇	◇

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OS システムエラー
3	IMP 処理タイムアウト
8	VIN 入力タイムアウト
1 0	ソース 0 画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 1	ソース 1 画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
3 0	ソース 0 画面ウィンドウ設定エラー
3 1	ソース 1 画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 9	ソース 0 画面またはソース 1 画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

ソース 0 画面とソース 1 画面の論理積を、デスティネーション画面に出力します。

演算式	$F = f \text{ and } g$

F : デスティネーション画面
f : ソース 0 画面
g : ソース 1 画面

2.10.2 論理和 (IP_Or)

□ C言語API

```
int ret = IP_Or(IMGID ImgSrc0, IMGID ImgSrc1, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc0 ソース 0 画面番号

IMGID ImgSrc1 ソース 1 画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇	◇	◇

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

2 OS システムエラー

3 IMP 処理タイムアウト

8 VIN 入力タイムアウト

1 0 ソース 0 画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 1 ソース 1 画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

3 0 ソース 0 画面ウィンドウ設定エラー

3 1 ソース 1 画面ウィンドウ設定エラー

3 3 デスティネーション画面ウィンドウ設定エラー

4 9 ソース 0 画面またはソース 1 画面が不定画面

— エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

ソース 0 画面とソース 1 画面の論理和を、デスティネーション画面に出力します。

演算式	$F = f \text{ or } g$

F : デスティネーション画面
f : ソース 0 画面
g : ソース 1 画面

2.10.3 排他的論理和 (IP_Xor)

□ C言語API

```
int ret = IP_Xor(IMGID ImgSrc0, IMGID ImgSrc1, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc0 ソース 0 画面番号

IMGID ImgSrc1 ソース 1 画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇	◇	◇

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

2 OSシステムエラー

3 IMP処理タイムアウト

8 VIN入力タイムアウト

1 0 ソース 0 画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 1 ソース 1 画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

3 0 ソース 0 画面ウィンドウ設定エラー

3 1 ソース 1 画面ウィンドウ設定エラー

3 3 デスティネーション画面ウィンドウ設定エラー

4 9 ソース 0 画面またはソース 1 画面が不定画面

— エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

ソース 0 画面とソース 1 画面の排他的論理和を、デスティネーション画面に出力します。

演算式	$F = f \text{ xor } g$

F : デスティネーション画面

f : ソース 0 画面

g : ソース 1 画面

2.10.4 入力反転付論理積 (IP_InvertAnd)

□ C言語API

```
int ret = IP_InvertAnd(IMGID ImgSrc0, IMGID ImgSrc1, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc0 ソース 0 画面番号

IMGID ImgSrc1 ソース 1 画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇	◇	◇

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

2 OS システムエラー

3 IMP 処理タイムアウト

8 VIN 入力タイムアウト

1 0 ソース 0 画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 1 ソース 1 画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

3 0 ソース 0 画面ウィンドウ設定エラー

3 1 ソース 1 画面ウィンドウ設定エラー

3 3 デスティネーション画面ウィンドウ設定エラー

4 9 ソース 0 画面またはソース 1 画面が不定画面

— エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

ソース 0 画面とソース 1 画面の入力反転付論理積を、デスティネーション画面に出力します。

演算式	$F = \sim f \text{ and } g$

F : デスティネーション画面
f : ソース 0 画面
g : ソース 1 画面

2.10.5 入力反転付論理和（IP_InvertOr）

□ C言語API

```
int ret = IP_InvertOr(IMGID ImgSrc0, IMGID ImgSrc1, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc0 ソース 0 画面番号

IMGID ImgSrc1 ソース 1 画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇	◇	◇

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2 OSシステムエラー

3 IMP処理タイムアウト

8 VIN入力タイムアウト

1 0 ソース 0 画面番号に対応する画面が未確保、または番号が有効範囲外
（不当画面番号エラー）

1 1 ソース 1 画面番号に対応する画面が未確保、または番号が有効範囲外
（不当画面番号エラー）

1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
（不当画面番号エラー）

3 0 ソース 0 画面ウィンドウ設定エラー

3 1 ソース 1 画面ウィンドウ設定エラー

3 3 デスティネーション画面ウィンドウ設定エラー

4 9 ソース 0 画面またはソース 1 画面が不定画面

— エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース 0 画面とソース 1 画面の入力反転付論理和を、デスティネーション画面に出力します。

演算式	$F = \sim f \text{ or } g$

F：デスティネーション画面
f：ソース 0 画面
g：ソース 1 画面

2.10.6 排他的否定論理和 (IP_Xnor)

□ C言語API

```
int ret = IP_Xnor(IMGID ImgSrc0, IMGID ImgSrc1, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc0 ソース 0 画面番号

IMGID ImgSrc1 ソース 1 画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc0	○	△R	○	○	○	×	○	○	○
ImgSrc1	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇	◇	◇

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

2 OS システムエラー

3 IMP 処理タイムアウト

8 VIN 入力タイムアウト

1 0 ソース 0 画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 1 ソース 1 画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

3 0 ソース 0 画面ウィンドウ設定エラー

3 1 ソース 1 画面ウィンドウ設定エラー

3 3 デスティネーション画面ウィンドウ設定エラー

4 9 ソース 0 画面またはソース 1 画面が不定画面

— エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

ソース 0 画面とソース 1 画面の排他的否定論理和を、デスティネーション画面に出力します。

演算式	$F = f \text{ xnor } g$

F : デスティネーション画面
f : ソース 0 画面
g : ソース 1 画面

2.11 2値画像形状変換

2.11.1 2値画像ノイズ除去(4連結) (IP_PickNoise4)

□ C言語API

```
int ret = IP_PickNoise4(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○
ImgDst	○	△R	○	○	○	×	-	-	◆

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

2値画像中の独立した1画素（黒で囲まれた白、または白で囲まれた黒）を除去します。
 本コマンドは、ソース画面が濃淡画像の場合も2値画像として処理します。最上位ビットが1の場合は白、0の場合は黒として処理します。結果画像は0と255の2値画像になります。
 指定領域の周辺(内側1画素の領域)は、不定値になります。
 処理内容は、次ページの『IP_PickNoise8()』コマンドを参照して下さい。

2.11.2 2値画像ノイズ除去（8連結）（IP_PickNoise8）

□ C言語API

```
int ret = IP_PickNoise8(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○
ImgDst	○	△R	○	○	○	×	-	-	◆

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

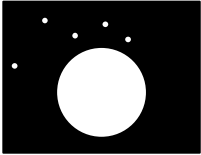
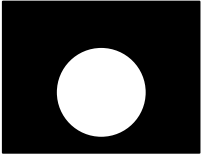
2 OSシステムエラー
3 IMP処理タイムアウト
8 VIN入力タイムアウト
1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
3 0 ソース画面ウィンドウ設定エラー
3 3 デスティネーション画面ウィンドウ設定エラー
4 9 ソース画面が不定画面
- エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

2値画像中の独立した1画素（黒で囲まれた白、または白で囲まれた黒）を除去します。
本コマンドは、ソース画面が濃淡画像の場合も2値画像として処理します。最上位ビットが1の場合は白、0の場合は黒として処理します。結果画像は0と255の2値画像になります。
指定領域の周辺(内側1画素の領域)は、不定値になります。

4連結と8連結の処理内容を次ページに示します。

	4 連結	8 連結
処 理 内 容	処理前 処理後 <pre> 1 0 1 0 → 1 0 0 1 0 0 </pre> 処理前 処理後 <pre> 0 0 1 0 → 0 0 0 0 0 0 </pre> 処理前 処理後 <pre> 1 1 0 1 → 1 1 1 1 1 1 </pre> 中央の画素が上下左右の全ての画素と違う値のときに、除去する。	処理前 処理後 <pre> 1 0 0 0 1 0 → 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 </pre> 処理前 処理後 <pre> 0 0 0 0 1 0 → 0 0 0 0 0 0 0 0 0 </pre> 処理前 処理後 <pre> 1 1 1 1 0 1 → 1 1 1 1 1 1 1 1 1 </pre> 中央の画素が上下左右、斜めの全ての画素と違う値のときに、除去する。
処 理 例	  白の独立点除去	

2.11.3 2値画像輪郭抽出（4連結）（IP_Outline4）

□ C言語API

```
int ret = IP_Outline4(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○
ImgDst	○	△R	○	○	○	×	-	-	◆

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2 OSシステムエラー
3 IMP処理タイムアウト
8 VIN入力タイムアウト
1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
3 0 ソース画面ウィンドウ設定エラー
3 3 デスティネーション画面ウィンドウ設定エラー
4 9 ソース画面が不定画面
- エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

2値画像の白の部分の輪郭を抽出します。

本コマンドは、ソース画面が濃淡画像の場合も2値画像として処理します。最上位ビットが1の場合は白、0の場合は黒として処理します。結果画像は0と255の2値画像になります。

指定領域の周辺(内側1画素の領域)は、0を出力します。

処理内容は、次ページの『IP_Outline8()』コマンドを参照して下さい。

2.11.4 2値画像輪郭抽出（8連結）（IP_Outline8）

□ C言語API

```
int ret = IP_Outline8(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○
ImgDst	○	△R	○	○	○	×	-	-	◆

□ リターンパラメータ

int ret 正常終了（0）または異常終了（- 1）

□ エラーコード

2 OSシステムエラー
3 IMP処理タイムアウト
8 VIN入力タイムアウト
1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
3 0 ソース画面ウィンドウ設定エラー
3 3 デスティネーション画面ウィンドウ設定エラー
4 9 ソース画面が不定画面
- エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

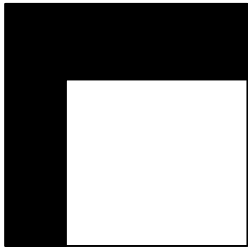
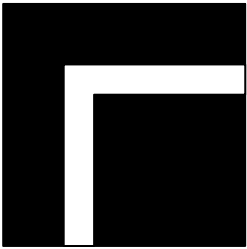
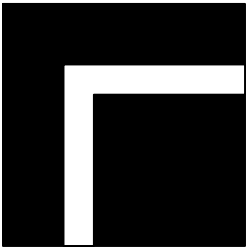
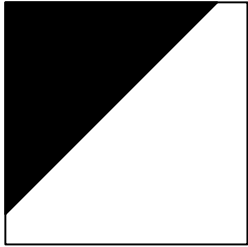
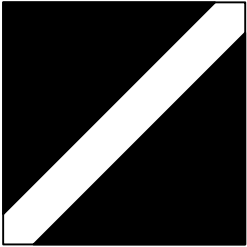
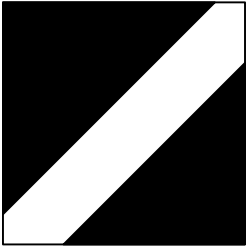
□ 機能

2 値画像の白の部分の輪郭を抽出します。

本コマンドは、ソース画面が濃淡画像の場合も2 値画像として処理します。最上位ビットが1 の場合は白、0 の場合は黒として処理します。結果画像は0 と2 5 5 の2 値画像になります。

指定領域の周辺(内側1 画素の領域)は、0 を出力します。

4 連結と8 連結の処理内容を次ページに示します。

	処理前	4 連結処理後	8 連結処理後
処 理 例 1	<pre> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 </pre> 	<pre> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 </pre> 	<pre> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 </pre> 
処 理 例 2	<pre> 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 </pre> 	<pre> 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 </pre> 	<pre> 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 </pre> 

2.11.5 2値画像膨張（4連結）（IP_Dilation4）

□ C言語API

```
int ret = IP_Dilation4(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○
ImgDst	○	△R	○	○	○	×	-	-	◆

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2 OSシステムエラー
3 IMP処理タイムアウト
8 VIN入力タイムアウト
1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
3 0 ソース画面ウィンドウ設定エラー
3 3 デスティネーション画面ウィンドウ設定エラー
4 9 ソース画面が不定画面
- エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

2値画像の白の部分を外側へ1画素膨張させます。

本コマンドは、ソース画面が濃淡画像の場合も2値画像として処理します。最上位ビットが1の場合は白、0の場合は黒として処理します。結果画像は0と255の2値画像になります。

指定領域の周辺(内側1画素の領域)は、不定値になります。

処理内容は、次ページの『IP_Dilation8()』コマンドを参照して下さい。

2.11.6 2値画像膨張（8連結）（IP_Dilation8）

□ C言語API

```
int ret = IP_Dilation8(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○
ImgDst	○	△R	○	○	○	×	-	-	◆

□ リターンパラメータ

int ret 正常終了（0）または異常終了（- 1）

□ エラーコード

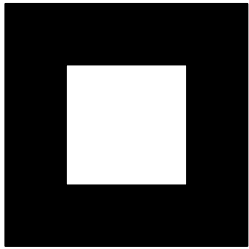
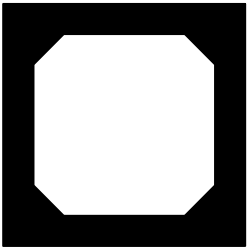
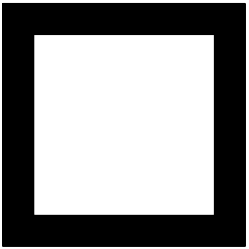
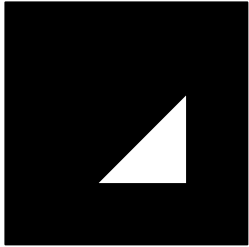
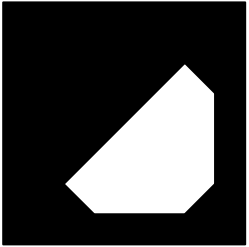
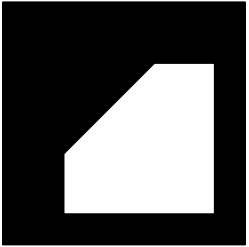
2 OSシステムエラー
3 IMP処理タイムアウト
8 VIN入力タイムアウト
1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
3 0 ソース画面ウィンドウ設定エラー
3 3 デスティネーション画面ウィンドウ設定エラー
4 9 ソース画面が不定画面
- エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

2 値画像の白の部分を外側へ1画素膨張させます。
本コマンドは、ソース画面が濃淡画像の場合も2 値画像として処理します。最上位ビットが1 の場合は白、0 の場合は黒として処理します。結果画像は0 と2 5 5 の2 値画像になります。
指定領域の周辺(内側1画素の領域)は、不定値になります。

4 連結と 8 連結の処理内容を以下に示します。

	処理前	4 連結処理後	8 連結処理後
処 理 例 1	<pre> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 </pre> 	<pre> 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 </pre> 	<pre> 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 </pre> 
処 理 例 2	<pre> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 </pre> 	<pre> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 </pre> 	<pre> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 </pre> 

4連結 上下左右に1画素膨張

8連結 上下左右斜めに1画素膨張

2.11.7 2値画像収縮（4連結）（IP_Erosion4）

□ C言語API

```
int ret = IP_Erosion4(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○
ImgDst	○	△R	○	○	○	×	-	-	◆

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2 OSシステムエラー
3 IMP処理タイムアウト
8 VIN入力タイムアウト
1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
3 0 ソース画面ウィンドウ設定エラー
3 3 デスティネーション画面ウィンドウ設定エラー
4 9 ソース画面が不定画面
- エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

2値画像の白の部分を内側へ1画素収縮させます。

本コマンドは、ソース画面が濃淡画像の場合も2値画像として処理します。最上位ビットが1の場合は白、0の場合は黒として処理します。結果画像は0と255の2値画像になります。

指定領域の周辺(内側1画素の領域)は、0を出力します。

処理内容は、次ページの『IP_Erosion8()』コマンドを参照して下さい。

2.11.8 2値画像収縮（8連結）（IP_Erosion8）

□ C言語API

```
int ret = IP_Erosion8(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○
ImgDst	○	△R	○	○	○	×	-	-	◆

□ リターンパラメータ

int ret 正常終了（0）または異常終了（- 1）

□ エラーコード

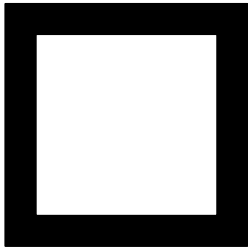
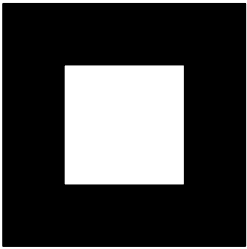
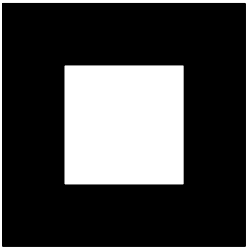
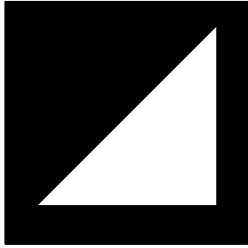
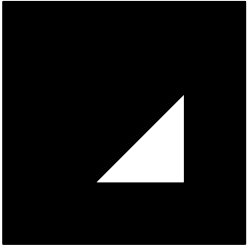
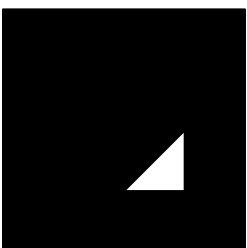
2 OSシステムエラー
3 IMP処理タイムアウト
8 VIN入力タイムアウト
1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
3 0 ソース画面ウィンドウ設定エラー
3 3 デスティネーション画面ウィンドウ設定エラー
4 9 ソース画面が不定画面
- エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

2 値画像の白の部分を内側へ1 画素収縮させます。
本コマンドは、ソース画面が濃淡画像の場合も2 値画像として処理します。最上位ビットが1 の場合は白、0 の場合は黒として処理します。結果画像は0 と2 5 5 の2 値画像になります。
指定領域の周辺(内側1 画素の領域)は、0 を出力します。

4連結と8連結の処理内容を以下に示します。

	処理前	4連結処理後	8連結処理後
処 理 例 1	<pre> 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 </pre> 	<pre> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 </pre> 	<pre> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 </pre> 
処 理 例 2	<pre> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 </pre> 	<pre> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 </pre> 	<pre> 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 </pre> 

4連結 上下左右に1画素収縮

8連結 上下左右斜めに1画素収縮

2.11.9 2値画像細線化（4連結）（IP_Thin4）

□ C言語API

```
int ret = IP_Thin4(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○
ImgDst	○	△R	○	○	○	×	-	-	◆

□ リターンパラメータ

int ret 正常終了（0）または異常終了（- 1）

□ エラーコード

2 OSシステムエラー
 3 IMP処理タイムアウト
 8 VIN入力タイムアウト
 1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
 1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
 3 0 ソース画面ウィンドウ設定エラー
 3 3 デスティネーション画面ウィンドウ設定エラー
 4 9 ソース画面が不定画面
 - エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

2値画像に対し、4連結で細線化処理を行います。

本コマンドは、ソース画面が濃淡画像の場合も2値画像として処理します。最上位ビットが1の場合は白、0の場合は黒として処理します。結果画像は0と255の2値画像になります。

指定領域の周辺(内側2画素の領域)は、0を出力します。

処理画面によっては1回の細線化処理で物体外周の1画素分だけしか削れないことがありますので、完全に細線化するには、複数回処理を行う必要があります。

処理内容は、次ページの『IP_Thin8()』コマンドを参照して下さい。

2.11.10 2値画像細線化（8連結）（IP_Thin8）

□ C言語API

```
int ret = IP_Thin8(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○
ImgDst	○	△R	○	○	○	×	-	-	◆

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2 OSシステムエラー
 3 IMP処理タイムアウト
 8 VIN入力タイムアウト
 1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
 1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
 3 0 ソース画面ウィンドウ設定エラー
 3 3 デスティネーション画面ウィンドウ設定エラー
 4 9 ソース画面が不定画面
 - エラーリセットコマンド（ClearIPError）未発行

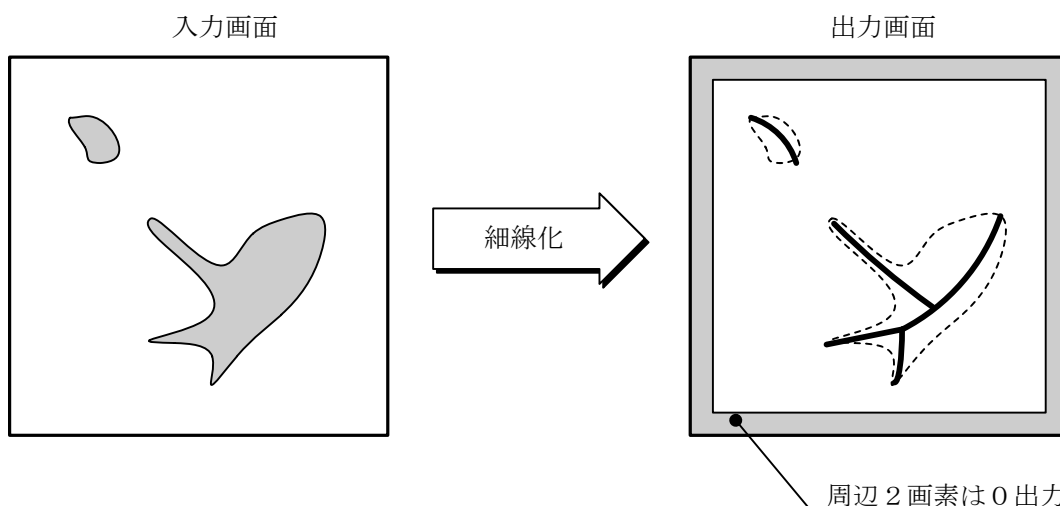
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

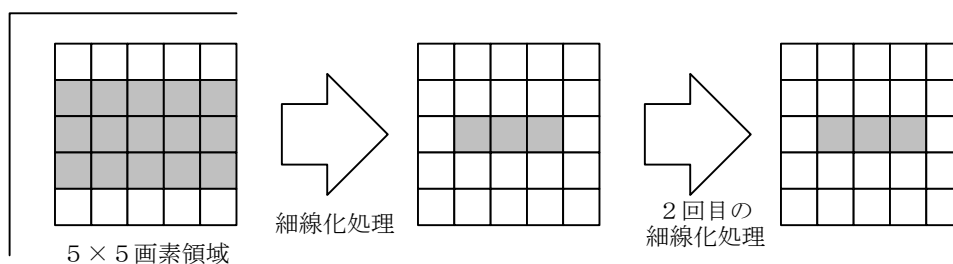
2値画像に対し、8連結で細線化処理を行います。

本コマンドは、ソース画面が濃淡画像の場合も2値画像として処理します。最上位ビットが1の場合は白、0の場合は黒として処理します。結果画像は0と255の2値画像になります。

細線化処理では、入力画面（2値画像）の指定領域内の全画素に対して細線化処理を行い、結果を出力します。また、指定領域の周辺(内側2画素の領域)は、0を出力します。



1回の画像処理で、 5×5 画素領域で局所的な細線化処理を行います。処理画面によっては1回の細線化処理で物体外周の1画素分だけしか削れないことがありますので、完全に細線化するには、複数回処理を行う必要があります。



[細線化処理を完全に行う場合のプログラム例]

2値画像の管理が、物体：255、背景：0で管理している為、物体の面積の変化がなくなったことをヒストグラム処理を用いて検知する（変化しなくなるまで処理する）。

```

IMGID          ImgSrc, ImgDst;
long           Tbl[256];
IPGOFeatureTbl FtrTbl;
int            opt, count;

opt  = 0;
count = 0;      // 2値データの画素数カウント用

for(;;){
    IP_Thin4( ImgSrc, ImgDst );
    ChangeImgDataType( ImgDst, (enum DataType)UNSIGN8_DATA );
    IP_Histogram( ImgDst, Tbl, &FtrTbl, opt );
    if( count == Tbl[255] )
        break;
    count = Tbl[255];
    IP_Copy( ImgDst, ImgSrc );
}

```

2.11.11 2値画像縮退化（4連結）（IP_Shrink4）

□ C言語API

```
int ret = IP_Shrink4(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○
ImgDst	○	△R	○	○	○	×	-	-	◆

□ リターンパラメータ

int ret 正常終了（0）または異常終了（- 1）

□ エラーコード

2 OSシステムエラー
3 IMP処理タイムアウト
8 VIN入力タイムアウト
1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
 (不当画面番号エラー)
1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
 (不当画面番号エラー)
3 0 ソース画面ウィンドウ設定エラー
3 3 デスティネーション画面ウィンドウ設定エラー
4 9 ソース画面が不定画面
- エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

2値画像に対し、4連結で縮退化処理を行います。

本コマンドは、ソース画面が濃淡画像の場合も2値画像として処理します。最上位ビットが1の場合は白、0の場合は黒として処理します。結果画像は0と255の2値画像になります。

指定領域の周辺(内側2画素の領域)は、0を出力します。

処理画面によっては1回の縮退化処理で物体外周の1画素分だけしか削れないことがありますので、完全に縮退化するには、複数回処理を行う必要があります。

処理内容は、次ページの『IP_Shrink8()』コマンドを参照して下さい。

2.11.12 2値画像縮退化（8連結）（IP_Shrink8）

□ C言語API

```
int ret = IP_Shrink8(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○
ImgDst	○	△R	○	○	○	×	-	-	◆

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2 OSシステムエラー
3 IMP処理タイムアウト
8 VIN入力タイムアウト
1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
3 0 ソース画面ウィンドウ設定エラー
3 3 デスティネーション画面ウィンドウ設定エラー
4 9 ソース画面が不定画面
- エラーリセットコマンド（ClearIPError）未発行

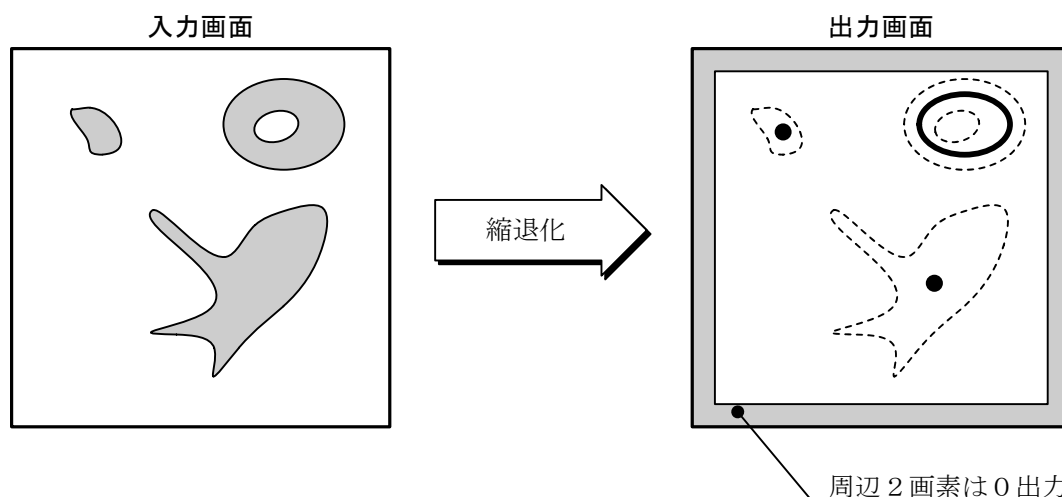
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

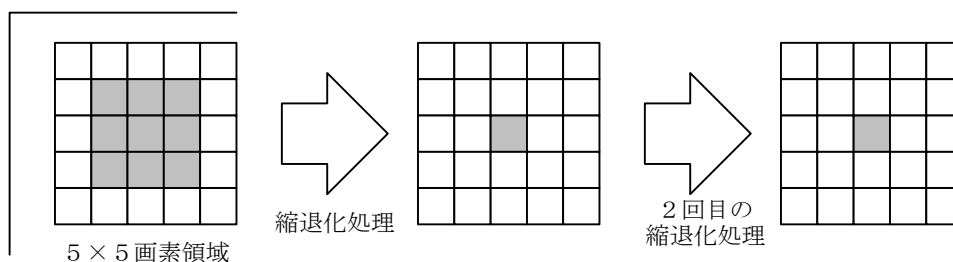
2値画像に対し、8連結で縮退化を行います。

本コマンドは、ソース画面が濃淡画像の場合も2値画像として処理します。最上位ビットが1の場合は白、0の場合は黒として処理します。結果画像は0と255の2値画像になります。

縮退化処理では、入力画面（2値画像）の指定領域内の全画素に対して縮退化処理を行い、結果を出力します。また、指定領域の周辺(内側2画素の領域)は、0を出力します。



1回の画像処理で、 5×5 画素領域で局所的な縮退処理を行います。処理画面によっては1回の縮退処理で物体外周の1画素分だけしか削れないことがありますので、完全に縮退するには、複数回処理を行う必要があります。また、縮退処理は細線化処理の端点を縮退する処理です。物体数の解析等に使用できます。



[縮退処理を完全に行う場合のプログラム例]

2値画像の管理が、物体：255、背景：0で管理している為、物体の面積の変化がなくなったことをヒストグラム処理を用いて検知する（変化しなくなるまで処理する）。

```

IMGID          ImgSrc, ImgDst;
long           Tbl[256];
IPGOFeatureTbl FtrTbl;
int            opt, count;

opt  = 0;
count = 0;           // 2値データの画素数カウント用

for(;;){
    IP_Shrink8( ImgSrc, ImgDst );
    ChangeImgDataType( ImgDst, (enum DataType)UNSIGN8_DATA );
    IP_Histogram( ImgDst, Tbl, &FtrTbl, opt );
    if( count == Tbl[255] )
        break;
    count = Tbl[255];
    IP_Copy( ImgDst, ImgSrc );
}

```

2.12 コンボリューション

2.12.1 平滑化（IP_SmoothFLT）

□ C言語API

```
int ret = IP_SmoothFLT(IMGID ImgSrc, IMGID ImgDst, int scale, int *COEFF);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	scale	ダウンシフト量（0～15）
int	*COEFF	係数テーブルを格納したアドレス テーブルはint型で9個の容量を確保して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	—

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
22	ダウンシフト量範囲外
23	係数テーブルを格納したアドレスが不正
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

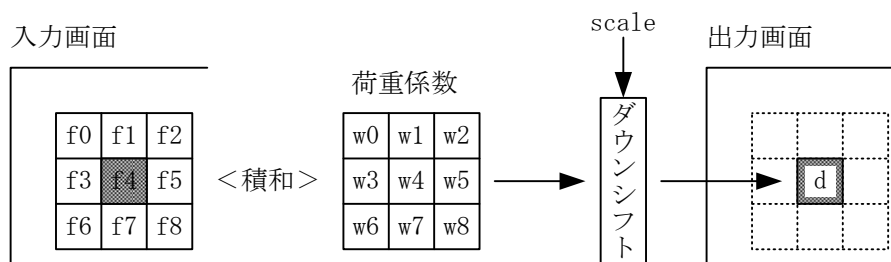
□ 機能

濃淡画像の平滑化を行います。平滑化では、入力画面の指定領域内の全画素に対して、注目画素を中心とした3×3近傍の局所領域で与えられた荷重係数との積和演算を行います。また、対象領域の周辺（3×3近傍に領域外が含まれる部分）処理の場合は中央画素値を出力します。

画面データタイプが符号付8ビットの場合は、ダウンシフトは算術シフトになります。

画面データタイプが2値の場合は、画素データの最上位ビット（1または0）を演算に使用します。例えば、画素値255は1として処理されます。

また、ダウンシフトは四捨五入を行い、ソース画面データタイプが符号付8ビットの場合-0.5は0になります。ダウンシフトした結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付8ビットの場合は127／-128、符号なし8ビットの場合は255／0を出力します。



$$d = (f0 \times w0 + f1 \times w1 + f2 \times w2 + f3 \times w3 + f4 \times w4 + f5 \times w5 + f6 \times w6 + f7 \times w7 + f8 \times w8) \gg \text{scale}$$

係数テーブルのフォーマット及び荷重係数の例を以下に示します。

荷重係数

#0	#1	#2
#3	#4	#5
#6	#7	#8

荷重係数の例

例1

16	16	0
16	16	0
0	0	0

例2

7	7	7
7	8	7
7	7	7

	31	9	8	7	0
COEFF [0]					係数 # 0
[1]					# 1
[2]					# 2
[3]					# 3
[4]					# 4
[5]					# 5
[6]					# 6
[7]					# 7
[8]					係数 # 8

符号ビット

例3

5	8	5
8	12	8
5	8	5

例4

5	5	5
5	20	5
5	5	5

本例の場合、ダウシフト量は6です。

2.12.2 濃淡画像輪郭強調 (IP_EdgeFLT)

□ C言語API

```
int ret = IP_EdgeFLT(IMGID ImgSrc, IMGID ImgDst, int scale, int *COEFF);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	scale	ダウンシフト量 (0～15)
int	*COEFF	係数テーブルを格納したアドレス テーブルはint型で9個の容量を確保して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
22	ダウンシフト量範囲外
23	係数テーブルを格納したアドレスが不正
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

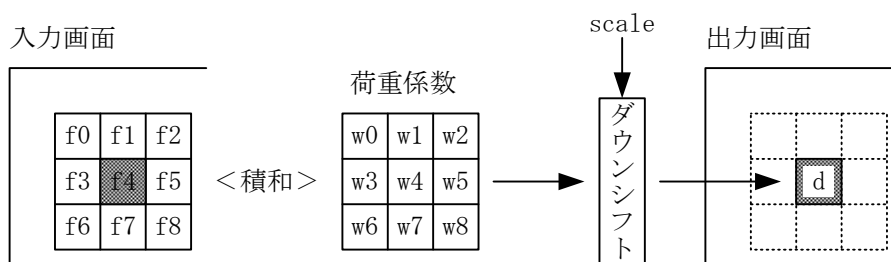
□ 機能

濃淡画像の輪郭強調を行います。輪郭強調では、入力画面の指定領域内の全画素に対して、注目画素を中心とした3×3近傍の局所領域で与えられた荷重係数との積和演算を行います。また、対象領域の周辺 (3×3近傍に領域外が含まれる部分) 処理の場合は0を出力します。

画面データタイプが符号付8ビットの場合は、ダウンシフトは算術シフトになります。

画面データタイプが2値の場合は、画素データの最上位ビット (1または0) を演算に使用します。例えば、画素値255は1として処理されます。

また、ダウンシフトは四捨五入を行い、ソース画面データタイプが符号付8ビットの場合-0.5は0になります。ダウンシフトした結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付8ビットの場合は127／-128、符号なし8ビットの場合は255／0を出力します。



$$d = (f0 \times w0 + f1 \times w1 + f2 \times w2 + f3 \times w3 + f4 \times w4 + f5 \times w5 + f6 \times w6 + f7 \times w7 + f8 \times w8) \gg \text{scale}$$

荷重係数の例を以下に示します。また、係数テーブルのフォーマットは、『IP_SmoothFLT()』コマンドを参照して下さい。

荷重係数の例

例1	-1	0	-1
	-1	0	-1
	-1	0	-1

例2	-1	-1	-1
	0	0	0
	-1	-1	-1

本例の場合、ダウンシフト量は0です。

2.12.3 濃淡画像輪郭強調（絶対値）（IP_EdgeFLTAbs）

□ C言語API

```
int ret = IP_EdgeFLTAbs(IMGID ImgSrc, IMGID ImgDst, int scale, int *COEFF);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	scale	ダウンシフト量（0～15）
int	*COEFF	係数テーブルを格納したアドレス テーブルはint型で9個の容量を確保して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
22	ダウンシフト量範囲外
23	係数テーブルを格納したアドレスが不正
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

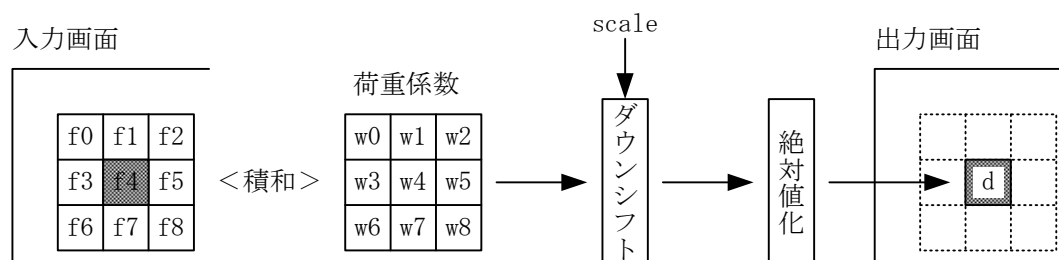
□ 機能

濃淡画像の輪郭強調を行います。輪郭強調（絶対値）では、入力画面の指定領域内の全画素に対して、注目画素を中心とした 3×3 近傍の局所領域で与えられた荷重係数との積和演算を行い、結果を絶対値化します。また、対象領域の周辺（ 3×3 近傍に領域外が含まれる部分）処理の場合は 0 を出力します。

画面データタイプが符号付 8 ビットの場合は、ダウンシフトは算術シフトになります。

画面データタイプが 2 値の場合は、画素データの最上位ビット（1 または 0）を演算に使用します。例えば、画素値 2 5 5 は 1 として処理されます。

また、ダウンシフトは四捨五入で行います。ダウンシフトした結果がオーバーフローしたとき、システムデータタイプが符号付 8 ビットの場合は 1 2 7、符号なし 8 ビットの場合は 2 5 5 を出力します。



$$d = | (f0 \times w0 + f1 \times w1 + f2 \times w2 + f3 \times w3 + f4 \times w4 + f5 \times w5 + f6 \times w6 + f7 \times w7 + f8 \times w8) \gg \text{scale} |$$

係数テーブルのフォーマットは『IP_SmoothFLT()』コマンドを、荷重係数の例は『IP_EdgeFLT()』コマンドを参照して下さい。

2.12.4 ラプラシアン（4連結）（IP_Lapl4FLT）

□ C言語API

```
int ret = IP_Lapl4FLT(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2 OSシステムエラー
3 IMP処理タイムアウト
8 VIN入力タイムアウト
1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
3 0 ソース画面ウィンドウ設定エラー
3 3 デスティネーション画面ウィンドウ設定エラー
4 9 ソース画面が不定画面
- エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

濃淡画像において、下記に示すフィルタで2次微分を行います。対象領域の周辺（3×3近傍に領域外が含まれる部分）処理の場合は0を出力します。

ソース画面データタイプが2値の場合は、画素データの最上位ビット（1または0）を演算に使用します。例えば、画素値255は1として処理されます。

結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付8ビットの場合は127／-128、符号なし8ビットの場合は255／0を出力します。

0	-1	0
-1	4	-1
0	-1	0

2.12.5 ラプラシアン（8連結）（IP_Lapl8FLT）

□ C言語API

```
int ret = IP_Lapl8FLT(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2 OSシステムエラー
3 IMP処理タイムアウト
8 VIN入力タイムアウト
1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
 （不当画面番号エラー）
3 0 ソース画面ウィンドウ設定エラー
3 3 デスティネーション画面ウィンドウ設定エラー
4 9 ソース画面が不定画面
- エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

濃淡画像において、下記に示すフィルタで2次微分を行います。対象領域の周辺（3×3近傍に領域外が含まれる部分）処理の場合は0を出力します。

ソース画面データタイプが2値の場合は、画素データの最上位ビット（1または0）を演算に使用します。例えば、画素値255は1として処理されます。

結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付8ビットの場合は127／-128、符号なし8ビットの場合は255／0を出力します。

-1	-1	-1
-1	8	-1
-1	-1	-1

2.12.6 ラプラシアン（4連結・絶対値）（IP_Lapl4FLTAbs）

□ C言語API

```
int ret = IP_Lapl4FLTAbs(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
3 0	ソース画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 9	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

濃淡画像において、下記に示すフィルタで2次微分を行い、絶対値を出力します。対象領域の周辺（3×3近傍に領域外が含まれる部分）処理の場合は0を出力します。

ソース画面データタイプが2値の場合は、画素データの最上位ビット（1または0）を演算に使用します。例えば、画素値255は1として処理されます。

結果がオーバーフローしたとき、システムデータタイプが符号付8ビットの場合は127、符号なし8ビットの場合は255を出力します。

0	-1	0
-1	4	-1
0	-1	0

2.12.7 ラプラシアン（8連結・絶対値）（IP_Lapl8FLTAbs）

□ C言語API

```
int ret = IP_Lapl8FLTAbs(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
3 0	ソース画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 9	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

濃淡画像において、下記に示すフィルタで2次微分を行い、絶対値を出力します。対象領域の周辺（3×3近傍に領域外が含まれる部分）処理の場合は0を出力します。

ソース画面データタイプが2値の場合は、画素データの最上位ビット（1または0）を演算に使用します。例えば、画素値255は1として処理されます。

結果がオーバーフローしたとき、システムデータタイプが符号付8ビットの場合は127、符号なし8ビットの場合は255を出力します。

-1	-1	-1
-1	8	-1
-1	-1	-1

2.12.8 ラインフィルタ (IP_LineFLT)

□ C言語API

```
int ret = IP_LineFLT(IMGID ImgSrc, IMGID ImgDst, int scale, int *COEFF);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	scale	ダウンシフト量 (0～15)
int	*COEFF	係数テーブルを格納したアドレス テーブルはint型で9個の容量を確保して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
22	ダウンシフト量範囲外
23	係数テーブルを格納したアドレスが不正
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

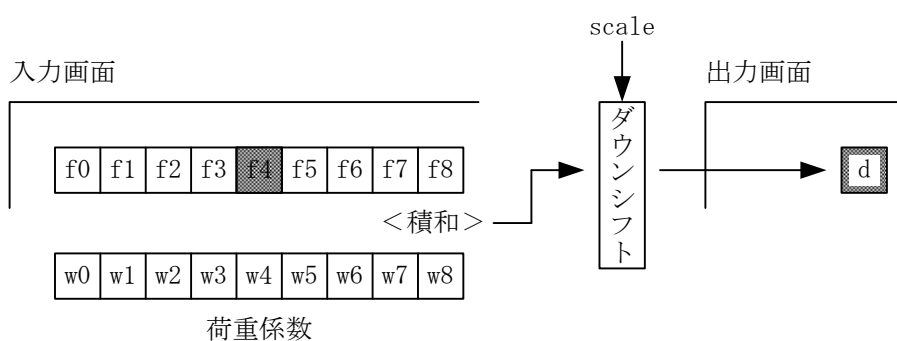
□ 機能

ソース画面に対し、ユーザの登録パターンでラインフィルタ処理を行い、デスティネーション画面に出力します。ラインフィルタでは、入力画面の指定領域内の全画素に対して、注目画素を中心とした1×9近傍の局所領域で与えられた荷重係数との積和演算を行います。また、対象領域の周辺(1×9近傍に領域外が含まれる部分)処理の場合は不定になります。

画面データタイプが符号付8ビットの場合は、ダウンシフトは算術シフトになります。

画面データタイプが2値の場合は、画素データの最上位ビット(1または0)を演算に使用します。例えば、画素値255は1として処理されます。

また、ダウンシフトは四捨五入を行い、ソース画面データタイプが符号付8ビットの場合-0.5は0になります。ダウンシフトした結果がオーバー／アンダーフローしたとき、システムデータタイプが符号付8ビットの場合は127／-128、符号なし8ビットの場合は255／0を出力します。



$$d = (f0 \times w0 + f1 \times w1 + f2 \times w2 + f3 \times w3 + f4 \times w4 + f5 \times w5 + f6 \times w6 + f7 \times w7 + f8 \times w8) \gg \text{scale}$$

係数テーブルのフォーマット及び荷重係数の例を、下記に示します。

荷重係数

#0	#1	#2	#3	#4	#5	#6	#7	#8

荷重係数の例（横方向の平滑化）

7	7	7	7	7	7	7	7	7

本例の場合、ダウシフト量は6です。

	31	9	8	7	0
COEFF [0]	_____				係数 # 0
[1]	_____				# 1
[2]	_____				# 2
[3]	_____				# 3
[4]	_____				# 4
[5]	_____				# 5
[6]	_____				# 6
[7]	_____				▼ # 7
[8]	_____				係数 # 8

符号ビット

2.12.9 ラインフィルタ（絶対値）（IP_LineFLTAbs）

□ C言語API

```
int ret = IP_LineFLTAbs(IMGID ImgSrc, IMGID ImgDst, int scale, int *COEFF);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	scale	ダウンシフト量（0～15）
int	*COEFF	係数テーブルを格納したアドレス テーブルはint型で9個の容量を確保して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
22	ダウンシフト量範囲外
23	係数テーブルを格納したアドレスが不正
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

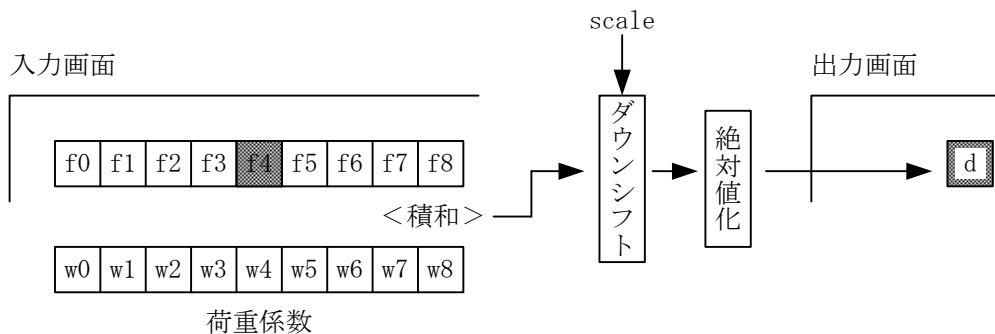
□ 機能

ソース画面に対し、ユーザの登録パターンでラインフィルタ絶対値処理を行い、デスティネーション画面に出力します。ラインフィルタでは、入力画面の指定領域内の全画素に対して、注目画素を中心とした 1×9 近傍の局所領域で与えられた荷重係数との積和演算を行い、絶対値化します。また、対象領域の周辺（ 1×9 近傍に領域外が含まれる部分）処理の場合は不定になります。

画面データタイプが符号付 8 ビットの場合は、ダウンシフトは算術シフトになります。

画面データタイプが 2 値の場合は、画素データの最上位ビット（1 または 0）を演算に使用します。例えば、画素値 2 5 5 は 1 として処理されます。

また、ダウンシフトは四捨五入で行います。ダウンシフトした結果がオーバーフローしたとき、システムデータタイプが符号付 8 ビットの場合は 1 2 7、符号なし 8 ビットの場合は 2 5 5 を出力します。



$$d = | (f0 \times w0 + f1 \times w1 + f2 \times w2 + f3 \times w3 + f4 \times w4 + f5 \times w5 + f6 \times w6 + f7 \times w7 + f8 \times w8) \gg \text{scale} |$$

係数テーブルのフォーマット及び荷重係数の例は、『IP_LineFLT()』コマンドを参照して下さい。

2.13 ミニ／マックスフィルタ

2.13.1 局所最小値フィルタ（IP_MinFLT）

□ C言語API

```
int ret = IP_MinFLT(IMGID ImgSrc, IMGID ImgDst, int calptn);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	calptn	演算パターン（下位9ビット有効）

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド（ClearIPError）未発行

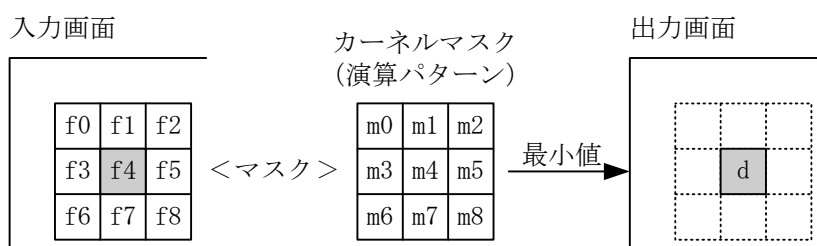
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

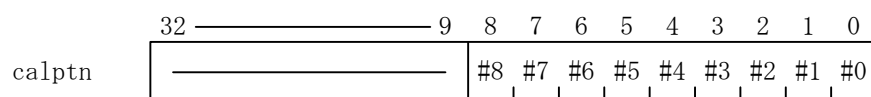
ソース画像に対し、パラメータで指定されたパターンで最小濃度値を見つけ、デスティネーション画面に出力します。局所最小値フィルタでは、入力画面の指定領域内の全画素に対して、注目画素を中心とした3×3近傍の局所領域を与えられた演算パターンでマスク処理し、有効となった画素の中から最小の画素を出力します。指定領域の周辺（内側1画素の領域）は、ソース画面の画素をそのまま出力します。

マスクされた画素の値は最大値として扱われます。calptn=0を指定した場合は、周辺部を除いてシステムデータタイプが符号なし8ビットの場合には0、システムデータタイプが符号付8ビットの場合には-128が出力されます。周辺部については周辺処理が優先します。

また、ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します（255は-1として処理します）。



パラメータcalptnのフォーマットを以下に示します。



0 : 演算マスク

1 : 演算イネーブル

#0	#1	#2
#3	#4	#5
#6	#7	#8

← 演算パターン

2.13.2 局所最小値フィルタ（4連結）（IP_MinFLT4）

□ C言語API

```
int ret = IP_MinFLT4(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

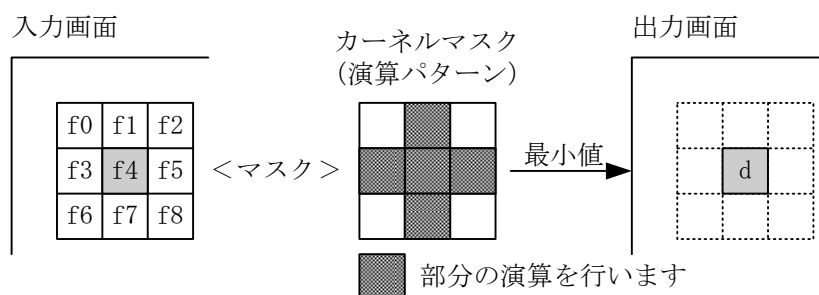
2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画像に対し、4連結で最小濃度値を見つけ、デスティネーション画面に出力します。局所最小値フィルタでは、入力画面の指定領域内の全画素に対して、注目画素を中心とした3×3近傍の局所領域を4連結の演算パターンでマスク処理し、有効となった画素の中から最小の画素を出力します。指定領域の周辺(内側1画素の領域)は、ソース画面の画素をそのまま出力します。

また、ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します（255は-1として処理します）。



2.13.3 局所最小値フィルタ（8連結）（IP_MinFLT8）

□ C言語API

```
int ret = IP_MinFLT8(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

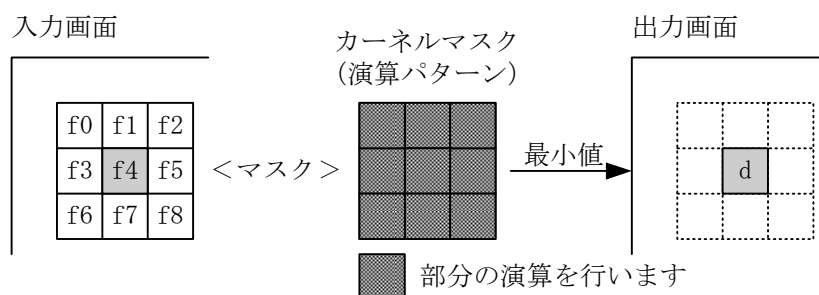
2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画像に対し、8連結で最小濃度値を見つけ、デスティネーション画面に出力します。局所最小値フィルタでは、入力画面の指定領域内の全画素に対して、注目画素を中心とした3×3近傍の局所領域を8連結の演算パターンでマスク処理し、有効となった画素の中から最小の画素を出力します。指定領域の周辺(内側1画素の領域)は、ソース画面の画素をそのまま出力します。

また、ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します（255は-1として処理します）。



2.13.4 局所最大値フィルタ (IP_MaxFLT)

□ C言語API

```
int ret = IP_MaxFLT(IMGID ImgSrc, IMGID ImgDst, int calptn);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	calptn	演算パターン (下位9ビット有効)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

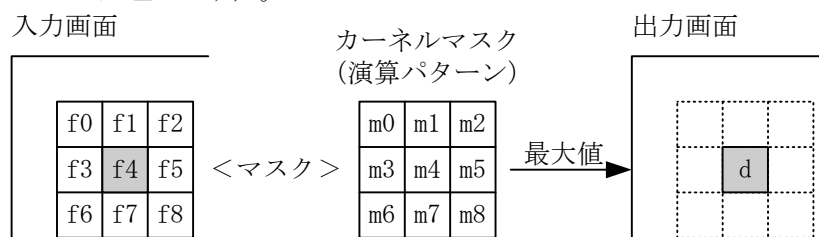
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画像に対し、パラメータで指定されたパターンで最大濃度値を見つけ、デスティネーション画面に出力します。局所最大値フィルタでは、入力画面の指定領域内の全画素に対して、注目画素を中心とした3×3近傍の局所領域を与えられた演算パターンでマスク処理し、有効となった画素の中から最大の画素を出力します。指定領域の周辺(内側1画素の領域)は、ソース画面の画素をそのまま出力します。

マスクされた画素の値は最小値として扱われます。calptn=0を指定した場合は、周辺部を除いてシステムデータタイプが符号なし8ビットの場合には0、システムデータタイプが符号付8ビットの場合には-128が出力されます。周辺部については周辺処理が優先します。

また、ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します(255は-1として処理します)。



パラメータcalptnのフォーマットは、『IP_MinFLT()』コマンドを参照して下さい。

2.13.5 局所最大値フィルタ（4連結）（IP_MaxFLT4）

□ C言語API

```
int ret = IP_MaxFLT4(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

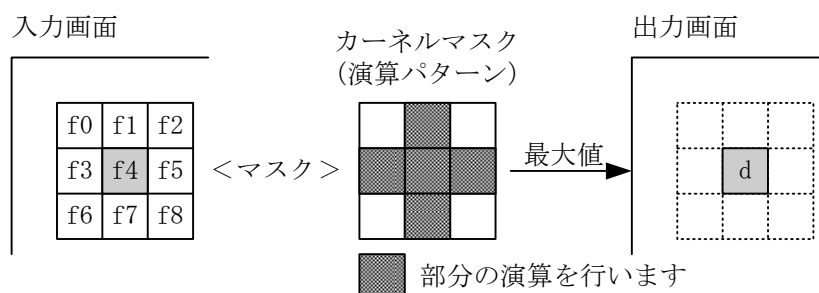
2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画像に対し、4連結で最大濃度値を見つけ、デスティネーション画面に出力します。局所最大値フィルタでは、入力画面の指定領域内の全画素に対して、注目画素を中心とした3×3近傍の局所領域を4連結の演算パターンでマスク処理し、有効となった画素の中から最大の画素を出力します。指定領域の周辺(内側1画素の領域)は、ソース画面の画素をそのまま出力します。

また、ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します（255は-1として処理します）。



2.13.6 局所最大値フィルタ（8連結）（IP_MaxFLT8）

□ C言語API

```
int ret = IP_MaxFLT8(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

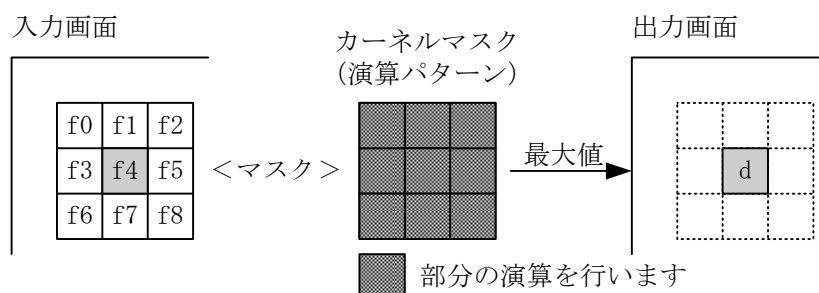
2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画像に対し、8連結で最大濃度値を見つけ、デスティネーション画面に出力します。局所最大値フィルタでは、入力画面の指定領域内の全画素に対して、注目画素を中心とした3×3近傍の局所領域を8連結の演算パターンでマスク処理し、有効となった画素の中から最大の画素を出力します。指定領域の周辺(内側1画素の領域)は、ソース画面の画素をそのまま出力します。

また、ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します（255は-1として処理します）。



2.13.7 ライン局所最小値フィルタ（IP_LineMinFLT）

□ C言語API

```
int ret = IP_LineMinFLT(IMGID ImgSrc, IMGID ImgDst, int calptn);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	calptn	演算パターン（下位9ビット有効）

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド（ClearIPError）未発行

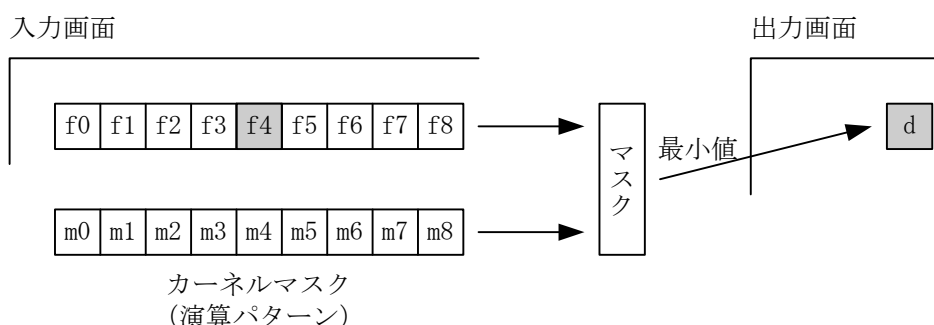
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

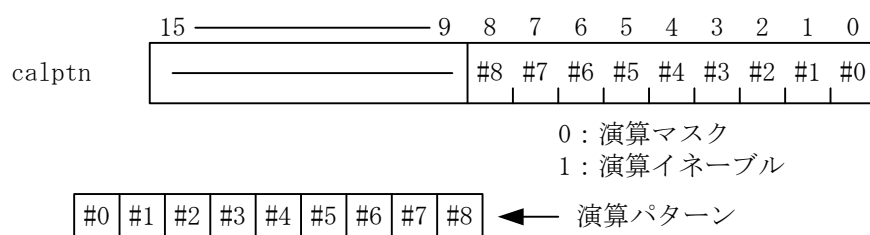
ソース画像に対し、パラメータで指定されたパターンで最小濃度値を見つけ、デスティネーション画面に出力します。ライン局所最小値フィルタでは、入力画面の指定領域内の全画素に対して、注目画素を中心とした1×9近傍の局所領域を与えられた演算パターンでマスク処理し、有効となった画素の中から最小の画素を出力します。指定領域の周辺（内側1画素の領域）は、ソース画面の画素をそのまま出力します。

マスクされた画素の値は最小値として扱われます。calptn=0を指定した場合は、周辺部を除いてシステムデータタイプが符号なし8ビットの場合には0、システムデータタイプが符号付8ビットの場合には-128が出力されます。周辺部については周辺処理が優先します。

また、ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します（255は-1として処理します）。



パラメータcalptnのフォーマットを下記に示します。



2.13.8 ライン局所最大値フィルタ（IP_LineMaxFLT）

□ C言語API

```
int ret = IP_LineMaxFLT(IMGID ImgSrc, IMGID ImgDst, int calptn);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	calptn	演算パターン（下位9ビット有効）

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

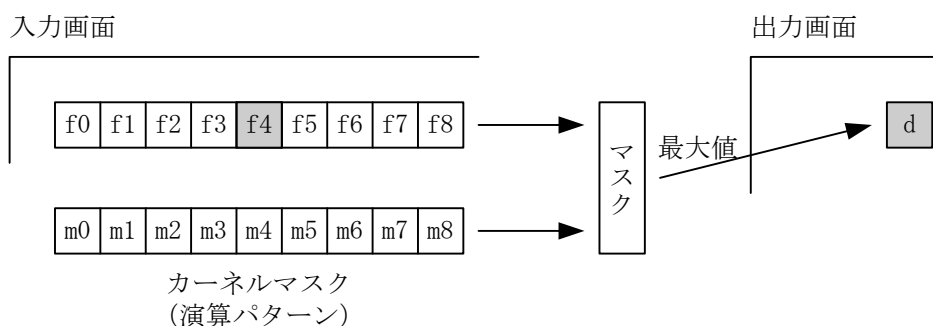
2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画像に対し、パラメータで指定されたパターンで最大濃度値を見つけ、デスティネーション画面に出力します。ライン局所最大値フィルタでは、入力画面の指定領域内の全画素に対して、注目画素を中心とした 1×9 近傍の局所領域を与えられた演算パターンでマスク処理し、有効となった画素の中から最大の画素を出力します。指定領域の周辺(内側1画素の領域)は、ソース画面の画素をそのまま出力します。

マスクされた画素の値は最小値として扱われます。calptn=0を指定した場合は、周辺部を除いてシステムデータタイプが符号なし8ビットの場合には0、システムデータタイプが符号付8ビットの場合には-128が出力されます。周辺部については周辺処理が優先します。また、ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します(255は-1として処理します)。



パラメータcalptnのフォーマットは、『IP_LineMinFLT()』コマンドを参照して下さい。

2.14 ランクフィルタ

2.14.1 局所ランクフィルタ（IP_RankFLT）

□ C言語API

```
int ret = IP_RankFLT(IMGID ImgSrc, IMGID ImgDst, int calptn, int rank);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	calptn	演算パターン（下位 9 ビット有効）
int	rank	ランク指定（0～8）

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

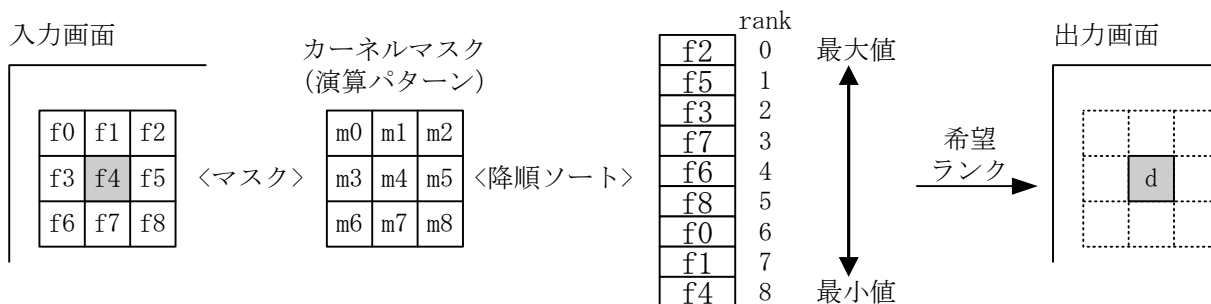
□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
2 3	ランク指定範囲外
3 0	ソース画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 9	ソース画面が不定画面
-	エラーリセットコマンド（ClearIPError）未発行

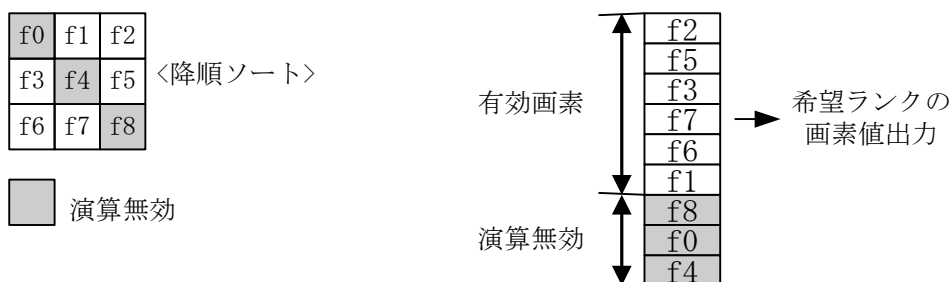
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

機能

ソース画面に対し、パラメータで指定された有効パターンで希望のランクの濃度値を見つけ、デスティネーション画面に出力します。局所ランクフィルタでは、入力画面の指定領域内の全画素に対して、注目画素を中心とした 3×3 近傍の局所領域を与えられた演算パターンでマスク処理し、有効となった画素の値を降順ソートして、希望のランクの数値を抽出し出力します。指定領域の周辺(内側1画素の領域)は、ソース画面の画素をそのまま出力します。また、ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します(255は-1として処理します)。

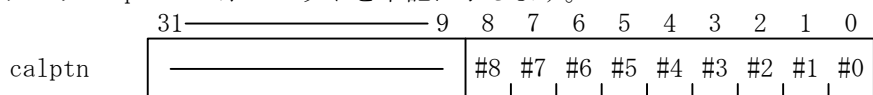


注) マスクされた画素の値は最小値として扱われます。したがって、ランク値を指定するときは、最大値(ランク0)からの順番を指定して下さい。calptn=0を指定した場合や指定されたランクがcalptnマスク後の有効画素の範囲を超えている場合は、周辺部を除いてシステムデータタイプが符号なし8ビットの場合には0、システムデータタイプが符号付8ビットの場合には-1 2 8が出力されます。周辺部については周辺処理が優先します。



最大値出力の場合 常にランク0 (rank = 0)
 中間値の場合 有効にした画素数の半分の数値
 (例: 有効画素数6画素であれば、ランク3 (rank = 3))
 最小値の場合 有効にした画素数
 (例: 有効画素数6画素であれば、ランク5 (rank = 5))

パラメータcalptnのフォーマットを下記に示します。



0: 演算無効 (-255を代入)
 1: 演算イネーブル (処理対象)

#0	#1	#2
#3	#4	#5
#6	#7	#8

カーネルマスク
(画面との対応)

2.14.2 局所ランクフィルタ(4連結) (IP_Rank4FLT)

□ C言語API

```
int ret = IP_Rank4FLT(IMGID ImgSrc, IMGID ImgDst, int rank);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	rank	ランク指定 (0 ~ 4)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

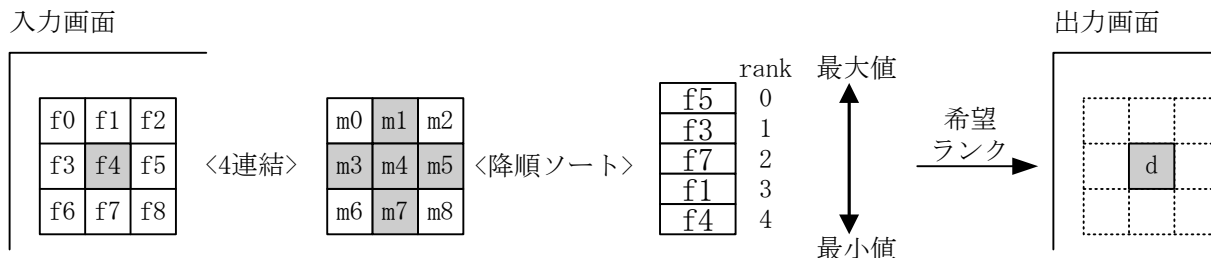
2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 2	ランク指定範囲外
3 0	ソース画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 9	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面に対し、4連結パターンで希望のランクの濃度値を見つけ、デスティネーション画面に出力します。局所ランクフィルタ（4連結）では、入力画面の指定領域内の全画素に対して、注目画素を中心とした3×3近傍の4連結局所領域内で、有効となった画素の値を降順ソートして、希望のランクの数値を抽出し出力します。指定領域の周辺(内側1画素の領域)は、ソース画面の画素をそのまま出力します。

また、ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します（255は-1として処理します）。



最大値出力の場合

ランク 0 (rank = 0)

中間値の場合

ランク 2 (rank = 2)

最小値の場合

ランク 4 (rank = 4)

2.14.3 局所ランクフィルタ(8連結) (IP_Rank8FLT)

□ C言語API

```
int ret = IP_Rank8FLT(IMGID ImgSrc, IMGID ImgDst, int rank);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	rank	ランク指定 (0 ~ 8)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

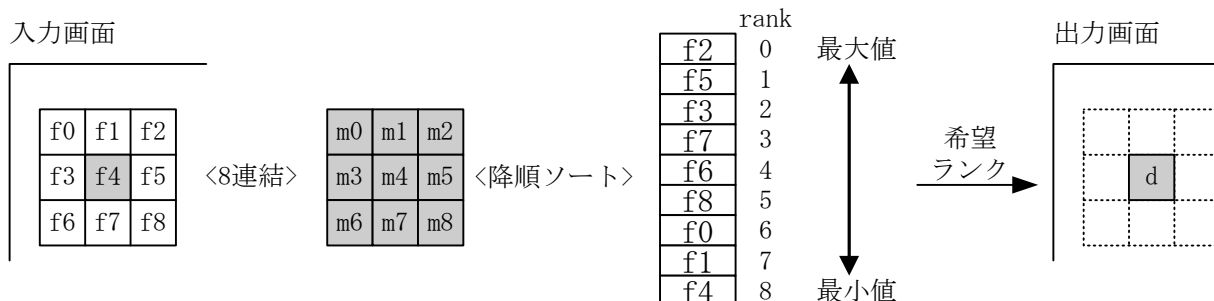
2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 2	ランク指定範囲外
3 0	ソース画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 9	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面に対し、8連結パターンで希望のランクの濃度値を見つけ、デスティネーション画面に出力します。局所ランクフィルタ（8連結）では、入力画面の指定領域内の全画素に対して、注目画素を中心とした3×3近傍の8連結局所領域内で、有効となった画素の値を降順ソートして、希望のランクの数値を抽出し出力します。指定領域の周辺(内側1画素の領域)は、ソース画面の画素をそのまま出力します。

また、ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します（255は-1として処理します）。



最大値出力の場合

中間値の場合

最小値の場合

ランク 0 (rank = 0)

ランク 4 (rank = 4)

ランク 8 (rank = 8)

2.14.4 局所メディアンフィルタ（IP_MedFLT）

□ C言語API

```
int ret = IP_MedFLT(IMGID ImgSrc, IMGID ImgDst, int calptn);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	calptn	演算パターン（下位9ビット有効）

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

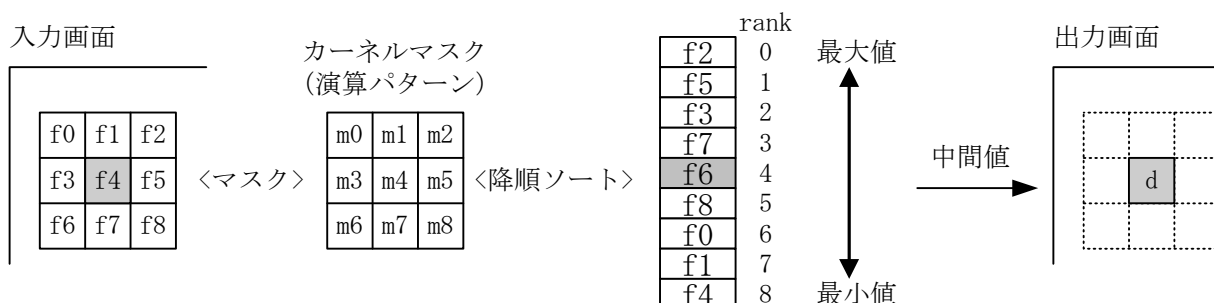
2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

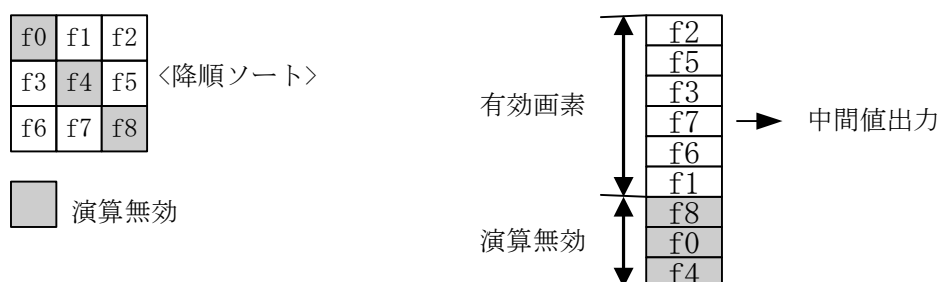
機能

ソース画面に対し、パラメータで指定された有効パターンで中間の濃度値を見つけ、デスティネーション画面に出力します。局所メディアンフィルタでは、入力画面の指定領域内の全画素に対して、注目画素を中心とした 3×3 近傍の局所領域を与えられた演算パターンでマスク処理し、有効となった画素の値の中間値を抽出し出力します。指定領域の周辺(内側1画素の領域)は、ソース画面の画素をそのまま出力します。

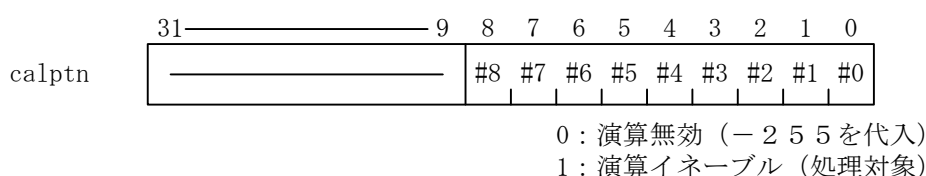
また、ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します(255は-1として処理します)。



注) 局所メディアンフィルタでは、有効画素の中間値を出力します。有効画素が偶数の場合には、以下に示すように最大値(ランク0)から有効画素/2 + 1 番目を出力します。マスクされた画素の値は最小値として扱われます。calptn=0を指定した場合は、周辺部を除いてシステムデータタイプが符号なし8ビットの場合には0、システムデータタイプが符号付8ビットの場合には-128が出力されます。周辺部については周辺処理が優先します。



マスクに使用するパラメータcalptnのフォーマットを下記に示します。



#0	#1	#2
#3	#4	#5
#6	#7	#8

カーネルマスク
(画面との対応)

2.14.5 局所メディアンフィルタ(4連結) (IP_Med4FLT)

□ C言語API

```
int ret = IP_Med4FLT(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

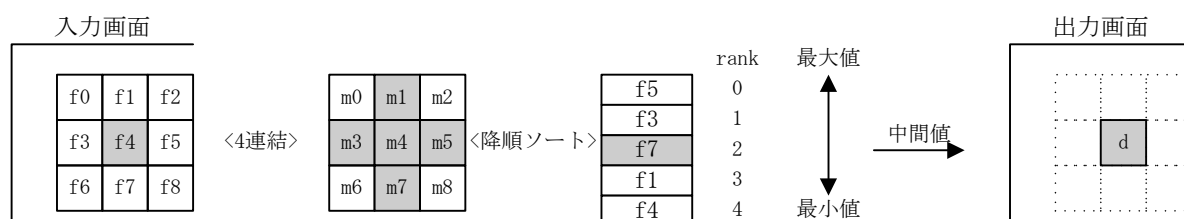
2 OS システムエラー
3 IMP 処理タイムアウト
8 VIN 入力タイムアウト
1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)
1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)
3 0 ソース画面ウィンドウ設定エラー
3 3 デスティネーション画面ウィンドウ設定エラー
4 9 ソース画面が不定画面
- エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面に対し、4連結パターンで中間の濃度値を見つけ、デスティネーション画面に出力します。局所メディアンフィルタ(4連結)では、入力画面の指定領域内の全画素に対して、注目画素を中心とした3×3近傍の4連結局所領域内で、有効となった画素の値の中間値を抽出し出力します。指定領域の周辺(内側1画素の領域)は、ソース画面の画素をそのまま出力します。

また、ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します(255は-1として処理します)。



2.14.6 局所メディアンフィルタ(8連結) (IP_Med8FLT)

□ C言語API

```
int ret = IP_Med8FLT(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

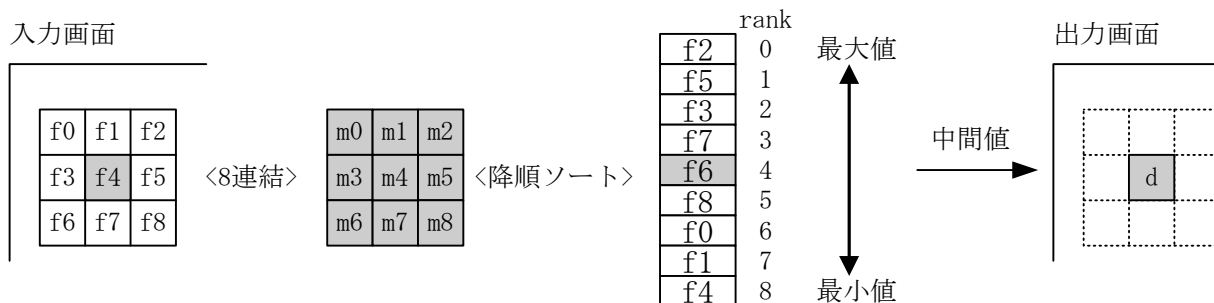
2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面に対し、8連結パターンで中間の濃度値を見つけ、デスティネーション画面に出力します。局所メディアンフィルタ(8連結)では、入力画面の指定領域内の全画素に対して、注目画素を中心とした3×3近傍の8連結局所領域内で、有効となった画素の値の中間値を抽出し出力します。指定領域の周辺(内側1画素の領域)は、ソース画面の画素をそのまま出力します。

また、ソース画面データタイプが2値の場合は、符号付8ビットに変換して処理します(255は-1として処理します)。



2.15 ラベリング

2.15.1 ラベリング(4連結) (IP_Label4)

□ C言語API

```
int ret = IP_Label4(IMGID ImgSrc, IMGID ImgDst, enum IP_Label_opt opt);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

enum IP_Label_optopt ラベリング処理オプション

LABEL_OBJ	(0)	白を物体としてラベリング
LABEL_BKG	(1)	黒を物体としてラベリング

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B2	△B2	○
ImgDst	○	△R	○	○	○	×	◆	-	-

□ リターンパラメータ

int ret ラベル数 (0 ~ 255)
 異常終了 (-1)
 ラベルオーバーフロー (0 x 7 F F F)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 2	オプション設定値範囲外
3 0	ソース画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 8	動的メモリ確保エラー
4 9	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ラベリングとは、つながっている全ての画素（連結成分）に同じラベル（番号）を付ける処理です。本コマンドは、ソース画面（2 値）に対し、4 連結でラベリング処理を行い、デスティネーション画面（濃淡）に出力します。

ソース画面に濃淡画像を指定した場合は、濃淡画像を 2 値画像として処理します。画素データの最上位ビットが 1 の場合は白、0 の場合は黒として処理します。

ラベリング処理は、仮ラベル付け、合流対解析、真ラベル付けの 3 ステップで行われます。仮ラベル付けでは同じ物体を異なるラベルとして認識するため、次ステップの合流対（同じ物体であることを示す仮ラベル対）解析で同じ物体に対して複数の仮ラベル番号が付いていないかを調べ、この結果をもとに真ラベル付けを行います。

ラベル付けの最大数は 255 個です。ラベルオーバーフローは、ラベル数が 256 個以上、仮ラベル数が 1023 個以上、合流対が 1024 個以上のいずれかで発生します。

ラベルオーバーフローの場合はコマンドエラーにならないため `ClearIPError()` は必要ありません。引き続き次の処理を実行します。

ラベル付けはラスタスキャンにて行うため、ラベルはラベル付け例の図に示すように左上から順に濃度 (1~255) が付けられます。ラベル以外 (背景色) はラベリング処理オプションにかかわらず常に黒になります。

連結成分を判定する方法とラベル付け例は、『`IP_Label()`』コマンドを参照してください。

2.15.2 ラベリング(8連結) (IP_Label8)

□ C言語API

```
int ret = IP_Label8(IMGID ImgSrc, IMGID ImgDst, enum IP_Label_opt opt);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

enum IP_Label_optopt ラベリング処理オプション

LABEL_OBJ	(0)	白を物体としてラベリング
LABEL_BKG	(1)	黒を物体としてラベリング

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B2	△B2	○
ImgDst	○	△R	○	○	○	×	◆	-	-

□ リターンパラメータ

int ret ラベル数 (0 ~ 255)
 異常終了 (-1)
 ラベルオーバーフロー (0 x 7 F F F)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 2	オプション設定値範囲外
3 0	ソース画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 8	動的メモリ確保エラー
4 9	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ラベリングとは、つながっている全ての画素（連結成分）に同じラベル（番号）を付ける処理です。本コマンドは、ソース画面（2値）に対し、8連結でラベリング処理を行い、デスティネーション画面（濃淡）に出力します。

ソース画面に濃淡画像を指定した場合は、濃淡画像を2値画像として処理します。画素データの最上位ビットが1の場合は白、0の場合は黒として処理します。

ラベリング処理は、仮ラベル付け、合流対解析、真ラベル付けの3ステップで行われます。仮ラベル付けでは同じ物体を異なるラベルとして認識するため、次ステップの合流対（同じ物体であることを示す仮ラベル対）解析で同じ物体に対して複数の仮ラベル番号が付いていないかを調べ、この結果をもとに真ラベル付けを行います。

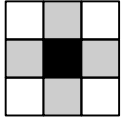
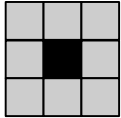
ラベル付けの最大数は255個です。ラベルオーバーフローは、ラベル数が256個以上、仮ラベル数が1023個以上、合流対が1024個以上のいずれかで発生します。

ラベルオーバーフローの場合はコマンドエラーにならないためClearIPError()は必要ありません。引き続き次の処理を実行します。

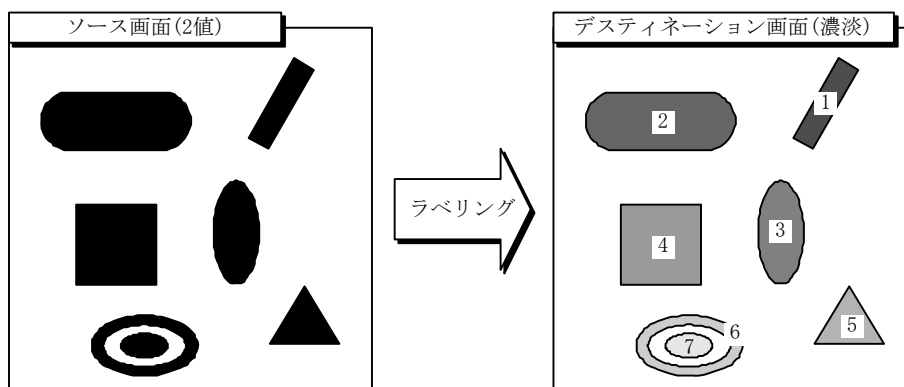
ラベル付けはラスタスキャンにて行うため、ラベルはラベル付け例の図に示すように左上から順に濃度（1～255）が付けられます。ラベル以外（背景色）はラベリング処理オプションにかかわらず常に黒になります。

連結成分を判定する方法とラベル付け例を以下に示します。

<連結成分判定>

4連結		上下左右の何れかが接していれば 連結成分とする
8連結		周辺の何れかでも接していれば 連結成分とする

<ラベル付け例>



注意：ラスタスキャン方向により、左上から右下にかけてラベル付けされない場合があります。

2.15.3 面積フィルタ付きラベリング (4連結) (IP_Label4withAreaFLT)

□ C言語API

```
int ret = IP_Label4withAreaFLT(IMGID ImgSrc, IMGID ImgDst,
                               long thrmin, long thrmax, enum IP_Label_opt opt);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
long	thrmin	面積フィルタしきい値 (最小値 ≥ 1)
long	thrmax	面積フィルタしきい値 (最大値)
enum IP_Label_optopt	ラベリング処理オプション	
	LABEL_OBJ	(0) 白を物体としてラベリング
	LABEL_BKG	(1) 黒を物体としてラベリング

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B2	△B2	○
ImgDst	○	△R	○	○	○	×	◆	-	-

□ リターンパラメータ

int ret ラベル数 (0 ~ 255)
 異常終了 (-1)
 ラベルオーバーフロー (0 x 7 F F F)

□ エラーコード

2 OSシステムエラー
 3 IMP処理タイムアウト
 8 VIN入力タイムアウト
 1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
 (不当画面番号エラー)
 1 3 デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
 (不当画面番号エラー)
 2 2、2 3 面積フィルタしきい値範囲設定エラー
 2 4 オプション設定値範囲外
 3 0 ソース画面ウィンドウ設定エラー
 3 3 デスティネーション画面ウィンドウ設定エラー
 4 8 動的メモリ確保エラー
 4 9 ソース画面が不定画面
 - エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面（2値）に対し、4連結でラベリング処理を行い、指定しきい値で面積フィルタ処理後に再度ラベル付けを行い、デスティネーション画面（濃淡）に出力します。

ソース画面に濃淡画像を指定した場合は、濃淡画像を2値画像として処理します。画素データの最上位ビットが1の場合は白、0の場合は黒として処理します。

ラベリング処理は、仮ラベル付け、合流対解析、真ラベル付けの3ステップで行われます。仮ラベル付けでは同じ物体を異なるラベルとして認識するため、次ステップの合流対（同じ物体であることを示す仮ラベル対）解析で同じ物体に対して複数の仮ラベル番号が付いていないかを調べ、この結果をもとに真ラベル付けを行います。

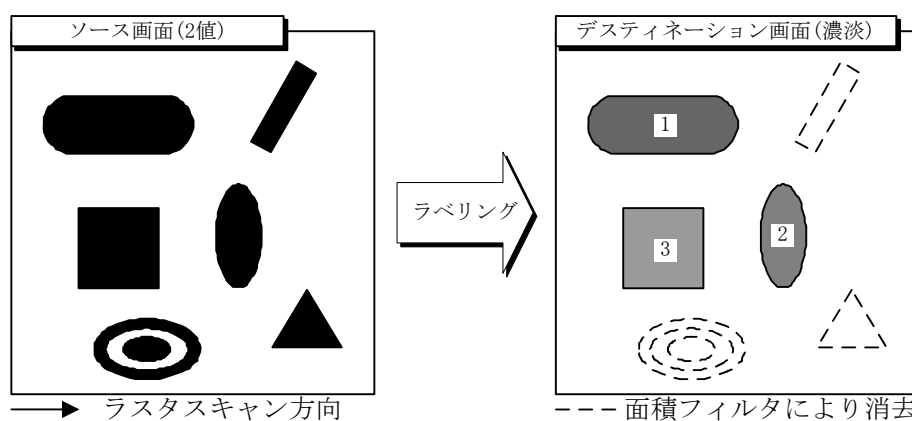
ラベル付けの最大数は255個です。ラベルオーバーフローは、ラベル数が256個以上、仮ラベル数が1023個以上、合流対が1024個以上のいずれかで発生します。

ラベルオーバーフローの場合はコマンドエラーにならないためClearIPError()は必要ありません。引き続き次の処理を実行します。

面積フィルタ処理は、合流対解析の段階で行いますが、仮ラベル付けで仮ラベルが1023を超えるとラベルオーバーフローとして処理します。したがって、面積フィルタの設定で全てのラベルオーバーフローを回避することはできません。

ラベル付けはラスタスキャンにて行うため、図に示すように $\text{thrmin} \leq \text{面積} \leq \text{thrmax}$ のラベルに対し、左上から順に番号が付けられます。

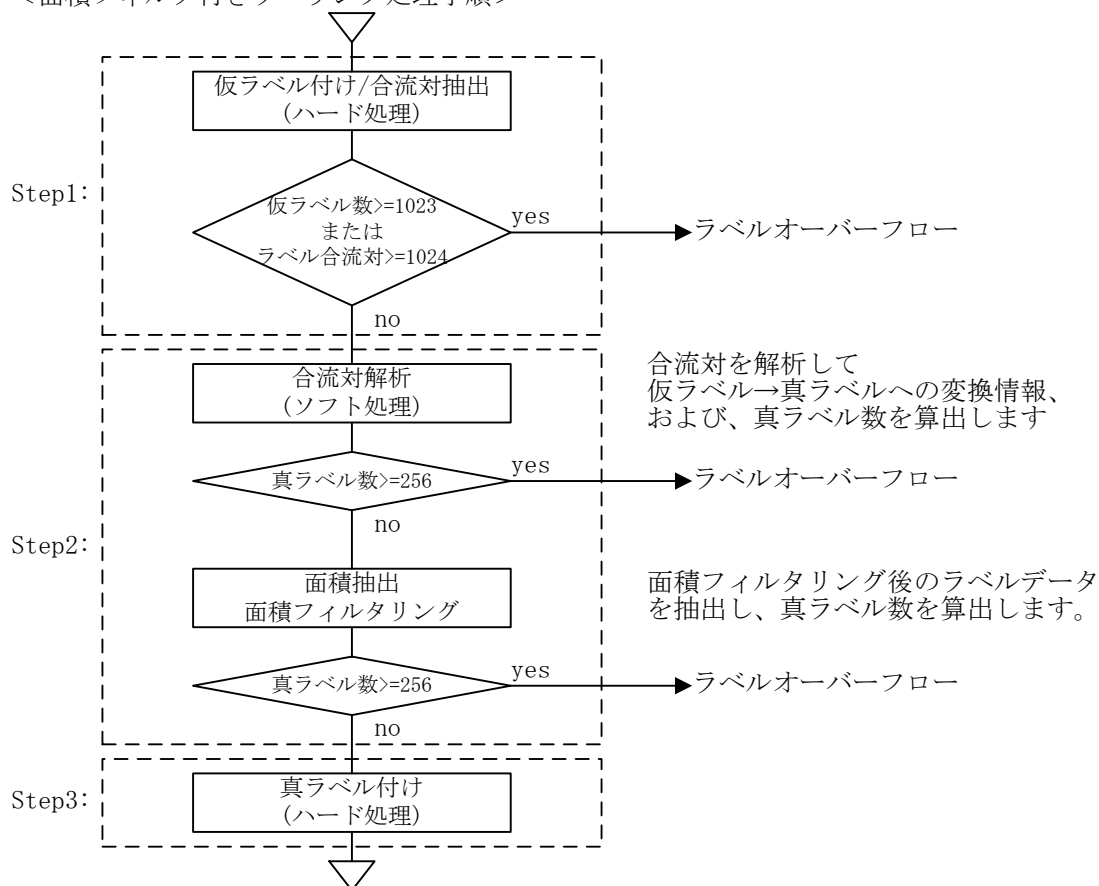
<ラベル付け例>



注意：仮ラベル付け等の状況により、左上から右下にかけてラベル付けされない場合があります。

以下に面積フィルタ付きラベリング処理手順を示します。

<面積フィルタ付きラベリング処理手順>



2.15.4 面積フィルタ付きラベリング(8連結) (IP_Label8withAreaFLT)

□ C言語API

```
int ret = IP_Label8withAreaFLT(IMGID ImgSrc, IMGID ImgDst,
                                long thrmin, long thrmax, enum IP_Label_opt opt);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
long	thrmin	面積フィルタしきい値 (最小値 ≥ 1)
long	thrmax	面積フィルタしきい値 (最大値)
enum IP_Label_optopt	ラベリング処理オプション	
	LABEL_OBJ	(0) 白を物体としてラベリング
	LABEL_BKG	(1) 黒を物体としてラベリング

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B2	△B2	○
ImgDst	○	△R	○	○	○	×	◆	-	-

□ リターンパラメータ

int ret ラベル数 (0 ~ 255)
 異常終了 (-1)
 ラベルオーバーフロー (0 x 7 F F F)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
22、23	面積フィルタしきい値範囲設定エラー
24	オプション設定値範囲外
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
48	動的メモリ確保エラー
49	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面（2値）に対し、8連結でラベリング処理を行い、指定しきい値で面積フィルタ処理後に再度ラベル付けを行い、デスティネーション画面（濃淡）に出力します。

ソース画面に濃淡画像を指定した場合は、濃淡画像を2値画像として処理します。画素データの最上位ビットが1の場合は白、0の場合は黒として処理します。

ラベリング処理は、仮ラベル付け、合流対解析、真ラベル付けの3ステップで行われます。仮ラベル付けでは同じ物体を異なるラベルとして認識するため、次ステップの合流対(同じ物体であることを示す仮ラベル対)解析で同じ物体に対して複数の仮ラベル番号が付いていないかを調べ、この結果をもとに真ラベル付けを行います。

ラベル付けの最大数は255個です。ラベルオーバーフローは、ラベル数が256個以上、仮ラベル数が1023個以上、合流対が1024個以上のいずれかで発生します。

ラベルオーバーフローの場合はコマンドエラーにならないためClearIPError()は必要ありません。引き続き次の処理を実行します。

面積フィルタ処理は、合流対解析の段階で行いますが、仮ラベル付けで仮ラベルが1023を超えるとラベルオーバーフローとして処理します。したがって、面積フィルタの設定で全てのラベルオーバーフローを回避することはできません。

ラベル付けはラスタスキャンにて行うため、 $\text{thrmin} \leq \text{面積} \leq \text{thrmax}$ のラベルに対し、左上から順に番号が付けられます。

図（ラベル付け例）および面積フィルタ付きラベリング処理手順は、『IP_Label4withAreaFLT ()』コマンドを参照して下さい。

□ 機能

ソース画面（2値）に対し、4連結でラベリング処理を行い、指定しきい値で面積フィルタ処理後に面積でソートして再度ラベル付けを行い、デスティネーション画面（濃淡）に出力します。

ソース画面に濃淡画像を指定した場合は、濃淡画像を2値画像として処理します。画素データの最上位ビットが1の場合は白、0の場合は黒として処理します。

ラベリング処理は、仮ラベル付け、合流対解析、真ラベル付けの3ステップで行われます。仮ラベル付けでは同じ物体を異なるラベルとして認識するため、次ステップの合流対(同じ物体であることを示す仮ラベル対)解析で同じ物体に対して複数の仮ラベル番号が付いていないかを調べ、この結果をもとに真ラベル付けを行います。

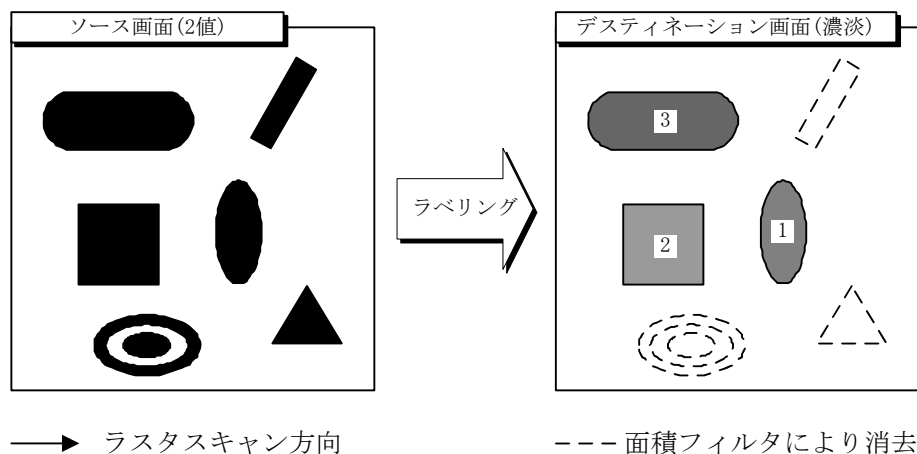
ラベル付けの最大数は255個です。ラベルオーバーフローは、ラベル数が256個以上、仮ラベル数が1023個以上、合流対が1024個以上のいずれかで発生します。

面積フィルタ処理および面積ソーティング処理は合流対解析の段階で行いますが、仮ラベル付けで仮ラベルが1023を超えるとラベルオーバーフローとして処理します。したがって、面積フィルタの設定で全てのラベルオーバーフローを回避することはできません。

ラベルオーバーフローの場合はコマンドエラーにならないためClearIPError()は必要ありません。引き続き次の処理を実行します。

ラベル付けはラスタスキャンにて行うため、図に示すように $\text{thrmin} \leq \text{面積} \leq \text{thrmax}$ のラベルに対し、opt2で指定されたソーティング方法で、左上から順にソーティング後の番号が付けられます。面積が同じ場合には、基本的に左上から順に番号がつけられます。

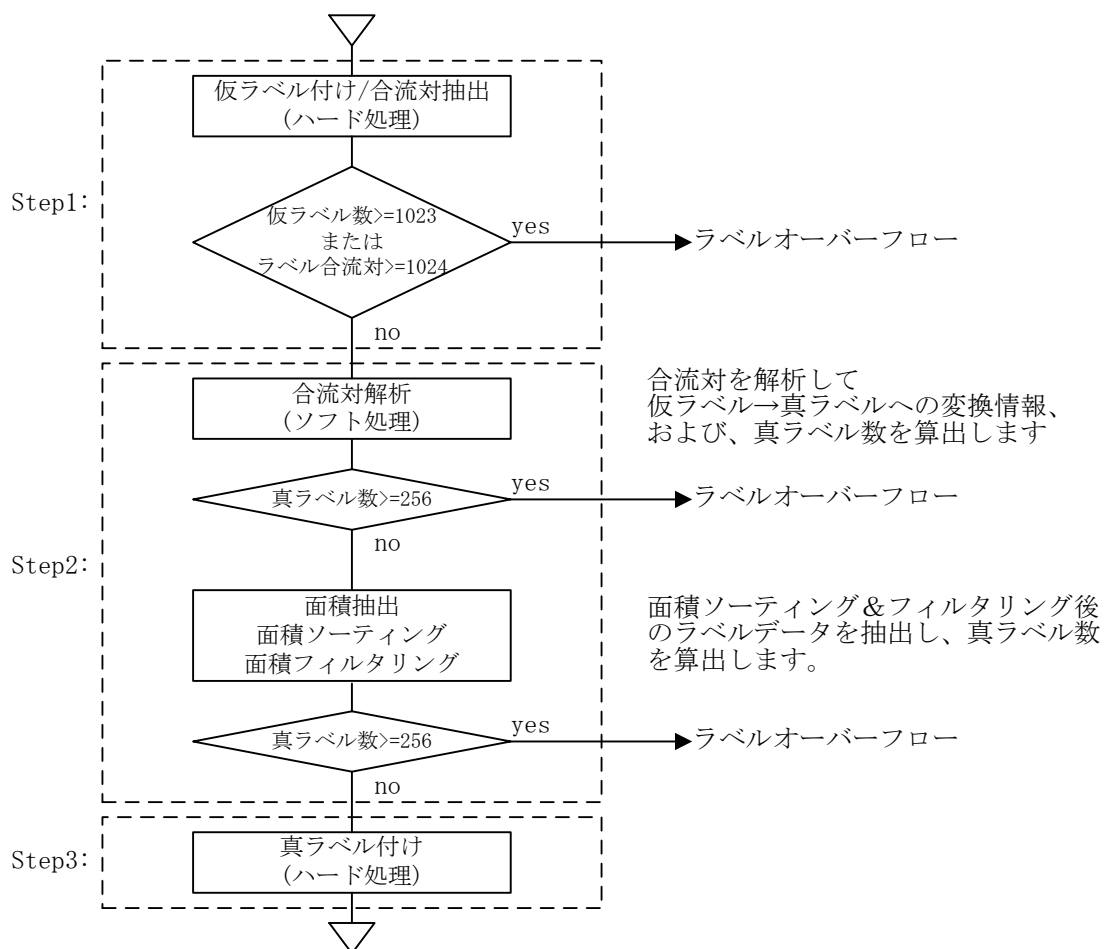
<ラベル付け例：面積が小さい順にソート>



注意：仮ラベル付け等の状況により、左上から右下にかけてラベル付けされない場合があります。

以下に面積ソート&フィルタ付きラベリング処理フローを示します。

<面積ソート&フィルタ付きラベリング処理手順>



□ 機能

ソース画面（2値）に対し、8連結でラベリング処理を行い、指定しきい値で面積フィルタ処理後に面積でソートして再度ラベル付けを行い、デスティネーション画面（濃淡）に出力します。

ソース画面に濃淡画像を指定した場合は、濃淡画像を2値画像として処理します。画素データの最上位ビットが1の場合は白、0の場合は黒として処理します。

ラベリング処理は、仮ラベル付け、合流対解析、真ラベル付けの3ステップで行われます。仮ラベル付けでは同じ物体を異なるラベルとして認識するため、次ステップの合流対(同じ物体であることを示す仮ラベル対)解析で同じ物体に対して複数の仮ラベル番号が付いていないかを調べ、この結果をもとに真ラベル付けを行います。

ラベル付けの最大数は255個です。ラベルオーバーフローは、ラベル数が256個以上、仮ラベル数が1023個以上、合流対が1024個以上のいずれかで発生します。

面積フィルタ処理および面積ソーティング処理は合流対解析の段階で行いますが、仮ラベル付けで仮ラベルが1023を超えるとラベルオーバーフローとして処理します。したがって、面積フィルタの設定で全てのラベルオーバーフローを回避することはできません。

ラベルオーバーフローの場合はコマンドエラーにならないためClearIPError()は必要ありません。引き続き次の処理を実行します。

ラベル付けはラスタスキャンにて行うため、図に示すように $\text{thrmin} \leq \text{面積} \leq \text{thrmax}$ のラベルに対し、opt2で指定されたソーティング方法で、左上から順に番号が付けられます。面積が同じ場合には、基本的に左上から順に番号がつけられます。

図（ラベル付け例）および＜面積ソート&フィルタ付きラベリング処理手順＞は、『IP_Label4withAreaFLTSort ()』コマンドを参照して下さい。

2.15.7 ラベル毎領域X座標抽出（IP_ExtractLORegionX）

□ C言語API

```
int ret = IP_ExtractLORegionX(IMGID ImgSrc, int *TblMinX, int *TblMaxX);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
int	*TblMinX	最小X座標を返すテーブルのアドレス テーブルはint型で256個の容量を確保して下さい。
int	*TblMaxX	最大X座標を返すテーブルのアドレス テーブルはint型で256個の容量を確保して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	△U	△U

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）
int	*TblMinX	最小X座標を格納したテーブルのアドレス
int	*TblMaxX	最大X座標を格納したテーブルのアドレス

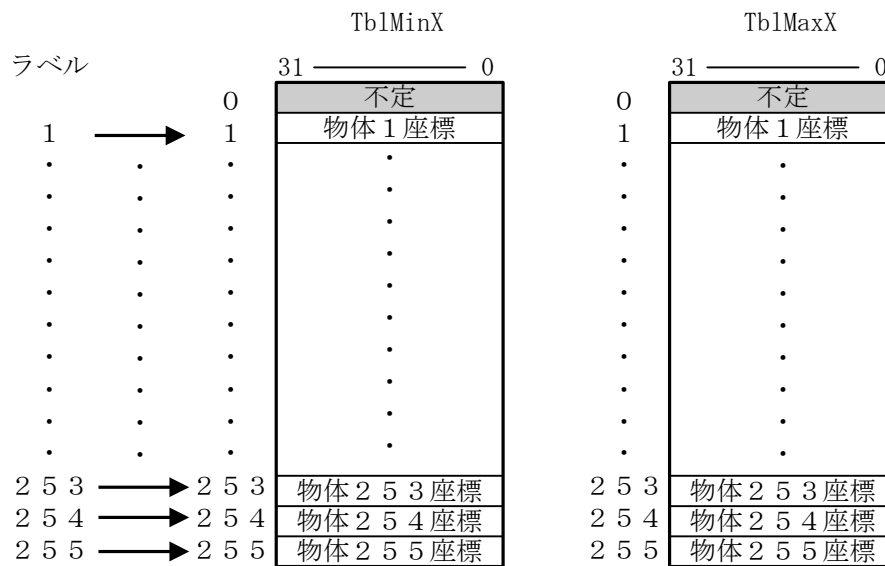
□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
21	最小X座標格納テーブルがアドレス不正
22	最大X座標格納テーブルがアドレス不正
30	ソース画面ウィンドウ設定エラー
48	動的メモリ確保エラー
49	ソース画面が不定画面
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面（ラベル画像）に対し、ラベル毎の最小／最大X座標を抽出します。
 ラベル画像に適用されるウィンドウは、ソース画面 1 のウィンドウ（SRC0_WIN）です。
 TblMinX、TblMaxXのフォーマットを以下に示します。TblMinX、TblMaxXには、下位 10 ビット
 に座標値が入り、上位 22 ビットは0になります。



本コマンドで得られる最小／最大X座標は、ソース画面 1 ウィンドウからの相対座標になります。
 画面上の座標位置はそれぞれ、

$$\begin{aligned} \text{最小X座標} &= \text{TblMinX} [\text{ラベル番号}] + \text{sx} \\ \text{最大X座標} &= \text{TblMaxX} [\text{ラベル番号}] + \text{sx} \end{aligned}$$

sx : ソース画面ウィンドウの始点X座標

となります。

2.15.8 ラベル毎領域Y座標抽出（IP_ExtractLORegionY）

□ C言語API

```
int ret = IP_ExtractLORegionY(IMGID ImgSrc, int *TblMinY, int *TblMaxY);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
int	*TblMinY	最小Y座標を返すテーブルのアドレス テーブルはint型で256個の容量を確保して下さい。
int	*TblMaxY	最大Y座標を返すテーブルのアドレス テーブルはint型で256個の容量を確保して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	△U	△U

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）
int	*TblMinY	最小Y座標を格納したテーブルのアドレス
int	*TblMaxY	最大Y座標を格納したテーブルのアドレス

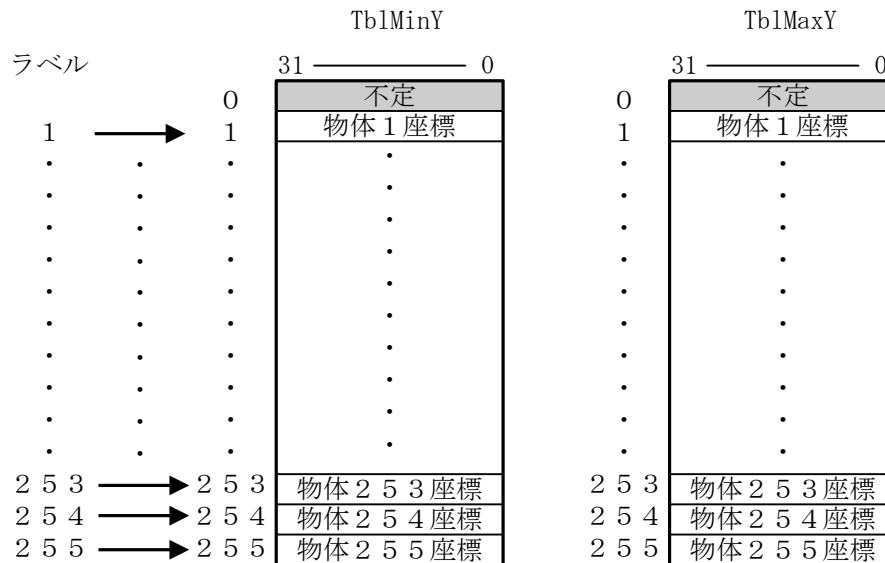
□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
21	最小Y座標格納テーブルがアドレス不正
22	最大Y座標格納テーブルがアドレス不正
30	ソース画面ウィンドウ設定エラー
48	動的メモリ確保エラー
49	ソース画面が不定画面
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面（ラベル画像）に対し、ラベル毎の最小／最大Y座標を抽出します。
 ラベル画像に適用されるウィンドウは、ソース画面 1 のウィンドウ（SRC0_WIN）です。。
 TblMinY、TblMaxYのフォーマットを以下に示します。TblMinY、TblMaxYには、下位 9 ビットに座標値が入り、上位 23 ビットは 0 になります。



本コマンドで得られる最小／最大Y座標は、ソース画面 1 ウィンドウからの相対座標になります。
 画面上の座標位置はそれぞれ、

$$\begin{aligned} \text{最小Y座標} &= \text{TblMinY} [\text{ラベル番号}] + \text{sy} \\ \text{最大Y座標} &= \text{TblMaxY} [\text{ラベル番号}] + \text{sy} \end{aligned}$$

sy : ソース画面ウィンドウの始点Y座標

となります。

2.15.9 ラベル毎面積抽出（IP_ExtractLOArea / IP_ExtractLOAreaExt）

□ C言語API

```
int ret = IP_ExtractLOArea(IMGID ImgSrc, int *Tbl);
int ret = IP_ExtractLOAreaExt(IMGID ImgSrc, long *Tbl);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
int	*Tbl	<IP_ExtractLOArea> ラベル毎の面積を返すテーブルのアドレス テーブルはint型で256個の容量を確保して下さい。
long	*Tbl	<IP_ExtractLOAreaExt> ラベル毎の面積を返すテーブルのアドレス テーブルはlong型で256個の容量を確保して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	△U	△U

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）
int	*Tbl	<IP_ExtractLOArea> ラベル毎の面積を格納したテーブルのアドレス
long	*Tbl	<IP_ExtractLOAreaExt> ラベル毎の面積を格納したテーブルのアドレス

□ エラーコード

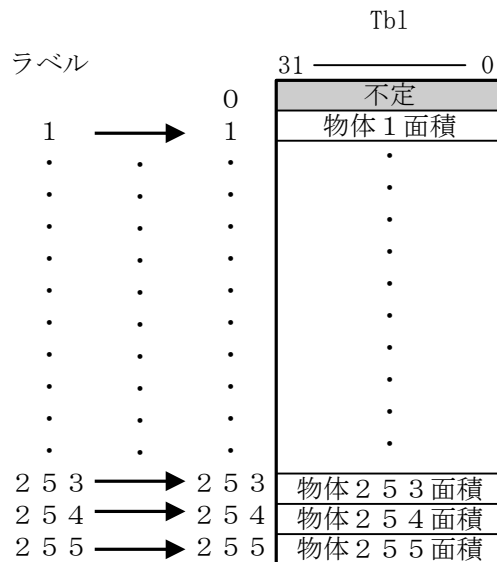
2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
21	面積格納テーブルがアドレス不正
30	ソース画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面（ラベル画像）に対し、ラベル毎の面積を抽出します。『IP_ExtractLOArea()』コマンドと『IP_ExtractLOAreaExt()』コマンドは同機能のコマンドです。
ラベル画像に適用されるウィンドウは、ソース画面 1 のウィンドウ（SRC0_WIN）です。

（注）『ClearIPError()』コマンドで読み出した時のエラー発生コマンド名は、“IP_ExtractLOArea”となります。



ラベル個数よりも大きい部分は不定です

2.15.10 ラベル毎重心座標抽出（IP_ExtractLOGravity）

□ C言語API

```
int ret = IP_ExtractLOGravity(IMGID ImgSrc, IPL0GravityTbl *Tbl);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IPL0GravityTbl	*Tbl	ラベル毎の重心と面積を返すテーブルのアドレス テーブルはIPL0GravityTbl型で256個の容量を確保して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	△U	△U

□ リターンパラメータ

int	ret	ラベル個数（0～255）または異常終了（-1）
IPL0GravityTbl	*Tbl	ラベル毎の重心と面積を格納したテーブルのアドレス

□ パケットの構造

```
typedef struct {
    float    X;           重心X座標
    float    Y;           重心Y座標
    long     AREA;        ラベル毎の面積
} IPL0GravityTbl;
```

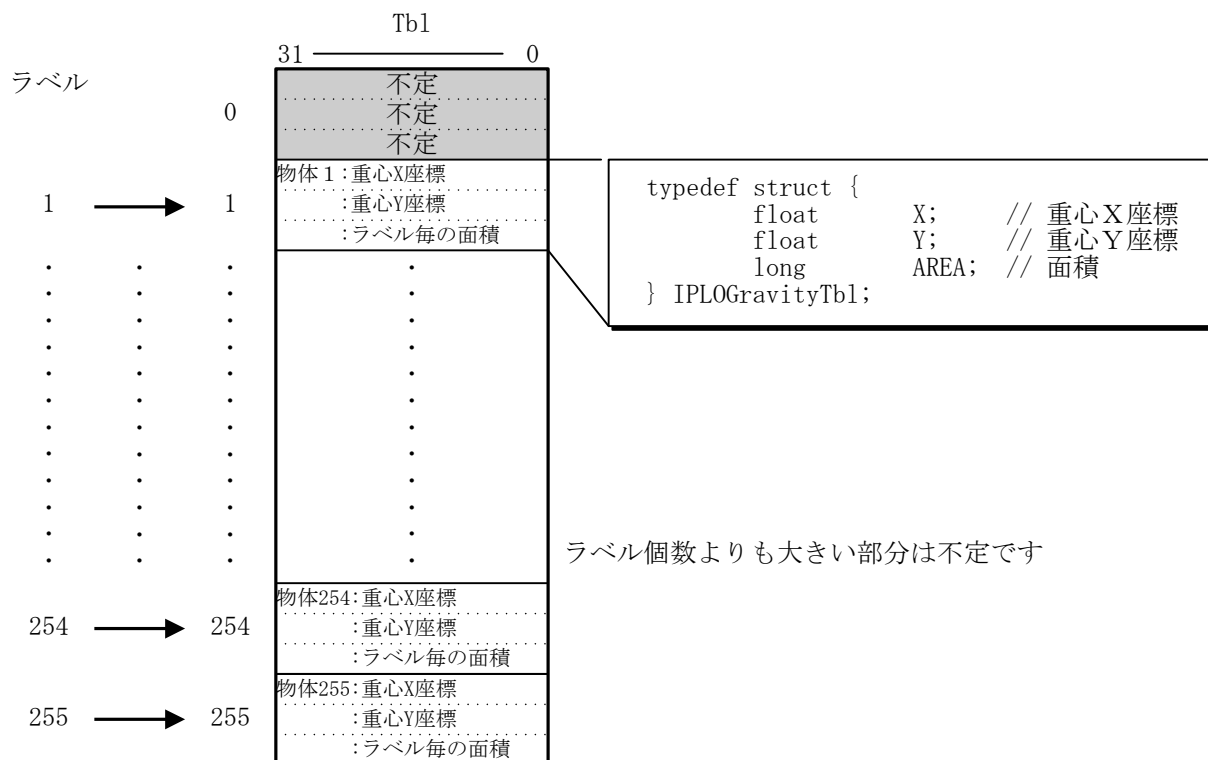
□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
21	重心/面積格納テーブルがアドレス不正
30	ソース画面ウィンドウ設定エラー
48	動的メモリ確保エラー
49	ソース画面が不定画面
-	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面（ラベル画像）に対し、ラベル毎の重心座標と面積を抽出します。
ラベル画像に適用されるウィンドウは、ソース画面 1 のウィンドウ（SRC0_WIN）です。



本コマンドで得られる重心X／Y座標は、ソース画面ウィンドウからの相対座標になります。
画面上の座標位置はそれぞれ、

重心X座標 = Tbl [ラベル番号].X + sx
重心Y座標 = Tbl [ラベル番号].Y + sy

sx, sy : ソース画面ウィンドウの始点座標

となります。

2.16 濃淡画像特徴量抽出

2.16.1 濃淡画像基本特徴量抽出（IP_ExtractGOFeatures）

□ C言語API

```
int ret = IP_ExtractGOFeatures(IMGID ImgSrc, IPGOFeatureTbl *FtrTbl, enum IPGOFeatureOpt opt);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IPGOFeatureTbl	*FtrTbl	基本特徴量を返すアドレス
enum IPGOFeatureOpt	opt	基本特徴量抽出オプション GRAY_REGISTER (0) 特徴量のみ抽出する

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	△U

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）
IPGOFeatureTbl	*FtrTbl	基本特徴量を格納したアドレス

□ パケットの構造

```
typedef struct {
    int    MIN_LEVEL_X;    最小濃度値をとる点のX座標
    int    MIN_LEVEL_Y;    最小濃度値をとる点のY座標
    int    MAX_LEVEL_X;    最大濃度値をとる点のX座標
    int    MAX_LEVEL_Y;    最大濃度値をとる点のY座標
    long   ACC_LEVEL;      濃度累積値
    int    MIN_LEVEL;      最小濃度値
    int    MAX_LEVEL;      最大濃度値
    int    TYPICAL_LEVEL;  予備
} IPGOFeatureTbl;
```

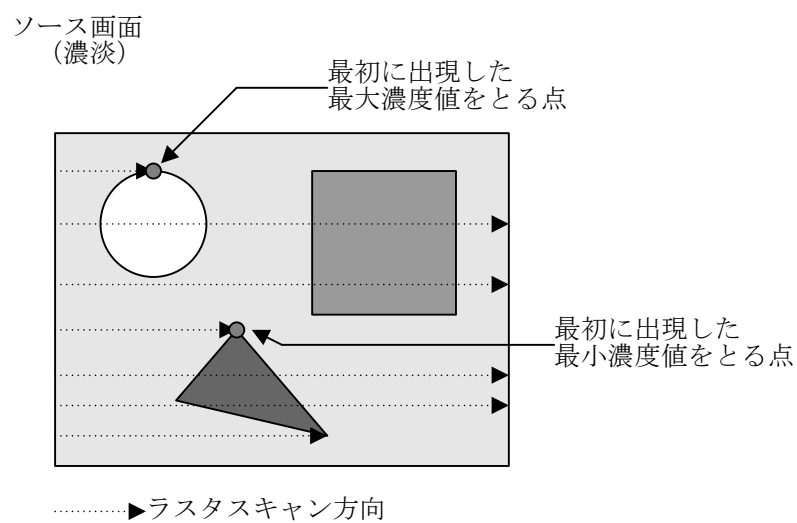
□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
21	基本特徴量格納アドレス不正
22	基本特徴量抽出オプション不正
30	ソース画面ウィンドウ設定エラー
49	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面（濃淡）に対し、基本特徴量抽出を行います。最小／最大濃度値をとる点とは、ラスタスキャンした際に最初に出現した最小／最大濃度値をとる点を指します。



2.16.2 濃度ヒストグラム (IP_Histogram / IP_HistogramShort)

□ C言語API

```
int ret = IP_Histogram(IMGID ImgSrc, long *Tbl, IPG0FeatureTbl *FtrTbl, int opt);
int ret = IP_HistogramShort(IMGID ImgSrc, int *Tbl, IPG0FeatureTbl *FtrTbl, int opt);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
long	*Tbl	<IP_Histogram> 濃度頻度分布を返すアドレス テーブルはlong型で256個の容量を確保して下さい。
int	*Tbl	<IP_HistogramShort> 濃度頻度分布を返すアドレス テーブルはint型で256個の容量を確保して下さい。
IPG0FeatureTbl	*FtrTbl	基本特徴量を返すアドレス
int	opt	ヒストグラム処理オプション 0を指定して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	×

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)
long	*Tbl	<IP_Histogram> 濃度頻度分布を格納したアドレス
int	*Tbl	<IP_HistogramShort> 濃度頻度分布を格納したアドレス
IPG0FeatureTbl	*FtrTbl	基本特徴量を格納したアドレス

□ パケットの構造

IPG0FeatureTbl構造体については、『IP_ExtractG0Features()』コマンドを参照して下さい。

□ エラーコード

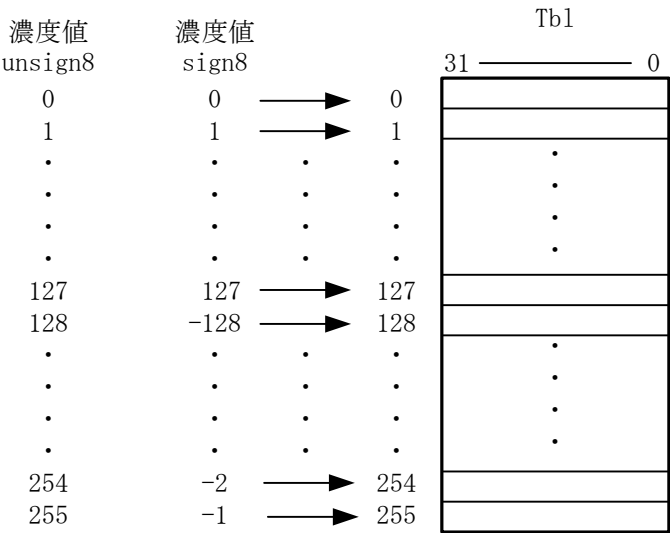
2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
21	濃度頻度分布格納アドレス不正
22	基本特徴量格納アドレス不正
23	ヒストグラム処理オプション不正
30	ソース画面ウィンドウ設定エラー
49	ソース画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

機能

ソース画面に対しヒストグラム処理を行い、濃度頻度分布及び濃淡画像基本特徴量抽出を行います。
『IP_Histogram()』コマンドと『IP_HistogramShort()』コマンドは同機能のコマンドです。
ソース画面データタイプが2値の場合は、異常終了(エラーコード10)します。

(注) 『ClearIPErr()』コマンドで読み出した時のエラー発生コマンド名は、“IP_Histogram”となります。



2.16.3 濃度ヒストグラム特徴量抽出 (IP_HistogramFeatures)

□ C言語API

```
int ret = IP_HistogramFeatures( IMGID ImgSrc , long *Tbl , HISTOGRAM_FEATURE *RegTbl ,
                                int opt );
```

□ パラメータ

IMGID ImgSrc ソース画面の画面番号

long *Tbl 濃度頻度分布を格納する領域へのポインタ。
4 バイト×256 の容量を確保して下さい。
「NULL」を指定すると濃度頻度分布は出力されません。

HISTOGRAM_FEATURE *RegTbl ヒストグラム特徴量を格納する領域へのポインタ

int opt オプション。0 を設定して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	×

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

1 0 指定画面空き領域、または範囲外 (不当画面番号エラー)
4 9 アクセス画面が不定画面
— エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面に対しヒストグラム処理を行い、濃度頻度累積及び濃度画像ヒストグラム特徴量抽出を行います。

HISTOGRAM_FEATUREのフォーマット及び内容を、下記に示します。

また、濃度累積値（ACC_LEVEL）が0以外のときのみ、テーブル内データが有効です。

ここで、同じ値の複数の最小/最大濃度値が存在する場合の最小/最大濃度値をとる点とはラストスキャンした際に最初に出現した最小/最大濃度値をとる点を指します。

vp900.hで宣言しています

```
typedef struct {
    int    MIN_LEVEL_X;
    int    MIN_LEVEL_Y;
    int    MAX_LEVEL_X;
    int    MAX_LEVEL_Y;
    long   ACC_LEVEL;
    int    MIN_LEVEL;
    int    MAX_LEVEL;
    int    TYPICAL_LEVEL;
    long   N;
    float  MEAN;
    float  VARIANCE;
    float  DEVIATION;
} HISTOGRAM_FEATURE;
```

基本特徴テーブル	説明
MIN_LEVEL_X	最小濃度値をとる点のX座標
MIN_LEVEL_Y	最小濃度値をとる点のY座標
MAX_LEVEL_X	最大濃度値をとる点のX座標
MAX_LEVEL_Y	最大濃度値をとる点のY座標
ACC_LEVEL	濃度累積
MIN_LEVEL	最小濃度値
MAX_LEVEL	最大濃度値
TYPICAL_LEVEL	予備
N	総画素数
MEAN	平均濃度値
VARIANCE	分散値
DEVIATION	標準偏差

□ 制限事項

ウィンドウサイズ1×1の画像メモリに対して本コマンドを発行した場合、正しい結果が得られません。

2.16.4 X/Y軸への濃度累積値投影(濃淡画像)(IP_ProjectGO)

□ C言語API

```
int ret = IP_ProjectGO(IMGID ImgSrc, int *TblX, int *TblY, IPGOFeatureTbl *FtrTbl);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
int	*TblX	濃度累積値(X軸への投影)を返すアドレス テーブルサイズはint型で処理サイズのX方向、Y方向の大きい方のサイズ分の容量を確保して下さい。
int	*TblY	濃度累積値(Y軸への投影)を返すアドレス テーブルサイズはint型で処理サイズのX方向、Y方向の大きい方のサイズ分の容量を確保して下さい。
IPGOFeatureTbl	*FtrTbl	基本特徴量を返すアドレス

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	×

□ リターンパラメータ

int	ret	正常終了(0)または異常終了(-1)
int	*TblX	濃度累積値(X軸への投影)を格納したアドレス
int	*TblY	濃度累積値(Y軸への投影)を格納したアドレス
IPGOFeatureTbl	*FtrTbl	基本特徴量を格納したアドレス

□ パケットの構造

IPGOFeatureTbl構造体については、『IP_ExtractGOFeatures()』コマンドを参照して下さい

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外(不当画面番号エラー)
21	濃度累積値(X軸への投影)格納アドレス不正
22	濃度累積値(Y軸への投影)格納アドレス不正
23	基本特徴量格納アドレス不正
30	ソース画面ウィンドウ設定エラー
48	動的メモリ確保エラー
49	ソース画面が不定画面
—	エラーリセットコマンド(ClearIPError)未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

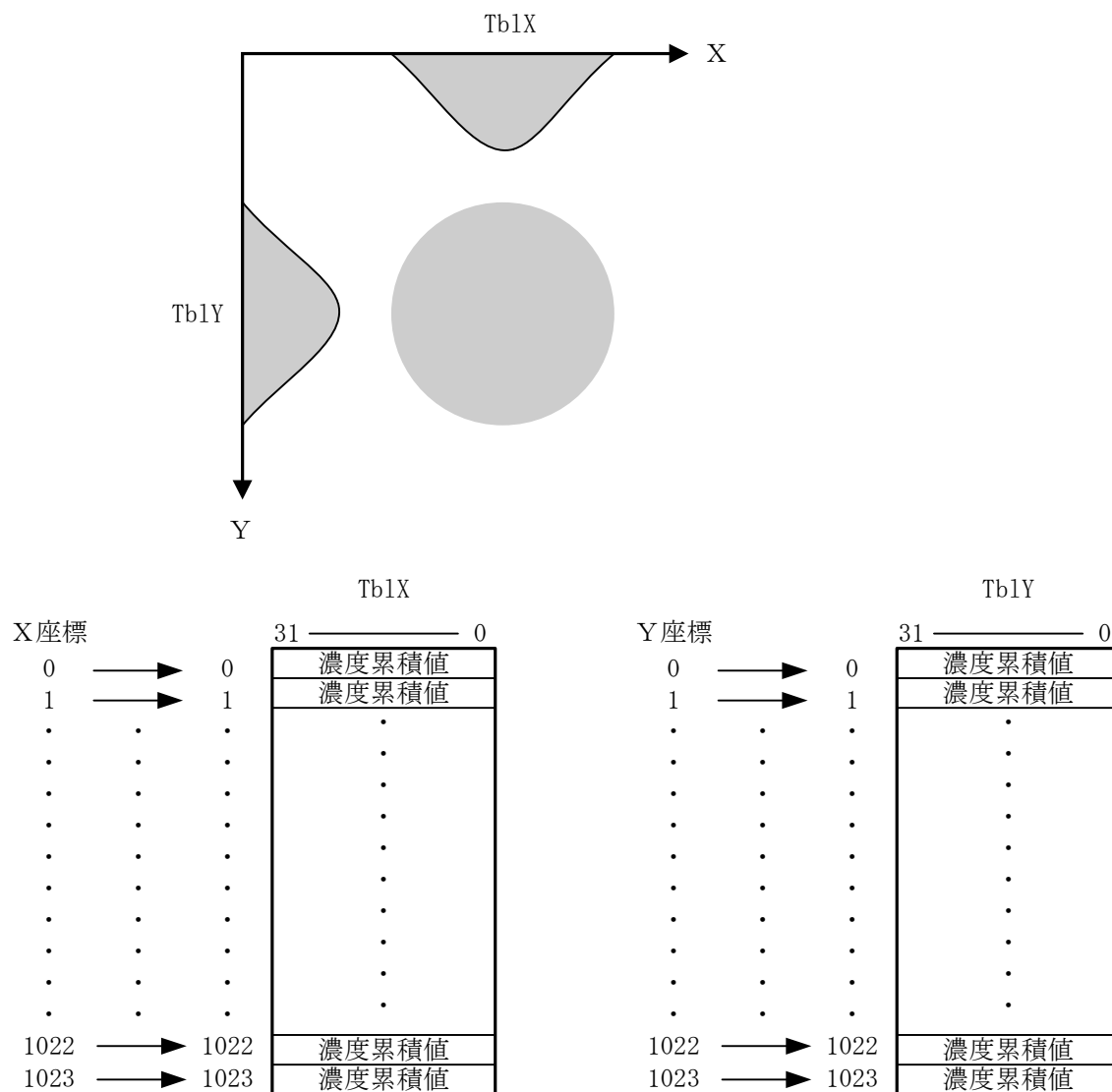
□ 機能

濃淡画像でのX/Y軸への投影を行い、濃度累積及び濃淡画像基本特徴量抽出を行います。本コマンドの濃度累積値の有効ビット数は16ビットです。結果格納する32ビットの内上位16ビットは、符号なし8ビットの場合0拡張をし、符号付8ビットの場合は、15ビット目の値を符号拡張した値となります。

書き込まれるテーブルのサイズ（範囲）は、ソース画面ウィンドウのサイズでX方向、Y方向の大きい方の範囲で書き込まれます。濃度累積値格納テーブルは大きい方のサイズにあわせて確保して下さい。このとき、本来のサイズを上回って結果が格納される部分には0が書き込まれます。

結果がオーバーフロー/アンダーフローした場合には、オーバーフロー/アンダーフローの直前の値を出力するため、正しい結果となりません。

ソース画面データタイプが2値の場合は、異常終了（エラーコード10）します。



本コマンドで得られるTb1X、Tb1Yの濃度累積値は、ソース画面ウィンドウからの相対座標になります。画面上の座標位置での濃度累積値はそれぞれ、

$$\begin{aligned} \text{X軸濃度累積値} &= \text{Tb1X} [X - \text{sx}] \\ \text{Y軸濃度累積値} &= \text{Tb1Y} [Y - \text{sy}] \end{aligned}$$

X : 画面X座標
Y : 画面Y座標
sx : ソース画面ウィンドウの始点X座標
sy : ソース画面ウィンドウの始点Y座標

となります。

2.16.5 X軸への濃度累積値投影(濃淡画像)(IP_ProjectGOonX)

□ C言語API

```
int ret = IP_ProjectGOonX(IMGID ImgSrc, long *TblX, IPGOFeatureTbl *FtrTbl);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

long *TblX 濃度累積値(X軸への投影)を返すアドレス
テーブルサイズはlong型で処理サイズの大きさを確保して下さい。

IPGOFeatureTbl *FtrTbl 基本特徴量を返すアドレス

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	×

□ リターンパラメータ

int ret 正常終了(0)または異常終了(-1)

long *TblX 濃度累積値(X軸への投影)を格納したアドレス

IPGOFeatureTbl *FtrTbl 基本特徴量を格納したアドレス

□ パケットの構造

IPGOFeatureTbl構造体については、『IP_ExtractGOFeatures()』コマンドを参照して下さい

□ エラーコード

2 OSシステムエラー

3 IMP処理タイムアウト

8 VIN入力タイムアウト

1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

2 1 濃度累積値(X軸への投影)格納アドレス不正

2 2 基本特徴量格納アドレス不正

3 0 ソース画面ウィンドウ設定エラー

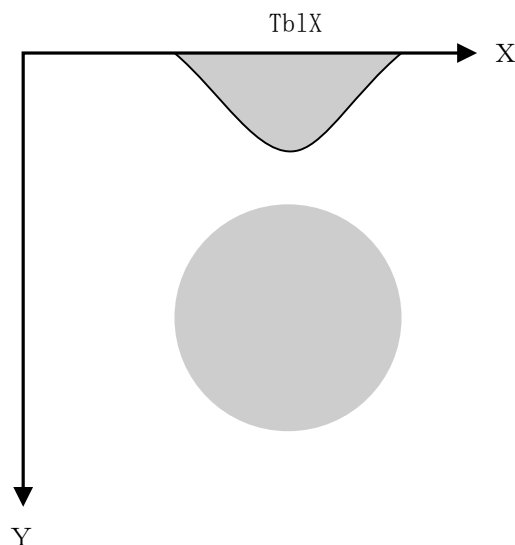
4 9 ソース画面が不定画面

— エラーリセットコマンド(ClearIPError)未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

濃淡画像でのX軸への投影を行い、濃度累積及び濃淡画像基本特徴量抽出を行います。
書き込まれるテーブルのサイズ（範囲）は、ソース画面ウィンドウのX方向サイズになります。
ソース画面データタイプが2値の場合は、異常終了（エラーコード10）します。



※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

			TblX	
X座標			31	0
0	→	0	濃度累積値	
1	→	1	濃度累積値	
⋮	⋮	⋮	⋮	
⋮	⋮	⋮	⋮	
⋮	⋮	⋮	⋮	
⋮	⋮	⋮	⋮	
⋮	⋮	⋮	⋮	
⋮	⋮	⋮	⋮	
⋮	⋮	⋮	⋮	
1022	→	1022	濃度累積値	
1023	→	1023	濃度累積値	

(注) ウィンドウサイズを越える部分は不定です。

本コマンドで得られるTblXの濃度累積値は、ソース画面ウィンドウからの相対座標になります。
画面上の座標位置での濃度累積値は、

$$\text{X軸濃度累積値} = \text{TblX} [X - \text{sx}]$$

X : 画面X座標
sx : ソース画面ウィンドウの始点X座標
となります。

2.16.6 Y軸への濃度累積値投影(濃淡画像)(IP_ProjectGOonY)

□ C言語API

```
int ret = IP_ProjectGOonY(IMGID ImgSrc, long *TblY, IPGOFeatureTbl *FtrTbl);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

long *TblY 濃度累積値(Y軸への投影)を返すアドレス
テーブルサイズはlong型で処理サイズの大きさを確保して下さい。

IPGOFeatureTbl *FtrTbl 基本特徴量を返すアドレス

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	×

□ リターンパラメータ

int ret 正常終了(0)または異常終了(-1)

long *TblY 濃度累積値(Y軸への投影)を格納したアドレス

IPGOFeatureTbl *FtrTbl 基本特徴量を格納したアドレス

□ パケットの構造

IPGOFeatureTbl構造体については、『IP_ExtractGOFeatures()』コマンドを参照して下さい

□ エラーコード

2 OSシステムエラー

3 IMP処理タイムアウト

8 VIN入力タイムアウト

1 0 ソース画面番号に対応する画面が未確保、または番号が有効範囲外
(不当画面番号エラー)

2 1 濃度累積値(Y軸への投影)格納アドレス不正

2 2 基本特徴量格納アドレス不正

3 0 ソース画面ウィンドウ設定エラー

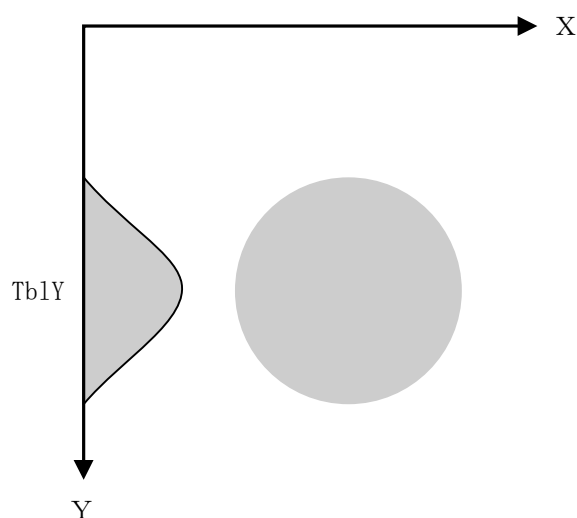
4 9 ソース画面が不定画面

— エラーリセットコマンド(ClearIPError)未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

濃淡画像でのY軸への投影を行い、濃度累積及び濃淡画像基本特徴量抽出を行います。
書き込まれるテーブルのサイズ（範囲）は、ソース画面ウィンドウのY方向サイズになります。
ソース画面データタイプが2値の場合は、異常終了（エラーコード10）します。



		TblY	
Y座標		31	0
0	→	0	濃度累積値
1	→	1	濃度累積値
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
1022	→	1022	濃度累積値
1023	→	1023	濃度累積値

(注) ウィンドウサイズを越える部分は不定です。

本コマンドで得られるTblYの濃度累積値は、ソース画面ウィンドウからの相対座標になります。
画面上の座標位置での濃度累積値は、

$$Y \text{ 軸濃度累積値} = \text{TblY} [Y - sy]$$

Y : 画面Y座標
sy : ソース画面ウィンドウの始点Y座標
となります。

2.16.7 X/Y軸への最大濃度値投影(濃淡画像)(IP_ProjectGOMaxValue)

□ C言語API

```
int ret = IP_ProjectGOMaxValue(IMGID ImgSrc, int *TblX, int *TblY, IPGOFeatureTbl *FtrTbl);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
int	*TblX	最大濃度値(X軸への投影)を返すアドレス テーブルサイズはint型で処理サイズのX方向、Y方向の大きい方のサイズ分の容量を確保して下さい。
int	*TblY	最大濃度値(Y軸への投影)を返すアドレス テーブルサイズはint型で処理サイズのX方向、Y方向の大きい方のサイズ分の容量を確保して下さい。
IPGOFeatureTbl	*FtrTbl	基本特徴量を返すアドレス

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	×

□ リターンパラメータ

int	ret	正常終了(0)または異常終了(-1)
int	*TblX	最大濃度値(X軸への投影)を格納したアドレス
int	*TblY	最大濃度値(Y軸への投影)を格納したアドレス
IPGOFeatureTbl	*FtrTbl	基本特徴量を格納したアドレス

□ パケットの構造

IPGOFeatureTbl構造体については、『IP_ExtractGOFeatures()』コマンドを参照して下さい

□ エラーコード

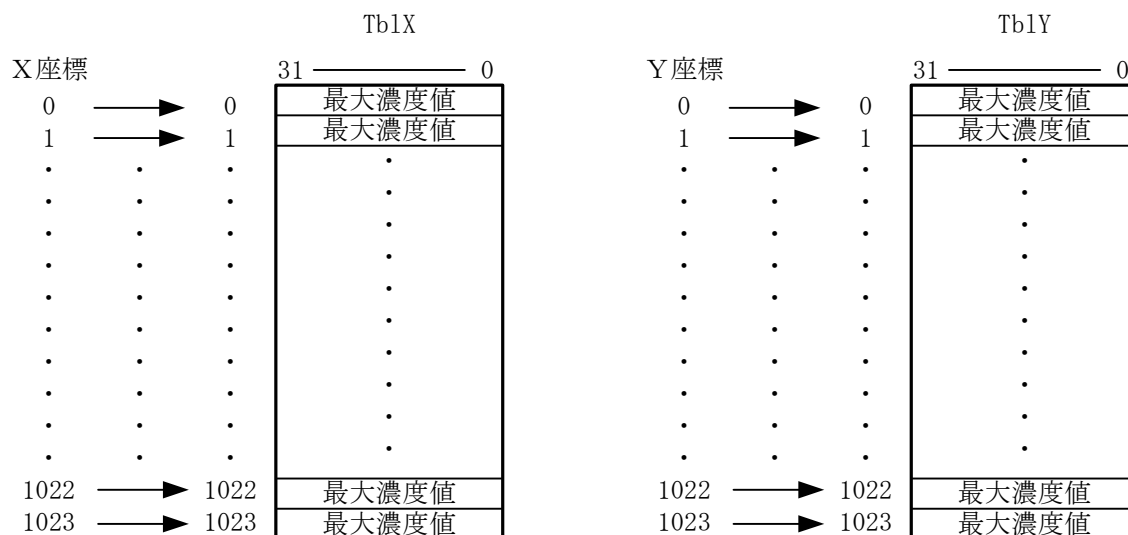
2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外(不当画面番号エラー)
21	最大濃度値(X軸への投影)格納アドレス不正
22	最大濃度値(Y軸への投影)格納アドレス不正
23	基本特徴量格納アドレス不正
30	ソース画面ウィンドウ設定エラー
48	動的メモリ確保エラー
49	ソース画面が不定画面
—	エラーリセットコマンド(ClearIPError)未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

濃淡画像でのX/Y軸への最大濃度値投影を行い、その結果及び濃淡画像基本特徴量抽出を行います。書き込まれるテーブルのサイズ（範囲）は、ソース画面ウィンドウのサイズでX方向、Y方向の大きい方の範囲で書き込まれます。最大濃度値格納テーブルは大きい方のサイズにあわせて確保して下さい。このとき、本来のサイズを上回って結果が格納される部分には、ソース画面データタイプが符号なし8ビットの場合は3 2 7 6 8（0 x 8 0 0 0）、符号付8ビットの場合は- 3 2 7 6 8（0 x F F F F 8 0 0 0）になります。

ソース画面データタイプが2値の場合は、異常終了（エラーコード1 0）します。

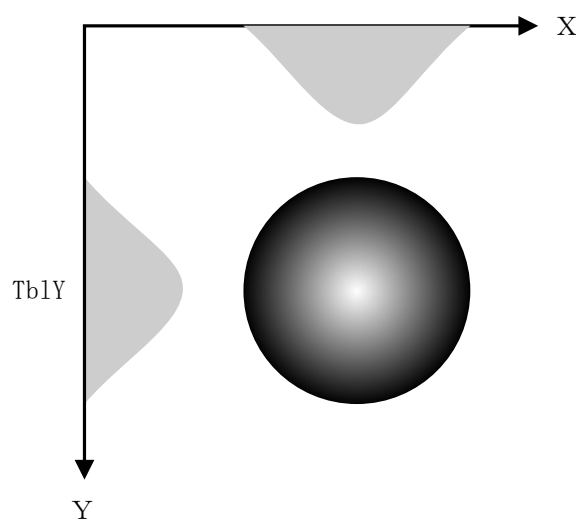


（注）ウィンドウサイズを越える部分は不定です。

本コマンドで得られるTb1X、Tb1Yの最大濃度値は、ソース画面ウィンドウからの相対座標になります。画面上の座標位置での最大濃度値はそれぞれ、

$$\begin{aligned} \text{X 軸最大濃度値} &= \text{Tb1X} [X - \text{sx}] \\ \text{Y 軸最大濃度値} &= \text{Tb1Y} [Y - \text{sy}] \end{aligned}$$

X : 画面X座標
Y : 画面Y座標
sx : ソース画面ウィンドウの始点X座標
sy : ソース画面ウィンドウの始点Y座標
となります。



2.16.8 X/Y軸への最小濃度値投影(濃淡画像)(IP_ProjectGOMinValue)

□ C言語API

```
int ret = IP_ProjectGOMinValue(IMGID ImgSrc, int *TblX, int *TblY, IPGOFeatureTbl *FtrTbl);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
int	*TblX	最小濃度値(X軸への投影)を返すアドレス テーブルサイズはint型で処理サイズのX方向、Y方向の大きい方のサイズ分の容量を確保して下さい。
int	*TblY	最小濃度値(Y軸への投影)を返すアドレス テーブルサイズはint型で処理サイズのX方向、Y方向の大きい方のサイズ分の容量を確保して下さい。
IPGOFeatureTbl	*FtrTbl	基本特徴量を返すアドレス

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	×

□ リターンパラメータ

int	ret	正常終了(0)または異常終了(-1)
int	*TblX	最小濃度値(X軸への投影)を格納したアドレス
int	*TblY	最小濃度値(Y軸への投影)を格納したアドレス
IPGOFeatureTbl	*FtrTbl	基本特徴量を格納したアドレス

□ パケットの構造

IPGOFeatureTbl構造体については、『IP_ExtractGOFeatures()』コマンドを参照して下さい

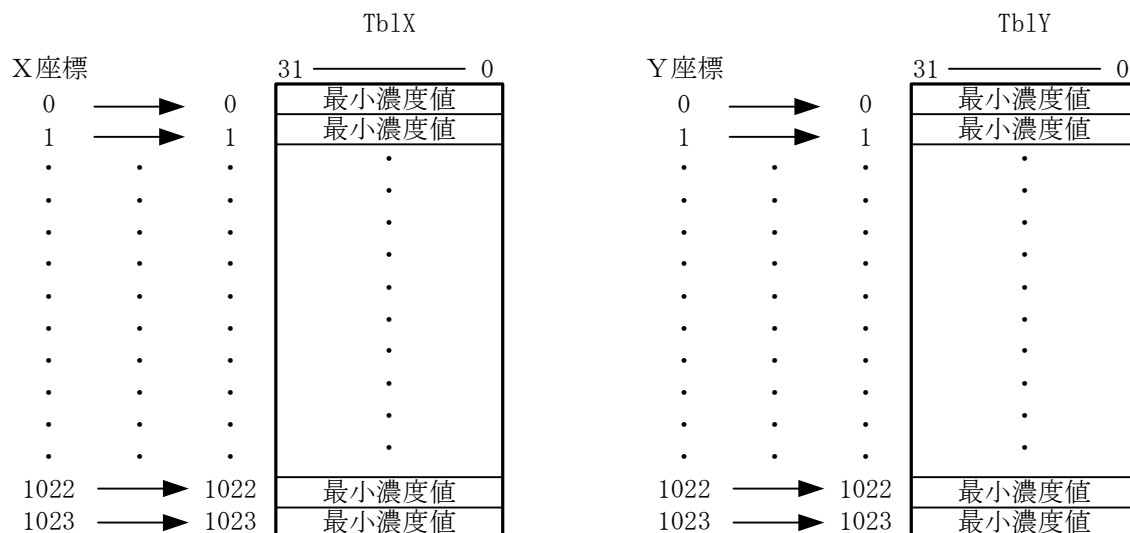
□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外(不当画面番号エラー)
21	最小濃度値(X軸への投影)格納アドレス不正
22	最小濃度値(Y軸への投影)格納アドレス不正
23	基本特徴量格納アドレス不正
30	ソース画面ウィンドウ設定エラー
48	動的メモリ確保エラー
49	ソース画面が不定画面
—	エラーリセットコマンド(ClearIPError)未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

濃淡画像でのX/Y軸への最小濃度値投影を行い、その結果及び濃淡画像基本特徴量抽出を行います。書き込まれるテーブルのサイズ（範囲）は、ソース画面ウィンドウのサイズでX方向、Y方向の大きい方の範囲で書き込まれます。最小濃度値格納テーブルは大きい方のサイズにあわせて確保して下さい。このとき、本来のサイズを上回って結果が格納される部分には、ソース画面データタイプが符号なし8ビット、符号付8ビットともに32767（0x7FFF）になります。ソース画面データタイプが2値の場合は、異常終了（エラーコード10）します。

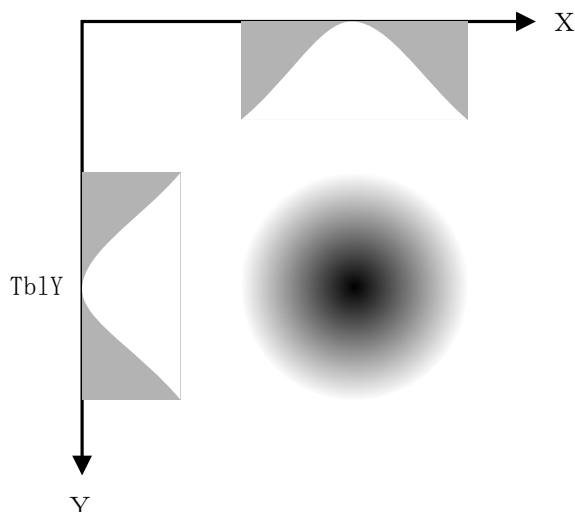


（注）ウィンドウサイズを越える部分は不定です。

本コマンドで得られるTb1X、Tb1Yの最小濃度値は、ソース画面ウィンドウからの相対座標になります。画面上の座標位置での最小濃度値はそれぞれ、

$$\begin{aligned} \text{X軸最小濃度値} &= \text{Tb1X} [X - s_x] \\ \text{Y軸最小濃度値} &= \text{Tb1Y} [Y - s_y] \end{aligned}$$

X : 画面X座標
Y : 画面Y座標
sx : ソース画面ウィンドウの始点X座標
sy : ソース画面ウィンドウの始点Y座標
となります。



2.16.9 領域毎濃淡画像濃度累積（IP_ProjectBlockGO）

□ C言語API

```
int ret = IP_ProjectBlockGO(IMGID ImgSrc, long *Tbl,
                           IPGOFeatureTbl *FtrTbl, IPDivideTbl *DivideTbl);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
long	*Tbl	領域毎濃度累積値データを返すアドレス テーブルサイズはlong型で領域分割数 (X軸分割数×Y軸分割数)用意して下さい。 (最大1024個)
IPGOFeatureTbl	*FtrTbl	基本特徴量を返すアドレス
IPDivideTbl	*DivideTbl	領域分割数を格納したアドレス

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	×

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）
long	*Tbl	領域毎濃度累積値データを格納したアドレス
IPGOFeatureTbl	*FtrTbl	基本特徴量を格納したアドレス

□ パケットの構造

IPGOFeatureTbl構造体については、『IP_ExtractGOFeatures()』コマンドを参照して下さい

```
typedef struct {
    int    Divide_X;    領域毎抽出X方向分割数
    int    Divide_Y;    領域毎抽出Y方向分割数
} IPDivideTbl;
```

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
21	領域毎濃度累積値データ格納アドレス不正
22	基本特徴量格納アドレス不正
23	領域分割数格納アドレス不正、または領域分割数が不正
30	ソース画面ウィンドウ設定エラー
48	動的メモリ確保エラー
49	ソース画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画像（濃淡）の画面領域を設定したパラメータで分割し、各領域毎の濃度累積値及び濃淡画像基本特徴量抽出を行います。

ソース画面データタイプが2値の場合は、異常終了（エラーコード10）します。

画面分割数と処理領域の間に次の関係が成立しないと特徴量抽出で誤差が発生します。

- 条件(1) (画面分割数) * 1024 / (処理領域の長さ) が割り切れること
 (2) (処理領域の長さ) / (画面分割数) が割り切れること
 (3) (画面分割数) と (処理領域の長さ) が2のN乗であること

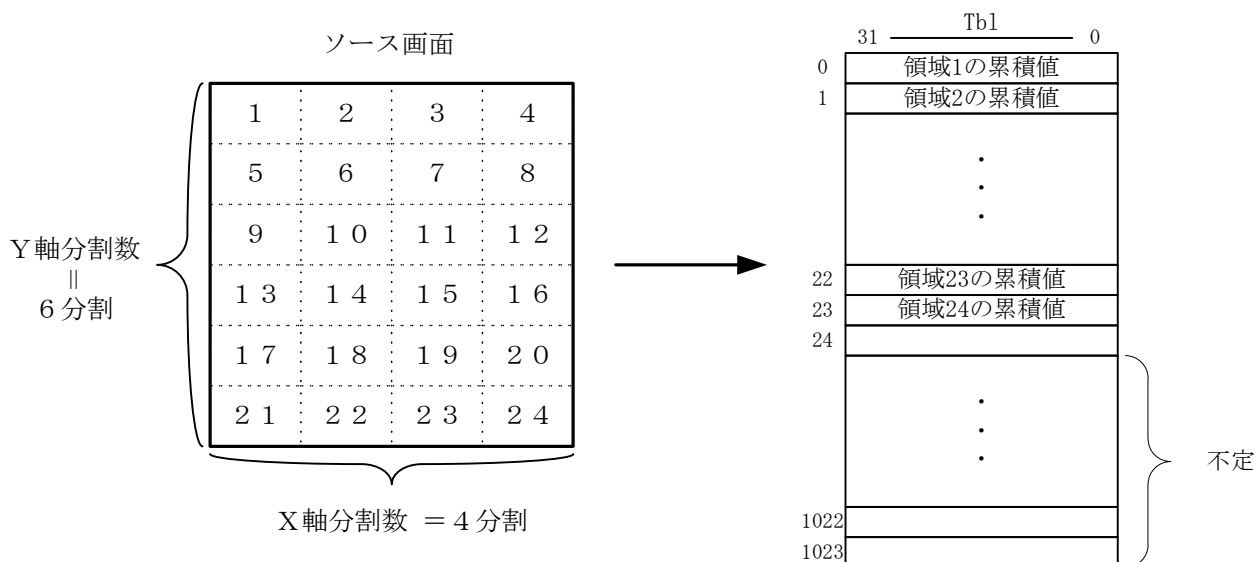
これは、『IP_ProjectBlockB0()』コマンド及び『IP_ProjectBlockG0()』コマンドを行う場合のヒストグラムメモリ（HM）への画像処理LSI内部のアドレス発生回路が固定小数点演算で行っており、上記の条件を満たさないとHMのアドレスに誤差が生じる為です。なお、『IP_ProjectBlockB0()』コマンド、『IP_ProjectBlockG0MinMaxValue()』コマンド、及び『IP_ProjectBlockG0()』コマンド以外では、このアドレッシングを使用しないので、他のコマンドには影響ありません。

X、Y軸の分割数はウィンドウの画素数を越えて設定できません。分割数の指定には以下の制約がありますのでご注意ください。

また、パイプラインモードの場合で、X方向分割数が17～32かつY方向分割数が16より大きい場合には、ペンディングされているコマンドを実行（パージ）してから濃度累積処理を行います。この条件ではパイプライン処理にならないので注意してください。

X分割数	Y分割数
1、2	1～512
3、4	1～256
5～8	1～128
9～16	1～64
17～32	1～32

X分割数	Y分割数
33～64	1～16
65～128	1～8
129～256	1～4
257～512	1、2
513～704	1



2.16.10 領域毎濃淡画像最大／最小濃度抽出 (IP_ProjectBlockGOMinMaxValue)

□ C言語API

```
int ret = IP_ProjectBlockGOMinMaxValue(IMGID ImgSrc, IPGOMinMaxTbl *Tbl,
                                         IPDivideTbl *DivideTbl);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IPGOMinMaxTbl	*Tbl	領域毎最小／最大濃度値データを返すアドレス テーブルサイズはIPGOMinMaxTbl型で領域分割 数 (X軸分割数×Y軸分割数) 用意して下さい。 (最大1024個)
IPDivideTbl	*DivideTbl	領域分割数を格納したアドレス

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	×

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)
IPGOMinMaxTbl	*Tbl	領域毎最小／最大濃度値データを格納したアドレス

□ パケットの構造

IPDivideTbl構造体については『IP_ProjectBlockG0()』コマンドを参照して下さい

```
typedef struct {
    int    MIN_LEVEL;    最小濃度値
    int    MAX_LEVEL;    最大濃度値
} IPGOMinMaxTbl;
```

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
21	領域毎最小／最大濃度値データ格納アドレス不正
22	領域分割数格納アドレス不正、または領域分割数が不正
30	ソース画面ウィンドウ設定エラー
48	動的メモリ確保エラー
49	ソース画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画像（濃淡）の画面領域を設定したパラメータで分割し、各領域毎の最小／最大濃度値抽出を行い、結果をユーザのテーブルへ返します。

ソース画面データタイプが2値の場合は、異常終了（エラーコード10）します。
画面分割数と処理領域の間に次の関係が成立しないと特徴量抽出で誤差が発生します。

- 条件(1) (画面分割数) * 1024 / (処理領域の長さ) が割り切れること
 (2) (処理領域の長さ) / (画面分割数) が割り切れること
 (3) (画面分割数) と (処理領域の長さ) が2のN乗であること

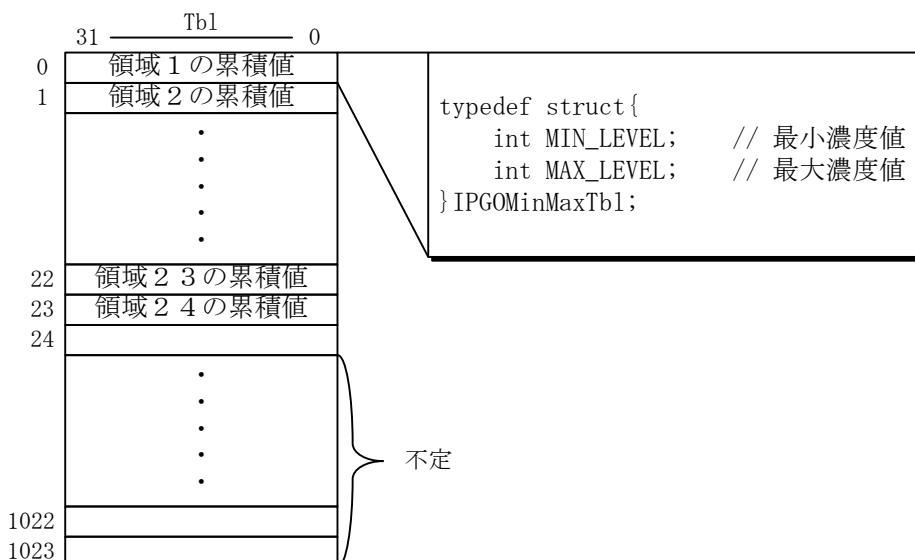
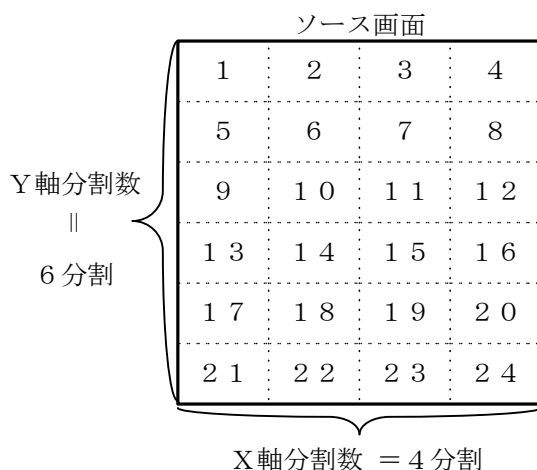
これは、『IP_ProjectBlockB0()』コマンド及び『IP_ProjectBlockG0()』コマンドを行う場合のヒストグラムメモリ（HM）への画像処理LSI内部のアドレス発生回路が固定小数点演算で行っており、上記の条件を満たさないとHMのアドレスに誤差が生じる為です。なお、『IP_ProjectBlockB0()』コマンド、『IP_ProjectBlockG0MinMaxValue()』コマンド、及び『IP_ProjectBlockG0()』コマンド以外では、このアドレッシングを使用しないので、他のコマンドには影響ありません。

X、Y軸の分割数はウィンドウの画素数を越えて設定できません。分割数の指定には以下の制約がありますのでご注意ください。

また、パイプラインモードの場合で、X方向分割数が17～32かつY方向分割数が16より大きい場合には、ペンディングされているコマンドを実行（パージ）してから最大／最小濃度抽出処理を行います。この条件ではパイプライン処理にならないので注意してください。

X分割数	Y分割数
1、2	1～512
3、4	1～256
5～8	1～128
9～16	1～64
17～32	1～32

X分割数	Y分割数
33～64	1～16
65～128	1～8
129～256	1～4
257～512	1、2
513～704	1



2.16.11 ラベル毎濃淡画像濃度累積（IP_ProjectLabelGO）

□ C言語API

```
int ret = IP_ProjectLabelGO(IMGID ImgSrc, IMGID ImgLabel, long *Tbl);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgLabel	ラベル画面番号
long	*Tbl	ラベル毎濃度累積値を返すアドレス テーブルはlong型で256個の容量を確保して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	×
ImgLabel	○	△R	○	○	○	×	○	△U	×

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）
long	*Tbl	ラベル毎濃度累積値を格納したアドレス

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
11	ラベル画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
22	ラベル毎濃度累積値格納アドレス不正
30	ソース画面ウィンドウ設定エラー
31	ラベル画面ウィンドウ設定エラー
49	ソース画面またはラベル画面が不定画面
-	エラーリセットコマンド（ClearIPError）未発行

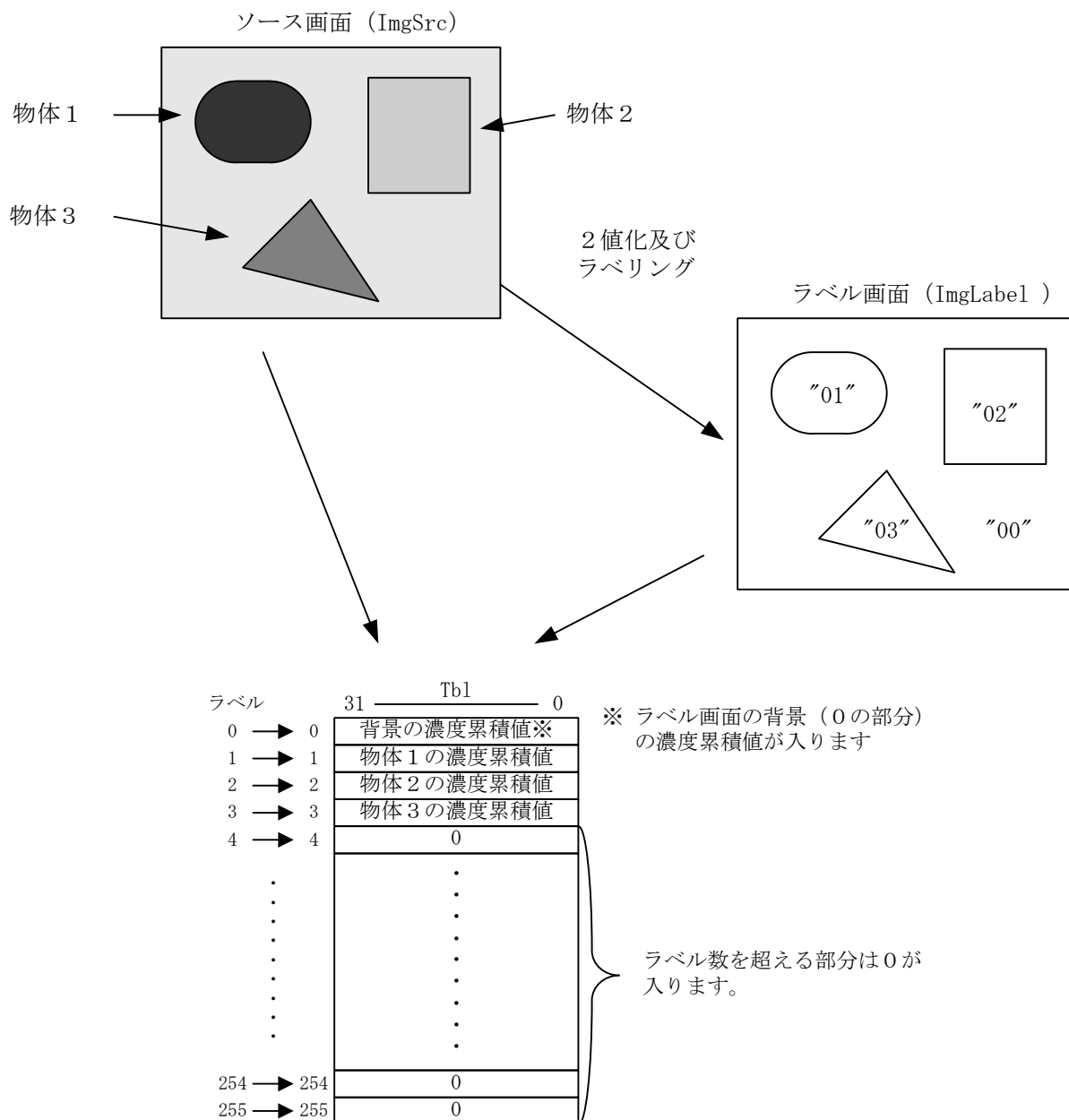
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画像（濃淡）をラベリング処理したラベル画面により、ラベル毎にソース画面の濃度累積を行い、結果をユーザのテーブルへ返します。

ソース画像、ラベル画像に適用されるウィンドウは、それぞれソース画面 1 のウィンドウ（SRC0_WIN）、ソース画面 2 のウィンドウ（SRC1_WIN）です。

ソース画面データタイプまたはラベル画面データタイプが 2 値の場合は異常終了し、それぞれのエラーコードは 1 0、1 3 になります。



2.16.12 ラベル毎濃淡画像最大／最小濃度抽出（IP_ProjectLabelGOMinMaxValue）

□ C言語API

```
int ret = IP_ProjectLabelGOMinMaxValue(IMGID ImgSrc, IMGID ImgLabel, IPGOMinMaxTbl *Tbl);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgLabel	ラベル画面番号
IPGOMinMaxTbl	*Tbl	ラベル毎最小／最大濃度値データを返すアドレス テーブルはIPGOMinMaxTbl型で256個の容量 を確保して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	×
ImgLabel	○	△R	○	○	○	×	○	△U	×

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）
IPGOMinMaxTbl	*Tbl	ラベル毎最小／最大濃度値データを格納したアドレス

□ パケットの構造

IPGOMinMaxTbl構造体については、『IP_ProjectBlockGOMinMaxValue()』コマンドを参照して下さい。

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
11	ラベル画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
22	ラベル毎最小／最大濃度値格納アドレス不正
30	ソース画面ウィンドウ設定エラー
31	ラベル画面ウィンドウ設定エラー
48	動的メモリ確保エラー
49	ソース画面またはラベル画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

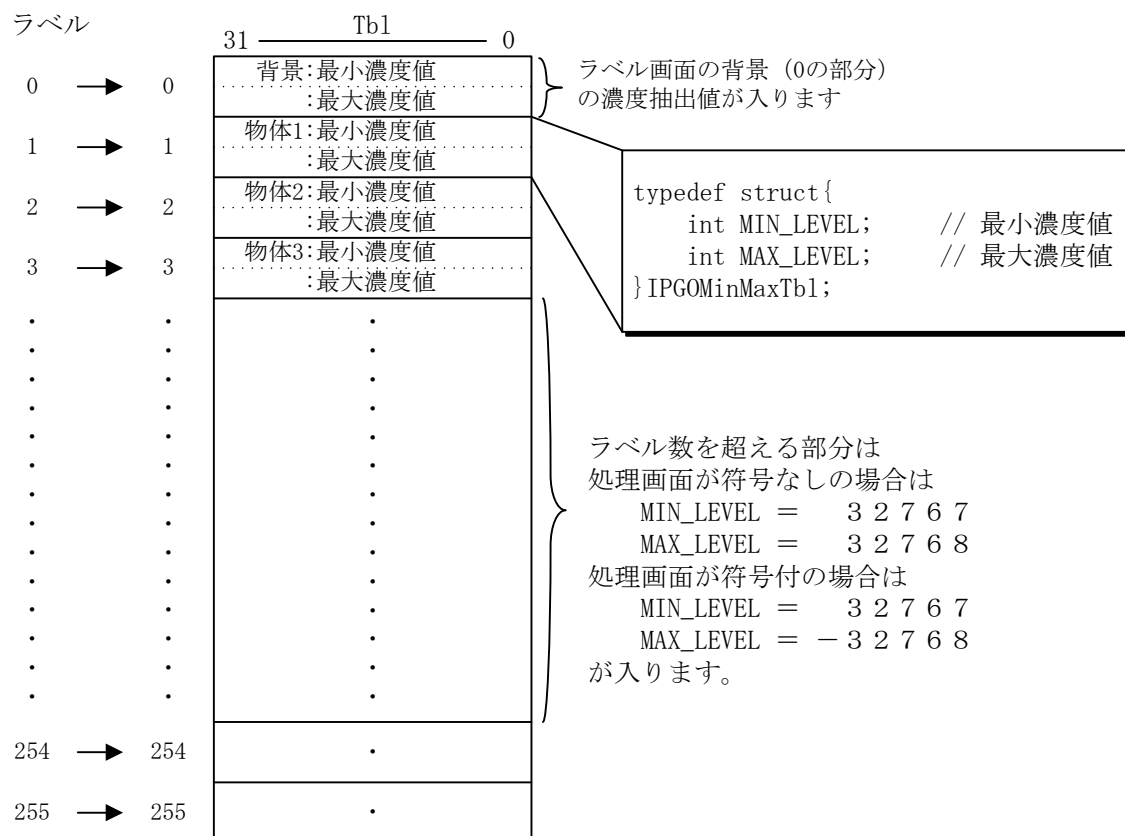
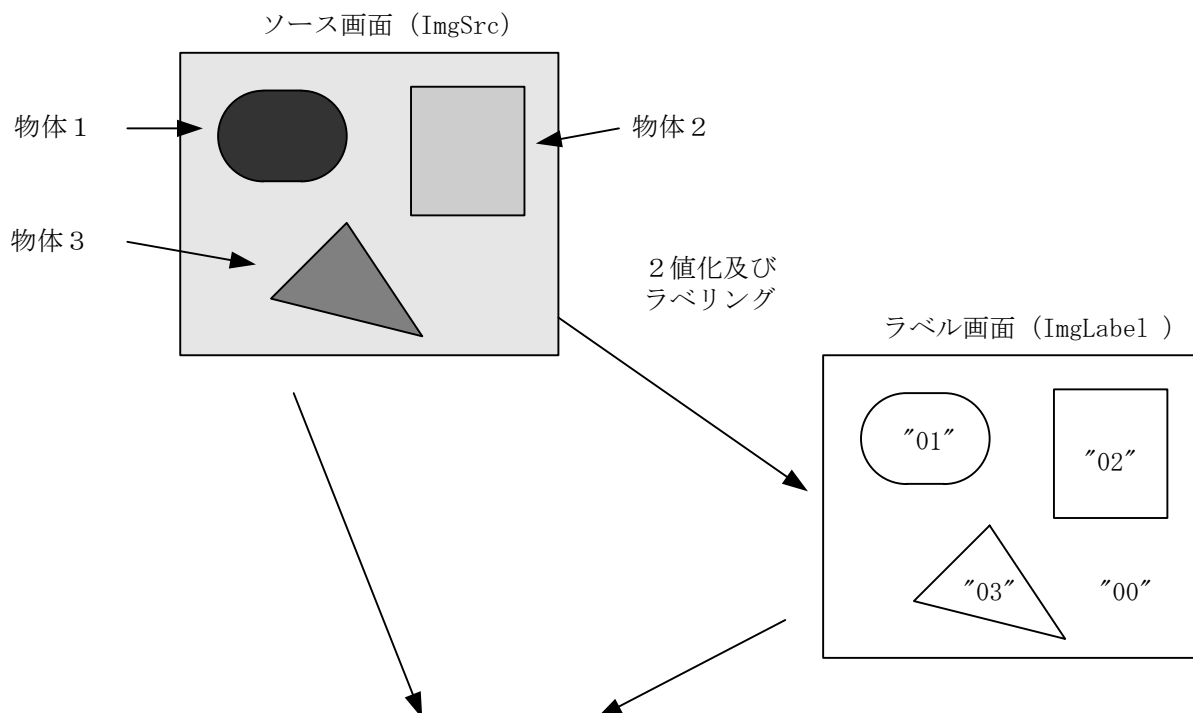
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画像（濃淡）をラベリング処理したラベル画面により、ラベル毎にソース画面の最小／最大濃度値抽出を行い、結果をユーザのテーブルへ返します。

ソース画像、ラベル画像に適用されるウィンドウは、それぞれソース画面 1 のウィンドウ（SRC0_WIN）、ソース画面 2 のウィンドウ（SRC1_WIN）です。

ソース画面データタイプまたはラベル画面データタイプが 2 値の場合は異常終了し、それぞれのエラーコードは 1 0、1 3 になります。



2.17 2値画像特徴量抽出

2.17.1 2値画像累積値抽出（ IP_ExtractBOArea ）

□ C言語API

int ret = IP_ExtractBOArea(IMGID ImgSrc , long *Area);

□ パラメータ

IMGID ImgSrc ソース画面番号

long *Area 2 値画像データの面積格納先アドレス

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

long *Area 2 値画像データの面積を格納したアドレス

□ エラーコード

1 0 画面空き領域、または範囲外（不当画面番号エラー）

— エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画像（2 値）に対し、2 値の面積累積値を抽出する。

2 値物体（白）が複数個あった場合には、それらの面積を累積した値になる。

□ 制限事項

ウィンドウサイズ1×1の画像メモリに対して本コマンドを発行した場合、正しい結果が得られません。

2.17.2 2値画像基本特徴量抽出 (IP_ExtractBOFeatures)

□ C言語API

```
int ret = IP_ExtractBOFeatures(IMGID ImgSrc, IPBOFeatureTbl *FtrTbl, enum IPBOFeatureOpt opt);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IPBOFeatureTbl	*FtrTbl	基本特徴量を返すアドレス
enum IPBOFeatureOpt	opt	基本特徴量オプション
	BINARY_REGISTER	(0) 基本特徴量のみ
	BINARY_CENTER_of_GRAVITY	(1) 基本特徴量+重心
	BINARY_MOMENT	(2) 基本特徴量+重心+角度 +各種モーメント

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)
IPBOFeatureTbl	*FtrTbl	基本特徴量を格納したアドレス

□ パケットの構造

```
typedef struct {
    int    LEFT_UP_POINT_X;      2 値物体 (白) が最初に出現する点の X 座標
    int    LEFT_UP_POINT_Y;      2 値物体 (白) が最初に出現する点の Y 座標
    int    RIGHT_DOWN_POINT_X;    2 値物体 (白) が最後に出現する点の X 座標
    int    RIGHT_DOWN_POINT_Y;    2 値物体 (白) が最後に出現する点の Y 座標
    long   AREA;                  2 値物体 (白) の面積
    int    REGION_X0;              2 値物体 (白) が存在する最小 X 座標
    int    REGION_Y0;              2 値物体 (白) が存在する最小 Y 座標
    int    REGION_X1;              2 値物体 (白) が存在する最大 X 座標
    int    REGION_Y1;              2 値物体 (白) が存在する最大 Y 座標
    float  CENTER_of_GRAVITY_X;    重心 X 座標
    float  CENTER_of_GRAVITY_Y;    重心 Y 座標
    float  ANGLE;                  慣性主軸と X 軸とのなす角度 (degree) (※)
    float  MOMENT_SIGMA_X;          X 軸への 1 次モーメント
    float  MOMENT_SIGMA_Y;          Y 軸への 1 次モーメント
    float  MOMENT_SIGMA_XX; X 軸への 2 次モーメント
    float  MOMENT_SIGMA_XY; 共役モーメント
    float  MOMENT_SIGMA_YY; Y 軸への 2 次モーメント
} IPBOFeatureTbl;
```

※ 基本特徴量とは LEFT_UP_POINT_X ~ REGION_Y1 までのデータを言います。

□ エラーコード

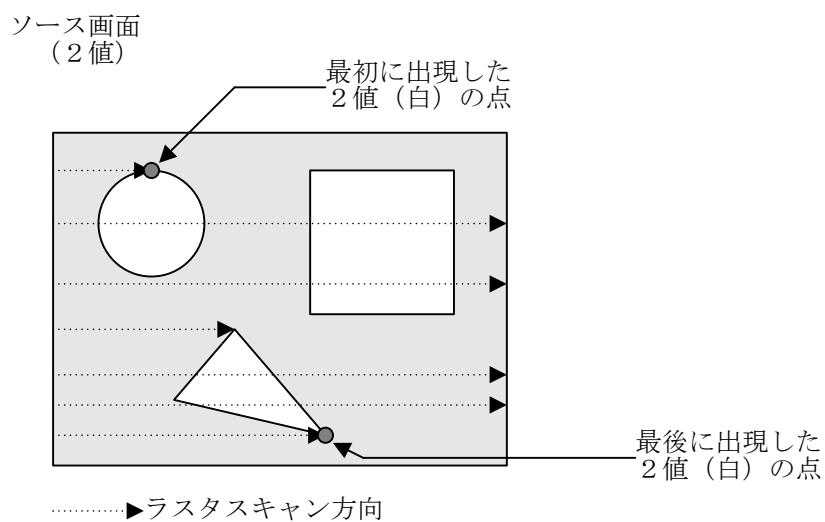
2	OS システムエラー
3	IMP 処理タイムアウト
8	VIN 入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
21	基本特徴量格納アドレス不正
22	オプション設定値範囲外
30	ソース画面ウィンドウ設定エラー
48	動的メモリ確保エラー
49	ソース画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

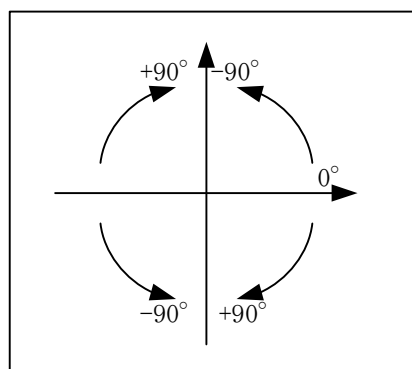
□ 機能

ソース画面（2値）に対し、基本特徴量抽出を行います。2値物体（白）が最初／最後に出現する点とは、ラスタスキャンした際に最初／最後に出現した2値（白）の点を指します。2値物体（白）の面積が0以外のときのみ、テーブル内データが有効です。また、重心、角度及び各種モーメントは、2値物体が1つのときのみ有効です。

ソース画面に濃淡画像を指定した場合は、濃淡画像を2値画像として処理します。画素データの最上位ビットが1の場合は白、0の場合は黒として処理します。



(※) IPB0FeatureTbl.ANGLEの値の説明



2.17.3 X/Y軸への投影(2値画像)(IP_ProjectBO)

□ C言語API

```
int ret = IP_ProjectBO(IMGID ImgSrc, int *TblX, int *TblY);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
int	*TblX	濃度累積値(X軸への投影)を返すアドレス テーブルサイズはint型で処理サイズのX方向、Y方向 の大きい方のサイズ分の容量を確保して下さい。
int	*TblY	濃度累積値(Y軸への投影)を返すアドレス テーブルサイズはint型で処理サイズのX方向、Y方向 の大きい方のサイズ分の容量を確保して下さい。

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○

□ リターンパラメータ

int	ret	正常終了(0)または異常終了(-1)
int	*TblX	濃度累積値(X軸への投影)を格納したアドレス
int	*TblY	濃度累積値(Y軸への投影)を格納したアドレス

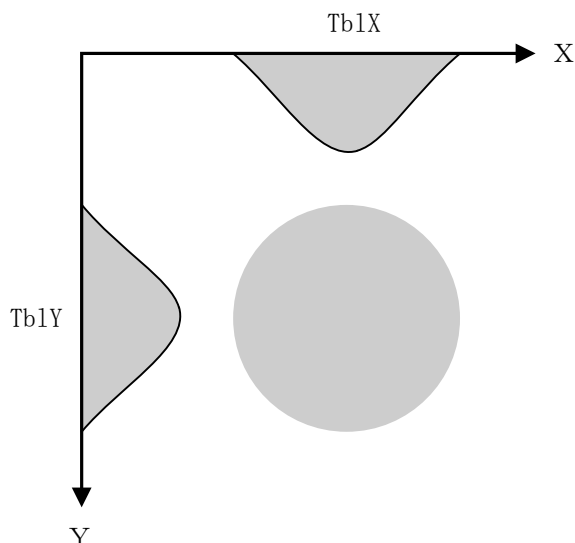
□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
21	濃度累積値(X軸への投影)格納アドレス不正
22	濃度累積値(Y軸への投影)格納アドレス不正
30	ソース画面ウィンドウ設定エラー
48	動的メモリ確保エラー
49	ソース画面が不定画面
—	エラーリセットコマンド(ClearIPError)未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

2値画像でのX/Y軸への投影を行い、物体（白）の濃度累積値を算出します。
書き込まれるテーブルのサイズ（範囲）は、ソース画面ウィンドウのサイズでX方向、Y方向の大きい方の範囲で書き込まれます。濃度累積値格納テーブルは大きい方のサイズにあわせて確保して下さい。このとき、本来のサイズを上回って結果が書き込まれる部分には0が書き込まれます。
ソース画面に濃淡画像を指定した場合は、濃淡画像を2値画像として処理します。画素データの最上位ビットが1の場合は白、0の場合は黒として処理します。



Tb1X、Tb1Yのフォーマットを下記に示します。

			Tb1X						Tb1Y		
X座標			31	0		Y座標			31	0	
0	→	0	濃度累積値			0	→	0	濃度累積値		
1	→	1	濃度累積値			1	→	1	濃度累積値		
.	.	.	.			.	.	.	.		
.	.	.	.			.	.	.	.		
.	.	.	.			.	.	.	.		
.	.	.	.			.	.	.	.		
.	.	.	.			.	.	.	.		
.	.	.	.			.	.	.	.		
.	.	.	.			.	.	.	.		
.	.	.	.			.	.	.	.		
1022	→	1022	濃度累積値			1022	→	1022	濃度累積値		
1023	→	1023	濃度累積値			1023	→	1023	濃度累積値		

本コマンドで得られるTb1X、Tb1Yの濃度累積値は、ソース画面ウィンドウからの相対座標の濃度累積値になります。

画面上の座標位置での濃度累積値はそれぞれ、

$$\begin{aligned} \text{X軸濃度累積値} &= \text{Tb1X} [\text{X} - \text{sx}] \\ \text{Y軸濃度累積値} &= \text{Tb1Y} [\text{Y} - \text{sy}] \end{aligned}$$

X : 画面X座標
Y : 画面Y座標
sx : ソース画面ウィンドウの始点X座標
sy : ソース画面ウィンドウの始点Y座標
となります。

2.17.4 領域X座標抽出 (Y軸へのMin／Max X投影) (IP_ProjectBORegionX)

□ C言語API

```
int ret = IP_ProjectBORegionX(IMGID ImgSrc, int *TblMinX, int *TblMaxX);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
int	*TblMinX	物体が存在するMinX座標を返すアドレス テーブルサイズはint型で処理サイズの大きさを確保して下さい。
int	*TblMaxX	物体が存在するMaxX座標を返すアドレス テーブルサイズはint型で処理サイズの大きさを確保して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)
int	*TblMinX	物体 (白) が存在するMinX座標を格納したアドレス
int	*TblMaxX	物体 (白) が存在するMaxX座標を格納したアドレス

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 1	MinX座標格納アドレス不正
2 2	MaxX座標格納アドレス不正
3 0	ソース画面ウィンドウ設定エラー
4 8	動的メモリ確保エラー
4 9	ソース画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

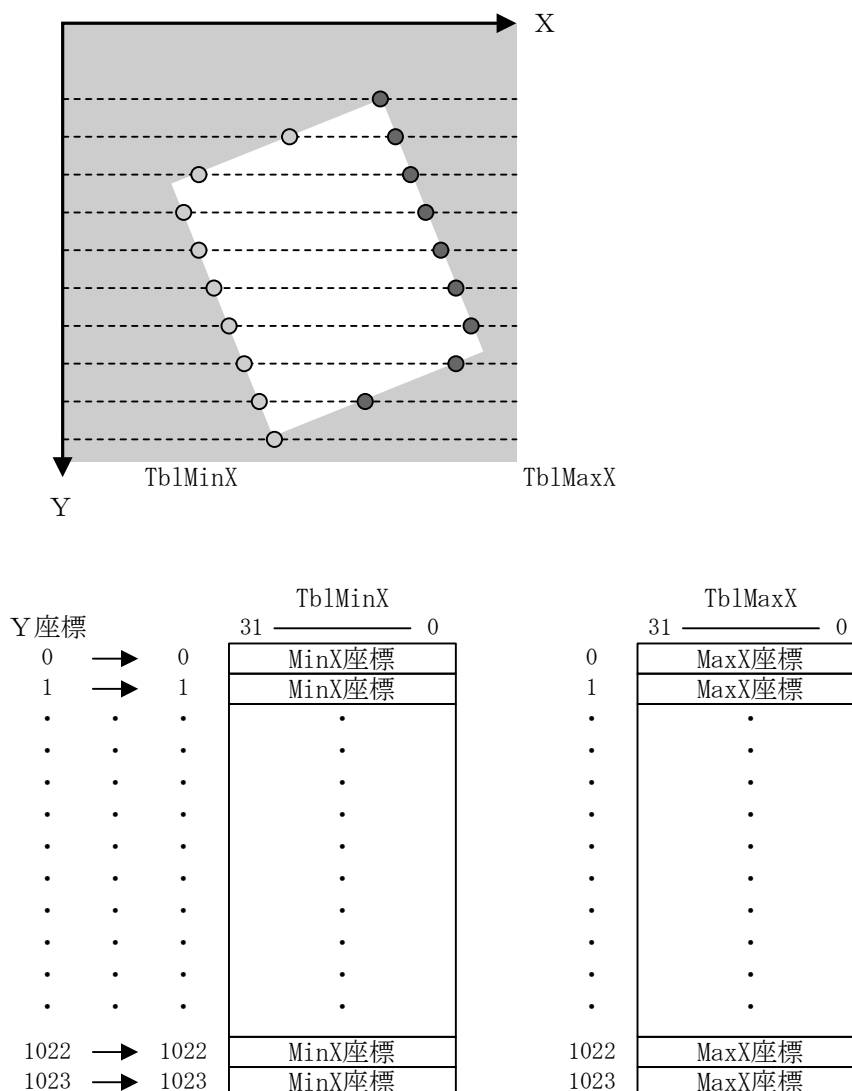
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

2 値画像での Y 軸への投影を行い、物体（白）が存在する Min/MaxX 座標の抽出を行います。

処理ウィンドウ範囲内で、物体（白）が存在しない Y 座標上の MinX 座標には 3 2 7 6 7 (0 x 7 F F F)、MaxX 座標には 0 を出力します。処理ウィンドウ範囲外の TblMinX、TblMaxX には、ともに不定値が出力されます。

ソース画面に濃淡画像を指定した場合は、濃淡画像を 2 値画像として処理します。画素データの最上位ビットが 1 の場合は白、0 の場合は黒として処理します。



本コマンドで得られる TblMinX、TblMaxX の座標値は、ソース画面ウィンドウからの相対座標になります。画面上の座標位置はそれぞれ、

$$\begin{aligned} \text{最小 X 座標} &= \text{TblMinX} [Y - sy] + sx \\ \text{最大 X 座標} &= \text{TblMaxX} [Y - sy] + sx \end{aligned}$$

Y : 画面 Y 座標
sx : ソース画面ウィンドウの始点 X 座標
sy : ソース画面ウィンドウの始点 Y 座標
となります。

2.17.5 領域Y座標抽出 (X軸へのMin/Max Y投影) (IP_ProjectBORegionY)

□ C言語API

```
int ret = IP_ProjectBORegionY(IMGID ImgSrc, int *TblMinY, int *TblMaxY);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
int	*TblMinY	物体が存在するMinY座標を返すアドレス テーブルサイズはint型で処理サイズの大きさを確保して下さい。
int	*TblMaxY	物体が存在するMaxY座標を返すアドレス テーブルサイズはint型で処理サイズの大きさを確保して下さい。

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)
int	*TblMinY	物体 (白) が存在するMinY座標を格納したアドレス
int	*TblMaxY	物体 (白) が存在するMaxY座標を格納したアドレス

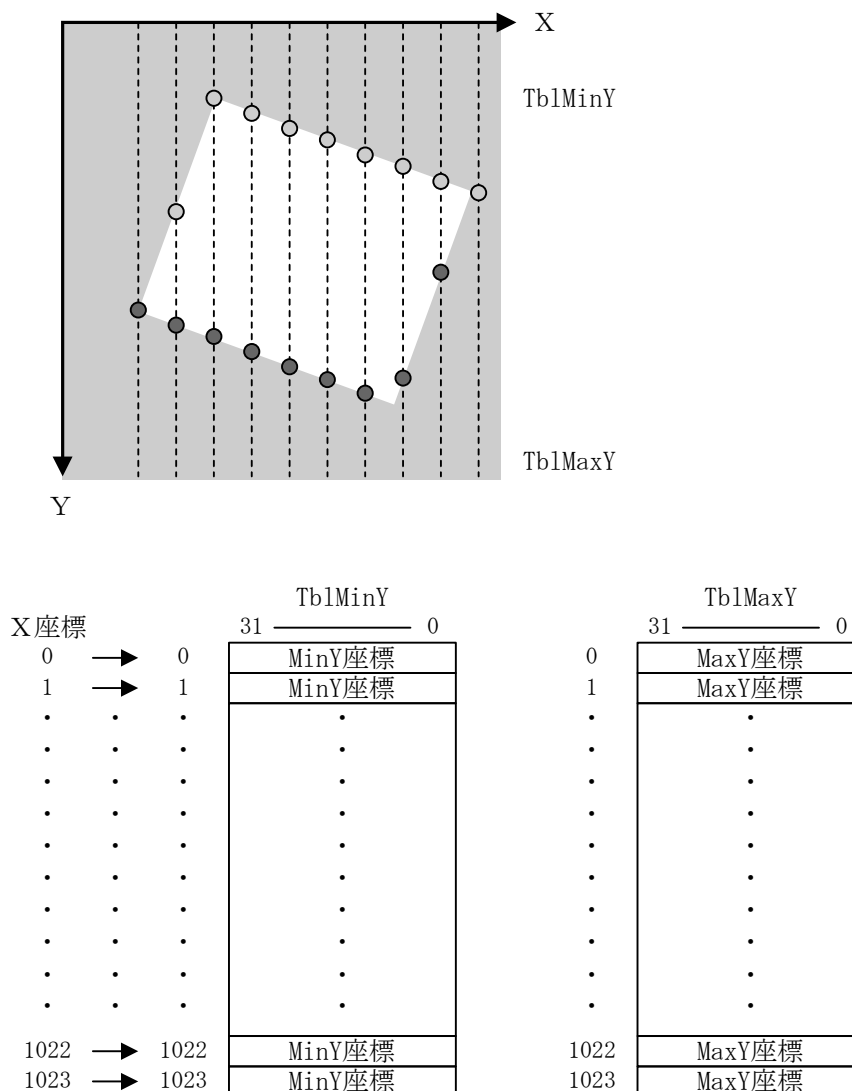
□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
21	MinY座標格納アドレス不正
22	MaxY座標格納アドレス不正
30	ソース画面ウィンドウ設定エラー
48	動的メモリ確保エラー
49	ソース画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

2値画像でのX軸への投影を行い、物体（白）が存在するMin/MaxY座標の抽出を行います。
 処理ウィンドウ範囲内で、物体（白）が存在しないX座標上のMinY座標には3 2 7 6 7（0 x 7 F F F）を、MaxY座標には0を出力します。処理ウィンドウ範囲外のTblMinY、TblMaxYには、ともに不定値が出力されます。
 ソース画面に濃淡画像を指定した場合は、濃淡画像を2値画像として処理します。画素データの最上位ビットが1の場合は白、0の場合は黒として処理します。



本コマンドで得られるTblMinY、TblMaxYの座標値は、ソース画面ウィンドウからの相対座標になります。画面上の座標位置はそれぞれ、

$$\begin{aligned} \text{最小Y座標} &= \text{TblMinY} [X - \text{sx}] + \text{sy} \\ \text{最大Y座標} &= \text{TblMaxY} [X - \text{sx}] + \text{sy} \end{aligned}$$

X : 画面X座標
 sx : ソース画面ウィンドウの始点X座標
 sy : ソース画面ウィンドウの始点Y座標
 となります。

2.17.6 領域毎2値画像濃度累積 (IP_ProjectBlockBO)

□ C言語API

```
int ret = IP_ProjectBlockBO(IMGID ImgSrc, long *Tbl, IPDivideTbl *DivideTbl);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
long	*Tbl	領域毎濃度累積値データを返すアドレス テーブルサイズはlong型で領域分割数 (X軸分割数×Y軸分割数) 用意して下さい (最大1024個)
IPDivideTbl	*DivideTbl	領域分割数を格納したアドレス

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)
long	*Tbl	領域毎濃度累積値データを格納したアドレス

□ パケットの構造

IPDivideTbl構造体については、『IP_ProjectBlockGO()』コマンドを参照して下さい。

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
21	領域毎濃度累積値データ格納アドレス不正
22	領域分割数格納アドレス不正、または領域分割数が不正
30	ソース画面ウィンドウ設定エラー
48	動的メモリ確保エラー
49	ソース画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画像（2値）の画面領域を設定したパラメータで分割し、各領域毎の濃度累積値をユーザのテーブルへ返します。

ソース画面に濃淡画像を指定した場合は、濃淡画像を2値画像として処理します。画素データの最上位ビットが1の場合は白、0の場合は黒として処理します。

画面分割数と処理領域の間に以下の関係が成立しないと特徴量抽出で誤差が発生します。

- 条件(1) (画面分割数) * 1024 / (処理領域の長さ) が割り切れること
- (2) (処理領域の長さ) / (画面分割数) が割り切れること
- (3) (画面分割数) と (処理領域の長さ) が2のN乗であること

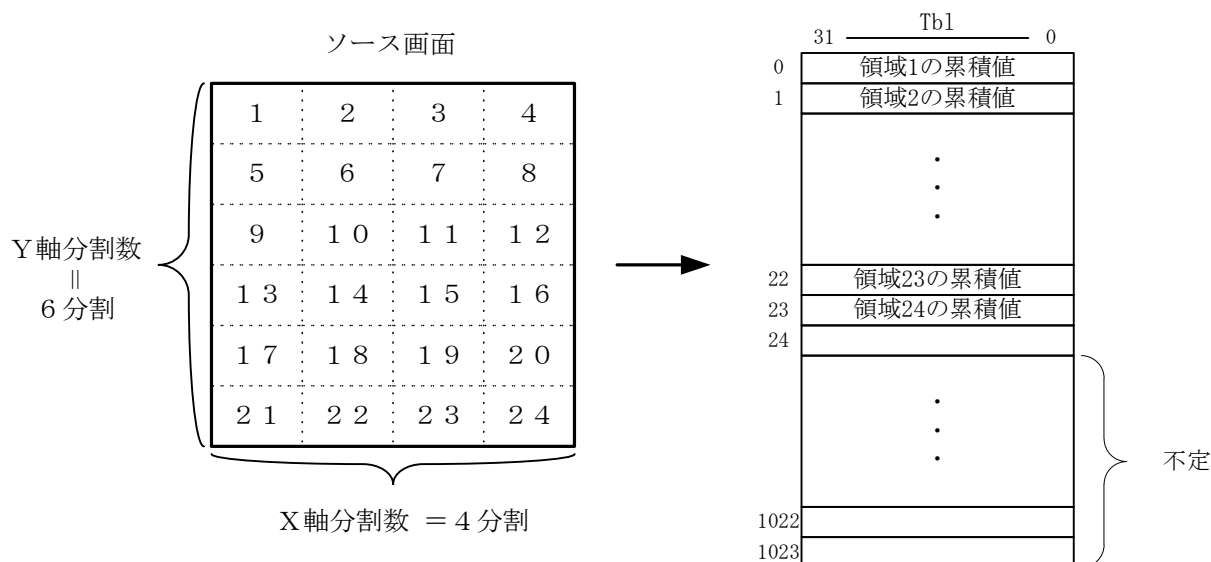
これは、『IP_ProjectBlockB0()』コマンド及び『IP_ProjectBlockG0()』コマンドを行う場合のヒストグラムメモリ(HM)への画像処理LSI内部のアドレス発生回路が固定小数点演算で行っており、上記の条件を満たさないとHMのアドレスに誤差が生じる為です。なお、『IP_ProjectBlockB0()』コマンド、『IP_ProjectBlockG0MinMaxValue()』コマンド、及び『IP_ProjectBlockG0()』コマンド以外は、このアドレッシングを使用しないので影響ありません。

X、Y軸の分割数はウィンドウの画素数を越えて設定できません。分割数の指定には以下の制約がありますのでご注意ください。

また、パイプラインモードの場合で、X方向分割数が17～32かつY方向分割数が16より大きい場合には、ペンディングされているコマンドを実行（パージ）してから濃度累積処理を行います。この条件ではパイプライン処理にならないので注意してください。

X分割数	Y分割数
1、2	1～512
3、4	1～256
5～8	1～128
9～16	1～64
17～32	1～32

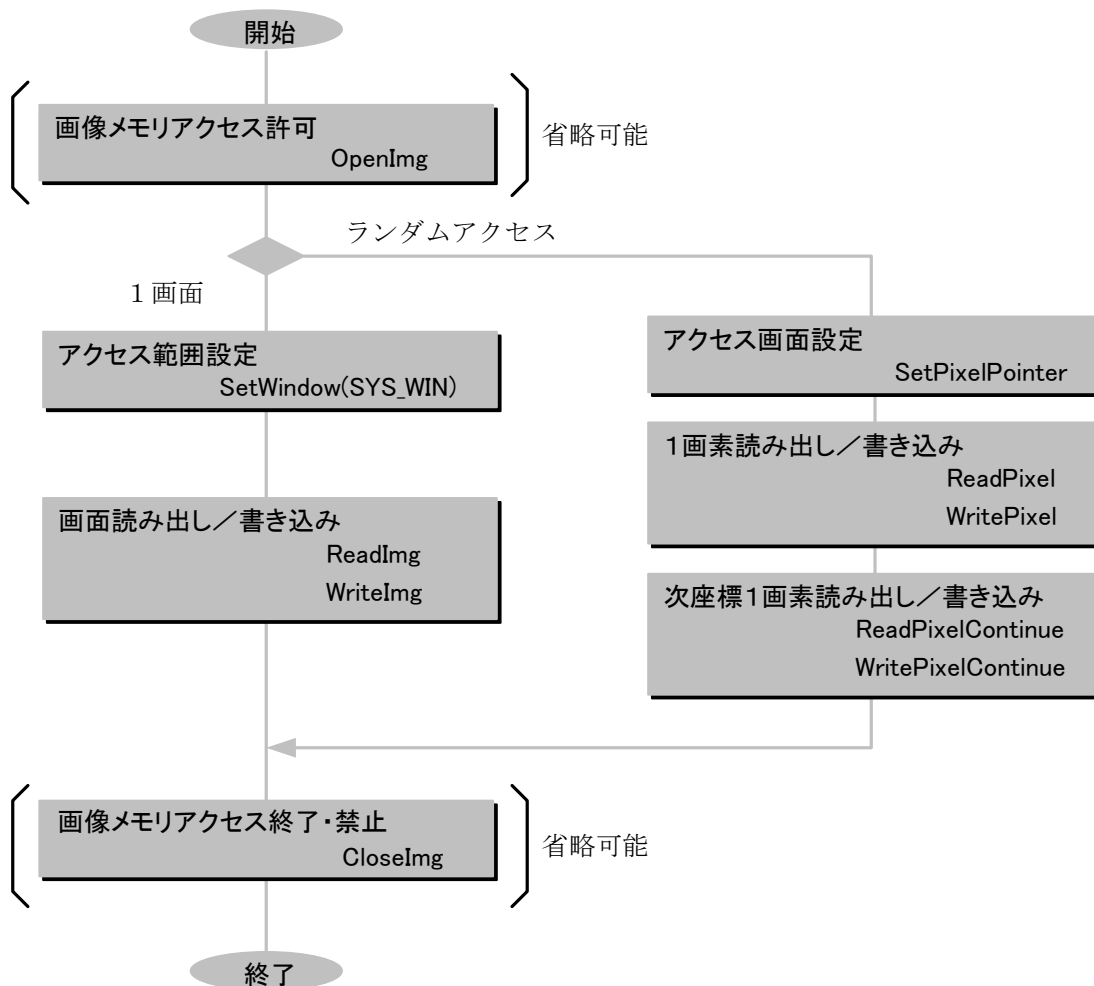
X分割数	Y分割数
33～64	1～16
65～128	1～8
129～256	1～4
257～512	1、2
513～704	1



2.18 画像メモリアクセス

画像メモリアクセスコマンドでは、画像メモリに対するアクセスが可能です。また、画像メモリにアクセスを行う場合は、下記の手順に従って下さい。

画像メモリアクセス手順



画像メモリはUMA方式でオンボードCPUからダイレクトに画像メモリをアクセスできます。詳細は、`OpenImgDirect()`、`CloseImgDirect()` コマンドを参照して下さい。

2.18.1 画像メモリアクセス可能（OpenImg）

□ C言語API

```
int ret = OpenImg(enum WaitMode mode);
```

□ パラメータ

enum WaitMode	mode	ウェイトモード	
		WAIT_FOREVER	(0) オープン処理実行
		NOWAIT	(1) オープン処理実行

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（1）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
20	ウェイトモード設定値範囲外
-	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

画像メモリをアクセス可能状態にします。
 本コマンドは、従来のIPコマンドとの互換性の為に残されているものであり、省略しても画像メモリへアクセス可能です。

2.18.2 画像メモリアクセス可能 (指定画面) (OpenImgExt)

□ C言語API

```
int ret = OpenImgExt(IMGID ImgSrc, enum WaitMode mode);
```

□ パラメータ

IMGID	ImgSrc	画面番号		
enum WaitMode	mode	ウェイトモード		
		WAIT_FOREVER	(0)	オープン処理実行
		NOWAIT	(1)	オープン処理実行

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	—	—	—	—	—	○	○	○

□ リターンパラメータ

int	ret	正常終了 (1) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
21	ウェイトモード設定値範囲外
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

パラメータで指定された画面番号の画像メモリをアクセス可能状態にします。
 本コマンドは、従来のIPコマンドとの互換性の為に残されているものであり、省略しても画像メモリへアクセス可能です。

2.18.3 画像メモリアクセス不可 (CloseImg)

□ C言語API

```
int ret = CloseImg();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

全画像メモリをアクセス不可状態にします。
本コマンドは、従来のIPコマンドとの互換性の為に残されているものであり、省略しても画像メモリへアクセスには影響がありません。

2.18.4 画像メモリ読み出し(指定画素数) (ReadImg)

□ C言語API

```
long ret = ReadImg(IMGID ImgSrc, unsigned char *ImgTbl, long count);
```

□ パラメータ

IMGID	ImgSrc	画面番号
unsigned char	*ImgTbl	読み出しデータを返すアドレス
long	count	画素数

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	○	○	○	○

□ リターンパラメータ

long	ret	読み出したデータ数 (1 以上) または異常終了 (- 1)
unsigned char	*ImgTbl	読み出しデータを格納したアドレス

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 1	読み出しデータを返すアドレスが不正
2 2	読み出し画素数範囲外
3 4	画像メモリアクセスウィンドウ設定エラー
4 9	読み出しを行う画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

パラメータで指定された画面番号の画像メモリから指定画素数分読み出します。指定画素数が画面（ウィンドウ）を超える場合は、越える直前までのデータを読み出します。画像メモリからの読み出しサイズは、ウィンドウが有効の時は画像メモリアクセスウィンドウサイズ（SYS_WIN）、無効の時は映像入出力サイズ（ビデオフレームサイズ）に従います。RGB16画面の読み出しは、16bit/pixel単位で行います。読み出しデータ格納配列（ImgTbl）は、count×2バイト分確保して下さい。RGB画面は、R、G、Bの画面をそれぞれY画面として読み出します。RGB画面を読み出し画面として指定した場合には、R画面から指定画素数(count)読み出します。G、B画面のデータを読み出すには、GetGImgID、GetBImgIDコマンドでそれぞれの画面IDを取得してY画面として読み出してください。

また、パラメータで指定された画面番号の画像メモリは、X方向画面サイズ ≥ 32、32バイトアラインの制限がありますので、ご注意ください。

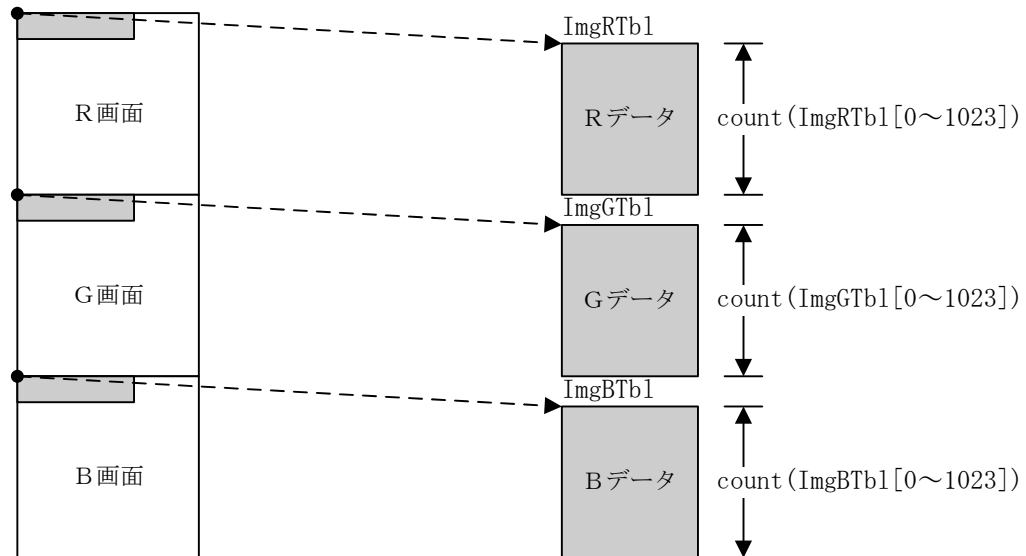
[R G B 画面読み出し例]

```

IMGID      ImgRGB, ImgG, ImgB;
unsigned char  ImgRTbl[256*4];          // X=256, Y=4の領域をR画面用に確保
unsigned char  ImgGTbl[256*4];          // X=256, Y=4の領域をG画面用に確保
unsigned char  ImgBTbl[256*4];          // X=256, Y=4の領域をB画面用に確保
long          count = 256*4;            // 読み出す画素数 (1画面分)
long          ret;

ImgRGB = AllocRGBImg( IMG_FS_512H_512V );
ImgG = GetGImgID( ImgRGB );
ImgB = GetBImgID( ImgRGB );
...
SetWindow( SYS_WIN, 0, 0, 255, 3 );    // 読み出しサイズを設定
ret = ReadImg( ImgRGB, ImgRTbl, count );
ret = ReadImg( ImgG,  ImgGTbl, count );
ret = ReadImg( ImgB,  ImgBTbl, count );

```



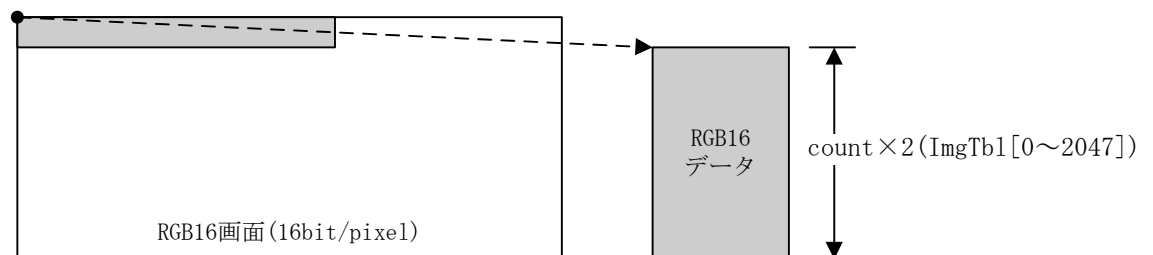
[R G B 1 6 画面読み出し例]

```

IMGID      ImgRGB16;
unsigned char  ImgTbl[256*2*4];          // X=256×2, Y=4の領域を確保
long          count = 256*4;            // 読み出す画素数 (16bit/pixel)
long          ret;

ImgRGB16 = AllocRGB16Img( IMG_FS_512H_512V );
...
SetWindow( SYS_WIN, 0, 0, 255, 3 );    // 読み出しサイズを設定
ret = ReadImg( ImgRGB16, ImgTbl, count );

```



2.18.5 画像メモリ書き込み(指定画素数)(WriteImg)

□ C言語API

```
long ret = WriteImg(IMGID ImgDst, unsigned char *ImgTbl, long count);
```

□ パラメータ

IMGID	ImgDst	画面番号
unsigned char	*ImgTbl	書き込みデータを格納したアドレス
long	count	画素数

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	○	□	□	□

□ リターンパラメータ

long	ret	書き込んだデータ数（1以上）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
2 1	書き込みデータを格納したアドレスが不正
2 2	書き込み画素数範囲外
3 4	画像メモリアクセスウィンドウ設定エラー
—	エラーリセットコマンド（ClearIPErrors）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

パラメータで指定された画面番号の画像メモリに指定画素数分書き込みます。指定画素数が画面（ウィンドウ）を超える場合は、越える直前までのデータが書き込まれます。画像メモリへの書き込みサイズは、ウィンドウが有効の時は画像メモリアクセスウィンドウサイズ（SYS_WIN）、無効の時は画面サイズに従います。

R G B 1 6 画面の書き込みは、1 6 bit/pixel単位で行います。書き込みデータ格納配列(ImgTbl)は、count×2バイト分確保して下さい。

R G B 画面は、R、G、Bの画面をそれぞれY画面として書き込みます。R G B 画面を書き込み画面として指定した場合には、R画面に指定画素数(count)分書き込みます。G、B画面のデータを書き込むには、G e t G I m g I D、G e t B I m g I D コマンドでそれぞれの画面 I D を取得してY画面として書き込んでください。

また、パラメータで指定された画面番号の画像メモリは、X方向画面サイズ >= 3 2、3 2 バイトアラインの制限がありますので、ご注意ください。

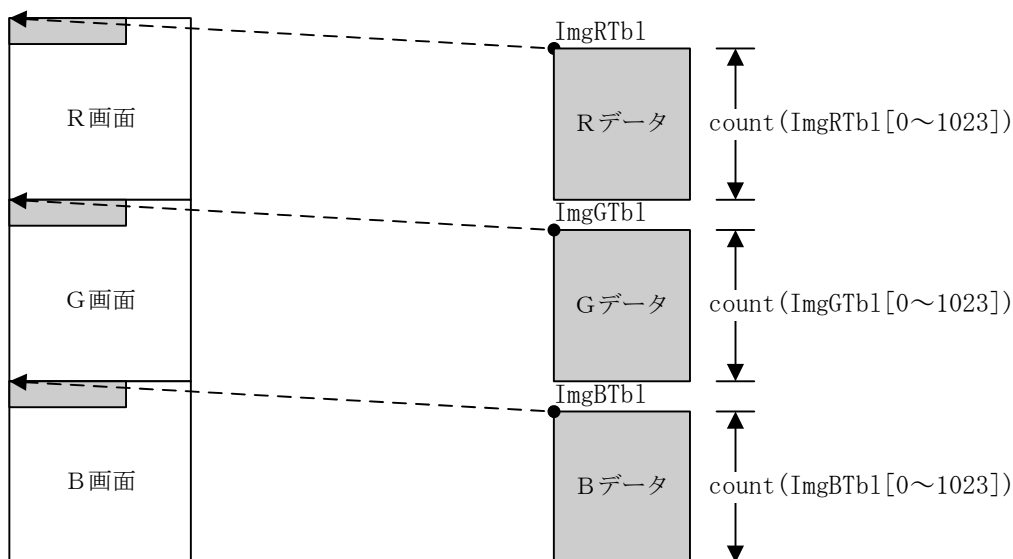
[R G B画面書込み例]

```

IMGID      ImgRGB, ImgG, ImgB;
unsigned char  ImgRTbl[256*4];           // X=256, Y=4の領域をR画面用に確保
unsigned char  ImgGTbl[256*4];           // X=256, Y=4の領域をG画面用に確保
unsigned char  ImgBTbl[256*4];           // X=256, Y=4の領域をB画面用に確保
long          count = 256*4;             // 読み出す画素数 (1画面分)
long          ret;

ImgRGB = AllocRGBImg( IMG_FS_512H_512V );
ImgG = GetGImgID( ImgRGB );
ImgB = GetBImgID( ImgRGB );
...
SetWindow( SYS_WIN, 0, 0, 255, 3 );      // 読み出しサイズを設定
ret = WriteImg( ImgRGB, ImgRTbl, count );
ret = WriteImg( ImgG, ImgGTbl, count );
ret = WriteImg( ImgB, ImgBTbl, count );

```



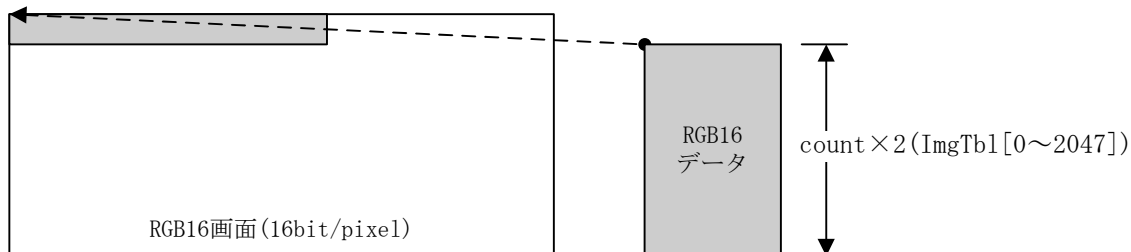
[R G B 1 6画面書込み例]

```

IMGID      ImgRGB16;
unsigned char  ImgTbl[256*2*4];           // X=256×2, Y=4の領域を確保
long          count = 256*4;             // 書き込む画素数 (16bit/pixel)
long          ret;

ImgRGB16 = AllocRGB16Img( IMG_FS_512H_512V );
...
SetWindow( SYS_WIN, 0, 0, 255, 3 );      // 書き込みサイズを設定
ret = WriteImg( ImgRGB16, ImgTbl, count );

```



2.18.6 画像メモリ1画素アクセス画面設定 (SetPixelPointer)

□ C言語API

```
int ret = SetPixelPointer(IMGID ImgSrc);
```

□ パラメータ

IMGID ImgSrc 画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	○	○	○	○

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

2 OSシステムエラー
3 IMP処理タイムアウト
8 VIN入力タイムアウト
1 0 画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
3 4 画像メモリアクセスウィンドウ設定エラー
- エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

画像メモリに1画素アクセスする為の初期設定をします。本コマンドにより、画像メモリ制御コマンド (『ReadPixel()』、『WritePixel()』、『ReadPixelContinue()』、『WritePixelContinue()』) の対象となる画面番号と処理領域を設定します。処理領域は画像メモリアクセスウィンドウ (SYS_WIN) となります。本コマンドを実行する前に画像メモリアクセスウィンドウを設定して下さい。

2.18.7 画像メモリ読み出し(座標指定、1画素) (ReadPixel)

□ C言語API

```
int ret = ReadPixel(int x, int y, char *data);
```

□ パラメータ

int	x	X座標 (ウィンドウ相対値)
int	y	Y座標 (ウィンドウ相対値)
char	*data	読み出しデータを返すアドレス

□ 対応画面タイプ/画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)
char	*data	読み出しデータを格納したアドレス

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	画像メモリ1画素アクセス画面に設定した画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
20	X座標設定範囲外
21	Y座標設定範囲外
22	読み出しデータを返すアドレスが不正
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

パラメータで指定された座標から1画素読み出します。本コマンドを実行する前に『SetPixelPointer()』コマンドで処理対象とする画面番号と処理領域を設定する必要があります。画像メモリ制御コマンドでの座標系はウィンドウ相対であり、画像メモリアクセスウィンドウ (SYS_WIN) の始点が座標 (0, 0) になります。設定されているウィンドウ外の座標が指定された場合はエラーとなります。

2.18.8 画像メモリ書き込み(座標指定、1画素) (WritePixel)

□ C言語API

```
int ret = WritePixel(int x, int y, char data);
```

□ パラメータ

int	x	X座標 (ウィンドウ相対値)
int	y	Y座標 (ウィンドウ相対値)
char	data	書き込みデータ

□ 対応画面タイプ/画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	画像メモリ1画素アクセス画面に設定した画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 0	X座標設定範囲外
2 1	Y座標設定範囲外
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

パラメータで指定された座標へ1画素書き込みます。本コマンドを実行する前に『SetPixelPointer()』コマンドで処理対象とする画面番号と処理領域を設定する必要があります。

画像メモリ制御コマンドでの座標系はウィンドウ相対であり、画像メモリアクセスウィンドウ (SYS_WIN) の始点が座標 (0, 0) になります。

設定されているウィンドウ外の座標が指定された場合はエラーとなります。

2.18.9 画像メモリ読み出し(次座標、1画素) (ReadPixelContinue)

□ C言語API

```
int ret = ReadPixelContinue(char *data);
```

□ パラメータ

char	*data	読み出しデータを返すアドレス

□ 対応画面タイプ/画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)
char	*data	読み出しデータを格納したアドレス

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	画像メモリ1画素アクセス画面に設定した画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 0	読み出しデータを返すアドレスが不正
4 4	座標が画面 (ウィンドウ) 外
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

前回読み出した次座標から1画素読み出します。

『ReadPixelContinue()』コマンドは、『ReadPixel()』コマンドを最初に実行し、座標を設定してから実行して下さい。『ReadPixel()』コマンドを実行せずに実行すると、読み出し座標が不定のままである為、処理結果は不定です。

また、『ReadPixelContinue()』コマンドと『WritePixelContinue()』コマンドを混在して使用しないで下さい。読み出し座標のポインタは、『ReadPixelContinue()』コマンドと『WritePixelContinue()』コマンドで共通な為、『WritePixelContinue()』コマンドを実行すると、座標ポインタがインクリメントされ、次に『ReadPixelContinue()』コマンドを実行すると、座標ポインタは『WritePixelContinue()』コマンドでインクリメントされた座標になります。

このコマンドは画面サイズ (ウィンドウサイズ) によって座標を更新しますが、座標が画面サイズ (ウィンドウサイズ) を超えた場合はエラーになります。

2.18.10 画像メモリ書き込み（次座標、1画素）（WritePixelContinue）

□ C言語API

```
int ret = WritePixelContinue(char data);
```

□ パラメータ

char	data	書き込みデータ

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	画像メモリ1画素アクセス画面に設定した画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
4 4	座標が画面（ウィンドウ）外
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

前回書き込んだ次座標へ1画素書き込みます。

『WritePixelContinue()』コマンドは、『WritePixel()』コマンドを最初に実行し、座標を設定してから実行して下さい。『WritePixel()』コマンドを実行せずに実行すると、書き込み座標が不定のままである為、処理結果は不定です。

また、『WritePixelContinue()』コマンドと『ReadPixelContinue()』コマンドを混在して使用しないで下さい。書き込み座標のポインタは、『WritePixelContinue()』コマンドと『ReadPixelContinue()』コマンドで共通な為に、『ReadPixelContinue()』コマンドを実行すると、座標ポインタがインクリメントされ、次に『WritePixelContinue()』コマンドを実行すると、座標ポインタは『WritePixelPointer()』コマンドでインクリメントされた座標になります。

このコマンドは画面サイズ（ウィンドウサイズ）によって座標を更新しますが、座標が画面サイズ（ウィンドウサイズ）を超えた場合はエラーになります。

2.18.11 画像メモリのリフレッシュ（ RefreshImg ）

□ C言語API

```
int ret = RefreshImg(IMGID ImgSrc);
```

□ パラメータ

IMGID ImgSrc 画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	○	○	○	○	○	○	○	○

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2 OSシステムエラー
3 IMP処理タイムアウト
8 VIN入力タイムアウト
10 画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
49 リフレッシュ対象画面が不定画面
- エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

リフレッシュ対象画面の最新画像データがSHのキャッシュ上に存在する場合、最新画像データを画像メモリへ書き出します。

図形描画コマンド、文字描画コマンドでは、描画データがSHのキャッシュに一時的に蓄えられる為、実際の画像メモリには書き込まれていない部分が出てきます。その為、描画が全て終了した時点で、本コマンドによりキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。なお、リフレッシュ対象画面のデータタイプは変更されません。

2.18.12 画像メモリ直接アクセス可能 (OpenImgDirect)

□ C言語API

```
int ret = OpenImgDirect(IMGID ImgSrc, int *xsize, int *ysize, char **address);
```

□ パラメータ

IMGID	ImgSrc	オープン対象の画面番号
int	*xsize	オープン対象画面のX方向の長さを返すアドレス
int	*ysize	オープン対象画面のY方向の長さを返すアドレス
char	**address	オープン対象画面の先頭アドレスを返すアドレス

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	○	○	○	○	○	○	○	○

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)
int	*xsize	オープン対象画面のX方向の長さを格納したアドレス
int	*ysize	オープン対象画面のY方向の長さを格納したアドレス
char	**address	オープン対象画面の先頭アドレスを格納したアドレス

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	オープン対象画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 1	オープン対象画面のX方向の長さを返すアドレスが不正
2 2	オープン対象画面のY方向の長さを返すアドレスが不正
2 3	オープン対象画面の先頭アドレスを返すアドレスが不正
4 9	オープン対象画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

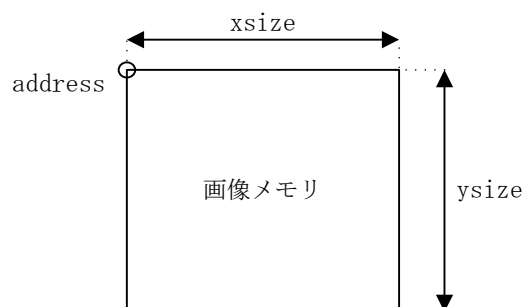
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

本コマンドは $xsize * ysize * \text{対象画面の画素サイズ}$ の領域をシステムが管理している領域に確保し、ここにオープンした画面の画像メモリのデータを格納します。返されるアドレスはこの領域の先頭アドレスになります。

データの変更後、『CloseImgDirect()』を実行してください。

※ 『CloseImgDirect()』コマンドを実行しないと確保した領域の解放と変更データが画像メモリに反映されませんので必ず『CloseImgDirect()』コマンドを実行してください。



2.18.13 画像メモリ直接アクセス不可 (CloseImgDirect)

□ C言語API

```
int ret = CloseImgDirect(IMGID ImgSrc);
```

□ パラメータ

IMGID ImgSrc クローズ対象の画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	○	○	○	○	○	○	○	○

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

2 OSシステムエラー
3 IMP処理タイムアウト
8 VIN入力タイムアウト
1 0 クローズ対象画面番号に対応する画面が未確保、または番号が有効範囲外
 (不当画面番号エラー)
4 9 クローズ対象画面が不定画面
- エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

『OpenImgDirect()』コマンドで確保した領域の画面データをパラメータで指定された画面番号の画像メモリに書き戻し、確保領域を解放します。
アクセスを終了する場合は、本コマンドを必ず実行してください。

2.19 2値パイプラインフィルタ

2.19.1 形状変換パイプライン処理設定 (SetTrsPipelineFLTMode)

□ C言語API

```
int ret = SetTrsPipelineFLTMode(enum PipelineMode pmode, PipelineStageMode *fmode);
```

□ パラメータ

enum PipelineMode pmode パイプライン処理出力段数／演算設定

PIPELINE0	(0)	パイプライン処理 1 段目出力
PIPELINE1	(1)	パイプライン処理 2 段目出力
PIPELINE2	(2)	パイプライン処理 3 段目出力
PIPELINE3	(3)	パイプライン処理 4 段目出力
PIPELINE4	(4)	パイプライン処理 5 段目出力
PIPELINE5	(5)	パイプライン処理 6 段目出力
PIPELINE6	(6)	パイプライン処理 7 段目出力
PIPELINE7	(7)	パイプライン処理 8 段目出力
PIPELINEAND	(8)	パイプライン処理 4 段目AND出力 (※)
PIPELINEOR	(9)	パイプライン処理 4 段目OR出力 (※)
PIPELINEXOR	(10)	パイプライン処理 4 段目Xor出力 (※)
PIPELINEXNOR	(11)	パイプライン処理 4 段目Xnor出力 (※)

※ 8 段あるパイプラインの処理を 4 段×2 並列の処理構成に変更し、4 段パイプライン処理を行った結果をAND、OR、Xor、Xnor 処理します。

PipelineStageMode *fmode パイプライン処理機能／内容を格納したアドレス

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ パケットの構造

```
typedef struct {
    int    Stage0;    パイプライン処理 1 段目処理機能
    int    Stage1;    パイプライン処理 2 段目処理機能
    int    Stage2;    パイプライン処理 3 段目処理機能
    int    Stage3;    パイプライン処理 4 段目処理機能
    int    Stage4;    パイプライン処理 5 段目処理機能
    int    Stage5;    パイプライン処理 6 段目処理機能
    int    Stage6;    パイプライン処理 7 段目処理機能
    int    Stage7;    パイプライン処理 8 段目処理機能
} PipelineStageMode;
```

□ エラーコード

2 0	パイプライン処理出力段数／演算設定値範囲外
2 1	パイプライン処理機能／内容の設定値範囲外、またはアドレス不正
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

機能

形状変換パイプライン（8段2値フィルタ）の処理内容を設定します。
各処理演算内容をStageFunction列挙型から選択し、PipelineStageMode構造体に設定して下さい。

• StageFunction列挙型

STAGE_NOOP	(0)	処理無し
STAGE_NOOP1	(1)	処理無し
DELETERNOISE4	(2)	ノイズ除去4連結
DELETERNOISE8	(3)	ノイズ除去8連結
DILATION4	(4)	4連結膨張
DILATION8	(5)	8連結膨張
EROSION4	(6)	4連結収縮
EROSION8	(7)	8連結収縮
OUTLINE4	(8)	4連結輪郭抽出
OUTLINE8	(9)	8連結輪郭抽出
DILATION4X	(10)	4連結X方向膨張
DILATION8X	(11)	8連結X方向膨張
EROSION4X	(12)	4連結X方向収縮
EROSION8X	(13)	8連結X方向収縮
DILATION4Y	(14)	4連結Y方向膨張
DILATION8Y	(15)	8連結Y方向膨張
EROSION4Y	(16)	4連結Y方向収縮
EROSION8Y	(17)	8連結Y方向収縮
DILATION4RU	(18)	4連結右上り方向膨張
DILATION8RU	(19)	8連結右上り方向膨張
EROSION4RU	(20)	4連結右上り方向収縮
EROSION8RU	(21)	8連結右上り方向収縮
DILATION4RD	(22)	4連結右下り方向膨張
DILATION8RD	(23)	8連結右下り方向膨張
EROSION4RD	(24)	4連結右下り方向収縮
EROSION8RD	(25)	8連結右下り方向収縮

フィルタ処理の注目画素

[ノイズ除去、輪郭抽出]

	方向指定なし
4連結	
8連結	

[膨張、収縮]

	方向指定なし	X方向	Y方向	右上り方向	右下り方向
4連結					
8連結					

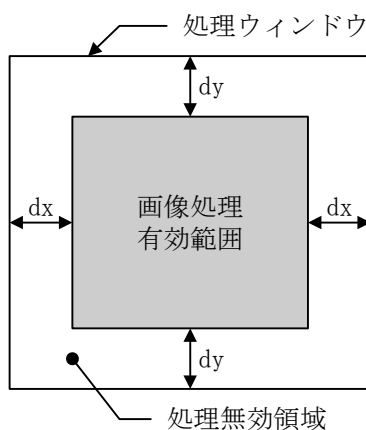
 注目画素

□ フィルタ処理の注目画素

フィルタ	内容
ノイズ除去	・注目画素が全て0の時、中心画素に0を出力 ・注目画素が全て1の時、中心画素に1を出力 ・上記以外の時、中心画素に入力画素値を出力
膨張	・注目画素の少なくとも一つが1の時、中心画素に1を出力 ・注目画素が全て0の時、中心画素に0を出力
収縮	・注目画素の少なくとも一つが0の時、中心画素に0を出力 ・注目画素が全て1の時、中心画素に1を出力
輪郭抽出	・注目画素が全て1の時、中心画素に0を出力 ・注目画素の少なくとも一つが0の時、中心画素に入力画素値を出力

□ 画像処理指定領域の周辺処理

3×3カーネルを用いて画像処理を行う為、指定領域の周辺は不定値になります。この不定値の領域は、パイプラインの段数によって決まります。
段数による不定値の領域は以下の通りです。



パイプライン段数	dx	dy
1段	1	1
2段	2	2
3段	3	3
4段	4	4
5段	5	5
6段	6	6
7段	7	7
8段	8	8

(注) 4段論理演算モードは4段の値になります。

2.19.2 形状変換パイプライン実行（IP_TrspipelineFLT）

□ C言語API

```
int ret = IP_TrspipelineFLT(IMGID ImgSrc, IMGID ImgDst);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○
ImgDst	○	△R	○	○	○	×	-	-	◆

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	ソース画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
3 0	ソース画面ウィンドウ設定エラー
3 3	デスティネーション画面ウィンドウ設定エラー
4 5	形状変換パイプライン処理設定エラー
4 9	ソース画面が不定画面
-	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

形状変換パイプライン（8段2値フィルタ）を実行します。
 本コマンドは、ハードウェアの制限事項として、X方向の処理領域が6以上必要です。X方向の処理領域が6未満の場合、正常な処理を行うことはできません。
 ソース画面に濃淡画像を指定した場合は、濃淡画像を2値画像として処理します。画素データの最上位ビットが1の場合は白、0の場合は黒として処理します。結果画像は0と255の2値画像です。
 演算内容は『SetTrsPipelineFLTMode()』コマンドで設定して下さい。

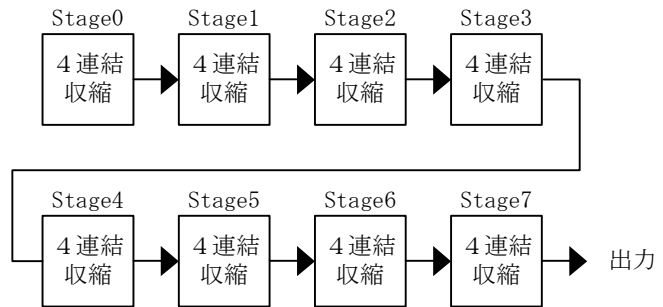
以下に実行例を示します。

（実行例1）8段全て4連結収縮とする場合

```
PipelineStageMode fmode;
IMGID ImgSrc, ImgDst;
int ret;
```

```
fmode.Stage0 = EROSION4;
fmode.Stage1 = EROSION4;
fmode.Stage2 = EROSION4;
fmode.Stage3 = EROSION4;
fmode.Stage4 = EROSION4;
fmode.Stage5 = EROSION4;
fmode.Stage6 = EROSION4;
fmode.Stage7 = EROSION4;
```

```
ret = SetTrsPipelineFLTMode(PIPELINE7, &fmode);
ret = IP_TrspipelineFLT(ImgSrc, ImgDst);
```

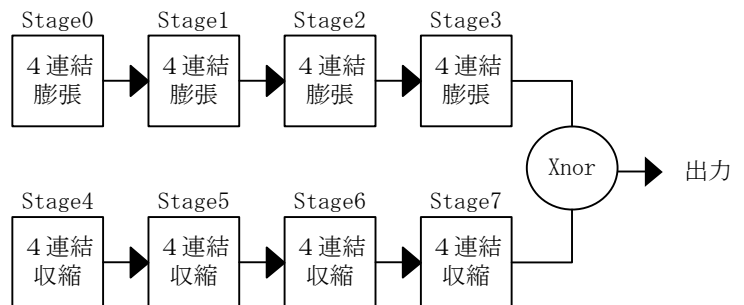


（実行例2）4段膨張した結果と4段収縮した結果をXnorする場合

```
PipelineStageMode fmode;
```

```
fmode.Stage0 = DILATION4;
fmode.Stage1 = DILATION4;
fmode.Stage2 = DILATION4;
fmode.Stage3 = DILATION4;
fmode.Stage4 = EROSION4;
fmode.Stage5 = EROSION4;
fmode.Stage6 = EROSION4;
fmode.Stage7 = EROSION4;
```

```
ret = SetTrsPipelineFLTMode(PIPELINEXNOR, &fmode);
ret = IP_TrspipelineFLT(ImgSrc, ImgDst);
```



2.20 パイプライン制御

2.20.1 パイプラインモード設定（EnablePipeline）

□ C言語API

```
int ret = EnablePipeline();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

— エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

システムが管理しているテーブルにパイプラインモード有効フラグを設定し、画像処理モードをパイプラインモードにします。パイプラインモードでは、画像処理、2値化、ヒストグラムの一連の処理を1画面の処理時間で実行することができます。

パイプラインモードでパイプライン処理を行うためには、パイプラインが構成可能な処理と順番にすることと、1段目のデスティネーション画面と2段目のソース画面を同一の画面に設定することが必要になります。

パイプライン処理終了時には、必ずパイプラインモード解除コマンド（DisablePipeline）を発行して下さい。

パイプラインが構成可能な処理と順番を以下に示します。他の組み合わせについては結果を保証しません。

● パイプライン処理構成

No.	処理と順番	備考
1	画像処理(濃淡) ⇒ ヒストグラム(濃淡)	注意事項2
2	画像処理(2値) ⇒ ヒストグラム(2値)	
3	2値化 ⇒ ヒストグラム(2値)	
4	画像処理(濃淡) ⇒ 2値化 ⇒ ヒストグラム(2値)	注意事項5

パイプライン処理を構成するコマンドとして、画像処理(濃淡)、画像処理(2値)、2値化処理、ヒストグラム処理(濃淡)、ヒストグラム(2値)、および、パイプライン処理とは無関係に動作するコマンドを以下に示します。

● 画像処理(濃淡)コマンド

項目	コマンド名	備考
画像転送／アフィン変換	IP_Copy	分離YUV画面はバイパス不可
	IP_ZoomOut	注意事項 2
	IP_ZoomOutExt	注意事項 2
	IP_ZoomIn	注意事項 2
	IP_ZoomInExt	注意事項 2
画素変換	IP_Invert	
	IP_Minus	
	IP_Abs	
	IP_AddConst	
	IP_SubConst	
	IP_SubConstAbs	
	IP_MultConst	
	IP_MinConst	
	IP_MaxConst	
	IP_ConvertLUT	
	IP_ShiftDown	
画像間算術演算	IP_Add	
	IP_Sub	
	IP_SubAbs	
	IP_Comb	
	IP_CombAbs	
	IP_Mult	
	IP_Average	
	IP_Min	
	IP_Max	
	IP_SubConstAbsAdd	
	IP_SubConstMultAdd	
	IP_SubConstMult	
	IP_CombDrop	
画像間論理演算	IP_And	
	IP_Or	
	IP_Xor	
	IP_InvertAnd	
	IP_InvertOr	
	IP_Xnor	
コンボリューション	IP_SmoothFLT	
	IP_EdgeFLT	
	IP_EdgeFLTAbs	
	IP_Lap14FLT	
	IP_Lap18FLT	
	IP_Lap14FLTAbs	
	IP_Lap18FLTAbs	
	IP_LineFLT	
ミニ／マックスフィルタ	IP_LineFLTAbs	
	IP_MinFLT	
	IP_MinFLT4	
	IP_MinFLT8	
	IP_MaxFLT	
	IP_MaxFLT4	
	IP_MaxFLT8	
	IP_LineMinFLT	
ランクフィルタ	IP_LineMaxFLT	
	IP_RankFLT	
	IP_Rank4FLT	
	IP_Rank8FLT	
	IP_MedFLT	
	IP_Med4FLT	
2 値マッチングフィルタ	IP_Med8FLT	
	IP_BinMatchFLT	

● 画像処理(2値)コマンド

項目	コマンド名	備考
2 値画像形状変換	IP_PickNoise4	
	IP_PickNoise8	
	IP_Outline4	
	IP_Outline8	
	IP_Dilation4	
	IP_Dilation8	
	IP_Erosion4	
	IP_Erosion8	
	IP_Thin4	
	IP_Thin8	
	IP_Shrink4	
	IP_Shrink8	
2 値パイプラインフィルタ	IP_TrspipelineFLT	

● 2 値化処理コマンド

項目	コマンド名	備考
2 値化	IP_Binarize	
	IP_BinarizeExt	

● ヒストグラム処理(濃淡)コマンド

項目	コマンド名	備考
濃淡画像特徴量抽出	IP_ExtractG0Features	注意事項 3
	IP_Histogram	
	IP_HistogramShort	
	IP_ProjectG0	
	IP_ProjectG0onX	
	IP_ProjectG0onY	
	IP_ProjectG0MaxValue	
	IP_ProjectG0MinValue	
	IP_ProjectBlockG0	注意事項 6
	IP_ProjectBlockG0MinMaxValue	注意事項 6
ラベリング	IP_ExtractLORegionX	注意事項 3
	IP_ExtractLORegionY	注意事項 3
	IP_ExtractLOArea	注意事項 3
	IP_ExtractLOAreaExt	注意事項 3
	IP_ExtractLOGravity	注意事項 3

● ヒストグラム処理(2値)コマンド

項目	コマンド名	備考
2 値画像特徴量抽出	IP_ExtractB0Features	
	IP_ProjectB0	
	IP_ProjectB0RegionX	
	IP_ProjectB0RegionY	
	IP_ProjectBlockB0	注意事項 6

● パイプライン処理と無関係に動作するコマンド

パイプライン処理と無関係に動作するコマンドには、ペンディングされているコマンドを実行（ページ）してから動作するコマンドと、ペンディングされているコマンドを実行せずに単独で動作するコマンドがあります。

[ページ]○:有 / ×:無

項目	コマンド名	ページ	備考
システム制御	InitIP	×	
	InitIPExt	×	
	ReadIPErrorTable	×	
	ClearIPError	×	
	SetIPDataType	×	
	GetIPVersionInfo	×	
	DisableIPErrorMessage	×	
	EnableIPErrorMessage	×	
画像メモリ領域管理	AllocImg	×	
	AllocComponentImg	×	
	FreeImg	×	
	FreeAllImg	×	
	ReadImgTable	×	
	GetElementImgID	×	
	ChangeImgDataType	×	
	SetWindow	×	注意事項 7
	SetAllWindows	×	注意事項 7
	ResetAllWindows	×	注意事項 7
	EnableIPWindow	×	注意事項 7
	DisableIPWindow	×	注意事項 7
	ReadWindow	×	
映像入力	ActiveVideoPort	×	
	SetVideoFrame	×	
	SelectCamera	×	
	GetCamera	×	
	SetVFDelay	×	
	CaptureContinuous	×	
	StopCaptureContinuous	×	
	GetCameraReqScene	×	
	GetCameraResScene	×	
	SetConfigCamera	×	
	ExitIPCamera	×	
映像出力	SetConfigDisp	×	
	SetDispPalette	×	
	SelectDisp	×	
	DispCamera	×	
	DisableDisp	×	
	DispImg	×	
	NoDisp	×	
	DispOverlap	×	
	DisableOverlap	×	
	SetConfigView	×	
	ExitIPView	×	
画像クリア	IP_ClearAllImg	○	
	IP_ClearImg	○	
	IP_Const	○	
画像転送／アフィン変換	IP_Zoom	○	
	IP_ZoomExt	○	
	IP_ZoomS	○	
	IP_Shift	○	
	IP_Rotate	○	
画素変換	WriteConvertLUT	○	
	IP_ShiftUp	○	

[ページ]○:有 / ×:無

項目	コマンド名	ページ	備考
ラベリング	IP_Label4	○	
	IP_Label8	○	
	IP_Label4withAreaFLT	○	
	IP_Label8withAreaFLT	○	
	IP_Label4withAreaFLTSort	○	
	IP_Label8withAreaFLTSort	○	
濃淡画像特徴量抽出	IP_ProjectLabelG0	○	
	IP_ProjectLabelG0MinMaxValue	○	
画像メモリアクセス	OpenImg	○	
	OpenImgExt	○	
	CloseImg	○	
	ReadImg	○	
	WriteImg	○	
	SetPixelPointer	○	
	ReadPixel	×	
	WritePixel	×	
	ReadPixelContinue	×	
	WritePixelContinue	×	
	RefreshImg	○	
	OpenImgDirect	○	
	CloseImgDirect	○	
2 値パイプラインフィルタ	SetTrsPipelineFLTMode	×	
パイプライン制御	EnablePipeline	×	
	DisablePipeline	○	
2 値マッチングフィルタ	SetBinMatchTemplate	×	
正規化関連	SetCorrMode	×	
	DisableCorrMask	×	
	EnableCorrMask	×	
	SetCorrBreakThr	×	
	DisableCorrBreak	×	
	EnableCorrBreak	×	
	SetCorrControl	×	
	SetCorrTemplate	○	
	SetCorrTemplateExt	○	
	IP_Corr	○	
	IP_CorrPrecise	○	
グラフィックス	SetDrawMode	×	
	SetStringAttributes	×	
	RefreshGraphics	○	
	DrawString	○	
	DrawLine	○	
	DrawSegments	○	
	DrawLines	○	
	DrawRectangle	○	
	DrawPolygon	○	
	DrawArc	○	
画像ファイリング	LoadBMPPFile	×	
	SaveBMPPFile	×	

[注意事項]

1. パイプラインモードでパイプライン処理とは無関係に動作するコマンドを実行した場合、パイプライン有効/無効で処理の順序が変わる為、処理結果が一致しない場合があります。
2. パイプラインモードで以下のアフィン変換コマンドを実行した場合、パイプライン有効/無効で処理領域が変わる為、処理結果が一致しない場合があります。パイプライン有効時には、アフィン変換コマンド実行後の処理領域はアフィン変換の結果領域になります。

IP_ZoomOut、IP_ZoomOutExt、IP_ZoomIn、IP_ZoomInExt

3. ヒストグラム処理（濃淡）対象が2値画像である場合、以下コマンドは2値画像を符号なし8ビット濃淡画像として処理します。

IP_ExtractGOFeatures、IP_ExtractLORegionX、IP_ExtractLORegionY、
IP_ExtractLOArea、IP_ExtractLOAreaExt、IP_ExtractLOGravity

4. パイプライン処理では、1段目のデスティネーション画面と2段目のソース画面を同一の画面に設定しますが、この画面には1段目の処理結果が格納されません。そこで、有効データが格納されている画面と区別する為に、画面データタイプを不定画面としています。不定画面は、映像入出力画面およびデスティネーション画面に使用することは可能ですが、ソース画面として使用するとエラーになります。不定画面をソース画面として使用する為には、ChangeImgDataTypeコマンドで画面データタイプを変更する必要があります。

以下に不定画面の発生例を示します。ImgSrc0とImgSrc1を加算処理後、2値化処理を実行し、結果をImgDstに格納する処理です。ImgIDは加算処理の処理結果が格納されず、不定画面になります。

```
EnablePipeline();           // パイプラインモード設定
IP_Add(ImgSrc0, ImgSrc1, ImgID); // 加算処理
IP_Binarize(ImgID, ImgDst, thr); // 2値化処理
DisablePipeline();          // パイプラインモード解除
```

5. パイプラインモードで、2値化に先行する画像処理（濃淡）のデスティネーション画面データタイプが符号付となる論理演算である場合、パイプライン処理しないで実行（ページ）されます。
6. パイプラインモードで、画面領域を分割して処理する以下のコマンドは、X方向分割数が17～32かつY方向分割数が16より大きい場合には、ペンディングされているコマンドを実行（ページ）してから濃度累積処理を行うため、パイプライン処理になりません。

IP_ProjectBlockGO、IP_ProjectBlockGOMinMaxValue、
IP_ProjectBlockBO

7. パイプラインモード中の以下コマンドによるウィンドウ操作は禁止します。

SetWindow、SetAllWindows、ResetAllWindows、
EnableIPWindow、DisableIPWindow

2.20.2 パイプラインモード解除 (DisablePipeline)

□ C言語API

```
int ret = DisablePipeline();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

システムが管理しているテーブルにパイプラインモード無効フラグを設定し、画像処理でのパイプラインモードを解除します。

また、パイプラインモードを解除することにより、パイプラインモード状態で実行待ちとなっていた処理がある場合、その処理を実行します。

2.21 2値マッチングフィルタ

2.21.1 2値画像マッチングフィルタテンプレート登録（SetBinMatchTemplate）

□ C言語API

```
int ret = SetBinMatchTemplate(int *Temp, int TmpID);
```

□ パラメータ

int	*Temp	テンプレート登録テーブルを格納したアドレス テーブルサイズはint型で9×9個の容量を確保して下さい。
int	TmpID	テンプレート番号（0～CFG_BMFTMPSZ - 1） ※CFG_BMFTMPSZはシステムで定義しています。

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2 0	テンプレート登録テーブルを格納したアドレスが不正
8 1	テンプレート番号が有効範囲外
—	エラーリセットコマンド（ClearIPError）未発行

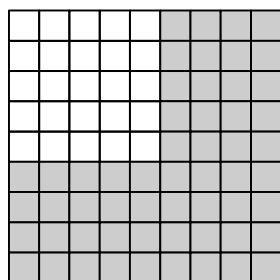
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

2値画像マッチングフィルタのテンプレート登録を行います。
テンプレートのサイズは9×9画素です。その設定は、Temp[Y][X]（2次元配列）で記述して下さい。テンプレート登録テーブルに格納した値は、最下位1ビットのみ有効となります。また、テンプレートは9×9画素を正しく設定して下さい。正しく設定しない場合は、動作保証をいたしません。

<記述例>

テンプレート(9×9画素)



テンプレート登録テーブル設定

```
int Temp[9][9] = {{0, 0, 0, 0, 0, 1, 1, 1, 1},
                  {0, 0, 0, 0, 0, 1, 1, 1, 1},
                  {0, 0, 0, 0, 0, 1, 1, 1, 1},
                  {0, 0, 0, 0, 0, 1, 1, 1, 1},
                  {0, 0, 0, 0, 0, 1, 1, 1, 1},
                  {1, 1, 1, 1, 1, 1, 1, 1, 1},
                  {1, 1, 1, 1, 1, 1, 1, 1, 1},
                  {1, 1, 1, 1, 1, 1, 1, 1, 1},
                  {1, 1, 1, 1, 1, 1, 1, 1, 1}};
```

2.21.2 2値画像マッチングフィルタ実行（IP_BinMatchFLT）

□ C言語API

```
int ret = IP_BinMatchFLT(IMGID ImgSrc, IMGID ImgDst, int TmpID, enum IPBinMatchMode mode);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	TmpID	テンプレート番号（0～CFG_BMFTMPSZ－1） ※CFG_BMFTMPSZはシステムで定義しています。
enum IPBinMatchMode	mode	マッチング処理オプション
	MATCH_AND	(0) AND処理
	MATCH_XNOR	(1) XNOR処理

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B1	△B1	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

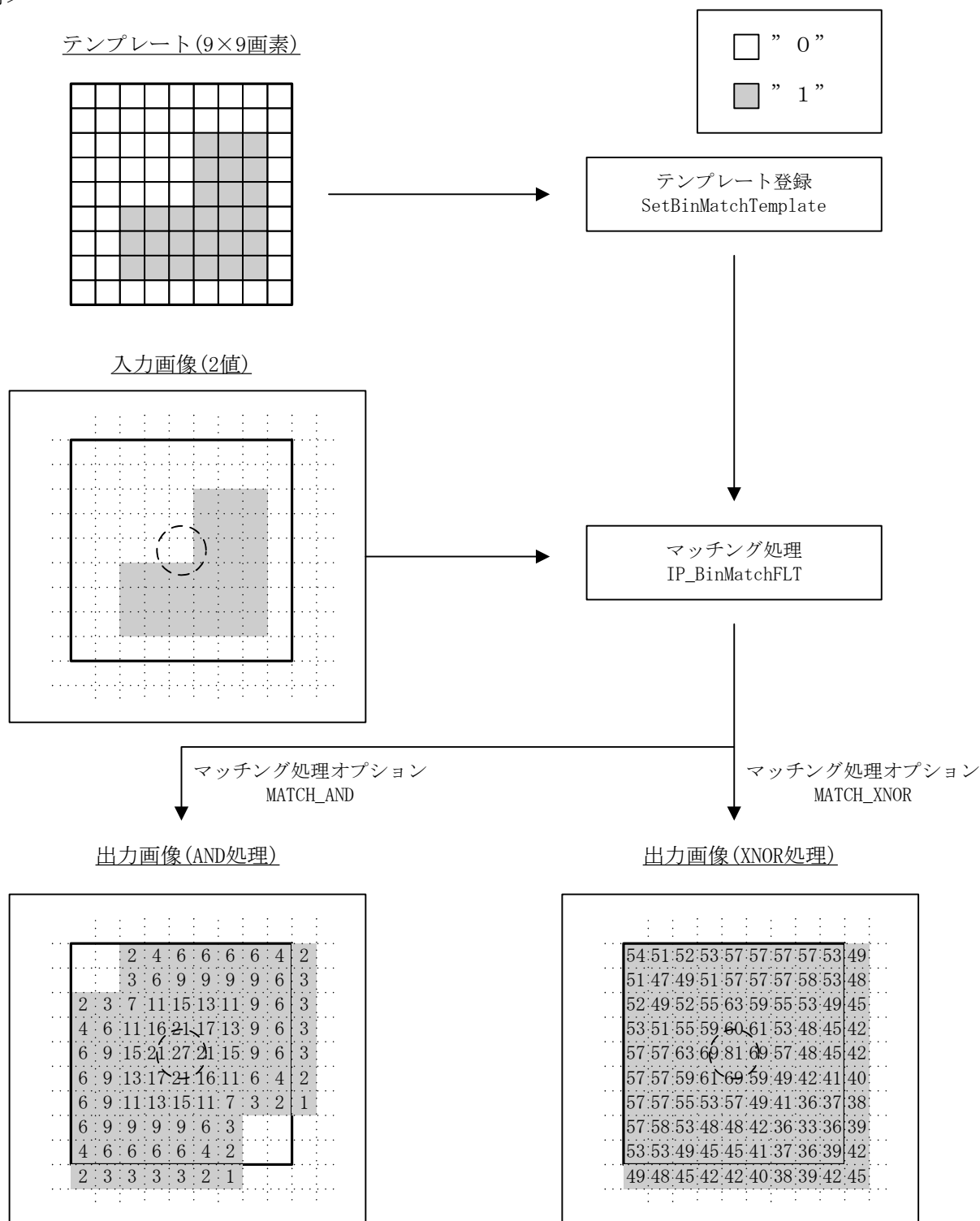
2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
22	テンプレート番号設定値範囲外
23	オプション設定値範囲外
30	ソース画面ウィンドウ設定エラー
33	デスティネーション画面ウィンドウ設定エラー
49	ソース画面が不定画面
87	テンプレート未設定エラー
-	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

2 値画像マッチングフィルタの実行を行います。
 9 × 9 画素のテンプレートにより入力画面（2 値）のマッチング処理を行い、結果を濃淡画面に出力します。
 ウィンドウサイズを 9 × 9 未満で実行した場合、処理領域は全て周辺処理により 0 クリアされます。
 ソース画面に濃淡画像を指定した場合は、濃淡画像を 2 値画像として処理します。画素データの最上位ビットが 1 の場合は白、0 の場合は黒として処理します。

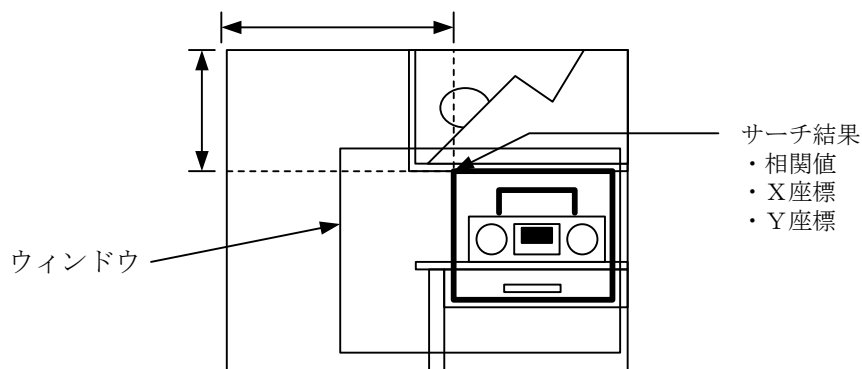
<例>



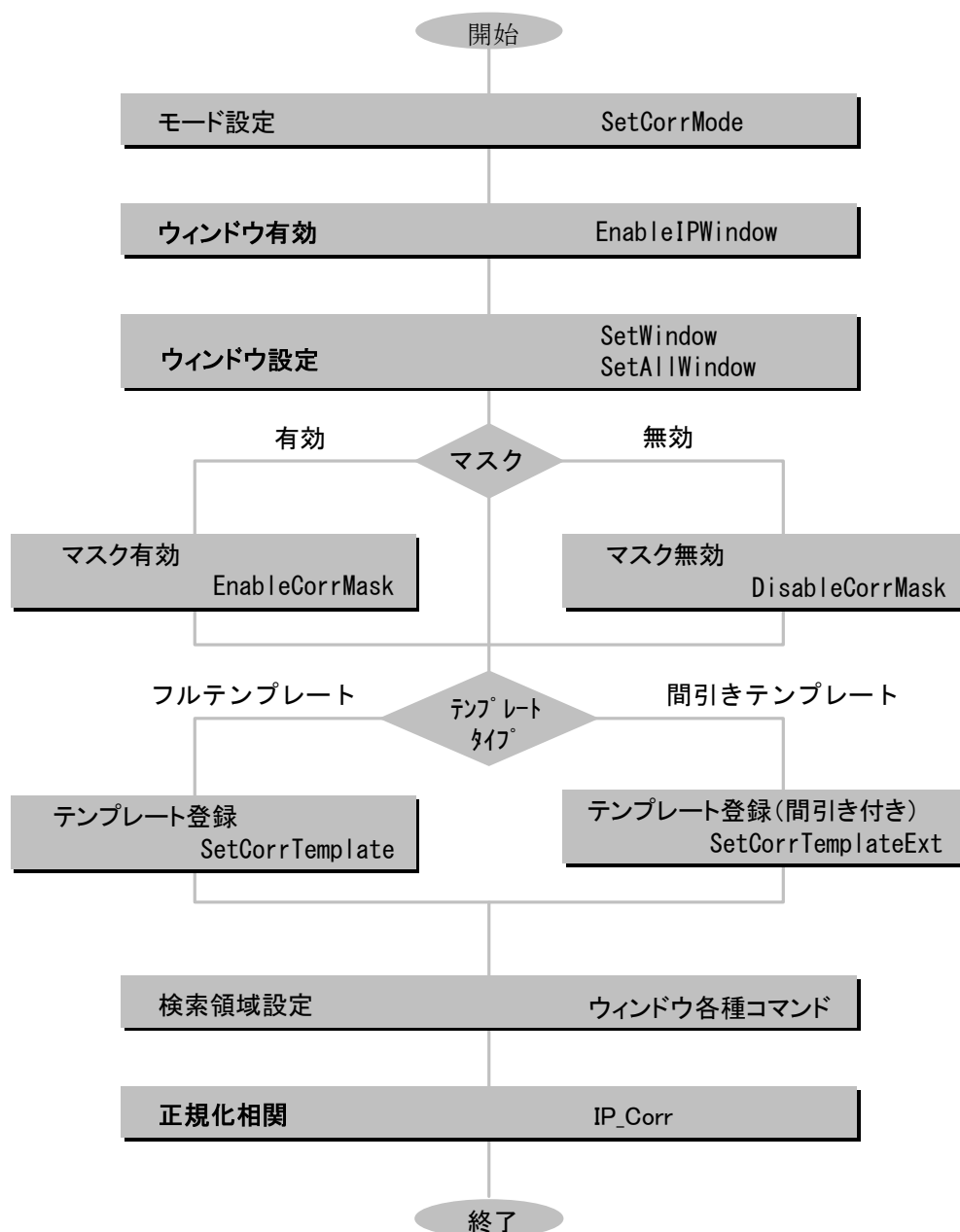
入力画像の () を中心とした場合、オプションMATCH_ANDでは入力画像とテンプレートのAND処理結果から“1”の総数である“27”が出力され、オプションMATCH_XNORではXNOR処理結果から“1”の総数である“81”が出力されます。
 ※ 指定領域の周辺(内側4画素の領域)は、0を出力します。

2.22 正規化相関

正規化相関コマンドでは、正規化相関演算が実行できます。
 正規化相関コマンドでの座標系は、画面絶対座標であり、画面始点が座標 0 になります。
 よって、正規化相関処理結果の座標は画面始点を基準とした座標です。



(1) 正規化相関実行手順を下図のフローで実行します。



(2) テンプレートタイプ

正規化相関のテンプレートには、2種類のタイプがあります。

- フルテンプレート

SetCorrTemplate() コマンドで登録されたテンプレートです。フルテンプレートは、IP_Corr() 実行時に、処理対象画面内全画素を処理する（フルサーチ）ので、高精度な処理ができます。

- 間引きテンプレート

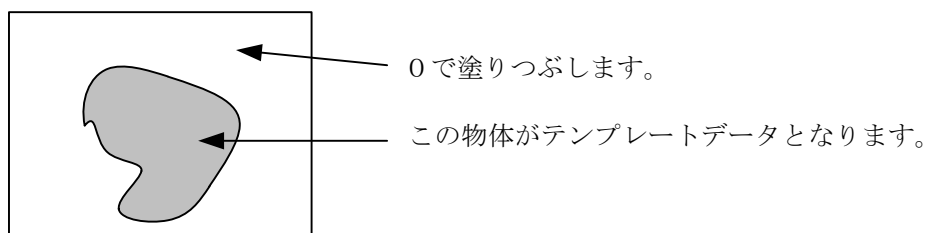
SetCorrTemplateExt() コマンドで登録されたテンプレートです。間引きテンプレートは、X/Y方向の間引き率を登録できます。この間引き率により、テンプレートデータおよび処理対象画面を間引いて演算処理するので、精度はフルテンプレートに対して落ちますが、演算回数が減るため、高速処理ができます。

処理対象画像に応じて、これら2つのテンプレートタイプを評価し使用してください。

(3) 正規化相関マスク

正規化相関マスクは、テンプレート内で濃度0の画素を処理対象外にします。よって、矩形でない任意の形をテンプレートとしたい場合に有効です。矩型のテンプレート内で必要物体部以外を0で塗りつぶしておき、IP_Corr() を実行する前に、EnableCorrMask() を実行すれば、0画素部分は処理されないため、任意形のテンプレートで処理したことと同じになります。

正規化相関マスクはEnableCorrMask() コマンドで設定します。



2.22.1 正規化相関モード設定 (SetCorrMode)

□ C言語API

```
int ret = SetCorrMode(int SearchMode, int count, float CorrThr);
```

□ パラメータ

int	SearchMode	走査方法 0 : ラスタ型に走査 1 : 同上 (ただし、相関値以上の物体を指定個数検知すると停止します) 2 : スパイラル型に走査 3 : 同上 (ただし、相関値以上の物体を指定個数検知すると停止します)
int	count	検知物体数 (1 ~ 100)
float	CorrThr	相関しきい値 (0 ~ 1)

□ 対応画面タイプ/画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

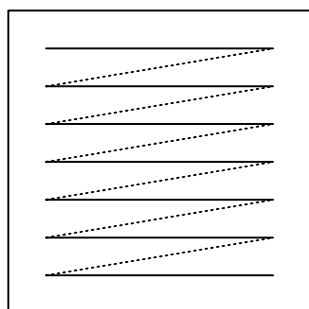
20	走査方法設定値範囲外
84	検知物体数エラー
85	相関値エラー
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

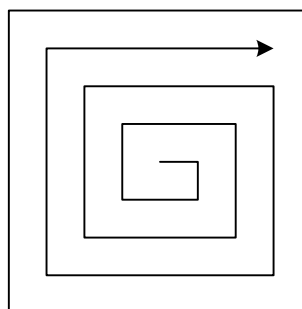
□ 機能

正規化相関での処理モードを設定します。
 走査方法は以下に示すとおり、ラスタ型は左上から、スパイラル型は中央から走査します。
 相関しきい値は正の相関値に対する値です。『SetCorrControl()』コマンドで「相関値を符号付で処理する」として正規化相関を行った場合、相関値が負 (テンプレートとサーチ画像がポジ/ネガの関係) となる物体は、相関しきい値を0としても未検出になります。

ラスタ型



スパイラル型



2.22.2 正規化関連マスク有効 (EnableCorrMask)

□ C言語API

```
int ret = EnableCorrMask();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

正規化関連処理のテンプレート内0値のマスクを有効にします。
 正規化関連マスクはテンプレート内で濃度0の画素を処理対象外にします。よって、矩型でない任意の形をテンプレートとしたい場合に有効です。矩型のテンプレート内で必要物体部以外を0で塗りつぶしておき、『IP_Corr()』コマンドを実行する前に、『EnableCorrMask()』コマンドを実行すれば、0画素部分は処理されないため、任意形のテンプレートで処理したことに同じになります。
 『EnableCorrMask()』コマンドは、テンプレート登録 (SetCorrTemplate) をする前に実行して下さい。また、正規化関連マスクを無効にする場合は『DisableCorrMask()』コマンドを実行して下さい。

2.22.3 正規化相関マスク無効 (DisableCorrMask)

□ C言語API

```
int ret = DisableCorrMask();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

正規化相関のデフォルト設定は、正規化相関マスク無効となります。
『EnableCorrMask()』コマンドで正規化相関マスク有効にした処理を無効にします。

2.22.4 打ち切りしきい値設定 (SetCorrBreakThr)

□ C言語API

```
int ret = SetCorrBreakThr(int thr);
```

□ パラメータ

int thr 累積誤差しきい値設定パラメータ (0 ~ 255)

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

20 累積誤差しきい値設定パラメータ設定範囲外
- エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

システムが管理しているテーブルに累積誤差しきい値計算用パラメータを設定します。
本コマンドは、指定累積誤差オーバによる相関演算途中打ち切り制御に使用します。相関演算途中打ち切り制御とは、相関演算処理中、累積誤差が設定したしきい値を超えた時点でサーチ対象画像がテンプレート画像と明らかに異なるものと判断し相関演算を打ち切ります。途中で演算を打ち切ることにより処理の高速化を図ることができます。
なお、本コマンドはアルゴリズムの関係上、相関演算途中打ち切りが発生しない場合があります。

● しきい値の設定

実際に演算を途中で打ち切るための累積誤差しきい値は、

累積誤差しきい値 = thr (累積誤差しきい値設定パラメータ) × テンプレート画素数

(テンプレート画素数 = テンプレートX方向長 / X方向間引き率
× テンプレートY方向長 / Y方向間引き率)

で計算されます。

累積誤差しきい値が2の23乗-1 (0x007F:FFFF) を超えると未検出となりますので注意して下さい。

また、thr = 0 を設定した場合、自動しきい値変更モードとなり、常に最小値をしきい値として、演算の打ち切りを行います (SSDA法の適用)。この場合、照明変動には弱くなりますので十分に注意して使用して下さい。また、照明変動がほとんどない場合はthr を小さく、照明変動がある場合には、thrを大きく設定して下さい。

● 相関演算打ち切り処理許可／禁止は、

EnableCorrBreak(); 相関演算打ち切り処理許可
DisableCorrBreak(); 相関演算打ち切り処理禁止(デフォルト)

を使用して下さい。

2.22.5 相関演算打ち切り許可 (EnableCorrBreak)

□ C言語API

```
int ret = EnableCorrBreak();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

相関演算途中打ち切り処理を許可します。
 本コマンドは、指定累積誤差オーバによる相関演算途中打ち切り制御に使用します。サーチ対象画像がテンプレート画像と明らかに異なる場合、相関演算実行中に累積誤差が非常に大きくなると考えられます。相関演算途中打ち切り制御は、この累積誤差がしきい値を超えた時点で、ミスマッチと判断し、途中で相関演算を中止させます。途中で演算を中止することにより処理の高速化を図ることができます。

しきい値の設定については、『SetCorrBreakThr()』コマンドを参照して下さい。
 また、相関演算途中打ち切り処理を禁止する場合は『DisableCorrBreak()』コマンドを実行して下さい。

2.22.6 相関演算打ち切り禁止（DisableCorrBreak）

□ C言語API

```
int ret = DisableCorrBreak();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

-	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

正規化相関のデフォルト設定は、相関演算途中打ち切り禁止となります。
『EnableCorrBreak()』コマンドで相関演算途中打ち切りを許可にした処理を禁止します。

2.22.7 正規化相関制御 (SetCorrControl)

□ C言語API

```
int ret = SetCorrControl(IPCorrControl *cnt);
```

□ パラメータ

IPCorrControl *cnt 制御モードを格納したアドレス

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ パケットの構造

```
typedef struct {
    int      SignOperation;   相関値の符号設定
                                0 : 相関値を絶対値で処理 (デフォルト)
                                1 : 相関値を符号付で処理
                                2 : 相関値は絶対値で処理するが、戻り値には符号を付ける
    int      ReturnValueOfSetCorrTemplate; 戻り値のタイプ設定
                                0 : 正常終了 (0) または異常終了 (-1) (デフォルト)
                                1 : テンプレートの分散値を返す、エラー時 -1
} IPCorrControl;
```

□ エラーコード

2 0 相関値の符号/戻り値タイプ設定範囲外、またはアドレス不正
- エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

正規化相関での相関値を符号付の値で扱うか、絶対値で扱うかを制御します。
本コマンドにより、テンプレート登録コマンド (SetCorrTemplate, SetCorrTemplateExt) の戻り値として、テンプレートの分散値を返すように設定できます。
また、正規化相関での相関値を符号付か、絶対値のいずれかで扱うかを指定できます。絶対値で扱うとポジ／ネガの関係の画像でも一致していると評価されます。

2.22.8 正規化関連テンプレート登録（SetCorrTemplate）

□ C言語API

```
int ret = SetCorrTemplate(IMGID ImgTmp, int TmpID);
```

□ パラメータ

IMGID	ImgTmp	テンプレート登録画面番号
int	TmpID	テンプレート番号（0～CFG_CORTMPSZ - 1） ※CFG_CORTMPSZはシステムで定義しています。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	×	×

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1） 『SetCorrControl()』コマンドで、戻り値をテンプレートの分散値に設定した場合は、分散値を返します。

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	テンプレート登録画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
30	テンプレート画面ウィンドウ設定エラー
49	テンプレート画面が不定画面
80	テンプレートサイズエラー
81	テンプレート番号が有効範囲外
83	テンプレート内同一画素エラー
86	マスク処理でのテンプレート登録エラー
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

正規化関連で使用する為のテンプレートを登録します。テンプレートの位置およびサイズは、ウィンドウ（SRCO_WIN）によって決まります。このコマンド発行前に必ず、ウィンドウ設定コマンドによりSRCO_WINウィンドウを設定して下さい。

また、テンプレートのサイズは2～65536画素（256×256相当）までとし、テンプレート内が同一画素値（濃度）のときはエラーとなります。

正規化関連テンプレート登録では、テンプレート画像の存在する画面番号（ImgTmp）と座標情報のみ登録します。よって、テンプレート登録画面は解放（FreeImg()）したり、画像処理によるデータ変更をしないで下さい。

2.22.9 正規化関連テンプレート登録(間引き付き)(SetCorrTemplateExt)

□ C言語API

```
int ret = SetCorrTemplateExt(IMGID ImgTmp, int TmpID, int xmag, int ymag);
```

□ パラメータ

IMGID	ImgTmp	テンプレート登録画面番号
int	TmpID	テンプレート番号 (0 ~ CFG_CORTMPSZ - 1) ※CFG_CORTMPSZはシステムで定義しています。
int	xmag	X方向の間引き率 (1 ~ 8)
int	ymag	Y方向の間引き率 (1 ~ 8)

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	×	×

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1) 『SetCorrControl()』コマンドで、戻り値をテンプレートの分散値 に設定した場合は、分散値を返します。

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	テンプレート登録画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
22	X方向の間引き率設定値範囲外
23	Y方向の間引き率設定値範囲外
30	テンプレート画面ウィンドウ設定エラー
49	テンプレート画面が不定画面
80	テンプレートサイズエラー
81	テンプレート番号が有効範囲外
83	テンプレート内同一画素エラー
86	マスク処理でのテンプレート登録エラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

正規化相関で使用する為、テンプレート（間引きテンプレート）を登録します。テンプレートの位置およびサイズは、ウィンドウ（SRC0_WIN）によって決まります。このコマンド発行前に必ず、ウィンドウ設定コマンドによりSRC0_WINウィンドウを設定して下さい。

また、テンプレートのサイズは2～65536画素（256×256相当）までとし、テンプレート内が同一画素値（濃度）のときはエラーとなります。

正規化相関テンプレート登録（間引き付き）では、テンプレート画像の存在する画面番号（ImgTmp）と座標情報と間引き率を登録します。よって、テンプレート登録画面は解放（FreeImg()）したり、画像処理によるデータ変更をしないで下さい。

間引き率を2とすると、テンプレート内データは1画素おきで処理される為、データ量は半分になります。間引きテンプレートは、処理が高速ですが、精度がフルテンプレート（SetCorrTemplate()で登録）より落ちます。処理対象物体によって、フルテンプレート／間引きテンプレートどちらが有効か、また、間引きテンプレートの間引き率は、実際に評価をしてから使用して下さい。

精度	フルテンプレート > 間引きテンプレート
処理速度	フルテンプレート < 間引きテンプレート

2.22.10 正規化相関 (IP_Corr)

□ C言語API

```
int ret = IP_Corr(IMGID ImgSrc, int TmpID, IPCorrTbl *Tbl);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号 (サーチ対象画面番号)
int	TmpID	テンプレート番号 (0 ~ CFG_CORTMPSZ - 1) ※CFG_CORTMPSZはシステムで定義しています。
IPCorrTbl	*Tbl	相関値および座標を返すアドレス 『SetCorrMode()』コマンドで設定した検知物体 数分を確保して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	×	×

□ リターンパラメータ

int	ret	検知した物体の数 (0 以上) または異常終了 (-1)
IPCorrTbl	*Tbl	相関値および座標を格納したアドレス

□ パケットの構造

```
typedef struct {
    float    VALUE;           相関値
    int      X;               画面原点相対X座標
    int      Y;               画面原点相対Y座標
} IPCorrTbl;
```

□ エラーコード

2	OS システムエラー
3	IMP 処理タイムアウト
8	VIN 入力タイムアウト
1 0	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 2	相関値および座標を返すアドレスが不正
4 9	ソース画面が不定画面
8 0	テンプレートサイズエラー
8 1	テンプレート番号が有効範囲外
8 7	テンプレート未設定エラー
8 8	テンプレート登録画面番号エラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

正規化相関を実行し、検知した物体数をリターン値として返します。また、そのときの相関値および座標を、ユーザテーブルへセットします。

『IP_Corr()』コマンドでは、使用するテンプレートタイプにより、処理内容が異なります。フルテンプレート (SetCorrTemplate()) で登録の場合は、処理領域内全画素を演算処理します。間引きテンプレート (SetCorrTemplateExt()) で登録の場合は、登録された間引き率で処理領域内を間引いて処理します。

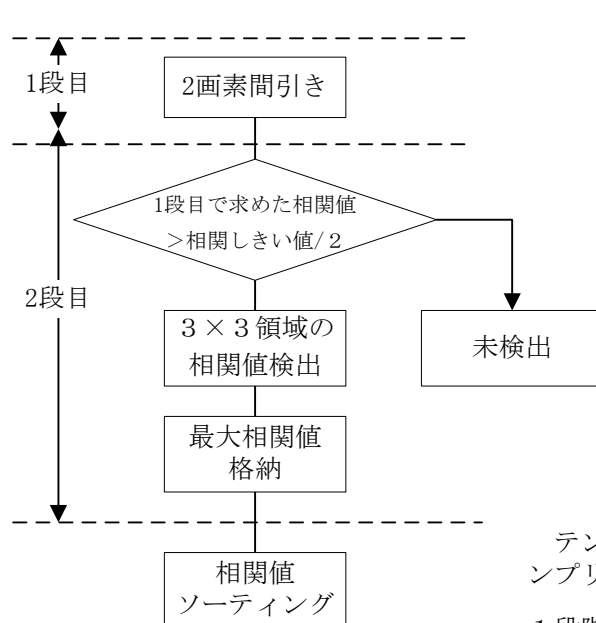
本コマンドは、2段階の正規化相関処理で物体検出を行います。1段階目の処理では、サーチ始点座標を2画素間引きで実行します。間引きテンプレートの場合は、間引き率に応じた間引き後にサーチ始点座標の2画素間引きが適用されます。

2段階目の処理では、1段階目の処理で求めた相関値が『SetCorrMode()』コマンドで設定した相関しきい値の半分の値より大きい場合、このサーチ始点座標を中心とした3×3の領域について正規化相関を実行します。そして、この9点のなかで相関値の最も高いデータをテーブルへ格納します。

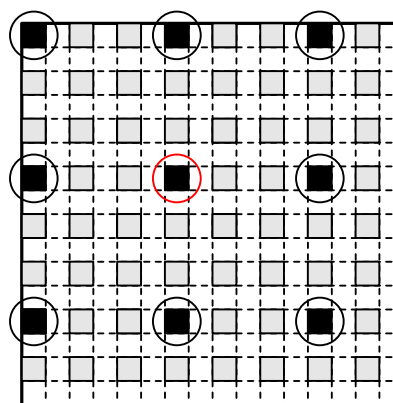
全処理領域の正規化相関終了後、テーブル内のデータを相関値の大きさにソートし出力します。

[制限事項]

- 1段階目の処理では、サーチ始点座標を2画素間引き間隔で実行する為、間引きによる位置ずれにより1段階目の相関値が低下する場合があります。これはテンプレートタイプに関わりません。この相関値が『SetCorrMode()』コマンドで設定した相関しきい値の半分以下の場合、2段階目の処理に移行せず、未検出となります。相関しきい値の設定の際には、1段階目の位置ずれによる相関値の低下を考慮する必要がありますが、実際には使用する環境に応じて数パターンの相関値を取得しながら最終的な相関しきい値を決定する方法が有効です。
- テンプレートサイズが小さい場合、間引きによる位置ずれの影響が大きくなり、相関値が低下する場合があります。
- 走査方式が0または2で、正規化相関実行中に検知物体数を超えた場合、検出済み相関値データを格納しているテーブル内で相関値の最も低い要素が上書きされます。走査方式および検知物体数は『SetCorrMode()』コマンドで設定します。



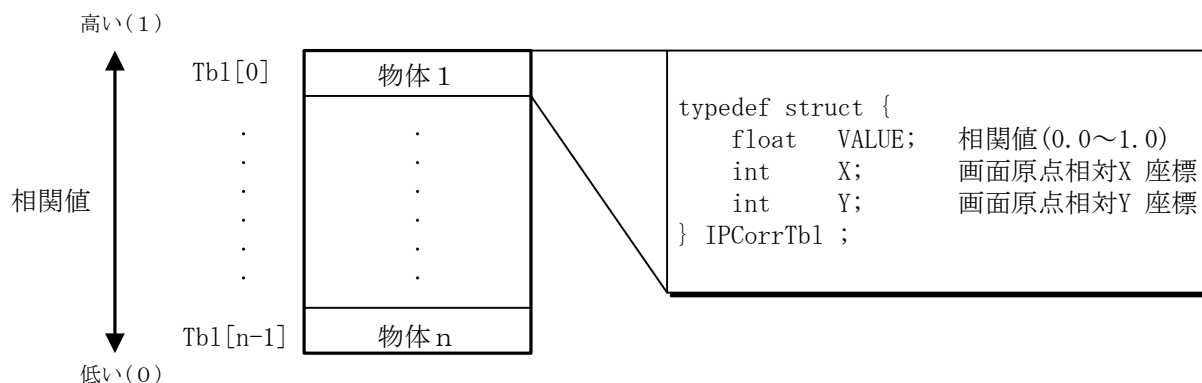
<例：間引き率2>



テンプレートの間引き率が2の場合、灰色で示した座標がサンプリングされます。

1段階目：2画素間引きで処理されるため黒色座標を始点座標として相関値検出。

2段階目：相関値が『SetCorrMode()』コマンドで設定した相関しきい値の1/2以上だった場合(赤丸)、3×3領域(黒丸で示した座標を始点座標とする9点)について相関値を検出。9点のなかで相関値の最も高いデータをテーブルへ格納。



2.22.11 正規化相関(高精度位置決め)(IP_CorrPrecise)

□ C言語API

```
int ret = IP_CorrPrecise(IMGID ImgSrc, int TmpID, int xc, int yc,
                        IPCorrPreciseTbl *Tbl, enum IPCorrPreciseOpt opt);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号 (サーチ対象画面番号)
int	TmpID	テンプレート番号 (0 ~ CFG_CORTMPSZ - 1) ※CFG_CORTMPSZはシステムで定義しています。
int	xc	サーチ中心X座標指定
int	yc	サーチ中心Y座標指定
IPCorrPreciseTbl	*Tbl	サブピクセル単位の座標を返すアドレス
enum IPCorrPreciseOpt	opt	位置決めオプション指定 CRP_CENTER (0) (xc, yc) を中心座標とみなす CRP_MAXVAL (1) (xc, yc) と上下左右の相関値を求めて 最大となった位置を中心座標とみなす

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	×	×

□ リターンパラメータ

int	ret	0 : 正常終了 (4近傍の範囲以内で高精度位置を計測し処理を終了) -1 : 異常終了 (エラーコード参照) 1 : 4近傍の範囲外で高精度位置を計測し終了 (4近傍の5点内で最も 相関値の高い座標をテーブルにセット)
IPCorrPreciseTbl	*Tbl	サブピクセル単位の座標を格納したアドレス

□ パケットの構造

```
typedef struct{
    float    X;           X座標：画面相対アドレス
    float    Y;           Y座標：画面相対アドレス
} IPCorrPreciseTbl;
```

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
24	サブピクセル単位の座標を返すアドレスが不正
25	オプション設定値範囲外
49	ソース画面が不定画面
81	テンプレート番号が有効範囲外
82	サーチ領域が画面をはみ出すようなサーチポイントを検出
87	テンプレート未設定エラー
88	テンプレート登録画面番号エラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

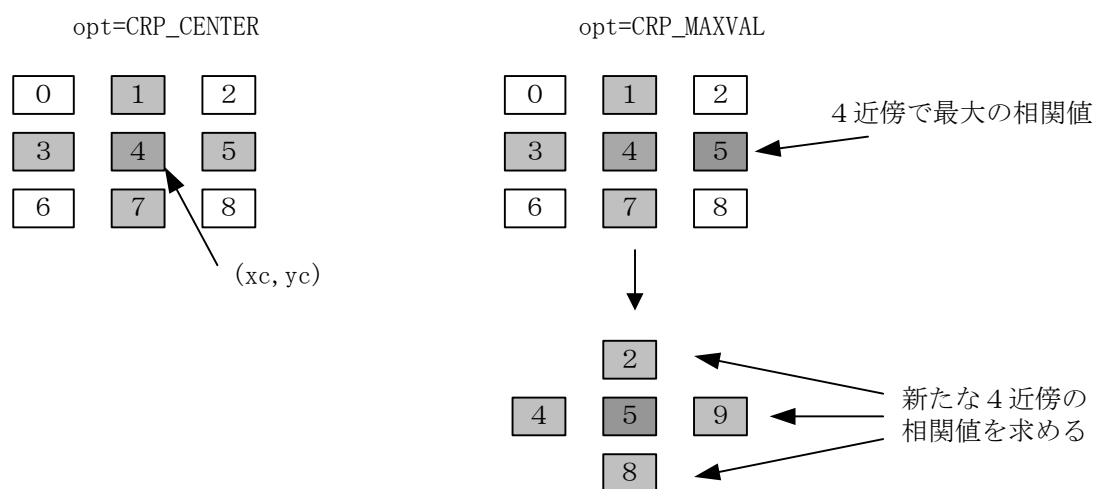
機能

正規化相関を (xc, yc) の近傍で実行し、最も相関値の高い位置のサブピクセル単位（1画素単位以下）の座標値をユーザテーブルへセットします。

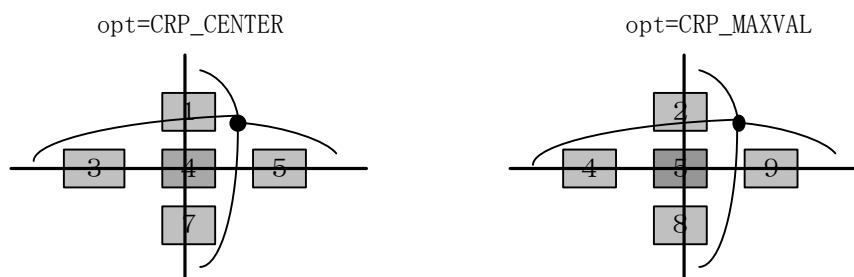
- (xc, yc) は、画面相対座標で指定して下さい。
- 間引き指定付きテンプレートを用いると、4近傍のアドレス計算にも間引き率が反映されます。
（例）間引き率2を指定していると、4近傍のアドレスは中心から2画素離れます。
- ウィンドウ指定、『SetCorrMode()』コマンドは無視されます。
- 相関演算打ち切りモードは無効にして処理されます。
- 本コマンド『IP_CorrPrecise()』は近傍の範囲しか見ない為、まず、『IP_Corr()』コマンドを用いて範囲を絞り込んでおく必要があります。

<処理仕様>

(A) オプション指定に従い4近傍の相関値を求める。



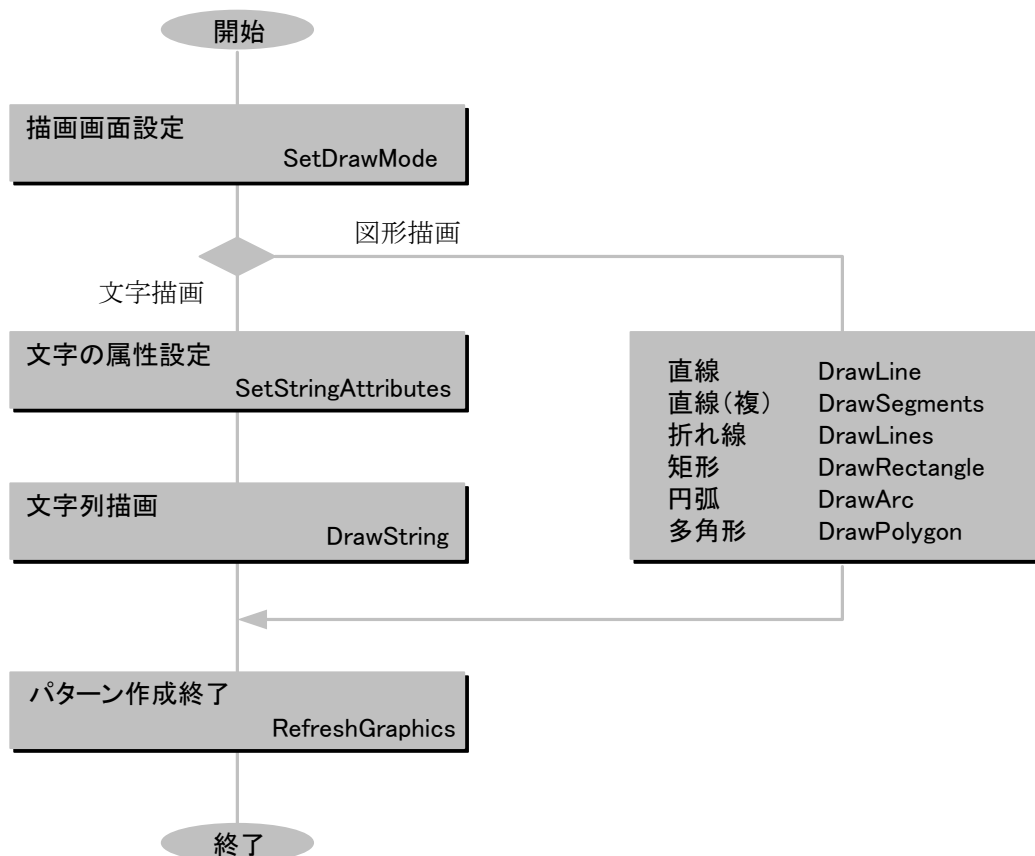
(B) 求めたX方向の3点とY方向の3点の相関値を用いて、2次近似により最大値の存在するX座標とY座標を得る。



2.23 グラフィックス

グラフィックスコマンドでは、文字列、図形描画等のパターン作成ができます。
グラフィックスは、下記手順のフローに従って実行してください。なお、グラフィックスコマンドでの座標系はウィンドウ相対であり、ウィンドウの始点が座標0になります。

パターン作成手順



2.23.1 描画面面設定 (SetDrawMode)

□ C言語API

```
int ret = SetDrawMode(IMGID ImgSrc, enum DRAW_MODE mode, int color);
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号
enum DRAW_MODE	mode	描画モード DRAW_DIRECT (0) 上書き DRAW_EOR (1) 元の画素とEORして描画 DRAW_DIRECT_EXT (2) 描画データのみ上書き
int	color	描画カラー (0～65535)



上位16bitは0となるように設定して下さい。
 背景の色は、描画モードをDRAW_DIRECTに設定して、
 文字描画 (DrawString()) を行う場合のみ有効となります。

※描画カラーとしてDRAW_COLOR列挙型を使用できます。

COLOR_WHITE	(255)	白
COLOR_BLACK	(0)	黒

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

1 0	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 1	描画モード設定値範囲外
2 2	描画カラー設定値範囲外
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

パターン作成時の画面番号と描画カラーを設定します。

2.23.2 文字の属性設定 (SetStringAttributes)

□ C言語API

```
int ret = SetStringAttributes(enum STRING_SIZE size, int C_Interval, int L_Interval);
```

□ パラメータ

enum STRING_SIZE size	文字サイズ
STRING_16x16	(0) 標準サイズ (16 × 16)
STRING_32x32	(1) 4倍サイズ (32 × 32)
STRING_48x48	(2) 9倍サイズ (48 × 48)
STRING_64x64	(3) 16倍サイズ (64 × 64)
int C_Interval	文字間隔 (0 ~ 128)
int L_Interval	行間隔 (0 ~ 128)

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

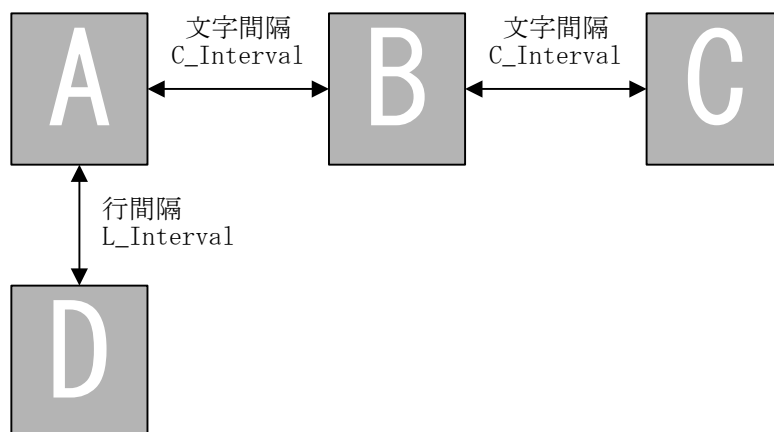
□ エラーコード

20	文字サイズ設定値範囲外
21	文字間隔設定値範囲外
22	行間隔設定値範囲外
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

システムが管理しているテーブルに、描画する文字のサイズ、文字間隔、行間隔を設定します。本ライブラリで描画可能な文字は、半角英数字、半角カタカナ、記号です。このとき、描画文字のX方向のサイズは、本コマンドで指定した文字サイズの半分になります。全角文字を入力した場合は、全角スペース文字に変換され出力します。



2.23.3 パターン作成終了（RefreshGraphics）

□ C言語API

```
int ret = RefreshGraphics();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
4 9	描画画面（SetDrawModeで設定したソース画面）が不定画面
—	エラーリセットコマンド（ReadIPErrorTable）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

パターン作成した画面のキャッシュページを行います。

（注）『ReadIPErrorTable()』コマンドで読み出した時のエラー発生コマンド名は、“RefreshImg”となります。

2.23.4 文字列描画 (DrawString)

□ C言語API

```
int ret = DrawString(int x, int y, char *str);
```

□ パラメータ

int	x	始点X座標 (ウィンドウ相対)
int	y	始点Y座標 (ウィンドウ相対)
char	*str	文字列を格納したアドレス

□ 対応画面タイプ/画面データタイプ

なし

□ リターンパラメータ

int	ret	書き込みを行った行数 (0 以上) または異常終了 (- 1)

□ エラーコード

2	OS システムエラー
3	IMP 処理タイムアウト
8	VIN 入力タイムアウト
1 0	描画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 0	始点X座標設定エラー
2 1	始点Y座標設定エラー
2 2	文字列を格納したアドレスが不正
3 4	画像メモリアクセスウィンドウ (SYS_WIN) 設定エラー
4 4	座標設定エラー (文字列の1文字目が画像メモリアクセスウィンドウ範囲外)
—	エラーリセットコマンド (ClearIPError) 未発行

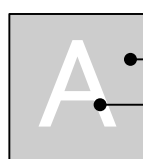
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

描画面設定及び文字の属性設定コマンドで設定した内容にもとづき、文字列を書き込みます。処理範囲は画像メモリアクセスウィンドウ (SYS_WIN) となります。グラフィックスコマンドでの座標系はウィンドウ相対であり、ウィンドウの始点が座標 (0, 0) になります。なお、描画文字の下端が処理範囲の下端をはみだした場合は、文字を描画せずに処理を終了します。

表示する文字列は、ASCII文字またはシフトJIS文字で作成して下さい。また、文字列終了コードには NULL (0) を入れて下さい。なお、本コマンドで描画可能な文字は、半角英数字、半角カタカナ、記号のみです。サポート外の文字が入力された場合は、スペース文字を出力します。このスペース文字は通常の文字と同様に扱われ、描画処理の終了判定、改行処理、リターンパラメータ (書き込みを行った行数) についても同様の処理を行います。

改行は、文字列に0x0Aを指定した場合と、描画文字の右端が処理範囲の右端を越える場合に行われます。改行後の始点X座標は、処理範囲の始点X座標になります。また、改行した時に描画文字の下端が処理範囲の下端をはみだした場合は、文字を描画せずに処理を終了します。



背景の色

文字の色

※ 『SetDrawMode()』 コマンドの color で指定します。

以下に、描画可能な文字の範囲及び文字コード対応表を示します。

文字の範囲	内容
0x01～0x09, 0x0B～0x1F	ダミー文字
0x20～0x7E	半角英数字、記号
0x7F～0x9F	半角スペース
0xA0～0xDF	半角カタカナ、記号
0xE0～0xFF	半角スペース
第1バイト: 0x81～0x9F, 0xE0～0xEF 第2バイト: 0x40～0x7E, 0x80～0xFC	全角スペース

		下位4ビット															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
上 位 4 ビ ット	0		◆	⦿	H _T	F _F	C _R	L _F	○	±	N _L	⌈	⌋	⌈	⌋	⌈	⌋
	1	—	—	—	—	—	—	—	—	—	—	≤	≥	π	≠	£	·
	2		!	”	#	\$	%	&	'	(	)	*	+	,	-	.	/
	3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	5	P	Q	R	S	T	U	V	W	X	Y	Z	[	¥	]	^	_
	6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
	7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
	8																
	9																
	A		。	「	」	、	・	ヲ	ア	イ	ウ	エ	オ	ヤ	ユ	ヨ	ツ
	B	ー	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ
	C	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ
	D	ミ	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ン	、	。
	E																
	F																

2.23.5 直線描画 (DrawLine)

□ C言語API

```
int ret = DrawLine(int sx, int sy, int ex, int ey);
```

□ パラメータ

int	sx	線分の始点X座標 (ウィンドウ相対)
int	sy	線分の始点Y座標 (ウィンドウ相対)
int	ex	線分の終点X座標 (ウィンドウ相対)
int	ey	線分の終点Y座標 (ウィンドウ相対)

□ 対応画面タイプ/画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

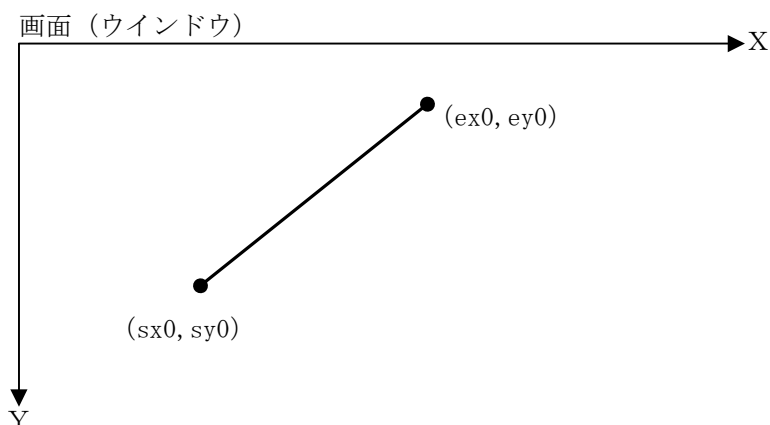
□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	描画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 0	始点X座標設定エラー
2 1	始点Y座標設定エラー
2 2	終点X座標設定エラー
2 3	終点Y座標設定エラー
3 4	画像メモリアクセスウィンドウ (SYS_WIN) 設定エラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

直線を描画します。処理範囲は画像メモリアクセスウィンドウ (SYS_WIN) となります。グラフィックスコマンドでの座標系はウィンドウ相対であり、ウィンドウの始点が座標 (0、0) になります。



2.23.6 直線描画(複数) (DrawSegments)

□ C言語API

```
int ret = DrawSegments(IPX_SEGMENT *segments, int ns);
```

□ パラメータ

IPX_SEGMENT	*segments	線分設定テーブルを格納したアドレス
int	ns	描画する線分数 (1 以上)

□ 対応画面タイプ/画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ パケットの構造

```
typedef struct {
    int    sx, sy;    線分の始点座標 (ウィンドウ相対)
    int    ex, ey;    線分の終点座標 (ウィンドウ相対)
} IPX_SEGMENT;
```

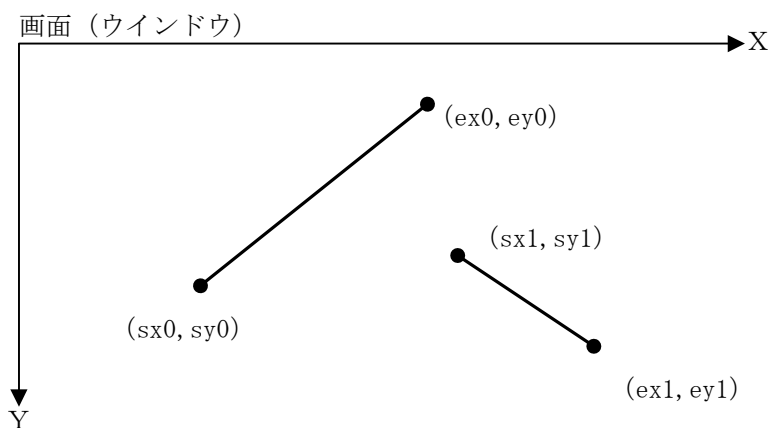
□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	描画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 0	線分設定テーブルを格納したアドレスが不正
2 1	描画線分数が0以下
3 4	画像メモリアクセスウィンドウ (SYS_WIN) 設定エラー
4 4	座標設定エラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

複数の直線を描画します。処理範囲は画像メモリアクセスウィンドウ (SYS_WIN) となります。グラフィックスコマンドでの座標系はウィンドウ相対であり、ウィンドウの始点が座標 (0、0) になります。



2.23.7 折れ線描画 (DrawLines)

□ C言語API

```
int ret = DrawLines(IPX_POINT *points, int np);
```

□ パラメータ

IPX_POINT	*points	線分設定テーブルを格納したアドレス
int	np	座標数 (2 以上)

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ パケットの構造

```
typedef struct {
    int x, y;          線分の始点及び終点座標 ( ウィンドウ相対 )
} IPX_POINT ;
```

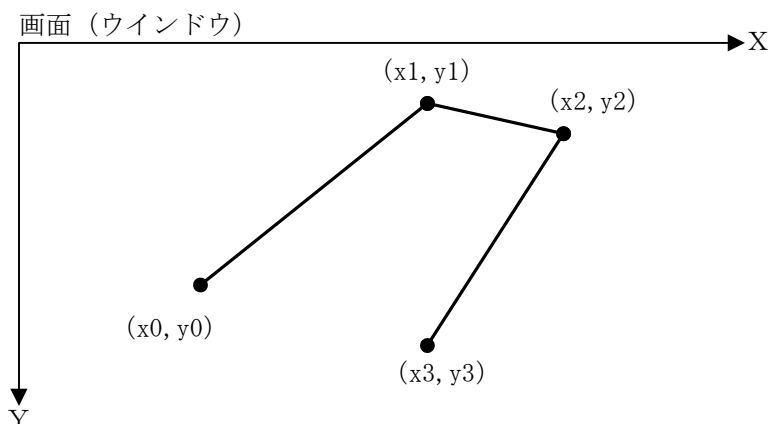
□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	V I N入力タイムアウト
1 0	描画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 0	線分設定テーブルを格納したアドレスが不正
2 1	設定座標数が 1 以下
3 4	画像メモリアクセスウィンドウ (SYS_WIN) 設定エラー
4 4	座標設定エラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

折れ線を描画します。処理範囲は画像メモリアクセスウィンドウ (SYS_WIN) となります。グラフィックスコマンドでの座標系はウィンドウ相対であり、ウィンドウの始点が座標 (0、0) になります。座標数 (np) には “ 2 ” 以上を設定して下さい。 “ 2 ” 未満を設定するとエラーとなります。



2.23.8 矩形描画 (DrawRectangle)

□ C言語API

```
int ret = DrawRectangle(int x, int y, int width, int height);
```

□ パラメータ

int	x	始点X座標 (ウィンドウ相対)
int	y	始点Y座標 (ウィンドウ相対)
int	width	幅
int	height	高さ

□ 対応画面タイプ/画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

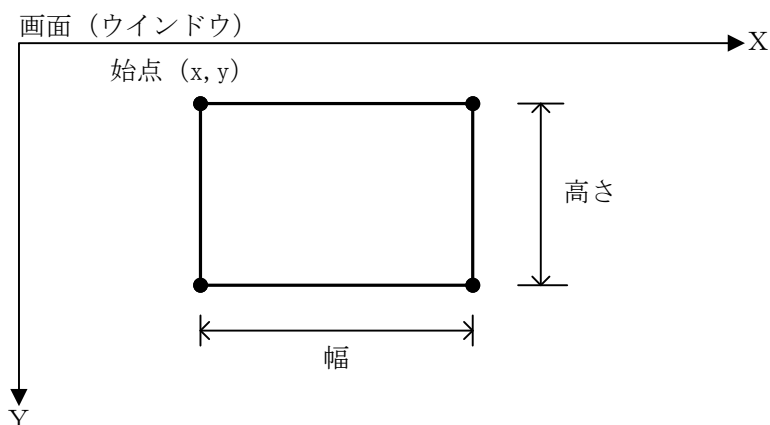
□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	描画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
20	始点X座標設定エラー
21	始点Y座標設定エラー
22	幅設定エラー
23	高さ設定エラー
34	画像メモリアクセスウィンドウ (SYS_WIN) 設定エラー
—	エラーリセットコマンド (ClearIPErrors) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

矩形を描画します。処理範囲は画像メモリアクセスウィンドウ (SYS_WIN) となります。グラフィックスコマンドでの座標系はウィンドウ相対であり、ウィンドウの始点が座標 (0、0) になります。幅、高さを1と指定したときは、始点座標へ1点を描画します。



2.23.9 多角形描画 (DrawPolygon)

□ C言語API

```
int ret = DrawPolygon(IPX_POINT *points, int np);
```

□ パラメータ

IPX_POINT	*points	線分設定テーブルを格納したアドレス
int	np	座標数 (2 以上)

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ パケットの構造

IPX_POINT構造体については、『DrawLines()』コマンドを参照して下さい。

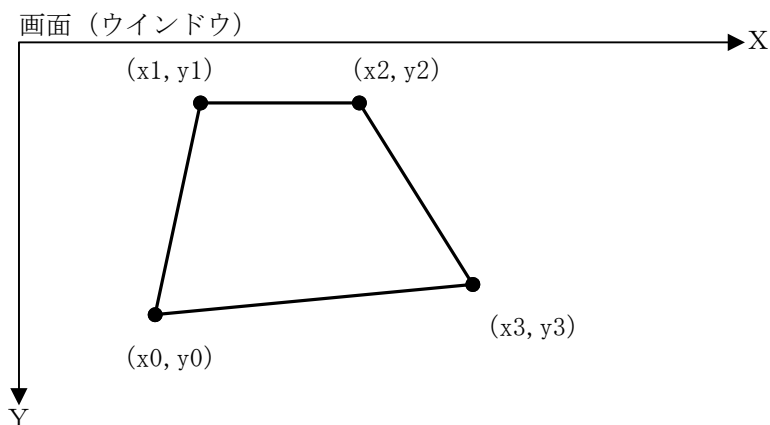
□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	描画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 0	線分設定テーブルを格納したアドレスが不正
2 1	設定座標数が1以下
3 4	画像メモリアクセスウィンドウ (SYS_WIN) 設定エラー
4 4	座標設定値エラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

多角形を描画します。処理範囲は画像メモリアクセスウィンドウ (SYS_WIN) となります。グラフィックスコマンドでの座標系はウィンドウ相対であり、ウィンドウの始点が座標 (0、0) になります。始点座標と終点座標が一致しないときは自動的に始点座標と終点座標を結びます。



2.23.10 円弧描画 (DrawArc)

□ C言語API

```
int ret = DrawArc(int x, int y, int width, int height, int angle1, int angle2);
```

□ パラメータ

int	x	円弧を内接する長方形の左隅X座標 (ウィンドウ相対)
int	y	円弧を内接する長方形の左隅Y座標 (ウィンドウ相対)
int	width	円弧を内接する長方形の幅
int	height	円弧を内接する長方形の高さ
int	angle1	弧の開始角度 (°)
int	angle2	弧の開始から終了までの角度 (°)

□ 対応画面タイプ/画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

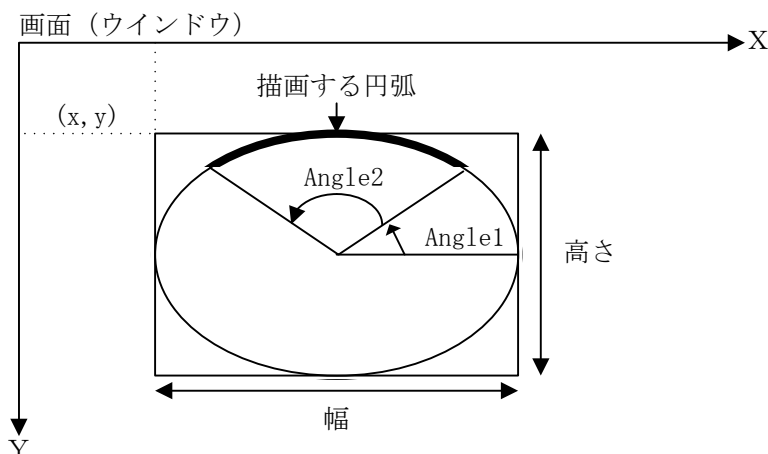
□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	描画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 0	X座標設定エラー
2 1	Y座標設定エラー
2 2	幅設定エラー (4 以下)
2 3	高さ設定エラー (4 以下)
2 4	弧の開始角度設定エラー
2 5	弧の存在する角度設定エラー
3 4	画像メモリアクセスウィンドウ (SYS_WIN) 設定エラー
4 5	パラメータの設定条件違反
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

円弧を描画します。処理範囲は画像メモリアクセスウィンドウ (SYS_WIN) となります。グラフィックスコマンドでの座標系はウィンドウ相対であり、ウィンドウの始点が座標 (0、0) になります。



2.23.11 矩形描画(塗りつぶし) (FillRectangle)

□ C言語API

```
int ret = FillRectangle( int x , int y , int width , int height);
```

□ パラメータ

int	x	始点 X 座標

int	y	始点 Y 座標

int	width	幅

int	height	高さ

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2 0	始点 X 座標設定エラー
2 1	始点 Y 座標設定エラー
2 2	幅設定エラー
2 3	高さ設定エラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

矩形 (塗りつぶし) を描画します。

幅、高さを 0 に指定した場合は、始点座標 (1 画素) を指定カラーにします。

2.23.12 多角形描画(塗りつぶし) (FillPolygon)

□ C言語API

```
int ret = FillPolygon( IPX_POINT *points , int np);
```

□ パラメータ

IPX_POINT	*points	線分設定テーブルのポインタ
int	np	座標数

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ パケットの構造

IPX_POINT構造体については、『DrawLines()』コマンドを参照して下さい。

□ エラーコード

2 1	設定座標数が 1 以下
4 4	座標設定エラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

多角形 (塗りつぶし) を描画します。そのとき、始点座標と終点座標が一致しないときは自動的に始点座標と終点座標を結び、塗りつぶします。

2.24 画像ファイリング

2.24.1 BMPファイルのロード (LoadBMPFile)

□ C言語API

```
int ret = LoadBMPFile(IMGID ImgID, char *filename);
```

□ パラメータ

IMGID	ImgID	転送先の画面番号
char	*filename	BMPファイル名を格納したアドレス ファイル名は拡張子まで含みます

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	○	○	○	○	○	○	○	○

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

1 0	画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 0 0	画像ファイルオープンエラー
1 0 1	不当画像ファイルエラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

BMPファイルの画像データを変換し、ImgIDで示される画面へロードします。
画面内の転送領域は画像メモリアクセスウインドウ (SYS_WIN) で指定します。ただし、データ転送領域の始点X座標を偶数、終点X座標を奇数に設定する必要があります。
また、ファイル名の拡張子には『.bmp』を指定して下さい。

以下に示す条件でビットマップデータを画像メモリのデータに変換し、ロードします。

BMP ファイル 画面	パレット 変換	B/W (モノクロ)	YUV	RGB (24/32ビット)
モノクロ (Y)	有	パレットでB/WデータをYUVデータに変換し、YデータをY画面にロードする	—	—
	無	B/WデータをY画面にロードする	YデータをY画面にロードする	RGBデータをYデータに変換し、Y画面にロードする
YUV	有	パレットでB/WデータをYUVデータに変換し、YUV画面にロードする	—	—
	無	B/WデータをY画面にロードする。UV画面は128でクリアする	YUVデータをYUV画面にロードする	RGBデータをYUVデータに変換し、YUV画面にロードする
CYUV	有	パレットでB/WデータをCYUVデータに変換し、CYUV画面にロードする	—	—
	無	B/WデータをY画面にロードする。UV画面は128でクリアする	YUVデータをCYUV画面にロードする	RGBデータをCYUVデータに変換し、CYUV画面にロードする
RGB	有	パレットでB/WデータをRGBデータに変換し、RGB画面にロードする	—	—
	無	B/WデータをRGB画面(R, G, Bそれぞれ)にロードする。	エラー	RGBデータをRGB画面にロードする。
RGB_R	有	パレットでB/WデータをYUVデータに変換し、YデータをRGB_R画面にロードする	—	—
	無	B/WデータをRGB_R画面にロードする。	YデータをRGB_R画面にロードする	RGBデータをYデータに変換し、RGB_R画面にロードする
RGB_G	有	パレットでB/WデータをYUVデータに変換し、YデータをRGB_G画面にロードする	—	—
	無	B/WデータをRGB_G画面にロードする。	YデータをRGB_G画面にロードする	RGBデータをYデータに変換し、RGB_G画面にロードする
RGB_B	有	パレットでB/WデータをYUVデータに変換し、YデータをRGB_B画面にロードする	—	—
	無	B/WデータをRGB_B画面にロードする。	YデータをRGB_B画面にロードする	RGBデータをYデータに変換し、RGB_B画面にロードする
RGB16	有	パレットでB/WデータをRGB16データに変換し、RGB16データをRGB16画面にロードする	—	—
	無	B/WデータをRGB16画面にロードする。	エラー	RGBデータをRGB16に変換し、RGB16画面にロードする

※ パレット変換有. . . BMPファイルのパレット情報が標準(R、G、Bがそれぞれ0～255)と異なる場合

※ YUV画面、CYUV画面には対応していません。

画像メモリには以下のファイル形式のデータをロードすることができます。

- B/W（モノクロ）ビットマップ（ビットマップファイル準拠）
Y画像（8ビット階調）をそのままのイメージで画像メモリにロード
- YUVビットマップ（画像認識ライブラリ専用フォーマット）
Y画像とUV画像（各8ビット階調）をそのままのイメージで画像メモリにロード
- RGBビットマップ（ビットマップファイル準拠）
R、G、B成分（各8ビット階調）に変換し画像メモリにロード

注1）白黒（1ビット）ビットマップ、16色（4ビット）ビットマップ画像についてはサポートしていません。

注2）圧縮画像についてはサポートしていません。

注3）256色ビットマップファイルをYUV画面に書き込んだ場合、色が正常に表示されません。
カラービットマップファイルデータをYUV画面に書き込む場合は
24ビットビットマップファイルを使用して下さい。

注4）ファイルの画像データが画像メモリの転送領域より大きい場合はクリッピングされます。

2.24.2 BMPファイルのセーブ（SaveBMPFile）

□ C言語API

```
int ret = SaveBMPFile(IMGID ImgID, char *filename, enum BITMAP_MODE mode);
```

□ パラメータ

IMGID	ImgID	転送元の画面番号
char	*filename	BMP ファイル名を格納したアドレス ファイル名は拡張子まで含みます
enum BITMAP_MODE	mode	BMP ファイル転送モード
	BW_BITMAP	(0) BWビットマップ形式
	RGB_BITMAP	(1) RGBビットマップ形式
	YUV_BITMAP	(2) YUVビットマップ形式

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	○	○	○	○	○	○	○	○

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

1 0	画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
2 2	転送モード指定範囲外
4 9	セーブを行う画面が不定画面
1 0 0	画像ファイルオープンエラー
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ImgIDで示される画面内の画像データをBMPファイルに変換しセーブします。B/W（モノクロ）ビットマップ、RGBビットマップはWindows標準画像ファイルとして使用できます。

画面内の転送領域は画像メモリアクセスウインドウ（SYS_WIN）で指定します。ただし、データ転送領域の始点X座標を偶数、終点X座標を奇数に設定する必要があります。

また、ファイル名の拡張子には『.bmp』を指定して下さい。

以下に示す条件で画像メモリのデータをビットマップデータに変換し、セーブします

mode 画面	BW_BITMAP	RGB_BITMAP	YUV_BITMAP
モノクロ (Y)	Y画面をセーブする	Y画面をセーブする	Y画面をセーブする
YUV	Y画面のみセーブする	YUV画面をRGBデータに変換し、セーブする	Y画面を1面目、UV画面を2面目にセーブする
CYUV	Yデータをセーブする	CYUV画面をRGBデータに変換し、セーブする	Yデータをセーブする
RGB	RGB画面をYデータに変換し、セーブする	RGB画面をセーブする	RGB画面をYデータに変換し、Y画像のみセーブする
RGB_R	RGB_R画面をセーブする	RGB_R画面をセーブする	RGB_R画面をセーブする
RGB_G	RGB_G画面をセーブする	RGB_G画面をセーブする	RGB_G画面をセーブする
RGB_B	RGB_B画面をセーブする	RGB_B画面をセーブする	RGB_B画面をセーブする
RGB16	RGB16画面をYデータに変換し、セーブする	RGB16画面をRGBデータに変換し、セーブする	RGB16画面をYデータに変換し、セーブする

※ YUV画面、CYUV画面には対応していません。

画像メモリのデータをビットマップファイルとしてセーブする場合は以下のファイル形式を選択できます。

- ・ B/W (モノクロ) ビットマップ
- ・ RGBビットマップ
- ・ YUV画像ファイル (画像認識ライブラリ専用フォーマット)
- ・ Windows 標準の256色ビットマップファイル/24ビットビットマップファイル

注1) YUVビットマップは画像認識ライブラリフォーマットのため、ペイント等のアプリケーションでファイルを開くことができません。

注2) 白黒 (1ビット) ビットマップ、16色 (4ビット) ビットマップ画像についてはサポートしていません。

注3) 圧縮画像についてはサポートしていません。

注4) 256色ビットマップファイルをYUV画面に書き込んだ場合、色が正常に表示されません。カラービットマップファイルデータをYUV画面に書き込む場合は24ビットビットマップファイルを使用して下さい。

注5) ファイルの画像データが画像メモリの転送領域より大きい場合はクリッピングされます。

注6) カラー画像をRGBビットマップで保存する場合、高輝度/高濃色部分でデータの階調が足りなくなる場合があります。この部分のデータはクリッピングされますので元の画像イメージと多少異なります。

2.25 構成制御

2.25.1 構成制御 (SetConfigCamera)

□ C言語API

```
int ret = SetConfigCamera( enum CfgCameraType type, void *para, int len );
```

□ パラメータ

enum CfgCameraType type	映像入力種別
CFG_VP110_CAMERA	(0) NVP-Ax230からのカメラ映像入力 (デフォルト設定)
CFG_VFW_CAMERA	(1) キャプチャデバイスからのカメラ映像入力
CFG_AVI_FILE	(2) 動画ファイルからのカメラ映像入力
CFG_BMP_LIST	(3) BMPリストで指定されたファイルからのカメラ映像入力
CFG_NO_INPUT	(256) カメラからの入力をスルー (実行しない) 状態にする

void	*para	映像入力構成制御テーブルを返すアドレス ・CFG_VP110_CAMERA、CFG_NO_INPUTの場合、NULLを指定して下さい。 ・CFG_VFW_CAMERA、CFG_AVI_FILE、CFG_BMP_LISTの場合、ConfigVfwPara構造体にパラメータを設定し、その先頭アドレスを指定して下さい。
int	len	映像入力構成制御テーブルのバイトサイズ

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ パケットの構造

```
typedef struct{
    char    *name;          ファイル名 (CFG_AVI_FILE、CFG_BMP_LIST)
                                ファイルへのパスを含みます
                                デバイス番号 (CFG_VFW_CAMERA)
                                NULLを指定するとデバイス番号が0に設定されます
    int     loop_mode;      最終フレームに達したときの処理
                                0 : 先頭フレームに戻ります
                                1 : GetCameraの映像入力を停止します。このとき、
                                    GetCameraは1を返します
    int     sx;             映像入力画面の始点X座標 (0～2047)
    int     sy;             映像入力画面の始点Y座標 (0～2047)
    int     xlng;           映像入力画面のX方向の長さ (1～2048)
    int     ylng;           映像入力画面のY方向の長さ (1～2048)
} ConfigVfwPara;
```

□ エラーコード

2 0	映像入力種別設定範囲外
2 1	映像入力構成制御テーブル設定範囲外
2 2	映像入力構成制御テーブルのバイトサイズが不正
1 1 0	『VFWCam2.exe』起動タイムアウト
1 1 1	『VFWCam2.exe』起動エラー(オブジェクトがすでに放棄されている)
1 1 2	『VFWCam2.exe』起動エラー(ファイルが見つからない)
1 2 0	WindowsAPIコマンドエラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

映像入力構成を設定します。映像入力種別はデフォルトでCFG_VP110_CAMERAが選択されています。

- 全共通
 - (1) 映像入力画面と画像メモリの画面サイズが異なるときは、AND領域に画像データを入力します。
- 映像入力種別がCFG_VP110_CAMERAのときは、
 - (1) NVP-Ax230から直接映像を入力します。
 - (2) 『ExitIPCamera』コマンド以外のすべての映像入力コマンドが実行できます
- 映像入力種別がCFG_VFW_CAMERA、またはCFG_AVI_FILE、CFG_BMP_LISTのときは、
 - (1) Microsoft® DirectShow® により、CFG_VFW_CAMERAのときはキャプチャデバイスからのカメラ映像入力、CFG_AVI_FILEのときは動画ファイルのキャプチャ、CFG_BMP_LISTのときはビットマップファイル入力を行います。
 - (2) 『GetCamera』コマンドと『ExitIPCamera』コマンドのみ有効になります。
 - (3) 別プロセスにてカメラ映像入力プログラム (VFWCam2.exe) が起動します。
 - (4) カメラ映像入力プログラムを終了する場合は、『ExitIPCamera』コマンドを実行するか、または終了ボタンを押して下さい。
 - (5) ビデオ画面サイズは映像入力構成制御テーブル (ConfigVfwPara) で設定します。

※1 『SelectCamera』コマンド、『SetVideoFrame』コマンドで設定した値は引き継がれません。

※2 Microsoft® DirectShow®の映像入力画面の長さは動画ファイルに依存しています。そのため、動画ファイルによっては任意サイズ指定ができない場合があります。

※3 BMPリストおよび『VFWCam2.exe』については別紙『VFWCam操作説明書』を参照して下さい。

(例1) 映像入力種別がCFG_VFW_CAMERAのとき

```
ConfigVfwPara VfwConfig;
memset(&VfwConfig, 0, sizeof(VfwConfig));
VfwConfig.name = NULL;    <--- NULLを指定するとデバイス番号が0に設定されます
VfwConfig.loop_mode= 0;
VfwConfig.sx      = 0;
VfwConfig.sy      = 0;
VfwConfig.xlng    = 640;
VfwConfig.ylng    = 480;
SetConfigCamera( CFG_VFW_CAMERA, &VfwConfig, sizeof(VfwConfig) );
GetCamera( imgid01 );
```

(例2) 映像入力種別がCFG_AVI_FILEのとき

```
ConfigVfwPara VfwConfig;
memset(&VfwConfig, 0, sizeof(VfwConfig));
VfwConfig.name = "sample.avi";
VfwConfig.loop_mode= 0;
VfwConfig.sx      = 0;
VfwConfig.sy      = 0;
VfwConfig.xlng    = 640;
VfwConfig.ylng    = 480;
SetConfigCamera( CFG_AVI_FILE, &VfwConfig, sizeof(VfwConfig) );
GetCamera( imgid01 );
```

- 映像入力種別がCFG_NO_INPUTのときは、
 - (1) カメラからの映像入力を行いません。
 - (2) 『ExitIPCamera』コマンド以外のすべての映像入力コマンドが無効になります。

2.25.2 カメラ映像入力プログラム終了（ExitIPCamera）

□ C言語API

```
int ret = ExitIPCamera();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

1 1 3	『VFWCam2.exe』 ツール通信エラー
1 3 8	『VFWCam2.exe』 ツール未起動
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

NVP-Ax230／Microsoft® DirectShow®（動画ファイル）からのカメラ映像入力プログラムのプロセスを終了します。本コマンドが正常終了した場合は、映像入力種別がNVP-Ax230からのカメラ映像入力（CFG_VP110_CAMERA）に設定されます。

カメラ映像入力プログラムが終了している状態で本コマンドを実行した場合はエラーになります。

2.25.3 画像メモリ表示構成制御設定 (SetConfigView)

□ C言語API

```
int ret = SetConfigView(enum CfgViewType type, void *para, int len);
```

□ パラメータ

enum CfgViewType type	画像メモリ表示種別
CFG_DISP_VP110 (0)	NVP-Ax230のRGB出力画面に表示する (デフォルト設定)
CFG_DISP_VIEW (1)	画面番号の画像データをパソコン上の画面に表示する
CFG_NO_OUTPUT (256)	画像メモリ表示処理をスルー (実行しない) 状態にする
void *para	画像メモリ表示構成制御テーブルを返すアドレス ・CFG_DISP_VP110、CFG_NO_OUTPUTの場合、NULLを指定して下さい。 ・CFG_DISP_VIEWの場合ConfigViewPara構造体にパラメータを設定し、その先頭アドレスを指定して下さい。
int len	画像メモリ表示構成制御テーブルのバイトサイズ

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret	正常終了 (0) または異常終了 (-1)

□ パケットの構造

```
typedef struct {
    int type;           映像出力形式 (未使用)
                        0を設定して下さい
    int xlng;           画像メモリ表示画面のX方向の長さ (1～2048)
    int ylng;           画像メモリ表示画面のY方向の長さ (1～2048)
} ConfigViewPara;
```

□ エラーコード

2 0	画像メモリ表示種別設定範囲外
2 1	画像メモリ表示構成制御テーブル設定範囲外
2 2	画像メモリ表示構成制御テーブルのバイトサイズが不正
1 1 0	『IPView2.exe』 起動タイムアウト
1 1 1	『IPView2.exe』 起動エラー (オブジェクトがすでに放棄されている)
1 1 2	『IPView2.exe』 起動エラー (ファイルが見つからない)
1 1 3	『IPView2.exe』 ツール通信エラー
1 2 0	WindowsAPIコマンドエラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

画像メモリ表示構成を設定します。画像メモリ表示種別はデフォルトでCFG_DISP_VP110が選択されています。

● 全共通

(1) 画像メモリ表示画面と画像メモリの画面サイズが異なるときは、AND領域に画像データを出力します。

● 画像メモリ表示種別がCFG_DISP_VP110のときは、

(1) NVP-Ax230からRGB出力画面に映像を表示します。
(2) 『ExitIPView』コマンド以外のすべての映像出力コマンドが実行できます。

- 画像メモリ表示種別がCFG_DISP_VIEWのときは、
 - (1) NVP-Ax230からパソコン上の画面に映像を出力します。
 - (2) 『DispImg』 コマンドと 『ExitIPView』 コマンドのみ有効になります。
 - (3) 別プロセスにて画像メモリ表示プログラム (IPView2.exe) が起動します。
 - (4) 画像メモリ表示プログラムを終了する場合は、 『ExitIPView』 コマンドを実行するか、または終了ボタンを押して下さい。
- ※1 画像メモリ表示プログラムが起動している状態で本コマンドを実行した場合、画像メモリ表示プログラムにはパラメータが反映されません。パラメータを反映させるためには一度画像メモリ表示プログラムを終了する必要があります。
- ※2 『IPView2.exe』 については別紙『IPView2操作説明書』を参照して下さい。
- 映像入力種別がCFG_NO_OUTPUTのときは、
 - (1) カメラからの映像出力を行いません。
 - (2) 『ExitIPView』 コマンド以外のすべての映像出力コマンドが無効になります。
- ※1 表示の停止は行いませんので、すでにDispImgでRGB出力画面に映像表示を行っている場合は引き続き表示が行われます。

2.25.4 画像メモリ表示プログラム終了（ExitIPView）

□ C言語API

```
int ret = ExitIPView();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

1 1 3	『IPView2.exe』 ツール通信エラー
1 3 8	『IPView2.exe』 ツール未起動
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

画像メモリ表示プログラム（IPView2.exe）のプロセスを終了します。本コマンドが正常終了した場合は、画像メモリ表示種別がNVP-Ax230からRGB出力画面への映像表示（CFG_DISP_VP110）に設定されます。画像メモリ表示プログラムが終了している状態で本コマンドを実行した場合はエラーになります。

□ 機能

ソース画面（2値）に対し、4連結でラベリング処理と基本特徴量の抽出を行います。本コマンドでは、254個までのラベル特徴量を抽出可能です。それ以上のラベル特徴量の抽出が必要な場合は、IP_LabelCombine()コマンドを使用して下さい。

ラベル付けの最大数は254個です。ラベルオーバーフローは、ラベル数が255個以上、仮ラベル数が512個以上のいずれかで発生します。

基本特徴量テーブルのフォーマット及び内容を下記に示します。

ipxsys.h で宣言しています。

```
typedef struct {
    long    AREA;           /* 面積          */
    short   LEFT_UP_POINT_X; /* 最小X座標     */
    short   LEFT_UP_POINT_Y; /* 最小Y座標     */
    short   RIGHT_DOWN_POINT_X; /* 最大X座標    */
    short   RIGHT_DOWN_POINT_Y; /* 最大Y座標    */
    float   CENTER_of_GRAVITY_X; /* 重心X座標    */
    float   CENTER_of_GRAVITY_Y; /* 重心Y座標    */
} IPLabelBasicTbl
```

基本特徴量のテーブルはラベル番号に対応させています。0番目は不定で、1番目から物体の特徴量が入ります。このため、特徴量テーブルは必ず出力最大数（254）+1 確保しておく必要があります。

本コマンドで得られる各座標はSRC0_WINで設定されたウィンドウからの相対座標になります。画面上の座標位置はそれぞれ

```
MinX = BasicTbl[ラベル番号].LEFT_UP_POINT_X + sx
MinY = BasicTbl[ラベル番号].LEFT_UP_POINT_Y + sy
MaxX = BasicTbl[ラベル番号].RIGHT_DOWN_POINT_X + sx
MaxY = BasicTbl[ラベル番号].RIGHT_DOWN_POINT_Y + sy
Gx   = BasicTbl[ラベル番号].CENTER_of_GRAVITY_X + sx
Gy   = BasicTbl[ラベル番号].CENTER_of_GRAVITY_Y + sy
```

```
MinX   : 最小X座標
MinY   : 最小Y座標
MaxX   : 最大X座標
MaxY   : 最大Y座標
Gx     : 重心X座標
Gy     : 重心Y座標
sx,sy  : SRC0_WIN(sx, sy)
```

となります。

□ 機能

ソース画面（2 値）に対し、8 連結でラベリング処理と基本特徴量の抽出を行います。本コマンドでは、2 5 4 個までのラベル特徴量を抽出可能です。それ以上のラベル特徴量の抽出が必要な場合は、IP_LabelCombine() コマンドを使用して下さい。

ラベル付けの最大数は 2 5 4 個です。ラベルオーバーフローは、ラベル数が 2 5 5 個以上、仮ラベル数が 5 1 2 個以上のいずれかで発生します。

基本特徴量テーブルのフォーマット及び内容については、IP_Label14byRL() コマンドを参照して下さい。

基本特徴量のテーブルはラベル番号に対応させています。0 番目は不定で、1 番目から物体の特徴量が入ります。このため、特徴量テーブルは必ず出力最大数（2 5 4）+1 確保しておく必要があります。

本コマンドで得られる各座標はSRC0_WINで設定されたウィンドウからの相対座標になります。詳細はIP_Label14byRL() コマンドを参照して下さい。

□ 機能

ソース画面（2 値）に対し、4 連結でラベリング処理を行い、指定しきい値で面積フィルタ処理後に、再度ラベル付けと基本特徴量の抽出を行います。本コマンドでは、2 5 4 個までのラベル特徴量を抽出可能です。それ以上のラベル特徴量の抽出が必要な場合は、IP_LabelCombine() コマンドを使用して下さい。

ラベル付けの最大数は 2 5 4 個です。ラベルオーバーフローは、ラベル数が 2 5 5 個以上、仮ラベル数が 5 1 2 個以上のいずれかで発生します。

基本特徴量テーブルのフォーマット及び内容については、IP_Label4byRL() コマンドを参照して下さい。

基本、拡張特徴量のテーブルはラベル番号に対応させています。0 番目は不定で、1 番目から物体の特徴量が入ります。このため、特徴量テーブルは必ず出力最大数（2 5 4）+1 確保しておく必要があります。

本コマンドで得られる各座標はSRCO_WINで設定されたウィンドウからの相対座標になります。詳細はIP_Label4byRL() コマンドを参照して下さい。

□ 機能

ソース画面（2値）に対し、8連結でラベリング処理を行い、指定しきい値で面積フィルタ処理後に、再度ラベル付けと基本特徴量の抽出を行います。本コマンドでは、254個までのラベル特徴量を抽出可能です。それ以上のラベル特徴量の抽出が必要な場合は、IP_LabelCombine() コマンドを使用して下さい。

ラベル付けの最大数は254個です。ラベルオーバーフローは、ラベル数が255個以上、仮ラベル数が512個以上のいずれかで発生します。

基本特徴量テーブルのフォーマット及び内容については、IP_Label4byRL() コマンドを参照して下さい。

基本、拡張特徴量のテーブルはラベル番号に対応させています。0番目は不定で、1番目から物体の特徴量が入ります。このため、特徴量テーブルは必ず出力最大数（254）+1 確保しておく必要があります。

本コマンドで得られる各座標はSRC0_WINで設定されたウィンドウからの相対座標になります。詳細はIP_Label4byRL() コマンドを参照して下さい。

2.26.5 ランレングス・面積フィルタ付ソートラベリング(4連結) (IP_Label4byRLwithAreaFLTSort)

□ C言語API

```
int ret = IP_Label4byRLwithAreaFLTSort( IMGID ImgSrc , IMGID ImgDst , long thrmin ,
    long thrmax , enum IP_Label_opt opt , enum IP_Label_Sort_opt Sort_opt ,
    IPLabelBasicTbl *BasicTbl , enum IP_Label_opt2 opt2 );
```

□ パラメータ

IMGID	ImgSrc	ソース画像
IMGID	ImgDst	デスティネーション画像
long	thrmin	面積フィルタしきい値 (最小値)
long	thrmax	面積フィルタしきい値 (最大値)
enum	IP_Label_opt opt	ラベリング処理オプション 1

ラベリング処理オプション 1	対応定数	内容
LABEL_OBJ	0	白を物体としてラベリング
LABEL_BKG	1	黒を物体としてラベリング

enum IP_Label_Sort_opt Sort_opt ソート処理オプション

ソート処理オプション	対応定数	内容
LABEL_SORT_MAX	0	面積でのソートを大きい順に行う
LABEL_SORT_MIN	1	面積でのソートを小さい順に行う

IPLabelBasicTbl *BasicTbl 基本特徴量テーブル。2 5 5 個以上確保して下さい。

enum IP_Label_opt2 opt2 ラベリング処理オプション 2

ラベリング処理オプション 2	対応定数	内容
LABEL_NO_DST	0	結果のデスティネーション画面未出力
LABEL_DST	1	結果のデスティネーション画面出力

2.26.6 ランレングス・面積フィルタ付ソートラベリング(8連結) (IP_Label8byRLwithAreaFLTSort)

□ C言語API

```
int ret = IP_Label8byRLwithAreaFLTSort( IMGID ImgSrc , IMGID ImgDst , long thrmin ,
                                         long thrmax , enum IP_Label_opt opt , enum IP_Label_Sort_opt Sort_opt ,
                                         IPLabelBasicTbl *BasicTbl , enum IP_Label_opt2 opt2 );
```

□ パラメータ

IMGID	ImgSrc	ソース画像	
IMGID	ImgDst	デスティネーション画像	
long	thrmin	面積フィルタしきい値 (最小値)	
long	thrmax	面積フィルタしきい値 (最大値)	
enum	IP_Label_opt	opt	ラベリング処理オプション 1

ラベリング処理オプション 1	対応定数	内容
LABEL_OBJ	0	白を物体としてラベリング
LABEL_BKG	1	黒を物体としてラベリング

enum	IP_Label_Sort_opt	Sort_opt	ソート処理オプション

ソート処理オプション	対応定数	内容
LABEL_SORT_MAX	0	面積でのソートを大きい順に行う
LABEL_SORT_MIN	1	面積でのソートを小さい順に行う

IPLabelBasicTbl	*BasicTbl	基本特徴量テーブル。255個以上確保して下さい。

enum	IP_Label_opt2	opt2	ラベリング処理オプション2

ラベリング処理オプション2	対応定数	内容
LABEL_NO_DST	0	結果のデスティネーション画面未出力
LABEL_DST	1	結果のデスティネーション画面出力

□ 機能

ソース画面（2値）に対し、4連結でラベリング処理と拡張特徴量の抽出を行います。本コマンドでは、254個までのラベル特徴量を抽出可能です。それ以上のラベル特徴量の抽出が必要な場合は、IP_LabelCombine()コマンドを使用して下さい。

ラベル付けの最大数は254個です。ラベルオーバーフローは、ラベル数が255個以上、仮ラベル数が512個以上のいずれかで発生します。

基本特徴量テーブルのフォーマット及び内容をご示します。

ipxsys.h で宣言しています。

```
typedef struct {
    long AREA; /* 面積 */
    short LEFT_UP_POINT_X; /* 最小X座標 */
    short LEFT_UP_POINT_Y; /* 最小Y座標 */
    short RIGHT_DOWN_POINT_X; /* 最大X座標 */
    short RIGHT_DOWN_POINT_Y; /* 最大Y座標 */
    float CENTER_of_GRAVITY_X; /* 重心X座標 */
    float CENTER_of_GRAVITY_Y; /* 重心Y座標 */
    float ANGLE; /* 慣性主軸とX軸とのなす角度 (degree) */
    float AXIS_LONG; /* 予備 */
    float AXIS_SHORT; /* 予備 */
    long Round_Length; /* 予備 */
} ILabelExtTbl;
```

拡張特徴量のテーブルは、IP_Label4byRL()コマンドの基本特徴量テーブルに拡張情報が追加されており、この拡張情報がIP_Label4byRL()コマンドとの相違点になっています。

拡張特徴量のテーブルはラベル番号に対応させています。0番目は不定で、1番目から物体の特徴量が入ります。このため、特徴量テーブルは必ず出力最大数（254）+1 確保しておく必要があります。

本コマンドで得られる各座標はSRC0_WINで設定されたウィンドウからの相対座標になります。画面上の座標位置はそれぞれ

```
MinX = ExtTbl[ラベル番号].LEFT_UP_POINT_X + sx
MinY = ExtTbl[ラベル番号].LEFT_UP_POINT_Y + sy
MaxX = ExtTbl[ラベル番号].RIGHT_DOWN_POINT_X + sx
MaxY = ExtTbl[ラベル番号].RIGHT_DOWN_POINT_Y + sy
Gx = ExtTbl[ラベル番号].CENTER_of_GRAVITY_X + sx
Gy = ExtTbl[ラベル番号].CENTER_of_GRAVITY_Y + sy
```

```
MinX      : 最小X座標
MinY      : 最小Y座標
MaxX      : 最大X座標
MaxY      : 最大Y座標
Gx        : 重心X座標
Gy        : 重心Y座標
sx,sy     : SRC0_WIN(sx, sy)
```

となります。

□ 機能

ソース画面（2 値）に対し、8 連結でラベリング処理と拡張特徴量の抽出を行います。本コマンドでは、2 5 4 個までのラベル特徴量を抽出可能です。それ以上のラベル特徴量の抽出が必要な場合は、IP_LabelCombine() コマンドを使用して下さい。

ラベル付けの最大数は 2 5 4 個です。ラベルオーバーフローは、ラベル数が 2 5 5 個以上、仮ラベル数が 5 1 2 個以上のいずれかで発生します。

拡張特徴量テーブルのフォーマット及び内容については、IP_Label4byRLExt() コマンドを参照して下さい。

拡張特徴量のテーブルは、IP_Label4byRL() コマンドの基本特徴量テーブルに拡張情報が追加されており、この拡張情報がIP_Label4byRL() コマンドとの相違点になっています。

拡張特徴量のテーブルはラベル番号に対応させています。0 番目は不定で、1 番目から物体の特徴量が入ります。このため、特徴量テーブルは必ず出力最大数（2 5 4）+1 確保しておく必要があります。

本コマンドで得られる各座標はSRC0_WINで設定されたウィンドウからの相対座標になります。

詳細はIP_Label4byRLExt() コマンドを参照して下さい。

□ 機能

ソース画面（2 値）に対し、4 連結でラベリング処理を行い、指定しきい値で面積フィルタ処理後に、再度ラベル付けと拡張特徴量の抽出を行います。本コマンドでは、2 5 4 個までのラベル特徴量を抽出可能です。それ以上のラベル特徴量の抽出が必要な場合は、IP_LabelCombine() コマンドを使用して下さい。

ラベル付けの最大数は 2 5 4 個です。ラベルオーバーフローは、ラベル数が 2 5 5 個以上、仮ラベル数が 5 1 2 個以上のいずれかで発生します。

拡張特徴量テーブルのフォーマット及び内容については、IP_Label4byRLExt() コマンドを参照して下さい。

拡張特徴量のテーブルはラベル番号に対応させています。0 番目は不定で、1 番目から物体の特徴量が入ります。このため、特徴量テーブルは必ず出力最大数（2 5 4）+1 確保しておく必要があります。

本コマンドで得られる各座標はSRC0_WINで設定されたウィンドウからの相対座標になります。詳細はIP_Label4byRLExt() コマンドを参照して下さい。

2.26.10 ランレングス・拡張面積フィルタ付きラベリング(8連結)

(IP_Label8byRLwithAreaFLTExt)

□ C言語API

```
int ret = IP_Label8byRLwithAreaFLTExt( IMGID ImgSrc , IMGID ImgDst , long thrmin , long thrmax ,
enum IP_Label_opt opt , IPLabelExtTbl *ExtTbl , enum IP_Label_opt2 opt2 );
```

□ パラメータ

IMGID	ImgSrc	ソース画像
IMGID	ImgDst	デスティネーション画像
long	thrmin	面積フィルタしきい値 (最小値)
long	thrmax	面積フィルタしきい値 (最大値)
enum	IP_Label_opt opt	ラベリング処理オプション 1

ラベリング処理オプション 1	対応定数	内容
LABEL_OBJ	0	白を物体としてラベリング
LABEL_BKG	1	黒を物体としてラベリング

IPLabelExtTbl *ExtTbl 拡張特徴量テーブル。2 5 5 個以上確保して下さい。

enum IP_Label_opt2 opt2 ラベリング処理オプション 2

ラベリング処理オプション 2	対応定数	内容
LABEL_NO_DST	0	結果のデスティネーション画面未出力
LABEL_DST	1	結果のデスティネーション画面出力

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B2	△B2	○
ImgDst	○	△R	○	○	○	×	◆	-	-

□ リターンパラメータ

int	ret	0 ≤ ret ≤ 2 5 4 : 正常終了 (ラベル数) - 1 : 異常終了 LABEL_OVERFLOW : ラベルオーバーフロー
IPLabelExtTbl	*ExtTbl	拡張特徴量を格納したアドレス

□ エラーコード

2	OSでのエラー(タイムアウト以外)
3	IMP処理実行タイムアウト
1 0 , 1 3	指定画面空き領域、または範囲外 (不当画面番号エラー)
2 2 , 2 3 , 2 4 , 2 5 , 2 6	オプション設定値範囲外
3 0	S R C O ウィンドウ幅エラー (ウィンドウ幅は2以上にして下さい)
3 3	D S T ウィンドウ幅エラー (ウィンドウ幅は2以上にして下さい)
4 8	動的メモリ確保エラー (ユーザープログラムによるメモリ破壊がないか確認して下さい)
4 9	アクセス画面が不定画面
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面（2値）に対し、8連結でラベリング処理を行い、指定しきい値で面積フィルタ処理後に、再度ラベル付けと拡張特徴量の抽出を行います。本コマンドでは、254個までのラベル特徴量を抽出可能です。それ以上のラベル特徴量の抽出が必要な場合は、IP_LabelCombine()コマンドを使用して下さい。

ラベル付けの最大数は254個です。ラベルオーバーフローは、ラベル数が255個以上、仮ラベル数が512個以上のいずれかで発生します。

拡張特徴量テーブルのフォーマット及び内容については、IP_Label4byRLExt()コマンドを参照して下さい。

拡張特徴量のテーブルはラベル番号に対応させています。0番目は不定で、1番目から物体の特徴量が入ります。このため、特徴量テーブルは必ず出力最大数（254）+1 確保しておく必要があります。

本コマンドで得られる各座標はSRC0_WINで設定されたウィンドウからの相対座標になります。詳細はIP_Label4byRLExt()コマンドを参照して下さい。

(IP Label4byRLwithAreaFLTSortExt)

□ C言語API

```
int ret = IP_Label4byRLwithAreaFLTSortExt( IMGID ImgSrc , IMGID ImgDst , long thrmin ,
long thrmax ,enum IP_Label_opt opt , enum IP_Label_Sort_opt Sort_opt ,
IPLabelExtTbl *ExtTbl , enum IP_Label_opt2 opt2 );
```

□ パラメータ

IMGID	ImgSrc	ソース画像	
IMGID	ImgDst	デスティネーション画像	
long	thrmin	面積フィルタしきい値 (最小値)	
long	thrmax	面積フィルタしきい値 (最大値)	
enum	IP_Label_opt	opt	ラベリング処理オプション 1

ラベリング処理オプション 1	対応定数	内容
LABEL_OBJ	0	白を物体としてラベリング
LABEL_BKG	1	黒を物体としてラベリング

enum IP_Label_Sort_opt Sort_opt ソート処理オプション

ソート処理オプション	対応定数	内容
LABEL_SORT_MAX	0	面積でのソートを大きい順に行う
LABEL_SORT_MIN	1	面積でのソートを小さい順に行う

IPLabelExtTbl *ExtTbl 拡張特徴量テーブル。255個以上確保して下さい。

enum	IP_Label_opt2	opt2	ラベリング処理オプション2

ラベリング処理オプション2	対応定数	内容
LABEL_NO_DST	0	結果のデスティネーション画面未出力
LABEL_DST	1	結果のデスティネーション画面出力

2.26.12 ランレングス・拡張面積フィルタ付ソートラベリング(8連結) (IP_Label8byRLwithAreaFLTSortExt)

□ C言語API

```
int ret = IP_Label8byRLwithAreaFLTSortExt( IMGID ImgSrc , IMGID ImgDst , long thrmin ,
long thrmax , enum IP_Label_opt opt , enum IP_Label_Sort_opt Sort_opt ,
IPLabelExtTbl *ExtTbl , enum IP_Label_opt2 opt2 );
```

□ パラメータ

IMGID	ImgSrc	ソース画像
IMGID	ImgDst	デスティネーション画像
long	thrmin	面積フィルタしきい値 (最小値)
long	thrmax	面積フィルタしきい値 (最大値)
enum	IP_Label_opt opt	ラベリング処理オプション 1

ラベリング処理オプション 1	対応定数	内容
LABEL_OBJ	0	白を物体としてラベリング
LABEL_BKG	1	黒を物体としてラベリング

enum IP_Label_Sort_opt Sort_opt ソート処理オプション

ソート処理オプション	対応定数	内容
LABEL_SORT_MAX	0	面積でのソートを大きい順に行う
LABEL_SORT_MIN	1	面積でのソートを小さい順に行う

IPLabelExtTbl *ExtTbl 拡張特徴量テーブル。2 5 5 個以上確保して下さい。

enum IP_Label_opt2 opt2 ラベリング処理オプション 2

ラベリング処理オプション 2	対応定数	内容
LABEL_NO_DST	0	結果のデスティネーション画面未出力
LABEL_DST	1	結果のデスティネーション画面出力

□ リターンパラメータ

int	ret	0 ≤ ret ≤ 254 : 正常終了 (ラベル数) - 1 : 異常終了
		LABEL_OVERFLOW : ラベルオーバーフロー
IPLabelExtTbl	*ExtTbl	拡張特徴量を格納したアドレス

2	OSでのエラー(タイムアウト以外)
3	IMP処理実行タイムアウト
1 0, 1 3	指定画面空き領域、または範囲外 (不当画面番号エラー)
2 2, 2 3, 2 4, 2 5, 2 6, 2 7	オプション設定値範囲外
3 0	SRC0 ウィンドウ幅エラー (ウィンドウ幅は2以上にして下さい)
3 3	DST ウィンドウ幅エラー (ウィンドウ幅は2以上にして下さい)
4 8	動的メモリ確保エラー (ユーザープログラムによるメモリ破壊がないか確認して下さい)
4 9	アクセス画面が不定画面
—	エラーリセットコマンド (ClearIPErrors) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

ソース画面（2 値）に対し、8 連結でラベリング処理を行い、指定しきい値で面積フィルタ処理後に、面積でソートして再度ラベル付けと拡張特徴量の抽出を行います。本コマンドでは、254 個までのラベル特徴量を抽出可能です。それ以上のラベル特徴量の抽出が必要な場合は、IP LabelCombine() コマンドを使用して下さい。

拡張特徴量のテーブルはラベル番号に対応させています。0番目は不定で、1番目から物体の特徴量が入ります。このため、特徴量テーブルは必ず出力最大数（254）+1 確保しておく必要があります。

本コマンドで得られる各座標はSRC0_WINで設定されたウィンドウからの相対座標になります。詳細はIP_Label4bvRLExt() コマンドを参照して下さい。

2.26.13 統合ラベリング (IP_LabelCombine)

□ C言語API

```
int ret = IP_LabelCombine( IMGID ImgSrc , IMGID ImgDst , enum IPCombineType type ,
                          IPBinarizeThr *binarizeThr , IPLabelCtl *labelCtl , IPCombineCtl *combineCtl ,
                          IPCombineTbl *combineTbl , IPCombineTblExt *combineTblExt );
```

□ パラメータ

IMGID ImgSrc ソース画像

IMGID ImgDst デスティネーション画像

enum IPCombineType type 統合ラベリングオプション

統合ラベリングオプション	内容
NORMAL_FEATURE	基本特徴量, ラベル画像を出力します
EXTRA_FEATURE	基本特徴量, 拡張特徴量, ラベル画像を出力します
NORMAL_FEATURE_NODST	基本特徴量のみ出力します デスティネーション画面にラベリング結果は出力しません
EXTRA_FEATURE_NODST	基本特徴量, 拡張特徴量を出力します デスティネーション画面にラベリング結果は出力しません

IPBinarizeThr *binarizeThr 2 値化しきい値

IPLabelCtl *labelCtl ラベリング管理テーブル

IPCombineCtl *combineCtl 統合管理テーブル。現在未サポートの機能です。
必ず以下のように設定して下さい。
(下記以外の設定値では正常動作しません)

```
combineCtl.including = NO_INCLUDE ;       /* 包含無効 */
combineCtl.width      = 0 ;               /* 統合無し */
combineCtl.height     = 0 ;               /* 統合無し */
```

IPCombineTbl *combineTbl (2 ～ 4 0 0 1) 基本特徴量テーブルのポインタ

IPCombineTblExt *combineTblExt (2 ～ 4 0 0 1) 拡張特徴量テーブルのポインタ

(注) 特徴量のテーブルサイズは、必ず
「ラベリング管理テーブルの出力最大数+1」
確保してください

□ リターンパラメータ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	△U	△U
ImgDst	○	△R	○	○	○	×	◆	-	-

int	ret	0 ≤ ret ≤ 2 5 4 : 正常終了 (ラベル数) - 1 : 異常終了
		LABEL_OVERFLOW : ラベルオーバーフロー
IPCombineTbl	*combineTbl	基本特徴量テーブルを格納したアドレス
IPCombineTblExt	*combineTblExt	拡張特徴量テーブルを格納したアドレス

2	OSでのエラー(タイムアウト以外)
3	IMP処理実行タイムアウト
1 0, 1 3	指定画面空き領域、または範囲外 (不当画面番号エラー)
2 2, 2 3, 2 4, 2 5, 2 6, 2 7	オプション設定値範囲外
3 0	S R C 0 ウィンドウ幅エラー (ウィンドウ幅は2以上にしてください)
3 3	D S T ウィンドウ幅エラー (ウィンドウ幅は2以上にしてください)
4 9	アクセス画面が不定画面
5 1	ワーク画面確保エラー (不要な画面を解放してリトライしてください)
—	エラーリセットコマンド (ClearIPErrors) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

ソース画面（多値）を2値化してラベリング処理を行い、統合処理を行ったラベルの特徴量を算出及びラベル画像を出力します。本コマンドでは、4000個（仮ラベル数）までのラベル特徴量を抽出可能です。

最大ラベル数は4000個ですが、あくまでも仮ラベルの最大数です。実際はその半分程度の2000個程度までのラベル数になると考えられます。また、実際ソース画像のラベル数により、処理時間が大幅に変化しますので注意して下さい。

2 値化しきい値構造体、ラベリング管理テーブル、統合管理テーブル、基本特徴量テーブル、拡張特徴量テーブルのフォーマット及び内容を下記に示します。
これらは `ipxdef.h`, `ipxsys.h` で宣言しています。

IPBinarizeThr : 2 値化しきい値

```
typedef struct {
    int thrmin;    /* 最小しきい値 ( 0～thrmax ) */
    int thrmax;    /* 最大しきい値 (thrmin ～255 ) */
} IPBinarizeThr;
```

IPLabelCtl : ラベリング管理テーブル

連結手法	内容
LABEL4	4 連結
LABEL8	8 連結

ソート手法	内容
LABEL_SORT_MAX	面積でのソートを大きい順に行う
LABEL_SORT_MIN	面積でのソートを小さい順に行う
LABEL_SORT_NON	出現順

```
typedef struct {
    int          max_num;      /* 出力最大数 ( 1~4000 )          */
    enum IPLabelConnect connect; /* 連結手法                      */
    int          area_min;     /* 最小面積しきい値 ( 1 ~ thrmax ) */
    int          area_max;     /* 最大面積しきい値 ( thrmin ~ INT_MAX */
    enum IP_Label_Sort_opt sort; /* ソート手法                    */
} IPLabelCtl;
```

IPCombineInc : 統合管理テーブル

包含による統合	内容
INCLUDE	包含有効
NO_INCLUDE	包含無効

```
typedef struct {
    int          width;      /* 統合幅範囲 (0 ~ 横ウインドウAND サイズ */
                                /*                      /*      =0 : 統合なし )          */
    int          height;     /* 統合高さ範囲 (0 ~ 縦ウインドウAND サイズ) */
    enum IPCombineInc including; /* 包含による統合          */
} IPCombineCtl;
```

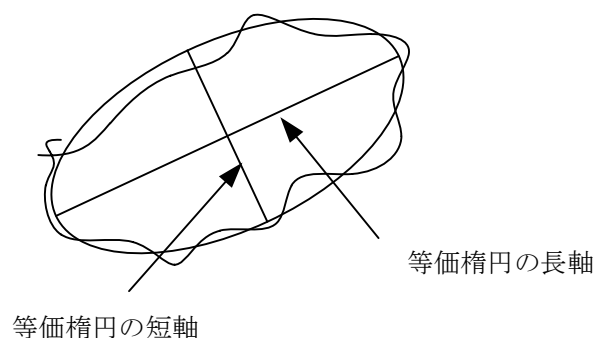
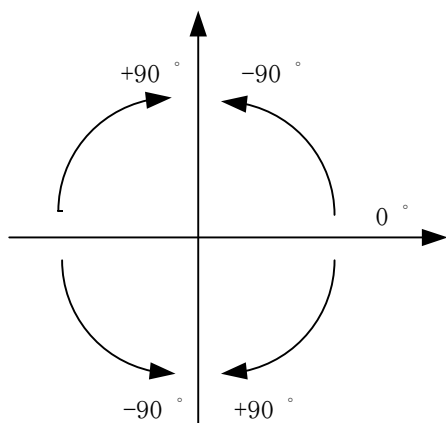
IPCombineTbl : 基本特徴量テーブルのポインタ

```
typedef struct {
    int      area;      /* 物体面積 */
    int      xmin;     /* 物体を囲む最小X座標 */
    int      ymin;     /* 物体を囲む最小Y座標 */
    int      xmax;     /* 物体を囲む最大X座標 */
    int      ymax;     /* 物体を囲む最大Y座標 */
    int      length;    /* 物体周囲長 */
} IPCombineTbl;
```

IPCombineTblExt : 拡張特徴量テーブルのポインタ

```
typedef struct {
    float     xmean;    /* 重心X座標 */
    float     ymean;    /* 重心Y座標 */
    float     tilt;     /* 物体傾き ( -90 ° ~ 90 ° ) (注) */
    float     axs_ratio; /* 短軸の長さ÷長軸の長さ (物体の等価楕円) */
} IPCombineTblExt;
```

(注)



- 基本、拡張特徴量のテーブルはラベル番号に対応させています。0番目は不定で、1番目から物体の特徴量が入ります。このため特徴量テーブルは、必ずラベリング管理テーブルで設定した出力最大数(max_num)+1 確保しておく必要があります。
- 仮ラベル付けの最大数は4 0 0 0個です。ラベルオーバーフローは、仮ラベル数4 0 0 1個以上で発生します。真ラベル付けの最大数は4 0 0 0個です。
- 現状でのラベル画像のデータタイプはUNSIGN8 なので、2 5 5番目以上のラベル毎の特徴量は出力されますが、ラベル画像には出力されません。
出力ラベル数が「0」の時、出力画面は不定になります。

2.27 2値化閾値算出支援

2.27.1 判別分析法 (HistAnalyze)

□ C言語API

```
int ret = HistAnalyze( HISTANATBL *tbl , int lower , int upper );
```

□ パラメータ

HISTANATBL *tbl 判別分析テーブル

int lower 下位レベル

int upper 上位レベル

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	≥ 0 : 抽出された2値化しきい値 -1 : 異常終了
HISTANATBL	*tbl	判別分析結果を格納したアドレス

□ エラーコード

2 0	パラメータエラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

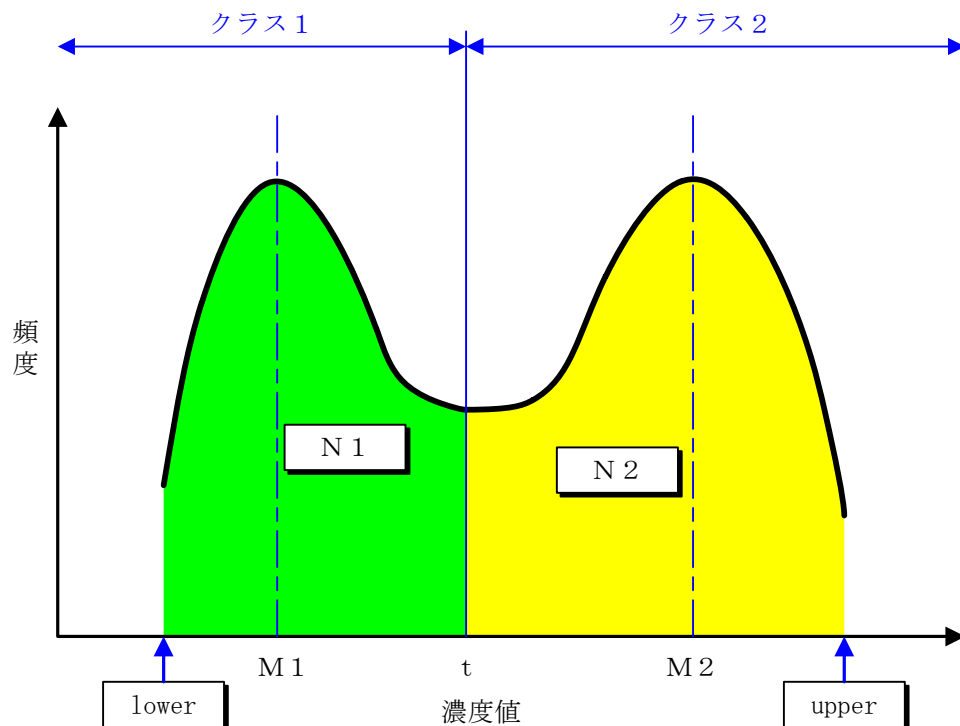
tblに設定されたヒストグラムテーブルの判別分析を行い、その特徴量を求めます。

HISTANATBLのフォーマット及び内容を、下記に示します。

ipxsys.hで宣言しています

```
typedef struct {
    long      HistTbl[256];
    long      n1;
    long      n2;
    int       m1;
    int       m2;
    int       thr;
    int       ct;
} HISTANATBL;
```

判別分析テーブル	説明
HistTbl[256] (入力)	ヒストグラムテーブル。Histogram() コマンドで得られるヒストグラムテーブルをコピーするか、または、このポインタをHistogram() コマンドのパラメータに設定して下さい。
n1 (出力)	クラス 1 の総画素数
n2 (出力)	クラス 2 の総画素数
m1 (出力)	クラス 1 の平均濃度値
m2 (出力)	クラス 2 の平均濃度値
thr (出力)	2 値化しきい値
ct (出力)	m 2 - m 1 (簡易コントラスト)



上図において

N 1 : クラス 1 の総画素数
 N 2 : クラス 2 の総画素数
 M 1 : クラス 1 の平均濃度値
 M 2 : クラス 2 の平均濃度値
 t : レベル

のとき 2 つのクラス間分散は、

$$\sigma^2 = N 1 \cdot N 2 \cdot (M 1 - M 2)^2$$

により求められます。そして、「t」の値を「lower」から「upper」まで変化させ、 σ を最大にする t の値を高速に計算しそのときのレベルをしきい値「thr」とする N 1, N 2, M 1, M 2 を求めます。

2.28 拡張コンボリューション

2.28.1 平滑化(5x5近傍) (IP_SmoothFLT5x5)

□ C言語API

```
int ret = IP_SmoothFLT5x5( IMGID ImgSrc , IMGID ImgDst , int scale , int *COEFF );
```

□ パラメータ

IMGID ImgSrc ソース画面の画面番号

IMGID ImgDst デスティネーション画面の画面番号

int scale ダウンシフト量 (0 ～ 1 5)

int *COEFF 荷重係数テーブルのポインタ

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

2 OSでのエラー(タイムアウト以外)

3 IMP処理実行タイムアウト

1 0 , 1 3 画面空き領域、または範囲外 (不当画面番号エラー)

2 2 ダウンシフト量設定エラー

2 3 荷重係数設定エラー

4 8 動的メモリ確保エラー
(ユーザープログラムによるメモリ破壊がないか確認して下さい)

3 0 SRC 0 ウィンドウ設定エラー

3 3 DST ウィンドウ設定エラー

4 9 アクセス画面が不定画面

— エラーリセットコマンド (ClearIPError) 未発行

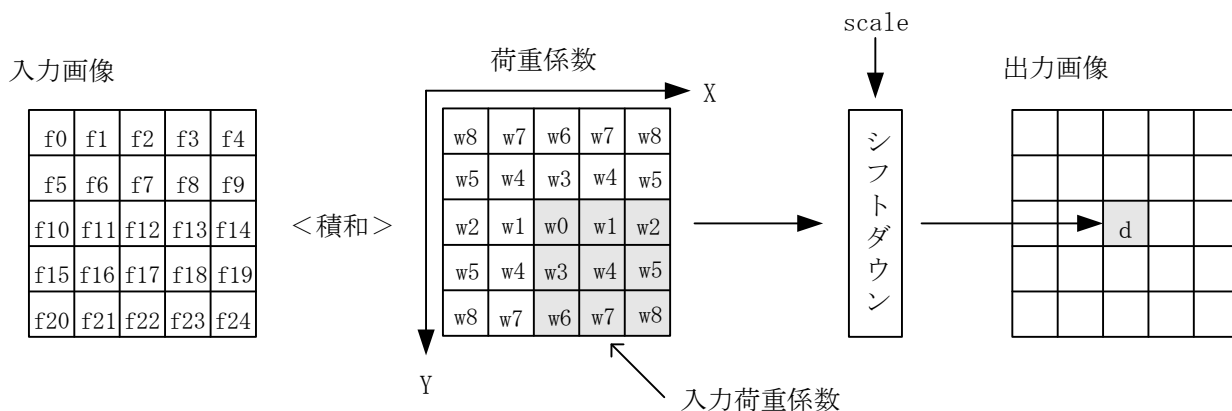
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

機能

ソース画面（濃淡）に対し、 5×5 近傍で平滑化を行います。

平滑化では、入力画面の指定領域内の全画素に対して、注目画素を中心とした 5×5 近傍の局所領域で与えられた荷重係数との積和演算を行います。

また、対象領域の周辺（内側 3 画素の領域）処理の場合は中央画素値を出力します。



$$\begin{aligned}
 d = & (f0 \times w8 + f1 \times w7 + f2 \times w6 + f3 \times w7 + f4 \times w8 \\
 & + f5 \times w5 + f6 \times w4 + f7 \times w3 + f8 \times w4 + f9 \times w5 \\
 & + f10 \times w2 + f11 \times w1 + f12 \times w0 + f13 \times w1 + f14 \times w2 \\
 & + f15 \times w5 + f16 \times w4 + f17 \times w3 + f18 \times w4 + f19 \times w5 \\
 & + f20 \times w8 + f21 \times w7 + f22 \times w6 + f23 \times w7 + f24 \times w8) \\
 & \gg \text{scale}
 \end{aligned}$$

係数テーブルのフォーマット及び荷重係数の例を、下記に示します。

荷重係数

#0	#1	#2
#3	#4	#5
#6	#7	#8

	31	8	7	0
COEFF [0]	——			係数 # 0
[1]	——			# 1
[2]	——			# 2
[3]	——			# 3
[4]	——			# 4
[5]	——			# 5
[6]	——			# 6
[7]	——		▼	# 7
[8]	——			係数 # 8

< 例1 >

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

< 例2 >

0	0	0	0	0
0	3	3	3	0
0	3	3	3	0
0	3	3	3	0
0	0	0	0	0

この例の場合、ダウンシフト量は5です。

2.28.2 平滑化(7x7近傍) (IP_SmoothFLT7x7)

□ C言語API

```
int ret = IP_SmoothFLT7x7( IMGID ImgSrc , IMGID ImgDst , int scale , int *COEFF );
```

□ パラメータ

IMGID ImgSrc ソース画面の画面番号

IMGID ImgDst デスティネーション画面の画面番号

int scale ダウンシフト量 (0 ～ 15)

int *COEFF 荷重係数テーブルのポインタ

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

2 OSでのエラー(タイムアウト以外)

3 IMP処理実行タイムアウト

1 0 , 1 3 画面空き領域、または範囲外 (不当画面番号エラー)

2 2 ダウンシフト量設定エラー

2 3 荷重係数設定エラー

4 8 動的メモリ確保エラー

 (ユーザープログラムによるメモリ破壊がないか確認して下さい)

3 0 SRC0ウインドウ設定エラー

3 3 DSTウインドウ設定エラー

4 9 アクセス画面が不定画面

- エラーリセットコマンド (ClearIPError) 未発行

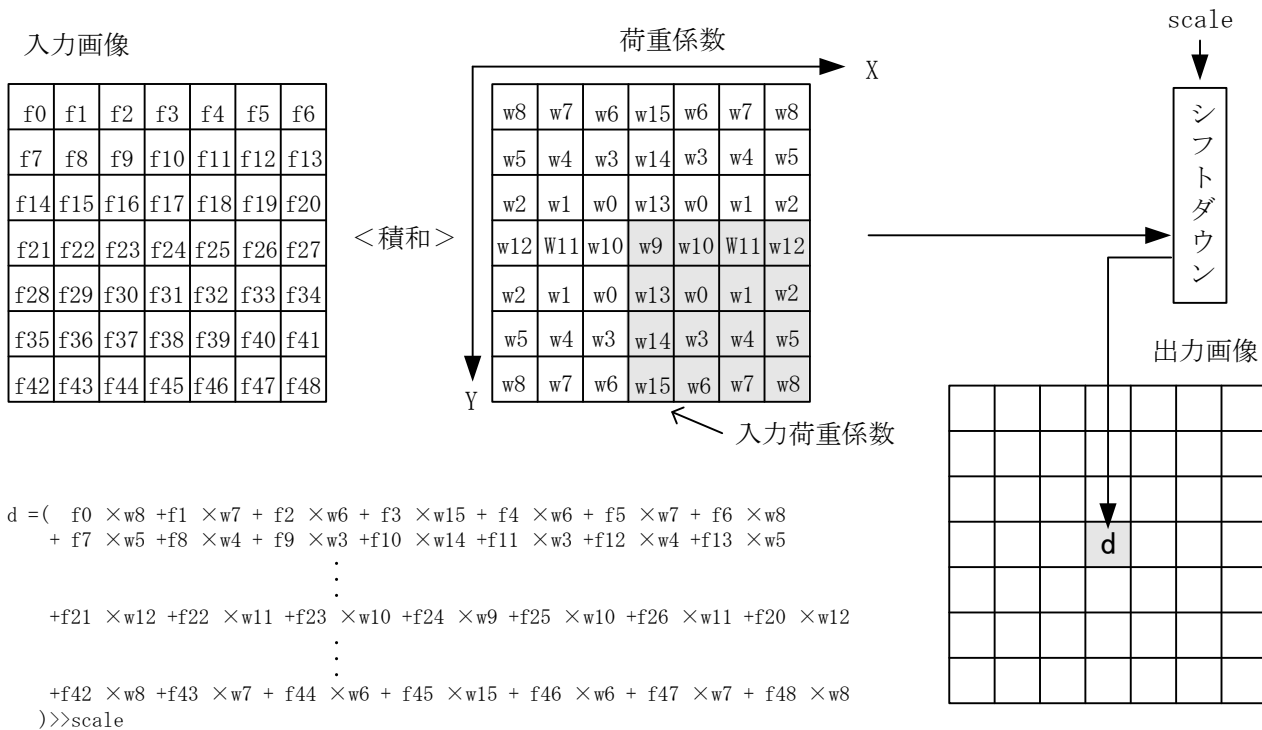
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

機能

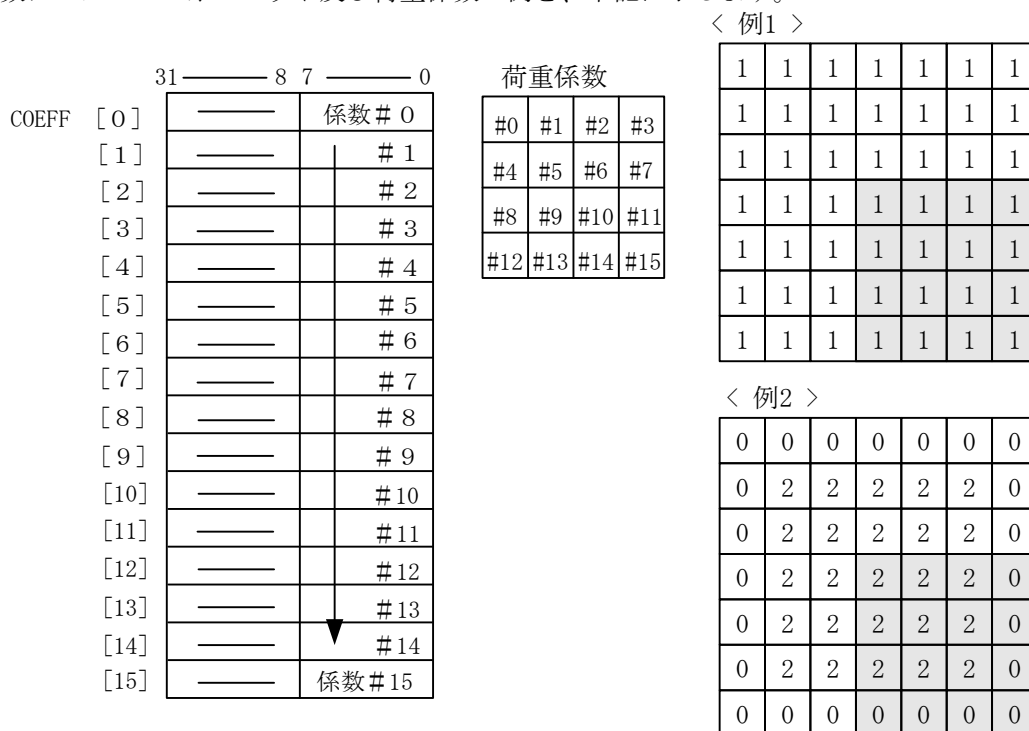
ソース画面（濃淡）に対し、 7×7 近傍で平滑化を行います。

平滑化では、入力画面の指定領域内の全画素に対して、注目画素を中心とした 7×7 近傍の局所領域で与えられた荷重係数との積和演算を行います。

また、対象領域の周辺（内側 3 画素の領域）処理の場合は中央画素値を出力します。



係数テーブルのフォーマット及び荷重係数の例を、下記に示します。



この例の場合、ダウンシフト量は6です。

2.28.3 濃淡画像輪郭強調(5×5近傍) (IP_EdgeFLT5x5)

□ C言語API

```
int ret = IP_EdgeFLT5x5( IMGID ImgSrc , IMGID ImgDst , int scale , int *COEFF );
```

□ パラメータ

IMGID ImgSrc ソース画面の画面番号

IMGID ImgDst デスティネーション画面の画面番号

int scale ダウンシフト量 (0 ～ 1 5)

int *COEFF 荷重係数テーブルのポインタ

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	—

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

2 OSでのエラー(タイムアウト以外)

3 IMP処理実行タイムアウト

1 0 , 1 3 画面空き領域、または範囲外 (不当画面番号エラー)

2 2 ダウンシフト量設定エラー

2 3 荷重係数設定エラー

4 8 動的メモリ確保エラー

 (ユーザープログラムによるメモリ破壊がないか確認して下さい)

3 0 SRC0ウインドウ設定エラー

3 3 DSTウインドウ設定エラー

4 9 アクセス画面が不定画面

— エラーリセットコマンド (ClearIPError) 未発行

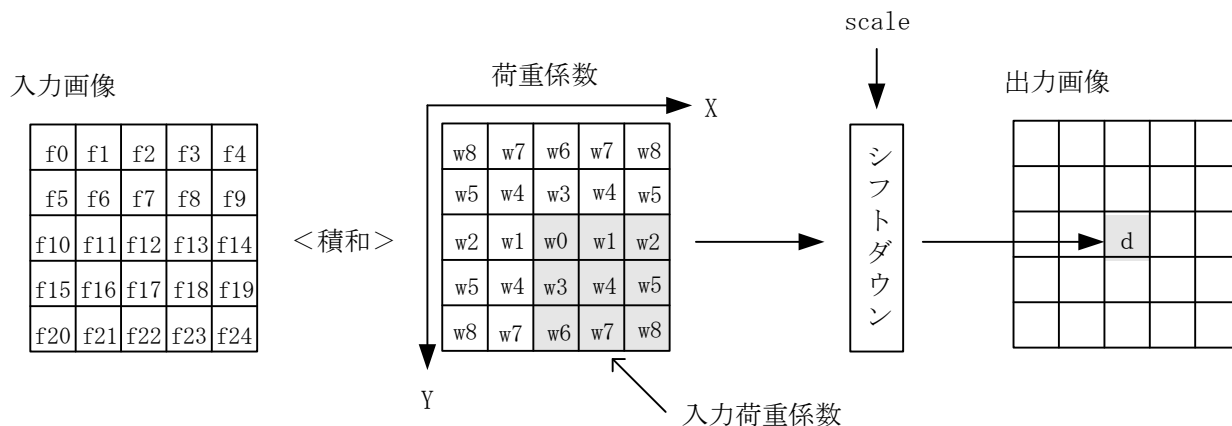
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面（濃淡）に対し、 5×5 近傍で輪郭強調を行います。

輪郭強調では、入力画面の指定領域内の全画素に対して、注目画素を中心とした 5×5 近傍の局所領域で与えられた荷重係数との積和演算を行います。

また、対象領域の周辺（内側 3 画素の領域）処理の場合は 0 を出力します。



$$\begin{aligned}
 d = & (f0 \times w8 + f1 \times w7 + f2 \times w6 + f3 \times w7 + f4 \times w8 \\
 & + f5 \times w5 + f6 \times w4 + f7 \times w3 + f8 \times w4 + f9 \times w5 \\
 & + f10 \times w2 + f11 \times w1 + f12 \times w0 + f13 \times w1 + f14 \times w2 \\
 & + f15 \times w5 + f16 \times w4 + f17 \times w3 + f18 \times w4 + f19 \times w5 \\
 & + f20 \times w8 + f21 \times w7 + f22 \times w6 + f23 \times w7 + f24 \times w8 \\
 &) \gg \text{scale}
 \end{aligned}$$

荷重係数の例を、以下に示します。また、係数テーブルのフォーマットは、IP_SmoothFLT5x5 の備考の欄を参照してください。

荷重係数の例

< 例1 > X方向微分

-1	1	0	1	-1
-1	1	0	1	-1
-1	1	0	1	-1
-1	1	0	1	-1
-1	1	0	1	-1

< 例2 > Y方向微分

-1	-1	-1	-1	-1
1	1	1	1	1
0	0	0	0	0
1	1	1	1	1
-1	-1	-1	-1	-1

この例の場合、ダウンシフト量は0です。

2.28.4 濃淡画像輪郭強調(7×7近傍) (IP_EdgeFLT7x7)

□ C言語API

```
int ret = IP_EdgeFLT7x7( IMGID ImgSrc , IMGID ImgDst , int scale , int *COEFF );
```

□ パラメータ

IMGID ImgSrc ソース画面の画面番号

IMGID ImgDst デスティネーション画面の画面番号

int scale ダウンシフト量 (0 ～ 15)

int *COEFF 荷重係数テーブルのポインタ

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

2 OSでのエラー(タイムアウト以外)

3 IMP処理実行タイムアウト

1 0 , 1 3 画面空き領域、または範囲外 (不当画面番号エラー)

2 2 ダウンシフト量設定エラー

2 3 荷重係数設定エラー

4 8 動的メモリ確保エラー

 (ユーザープログラムによるメモリ破壊がないか確認して下さい)

3 0 SRC0 ウィンドウ設定エラー

3 3 DST ウィンドウ設定エラー

4 9 アクセス画面が不定画面

- エラーリセットコマンド (ClearIPError) 未発行

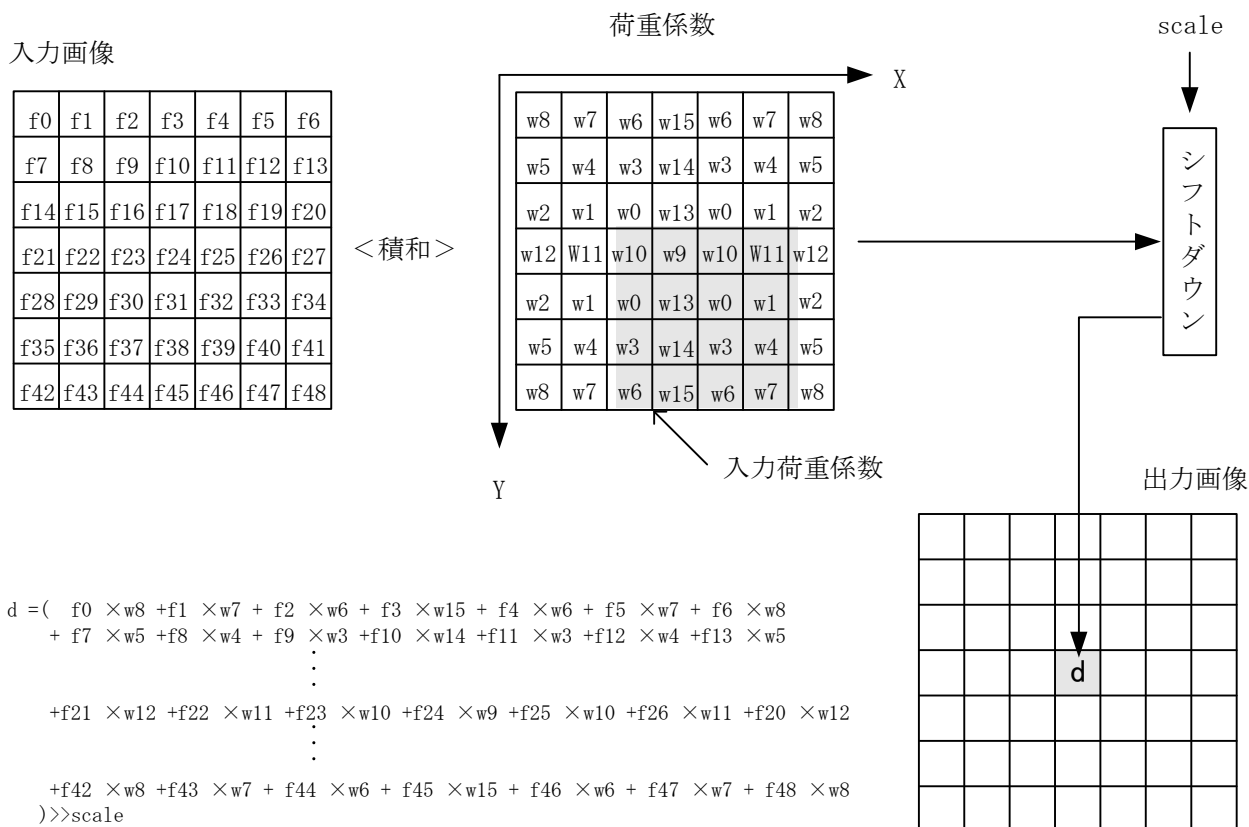
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面（濃淡）に対し、 7×7 近傍で輪郭強調を行います。

輪郭強調では、入力画面の指定領域内の全画素に対して、注目画素を中心とした 7×7 近傍の局所領域で与えられた荷重係数との積和演算を行います。

また、対象領域の周辺（内側 3 画素の領域）処理の場合は 0 を出力します。



荷重係数の例を、以下に示します。また、係数テーブルのフォーマットは、IP_SmoothFLT7x7の備考の欄を参照してください。

荷重係数の例

＜例1＞ X方向微分

0	-1	1	0	1	-1	0
0	-1	1	0	1	-1	0
0	-1	1	0	1	-1	0
0	-1	1	0	1	-1	0
0	-1	1	0	1	-1	0
0	-1	1	0	1	-1	0
0	-1	1	0	1	-1	0

＜例2＞ Y方向微分

0	0	0	0	0	0	0
-1	-1	-1	-1	-1	-1	-1
1	1	1	1	1	1	1
0	0	0	0	0	0	0
1	1	1	1	1	1	1
-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0

この例の場合、ダウンシフト量は0です。

2.28.5 局所最小値フィルタ(5x5近傍) (IP_MinFLT5x5)

□ C言語API

```
int ret = IP_MinFLT5x5( IMGID ImgSrc , IMGID ImgDst , int calptn );
```

□ パラメータ

IMGID ImgSrc ソース画面の画面番号

IMGID ImgDst デスティネーション画面の画面番号

int calptn 演算パターン（下位9ビット有効）

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2 OSでのエラー(タイムアウト以外)
 3 IMP処理実行タイムアウト
 10, 13 画面空き領域、または範囲外（不当画面番号エラー）
 48 動的メモリ確保エラー
 (ユーザープログラムによるメモリ破壊がないか確認して下さい)
 30 SRC0ウインドウ設定エラー
 33 DSTウインドウ設定エラー
 49 アクセス画面が不定画面
 - エラーリセットコマンド (ClearIPError) 未発行

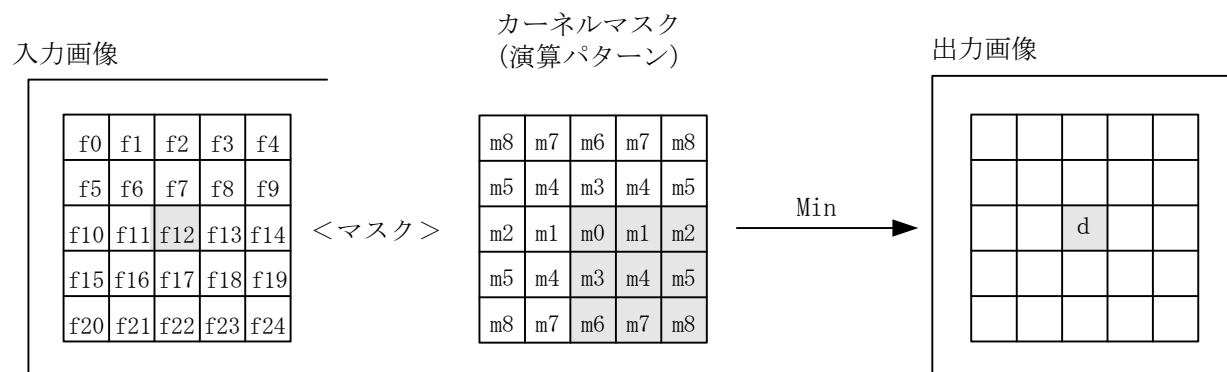
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

機能

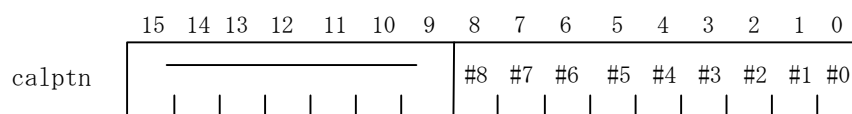
ソース画面に対し、 5×5 近傍をパラメータで指定されたパターンにて最小濃度値を見つけ、デスティネーション画面に出力します。

局所最小値フィルタ（ 5×5 近傍）では、入力画面の指定領域内の全画素に対して、注目画素を中心とした 5×5 近傍の局所領域で与えられた演算パターンでマスク処理し、有効となった画素の中から最小の画素を出力します。

また、対象領域の周辺（内側3画素の領域）処理の場合は不定値を出力します。



パラメータcalptn のフォーマットを下記に示します。



0 : 演算マスク

1 : 演算イネーブル

#0	#1	#2
#3	#4	#5
#6	#7	#8

← 演算パターン

2.28.6 局所最大値フィルタ(5x5近傍) (IP_MaxFLT5x5)

□ C言語API

```
int ret = IP_MaxFLT5x5( IMGID ImgSrc , IMGID ImgDst , int calptn );
```

□ パラメータ

IMGID ImgSrc ソース画面の画面番号

IMGID ImgDst デスティネーション画面の画面番号

int calptn 演算パターン（下位9ビット有効）

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2 OSでのエラー(タイムアウト以外)
 3 IMP処理実行タイムアウト
 10, 13 画面空き領域、または範囲外（不当画面番号エラー）
 48 動的メモリ確保エラー
 (ユーザープログラムによるメモリ破壊がないか確認して下さい)
 30 SRC0ウインドウ設定エラー
 33 DSTウインドウ設定エラー
 49 アクセス画面が不定画面
 - エラーリセットコマンド (ClearIPError) 未発行

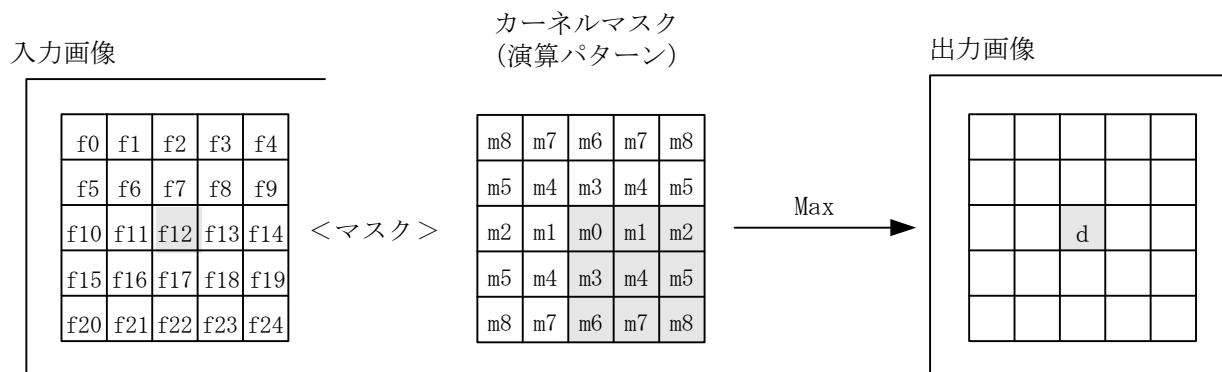
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

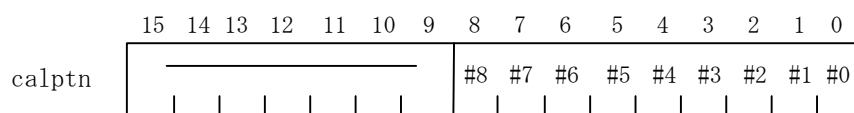
ソース画面に対し、 5×5 近傍で、パラメータで指定されたパターンで最大濃度値を見つけ、デスティネーション画面に出力します。

局所最大値フィルタ（ 5×5 近傍）では、入力画面の指定領域内の全画素に対して、注目画素を中心とした 5×5 近傍の局所領域で与えられた演算パターンでマスク処理し、有効となった画素の中から最大の画素を出力します。

また、対象領域の周辺（内側3画素の領域）処理の場合は不定値を出力します。

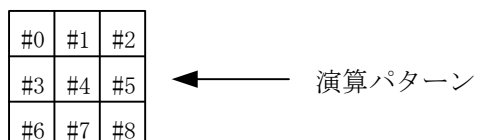


パラメータcalptn のフォーマットを下記に示します。



0 : 演算マスク

1 : 演算イネーブル



2.28.7 平滑化フィルタ(拡張版) (IP_SmoothFLTExt)

□ C言語API

```
int ret = IP_SmoothFLTExt( IMGID ImgSrc , IMGID ImgDst , enum IPMSmoothMethod method , int n );
```

□ パラメータ

IMGID ImgSrc ソース画面の画面番号

IMGID ImgDst デスティネーション画面の画面番号

enum IPMSmoothMethod method 平滑化手法

平滑化手法	内容
SMOOTHNOOP	コピー
SMOOTH1	平滑化フィルタ 1 : 詳細情報参照
SMOOTH2	平滑化フィルタ 2 : 詳細情報参照
SMOOTH3	平滑化フィルタ 3 : 詳細情報参照
SMOOTH4	平滑化フィルタ 4 : 詳細情報参照
MEDIAN8	メディアンフィルタ (8 連結)
MEDIAN4	メディアンフィルタ (4 連結)
MEDIAN84	メディアンフィルタ (8 → 4 → 8 ・ ・ 連結)
MEDIAN48	メディアンフィルタ (4 → 8 → 4 ・ ・ 連結)
SENSUP2	2 倍感度 U P フィルタ : 詳細情報参照
SENSUP4	4 倍感度 U P フィルタ : 詳細情報参照
MAX8	局所最大フィルタ (8 連結)
MIN8	局所最小フィルタ (8 連結)

int n 繰り返し回数 (1 ~ 16)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

2	OSでのエラー(タイムアウト以外)
3	IMP処理実行タイムアウト
10, 13	画面空き領域、または範囲外（不当画面番号エラー）
22	繰り返し回数設定エラー
23	平滑化方式設定エラー
48	動的メモリ確保エラー (ユーザープログラムによるメモリ破壊がないか確認して下さい)
30	SRC0ウィンドウ設定エラー
33	DSTウィンドウ設定エラー
49	アクセス画面が不定画面
-	エラーリセットコマンド (ClearIPErrors) 未発行

※ 詳細は「コマンドエラー発生時の対策と手順」の章を参照して下さい。

□ 機能

濃淡画像をパラメータ指定の平滑化手法で平滑化します。なお、同一映像に対して最大16回まで連続して実行処理できます。

スムージングフィルタ・感度UPフィルタの荷重係数

平滑化フィルタ 1

7	7	7
7	8	7
7	7	7

6ビットシフトダウン

平滑化フィルタ 2

6	7	6
7	12	7
6	7	6

6ビットシフトダウン

平滑化フィルタ 3

2	4	2
4	8	4
2	4	2

5ビットシフトダウン

平滑化フィルタ 4

2	3	2
3	12	3
2	3	2

5ビットシフトダウン

2倍感度UP フィルタ

1	2	1
2	4	2
1	2	1

3ビットシフトダウン

4倍感度UP フィルタ

1	2	1
2	4	2
1	2	1

2ビットシフトダウン

デスティネーション画面に画像処理の途中結果を格納しているため、デスティネーション画面を表示画面にしていると、映像がちらつく場合があります。

2.28.8 輪郭強調フィルタ・絶対値(拡張版) (IP EdgeFLTAbsExt)

□ C言語API

```
int ret = IP_EdgeFLTAbsExt( IMGID ImgSrc , IMGID  ImgDst , enum IP_EdgeMethod method ,
                           float gain ,int lutno);
```

□ パラメータ

IMGID	ImgSrc	ソース画面の画面番号

IMGID	ImgDst	デスティネーション画面の画面番号

enum IPMEdgeMethod method 輪郭強調手法

輪郭強調手法	内容
EDGENOOP	コピー
EDGEMAXVH	$\max(\text{水平エッジ} , \text{垂直エッジ})$: 詳細情報 (A), (B) 参照
EDGEADDVH	$ \text{水平エッジ} + \text{垂直エッジ} $: 詳細情報 (A), (B) 参照
EDGEV	$ \text{垂直エッジ} $: 詳細情報 (A), (B) 参照
EDGEH	$ \text{水平エッジ} $: 詳細情報 (A), (B) 参照
EDGE_SOBEL_DASH	ソーベル' : 詳細情報 (A), (B) 参照
EDGE_ZEROCROSS8_ALL	ゼロクロッシング 1 : 詳細情報 (C) 参照
EDGE_ZEROCROSS4_ALL	ゼロクロッシング 2 : 詳細情報 (C) 参照
EDGE_ZEROCROSS8_SEQ	ゼロクロッシング 3 : 詳細情報 (D) 参照
EDGE_ZEROCROSS4_SEQ	ゼロクロッシング 4 : 詳細情報 (D) 参照
EDGE_MINMAX	Min, Max フィルタによるエッジ抽出 : 詳細情報 (B), (E) 参照
EDGE_SOBEL	$ \text{水平エッジ} + \text{垂直エッジ} $: 詳細情報 (A), (B) 参照

float	gain	エッジ出力時	— 輪郭強調ゲイン (0.5 ~ 0.7)
		ゼロクロッシング時	— 乗算のダウンシフト量 (0 ~ 8)

int	lutno	拡張濃度変換テーブル登録番号

0 : N00P
1 ~ 8 : 濃度適応補正処理

IP_RegisterLUT(int lutno, CNVLUT *lut)コマンドで設定された拡張濃度変換テーブル登録番号 (lutno) で、濃度変換を実行します。但し、以下の条件時には、濃度適応は行いません。

- ・拡張濃度変換テーブル登録番号が0の場合
- ・輪郭強調手法が「EDGENOOP」の場合

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	－

□ リターンパラメータ

int ret 正常終了（0）または異常終了（－1）

□ エラーコード

2 OSでのエラー(タイムアウト以外)
3 IMP処理実行タイムアウト
1 0, 1 3 画面空き領域、または範囲外（不当画面番号エラー）
2 2 平滑化手法設定範囲外
2 3 繰り返し回数設定範囲外
4 8 動的メモリ確保エラー
 (ユーザープログラムによるメモリ破壊がないか確認して下さい)
3 0 SRC0ウィンドウ設定エラー
3 3 DSTウィンドウ設定エラー
4 9 アクセス画面が不定画面
－ エラーリセットコマンド (ClearIPError) 未発行

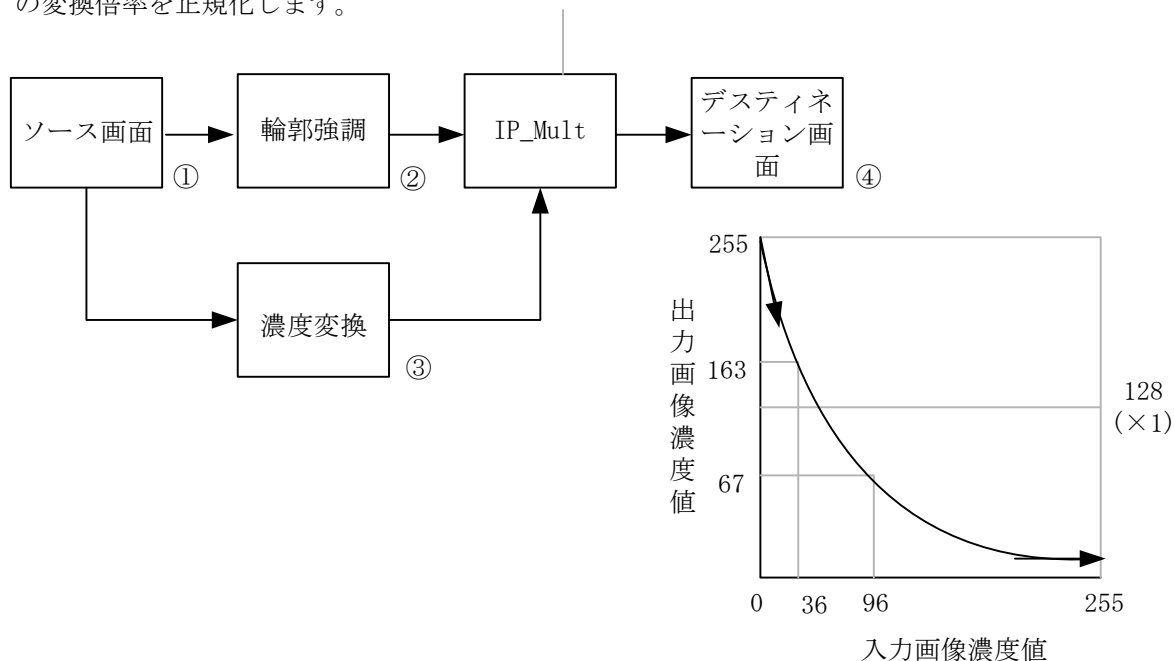
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

機能

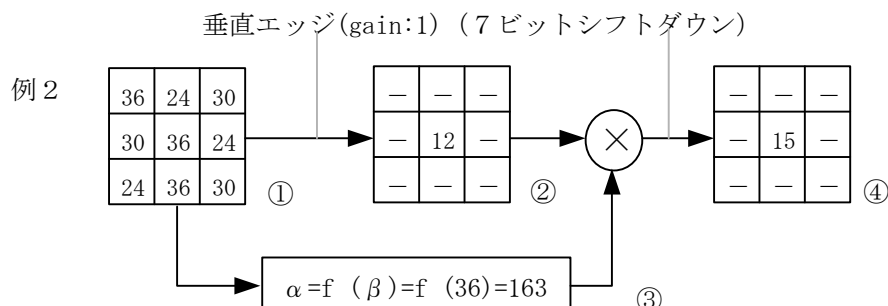
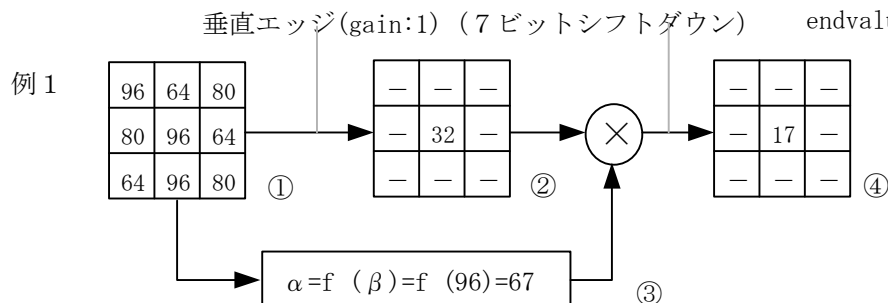
濃淡画像をパラメータ指定の輪郭強調手法で、濃淡画像の輪郭強調を行います。

[A] 濃度適応処理は、下図左の処理を行います。下図右の濃度変換テーブルを登録・指定して濃度適応処理を実行させた場合、暗い部分のエッジは明るくなり・明るい部分のエッジは暗くなります。

乗算後、7ビットシフトダウン（ $\div 128$ ）し、濃度補正の変換倍率を正規化します。



startvalue:255 startgrad:-3.0
endvalue :16 endgrad :-0.025



左図の例 1、2 の様に濃度補正を行う事で、同一パターンで出力される値が均一化されます。

濃度変換は、上図のテーブルを指定します。

①ソース画面のサンプル

②垂直エッジの濃度値

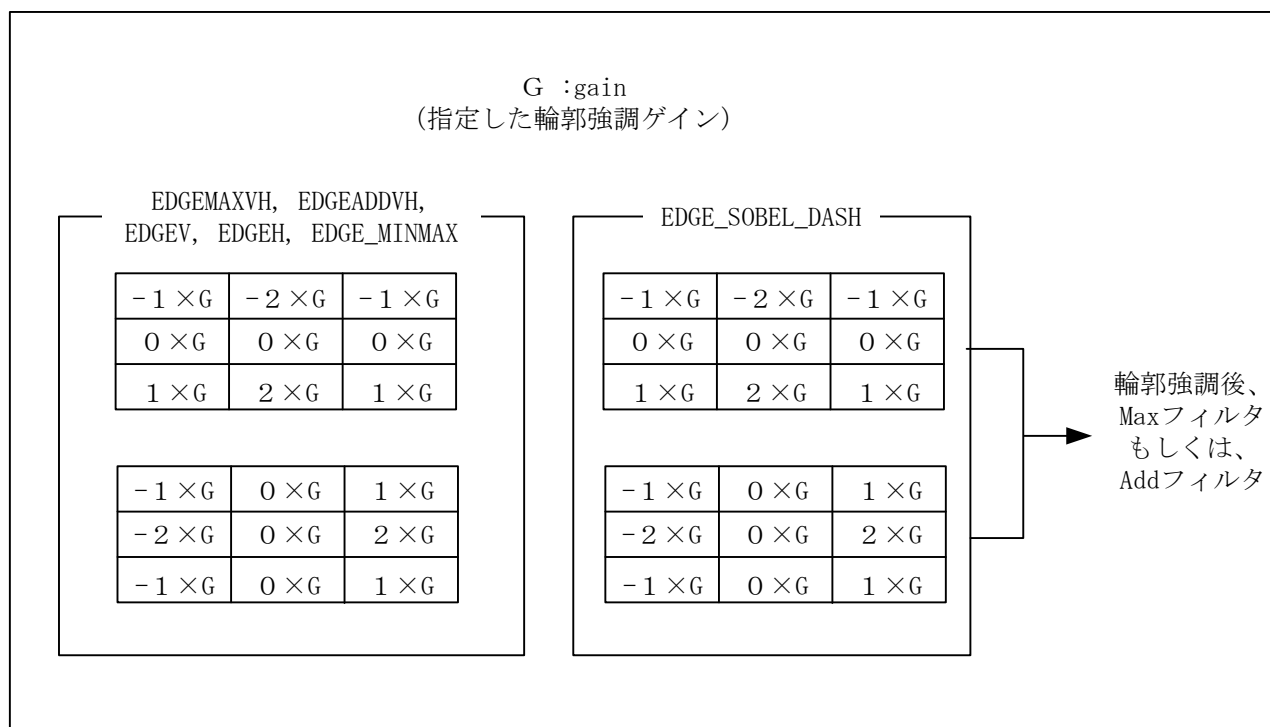
③濃度変換 $\alpha = f(\beta)$ (右上図)

④濃度適応処理結果 (四捨五入されます)

[B] 係数テーブル

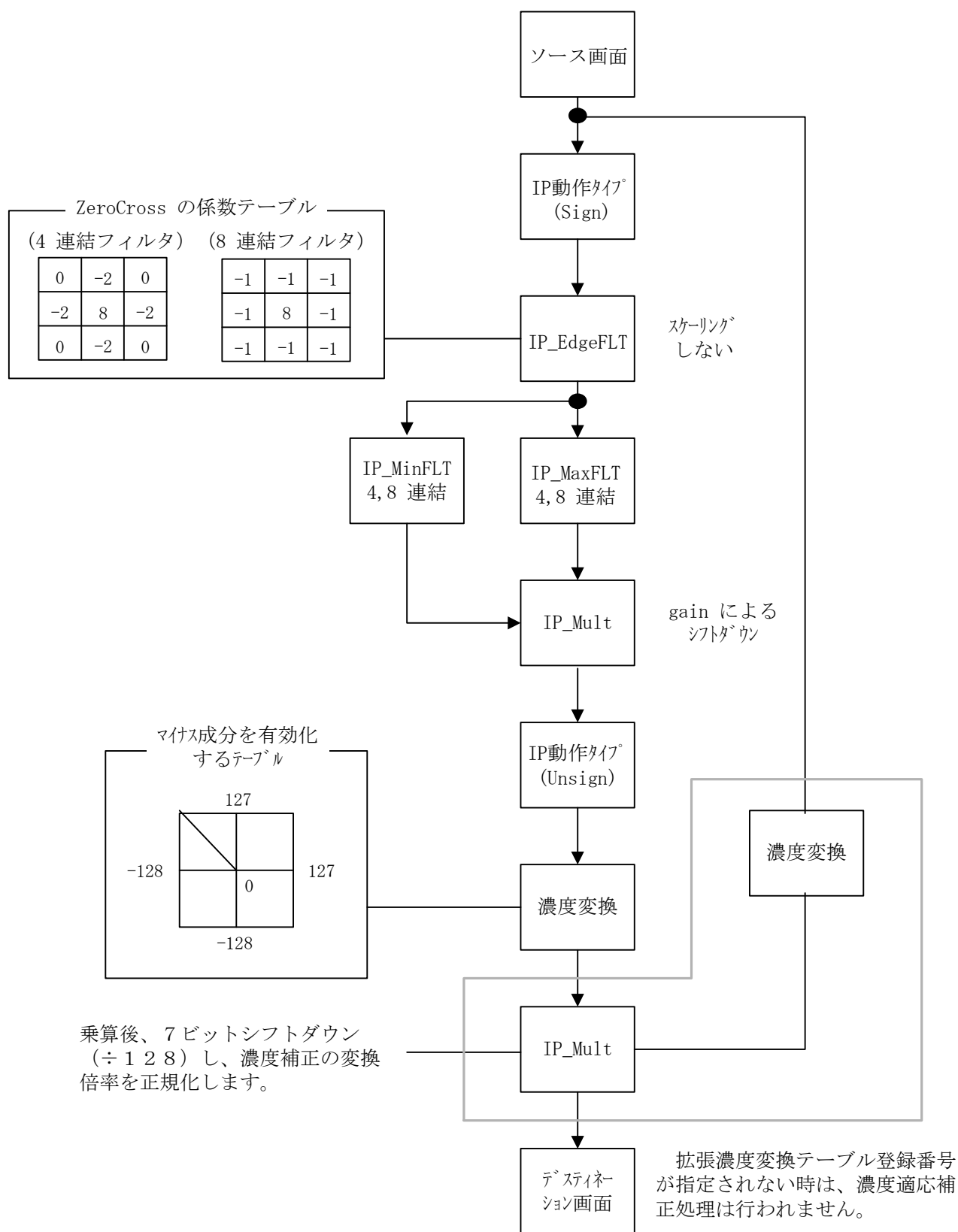
輪郭強調フィルタで、EDGE_MAXVH, EDGE_ADDVH, Sobel, Sobel' フィルタの係数テーブルは次の通りです。

係数テーブルの構造



[C] ZeroCross8, 4ALL

下記の係数テーブルで取得した微分符号付き画像に対し、局所最小・最大値フィルタを乗算した画像で、負の値（0 をクロスする）を抽出します。乗算結果の値が小さいほど、エッジ濃度は大きくなります。

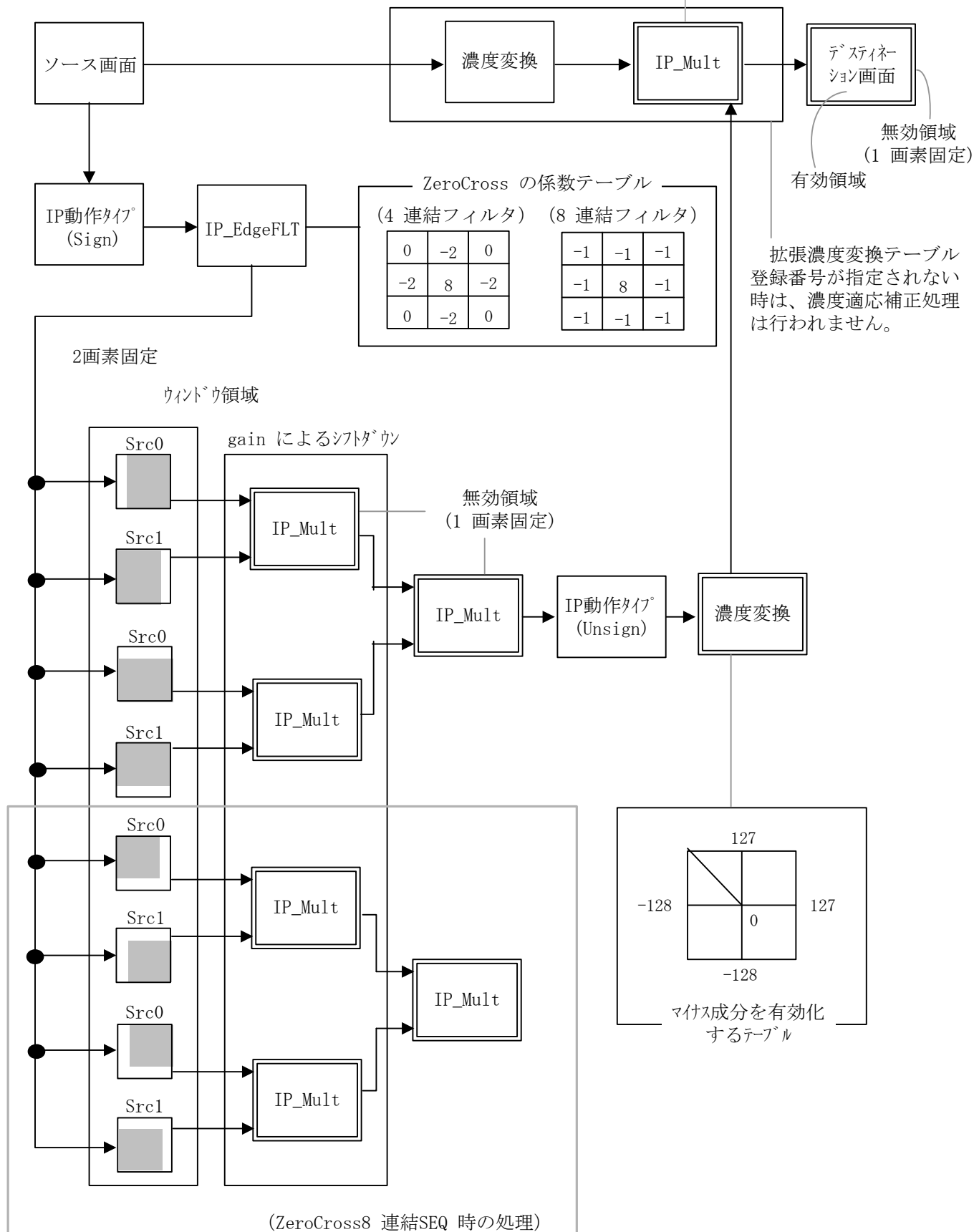


[D] ZeroCross8, 4SEQ

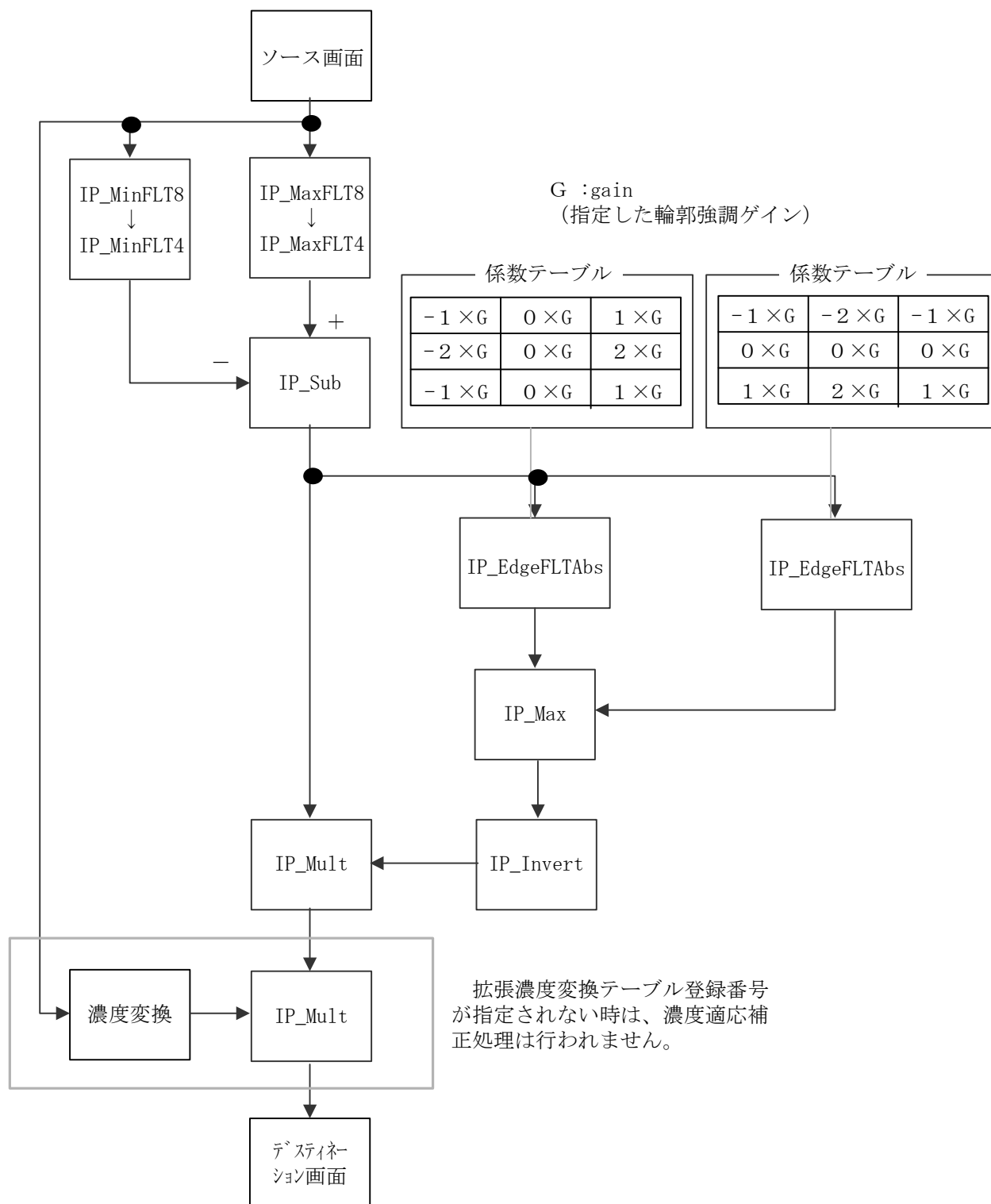
下記の係数テーブルで取得した微分符号付き画像に対し、座標をずらし乗算した画像で、負の値（0をクロスする）を抽出します。乗算結果の値が小さいほど、エッジ濃度は大きくなります。

※ウィンドウサイズが、縦、もしくは横が1ラインの時は、NOOP に切り換えます。

乗算後、7ビットシフトダウン（ $\div 128$ ）し、濃度補正の変換倍率を正規化します。



[E] Min, Max フィルタによるエッジ抽出



2.28.9 拡張平滑化(5×5) (IP_SmoothFLT5x5Ext)

□ C言語API

```
int ret = IP_SmoothFLT5x5Ext( IMGID ImgSrc , IMGID ImgDst , int scale , enum IPConvExtOpt opt ,
                               int *COEFF );
```

□ パラメータ

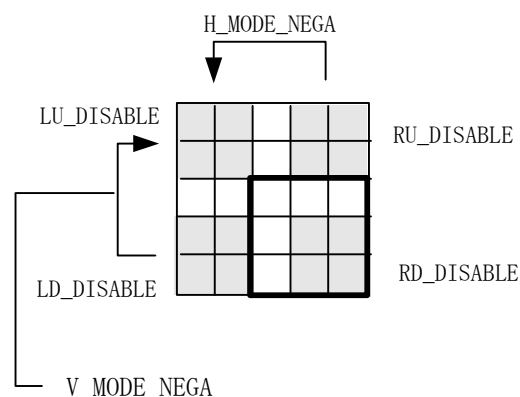
IMGID ImgSrc ソース画面の画面番号

IMGID ImgDst デスティネーション画面の画面番号

int scale ダウンシフト量 (0 ～ 15)

enum IPConvExtOpt opt 拡張コンボリユーションオプション

オプション	対応定数	内容
H_MODE_NEGA	3 2	負対称 (水平方向)
V_MODE_NEGA	6 4	負対称 (垂直方向)
RD_DISABLE	1	右下演算処理無効
LD_DISABLE	2	左下演算処理無効
RU_DISABLE	4	右上演算処理無効
LU_DISABLE	8	左上演算処理無効



int *COEFF 荷重係数テーブルのポインタ

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

2 OSでのエラー(タイムアウト以外)
3 IMP処理実行タイムアウト
1 0, 1 3 画面空き領域、または範囲外 (不当画面番号エラー)
2 2 ダウンシフト量設定エラー
2 3 荷重係数設定エラー
4 8 動的メモリ確保エラー
(ユーザープログラムによるメモリ破壊がないか確認して下さい)
3 0 SRC0 ウィンドウ設定エラー
3 3 DST ウィンドウ設定エラー
4 9 アクセス画面が不定画面
- エラーリセットコマンド (ClearIPError) 未発行

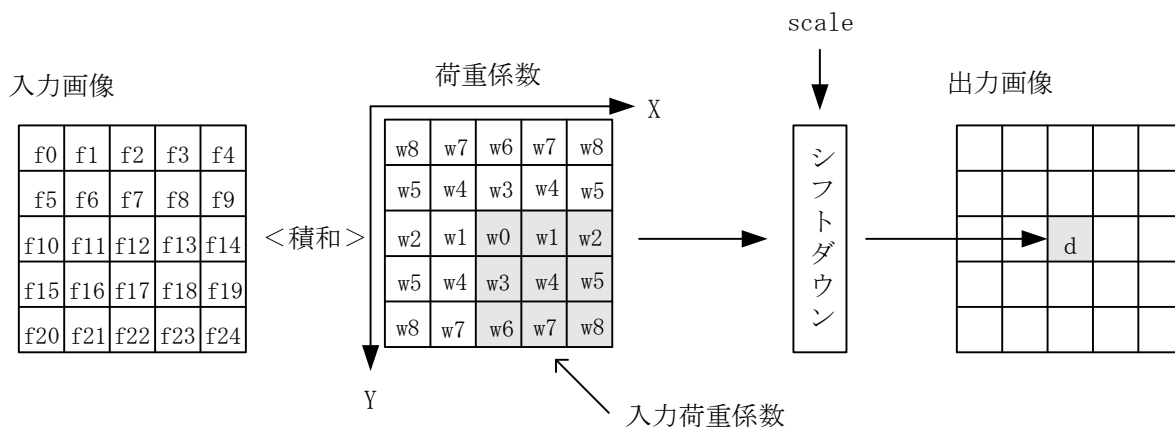
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

機能

ソース画面（濃淡）に対し、 5×5 近傍で平滑化を行います。

平滑化では、入力画面の指定領域内の全画素に対して、注目画素を中心とした 5×5 近傍の局所領域で与えられた荷重係数との積和演算を行います。

また、対象領域の周辺（内側3画素の領域）処理の場合は中央画素値を出力します。

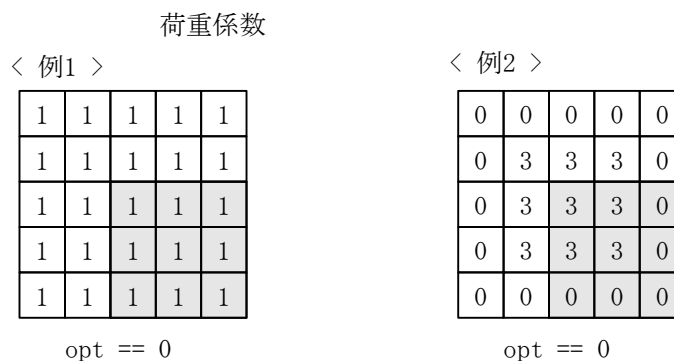


```

d = ( f0 × w8 + f1 × w7 + f2 × w6 + f3 × w7 + f4 × w8
    + f5 × w5 + f6 × w4 + f7 × w3 + f8 × w4 + f9 × w5
    + f10 × w2 + f11 × w1 + f12 × w0 + f13 × w1 + f14 × w2
    + f15 × w5 + f16 × w4 + f17 × w3 + f18 × w4 + f19 × w5
    + f20 × w8 + f21 × w7 + f22 × w6 + f23 × w7 + f24 × w8
    ) >> scale

```

荷重係数の例を、以下に示します。また、係数テーブルのフォーマットは、`IP_SmoothFLT5x5()` の備考の欄を参照してください。



この例の場合、ダウンシフト量は5です。

[拡張コンボリューション設定例]

全カーネル有効で負対称（垂直方向）
`opt = ( enum IPConvExtOpt )( V_MODE_NEGA )`

右上と左下カーネル無効で負対称（水平方向）
`opt = ( enum IPConvExtOpt )( LD_DISABLE | RU_DISABLE | H_MODE_NEGA );`

（注）Opt=0 のとき、Extなしの拡張コンボリューションに一致

2.28.10 拡張平滑化(7×7) (IP_SmoothFLT7x7Ext)

□ C言語API

```
int ret = IP_SmoothFLT7x7Ext( IMGID ImgSrc , IMGID ImgDst , int scale , enum IPConvExtOpt opt ,
                               int *COEFF );
```

□ パラメータ

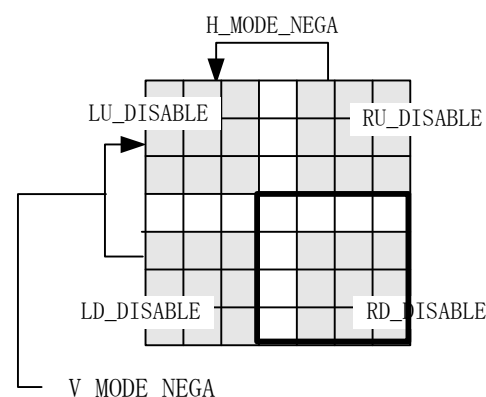
IMGID ImgSrc ソース画面の画面番号

IMGID ImgDst デスティネーション画面の画面番号

int scale ダウンシフト量 (0 ～ 1 5)

enum IPConvExtOpt opt 拡張コンボリキュレーションオプション

オプション	対応定数	内容
H_MODE_NEGA	3 2	負対称 (水平方向)
V_MODE_NEGA	6 4	負対称 (垂直方向)
RD_DISABLE	1	右下演算処理無効
LD_DISABLE	2	左下演算処理無効
RU_DISABLE	4	右上演算処理無効
LU_DISABLE	8	左上演算処理無効



int *COEFF 荷重係数テーブルのポインタ

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

2 OSでのエラー(タイムアウト以外)
3 IMP処理実行タイムアウト
1 0, 1 3 画面空き領域、または範囲外 (不当画面番号エラー)
2 2 ダウンシフト量設定エラー
2 3 荷重係数設定エラー
4 8 動的メモリ確保エラー
(ユーザープログラムによるメモリ破壊がないか確認して下さい)
3 0 SRC 0 ウィンドウ設定エラー
3 3 DST ウィンドウ設定エラー
4 9 アクセス画面が不定画面
- エラーリセットコマンド (ClearIPError) 未発行

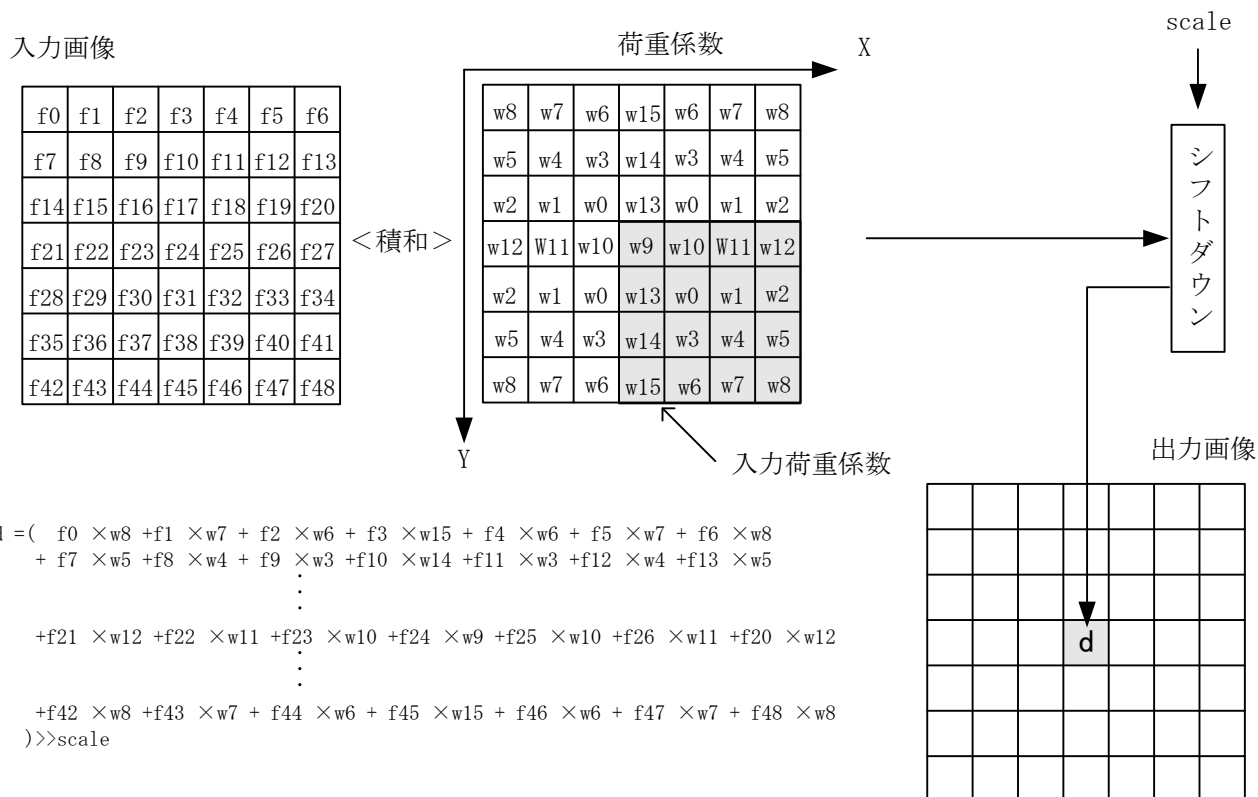
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

機能

ソース画面（濃淡）に対し、 7×7 近傍で平滑化を行います。

平滑化では、入力画面の指定領域内の全画素に対して、注目画素を中心とした 7×7 近傍の局所領域で与えられた荷重係数との積和演算を行います。

また、対象領域の周辺（内側3画素の領域）処理の場合は中央画素値を出力します。



荷重係数の例を、以下に示します。また、係数テーブルのフォーマットは、`IP_SmoothFLT7x7()` の備考の欄を参照してください。

＜例1＞ 荷重係数

1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

opt == 0

この例の場合、ダウンシフト量は6です。

[拡張コンボリユーション設定例]

全カーネル有効で負対称（垂直方向）

```
opt = ( enum IPConvExtOpt )( V_MODE_NEGA )
```

右上と左下カーネル無効で負対称（水平方向）

```
opt = ( enum IPConvExtOpt )( LD_DISABLE | RU_DISABLE | H_MODE_NEGA );
```

（注）Opt=0 のとき、Extなしの拡張コンボリユーションに一致

2.28.11 拡張濃淡画像輪郭強調(5x5近傍) (IP_EdgeFLT5x5Ext)

□ C言語API

```
int ret = IP_EdgeFLT5x5Ext( IMGID ImgSrc , IMGID ImgDst , int scale , enum IPConvExtOpt opt ,
                           int *COEFF );
```

□ パラメータ

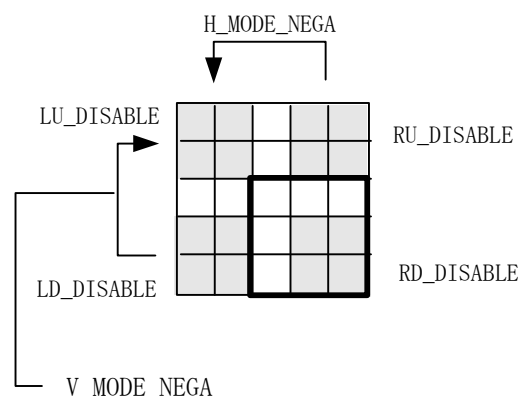
IMGID ImgSrc ソース画面の画面番号

IMGID ImgDst デスティネーション画面の画面番号

int scale ダウンシフト量 (0 ～ 1 5)

enum IPConvExtOpt opt 拡張コンボリユーションオプション

オプション	対応定数	内容
H_MODE_NEGA	3 2	負対称 (水平方向)
V_MODE_NEGA	6 4	負対称 (垂直方向)
RD_DISABLE	1	右下演算処理無効
LD_DISABLE	2	左下演算処理無効
RU_DISABLE	4	右上演算処理無効
LU_DISABLE	8	左上演算処理無効



int *COEFF 荷重係数テーブルのポインタ

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

2 OSでのエラー(タイムアウト以外)
3 IMP処理実行タイムアウト
1 0, 1 3 画面空き領域、または範囲外 (不当画面番号エラー)
2 2 ダウンシフト量設定エラー
2 3 荷重係数設定エラー
4 8 動的メモリ確保エラー
(ユーザープログラムによるメモリ破壊がないか確認して下さい)
3 0 SRC0 ウィンドウ設定エラー
3 3 DST ウィンドウ設定エラー
4 9 アクセス画面が不定画面
- エラーリセットコマンド (ClearIPErrors) 未発行

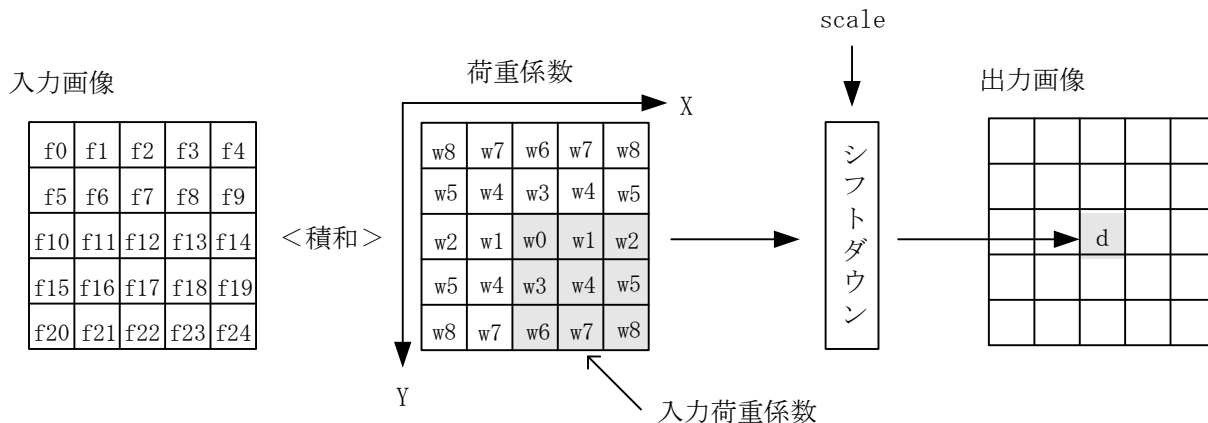
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

機能

ソース画面（濃淡）に対し、5×5近傍で輪郭強調を行います。

輪郭強調では、入力画面の指定領域内の全画素に対して、注目画素を中心とした5×5近傍の局所領域で与えられた荷重係数との積和演算を行います。

また、対象領域の周辺（内側3画素の領域）処理の場合は0を出力します。



$$\begin{aligned}
 d = & (f0 \times w8 + f1 \times w7 + f2 \times w6 + f3 \times w7 + f4 \times w8 \\
 & + f5 \times w5 + f6 \times w4 + f7 \times w3 + f8 \times w4 + f9 \times w5 \\
 & + f10 \times w2 + f11 \times w1 + f12 \times w0 + f13 \times w1 + f14 \times w2 \\
 & + f15 \times w5 + f16 \times w4 + f17 \times w3 + f18 \times w4 + f19 \times w5 \\
 & + f20 \times w8 + f21 \times w7 + f22 \times w6 + f23 \times w7 + f24 \times w8 \\
 &) \gg \text{scale}
 \end{aligned}$$

荷重係数の例を、以下に示します。また、係数テーブルのフォーマットは、IP_EdgeFLT5x5()の備考の欄を参照してください。

荷重係数の例

＜例1＞ X方向微分

0	-1	0	1	0
0	-1	0	1	0
0	-1	0	1	0
0	-1	0	1	0
0	-1	0	1	0

opt == H_MODE_NEGA

＜例2＞ Y方向微分

0	0	0	0	0
-1	-1	-1	-1	-1
0	0	0	0	0
1	1	1	1	1
0	0	0	0	0

opt == V_MODE_NEGA

この例の場合、ダウンシフト量は0です。

＜例3＞ X方向微分

2	1	0	-1	-2
2	1	0	-1	-2
2	1	0	-1	-2
2	1	0	-1	-2
2	1	0	-1	-2

opt == H_MODE_NEGA

この例の場合、ダウンシフト量は0です。

2.28.12 拡張濃淡画像輪郭強調(7x7近傍) (IP_EdgeFLT7x7Ext)

□ C言語API

```
int ret = IP_EdgeFLT7x7Ext( IMGID ImgSrc , IMGID ImgDst , int scale , enum IPConvExtOpt opt ,
                           int *COEFF );
```

□ パラメータ

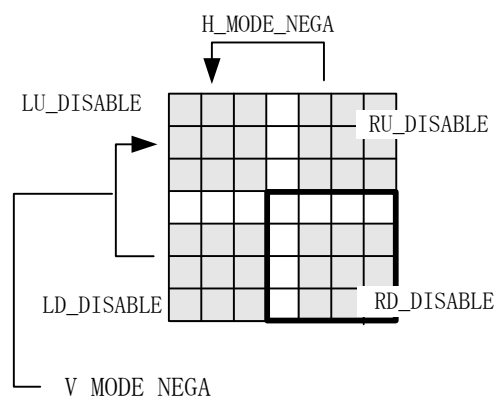
IMGID ImgSrc ソース画面の画面番号

IMGID ImgDst デスティネーション画面の画面番号

int scale ダウンシフト量 (0 ～ 1 5)

enum IPConvExtOpt opt 拡張コンボリユーションオプション

オプション	対応定数	内容
H_MODE_NEGA	3 2	負対称 (水平方向)
V_MODE_NEGA	6 4	負対称 (垂直方向)
RD_DISABLE	1	右下演算処理無効
LD_DISABLE	2	左下演算処理無効
RU_DISABLE	4	右上演算処理無効
LU_DISABLE	8	左上演算処理無効



int *COEFF 荷重係数テーブルのポインタ

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

2 OSでのエラー(タイムアウト以外)
3 IMP処理実行タイムアウト
1 0, 1 3 画面空き領域、または範囲外 (不当画面番号エラー)
2 2 ダウンシフト量設定エラー
2 3 荷重係数設定エラー
4 8 動的メモリ確保エラー
(ユーザープログラムによるメモリ破壊がないか確認して下さい)
3 0 SRC0ウインドウ設定エラー
3 3 DSTウインドウ設定エラー
4 9 アクセス画面が不定画面
- エラーリセットコマンド (ClearIPError) 未発行

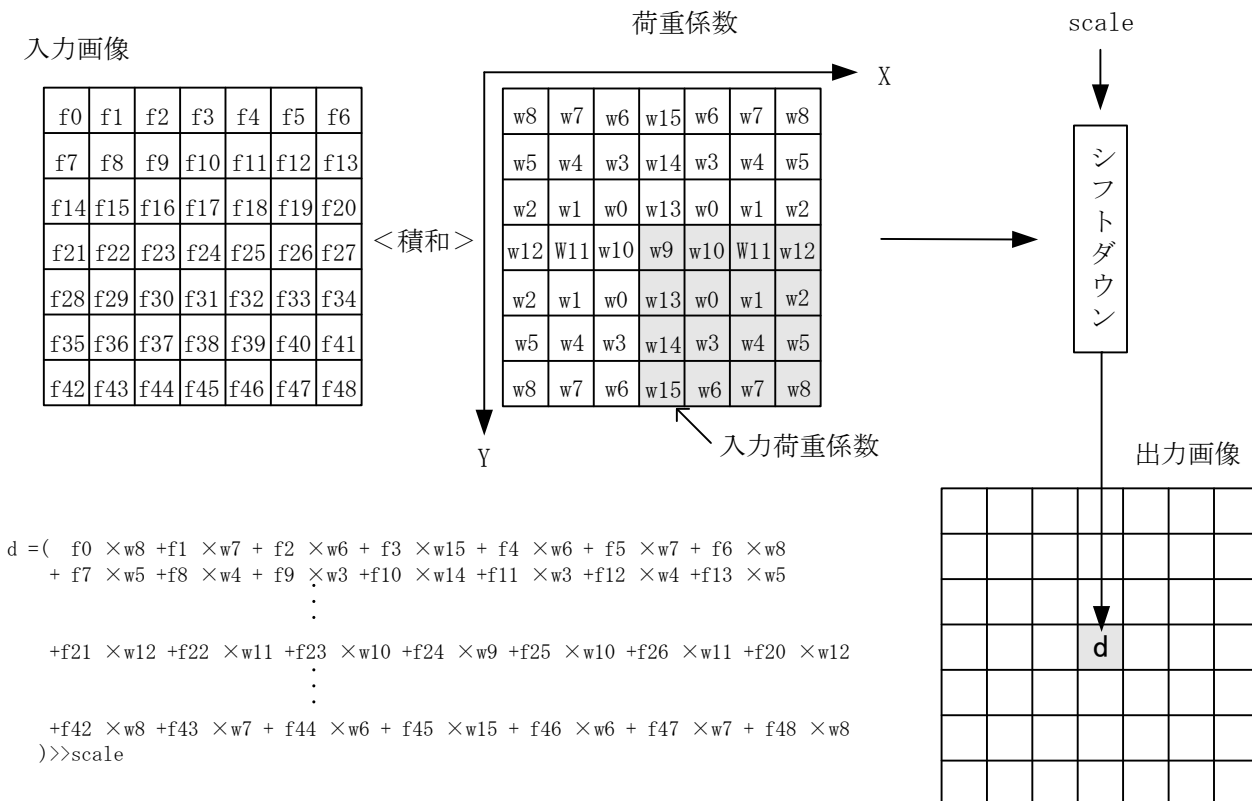
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

機能

ソース画面（濃淡）に対し、 7×7 近傍で輪郭強調を行います。

輪郭強調では、入力画面の指定領域内の全画素に対して、注目画素を中心とした 7×7 近傍の局所領域で与えられた荷重係数との積和演算を行います。

また、対象領域の周辺（内側3画素の領域）処理の場合は0を出力します。



荷重係数の例を、以下に示します。また、係数テーブルのフォーマットは、IP_EdgeFLT7x7() の備考の欄を参照してください。

＜例1＞

-3	-2	-1	0	1	2	3
-3	-2	-1	0	1	2	3
-3	-2	-1	0	1	2	3
-3	-2	-1	0	1	2	3
-3	-2	-1	0	1	2	3
-3	-2	-1	0	1	2	3
-3	-2	-1	0	1	2	3

opt == H_MODE_NEGA

荷重係数 ＜例2＞

-3	-3	-3	-3	-3	-3	-3
-2	-2	-2	-2	-2	-2	-2
-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0
1	1	1	1	1	1	1
2	2	2	2	2	2	2
3	3	3	3	3	3	3

opt == V_MODE_NEGA

この例の場合、ダウンシフト量は0です。

[拡張コンボリューション設定例]

全カーネル有効で負対称（垂直方向）

```
opt = ( enum IPConvExtOpt ) ( V_MODE_NEGA )
```

右上と左下カーネル無効で負対称（水平方向）

```
opt = ( enum IPConvExtOpt ) ( LD_DISABLE | RU_DISABLE | H_MODE_NEGA );
```

（注）Opt=0 のとき、Extなしの拡張コンボリューションに一致

2.29 拡張画像処理

2.29.1 輪郭強調(ソーベル) (IP_Sobel)

□ C言語API

```
int ret = IP_Sobel( IMGID ImgSrc , IMGID ImgDst , int gain );
```

□ パラメータ

IMGID ImgSrc ソース画面の画面番号

IMGID ImgDst デスティネーション画面の画面番号

int gain ゲイン (1 ~ 8)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

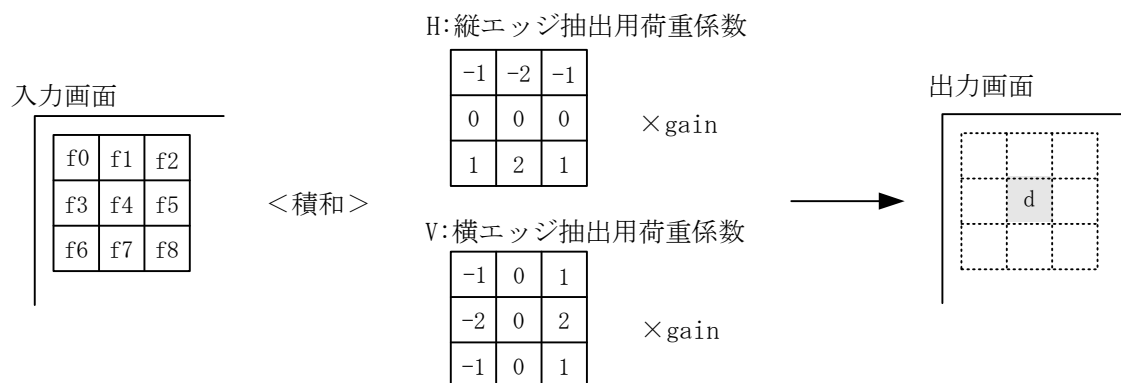
2 OSでのエラー(タイムアウト以外)
 3 IMP処理実行タイムアウト
 1 0, 1 3 画面空き領域、または範囲外 (不当画面番号エラー)
 2 2 gain値設定範囲外
 4 8 動的メモリ確保エラー
 (ユーザープログラムによるメモリ破壊がないか確認して下さい)
 3 0 SRCOウインドウ設定エラー
 3 3 DSTウインドウ設定エラー
 4 9 アクセス画面が不定画面
 5 1 ワーク画面確保エラー(不要な画面を解放してリトライして下さい)
 - エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面からソーベルによって輪郭強調を行い、デスティネーション画面に出力します。

輪郭強調では、入力画面の指定領域内の全画素に対して、注目画素を中心とした 3×3 近傍の局所領域で与えられた荷重係数との積和演算を行います。また、対象領域の周辺 (3×3 近傍に領域外が含まれる部分) 処理の場合は 0 を出力します。



$$d = \begin{vmatrix} f0 \times (-\text{gain}) + f1 \times (-2) \times \text{gain} + f2 \times (-\text{gain}) + f6 \times \text{gain} + f7 \times 2 \times \text{gain} + f8 \times \text{gain} \\ + f0 \times (-\text{gain}) + f3 \times (-2) \times \text{gain} + f6 \times (-\text{gain}) + f2 \times \text{gain} + f5 \times 2 \times \text{gain} + f8 \times \text{gain} \end{vmatrix}$$

2.29.2 輪郭強調2値化(ソーベル2値化) (IP_SobelBinarize)

□ C言語API

```
int ret = IP_SobelBinarize( IMGID ImgSrc , IMGID ImgDst , int thr );
```

□ パラメータ

IMGID ImgSrc ソース画面の画面番号

IMGID ImgDst デスティネーション画面の画面番号

int thr 2 値化しきい値

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	-	-	◆

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

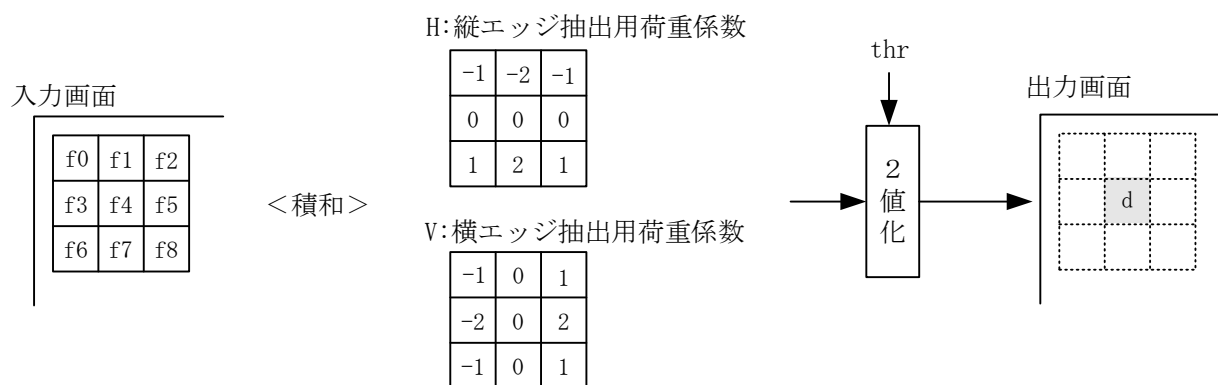
2 OSでのエラー(タイムアウト以外)
3 IMP処理実行タイムアウト
1 0, 1 3 画面空き領域、または範囲外 (不当画面番号エラー)
2 2 2 値化しきい値設定範囲外
4 8 動的メモリ確保エラー
 (ユーザープログラムによるメモリ破壊がないか確認して下さい)
3 0 SRCO ウィンドウ設定エラー
3 3 DST ウィンドウ設定エラー
4 9 アクセス画面が不定画面
5 1 ワーク画面確保エラー (不要な画面を解放してリトライして下さい)
- エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

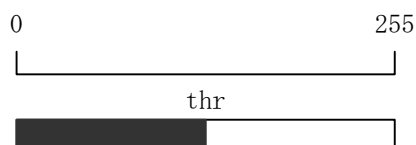
機能

濃淡画像からソーベルによって輪郭強調を行い、その結果をしきい値によって2値化します。

輪郭強調では、入力画面の指定領域内の全画素に対して、注目画素を中心とした 3×3 近傍の局所領域で与えられた荷重係数との積和演算を行います。その結果をパラメータ `thr` で指定するしきい値以上の値を白、それより小さい値を黒で2値化します。また、対象領域の周辺（ 3×3 近傍に領域外が含まれる部分）処理の場合は0を出力します。



$$d = \begin{vmatrix} f_0 \times (-1) + f_1 \times (-2) + f_2 \times (-1) + f_6 + f_7 \times 2 + f_8 \\ f_0 \times (-1) + f_3 \times (-2) + f_6 \times (-1) + f_2 + f_5 \times 2 + f_8 \end{vmatrix}$$



2.29.3 濃度変換テーブルの登録（IP_RegisterLUT）

□ C言語API

```
int ret = IP_RegisterLUT( int lutno , CNVLUT *lut );
```

□ パラメータ

int lutno 濃度変換テーブル登録番号（1～8）

CNVLUT *lut 濃度変換データ格納テーブル（32bit×256）のポインタ。
このテーブルに濃度変換のデータを設定して下さい。

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

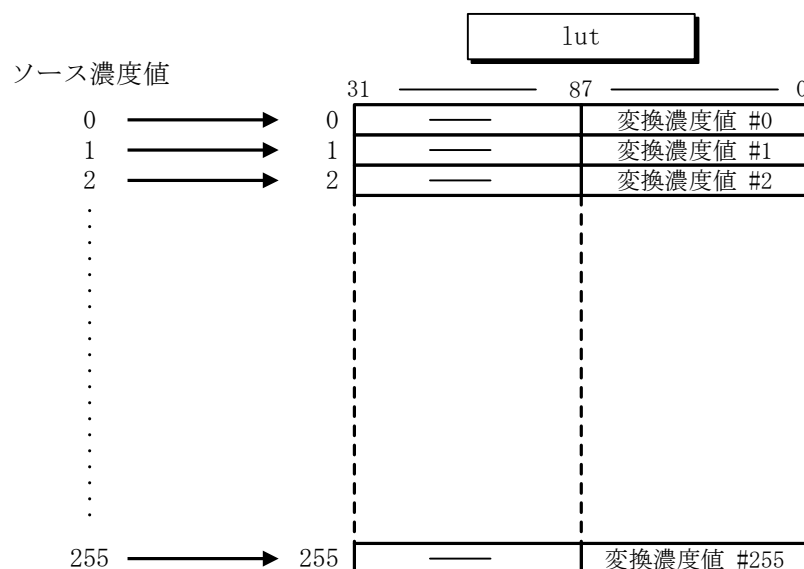
2 0 濃度変換テーブル登録番号設定範囲外
2 1 濃度変換データ格納テーブル設定エラー
- エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

IP_EdgeFLTAbsExt() コマンドの濃度変換データを設定します。
また、ここで設定したLUTのデータは、ラベリングコマンドを実行すると書き換えられますので注意して下さい。

濃度変換テーブルの詳細



2.29.4 輪郭強調(プレビット) (IP_Prewitt)

□ C言語API

```
int ret = IP_Prewitt( IMGID ImgSrc , IMGID ImgDst , int gain );
```

□ パラメータ

IMGID ImgSrc ソース画面の画面番号

IMGID ImgDst デスティネーション画面の画面番号

int gain ゲイン (1 ~ 8) 通常は「1」を指定してください。

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇S	◇S	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

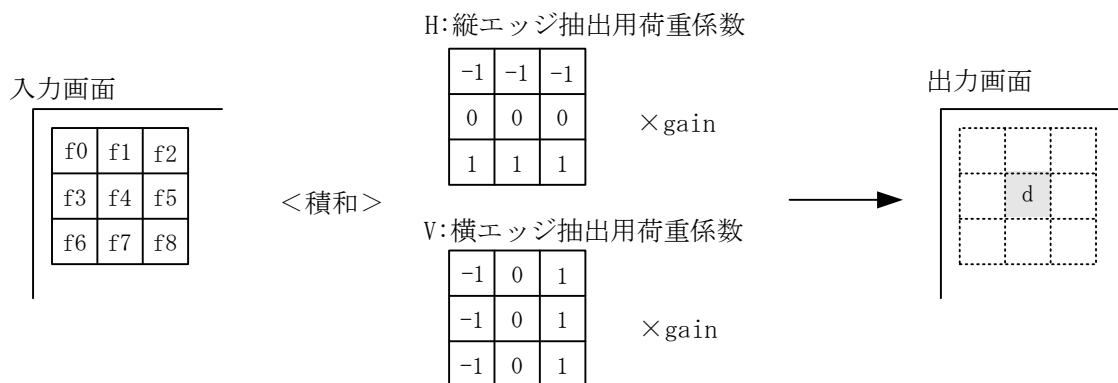
2	OSでのエラー(タイムアウト以外)
3	IMP処理実行タイムアウト
1 0 , 1 3	画面空き領域、または範囲外 (不当画面番号エラー)
2 2	gain値設定範囲外
4 8	動的メモリ確保エラー (ユーザープログラムによるメモリ破壊がないか確認して下さい)
3 0	S R C O ウィンドウ設定エラー
3 3	D S T ウィンドウ設定エラー
4 9	アクセス画面が不定画面
5 1	ワーク画面確保エラー(不要な画面を解放してリトライして下さい)
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソース画面からプレビットにより輪郭強調を行い、デスティネーション画面に出力します。

輪郭強調では、入力画面の指定領域内の全画素に対して、注目画素を中心とした 3×3 近傍の局所領域で与えられた荷重係数との積和演算を行います。また、対象領域の周辺（ 3×3 近傍に領域外が含まれる部分）処理の場合は 0 を出力します。



$$d = |f0 \times (-\text{gain}) + f1 \times (-2) \times \text{gain} + f2 \times (-\text{gain}) + f6 \times \text{gain} + f7 \times 2 \times \text{gain} + f8 \times \text{gain}| \\ + |f0 \times (-\text{gain}) + f3 \times (-2) \times \text{gain} + f6 \times (-\text{gain}) + f2 \times \text{gain} + f5 \times 2 \times \text{gain} + f8 \times \text{gain}|$$

2.29.5 輪郭強調2値化(プレビット2値化) (IP_PrewittBinarize)

□ C言語API

```
int ret = IP_PrewittBinarize( IMGID ImgSrc , IMGID ImgDst , int thr );
```

□ パラメータ

IMGID ImgSrc ソース画面の画面番号

IMGID ImgDst デスティネーション画面の画面番号

int thr 2値化しきい値

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	○	○
ImgDst	○	△R	○	○	○	×	-	-	◆

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

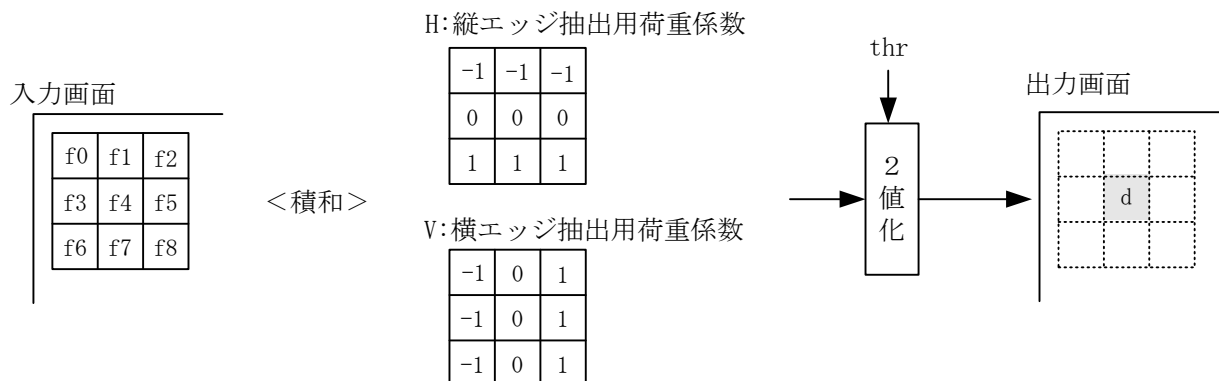
2 OSでのエラー(タイムアウト以外)
3 IMP処理実行タイムアウト
1 0, 1 3 画面空き領域、または範囲外 (不当画面番号エラー)
2 2 2値化しきい値設定範囲外
4 8 動的メモリ確保エラー
 (ユーザープログラムによるメモリ破壊がないか確認して下さい)
3 0 SRCOウインドウ設定エラー
3 3 DSTウインドウ設定エラー
4 9 アクセス画面が不定画面
5 1 ワーク画面確保エラー(不要な画面を解放してリトライして下さい)
- エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

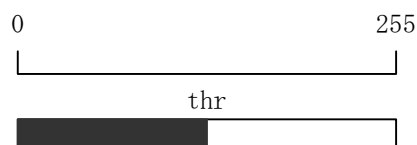
□ 機能

濃淡画像からプレビットにより輪郭強調を行い、その結果を2値化します。

輪郭強調では、入力画面の指定領域内の全画素に対して、注目画素を中心とした3×3近傍の局所領域で与えられた荷重係数との積和演算を行います。その結果をパラメータthrで指定するしきい値以上の値を白、それより小さい値を黒で2値化します。また、対象領域の周辺（3×3近傍に領域外が含まれる部分）処理の場合は0を出力します。



$$d = \begin{vmatrix} f0 \times (-1) + f1 \times (-2) + f2 \times (-1) + f6 + f7 \times 2 + f8 \\ f0 \times (-1) + f3 \times (-2) + f6 \times (-1) + f2 + f5 \times 2 + f8 \end{vmatrix}$$



2.30 線分化

2.30.1 線分列抽出 (ExtractPolyline)

□ C言語API

```
int ret = ExtractPolyline( IMGID ImgID , int sx , int sy , int col , int opt , POLY_TBL *poly ,
                          int maxnum );
```

□ パラメータ

IMGID	ImgID	画面番号
int	sx	探索開始X座標(画面絶対座標)
int	sy	探索開始Y座標(画面絶対座標)
int	col	探索対象物体の濃度値
int	opt	探索オプション

探索オプション	内容
0	探索した全ての座標を線分列抽出座標格納テーブルに出力します。
1	探索した座標のうち頂点のみを線分列抽出座標格納テーブルに出力します。

POLY_TBL *poly 線分列抽出座標格納テーブル。座標値は画面絶対座標です。

int maxnum 抽出線分列最大数

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	△R	○	○	○	×	○	○	○

□ リターンパラメータ

int ret > 0 : 抽出線分列数
 - 1 : 異常終了
 POLY_TBL *poly 線分列抽出座標を格納したアドレス

□ エラーコード

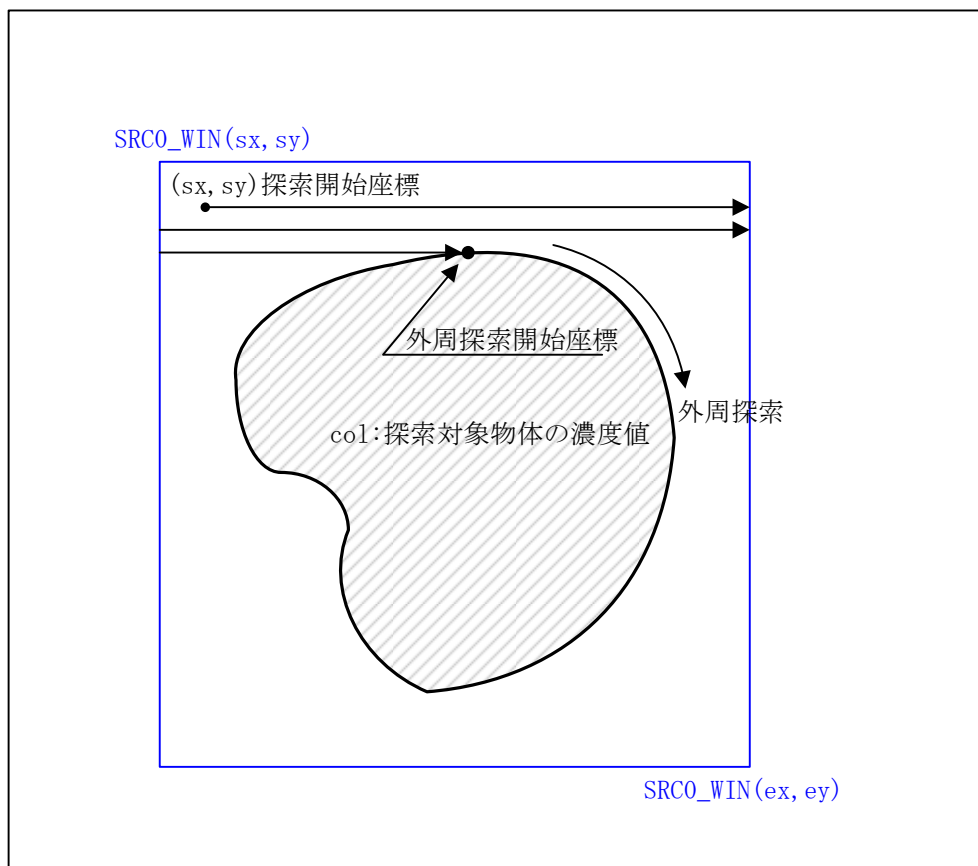
2	OSでのエラー(タイムアウト以外)
3	IMP処理実行タイムアウト
8	VIN入力タイムアウト
1 0	画面空き領域、または範囲外 (不当画面番号エラー)
2 1 ~ 2 6	パラメータエラー
3 0	S R C O ウィンドウ設定エラー
5 6	線分列未検出エラー
4 9	アクセス画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

指定画面の指定した2値画像オブジェクトの線分列（外周座標）の抽出を行います。

画面全体



ExtractPolyline() コマンドは、指定画面の探索開始座標 (sx, sy) からラスタースキャンで探索対象物体濃度値 (col) で指定された探索対象物体の外周探索開始座標を取得します。取得した外周探索開始座標から時計回りに探索対象物体の外周を探索し、再び外周探索開始座標に到達するか、抽出線分列最大数 (maxnum) になるまで探索し、線分列抽出座標格納テーブルに画面絶対座標を出力します。

ExtractPolyline() コマンドの探索開始座標 (sx, sy) が探索対象物体に近いほど処理時間は短くなります。また、探索対象物体がSRC0_WINで指定した座標に接しているか、それをまたいでいる場合、処理はエラーで終了します。

POLY_TBL のフォーマット及び内容を、下記に示します。

ipxsys.h で宣言しています。

```
typedef struct {
    int      num;    // 有効データ数
    WPOINT*  pos;    // X , Y 座標配列
} POLY_TBL;
```

```
typedef struct {
    short    x;      // X 座標
    short    y;      // Y 座標
} WPOINT;
```


2.30.2 線分列からの面積抽出 (PolyArea)

□ C言語API

```
int ret = PolyArea( POLY_TBL *poly , double *area );
```

□ パラメータ

POLY_TBL *poly 線分列抽出座標

double *area 面積(画素)

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)
double *area 面積(画素)を格納したアドレス

□ エラーコード

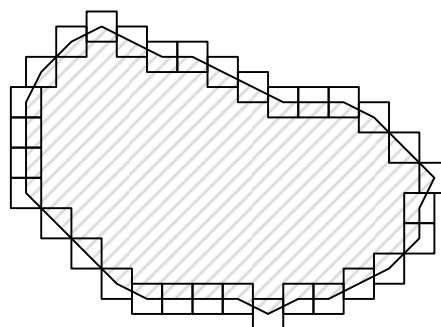
2 0 線分列抽出座標を格納したアドレスが不正、または有効データ数範囲外
2 1 面積を格納したアドレスが不正
- エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

線分列から面積値の抽出を行います。

面積は、外周座標値の中間点を結ぶ閉曲面内の面積を計算します。なお、穴があった場合でも穴の面積を除外しません。



POLY_TBL のフォーマットは、ExtractPolyline() の詳細情報の欄を参照して下さい。

2.30.3 線分列からの周囲長抽出 (PolyPerim)

□ C言語API

```
int ret = PolyPerim( POLY_TBL *poly , double *perim );
```

□ パラメータ

POLY_TBL *poly 線分列抽出座標

double *perim 周囲長(画素)

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)
double *perim 周囲長(画素)を格納したアドレス

□ エラーコード

2 0 線分列抽出座標を格納したアドレスが不正、または有効データ数範囲外
2 1 面積を格納したアドレスが不正
- エラーリセットコマンド (ClearIPError) 未発行

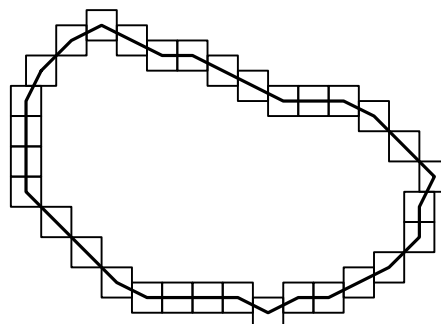
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

線分列から周囲長の抽出を行います。

POLY_TBL のフォーマット及び内容は、PolyArea() を参照して下さい。

周囲長は、外周座標値の中間点を結ぶ線の長さを計算します。



POLY_TBL のフォーマットは、ExtractPolyline() の詳細情報の欄を参照して下さい。

2.30.4 線分列からの重心座標抽出 (PolyGrav)

□ C言語API

```
int ret = PolyGrav( POLY_TBL *poly , float *gx , float *gy );
```

□ パラメータ

POLY_TBL	*poly	線分列抽出座標
float	*gx	重心X座標
float	*gy	重心Y座標

□ 対応画面タイプ/画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)
float	*gx	重心X座標を格納したアドレス
float	*gy	重心Y座標を格納したアドレス

□ エラーコード

2 0	線分列抽出座標を格納したアドレスが不正、または有効データ数範囲外
2 1	重心X座標を格納したアドレスが不正
2 2	重心Y座標を格納したアドレスが不正
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

線分列から重心座標の抽出を行います。

POLY_TBL のフォーマットは、ExtractPolyline() の詳細情報の欄を参照して下さい。

2.30.5 線分列からの形状特徴量抽出 (PolyFeatures)

□ C言語API

```
int ret = PolyFeatures( POLY_TBL *poly , POLY_FEATURE *tbl , int opt );
```

□ パラメータ

POLY_TBL *poly 線分列抽出座標

POLY_FEATURE *tbl ヒストグラム特徴量テーブル

int opt 抽出オプション。抽出する特徴量を指定して下さい。

抽出オプション	定数	内 容
POLY_CENTER_of_GRAVITY	0	外接座標+周囲長+面積+重心
POLY_MOMENT	1	上記+モーメント+慣性主軸角度
POLY_AXIS LENG	2	上記+慣性主軸長+パターン幅
POLY_MAX LENG	3	上記+絶対最大長+パターン幅

□ 対応画面タイプ/画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)
POLY_FEATURE *tbl ヒストグラム特徴量を格納したアドレス

□ エラーコード

2 0 線分列抽出座標を格納したアドレスが不正、または有効データ数範囲外
2 1 ヒストグラム特徴量テーブルのアドレスが不正
2 2 オプション設定範囲外
— エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

線分列から形状特徴量の抽出を行います。

POLY_TBL のフォーマットは、ExtractPolyline()の詳細情報の欄を参照して下さい。

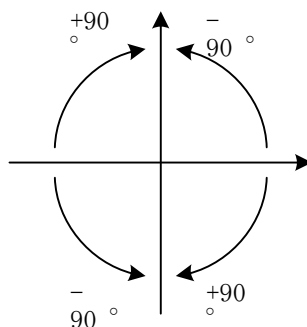
POLY_FEATURE のフォーマット及び内容を、下記に示します。

ipxsys.h で宣言しています。

```
typedef struct {
    int LEFT_UP_POINT_X;
    int LEFT_UP_POINT_Y;
    int REGION_X0;
    int REGION_Y0;
    int REGION_X1;
    int REGION_Y1;
    double PERIMETER;
    double AREA;
    double CENTER_of_GRAVITY_X;
    double CENTER_of_GRAVITY_Y;
    double ANGLE;
    double MOMENT_SIGMA_X;
    double MOMENT_SIGMA_Y;
    double MOMENT_SIGMA_XX;
    double MOMENT_SIGMA_XY;
    double MOMENT_SIGMA_YY;
    float AXIS LENG;
    float AXIS_WIDTH;
    int AXIS LENG_X0;
    int AXIS LENG_Y0;
    int AXIS LENG_X1;
    int AXIS LENG_Y1;
    float MAX LENG;
    float WIDTH;
    float ORIENTATION;
    int MAX LENG_X0;
    int MAX LENG_Y0;
    int MAX LENG_X1;
    int MAX LENG_Y1;
} POLY_FEATURE;
```

ヒストグラム特徴テーブル	内 容
LEFT_UP_POINT_X	物体が最初に出現する点のX座標
LEFT_UP_POINT_Y	物体が最初に出現する点のY座標
REGION_X0	物体が存在する最小X座標
REGION_Y0	物体が存在する最小Y座標
REGION_X1	物体が存在する最大X座標
REGION_Y1	物体が存在する最大Y座標
PERIMETER	周囲長
AREA	物体の面積
CENTER_of_GRAVITY_X	重心X座標
CENTER_of_GRAVITY_Y	重心Y座標
ANGLE	慣性主軸とX軸とのなす角度 (degree) (注)
MOMENT_SIGMA_X	X軸への1次モーメント
MOMENT_SIGMA_Y	Y軸への1次モーメント
MOMENT_SIGMA_XX	X軸への2次モーメント
MOMENT_SIGMA_XY	共役モーメント
MOMENT_SIGMA_YY	Y軸への2次モーメント
AXIS LENG	慣性主軸の軸長
AXIS_WIDTH	慣性主軸の軸長を挟む2本の直線間の距離
AXIS LENG_X0	慣性主軸の軸長を示す対角線X座標
AXIS LENG_Y0	慣性主軸の軸長を示す対角線Y座標
AXIS LENG_X1	慣性主軸の軸長を示す対角線X座標
AXIS LENG_Y1	慣性主軸の軸長を示す対角線Y座標
MAX LENG	絶対最大長
WIDTH	絶対最大長を挟む2本の直線間の距離
ORIENTATION	絶対最大長の方角と水平線との角度 (degree) (注)
MAX LENG_X0	絶対最大長を示す対角線X座標
MAX LENG_Y0	絶対最大長を示す対角線Y座標
MAX LENG_X1	絶対最大長を示す対角線X座標
MAX LENG_Y1	絶対最大長を示す対角線Y座標

(注)



2.31 2値画像の穴埋め

2.31.1 2値画像の穴埋め（IP_FillHole）

□ C言語API

```
int ret = IP_FillHole( IMGID ImgSrc , IMGID ImgDst , int opt , int col , OBJ_AREA_OPT *hole_opt );
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

int opt オプション

オプション	内容
0	4 連結ラベリングによる処理
1	8 連結ラベリングによる処理

int col 穴埋めする穴の色

色	内容
0	黒の穴を埋める
1	白の穴を埋める

OBJ_AREA_OPT *hole_opt 穴埋めする穴の面積閾値。
面積判定しない場合はNULLを指定してください。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B 2	△B 2	○
ImgDst	○	△R	○	○	○	×	—	—	◆

□ リターンパラメータ

int ret 正常終了（0）
異常終了（-1）
ラベルオーバーフロー（0 x 7 F F F）

□ エラーコード

1 0 ImgSrcの不当画面番号エラー
1 3 ImgDstの不当画面番号エラー
2 2～2 4 パラメータエラー
4 9 ImgSrcの不定画面エラー
5 1 ワーク画面確保エラー
— エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

2 値画像の穴埋めを行います。2 値のソース画像に対し、パラメータで指定された設定値でラベリング処理を行い、2 値画像の穴埋めをします。なお、ラベリング処理を用いているため、2 5 5 個以上の穴埋めはできません。

ラベルオーバーフローの場合はコマンドエラーにならないためClearIPError()は必要ありません。引き続き次の処理を実行します。

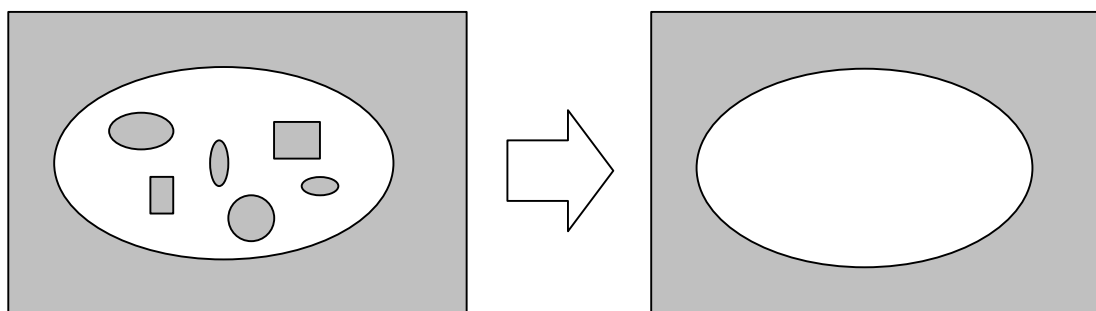
OBJ_AREA_OPTのフォーマットを下記に示します。

ipxsys.hで宣言しています。

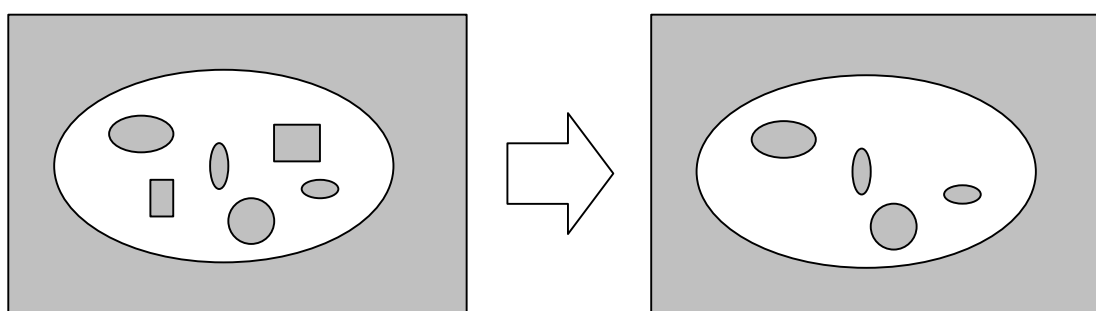
```
typedef struct {
    int    min;    // 最小面積
    int    max;    // 最大面積
} OBJ_AREA_OPT;
```

IP_FillHole() コマンドは、下図に示すように画面のSRC0_WINの指定領域中に対象物が 1 個の場合を想定しています。指定領域内に物体が複数存在する場合は、IP_FillHoleExt() コマンドを使用してください。

◆ hole_opt= NULL の場合



◆ hole_opt 指定の場合



2.31.2 2値画像の穴埋め(拡張) (IP_FillHoleExt)

□ C言語API

```
int ret = IP_FillHoleExt( IMGID ImgSrc , IMGID ImgDst , int opt , int col ,
                        OBJ_AREA_OPT *hole_opt , OBJ_AREA_OPT *obj_opt);
```

□ パラメータ

IMGID ImgSrc ソース画面番号

IMGID ImgDst デスティネーション画面番号

int opt オプション

オプション	内容
0	4 連結ラベリングによる処理
1	8 連結ラベリングによる処理

int col 穴埋めする穴の色

色	内容
0	黒の穴を埋める
1	白の穴を埋める

OBJ_AREA_OPT *hole_opt 穴埋めする穴の面積閾値。
面積判定しない場合はNULLを指定してください。

OBJ_AREA_OPT *obj_opt 穴埋めする対象物体の抽出面積閾値。
面積判定しない場合はNULLを指定してください。
NULLを指定すると、指定領域内で最大面積の物体を
抽出し、穴埋めを行います。

※OBJ_AREA_OPTのフォーマットは、IP_FillHole() コマンドを参照してください。

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	△B 2	△B 2	○
ImgDst	○	△R	○	○	○	×	—	—	◆

□ リターンパラメータ

int ret 正常終了 (0)
異常終了 (-1)
ラベルオーバーフロー (0 x 7 F F F)

□ エラーコード

1 0 ImgSrcの不当画面番号エラー
1 3 ImgDstの不当画面番号エラー
2 2 ~ 2 5 パラメータエラー
4 9 ImgSrcの不定画面エラー
5 1 ワーク画面確保エラー
— エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

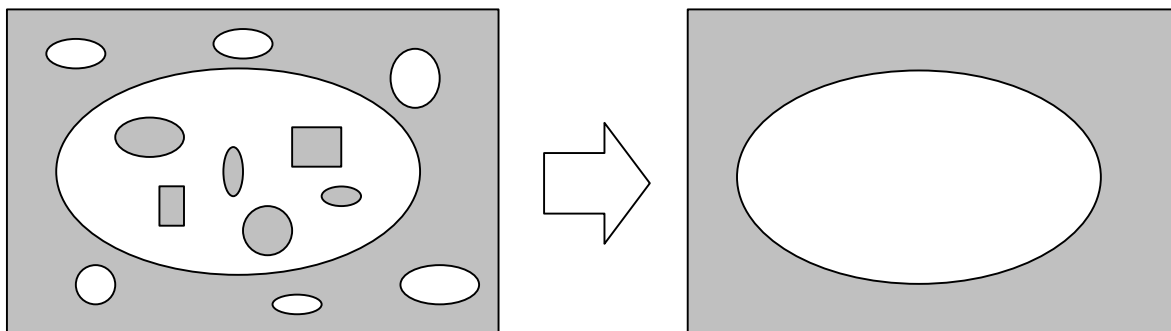
2 値画像の対象物体の抽出を行い抽出した対象物の穴埋めを行います。拡張機能として穴埋めする対象物体の抽出面積閾値の設定ができます。

IP_FillHoleExt() コマンドは、まず、2 値のソース画像に対してラベリング処理を行い、obj_opt の指定条件で対象物体を抽出します。このとき、指定領域内に複数物体が存在する場合には最大面積の物体が対象になり、それ以外の物体は削除されるので注意が必要です。次に抽出した対象物体に対して、hole_opt の指定条件で穴埋め処理を実行します。

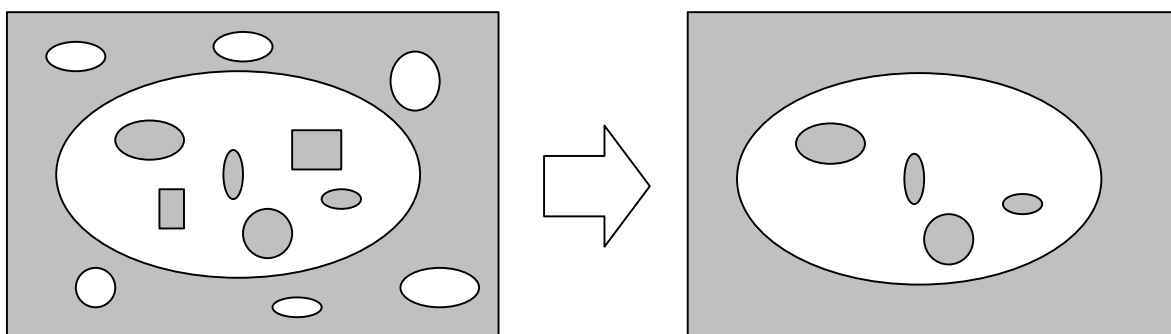
なお、本コマンドではラベリング処理を用いているため、255 個以上の穴埋めはできません。

ラベルオーバーフローの場合はコマンドエラーにならないため ClearIPError() は必要ありません。引き続き次の処理を実行します。

◆ hole_opt= NULL、obj_opt 指定の場合



◆ hole_opt、obj_opt 指定の場合



2.32 RGLUT変換

2.32.1 RGLUTオブジェクトの生成 (CreateRGLUT)

□ C言語API

```
HRGLUT ret = CreateRGLUT( int mode , int opt );
```

□ パラメータ

int mode L U Tモード

L U Tモード	定数値	内 容
RGLUT888	0	R:8bit / G:8bit / B:8bit (16,777,216byte)
RGLUT777	1	R:7bit / G:7bit / B:7bit (2,097,152byte)
RGLUT666	2	R:6bit / G:6bit / B:6bit (262,144byte)
RGLUT887	3	R:8bit / G:8bit / B:7bit (8,388,608byte)
RGLUT878	4	R:8bit / G:7bit / B:8bit (8,388,608byte)
RGLUT788	5	R:7bit / G:8bit / B:8bit (8,388,608byte)
RGLUT787	6	R:7bit / G:8bit / B:7bit (4,194,304byte)
RGLUT2CH88	7	R:8bit / G:8bit (65,536byte)
RGLUT565	8	R:5bit / G:6bit / B:5bit (65,536byte)

int opt オプション ("0"を設定してください)

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

HRGLUT ret R G B L U Tハンドル (0以外) または異常終了 (0)

□ エラーコード

2 0 L U Tモードの設定範囲外
 2 1 オプションの設定範囲外
 4 6 メモリヒープ領域空きなし
 - エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

機能

R G B L U Tオブジェクトの生成を行います。R G B L U TオブジェクトはオンボードC P U上に生成され、R G B L U Tハンドルで管理されます。R G B L U Tオブジェクトの構成を以下に示します。

L U Tモード(mode)は、濃度変換処理と濃度変換テーブル(L U T)の大きさを決定します。例えば、R G B L U T888は、Rデータが0～2 5 5 (8bit)、Gデータが0～2 5 5 (8bit)、Bデータが0～2 5 5 (8bit)のR G Bカラー画像を濃淡画像0～2 5 5 (8bit)に変換するモードで、本コマンド実行時に16,777,216byteの変換テーブルを確保します。

本コマンドを実行してから、OpenR G B L U T()の実行によりL U Tの先頭アドレスを取得して、直接、L U Tにアクセスすることができます。L U Tの設定後、IP_ConvertR G B L U T()およびIP_ConvertR G B L U TEx()で、R G BのL U T変換を実行します。

R G B L U T変換が終了したら、DeleteR G B L U T()でR G B L U Tオブジェクトを削除してください。R G B L U Tオブジェクトを削除しないで、本コマンドを連続して実行するとメモリ不足によるエラーが発生します。

H R G B L U Tハンドル h L U T

mode
opt
reserved
size
*heap
*lut

L U Tモード

オプション

(予約)

L U Tサイズ

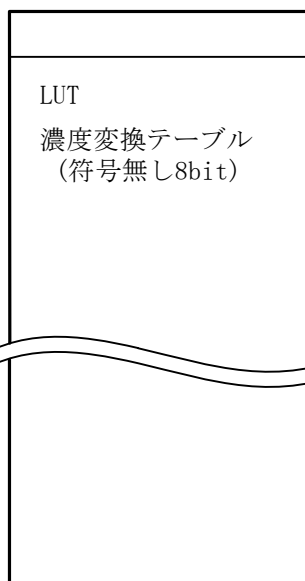
R G B L U T_HEADER

mallocで確保したアドレスへのポインタ

128byteアラインされたアドレスへのポインタ

heap

lut



濃度変換テーブル

2.32.2 RGLUTオブジェクトの削除 (DeleteRGLUT)

□ C言語API

```
int ret = DeleteRGLUT (HRGLUT hLUT);
```

□ パラメータ

HRGLUT hLUT LUTハンドル

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

— エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

RGLUTオブジェクトの削除を行います。
CleaveRGLUT() を実行した場合は、必ず、本コマンドを実行してください。

2.32.3 RGBLUTオブジェクトの使用開始 (OpenRGBLUT)

□ C言語API

```
int ret = OpenRGBLUT(HRGBLUT hLUT, int *size, void **RGBTbl, int mode);
```

□ パラメータ

HRGBLUT	hLUT	LUTハンドル
int	*size	RGBLUTのLUTサイズを返すアドレス
void	**RGBTbl	RGBLUTのLUT先頭アドレスを返すアドレス
int	mode	LUTテーブル読み出しモード 0 : LUTデータをSH側からPC側へ読み出さない。 1 : LUTデータをSH側からPC側へ読み出す。

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)
int	*size	RGBLUTのLUTサイズを格納したアドレス
void	**RGBTbl	RGBLUTのLUT先頭アドレスを格納したアドレス

□ エラーコード

2	OSシステムエラー
20	RGBLUTハンドル値設定範囲外
21	LUTのサイズを返すアドレスが不正
22	LUTの先頭アドレスを返すアドレスが不正
23	LUT読み出しモード設定範囲外
48	動的メモリ確保エラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

R G B L U T ハンドルで示される R G B L U T オブジェクトの濃度変換テーブル (L U T) のサイズと先頭アドレスを取得し、直接アクセスを可能にします。

L U T へは、直接データを書き込むか、あらかじめ「ipxdef.h」に定義されている以下のマクロを用いて設定します。ここで、lut は L U T の先頭アドレス、r、g、b はそれぞれ R G B 画像の R データ、G データ、B データ、d は変換データです。マクロ定義から分かるように、r、g、b 値が L U T のアドレスを示すパラメータになります。

L U T テーブルへのデータ書き込みマクロ

```
#define SetRGLUT888(lut,r,g,b,d) lut[r << 16 | g << 8 | b]= (d)
#define SetRGLUT777(lut,r,g,b,d) lut[(r & 0xfe) << 13 | (g & 0xfe) << 6 | b >> 1]= (d)
#define SetRGLUT666(lut,r,g,b,d) lut[(r & 0xfc) << 10 | (g & 0xfc) << 4 | b >> 2]= (d)
#define SetRGLUT887(lut,r,g,b,d) lut[r << 15 | g << 7 | b >> 1]= (d)
#define SetRGLUT878(lut,r,g,b,d) lut[r << 15 | (g & 0xfe) << 7 | b]= (d)
#define SetRGLUT788(lut,r,g,b,d) lut[(r & 0xfe) << 15 | g << 8 | b]= (d)
#define SetRGLUT787(lut,r,g,b,d) lut[(r & 0xfe) << 14 | g << 7 | b >> 1]= (d)
#define SetRGLUT2CH88(lut,r,g,d) lut[r << 8 | g]= (d)
#define SetRGLUT565(lut,r,g,b,d) lut[(r & 0xf8) << 8 | (g & 0xfc) << 5 | b >> 3]= (d)
```

L U T テーブルへのデータ読み出しマクロ

```
#define GetRGLUT888(lut,r,g,b) lut[r << 16 | g << 8 | b]
#define GetRGLUT777(lut,r,g,b) lut[(r & 0xfe) << 13 | (g & 0xfe) << 6 | b >> 1]
#define GetRGLUT666(lut,r,g,b) lut[(r & 0xfc) << 10 | (g & 0xfc) << 4 | b >> 2]
#define GetRGLUT887(lut,r,g,b) lut[r << 15 | g << 7 | b >> 1]
#define GetRGLUT878(lut,r,g,b) lut[r << 15 | (g & 0xfe) << 7 | b]
#define GetRGLUT788(lut,r,g,b) lut[(r & 0xfe) << 15 | g << 8 | b]
#define GetRGLUT787(lut,r,g,b) lut[(r & 0xfe) << 14 | g << 7 | b >> 1]
#define GetRGLUT2CH88(lut,r,g) lut[r << 8 | g]
#define GetRGLUT565(lut,r,g,b) lut[(r & 0xf8) << 8 | (g & 0xfc) << 5 | b >> 3]
```

リモートコマンドでは L U T サイズのワーク領域を確保し、このワーク領域にオンボード C P U に書き込まれている L U T データを読み出してデータの変更を行います。L U T をオンボード C P U から読み出す必要がない場合には、L U T 読み出しモード(mode)を"0"にしてください。L U T を読み出す時間を短縮することができます。

CloseRGLUT() では、L U T データをオンボード C P U に書き戻し、ワーク領域を解放しますので、L U T の設定が終了したら、必ず、CloseRGLUT() で L U T のアクセスを終了してください。

2.32.4 RGLUTオブジェクトの使用終了 (CloseRGLUT)

□ C言語API

```
int ret = CloseRGLUT(HRGLUT hLUT, void *RGBTbl);
```

□ パラメータ

HRGLUT	hLUT	LUTハンドル
void	*RGBTbl	RGLUTのLUT先頭アドレスを格納したアドレス

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
20	RGLUTハンドル値設定範囲外
21	LUTの先頭アドレスを格納したアドレスが不正
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

RGLUTハンドルで示されるRGLUTオブジェクトのLUT（濃度変換テーブル）の直接アクセスを終了します。

リモートコマンドではLUTデータをオンボードCPUに書き込み、OpenRGLUT()で確保したワーク領域を解放します。OpenRGLUT()実行後は、必ず、CloseRGLUT()を実行してください。

2.32.5 RGBLUT変換の実行 (IP_ConvertRGBLUT)

□ C言語API

```
int ret = IP_ConvertRGBLUT(IMGID ImgSrc , IMGID ImgDst , int opt , HRGBLUT hLUT);
```

□ パラメータ

IMGID	ImgSrc	R G B ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	opt	オプション ("0"を設定してください)
HRGBLUT	hLUT	L U Tハンドル

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	×	○	×	×	×	×	○	○	○
ImgDst	○	△R	○	○	○	×	◇1	◇1	◇1

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OSシステムエラー
3	IMP処理タイムアウト
8	V I N入力タイムアウト
10	R G B ソース画面の不当画面番号エラー
13	デスティネーション画面の不当画面番号エラー
22	オプションの設定範囲外
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

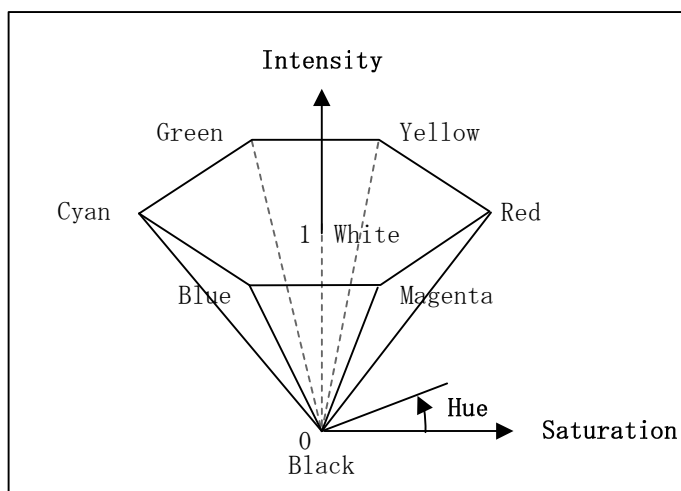
R G B画像のL U T変換を行います。L U Tハンドルで管理されるR G B L U TオブジェクトのL U Tモードに従い、R G BのL U T変換を実行します。
以下にL U Tモードに対する演算内容を示します。

LUTモード	演算内容
RGBLUT888	ImgDst = LUT[ImgSrc(R) << 16 ImgSrc(G) << 8 ImgSrc(B)]
RGBLUT777	ImgDst = LUT[(ImgSrc(R) & 0xFE) << 13 (ImgSrc(G) & 0xFE) << 6 ImgSrc(B) >> 1]
RGBLUT666	ImgDst = LUT[(ImgSrc(R) & 0xFC) << 10 (ImgSrc(G) & 0xFC) << 4 ImgSrc(B) >> 2]
RGBLUT887	ImgDst = LUT[ImgSrc(R) << 15 ImgSrc(G) << 7 ImgSrc(B) >> 1]
RGBLUT878	ImgDst = LUT[ImgSrc(R) << 15 (ImgSrc(G) & 0xFE) << 7 ImgSrc(B)]
RGBLUT788	ImgDst = LUT[(ImgSrc(R) & 0xFE) << 15 ImgSrc(G) << 8 ImgSrc(B)]
RGBLUT787	ImgDst = LUT[(ImgSrc(R) & 0xFE) << 14 ImgSrc(G) << 7 ImgSrc(B) >> 1]
RGBLUT2CH88	ImgDst = LUT[ImgSrc(R) << 8 ImgSrc(G)]
RGBLUT565	ImgDst = LUT[(ImgSrc(R) & 0xF8) << 8 (ImgSrc(G) & 0xFC << 5) ImgSrc(B) << 3]

[RGBカラー画像の Hue(色相)、Saturation(彩度)、Intensity(明度)画像への変換例]

IP_ConvertRGBLUT()やIP_ConvertRGBLUT()では、RGBのLUT変換を用いてRGBカラー画像をHue(色相)、Saturation(彩度)、Intensity(明度)の画像に変換することができます。

次に色相、彩度、明度の関係を示した図、画像変換の演算式およびRGBLUT変換例を示します。



演算式	$I = \max(R, G, B)$ $I = 0: S = 0: H = 0(\text{不定})$ $I \neq 0: i = \min(R, G, B)$ $S = (I - i) * 255 / I$ $R = I: H = (G - B) * 60 / (I - i)$ $G = I: H = (B - R) * 60 / (I - i) + 120$ $B = I: H = (R - G) * 60 / (I - i) + 240$ ($H < 0$ の時は、$H = H + 360$ とする) H: Hue, S: Saturation, I: Intensity

画像メモリには、0～255までしかデータを格納することができません。そこで、0～360度までのHueの値を0.71倍するなどしてLUTに設定してください。

R G B画像の色相画像への変換例を以下に示します。

```

/* 画面確保 */
ImgRGB = AllocRGBImg( IMG_FS_512H_512V );
ImgID  = AllocImg( IMG_FS_512H_512V );

.....

/* RGLUTオブジェクト作成 */
mode = RGLUT888;
hLUT = CreateRGLUT( mode , 0 );

/*****
  R G B画像から色相(Hue)画像へ変換
*****/
/* RGLUTのLUTテーブルアクセス可能 */
mode = 0;
OpenRGLUT( hLUT, &size, &lut, mode );

/* LUTテーブル設定[色相(Hue)] */
for( r = 0; r < 256; r++ ){
    for( g = 0; g < 256; g++ ){
        for( b = 0; b < 256; b++ ){
            I = max( max( b, g ), max( g, r ) );
            i = min( min( b, g ), min( g, r ) );
            if( I == 0 ){
                H = 0;
            }else{
                val = 60.0f / ( I - i );
                if( I == r ){
                    H = ( g - b ) * val;
                }else if( I == g ){
                    H = ( b - r ) * val + 120;
                }else{
                    H = ( r - g ) * val + 240;
                }
                if( H < 0 ) H += 360;
                H *= 0.71f;
            }
            SetRGLUT888( lut, r, g, b, H );
        }
    }
}

/* RGLUTのLUTテーブルアクセス禁止 */
CloseRGLUT( hLUT, lut );

/* RGLUT変換 */
IP_ConvertRGLUT( ImgRGB, ImgID , 0 , hLUT);

.....

/* RGLUTオブジェクト削除 */
DeleteRGLUT( hLUT );

/* 画面解放 */
FreeImg( ImgID );
FreeImg( ImgRGB );

```

r, g, b値(0~255)で
H(Hue)値が決定されます

r, g, b値のときのH(Hue)値
をlutへ設定します

R G B 画像の彩度画像への変換例を以下に示します。

```

/* 画面確保 */
ImgRGB = AllocRGBImg( IMG_FS_512H_512V );
ImgID  = AllocImg( IMG_FS_512H_512V );

.....

/* RGLUTオブジェクト作成 */
mode = RGLUT888;
hLUT = CreateRGLUT( mode , 0 );

/*****
   R G B 画像から彩度(Saturation)画像へ変換
*****/
/* RGLUTのLUTテーブルアクセス可能 */
mode = 0;
OpenRGLUT( hLUT, &size, &lut, mode );

/* LUTテーブル設定[彩度(Saturation)] */
for( r = 0; r < 256; r++ ){
    for( g = 0; g < 256; g++ ){
        for( b = 0; b < 256; b++ ){
            I = max( max( b, g ), max( g, r ) );
            i = min( min( b, g ), min( g, r ) );
            if( I == 0 ){
                S = 0;
            }else{
                S = ( I - i ) * 255 / I;
            }
            SetRGLUT888( lut, r, g, b, S );
        }
    }
}

/* RGLUTのLUTテーブルアクセス禁止 */
CloseRGLUT( hLUT, lut );

/* RGLUT変換 */
IP_ConvertRGLUT( ImgRGB, ImgID , 0 , hLUT);

.....

/* RGLUTオブジェクト削除 */
DeleteRGLUT( hLUT );

/* 画面解放 */
FreeImg( ImgID );
FreeImg( ImgRGB );

```

R G B 画像の明度画像への変換例を以下に示します。

```

/* 画面確保 */
ImgRGB = AllocRGBImg( IMG_FS_512H_512V );
ImgID  = AllocImg( IMG_FS_512H_512V );

.....

/* RGLUTオブジェクト作成 */
mode = RGLUT888;
hLUT = CreateRGLUT( mode , 0 );

/* **** R G B 画像から明度(Intensity)画像へ変換 ****
**** **** */
/* RGLUTのLUTテーブルアクセス可能 */
mode = 0;
OpenRGLUT( hLUT, &size, &lut, mode );

/* LUTテーブル設定[明度(Intensity)] */
for( r = 0; r < 256; r++ ){
    for( g = 0; g < 256; g++ ){
        for( b = 0; b < 256; b++ ){
            I = max( max( b, g ), max( g, r ) );
            SetRGLUT888( lut, r, g, b, I);
        }
    }
}

/* RGLUTのLUTアクセス禁止 */
CloseRGLUT( hLUT, lut );

/* RGLUT変換 */
IP_ConvertRGLUT( ImgRGB, ImgID , 0 , hLUT);

.....

/* RGLUTオブジェクト削除 */
DeleteRGLUT( hLUT );

/* 画面解放 */
FreeImg( ImgID );
FreeImg( ImgRGB );

```

2.32.6 RGBLUT変換の実行(拡張)(IP_ConvertRGBLUTEx)

□ C言語API

```
int ret = IP_ConvertRGBLUTEx(IMGID ImgR , IMGID ImgG , IMGID ImgB , IMGID ImgDst ,
                             int opt , HRGBLUT hLUT);
```

□ パラメータ

IMGID	ImgR	R ソース画面番号
IMGID	ImgG	G ソース画面番号
IMGID	ImgB	B ソース画面番号
IMGID	ImgDst	デスティネーション画面番号
int	opt	オプション ("0"を設定してください)
HRGBLUT	hLUT	LUTハンドル

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgR	○	△R	○	○	○	—	○	○	○
ImgG	○	△R	○	○	○	—	○	○	○
ImgB	○	△R	○	○	○	—	○	○	○
ImgDst	○	△R	○	○	○	×	◇1	◇1	◇1

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2	OS システムエラー
3	IMP 処理タイムアウト
8	VIN 入力タイムアウト
10	RGB ソース画面の不当画面番号エラー
13	デスティネーション画面の不当画面番号エラー
22	オプションの設定範囲外
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

分離されたR, G, B画像のLUT変換を行います。指定されたLUTオブジェクトのLUTモードに従いRGBのLUT変換を実行します。

以下にLUTモードに対する演算内容を示します。

LUTモード	演算内容
RGBLUT888	$\text{ImgDst} = \text{LUT}[\text{ImgSrc}(\text{R}) \ll 16 \mid \text{ImgSrc}(\text{G}) \ll 8 \mid \text{ImgSrc}(\text{B})]$
RGBLUT777	$\text{ImgDst} = \text{LUT}[(\text{ImgSrc}(\text{R}) \& 0\text{xFE}) \ll 13 \mid (\text{ImgSrc}(\text{G}) \& 0\text{xFE}) \ll 6 \mid \text{ImgSrc}(\text{B}) \gg 1]$
RGBLUT666	$\text{ImgDst} = \text{LUT}[(\text{ImgSrc}(\text{R}) \& 0\textxFC}) \ll 10 \mid (\text{ImgSrc}(\text{G}) \& 0\textxFC}) \ll 4 \mid \text{ImgSrc}(\text{B}) \gg 2]$
RGBLUT887	$\text{ImgDst} = \text{LUT}[\text{ImgSrc}(\text{R}) \ll 15 \mid \text{ImgSrc}(\text{G}) \ll 7 \mid \text{ImgSrc}(\text{B}) \gg 1]$
RGBLUT878	$\text{ImgDst} = \text{LUT}[\text{ImgSrc}(\text{R}) \ll 15 \mid (\text{ImgSrc}(\text{G}) \& 0\text{xFE}) \ll 7 \mid \text{ImgSrc}(\text{B})]$
RGBLUT788	$\text{ImgDst} = \text{LUT}[(\text{ImgSrc}(\text{R}) \& 0\text{xFE}) \ll 15 \mid \text{ImgSrc}(\text{G}) \ll 8 \mid \text{ImgSrc}(\text{B})]$
RGBLUT787	$\text{ImgDst} = \text{LUT}[(\text{ImgSrc}(\text{R}) \& 0\text{xFE}) \ll 14 \mid \text{ImgSrc}(\text{G}) \ll 7 \mid \text{ImgSrc}(\text{B}) \gg 1]$
RGBLUT2CH88	$\text{ImgDst} = \text{LUT}[\text{ImgSrc}(\text{R}) \ll 8 \mid \text{ImgSrc}(\text{G})]$
RGBLUT565	$\text{ImgDst} = \text{LUT}[(\text{ImgSrc}(\text{R}) \& 0\text{xF8}) \ll 8 \mid (\text{ImgSrc}(\text{G}) \& 0\textxFC} \ll 5) \mid (\text{ImgSrc}(\text{B}) \ll 3)]$

RGBLUT変換例は、IP_ConvertRGBLUT()の機能の欄を参照してください。

2.33 RAW/RGB変換

2.33.1 RAW/RGB変換データ生成 (CreateConvertRAWData)

□ C言語API

```
int ret = CreateConvertRAWData( int xlng, int ylng );
```

□ パラメータ

int	xlng	RAW/RGB変換データX方向長 『SetVideoFrame()』コマンドで設定した映像入力サイズ (ビデオフレームサイズ) のX方向長以上の値を設定して ください。
int	ylng	RAW/RGB変換データY方向長 『SetVideoFrame()』コマンドで設定した映像入力サイズ (ビデオフレームサイズ) のY方向長以上の値を設定して ください。

□ 対応画面タイプ/画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

2 0	RAW/RGB変換データX方向長設定値範囲外
2 1	RAW/RGB変換データY方向長設定値範囲外
4 6	画面ID空きなし
4 8	動的メモリ確保エラー
1 3 8	RAW/RGB変換データ生成済み
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

RAW/RGB変換コマンドで使用するシステム管理データを生成します。本コマンドは、RAW/RGB変換コマンドを実行する前に実行してください。本コマンドを実行せずにRAW/RGB変換コマンドを実行すると各コマンドでエラーになります。

RAW/RGB変換コマンド

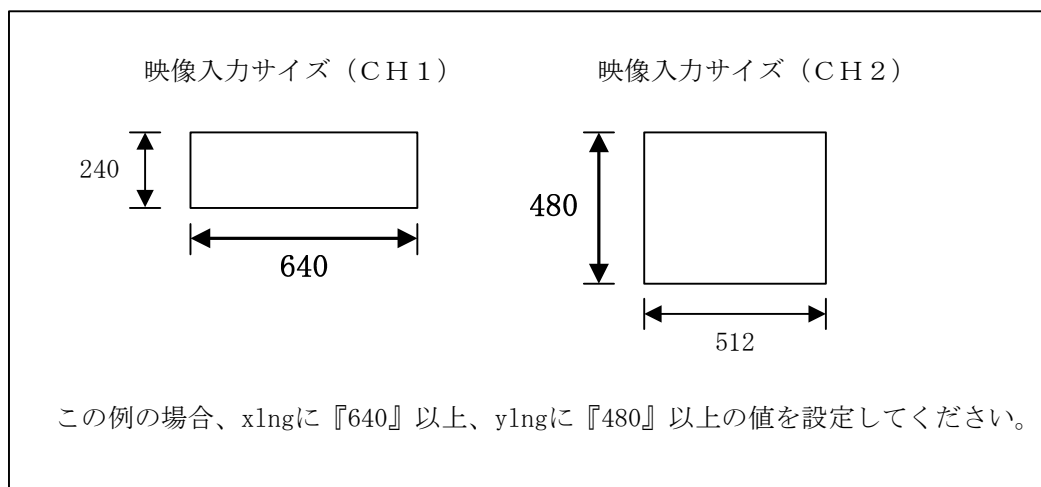
IP_ConvertRAWtoRGB
IP_ConvertRAWtoRGBEx

すでにRAW/RGB変換データを生成している場合、本コマンドを実行するとエラー (エラーコード: 1 3 8) になります。この場合は『DeleteConvertRAWData()』コマンドを実行してから本コマンドを実行してください。

RAW/RGB変換が不要になりましたら『DeleteConvertRAWData()』コマンドを実行してシステム管理データを削除してください。

RAW／RGB変換データX方向長／Y方向長 (xlng, ylng) は内部で使用するワーク画面のサイズです。ワーク画面は各カメラチャネルで共通になります。複数のカメラチャネルでRAW／RGB変換を行う場合は、各映像入力サイズの最大値以上の値を設定してください。

以下に設定例を示します。



本コマンドはワーク画面を2枚確保します。画像メモリの空きがなくワーク画面が確保できない場合は、エラー（エラーコード：48）となります。この場合は、不要な画面を『FreeImg()』コマンドで解放し、空き画面を確保してください。

2.33.2 RAW/RGB変換データ削除 (DeleteConvertRAWData)

□ C言語API

```
int ret = DeleteConvertRAWData( void );
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

—	エラーリセットコマンド (ClearIPErrors) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

RAW／RGB変換で使用するシステム管理データを削除します。

『CreateConvertRAWData()』コマンドを実行した場合は、必ず本コマンドを実行してください。

2.33.3 RAW/RGB変換ゲイン設定 (SetRAWtoRGBGain)

□ C言語API

```
int ret = SetRAWtoRGBGain( int ch, float Rgain, float Ggain, float Bgain );
```

□ パラメータ

int	ch	カメラチャネル番号 (0 ~ 1)
float	Rgain	R ゲイン (0 ~ 2 5 5)
float	Ggain	G ゲイン (0 ~ 2 5 5)
float	Bgain	B ゲイン (0 ~ 2 5 5)

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

2 0	カメラチャネル番号設定値範囲外
2 1	R ゲイン設定値範囲外
2 2	G ゲイン設定値範囲外
2 3	B ゲイン設定値範囲外
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

RAWカラーデータからRGB画像への変換時のゲインを設定します。『InitIP()』コマンドを実行するとデフォルトのゲインが設定されます。

デフォルトのゲイン設定値 (各チャネル共通)

R : 1. 2 3
G : 1. 0 0
B : 1. 9 5

2.33.4 RAWデータからRGBデータへの変換 (IP_ConvertRAWtoRGB)

□ C言語API

```
int ret = IP_ConvertRAWtoRGB( IMGID ImgSrc, IMGID ImgDst, int mode, int ch );
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号 (Y画面)
IMGID	ImgDst	デスティネーション画面番号 (RGB画面)
int	mode	変換モード (0～3)
int	ch	カメラチャネル番号 (0～1)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	×	×	×	×	×	○	○	○
ImgDst	×	○	×	×	×	×	◆	-	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

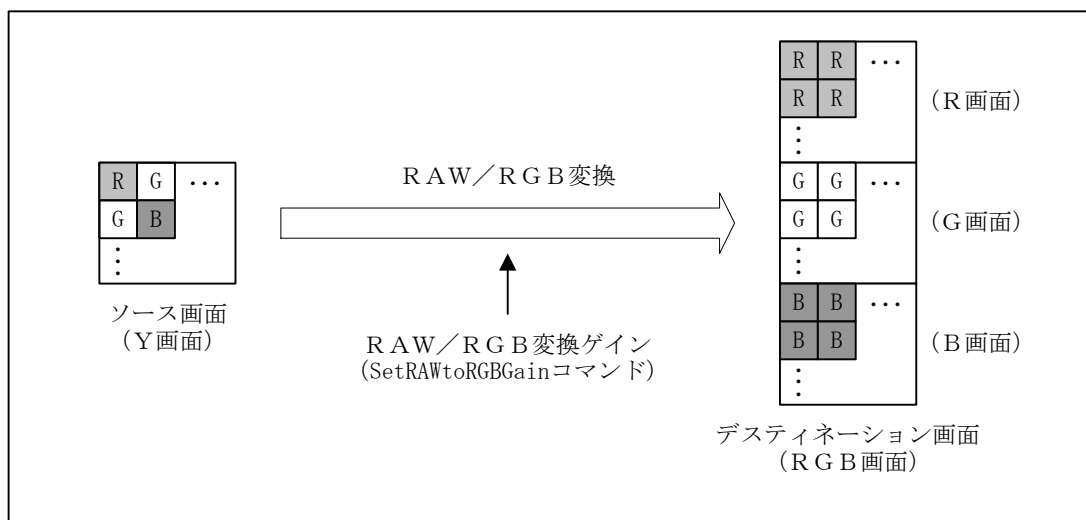
2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
1 0	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
1 3	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 2	変換モード設定値範囲外
2 3	カメラチャネル番号設定値範囲外
4 9	ソース画面が不定画面
1 3 8	RAW/RGB変換データ未生成 (CreateConvertRAWData未発行)、またはRAW/RGB変換データのワーク画面が映像入力サイズより小さい、または映像入力サイズ未設定 (SetVideoFrame未発行)
-	エラーリセットコマンド (ClearIPErrors) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

RAWデータを格納したソース画面をRGBデータに変換し、デスティネーション画面に出力します。処理領域は『SetVideoFrame()』コマンドで設定した映像入力サイズ (ビデオフレームサイズ) になります。また、デスティネーション画面の周辺1画素は0になります。

本コマンドの実行時に『CreateConvertRAWData()』コマンドでRAW/RGB変換データを生成していない場合、およびワーク画面が映像入力サイズより小さい場合、エラー (エラーコード: 1 3 8) になります。また、『SetVideoFrame()』コマンドを実行していない場合もエラー (エラーコード: 1 3 8) になります。カメラ接続を行わずにRAW/RGB変換を実行したい場合は、『IP_ConvertRAWtoRGBEx()』コマンドを使用してください。



変換モード (mode) はRAWデータのR/G/B配置情報です。ソース画面の原点 (0, 0) からのR/G/Bの配置に対応したモードを設定してください。配置と変換モードが合っていないとRAW/RGB変換で正しい画像を取得することができません。

変換モード	0	1	2	3
RAWデータ	(0, 0) X Y	(0, 0) X Y	(0, 0) X Y	(0, 0) X Y

RAWデータを表示する場合は、本コマンド実行後に『CombineRGB()』コマンドでRGB16画面に変換してください。

```

IP_ConvertRAWtoRGB( ImgBW, ImgRGB, mode, ch );    /* RAW ---> RGB 変換 */
CombineRGB( ImgRGB, ImgRGB16 );                    /* RGB ---> RGB16 変換 */
DispImg( ImgRGB16 );                               /* RGB16画面を表示 */

```

2.33.5 RAWデータからRGBデータへの変換(拡張) (IP_ConvertRAWtoRGBEx)

□ C言語API

```
int ret = IP_ConvertRAWtoRGBEx( IMGID ImgSrc, IMGID ImgDst, int mode, int ch,
                                int ssx, int ssy, int dsx, int dsy, int xlng, int ylng );
```

□ パラメータ

IMGID	ImgSrc	ソース画面番号 (Y画面)
IMGID	ImgDst	デスティネーション画面番号 (RGB画面)
int	mode	変換モード (0～3) 『IP_ConvertRAWtoRGB()』コマンドを参照してください。
int	ch	カメラチャネル番号 (0～1)
int	ssx	ソース画面転送開始点X座標
int	ssy	ソース画面転送開始点Y座標
int	dsx	デスティネーション画面書込み開始点X座標
int	dsy	デスティネーション画面書込み開始点Y座標
int	xlng	デスティネーション画面X方向転送幅
int	ylng	デスティネーション画面Y方向転送幅

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	×	×	×	×	×	○	○	○
ImgDst	×	○	×	×	×	×	◆	-	-

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

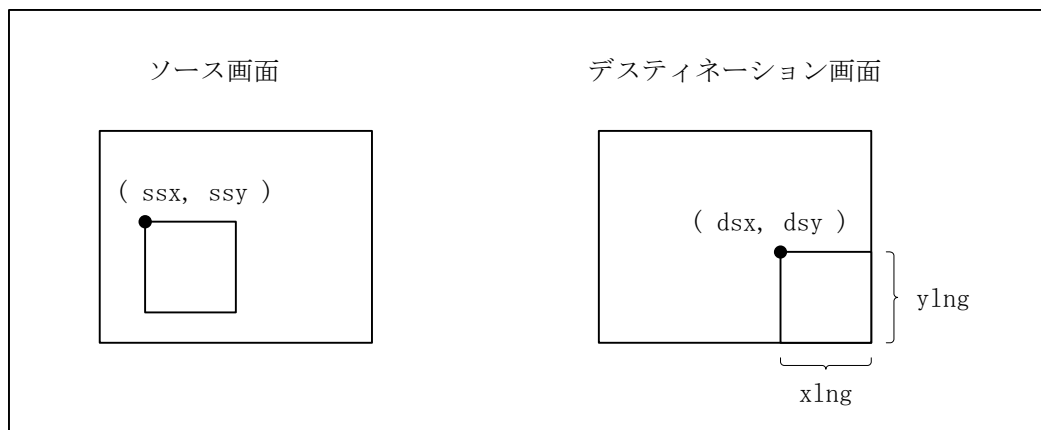
2	OSシステムエラー
3	IMP処理タイムアウト
8	VIN入力タイムアウト
10	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
13	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
22	変換モード設定値範囲外
23	カメラチャネル番号設定値範囲外
24	ソース画面転送開始点X座標設定値範囲外
25	ソース画面転送開始点Y座標設定値範囲外
26	デスティネーション画面書込み開始点X座標設定値範囲外
27	デスティネーション画面書込み開始点Y座標設定値範囲外
28	デスティネーション画面X方向転送幅設定値範囲外
29	デスティネーション画面Y方向転送幅設定値範囲外
49	ソース画面が不定画面
138	RAW/RGB変換データ未生成 (CreateConvertRAWData未発行)、またはRAW/RGB変換データのワーク画面がデスティネーション画面X/Y方向転送幅より小さい
-	エラーリセットコマンド (ClearIPErrors) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

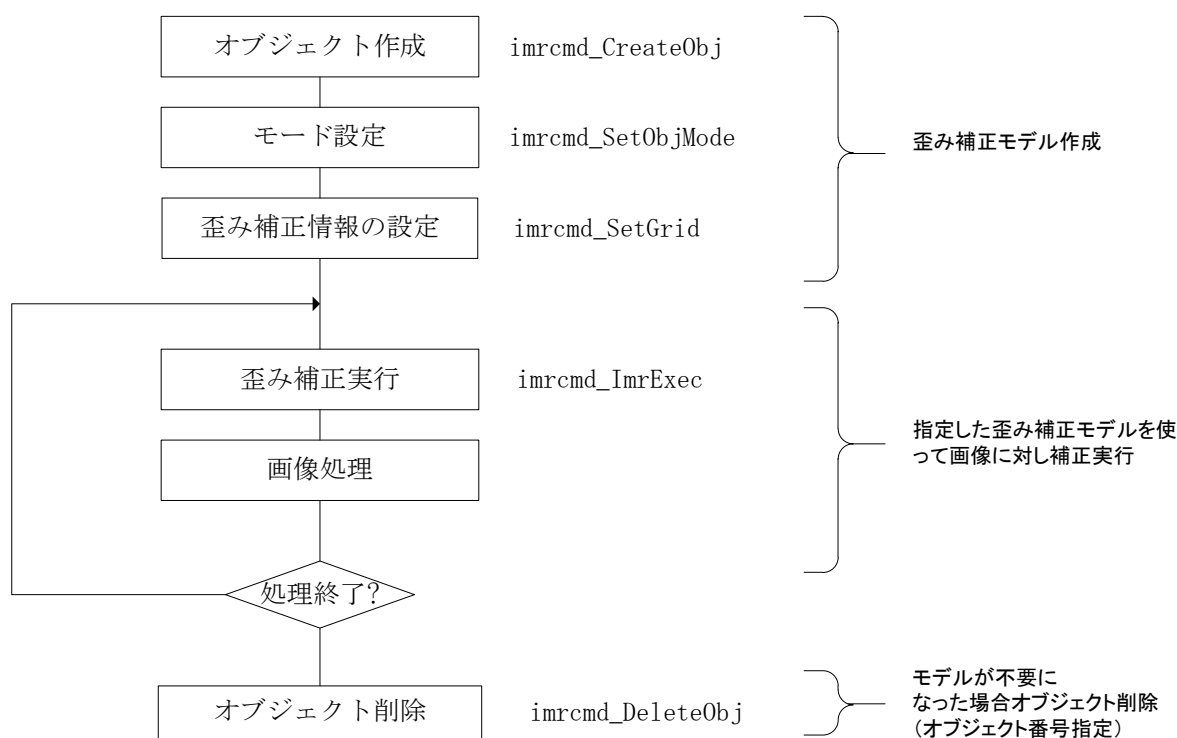
RAWデータを格納したソース画面をRGBデータに変換し、デスティネーション画面に出力します。処理領域はソース画面の (ssx, ssy) を開始座標とし、デスティネーション画面の (dsx, dsy) を開始点とする xlng, ylng の領域に書き込みます。また、処理領域の周辺1画素は0になります。

本コマンドの実行時に『CreateConvertRAWData()』コマンドでRAW/RGB変換データを生成していない場合、およびワーク画面が xlng, ylng より小さい場合、エラー (エラーコード: 138) になります。

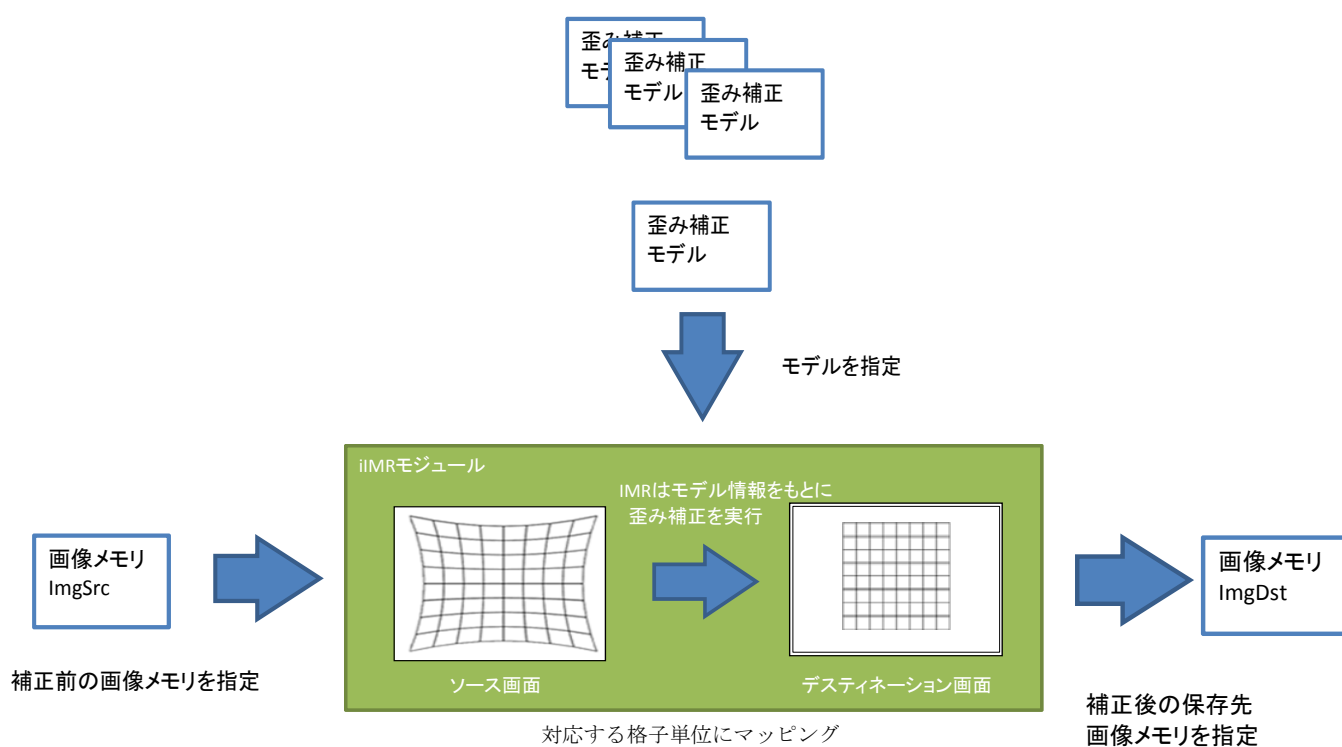


2.34 歪み補正

(1) 歪み補正コマンドの処理フローを示します。



基本フロー



歪み補正処理

(2) エラー処理について

本歪み補正コマンドは、画像認識ライブラリ上で動作します。エラー発生時は、画像処理コマンドのエラー処理と同じく『ReadIPError』『ClearIPError』コマンドを使って適宜処理して下さい。

歪み補正コマンドで発生するエラーコードは、「付録A エラーコード一覧」を参照してください。

2.34.1 歪み補正オブジェクト作成（メモリ確保）（imrcmd_CreateObj）

□ C言語API

```
int imrhdl = imrcmd_CreateObj( enum IMR_OBJTYPE type, IMR_VERTEX *VtxSize );
```

□ パラメータ

enum IMR_OBJTYPE type	オブジェクトタイプ※ IMR_OBJTYPE_IMR (0) IMR_OBJTYPE_IMRLS (1) ※0 固定でご使用下さい。
IMR_VERTEX *VtxSize	頂点データ数を格納したアドレス モデル生成コマンド (ex. 『imrcmd_SetGrid』) で設定する頂点座標 数を与えてください。

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int imrhdl	0: 異常終了, 0以外: 歪み補正オブジェクトハンドル

□ パケットの構造

```
typedef struct {
    long x;                /* X方向頂点数 */
    long y;                /* Y方向頂点数 */
} IMR_VERTEX;
```

□ エラーコード

※ 「IMRエラーコード一覧」を参照して下さい。

□ 機能

本コマンドで歪み補正オブジェクトを生成します。生成したオブジェクトに対し歪み補正モデルを設定しますが必要なオブジェクトサイズはモデル毎異なります。あらかじめ引数VtxSizeに歪み補正モデルの頂点数を指定してください。（頂点数については『imrcmd_SetGrid()』コマンドを参照してください。）

戻り値には歪み補正オブジェクトハンドルが返ります。オブジェクトの設定はこのハンドルを指定します。本コマンドで生成していないオブジェクトハンドル値を指定してIMRコマンドを実行した場合、動作が不定となりますのでご注意ください。

複数のタスクで同じオブジェクトは同時に使用できません。

オブジェクトサイズは、頂点数に応じて以下の通りになります。

頂点数 1024x1024 : 16MB

頂点数 512x512 : 4MB

頂点数 128x128 : 255KB

※小数点以下切り上げ

2.34.2 歪み補正オブジェクト削除（メモリ解放）（imrcmd_DeleteObj）

□ C言語API

```
int ret = imrcmd_DeleteObj( int imrhdl );
```

□ パラメータ

int	imrhdl	歪み補正オブジェクトハンドル

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

※ 「IMRエラーコード一覧」を参照して下さい。

□ 機能

オブジェクトを削除します。
実行中のオブジェクトを指定した場合は、実行終了を待って削除します。

2.34.3 歪み補正モード設定 (imrcmd_SetObjMode)

□ C言語API

```
int ret = imrcmd_SetObjMode( int imrhdl, unsigned long mode, unsigned long opt );
```

□ パラメータ

int	imrhdl	歪み補正オブジェクトハンドル	
unsigned long	mode	オブジェクト動作モード (1) 補完モード IMR_MODE_BFENABLE (0x00000001) バイリニア補完 (2) 描画モード IMR_MODE_DRAWDISABLE (0x00000002) 補正描画無効 (単色描画) ※各モードはOR指定が可能です。 その他のbitは0固定として下さい。	
unsigned long	opt	単色描画 (ライン描画) 色指定	Y画面 (0~255) RGB16画面 (下位16bitにRGB565指定)

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

※ 「IMRエラーコード一覧」を参照して下さい。

□ 機能

指定したオブジェクトのモードを設定します。歪み補正モデルを設定する前に本コマンドを実行してください。

● 補完モード

バイリニア補完を実行することが可能です。デフォルトは補完しません。

● 描画モード

IMR_MODE_DRAWDISABLEを指定した場合補正データをソース画面から参照せずopt指定した画素値で単色描画されます。デフォルトは、ソース画像から補正描画が行われます。

2.34.4 歪み補正モデルの生成（補正情報の設定）（imrcmd_SetGrid）

□ C言語API

```
int ret = imrcmd_SetGrid( int imrhdl, IMR_VERTEX *VtxSize, IMR_VERTEX *SrcVtxList,
                        IMR_VERTEX *DstVtxList, int Dmldgt );
```

□ パラメータ

int	imrhdl	歪み補正オブジェクトハンドル
IMR_VERTEX	*VtxSize	頂点データ数 ($2 \leq x \leq 32767$ 、 $2 \leq y$)
IMR_VERTEX	*SrcVtxList	補正前格子座標リストを示すアドレス ※座標値は整数または固定小数点5桁までの表記が可能です。
IMR_VERTEX	*DstVtxList	補正後格子座標リストを示すアドレス ※補正後の座標値は整数値での表記となります。
int	Dmldgt	SrcVtxListの固定小数点表記を指定して下さい。(0～5) ex. 整数表記の場合 (0)、固定小数点5桁表記の場合 (5)

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ パケットの構造

```
typedef struct {
    long x;                /* X方向頂点数 */
    long y;                /* Y方向頂点数 */
} IMR_VERTEX;
```

□ エラーコード

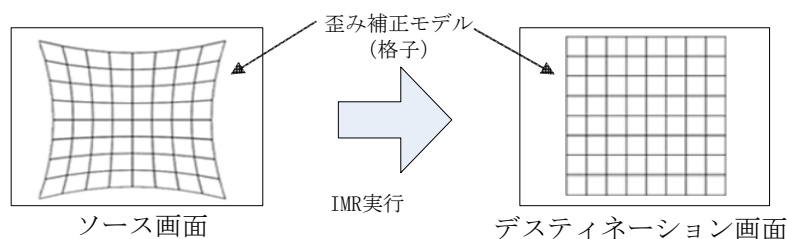
※ 「IMRエラーコード一覧」を参照して下さい。

□ 機能

指定したオブジェクトに歪み補正モデルを生成します。歪み補正はソース・デスティネーション画面それぞれに対応するVtxSizeに示す頂点数(X×Y)からなる格子を指定することで実行されます。この格子を本コマンドで設定します。ソース・デスティネーションそれぞれの格子の頂点座標を一次元配列に配置して、SrcVtxListに、DstVtxListに渡します。

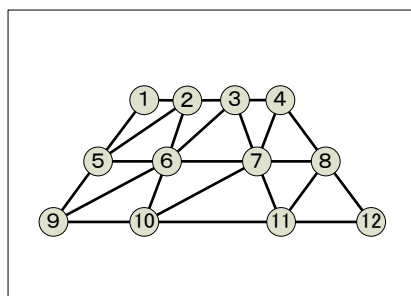
実行中のオブジェクトを指定した場合、実行終了後設定を行います。

頂点数が多いほどオブジェクトサイズが大きくなります。『imrcmd_CreateObj』でオブジェクトを生成する際に必要な頂点数を指定して生成してください。指定した値を超える頂点数を本コマンドで設定した場合オブジェクトサイズが足りずエラーとなります。

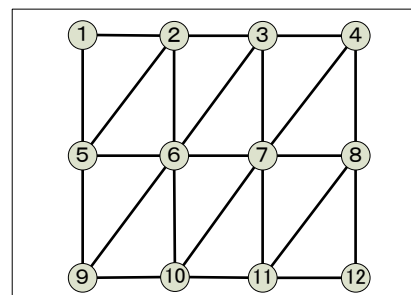


(1) 台形補正の設定例を示します。

頂点数は、 $X(4) \times Y(3)$ とします。ソース画面とデスティネーション画面における格子の頂点 (①～⑫) をリストに登録します。補正処理はこの頂点を結ぶ三角格子単位で実行され、たとえばソース画面の①、②、⑤で囲まれた領域は、デスティネーション画面の①、②、⑤の領域に貼り付けられます。したがって、格子サイズを小さくすればより滑らかに補正されます。(アフィン変換のような補正処理では格子数は4点で十分です。)

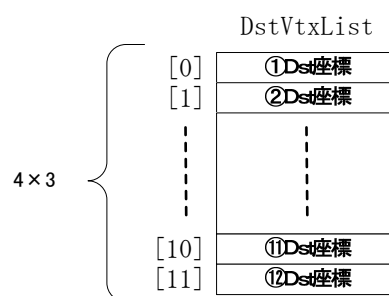
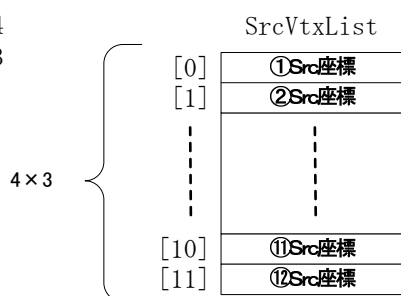


ソース画面
(ウィンドウ領域)



デスティネーション画面
(ウィンドウ領域)

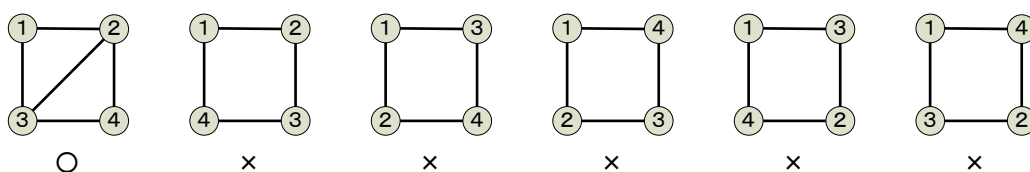
VtxSize.X = 4
VtxSize.Y = 3



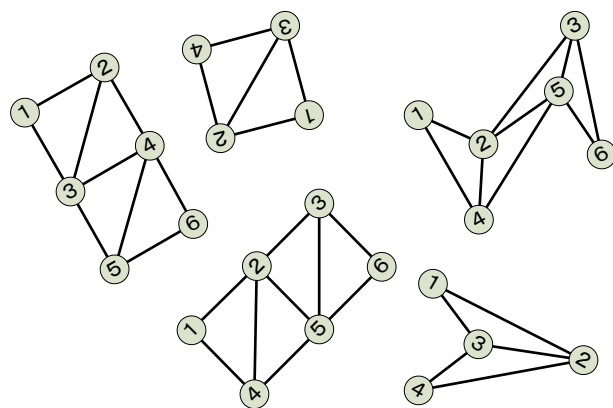
台形補正の設定例

(2) 座標リストへの登録順序について

格子の頂点を座標リストへ登録する順序に注意が必要です。逸脱した場合描画されません。また、格子領域が重なる場合も描画されません。以下に例を示します。



頂点の座標リストへの登録順序



O (すべて可)

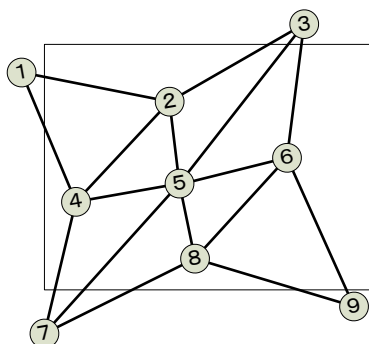
x (格子領域が重なる場合不可)

格子の例

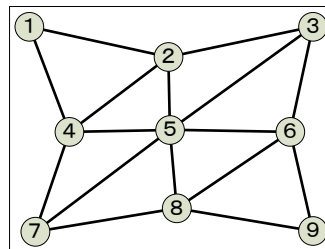
(3) ソース座標について

座標値は、ソースウィンドウ始点からの相対座標で設定してください。必ず使用するソース画面かつソースウィンドウ範囲内の座標 ($0 \leq X, Y \leq 2047$) で設定してください。それを超える座標を指定した場合、画面外の領域が参照され動作が不定となります。(ソースウィンドウおよびソース画面については『imrcmd_ImrExec』を参照ください。)

座標 (補正領域) は、必ずソースウィンドウ内に収まるように設定してください。



ソース画面ウィンドウ
x



ソース画面ウィンドウ
o

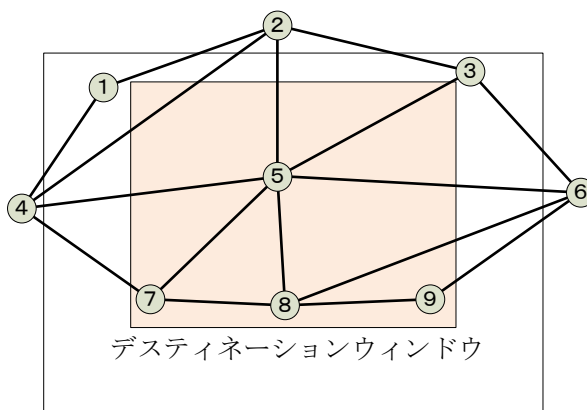
ソース画面と座標指定

(4) デスティネーション座標について

座標値は、デスティネーションウィンドウ始点からの相対座標で設定してください。符号付16bitの範囲で指定してください。それを超えた場合正しく描画できません。但し、描画されるのは $0 \leq X, Y \leq 1024$ の座標範囲となります。また、範囲内であっても、ウィンドウ領域を超えた領域には描画されません。

(※デスティネーションウィンドウおよびデスティネーション画面については『imrcmd_ImrExec』を参照ください。)

デスティネーションウィンドウ内の補正領域にのみ描画されます。



デスティネーションウィンドウ

デスティネーション画面

o

デスティネーション画面と座標指定

(5) 固定小数点表記について

ソース座標は、整数型または、固定小数点1～5桁での設定が可能です。

以下に float型(浮動小数点)から固定小数点データへの変換方法を示します。

unsigned long y : 変換後 固定少数点データ
float x : 変換前 フロートデータ(座標値)
int n : 固定少数点桁数 (0～5)
0の場合は整数(固定小数点なし)データとなります。

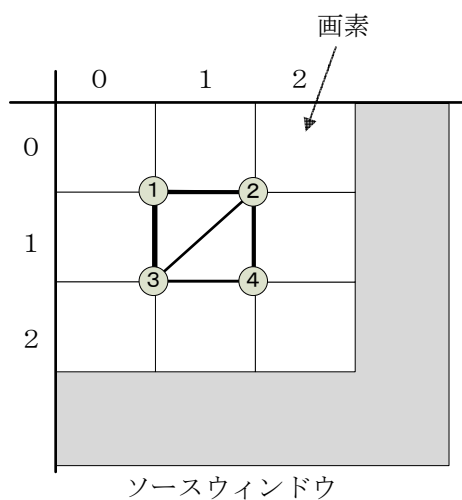
$$y = (\text{unsigned long})((2^n) * x)$$

変換の際に、整数型へのキャストにより小数点以下が切り捨てられますので誤差となります。ご理解の上ご使用ください。

(6) 頂点座標の描画基準について

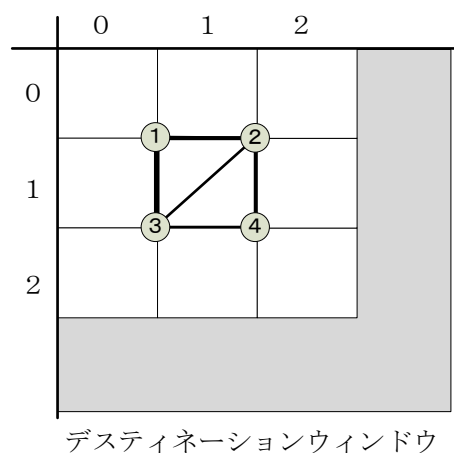
頂点座標は、画素中心ではなく画素の左上が基準となりますのでご注意ください。

以下に例を示します。この場合座標(1, 1)の画素がソース→デスティネーションに描画されます。合1画素)に描画が行われます。



頂点座標

- ① (1, 1)
- ② (2, 1)
- ③ (1, 2)
- ④ (2, 2)



頂点座標

- ① (1, 1)
- ② (2, 1)
- ③ (1, 2)
- ④ (2, 2)

2.34.5 歪み補正実行 (imrcmd_ImrExec)

□ C言語API

```
int ret = imrcmd_ImrExec( enum IMR_SELCH  imrch, int imrhdl, IMGID ImgSrc, IMGID ImgDst,
                          IMR_WINDOW *window, enum IMR_OPEOPT Opt );
```

□ パラメータ

enum IMR_SELCH	imrch	チャンネル IMR_CHO (0) IMR0を使用 IMR_CH1 (1) IMR1を使用
int	imrhdl	歪み補正オブジェクトハンドル
IMGID	ImgSrc	ソース画面 (補正前画像)
IMGID	ImgDst	デスティネーション画面 (補正後画像出力画面)
IMR_WINDOW	*window	IMRウィンドウ設定
enum IMR_OPEOPT	Opt	歪み補正実行オプション ・周辺処理 IMR_SPACE_OCLEAR (0) 周辺を 0 クリア IMR_SPACE_NOTOUCH (1) 周辺は変化させない IMR_SPACE_128CLEAR (2) 周辺を 1 2 8 クリア

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	○※	○	△U	△U
ImgDst	○	△R	○	○	○	○※	◆	-	-

※入出力画面が共にRGB16画面である場合に利用可能です。

※RGB16画面は未サポートです。

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ パケットの構造

```
typedef struct {
    long ssx; /* ソース描画始点 x座標値 ( 0 ≤ ssx ※ ) */
    long ssy; /* ソース描画始点 y座標値 ( 0 ≤ ssy ) */
    long dsx; /* デスティネーション描画始点 x座標値 ( 0 ≤ dsx ※ ) */
    long dsy; /* デスティネーション描画始点 y座標値 ( 0 ≤ dsy ) */
    long dxlng; /* デスティネーション描画幅 (X座標範囲) ( 2 ~ 1025 ) */
    long dylng; /* デスティネーション描画高さ (Y座標範囲) ( 2 ~ 1025 ) */
} IMR_WINDOW;
```

※アライン制限があります。

RGB16画面の場合は 1 6 画素アラインとして下さい。

それ以外の場合は 3 2 画素アラインとして下さい。

□ エラーコード

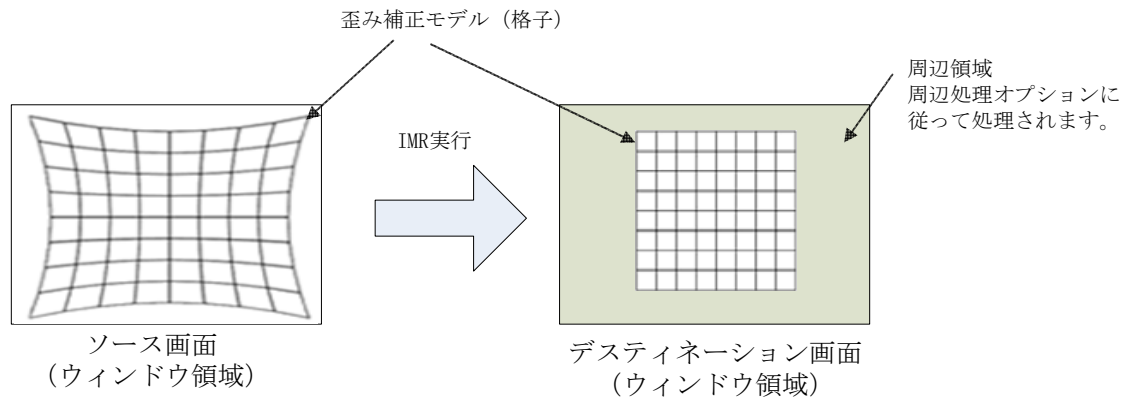
※ 「IMRエラーコード一覧」を参照して下さい。

□ 機能

指定したIMRチャネルを使用し、指定のオブジェクトに従って歪み補正実行します。描画終了後関数が戻ります。IMRがすでに実行中の場合、終了を待って実行します。

オブジェクトにはあらかじめモデル生成コマンド (ex. 『imrcmd_SetGrid()』) でモデルを設定してください。

引数の歪み補正オプションの周辺処理設定によりデスティネーションウィンドウの描画されない領域の処理を指定できます。



本コマンドには以下の制限があります。

●画面サイズ制限

ImgSrc, ImgDstに指定する画像メモリの横方向サイズは、32以上2048画素以下で32画素アラインの画面を使用してください。

●ウィンドウについて

歪み補正コマンドでのウィンドウ設定は、引数のIMR_WINDOW構造体にて設定します。モデル生成コマンド (ex. 『imrcmd_SetGrid』) で指定する座標は、このウィンドウ相対座標となります。

- ・ソースウィンドウは始点のみ設定可能です。ウィンドウサイズは2048 x 2048固定となります。座標がウィンドウサイズ、画面サイズを超える場合でもクリッピングはされませんので、必ず画像メモリの領域内およびウィンドウサイズ内を参照するような座標を設定してください。範囲外を参照した場合不定動作となります。
- ・デスティネーションウィンドウは始点とウィンドウサイズを指定します。ウィンドウ領域が画面サイズより大きい場合クリッピングされます。ウィンドウ領域外となる画素への描画はされません。

●その他ご注意

指定した画像メモリに対し画像処理を実行している場合、事前に『RefreshImg』コマンドを実施してください。※パイプラインモードが無効の場合や、最新画像データがSHのキャッシュ上に存在しない場合は不要です。詳細はNVP-Ax230SDKコマンドリファレンスの「パイプライン制御」の項、『RefreshImg』コマンドを参照ください。

第3章 VP互換コマンド

3.1 正規化関連コマンド

3.1.1 テンプレート特徴量データ領域確保（vpxAllocCorrTemplate）

□ C言語API

```
int ret = vpxAllocCorrTemplate( int Tempsize );
```

□ パラメータ

int	Tempsize	テンプレートサイズ（テンプレートの個数）

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

4 8	動的メモリ確保エラー
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

正規化関連サーチのテンプレート特徴量データ領域の確保を行います。

3.1.2 テンプレート特徴量データ領域解放(vpxFreeCorrTemplate)

☐ C言語API

```
void vpxFreeCorrTemplate();
```

☐ パラメータ

なし

☐ 対応画面タイプ／画面データタイプ

なし

☐ リターンパラメータ

なし

☐ エラーコード

なし

☐ 機能

正規化相関サーチのテンプレート特徴量データ領域の解放を行います。

3.1.3 テンプレート特徴量データ読み出し (vpxReadCorrTemplate)

□ C言語API

```
int ret = vpxReadCorrTemplate( int TmpID , CORRTEMPLATE *CorrTmp );
```

□ パラメータ

int	TmpID	テンプレート番号
CORRTEMPLATE	*CorrTmp	テンプレートデータテーブルを返すアドレス

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)
CORRTEMPLATE	*CorrTmp	テンプレートデータテーブルを格納したアドレス

□ エラーコード

2 1	テンプレートデータテーブルを返すアドレスが不正
8 1	テンプレート番号が有効範囲外
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

正規化関連サーチのテンプレート特徴量データを読み出します。

CORRTEMPLATEのフォーマット及び内容を、下記に示します。

vp900.hで宣言しています

```
typedef struct {
    RECTWINDOW Win;
    short xlen;
    short ylen;
    short v;
    short ImgID;
    short filter;
    short opt;
    short affsrc;
    short afftmp;
    int xmag;
    int ymag;
    float dx;
    float dy;
    unsigned long tn;
    unsigned long sum;
    unsigned long sqsum;
} CORRTEMPLATE;
```

```
typedef struct {
    short sx;
    short sy;
    short ex;
    short ey;
    short xlen;
    short ylen;
} RECTWINDOW;
```

テンプレートデータテーブル	説明
Win	テンプレートのウィンドウ情報
xlen	テンプレートのアフェイン変換前のX方向の長さ
ylen	テンプレートのアフェイン変換前のY方向の長さ
v	テンプレートの登録フラグ
ImgID	テンプレート画面番号
filter	フィルタの種類
opt	オプション
affsrc	ソース画面のアフェイン変換
afftmp	テンプレートのアフェイン変換
xmag	X方向間引き間隔
ymag	Y方向間引き間隔
dx	テンプレート中心X方向オフセット
dy	テンプレート中心Y方向オフセット
tn	テンプレート画素数
sum	テンプレートの濃度総和
sqsum	テンプレートの濃度二乗総和

RECTWINDOW	説明
sx	テンプレートの開始点X座標
sy	テンプレートの開始点Y座標
ex	テンプレートの終了点X座標
ey	テンプレートの終了点Y座標
xlen	テンプレートのX方向長さ
ylen	テンプレートのY方向長さ

3.1.4 テンプレート特徴量データ書き込み（vpxWriteCorrTemplate）

□ C言語API

```
int ret = vpxWriteCorrTemplate( int TmpID , CORRTEMPLATE *CorrTmp );
```

□ パラメータ

int	TmpID	テンプレート番号
CORRTEMPLATE	*CorrTmp	テンプレートデータテーブルを返すアドレス

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

2 1	テンプレートデータテーブルを返すアドレスが不正
8 1	テンプレート番号が有効範囲外
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

正規化相関サーチのテンプレート特徴量データを書き込みます。

CORRTEMPLATEのフォーマット及び内容は、vpxReadCorrTemplate()の詳細情報の欄を参照して下さい。

3.1.5 テンプレートマスクの有効化（vpxEnableCorrMask）

□ C言語API

```
int ret = vpxEnableCorrMask();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）

□ エラーコード

—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

正規化関連サーチのテンプレートパターンをマスクを有効にします。

正規化関連マスクは、テンプレート内で濃度0の画素を処理対象外にします。よって、矩形でない任意の形をテンプレートとしたい場合に有効です。矩形のテンプレート内で必要物体部以外を0で塗りつぶしておき、vpxIP_CorrStep()を実行する前に、vpxEnableCorrMask()を実行すれば、0画素部分は処理されないため、任意形のテンプレートで処理したことに同一となります。

vpxEnableCorrMask() コマンドは、テンプレート登録（vpxSetCorrTemplate()）を行う前に実行して下さい。

3.1.6 テンプレートマスクの無効化（vpxDisableCorrMask）

□ C言語API

```
int ret = vpxDisableCorrMask();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）

□ エラーコード

— エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

正規化関連サーチのテンプレートパターンをマスクを無効にします。

3.1.7 トレーニング（vpxSetCorrTemplate）

□ C言語API

```
int ret = vpxSetCorrTemplate( IMGID ImgTmp , int TmpID , int xmag , int ymag );
```

□ パラメータ

IMGID	ImgTmp	テンプレート画面番号
int	TmpID	テンプレート番号 vpxAllocCorrTemplate() コマンドで確保したテンプレートの範囲で設定して下さい。
int	xmag	X方向の間引き率（1～8）
int	ymag	Y方向の間引き率（1～8）

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgTmp	○	△R	○	○	○	×	○	×	×

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

3	I M P 処理タイムアウト
1 0	指定画面空き領域、または範囲外（不当画面番号エラー）、画面データタイプ不正、画面属性不正
2 2	X方向の間引き率設定値範囲外
2 3	Y方向の間引き率設定値範囲外
3 0	ソース0画面ウィンドウの始点座標が画面外
4 9	テンプレート画面が不定画面
8 0	テンプレートサイズエラー（サイズが256×256より大きい）
8 1	テンプレート番号が有効範囲外
8 3	テンプレート内同一画素エラー（分散値エラー）
8 6	マスク処理でのテンプレート登録エラー（有効面積が0以下）
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

正規化サーチのテンプレートパターンのテンプレート特徴量データの登録を行います。この関数でテンプレートの登録を行った場合は、正規化相関サーチの際にソース画像だけがxmag、ymagで示される間隔で間引きされます。つまり、テンプレートの大きさをxmag、ymagで指定した値で小さくできるということです。この場合、情報量はテンプレートの大きさと同じになります。

テンプレートの位置及びサイズは、ウィンドウ（SRC0_WIN）によって決まります。本コマンド設定前に必ず、ウィンドウ設定コマンドによりSRC0_WINウィンドウを設定して下さい。

また、テンプレートのサイズは4～65,536画素（256×256相当）までとし、テンプレート内が同一画素値の時はエラーとなります。

正規化テンプレート登録では、テンプレート画像の存在する画面番号（ImgTmp）と座標情報のみ登録します。よって、テンプレート登録画面は、解放（FreeImg）したり、画像処理によるデータ変更を行わないで下さい。

3.1.8 トレーニング（ vpxSetCorrTemplateExt ）

□ C言語API

```
int ret = vpxSetCorrTemplateExt( IMGID ImgTmp , int TmpID , int xmag , int ymag );
```

□ パラメータ

IMGID	ImgTmp	テンプレート画面番号
int	TmpID	テンプレート番号 vpxAllocCorrTemplate() コマンドで確保したテンプレートの範囲で設定して下さい。
int	xmag	X方向の間引き率（1～8）
int	ymag	Y方向の間引き率（1～8）

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgTmp	○	△R	○	○	○	×	○	×	×

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

3	I M P 処理タイムアウト
1 0	指定画面空き領域、または範囲外（不当画面番号エラー）、画面データタイプ不正、画面属性不正
2 2	X方向の間引き率設定値範囲外
2 3	Y方向の間引き率設定値範囲外
3 0	ソース 0 画面ウィンドウの始点座標が画面外
4 9	テンプレート画面が不定画面
8 0	テンプレートサイズエラー（サイズが 2 5 6 × 2 5 6 より大きい）
8 1	テンプレート番号が有効範囲外
8 3	テンプレート内同一画素エラー（分散値エラー）
8 6	マスク処理でのテンプレート登録エラー（有効面積が 0 以下）
—	エラーリセットコマンド（ClearIPErrors）未発行

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

正規化サーチのテンプレートパターンのテンプレート特徴量データの登録を行います。この関数でテンプレートの登録を行った場合は、正規化相関サーチの際にソース画像とテンプレート画像の両方がxmag、ymagで示される間隔で間引きされます。つまり、テンプレートの大きさを変更せずに情報量を制限できるということです。

テンプレートの位置及びサイズは、ウィンドウ（SRC0_WIN）によって決まります。本コマンド設定前に必ず、ウィンドウ設定コマンドによりSRC0_WINウィンドウを設定して下さい。

また、テンプレートのサイズは4～65,536画素（256×256相当）までとし、テンプレート内が同一画素値の時はエラーとなります。

正規化テンプレート登録（間引き付き）では、テンプレート画像の存在する画面番号（ImgTmp）と座標情報と間引き率を登録します。よって、テンプレート登録画面は、解放（FreeImg）したり、画像処理によるデータ変更を行わないで下さい。

間引き率を2とすると、テンプレート内データは1画素おきで処理されるため、データ量は半分にになります。間引きテンプレートは、処理が高速ですが、精度がフルテンプレートより落ちます。処理対象物体によって、フルテンプレート／間引きテンプレートどちらが有効か、また、間引きテンプレートによる間引き率は異なるため、実際に評価を行い、使用して下さい。

精度

フルテンプレート > 間引きテンプレート

処理速度

フルテンプレート < 間引きテンプレート

3.1.9 ラスターサーチ (vpxIP_CorrStep)

□ C言語API

```
int ret = vpxIP_CorrStep( IMGID ImgSrc , int TmpID , CORR_TBL *tbl , int cNum ,
                        int StepX , int StepY );
```

□ パラメータ

IMGID	ImgSrc	サーチ画面番号
int	TmpID	テンプレート番号
CORR_TBL	*tbl	正規化相関サーチ結果格納テーブルを返すアドレス
int	cNum	サーチ個数指定
int	StepX	X方向サーチ間隔
int	StepY	Y方向サーチ間隔

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	×	×

□ リターンパラメータ

int	ret	≥ 0 : 正常終了、サーチ個数 -1 : 異常終了
CORR_TBL	*tbl	正規化相関サーチ結果格納テーブルを格納したアドレス

□ エラーコード

3	IMP処理タイムアウト
1 0	指定画面空き領域、または範囲外（不当画面番号エラー）、画面データタイプ不正、画面属性不正
2 2	正規化相関サーチ結果格納テーブルを返すアドレスが不正
3 0	ソース 0 画面ウィンドウの始点座標が画面外
4 5	設定条件の範囲エラー（サーチ間隔および間引き率が正規化相関処理モードに対応しない）
4 8	動的メモリ確保エラー
4 9	サーチ画面が不定画面
8 0	テンプレートサイズがサーチ領域を超える
8 7	テンプレート未設定エラー
8 8	テンプレート登録画面番号エラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

矩形領域の正規化相関サーチを行います。
サーチで得られるスコア (score) は、実際の相関値 (r) の 2 乗を 1 0 0 0 0 倍した値です。

$$\text{score} = r * r * 10000$$

SRC0_WINで指定された矩形領域をStepX、StepYで指定された間隔でラスタースーチします。
そしてcNumで指定された個数の正規化相関データを相関係数の大きい順にソートしてtblに格納します。
この関数で得られるサーチ結果の座標は、テンプレートの左上のポイントです。
正規化相関サーチ結果格納テーブルのフォーマットを、下記に示します。

本テーブルはvp900.hで宣言しています。

```
typedef struct {  
    long    score;    // 正規化相関値  
    long    x;        // 画面原点相対X座標  
    long    y;        // 画面原点相対Y座標  
} CORRTBL;
```

3.1.10 近傍サーチ (vpxIP_CorrPoint)

□ C言語API

```
int ret = vpxIP_CorrPoint( IMGID ImgSrc , int TmpID , CORRTBL *inTbl , int inNum ,
                          CORRTBL *outTbl , int outNum , int LengX , int LengY , int StepX , int StepY );
```

□ パラメータ

IMGID	ImgSrc	サーチ画面番号
int	TmpID	テンプレート番号
CORRTBL	*inTbl	入力処理ポイントテーブルを返すアドレス
int	inNum	入力処理個数
CORRTBL	*outTbl	正規化相関サーチ結果格納テーブルを返すアドレス
int	outNum	サーチ個数指定
int	LengX	サーチするX方向の幅
int	LengY	サーチするY方向の幅
int	StepX	X方向サーチ間隔
int	StepY	Y方向サーチ間隔

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	×	×

□ リターンパラメータ

int	ret	≥ 0 : 正常終了、サーチ個数 -1 : 異常終了
CORRTBL	*outTbl	正規化相関サーチ結果格納テーブルを格納したアドレス

□ エラーコード

3	IMP処理タイムアウト
1 0	指定画面空き領域、または範囲外（不当画面番号エラー）、画面データタイプ不正、画面属性不正
2 2	入力処理ポイントテーブルを返すアドレスが不正
2 4	正規化相関サーチ結果格納テーブルを返すアドレスが不正
3 0	ソース0画面ウィンドウの始点座標が画面外
4 5	設定条件の範囲エラー（サーチ間隔および間引き率が正規化相関処理モードに対応しない）
4 9	サーチ画面が不定画面
8 0	テンプレートサイズがサーチ領域を超える
8 2	サーチ領域が画面をはみ出す（有効なサーチ点がない）
8 7	テンプレート未設定エラー
8 8	テンプレート登録画面番号エラー
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

近傍領域の正規化相関サーチを行います。
サーチで得られるスコア (score) は、実際の相関値 (r) の 2 乗を 1 0 0 0 0 倍した値です。

$$\text{score} = r * r * 10000$$

入力処理ポイントテーブル inTbl に入力された座標から、StepX、StepY で指定した間隔で LengX、LengY で指定した幅の正規化相関サーチを inNum 回数行い、outNum で指定された個数の正規化相関データを相関係数の大きい順にソートして outTbl に格納します。

この関数で得られるサーチ結果の座標は、テンプレートの左上のポイントです。
CORRTBL のフォーマットは、vpxIP_CorrStep() の詳細情報の欄を参照して下さい。

3.1.11 サブピクセルサーチ (vpxIP_CorrPrecise)

□ C言語API

```
int ret = vpxIP_CorrPrecise( IMGID ImgSrc , int TmpID , CORRTBL *inTbl ,FCORRTBL *outTbl );
```

□ パラメータ

IMGID	ImgSrc	サーチ画面番号
int	TmpID	テンプレート番号
CORRTBL	*inTbl	入力処理ポイントテーブルを返すアドレス
FCORRTBL	*outTbl	サブピクセル計算結果テーブルを返すアドレス

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgSrc	○	△R	○	○	○	×	○	×	×

□ リターンパラメータ

int	ret	1 : 正常終了 0 : サーチ個数が0 -1 : 異常終了
FCORRTBL	*outTbl	サブピクセル計算結果テーブルを格納したアドレス

□ エラーコード

1 0	指定画面空き領域、または範囲外（不当画面番号エラー）、画面データタイプ不正、画面属性不正
2 2	入力処理ポイントテーブルを返すアドレスが不正
2 3	サブピクセル計算結果テーブルを返すアドレスが不正
3 0	ソース0画面ウィンドウの始点座標が画面外
4 9	サーチ画面が不定画面
8 0	テンプレートサイズがサーチ領域を超える
8 2	サーチ領域が画面をはみ出す（有効なサーチ点がない）
8 7	テンプレート未設定エラー
8 8	テンプレート登録画面番号エラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

正規化相関サーチから得られたポイントから近傍領域で正規化相関サーチを行い、その相関値からサブピクセルの位置計算を行います。

サーチで得られるスコア (score) は、実際の相関値 (r) の2乗を10000倍した値です。

$$\text{score} = r * r * 10000$$

CORRTBLのフォーマットは、vpxIP_CorrStep()の詳細情報の欄を参照して下さい。
FCORRTBLはvp900.hで宣言しています。

```
typedef struct {
    long    score;    // 正規化相関値
    float   x;        // 画面原点相対X座標
    float   y;        // 画面原点相対Y座標
} FCORRTBL;
```


3.1.12 相関演算途中打ち切りの有効化（vpxEnableCorrBreak）

☐ C言語API

```
int ret = vpxEnableCorrBreak();
```

☐ パラメータ

なし

☐ 対応画面タイプ／画面データタイプ

なし

☐ リターンパラメータ

int	ret	正常終了（0）

☐ エラーコード

—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

☐ 機能

正規化相関サーチの相関演算のブレイク機能（相関演算途中打ち切り）を許可します。相関演算の値とテンプレートデータの値の差がvpxSetCorrBreakThr() コマンドで設定した値よりも大きい場合相関演算を途中で打ち切ります。

なお、この機能を使用すると、明るさの変動に影響されやすくなりますので注意して下さい。

3.1.13 相関演算途中打ち切りの無効化（vpxDisableCorrBreak）

□ C言語API

```
int ret = vpxDisableCorrBreak();
```

□ パラメータ

なし

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）

□ エラーコード

—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

vpxEnableCorrBreak() コマンドで有効化した正規化相関サーチの相関演算のブレイク機能を無効にします。

3.1.14 相関演算途中打ち切り閾値設定（vpxSetCorrBreakThr）

□ C言語API

```
int ret = vpxSetCorrBreakThr( int thr , int opt );
```

□ パラメータ

int thr 相関演算途中打ち切りを行うための差分累積値のしきい値。
 ここでは、1画素のレベル0～255の値を設定して下さい。
 ここで指定されたthrはテンプレートの画素数換算されたものが実際の
 差分累積しきい値になります。

 差分累積しきい値 = t h r × （テンプレートの画素数）

int opt オプション。処理打ち切り方法

オプション	内容
0	常にthrで指定したしきい値で演算をうち切る
1	残差逐次検定演算（S S D A）法により演算をうち切る

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了（0）

□ エラーコード

— エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

正規化相関値サーチの相関演算のブレイク機能での相関演算途中打ち切りを行うための差分累積値のしきい値を設定します。相関演算途中の総和値とテンプレートデータの総和値の差がt h rよりも大きい場合、ハード的に相関演算を途中で打ち切り、サーチを高速化します。
なお、この機能を使用すると、明るさの変動に影響されやすくなりますので注意して下さい。

しきい値（t h r）が小さいほど高速になりますが、明るさの変動に影響されやすくなります。
現実的なしきい値は、64程度かそれ以上と考えて下さい。
なお、t h rに「0」を指定すると最大値が設定されます。

3.1.15 サーチ除外領域の設定（vpxSetSearchDistance）

□ C言語API

```
int ret = vpxSetSearchDistance( int mode , int xleng , int yleng );
```

□ パラメータ

int mode サーチモード

サーチモード	内容
0	ノーマルモード。サーチ除外領域モード解除。
1	サーチ除外領域モード。xleng, ylengが有効になります。

int xleng 除外領域X間隔。間隔の1ステップは、正規化相関サーチのパラメータのX方向サーチ間隔になります。

int yleng 除外領域Y間隔。間隔の1ステップは、正規化相関サーチのパラメータのY方向サーチ間隔になります。

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了（0）

□ エラーコード

— エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

正規化相関サーチのサーチ領域の除外領域を設定します。正規化相関サーチのパラメータのサーチ個数が2以上の場合、抽出座標の間隔を設定し、近傍座標の抽出を防止します。
尚、除外領域X間隔、除外領域Y間隔の1ステップは、正規化相関サーチのパラメータのX方向サーチ間隔、Y方向サーチ間隔になります。

3.1.16 相関サーチモード設定（vpxSetCorrMode）

□ C言語API

```
int ret = vpxSetCorrMode( enum IPCorrMode mode );
```

□ パラメータ

enum IPCorrMode mode 正規化相関処理モード

正規化相関サーチモード	対応数	説明
CORR_NORMAL	0	ノーマルモード

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了（0）

□ エラーコード

— エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

正規化相関サーチでの処理モードの設定を行います。

3.1.17 サブピクセル算出桁数設定 (vpxSetCorrPrecise)

□ C言語API

```
int ret = vpxSetCorrPrecise( enum IPCorrPrecise precise );
```

□ パラメータ

enum IPCorrPrecise precise 正規化相関小数点出力桁数

正規化相関小数点出力桁数	対応数	説明
CORR_PRECISE_1	0	小数点 1 桁
CORR_PRECISE_2	1	小数点 2 桁
CORR_PRECISE_3	2	小数点 3 桁
CORR_PRECISE_4	3	小数点 4 桁

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了 (0)

□ エラーコード

— エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

vpxIP_CorrPrecise() コマンドで出力される X, Y 座標の小数点の桁数を指定します。

3.2 直線抽出 (VP-910互換)

3.2.1 離散データ座標からの直線抽出 (GetHoughLine)

□ C言語API

```
int ret = GetHoughLine( POINTTBL *PointTbl , LINERANGE *AngleRange , HOUGHLINE *HoughLine );
```

□ パラメータ

POINTTBL *PointTbl X, Y座標配列 (入力)

LINERANGE *AngleRange 検索角度範囲 (入力)

HOUGHLINE *HoughLine ハフ変換直線算出 (出力)

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	≥ 0 : 抽出点列数 - 1 : 異常終了
HOUGHLINE	*HoughLine	ハフ変換直線算出結果を格納したアドレス

□ エラーコード

2 0、2 1、2 2	パラメータエラー
4 8	メモリ確保エラー
2 0 1	直線のポイント数不足による直線抽出エラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

PointTblに格納された点列をLINERANGEで設定された角度の範囲でハフ変換を行い、ハフ変換直線をサブピクセルの単位まで求めます。

POINTTBL、LINERANGE、HOUGHLINEのフォーマットを、下記に示します。
vp900.hで宣言しています。

```
typedef struct {
    short    num;
    short    opt;
    WPOINT   point[ _POINTTBL_DATA_NUM ];
} POINTTBL;
```

```
typedef struct {
    short    x; // X座標
    short    y; // Y座標
} WPOINT;
```

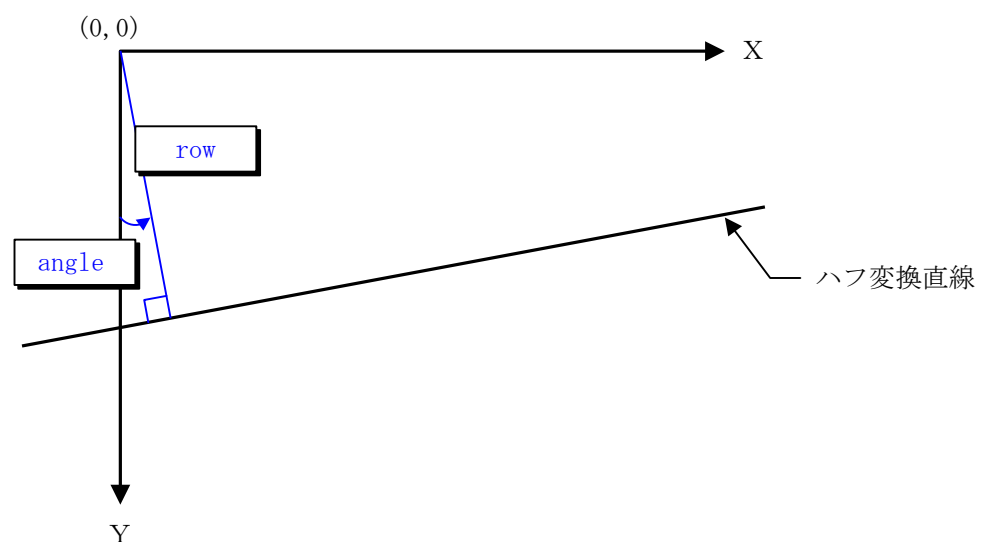
POINTTBL	説明
num	有効データ数
opt	未使用
point[_POINTTBL_DATA_NUM]	X, Y座標配列

```
typedef struct {
    short    lower;
    short    upper;
} LINERANGE;
```

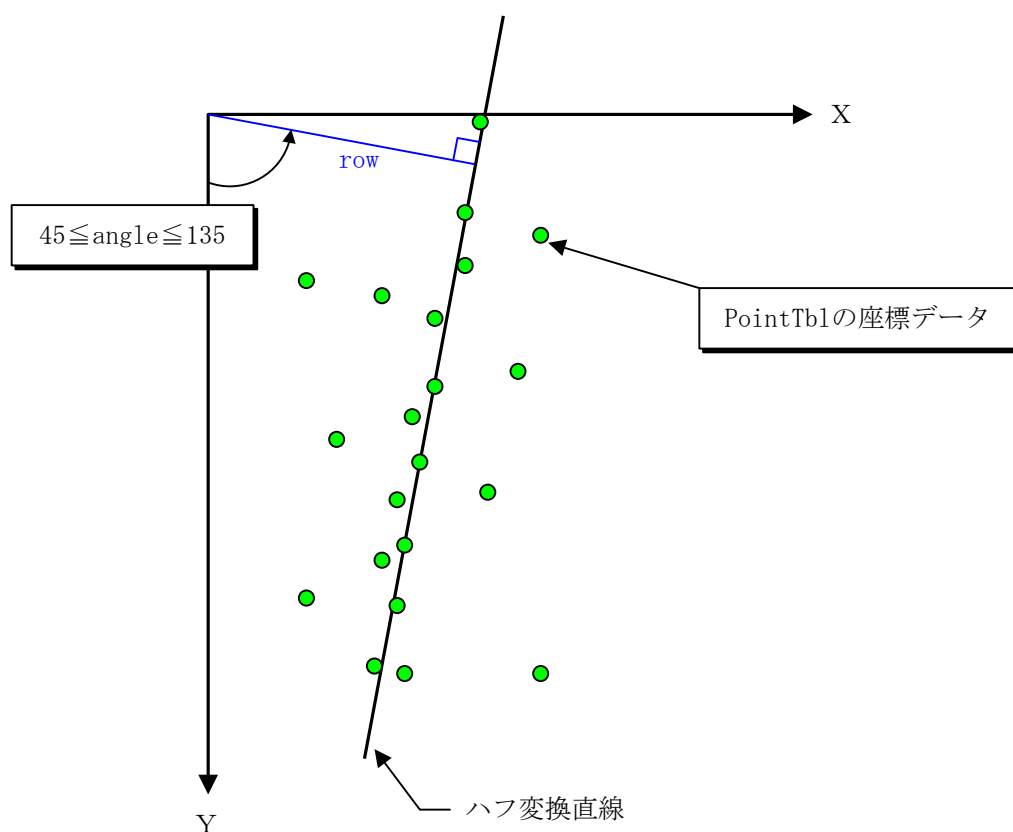
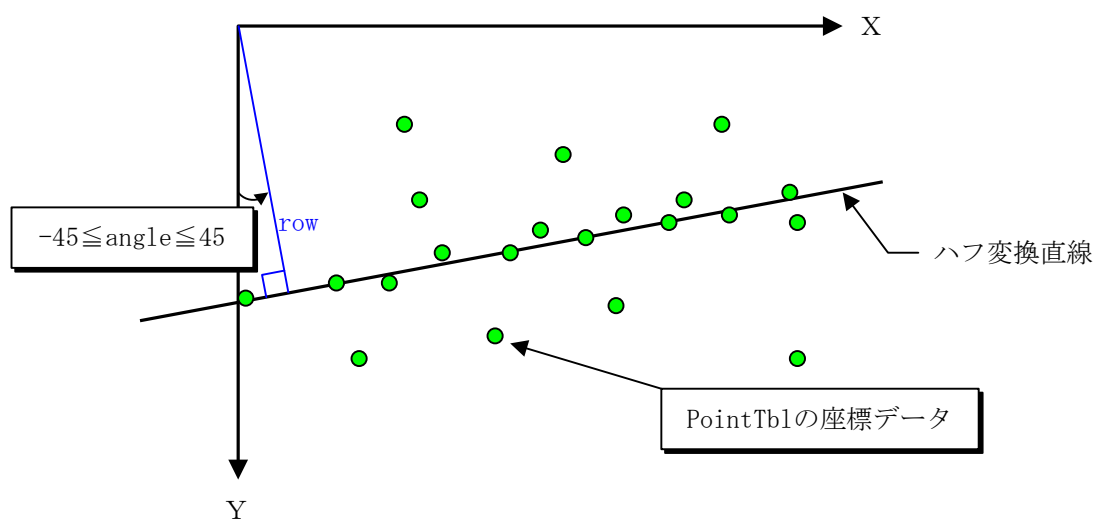
LINERANGE	説明
lower	検索範囲 下限
upper	検索範囲 上限

```
typedef struct {
    float    row;
    float    angle;
} HOUGHLINE;
```

HOUGHLINE	説明
row	ハフ変換直線 row
angle	ハフ変換直線 angle



本コマンドは、常に変換抽出数量が最大となる直線を抽出します。



ハフ変換のコマンドは、高速化のため、関数内部でPointTblに格納されたデータのバウンダリチェックを行いません。そのため、ユーザ側でバウンダリチェック及び領域の絞り込みを行う必要があります。

以下にその例を示しました。例では、領域X座標抽出(IP_ProjectB0RegionX())、領域Y座標抽出(IP_ProjectB0RegionY())での抽出座標のハフ変換を行っています。リストで「sx, sy, ex, ey」は、SetWindow()で指定した処理ウィンドウ領域です。またポイント座標(point.x, point.y)として指定できる範囲は0～511です。この範囲を超えるとシステム例外が発生する可能性がありますので注意して下さい。

```
int sx; /* 処理領域開始X座標 */
int sy; /* 処理領域開始Y座標 */
int ex; /* 処理領域終了X座標 */
int ey; /* 処理領域終了Y座標 */
POINT_TBL RegionYMin, RegionYMax, RegionXMin, RegionXMax;
POINT_TBL *RegionTbl;
int *LineTbl;
int TblMin[512], TblMax[512];
int TblX[512], TblY[512];
int lower, upper;
int i, j, x, y, n, val;

/** 領域YのMax, Min抽出 **/
IP_ProjectB0RegionY( BinID, TblMin, TblMax);
LineTbl= TblMin; RegionTbl= &RegionYMin;
lower= sy; upper= ey;
for(n=0, i=0; i<ex-sx+1; i++){
    val= LineTbl[i]+ sy;
    if(lower < val && val < upper){
        RegionTbl->point[n].x= i+ sx;
        RegionTbl->point[n].y= val;
        n++;
    }
}
RegionTbl->num= n;
LineTbl= TblMax; RegionTbl= &RegionYMax;
lower= sy; upper= ey;
for(n=0, i=0; i<ex-sx+1; i++){
    val= LineTbl[i]+ sy;
    if(lower < val && val < upper){
        RegionTbl->point[n].x= i+ sx;
        RegionTbl->point[n].y= val;
        n++;
    }
}
RegionTbl->num= n;
/** 領域XのMax, Min抽出 **/
IP_ProjectB0RegionX( BinID, TblMin, TblMax);
LineTbl= TblMin; RegionTbl= &RegionXMin;
lower= sx; upper= ex;
for(n=0, i=0; i<ey-sy+1; i++){
    val= LineTbl[i]+ sx;
    if(lower < val && val < upper){
        RegionTbl->point[n].x= val;
        RegionTbl->point[n].y= i+ sy;
        n++;
    }
}
RegionTbl->num= n;
LineTbl= TblMax; RegionTbl= &RegionXMax;
lower= sx; upper= ex;
for(n=0, i=0; i<ey-sy+1; i++){
    val= LineTbl[i]+ sx;
    if(lower < val && val < upper){
        RegionTbl->point[n].x= val;
        RegionTbl->point[n].y= i+ sy;
        n++;
    }
}
RegionTbl->num= n;
```

3.2.2 離散データ座標からの2直線抽出（GetHoughLineRow）

□ C言語API

```
int ret = GetHoughLineRow( POINTTBL *PointTbl , LINERANGE *AngleRange ,
                          HOUGHLINE *HoughLine , int mode );
```

□ パラメータ

POINTTBL *PointTbl X, Y座標配列（入力）

LINERANGE *AngleRange 検索角度範囲（入力）

HOUGHLINE *HoughLine ハフ変換直線算出（出力）

int mode ライン検出モード（入力）

検出モード	対応数	内容
MIN_LINE	0	row の最小値を検出
MAX_LINE	1	row の最大値を検出

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret ≥ 0 : 抽出点列数
 -1 : 異常終了
 HOUGHLINE *HoughLine ハフ変換直線算出結果を格納したアドレス

□ エラーコード

20～23 パラメータエラー
 48 メモリ確保エラー
 201 直線のポイント数不足による直線抽出エラー
 - エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

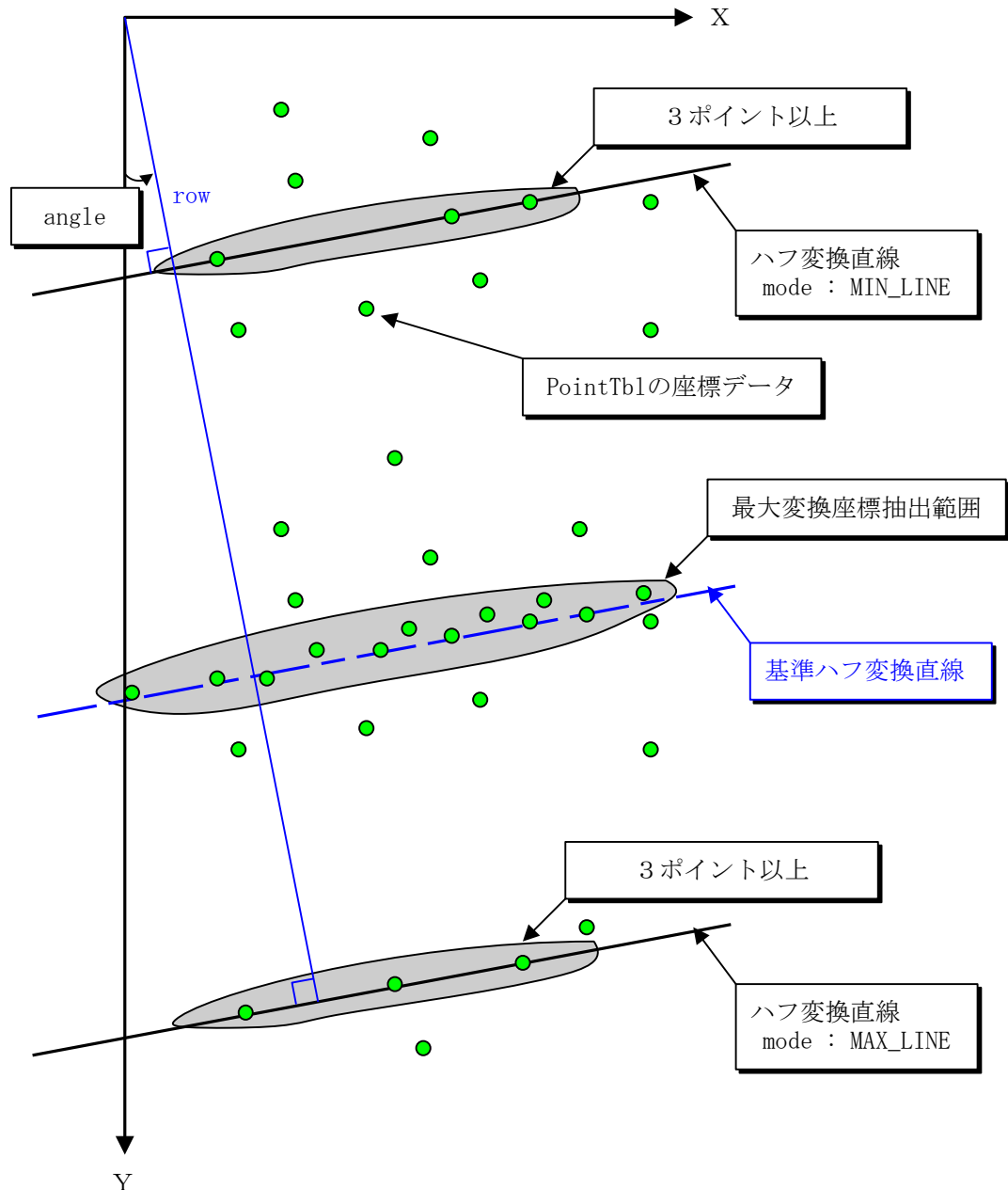
□ 機能

PointTblに格納された点列をLINERANGEで設定された角度の範囲でハフ変換を行い、同一角度の2直線を求め、modeにより、2直線のrowのうち最小値又は最大値を検出し、ハフ変換直線をサブピクセルの単位まで求めます。

なお、本コマンドを使用する場合のバウンダリチェック及び領域の絞り込みについては、GetHoughLine() コマンドの詳細情報を参照して下さい。

POINTTBL、LINERANGE、HOUGHLINEのフォーマットは、GetHoughLine()の詳細情報の欄を参照して下さい。

本コマンドは、まず変換抽出数量が最大となる直線を抽出し、その抽出した直線を基準ハフ変換直線にします。そして、基準ハフ変換直線の角度で変換抽出数のが3ポイント以上の直線を検索します。



3.2.3 離散データ座標からの2直線抽出(拡張) (GetHoughLineRowExt)

□ C言語API

```
int ret = GetHoughLineRowExt( POINTTBL *PointTbl , LINERANGE *AngleRange ,
                             HOUGHLINE *HoughLine , int thr , int mode );
```

□ パラメータ

POINTTBL *PointTbl X, Y座標配列 (入力)

LINERANGE *AngleRange 検索角度範囲 (入力)

HOUGHLINE *HoughLine ハフ変換直線算出 (出力)

int thr 変換抽出数量しきい値 (入力)

int mode ライン検出モード (入力)

検出モード	対応数	内容
MIN_LINE	0	row の最小値を検出
MAX_LINE	1	row の最大値を検出

□ 対応画面タイプ/画面データタイプ

なし

□ リターンパラメータ

int ret ≥ 0 : サーチ個数
 -1 : 異常終了
 HOUGHLINE *HoughLine ハフ変換直線算出結果を格納したアドレス

□ エラーコード

20 ~ 22, 24 パラメータエラー
 48 メモリ確保エラー
 201 直線のポイント数不足による直線抽出エラー
 - エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

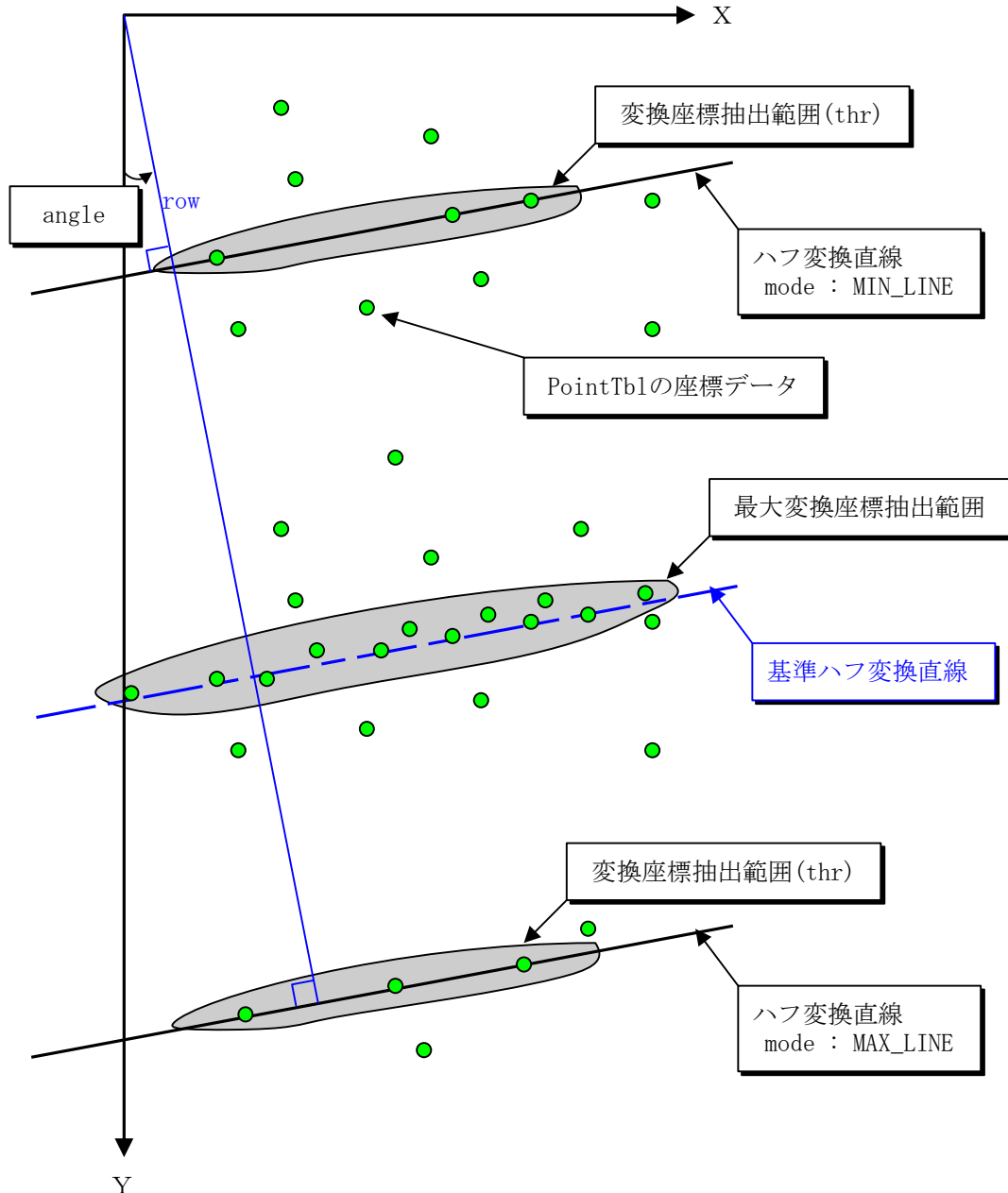
□ 機能

PointTblに格納された点列をLINERANGEで設定された角度の範囲でハフ変換を行い、thrで設定された変換抽出数量しきい値以上の同一角度の2直線を求め、modeにより、2直線のrowのうち最小値又は最大値を検出し、ハフ変換直線をサブピクセルの単位まで求めます。

なお、本コマンドを使用する場合のバウンダリチェック及び領域の絞り込みについては、GetHoughLine() コマンドの詳細情報を参照して下さい。

POINTTBL、LINERANGE、HOUGHLINEのフォーマットは、GetHoughLineの詳細情報の欄を参照して下さい。

本コマンドは、まず変換抽出数量が最大となる直線を抽出し、その抽出した直線を基準ハフ変換直線にします。そして、基準ハフ変換直線の角度で変換抽出数のしきい値 (thr) 以上の直線を検索します。



3.2.4 離散データ座標からの直線抽出(角度固定) (GetSideHoughLine)

□ C言語API

```
int ret = GetSideHoughLine( POINTTBL *PointTbl , float angle ,
                           HOUGHLINE *HoughLine , int thr , int mode );
```

□ パラメータ

POINTTBL	*PointTbl	X, Y座標配列 (入力)
float	angle	検索角度
HOUGHLINE	*HoughLine	ハフ変換直線算出 (出力)
int	thr	しきい値
int	mode	ライン検出モード (入力)

検出モード	対応数	内容
MIN_LINE	0	row の最小値を検出
MAX_LINE	1	row の最大値を検出

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	≥ 0 : 抽出点列数 -1 : 異常終了
HOUGHLINE	*HoughLine	ハフ変換直線算出結果を格納したアドレス

□ エラーコード

20、22、24	パラメータエラー
48	メモリ確保エラー
201	直線のポイント数不足による直線抽出エラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

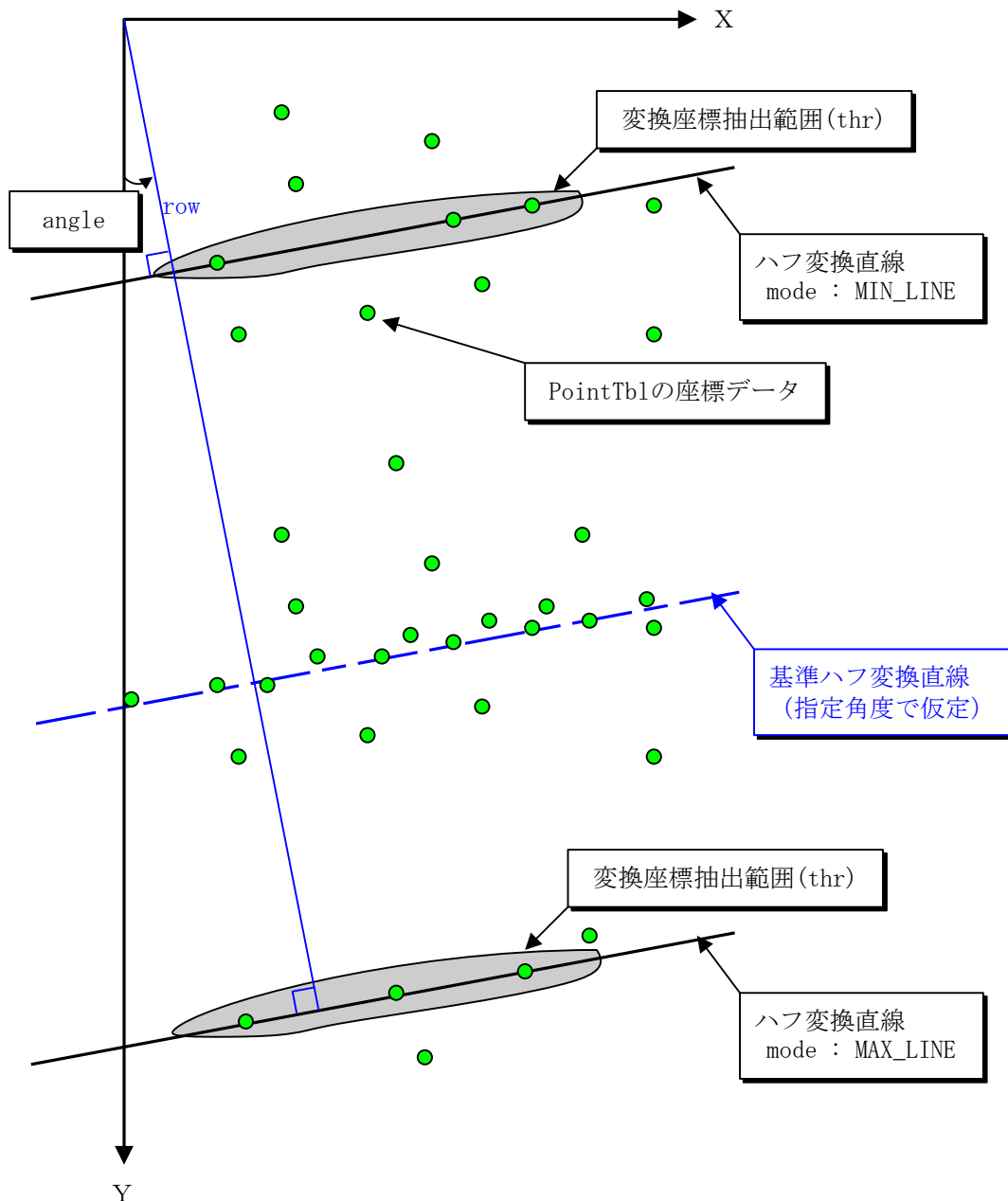
PointTblに格納された点列をangleで設定された角度でハフ変換を行い、thrで設定された変換抽出数量しきい値以上のポイントの直線を求めます。

また、modeにより直線のrowの最小値または、最大値を検出します。

なお、本コマンドを使用する場合のバウンダリチェック及び領域の絞り込みについては、GetHoughLine() コマンドの詳細情報を参照して下さい。

POINTTBL、HOUGHLINEのフォーマットは、GetHoughLineの詳細情報の欄を参照して下さい。

本コマンドは、angleで指定された角度で直線を基準ハフ変換直線を仮定します。そして、基準ハフ変換直線の角度で変換抽出数のしきい値（thr）以上の直線を検索します。



3.2.5 ハフ変換直線の交点座標算出 (GetCrossPoint)

□ C言語API

```
int ret = GetCrossPoint( HOUGHLINE *xline , HOUGHLINE *yline , FPOINT *point );
```

□ パラメータ

HOUGHLINE	*xline	$-45^{\circ} \leq \text{angle} \leq 45^{\circ}$ までのハフ変換直線
HOUGHLINE	*yline	$45^{\circ} \leq \text{angle} \leq 135^{\circ}$ までのハフ変換直線
FPOINT	*point	直線の交点座標

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)
FPOINT	*point	直線の交点座標を格納したアドレス

□ エラーコード

20, 21, 22	パラメータエラー)
201	0除算エラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

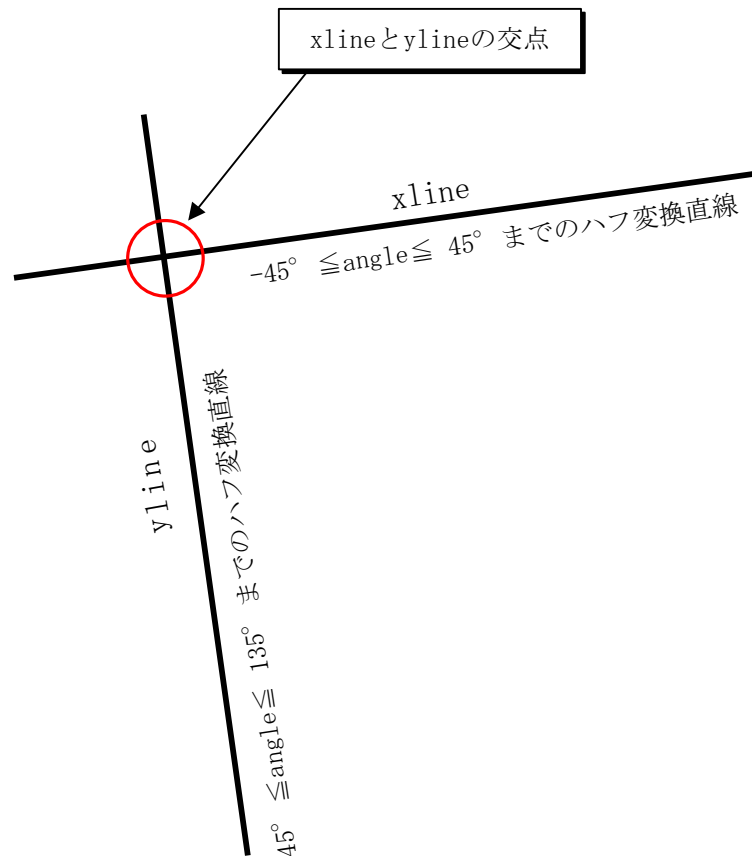
□ 機能

ハフ変換2直線の交点座標の算出を行います。

HOUGHLINEのフォーマットは、GetHoughLineの詳細情報の欄を参照して下さい。
FPOINTのフォーマットを下記に示します。

vp900.hで宣言しています。

```
typedef struct {
    float    x;  // X座標
    float    y;  // Y座標
} FPOINT;
```



3.2.6 ハフ変換直線の矩形交点座標の算出 (GetRectPoint)

□ C言語API

```
int ret = GetRectPoint( HOUGHLINE *block0 , HOUGHLINE *block1 ,
                      HOUGHLINE *block2 , HOUGHLINE *block3 , FRECTPOINT *RectPoint );
```

□ パラメータ

HOUGHLINE *block0 $-45^{\circ} \leq \text{angle} \leq 45^{\circ}$ までのブロック 0 のハフ変換直線

HOUGHLINE *block1 $-45^{\circ} \leq \text{angle} \leq 45^{\circ}$ までのブロック 1 のハフ変換直線

HOUGHLINE *block2 $45^{\circ} \leq \text{angle} \leq 135^{\circ}$ までのブロック 2 のハフ変換直線

HOUGHLINE *block3 $45^{\circ} \leq \text{angle} \leq 135^{\circ}$ までのブロック 3 のハフ変換直線

FRECTPOINT *RectPoint 直線の交点座標

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret	正常終了 (0) または異常終了 (- 1)
FRECTPOINT *RectPoint	直線の交点座標を格納したアドレス

□ エラーコード

2 0 ~ 2 4	パラメータエラー
2 0 1	0 除算エラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ブロック毎のハフ変換4直線から矩形交点座標の算出を行います。

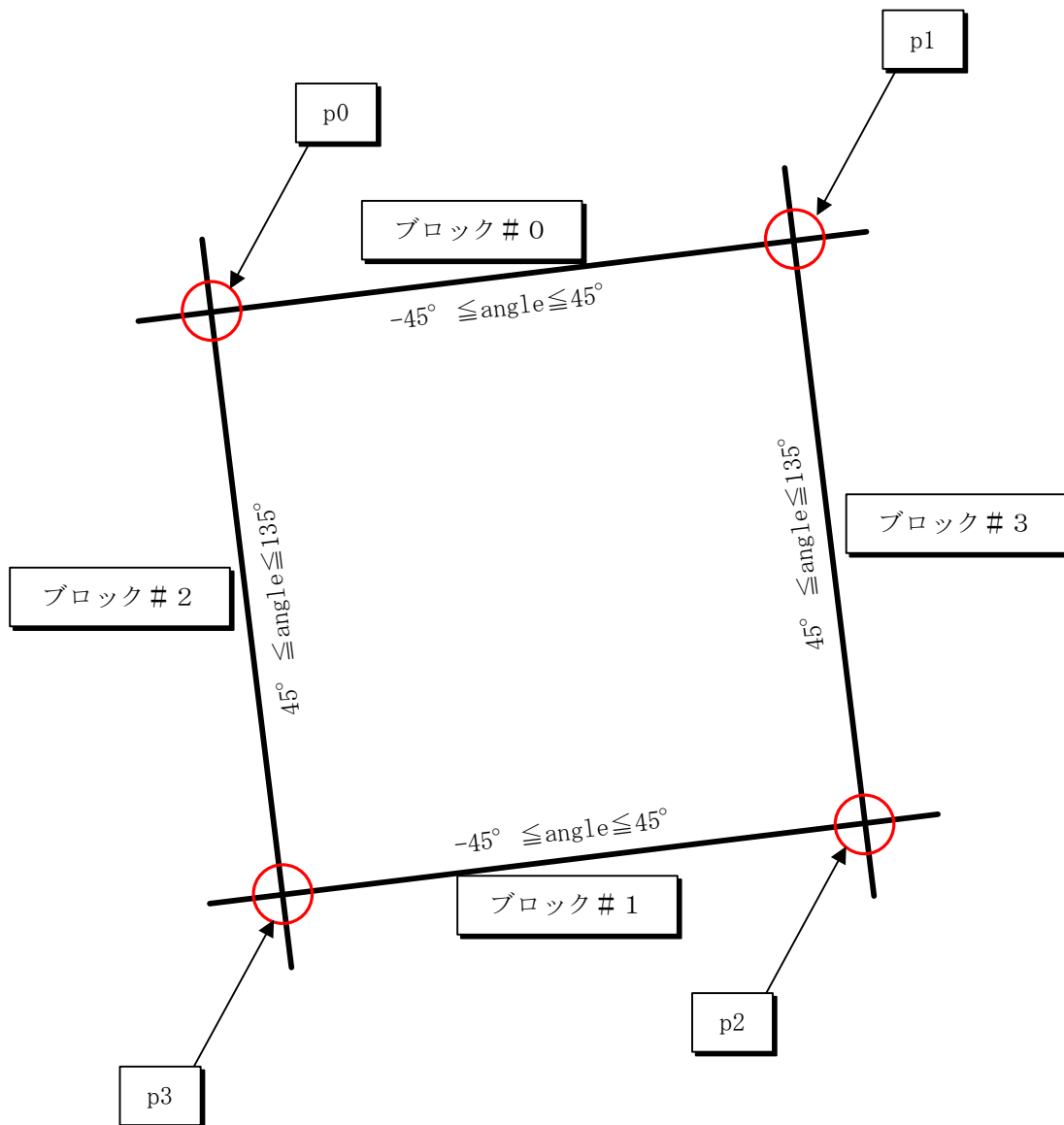
HOUGHLINEのフォーマットは、GetHoughLine()の詳細情報の欄を参照して下さい。
FRECTPOINTのフォーマットを下記に示します。

vp900.hで宣言しています。

```
typedef struct {
    FPOINT  p0; // 直線の交点座標0
    FPOINT  p1; // 直線の交点座標1
    FPOINT  p2; // 直線の交点座標2
    FPOINT  p3; // 直線の交点座標3
} FRECTPOINT;
```

```
typedef struct {
    float  x; // X座標
    float  y; // Y座標
} FPOINT;
```

FPOINTのフォーマットは、GetCrossPoint()の詳細情報の欄を参照して下さい。



3.2.7 矩形頂点中心座標の算出 (GetRectCenter)

□ C言語API

```
int ret = GetRectCenter( FPOINT *block0 , FPOINT *block1 , FPOINT *block2 ,
                        FPOINT *block3 , FPOINT *CenterPoint );
```

□ パラメータ

FPOINT *block0 ブロック 0 の頂点中心座標

FPOINT *block1 ブロック 1 の頂点中心座標

FPOINT *block2 ブロック 2 の頂点中心座標

FPOINT *block3 ブロック 3 の頂点中心座標

FPOINT *CenterPoint 矩形中心座標

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)
FPOINT	*CenterPoint	矩形中心座標を格納したアドレス

□ エラーコード

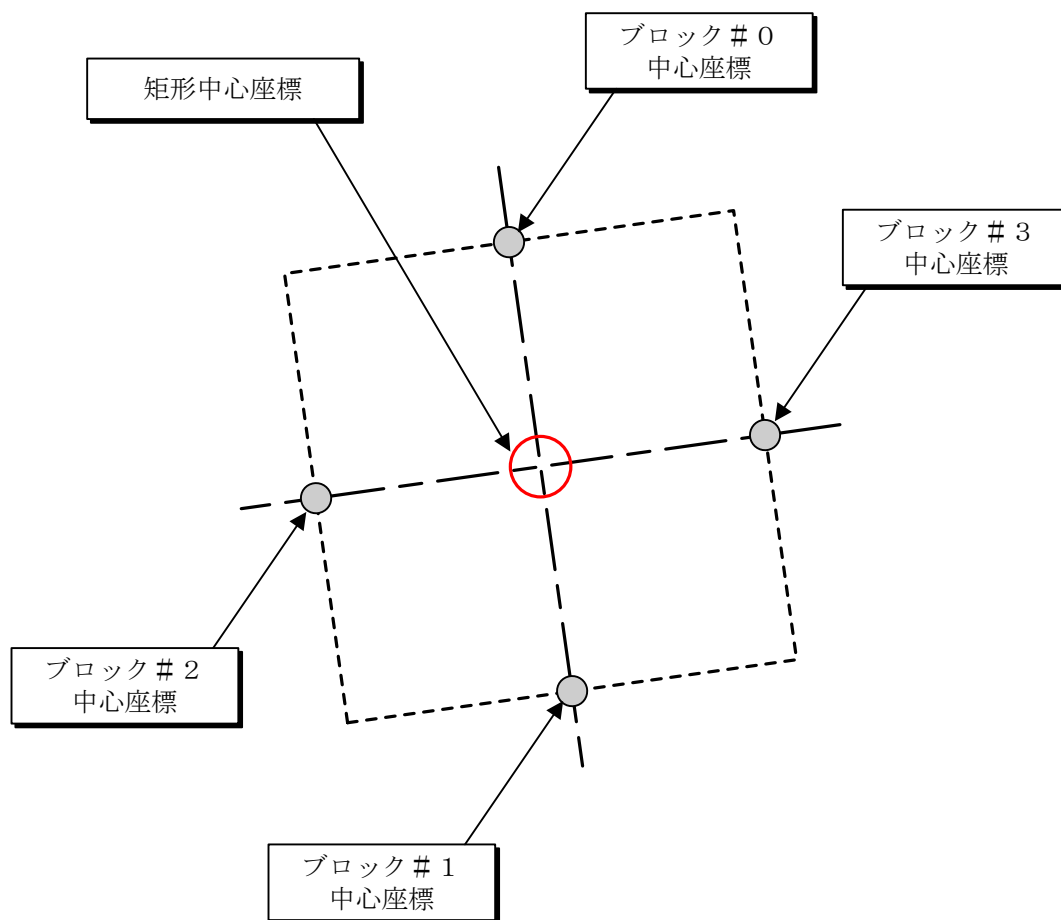
2 0 ~ 2 4	パラメータエラー
2 0 1	0 除算エラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ブロック毎の頂点中心座標から矩形中心座標の算出を行います。

FPOINTのフォーマットは、GetCrossPointの詳細情報の欄を参照して下さい。



3.2.8 矩形頂点間中心座標4ポイントからの角度算出（GetAnglePoint4）

□ C言語API

```
int ret = GetAnglePoint4( FPOINT *block0 , FPOINT *block1 , FPOINT *block2 ,
                          FPOINT *block3 , float *angle);
```

□ パラメータ

FPOINT *block0 ブロック 0 の頂点中心座標

FPOINT *block1 ブロック 1 の頂点中心座標

FPOINT *block2 ブロック 2 の頂点中心座標

FPOINT *block3 ブロック 3 の頂点中心座標

float *angle 矩形の傾き角度

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）
float	*angle	矩形の傾き角度を格納したアドレス

□ エラーコード

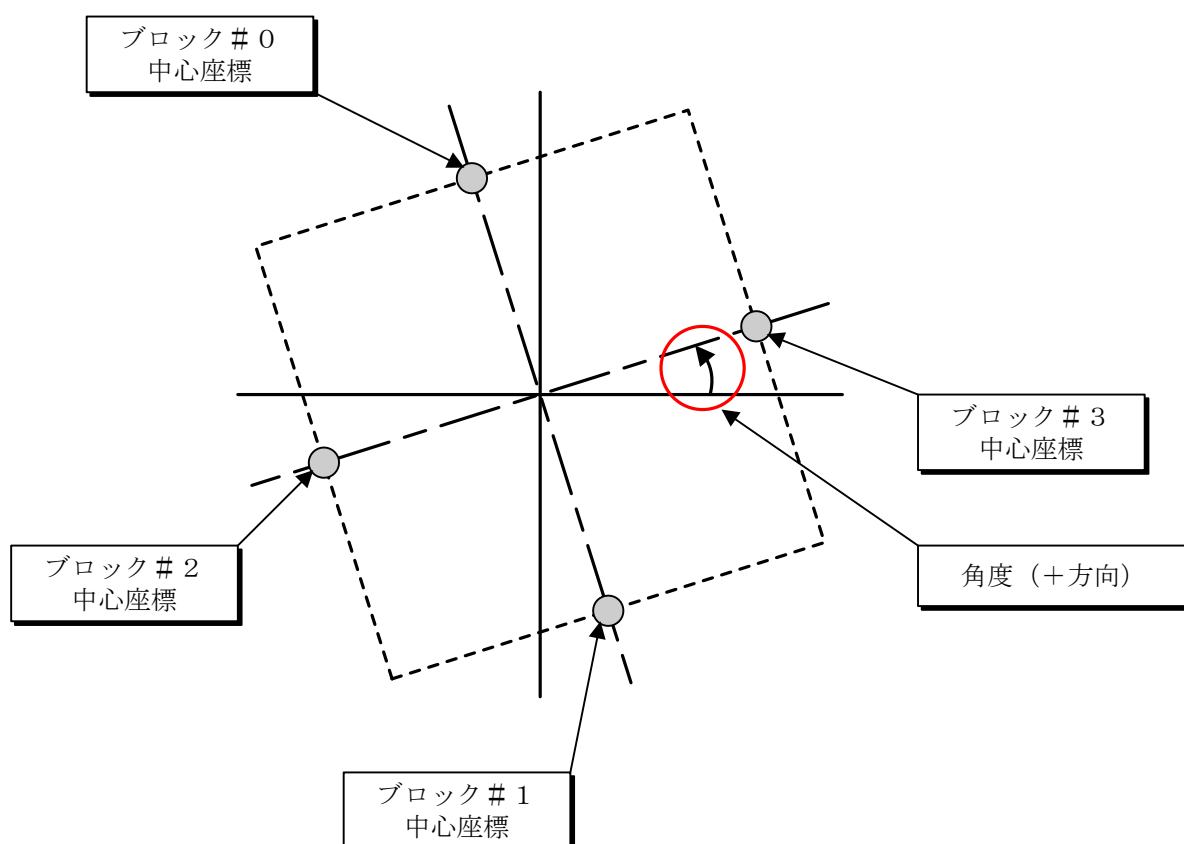
2 0 ~ 2 4	設定範囲外（パラメータエラー）
2 0 1	0 除算エラー
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ブロック毎の頂点中心座標 4 ポイントから矩形の傾き角度の算出を行います。

FPOINTのフォーマットは、GetCrossPoint()の詳細情報の欄を参照して下さい。



3.2.9 矩形頂点間中心座標2ポイントからの角度算出（GetAnglePoint2）

□ C言語API

```
int ret = GetAnglePoint2( FPOINT *block0 , FPOINT *block1 , float *Angle);
```

□ パラメータ

FPOINT *block0 ブロック # 0 , # 2 の頂点中心座標

FPOINT *block1 ブロック # 1 , # 3 の頂点中心座標

float *Angle 矩形の傾き角度

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）
float	*Angle	矩形の傾き角度を格納したアドレス

□ エラーコード

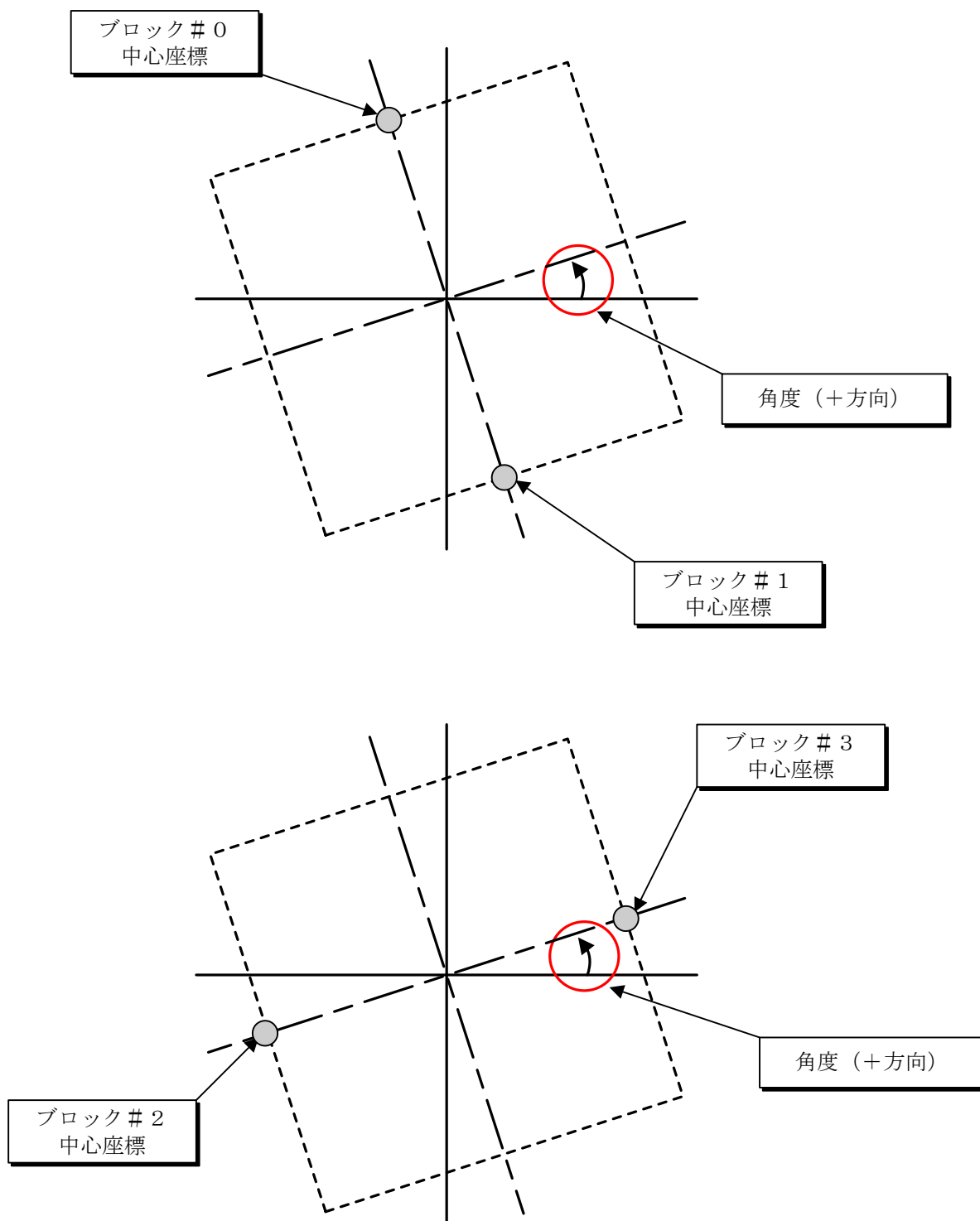
2 0 ~ 2 2	設定範囲外（パラメータエラー）
2 0 0	0 除算エラー
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ブロック毎の頂点中心座標 2 ポイントから矩形の傾き角度の算出を行います。

FPOINTのフォーマットは、GetCrossPoint()の詳細情報の欄を参照して下さい。



3.3 イメージキャリパ

3.3.1 傾斜矩形領域の投影 (ProjectLine)

□ C言語API

```
int ret = ProjectLine( IMGID ImgID , LINETBL *LineTbl , int mode , LINEWINDOW *LineWindow );
```

□ パラメータ

IMGID	ImgID	処理画面の画面番号
LINETBL	*LineTbl	読み出しデータテーブル
int	mode	モード（現在未使用）。「0」を設定して下さい。
LINEWINDOW	*LineWindow	投影領域（ラインウィンドウ）

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	△R	○	○	○	×	○	△U	△U

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）
LINETBL	*LineTbl	読み出しデータを格納したアドレス

□ エラーコード

2	OSでのエラー(タイムアウト以外)
3	IMP処理実行タイムアウト
8	VIN入力タイムアウト
1 0	画面空き領域、または範囲外（不当画面番号エラー）
2 1 ~ 2 3	パラメータエラー
4 9	アクセス画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ImgIDで指定された画面番号の画像メモリから、投影領域（sx、sy）を開始点座標、（ex、ey）を終了点座標とする直線上をleng幅で投影したデータを、データテーブルに読み出します。

LINETBL、LINEWINDOWのフォーマットおよび内容を下記に示します。
vp900.hで宣言しています。

```
typedef struct{
    short        lower;
    short        num;
    short        data[ _LINETBL_DATA_NUM ];
} LINETBL;
```

LINETBL	説明
lower	開始点座標（ sx / sy ）
num	有効データ数（ ex - sx + 1 / ey - sy + 1 ）
data[_LINETBL_DATA_NUM]	データ

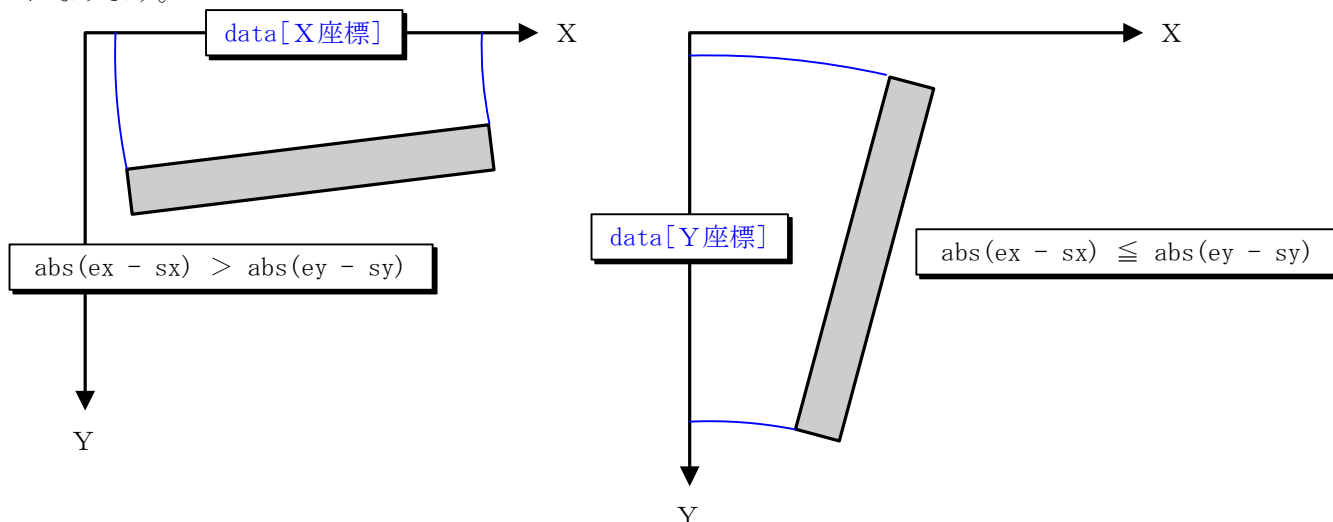
```
typedef struct {
    short        sx;
    short        sy;
    short        ex;
    short        ey;
    short        leng;
    short        opt;
} LINEWINDOW;
```

LINEWINDOW	説明
sx	開始点座標 X
sy	開始点座標 Y
ex	終了点座標 X
ey	終了点座標 Y
leng	投影幅
opt	オプション（「0」を設定して下さい）。 CaliperLPtoSP(), EdgeFinderLPtoSP() コマンドで意味を持ちます。詳細はそちらを参照して下さい。

読み出しデータは、data[0]からdata[num-1]に書き込まれます。そのときのdata[0]の投影座標がlowerになります。また、設定されたdata[]配列のポインタ（添字）は

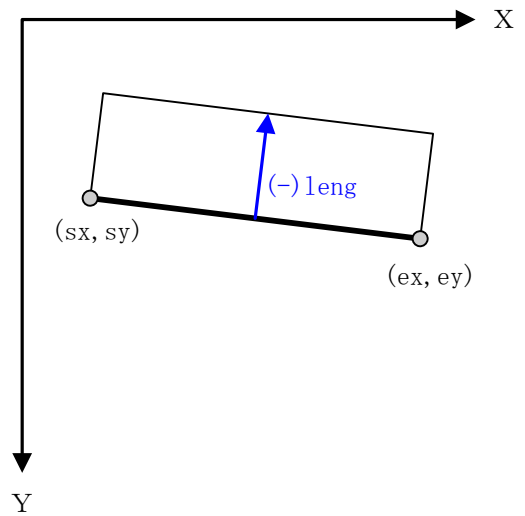
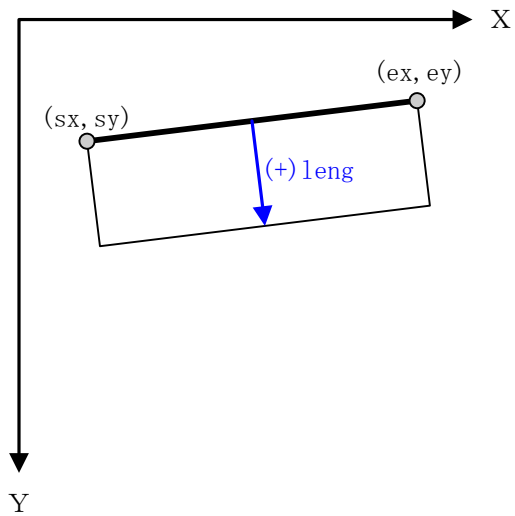
```
abs(ex - sx) > abs(ey - sy) : data[ X座標 ]
abs(ex - sx) ≤ abs(ey - sy) : data[ Y座標 ]
```

になります。



投影幅 `leng` は符号により以下のように投影する方向が変わります。

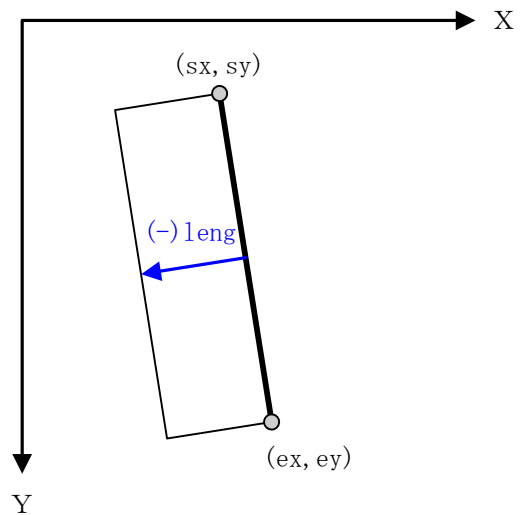
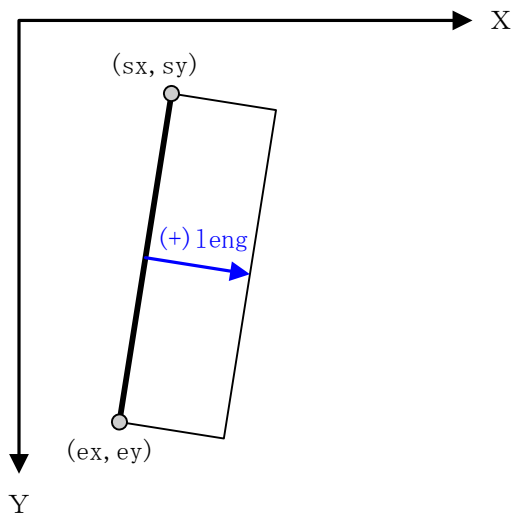
$$\text{abs}(ex - sx) > \text{abs}(ey - sy)$$



ラインテーブルに出力される値

LineTbl.lower = `sx`
 LineTbl.num = `ex - sx + 1`
 LineTbl.data[] = X軸上に投影されるデータ

$$\text{abs}(ex - sx) \leq \text{abs}(ey - sy)$$



ラインテーブルに出力される値

LineTbl.lower = `sy`
 LineTbl.num = `ey - sy + 1`
 LineTbl.data[] = Y軸上に投影されるデータ

3.3.2 エッジ抽出 (LineEdgeFilter)

□ C言語API

```
int ret = LineEdgeFilter( LINETBL *ProjectTbl , LINETBL *EdgeTbl );
```

□ パラメータ

LINETBL *ProjectTbl 投影データテーブル

LINETBL *EdgeTbl 出力エッジテーブル

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)
LINETBL	*EdgeTbl	エッジデータを格納したアドレス

□ エラーコード

20、21	パラメータエラー
—	エラーリセットコマンド (ClearIPError) 未発行

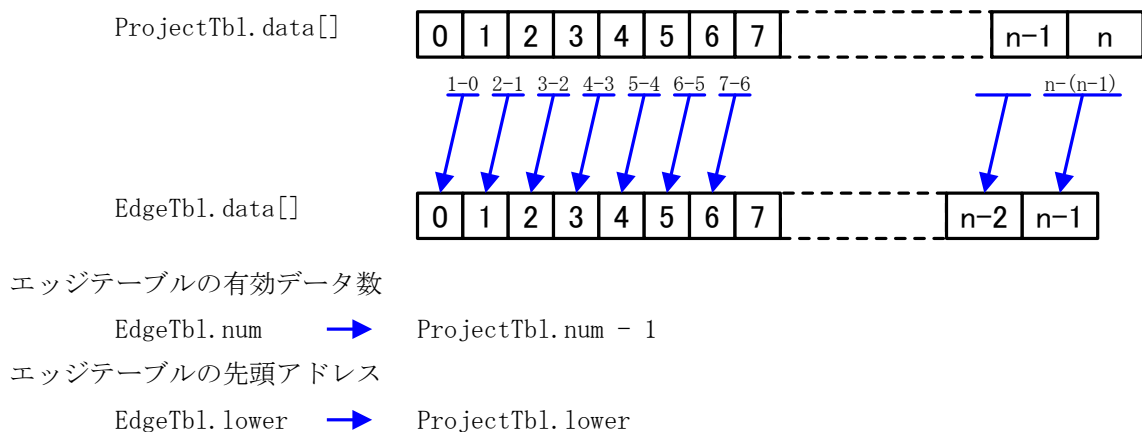
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ProjectTblに設定された画像データ及び投影データの微分を行い、そのエッジ成分を抽出し、EdgeTblに出力します。

LINETBLのフォーマット及び内容は、ProjectLine()の詳細情報の欄を参照して下さい。

本コマンドは、以下の演算を行いEdgeTblに出力します。



3.3.3 フィルタ付きエッジ抽出 (LineEdgeFilterExt)

□ C言語API

```
int ret = LineEdgeFilterExt( LINETBL *ProjectTbl , LINETBL *EdgeTbl , int offset , int scale ,
                           int size , short *COEFF );
```

□ パラメータ

LINETBL *ProjectTbl	投影データテーブル
LINETBL *EdgeTbl	出力エッジテーブル
int offset	オフセット。出力エッジテーブルの有効データが始まる位置を指定して下さい。 また、先頭から指定した分だけ「0」が設定されます。 通常は、 $\text{INT}(\text{size} / 2 - 0.5)$ を設定して下さい。
int scale	スケールダウン量。荷重係数との積和演算後の除算の除数ですので 「0」以外の値を設定して下さい。
int size	荷重係数のサイズ
short *COEFF	荷重係数テーブルへのポインタ

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int LINETBL	ret *EdgeTbl	正常終了 (0) または異常終了 (-1) エッジデータを格納したアドレス

□ エラーコード

20 ~ 25	パラメータエラー
—	エラーリセットコマンド (ClearIPError) 未発行

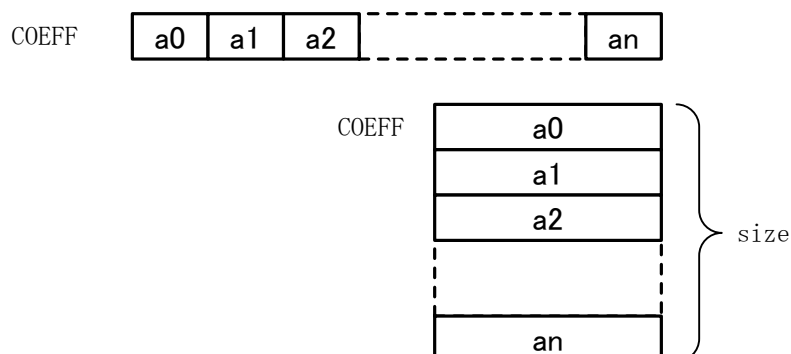
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

機能

ProjectTblに設定された画像データ及び投影データをCOEFFテーブルの空間オペレータでフィルタリングし、EdgeTblに出力します。

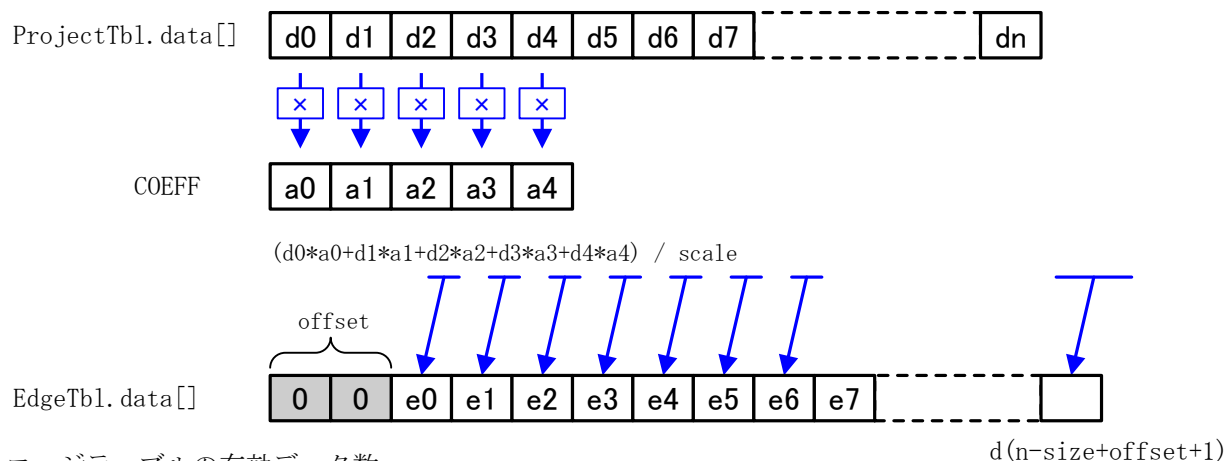
LINETBLのフォーマット及び内容は、ProjectLine()の詳細情報の欄を参照して下さい。

荷重係数 COEFF のフォーマットを以下に示します。



本コマンドは、以下の演算を行いEdgeTblに出力します。

荷重係数サイズ size = 5 , offset = 2 の時



エッジテーブルの有効データ数

EdgeTbl.num $\rightarrow$ ProjectTbl.num - size + offset + 1

エッジテーブルの先頭アドレス

EdgeTbl.lower $\rightarrow$ ProjectTbl.lower

offset の設定値の目安は、以下のようになります。

$$\text{offset} = \text{INT}(\text{size} / 2 - 0.5)$$

設定例

COEFF	<table><tr><td>-1</td><td>-1</td><td>0</td><td>1</td><td>1</td></tr></table>	-1	-1	0	1	1
-1	-1	0	1	1		
size	5					
scale	1					
offset	2					

3.3.4 ラインキャリパ (LineCaliper)

□ C言語API

```
int ret = LineCaliper( LINETBL *LineTbl , LINECALIPERTBL *CaliperTbl ,
                      enum IPCaliperMode cmode , int score );
```

□ パラメータ

LINETBL *LineTbl (入力) エッジ成分ラインテーブル

LINECALIPERTBL *CaliperTbl (出力) 一次元論理座標のキャリパテーブル

enum IPCaliperMode cmode (入力) 抽出モード

描画モード	対応数	説明
CALIPER_BRIGHT	0	明るいオブジェクトの抽出
CALIPER_DARK	1	暗いオブジェクトの抽出

int score (入力) エッジ判定しきい値

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (−1)

LINECALIPERTBL *CaliperTbl 一次元論理座標のキャリパデータを格納したアドレス

□ エラーコード

20 ~ 22 パラメータエラー

— エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

EdgeTblのエッジ成分からscoreをしきい値としてアップエッジとダウンエッジを検出し、抽出モードmodeに従いオブジェクトの開始点、終了点、中心点、幅を一次元論理座標で抽出し、LineCaliperTblに出力します。

LINECIPERTBLのフォーマット及び内容は、ProjectLine()の詳細情報の欄を参照して下さい。

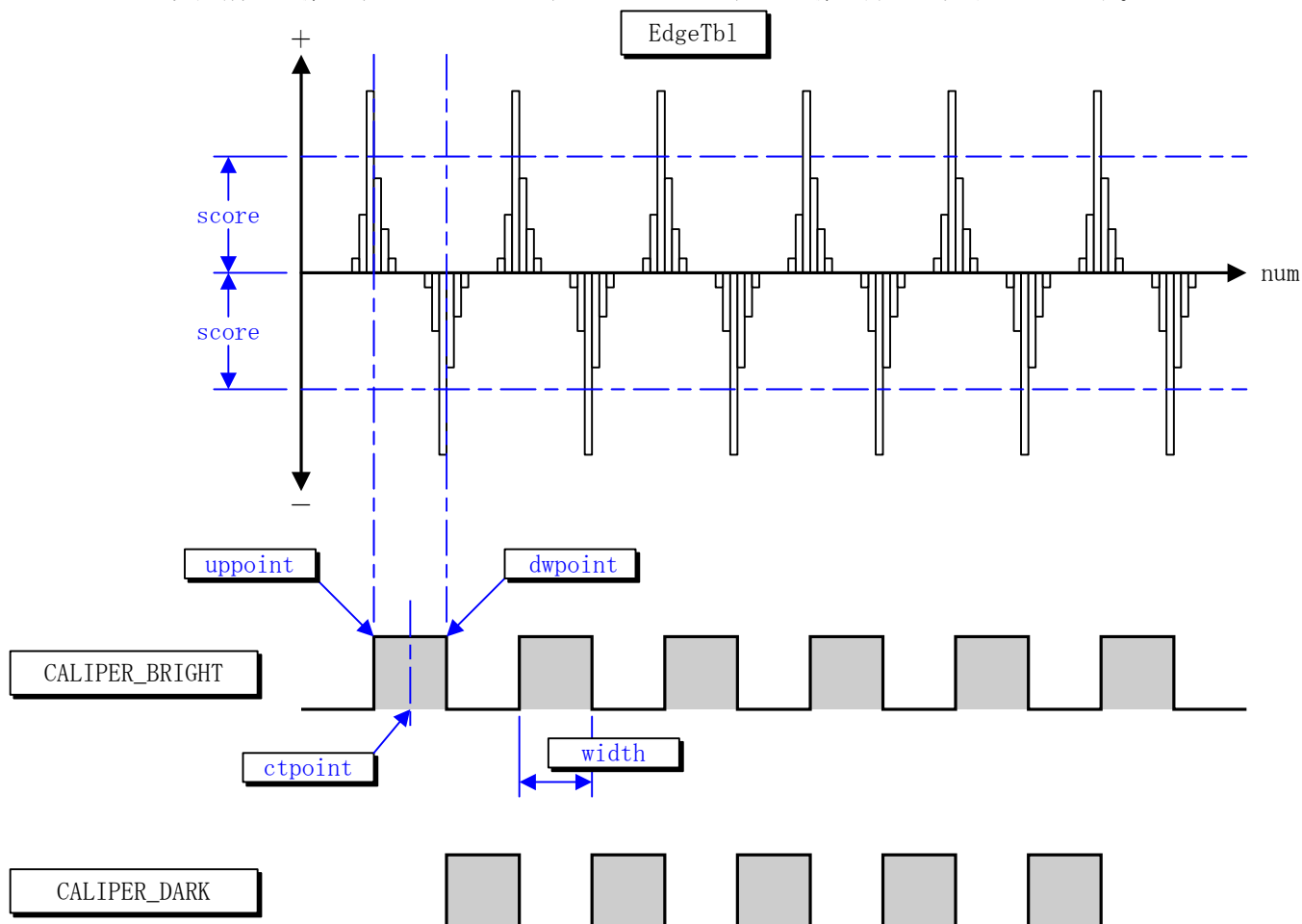
LINECALIPERTBLのフォーマット及び内容を、下記に示します。

vpx900.hで宣言しています。

```
typedef struct{
    short    num;
    short    opt;
    float    uppoint[ _CALIPER_DATA_NUM ];
    float    dwpoint[ _CALIPER_DATA_NUM ];
    float    ctpoint[ _CALIPER_DATA_NUM ];
    float    width[ _CALIPER_DATA_NUM ];
} LINECALIPERTBL;
```

LINEWINDOW	説明
num	抽出オブジェクトの個数
opt	未使用
uppoint[_CALIPER_DATA_NUM]	オブジェクトの開始位置（一次元論理座標）
dwpoint[_CALIPER_DATA_NUM]	オブジェクトの終了位置（一次元論理座標）
ctpoint[_CALIPER_DATA_NUM]	オブジェクトの中心位置（一次元論理座標）
width[_CALIPER_DATA_NUM]	オブジェクトの幅（論理幅）

一次元論理座標キャリパテーブル（LINECALIPERTBL）の座標の詳細を以下に示します。



3.3.5 ライン座標から平面座標への変換（CaliperLPtoSP）

□ C言語API

```
int ret = CaliperLPtoSP( LINECALIPERTBL *LineCaliperTbl , CALIPERTBL *CaliperTbl ,
                        LINEWINDOW *LineWindow , CALIBTBL *Calib );
```

□ パラメータ

LINECALIPERTBL *LineCaliperTbl (入力)
一次元論理座標のラインキャリパテーブル。
LineCaliper() コマンドで得られたラインキャリパテーブルをそのまま設定します。

CALIPERTBL *CaliperTbl (出力)
二次元論理座標のキャリパテーブル。
一次元論理座標のラインキャリパテーブルが二次元論理座標に変換され出力されます。

LINEWINDOW *LineWindow (入力)
投影を行ったラインウィンドウ。
ProjectLine() コマンドで指定したラインウィンドウを設定して下さい。

CALIBTBL *Calib (入力)
キャリブレーションデータテーブル。画素サイズを設定して下さい。

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）
LINECALIPERTBL	*CaliperTbl	二次元論理座標のキャリパデータを格納したアドレス

□ エラーコード

20～23	パラメータエラー
201	0除算エラー
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

LineCaliperコマンドで得られた一次元論理座標のLineCaliperTblのデータをCalibテーブルに設定されたキャリブレーションデータとLineWindowテーブルに設定されたウィンドウテーブルから二次元論理座標に変換し、CaliperTblに設定します。

LINECALIPERTBLのフォーマット及び内容は、LineCaliper()の詳細情報の欄を参照して下さい。
CALIPERTBL、CALIBTBLのフォーマットおよび内容を下記に示します。
vpx900.hで宣言しています。

```
typedef struct{
    LINEWINDOW      LineWindow;
    CFACTOR          calib;
    DFACTOR          disp;
    short            num;
    short            mode;
    FPOINT           uppoint[ _CALIPER_DATA_NUM ];
    FPOINT           dwpoint[ _CALIPER_DATA_NUM ];
    FPOINT           ctpoint[ _CALIPER_DATA_NUM ];
    float            width[ _CALIPER_DATA_NUM ];
    char             flag[ _CALIPER_DATA_NUM ];
} CALIPERTBL;
```

●二次元論理キャリパテーブル

CALIPERTBL	説明
LineWindow	ラインウィンドウ (LineWindowパラメータからコピーされます)
calib	物理座標変換係数
disp	論理座標変換係数
num	抽出ポイント数
mode	0 : $\text{abs}(dx) > \text{abs}(dy)$ 1 : $\text{abs}(dx) \leq \text{abs}(dy)$
uppoint[_CALIPER_DATA_NUM]	オブジェクトの開始ポイント (二次元論理座標)
dwpoint[_CALIPER_DATA_NUM]	オブジェクトの終了ポイント (二次元論理座標)
ctpoint[_CALIPER_DATA_NUM]	オブジェクトの中心ポイント (二次元論理座標)
width[_CALIPER_DATA_NUM]	オブジェクトの幅 (物理幅 単位: mm)
flag[_CALIPER_DATA_NUM]	オブジェクトの有効フラグ = 1 : 有効 = 0 : 無効

```
typedef struct {
    float          cx;
    float          cy;
    float          angle;
    float          pixelx;
    float          pixely;
} CALIBTBL;
```

●キャリブレーションテーブル

CALIBTBL	説明
cx	カメラ中心X座標
cy	カメラ中心Y座標
angle	カメラ傾き
pixelx	画素サイズX (mm)
pixely	画素サイズY (mm)

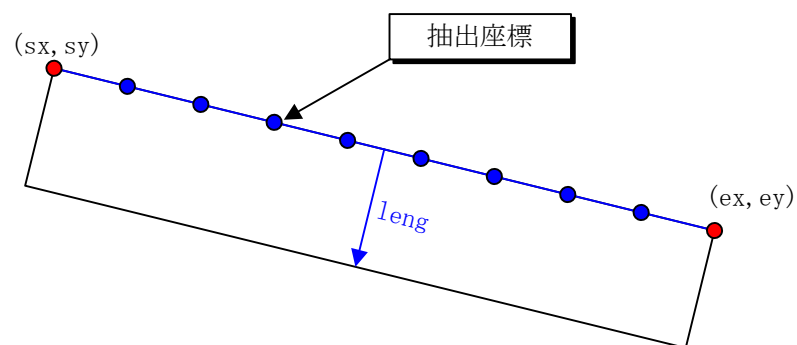
```
typedef struct {
    short    sx;
    short    sy;
    short    ex;
    short    ey;
    short    leng;
    short    opt;
} LINEWINDOW;
```

● ラインウィンドウテーブル

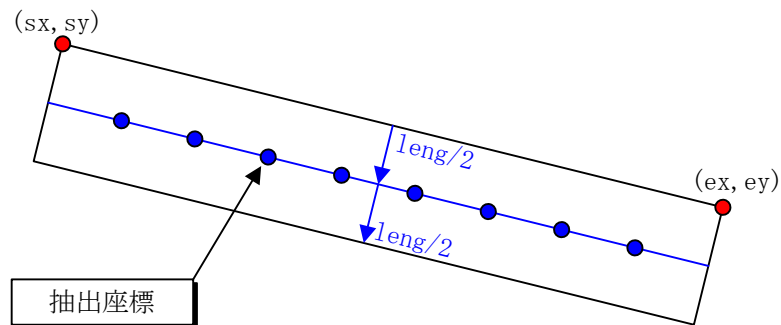
LINEWINDOW	説明								
sx	開始点座標 X								
sy	開始点座標 Y								
ex	終了点座標 X								
ey	終了点座標 Y								
leng	投影幅								
opt	オプション <table border="1"> <thead> <tr> <th>オプション</th><th>内容</th></tr> </thead> <tbody> <tr> <td>0</td><td>ラインウィンドウ直線上の座標を抽出</td></tr> <tr> <td>1</td><td>ラインウィンドウ直線とleng幅との中心座標を抽出</td></tr> <tr> <td>2</td><td>ラインウィンドウ直線とleng幅間隔の座標を抽出</td></tr> </tbody> </table> 詳細を以下に示します。	オプション	内容	0	ラインウィンドウ直線上の座標を抽出	1	ラインウィンドウ直線とleng幅との中心座標を抽出	2	ラインウィンドウ直線とleng幅間隔の座標を抽出
オプション	内容								
0	ラインウィンドウ直線上の座標を抽出								
1	ラインウィンドウ直線とleng幅との中心座標を抽出								
2	ラインウィンドウ直線とleng幅間隔の座標を抽出								

● LINEWINDOWのメンバ変数 opt の詳細

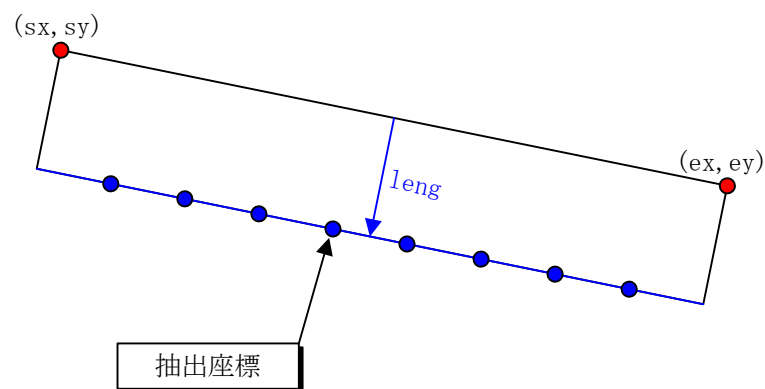
opt : 0



opt : 1



opt : 2



DFACTOR、FPOINTのフォーマットを下記に示します。
vp900.hで宣言しています。

```
typedef struct {
    float    sin;
    float    cos;
    float    Factor;
} DFACTOR;
```

```
typedef struct {
    float    x;
    float    y;
} FPOINT;
```

●論理座標変換係数

DFACTOR	説明
sin	mode(0): $\sin(\text{atan}(\text{dy}/\text{dx}))$ // mode(1): $\sin(\text{atan}(\text{dx}/\text{dy}))$
cos	mode(0): $\cos(\text{atan}(\text{dy}/\text{dx}))$ // mode(1): $\cos(\text{atan}(\text{dx}/\text{dy}))$
Factor	mode(0): $\cos(\text{atan}(\text{dy}/\text{dx}))$ // mode(1): $\sin(\text{atan}(\text{dx}/\text{dy}))$

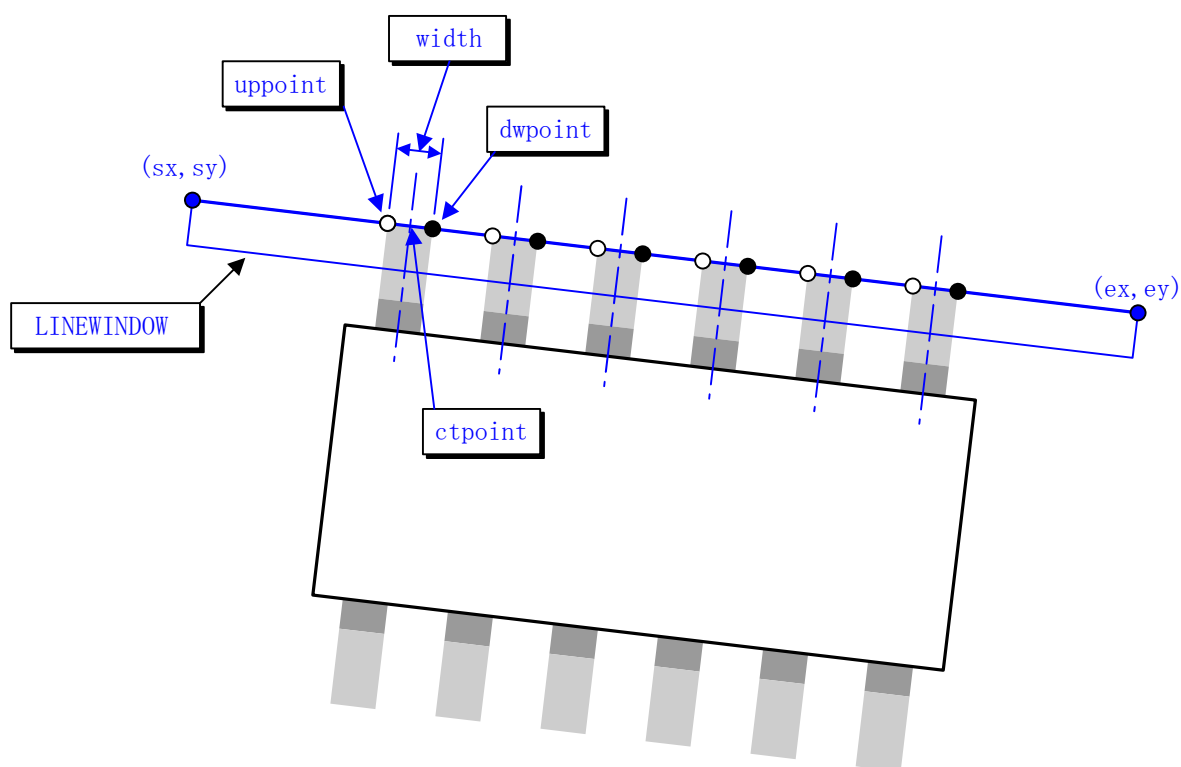
CFACTORのフォーマットを下記に示します。
vp900.hで宣言しています。

```
typedef struct {
    float    pixelx;
    float    pixely;
    float    angle;
    float    sinFactor;
    float    cosFactor;
    float    divFactor;
    float    divSinFactor;
    float    divCosFactor;
} CFACTOR;
```

●物理座標変換係数

CFACTOR	説明
pixelx	Calib.pixelxのコピー
pixely	Calib.pixelyのコピー
angle	Calib.angleのコピー
sinFactor	mode(0): pixelx * calibSin // mode(1): pixely * calibSin
cosFactor	mode(0): pixelx * calibCos // mode(1): pixely * calibCos
divFactor	mode(0): pixelx / calibCos // mode(1): pixely / calibSin
divSinFactor	mode(0): calibSin / pixelx // mode(1): calibSin / pixely
divCosFactor	mode(0): calibCos / pixelx // mode(1): calibCos / pixely

二次元論理座標キャリパテーブル (CALIPERTBL) の座標の詳細を以下に示します。



3.3.6 エッジ判定しきい値の算出 (GetCaliperScore)

□ C言語API

```
int ret = GetCaliperScore( LINETBL *LineTbl );
```

□ パラメータ

LINETBL *LineTbl エッジ成分ラインテーブル

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int ret > 0 : エッジ判定しきい値
 ≤ 0 : 異常終了

□ エラーコード

2 0	パラメータエラー
2 0 1	しきい値算出に失敗。LineTblデータを確認してください。
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

EdgeTblからLineCaliper()に使用するエッジ判定しきい値を算出します。

LINETBLのフォーマット及び内容は、ProjectLine()の詳細情報の欄を参照して下さい。

3.3.7 投影幅の算出 (SetCaliperWidth)

□ C言語API

```
int ret = SetCaliperWidth( CALIBTBL *Calib , LINEWINDOW *LineWindow , float width , int mode );
```

□ パラメータ

CALIBTBL	*Calib	キャリブレーションデータテーブル
LINEWINDOW	*LineWindow	ラインウィンドウテーブル
float	width	投影幅
int	mode	ラインウィンドウ幅の方向 0 : 投影幅+ 1 : 投影幅-

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)
short	LineWindow->leng	投影幅

□ エラーコード

2 0 ~ 2 3	パラメータエラー
2 0 1	0 除算エラー
-	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

LineWindowの投影幅LineWindow->lengを設定します。LineWindowテーブルのsx、sy、ex、eyとキャリブレーションテーブルCalib、方向mode、物理座標の投影幅widthから論理座標投影幅を算出します。

LINETBLのフォーマット及び内容は、ProjectLine()の詳細情報の欄を、CALIBTBLのフォーマット及び内容は、CaliperLPtoSP()の詳細情報の欄を参照して下さい。

3.3.8 ラインエッジファインダ (LineEdgeFinder)

□ C言語API

```
int ret = LineEdgeFinder( LINETBL *EdgeTbl , LINEEDGEFINDERTBL *LineEdgeFinderTbl , int mode ,
                        int minthd , int minscore , int score);
```

□ パラメータ

LINETBL	*EdgeTbl (入力)	エッジ成分ラインテーブル
LINEEDGEFINDERTBL	*LineEdgeFinderTbl (出力)	一次元論理座標のラインエッジ抽出テーブル
int	mode (入力)	抽出モード。「0」を設定して下さい。
int	minthd (入力)	エッジ抽出領域を検出するための2次微分しきい値。 エッジ抽出の領域は minthd と minscoreどちらかの 条件が一致した部分になります。
int	minscore (入力)	エッジ抽出領域を検出するための微分しきい値。 エッジ抽出の領域は minthd と minscoreどちらかの 条件が一致した部分になります。
int	score (入力)	エッジ判定しきい値

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)
LINEEDGEFINDERTBL	*LineEdgeFinderTbl	一次元論理座標のラインエッジデータを格納したアドレス

□ エラーコード

20 ~ 22	パラメータエラー
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

機能

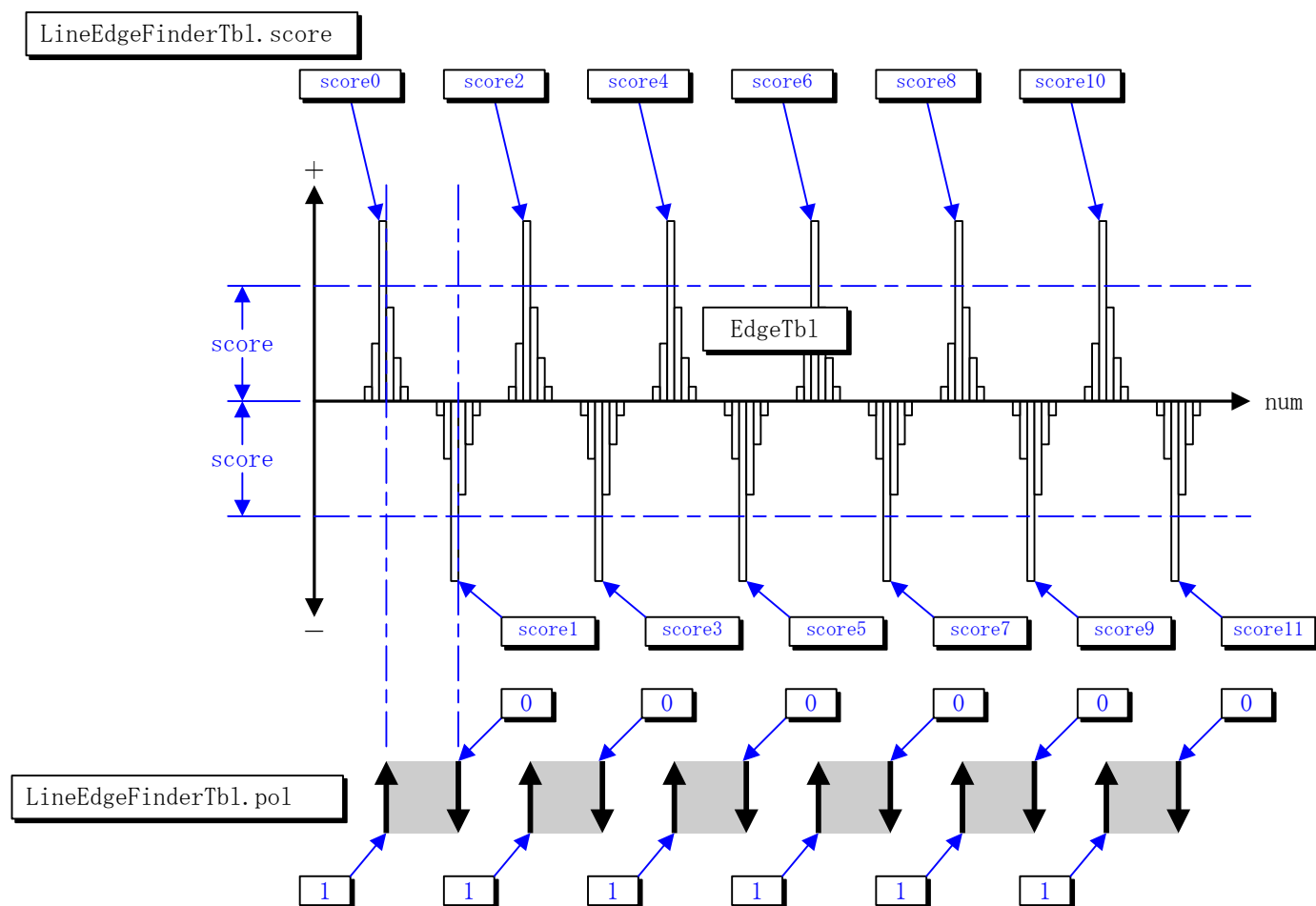
EdgeTblのエッジ成分から、scoreをしきい値としてエッジを検出し、そのエッジの位置を一次元論理座標で抽出し、LineEdgeFinderTblに出力します。
本コマンドでは、抽出したエッジの強さや極性を個別に抽出できます。

LINETBLのフォーマット及び内容は、ProjectLine()の詳細情報の欄を参照して下さい。
LINEEDGEFINDERTBLのフォーマットおよび内容を下記に示します。
vp900.hで宣言しています。

```
typedef struct{
    short      num;
    short      opt;
    float      pos[_EDGEFINDER_DATA_NUM ];
    short      pol[_EDGEFINDER_DATA_NUM ];
    short      score[_EDGEFINDER_DATA_NUM ];
} LINEEDGEFINDERTBL;
```

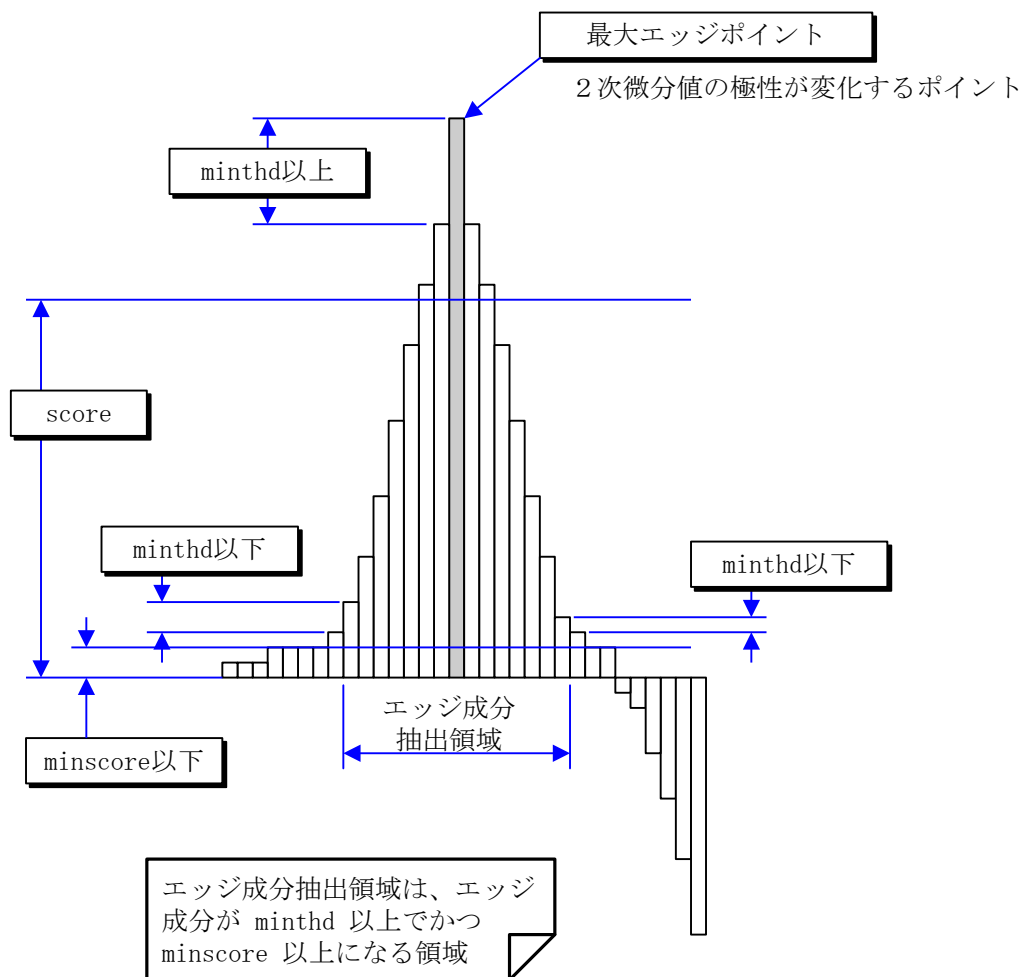
LINEEDGEFINDERTBL	説明	
num	抽出エッジの個数	
opt	未使用	
pos[_EDGEFINDER_DATA_NUM]	抽出エッジの位置（一次元論理座標）	
pol[_EDGEFINDER_DATA_NUM]	抽出エッジの属性	
	属性	内容
	0	明 ⇒ 暗
	1	暗 ⇒ 明
score[_EDGEFINDER_DATA_NUM]	抽出エッジのスコア	

一次元論理座標ラインエッジ抽出テーブル (LINEEDGEFINDER_TBL) の詳細を以下に示します。



- エッジ抽出領域検出 2 次微分しきい値 : minthd
 エッジ抽出領域検出微分しきい値 : minscore

エッジの抽出は、最大エッジポイントから左右にエッジ成分の範囲をサーチし、その範囲内でエッジの中心を求めることにより行います。その場合のエッジの終了点は、エッジ抽出領域検出 2 次微分しきい値 : minthd 以下または、エッジ抽出領域検出微分しきい値 : minscore 以下になるポイントです。



入力されたエッジテーブル（1 次微分値）内のデータをサーチし、score 以上のポイントを探し、次にエッジテーブル内の 2 次微分値を求め、2 次微分値の極性が変化したポイントを探し、最大のエッジポイントとして座標を求めます。そのとき、同時に 2 次微分値が minthd 以下になるかどうかをチェックしており、minthd 以下のポイントが出現した時点でそのポイントを最大のエッジポイントとします。次に、最大のエッジポイント座標を基点にエッジテーブル内のデータをサーチし、エッジ成分抽出領域範囲の座標を決定します。このときの条件は 1 次微分値が minscore 以下になるポイントか、または 2 次微分値が minthd 以下になるポイントです。

エッジ抽出領域検出 2 次微分しきい値 : minthd とエッジ抽出領域検出微分しきい値 : minscore に設定する値は、エッジ成分抽出領域を決める際に 2 次微分値で行うか 1 次微分で行うかを選択できます。1 次微分でエッジ成分抽出領域を決める場合は、minthd を比較的大きな値にする必要があります。逆に、2 次微分で行う場合は、minscore を大きな値にします。通常は、1 次微分のみで十分です。

エッジ成分抽出領域条件	minthd	minscore	備考
1 次微分のみ	score 以上	0 ～ 10 程度	エッジの立ち上がりが急な場合
2 次微分のみ	0 ～ 10 程度	score 以上	エッジの立ち上がりが平坦な場合
1 次微分、2 次微分両方	0 ～ 10 程度	0 ～ 10 程度	

3.3.9 ライン座標から平面座標への変換（EdgeFinderLPtoSP）

□ C言語API

```
int ret = EdgeFinderLPtoSP( LINEEDGEFINDERTBL *LineEdgeFinderTbl ,
                           EDGEFINDERTBL *EdgeFinderTbl , LINEWINDOW *LineWindow );
```

□ パラメータ

LINEEDGEFINDERTBL *LineEdgeFinderTbl（入力）	一次元論理座標ラインエッジ抽出テーブル。 LineEdgeFinder() コマンドで得られた ラインエッジ抽出テーブルをそのまま設定します。
EDGEFINDERTBL *EdgeFinderTbl（出力）	二次元論理座標エッジ抽出テーブル。 一次元論理座標のラインエッジ抽出テーブルが 二次元論理座標に変換され出力されます。
LINEWINDOW *LineWindow（入力）	投影を行ったラインウィンドウ。 ProjectLine() コマンドで指定した ラインウィンドウを設定して下さい。

□ 対応画面タイプ／画面データタイプ

なし

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）
EDGEFINDERTBL	*EdgeFinderTbl	二次元論理座標のエッジデータを格納したアドレス

□ エラーコード

20～22	パラメータエラー
—	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

LineEdgeFinder() コマンドで得られた一次元論理座標のLineEdgeFinderTblのデータをLineWindowテーブルに設定されたウィンドウデータから二次元論理座標に変換し、EdgeFinderTblに設定します。

EDGEFINDERTBL, LINEWINDOWのフォーマットおよび内容を下記に示します。
vp900.hで宣言しています。

```
typedef struct {
    short    num;
    short    opt;
    FPOINT    pos[ _EDGEFINDER_DATA_NUM ];
    short    pol[ _EDGEFINDER_DATA_NUM ];
    short    score[ _EDGEFINDER_DATA_NUM ];
} EDGEFINDERTBL;
```

```
typedef struct {
    float    x;
    float    y;
} FPOINT;
```

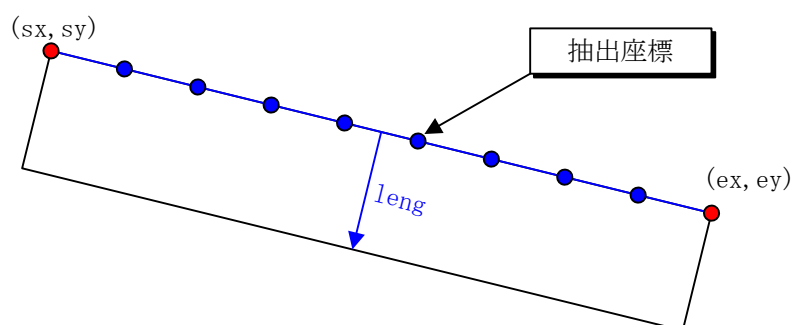
EDGEFINDERTBL	説明						
num	抽出エッジの個数						
opt	未使用						
pos[_EDGEFINDER_DATA_NUM]	抽出エッジの位置（二次元論理座標）						
pol[_EDGEFINDER_DATA_NUM]	抽出エッジの属性 <table border="1"> <tr> <th>属性</th><th>内容</th></tr> <tr> <td>0</td><td>明 ⇒ 暗</td></tr> <tr> <td>1</td><td>暗 ⇒ 明</td></tr> </table>	属性	内容	0	明 ⇒ 暗	1	暗 ⇒ 明
属性	内容						
0	明 ⇒ 暗						
1	暗 ⇒ 明						
score[_EDGEFINDER_DATA_NUM]	抽出エッジのスコア						

```
typedef struct {
    short    sx;
    short    sy;
    short    ex;
    short    ey;
    short    leng;
    short    opt;
} LINEWINDOW;
```

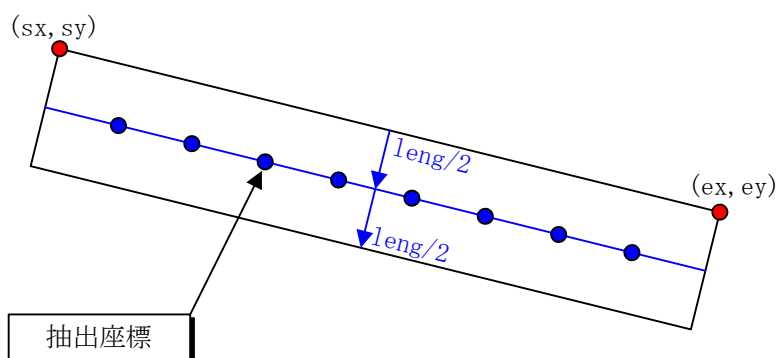
LINEWINDOW	説明								
sx	開始点座標 X								
sy	開始点座標 Y								
ex	終了点座標 X								
ey	終了点座標 Y								
leng	投影幅								
opt	オプション <table border="1"> <tr> <th>オプション</th><th>内容</th></tr> <tr> <td>0</td><td>ラインウィンドウ直線上の座標を抽出</td></tr> <tr> <td>1</td><td>ラインウィンドウ直線とleng幅との中心座標を抽出</td></tr> <tr> <td>2</td><td>ラインウィンドウ直線とleng幅間隔の座標を抽出</td></tr> </table> 詳細を次ページに示します。	オプション	内容	0	ラインウィンドウ直線上の座標を抽出	1	ラインウィンドウ直線とleng幅との中心座標を抽出	2	ラインウィンドウ直線とleng幅間隔の座標を抽出
オプション	内容								
0	ラインウィンドウ直線上の座標を抽出								
1	ラインウィンドウ直線とleng幅との中心座標を抽出								
2	ラインウィンドウ直線とleng幅間隔の座標を抽出								

● LINEWINDOWのメンバ変数 opt の詳細

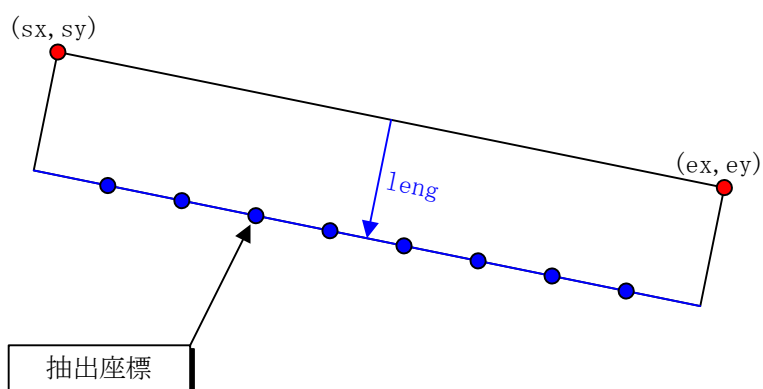
opt : 0



opt : 1



opt : 2



3.4 図形・文字描画

3.4.1 線・矩形描画（PutLine）

□ C言語API

```
int ret = PutLine( IMGID ImgID , enum IPDrawLineMode mode , int sx , int sy , int ex , int ey ,
                  int col);
```

□ パラメータ

IMGID ImgID 描画面の画面番号

enum IPDrawLineMode mode 描画モード

描画モード	対応数	説明
DRAW_LINE	0	線描画 
DRAW_RECT	1	矩形描画 
DRAW_FILL	2	フィル 
DRAW_CROSS_RECT	3	中心線付矩形 
DRAW_LEFT_TOP_RECT	4	左上ポイント付矩形 
DRAW_RIGHT_BOTTOM_RECT	5	右下ポイント付矩形 

int sx 書き込み開始点X座標

int sy 書き込み開始点Y座標

int ex 書き込み終了点X座標

int ey 書き込み終了点Y座標

int col 書き込みデータ（0～255）

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	△R	○	○	○	—	○	○	○

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

1 0 画面空き領域、または範囲外（不当画面番号エラー）、画面属性不正
 2 1 描画モード設定範囲外
 — エラーリセットコマンド（ClearIPError）未発行

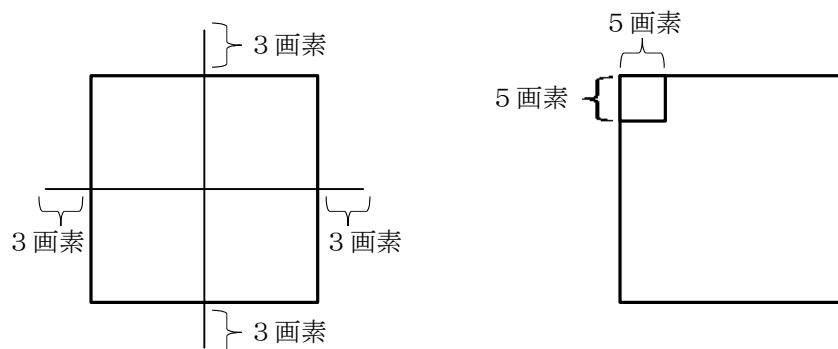
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

描画モードに従い線または矩形を描画します。

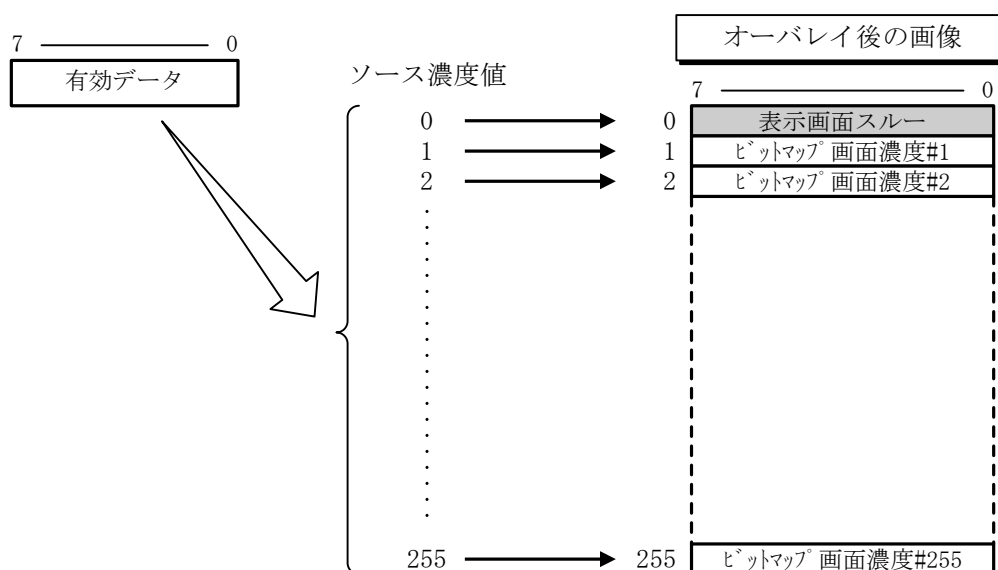
図形描画コマンドの描画は、ローカルCPU（SH）で実行します。コマンド実行中はSHのキャッシュに一時的に蓄えられるため、実際の画像メモリ（IM）には書き込まれていない部分が出てきます。RefreshImg（）コマンドにより描画が全て終了した時点で、そのキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。

十字およびポイントのサイズは図のようになっています。



●ビットマップ画面での表示について

ビットマップ画面（オーバーレイ画面）のオーバーレイ表示はビットマップ画面のデータを変換し、表示します。ビットマップ画面データが「0」の部分は表示画面、「1～255」の部分はビットマップ画面の映像を表示します。



3.4.2 十字カーソル描画 (PutCross)

□ C言語API

```
int ret = PutCross( IMGID ImgID , int mode , int xp , int yp , int col );
```

□ パラメータ

IMGID ImgID 描画画面の画面番号

int mode 描画モード

描画モード	説明
0	十字 
1	×印 
2	点 
3	点十字 
4	点×印 
5	点■印 

int xp 中心 X 座標

int yp 中心 Y 座標

int col 書き込みデータ

ビットマップ画面への書き込みデータについては、PutLine() コマンドの詳細情報を参照して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	△R	○	○	○	—	○	○	○

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

1 0 画面空き領域、または範囲外 (不当画面番号エラー)、画面属性不正
 2 1 描画モード設定範囲外
 — エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録 A 「エラーコード一覧」 を参照して下さい。

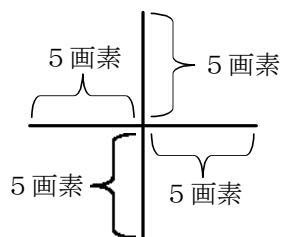
□ 機能

十字カーソルを描画します。

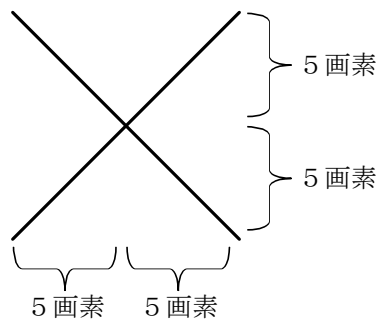
図形描画コマンドの描画は、ローカルCPU（SH）で実行します。コマンド実行中はSHのキャッシュに一時的に蓄えられるため、実際の画像メモリ（IM）には書き込まれていない部分が出てきます。RefreshImg（）コマンドにより描画が全て終了した時点で、そのキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。

十字および×印のサイズは図のようになっています。

十字(mode : 0)



×印(mode : 1)



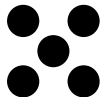
点(mode : 2)



点＋字(mode : 3)



点×印(mode : 4)



点■印(mode : 5)



3.4.3 ラインウィンドウ描画（PutLineWindow）

□ C言語API

```
int ret = PutLineWindow( IMGID ImgID , LINEWINDOW *LineWindow , int col );
```

□ パラメータ

IMGID ImgID 描画画面の画面番号

LINEWINDOW *LineWindow ラインウィンドウを返すアドレス

int col 書き込みデータ。ビットマップ画面への書き込みデータについては、PutLine() コマンドの詳細情報を参照して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	○	△R	○	○	—	○	○	○

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

1 0 画面空き領域、または範囲外（不当画面番号エラー）、画面属性不正
 2 1 ラインウィンドウを返すアドレスが不正
 — エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

機能

ラインウィンドウの外形を線描画します 図形描画コマンドの描画は、ローカルCPU（SH）で実行します。コマンド実行中はSHのキャッシュに一時的に蓄えられるため、実際の画像メモリ（IM）には書き込まれていない部分が出てきます。RefreshImg（）コマンドにより描画が全て終了した時点で、そのキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。

LINEWINDOWのフォーマットおよび内容を下記に示します。
vp900.hで宣言しています。

```
typedef struct {
    short      sx;
    short      sy;
    short      ex;
    short      ey;
    short      leng;
    short      opt;
} LINEWINDOW;
```

LINEWINDOW	説明
sx	開始点座標X
sy	開始点座標Y
ex	終了点座標X
ey	終了点座標Y
leng	投影幅
opt	オプション（未使用）

3.4.4 多角形描画（PutPolygon）

□ C言語API

```
int ret = PutPolygon( IMGID ImgID , enum IPDrawLineMode mode , WPOINT *point ,
                     int num , int col );
```

□ パラメータ

IMGID ImgID 処理画面の画面番号

enum IPDrawLineMode mode 描画モード

描画モード	対応数	説明
DRAW_LINE	0	線描画
DRAW_FILL	2	フィル

WPOINT *point 頂点X、Y座標列を返すアドレス

int num 頂点数

int col 書き込みデータ。ビットマップ画面への書き込みデータについては、PutLine() コマンドの詳細情報を参照して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	△R	○	○	○	—	○	○	○

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

1 0 画面空き領域、または範囲外（不当画面番号エラー）、画面属性不正
 2 2 頂点X、Y座標列を返すアドレスが不正
 2 3 頂点数が0未満
 4 8 動的メモリ確保エラー
 — エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

point座標列を頂点とする多角形を描画モードに従い描画します。

図形描画コマンドの描画は、ローカルCPU（SH）で実行します。コマンド実行中はSHのキャッシュに一時的に蓄えられるため、実際の画像メモリ（IM）には書き込まれていない部分が出てきます。RefreshImg（）コマンドにより描画が全て終了した時点で、そのキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。

WPOINTのフォーマットを下記に示します。

vp900.hで宣言しています。

```
typedef struct {  
    short    x;    // X座標  
    short    y;    // Y座標  
} WPOINT;
```


□ C言語API

パラメータ

enum IPDrawLineMode mode 描画モード

□ 対応画面タイプ／画面データタイプ

□ リターンパラメータ

❏ エラーコード

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

3 - 72

3.4.6 円弧描画（PutArc）

□ C言語API

```
int ret = PutArc( IMGID ImgID , enum IPDrawLineMode mode , int xp , int yp , int sx , int sy ,
float angle , int col );
```

□ パラメータ

IMGID ImgID 描画面面の画面番号

enum IPDrawLineMode mode 描画モード

描画モード	対応数	説明
DRAW_LINE	0	線描画
DRAW_FILL	2	フィル

int xp 中心X座標

int yp 中心Y座標

int sx 始点X座標

int sy 始点Y座標

float angle 円弧の角度（degree）

int col 書き込みデータ。ビットマップ画面への書き込みデータについては、PutLine() コマンドの詳細情報を参照して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	△R	○	○	○	—	○	○	○

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

1 0 画面空き領域、または範囲外（不当画面番号エラー）、画面属性不正
2 1 描画モード設定範囲外
4 8 動的メモリ確保エラー
— エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

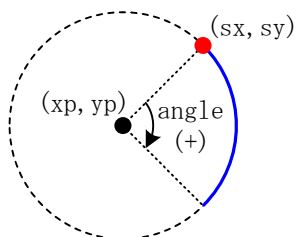
□ 機能

(xp, yp) を中心、(sx, sy) を始点とする角angleの円弧を描画します。

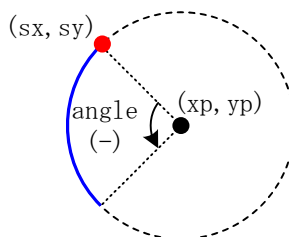
図形描画コマンドの描画は、ローカルCPU (SH) で実行します。コマンド実行中はSHのキャッシュに一時的に蓄えられるため、実際の画像メモリ (IM) には書き込まれていない部分が出てきます。RefreshImg() コマンドにより描画が全て終了した時点で、そのキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。

円弧の中心と始点、描画角度の関係を示します。

angle ≥ 0
(時計回り描画)



angle < 0
(反時計回り描画)



なお、描画モードが DRAW_FILL のとき angle が 180° を超えると正しく描画できません。

3.4.7 矩形描画（PutRectangle）

□ C言語API

```
int ret = PutRectangle( IMGID ImgID , enum IPDrawLineMode mode , int xp , int yp , int lengh ,
                        int lengv , int round , float angle , int col );
```

□ パラメータ

IMGID ImgID 描画面面の画面番号

enum IPDrawLineMode mode 描画モード

描画モード	対応数	説明
DRAW_LINE	0	線描画
DRAW_FILL	2	フィル

int xp 中心 X 座標

int yp 中心 Y 座標

int lengh 横幅

int lengv 縦幅

int round 角の円弧の半径

float angle 傾き（degree）

int col 書き込みデータ。ビットマップ画面への書き込みデータについては、PutLine() コマンドの詳細情報を参照して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	△R	○	○	○	—	○	○	○

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

1 0 画面空き領域、または範囲外（不当画面番号エラー）、画面属性不正
 2 1 描画モード設定範囲外
 2 4 横幅が 0 未満
 2 5 縦幅が 0 未満
 4 8 動的メモリ確保エラー
 — エラーリセットコマンド（ClearIPError）未発行

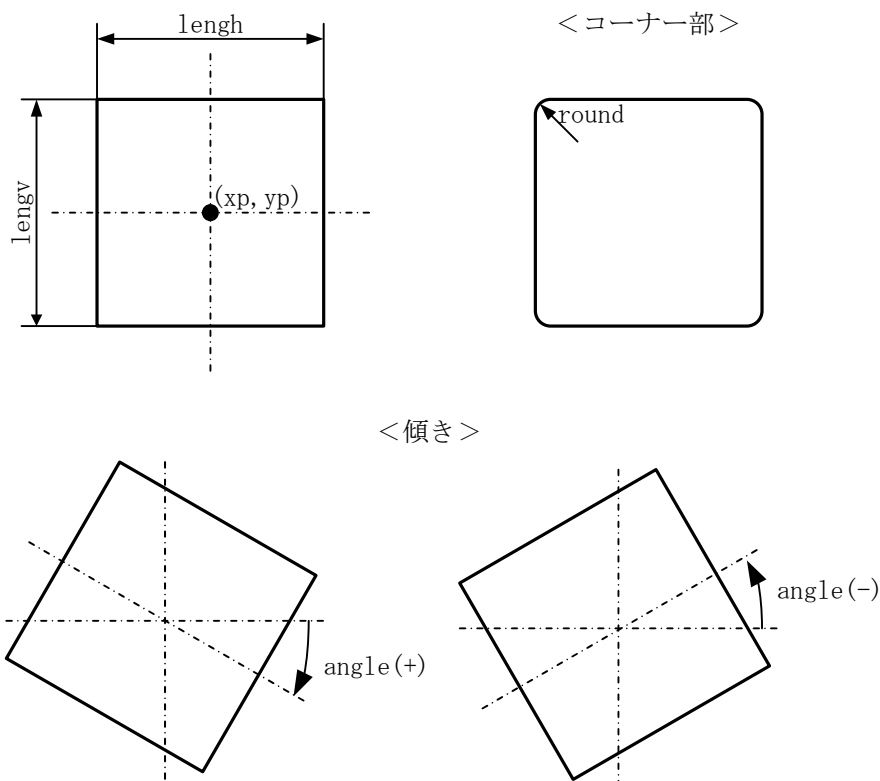
※ 詳細は、付録 A「エラーコード一覧」を参照して下さい。

□ 機能

(xp, yp) を中心座標、横幅 (length) 、縦幅 (length) 、角の円弧の半径 (round) 、傾き (angle) とする四角形を描画します。

図形描画コマンドの描画は、ローカルCPU (SH) で実行します。コマンド実行中はSHのキャッシュに一時的に蓄えられるため、実際の画像メモリ (IM) には書き込まれていない部分が出てきます。RefreshImg() コマンドにより描画が全て終了した時点で、そのキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。

四角形と各パラメータの関係を示します。



3.4.8 三角形描画（PutTriangle）

□ C言語API

```
int ret = PutTriangle( IMGID ImgID , enum IPDrawLineMode mode , int xp , int yp , int lenth ,
                      int lengv , int round , float angle , int col );
```

□ パラメータ

IMGID ImgID 描画面面の画面番号

enum IPDrawLineMode mode 描画モード

描画モード	対応数	説明
DRAW_LINE	0	線描画
DRAW_FILL	2	フィル

int xp 中心 X 座標

int yp 中心 Y 座標

int lenth 横幅

int lengv 縦幅

int round 角の円弧の半径

float angle 傾き（degree）

int col 書き込みデータ。ビットマップ画面への書き込みデータについては、PutLine() コマンドの詳細情報を参照して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	△R	○	○	○	—	○	○	○

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

1 0 画面空き領域、または範囲外（不当画面番号エラー）、画面属性不正
 2 1 描画モード設定範囲外
 2 4 横幅が 0 未満
 2 5 縦幅が 0 未満
 4 8 動的メモリ確保エラー
 — エラーリセットコマンド（ClearIPError）未発行

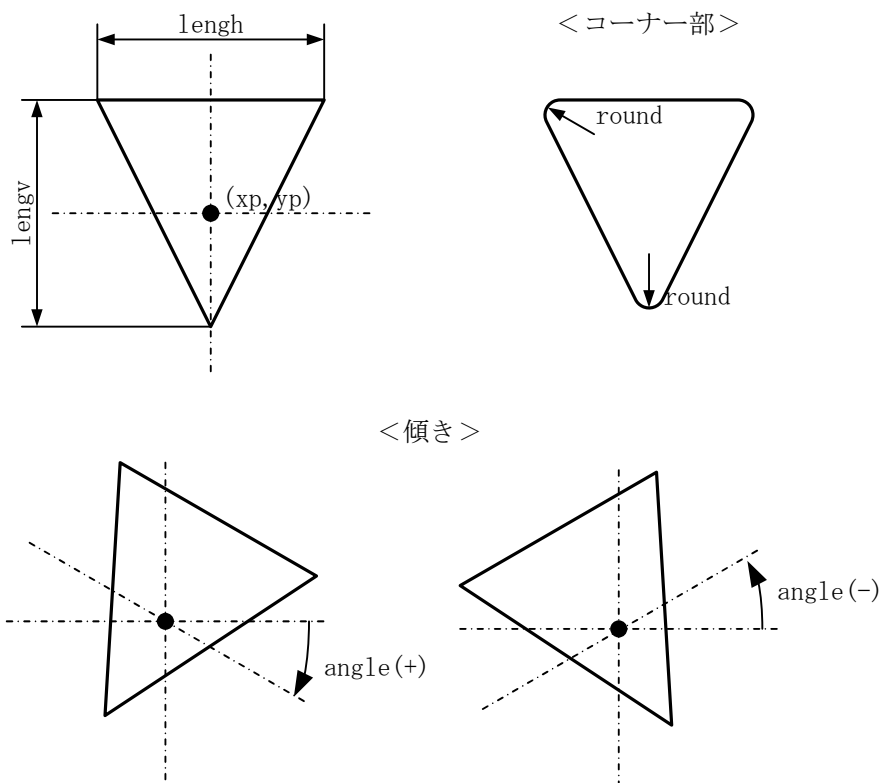
※ 詳細は、付録 A「エラーコード一覧」を参照して下さい。

□ 機能

(xp, yp) を中心座標、横幅 (lengh) 、縦幅 (lengv) 、角の円弧の半径 (round) 、傾き (angle) とする三角形を描画します。

図形描画コマンドの描画は、ローカルCPU (SH) で実行します。コマンド実行中はSHのキャッシュに一時的に蓄えられるため、実際の画像メモリ (IM) には書き込まれていない部分が出てきます。RefreshImg() コマンドにより描画が全て終了した時点で、そのキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。

三角形と各パラメータの関係を示します。



3.4.9 菱形描画（PutDiamond）

□ C言語API

```
int ret = PutDiamond( IMGID ImgID , enum IPDrawLineMode mode , int xp , int yp , int lengh ,
                     int lengv , int round , float angle , int col );
```

□ パラメータ

IMGID ImgID 描画面面の画面番号

enum IPDrawLineMode mode 描画モード

描画モード	対応数	説明
DRAW_LINE	0	線描画
DRAW_FILL	2	フィル

int xp 中心 X 座標

int yp 中心 Y 座標

int lengh 横幅

int lengv 縦幅

int round 角の円弧の半径

float angle 傾き（degree）

int col 書き込みデータ。ビットマップ画面への書き込みデータについては、PutLine() コマンドの詳細情報を参照して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	△R	○	○	○	—	○	○	○

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

1 0 画面空き領域、または範囲外（不当画面番号エラー）、画面属性不正
 2 1 描画モード設定範囲外
 2 4 横幅が 0 未満
 2 5 縦幅が 0 未満
 4 8 動的メモリ確保エラー
 — エラーリセットコマンド（ClearIPError）未発行

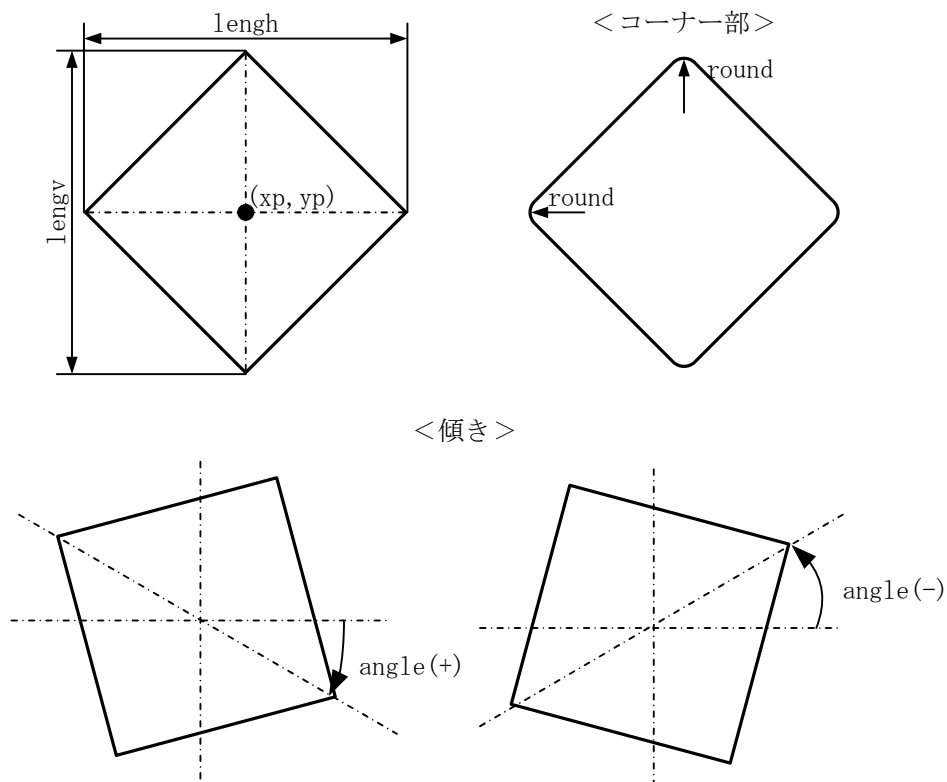
※ 詳細は、付録 A「エラーコード一覧」を参照して下さい。

□ 機能

(x_p, y_p) を中心座標、横幅 (length)、縦幅 (lengv)、角の円弧の半径 (round)、傾き (angle) とするひし形を描画します。

図形描画コマンドの描画は、ローカルCPU (SH) で実行します。コマンド実行中はSHのキャッシュに一時的に蓄えられるため、実際の画像メモリ (IM) には書き込まれていない部分が出てきます。RefreshImg() コマンドにより描画が全て終了した時点で、そのキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。

ひし形と各パラメータの関係を示します。



3.4.10 十字形描画（PutCrossRect）

□ C言語API

```
int ret = PutCrossRect( IMGID ImgID , enum IPDrawLineMode mode , int xp , int yp , int lengh ,
                        int lengv , int width , int round , float angle , int col );
```

□ パラメータ

IMGID ImgID 描画面面の画面番号

enum IPDrawLineMode mode 描画モード

描画モード	対応数	説明
DRAW_LINE	0	線描画
DRAW_FILL	2	フィル
DRAW_LINE_R	6	内側ラウンド付線描画
DRAW_FILL_R	7	内側ラウンド付フィル

int xp 中心X座標

int yp 中心Y座標

int lengh 横幅

int lengv 縦幅

int width 腕幅

int round 角の円弧の半径

float angle 傾き（degree）

int col 書き込みデータ。ビットマップ画面への書き込みデータについては、PutLine() コマンドの詳細情報を参照して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	△R	○	○	○	—	○	○	○

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

1 0 画面空き領域、または範囲外（不当画面番号エラー）、画面属性不正
2 1 描画モード設定範囲外
2 4 横幅が0未満
2 5 縦幅が0未満
2 6 腕幅が0未満
4 8 動的メモリ確保エラー
— エラーリセットコマンド（ClearIPError）未発行

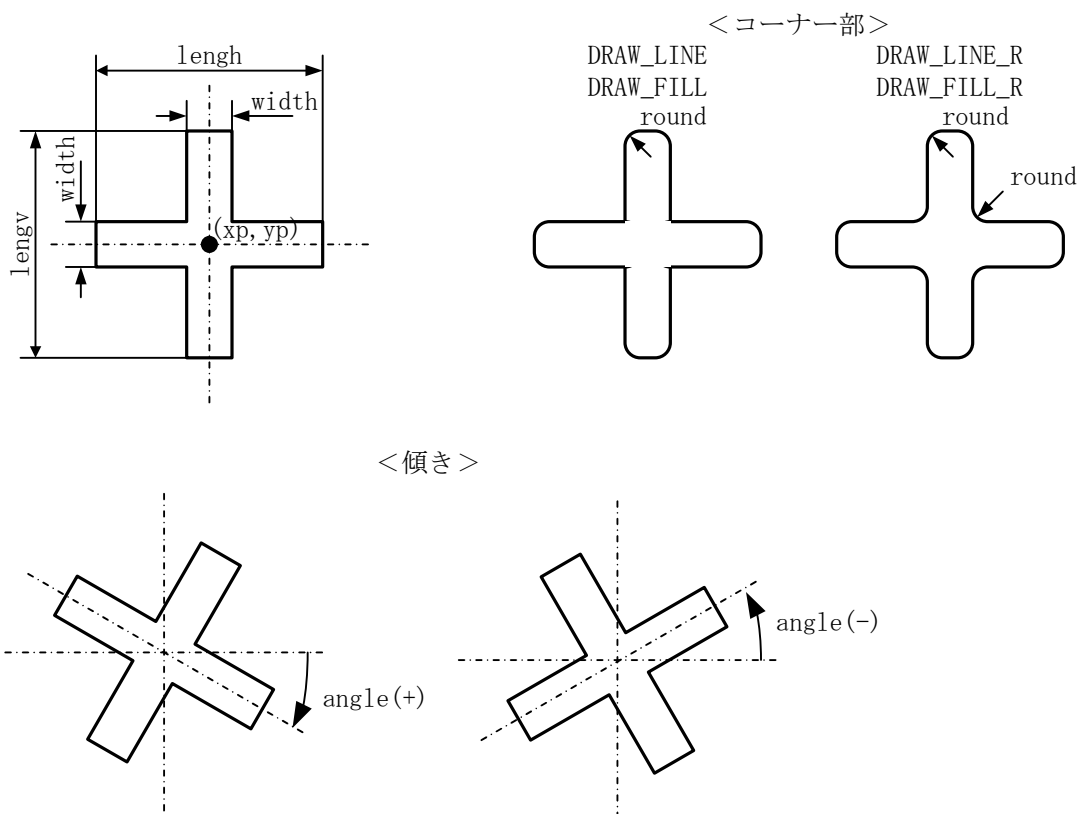
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

(xp, yp) を中心座標、横幅 (length)、縦幅 (length)、腕幅 (width)、角の円弧の半径 (round)、傾き (angle) とする十字形を描画します。

図形描画コマンドの描画は、ローカルCPU (SH) で実行します。コマンド実行中はSHのキャッシュに一時的に蓄えられるため、実際の画像メモリ (IM) には書き込まれていない部分が出てきます。RefreshImg() コマンドにより描画が全て終了した時点で、そのキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。

十字形と各パラメータの関係を示します。



3.4.11 2連矩形描画（PutTwinRectangle）

□ C言語API

```
int ret = PutTwinRectangle( IMGID ImgID , enum IPDrawLineMode mode , int xp ,int yp ,
                           int lengh , int lengv , int width , int round , float angle , int col );
```

□ パラメータ

IMGID ImgID 描画面面の画面番号

enum IPDrawLineMode mode 描画モード

描画モード	対応数	説明
DRAW_LINE	0	線描画
DRAW_FILL	2	フィル
DRAW_LINE_R	6	内側ラウンド付線描画
DRAW_FILL_R	7	内側ラウンド付フィル

int xp 中心X座標

int yp 中心Y座標

int lengh 横幅

int lengv 縦幅

int width 腕幅

int round 角の円弧の半径

float angle 傾き（degree）

int col 書き込みデータ。ビットマップ画面への書き込みデータについては、PutLine() コマンドの詳細情報を参照して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	△R	○	○	○	—	○	○	○

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

1 0 画面空き領域、または範囲外（不当画面番号エラー）、画面属性不正
 2 1 描画モード設定範囲外
 2 4 横幅が0未満
 2 5 縦幅が0未満
 2 6 腕幅が0未満
 4 8 動的メモリ確保エラー
 — エラーリセットコマンド（ClearIPError）未発行

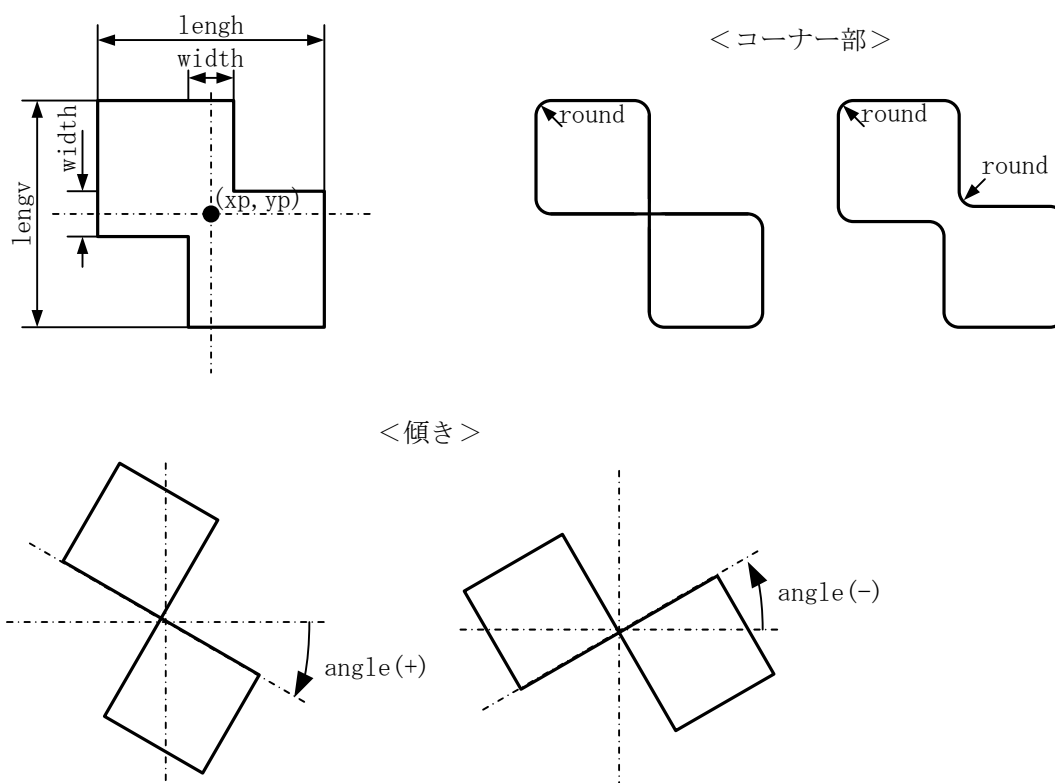
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

(xp, yp) を中心座標、横幅 (length)、縦幅 (length)、腕幅 (width)、角の円弧の半径 (round)、傾き (angle) とする 2 四角形を描画します。

図形描画コマンドの描画は、ローカルCPU (SH) で実行します。コマンド実行中はSHのキャッシュに一時的に蓄えられるため、実際の画像メモリ (IM) には書き込まれていない部分が出てきます。RefreshImg () コマンドにより描画が全て終了した時点で、そのキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。

2 四角形と各パラメータの関係を示します。



3.4.12 円弧描画(拡張) (PutArcExt)

□ C言語API

```
int ret = PutArcExt( IMGID ImgID , enum IPDrawLineMode mode , float xp , float yp ,
                    float sx , float sy , float angle , int col );
```

□ パラメータ

IMGID ImgID 描画面面の画面番号

enum IPDrawLineMode mode 描画モード

描画モード	対応数	説明
DRAW_LINE	0	線描画
DRAW_RECT	1	
DRAW_FILL	2	フィル

float xp 中心X座標

float yp 中心Y座標

float sx 始点X座標

float sy 始点Y座標

float angle 円弧の角度 (degree)

int col 書き込みデータ。ビットマップ画面への書き込みデータについては、PutLine() コマンドの詳細情報を参照して下さい。

□ 対応画面タイプ/画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	△R	○	○	○	—	○	○	○

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

1 0 画面空き領域、または範囲外 (不当画面番号エラー)、画面属性不正
 2 1 描画モード設定範囲外
 4 8 動的メモリ確保エラー
 — エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

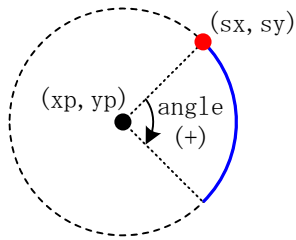
□ 機能

(x_p, y_p) を中心、(s_x, s_y) を始点とする角 $angle$ の円弧を描画します。なお、パラメータがフロート型である以外は `PutArc()` と同じです。

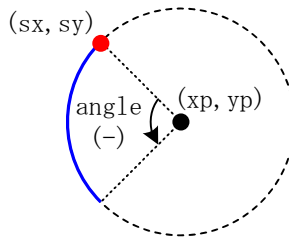
図形描画コマンドの描画は、ローカルCPU (SH) で実行します。コマンド実行中はSHのキャッシュに一時的に蓄えられるため、実際の画像メモリ (IM) には書き込まれていない部分が出てきます。RefreshImg() コマンドにより描画が全て終了した時点で、そのキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。

円弧の中心と始点、描画角度の関係を示します。

$angle \geq 0$
(時計回り描画)



$angle < 0$
(反時計回り描画)



尚、描画モードが `DRAW_FILL` のとき $angle$ が 180° を超えると正しく描画できません。

3.4.13 楕円描画処理（PutEllipse）

□ C言語API

```
int ret = PutEllipse( IMGID ImgID , enum IPDrawLineMode mode , float cx , float cy ,
                    float width , float height , float angle , int col );
```

□ パラメータ

IMGID ImgID 描画面の画面番号

enum IPDrawLineMode mode 描画モード

描画モード	対応数	説明
DRAW_LINE	0	線描画
DRAW_FILL	2	フィル

float cx 楕円の中心X座標。

float cy 楕円の中心Y座標。

float width 楕円を内接する長方形の幅（5～）。

float height 楕円を内接する長方形の高さ（5～）。

float angle 回転角度（°）。

int col 書き込みデータ。ビットマップ画面への書き込みデータについては、PutLine() コマンドの詳細情報を参照して下さい。

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	△R	○	○	○	—	○	○	○

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

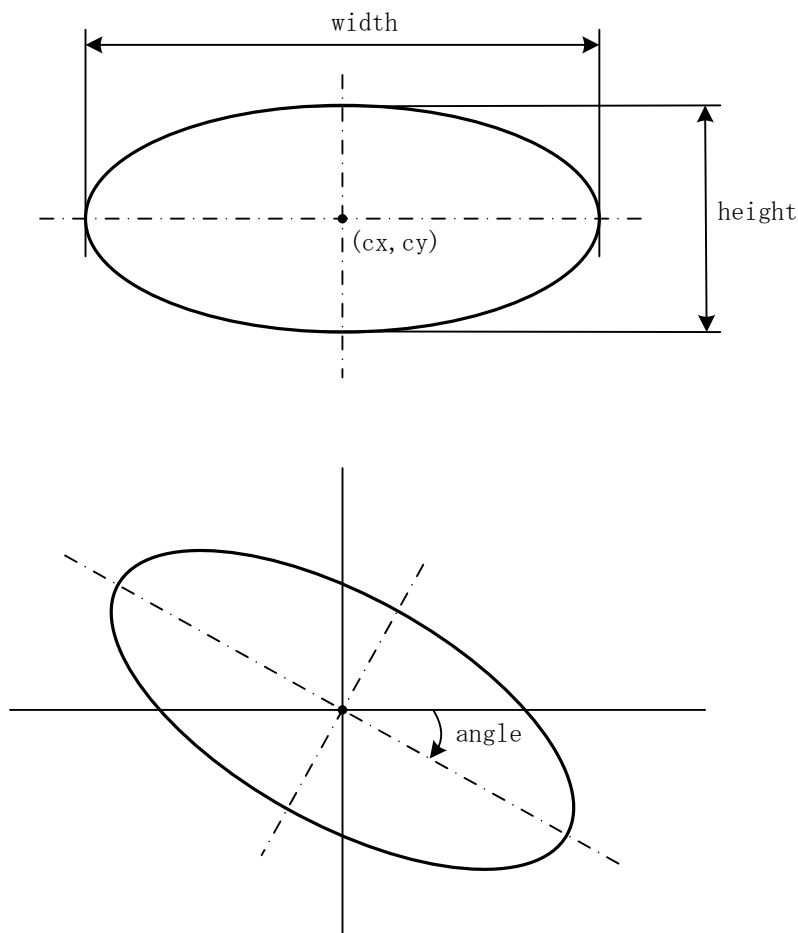
1 0 画面空き領域、または範囲外（不当画面番号エラー）、画面属性不正
2 1 描画モード設定範囲外
4 5 パラメータの設定条件違反
4 8 動的メモリ確保エラー
— エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

楕円描画処理を行います。

文字描画コマンドの描画は、ローカルCPU（SH）で実行します。コマンド実行中はSHのキャッシュに一時的に蓄えられるため、実際の画像メモリ（IM）には書き込まれていない部分が出てきます。RefreshImg（）コマンドにより描画が全て終了した時点で、そのキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。



本コマンドでは、以下の条件が1つでも成り立つ場合、パラメータの設定条件違反（エラーコード：45）が発生します。

(条件1) $b * ((2 * (\text{long})x\_radius) - 1) < 2 * a$

(条件2) $2 * b \geq a * ((2 * (\text{long})y\_radius) - 1)$

x_radius : 楕円の半径X方向長

y_radius : 楕円の半径Y方向長

a : 楕円の半径X方向長の2乗

b : 楕円の半径Y方向長の2乗

3.4.14 カナ漢字文字列の描画(シフトJIS) (PutString)

□ C言語API

```
int ret = PutString( IMGID ImgID , char *string , int mode , int sx , int sy , int col);
```

□ パラメータ

IMGID ImgID 描画面面の画面番号

char *string 文字列テーブルを返すアドレス

int mode 文字描画モード

31	16	15	12	11	8	7	4	3	0
—			Vsize		Hsize		描画モード		

※Hsize、VsizeはそれぞれX方向、Y方向の文字倍率です。

Hsize, Vsize	対応数	内容
STR_SIZE_1	0	X (Y) 方向を倍率 1 倍で描画
STR_SIZE_2	1	X (Y) 方向を倍率 2 倍で描画
STR_SIZE_3	2	X (Y) 方向を倍率 3 倍で描画
STR_SIZE_4	3	X (Y) 方向を倍率 4 倍で描画

文字描画モード	対応数	内容
BACKCOL_NOWRITE	0	文字部分のみcolF色で描画
BACKCOL_NOWRITE_REVERSE	1	文字背景のみcolB色で描画
BACKCOL_WRITE	2	文字部分をcolF色、背景をcolBで描画
BACKCOL_WRITE_REVERSE	3	文字部分をcolB色、背景をcolFで描画

int sx 開始X座標

int sy 開始Y座標

int col 書き込みデータ (詳細説明を参照)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	△R	○	○	○	—	○	○	○

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

1 0 画面空き領域、または範囲外（不当画面番号エラー）、画面属性不正
 2 1 文字列テーブルを返すアドレスが不正
 2 2 文字描画モード設定範囲外
 — エラーリセットコマンド（ClearIPErrors）未発行

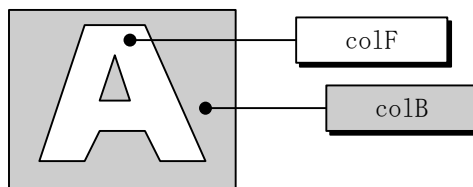
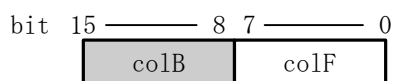
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

指定位置にカナ漢字文字列を描画します。カナ漢字文字列はシフトJISに対応しています。

文字描画コマンドの描画は、ローカルCPU（SH）で実行します。コマンド実行中はSHのキャッシュに一時的に蓄えられるため、実際の画像メモリ（IM）には書き込まれていない部分が出てきます。RefreshImg（）コマンドにより描画が全て終了した時点で、そのキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。

●書き込みデータ col のビット構成



PutString(ImgID, "漢字が使用できます", BACKCOL_NOWRITE, 128, 64, 15)

PutString(ImgID, "半角かなも使用できます", BACKCOL_NOWRITE, 128, 80, 15)

第4章 ダウンロードモジュール制御コマンド

4.1 モジュールサポート

4.1.1 ダウンロードモジュールのロード (LoadIPDLM)

□ C言語API

```
int ret = LoadIPDLM(char *pFileName, int mode, char *pModuleName,
                    IPDLMSymbolTbl *pInOpt, IPDLMEntryInf *pOutInf);
```

□ パラメータ

char	*pFileName	ロードするファイルへのパス ELF/DWARF形式のアブソリュートロードモジュールファイル (拡張子: ABS) へのパスを指定して下さい。
int	mode	モード (未使用) 0 を指定して下さい。
char	*pModuleName	モジュール名 ロード領域の管理に使用します。” NULL ” を指定すると pFileNameに指定したパスから拡張子を除いたファイル名 がモジュール名となります。
IPDLMSymbolTbl	*pInOpt	ダウンロードモジュール情報構造体を返すアドレス
IPDLMEntryInf	*pOutInf	ダウンロードモジュールエントリ情報構造体を返すアドレス pInOptで指定したシンボルのアドレス、ロード領域の先頭 アドレス、サイズが格納されます。

□ リターンパラメータ

int	ret	正常終了 (0) またはエラーコード
IPDLMEntryInf	*pOutInf	ダウンロードモジュールエントリ情報を格納したアドレス

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ パケットの構造

typedef struct {		
int	style;	シンボル形式 IPDLM_SYMBOL_C_STYLE (0) C言語形式 pSymbolで指定したシンボル名に” _ ” を添付してシンボル を検索します。 IPDLM_SYMBOL_ASM_STYLE (1) アセンブラ形式 pSymbolで指定したシンボル名でシンボルを検索します。
int	count;	シンボルの数
char	**pSymbol;	シンボル名設定領域のアドレス アドレス出力を行うシンボル名を設定して下さい。この領 域に設定するシンボル名の数 count で指定した数よりも 少ない場合、動作の保障はいたしません。
}	IPDLMSymbolTbl;	
typedef struct {		
unsigned long	*pAdr;	シンボルアドレスを格納したアドレス pSymbolで指定したシンボル名に対応したアドレスが格納 されます。シンボルが見つからない場合は0が出力され ます。
unsigned long	SecAdr;	ロード領域先頭アドレス このアドレスからSecSizバイトの領域がモジュール名で管 理されます。
unsigned long	SecSiz;	ロード領域サイズ サイズは4バイトアラインになります。
}	IPDLMEntryInf;	

□ エラーコード

- 1 7	パラメータエラー
- 1 0 0	ドライバインタフェースエラー(PC側のみ)
- 1 0 3	ファイルオープンエラー
- 1 0 4	ファイルタイプエラー (ELF/DWARF形式以外のファイルを指定)
- 1 0 5	ファイルアクセスエラー
- 1 0 6	作業領域メモリ確保エラー
- 1 0 7	ダウンロード領域のメモリ確保エラー
- 1 0 8	ベリファイエラー

□ 機能

パソコン上のダウンロードモジュール（以下、DLM（DownLoadModule）と略します）ファイルを画像認識ボード上のメモリにロードします。

DLMファイルからプログラムデータの先頭アドレスとサイズを取得し、その領域をメモリ確保してロードを行います。

ロード領域はモジュール名で管理します。本コマンドによりロードした後、DLMをインテリジェントモジュール化やタスク化することができます。ロード領域は『UnloadIPDLM』コマンドでアンロードするまで保持されます。同一領域にDLMを再ロードする場合は必ず『UnloadIPDLM』コマンドを実行して領域を解放して下さい。一度ロードされた領域は『ResetIP』コマンドでボードをリセットするか、『UnloadIPDLM』コマンドで解放しないかぎり解放されません。

（注1）プログラムデータの先頭アドレスからサイズ分の領域内に、既にメモリ確保済みの領域がある場合はエラーになります。

（注2）DLMファイルが異なっても、既にロードしているモジュール名と同一のモジュール名を指定するとエラーになります。

（注3）『InitIP』コマンドや『OpenIPDev』コマンドで初期化を行ってもロード領域は保持されています。

（注4）ダウンロードモジュール情報（pInOpt）、ダウンロードモジュールエントリ情報（pOutInf）には ” NULL ” を指定することもできます。

pInOpt	pOutInf	内容
NULL 以外	NULL 以外	・ロード領域先頭アドレス、サイズ、シンボルアドレスを出力 ・countが0、または pSymbolが ” NULL ” の場合、エラー
NULL 以外	NULL	エラー
NULL	NULL 以外	ロード領域先頭アドレス、サイズのみ出力
NULL	NULL	出力なし

□ サンプルコード

(1) DLMをロードし、インテリジェントモジュール化して実行する

```

void chapter_xx_xx( void )
{
    int          ret;                /* 関数戻り値 */
    IPErrorTbl   errTbl;            /* エラー情報テーブル */
    char         *filename = "intelimod.abs"; /* ファイルパス */
    char         *modulename = NULL; /* モジュール名 */
    int          mode = 0;          /* モード */
    IPDLMSymbolTbl lnOpt;           /* ダウンロードモジュール情報構造体 */
    IPDLMEntryInf OutInf;           /* ダウンロードモジュールエントリ情報構造体 */
    unsigned long symadr[3];        /* シンボルアドレス */
    char         *symbol[] = {      /* シンボル名 */
        "inteli_module0",
        "inteli_module1",
        "inteli_module2"
    };
    MODULE_PARAM_TBL prmtbl;        /* モジュールパラメータテーブル */

    /* 変数初期化 */
    memset( &lnOpt, 0, sizeof( lnOpt ) );
    memset( &OutInf, 0, sizeof( OutInf ) );
    memset( symadr, 0, sizeof( symadr ) );

    /* ダウンロードモジュール情報設定 */
    lnOpt.style = IPDLM_SYMBOL_C_STYLE;
    lnOpt.count = 3;
    lnOpt.pSymbol = symbol;

    /* ダウンロードモジュールエントリ情報設定 */
    OutInf.pAdr = symadr;

    /* ダウンロードモジュールのロード */
    ret = LoadIPDLM( filename, mode, modulename, &lnOpt, &OutInf );
    if( ret ){
        ReadIPErrorTable( &errTbl );
        printf( "errTbl.ErrorCode = %d\n", errTbl.ErrorCode );
        return;
    }

    /* インテリジェントモジュールの登録 */
    ret = RegistInteliIPModule( MODULE_0, OutInf.pAdr[0], 0 );
    ret = RegistInteliIPModule( MODULE_1, OutInf.pAdr[1], 0 );
    ret = RegistInteliIPModule( MODULE_2, OutInf.pAdr[2], 0 );

    /* インテリジェントモジュールの実行 */
    InitIPParamTable( &prmtbl, 1, INTELI_MODULE, 0, 0 );
    SetIntegerParam( &prmtbl, PARAM_1, 0x01234567 );
    ExecuteInteliIPModule( MODULE_0, &prmtbl, NULL );

    InitIPParamTable( &prmtbl, 2, INTELI_MODULE, 0, 0 );
    SetIntegerParam( &prmtbl, PARAM_1, 0x12345678 );
    SetIntegerParam( &prmtbl, PARAM_2, 0x23456789 );
    ExecuteInteliIPModule( MODULE_1, &prmtbl, NULL );

    InitIPParamTable( &prmtbl, 3, INTELI_MODULE, 0, 0 );
    SetIntegerParam( &prmtbl, PARAM_1, 0x34567890 );
    SetIntegerParam( &prmtbl, PARAM_2, 0x45678901 );
    SetIntegerParam( &prmtbl, PARAM_3, 0x56789012 );
    ExecuteInteliIPModule( MODULE_2, &prmtbl, NULL );

    /* ダウンロードモジュールのアンロード */
    UnloadIPDLM( "intelimod" );

    return;
}

```

(2) DLMをロードし、タスク化して実行する

```

void chapter_xx_xx( void )
{
    int          ret;                /* 関数戻り値 */
    IPErrorTbl   errTbl;            /* エラー情報テーブル */
    char          *filename = "piotask.abs"; /* ファイルパス */
    char          *modulename = NULL; /* モジュール名 */
    int          mode = 0;          /* モード */
    IPDLMSymbolTbl lnOpt;          /* ダウンロードモジュール情報構造体 */
    IPDLMEEntryInf OutInf;         /* ダウンロードモジュールエントリ情報構造体 */
    unsigned long symadr[2];        /* シンボルアドレス */
    char *symbol[] = {
        "pioint0_task_init",
        "pioint0_task"
    };
};
MODULE_PARAM_TBL prmtbl;          /* モジュールパラメータテーブル */
int          tskid;               /* タスクID */
CREATE_TASK_TBL iptsk;           /* タスク生成情報テーブル */
INT_DEVICE_OBJ intobj;           /* 割り込みデバイスオブジェクト */

/* 変数初期化 */
memset( &lnOpt, 0, sizeof( lnOpt ) );
memset( &OutInf, 0, sizeof( OutInf ) );
memset( symadr, 0, sizeof( symadr ) );

/* ダウンロードモジュール情報設定 */
lnOpt.style = IPDLM_SYMBOL_C_STYLE;
lnOpt.count = 2;
lnOpt.pSymbol = symbol;

/* ダウンロードモジュールエントリ情報設定 */
OutInf.pAdr = symadr;

/* ダウンロードモジュールのロード */
ret = LoadIPDLM( filename, mode, modulename, &lnOpt, &OutInf );
if( ret ){
    ReadIPErrorTable( &errTbl );
    printf( "errTbl.ErrorCode = %d\n", errTbl.ErrorCode );
    return;
}

/* タスクの生成 */
iptsk.task_addr = 0;
iptsk.priority = TASK_PRI_NORMAL;
iptsk.pbuff_size = 0x1000;
iptsk.stack_size = 0x1000;
iptsk.task_opt = 0;
iptsk.param_opt = 1;
tskid = CreateIPTask( &iptsk );

/* 割り込みオブジェクトの生成 */
intobj.intdev = INTDEV_PIO;
intobj.evtpri = 0;
intobj.intbit = 0;
CreateInterruptLink( &intobj, tskid, INTEVENT_WAKEUP, 0 );

/* タスクの登録 */
RegistIPTask( tskid, OutInf.pAdr[0], OutInf.pAdr[1], 0 );

/* 割込モジュールのサービス開始 (pioint0_task_init) */
InitIPParamTable( &prmtbl, 0, PIOINT_MODULE, tskid, 0 );
StartIPTaskwithParam( tskid, &prmtbl );
Sleep(1000); /* タスク終了ウェイト 1000ms */

/* 割込モジュールの実行 (pioint0_task) */
WakeupIPTaskwithParam( tskid[0], &prmtbl, SELF_WAKEUP );
Sleep(1000); /* タスク終了ウェイト 1000ms */

/* タスクの削除 */
DeleteIPTask( tskid );
UnloadIPDLM( "piotask" );

return;
}

```

前述のサンプルコード(1)(2)でロードするDLMの例を以下に示します。

(1)のDLM

DLMファイル intelimod.abs
 セクション名 : P,C
 アドレス : 0x08800000

```
void inteli_module0(int p1)
{
    printf("Inteli Module P1:%x\n", p1);
}

void inteli_module1(int p1, int p2)
{
    printf("Inteli Module P1:%x P2:%x\n", p1, p2);
}

void inteli_module2(int p1, int p2, int p3)
{
    printf("Inteli Module P1:%x P2:%x P3:%x\n", p1, p2, p3);
}
```

(2)のDLM

```
void pioint0_task_init(int p1)
{
    printf("PIO_INT0_TASK Init P1:%x\n", p1);
}

void pioint0_task(int p1, int p2)
{
    printf("PIO_INT0 Triggered P1:%x P2:%x\n", p1, p2);
}
```


4.1.2 ダウンロードモジュールのアンロード (UnloadIPDLM)

□ C言語API

```
int ret = UnloadIPDLM(char *pModuleName);
```

□ パラメータ

char	*pModuleName	モジュール名 『LoadIPDLM』コマンドで指定したモジュール名を指定 して下さい。大文字と小文字は区別されます。

□ リターンパラメータ

int	ret	正常終了 (0) またはエラーコード

□ エラーコード

- 1 7	パラメータエラー
- 2 5	コンテキストエラー (許可されていないシステム状態からの呼び出し)
- 2 8	V P - A x ダウンロードシステムが終了済み
- 4 1	V P - A x ダウンロードシステムにモジュール名が登録されていない

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

モジュール名で指定したロード領域をアンロードします。pModuleNameで指定したモジュール名がロードされていない場合や既にアンロードされている場合は、エラーになります。

(注) DLMをロードし、インテリジェントモジュール化やタスク化して実行している場合は終了処理をしてから本コマンドを実行して下さい。アンロード後はロード領域のメモリが解放されるため、プログラムデータが上書きされて予期しない動作をする危険があります。

4.1.3 モジュールのパラメータテーブルの初期化（InitIPParamTable）

□ C言語API

```
int ret = InitIPParamTable( MODULE_PARAM_TBL *tbl , int ParamCount , enum IPModuleType type ,
                           int taskID , int opt );
```

□ パラメータ

MODULE_PARAM_TBL *tbl モジュールパラメータテーブル構造体を返すアドレス
MODULE_PARAM_TBLテーブルの領域をあらかじめ確保して下さい。

int ParamCount パラメータ数
0～32個のパラメータをオンボードCPUにC言語のパラメータとして
わたすことができます。

enum IPModuleType type ターゲットのモジュールタイプを指定します。

モジュールタイプ	対応数	説明
INTELI_MODULE	0	インテリジェントモジュール
PIOINT_MODULE	1	P I O割込起動モジュール

int taskID 割込モジュールのタスクID
CreateIPTask() コマンドで生成したタスクIDを設定して下さい。
設定範囲は1～64です。インテリジェントモジュールの場合は
常に「0」を設定して下さい。

int opt オプション（現在未使用）。「0」を設定して下さい。

□ リターンパラメータ

int ret 正常終了（0）またはエラーコード
MODULE_PARAM_TBL *tbl 初期化したモジュールパラメータテーブル構造体を格納したアドレス

□ パケットの構造

```
typedef struct {
    unsigned long    module_type;    // モジュールタイプ
    unsigned long    task_id;        // タスクID
    unsigned long    param_tbl;      // パラメータテーブルアドレス
    unsigned long    pbuff_addr;     // パラメータバッファアドレス
    unsigned long    pbuff_size;     // パラメータバッファサイズ
    unsigned long    funcNo;         // モジュール番号
    unsigned long    count;          // パラメータ数
    unsigned long    format;         // パラメータ型
    unsigned long    reserve;        // 予約
    unsigned long    param[32];      // パラメータ
} MODULE_PARAM_TBL;
```

□ エラーコード

－1 7 パラメータエラー
－3 3 タスク情報領域未確保
－3 4 タスクID不正
－4 1 内部オブジェクトエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

インテリジェントモジュール実行コマンド（ExecuteInteriIPModule()）及びP I O割込モジュール実行コマンド（StartIPTaskwithParam(), WakeupIPTaskwithParam(), WakeupIPTaskExt()）で使用するモジュールパラメータテーブルの初期化を行います。モジュールパラメータテーブルとは、オンボードCPU側のプログラムにC言語関数の仕様でわたされるパラメータを設定するためのテーブルです。

4.1.4 整数型パラメータのセット (SetIntegerParam)

□ C言語API

```
int ret = SetIntegerParam( MODULE_PARAM_TBL *tbl , enum IPModuleParamNo paramNo , int val );
```

□ パラメータ

MODULE_PARAM_TBL *tbl モジュールパラメータテーブル構造体を返すアドレス
MODULE_PARAM_TBLテーブルの領域をInitIPParamTable() コマンドで
初期化して下さい。

enum IPModuleParamNo paramNo 設定するパラメータの順番

パラメータの順番	対応数	説明
PARAM_1	0	第1パラメータ
PARAM_2	1	第2パラメータ
PARAM_3	2	第3パラメータ
PARAM_32	31	第32パラメータ

int val モジュールパラメータテーブルにセットする整数

□ リターンパラメータ

int ret 正常終了 (0) またはエラーコード

□ エラーコード

- 17 パラメータエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

整数型のパラメータをモジュールパラメータテーブルにセットします。

4.1.5 実数型パラメータのセット (SetFloatParam)

□ C言語API

```
int ret = SetFloatParam( MODULE_PARAM_TBL *tbl , enum IPModuleParamNo paramNo , float val );
```

□ パラメータ

MODULE_PARAM_TBL *tbl モジュールパラメータテーブル構造体返すアドレス
MODULE_PARAM_TBLテーブルの領域をInitIPParamTable() コマンドで初期化して下さい。

enum IPModuleParamNo paramNo 設定するパラメータの順番

パラメータの順番	対応数	説明
PARAM_1	0	第1パラメータ
PARAM_2	1	第2パラメータ
PARAM_3	2	第3パラメータ
PARAM_32	31	第32パラメータ

float val モジュールパラメータテーブルにセットする整数

□ リターンパラメータ

int ret 正常終了 (0) またはエラーコード

□ エラーコード

- 17 パラメータエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

実数型のパラメータをモジュールパラメータテーブルにセットします。

4.1.6 配列データ(入力)パラメータのセット (SetParamTable)

□ C言語API

```
int ret = SetParamTable( MODULE_PARAM_TBL *tbl , enum IPModuleParamNo paramNo , void *pctbl ,
                        int num );
```

□ パラメータ

MODULE_PARAM_TBL *tbl モジュールパラメータテーブル構造体を返すアドレス
MODULE_PARAM_TBLテーブルの領域をInitIPParamTable() コマンドで
初期化して下さい。

enum IPModuleParamNo paramNo 設定するパラメータの順番

パラメータの順番	対応数	説明
PARAM_1	0	第1パラメータ
PARAM_2	1	第2パラメータ
PARAM_3	2	第3パラメータ
PARAM_32	31	第32パラメータ

void *pctbl P C側のデータが格納されている配列を返すアドレス
4バイト単位に領域を確保して下さい。

int num セットする配列のバイト数。4バイト単位に設定して下さい。

□ リターンパラメータ

int ret 正常終了 (0) またはエラーコード

□ エラーコード

－ 1 7 パラメータエラー
－ 5 8 確保しようとしている容量がオンボードC P Uのパラメータ領域を越えた
－ 1 0 0 ドライバインタフェースエラー(PC側のみ)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

P C上にある配列データをオンボードC P Uのプログラムにパラメータとして引き渡します。オンボードC P Uのプログラムは、たんなる配列のポインタ渡しで受け取ることができます。

本コマンドにより、配列データ (入力) の領域をオンボードC P Uシステムメモリに確保し、その配列データを転送します。そして、確保した領域のポインタをモジュールパラメータテーブルにセットします。

本コマンドでは、オンボードC P Uシステムメモリにパラメータ領域としてあらかじめ確保されている領域から確保します。そのため、パラメータ領域の容量を超える領域は確保できません。パラメータ領域の容量は約64 Kバイトです。また、SetParamTableExt() コマンドでは、その制限はありません。

4.1.7 配列データ(出力)パラメータ領域の確保 (AllocParamTable)

□ C言語API

```
int ret = AllocParamTable( MODULE_PARAM_TBL *tbl , enum IPModuleParamNo paramNo , int num );
```

□ パラメータ

MODULE_PARAM_TBL *tbl モジュールパラメータテーブル構造体を返すアドレス
MODULE_PARAM_TBLテーブルの領域をInitIPParamTable() コマンドで
初期化して下さい。

enum IPModuleParamNo paramNo 設定するパラメータの順番

パラメータの順番	対応数	説明
PARAM_1	0	第1パラメータ
PARAM_2	1	第2パラメータ
PARAM_3	2	第3パラメータ
PARAM_32	31	第32パラメータ

int num セットする配列のバイト数。4バイト単位に設定して下さい。

□ リターンパラメータ

int ret 正常終了 (0) またはエラーコード

□ エラーコード

— 1 7 パラメータエラー
— 5 8 確保しようとしている容量がオンボードCPUのパラメータ領域を越えた

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

配列データ (入力) の領域をオンボードCPUシステムメモリに確保します。本コマンドでは、オンボードCPUシステムメモリにパラメータ領域としてあらかじめ確保されている領域から確保します。そのため、パラメータ領域の容量を超える領域は確保できません。パラメータ領域の容量は約64Kバイトです。また、AllocParamTableExt() コマンドでは、その制限はありません。

GetParamTable() コマンドでは、オンボードCPUのプログラムからPC上に確保した領域にデータを転送します。その場合、オンボードCPUのプログラムは、たんなる配列のポインタ渡しで出力することができます。AllocParamTable() コマンドはその配列データ (入力) の領域をオンボードCPUシステムメモリに確保するわけです。

この機能は、AllocParamTable() コマンドとGetParamTable() コマンドを組み合わせで使用します。通常、本コマンドにより、配列データ (入力) の領域をオンボードCPUシステムメモリに確保し、確保した領域のポインタをモジュールパラメータテーブルにセットします。そして、インテリジェントモジュール及びPIO割込モジュールを実行後、GetParamTable() コマンドでそのオンボードCPUシステムメモリの配列データをPC上のメモリに転送します。

4.1.8 配列データ(出力)パラメータの読み出し (GetParamTable)

□ C言語API

```
int ret = GetParamTable( MODULE_PARAM_TBL *tbl , enum IPModuleParamNo paramNo , void *pctbl ,
                        int num );
```

□ パラメータ

MODULE_PARAM_TBL *tbl モジュールパラメータテーブル構造体を返すアドレス
MODULE_PARAM_TBLテーブルの領域をInitIPParamTable() コマンドで
初期化して下さい。

enum IPModuleParamNo paramNo 設定するパラメータの順番

パラメータの順番	対応数	説明
PARAM_1	0	第1パラメータ
PARAM_2	1	第2パラメータ
PARAM_3	2	第3パラメータ
PARAM_32	31	第32パラメータ

void *pctbl オンボードCPUシステムメモリからデータを読み出すPC側の
データ領域を返すアドレス。4バイト単位に領域を確保して下さい。

int num 読み出しを行うバイト数。4バイト単位に設定して下さい。

□ リターンパラメータ

int ret 正常終了 (0) またはエラーコード

void *pctbl オンボードCPUシステムメモリからデータを読み出す
PC側のデータ領域を格納したアドレス。

□ エラーコード

- 1 7 パラメータエラー
- 1 0 0 ドライバインタフェースエラー(PC側のみ)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

オンボードCPUのプログラムからAllocParamTable() 及びAllocParamTableExt() コマンドでPC上に確保した領域にデータを転送します。その場合、オンボードCPUのプログラムは、たんなる配列のポインタ渡しで出力することができます。

この機能は、AllocParamTable() コマンドとGetParamTable() コマンドを組み合わせで使用します。通常、AllocParamTable() 及びAllocParamTableExt() コマンドにより、配列データ (入力) の領域をオンボードCPUシステムメモリに確保し、確保した領域のポインタをモジュールパラメータテーブルにセットします。そして、インテリジェントモジュール及びPIO割込モジュールを実行後、本コマンドでそのオンボードCPUシステムメモリの配列データをPC上のメモリに転送します。

4.1.9 配列データ(入力)パラメータのセット (SetParamTableExt)

□ C言語API

```
int ret = SetParamTableExt( MODULE_PARAM_TBL *tbl , enum IPModuleParamNo paramNo ,
                           unsigned long iptbl_addr , void *pctbl , int num );
```

□ パラメータ

MODULE_PARAM_TBL *tbl モジュールパラメータテーブル構造体を返すアドレス
MODULE_PARAM_TBLテーブルの領域をInitIPParamTable() コマンドで
初期化して下さい。

enum IPModuleParamNo paramNo 設定するパラメータの順番

パラメータの順番	対応数	説明
PARAM_1	0	第1パラメータ
PARAM_2	1	第2パラメータ
PARAM_3	2	第3パラメータ
PARAM_32	31	第32パラメータ

unsigned long iptbl_addr 書き込みを行うオンボードCPUシステムメモリのアドレス
4バイト単位でアドレスして下さい。

void *pctbl PC側のデータが格納されている配列を返すアドレス
4バイト単位に領域を確保して下さい。

int num セットする配列のバイト数。4バイト単位に設定して下さい。

□ リターンパラメータ

int ret 正常終了 (0) またはエラーコード

□ エラーコード

- 1 7 パラメータエラー
- 1 0 0 ドライバインタフェースエラー(PC側のみ)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

PC上にある配列データをオンボードCPUのプログラムにパラメータとして引き渡します。オンボードCPUのプログラムは、たんなる配列のポインタ渡しで受け取ることができます。

本コマンドにより、指定されたオンボードCPUシステムメモリアドレスからPC側の配列データを転送します。そして、指定した領域のアドレスをモジュールパラメータテーブルにセットします。

SetParamTable() コマンドでは、オンボードCPUシステムメモリにパラメータ領域としてあらかじめ確保されている領域から確保します。そのため、64Kバイトの容量を超える領域は確保できませんがSetParamTableExt() コマンドでは、その制限はありません。任意の領域をユーザが確保し管理する必要があります。

4.1.10 配列データ(出力)パラメータ領域の確保 (AllocParamTableExt)

□ C言語API

```
int ret = AllocParamTableExt( MODULE_PARAM_TBL *tbl , enum IPModuleParamNo paramNo ,
                             unsigned long iptbl_addr , int num );
```

□ パラメータ

MODULE_PARAM_TBL *tbl モジュールパラメータテーブル構造体を返すアドレス
MODULE_PARAM_TBLテーブルの領域をInitIPParamTable() コマンドで
初期化して下さい。

enum IPModuleParamNo paramNo 設定するパラメータの順番

パラメータの順番	対応数	説明
PARAM_1	0	第1パラメータ
PARAM_2	1	第2パラメータ
PARAM_3	2	第3パラメータ
PARAM_32	31	第32パラメータ

unsigned long iptbl_addr 書き込みを行うオンボードCPUシステムメモリのアドレス
4バイト単位のアドレスして下さい。

int num セットする配列のバイト数。4バイト単位に設定して下さい。

□ リターンパラメータ

int ret 正常終了 (0) またはエラーコード

□ エラーコード

- 17 パラメータエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

配列データ (入力) の領域をオンボードCPUのプログラムにパラメータとして引き渡します。オンボードCPUのプログラムは、たんなる配列のポインタ渡しで受け取ることができます。SetParamTableExtコマンドは、指定されたオンボードCPUシステムメモリアドレスからPC側の配列データを転送しますが、本コマンドは転送しないで指定された領域のアドレスをモジュールパラメータテーブルにセットします。

AllocParamTable() コマンドでは、オンボードCPUシステムメモリにパラメータ領域としてあらかじめ確保されている領域から確保します。そのため、64Kバイトの容量を超える領域は確保できませんがAllocParamTableExt() コマンドでは、その制限はありません。任意の領域をユーザが確保し管理する必要があります。

この機能は、AllocParamTableExt() コマンドとGetParamTable() コマンドを組み合わせて使用します。通常、本コマンドにより、配列データ (入力) の領域をオンボードCPUシステムメモリに確保し、確保した領域のポインタをモジュールパラメータテーブルにセットします。そして、インテリジェントモジュール及びPIO割込モジュールを実行後、GetParamTable() コマンドでそのオンボードCPUシステムメモリの配列データをPC上のメモリに転送します。

4.1.11 ボード側システムメモリデータリード(ReadIPMemData)

□ C言語API

```
int ret = ReadIPMemData( void *pPCMem, unsigned long shmem_addr, unsigned long Count,
int Mode );
```

□ パラメータ

void *pPCMem	P C側データ領域を返すアドレス
unsigned long shmem_addr	ボードC P U システムメモリアドレス
unsigned long Count	データ数 (アクセス数)
int Mode	転送モード
	0 : 4 B アクセス
	1 : 1 B アクセス
	2 : 2 B アクセス

□ リターンパラメータ

int	ret	正常終了 (0) またはエラーコード
void	*pPCMem	P C側データ領域を格納したアドレス

□ エラーコード

- 1 0 0	ドライバインタフェースエラー (PC側のみ)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ボード側システムメモリからデータを読み出し、P C側のデータ領域に書き込みます。

4.1.12 ボード側システムメモリデータライト(WriteIPMemData)

□ C言語API

```
int ret = WriteIPMemData( void *pPCMem, unsigned long shmem_addr, unsigned long Count,
int Mode );
```

□ パラメータ

void *pPCMem	P C側データ領域を返すアドレス
unsigned long shmem_addr	ボードC P U システムメモリアドレス
unsigned long Count	データ数 (アクセス数)
int Mode	転送モード
	0 : 4 Bアクセス
	1 : 1 Bアクセス
	2 : 2 Bアクセス

□ リターンパラメータ

int	ret	正常終了 (0) またはエラーコード

□ エラーコード

- 1 0 0	ドライバインタフェースエラー(PC側のみ)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

P C側のデータをボード側システムメモリに書き込みます。

4.2 インテリジェントモジュール制御

4.2.1 インテリジェントモジュールの登録 (RegistInteliIPModule)

□ C言語API

```
int ret = RegistInteliIPModule( int module , unsigned long module_addr , int opt );
```

□ パラメータ

int module 登録するインテリジェントモジュールのモジュール番号

モジュール番号	説明
0	モジュール番号 # 0
1	モジュール番号 # 1
2	モジュール番号 # 2
255	モジュール番号 # 2 5 5

unsigned long module_addr インテリジェントモジュールのアドレス。
オンボードCPU上のインテリジェントモジュールの
プログラムセクションのアドレスを指定して下さい。

int opt オプション。「0」を設定して下さい。

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

2 0 モジュールID設定範囲外
- エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

オンボードCPUで実行されるインテリジェントモジュールを登録します。登録したモジュールは、ExecuteInteliIPModule() コマンドで実行します。
インテリジェントモジュールは256個まで登録できます。

4.2.2 インテリジェントモジュールの実行 (ExecuteInteliIPModule)

□ C言語API

```
int ret = ExecuteInteliIPModule( int module , MODULE_PARAM_TBL *prmtbl , void *ovl);
```

□ パラメータ

int module 実行するインテリジェントモジュールのモジュール番号

モジュール番号	説明
0	モジュール番号 # 0
1	モジュール番号 # 1
2	モジュール番号 # 2
255	モジュール番号 # 2 5 5

MODULE_PARAM_TBL *prmtbl

モジュールのパラメータテーブルのポインタアドレス。初期化及び設定についての詳細は、InitIPParamTable(), SetIntegerParam(), SetFloatParam() コマンドなどのモジュールサポートコマンドを参照して下さい。

void *ovl NULLを設定して下さい。

□ リターンパラメータ

このコマンドのリターンコードは登録モジュールから返されるリターンコードです。型は int 型のみで、登録モジュールから返されるリターンコードがそれ以外の場合、正しくデータが渡されないので注意してください。

□ エラーコード

なし

□ 機能

RegistInteliIPModule() コマンドで登録したインテリジェントモジュールの実行を行います。インテリジェントモジュールは画像処理コマンドの一部として動作しますが、タイムアウトがないので注意が必要です。
このコマンドではインテリジェントモジュールにC言語の関数形式でパラメータを渡すことができます。

□ 機能

割込リンクオブジェクトの生成。本コマンドにより、P I Oなどの割込オブジェクトからの割込によりウェイクアップ等の動作を行うように設定します。

INT_DEVICE_OBJ のフォーマット及び内容を、下記に示します。

```
typedef struct {
    int          intdev;
    int          intbit;
    int          evtpri;
    int          opt;
} INT_DEVICE_OBJ;
```

割込デバイスオブジェクト	説明		
intdev	割込デバイス番号		
	デバイス番号	対応数	説明
	INTDEV_UNLINK	0	割込デバイスなし(現在指定不可)
	INTDEV_PIO	2	ボード上のP I O割込
intbit	割込デバイスの対象ビット。以下にデバイス番号に対する対応ビットを示します。		
	デバイス番号	対応数	説明
	INTDEV_PIO	0-15	ボード上のP I Oのビットに対応しています。
evtpri	「0」を設定して下さい。		
opt	オプション（「0」を指定）		

4.3.2 割込オブジェクトの削除 (DeleteInterruptLink)

□ C言語API

```
int ret = DeleteInterruptLink( INT_DEVICE_OBJ *obj );
```

□ パラメータ

INT_DEVICE_OBJ *obj 割込デバイスオブジェクト情報。
リンクを解除する割込デバイスの情報を設定します。

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

なし

□ 機能

CreateInterruptLink()により生成した割込リンクオブジェクトを削除します。
本コマンドにより、P I Oなどの割込オブジェクトからの割込による動作を解除します。

4.3.3 割込オブジェクトの有効化（ EnableInterruptObject ）

□ C言語API

```
int ret = EnableInterruptObject( INT_DEVICE_OBJ *obj , int opt );
```

□ パラメータ

INT_DEVICE_OBJ *obj	割込デバイスオブジェクト情報。 リンクを解除する割込デバイスの情報を設定します。
int opt	現在、未使用。「0」を設定して下さい。

□ リターンパラメータ

int	ret	正常終了（0）または異常終了（-1）

□ エラーコード

なし

□ 機能

本コマンドにより、P I Oなどの割込オブジェクトからの割込動作を有効にします。
「INT_DEVICE_OBJ」構造体の詳細は、CreateInterruptLink() コマンドの詳細情報を参照して下さい。

4.3.4 割込オブジェクトの無効化 (DisableInterruptObject)

□ C言語API

```
int ret = DisableInterruptObject( INT_DEVICE_OBJ *obj );
```

□ パラメータ

INT_DEVICE_OBJ *obj 割込デバイスオブジェクト情報。
リンクを解除する割込デバイスの情報を設定します。

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

なし

□ 機能

本コマンドにより、P I Oなどの割込オブジェクトからの割込による動作を無効にします。
「INT_DEVICE_OBJ」構造体の詳細は、CreateInterruptLink() コマンドの詳細情報を参照して下さい。

4.4 タスク制御(PC側)

4.4.1 ユーザタスクの生成 (CreateIPTask)

□ C言語API

```
int ret = CreateIPTask( CREATE_TASK_TBL *tbl );
```

□ パラメータ

CREATE_TASK_TBL *tbl タスク生成情報テーブル。
この情報テーブルに設定された情報をもとにタスクを生成します。

□ リターンパラメータ

int ret

> 0 : タスク I D
≤ 0 : 異常終了
- 5 1 ~ - 9 : μ ITRONサービスコールエラー

□ エラーコード

なし

□ 機能

ユーザタスクの生成を行います。タスクの生成に成功するとタスク ID をリターン値として返します。

CREATE_TASK_TBLのフォーマット及び内容を、下記に示します。

```
typedef struct {
    unsigned long    task_addr;
    int              priority;
    unsigned long    pbuff_size;
    unsigned long    stack_size;
    int              param_opt;
    int              task_opt;
} CREATE_TASK_TBL;
```

タスク情報生成テーブル	説明			
task_addr	タスクの先頭アドレス			
	task_addr	内容		
	0	割込起動モジュールに対してパラメータを引き渡す機能のあるフレームを使用します。この機能を選択した場合は、RegistIPTask()コマンドでモジュールのアドレスを設定して下さい。		
	0 以外	指定したモジュールを直接起動します。割込起動モジュールに対してパラメータを引き渡すことができませんが、RegistIPTask()コマンドでのモジュールのアドレス指定も必要ありません。		
priority	タスクのプライオリティ			
	プライオリティ	対応数	説明	
	TASK_PRI_HIGHEST	31	ノーマル－ 2	高い 低い
	TASK_PRI_ABOVE_NORMAL	32	ノーマル－ 1	
	TASK_PRI_NORMAL	33	ノーマル（初期値）	
	TASK_PRI_BELOW_NORMAL	34	ノーマル＋ 1	
	TASK_PRI_LOWEST	35	ノーマル＋ 2	
pbuff_size	パラメータバッファのサイズ（バイト）			
stack_size	スタックサイズ（バイト）			
param_opt	パラメータオプション			
	task_addr 情報に「0」を設定した場合、割込起動モジュールに対してパラメータを引き渡す機能を使用するかどうか指定します。task_addr 情報が「0」以外の場合は、「0」を指定して下さい。			
	param_opt	内容		
	0	パラメータを渡さない		
	1	パラメータを渡す		
task_opt	タスクオプション			
	未使用。0を設定して下さい。			

4.4.2 割込モジュールの登録 (RegistIPTask)

□ C言語API

```
int ret = RegistIPTask( int taskID , unsigned long init_task , unsigned long ppoint_task ,
                      int opt );
```

□ パラメータ

int taskID 登録する割込モジュールのタスクID。
CreateIPTask() コマンドで生成したタスクIDを設定して下さい。
設定範囲は1～64です。

unsigned long init_task 初期化モジュールのアドレス。
LoadIPDLM() で取得したシンボルアドレスを指定して下さい。
初期化モジュールが不要な場合、0を設定して下さい。

unsigned long ppoint_task PIOからの割込で起動されるモジュールのアドレス。
LoadIPDLM() で取得したシンボルアドレスを指定して下さい。
0を設定するとNOPモジュールが設定されます。

int opt オプション。通常は「0」を指定して下さい。

タスクオプション	対応数	説明
TASK_NO_OPTION	0x00000000	・ウェイクアップ要求をキューイングする。 ・スタートタスク実行後、ハード的に割込マスクを解除し、PIOからの割込によりタスクがウェイクアップされるようにします。
CANCEL_WAKEUP	0x00000001	タスク実行中にキューイングされたウェイクアップ要求をキャンセルします。
DISABLE_INTERRUPT	0x00000100	PIOからの割込をハード的にマスクし、割込によりタスクがウェイクアップされないようにします。

複数のオプションを指定する場合は、論理和をとって指定して下さい。

□ リターンパラメータ

int ret > 0 : タスクID
 ≤ 0 : 異常終了
 -51～-9 : μITRONサービスコールエラー

□ エラーコード

なし

□ 機能

オンボードCPUで実行される割込起動モジュールを登録します。登録するモジュールは、初期化部とPIOなどのデバイスからの割込で起動される部分の2つのペアです。

4.4.3 割込モジュールのサービス開始 (StartIPTaskwithParam)

□ C言語API

```
int ret = StartIPTaskwithParam( int taskID , MODULE_PARAM_TBL *prmtbl );
```

□ パラメータ

int taskID 割込モジュールのタスク I D。
CreateIPTask() コマンドで生成しRegistIPTask() コマンドで登録された
タスク I Dを設定して下さい。設定範囲は 1 ～ 6 4 です。

MODULE_PARAM_TBL *prmtbl モジュールのパラメータテーブルのポインタアドレス。
初期化及び設定についての詳細は、InitIPParamTable(), SetIntegerParam(),
SetFloatParam() コマンドなどのモジュールサポートコマンドを参照して下さい。

□ リターンパラメータ

int ret 0 : 正常終了
 - 1 : 異常終了
 - 5 1 ～ - 9 : μ ITRONサービスコールエラー

□ エラーコード

なし

□ 機能

RegistIPTask() コマンドで登録された割込初期化モジュール(init_task)を実行し、割込起動モジュール(pioint_task)に対するサービスを開始します。RegistIPTask() コマンドで登録したtaskIDに対応するinit_taskが実行された後、対応する割込のハード的なマスクが解除され、ウェイクアップコマンドでpioint_taskが起動される状態になります。

また、このコマンドではinit_taskにC言語の関数形式でパラメータをわたすことができます。

4.4.4 割込モジュールの実行（WakeupIPTaskwithParam）

□ C言語API

```
int ret = WakeupIPTaskwithParam( int taskID , MODULE_PARAM_TBL *prmtbl , int opt );
```

□ パラメータ

int taskID 割込モジュールのタスク I D。CreateIPTask() コマンドで生成し
RegistIPTask() コマンドで登録されたタスク I Dを設定して下さい。
設定範囲は 1 ～ 6 4 です。

MODULE_PARAM_TBL *prmtbl モジュールのパラメータテーブルのポインタアドレス。
初期化及び設定についての詳細は、InitIPParamTable(), SetIntegerParam(),
SetFloatParam() コマンドなどのモジュールサポートコマンドを参照して下さい。

int opt 起動オプション

起動オプション	説明
SELF_WAKEUP	自己起動します
PIO_WAKEUP	P I O 入力の割込で起動します

□ リターンパラメータ

int ret 0 : 正常終了
 - 1 : 異常終了
 - 5 1 ～ - 9 : μ ITRON サービスコールエラー

□ エラーコード

なし

□ 機能

本コマンドの起動オプション(opt)がSELF_WAKEUPの場合、RegistIPTask() コマンドで登録された割込起動モジュール(pioint_task)を自己起動します。また、本コマンドの起動オプション(opt)がPIO_WAKEUPの場合、RegistIPTask() コマンドで登録された割込起動モジュール(pioint_task)をtaskIDで示される P I O の割込により起動します。

また、このコマンドではpioint_taskに C 言語の関数形式でパラメータをわたすことができます。

4.4.5 タスクモジュールからの終了シグナルウェイト（WaitforIPTaskSignal）

□ C言語API

```
int ret = WaitforIPTaskSignal( int Signal , int Mode , int Tmo , WaiIPSigInf *pInf );
```

□ パラメータ

int Signal ユーザシグナル番号。
ローカルCPUからPCへのシグナルを送信する場合、ローカルCPU側からの SendSignaltoPC() コマンドの signal パラメータに対応するユーザシグナル番号に応答して本コマンドのウェイトが解除されます。

ユーザシグナル番号	対応数	説明
PCI_SIGNAL_USER_0	0	ユーザシグナル # 0
PCI_SIGNAL_USER_1	1	ユーザシグナル # 1
PCI_SIGNAL_USER_2	2	ユーザシグナル # 2
PCI_SIGNAL_USER_7	7	ユーザシグナル # 7

int Mode モード（現在未使用）。「0」を設定して下さい。

int Tmo タイムアウト時間（ms）。「0」を指定するとタイムアウトしません。

WaiIPSigInf *pInf 受信シグナル情報出力テーブル。
NULLを設定すると受信シグナル情報が出力されません。

□ リターンパラメータ

int ret 0 : 正常終了
 - 1 : 異常終了
 - 5 1 ~ - 9 : μ ITRONサービスコールエラー
 - 1 0 0 : ドライバインタフェースエラー(PC側のみ)

□ エラーコード

2 0 ユーザシグナル番号設定範囲外
2 1 モード設定範囲外

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

オンボードCPUの割込モジュールとタスクモジュールが終了シグナルを発信するまでウェイトします。オンボードCPUの割込モジュールは、SendSignaltoPC() コマンドで終了シグナルを発信し、割込モジュールの処理が終了したことをPCに伝えます。そのとき、SendSignaltoPC() で設定したCodeパラメータが、受信シグナル情報出力テーブル(WaiIPSigInf)の受信コードに反映されます。

また、1つのスレッドまたは1つのデバイスIDに対してシグナルポートは1つだけ設定可能です。複数のスレッド、複数のデバイスIDで同じシグナルポート番号は使用できません。

なお、SELF_WAKEUPモードのWakeupIPTaskwithParam() コマンドとWaitforIPTaskSignal() コマンドにより、タスクの制御を行うとWakeup動作とWait動作を単一のドライバ内で行うことができないため、WakeupIPTaskwithParam() コマンドとWaitforIPTaskSignal() コマンドの間に発生したオンボードCPUからのシグナルに対して応答する事ができず、そのスレッドは、別のスレッドからCancelIPTaskWait() コマンドを発行するまでデッドロック状態になりますので注意してください。

本コマンドの前に必ずReadyforWaitIPTaskSignalを実行してください。

WaiIPSigInf のフォーマット及び内容を、下記に示します。

```
typedef struct {
    long      Status;    /* 受信ステータス */
    long      Code;      /* 受信コード */
    long      Info[4];   /* 受信拡張情報 */
} WaiIPSigInf;
```

受信シグナル情報出力テーブル

受信シグナル情報	説明
Status	受信ステータス 0 : シグナル正常受信 1 : キャンセル 2 : タイムアウト 3 : システムダウン 4 : その他のエラー
Code	受信コード。SendSignaltoPC() で設定したCodeパラメータが反映されます。
Info[4]	受信拡張情報 (現在未使用のため、0 が出力されます。)

4.4.6 タスクモジュールからの終了シグナルウェイト準備 (ReadyforWaitIPTaskSignal)

□ C言語API

```
int ret = ReadyforWaitIPTaskSignal( int signal );
```

□ パラメータ

int signal ユーザシグナル番号。
ローカルCPUからPCへのシグナルを送信する場合、ローカルCPU側からの SendSignaltoPC() コマンドの signal パラメータに対応するユーザシグナル番号に応答して本コマンドのウェイトが解除されます。

ユーザシグナル番号	対応数	説明
PCI_SIGNAL_USER_0	0	ユーザシグナル # 0
PCI_SIGNAL_USER_1	1	ユーザシグナル # 1
PCI_SIGNAL_USER_2	2	ユーザシグナル # 2
PCI_SIGNAL_USER_7	7	ユーザシグナル # 7

□ リターンパラメータ

int ret 0 : 正常終了
 - 1 : 異常終了
 - 1 0 0 : ドライバインタフェースエラー

□ エラーコード

2 0 ユーザシグナル番号設定範囲外

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

SELF_WAKEUPモードのWakeupIPTaskwithParam() コマンドとWaitforIPTaskSignal() コマンドにより、タスクの制御を行うとWakeup動作とWait動作を単一のドライバ内で行うことができないため、WakeupIPTaskwithParam() コマンドとWaitforIPTaskSignal() コマンドの間に発生したオンボードCPUからのシグナルに対して応答する事ができず、そのスレッドは、デッドロック状態になります。そのような状態になることが予想されるアプリケーションではWakeupIPTaskwithParam() コマンドを発行する直前にReadyforWaitIPTaskSignal() コマンドを発行することでデッドロック状態になることを回避できます。

4.4.7 割込モジュールの実行2 (WakeupIPTaskExt)

□ C言語API

```
int ret = WakeupIPTaskExt( int taskID , MODULE_PARAM_TBL *prmtbl , int opt , void *ovl );
```

□ パラメータ

int taskID 割込モジュールのタスク I D。
CreateIPTask() コマンドで生成したタスク I Dを設定して下さい。
設定範囲は1～64です。

MODULE_PARAM_TBL *prmtbl モジュールのパラメータテーブルのポインタアドレス。
初期化及び設定についての詳細は、InitIPParamTable(), SetIntegerParam(),
SetFloatParam() コマンドなどのモジュールサポートコマンドを参照して下さい。

int opt 起動オプション

起動オプション	説明
SELF_WAKEUP	自己起動します
PIO_WAKEUP	P I O入力の割込で起動します

void *ovl N U L Lを設定して下さい。

□ リターンパラメータ

int ret 0 : 正常終了
 - 1 : 異常終了
 - 5 1 ～ - 9 : μ ITRONサービスコールエラー
 - 1 0 0 : ドライバインタフェースエラー

□ エラーコード

なし

□ 機能

機能的には、WakeupIPTaskwithParam() コマンドを起動オプション(opt)がSELF_WAKEUPで実行し、WaitforIPTaskSignal() コマンドで登録されたP I O割込起動モジュール(pioint_task)からのシグナルをウェイトする動作と同じです。しかし、本コマンドでは、Wakeup動作とWait動作を単一のドライバ内で行うため、WakeupIPTaskwithParam() コマンドとWaitforIPTaskSignal() コマンドの間に発生したオンボードC P Uからのシグナルに対しても確実に応答する事ができます。

複数のスレッドから本コマンドを同時に実行することはできません。

4.4.8 終了シグナルウェイトの解除 (CancelIPTaskWait)

□ C言語API

```
int ret = CancelIPTaskWait( int signal , int mode );
```

□ パラメータ

int signal	WaitforIPTaskSignal() コマンドでウェイトしているシグナルの番号。

ユーザシグナル番号	対応数	説明
PCI_SIGNAL_USER_0	0	ユーザシグナル # 0
PCI_SIGNAL_USER_1	1	ユーザシグナル # 1
PCI_SIGNAL_USER_2	2	ユーザシグナル # 2
PCI_SIGNAL_USER_7	7	ユーザシグナル # 7

int mode モード（現在未使用）。「0」を設定して下さい。

□ リターンパラメータ

```
int      ret          0 : 正常終了  
          -1 : 異常終了  
          -1000 : ドライバインタフェースエラー
```

❏ エラーコード

20 タスク ID 設定範囲外

機能

WaitforIPTaskSignal() コマンドでの終了シグナルウェイトを解除します。本コマンドは、WaitforIPTaskSignal() でウェイトしているスレッドと別のスレッドから発行することによりウェイトの解除を行います。WaitforIPTaskSignal コマンドタイムアウト無しで使用している場合、シグナルが受信されない状態でのアプリケーション終了処理に本コマンドを使用してWaitforIPTaskSignal() コマンドを確実に終了させる必要があります。また、WaitforIPTaskSignal() コマンドでオンボードCPUのPIO割込モジュールからの終了シグナルをウェイトしている場合、何らかの理由でオンボードCPUのPIO割込モジュールが応答できなくなった場合も本コマンドを使用してWaitforIPTaskSignal() コマンドを確実に終了させて下さい。

4.4.9 タスクモジュールのサービス開始 (StartIPTask)

□ C言語API

```
int ret = StartIPTask( int taskID );
```

□ パラメータ

int taskID タスクモジュールのタスク I D。
CreateIPTask() コマンドで生成したタスク I Dを設定して下さい。
設定範囲は1～64です。

□ リターンパラメータ

int ret 0 : 正常終了
 - 1 : 異常終了
 - 51～-9 : μ ITRONサービスコールエラー

□ エラーコード

なし

□ 機能

StartIPTaskwithParam() コマンドと同等の処理を行いますが、本コマンドでは、パラメータを設定することはできません。また、本コマンドを使用する場合は、必ずWakeupIPTask() コマンドをペアで使用して下さい。WakeupIPTaskwithParam() コマンドとペアで使用しないで下さい。

4.4.10 タスクモジュールの起動（WakeupIPTask）

□ C言語API

```
int ret = WakeupIPTask( int taskID );
```

□ パラメータ

int taskID タスクモジュールのタスク I D。
CreateIPTask() コマンドで生成したタスク I Dを設定して下さい。
設定範囲は 1 ～ 6 4 です。

□ リターンパラメータ

int ret 0 : 正常終了
 - 1 : 異常終了
 - 5 1 ～ - 9 : μ ITRON サービスコールエラー

□ エラーコード

なし

□ 機能

WakeupIPTaskwithParam() コマンドと同等の処理を行いますが、本コマンドでは、パラメータを設定することはできません。また、本コマンドを使用する場合は、必ずStartIPTask() コマンドをペアで使用して下さい。StartIPTaskwithParam() コマンドとペアで使用しないで下さい。

4.4.11 タスクモジュールのサービス終了 (TerminateIPTask)

□ C言語API

```
int ret = TerminateIPTask( int taskID );
```

□ パラメータ

```
int taskID
```

タスクモジュールのタスク I D。
CreateIPTask() コマンドで生成したタスク I Dを設定して下さい。
設定範囲は1～64です。

□ リターンパラメータ

```
int      ret
```

0 : 正常終了
- 1 : 異常終了
- 5 1 ~ - 9 : μ ITRONサービスコールエラー

□ エラーコード

なし

□ 機能

StartIPTaskwithParam() 及び StartIPTask() コマンドでサービスが開始されている P I O 割込モジュールとタスクモジュールのサービスを終了します。ただし、本コマンドは指定したタスクを強制終了させるため、処理の実行が途中で終了したりタスク内のリソースのクリーンアップ処理などが不完全で終了し、結果的に他のタスクに影響を与える場合がありますので注意してください。再びサービスを開始する場合は、StartIPTaskwithParam() 及び StartIPTask() コマンドを実行して下さい。

4.4.12 タスクモジュールサービスの再開 (ResumeIPTask)

□ C言語API

```
int ret = ResumeIPTask( int taskID );
```

□ パラメータ

```
int taskID
```

タスクモジュールのタスク I D。
CreateIPTask() コマンドで生成したタスク I Dを設定して下さい。
設定範囲は 1 ～ 6 4 です。

□ リターンパラメータ

```
int      ret
```

0 : 正常終了
- 1 : 異常終了
- 5 1 ～ - 9 : μ ITRONサービスコールエラー

□ エラーコード

なし

□ 機能

SuspendIPTask() コマンドで一時停止した P I O割込モジュールとタスクモジュールのサービスを再開します。

4.4.13 タスクモジュールサービスの一時停止 (SuspendIPTask)

□ C言語API

```
int ret = SuspendIPTask( int taskID );
```

□ パラメータ

int taskID タスクモジュールのタスク I D。
CreateIPTask() コマンドで生成したタスク I Dを設定して下さい。
設定範囲は0～64です。
TASK_SELF(0)を指定することにより、自タスクも指定できます。

□ リターンパラメータ

int ret 0 : 正常終了
 - 1 : 異常終了
 - 5 1 ～ - 9 : μ ITRONサービスコールエラー

□ エラーコード

なし

□ 機能

任意のP I O割込モジュールとタスクモジュールのサービスを一時停止します。
ResumeIPTask() コマンドでサービスを再開できます。

4.4.14 タスクモジュールの削除 (DeleteIPTask)

□ C言語API

```
int ret = DeleteIPTask( int taskID );
```

□ パラメータ

int taskID タスクモジュールのタスク I D。
CreateIPTask() コマンドで生成したタスク I Dを設定して下さい。
設定範囲は 1 ～ 6 4 です。

□ リターンパラメータ

int ret 0 : 正常終了
 - 1 : 異常終了
 - 5 1 ～ - 9 : μ ITRONサービスコールエラー

□ エラーコード

なし

□ 機能

P I O割込モジュールとタスクモジュールを削除します。

4.5 PCとの同期

4.5.1 PIO割込モジュールからのシグナル送信（ SendSignaltoPC ）

□ C言語API

```
int ret = SendSignaltoPC( int signal , int code , int opt );
```

□ パラメータ

int signal ローカルC P UからP C へのシグナルを送信する場合の番号。

ユーザシグナル番号	対応数	説明
PCI_SIGNAL_USER_0	0	ユーザシグナル # 0
PCI_SIGNAL_USER_1	1	ユーザシグナル # 1
PCI_SIGNAL_USER_2	2	ユーザシグナル # 2
PCI_SIGNAL_USER_7	7	ユーザシグナル # 7

int code 終了コード。ここでセットした値がWaitforIPTaskSignal()の受信シグナル情報出力テーブル（WaiIPSigInf）のCodeメンバに設定されます。

int opt 現在、未使用。「0」を設定して下さい。

□ リターンパラメータ

int ret 0 : 正常終了
 - 1 : 異常終了
 - 5 1 ~ - 9 : μ ITRONサービスコールエラー

□ エラーコード

なし

□ 機能

オンボードC P U側のP I O割込モジュールからP C側に終了シグナルを送信する場合に使用します。P C側は、WaitforIPTaskSignal() コマンド等でオンボードC P U側からの終了シグナルをウェイトしていますが、本コマンドをオンボードC P U側から実行することにより、ウェイト状態が解除されます。

また、1つのスレッドまたは1つのデバイスIDに対してシグナルポートは1つだけ設定可能です。複数のスレッド、複数のデバイスIDで同じシグナルポート番号は使用できません。

本コマンドは、オンボードC P U側専用のコマンドですが、オンボードC P U側のプログラムをP C側でシミュレートできるようにP C側のコマンドとしてもダミーで実装しています。

第5章 ボード内デバイス制御コマンド

5.1 I/O入出力

5.1.1 I/Oポート入力（ipport_Read）

□ C言語API

```
int ret = ipport_Read( int Ch, int *pVal );
```

□ パラメータ

int	Ch	チャンネル（将来拡張用：0を設定すること）
int	*pVal	読み出しデータを返すアドレス

□ リターンパラメータ

int	ret	正常終了（0）またはエラーコード
int	*pVal	読み出しデータを格納したアドレス

□ エラーコード

- 1 7	パラメータエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

I/Oポートからデータ入力を行います。

NVP-Ax230CL/235CLのI/Oポートは8ビットです。ビット31～8は不定値となりますので注意ください。

5.1.2 I/Oポート出力 (ipport_Write)

□ C言語API

```
int ret = ipport_Write( int Ch, int Val, int Msk );
```

□ パラメータ

int Ch チャンネル (将来拡張用 : 0 を設定すること)

int Val 書き込みデータ

int Msk マスクデータ

マスクオプションの指定を行います。

オプション	内容
0	全ビット書き換えます。
≠ 0	タスク間で排他制御を行い有効な 16 ビットのうち「1」に対応するビットだけを書き換えます。

□ リターンパラメータ

int ret 正常終了 (0) またはエラーコード

□ エラーコード

- 17 パラメータエラー

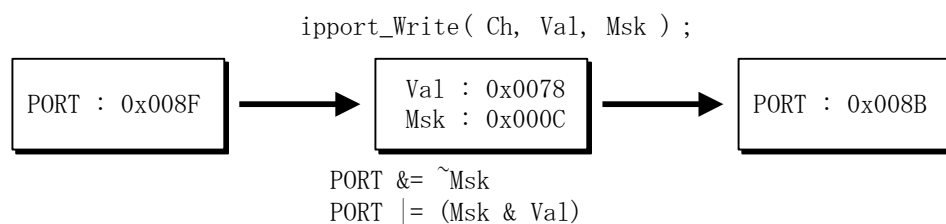
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

I/Oポートにデータを出力します。

NVP-Ax230CL/235CLは8ビットありますが、オプションでマスクを指定することにより、指定ビットだけを変更することができます。

マスクオプションの例



PORT : I/Oポートの状態

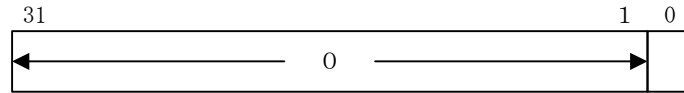
5.1.3 I/Oポートモード設定 (ipport_SetDIOMode)

□ C言語API

```
int ret = ipport_SetDIOMode( int mode );
```

□ パラメータ

int mode I/Oポートモード



mode	ポートモード	機能
bit0	D0ポートモード	0 : D0[6] = IOD0[6]出力 D0[7] = IOD0[7]出力 CPU_ERR = 使用不可 1 : D0[6] = CH2ストロボ信号出力 D0[7] = CH3ストロボ信号出力 CPU_ERR = WDTERR/CPUERR出力
bit1 ～ bit31	Reserve	0 (必ず0を設定してください)

□ リターンパラメータ

int ret 正常終了 (0) またはエラーコード

□ エラーコード

- 1 7 パラメータエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

I/Oポートモードを設定します。

5.2 SCIコントロール

5.2.1 SCIセットアップ (scicom_Setup)

□ C言語API

```
int ret = scicom_Setup( int Ch, SCICOM_SETUP_INFO *pInfo );
```

□ パラメータ

int Ch SCIのチャンネル (0 ~ 1)

SCICOM_SETUP_INFO *pInfo SCI設定情報を返すアドレス

□ パケットの構造

```
typedef struct {
    int                      Rate;                      ビットレート
                                     SCICOM_1200      (1200)      1200 bit/sec
                                     SCICOM_2400      (2400)      2400 bit/sec
                                     SCICOM_4800      (4800)      4800 bit/sec
                                     SCICOM_9600      (9600)      9600 bit/sec
                                     SCICOM_19200     (19200)     19200 bit/sec
                                     SCICOM_38400     (38400)     38400 bit/sec
                                     SCICOM_57600     (57600)     57600 bit/sec
    int                      Lng;                      キャラクタ長
                                     SCICOM_8Bit      (0)              8 bit データ
                                     SCICOM_7Bit      (1)              7 bit データ
    int                      Par;                      パリティ (※)
                                     SCICOM_NonParity    (0)              パリティ無し
                                     SCICOM_EvenParity   (1)              偶数パリティ
                                     SCICOM_OddParity    (2)              奇数パリティ
    int                      Stop;                    ストップビット
                                     SCICOM_StopBit1    (0)              1ストップビット
                                     SCICOM_StopBit2    (1)              2ストップビット
    int                      RcvBufSiz;              受信バッファサイズ
    int                      SndBufSiz;              送信バッファサイズ
} SCICOM_SETUP_INFO;
```

※ 偶数パリティに設定すると送信時には、パリティビットと送信キャラクタをあわせて、その中の 1 の数の合計が偶数になるようにパリティビットを付加して送信します。受信時には、パリティビットと受信キャラクタを合わせて、その中の 1 の数の合計が偶数であるかどうかのチェックを行います。

奇数パリティに設定すると送信時には、パリティビットと送信キャラクタをあわせて、その中の 1 の数の合計が奇数になるようにパリティビットを付加して送信します。受信時には、パリティビットと受信キャラクタを合わせて、その中の 1 の数の合計が奇数であるかどうかのチェックを行います。

□ リターンパラメータ

int ret 正常終了 (0) またはエラーコード

□ エラーコード

- 1 7 パラメータエラー

- 3 3 メモリ確保エラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

SCI (Serial Communication Interface) デバイスに対して、ビットレート、キャラクタ長、パリティ、ストップビットの設定を行います。

5.2.2 SCI入力バッファのパージ (scicom_Purge)

□ C言語API

```
int ret = scicom_Purge( int Ch );
```

□ パラメータ

int Ch SCIのチャネル (0 ~ 1)

□ リターンパラメータ

int ret 正常終了 (0) またはエラーコード

□ エラーコード

- 1 7 パラメータエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

SCI バッファのパージを行います。本コマンドにより、バッファにプールされている入力データが全て捨てられ、バッファが初期化されます。

5.2.3 SCIデータ受信 (scicom_Recv)

□ C言語API

```
int ret = scicom_Recv( int Ch, char *pBuf, int len );
```

□ パラメータ

int	Ch	SCIのチャネル (0 ~ 1)
char	*pBuf	受信データを返すアドレス
int	len	受信データバイト数

□ リターンパラメータ

int	ret	正常終了 (0) またはエラーコード
char	*pBuf	受信データを格納したアドレス

□ エラーコード

- 1 7	パラメータエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

SCIバッファから指定されたバイト数のデータを読み出します。バッファがエンプティ状態になると入力待ちになります。入力待ちにしない場合は、scicom_GetRecvStatus() コマンドでバッファに入力されているデータ数をチェックして下さい。

5.2.4 SCIデータ受信(タイムアウト付き) (scicom_tRecv)

□ C言語API

```
int ret = scicom_tRecv( int Ch, char *pBuf, int len, long tmo );
```

□ パラメータ

int	Ch	SCIのチャンネル (0～1)
char	*pBuf	受信データを返すアドレス
int	len	受信データバイト数
long	tmo	タイムアウト(ms) 「-1」を指定するとタイムアウト処理は行わず、scicom_Recv() コマンドと同じ処理を行います。

□ リターンパラメータ

int	ret	正常終了 (0) またはエラーコード
char	*pBuf	受信データを格納したアドレス

□ エラーコード

-17	パラメータエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

SCIバッファから指定されたバイト数のデータをタイムアウト付きで読み出します。バッファがエンプティ状態になると入力待ちになります。入力待ちにたくない場合は、scicom_GetRecvStatus() コマンドでバッファに入力されているデータ数をチェックして下さい。

5.2.5 SCIデータ送信 (scicom_Send)

□ C言語API

```
int ret = scicom_Send( int Ch, char *pBuf, int len );
```

□ パラメータ

int	Ch	SCIのチャネル (0 ~ 1)
char	*pBuf	送信データ格納アドレス
int	len	送信データバイト数

□ リターンパラメータ

int	ret	正常終了 (0) またはエラーコード

□ エラーコード

- 1 7	パラメータエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

指定したバッファから指定されたバイト数のデータをS C Iに出力します。

5.2.6 受信ステータスの取得 (scicom_GetRecvStatus)

□ C言語API

```
int ret = scicom_GetRecvStatus( int Ch, SCICOM_RCVSTS_INFO *pSts );
```

□ パラメータ

int	Ch	SCIのチャネル (0～1)
SCICOM_RCVSTS_INFO	*pSts	受信ステータス情報を返すアドレス

□ パケットの構造

```
typedef struct {
    int      Count;      受信バイト数
    int      BufFull;    受信バッファフルステータス
                        0 : バッファ空きあり
                        1 : バッファフル
} SCICOM_RCVSTS_INFO;
```

□ リターンパラメータ

int	ret	正常終了 (0) またはエラーコード
SCICOM_RCVSTS_INFO	*pSts	受信ステータス情報を格納したアドレス

□ エラーコード

- 1 7	パラメータエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

受信しているデータのバイト数とバッファフルステータスを取得します。

5.2.7 SCIの通信エラー取得 (scicom_GetError)

□ C言語API

```
int ret = scicom_GetError( int Ch, unsigned long *pErr );
```

□ パラメータ

int	Ch	SCIのチャネル (0 ~ 1)
unsigned long	*pErr	エラー情報を返すアドレス

□ リターンパラメータ

int	ret	エラーなし (0) またはエラーコード
unsigned long	*pErr	エラー情報を格納するアドレス

□ エラーコード

- 1 7	パラメータエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

SCIでの現在までの通信エラーを調べます。
このコマンド実行後、エラーは自動的にクリアされます。
以下にエラー情報の一覧を示します。

エラー情報

SCICOM_ERR_PER	(0x01)	受信時のパリティエラー
SCICOM_ERR_FER	(0x02)	受信時のフレーミングエラー
SCICOM_ERR_RDF	(0x04)	受信時のオーバランエラー
SCICOM_ERR_BRK	(0x08)	ブレーク検出

5.3 タイマ

5.3.1 時間計測モード設定 (SetIPTimeMode)

□ C言語API

```
int ret = SetIPTimeMode( enum IPTimeMode mode );
```

□ パラメータ

enum IPTimeMode mode 時間計測モード

時間計測モード	対応数	説明
U_SEC	0	μ S 単位の計測
M_SEC	1	m S 単位の計測
S_SEC	2	秒 単位の計測

□ リターンパラメータ

int ret 正常終了 (0) またはエラーコード

□ エラーコード

- 1 7 パラメータエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

timeGetIPTime() コマンドによる時間計測のモード設定を行います。なお、デフォルトは m S 単位に設定されています。

以下に処理時間の計測例を示します。

```
unsigned long time,stime,etime;

SetIPTimeMode(U_SEC);           //  $\mu$  S 計測設定

startGetIPTime();                // タイマスタート

stime = timeGetIPTime();         // 開始時間
. . . . .
処理
. . . . .

etime = timeGetIPTime();         // 処理終了時間

stopGetIPTime();                // タイマ停止

time = etime - stime;            // 処理時間算出
printf("Total Time : %d uS\n",time);
```

5.3.2 タイマスタート (startGetIPTime)

□ C言語API

```
int ret = startGetIPTime();
```

□ パラメータ

なし

□ リターンパラメータ

int ret 正常終了（0）またはエラーコード

□ エラーコード

なし

□ 機能

時間計測のためのタイマを起動します。

以下に処理時間の計測例を示します。

```
unsigned long time,stime,etime;

startGetIPTime();           // タイマスタート

stime = timeGetIPTime();    // 開始時間

. . . . .
処理
. . . . .

etime = timeGetIPTime();    // 処理終了時間

stopGetIPTime();           // タイマ停止

time = etime - stime;       // 処理時間算出
printf("Total Time : %d mS\n",time);
```

5.3.3 タイマ時間読み出し (timeGetIPTime)

□ C言語API

```
unsigned long ret = timeGetIPTime();
```

□ パラメータ

なし

□ リターンパラメータ

```
unsigned long    ret    0xFFFFFFFF以外 : 経過時間 (デフォルト : m S   Max値 : 87,960,930[ms])
                  0xFFFFFFFF      : 異常終了
```

□ エラーコード

なし

□ 機能

時間計測のためのタイマの現在の経過時間を読み出します。読み出される時間は、デフォルトで m S 単位ですが、SetIPTimeMode() コマンドにより、 μ S, m S, S 単位での計測が可能です。

以下に処理時間の計測例を示します。

```
unsigned long time, stime, etime;

startGetIPTime();                // タイマスタート

stime = timeGetIPTime();         // 開始時間

. . . . .
処理
. . . . .

etime = timeGetIPTime();         // 処理終了時間

stopGetIPTime();                // タイマ停止

time = etime - stime;            // 処理時間算出
printf("Total Time : %d mS\n", time);
```


5.3.4 タイマストップ (stopGetIPTime)

□ C言語API

```
int ret = stopGetIPTime();
```

□ パラメータ

なし

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (-1)

□ エラーコード

— エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

時間計測のためのタイマを停止します。

以下に処理時間の計測例を示します。

```
unsigned long time,stime,etime;

startGetIPTime();           // タイマスタート

stime = timeGetIPTime();    // 開始時間

.....
処理
.....

etime = timeGetIPTime();    // 処理終了時間

stopGetIPTime();           // タイマ停止

time = etime - stime;       // 処理時間算出
printf("Total Time : %d mS\n",time);
```

5.3.5 パフォーマンスカウンターの周波数取得（ GetPerformanceFrequency ）

□ C言語API

```
int ret = GetPerformanceFrequency( long long *lpFrequency );
```

□ パラメータ

long long	*lpFrequency	周波数（H z）を返すアドレス

□ リターンパラメータ

int	ret	正常終了（0）または 異常終了（- 1）

long long	*lpFrequency	周波数（H z）を格納したアドレス

□ エラーコード

なし

□ 機能

パフォーマンスカウンターの周波数を取得します。

5.3.6 パフォーマンスカウンターの現在値取得（ GetPerformanceCounter ）

□ C言語API

```
int ret = GetPerformanceCounter( long long *lpPerformanceCount );
```

□ パラメータ

long long	*lpPerformanceCount	カウンタ値を返すアドレス

□ リターンパラメータ

int	ret	正常終了（0）または 異常終了（-1）

long long	*lpPerformanceCount	カウンタ値を格納したアドレス

□ エラーコード

なし

□ 機能

パフォーマンスカウンタの現時点の値を取得します。

5.3.7 タイマウェイト（IPSleep）

□ C言語API

```
int ret = IPSleep( unsigned long wait );
```

□ パラメータ

unsigned long wait ウェイト時間。mS単位で設定して下さい。

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

— エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

任意時間ウェイトします。

5.3.8 時刻の登録 (SetIPTime)

□ C言語API

```
int  ret = SetIPTime( IP_SYSTEM_TIME *time );
```

□ パラメータ

IP_SYSTEM_TIME	*time	仮想リアルタイムクロックに登録する時刻 (西暦/月/日/時/分/秒)を返すアドレス 曜日のデータは登録されません

□ リターンパラメータ

```
int      ret      0 : 正常終了。  
          -1 : 異常終了。R T Cに登録する時刻（西暦／月／日／時／分／秒）が不適切。
```

□ パケットの構造

```
typedef struct {
    short    year;           西暦
    short    month;         月
    short    day_of_week;   曜日
    short    day;           日
    short    hour;          時(24時間)
    short    minute;        分
    short    second;        秒
    short    milliseconds;  mS
} IP_SYSTEM_TIME;
```

メンバ名	説明														
year	年（西暦）														
month	月（1-1月，2-2月・・・）														
day_of_week	曜日 <table><tr><td>日</td><td>月</td><td>火</td><td>水</td><td>木</td><td>金</td><td>土</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr></table>	日	月	火	水	木	金	土	0	1	2	3	4	5	6
日	月	火	水	木	金	土									
0	1	2	3	4	5	6									
day	日（1～31）														
hour	時（24時間制：0～23）														
minute	分（0～59）														
second	秒（0～59）														
milliseconds	m sec（10m sec単位：0～990）														

❏ エラーコード

なし

機能

μ ITRON上の仮想リアルタイムクロックに対して、時刻を登録します。GetIPTime() コマンドでは、本コマンドで登録した時刻から積算された時刻が読み出されます。

SetIPTime() コマンドでは、曜日のデータは登録されませんが、GetIPTime() コマンドで、曜日のデータが取得されます。

5.3.9 時刻の取得 (GetIPTime)

□ C言語API

```
int  ret = GetIPTime( IP_SYSTEM_TIME *time );
```

□ パラメータ

IP_SYSTEM_TIME	*time	仮想リアルタイムクロックに登録する時刻 (西暦/月/日/時/分/秒)を返すアドレス

□ リターンパラメータ

```
int          ret          正常終了 (0)
```

IP_SYSTEM_TIME	*time	仮想リアルタイムクロックに登録する時刻 (西暦/月/日/時/分/秒)を格納したアドレス

□ パケットの構造

```
typedef struct {
    short    year;           西暦
    short    month;         月
    short    day_of_week;   曜日
    short    day;           日
    short    hour;          時(24時間)
    short    minute;        分
    short    second;        秒
    short    milliseconds;  mS
} IP_SYSTEM_TIME;
```

メンバ名	説明																				
year	年（西暦）																				
month	月（1-1月，2-2月・・・）																				
day_of_week	曜日 <table><tr><td>日</td><td>月</td><td>火</td><td>水</td><td>木</td><td>金</td><td>土</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr></table>							日	月	火	水	木	金	土	0	1	2	3	4	5	6
日	月	火	水	木	金	土															
0	1	2	3	4	5	6															
day	日（1～31）																				
hour	時（24時間制：0～23）																				
minute	分（0～59）																				
second	秒（0～59）																				
milliseconds	m sec（1m sec単位：0～999）※RTCあり = 0																				

❏ エラーコード

なし

機能

RTC機能がない製品では、 μ ITRON上の仮想リアルタイムクロックの現在時刻を取得します。正確には、SetIPTime() コマンドで登録した時刻から積算された時刻が読み出されます。従って、SetIPTime() コマンドで時刻をあらかじめ登録する必要があります。システムがイニシャライズされると 2000/01/01 0:0:0:0 にリセットされます。

5.4 USB－HID制御コマンド

USB-HID制御コマンドを使用する場合は、“usbapi.h” をインクルードして下さい。

5.4.1 USB_HID 制御の開始 (USB_HID_Start)

□ C言語API

```
int ret = USB_HID_Start( USB_HID_CALLBACK CallBack );
```

□ パラメータ

USB_HID_CALLBACK	*CallBack	HIDデバイス挿抜時に呼び出されるコールバックルーチン

□ リターンパラメータ

int	ret	(0) : 正常終了 (- 1) : 異常終了

□ エラーコード

なし

□ 機能

USB HIDデバイス挿抜時に呼び出されるコールバックルーチン(ユーザ作成)を登録し、HIDドライバ制御を起動します。

※HIDドライバで制御するデバイスは、キーボード、マウスです。

5.4.2 USB_HID 制御の終了 (USB_HID_Terminate)

□ C言語API

```
int ret = USB_HID_Terminate( void );
```

□ パラメータ

なし

□ リターンパラメータ

int	ret	(0)	: 正常終了
		(- 1)	: 異常終了

□ エラーコード

なし

□ 機能

USB HIDドライバ制御を停止します。

5.4.3 USB HIDデバイスからの情報取得（USB_HID_GetStatus）

□ C言語API

```
int ret = USB_HID_GetStatus( USB_HID_HPIPE hPipe, unsigned char *Buff, int size, int tmout );
```

□ パラメータ

USB_HID_HPIPE	hPipe	HIDデバイスアクセス用パイプテーブルポインタ
unsigned char	*Buff	取得情報を返すアドレス
int	size	読み出す最大データサイズ
int	tmout	読み出し時のタイムアウト

□ リターンパラメータ

int	ret	(≥ 0) : 正常終了 (読み出したサイズ) (< 0) : 異常終了
unsigned char	*Buff	取得情報を格納したアドレス

□ エラーコード

なし

□ 機能

パイプインデックスに対応するパイプから読み出す最大サイズで指定されたサイズ分のデータを読み出し、取得情報格納領域のポインタで指定されたエリアに格納します。

パラメータとして必要なデータは、コールバックルーチンを介して取得できます。コールバックルーチンから渡されるテーブル情報をもとにパラメータを設定します。

5.4.4 USB HIDデバイスへの情報設定（USB_HID_SetReport）

□ C言語API

```
int ret = USB_HID_SetReport( int fd, int type, int id, int len, void *rep );
```

□ パラメータ

int	fd	設定対象デバイス用ファイルディスクリプタ
int	type	設定対象デバイスタイプ
int	id	設定対象デバイスID
int	len	設定情報サイズ
void	*rep	設定情報を返すアドレス

□ リターンパラメータ

int	ret	(≥ 0) : 正常終了（読み出したサイズ） (< 0) : 異常終了
void	*rep	設定情報を格納したアドレス

□ エラーコード

なし

□ 機能

設定情報格納領域のポインタで指定されたエリアに格納されているデータを、デバイスタイプ、デバイスIDに対応するファイルディスクリプタへ、設定情報サイズで指定されたサイズ分のデータを設定します（書き込み）。

パラメータとして必要なデータは、コールバックルーチンを介して取得できます。コールバックルーチンから渡されるテーブル情報をもとにパラメータを設定します。

5.5 WDTコマンド

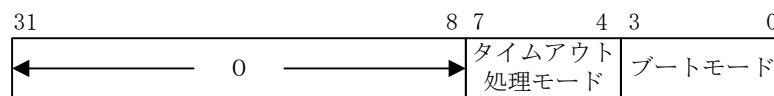
5.5.1 WDTの起動（WDT_Start）

□ C言語API

```
int ret = WDT_Start( int mode, int tmo );
```

□ パラメータ

int mode WDT動作モード（WDTタイムアウト処理モードとブートモードのOR値を設定してください）
 [ブートモード]
 0：ノーマルブート
 1：セーフブート
 [WDTタイムアウト処理モード]
 0：システムリセットする
 1：システムリセットしない



int tmo タイムアウト時間（mS）

0～	1：	Min	1 mS	Max	3 mS
2～	10：	Min	10 mS	Max	30 mS
11～	30：	Min	30 mS	Max	90 mS
31～	100：	Min	100 mS	Max	300 mS
101～	1000：	Min	1 S	Max	3 S
1001～	10000：	Min	10 S	Max	30 S
10001～	60000：	Min	60 S	Max	180 S

□ リターンパラメータ

int ret 正常終了（0）またはエラーコード

□ エラーコード

- 17 パラメータエラー

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

WDT（Watchdog Timer）を起動します。

5.5.2 WDTの停止（WDT_Stop）

□ C言語API

```
int ret = WDT_Stop();
```

□ パラメータ

なし

□ リターンパラメータ

int	ret	正常終了（0）

□ エラーコード

なし

□ 機能

WDT（Watchdog Timer）を停止します。

5.5.3 WDTカウンタのリセット (WDT_Reset)

□ C言語API

```
int ret = WDT_Reset();
```

□ パラメータ

なし

□ リターンパラメータ

int	ret	正常終了 (0)

□ エラーコード

なし

□ 機能

WDT (Watchdog Timer) カウンタをリセットします。

5.5.4 WDTステータスの取得 (WDT_GetStatus)

□ C言語API

```
int ret = WDT_GetStatus( int *pStatus );
```

□ パラメータ

int *pStatus WDTステータスを返すアドレス

□ リターンパラメータ

int ret 正常終了 (0)

int *pStatus WDTステータスを格納したアドレス
 0 : 通常のPOWER ONリセットからの起動
 1 : WDTタイムアウトでリセット発生(パワーオンブートモード)
 2 : WDTタイムアウトでリセット発生(セーフブートモード)
 3 : ソフトリセットからの起動(パワーオンブートモード)
 4 : ソフトリセットからの起動(セーフブートモード)
 5 : WDT動作中

□ エラーコード

なし

□ 機能

WDT (Watchdog Timer) のステータスを取得します。

WDTの条件がソフトリセットまたはセーフブートのときに 5 : WDT動作中 となります。

ソフトリセットは、ResetIPコマンドを実行した場合にだけ発生します。そのため、3 : ソフトリセットからの起動(パワーオンブートモード)と 4 : ソフトリセットからの起動(セーフブートモード)は、ResetIPコマンドの実行により起動されたことを示します。

5.5.5 WDTステータスのクリア (WDT_ClearStatus)

□ C言語API

```
int ret = WDT_ClearStatus( );
```

□ パラメータ

なし

□ リターンパラメータ

int	ret	正常終了 (0)

□ エラーコード

なし

□ 機能

WDT (Watchdog Timer) のステータスをクリアします。

5.6 CPUエラーコマンド

5.6.1 CPUエラーの出力（CPUERR_Set）

□ C言語API

```
int ret = CPUERR_Set( );
```

□ パラメータ

なし

□ リターンパラメータ

int	ret	正常終了（0）

□ エラーコード

なし

□ 機能

CPUエラーを出力します。

5.6.2 CPUエラーのクリア (CPUERR_Clear)

□ C言語API

```
int ret = CPUERR_Clear();
```

□ パラメータ

なし

□ リターンパラメータ

int	ret	正常終了 (0)

□ エラーコード

なし

□ 機能

CPUエラーをクリアします。

6.1.2 ファイルクローズ (fmClose)

□ C言語API

```
int ret = fmClose( FM_HFILE fh );
```

□ パラメータ

FM_HFILE fh クローズするファイルのファイルハンドル

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ エラーコード

異常終了した場合、fmGetErrorでエラー情報を取得できます。

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

オープンしたファイルをクローズします。

ファイル作業領域の内容が更新されていた場合、その内容をオンボードディスクに書き込みます。
また、ファイル作業領域の解放を行います。

6.1.3 ファイル読み込み（fmRead）

□ C言語API

```
int ret = fmRead( FM_HFILE fh , char *buff , unsigned long size );
```

□ パラメータ

FM_HFILE fh	読み込むファイルのファイルハンドル
char *buff	読み込むデータを返すアドレス
unsigned long size	読み込むデータサイズ（バイト）

□ リターンパラメータ

int ret	正常終了（0）または異常終了（-1）
char *buff	読み込むデータを格納したアドレス

□ エラーコード

異常終了した場合、fmGetErrorでエラー情報を取得できます。

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

オープンしたファイルの内容を読み込みます。
本関数は、ファイル作業領域から読み込みを行います。

6.1.4 ファイル書き込み（fmWrite）

□ C言語API

```
int ret = fmWrite( FM_HFILE fh , char *buff , unsigned long size );
```

□ パラメータ

FM_HFILE fh	書き込むファイルのファイルハンドル
char *buff	書き込むデータを返すアドレス
unsigned long size	書き込むデータサイズ

□ リターンパラメータ

int ret	正常終了（0）または異常終了（-1）

□ エラーコード

異常終了した場合、fmGetErrorでエラー情報を取得できます。

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

オープンしたファイルにデータを書き込みます。
 本関数は、ファイル作業領域に対して書き込みを行います。
 オンボードディスクへの書き込みタイミングは、ファイル作業領域サイズを超える場合に行います。

6.1.5 ファイルサイズ取得 (fmGetFileSize)

□ C言語API

```
unsigned long ret = fmGetFileSize( FM_HFILE fh );
```

□ パラメータ

FM_HFILE fh	ファイルサイズを取得するファイルのファイルハンドル

□ リターンパラメータ

unsigned long	ret	≥ 0 : ファイルサイズ(byte)
		- 1 : 異常終了

❏ エラーコード

異常終了した場合、fmGetErrorでエラー情報を取得できます。

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

 機能

オープンしたファイルのサイズを取得します。

6.1.6 ファイル削除（fmDelete）

□ C言語API

```
int ret = fmDelete( char *filename );
```

□ パラメータ

char *filename ファイル名(パス名付き)

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

異常終了した場合、fmGetErrorでエラー情報を取得できます。

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

指定ファイルを削除します。

6.1.7 ファイル名変更（fmRename）

□ C言語API

```
int ret = fmRename( char *srcfilename , char *dstfilename );
```

□ パラメータ

char *srcfilename 変更前ファイル名(パス名付き)

char *dstfilename 変更後ファイル名

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

異常終了した場合、fmGetErrorでエラー情報を取得できます。

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

指定ファイル名を変更します。変更後ファイル名にパス名付きのファイル名を指定した場合、パスは無視されます。

6.1.8 ファイル検索 (fmFindFile)

□ C言語API

```
int ret = fmFindFile( char *filename );
```

□ パラメータ

char	*filename	ファイル名

□ リターンパラメータ

int	ret	1 : ファイル有り
		0 : ファイル無し
		-1 : 異常終了

□ エラーコード

異常終了した場合、fmGetErrorでエラー情報を取得できます。

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

指定されたファイルが存在するかどうかをチェックします。

6.1.9 ファイル属性の変更（fmChgAttr）

□ C言語API

```
int ret = fmChgAttr( char *filename , int attr , int flg );
```

□ パラメータ

char *filename ファイル名

int attr 属性

属性	対応数	説明
FM_ATTR_DEFAULT	0x00	デフォルト
FM_ATTR_RDONLY	0x01	リードオンリ
FM_ATTR_WRONLY	0x02	ライトオンリ
FM_ATTR_RDWR	0x04	リードライト
FM_ATTR_SECRET	0x10	シークレット

int flg 属性ビットを有効／無効にするフラグ

フラグ	対応数	説明
FM_RESET	0	リセット（無効）
FM_SET	1	セット（有効）

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

異常終了した場合、fmGetErrorでエラー情報を取得できます。

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

指定されたファイルの属性を変更します。

6.1.10 PC-オンボードディスク間のファイルコピー（fmFileCopy）

□ C言語API

```
int ret = fmFileCopy( int mode , char *srcfilename , char *dstfilename );
```

□ パラメータ

int mode コピーモード

モード	対応数	説明
HDD_TO_FLASH	0	HDD→オンボードディスク
FLASH_TO_HDD	1	オンボードディスク→HDD
FLASH_TO_FLASH	2	オンボードディスク→オンボードディスク

char *srcfilename ソース（コピー元）ファイル名（パス名付き）

char *dstfilename デスティネーション（コピー先）ファイル名（パス名付き）

□ リターンパラメータ

int ret 正常終了（0）またはエラーコード

□ エラーコード

- 1 異常終了（内部関数エラー）
 - 1 7 パラメータエラー
 - 4 1 内部オブジェクトエラー
 - 1 0 3 ファイルオープンエラー
 - 1 0 6 作業領域メモリ確保エラー
 - 1 1 0 オンボードディスクディスク容量不足

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

PC→オンボードディスク／オンボードディスク→PC／オンボードディスク→オンボードディスクへファイルをコピーします。PC側にのみ実装されていますのでSH側からの呼び出しはできません。

6.1.11 エラーコード取得（ fmGetError ）

□ C言語API

```
int Error = fmGetError( void );
```

□ パラメータ

なし

□ リターンパラメータ

int	Error	エラー無し（0）それ以外はエラーコード

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ファイルアクセスコマンドで発生したエラーの情報を読み出します。

6.1.12 ファイルポインタの移動（fmSeek）

□ C言語API

```
int ret = fmSeek( FM_HFILE fh, long offset, int origin );
```

□ パラメータ

FM_HFILE fh ファイルハンドル

long offset originからのバイト数
 (負を設定するとポインタ位置が指定バイト数分戻る)

int origin 初期位置

初期位置	対応数	説明
FM_SEEK_CUR	0	ファイルポインタの位置から
FM_SEEK_END	1	ファイルの終端から
FM_SEEK_SET	2	ファイルの先頭から

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

異常終了した場合、fmGetErrorでエラー情報を取得できます。

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

hFileでオープンされているファイルの指定された位置にファイルポインタを移動させます。
 originからoffsetバイトの新しい位置に移動します。ファイルに対するその次の操作は、この新しい位置で行われます。

6.1.13 ディレクトリの作成 (fmCreateDirectory)

□ C言語API

```
int ret = fmCreateDirectory( char *pathName, int attr );
```

□ パラメータ

char	*pathName	ディレクトリ名(パス名付き)
int	attr	アトリビュート(0を指定)

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)

□ エラーコード

異常終了した場合、fmGetErrorでエラー情報を取得できます。

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ディレクトリを作成します。パス名に含まれるディレクトリが存在しない場合は、ディレクトリを作成します。既存ディレクトリ名を指定した場合は、処理をせずにリターンパラメータに0を返します。

6.1.14 ディレクトリの削除（ fmRemoveDirectory ）

□ C言語API

```
int ret = fmRemoveDirectory( char *pathName );
```

□ パラメータ

char *pathName ディレクトリ名(パス名付き)

□ リターンパラメータ

int ret 正常終了（0）または異常終了（-1）

□ エラーコード

異常終了した場合、fmGetErrorでエラー情報を取得できます。

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ディレクトリを削除します。指定したディレクトリ内の全てのファイルとディレクトリを削除します。

6.1.15 ファイルの検索と検索ハンドルのオープン(fmFindFirstFile)

□ C言語API

```
FM_HFIND hFind = fmFindFirstFile( char *pathName, FM_FIND_TBL *tbl );
```

□ パラメータ

char	*pathName	ファイル名(パス名付き)
FM_FIND_TBL	*tbl	ファイル検索結果を格納する領域のアドレス

□ リターンパラメータ

FM_HFIND	hFind	検索ハンドル (≠0) または異常終了 (0)
FM_FIND_TBL	*tbl	ファイル検索結果

□ エラーコード

異常終了した場合、fmGetErrorでエラー情報を取得できます。

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ディレクトリから指定された検索ファイル名が一致する最初のファイルを検索します。この関数を呼び出した後、fmFindNextFile関数を繰り返し呼び出すことにより、検索ファイル名と一致する全てのファイルとディレクトリを取得することが出来ます。

FM_FIND_TBL のフォーマット及び内容を、下記に示します。

```
typedef struct {
    long        utime;           // 上位16bit
    unsigned long ltime;        // 下位32bit
} IP_FILE_TIME;

typedef struct {
    unsigned long attr;         // ファイル属性
    unsigned long size;        // ファイルサイズ
    IP_FILE_TIME time;         // 最終書き込み時間 (8B)
    unsigned long reserved[2]; // リザーブ
    char        name[C_FSZ_FILENAME]; // ファイル名(128文字(NULL含む))
} FM_FIND_TBL;
```


6.1.16 ファイルの検索 (fmFindNextFile)

□ C言語API

```
int ret = fmFindNextFile( FM_HFIND hFind, FM_FIND_TBL *tbl );
```

□ パラメータ

FM_HFIND	hFind	検索ハンドル
FM_FIND_TBL	*tbl	ファイル検索結果を格納する領域のアドレス

□ リターンパラメータ

int	ret	全てのファイルの検索終了 (0) まだファイルが存在している (1)
FM_FIND_TBL	*tbl	ファイル検索結果

□ エラーコード

異常終了した場合、fmGetErrorでエラー情報を取得できます。

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

前のfmFindFirstFile()呼び出しで指定した検索名と一致する次のファイルを検索します。全てのファイルが呼び出され「0」が返されるまでfmFindNextFile()呼び出します。

FM_FIND_TBL のフォーマット及び内容を、下記に示します。

```
typedef struct {
    long        utime;           // 上位16bit
    unsigned long ltime;        // 下位32bit
} IP_FILE_TIME;

typedef struct {
    unsigned long attr;          // ファイル属性
    unsigned long size;          // ファイルサイズ
    IP_FILE_TIME time;           // 最終書き込み時間 (8B)
    unsigned long reserved[2];   // リザーブ
    char          name[C_FSZ_FILENAME]; // ファイル名(128文字(NULL含む))
} FM_FIND_TBL;
```

6.1.17 検索ハンドルのクローズ (fmFindClose)

□ C言語API

```
int ret = fmFindClose( FM_HFIND hFind );
```

□ パラメータ

FM_HFIND	hFind	検索ハンドル

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

異常終了した場合、fmGetErrorでエラー情報を取得できます。

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

fmFindFirstFile() で返された検索ハンドルをクローズします。

6.1.18 ファイルの属性参照(ファイル名指定) (fmRefAttr)

□ C言語API

```
int ret = fmRefAttr( char *pathName, int *attr );
```

□ パラメータ

char	*pathName	ファイル名(パス名付き)
int	*attr	属性を格納する領域のアドレス

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)
int	*attr	属性

属性	対応数	説明
FM_ATTR_DEFAULT	0x00	デフォルト
FM_ATTR_RDONLY	0x01	リードオンリ
FM_ATTR_WRONLY	0x02	ライトオンリ
FM_ATTR_RDWR	0x04	リードライト
FM_ATTR_SECRET	0x10	シークレット
FM_ATTR_DIRECTORY	0x1000	ディレクトリ

□ エラーコード

異常終了した場合、fmGetErrorでエラー情報を取得できます。

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

指定されたファイルの属性を参照します。

6.1.19 ディスクドライブのフォーマット (fmDriveFormat)

□ C言語API

```
int ret = fmDriveFormat( char *name );
```

□ パラメータ

char	*name	ドライブ名

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

異常終了した場合、fmGetErrorでエラー情報を取得できます。

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

システムにマウントされているディスクドライブのフォーマットを行います。

(注) 大容量のディスクをフォーマットする場合、本コマンドを使用すると非常に時間かかりますので、ご使用のパソコンでフォーマットすることをお勧めします。

6.1.20 ディスクサイズと空きエリア容量の取得 (fmDriveSpace)

□ C言語API

```
int ret = fmDriveSpace( char *name, long long *disksize, long long *freesize );
```

□ パラメータ

char	*name	ドライブ名
long long	*disksize	ドライブのディスクサイズを格納する領域のアドレス
long long	*freesize	ドライブの空きエリア容量を格納する領域のアドレス

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)
long long	*disksize	ドライブのディスクサイズ
long long	*freesize	ドライブの空きエリア容量

□ エラーコード

異常終了した場合、fmGetErrorでエラー情報を取得できます。

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

システムにマウントされているドライブのディスクサイズとドライブドライブの空きエリア容量を取得します。

6.1.21 オンボードディスク間のファイルコピー（fmCopyFile）

□ C言語API

```
int ret = fmCopyFile( char *srcfilename , char *dstfilename );
```

□ パラメータ

char	*srcfilename	ソース（コピー元）ファイル名（パス名付き）
char	*dstfilename	デスティネーション（コピー先）ファイル名（パス名付き）

□ リターンパラメータ

int	ret	正常終了（0）またはエラーコード

□ エラーコード

- 1	異常終了（内部関数エラー）
- 1 7	パラメータエラー
- 4 1	内部オブジェクトエラー
- 1 0 3	ファイルオープンエラー
- 1 0 6	作業領域メモリ確保エラー
- 1 1 0	オンボードディスクディスク容量不足

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

オンボードディスク上のファイルをコピーします。本コマンドは fmFileCopy のコピーモード FLASH_TO_FLASH に相当します。

6.1.22 システム時刻形式変換（FileTimeToIPTime）

□ C言語API

```
int ret = FileTimeToIPTime( IP_FILE_TIME *FileTime, IP_SYSTEM_TIME *time );
```

□ パラメータ

IP_FILE_TIME	*FileTime	ファイル時刻を格納したアドレス
IP_SYSTEM_TIME	*time	システム時刻を返すアドレス

□ リターンパラメータ

int	ret	正常終了（0）またはエラーコード
IP_SYSTEM_TIME	*time	システム時刻を格納したアドレス

□ パケットの構造

```
typedef struct {
    long          utime;          上位16bit
    unsigned long ltime;          下位32bit
} IP_FILE_TIME;

typedef struct {
    short          year;          西暦
    short          month;         月
    short          day_of_week;   曜日
    short          day;           日
    short          hour;          時(24時間)
    short          minute;        分
    short          second;        秒
    short          milliseconds;  mS
} IP_SYSTEM_TIME;
```

メンバ名	説明														
year	年（西暦）														
month	月（1-1月，2-2月・・・）														
day_of_week	曜日 <table><tr><td>日</td><td>月</td><td>火</td><td>水</td><td>木</td><td>金</td><td>土</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr></table>	日	月	火	水	木	金	土	0	1	2	3	4	5	6
日	月	火	水	木	金	土									
0	1	2	3	4	5	6									
day	日（1～31）														
hour	時（24時間制：0～23）														
minute	分（0～59）														
second	秒（0～59）														
milliseconds	m sec（10m sec単位：0～990）														

□ エラーコード

なし

□ 機能

ファイル時刻をシステム時刻形式に変換します。ファイル時刻はfmFindFirstFileやfmFindNextFileで取得します。

第7章 μ ITRONサービスコール

7.1 μ ITRONサービスコール

NVP-Ax230シリーズは、リアルタイムOS： μ ITRON4.0で動作しており、以下のサービスコールをオンボードアプリケーションで使用可能です。

なお、 μ ITRON4.0仕様は、トロン協会が中心となって策定されたオープンなリアルタイムカーネル仕様です。 μ ITRON4.0仕様の仕様書は、トロン協会のホームページ (<http://www.assoc.tron.org/>) から入手することができます。

この章では、 μ ITRON4.0のサービスコールの補足説明を記載します。

サービスコール一覧

NO	サービスコール	機能	備考
1	acre_tsk	タスクの生成	
2	del_tsk	タスクの削除	
3	act_tsk	タスクの起動	
4	ext_tsk	自タスクの終了	
5	ter_tsk	タスクの強制終了	
6	chg_pri	タスク優先度の変更	
7	ref_tsk	タスクの状態参照	
8	slp_tsk	起床待ち	
9	wup_tsk	タスクの起床	
10	can_wup	タスク起床要求のキャンセル	
11	sus_tsk	強制待ち状態への移行	
12	rsm_tsk	強制待ち状態からの再開	
13	dly_tsk	タスク遅延	
14	acre_sem	セマフォの生成	
15	del_sem	セマフォの削除	
16	sig_sem	セマフォ資源の返却	
17	wai_sem	セマフォ資源の獲得	
18	ref_sem	セマフォの状態参照	
19	acre_flg	イベントフラグの生成	
20	del_flg	イベントフラグの削除	
21	set_flg	イベントフラグのセット	
22	clr_flg	イベントフラグのクリア	
23	wai_flg	イベントフラグ待ち	
24	ref_flg	イベントフラグの状態参照	

7.1.1 タスクの生成 (acre_tsk)

□ C言語API

```
ER_ID tskid = acre_tsk(T_CTSK *pk_ctsk);
```

□ パラメータ

T_CTSK *pk_ctsk タスク生成情報を格納したパケットへのポインタ

□ リターンパラメータ

ER_ID tskid 生成したタスクID(正の値)またはエラーコード

□ パケットの構造

```
typedef struct t_ctsk {
    ATR      tskatr;          タスク属性
    VP_INT   exinf;          拡張情報
    FP       task;           タスク起動アドレス
    PRI      itskpri;         初期タスク優先度
    SIZE     stksz;          タスクスタックサイズ
    VP       stk;            タスクスタック領域の先頭アドレス
    VP       name;           タスク名(拡張)
    VP       path;           標準入出力パス名(拡張)
} T_CTSK;
```

□ エラーコード

E_NOMEM	メモリ不足(タスクスタック領域を確保できない)
E_RSATR	予約属性(tskatr が不正)
E_PAR	パラメータエラー(pk_ctsk が4 の倍数以外、task が奇数、stksz が4 の倍数以外、 stksz=0、stksz $\geq$ 0x80000000、itskpri $\leq$ 0、itskpri>CFG_MAXTSKPRI, stk が NULL 以外で4 の倍数以外)
E_OBJ	オブジェクト状態不正(tskid のタスクが休止状態でない、または自タスク指定)
E_NOID	空きID なし

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

acre_tsk サービスコールはシステムから要求した容量のスタックを割り当てます。生成したタスクは、TA_ACT 属性の指定がない場合は休止状態へ、TA_ACT 属性の指定がある場合は実行可能状態に移行します。

以下に各パラメータの意味を示します。

(1) tskid

未登録のタスクID を検索してそのID を持つタスクをpk_ctsk で示された内容で生成し、そのID をリターンパラメータとして返します。

(2) tskatr

tskatr には属性として、タスクを記述した言語およびコプロセッサの使用を指定します。

tskatr := ((TA_HLNG || TA_ASM) [|TA_ACT] [|TA_COP0] [|TA_COP1] [|TA_COP2]
[|TA_NAME] [|TA_PATH])

- TA_HLNG (H' 00000000) 高級言語記述
- TA_ASM (H' 00000001) アセンブラ記述
- TA_ACT (H' 00000002) タスク生成後にタスクを実行可能状態へ
- TA_COP1 (H' 00000200) FPU のバンク0 を使用 *
- TA_COP2 (H' 00000400) FPU のバンク1 を使用 *
- TA_NAME (H' 00001000) タスク名を指定(nameメンバにタスク名を指定)
- TA_PATH (H' 00002000) 標準入出力のパス名を指定(pathメンバにパス名を指定)

* FPSCR の初期値は、H' 00040001(バンク0)です。

TA_ACT 属性を指定した場合、対象タスクには拡張情報(exinf)がパラメータとして渡ります。

TA_COPn 属性の指定により、当該コプロセッサのレジスタもタスクコンテキストとして保存されるようになります。なお、TA_COPn, TA_NAME, TA_PATH属性は μ ITRON4. 0仕様の範囲外の属性です。

(3) exinf

exinf は、ユーザが生成するタスクに関する情報を設定するなどの目的で自由に使用できます。

(4) task

task には、タスクの起動アドレスを指定します。

(5) itskpri

itskpri には、タスク起動時の優先度の初期値として1~CFG_MAXTSKPRI の値を指定します。

(6) stksz

生成するタスクのスタックサイズを指定します。スタックサイズには4の倍数の値を指定します。

(7) stk

stk にNULL を指定した場合は、スタックはシステムから割り付けられます(通常はNULLを指定してください)。また、生成するタスクのスタックアドレスを指定することもできます。この場合、stksz で指定したサイズのスタック領域を確保し、その先頭アドレスを指定してください。

(8) name

name には、タスク名文字列の先頭アドレスを指定します。タスク名を指定するには、tskatr にTA_NAME の指定が必要です。TA_NAMEを指定しない場合は、NULLを指定してください。

(9) path

path には、標準入出力を行うパス名文字列の先頭アドレスを指定します。パス名を指定するには、tskatr にTA_PATH の指定が必要です。TA_PATHを指定しない場合は、NULLを指定してください。

パス名	標準入出力先	備考
¥¥con1	COM1	
¥¥con2	COM2	
¥¥xterm1	XTERM CH1	
¥¥xterm2	XTERM CH2	

7.1.2 タスクの削除 (del_tsk)

□ C言語API

```
ER ercd = del_tsk(ID tskid);
```

□ パラメータ

ID	tskid	タスクID

□ リターンパラメータ

ER ercd	正常終了 (E_OK) またはエラーコード

□ エラーコード

E_ID	不正ID 番号($tskid \leq 0$ 、 $tskid > CFG\_MAXTSKID$)
E_NOEXS	未登録 ($tskid$ のタスクが存在しない)
E_OBJ	オブジェクト状態不正 ($tskid$ のタスクが休止状態でない、または自タスク指定)
E_CTX	コンテキストエラー (許可されていないシステム状態からの呼び出し)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

$tskid$ で示されたタスクを削除します。削除されたタスクは、未登録状態へ移行します。

7.1.3 タスクの起動 (act_tsk)

□ C言語API

```
ER ercd = act_tsk(ID tskid);
```

□ パラメータ

ID	tskid	タスクID

□ リターンパラメータ

ER ercd	正常終了 (E_OK) またはエラーコード

□ エラーコード

E_ID	不正ID 番号 (tskid < 0、tskid > CFG_MAXTSKID、非タスクコンテキストで tskid=TSK_SELF (0) を指定)
E_NOEXS	未登録 (tskid のタスクが存在しない)
E_QOVR	キューイングのオーバーフロー (actcnt > 15)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

tskid で指定されたタスクを起動します。起動したタスクは休止状態から実行可能状態へ移行します。
tskid=TSK_SELF (0) の指定により、自タスクの指定になります。
対象タスクには、タスク生成時に指定したタスクの拡張情報がパラメータとして渡ります。
対象タスクが休止状態でない場合には、本サービスコールによるタスクの起動要求は、最大15 回まで記憶されます。

7.1.4 自タスクの終了 (ext_tsk) 自タスクの終了と削除 (exd_tsk)

□ C言語API

```
void ext_tsk(void);
void exd_tsk(void);
```

□ パラメータ

なし

□ リターンパラメータ

サービスコールの呼び出し元には戻りません。
ただし、本サービスコールをシステム構築時に組み込まずに呼び出した場合には、R0 にE_RSFN が設定されてリターンします。
また、以下のエラーが発生するとシステムダウンとなります。

E_CTX コンテキストエラー(許可されていないシステム状態からの呼び出し)

□ エラーコード

なし

□ 機能

ext_tsk サービスコールは、自タスクを正常終了します。タスクの状態は、実行状態から休止状態へ移行します。起動要求がキューイングされている場合は、自タスクをいったん終了させた後に再起動します。

exd_tsk サービスコールは、自タスクを正常終了し、さらに削除します。タスクの状態は、実行状態から未登録状態へ移行します。

ext_tsk, exd_tsk サービスコールは、タスクが占有していた資源（セマフォやメモリブロックなど）を自動的に解放する機能はありません。タスクは、必ず終了する前に資源の解放を行ってください。

7.1.5 タスクの強制終了 (ter_tsk)

□ C言語API

```
ER ercd = ter_tsk(ID tskid);
```

□ パラメータ

ID	tskid	タスクID

□ リターンパラメータ

ER ercd	正常終了 (E_OK) またはエラーコード

□ エラーコード

E_ID	不正ID 番号($tskid \leq 0$, $tskid > CFG\_MAXTSKID$)
E_NOEXS	未登録($tskid$ のタスクが存在しない)
E_OBJ	オブジェクト状態不正($tskid$ のタスクが休止状態)
E_ILUSE	サービスコール不正使用(対象タスクが自タスク)
E_CTX	コンテキストエラー(許可されていないシステム状態からの呼び出し)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

$tskid$ で指定された他タスクを強制的に終了させます。終了させた他タスクは休止状態へ移行します。起動要求がキューイングされている場合には、タスクを起動する際に行うべき処理を行い、対象タスクを実行可能状態に移行します。

本サービスコールは、タスクが占有していた資源（セマフォやメモリブロックなど）を自動的に解放する機能はありません。タスクは、必ず終了する前に資源の解放を行ってください。

7.1.6 タスク優先度の変更 (chg_pri)

□ C言語API

```
ER ercd = chg_pri(ID tskid, PRI tskpri);
```

□ パラメータ

ID	tskid	タスク ID
PRI	tskpri	タスクのベース優先度

□ リターンパラメータ

ER	ercd	正常終了 (E_OK) またはエラーコード

□ エラーコード

E_PAR	パラメータエラー (tskpri < 0, tskpri > CFG_MAXTSKPRI)
E_ID	不正ID 番号 (tskid < 0, tskid > CFG_MAXTSKID、非タスクコンテキストで tskid=TSK_SELF (0) を指定)
E_NOEXS	未登録 (tskid のタスクが存在しない)
E_ILUSE	サービスコール不正使用 (上限優先度の違反)
E_OBJ	オブジェクト状態不正 (タスクが休止状態である)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

tskid で示されたタスクのベース優先度を、tskpri で示された値に変更します。それに伴い、タスクの現在優先度も変更します。tskid=TSK_SELF (0) の指定により、自タスク指定となります。

tskpri=TPRI_INI (0) の指定により、タスク生成時に指定した初期タスク優先度に戻します。

変更したタスク優先度は、タスクが終了、または本サービスコールを呼び出すまで有効です。タスクが休止状態になると終了前のタスク優先度は無効になり、次に起動されたときにはタスク生成時に指定した初期タスク優先度になります。

7.1.7 タスクの状態参照 (ref_tsk)

□ C言語API

```
ER ercd = ref_tsk(ID tskid, T_RTSK *pk_rtsk);
```

□ パラメータ

ID	tskid	タスクID
T_RTSK	*pk_rtsk	タスク状態を返すパケットへのポインタ

□ リターンパラメータ

ER	ercd	正常終了 (E_OK) またはエラーコード
T_RTSK	*pk_rtsk	タスク状態を格納したパケットへのポインタ

□ パケットの構造

```
typedef struct t_rtsk {
    STAT    tskstat;      タスク状態
    PRI     tskpri;       タスクの現在優先度
    PRI     tsbpri;       タスクのベース優先度
    STAT    tskwait;      待ち要因
    ID      wobjid;       待ちオブジェクトID
    TMO     lefttmo;      タイムアウトするまでの時間
    UINT    actcnt;       起動要求キューイング数
    UINT    wupcnt;       起床要求キューイング数
    UINT    suscnt;       強制待ち要求ネスト数
    UINT    tskmode;      タスクの実行モード
    UINT    tflptn;       現在のタスク付属イベントフラグの値
} T_RTSK;
```

□ エラーコード

E_PAR	パラメータエラー(pk_rtsk が4 の倍数以外)
E_ID	不正ID 番号(tskid<0、tskid>CFG_MAXTSKID、非タスクコンテキストで tskid=TSK_SELF(0)を指定)
E_NOEXS	未登録(tskid のタスクが存在しない)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

tskid で示されたタスクの状態を参照し、pk_rtsk が指す領域に返します。
tskid=TSK_SELF (0) の指定により自タスクの指定になります。

pk_rtsk の指す領域には、以下の値を返します。なお、*のデータはタスクが休止状態の場合は無効です。また、機能が組み込まれていないものに対する情報は不定となります。

◆ tskstat

現在のタスクの状態です。tskstat には、次の値を返します。

- ・ TTS_RUN (H' 00000001) 実行状態
- ・ TTS_RDY (H' 00000002) 実行可能状態
- ・ TTS_WAI (H' 00000004) 待ち状態
- ・ TTS_SUS (H' 00000008) 強制待ち状態
- ・ TTS_WAS (H' 0000000c) 二重待ち状態
- ・ TTS_DMT (H' 00000010) 休止状態
- ・ TTS_STK (H' 40000000) 共有スタック解放待ち状態

◆ tskpri

タスクの現在優先度です。タスクが休止状態の場合は、タスクの初期優先度を返します。

◆ tskbpri

タスクのベース優先度です。タスクが休止状態の場合は、タスクの初期優先度を返します。

◆ tsawait *

tskstat がTTS_WAI、TTS_WAS のときに有効で、次の値を返します。

- ・ TTW_SLP (H' 00000001) slp_tsk、tslp_tsk サービスコールによる待ち
- ・ TTW_DLY (H' 00000002) dly_tsk サービスコールによる待ち
- ・ TTW_SEM (H' 00000004) wai_sem、twai_sem サービスコールによる待ち
- ・ TTW_FLG (H' 00000008) wai_flg、twai_flg サービスコールによる待ち
- ・ TTW_SDTQ (H' 00000010) snd_dtq、tsnd_dtq サービスコールによる待ち
- ・ TTW_RDTQ (H' 00000020) rcv_dtq、trcv_dtq サービスコールによる待ち
- ・ TTW_MBX (H' 00000040) rcv_mbx、trcv_mbx サービスコールによる待ち
- ・ TTW_MTX (H' 00000080) loc_mtx、tloc_mtx サービスコールによる待ち
- ・ TTW_SMBF (H' 00000100) snd_mbf、tsnd_mbf サービスコールによる待ち
- ・ TTW_RMBF (H' 00000200) rcv_mbf、trcv_mbf サービスコールによる待ち
- ・ TTW_MPF (H' 00002000) get_mpf、tget_mpf サービスコールによる待ち
- ・ TTW_MPL (H' 00004000) get_mpl、tget_mpl サービスコールによる待ち
- ・ TTW_TFL (H' 00008000) vwai_tfl、vtwai_tfl サービスコールによる待ち

◆ wobjid *

tskstat がTTS_WAI、TTS_WAS のときに有効で、待ち対象のオブジェクトID を返します。

◆ lefttmo *

対象タスクがタイムアウトするまでの時間を返します。対象タスクがdly_tsk サービスコールによる待ち状態の場合は、この値は不定値となります。

◆ actcnt *

現在の起動要求キューイング数を返します。

◆ wupcnt *

現在の起床要求キューイング数を返します。

◆ suscnt *

現在の強制待ち要求ネスト数を返します。

◆ tskmode *

vchg_tmd サービスコールで設定したタスク実行モードと、それによって遅延されている要求があるかを返します。tskmode には次の値を返します。

- ・ ECM_SUS (H' 00000001) 強制待ち要求がマスクされている
- ・ ECM_TER (H' 00000002) 強制終了要求がマスクされている
- ・ PND_SUS (H' 00000004) 強制待ち要求が遅延されている
- ・ PND_TER (H' 00000008) 強制終了要求が遅延されている

◆ tflptn *

現在のタスク付属イベントフラグの値を返します。ただし、タスク付属イベントフラグ機能を組み込んでいない場合には、この値は不定値を返します。

tskmode およびtflptn は、 μ ITRON4.0 仕様の範囲外のメンバです。

7.1.8 起床待ち (slp_tsk, tslp_tsk)

□ C言語API

```
ER ercd = slp_tsk(void);
ER ercd = tslp_tsk(TMO tmout);
```

□ パラメータ

《tslp_tsk》	
TMO tmout	タイムアウト指定

□ リターンパラメータ

ER ercd	正常終了 (E_OK) またはエラーコード

□ エラーコード

E_PAR	パラメータエラー (tmout $\leq$ -2)
E_CTX	コンテキストエラー (許可されていないシステム状態からの呼び出し)
E_TMOUT	タイムアウト
E_RLWAI	待ち状態強制解除 (待ちの間に rel_wai サービスコールが呼び出された)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

自タスクを起床待ち状態に移行させます。ただし、自タスクに対する起床要求がキューイングされている場合は、起床要求カウントを1減らしてそのまま実行を継続します。

起床待ち状態は、wup_tsk サービスコールによって解除されます。

tslp_tsk サービスコールでは、tmout には待ち時間を指定します。

tmout に正の値を指定した場合、待ち状態のままtmout 時間が経過すると、待ち状態は解除され、エラーコードとしてE_TMOUT が返ります。

tmout=TMO_POL (0) を指定した場合、起床要求カウントが正なら起床要求カウントを1減らして実行を継続し、0 ならエラーコードとしてE_TMOUT を返します。

tmout=TMO_FEVR (-1) を指定した場合、タイムアウト監視を行いません。この場合、slp_tsk サービスコールと同じ動作となります。

7.1.9 タスクの起床 (wup_tsk)

□ C言語API

```
ER ercd = wup_tsk(ID tskid);
```

□ パラメータ

ID	tskid	タスクID

□ リターンパラメータ

ER	ercd	正常終了 (E_OK) またはエラーコード

□ エラーコード

E_ID	不正ID 番号 (tskid < 0、tskid > CFG_MAXTSKID、非タスクコンテキストで tskid=TSK_SELF(0) を指定)
E_NOEXS	未登録 (tskid のタスクが存在しない)
E_OBJ	オブジェクト状態不正 (tskid のタスクが休止状態である)
E_QOVR	キューイングのオーバーフロー (wupcnt > 15)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

slp_tsk、またはtslp_tsk サービスコールの呼び出しにより待ち状態になっているタスクの待ち状態を解除します。

対象タスクがslp_tsk、またはtslp_tsk サービスコールによる待ち状態でない場合には、本サービスコールによる起床要求は、最大15 回まで記憶されます。

tskid=TSK_SELF (0) の指定により自タスクの指定になります。

7.1.10 タスク起床要求のキャンセル (can_wup)

□ C言語API

```
ER_UINT wupcnt = can_wup(ID tskid);
```

□ パラメータ

ID	tskid	タスクID

□ リターンパラメータ

ER_UINT	wupcnt	キューイングされていた起床要求の回数(正の値または0) またはエラーコード

□ エラーコード

E_ID	ID 範囲外 (tskid<0、tskid>CFG_MAXTSKID、非タスクコンテキストで tskid=TSK_SELF(0)を指定)
E_NOEXS	未登録 (tskid のタスクが生成されていない)
E_OBJ	オブジェクト状態不正 (tskid のタスクが休止状態である)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

tskid で示されたタスクにキューイングされていた起床要求回数を求め、その結果をリターンパラメータとして返し、同時にその起床要求を全て無効にします。

tskid=TSK_SELF (0) の指定により、自タスクの指定になります。

7.1.11 強制待ち状態への移行 (sus_tsk)

□ C言語API

```
ER ercd = sus_tsk(ID tskid);
```

□ パラメータ

ID	tskid	タスクID

□ リターンパラメータ

ER	ercd	正常終了 (E_OK) またはエラーコード

□ エラーコード

E_ID	不正ID 番号($tskid < 0$ 、 $tskid > CFG\_MAXTSKID$ 、非タスクコンテキストで $tskid=TSK\_SELF(0)$ を指定)
E_NOEXS	未登録($tskid$ のタスクが存在しない)
E_OBJ	オブジェクト状態不正($tskid$ のタスクが休止状態である)
E_CTX	コンテキストエラー(タスクコンテキストかつディスパッチ禁止状態で $tskid=TSK\_SELF(0)$ または自タスクID を指定)
E_QOVR	キューイングのオーバーフロー($suscnt > 15$)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

$tskid$ で示されたタスクの実行を中断させ、強制待ち状態へ移行します。 $tskid$ で示されたタスクが待ち状態にある場合は、二重待ち状態へ移行します。

$tskid=TSK\_SELF(0)$ の指定により自タスクの指定になります。

強制待ち状態は、`rsm_tsk` サービスコールの呼び出しにより解除されます。

本サービスコールによる強制待ちの要求はネストします。強制待ち要求のネスト数は最大15 回まで記憶されます。

7.1.12 強制待ち状態からの再開 (rsm_tsk)

□ C言語API

```
ER ercd = rsm_tsk(ID tskid);
```

□ パラメータ

ID	tskid	タスクID

□ リターンパラメータ

ER	ercd	正常終了 (E_OK) またはエラーコード

□ エラーコード

E_ID	不正ID 番号($tskid < 0$ 、 $tskid > CFG\_MAXTSKID$)
E_NOEXS	未登録($tskid$ のタスクが存在しない)
E_OBJ	オブジェクトの状態不正($tskid$ のタスクが休止状態である、 $tskid$ のタスクが強制待ち状態でない、または自タスク指定)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

$tskid$ で示されたタスクの強制待ち状態を解除します。

具体的には、rsm_tskサービスコールは、 $tskid$ で示されたタスクの強制待ち要求ネスト数を1 減算し、その結果が0 になった場合に強制待ち状態を解除します。

7.1.13 タスク遅延 (dly_tsk)

□ C言語API

```
ER ercd = dly_tsk(RELTIM dlytim);
```

□ パラメータ

RELTIM	dlytim	遅延時間

□ リターンパラメータ

ER	ercd	正常終了 (E_OK) またはエラーコード

□ エラーコード

E_CTX	コンテキストエラー(許可されていないシステム状態からの呼び出し)
E_RLWAI	待ち状態強制解除(待ちの間にrel_wai サービスコールが呼び出された)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

自タスクの状態を実行状態から時間経過待ち状態へ移行し、dlytim で指定された時間が経過するのを待ちます。dlytim で指定された時間が経過した時点で、自タスクの状態を実行可能状態に移行します。dlytim=0 を指定した場合にも、自タスクを待ち状態に移行させます。

本サービスコールは、tslp_tsk サービスコールとは異なり、dlytim 時間だけ実行を遅延して終了した場合に正常終了します。また、遅延時間中にwup_tsk サービスコールが実行されても、待ち状態は解除されません。遅延時間が経過する前に待ち状態を解除するのは、rel_wai または ter_tsk サービスコールが呼び出された場合に限られます。

7.1.14 セマフォの生成 (acre_sem)

□ C言語API

```
ER_ID semid = acre_sem(T_CSEM *pk_csem);
```

□ パラメータ

T_CSEM	*pk_csem	セマフォ生成情報を格納したパケットへのポインタ

□ リターンパラメータ

ER_ID	semid	生成したセマフォのID 番号 (正の値) またはエラーコード

□ パケットの構造

```
typedef struct t_csem {
    ATR      sematr;      セマフォ属性
    UINT      isemcnt;    セマフォの資源数の初期値
    UINT      maxsem;     セマフォの最大資源数
} T_CSEM;
```

□ エラーコード

E_RSATR	属性不正 (sema _{tr} が不正)
E_PAR	パラメータエラー (pk_csem が4 の倍数以外、maxsem=0、maxsem>H' ffff、isemcnt>maxsem)
E_ID	不正ID 番号 (semid≤0、semid>CFG_MAXSEMI _D)
E_OBJ	オブジェクト状態不正 (semid のセマフォが存在)
E_NOID	空きID なし

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

acre_sem サービスコールは、未登録のセマフォID を検索して、そのID を持つセマフォをpk_csem で示された内容で生成し、そのID をリターンパラメータとして返します。未登録のセマフォID を検索する範囲は1~CFG_MAXSEMI_D です。

sema_{tr} には属性として、セマフォ資源獲得待ちタスクが待ち行列に並ぶ際の並び方を指定します。

sema_{tr} := (TA_TFIFO || TA_TPRI)

- ・TA_TFIFO (H' 00000000) 待ちタスクのキューイングはFIFO
- ・TA_TPRI (H' 00000001) 待ちタスクのキューイングは優先度順

isemcnt には、生成するセマフォの初期値を指定します。指定できる値の範囲は、0 からmaxsem です。

maxsem には、生成するセマフォの最大資源数を指定します。指定できる値の範囲は、1 から65,535 です。

7.1.15 セマフォの削除 (del_sem)

□ C言語API

```
ER ercd = del_sem(ID semid);
```

□ パラメータ

ID	semid	セマフォID

□ リターンパラメータ

ER	ercd	正常終了 (E_OK) またはエラーコード

□ エラーコード

E_ID	不正ID 番号 (semid $\leq$ 0、semid $>$ CFG_MAXSEMID)
E_NOEXS	未登録 (semid のセマフォが存在しない)
E_CTX	コンテキストエラー (許可されていないシステム状態からの呼び出し)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

semid で示されたセマフォを削除します。

semid で示されたセマフォで資源獲得を待っているタスクがあった場合でもエラーにはなりません
が、待ち状態だったタスクは待ち状態が解除され、エラーコードとしてE_DLT が返されます。

7.1.16 セマフォ資源の返却 (sig_sem)

□ C言語API

```
ER ercd = sig_sem(ID semid);
```

□ パラメータ

ID	semid	セマフォID

□ リターンパラメータ

ER	ercd	正常終了 (E_OK) またはエラーコード

□ エラーコード

E_ID	不正ID 番号 (semid $\leq$ 0、semid $>$ CFG_MAXSEMIC)
E_NOEXS	未登録 (semid のセマフォが存在しない)
E_QOVR	キューイングオーバーフロー (semcnt $>$ maxsem) ※1

※1 maxsem: セマフォ生成時に指定した最大セマフォ資源数

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

semid で示されたセマフォに資源をひとつ返却します。対象セマフォで待っているタスクがあれば、セマフォの待ち行列先頭タスクに資源を割り付けて待ち状態を解除します。セマフォに対して待っているタスクがなければ、そのセマフォのカウント値を1 増やします。

セマフォのカウント値の最大値は、セマフォ生成時に指定したmaxsem です。

7.1.17 セマフォ資源の獲得 (wai_sem, pol_sem, twai_sem)

□ C言語API

```
ER ercd = wai_sem(ID semid);
ER ercd = pol_sem(ID semid);
ER ercd = twai_sem(ID semid, TMO tmout);
```

□ パラメータ

ID	semid	セマフォID
《twai_sem》		
TMO	tmout	タイムアウト指定

□ リターンパラメータ

ER	ercd	正常終了 (E_OK) またはエラーコード

□ エラーコード

E_PAR	パラメータエラー (tmout $\leq$ -2)
E_ID	不正ID 番号 (semid $\leq$ 0, semid > CFG_MAXSEMIC)
E_NOEXS	未登録 (semid のセマフォが存在しない)
E_CTX	コンテキストエラー (許可されていないシステム状態からの呼び出し)
E_DLT	待ちオブジェクト削除 (semid のセマフォが削除された)
E_RLWAI	待ち状態強制解除 (待ちの間にrel_wai サービスコールが呼び出された)
E_TMOUT	ポーリング失敗、またはタイムアウト

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

semid で指定されるセマフォから、資源をひとつ獲得します。

対象セマフォの資源数が 1 以上の場合には、セマフォの資源数から 1 を減じ、実行を続けます。資源数が 0 の場合には、wai_sem, twai_sem サービスコールでは呼び出しタスクはそのセマフォの待ち行列につながれ、pol_sem サービスコールでは直ちにエラーE_TMOUT で終了します。待ち行列は、生成時に指定した属性にしたがって管理されます。

twai_sem サービスコールの場合、tmout には待ち時間を指定します。

tmout に正の値を指定した場合、待ち解除の条件が満たされないままtmout 時間が経過すると、エラーコードとしてE_TMOUT を返します。tmout=TMO_POL (0) を指定した場合、pol_sem サービスコールと同じ処理を行います。tmout=TMO_FEVR (-1) を指定した場合、タイムアウト監視を行いません。この場合、wai_sem サービスコールと同じ動作となります。

7.1.18 セマフォの状態参照 (ref_sem)

□ C言語API

```
ER ercd = ref_sem(ID semid, T_RSEM *pk_rsem);
```

□ パラメータ

ID	semid	セマフォID
T_RSEM	*pk_rsem	セマフォ状態を返すパケットへのポインタ

□ リターンパラメータ

ER	ercd	正常終了 (E_OK) またはエラーコード
T_RSEM	*pk_rsem	セマフォ状態を格納したパケットへのポインタ

□ パケットの構造

```
typedef struct t_rsem {
    ID wtskid;           待ちタスクID
    UINT semcnt;        現在のセマフォカウント値
} T_RSEM;
```

□ エラーコード

E_PAR	パラメータエラー(pk_rsem が4 の倍数以外)
E_ID	不正ID 番号(semid $\leq$ 0、semid>CFG_MAXSEMid)
E_NOEXS	未登録(semid のセマフォが存在しない)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

semid で示されたセマフォの状態を参照します。
 pk_rsem が指す領域に、待ち行列の先頭タスクID(wtskid)、現在のセマフォカウント値(semcnt)を返します。
 対象セマフォの待ちタスクが無い場合は、待ちタスクID としてTSK_NONE (0) を返します。

7.1.19 イベントフラグの生成 (acre_flg)

□ C言語API

```
ER_ID flgid = acre_flg(T_CFLG *pk_cflg);
```

□ パラメータ

T_CFLG *pk_cflg イベントフラグ生成情報を格納したパケットへのポインタ

□ リターンパラメータ

ER_ID flgid 生成したイベントフラグのID 番号 (正の値)
 またはエラーコード

□ パケットの構造

```
typedef struct t_cflg {
    ATR      flgatr;           イベントフラグ属性
    FLGPTN   iflgptn;         イベントフラグの初期値
} T_CFLG;
```

□ エラーコード

E_RSATR	属性不正 (flgatr が不正)
E_PAR	パラメータエラー (pk_cflg が4 の倍数以外)
E_ID	不正ID 番号 (flgid ≤ 0, flgid > CFG_MAXFLGID)
E_OBJ	オブジェクト状態不正 (flgid のイベントフラグが存在)
E_NOID	空きID なし

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

acre_flg サービスコールは、未登録のイベントフラグID を検索してそのID を持つイベントフラグを pk_cflg で示された内容で生成し、そのID をリターンパラメータとして返します。未登録のイベントフラグID を検索する範囲は1〜CFG_MAXFLGID です。

flgatr には属性として、イベントフラグ待ちタスクが待ち行列に並ぶ際の並び方と、生成するイベントフラグに待つことのできるタスク数を指定します。

```
flgatr := ((TA_TFIFO || TA_TPRI) | (TA_WSGL || TA_WMUL) | [TA_CLR])
```

- ・ TA_TFIFO (H' 00000000) 待ちタスクのキューイングはFIFO
- ・ TA_TPRI (H' 00000001) 待ちタスクのキューイングは優先度順
- ・ TA_WSGL (H' 00000000) 複数タスクの待ちを許さない
- ・ TA_WMUL (H' 00000002) 複数タスクの待ちを許す
- ・ TA_CLR (H' 00000004) 待ち解除時にイベントフラグを0 クリア

TA_WSGL 属性のイベントフラグでは、待つことのできるタスクは1つになります。この場合は、TA_TFIFO とTA_TPRI のどちらの属性を指定してもイベントフラグの動作は同じになります。これに対して、TA_WMUL 属性のイベントフラグでは、複数のタスクが待ち状態に遷移することができます。TA_CLR 属性が指定された場合は、タスクをイベントフラグ待ち状態から解除する時に、イベントフラグのビットパターンすべてのビットをクリアします。

iflgptn には、生成するイベントフラグの初期値を指定します。

7.1.20 イベントフラグの削除 (del_flg)

□ C言語API

```
ER ercd = del_flg(ID flgid);
```

□ パラメータ

ID	flgid	イベントフラグID

□ リターンパラメータ

ER	ercd	正常終了 (E_OK) またはエラーコード

□ エラーコード

E_ID	不正ID 番号($flgid \leq 0$ 、 $flgid > CFG\_MAXFLGID$)
E_NOEXS	未登録($flgid$ のイベントフラグが存在しない)
E_CTX	コンテキストエラー(許可されていないシステム状態からの呼び出し)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

flgid で示されたイベントフラグを削除します。

flgid で示されたイベントフラグで条件成立を待っているタスクがあった場合でもエラーにはなりません、待ち状態だったタスクは待ち状態が解除され、エラーコードとしてE_DLT が返されます。

7.1.21 イベントフラグのセット (set_flg)

□ C言語API

```
ER ercd = set_flg(ID flgid, FLGPTN setptn);
```

□ パラメータ

ID	flgid	イベントフラグID
FLGPTN	setptn	セットするビットパターン

□ リターンパラメータ

ER	ercd	正常終了 (E_OK) またはエラーコード

□ エラーコード

E_ID	不正ID 番号($flgid \leq 0$, $flgid > CFG\_MAXFLGID$)
E_NOEXS	未登録($flgid$ のイベントフラグが存在しない)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

flgid で示されたイベントフラグを、setptn で示された値との論理和 (OR) に更新します。

イベントフラグ値の更新の結果、そのイベントフラグで待っているタスクの待ち解除条件を満たせば、そのタスクの待ち状態を解除します。なお、待ち解除条件は待ち行列の順に評価されます。この時、対象のイベントフラグ属性にTA_CLR 属性が指定されている場合には、イベントフラグのビットパターンのすべてのビットをクリアし、サービスコールの処理を終了します。

イベントフラグにTA_WMUL 属性が指定されており、TA_CLR 属性が指定されていない場合、set_flg の1 回の呼び出しで複数のタスクが待ち解除される可能性があります。待ち解除されるタスクが複数ある場合に、イベントフラグの待ち行列につながれていた順序で待ち解除されます。

7.1.22 イベントフラグのクリア (clr_flg)

□ C言語API

```
ER ercd = clr_flg(ID flgid, FLGPTN clrptn);
```

□ パラメータ

ID	flgid	イベントフラグID
FLGPTN	clrptn	クリアするビットパターン

□ リターンパラメータ

ER	ercd	正常終了 (E_OK) またはエラーコード

□ エラーコード

E_ID	不正ID 番号($flgid \leq 0$ 、 $flgid > CFG\_MAXFLGID$)
E_NOEXS	未登録($flgid$ のイベントフラグが存在しない)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

$flgid$ で示されたイベントフラグを、 $clrptn$ で示された値との論理積 (AND) に更新します。

7.1.23 イベントフラグ待ち (wai_flg, pol_flg, twai_flg)

□ C言語API

```
ER ercd = wai_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn);
ER ercd = pol_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn);
ER ercd = twai_flg(ID flgid, FLGPTN waiptn, MODE wfmode, FLGPTN *p_flgptn, TMO tmout);
```

□ パラメータ

ID	flgid	イベントフラグID
FLGPTN	waiptn	待ちビットパターン
MODE	wfmode	待ちモード
FLGPTN	*p_flgptn	待ち解除時のビットパターンを返す領域へのポインタ
《twai_flg》		
TMO	tmout	タイムアウト値

□ リターンパラメータ

ER	ercd	正常終了 (E_OK) またはエラーコード
FLGPTN	*p_flgptn	待ち解除時のビットパターンへのポインタ

□ エラーコード

E_PAR	パラメータエラー(p_flgptn が4 の倍数以外、waiptn=0、wfmode が不正、tmout $\leq$ -2)
E_ID	不正ID 番号(flgid $\leq$ 0、flgid>CFG_MAXFLGID)
E_NOEXS	未登録(flgid のイベントフラグが存在しない)
E_ILUSE	サービスコール不正使用(TA_WSGL 属性のイベントフラグに待ちタスクが存在)
E_CTX	コンテキストエラー(許可されていないシステム状態からの呼び出し)
E_DLT	待ちオブジェクト削除(flgid のイベントフラグが削除された)
E_TMOUT	ポーリング失敗、またはタイムアウト
E_RLWAI	待ち状態強制解除(待ちの間にrel_wai サービスコールが呼び出された)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

flgid で指定されるイベントフラグのビットパターンが、waiptn とwfmode で指定される待ち解除条件を満たすのを待ちます。p_flgptn の指す領域には、待ち解除される時のイベントフラグのビットパターンを返します。

対象イベントフラグがTA_WSGL 属性で、すでに他のタスクが待っている場合は、本サービスコールはエラーE_ILUSE となります。

本サービスコール呼び出し時にすでに待ち解除条件が成立している場合は、本サービスコールは直ちに終了します。待ち解除条件が成立していない場合は、wai_flg, twai_flg サービスコールの場合はイベント待ち行列につながれ、pol_flg, ipol_flg サービスコールの場合は直ちにエラーE_TMOUT で終了します。

wfmode には、次のような指定を行います。

wfmode := ((TWF_ANDW || TWF_ORW))

・TWF_ANDW (H' 00000000) AND 待ち

・TWF_ORW (H' 00000001) OR 待ち

TWF_ANDW では、waiptn で指定したビットの全てがセットされるのを待ちます。TWF_ORW では、flgid で示されたイベントフラグのうちwaiptn で指定したビットのいずれかがセットされるのを待ちます。

twai_flg サービスコールの場合、tmout には待ち時間を指定します。

tmout に正の値を指定した場合、待ち条件が満たされないままtmout 時間が経過すると、エラーコードとしてE_TMOUT を返します。tmout=TMO_POL (0) を指定した場合、pol_flg サービスコールと同じ処理を行います。tmout=TMO_FEVR (-1) を指定した場合、タイムアウト監視を行いません。この場合、wai_flg サービスコールと同じ動作となります。

7.1.24 イベントフラグの状態参照 (ref_flg)

□ C言語API

```
ER ercd = ref_flg(ID flgid, T_RFLG *pk_rflg);
```

□ パラメータ

ID	flgid	イベントフラグID
T_RFLG	*pk_rflg	イベントフラグ状態を返すパケットへのポインタ

□ リターンパラメータ

ER	ercd	正常終了 (E_OK) またはエラーコード
T_RFLG	*pk_rflg	イベントフラグ状態を格納したパケットへのポインタ

□ パケットの構造

```
typedef struct t_rflg {
    ID      wtskid;      待ちタスクID
    FLGPNTN flgpntn;    イベントフラグのビットパターン
} T_RFLG;
```

□ エラーコード

E_PAR	パラメータエラー(pk_rflg が4 の倍数以外)
E_ID	不正ID 番号 (flgid $\leq$ 0、flgid $>$ CFG_MAXFLGID)
E_NOEXS	未登録 (flgid のイベントフラグが存在しない)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

flgid で示されたイベントフラグの状態を参照します。
 pk_rflg の指す領域に、待ち行列の先頭タスクID(wtskid)、現在のイベントフラグのビットパターン(flgpntn)を返します。
 対象イベントフラグの待ちタスクが無い場合は、待ちタスクID としてTSK_NONE (0) を返します。

第8章 Windowsアプリケーションサポートコマンド

8.1 Windowsアプリケーションサポート

8.1.1 IMデータを BMPに変換し転送 (ReadBMPImg)

□ C言語API

```
int ret = ReadBMPImg( IMGID ImgID, ISizeInfo *pISize, char *pBmpImg,
                    void *pBmpInf, enum BITMAP_MODE mode );
```

□ パラメータ

IMGID	ImgID	画面番号
ISizeInfo	*pISize	転送範囲 IM上の転送開始座標と幅、高さを設定して下さい。 NULLを設定すると全画面サイズになります。
char	*pBmpImg	ビットマップ画像データを返すアドレス 転送サイズ分の領域を確保して下さい。
void	*pBmpInf	ビットマップ情報ヘッダ&LUTを返すアドレス BITMAPINFOHEADER構造体とRGBQUAD構造体 (BW_BITMAP設定時) のサイズ分の領域を確保して下さい。 NULLを設定するとビットマップ情報ヘッダ&LUTを出力しません。
enum BITMAP_MODE	mode	ビットマップファイル転送モード
	BW_BITMAP (0)	モノクロ画像 (8bit/dot: 256色)
	RGB_BITMAP (1)	カラー画像 (24bit/dot:16777216色)
	YUV_BITMAP (2)	YUV画像 (IP-X専用フォーマット)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	○	○	○	○	○	○	○	○

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)
char	*pBmpImg	ビットマップ画像データを格納したアドレス
void	*pBmpInf	ビットマップ情報ヘッダ&LUTを格納したアドレス

□ パケットの構造

```
typedef struct {
    int    sx;      X方向開始点座標
    int    sy;      Y方向開始点座標
    int    xlng;    X方向処理範囲
    int    ylng;    Y方向処理範囲
} ISizeInfo;
```

□ エラーコード

1 0	画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 2	転送モード指定範囲外
4 8	動的メモリ確保エラー
4 9	セーブを行う画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

I Mのデータをビットマップ画像データに変換してpBmpImgに読み出します。データはビットマップの形式に従い、左から右へ、下から上に向かって格納されます。また、水平方向のバイト数が4の倍数でない場合は0で埋めて4の倍数にします。このため、pBmpImgに必要な領域は以下のようになります。

ビットマップファイル 転送モード	水平方向のバイト数	pBmpImgに必要な領域 (バイト)
BW_BITMAP	$xlng \% 4 == 0$	$xlng * ylng$
	$xlng \% 4 != 0$	$(xlng + (4 - xlng \% 4)) * ylng$
RGB_BITMAP	$(xlng * 3) \% 4 == 0$	$xlng * ylng * 3$
	$(xlng * 3) \% 4 != 0$	$(xlng * 3 + (4 - (xlng * 3) \% 4)) * ylng$

pBmpImgの領域が足りない場合はメモリ破壊が発生しますので注意して下さい。

pBmpInfがNULL以外の場合、ビットマップ情報ヘッダとLUT情報がBmpInfに書き込まれます。NULLを設定すると書き込みは行いません。

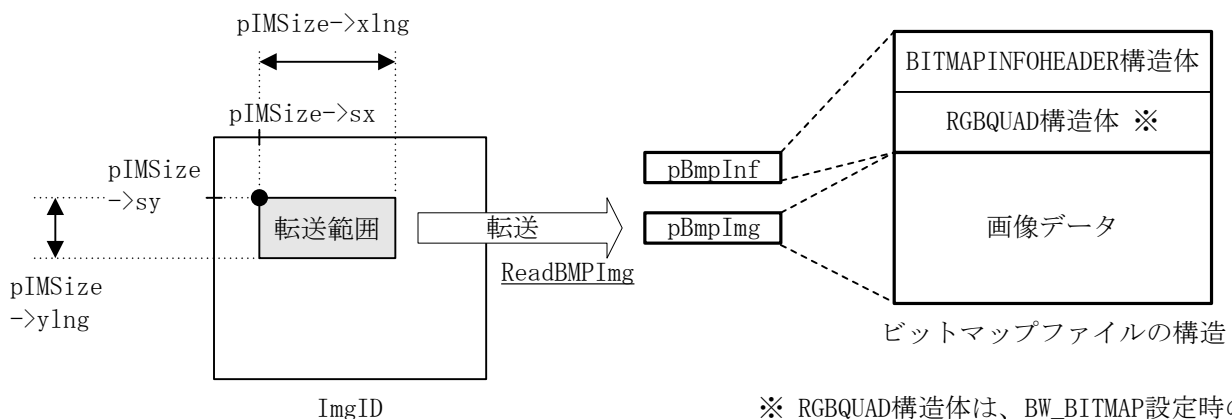
ビットマップ情報ヘッダ (BITMAPINFOHEADER) とLUT情報 (RGBQUAD) の構造体を以下に示します。

BITMAPINFOHEADER構造体

フィールド	バイト数	内容
biSize	4	構造体サイズ (40 B)
biWidth	4	ビットマップの幅 (ピクセル)
biHeight	4	ビットマップの高さ (ピクセル)
biPlanes	2	プレーン数
biBitCount	2	ピクセルあたりの色数 (1, 4, 8, 24)
biCompression	4	圧縮形式 (0なら圧縮なし)
biSizeImage	4	ビットマップサイズ (圧縮時のみ必要)
biXPelsPerMeter	4	水平解像度 (ピクセル/m)
biYPelsPerMeter	4	垂直解像度 (ピクセル/m)
biClrUsed	4	使われている色数
biClrImportant	4	使われている重要な色数

RGBQUAD構造体

フィールド	バイト数	内容
rgbBlue	1	青の強さ
rgbGreen	1	緑の強さ
rgbRed	1	赤の強さ
rgbReserved	1	予約領域 (0)



※ RGBQUAD構造体は、BW_BITMAP設定時のみ必要

以下に示す条件で画像メモリのデータをビットマップデータに変換し、読み出します。

mode 画面	BW_BITMAP	RGB_BITMAP	YUV_BITMAP
モノクロ (Y)	Y画面を読み出す	Y画面を読み出す	Y画面を読み出す
YUV	Y画面のみ読み出す	YUV画面をRGBデータに変換し、読み出す	Y画面を1面目、UV画面を2面目に読み出す
CYUV	Yデータを読み出す	CYUV画面をRGBデータに変換し、読み出す	Yデータを読み出す
RGB	R画面のみ読み出す	RGB画面を読み出す	R画面のみ読み出す
RGB_R	RGB_R画面を読み出す	RGB_R画面を読み出す	RGB_R画面を読み出す
RGB_G	RGB_G画面を読み出す	RGB_G画面を読み出す	RGB_G画面を読み出す
RGB_B	RGB_B画面を読み出す	RGB_B画面を読み出す	RGB_B画面を読み出す
RGB16	RGB16画面をYデータに変換し、読み出す	RGB16画面をRGBデータに変換し、読み出す	エラー

※ YUV画面、CYUV画面には対応していません。

□ サンプルコード

使用例)

```

IMSizeInfo IMSize;
LPSTR pBmpImg;
LPBITMAPINFOHEADER pBmpInf;
int width, height;

width = 640;
height = 512;

/* ビットマップ情報ヘッダ&LUTとビットマップ画像データの領域を確保 */
pBmpInf = (LPBITMAPINFOHEADER)malloc( sizeof(BITMAPINFOHEADER)+
                                       sizeof(RGBQUAD) * 256 + ( width * height ) );

/* ビットマップ画像データを返すアドレスをpBmpImgに設定 */
pBmpImg = (LPSTR)((LPSTR)pBmpInf + sizeof(BITMAPINFOHEADER)+ sizeof(RGBQUAD) * 256);

/* 転送範囲を設定 */
IMSize.sx = 0;
IMSize.sy = 0;
IMSize.xleng = width;
IMSize.yleng = height;

/* IMデータをBMPに変換し転送 */
ReadBMPImg( ImgID, &IMSize, pBmpImg, pBmpInf, BW_BITMAP );

/* BMPデータを表示(Windows API) */
StretchDIBits( dc.m_hDC, 0, 0, width, height, 0, 0, width, height,
               pBmpImg, (LPBITMAPINFO)pBmpInf, DIB_PAL_COLORS, SRCCOPY );

```

8.1.2 BMPデータをIMに転送 (WriteBMPImg)

□ C言語API

```
int ret = WriteBMPImg( IMGID ImgID, IMSizeInfo *pIMSize, char *pBmpImg,
                      void *pBmpInf, void *pOpt );
```

□ パラメータ

IMGID	ImgID	画面番号
IMSizeInfo	*pIMSize	転送範囲 IM上の転送開始座標と幅、高さを設定して下さい。 NULLを設定すると全画面サイズになります。
char	*pBmpImg	ビットマップ画像データを返すアドレス 転送サイズ分の領域を確保して下さい。
void	*pBmpInf	ビットマップ情報ヘッダ&LUTを返すアドレス BITMAPINFOHEADER構造体とRGBQUAD構造体 (biBitCount≤8の場合) のサイズ分の領域を確保して下さい。
void	*pOpt	オプションテーブル 現在、未使用の為、NULLを指定して下さい

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	○	○	○	○	○	○	○	○

□ リターンパラメータ

int ret 正常終了 (0) または異常終了 (- 1)

□ パケットの構造

```
typedef struct {
    int    sx;      X方向開始点座標
    int    sy;      Y方向開始点座標
    int    xlng;    X方向処理範囲
    int    ylng;    Y方向処理範囲
} IMSizeInfo;
```

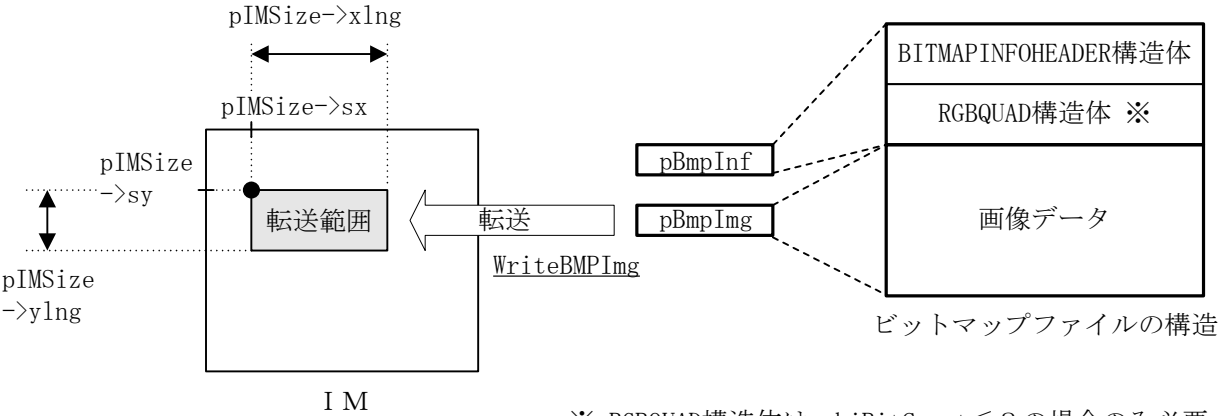
□ エラーコード

1 0 画面番号に対応する画面が未確保、または番号が有効範囲外
 (不当画面番号エラー)
4 8 動的メモリ確保エラー
1 0 1 不当画像ファイルエラー
— エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

pBmpImgに指定したビットマップ画像データ (左から右へ、下から上に向かってデータ) を pBmpInfのビットマップ情報ヘッダとLUTに従いIMに書き込みます。pBmpInfにNULLを設定するとエラーになります。ビットマップ情報ヘッダのビットマップの幅 (biWidth) が4の倍数でない場合、ビットマップ画像データの水平方向のバイト数を4の倍数にするために0で埋めた分を読み飛ばして、書き込みを行います。



※ RGBQUAD構造体は、biBitCount ≤ 8 の場合のみ必要。
BITMAPINFOHEADER構造体、およびRGBLUT構造体については、
『ReadBMPImg()』 コマンドを参照して下さい。

以下に示す条件でビットマップデータを画像メモリのデータに変換し、書き込みます。

BMP ファイル 画面	パレット 変換	B/W (モノクロ)	YUV	R G B (2 4 / 3 2 ビット)
モノクロ (Y)	有	パレットでB/WデータをYUVデータに変換し、YデータをY画面に書き込む	—	—
	無	B/WデータをY画面に書き込む	YデータをY画面に書き込む	R G BデータをYデータに変換し、Y画面に書き込む
YUV	有	パレットでB/WデータをYUVデータに変換し、YUV画面に書き込む	—	—
	無	B/WデータをY画面に書き込む。 UV画面は1 2 8でクリアする	YUVデータをYUV画面に書き込む	R G BデータをYUVデータに変換し、YUV画面に書き込む
C Y U V	有	パレットでB/WデータをC Y U Vデータに変換し、C Y U V画面に書き込む	—	—
	無	B/WデータをY画面に書き込む。 UV画面は1 2 8でクリアする	YUVデータをC Y U V画面に書き込む	R G BデータをC Y U Vデータに変換し、C Y U V画面に書き込む
R G B	有	パレットでB/WデータをR G Bデータに変換し、R G B画面に書き込む	—	—
	無	B/WデータをR G B画面 (R, G, Bそれぞれ) に書き込む。	エラー	R G BデータをR G B画面に書き込む。
R G B _ R	有	パレットでB/WデータをYUVデータに変換し、YデータをR G B _ R画面に書き込む	—	—
	無	B/WデータをR G B _ R画面に書き込む。	YデータをR G B _ R画面に書き込む	R G BデータをYデータに変換し、R G B _ R画面に書き込む
R G B _ G	有	パレットでB/WデータをYUVデータに変換し、YデータをR G B _ G画面に書き込む	—	—
	無	B/WデータをR G B _ G画面に書き込む。	YデータをR G B _ G画面に書き込む	R G BデータをYデータに変換し、R G B _ G画面に書き込む
R G B _ B	有	パレットでB/WデータをYUVデータに変換し、YデータをR G B _ B画面に書き込む	—	—
	無	B/WデータをR G B _ B画面に書き込む。	YデータをR G B _ B画面に書き込む	R G BデータをYデータに変換し、R G B _ B画面に書き込む

BMP ファイル 画面	パレット 変換	B/W (モノクロ)	YUV	RGB (24/32ビット)
RGB16	有	パレットでB/WデータをRGB16データに変換し、RGB16データをRGB16画面に書き込む	—	—
	無	B/WデータをRGB16画面に書き込む。	エラー	RGBデータをRGB16に変換し、RGB16画面に書き込む

※ パレット変換有. . . BMPファイルのパレット情報が標準（R、G、Bがそれぞれ0～255）と異なる場合

※ YUV画面、CYUV画面には対応していません。

□ サンプルコード

使用例)

```

IMSizeInfo IMSize;
LPSTR pBmpImg;
RGBQUAD *pRGBTbl;
LPBITMAPINFOHEADER pBmpInf;
int width, height;
int i;

width = 640;
height = 512;

/* ビットマップ情報ヘッダ&LUTとビットマップ画像データの領域を確保 */
pBmpInf = (LPBITMAPINFOHEADER)malloc( sizeof(BITMAPINFOHEADER) +
                                       sizeof(RGBQUAD) * 256 + ( width * height ) );

/* LUTのアドレスを設定 */
pRGBTbl = (RGBQUAD *)((LPSTR)pBmpInf + sizeof(BITMAPINFOHEADER));

/* ビットマップ画像データのアドレスを設定 */
pBmpImg = (LPSTR)((LPSTR)pRGBTbl + sizeof(RGBQUAD) * 256);

/* LUT作成 */
for(i=0; i<256; i++){
    pRGBTbl[i].rgbRed      = i;
    pRGBTbl[i].rgbGreen   = i;
    pRGBTbl[i].rgbBlue    = i;
    pRGBTbl[i].rgbReserved = 0;
}

/* ビットマップ画像データ作成 */
Pattern( pBmpImg, width, height );

/* ビットマップ情報ヘッダ設定 */
pBmpInf->biSize = sizeof( BITMAPINFOHEADER );
pBmpInf->biSizeImage = width * height;
pBmpInf->biWidth  = width;
pBmpInf->biHeight = height;
pBmpInf->biPlanes = 1;
pBmpInf->biBitCount = 8;

/* 転送範囲を設定 */
IMSize.sx = 0;
IMSize.sy = 0;
IMSize.xlmg = width;
IMSize.ylmg = height;

/* BMPデータをIMに転送 */
WriteBMPImg( ImgID, &IMSize, pBmpImg, pBmpInf, NULL );

```


8.1.3 画像メモリビットマップオブジェクトの生成 (CreateBMPImgObject)

□ C言語API

```
int ret = CreateBMPImgObject( void **pBmpObject, IMGID ImgID,
                             ISizeInfo *pIMSize, enum BITMAP_MODE mode );
```

□ パラメータ

void	**pBmpObject	画像メモリビットマップオブジェクトアドレスを返すアドレス
IMGID	ImgID	画面番号
ISizeInfo	*pIMSize	転送範囲 IM上の転送開始座標と幅、高さを設定して下さい。 NULLを設定すると全画面サイズになります。
enum BITMAP_MODE	mode	ビットマップファイル転送モード
	BW_BITMAP (0)	モノクロ画像 (8bit/dot: 256色)
	RGB_BITMAP (1)	カラー画像 (24bit/dot:16777216色)
	YUV_BITMAP (2)	YUV画像 (IP-X専用フォーマット)

□ 対応画面タイプ／画面データタイプ

	画面タイプ						画面データタイプ		
	Y	RGB	RGB. R	RGB. G	RGB. B	RGB16	U	S	B
ImgID	○	○	○	○	○	○	○	○	○

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)
void	**pBmpObject	画像メモリビットマップオブジェクトアドレスを格納するアドレス

□ パケットの構造

```
typedef struct {
    int    sx;      X方向開始点座標
    int    sy;      Y方向開始点座標
    int    xlng;    X方向処理範囲
    int    ylng;    Y方向処理範囲
} ISizeInfo;
```

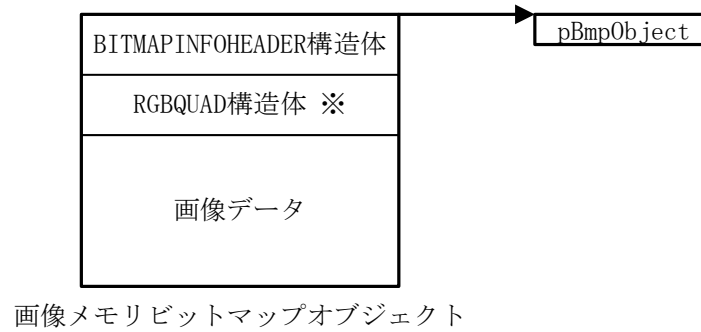
□ エラーコード

1 0	画面番号に対応する画面が未確保、または番号が有効範囲外 (不当画面番号エラー)
2 2	転送モード指定範囲外
4 8	動的メモリ確保エラー
4 9	セーブを行う画面が不定画面
—	エラーリセットコマンド (ClearIPError) 未発行

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

画像メモリビットマップオブジェクトの生成を行います。画像メモリビットマップオブジェクトはビットマップ情報ヘッダ&LUT (BW_BITMAP設定時) とビットマップ画像データで構成されています。本コマンドは、画像メモリビットマップオブジェクトの領域を内部で確保し、ビットマップ情報ヘッダ&LUT作成、IMデータのBMP変換、転送を行い、pBmpObjectに画像メモリビットマップオブジェクトの先頭アドレスを返します。



※ RGBQUAD構造体は、 $biBitCount \leq 8$ の場合のみ必要。
 BITMAPINFOHEADER構造体、およびRGBLUT構造体については、
 『ReadBMPImg()』 コマンドを参照して下さい。

・画面タイプとビットマップファイル転送モードの対応については、『ReadBMPImg()』 コマンドを参照して下さい。

□ サンプルコード

使用例)

```
IMSizeInfo IMSize;
LPSTR pBmpImg;
LPBITMAPINFOHEADER pBmpInf;
void *pBmpObject;
int width, height;

width = 640;
height = 512;

/* 転送範囲を設定 */
IMSize.sx = 0;
IMSize.sy = 0;
IMSize.xleng = width;
IMSize.yleng = height;

/* 画像メモリビットマップオブジェクト生成 */
CreateBMPImgObject( &pBmpObject, ImgID, &IMSize, BW_BITMAP );

/* ビットマップ情報ヘッダの先頭アドレス取得 */
pBmpInf = (LPBITMAPINFOHEADER)GetBMPImgInfoEntry( pBmpObject );

/* ビットマップ画像データの先頭アドレス取得 */
pBmpImg = (char*)GetBMPImgBitsEntry( pBmpObject );

/* BMPデータを表示(Windows API) */
StretchDIBits( dc.m_hDC, 0, 0, width, height, 0, 0, width, height,
               pBmpImg, (LPBITMAPINFO)pBmpInf, DIB_PAL_COLORS, SRCCOPY );

/* 画像メモリビットマップオブジェクト削除 */
DeleteBMPImgObject( pBmpObject );
```

8.1.4 画像メモリビットマップオブジェクトの削除 (DeleteBMPImgObject)

□ C言語API

```
int ret = DeleteBMPImgObject( void *pBmpObject );
```

□ パラメータ

void *pBmpObject 画像メモリビットマップオブジェクトを返すアドレス

□ リターンパラメータ

int ret 正常終了 (0)

□ エラーコード

なし

□ 機能

CreateBMPImgObjectで生成した画像メモリビットマップオブジェクトの削除を行います。

8.1.5 ビットマップ情報ヘッダー先頭アドレスの取得 (GetBMPImgInfoEntry)

□ C言語API

```
void *pBmpInf = *GetBMPImgInfoEntry( void *pBmpObject );
```

□ パラメータ

void *pBmpObject 画像メモリビットマップオブジェクトを返すアドレス

□ リターンパラメータ

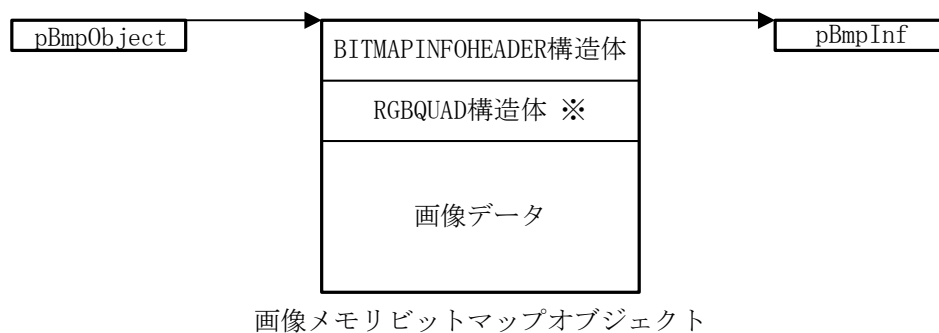
void *pBmpInf != NULL ビットマップ情報ヘッダーの先頭アドレス
 == NULL エラー

□ エラーコード

なし

□ 機能

CreateBMPImgObjectで生成した画像メモリビットマップオブジェクトからビットマップ情報ヘッダーの先頭アドレスを取得します。



※ RGBQUAD構造体は、BW_BITMAP設定時のみ必要

8.1.6 ビットマップデータ先頭アドレスの取得 (GetBMPImgBitsEntry)

□ C言語API

```
void *pBmpImg = *GetBMPImgBitsEntry( void *pBmpObject );
```

□ パラメータ

void *pBmpObject 画像メモリビットマップオブジェクトを返すアドレス

□ リターンパラメータ

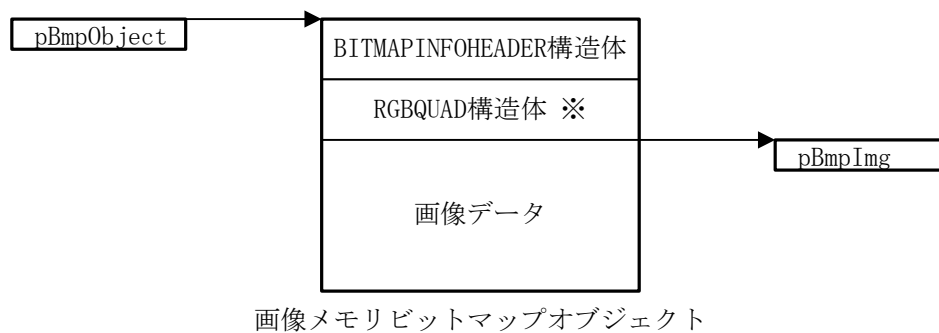
void *pBmpImg != NULL ビットマップ画像データの先頭アドレス
 == NULL エラー

□ エラーコード

なし

□ 機能

CreateBMPImgObjectで生成した画像メモリビットマップオブジェクトからビットマップ画像データの先頭アドレスを取得します。



※ RGBQUAD構造体は、BW_BITMAP設定時のみ必要

第9章 イーサネット通信コマンド

9.1 BSDソケットコマンド

B S Dソケットコマンドを使用する場合は、“bsdsock.h” をインクルードして下さい。

9.1.1 ソケット生成 (socket)

□ C言語API

```
int ret = socket( int family, int type, int protocol );
```

□ パラメータ

int	family	アドレスファミリ
int	type	サービス型
int	protocol	使用するプロトコル番号

□ リターンパラメータ

int	ret	生成したソケットの識別子 (> 0) または異常終了 (- 1)

□ エラーコード

BSD_EPERM	関数実行不可
BSD_ENFILE	ファイルテーブル不定 (ソケット数に空きがない)
BSD_EPROTONOSUPPORT	プロトコル未サポート (アドレスファミリ、サービスまたはプロトコルが不正)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソケットを生成し、ソケット識別子を返します。

本ソケットA P I関数においてサポートしているプロトコルは、T C PとU D Pのみです。
familyにはAF_INET (2)を指定してください (PF_INET (2)でも可)。typeには、T C PではSOCK_STREAM (1)、U D PではSOCK_DGRAM (2)を指定してください (SOCK_RAW (3)は指定できません)。protocolには、T C Pでは0またはT C Pのプロトコル番号(6)を、U D Pでは0またはU D Pのプロトコル番号(17)を指定してください。

B S Dソケットコマンドを使用する場合は、“VPXSOCK.H”をインクルードして下さい。

空きのソケットがなく、ソケットが生成できない場合は、大域変数errnoにENFILEを設定し-1 (エラー)を返します。

family、type、protocolが表にない組み合わせの場合は、大域変数errnoにEPROTONOSUPPORTを設定し-1 (エラー)を返します。

使用するプロトコルごとのfamily、type、protocolの設定を下の表に示します。

使用するプロトコル	Family	type	protocol
TCPのプロトコル	AF_INET (2) または PF_INET (2)	SOCK_STREAM (1)	0
			TCPのプロトコル番号 (6)
UDPのプロトコル		SOCK_DGRAM (2)	0
			UDPのプロトコル番号 (17)

9.1.2 通信アドレス情報の指定 (bind)

□ C言語API

```
int ret = bind( int sockfd, struct sockaddr *p_localaddr, socklen_t addrlen );
```

□ パラメータ

int	sockfd	ソケット識別子
struct sockaddr	*p_localaddr	自分の通信アドレス情報を格納した領域のアドレス
socklen_t	addrlen	自分の通信アドレス情報を格納した領域の長さ

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EBADF	識別子無効 (sockfd < 0)
BSD_ENOMEM	メモリ不足 (UDPソケットを生成するための受信バッファ資源が足りない)
BSD_EFAULT	ポインタアドレス不正 (p_localaddr が 4 の倍数以外)
BSD_EINVAL	引数不正 ①指定したソケットはすでに自通信アドレス情報が決定している ② addrlen が構造体 struct sockaddr_in 型の長さと異なる 指定したソケットに対し shutdown() が実行されている
BSD_EPIPE	識別子ソケット未指定 (sockfd と同じ識別子のソケットが存在しない)
BSD_ENOTSOCK	アドレスファミリが不正 (アドレスファミリが AF_INT(2) 以外)
BSD_EAFNOSUPPORT	通信アドレス情報使用中 ①指定した通信アドレス情報において、自動割り当て以外の指定をしたIP アドレスまたはポート番号が、他のソケットにおいて使用中 ②指定した通信アドレス情報において、自動割り当て指定をしたIPアドレ スまたはポート番号に空きが無い
BSD_EADDRINUSE	IPアドレス利用不可
BSD_EADDRNOTAVAIL	プロセス超過 (TCP/IPマネージャのUDP通信端点が足りない)
BSD_EPROCLIM	重複操作 (指定したソケットを他のBSD 関数が使用中)
BSD_EUSERS	

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソケットに対し自分の通信アドレス情報を割り当てます。
`socket()` を用いて作成された直後のソケットには、通信アドレス情報が割り当てられていません。
`bind()` は `sockfd` が指すソケットに、`p_localaddr` で示す領域の通信アドレス情報を割り当てます。
 通信アドレス情報は `struct sockaddr_in` 型です。この領域のアドレスを関数に渡す際には、
`struct sockaddr` へのポインタ型にキャストしてください。

指定した通信アドレス情報のうち、IP アドレスまたはポート番号が他のソケットにおいて使用中の場合は、大域変数 `errno` に `EADDRINUSE`*2 を設定し-1 (エラー) を返します。

`sockfd` が指定したソケットにすでに通信アドレス情報が割り当てられている場合は、通信アドレス情報を割り当てることはできないため、大域変数 `errno` に `EINVAL` を設定し-1 (エラー) を返します。

`addrlen` が構造体 `struct sockaddr_in` 型の長さとは異なる場合は、大域変数 `errno` に `EINVAL` を設定し-1 (エラー) を返します。

`sin_port` に 0 が指定された場合には登録されているポート番号の範囲から未使用のポートを検索して割り当てます。未使用のポートが無い場合は、大域変数 `errno` に `EADDRINUSE` を設定し-1 (エラー) を返します。

`sin_addr` で指定した IP アドレスが登録されていない場合は、大域変数 `errno` に `EADDRNOTAVAIL` を設定し-1 (エラー) を返します。

`sin_addr` に 0 が指定された場合には登録されている IP アドレスから未使用の IP アドレスを検索して割り当てます。未使用の IP アドレスが無い場合は、大域変数 `errno` に `EADDRINUSE` を設定し-1 (エラー) を返します。

TCP/IP マネージャで確保したUDP 通信端点に空きがなく処理ができない場合は、大域変数 `errno` に `EPROCLIM` を設定し-1 (エラー) を返します。

UDP ソケットが送受信に使用するバッファが確保できない場合は、大域変数 `errno` に `ENOMEM` を設定し-1 (エラー) を返します。

*2: ソケットオプション `SO_REUSEADDR` が OFF の時のものです。
 詳細は、`getsockopt()` の解説を参照してください。

◆ パケットの構造 ◆

< IP アドレスの構造体 >

```
struct in_addr {
    in_addr_t s_addr;
};
```

< 通信アドレス情報の構造体 >

```
struct sockaddr_in {
    uint8_t      sin_len;           // 構造体の長さ (16 バイト固定)
    sa_family_t  sin_family;       // アドレスファミリー (AF_INET(2))
    in_port_t    sin_port;         // TCP あるいはUDP のポート番号
    struct in_addr sin_addr;       // IP アドレス (ネットワークバイトオーダー)
    char         sin_zero[8];      // 未使用
};
```


9.1.3 受信モード設定 (listen)

□ C言語API

```
int ret = listen( int sockfd, int queuelen );
```

□ パラメータ

int	sockfd	ソケット識別子
int	queuelen	受け入れ可能なコネクション要求の数

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EBADF	識別子無効 (sockfd < 0)
BSD_ENOMEM	メモリ不足 (ソケット用のバッファが不足している)
BSD_EINVAL	引数不正 ① queuelen ≤ 0 ② 指定したソケットを、受動モードへ変更できない
BSD_ENOTSOCK	識別子ソケット未指定 (sockfd と同じ識別子のソケットが存在しない)
BSD_EOPNOTSUPP	未サポートソケット (指定したソケットがTCPのソケットでない)
BSD_EADDRINUSE	通信アドレス情報使用中 ① 登録されている通信アドレス情報のポート番号が、他のソケットにおいて使用中 (自動割り当て指定でない場合) ② 通信アドレス情報に割り当てるポート番号に空きが無い (自動割り当て指定の場合)
BSD_EPROCLIM	プロセス超過 (TCP/IP マネージャのTCP 受付口が足りない)
BSD_EUSERS	重複操作 (指定したソケットを他のBSD 関数が使用中)
BSD_EDEADLK	デッドロック条件 ① BSDで生成した通信端点またはポートを他のプロセスが使用中 ② BSDで生成した通信端点が他のプロセスによって削除された ③ BSDで生成した受付口が他のプロセスによって削除された
BSD_EPROTO	プロトコルエラー (使用するIPアドレスが動作していない)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソケットを受動オープンし、受動モードに設定します。

queuelenには、このソケットが受け付けることができるコネクション要求の最大数を設定します。

queuelenには1以上を指定してください。0以下の値を指定した場合は、大域変数 `errno` に `EINVAL` を設定し-1（エラー）を返します。

対象ソケットが下記に示す(1)と(2)の状態以外の場合は、大域変数 `errno` に `EINVAL` を設定し-1（エラー）を返します。

- (1) `socket()` で生成した直後のソケット
- (2) `bind()` が成功した直後のソケット

`listen()` は、TCPのソケットが対象です。指定されたソケットがTCPでない場合は、大域変数 `errno` に `EOPNOTSUPP` を設定し-1（エラー）を返します。

ソケットに登録されている通信アドレス情報のポート番号が、他のソケットにおいて既に使用中の場合は、大域変数 `errno` に `EADDRINUSE` を設定し-1（エラー）を返します。

通信アドレス情報のポート番号を自動割り当てに指定し、割り当てるポート番号に空きが無い場合は、大域変数 `errno` に `EADDRINUSE` を設定し-1（エラー）を返します。

ソケットが送受信に使用するバッファが確保できない場合は、大域変数 `errno` に `ENOMEM` を設定し-1（エラー）を返します。

TCP/IPマネージャで確保したTCP受付口に空きがなく処理が行えない場合は、大域変数 `errno` に `EPROCLIM` を設定し-1（エラー）を返します。

マルチタスク動作において、指定したソケットを他のプロセスから呼び出されたBSD関数が操作中の場合は、大域変数 `errno` に `EUSERS` を設定し-1（エラー）を返します。

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数 `errno` に `EDEADLK` を設定し-1（エラー）を返します。

自分の通信アドレス情報として設定されているIPアドレスがTCP/IPマネージャ上で動作していない場合は、大域変数 `errno` に `EPROTO` を設定し-1（エラー）を返します。

9.1.4 ソケットに対するコネクションの受入れ (accept)

□ C言語API

```
int ret = accept( int sockfd, struct sockaddr *p_addr, socklen_t *addrlen );
```

□ パラメータ

int	sockfd	ソケット識別子
struct sockaddr	*p_addr	接続相手の通信アドレス情報を格納する領域のアドレス
socklen_t	*addrlen	通信アドレス情報の長さを格納した領域のアドレス

□ リターンパラメータ

int	ret	新しく生成されたソケット識別子 (≥ 0) または異常終了 (-1)
struct sockaddr_in	*p_addr	格納した接続相手の通信アドレス情報
socklen_t	*addrlen	格納した接続相手の通信アドレス情報の長さ

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EINTR	割り込みエラー (待ち状態が強制的に解除された)
BSD_EBADF	識別子無効 (sockfd < 0)
BSD_EFAULT	ポインタアドレス不正 (p_addr もしくは addrlen の示す値が4の倍数以外)
BSD_EINVAL	引数不正
	① 指定したソケットが受動モードでない
	② addrlen が構造体 struct sockaddr_in 型の長さと異なる)
BSD_ENFILE	ファイルテーブル不足 (ソケット数に空きがない)
BSD_EWOULDBLOCK	ノンブロッキングリターン (ソケットがノンブロッキングモードでかつキュー上に待ち状態のコネクションがない)
BSD_ENOTSOCK	識別子ソケット未指定 (sockfd と同じ識別子のソケットが存在しない)
BSD_EOPNOTSUPP	未サポートソケット (指定したソケットがTCPのソケットでない)
BSD_EPROCLIM	プロセス超過 (TCP/IPマネージャのTCP通信端点が足りない)
BSD_EUSERS	重複操作 (指定したソケットを他のBSD 関数が使用中)
BSD_EDEADLK	デッドロック条件
	① BSDで生成した通信端点またはポートを他のプロセスが使用中
	② BSDで生成した通信端点が他のプロセスによって削除された
	③ BSDで生成した受付口が他のプロセスによって削除された

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

確立済みコネクションから接続済みのソケットを生成します。

確立済みコネクションから接続済みのソケットを生成してデータの送受信を可能にした後、`p_addr`の指し示す領域に接続相手の通信アドレス情報を返します。

`p_addr` が示す領域は`struct sockaddr_in`型です。関数に渡す際には`struct sockaddr`型のポインタにキャストする必要があります。

`accept()`はコネクション待ちキューにある確立済みコネクションを引き出し、確立済みコネクションから接続済みの能動モードのソケットを作成して、そのソケットに新しいソケット識別子を割り当てリターン値として返します。新しく生成された接続済みソケットは、`sockfd`の示すソケットとは別のソケットです。

キュー上に待ち状態のコネクションがない場合、コネクションができるまで`accept()`からリターンしません（ソケットのノンブロッキングモード^{*1}がOFFの場合）。ソケットのノンブロッキングモード^{*1}がONの場合は、大域変数 `errno` に `EWOULDBLOCK` を設定しリターン値に`-1`を返します。

`accept()`により新しく生成された接続済みのソケットは、能動モードであり、データの送受信のために使用されます。

`sockfd` が示すソケットは、`accept()`からリターンしたあとも受動モードのまま存在するため、再び`accept()`を実行することが可能です。

`accept()`は、TCPのソケットが対象です。指定されたソケットがTCPでない場合は、大域変数 `errno` に `EOPNOTSUPP` を設定し`-1`（エラー）を返します。

`sockfd`の示すソケットは、受動モードかつ`shutdown()`が実行されていない状態である必要があります。

この条件を満たしていないソケットが指定された場合は、大域変数 `errno` に `EINVAL` を設定し`-1`（エラー）を返します。

`addrlen` が構造体`struct sockaddr_in`型の長さと異なる場合は、大域変数 `errno` に `EINVAL` を設定し`-1`（エラー）を返します。

ソケットに空きが無く、新しいソケットが生成できない場合は、大域変数 `errno` に `ENFILE` を設定し`-1`（エラー）を返します。

他のプロセスによって待ち状態が強制的に解除された場合は、大域変数 `errno` に `EINTR` を設定し`-1`（エラー）を返します。

マルチタスク動作において、指定したソケットを他のプロセスから呼び出されたBSD関数が操作中の場合は、大域変数 `errno` に `EUSERS` を設定し`-1`（エラー）を返します。

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数 `errno` に `EDEADLK` を設定し`-1`（エラー）を返します。

^{*1} : ノンブロッキングモードとそのON/OFFの切り替えについては `set_blocking_socket()`を参照してください。

◆ パケットの構造体 ◆

< IP アドレスの構造体 >

```
struct in_addr {
    in_addr_t s_addr;
};
```

< 通信アドレス情報の構造体 >

```
struct sockaddr_in {
    uint8_t      sin_len;           // 構造体の長さ (16バイト固定)
    sa_family_t  sin_family;       // アドレスファミリー (AF_INET(2))
    in_port_t    sin_port;         // TCPあるいはUDPのポート番号
    struct in_addr sin_addr;       // IPアドレス (ネットワークバイトオーダー)
    char         sin_zero[8];      // 未使用
};
```

9.1.5 ソケットの接続の開始 (connect)

□ C言語API

```
int ret = connect( int sockfd, struct sockaddr *p_addr, socklen_t addrlen );
```

□ パラメータ

int	sockfd	ソケット識別子
struct sockaddr	*p_addr	相手通信アドレス情報を格納する領域のアドレス
socklen_t	addrlen	相手通信アドレス情報の長さを格納した領域のアドレス

□ リターンパラメータ

int	ret	新しく生成されたソケット識別子 (≥ 0) または異常終了 (-1)

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EINTR	割り込みエラー (待ち状態が強制的に解除された)
BSD_EBADF	識別子無効 (sockfd < 0)
BSD_ENOMEM	メモリ不足 (ソケット用のバッファが不足している)
BSD_EFAULT	ポインタアドレス不正 (p_addr が 0 または 4 の倍数以外)
BSD_EINVAL	引数不正
	① 指定したソケットが受動モード
	② 指定したソケットが接続失敗状態
	③ addrlen が構造体 struct sockaddr_in 型の長さと異なる
BSD_EPIPE	指定したソケットに対し shutdown() が実行されている
BSD_EINPROGRESS	処理開始 (ノンブロッキング設定で、かつ接続がすぐに完了しない)
BSD_ENOTSOCK	識別子ソケット未指定 (sockfd と同じ識別子のソケットが存在しない)
BSD_EAFNOSUPPORT	アドレスファミリ未サポート
	① TCP において sin_family $\neq$ AF_INET (2)
	② UDP において sin_family $\neq$ AF_INET (2) かつ sin_family $\neq$ AF_UNSPEC (0)
BSD_EADDRINUSE	通信アドレス情報割り当て不可 (自分のポート番号に空きがない)
BSD_EADDRNOTAVAIL	通信アドレス情報利用不可 (相手の IP アドレスまたはポート番号が 0)
BSD_EISCONN	接続済みソケット指定
BSD_ETIMEDOUT	タイムアウトエラー (プロトコルがコネクション確立前に 75 秒経過)
BSD_ECONNREFUSED	接続失敗 (コネクションが相手によって拒絶された)
BSD_EPROCLIM	プロセス超過 (TCP/IP マネージャの UDP 通信端点が足りない)
BSD_EUSERS	重複操作 (指定したソケットを他の BSD 関数が使用中)
BSD_EDEADLK	デッドロック条件
	① BSD で生成した通信端点またはポートを他のプロセスが使用中
	② BSD で生成した通信端点 that 他 のプロセスによって削除された
BSD_EPROTO	プロトコルエラー (使用する IP アドレスが動作していない)

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

通信相手とのコネクションを確立します。
p_addr が示す領域はstruct sockaddr_in型です。関数に渡す際にはstruct sockaddr型のポインタにキャストする必要があります。

ソケットが送受信に使用するバッファが確保できない場合は、大域変数 errno に ENOMEM を設定し-1（エラー）を返します。

マルチタスク動作において、指定したソケットを他のプロセスから呼び出されたBSD関数が操作中の場合は、大域変数 errno に EUSERS を設定し-1（エラー）を返します。

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数 errno に EDEADLK を設定し-1（エラー）を返します。

自分の通信アドレス情報として設定されているIPアドレスがTCP/IPマネージャ上で動作していない場合は、大域変数 errno に EPROTO を設定し-1（エラー）を返します。

次にプロトコルごとの説明を行います。

(1) TCPのソケット

TCPのソケットの場合には、connect()は3ウェイハンドシェイクを使ってp_addrの示す相手に対しコネクションを確立します。コネクションの確立に成功すると、0（正常終了）を返します。
コネクションの確立に失敗した場合、そのソケットは使用不可になり、同じソケットに対し再度connect()が呼び出された場合は必ず異常終了となります。

使用可能なIPアドレスやポート番号に空きがない場合は、大域変数errnoにEADDRINUSEを設定し-1（エラー）を返します。
コネクションが確立しないで一定時間（75秒）経過した場合は、大域変数errnoにETIMEDOUTを設定し-1（エラー）を返します。
コネクションが相手によって拒絶された場合は、大域変数errnoにECONNREFUSEDを設定し-1（エラー）を返します。

ノンブロッキングモードがOFFの場合、コネクションが確立するかエラーが発生するまでは接続中（TCP/IPマネージャによる接続を待つ）の状態となりリターンしません。
他のプロセスによって待ち状態が強制的に解除された場合は、大域変数errnoにEINTRを設定し-1（エラー）を返します。

ノンブロッキングモードがONの場合は、大域変数errnoにEINPROGRESSを設定し-1（エラー）を返しますが異常ではありません。対象となっていたソケットはコネクションを開始し、接続中の状態になります。
コネクションが終了する前に再度connect()を実行した場合は、大域変数errnoにEINVALを設定し-1（エラー）を返します。
コネクションが正常に終了した後に再度connect()を実行した場合は、大域変数errnoにEISCONNを設定し-1（エラー）を返します。

(2) UDPのソケット

UDPのソケットの場合には相手側の通信アドレス情報を記録し、すぐにリターンします。これにより、UDPソケットを相手設定済みの状態にします。

UDPのソケットの場合、同じソケットに対し何度もconnect()が呼び出せます。このとき、別の相手側通信アドレスを設定すると、登録してある相手側通信アドレスが変更されます。
相手側通信アドレスのp_addrにNULLを、addrlenに0を設定しconnect()を呼び出した場合は、大域変数 errno に EADDRNOTAVAIL を設定し-1（エラー）を返しますが異常ではありません。この場合、ソケットは相手未設定の状態になります。

送信側および受信側が共にshutdown()によって閉じられている場合は、大域変数 errno に EPIPE を設定し-1（エラー）を返します。

TCP/IPマネージャで確保したUDPの通信端点に空きがなく処理が行えない場合は、大域変数 errno に EPROCLIM を設定し-1（エラー）を返します。

◆ パケットの構造体 ◆

＜ IP アドレスの構造体 ＞

```
struct in_addr {  
    in_addr_t s_addr;  
};
```

＜ 通信アドレス情報の構造体 ＞

```
struct sockaddr_in {  
    uint8_t      sin_len;      // 構造体の長さ (16バイト固定)  
    sa_family_t  sin_family;   // アドレスファミリ  
    in_port_t    sin_port;     // TCPあるいはUDPのポート番号  
    struct in_addr sin_addr;    // IPアドレス (ネットワークバイトオーダー)  
    char         sin_zero[8];  // 未使用  
};
```

9.1.6 相手側通信アドレス情報取得 (getpeername)

□ C言語API

```
int ret = getpeername( int sockfd, struct sockaddr *p_remaddr, socklen_t *addrlen );
```

□ パラメータ

int	sockfd	ソケット識別子
struct sockaddr	*p_remaddr	相手通信アドレス情報を格納する領域
socklen_t	*addrlen	相手通信アドレス情報を格納する領域の長さ

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (- 1)
struct sockaddr_in	*p_remaddr	相手側の通信アドレス情報
socklen_t	*addrlen	相手側の通信アドレス情報の長さ

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EBADF	識別子無効 (sockfd < 0)
BSD_EFAULT	ポインタアドレス不正 (p_remaddr, addrlen が 0 または 4 の倍数以外)
BSD_EINVAL	引数不正 (*addrlen が構造体 struct sockaddr_in 型の長さより小さい)
BSD_ENOTSOCK	識別子ソケット未指定 (sockfd と同じ識別子のソケットが存在しない)
BSD_ENOTCONN	ソケット未接続
	① TCPにおいて通信相手と接続されていない
	② UDPにおいて通信相手が設定されていない

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

接続されているソケットの相手側通信アドレス情報を取得します。

p_remaddr が示す領域は struct sockaddr_in 型です。関数に渡す際には struct sockaddr 型のポインタにキャストする必要があります。対象ソケットが接続されておらず、相手がいない場合は、大域変数 errno には ENOTCONN を設定し - 1 (エラー) を返します。

パケットの構造は、connect () を参照して下さい。

9.1.7 自通信アドレス情報取得 (getsockname)

□ C言語API

```
int ret = getsockname( int sockfd, struct sockaddr *p_addr, socklen_t *addrlen );
```

□ パラメータ

int	sockfd	ソケット識別子
struct sockaddr	*p_addr	自通信アドレス情報を格納する領域
socklen_t	*addrlen	自通信アドレス情報の長さを格納する領域の先頭アドレス

□ リターンパラメータ

int	ret	正常終了 (0) または異常終了 (-1)
struct sockaddr_in	*p_addr	指定ソケットの通信アドレス情報
socklen_t	*addrlen	指定ソケットの通信アドレス情報の長さ

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EBADF	識別子無効 (sockfd < 0)
BSD_EFAULT	ポインタアドレス不正 (p_addr または addrlen が 4 の倍数以外)
BSD_EINVAL	引数不正 (*addrlen が構造体 struct sockaddr_in 型の長さより小さい)
BSD_ENOTSOCK	識別子ソケット未指定 (sockfd と同じ識別子のソケットが存在しない)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソケット識別子が示すソケットの通信アドレス情報を取得します。

p_addr が示す領域は struct sockaddr_in 型です。関数に渡す際には struct sockaddr 型のポインタにキャストする必要があります。指定のソケットに通信アドレス情報が設定されていない場合でも、正常終了になります。ただし、p_addr に設定するパラメータの値は保証しません。

パケットの構造は、connect() を参照して下さい。

9.1.8 ソケットからの受信データ取得 (recv)

□ C言語API

```
ssize_t ret = recv( int sockfd, void *buffer, size_t length, int flags );
```

□ パラメータ

int	sockfd	ソケット識別子
void	*buffer	受信データを格納する領域の先頭アドレス
size_t	length	受信データを格納する領域の長さ
int	flags	受信動作を指定するフラグ

□ リターンパラメータ

int	ret	受信データの長さ (≥ 0) または異常終了 (− 1)
void	*buffer	ソケットからの受信データ

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EINTR	割り込みエラー (待ち状態が強制的に解除された)
BSD_EBADF	識別子無効 (sockfd < 0)
BSD_EINVAL	引数不正
	① 指定したソケットが受動モード
	② length = 0 または length ≥ 0x80000000
	③ flags が不正
BSD_EPIPE	指定したソケットに対し shutdown() が実行されている
BSD_EWOULDBLOCK	ノンブロッキングリターン (ソケットがノンブロッキングモードでかつ受信済みのデータがない)
BSD_ENOTSOCK	識別子ソケット未指定 (sockfd と同じ識別子のソケットが存在しない)
BSD_ECONNRESET	コネクションリセット (コネクションがリセットされた)
BSD_ENOTCONN	未接続エラー (sockfdの示すソケットが未接続)
BSD_EUSERS	重複操作 (指定したソケットを他のBSD 関数が使用中)
BSD_EDEADLK	デッドロック条件
	① BSDで生成した通信端点またはポートを他のプロセスが使用中
	② BSDで生成した通信端点が他のプロセスによって削除された

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソケットからデータを受信します。
 ソケット内に読み取られていない受信データがありその長さがlengthより小さい場合は、全てのデータを取り出してその取り出したデータの長さを返します。lengthより大きい場合は、length分のデータを取り出してその長さを返します。
 ノンブロッキングモードがOFFの場合は、読み取られていない受信データが無いときは、データを受信するかエラーが発生するまでリターンしません。
 ノンブロッキングモードがONで受信データがない場合は、大域変数errnoにEWOULDBLOCKを設定し-1 (エラー)を返します。この場合は異常ではありません。

指定したソケットがTCPで接続されていない場合は、大域変数errnoにENOTCONNを設定し-1 (エラー)を返します。

指定したソケットの受信側が、shutdown()によって閉じられている場合は、大域変数errnoにEPIPEを設定し-1 (エラー)を返します。

指定したソケットがTCPで既にFINを受信しており、かつ受信バッファが空の場合は、正常終了として受信データ長0を返します (FINを受信しても受信データがあるうちは正常に受信できます)。

指定したソケットがTCPで既にコネクションがリセットされている場合は、大域変数errnoにECONNRESETを設定し-1 (エラー)を返します。

TCPソケットの受信待ち状態でFINを受信し、かつ受信データが無い場合は、正常終了として受信データ長0を返します。

TCPソケットの受信待ち状態でコネクションがリセットされた(RST受信またはRST送信)場合は、大域変数errnoにECONNRESETを設定し-1 (エラー)を返します。

他のプロセスによって待ち状態が強制的に解除された場合は、大域変数errnoにEINTRを設定し-1 (エラー)を返します。

マルチタスク動作において、指定したソケットを他のプロセスから呼び出されたBSD関数が操作中の場合は、大域変数errnoにEUSERSを設定し-1 (エラー)を返します。

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数errnoにEDEADLKを設定し-1 (エラー)を返します。

flagsに設定する値に対応した動作を行うことができます。また、値の論理和を設定することで2つ以上の動作を指定することができます。

flagsに不正な値が設定された場合は、大域変数errnoにEINVALを設定し-1 (エラー)を返します。

recv(),recvfrom()で flags に設定する値

define名	値	内 容	TCP	UDP
MSG_PEEK	0x0002	バッファリングデータを保持したままで読み込み実行	○	○
MSG_DONTWAIT	0x0040	1回の操作毎のノンブロッキング指定	○	○
MSG_WAITALL	0x0100	全データ到着待ち	○	×

注意：MSG_PEEK とMSG_WAITALL は同時に設定できません

MSG_PEEK : 受信バッファの状態とデータを変化させずに、受信データの内容を取り出します。

MSG_PEEKが設定されていない場合は、受信したデータ分、受信バッファの内容が変化します。
MSG_PEEKが設定されている場合は、受信バッファの状態とデータを変化させません。
(再度、受信を実行したとき、同じデータを読み込みます)
T C Pの場合は、再度受信を実行するとデータの長さが増える場合があります。
U D Pの場合は、再度受信を実行してもまったく同じデータを取得します。

MSG_DONTWAIT : ノンブロッキングモードON として動作します。

MSG_DONTWAITが設定されていない場合は、ソケット自身のノンブロッキングモードに従って動作します。
MSG_DONTWAITが設定されている場合は、関数内の処理はノンブロッキングモードONとして動作します。

MSG_WAITALL : 要求された長さ分のデータを受信します。

MSG_WAITALLが設定されていない場合は、受信データがあればその長さがlengthより小さい場合でもデータを取り出してリターンします。
MSG_WAITALLが設定されている場合は、要求された長さ分のデータを受信するまでリターンしません。ただし、次の場合は要求より少ないデータでリターンすることがあります。
(1) コネクションが切断された場合
(2) ソケットにエラーが発生した場合

9.1.9 受信データと送信者アドレス取得 (recvfrom)

□ C言語API

```
ssize_t ret = recvfrom( int sockfd, void *buffer, size_t length, int flags,
                        struct sockaddr *p_from, socklen_t *fromlen );
```

□ パラメータ

int	sockfd	ソケット識別子
void	*buffer	受信データを格納した領域の先頭アドレス
size_t	length	受信データを格納した領域の長さ
int	flags	各種の特殊な受信動作を指定するフラグ
struct sockaddr	*p_from	送信者のアドレス情報を格納する領域
socklen_t	*fromlen	送信者のアドレス情報を格納する領域の長さを格納した領域のアドレス

□ リターンパラメータ

int	ret	buffer に設定した受信データの長さ(≥ 0)または異常終了(-1)
void	*buffer	ソケットからの受信データ
struct sockaddr	*p_from	送信者のアドレス情報
socklen_t	*fromlen	送信者のアドレス情報の長さ

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EINTR	割り込みエラー (待ち状態が強制的に解除された)
BSD_EBADF	識別子無効 (sockfd < 0)
BSD_EFAULT	ポインタアドレス不正 (p_from または fromlen が 4 の倍数以外)
BSD_EINVAL	引数不正
	① 指定したソケットが受動モード
	② fromlen が構造体 struct sockaddr_in 型の長さと異なる
	③ length = 0 または length ≥ 0x80000000
	④ flags が不正
BSD_EPIPE	指定したソケットに対し shutdown() が実行されている
BSD_EWOULDBLOCK	ノンブロッキングリターン (ソケットがノンブロッキングモードでかつ受信済みのデータがない)
BSD_ENOTSOCK	識別子ソケット未指定 (sockfd と同じ識別子のソケットが存在しない)
BSD_ECONNRESET	コネクションリセット (コネクションがリセットされた)
BSD_ENOTCONN	未接続エラー (sockfd の示すソケットが未接続)
BSD_EUSERS	重複操作 (指定したソケットを他の BSD 関数が使用中)
BSD_EDEADLK	デッドロック条件
	① BSD で生成した通信端点またはポートを他のプロセスが使用中
	② BSD で生成した通信端点が他のプロセスによって削除された

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

受信したデータと、その送信者の通信アドレス情報を取得します。
 recvfrom()は相手未設定のUDPソケットを対象としていますが、相手設定済みのUDPソケットやTCPソケットの場合にも使用できます。

p_fromが示す領域はstruct sockaddr_in型です。関数に渡す際にはstruct sockaddr型のポインタにキャストする必要があります。p_fromの示す領域には送信者の通信アドレス情報を、fromlenの示す領域には送信者の通信アドレス情報の長さを返します。p_fromにNULLを指定するとrecv()と同じ動作を行います。その場合は、*p_from および*fromlenの領域には設定を行いません。

p_fromがNULL以外でfromlenが構造体struct sockaddr_in型の長さと異なる場合は、大域変数errnoにEINVALを設定し-1（エラー）を返します。

ソケット内に読み取られていない受信データがありその長さがlengthより小さい場合は、全てのデータを取り出してその取り出したデータの長さを返します。lengthより大きい場合は、length分のデータを取り出してその長さを返します。

ノンブロッキングモードがOFFの場合は、読み取られていない受信データが無いときは、データを受信するかエラーが発生するまでリターンしません。

ノンブロッキングモードがONで受信データがない場合は、大域変数errnoにEWOULDBLOCKを設定し-1（エラー）を返します。この場合は異常ではありません。

指定したソケットがTCPで接続されていない場合は、大域変数errnoにENOTCONNを設定し-1（エラー）を返します。

指定したソケットの受信側が、shutdown()によって閉じられている場合は、大域変数errnoにEPIPEを設定し-1（エラー）を返します。

指定したソケットがTCPで既にFINを受信しており、かつ受信バッファが空の場合は、正常終了として受信データ長0を返します（FINを受信しても受信データがあるうちは正常に受信できます）。

指定したソケットがTCPで既にコネクションがリセットされている場合は、大域変数errnoにECONNRESETを設定し-1（エラー）を返します。

TCPソケットの受信待ち状態でFINを受信し、かつ受信データが無い場合は、正常終了として受信データ長0を返します。

TCPソケットの受信待ち状態でコネクションがリセットされた(RST受信またはRST送信)場合は、大域変数errnoにECONNRESETを設定し-1（エラー）を返します。

他のプロセスによって待ち状態が強制的に解除された場合は、大域変数errnoにEINTRを設定し-1（エラー）を返します。

マルチタスク動作において、指定したソケットを他のプロセスから呼び出されたBSD関数が操作中の場合は、大域変数errnoにEUSERSを設定し-1（エラー）を返します。

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数errnoにEDEADLKを設定し-1（エラー）を返します。

flagsに設定する値に対応した動作を行うことができます。詳細については、recv()を参照してください。

パケットの構造は、connect()を参照して下さい。

9.1.10 ソケットへの送信データ設定 (send)

□ C言語API

```
ssize_t ret = send( int sockfd, void *buffer, size_t len, int flags );
```

□ パラメータ

int	sockfd	ソケット識別子
void	*buffer	送信データを格納した領域の先頭アドレス
size_t	len	送信データの長さ
int	flags	各種の特殊な送信動作を指定するフラグ

□ リターンパラメータ

int	ret	送信用ウインドウバッファに書き込んだ送信データの長さ (≥ 0) または 異常終了 (-1)

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EINTR	割り込みエラー (待ち状態が強制的に解除された)
BSD_EBADF	識別子無効 (sockfd < 0)
BSD_ENOMEM	メモリ不足 (UDPソケットを生成するための受信バッファ資源が足りない)
BSD_EACCES	許可が無い (SO_BROADCASTがOFFで相手アドレスにブロードキャストを指定)
BSD_EINVAL	引数不正
	① 指定したソケットが受動モード
	② len = 0 または len $\geq 0x80000000$
	③ flags が不正
BSD_EPIPE	指定したソケットに対しshutdown() が実行されている
BSD_EWOULDBLOCK	ノンブロッキングリターン (ソケットがノンブロッキングモードでかつ受信済みのデータがない)
BSD_ENOTSOCK	識別子ソケット未指定 (sockfd と同じ識別子のソケットが存在しない)
BSD_EDESTADDRREQ	相手側通信アドレス情報未指定 (sockfdの示すソケットが相手未設定のUDPソケット)
BSD_EADDRINUSE	通信アドレス情報使用中 (通信アドレス情報のポート番号が他のソケットにおいて使用されている)
BSD_ECONNRESET	コネクションリセット (コネクションがリセットされた)
BSD_ENOBUFS	利用可能エリア不足 (UDPの送信バッファが確保できない)
BSD_ENOTCONN	未接続エラー (sockfdの示すソケットが未接続)
BSD_EPROCLIM	プロセス超過 (TCP/IPマネージャのUDP通信端点が足りない)
BSD_EUSERS	重複操作 (指定したソケットを他のBSD 関数が使用中)
BSD_EDEADLK	デッドロック条件
	① BSDで生成した通信端点またはポートを他のプロセスが使用中
	② BSDで生成した通信端点が他のプロセスによって削除された

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソケットを用いてデータを送信します。

T C PでノンブロッキングモードがO F Fの場合は、送信用ウインドウバッファに対しlenの長さ分のデータを書き込み、書き込んだ長さを返します。lenの長さ分のデータ全てを書き込む領域がない場合は、lenの長さ分のデータ全てを書き込むかエラーが発生するまで内部で待ち状態となり、リターンしません。

T C PでノンブロッキングモードがO Nの場合は、送信用ウインドウバッファに1バイトでも書き込んだ場合は、正常終了として書きこんだバイト数を返します。送信用ウインドウバッファに空きがなく、1バイトも書き込めなかった場合は、大域変数errnoにEWOULDBLOCKを設定し-1（エラー）を返します。

送信用ウインドウバッファに書き込まれたデータはT C P/I Pマネージャによって送信されます。

sockfdに相手未設定のU D Pソケットを指定した場合は、大域変数errnoにEDESTADDRREQを設定し-1（エラー）を返します。

U D Pでは送信用バッファにlenの長さ分のデータを書き込み、書き込んだ長さを返します。また、送信データが1回で送信可能なデータサイズより大きくて全てを書き込めない場合は、書き込める分のデータを書き込み、書き込んだ長さを返します(注1)。いずれの場合も書き込んだ分のデータは送信が行います。

U D Pのソケットの場合、ノンブロッキングモードO F Fでもブロックすることはありません。

U D PでノンブロッキングモードがO F Fのときに送信用バッファに空きがなく、送信データを書き込めなかった場合は、大域変数errnoにENOBUFSを設定し-1（エラー）を返します。

U D PでノンブロッキングモードがO Nのときに送信用バッファに空きがなく、送信データを書き込めなかった場合は、大域変数errnoにEWOULDBLOCKを設定し-1（エラー）を返します。

指定したソケットがT C Pで接続されていない場合は、大域変数errnoにENOTCONNを設定し-1（エラー）を返します。

指定したソケットの送信側が、shutdown()によって閉じられている場合は、大域変数errnoにEPIPEを設定し-1（エラー）を返します。

指定したソケットがT C Pで既にコネクションがリセットされている場合は、大域変数errnoにECONNRESETを設定し-1（エラー）を返します。

T C Pソケットの送信待ち状態でコネクションがリセットされた(R S T受信またはR S T送信)場合は、大域変数errnoにECONNRESETを設定し-1（エラー）を返します。

他のプロセスによって待ち状態が強制的に解除された場合は、大域変数errnoにEINTRを設定し-1（エラー）を返します。

マルチタスク動作において、指定したソケットを他のプロセスから呼び出されたB S D関数が操作中の場合は、大域変数errnoにEUSERSを設定し-1（エラー）を返します。

本B S DソケットA P IはT C P/I PマネージャのITRON TCP/IP API で動作しています。B S DソケットA P IがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、B S DソケットA P Iの正常な動作が保証できない状態に陥った場合は、大域変数errnoにEDEADLKを設定し-1（エラー）を返します。

flagsに設定する値に対応した動作を行うことができます。ただし、本B S DソケットA P IではMSG_DONTWAITのみをサポートしています。

flagsに不正な値が設定された場合は、大域変数errnoにEINVALを設定し-1（エラー）を返します。

注1 : EMSGSIZE(40)のエラーは発生しません。また、要求したブロックの全データをIPで分割して送信することはありません。

send(), sendto()でflags に設定する値

define名	値	内 容	TCP	UDP
MSG_DONTWAIT	0 x 0040	1回操作毎のノンブロッキング指定	○	○

MSG_DONTWAIT : ノンブロッキングモードO Nとして動作します。

MSG_DONTWAITが設定されていない場合は、ソケット自身のノンブロッキングモードに従って動作します。

MSG_DONTWAITが設定されている場合は、関数内の処理はノンブロッキングモードO Nとして動作します。

9.1.11 送信データと送信先アドレス設定 (sendto)

□ C言語API

```
ssize_t ret = sendto( int sockfd, void *p_msg, size_t msglen, int flags,
                     struct sockaddr *p_to, int tolen );
```

□ パラメータ

int	sockfd	ソケット識別子
void	*p_msg	送信データを格納した領域のアドレス
size_t	msglen	送信データの長さ
int	flags	各種の特殊な送信動作を指定するフラグ
struct sockaddr	*p_to	送信先の通信アドレス情報を格納した領域の先頭アドレス
int	tolen	送信先の通信アドレス情報を格納した領域の長さ

□ リターンパラメータ

int	ret	送信用ウインドウバッファに書き込んだ送信データの長さ (≥ 0) または 異常終了 (- 1)

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EINTR	割り込みエラー (待ち状態が強制的に解除された)
BSD_EBADF	識別子無効 (sockfd < 0)
BSD_ENOMEM	メモリ不足 (UDPソケットを生成するための受信バッファ資源が足りない)
BSD_EACCES	許可が無い (SO_BROADCASTがOFFで相手アドレスにブロードキャストを指定)
BSD_EINVAL	引数不正
	① 指定したソケットが受動モード
	② msglen = 0 または msglen ≥ 0x80000000
	③ flagsが不正
	④ tolenが構造体struct sockaddr_in型の長さと異なる
BSD_EPIPE	指定したソケットに対しshutdown()が実行されている
BSD_EWOULDBLOCK	ノンブロッキングリターン (ソケットがノンブロッキングモードでかつ送信バッファに空きがない)
BSD_ENOTSOCK	識別子ソケット未指定 (sockfd と同じ識別子のソケットが存在しない)
BSD_EDESTADDRREQ	相手側通信アドレス情報未指定 (sockfdの示すソケットが相手未設定のUDPソケット、かつ指定した相手側通信アドレスが不正)
BSD_EADDRINUSE	通信アドレス情報使用中 (通信アドレス情報のポート番号が他のソケットにおいて使用されている)
BSD_ECONNRESET	コネクションリセット (コネクションがリセットされた)
BSD_ENOBUFS	利用可能エリア不足 (UDPの送信バッファが確保できない)
BSD_EISCONN	接続済みソケット指定
BSD_ENOTCONN	未接続エラー (sockfdの示すソケットが未接続)
BSD_EPROCLIM	プロセス超過 (TCP/IPマネージャのUDP通信端点が足りない)
BSD_EUSERS	重複操作 (指定したソケットを他のBSD 関数が使用中)
BSD_EDEADLK	デッドロック条件
	① BSDで生成した通信端点またはポートを他のプロセスが使用中
	② BSDで生成した通信端点が他のプロセスによって削除された

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

指定された通信アドレスに対し送信を行います。

送信先を指定する場合、`p_to`が示す領域は`struct sockaddr_in`型です。関数に渡す際には`struct sockaddr`型のポインタにキャストする必要があります。

送信先を指定しない場合、`p_to`に`NULL`を指定すると、`send()`と同じ動作を行います。

`p_to`が`NULL`以外で`tolen`が構造体`struct sockaddr_in`型の長さとは異なる場合は、大域変数`errno`に`EINVAL`を設定し-1（エラー）を返します。

(1) `sockfd`が相手未設定のUDPソケットを指している場合

送信用バッファに`msglen`の長さ分のデータを書き込み、書き込んだ長さを返します。また、送信データが1回で送信可能なデータサイズより大きくて全てを書き込めない場合は、書き込める分のデータを書き込み、書き込んだ長さを返します(注1)。いずれの場合も書き込んだ分のデータは送信を行います。

ノンブロッキングモードが`O F F`のときに送信用バッファに空きがなく、送信データを書き込めなかった場合は、大域変数`errno`に`ENOBUFS`を設定し-1（エラー）を返します。

ノンブロッキングモードが`O N`のときに送信用バッファに空きがなく、送信データを書き込めなかった場合は、大域変数`errno`に`EWouldBlock`を設定し-1（エラー）を返します。

送信先通信アドレス情報 (`p_to`) を指定しない (`NULL`) 場合は、大域変数`errno`に`EDESTADDRREQ`を設定し-1（エラー）を返します。

(2) `sockfd`が相手設定済みのUDPソケットまたはTCPソケットを指している場合

送信先通信アドレス情報 (`p_to`) を指定しない (`NULL`) 場合は、`send()`と同様の動作を行います。

送信先通信アドレス情報 (`p_to`) を指定した (`NULL`以外) 場合は、大域変数`errno`に`EISCONN`を設定し-1（エラー）を返します。

指定したソケットがTCPで接続されていない場合は、大域変数`errno`に`ENOTCONN`を設定し-1（エラー）を返します。

指定したソケットの送信側が、`shutdown()`によって閉じられている場合は、大域変数`errno`に`EPIPE`を設定し-1（エラー）を返します。

指定したソケットがTCPで既にコネクションがリセットされている場合は、大域変数`errno`に`ECONNRESET`を設定し-1（エラー）を返します。

注1 : `EMSGSIZE(40)`のエラーは発生しません。

また、要求したブロックの全データをIPで分割して送信することはありません。

TCPソケットの送信待ち状態でコネクションがリセットされた(RST受信またはRST送信)場合は、大域変数`errno`に`ECONNRESET`を設定し-1（エラー）を返します。

他のプロセスによって待ち状態が強制的に解除された場合は、大域変数`errno`に`EINTR`を設定し-1（エラー）を返します。

マルチタスク動作において、指定したソケットを他のプロセスから呼び出されたBSD関数が操作中の場合は、大域変数`errno`に`EUSERS`を設定し-1（エラー）を返します。

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数`errno`に`EDEADLK`を設定し-1（エラー）を返します。

`flags`に設定する値に対応した動作を行うことができます。

詳細については、`send()`を参照してください。

9.1.12 ソケットのクローズ(closesocket)

□ C言語API

```
int ret = closesocket( int sockfd );
```

□ パラメータ

```
int          sockfd          ソケット識別子
```

□ リターンパラメータ

```
int          ret          正常終了 ( 0 ) または 異常終了 ( - 1 )
```

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EBADF	識別子無効 (sockfd < 0)
BSD_EWOULDBLOCK	ノンブロッキングリターン (ソケットがノンブロッキングモードでかつ SO_LINGERがONの場合に、リンガー時間が経過した)
BSD_ENOTSOCK	識別子ソケット未指定 (sockfd と同じ識別子のソケットが存在しない)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

通信を終了し、ソケットを閉じます。

送信バッファ内に残っているデータは全て送信し、未読の受信データは破棄します。
 T C Pソケットの場合は、送信バッファ内のデータを全て送信した後にF I Nを送ります。
 送信データ、受信データに対する処理の終了後、ソケットを未生成の状態にします。

SO_LINGERオプションがO F Fの場合は、クローズ処理を開始後直ちに、0 (正常終了) を返します。

SO_LINGERオプションがO Nの場合は、リンガー時間が経過するまでclosesocket() からリターンしません。リンガー時間が経過した場合、ノンブロッキングモードがO F Fの場合は、0 (正常終了) を返しますが、ノンブロッキングモードがO Nの場合は、大域変数errnoにEWOULDBLOCK設定し- 1 (エラー) を返します。

T C PソケットではSO_LINGERオプションがO Nのとき、リンガー時間に0を指定した場合と、リンガー時間が経過した場合には、送信バッファ内に残っているデータを破棄してR S Tを送信します。

9.1.13 コネクション閉鎖 (shutdown)

□ C言語API

```
int ret = shutdown( int sockfd, int direction );
```

□ パラメータ

int	sockfd	ソケット識別子
int	direction	切断方向

□ リターンパラメータ

int	ret	正常終了 (0) または 異常終了 (- 1)

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EBADF	識別子無効 (sockfd < 0)
BSD_EINVAL	引数不正 (direction の値が対象外)
BSD_ENOTSOCK	識別子ソケット未指定 (sockfd と同じ識別子のソケットが存在しない)
BSD_ENOTCONN	ソケット未接続
BSD_EUSERS	重複操作 (指定したソケットを他のBSD関数が使用中)
BSD_EDEADLK	デッドロック条件 (BSDで生成した通信端点が他のプロセスによって削除された)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

コネクションを部分的に閉鎖します。

切断方向ごとの動作は以下になります。接続していないソケットに使用した場合は、大域変数errno に ENOTCONNを設定し-1 (エラー) を返します。

SHUT_RD (0) の場合：

コネクションの読み出し側をクローズし、受信バッファ内のデータやその後到着したデータを破棄します。TCPソケットの場合、到着データに対しては応答 (ACK) を返しますが、データ自体は破棄します。

SHUT_WR (1) の場合：

コネクションの書き込み側をクローズし、それ以後データ送信を行いません。送信バッファに残っているデータがあれば、そのデータは全て送信します。TCPソケットの場合、送信バッファ内データを全て送信した後にFINを送信します。

SHUT_RDWR (2) の場合：

コネクションの読み出し側、書き込み側の双方をクローズし、それ以後、データの送受信を行えなくします。

マルチタスク動作において、指定したソケットを他のプロセスから呼び出されたBSD関数が操作中の場合は、大域変数errnoにEUSERSを設定し-1 (エラー) を返します。

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数errno にEDEADLKを設定し-1 (エラー) を返します。

9.1.14 プロトコルのオプション取得 (getsockopt)

□ C言語API

```
int ret = getsockopt( int sockfd, int level, int opt, void *p_optval, int *optlen );
```

□ パラメータ

int	sockfd	ソケット識別子
int	level	プロトコルレベル
int	opt	オプション
void	*p_optval	オプションを格納する領域の先頭アドレス
int	*optlen	オプションを格納する領域の長さを格納する領域の先頭アドレス

□ リターンパラメータ

int	ret	正常終了 (0) または 異常終了 (-1)
void	p_optval	オプションデータ
int	optlen	オプションデータの長さ

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EBADF	識別子無効 (sockfd < 0)
BSD_EFAULT	ポインタアドレス不正 (p_optval もしくはoptlenが4の倍数以外)
BSD_EINVAL	引数不正 (optlenが短い)
BSD_ENOTSOCK	識別子ソケット未指定 (sockfd と同じ識別子のソケットが存在しない)
BSD_ENOPROTOOPT	プロトコル利用不可 (level、optの組み合わせが不正)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

プロトコルに対するオプションの値を読み出します。
 ソケットから取得したオプションデータをp_optvalの示す領域に返し、オプションデータの長さをoptlenに返します。
 オプションには、特定の機能の有効／無効を設定、取得するオプション（フラグ型オプション）と、特定の値の設定、取得を行うオプション（数値型オプション）の2種類に分けられます。
 フラグ型オプションの場合は、オプションデータが0ならオプションが無効になっていることを示し、0以外ならオプションが有効になっていることを示します。

以下に、本BSDソケットAPIにおいて使用するソケットオプションを示します。
levelとoptの組み合わせが正しくない場合は、大域変数errnoにENOPROTOOPTを設定し-1（エラー）を返します。

level	Opt	使用可能な関数	内 容	オプションの型	データ型
SOL_SOCKET	SO_REUSEADDR (0x0004)	getsockopt setsockopt	自通信アドレスの再利用の許可	フラグ型	int型
	SO_KEEPALIVE (0x0008)	getsockopt setsockopt	コネクションが有効か定期的な検査の実行	フラグ型	int型
	SO_LINGER (0x0080)	getsockopt setsockopt	クローズ時のリンガー動作実行	数値型	linger型
	SO_BROADCAST (0x0020)	getsockopt setsockopt	ブロードキャストメッセージの転送を許可（UDPのみ）	フラグ型	int型
	SO_SNDBUF (0x1001)	getsockopt setsockopt	出力バッファの大きさ	数値型	int型
	SO_RCVBUF (0x1002)	getsockopt setsockopt	入力バッファの大きさ	数値型	int型
	SO_TYPE (0x1008)	getsockopt	ソケットの型	数値型	int型
	SO_ERROR (0x1007)	getsockopt	ソケットの最新のエラー（取得によりクリア）	数値型	int型
IPPROTO_TCP	TCP_KEEPALIVE (0x02)	getsockopt setsockopt	TCPキープアライブプローブ送信までの待ち時間（単位は秒）	数値型	int型
	TCP_NODELAY (0x01)	getsockopt setsockopt	Nagle アルゴリズムの禁止	フラグ型	int型

（1）SO_REUSEADDR（0x0004） 自通信アドレスの再利用の許可

このオプションは、自分の通信アドレス情報重複許可のON/OFFを示します。
このオプションがOFFの場合は、同一のポートの通信アドレス情報は使用できません。bind()により既に存在するポート番号を指定すると大域変数errnoにEADDRINUSEを設定して-1（エラー）を返します。
このオプションがONの場合、自分の通信アドレス情報の重複を可能にします。TCPとUDPでは働きが異なります。

TCPソケットの場合、能動モードのソケットにおいて、ポート番号、IPアドレスともに同一の自通信アドレスを複数使用できるようにします。受動モードのソケットは、ポート番号、IPアドレスともに同一にすることはできません。自通信アドレスがすでに存在する受動モードソケットと同一の能動モードのソケットを生成することはできますが、この能動モードのソケットに対しlisten()をコールすると大域変数errnoにEADDRINUSEを設定して-1（エラー）を返します。また、能動モードのソケットの場合でも自通信アドレスと相手側通信アドレスの両方を同じにすることはできません。connect()により自通信アドレスと相手側通信アドレスが同一のソケットが生成される場合は、大域変数errnoにEINVALを設定して-1（エラー）を返します。

UDPソケットの場合、複数の自分の通信アドレス情報を同一のポートでできるようにします。ただし、通信アドレス情報毎のIPアドレスは異なっている必要があります。bind()によりポート番号、IPアドレスともに同一の自通信アドレスを指定すると大域変数errnoにEADDRINUSEを設定し-1（エラー）を返します。

(2) SO_KEEPALIVE (0x0008) コネクションが有効か定期的な検査の実行

このオプションは、TCPソケットのコネクション定期的に検査する機能のON/OFFを示します。このオプションがONかつソケットが接続されている場合、そのソケットによる送信受信ともに行われない状況が2時間以上続いたなら、そのTCPソケットは自動的に相手に向けてキープアライブプローブを送信します。このキープアライブプローブに対しての相手側の対応によって次の処理を行います。

- ① 相手がACKで応答
正常な状態であるのでアプリケーションに対しなにも通知しません。送信受信ともないまま再び2時間経過したら再びキープアライブプローブを送信します。
- ② 相手からの応答が無い
この場合、75秒経過するごとにキープアライブプローブの再送をおこないます。8回再送しても応答が無い場合、RSTを送信してコネクションを切断します。

(3) SO_LINGER (0x0080) クローズ時のリンガー動作実行

このオプションは、TCPソケットにおけるクローズ処理のリンガー動作の内容を示します。このオプションデータは、構造体linger型（以下に示す）のデータです。この型のデータをp_optvalを通じて参照、設定します。

```
struct linger {
    int    l_onoff;        // 0: リンガーオプションOFF、0以外: リンガーオプションON
    int    l_linger;       // リンガー時間単位は[1/100秒]
};
```

オプションデータの内容によってclosesocket()の動作が異なります。オプションデータ毎のclosesocket()の動作を次に示します。

- ① l_onoffが0の場合
l_lingerの値は無視します。
closesocket()からすぐにリターンしますが、送信バッファ中のデータを送信し終わるのを待ってFINを送ります。
- ② l_onoffが0以外、かつl_lingerが0の場合
closesocket()からすぐにリターンし、直ちにRSTを送信します。未送信のデータは破棄します。l_onoffが0以外、かつl_lingerが0以外の場合closesocket()からすぐにはリターンしません。送信バッファ中のデータを送信し終わるのを待ってFINを送りますが、l_lingerの指定する時間を経過した場合は、RSTを送信して未送信のデータを破棄します。

(4) SO_BROADCAST (0x0020) ブロードキャストメッセージの転送の許可

このオプションは、ブロードキャストメッセージの転送能力の有効/無効を示します。ブロードキャストはUDPでのみサポートします。このオプションがONの時にブロードキャストメッセージの転送が有効となります。

このオプションがOFFの状態ではデータの送信を行う（sendto()等の関数を使用する）時、相手側通信アドレス情報としてブロードキャストアドレスが設定されると、大域変数errnoにEACCESを設定して-1（エラー）を返します。

(5) SO_SNDBUF (0x1001) 出力バッファの大きさ

このオプションは、ソケットの送信バッファサイズを示します。このオプションはint型のデータであり、p_optvalを通じて参照、設定します。設定する値は2048以上である必要があります。UDPのソケットに対してこのオプションを指定することはできません。

(6) SO_RCVBUF (0x1002) 入力バッファの大きさ

このオプションは、ソケットの受信バッファサイズを示します。このオプションはint型のデータで、p_optvalを通じて参照、設定します。設定する値は2048以上である必要があります。UDPソケットの場合、データグラム受信キューの数を示します。

(7) SO_TYPE (0x1008) ソケットの型

このオプションを指定してgetsockopt()をコールすることで、ソケットの型の参照を行います。このオプションはint型のデータで、SOCK_STREAM(1) (TCPの場合)、SOCK_DGRAM(2) (UDPの場合)の値を取ります。このオプションは、setsockopt()では、使用できません。

(8) SO_ERROR (0x1007) ソケットの最近のエラー取得

このオプションを指定してgetsockopt()をコールすることで、ソケット内の変数so_errorの値を取得することができます。このとき、getsockopt()は変数so_errorを0にクリアします。このオプションは、setsockopt()では、使用できません。

(9) IP_TOS (0x02) IPヘッダ内のTOSの値

このオプションは、IPヘッダ内のTOSの値を参照または設定します。このオプションはint型のデータで、p_optvalを通じて参照、設定します。有効なデータの範囲は8ビットで0~255です。下位8ビット以外の設定データは無視します。sockfdにより指定したソケットの送信するIPヘッダのTOSだけが対象となります。

(10) IP_TTL (0x03) IPヘッダ内のTTLの値

このオプションは、IPヘッダ内のTTLの値を参照または設定します。このオプションはint型のデータで、p_optvalを通じて参照、設定します。有効なデータの範囲は8ビットで1~255です。下位8ビット以外の設定データは無視します。sockfdにより指定したソケットの送信するIPヘッダのTTLだけが対象となります。

(11) TCP_KEEPAIVE (0x02) キープアライブプローブ送信までのアイドル時間

このオプションは、TCPキープアライブプローブ送信までのアイドル時間を秒単位で設定します。このオプションはint型のデータで、p_optvalを通じて参照、設定します。また、SO_KEEPAIVEオプションがOFFのときは、このオプションは無効です。時間の単位は秒です。sockfdにより指定したソケットだけアイドル時間が変更されます。

(12) TCP_NODELAY (0x01) Nagleアルゴリズムの禁止

このオプションは、TCP Nagleアルゴリズムの禁止のON/OFFを示します。sockfdにより指定したソケットだけNagle アルゴリズムがON/OFFされます。

9.1.15 プロトコルのオプション変更 (setsockopt)

□ C言語API

```
int ret = setsockopt( int sockfd, int level, int opt, void *p_optval, int optlen );
```

□ パラメータ

int	sockfd	ソケット識別子
int	level	プロトコルレベル
int	opt	オプション
void	*p_optval	オプションデータを格納する領域の先頭アドレス
int	optlen	オプションデータの長さ

□ リターンパラメータ

int	ret	正常終了 (0) または 異常終了 (- 1)

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EBADF	識別子無効 (sockfd < 0)
BSD_EFAULT	ポインタアドレス不正 (p_optval もしくは optlen が 4 の倍数以外)
BSD_EINVAL	引数不正 (optlen が短い)
BSD_ENOTSOCK	識別子ソケット未指定 (sockfd と同じ識別子のソケットが存在しない)
BSD_ENOPROTOOPT	プロトコル利用不可 (level、opt の組み合わせが不正)
BSD_ENOBUFS	利用可能エリア不足 (バッファが確保できない)
BSD_EPROTO	プロトコルエラー (使用中の通信端点に対し実行できない)

※ 詳細は、付録 A 「エラーコード一覧」を参照して下さい。

□ 機能

プロトコルに対するオプションの値を設定します。
 オプションの設定値を p_optval の示す領域から取得しソケットに設定します。
 フラグ型オプションの場合、オプションの設定値が 0 なら、オプションを無効にし、0 以外ならオプションを有効にします。
 ソケットオプションについては、getsockopt() を参照してください。

level と opt の組み合わせが正しくない場合、およびオプションとして SO_TYPE または SO_ERROR が指定された場合は、大域変数 errno に ENOPROTOOPT を設定し -1 (エラー) を返します。

BSD ソケット API が生成した TCP/IP マネージャの通信端点を既に使用中で、指定したオプションの設定ができない状態の場合は、大域変数 errno に EPROTO を設定し -1 (エラー) を返します。
 (SO_KEEPALIVE、SO_SNDBUF、SO_RCVBUF、IP_TOS、IP_TTL、TCP_KEEPALIVE、TCP_NODELAY の各オプションは listen() または connect() 後の TCP ソケットには使えません)

9.1.16 ソケットの利用可能待ち(selectetsocket)

□ C言語API

```
int ret = selectsocket( int maxfdpl, fd_set *readset, fd_set *writeset, fd_set *exceptset,
                      struct timeval *timeout );
```

□ パラメータ

int	maxfdpl	検査するディスクリプタの範囲
fd_set	*readset	受信用ディスクリプタを指定した領域のアドレス
fd_set	*writeset	送信用ディスクリプタを指定した領域のアドレス
fd_set	*exceptset	例外用ディスクリプタを指定した領域のアドレス
struct timeval	*timeout	タイムアウト時間を格納した領域のアドレス

□ リターンパラメータ

int	ret	正常終了 (≥ 1) タイムアウト (0) 異常終了 (− 1)
fd_set	readset	受信可能となったディスクリプタを示す情報
fd_set	writeset	送信可能となったディスクリプタを示す情報
fd_set	exceptset	例外状態が発生したディスクリプタを示す情報

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EINTR	割り込みエラー (timeoutで指定したタイムアウト時間を経過した)
BSD_EBADF	識別子ソケット未指定 (ディスクリプタに対応するソケットが未生成)
BSD_EFAULT	ポインタアドレス不正 (readset、writeset、exceptset、timeoutが4の倍数以外)
BSD_EINVAL	引数不正 (maxfdpl<0 またはmaxfdpl>FD_SETSIZE(97)、0>tv_sec 、 0≤tv_usec ≤999999 以外)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソケットに対する複数のイベントのいずれかが発生するのを待ちます。UNIX系で使用されている `select()` のコマンド名でも使用できます。

イベントの発生が無い場合は、指定時間が経過するまで待ちます。
`maxfdp1`は、指定するソケット識別子の最大値を指定します（ソケット識別子最大番号プラス1の値）。
 実際には、`maxfdp1`の値をソケット識別子の最大値プラス1の値として取得します（ソケット識別子の最大値は、`maxfdp1-1`の値である）。

`readset`、`writeset`、`exceptset`の領域内のデータは、それぞれ受信可能、送信可能及び例外状態発生を検査するソケットを複数指定します。本BSDソケットAPIでは、例外状態に対応する状態は次の1つのみです。

①ソケットが緊急データを受信した

`readset`、`writeset`、`exceptset`の領域内のデータは、`unsigned long`型であり各ビットがそれぞれのソケット識別子を指定します（例えば、ソケット識別子が5のソケットを指定する場合はデータのbit 5をセットします）。

対象となるビットは、`maxfdp1`により決まります。`maxfdp1`の指定するビットより大きいビットは無視します。

`selectsocket()`がコールされたときには、`readset`、`writeset`、`exceptset`の領域から検査するソケットの指定を受け取ります。また、リターンする際には`readset`、`writeset`、`exceptset`の領域に対し、受信可能、送信可能及び例外状態発生となったソケットを指定する値を設定します。

リターン値は、`readset`、`writeset`、`exceptset`の領域に設定されたデータのビットの総計です（例えば、ソケット識別子5のソケットのみ利用可能でかつそのソケットが受信、送信とも可能な場合、リターン値は2となります）。

*`timeout`には、タイムアウト時間を指定します。`timeout`がNULLの場合は、利用可能なソケットが発生するまで待ち続けます。

`selectsocket()`によりコネクション中のTCPソケットを検査すると、受信不可かつ送信不可の状態という結果になります。コネクションの試みが終了すると（正常、異常問わず）、受信可能や送信可能（あるいは、両方とも可能）になります。そのため、`selectsocket()`により対象のソケットの受信可能／不可と送信可能／不可を検査することでコネクションの試みの終了を確認することができます。

コネクションの確立が成功したかどうかは、ソケットオプションSO_ERRORによりソケットのエラーを調べることで確認できます。ソケットのエラーが正常終了を示している（`errno=0`）ならコネクションの確立に成功しています。

◆ パケットの構造 ◆

<ディスクリプタ情報の構造体>

```
struct {
    fd_mask    fds_bits[4];
} fd_set;
```

<タイムアウト時間の構造体>

```
struct timeval {
    long tv_sec;      // 秒単位の値
    long tv_usec;    // μ秒単位の値
};
```

9.1.17 ブロッキングモード設定 (set_blocking_socket)

□ C言語API

```
int ret = set_blocking_socket( int sockfd, int blocking );
```

□ パラメータ

int	sockfd	ソケット識別子
int	blocking	設定するブロッキングモード (0 : ブロッキング、1 : ノンブロッキング)

□ リターンパラメータ

int	ret	正常終了 (0) または 異常終了 (-1)

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EBADF	識別子無効 (sockfd < 0)
BSD_EINVAL	引数不正 (blocking が 0, 1 以外)
BSD_ENOTSOCK	識別子ソケット未指定 (sockfd と同じ識別子のソケットが存在しない)
BSD_EUSERS	重複操作 (指定したソケットを他のBSD関数が使用中)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

指定したソケットのブロッキングモードを設定します。

blocking が 0, 1 以外の値の場合は、リターン値としてEINVALを返します。

9.1.18 エラーコード取得 (get_errno)

□ C言語API

```
int ret = get_errno( void );
```

□ パラメータ

なし

□ リターンパラメータ

int ret 大域変数に設定されていたエラーコード

□ エラーコード

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

BSDソケットAPI全体で発生した最新のエラーコードを取得します。

9.1.19 スレッドエラーコード取得 (get_thread_errno)

□ C言語API

```
int ret = get_thread_errno( ID threadid );
```

□ パラメータ

ID	threadid	スレッド (タスク) ID

□ リターンパラメータ

int	ret	タスクの大域変数に設定されていたエラーコード (> 0) 正常終了 (0) 関数実行不可 (-1)

□ エラーコード

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

BSDソケットAPIを使用したタスク毎に発生した最新のエラーコードを取得します。
threadidに0を指定した場合は、自タスクのエラーコードを取得します。
取得したタスクのエラーコード (大域変数) は0にクリアされます。

9.1.20 TIMEWAIT時間の変更 (set_sock_timewait)

□ C言語API

```
int ret = set_sock_timewait( int sockfd, short socktimewait );
```

□ パラメータ

int	sockfd	ソケット識別子
short	socktimewait	TIMEWAIT時間 (秒)

□ リターンパラメータ

int	ret	正常終了 (0) または 異常終了 (-1)

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EBADF	識別子無効 (sockfd < 0)
BSD_EINVAL	引数不正 (socktimewait < 1 または socktimewait > 240)
BSD_ENOTSOCK	識別子ソケット未指定 (sockfd と同じ識別子のソケットが存在しない)
BSD_EOPNOTSUPP	未サポートソケット (指定したソケットがTCPのソケットでない)
BSD_EDEADLK	デッドロック条件 (BSDで生成した通信端点が他のプロセスによって削除された)
BSD_EPROTO	プロトコルエラー (使用中の通信端点に対し実行できない)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

指定したソケットの TIMEWAIT 時間を変更します。

TIMEWAIT時間として指定できる範囲は1秒から4分です。socktimewaitが指定できる範囲の値でない場合は、大域変数errnoにEINVALを設定し-1 (エラー) を返します。

指定されたソケットがTCPでない場合は、大域変数errnoにEOPNOTSUPPを設定し-1 (エラー) を返します。

BSDソケットAPIが生成したTCP/IPマネージャの通信端点を既に使用中で、TIMEWAIT時間の変更ができない状態の場合は、大域変数errnoにEPROTOを設定し-1 (エラー) を返します。
(listen() または connect() 後は使えません)

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数errnoにEDEADLKを設定し-1 (エラー) を返します。

9.1.21 受信データ長取得 (get_sock_recvlen)

□ C言語API

```
int ret = get_sock_recvlen( int sockfd );
```

□ パラメータ

```
int          sockfd          ソケット識別子
```

□ リターンパラメータ

```
int          ret          受信データの長さ ( ≥ 0 ) または 異常終了 ( - 1 )
```

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EBADF	識別子無効 (sockfd < 0)
BSD_EPIPE	指定したソケットに対しshutdown() が実行されている
BSD_ENOTSOCK	識別子ソケット未指定 (sockfd と同じ識別子のソケットが存在しない)
BSD_ENOTCONN	未接続エラー (sockfd の示すソケットが未接続)
BSD_EDEADLK	デッドロック条件 (BSDで生成した通信端点が他のプロセスによって削除された)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

ソケットが受信しているデータ長を取得します。
 TCPソケットの場合は、受信用ウィンドバッファ内の有効な受信セグメント長を返します。
 UDPソケットの場合は、キューイングされている全てのデータグラムを合計した長さを返します。

指定したソケットがTCPで接続されていない場合は、大域変数errnoにENOTCONNを設定し-1 (エラー) を返します。

指定したソケットの受信側が、shutdown() によって閉じられている場合は、大域変数errnoにEPIPEを設定し-1 (エラー) を返します。

本BSDソケットAPIはTCP/IPマネージャのITRON TCP/IP API上で動作しています。BSDソケットAPIがITRON TCP/IP API上で生成した通信端点や受付口を他のプロセスがITRON TCP/IP APIを用いて直接操作したことにより、BSDソケットAPIの正常な動作が保証できない状態に陥った場合は、大域変数errnoにEDEADLKを設定し-1 (エラー) を返します。

9.1.22 キープアライブ制御情報の変更 (set_sock_keepalive)

□ C言語API

```
int ret = set_sock_keepalive( int sockfd, T_BSD_KEEPALIVE *p_ktbl );
```

□ パラメータ

int	sockfd	ソケット識別子
T_BSD_KEEPALIVE	*p_ktbl	キープアライブ制御情報を格納した領域のアドレス

□ リターンパラメータ

int	ret	正常終了 (0) または 異常終了 (- 1)

□ エラーコード

BSD_EPERM	関数実行不可
BSD_EBADF	識別子無効 (sockfd < 0)
BSD_EINVAL	引数不正 (keeptime < 1 , kretrytime < 1 , kretrycnt < 1)
BSD_ENOTSOCK	識別子ソケット未指定 (sockfd と同じ識別子のソケットが存在しない)
BSD_EOPNOTSUPP	未サポートソケット (指定したソケットがTCPのソケットでない)
BSD_EDEADLK	デッドロック条件 (BSDで生成した通信端点が他のプロセスによって削除された)
BSD_EPROTO	プロトコルエラー (使用中の通信端点に対し実行できない)

※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

□ 機能

指定したソケットのキープアライブプローブ機能の設定を変更します。

以下にパケットの構造を示します。

```
typedef struct {
    short    k_onoff;        // キープアライブ機能スイッチ (0:OFF/1:ON)
    short    keeptime;       // キープアライブパケット開始タイマ (秒)
    short    kretrytime;     // キープアライブパケットのリトライ間隔 (秒)
    short    kretrycnt;      // キープアライブパケットのリトライ回数
} T_BSD_KEEPALIVE;
```

※ keeptime , kretrytime , kretrycnt は1以上を指定して下さい。

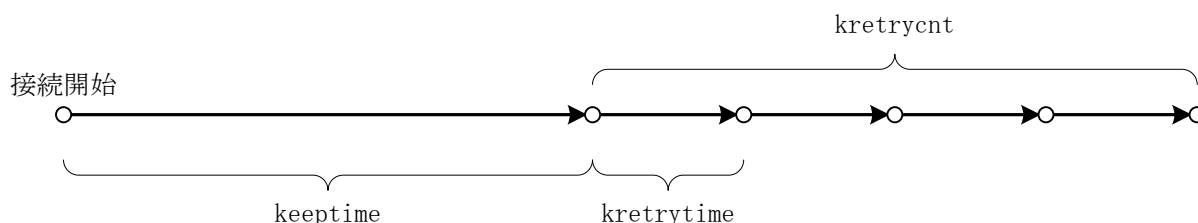
`k_onoff` には、キープアライブ機能の有効／無効を設定します。`k_onoff` に 0 を指定するとキープアライブ機能は無効になり、`keepalive`, `keepalivetime`, `keepalivetimeout` に指定した値は無視します。`k_onoff` に 1 (0 以外の値) を指定するとキープアライブ機能が有効になります。

`k_onoff` に 1 (0 以外の値) を指定した場合は、`keepalive`, `keepalivetime`, `keepalivetimeout` には 1 以上の値を指定して下さい。`keepalive`, `keepalivetime`, `keepalivetimeout` に 0 以下の値を指定した場合は、大域変数 `errno` に `EINVAL` を設定し「-1」(エラー)を返します。

指定されたソケットが TCP でない場合は、大域変数 `errno` に `EOPNOTSUPP` を設定し「-1」(エラー)を返します。

BSD ソケット API が生成した TCP/IP マネージャの通信端点を既に使用中で、キープアライブ機能の設定変更ができない場合は、大域変数 `errno` に `EPROTO` を設定し「-1」(エラー)を返します(`listen()` または `connect()` 後は使用できません)。

キープアライブプローブは、TCP 通信の接続が確立されてから開始されます。設定値のデフォルトは、それぞれ `keepalive`: 7200 秒(2 時間), `keepalivetime`: 75 秒, `keepalivetimeout`: 8 回です。



付録A エラーコード一覧

1 ハードウェア／システムエラー(1 ～ 9)

エラー番号		エラー内容	対策	備考
1	0x0001	—	—	
2	0x0002	OSでのエラー(タイムアウト以外)	OSのリソースなどを確認して下さい	
3	0x0003	内容：IMP処理実行タイムアウト	ドライバをクローズし、再度初期化処理を実行して下さい	
4	0x0004	—	—	
5	0x0005	—	—	
6	0x0006	—	—	
7	0x0007	—	—	
8	0x0008	VIN入力タイムアウト	映像及びカメラの同期信号が正常に入力されているか確認して下さい ドライバをクローズし、再度初期化処理を実行して下さい	

2 不当画面番号エラー(10 ～ 19)

エラー番号		エラー内容	対策	備考
10	0x000A	ImgSrc0の不当画面番号エラー	画面番号を確保しているか確認して下さい	
11	0x000B	ImgSrc1の不当画面番号エラー		
12	0x000C	ImgSrc2の不当画面番号エラー		
13	0x000D	ImgDstの不当画面番号エラー		
14	0x000E	ImgYUVの不当画面番号エラー		
15	0x000F	ImgRGBの不当画面番号エラー		

3 パラメータエラー(20 ～ 29)

エラー番号		エラー内容	対策	備考
20	0x0014	第1引数の設定エラー	引数の設定値を確認して下さい。	
21	0x0015	第2引数の設定エラー		
22	0x0016	第3引数の設定エラー		
23	0x0017	第4引数の設定エラー		
24	0x0018	第5引数の設定エラー		
25	0x0019	第6引数の設定エラー		
26	0x001A	第7引数の設定エラー		
27	0x001B	第8引数の設定エラー		
28	0x001C	第9引数の設定エラー		
29	0x001D	第10引数の設定エラー		

4 ウィンドウ設定エラー(30 ～ 39)

エラー番号		エラー内容	対策	備考
30	0x001E	Src0ウィンドウ設定エラー	ウィンドウの設定値を確認して下さい	
31	0x001F	Src1ウィンドウ設定エラー		
32	0x0020	Src2ウィンドウ設定エラー		
33	0x0021	Dstウィンドウ設定エラー		
34	0x0022	Sysウィンドウ設定エラー		
35	0x0023	Extウィンドウ設定エラー		
36	0x0024	Dst拡張ウィンドウ設定エラー		

5 一般的エラー(40 ～ 79)

エラー番号		エラー内容	対策	備考
40	0x0028	—	—	
41	0x0029	—	—	
42	0x002A	—	—	
43	0x002B	—	—	
44	0x002C	座標設定エラー	座標の設定値を確認して下さい	
45	0x002D	設定条件の範囲エラー	前処理での設定値を確認して下さい	
46	0x002E	画面ID空きなし	不要な画面を解放してリトライして下さい	
47	0x002F	—	—	
48	0x0030	動的メモリ確保エラー (ワークエリアが malloc で確保出来なかった)	まずユーザープログラムによるメモリ破壊がないか確認して下さい。 問題がなければ、開発元へ連絡して下さい	
49	0x0031	不定画面エラー (パイプライン処理で不定画面に設定された)	ユーザプログラムによる画像処理 (パイプライン) を確認して下さい	
50	0x0032	—	—	
51	0x0033	ワーク画面確保エラー	不要な画面を解放してリトライして下さい	
52	0x0034	—	—	
53	0x0035	—	—	
54	0x0036	—	—	
55	0x0037	—	—	

6 正規化関連エラー(80 ～ 89)

エラー番号		エラー内容	対策	備考
80	0x0050	正規化関連処理でのテンプレートサイズミスマッチエラー	テンプレートサイズを確認して下さい。 もしくは、SetCorrMode が正しく実行されているか確認して下さい	
81	0x0051	テンプレート画像の不当画面番号エラー	正しく確保した番号がセットされているか確認して下さい	
82	0x0052	サーチ領域が画面をはみ出すようなサーチポイントがあります	テンプレートサイズを考慮してサーチポイントを指定して下さい。サーチポイント数が正しくない可能性もあります	
83	0x0053	テンプレート画像の分散値エラー	特徴のある画像をテンプレート画像として登録して下さい	
84	0x0054	SetCorrMode での検知物体数エラー	SetCorrMode での検知物体数を確認して下さい	
85	0x0055	SetCorrMode での相関値エラー	SetCorrMode での相関値を確認して下さい	
86	0x0056	マスク処理でのテンプレート登録エラー	開発元へ連絡して下さい	
87	0x0057	テンプレート未設定エラー	テンプレートを登録して下さい	
88	0x0058	テンプレート登録画面番号エラー	登録されたテンプレート画面が有効になっているかを確認して下さい	
89	0x0059	オプティカルフロー未設定エラー	オプティカルフローのモードを設定して下さい	

7 画像ファイリングエラー(100 ～ 109)

エラー番号		エラー内容	対策	備考
100	0x0064	画像ファイルオープンエラー	画像ファイル名を確認して下さい	
101	0x0065	不当画像ファイルエラー	画像ファイルがビットマップファイルか確認して下さい	

8 SDKツール関連エラー(110 ~ 119)

エラー番号		エラー内容	対策	備考
110	0x006E	SDKツールタイムアウトエラー	PCがビジーになっていないか確認して下さい	
111	0x006F	SDKツール起動エラー	ツールのプロセスが残っていないか確認して下さい	
112	0x0070	SDKツール実行ファイル検索エラー	ツールが所定のフォルダにあるか確認して下さい	
113	0x0071	SDKツール通信エラー	ツールが正しく動作しているか確認して下さい	

9 WindowsAPIエラー(120 ~ 129)

エラー番号		エラー内容	対策	備考
120	0x0078	WindowsAPI コマンドエラー	—	
121	0x0079	—	—	

10 その他(130 ~ 209)

エラー番号		エラー内容	対策	備考
130	0x0082	IMPドライバーでのエラー	ドライバをクローズし、再度初期化処理を実行して下さい	
131	0x0083	VINドライバーでのエラー		
132	0x0084	DUドライバーでのエラー		
133	0x0085	アクティブビデオポートが未設定	ActiveVideoPort()でアクティブビデオポートを指定して下さい	
134	0x0086	映像入力サイズが不正です	適切な映像入力サイズを指定して下さい	
135	0x0087	画面サイズ又は画面タイプが不正です		
136	0x0088	システム管理データが不正です		
137	0x0089	画面管理データが不正です	—	
138	0x008A	手続きが正しくありません	正しい手続きで実行して下さい	
139	0x008B	コンフィグレーション設定データエラー	セットアップファイルのコンフィグレーションデータを確認して下さい	
140	0x008C	メモリ管理ドライバーでのエラー	ドライバをクローズし、再度初期化処理を実行して下さい	
160	0x00A0	IMRライブラリエラー	開発元へ連絡してください※	
161	0x00A1	IMR処理タイムアウト	IMRの処理時間が規定(60ms)を超えました。頂点座標の設定値が正しいか(範囲内かなど)を確認した上で、問題なければ開発元に連絡して下さい	
162	0x00A2	IMR資源確保タイムアウト	開発元へ連絡してください※	
165	0x00A5	オブジェクトサイズが不足しています	適切なサイズでオブジェクトを作成してください	
166	0x00A6	オブジェクトが不正です	実行に必要な設定が不足しています(オブジェクトにモデル情報を設定していないなど)	
167	0x00A7	IMRオブジェクトステータスエラー	複数タスクで同じオブジェクトが同時に使われました。適切に排他制御してください	
200	0x00C8	PLL設定エラー	カメラ設定を確認してください	
201	0x00C9	内部計算エラー	実行条件を確認してください	

※正常な使用では発生しません。

あらかじめユーザープログラムによるメモリ破壊がないか確認してください。

11 Windows Driverエラー

エラー番号		エラー内容	原因・対策	備考
131081	0x00020009	DriverEntry 割込みリソース取得エラー	Windowsのシステムリソース不足	
131083	0x0002000B	DriverEntry DMAリソース取得エラー	Windowsのシステムリソース不足	
131084	0x0002000C	タイマ初期化エラー	Windowsのシステムリソース不足	
196609	0x00030001	実行モードエラー	オンボードシステムが動作していません	リセットコマンドでボードをリセットし、初期化し復帰してください
196610	0x00030002	コマンドタイムアウト		
196611	0x00030003	DMA転送タイムアウト		
196612	0x00030004	オンボードシステム システムダウン		
196613	0x00030005	データ転送チェックサムエラー	ボードが壊れているかP C Iバスの接触不良の可能性があります。復帰するには、リセットコマンドでボードをリセットしてください	
131329	0x00020101	パラメータエラー	パラメータを確認下さい	
131330	0x00020102	ドライバが検出できません	ドライバのインストールを確認下さい	
131331	0x00020103	ドライバの情報が取得できません	ドライバのインストールを確認下さい	
131332	0x00020104	ボード枚数が多すぎます (MAX. 16枚)	ボード枚数を 1 6 枚以内にしてください	
131333	0x00020105	ボード番号が不正です (0～15以外)	ボード番号を確認してください	
131334	0x00020106	同じボード番号のボードがあります	ボード番号を確認してください	
131335	0x00020107	デバイス I D のリソースが足りません	不要なデバイス I D をクローズして下さい	
131336	0x00020108	ドライバエラー	ドライバでエラーが発生しました	
131337	0x00020109	オンボードシステムとの通信エラー	－	
131338	0x0002010A	SYSファイルオープンエラー	オンボードシステムファイルが存在しないか壊れています	
131339	0x0002010B	SYSファイルアクセスエラー	オンボードシステムファイルが壊れています	
131340	0x0002010C	SYSファイルのファイルタイプ違い	オンボードシステムファイルが壊れているかオンボードシステムファイルではない	
131341	0x0002010D	メモリ確保エラー	－	
131342	0x0002010E	ブートエラー	オンボードシステムがブート出来なかった	
131343	0x0002010F	ターゲットボード不一致	ターゲットボードに対するオンボードシステムファイル不一致	
131585	0x00020201	パラメータバッファ確保エラー	－	
131586	0x00020202	パラメータバッファオーバーフロー	－	
131587	0x00020203	DLLメモリ確保エラー	－	

12 ファイル管理エラー

エラー番号		エラー内容	備考	
3001	0x0BB9	パラメータエラー		
3002	0x0BBA	オープンモードエラー		
3003	0x0BBB	ファイル名, パス名, ドライブ名サイズオーバー		
3004	0x0BBC	ファイルワーク領域確保エラー		
3005	0x0BBD	ファイル多重オープンエラー		
3006	0x0BBE	追記モードシークエラー		
3007	0x0BBF	ファイル空き容量不足		
3008	0x0BC0	ファイルデータ・チェックサムエラー		
3009	0x0BC1	ファイルアトリビュートエラー		
3010	0x0BC2	ファイル読み出しサイズエラー		
3011	0x0BC3	ファイルハンドル無効		
3012	0x0BC4	ファイル検索エラー(ファイル無し)		
3013	0x0BC5	ドライブ名、ファイル名重複エラー		
3014	0x0BC6	ドライブ名、ファイル名, パス名, ドライブ名エラー		
3015	0x0BC7	無効セクタエラー		
3016	0x0BC8	無効サイズエラー		
3017	0x0BC9	ディスク破損エラー		
3018	0x0BCA	ファイル破損エラー		
3020	0x0BCC	フラッシュメモリセクタ消去エラー		
3021	0x0BCD	フラッシュメモリ書込みエラー		
3022	0x0BCE	フラッシュメモリベリファイエラー		
3023	0x0BCF	フラッシュメモリチェックサム書込みエラー		
3024	0x0BD0	フラッシュメモリデバイスコードエラー		
3030	0x0BD6	ディスクシステムエラー		
3031	0x0BD7			
3032	0x0BD8			
3033	0x0BD9			
3034	0x0BDA			
3035	0x0BDB			
3040	0x0BE0	デバイス検索エラー		USB, CF の アクセス時に 発生
3041	0x0BE1	デバイス名エラー		
3042	0x0BE2	ディレクトリ名エラー		
3043	0x0BE3	ファイルオープンエラー		
3044	0x0BE4	ファイル属性取得エラー		
3045	0x0BE5	ファイル属性設定エラー		
3046	0x0BE6	ファイル時刻取得エラー		
3047	0x0BE7	ファイル時刻設定エラー		
3048	0x0BE8	ファイル読み出しエラー		
3049	0x0BE9	ファイル書き込みエラー		
3050	0x0BEA	ファイルサイズ取得エラー		
3051	0x0BEB	ファイル削除エラー		
3052	0x0BEC	ファイル名変更エラー		
3053	0x0BED	ディレクトリ内容取得エラー		
3054	0x0BEE	デバイス情報取得エラー		
3055	0x0BEF	ディスクサイズ取得エラー		
3056	0x0BF0	ディスクフリーサイズ取得エラー		
3057	0x0BF1	ディレクトリ生成エラー		
3058	0x0BF2	ディレクトリ削除エラー		
3059	0x0BF3	ファイルシステム情報エラー		
3070	0x0BFE	ディスクシステムエラー		
3080	0x0C08			

13 制御系、μITRONエラー

エラー番号		エラー内容	備考
-9	0xFFFFFFFF7	未サポート機能	
-10	0xFFFFFFFF6	サービスコールが組み込まれていない	
-11	0xFFFFFFFF5	属性が不正	
-17	0xFFFFFFFFF	パラメータエラー	
-18	0xFFFFFFFFEE	不正 I D 番号	
-25	0xFFFFFFFFE7	コンテキストエラー	
-28	0xFFFFFFFFE4	サービスコール不正使用	
-33	0xFFFFFFFFDF	メモリ不足	
-34	0xFFFFFFFFDE	I D 番号不足	
-41	0xFFFFFFFFD7	内部オブジェクトエラー	
-42	0xFFFFFFFFD6	オブジェクトが存在しない	
-43	0xFFFFFFFFD5	キューイングまたはネストのオーバーフロー	
-49	0xFFFFFFFFCF	待ち状態強制解除	
-50	0xFFFFFFFFCE	ポーリング失敗またはタイムアウト	
-51	0xFFFFFFFFCD	待ちオブジェクトが削除された	
-52	0xFFFFFFFFCC	待ちオブジェクトの状態が変化	
-57	0xFFFFFFFFC7	ノンブロッキング受付	
-58	0xFFFFFFFFC6	バッファオーバーフロー	
-100	0xFFFFFFFF9C	ドライバインタフェースエラー (PC側のみ)	
-101	0xFFFFFFFF9B	オンボードシステムエラー (PC側のみ)	
-102	0xFFFFFFFF9A	画像処理コマンドエラー	
-103	0xFFFFFFFF99	ファイルオープンエラー	
-104	0xFFFFFFFF98	ファイルタイプエラー (ELF/DWARF形式以外のファイルを指定)	
-105	0xFFFFFFFF97	ファイルアクセスエラー	
-106	0xFFFFFFFF96	作業領域メモリ確保エラー	
-107	0xFFFFFFFF95	ダウンロード領域のメモリ確保エラー	
-108	0xFFFFFFFF94	ベリファイエラー	
-109	0xFFFFFFFF93	エントリシンボル無し	
-110	0xFFFFFFFF92	フラッシュメモリディスク容量不足	

14 イーサネット通信エラー

大域変数	対応定数	エラー内容
BSD_EPERM	1	関数実行不可
BSD_INTR	4	割り込みエラー（待ち状態が強制的に解除された）
BSD_EBADF	9	識別子無効（sockfd < 0）
BSD_ENOMEM	1 2	メモリ不足 （UDPソケットを生成するための受信バッファ資源が足りない）
BSD_EACCES	1 3	許可が無い （SO_BROADCASTがOFFで相手アドレスにブロードキャストを指定）
BSD_EFAULT	1 4	ポインタアドレス不正（p_localaddrが4の倍数以外）
BSD_EINVAL	2 2	引数不正 ①指定したソケットはすでに自通信アドレス情報が決定している ② addrrlen が構造体struct sockaddr_in 型の長さと異なる
BSD_ENFILE	2 3	ファイルテーブル不足（ソケット数に空きがない）
BSD_EPIPE	3 2	指定したソケットに対しshutdown() が実行されている
BSD_EWOULDBLOCK	3 5	ノンブロッキングリターン（ソケットがノンブロッキングモードでかつキュー上に待ち状態のコネクションがない）
BSD_EINPROGRESS	3 6	処理開始（ノンブロッキング設定で、かつ接続がすぐに完了しない）
BSD_ENOTSOCK	3 8	識別子ソケット未指定（sockfd と同じ識別子のソケットが存在しない）
BSD_EDESTADDRREQ	3 9	相手側通信アドレス情報未指定 （sockfdの示すソケットが相手未設定のUDPソケット）
BSD_ENOPROTOOPT	4 2	プロトコル利用不可（level、optの組み合わせが不正）
BSD_EPROTONOSUPPORT	4 3	プロトコル未サポート （アドレスファミリまたはサービスまたはプロトコルが不正）
BSD_EOPNOTSUPP	4 5	未サポートソケット（指定したソケットがTCPのソケットでない）
BSD_EAFNOSUPPORT	4 7	アドレスファミリが不正（アドレスファミリがAF_INT(2)以外）
BSD_EADDRINUSE	4 8	通信アドレス情報使用中 ①指定した通信アドレス情報において、自動割り当て以外の指定をしたIPアドレスまたはポート番号が、他のソケットにおいて使用中 ②指定した通信アドレス情報において、自動割り当て指定をしたIPアドレスまたはポート番号に空きが無い
BSD_EADDRNOTAVAIL	4 9	IPアドレス利用不可
BSD_ECONNRESET	5 4	コネクションリセット（コネクションがリセットされた）
BSD_ENOBUFS	5 5	利用可能エリア不足（UDPの送信バッファが確保できない）
BSD_EISCONN	5 6	接続済みソケット指定
BSD_ENOTCONN	5 7	ソケット未接続 ① TCPにおいて通信相手と接続されていない ② UDPにおいて通信相手が設定されていない
BSD_ETIMEDOUT	6 0	タイムアウトエラー（プロトコルがコネクション確立前に75秒経過）
BSD_ECONNREFUSED	6 1	接続失敗（コネクションが相手によって拒絶された）
BSD_EPROCLIM	6 7	プロセス超過（TCP/IPマネージャのUDP通信端点が足りない）
BSD_EUSERS	6 8	重複操作（指定したソケットを他のBSD 関数が使用中）
BSD_EDEADLK	7 8	デッドロック条件 ① BSDで生成した通信端点またはポートを他のプロセスが使用中 ② BSDで生成した通信端点が他のプロセスによって削除された ③ BSDで生成した受付口が他のプロセスによって削除された
BSD_EPROTO	8 6	プロトコルエラー（使用するIPアドレスが動作していない）

付録B カメラデータ一覧

1. SONY XCL_C500

テーブル名	要素	値	備考
VIDEO_CAM_FRM_INFO	inf1	2	
	inf2	0	
VIDEO_FRM_TBL	pluslng.Vsize	0	未使用
	pluslng.Hsize	0	未使用
	maxfs.Vsize	2052	最大フレームサイズ(有効画素) ※1
	maxfs.Hsize	2448	
VIDEO_CAM_TBL	name	"SONY_XCL_C500"	
	CamAtr	CAMATR_CLINK (0x07)	
	Hcycle	31500	
	OptInfo	0	未使用
	Reserve	0	未使用
VIDEO_CLINK_MODE_TBL	camtype	PoCL/CameraLink (0)	
	scantype	Area Scan(0)	
	tapform	2TAP-2X1Y-PXL (0x23)	TAPフォーマット ※2
	pxlform	8bpp (0)	画素フォーマット ※3
	pxlconv	Non (0)	
	seldval	DVAL (0)	
	vinmode	1TAP/2TAP-single (0)	2TAP-single
	reserve[0-12]	0	未使用
VIDEO_VPP_DATA_TBL	Data[0]	67000	10us単位で切り捨て
	Data[1]	CC1 (1)	トリガ出力先 ※4
	Data[2]	CC2 (2)	DIスルー出力先 ※4
	Data[3]	正極 (0)	CC極性 ※4 bit0~4=CC1~4
	Data[4]	0x00000000	bit0~15:Hoffset、bit16~31:Voffset
	Data[5-19]	0	未使用

※1 カメラデータ設定値 ≤ カメラ側設定 となる様にカメラ側設定を変更する必要があります(WIDTH/HEIGHT)。

※2 TAPフォーマットは カメラ側に2TAPを設定してください(CAMERALINK-TAP)。

※3 画素フォーマットはカメラ側に8bppを設定してください(PIXEL-DEPTH)。

※4 使用するトリガモードに合わせてカメラ側を設定してください(TRG-MODE/TRG-SRC/TRG-POL)。

DIスルー出力はDI端子への入力信号に対し反転した信号がCC2へ出力されます。

2. SONY XCL_C130

テーブル名	要素	値	備考
VIDEO_CAM_FRM_INFO	inf1	2	
	inf2	0	
VIDEO_FRM_TBL	pluslng.Vsize	0	未使用
	pluslng.Hsize	0	未使用
	maxfs.Vsize	964	最大フレームサイズ(有効画素) ※1
	maxfs.Hsize	1280	
VIDEO_CAM_TBL	name	"SONY_XCL_C130"	
	CamAttr	CAMATR_CLINK (0x07)	
	Hcycle	33200	
	OptInfo	0	未使用
	Reserve	0	未使用
VIDEO_CLINK_MODE_TBL	camtype	PoCL/CameraLink (0)	
	scantype	Area Scan(0)	
	tapform	2TAP-2X1Y-PXL (0x23)	TAPフォーマット ※2
	pxlform	8bpp (0)	画素フォーマット ※3
	pxlconv	Non (0)	
	seldval	DVAL (0)	
	vinmode	1TAP/2TAP-single (0)	2TAP-single
	reserve[0-12]	0	未使用
VIDEO_VPP_DATA_TBL	Data[0]	40000	10us単位で切り捨て
	Data[1]	CC1 (1)	トリガ出力先 ※4
	Data[2]	CC2 (2)	DIスルー出力先 ※4
	Data[3]	正極 (0)	CC極性 ※4 bit0~4=CC1~4
	Data[4]	0x00000000	bit0~15:Offset、bit16~31:Voffset
	Data[5-19]	0	未使用

※1 カメラデータ設定値 ≤ カメラ側設定 となる様にカメラ側設定を変更する必要があります(WIDTH/HEIGHT)。

※2 TAPフォーマットはカメラ側に2TAPを設定してください(CAMERALINK-TAP)。

※3 画素フォーマットはカメラ側に8bppを設定してください(PIXEL-DEPTH)。

※4 使用するトリガモードに合わせてカメラ側を設定してください(TRG-MODE/TRG-SRC/TRG-POL)。

DIスルー出力はDI端子への入力信号に対し反転した信号がCC2へ出力されます。

付録C 注意事項

1. 外部トリガを映像入力タイミングとする場合、ハードウェアトリガを利用することで遅延なく映像入力することができます(推奨)。ハードウェアトリガ設定の詳細は2.3.6章SetTriggerModeを参照してください(mode=DI_POSI、DI_NEGA、または、DI_THROUGH指定時)。

最大でタスク優先度3で動作するシステム処理がございます。マルチタスク環境下でソフトウェアトリガを利用する場合、映像入力タイミングが遅延する可能性がありますのでご注意ください。

ソフトウェアトリガご利用の際には映像入力タイミングについて十分評価の上実装ください。ソフトウェアトリガ設定の詳細は2.3.6章SetTriggerModeを参照してください(mode=TRIGGER指定時)。詳細は弊社までお問合せください。

画像認識ユニット NVP-Ax230
Software Development Kit
コマンドリファレンス(第5版)

(C) マクセルシステムテック株式会社

開発元

マクセルシステムテック株式会社

電子機器部

〒992-0021 山形県米沢市花沢3091-6

外販営業部

〒244-0801 神奈川県横浜市戸塚区品濃町549-2 三宅ビル

技術サポート窓口

URL <http://www.systemtech.maxell.co.jp>

E-mail vp-support@maxell.co.jp