第3版

画像処理ユニット　NVP-Ax430シリーズ

ソフトウェア開発キット

NVP-Ax430SDK

コマンドリファレンス

maxell
マクセルフロンティア株式会社

はじめに

このたびは、NVP-Ax430シリーズの画像処理ユニット(NVP-Ax430CL/430ACL/435CL/435FCL)および ソフトウェア開発キット NVP-Ax430SDK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-Ax430シリーズの画像処理ユニットを使用したアプリケーション作成のための基本ソフトウェアであるNVP-Ax430SDKで提供するAPIについての仕様、注意事項などについて詳細を記載しています。また、別紙「ユーザーズマニュアル」はNVP-Ax430SDKで提供する、仕様、操作方法、使用上の注意事項などについて詳細を記載しています。正しくご使用いただくために、各章の熟読をお勧めいたします。

別紙「ハードウェアマニュアル」にNVP-Ax430シリーズのハードウェア仕様、設置方法、注意事項などを記載しています。取扱い、設置方法を誤ると重大事故の可能性がありますので、ご一読をお願い致します。

ご注意

- 本アプリケーションの操作を行う前に、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ください。誤った操作によりシステムの故障の原因となる場合がありますので十分ご注意ください。

- 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。

- NVP-Ax430シリーズに搭載されているLinuxは、32ビットのLinuxシステムであり、管理できる日付と時刻の範囲は、UTC（協定世界時）時刻で1970年1月1日0時00分00秒～2038年1月19日3時14分7秒までとなります。それ以外の日付と時刻の管理が必要なアプリケーションには対応できませんのでご了承ください。

- NVP-Ax430シリーズに搭載されているLinuxは、リアルタイムOSではございません。お客様のアプリケーションで使用されるユースケースにて性能評価を実施頂き、実製品への適用可能性を十分検討頂くようお願い致します。

- 本マニュアルの内容について予告なく変更する場合があります。

すべての商標および登録商標は、それぞれの所有者に帰属します。

・Windows® の正式名称はMicrosoft® Windows® Operating Systemです。
・Microsoft、Windows、Visual Studioは、米国Microsoft Corporationの米国及びその他の国における商標または登録商標です。
・ARM、Cortexは、米国およびその他の国におけるARM Ltd.の登録商標または商標です。
・Linuxは、Linus Torvaldsの米国及びその他の国における登録商標あるいは商標です。
・本書では、商標または登録商標のマーク(® 、™)を省略して記載する場合がございます。

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	ＲＧＢ画面（3面１組）
RGB.R	ＲＧＢのＲ成分の画面
RGB.G	ＲＧＢのＧ成分の画面
RGB.B	ＲＧＢのＢ成分の画面
RGB16	ＲＧＢ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
ＡＰＩ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
ＡＰＩ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。
ＩＭＰ：画像処理モジュール
ＶＰＰ：映像前処理モジュール
ＶＩＮ：映像入力モジュール
ＤＵ　：映像出力モジュール

(6) 機能
ＡＰＩ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ＡＰＩを使用するにあたり、注意事項を説明します。

(1) 画面番号
ＡＰＩ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が、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。

(4) PCとNVPの通信
PCとNVP間で通信できなくなった場合、特に記載がない限り、以下のエラーコードが設定されます。
１、２項　　：0x00020108 ドライバエラー又は通信エラー
４、５、６、７項　：-100　　ドライバエラー又は通信エラー

Microsoft,Windows,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。

画像処理コマンドの最大処理サイズ

(4K:4096,3K:3072,2K:2048,1K:1024)

項目				コマンド名称	幅	高さ	備考
1	初期化ｺﾏﾝﾄﾞ	1.1	ﾃﾞﾊﾞｲｽID無しﾘﾓｰﾄｺﾏﾝﾄﾞのAPI初期化	1	StartIP	－	－	
				2	StopIP	－	－	
				3	ActiveIP	－	－	
		1.2	ﾃﾞﾊﾞｲｽID付きﾘﾓｰﾄｺﾏﾝﾄﾞのAPI初期化	1	OpenIPDev	－	－	
				2	CloseIPDev	－	－	
		1.3	API初期化ｻﾎﾟｰﾄｺﾏﾝﾄﾞ	1	EnumAttachIPDev	－	－	
				2	ResetIP	－	－	
				3	ShutdownIP	－	－	
				4	GetIPLastError	－	－	
		1.4	NVP-LinuxのAPI初期化	1	StartIP	－	－	
				2	StopIP	－	－	
		1.5	NVP-LinuxのAPI初期化ｻﾎﾟｰﾄｺﾏﾝﾄﾞ	1	ResetIP	－	－	
2	画像処理ｺﾏﾝﾄﾞ	2.1	ｼｽﾃﾑ制御	1	InitIP	－	－	
				2	InitIPExt	－	－	
				3	EnableIPErrorMessage	－	－	
				4	DisableIPErrorMessage	－	－	
				5	ReadIPErrorTable	－	－	
				6	ClearIPError	－	－	
				7	SetIPDataType	－	－	
				8	GetIPVersionInfo	－	－	
				9	OpenIMP	－	－	
				10	CloseIMP	－	－	
				11	ActiveIMP	－	－	
		2.2	画像ﾒﾓﾘ領域管理	1	AllocImg	16K	16K	幅は32byteｱﾗｲﾝ
				2	AllocComponentImg	16K	16K	幅は32byteｱﾗｲﾝ
				3	FreeImg	－	－	
				4	FreeAllImg	－	－	
				5	ReadImgTable	－	－	
				6	GetElementImgID	－	－	
				7	ChangeImgDataType	－	－	
				8	SetWindow	－	－	
				9	SetAllWindow	－	－	
				10	ResetAllWindow	－	－	
				11	EnableIPWindow	－	－	
				12	DisableIPWindow	－	－	
				13	ReadWindow	－	－	
				14	AllocRGBImg	16K	16K	幅は32byteｱﾗｲﾝ
				15	AllocRGB16Img	16K	16K	幅は32byteｱﾗｲﾝ
				16	GetGImgID	－	－	
				17	GetBImgID	－	－	
				18	AllocRGBYImg	16K	16K	幅は32byteｱﾗｲﾝ
				19	GetYImgID	－	－	
		2.3	映像入力	1	ActiveVideoPort	－	－	
				2	SelectCamera	－	－	
				3	SetVideoFrame	16K	4K	幅は32byteｱﾗｲﾝ
				4	GetCamera	－	－	
				5	SetVFDelay	－	－	
				6	SetTriggerMode	－	－	
				7	SetStrobeMode	－	－	
				8	SetVideoLUTMode	－	－	
				9	GetSceneCounter	－	－	
				10	SetSceneCounter	－	－	
				11	SetCameraData	－	－	
				12	GetCameraData	－	－	
				13	LoadCameraFile	－	－	
				14	CaptureContinuous	－	－	
				15	StopCaptureContinuous	－	－	
				16	GetCameraReqScene	－	－	
				17	GetCameraResScene	－	－	
				18	SetEncoderMode	－	－	
				19	EncoderStart	－	－	
				20	EncoderStop	－	－	
				21	EncCountStart	－	－	
				22	EncCountPause	－	－	
				23	EncCountResume	－	－	
				24	EncCountReset	－	－	
				25	SetVideoTimeout	－	－	
				26	SetCLComCH	－	－	

		2.4	映像表示	1	SetConfigDisp	4K	4K	最大表示ｻｲｽﾞは1280x720
				2	SetDispPalette	－	－	
				3	SelectDisp	－	－	
				4	DispCamera	－	－	
				5	DisableDisp	－	－	
				6	DispImg	－	－	
				7	NoDisp	－	－	
				8	DispOverlap	－	－	
				9	DisableOverlap	－	－	
				10	CombineRGB	4K	4K	
				11	CombineRGBEx	4K	4K	
		2.5	画像ｸﾘｱ	1	IP_ClearAllImg	4K	4K	
				2	IP_ClearImg	4K	4K	
				3	IP_Const	4K	4K	
		2.6	転送/ｱﾌｨﾝ変換	1	IP_Copy	4K	4K	
				2	IP_Zoom	4K	4K	
				3	IP_ZoomExt	4K	4K	
				4	IP_ZoomOut	4K	4K	
				5	IP_ZoomOutExt	4K	4K	
				6	IP_ZoomIn	4K	4K	
				7	IP_ZoomInExt	4K	4K	
				8	IP_ZoomS	4K	4K	
				9	IP_Shift	4K	4K	
				10	IP_Rotate	4K	4K	
		2.7	２値化	1	IP_Binarize	4K	4K	
				2	IP_BinarizeExt	4K	4K	
		2.8	画素変換	1	IP_Invert	4K	4K	
				2	IP_Minus	4K	4K	
				3	IP_Abs	4K	4K	
				4	IP_AddConst	4K	4K	
				5	IP_SubConst	4K	4K	
				6	IP_SubConstAbs	4K	4K	
				7	IP_MultConst	4K	4K	
				8	IP_MinConst	4K	4K	
				9	IP_MaxConst	4K	4K	
				10	WriteConvertLUT	4K	4K	
				11	IP_ConvertLUT	4K	4K	
				12	IP_ShiftDown	4K	4K	
				13	IP_ShiftUp	4K	4K	
		2.9	画像間算術演算	1	IP_Add	4K	4K	
				2	IP_Sub	4K	4K	
				3	IP_SubAbs	4K	4K	
				4	IP_Comb	4K	4K	
				5	IP_CombAbs	4K	4K	
				6	IP_Mult	4K	4K	
				7	IP_Average	4K	4K	
				8	IP_Min	4K	4K	
				9	IP_Max	4K	4K	
				10	IP_SubConstAbsAdd	4K	4K	
				11	IP_SubConstMultAdd	4K	4K	
				12	IP_SubConstMult	4K	4K	
				13	IP_CombDrop	4K	4K	
		2.10	画像間論理演算	1	IP_And	4K	4K	
				2	IP_Or	4K	4K	
				3	IP_Xor	4K	4K	
				4	IP_InvertAnd	4K	4K	
				5	IP_InvertOr	4K	4K	
				6	IP_Xnor	4K	4K	
		2.11	２値画像形状変換	1	IP_PickNoise4	3K	4K	
				2	IP_PickNoise8	3K	4K	
				3	IP_Outline4	3K	4K	
				4	IP_Outline8	3K	4K	
				5	IP_Dilation4	3K	4K	
				6	IP_Dilation8	3K	4K	
				7	IP_Erosion4	3K	4K	
				8	IP_Erosion8	3K	4K	
				9	IP_Thin4	3K	4K	
				10	IP_Thin8	3K	4K	
				11	IP_Shrink4	3K	4K	
				12	IP_Shrink8	3K	4K	

		2.12	ｺﾝﾎﾞﾘｭｰｼｮﾝ	1	IP_SmoothFLT	3K	4K	
				2	IP_EdgeFLT	3K	4K	
				3	IP_EdgeFLTAbs	3K	4K	
				4	IP_Lapl4FLT	3K	4K	
				5	IP_Lapl8FLT	3K	4K	
				6	IP_Lapl4FLTAbs	3K	4K	
				7	IP_Lapl8FLTAbs	3K	4K	
				8	IP_LineFLT	3K	4K	
				9	IP_LineFLTAbs	3K	4K	
		2.13	ﾐﾆ/ﾏｯｸｽﾌｨﾙﾀ	1	IP_MinFLT	3K	4K	
				2	IP_MinFLT4	3K	4K	
				3	IP_MinFLT8	3K	4K	
				4	IP_MaxFLT	3K	4K	
				5	IP_MaxFLT4	3K	4K	
				6	IP_MaxFLT8	3K	4K	
				7	IP_LineMinFLT	3K	4K	
				8	IP_LineMaxFLT	3K	4K	
		2.14	ﾗﾝｸﾌｨﾙﾀ	1	IP_RankFLT	3K	4K	
				2	IP_Rank4FLT	3K	4K	
				3	IP_Rank8FLT	3K	4K	
				4	IP_MedFLT	3K	4K	
				5	IP_Med4FLT	3K	4K	
				6	IP_Med8FLT	3K	4K	
		2.15	ﾗﾍﾞﾘﾝｸﾞ	1	IP_Label4	3K	4K	
				2	IP_Label8	3K	4K	
				3	IP_Label4withAreaFLT	2K	2K	
				4	IP_Label8withAreaFLT	2K	2K	
				5	IP_Label4withAreaFLTSort	2K	2K	
				6	IP_Label8withAreaFLTSort	2K	2K	
				7	IP_ExtractLORegionX	2K	2K	
				8	IP_ExtractLORegionY	2K	2K	
				9	IP_ExtractLOArea/	2K	2K	
					IP_ExtractLOAreaExt	2K	2K	
				10	IP_ExtractLOGravity	2K	2K	
		2.16	濃淡画像 特徴量抽出	1	IP_ExtractGOFeatures	2K	2K	
				2	IP_Histogram	2K	2K	
					IP_HistogramShort	2K	2K	
				3	IP_HistogramFeatures	2K	2K	
				4	IP_ProjectGO	2K	2K	
				5	IP_ProjectGOonX	2K	4K	
				6	IP_ProjectGOonY	4K	2K	
				7	IP_ProjectGOMaxValue	2K	2K	
				8	IP_ProjectGOMinValue	2K	2K	
				9	IP_ProjectBlockGO	2K	2K	
				10	IP_ProjectBlockGOMinMaxValue	2K	2K	
				11	IP_ProjectLabelGO	2K	2K	
				12	IP_ProjectLabelGOMinMaxValue	2K	2K	
		2.17	２値画像 特徴量抽出	1	IP_ExtractBOArea	4K	4K	
				2	IP_ExtractBOFeatures	2K	2K	
				3	IP_ProjectBO	2K	2K	
				4	IP_ProjectBORegionX	4K	2K	
				5	IP_ProjectBORegionY	2K	4K	
				6	IP_ProjectBlockBO	2K	2K	
		2.18	画像ﾒﾓﾘｱｸｾｽ	1	OpenImg	－	－	
				2	OpenImgExt	－	－	
				3	CloseImg	－	－	
				4	ReadImg	4K	4K	
				5	WriteImg	4K	4K	
				6	SetPixelPointer	4K	4K	
				7	ReadPixel	4K	4K	
				8	WritePixel	4K	4K	
				9	ReadPixelContinue	4K	4K	
				10	WritePixelContinue	4K	4K	
				11	RefreshImg	－	－	
				12	OpenImgDirect	－	－	
				13	CloseImgDirect	－	－	

		2.19	２値ﾊﾟｲﾌﾟﾗｲﾝﾌｨﾙﾀ	1	SetTrsPipelineFLTMode	－	－	
				2	IP_TrsPipelineFLT	3K	4K	
		2.20	ﾊﾟｲﾌﾟﾗｲﾝ制御	1	EnablePipeline	－	－	
				2	DisablePipeline	－	－	
		2.21	２値ﾏｯﾁﾝｸﾞﾌｨﾙﾀ	1	SetBinMatchTemplate	－	－	
				2	IP_BinMatchFLT	3K	4K	
		2.22	正規化相関	1	SetCorrMode	－	－	
				2	EnableCorrMask	－	－	
				3	DisableCorrMask	－	－	
				4	SetCorrControl	－	－	
				5	SetCorrTemplate	－	－	
				6	SetCorrTemplateExt	－	－	
				7	IP_Corr	4K	4K	
				8	IP_CorrPecise	4K	4K	
		2.23	ｸﾞﾗﾌｨｯｸｽ	1	SetDrawMode	－	－	
				2	SetStringAttributes	－	－	
				3	RefreshGraphics	－	－	
				4	DrawString	4K	4K	
				5	DrawLine	4K	4K	
				6	DrawSegments	4K	4K	
				7	DrawLines	4K	4K	
				8	DrawRectangle	4K	4K	
				9	DrawPolygon	4K	4K	
				10	DrawArc	4K	4K	
				11	FillRectangle	4K	4K	
				12	FillPolygon	4K	4K	
		2.24	画像ﾌｧｲﾘﾝｸﾞ	1	LoadBMPFile	4K	4K	
				2	SaveBMPFile	4K	4K	
		2.25	ﾗﾝﾚﾝｸﾞｽ･ﾗﾍﾞﾘﾝｸﾞ	1	IP_Label4byRL	3K	4K	
				2	IP_Label8byRL	3K	4K	
				3	IP_Label4byRLwithAreaFLT	3K	4K	
				4	IP_Label8byRLwithAreaFLT	3K	4K	
				5	IP_Label4byRLwithAreaFLTSort	3K	4K	
				6	IP_Label8byRLwithAreaFLTSort	3K	4K	
				7	IP_Label4byRLExt	3K	4K	
				8	IP_Label8byRLExt	3K	4K	
				9	IP_Label4byRLwithAreaFLTExt	3K	4K	
				10	IP_Label8byRLwithAreaFLTExt	3K	4K	
				11	IP_Label4byRLwithAreaFLTSortEx	3K	4K	
				12	IP_Label8byRLwithAreaFLTSortEx	3K	4K	
				13	IP_LabelCombine	3K	4K	
		2.26	2値化閾値算出支援	1	HistAnalyze	－	－	
		2.27	拡張ｺﾝﾎﾞﾘｭｰｼｮﾝ	1	IP_SmoothFLT5x5	1K	4K	
				2	IP_SmoothFLT7x7	1K	4K	
				3	IP_EdgeFLT5x5	1K	4K	
				4	IP_EdgeFLT7x7	1K	4K	
				5	IP_MinFLT5x5	1K	4K	
				6	IP_MaxFLT5x5	1K	4K	
				7	IP_SmoothFLTExt	3K	4K	
				8	IP_EdgeFLTAbsExt	3K	4K	
				9	IP_SmoothFLT5x5Ext	1K	4K	
				10	IP_SmoothFLT7x7Ext	1K	4K	
				11	IP_EdgeFLT5x5Ext	1K	4K	
				12	IP_EdgeFLT7x7Ext	1K	4K	
		2.28	拡張画像処理	1	IP_Sobel	3K	4K	
				2	IP_SobelBinarize	3K	4K	
				3	IP_RegisterLUT	－	－	
				4	IP_Prewitt	3K	4K	
				5	IP_PrewittBinarize	3K	4K	
		2.29	線分化	1	ExtractPolyline	4K	4K	
				2	PolyArea	4K	4K	
				3	PolyPerim	4K	4K	
				4	PolyGrav	4K	4K	
				5	PolyFeatures	4K	4K	
		2.30	2値画像の穴埋め	1	IP_FillHole	2K	2K	
				2	IP_FillHoleExt	2K	2K	

		2.31	RGBLUT変換	1	CreateRGBLUT	－	－	
				2	DeleteRGBLUT	－	－	
				3	OpenRGBLUT	－	－	
				4	CloseRGBLUT	－	－	
				5	IP_ConvertRGBLUT	4K	4K	
				6	IP_ConvertRGBLUTEx	4K	4K	
		2.32	RAW/RGB変換	1	CreateConvertRAWData	－	－	
				2	DeleteConvertRAWData	－	－	
				3	SetRAWtoRGBGain	－	－	
				4	IP_ConvertRAWtoRGB	3K	4K	
				5	IP_ConvertRAWtoRGBEx	3K	4K	
		2.33	歪み補正	1	imrcmd_CreateObj	－	－	
				2	imrcmd_DeleteObj	－	－	
				3	imrcmd_SetObjMode	－	－	
				4	imrcmd_SetGrid	－	－	
				5	imrcmd_ImrExec	2K	4K	Dstｳｨﾝﾄﾞｳｻｲｽﾞは幅1K
3	VP互換ｺﾏﾝﾄﾞ	3.1	正規化相関	1	vpxAllocCorrTemplate	－	－	
				2	vpxFreeCorrTemplate	－	－	
				3	vpxReadCorrTemplate	－	－	
				4	vpxWriteCorrTemplate	－	－	
				5	vpxEnableCorrMask	－	－	
				6	vpxDisableCorrMask	－	－	
				7	vpxSetCorrTemplate	－	－	
				8	vpxSetCorrTemplateExt	－	－	
				9	vpxIP_CorrStep	4k	4k	
				10	vpxIP_CorrPoint	4k	4k	
				11	vpxIP_CorrPrecise	4k	4k	
				12	vpxSetSearchDistance	－	－	
				13	vpxSetCorrMode	－	－	
				14	vpxSetCorrPrecise	－	－	
		3.2	直線抽出	1	GetHoughLine	4k	4k	
				2	GetHoughLineRow	4k	4k	
				3	GetHoughLineRowExt	4k	4k	
				4	GetSideHoughLine	4k	4k	
				5	GetCrossPoint	－	－	
				6	GetRectPoint	－	－	
				7	GetRectCenter	－	－	
				8	GetAnglePoint4	－	－	
				9	GetAnglePoint2	－	－	
		3.3	ｲﾒｰｼﾞｷｬﾘﾊﾟ	1	ProjectLine	4K	4K	
				2	LineEdgeFilter	－	－	
				3	LineEdgeFilterExt	－	－	
				4	LineCaliper	－	－	
				5	CaliperLPtoSP	－	－	
				6	GetCaliperScore	－	－	
				7	SetCaliperWidth	－	－	
				8	LineEdgeFinder	－	－	
				9	EdgeFinderLPtoSP	－	－	
		3.4	図形・文字描画	1	PutLine	4K	4K	
				2	PutCross	4K	4K	
				3	PutLineWindow	4K	4K	
				4	PutPolygon	4K	4K	
				5	PutCircle	4K	4K	
				6	PutArc	4K	4K	
				7	PutRectangle	4K	4K	
				8	PutTriangle	4K	4K	
				9	PutDiamond	4K	4K	
				10	PutCrossRect	4K	4K	
				11	PutTwinRectangle	4K	4K	
				12	PutArcExt	4K	4K	
				13	PutEllipse	4K	4K	
				14	PutString	4K	4K	

4	ﾘﾓｰﾄﾓｼﾞｭｰﾙ制御ｺﾏﾝﾄﾞ	4.1	ﾓｼﾞｭｰﾙｻﾎﾟｰﾄ	1	GetTargetAppInfo	－	－	
				2	AllocTargetAppInfo	－	－	
				3	FreeTargetAppInfo	－	－	
				4	InitIPParamTable	－	－	
				5	SetIntegerParam	－	－	
				6	SetFloatParam	－	－	
				7	SetParamTable	－	－	
				8	AllocParamTable	－	－	
				9	GetParamTable	－	－	
				10	SetParamTableExt	－	－	
				11	AllocParamTableExt	－	－	
				12	ReadIPMemData	－	－	
				13	WriteIPMemData	－	－	
		4.2	ｲﾝﾃﾘｼﾞｪﾝﾄﾓｼﾞｭｰﾙ制御	1	RegistInteliIPModule	－	－	
				2	ExecuteInteliIPModule	－	－	
		4.3	PIO割込ｵﾌﾞｼﾞｪｸﾄ制御	1	CreateInterruptLink	－	－	
				2	DeleteInterruptLink	－	－	
				3	EnableInterruptObject	－	－	
				4	DisableInterruptObject	－	－	
		4.4	ﾀｽｸ制御	1	CreateIPTask	－	－	
				2	RegistIPTask	－	－	
				3	StartIPTaskwithParam	－	－	
				4	WakeupIPTaskwithParam	－	－	
				5	WaitforIPTaskSignal	－	－	
				6	ReadyforWaitIPTaskSignal	－	－	
				7	CancelIPTaskWait	－	－	
				8	StartIPTask	－	－	
				9	WakeupIPTask	－	－	
				10	CancelIPTask	－	－	
				11	SleepIPTask	－	－	
		4.5	PCとの同期	1	SendSignaltoPC	－	－	
5	ﾎﾞｰﾄﾞ内ﾃﾞﾊﾞｲｽ制御ｺﾏﾝﾄﾞ	5.1	Ｉ／Ｏ入出力	1	ipport_Read	－	－	
				2	ipport_Write	－	－	
		5.2	SCIｺﾝﾄﾛｰﾙ	1	scicom_Setup	－	－	
				2	scicom_Purge	－	－	
				3	scicom_Recv	－	－	
				4	scicom_tRecv	－	－	
				5	scicom_Send	－	－	
		5.3	タイマ	1	SetIPTimeMode	－	－	
				2	startGetIPTime	－	－	
				3	timeGetIPTime	－	－	
				4	stopGetIPTime	－	－	
				5	GetPerformanceFrequency	－	－	
				6	GetPerformanceCounter	－	－	
				7	IPSleep	－	－	
				8	SetIPTime	－	－	
				9	GetIPTime	－	－	
				10	GetLinuxErrno	－	－	
		5.4	WDTｺﾏﾝﾄﾞ	1	WDT_Start	－	－	
				2	WDT_Stop	－	－	
				3	WDT_Reset	－	－	
				4	WDT_GetStatus	－	－	
				5	WDT_ClearStatus	－	－	

6	ﾌｧｲﾙｱｸｾｽｺﾏﾝﾄﾞ	6.1	ﾌｧｲﾙｱｸｾｽ	1	fmOpen	－	－	
				2	fmClose	－	－	
				3	fmRead	－	－	
				4	fmWrite	－	－	
				5	fmGetFileSize	－	－	
				6	fmDelete	－	－	
				7	fmRename	－	－	
				8	fmFindFile	－	－	
				9	fmChgAttr	－	－	
				10	fmFileCopy	－	－	
				11	fmSeek	－	－	
				12	fmCreateDirectory	－	－	
				13	fmRemoveDirectory	－	－	
				14	fmFindFirstFile	－	－	
				15	fmFindNextFile	－	－	
				16	fmFindClose	－	－	
				17	fmRefAttr	－	－	
				18	fmDriveSpace	－	－	
				19	fmCopyFile	－	－	
7	Windows ｱﾌﾟﾘｹｰｼｮﾝ ｻﾎﾟｰﾄｺﾏﾝﾄﾞ	7.1	Windows ｱﾌﾟﾘｹｰｼｮﾝｻﾎﾟｰﾄ	1	ReadBMPImg	4K	4K	
				2	WriteBMPImg	4K	4K	
				3	CreateBMPImgObject	－	－	
				4	DeleteBMPImgObject	－	－	
				5	GetBMPImgInfoEntry	－	－	
				6	GetBMPImgBitsEntry	－	－	

※グレーハッチングの項目は画像処理サイズに影響ありません。

画像処理コマンドのパラメータ制限

(1)パラメータの制限値

	項目	制限値	説明
1	画像メモリ (画面数)	256MB (VGAｻｲｽﾞ、8bitﾓﾉｸﾛで800面以上)	画像メモリは、画像メモリ確保コマンドで画面として確保される以外に、ヒープ領域に使用されます。確保できる画面数はヒープの使用状況により変わりますので注意が必要です。
2	ラベリング処理	ラベル数　　255個 仮ラベル数　4094個 合流対　　2046個	ラベリング処理の過程で、仮ラベル、合流対が抽出されますが、制限値を超えた場合はエラーとなります。
3	２値マッチング フィルタ	テンプレート数　64個	ユーザが登録することができるテンプレートの最大数です。テンプレート番号は0～63になります。
4	正規化相関	テンプレート数　256個	ユーザが登録することができるテンプレートの最大数です。テンプレート番号は0～255になります。
5	直線抽出	ポイント数　4096個	直線を抽出するために設定できるポイント数です。 最大ポイント数(_POINTTBL_DATA_NUM)はvp900.hで定義しています。
6	イメージキャリパ	直線上への投影データ数　4096個 エッジデータ数　　4096個	投影データ数およびエッジデータ数は、データを格納するテーブルサイズで制限されています。 最大データ数(_LINETBL_DATA_NUM)はvp900.hで定義しています。
		キャリパデータ数　　4096個	キャリパデータ数は、データを格納するテーブルサイズで制限されています。 最大キャリパデータ数(_CALIPER_DATA_NUM)はvp900.hで定義しています。
		エッジファインダデータ数　4096個	エッジファインダデータ数は、データを格納するテーブルサイズで制限されています。 最大エッジファインダデータ数(_EDGEFINDER_DATA_NUM)はvp900.hで定義しています。

(2)タイムアウト時間

	項目	制限値	説明
1	映像入力	タイムアウト時間　300ms	映像入力タイムアウト時間のデフォルト値です。 映像入力タイムアウト時間はSetVideoTimeoutコマンドで変更可能です。
2	画像処理	IMPタイムアウト時間　6000ms	画像認識プロセッサ（IMP)の画像処理タイムアウト時間です。
3	歪み補正	IMRタイムアウト時間　1000ms	歪み補正(IMR)処理のタイムアウト時間です。

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

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	
		CONNECT_PORT_DEAMON	リモートコマンドデーモンと接続 接続の前にリモートコマンドデーモンを実行している必要があります。SW1-1をONにしてLinuxを起動することで実行されます。
		CONNECT_PORT_USRAPL	ユーザーアプリケーションと接続 接続の前にユーザーアプリケーションを実行している必要があります
		CONNECT_KEEPALIVE	キープアライブ有効

❑ リターンパラメータ

int	ret	正常終了（０） 異常終了（－１） 対象スレッドで既にStartIPが実行されている又は内部データ登録エラー(－４１） ボード番号エラー（－１７）

❑ 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	パラメータエラー又は内部管理データエラー
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
0x00020107	ドライバを動作させるリソース不足
0x00020108	ドライバエラー又は通信エラー
0x0002010E	NVPシステムが動作していない
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との接続を行います。本コマンドはWindowsからのリモートコマンドでデバイス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、リモートシステムに設定した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（０～１５）

❑ リターンパラメータ

int	ret	正常終了（０） 異常終了（－１） 内部データ登録エラー(－４１） ボード番号エラー（－１７）

❑ 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　パラメータエラー又は内部管理データエラー

※ 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接続のクローズ処理を行います。本コマンドは、StartIPコマンドで接続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） 内部データ登録エラー(－４１） ボード番号エラー（－１７）

❑ 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。

CONNECT_PORT_DEAMON	リモートコマンドデーモンと接続 接続の前にリモートコマンドデーモンを実行している必要があります。SW1-1をONにしてLinuxを起動することで実行されます。
CONNECT_PORT_USRAPL	ユーザーアプリケーションと接続 接続の前にユーザーアプリケーションを実行している必要があります
CONNECT_KEEPALIVE	キープアライブ有効

❑ リターンパラメータ

DEVID	devID	＞０ ：デバイスＩＤ ０ ：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	パラメータエラー又は内部管理データエラー
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
0x00020107	ドライバを動作させるリソース不足
0x00020108	ドライバエラー又は通信エラー
0x0002010E	オンボードシステムが動作していない
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との接続を行いNVPに対する制御情報を識別するためのデバイスＩＤ（ devID ）を取得します。本コマンドはWindowsからのリモートコマンドでデバイスＩＤ有りの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、レジストリに設定した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）。

1.2.2 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ＩＤ

❑ 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ＩＤ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 API初期化サポートコマンド

1.3.1 装着ボードの列挙（ EnumAttachIPDev ）

❑ C言語API

```
int  Num = EnumAttachIPDev( int *pBoard );
```

❑ パラメータ

int	*pBoard	装着ボードのボード番号を格納するテーブル

❑ リターンパラメータ

int	Num	≧１：装着されているボード枚数 ０：ボードが装着されていない

❑ エラーコード

Numが０で終了した場合、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

※ 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に装着されているボードの枚数とボード番号を列挙します。本コマンドは、デバイスＩＤ無しのＡＰＩでもデバイスＩＤ付きのＡＰＩ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。

ボード番号格納テーブルには装着されているボード枚数分のボード番号が出力されます。装着される可能性があるボード枚数以上の領域を確保して下さい。本コマンドのリターンコードが装着されているボード枚数で、その枚数分のデータ以外は無効です。また、pBoardテーブルにはボード番号順にボード番号が格納されないことがあります。

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（０～１５）

int　　Mode　　モード

❑ 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

※ 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、Linuxを再起動します。
本コマンドは、デバイスＩＤ無しのＡＰＩでもデバイスＩＤ付きのＡＰＩ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、レジストリに設定した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）。

1.3.3 画像処理ボードのシャットダウン（ ShutdownIP）

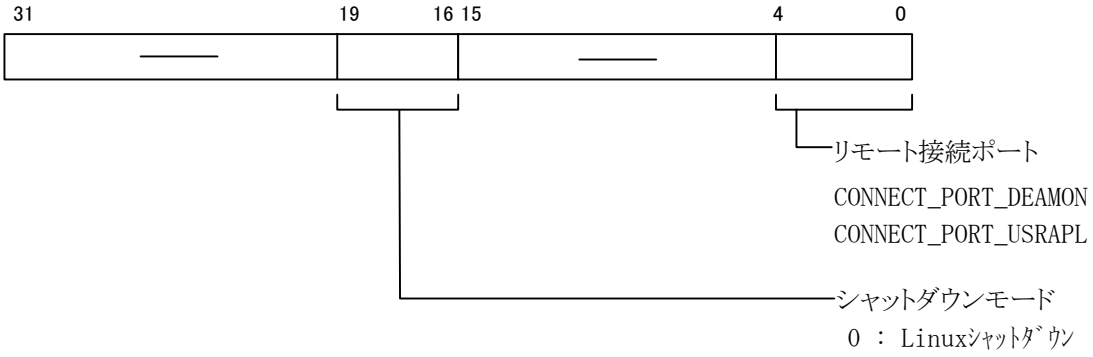
❑ C言語API

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

❑ パラメータ

int　　BoardNo　　ボード番号（０～１５）

int　　Mode　　モード



❑ 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

※ 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Linuxをシャットダウンします。
本コマンドは、デバイスＩＤ無しのＡＰＩでもデバイスＩＤ付きのＡＰＩ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、レジストリに設定した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）。

1.3.4 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

ＡＰＩ初期化コマンドで発生したエラー情報を読み出します。

1.4 NVP-LinuxのAPI初期化

1.4.1 NVP-LinuxのAPI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を指定してください。

int　　Opt　　オプション

0	スタンドアロン画像認識アプリケーション
REMOTE_SERVER_BOOT ｜ CONNECT_PORT_USRAPL	リモートサーバー起動ユーザーアプリケーション
PIOINT_SERVER_BOOT	PIO割込処理サーバー起動 PIO入力ポート:DI0～DI7
PIOINT1_SERVER_BOOT	PIO#1割込処理サーバー起動 PIO入力ポート:DI8～DI15（NVP-Ax430CL除く）

PIO割込処理サーバースレッドのCPUアフィニティ設定
　PIO割込処理サーバースレッドのCPUアフィニティを設定する場合は、「Opt」パラメータに下表の値を論理和で指定してください。

CPUアフィニティ	内容
PIOINT_SERVER_AFFINITY_CPU0	CPU0 が有効
PIOINT_SERVER_AFFINITY_CPU1	CPU1 が有効
PIOINT_SERVER_AFFINITY_CPU2	CPU2 が有効
PIOINT_SERVER_AFFINITY_CPU3	CPU3 が有効

❑ 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

－９	ドライバ初期化エラー
－２０	ボード番号エラー
－１３８	StartIP多重実行

※ 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側のユーザーアプリケーションは、本コマンドを使用し初期化を行ってください。
本コマンドは1.1.1 StartIPと同じですが、NVP-Linuxで使用する場合、ボード番号「０」を指定してください。
オプションは、スタンドアロンの画像認識アプリケーションの場合「０」を指定します。
リモートモジュールフレームワークを使用するアプリケーションの場合「REMOTE_SERVER_BOOT ｜ CONNECT_PORT_USRAPL」を指定してください。
また、PIOの割込み起動モジュールを使用する場合は「PIOINT_SERVER_BOOT」及び「PIOINT1_SERVER_BOOT」を論理和で指定して下さい。

PIO割込みサーバースレッドのCPUアフィニティ(実行するCPUの明示的な指定)を指定する場合、リモート、PIO割込みサーバー起動のオプション設定に論理和で「PIOINT_SERVER_AFFINITY_CPU0」～「PIOINT_SERVER_AFFINITY_CPU3」を指定してください。指定しない場合はCPUアフィニティ制御は無効になります。

1.4.2　NVP-LinuxのAPI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を指定してください。

❑ 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

－２０	ボード番号エラー

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

本コマンドは、StartIPコマンドで使用開始したNVP側ユーザー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。
本コマンドは1.1.2 StopIPと同じですが、NVP-Linuxで使用する場合、ボード番号は「０」を指定して下さい。

1.5 NVP-LinuxのAPI初期化サポートコマンド

1.5.1 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を指定してください。
int	Mode	モード(０を指定して下さい) ０：NVPボードリセット、再接続は行わない

❑ 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

－２０　　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

ボードをリセットし、Linuxを再起動します。
StartIPやOpenIPDevコマンド発行前でも使用可能です。
ボード番号は「０」を指定して下さい。

2. 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 －１：内部管理データエラー(PC側のみ) －１００：ドライバインタフェースエラー(PC側のみ) －１３８：画像処理システム未初期化

❑ エラーコード

なし

❑ 機能

画像処理システムのシステム管理テーブルをイニシャライズ後、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
正規化相関マスク	無効
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（ 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 －１：内部管理データエラー(PC側のみ) －１００：ドライバインタフェースエラー(PC側のみ) －１３８：画像処理システム未初期化

❑ 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	０：正常終了 －１：内部管理データエラー(PC側のみ)

❑ 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スレッド毎に管理されているためスレッド毎、スレッド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	０：正常終了 －１：内部管理データエラー(PC側のみ)

❑ 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スレッド毎に管理されているためスレッド毎、スレッド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、 　　または内部管理データエラー
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     ErrorINF0;              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スレッド毎に管理されているためスレッド毎、スレッド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： 内部管理データエラー

❑ 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スレッド毎に管理されているためスレッド毎、スレッド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

２０	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。

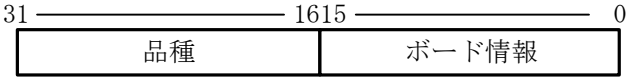
❑ 機能

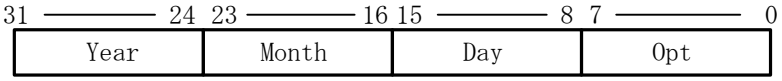
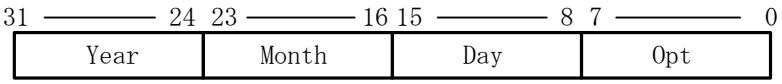
ＰＣドライバ、画像処理コマンド、ボード、ＲＯＭ等のバージョンを読み出します。

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。

<table>
<tr><th>メンバ名</th><th>内容</th></tr>
<tr><td>IPCMD_Env</td><td>画像処理コマンドの動作環境
<table><tr><th>対応数</th><th>内容</th></tr><tr><td>0</td><td>ボードが動作していない</td></tr><tr><td>1</td><td>未使用(出力されません)</td></tr><tr><td>2</td><td>NVP-Linuxシステムが動作している</td></tr></table></td></tr>
<tr><td>BOARD_Info</td><td>画像処理ボードの情報
 

品種　0x0087 ： NVP-Ax430CL
0x0085 ： NVP-Ax435CL
0x0086 ： NVP-Ax435FCL
0x0083 ： NVP-Ax430ACL</td></tr>
<tr><td>IPCMD_Version</td><td>画像処理コマンド（ボードＣＰＵ側）のバージョン情報
</td></tr>
<tr><td>IPCMD_Update</td><td>画像処理コマンド（ボードＣＰＵ側）の最終更新日
 
</td></tr>
<tr><td>ROM_Version</td><td>０（未使用）</td></tr>
<tr><td>ROM_Update</td><td>０（未使用）</td></tr>
<tr><td>※
PCDRV_Version</td><td>ＰＣドライバ（ＤＬＬ）のバージョン情報
</td></tr>
<tr><td>※
PCDRV_Update</td><td>ＰＣドライバ（ＤＬＬ）の最終更新日
 
</td></tr>
<tr><td>※
DEVDRV_Version</td><td>デバイスドライバのバージョン情報
</td></tr>
<tr><td>※
DEVDRV_OS</td><td>デバイスドライバの種類
<table><tr><th>対応数</th><th>内容</th></tr><tr><td>3</td><td>Windows WDM (デバイスドライバ)</td></tr><tr><td>4</td><td>TCP DLL(LAN通信インタフェース)</td></tr></table></td></tr>
<tr><td>Reserve</td><td>予約（２４バイト）</td></tr>
</table>

※Windows側のみ出力、ボード側では「０」出力

2.1.9 スレッド毎 マルチIMP動作環境準備（ OpenIMP ）

❑ C言語API

```
int ret = OpenIMP( 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	０：正常終了
		－２：ＯＳシステムエラー
		－２０３：画像処理スレッドシステムテーブル空きなし
		－２０５：２重オープン(OpenIMP２重発行)

❑ エラーコード

なし

❑ 機能

NVP-Ax430シリーズは、画像認識プロセッサ（ＩＭＰ）を４個搭載しており、個々に実行することが可能です。NVP-Ax430シリーズでは、１プロセス内でスレッド毎に使用するＩＭＰを指定し管理することで複数のＩＭＰを制御する仕組みになっています。複数の画像認識プロセッサを複数スレッドで同時に使用する動作モードをマルチプロセッサモード、１つの画像認識プロセッサを１スレッドのみで使用する場合の動作モードをシングルプロセッサモードを呼びます。

マルチプロセッサモードを使用する場合、OpenIMPコマンドでスレッド毎に複数のＩＭＰ動作に対する環境を準備する必要があります。複数の画像認識プロセッサを使用する場合、OpenIMPを発行することで、発行したスレッドに対しＩＭＰ管理テーブルを割り当て、複数のＩＭＰを実行する準備をします。OpenIMP発行時のデフォルトのアクティブなIMP番号は、IMP#0になっています。

なお、割り当てたＩＭＰ管理テーブルは、スレッド削除前にCloseIMPコマンドで解放して下さい。

■シングルプロセッサモードとマルチプロセッサモードの相違点

【１】システムリソース

シングルプロセッサモードでは、システムリソースは全て共通管理でしたが、マルチプロセッサモードでは、システムリソースにスレッド毎に管理されるものがあります。以下にスレッド毎に管理されるものとシステム共通に管理されるものを示します。

なお、スレッド毎に管理されるものに関しては、他のスレッドから変更ができませんので注意してください。

スレッド毎に管理	システム共通管理
・SetWindow等で設定したWindow情報 ・システムデータタイプ ・パイプラインモード ・正規化相関制御情報（正規化相関モード） ・８段２値パイプラインフィルタモード情報 ・グラフィックス情報 ・濃度変換テーブル情報 ・画像メモリランダムアクセス情報	・画像メモリ ・２値画像マッチングフィルタテンプレート ・正規化相関テンプレート ・８段２値パイプラインフィルタモード情報 ・映像入力構成情報 ・画面表示構成情報

【２】処理手順

複数の画像認識プロセッサを使用する場合、以下の要領で行って下さい。

① OpenIMPコマンドでスレッド毎のマルチＩＭＰ動作環境準備
② ActiveIMPコマンドでスレッド毎の画像認識プロセッサ番号を設定

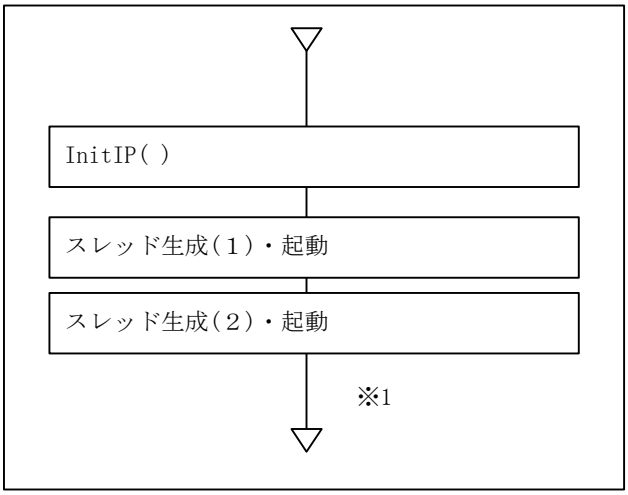
画像認識処理

③ CloseIMPコマンドでスレッド毎のマルチＩＭＰ環境解放

なお、映像入力関連のコマンドはマルチプロセッサモードから独立しており、映像入力関連のコマンドのみのスレッドの場合、OpenIMPコマンドでスレッド毎のマルチＩＭＰ動作環境準備は必要ありません。

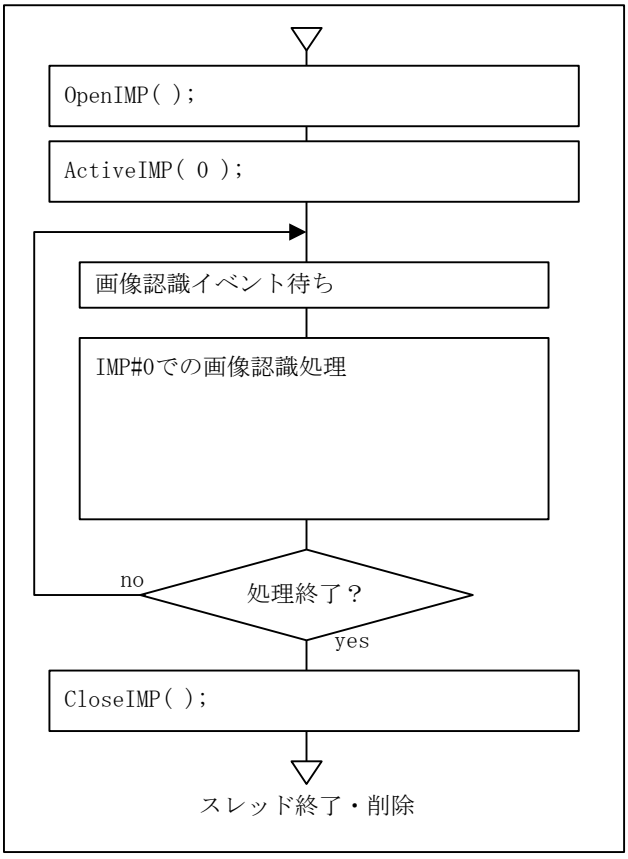
以下にマルチプロセッサモードの例を示します。

main( )

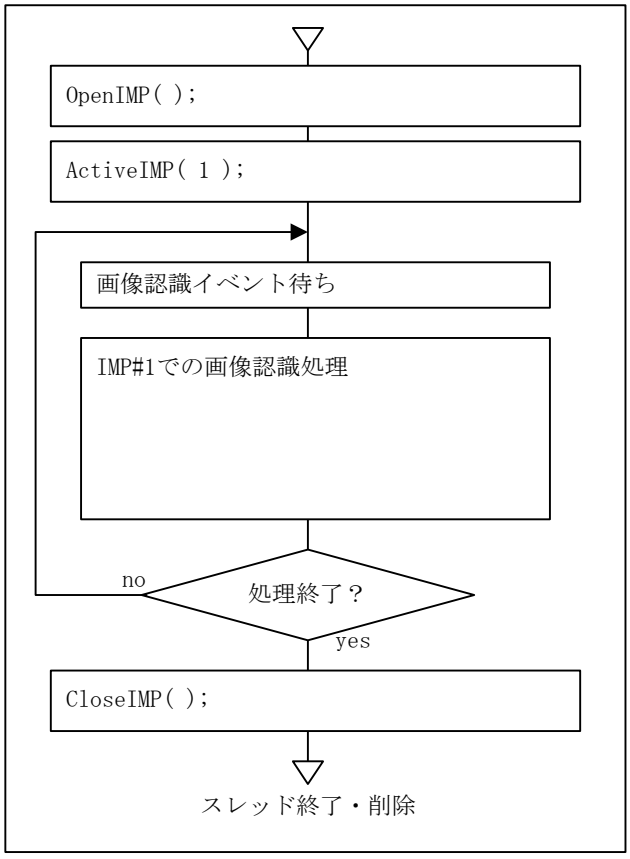


※1 アプリケーションのメインスレッド(main())は、スレッド生成、起動後、何らかの処理を継続するか、またはLinuxの「pause()」等でウェイトし、生成したスレッドが終了する前に終了しないようにしてください。

スレッド(1)



スレッド(2)



■マルチプロセッサモード動作時の注意点

マルチプロセッサモード動作時は、以下の点に注意しアプリケーション作成を行って下さい。

・対応プロセス、スレッド
マルチプロセッサモードでは、１プロセス内での複数のスレッドに対する動作に対応しています。複数プロセス間の動作には対応していませんので注意してください。

・画面番号
画面番号は、各ＩＭＰ、スレッド間で共通に管理されています。複数スレッドからの同一番号の画面番号に対する処理を実行した場合の動作の保証は致しません。必要に応じ、アプリケーション内で排他制御等の資源管理を行って下さい。また、FreeAllImg／IP_ClearAllImg は、全てのスレッドで使用している画面に影響を与えますので使用する際は注意してください。

・テンプレート番号（２値マッチングフィルタ、正規化相関）
テンプレート番号は、各ＩＭＰ、スレッド間で共通に管理されています。複数スレッドからの同一番号のテンプレート番号に対する処理を実行した場合の動作の保証は致しません。必要に応じ、アプリケーション内で排他制御等の資源管理を行って下さい。

・全画面対象の処理コマンド（FreeAllImg／IP_ClearAllImg）
「FreeAllImg／IP_ClearAllImg」は、全てのスレッドで使用している画面に影響を与えますので使用する際は注意してください。

・vpxAllocCorrTemplate／vpxFreeCorrTemplate
ＶＰ互換正規化相関コマンド「vpxAllocCorrTemplate／vpxFreeCorrTemplate」は全てのスレッドで使用しているＶＰ互換正規化相関コマンドのテンプレートに影響を与えますので使用する際は注意してください。なお、システム起動時にはテンプレートサイズ２５６個にて確保済です。２５６個で不足する場合のみ、vpxAllocCorrTemplateを発行してください。発行タイミングはInitIP直後でマルチプロセッサモード動作前に行ってください。

・映像入力
映像入力で使用するビデオポート情報は共通に管理されています。「ActiveVideoPort」コマンド以外、複数スレッドから同一ビデオポートに対する制御を行った場合の動作保証は致しません。

・映像表示
映像表示で使用する映像表示情報は共通に管理されています。複数スレッドから映像表示コマンドを同時に行った場合の動作保証は致しません。

・RAW/RGB変換
CreateConvertRAWData()、DeleteConvertRAWData()、SetRAWtoRGBGain()で生成されるリソースは、各ＩＭＰ、スレッド間で共通に管理されており、複数のスレッドで同時に実行することはできません。CreateConvertRAWData()を使用する場合は、マルチプロセッサモード動作前のアプリケーション初期化時、DeleteConvertRAWData()は、マルチプロセッサモード動作終了後のアプリケーション終了処理で使用してください。

2.1.10　スレッド毎 マルチIMP動作終了 （ CloseIMP ）

❑ C言語API

```
int ret = CloseIMP( 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	０：正常終了
		－２：ＯＳシステムエラー
		－２０４：OpenIMP未発行

❑ エラーコード

なし

❑ 機能

本コマンドにより、OpenIMP発行でスレッドに割り当てたＩＭＰ管理テーブルを解放します。OpenIMP発行したスレッドを削除する場合、必ずスレッド削除前にCloseIMPコマンドでＩＭＰ管理テーブルを解放して下さい。

2.1.11　スレッド毎 画像認識プロセッサ指定 （ ActiveIMP ）

❑ C言語API

```
int ret = ActiveIMP( int impid );
```

❑ パラメータ

int	impid	IMP番号(0～3) 0 ： IMP#0 1 ： IMP#1 2 ： IMP#2 3 ： IMP#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	０：正常終了 －２０：パラメータエラー(ＩＭＰ番号範囲外)

❑ エラーコード

なし

❑ 機能

使用する画像認識プロセッサ(IMP)を選択します。マルチプロセッサモードで動作する場合、OpenIMPコマンドでスレッド毎に複数のＩＭＰ動作に対する環境を準備した後、ActiveIMPコマンドにより、OpenIMPでスレッドに割当てた管理領域に対し画像認識プロセッサ番号(IMP_ID)を設定します。

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

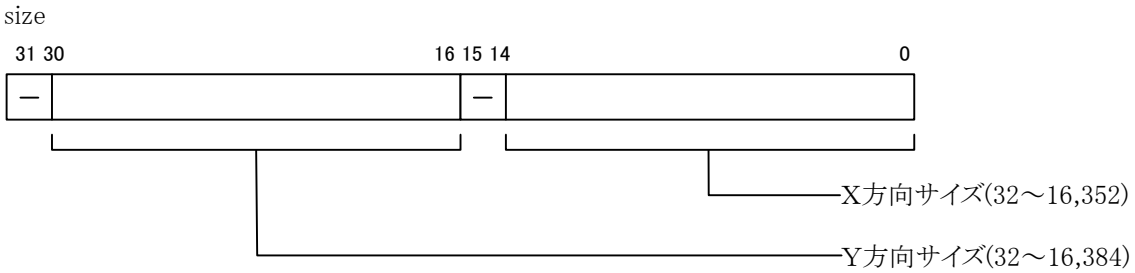
```
IMGID ImgID = AllocImg(enum ImageFrameSize size);
```

❑ パラメータ

enum ImageFrameSize　　size　　画面サイズ
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

画面サイズ(size)はX方向サイズ、Y方向サイズを任意に設定することが可能です。表中の画面サイズを使用しない場合は、下記フォーマットに従い、32byteアラインで設定してください。



❑ 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
ImgID	○	－	－	－	－	－	◆	-	-

❑ リターンパラメータ

IMGID　　ImgID　　画面番号または異常終了（－１）

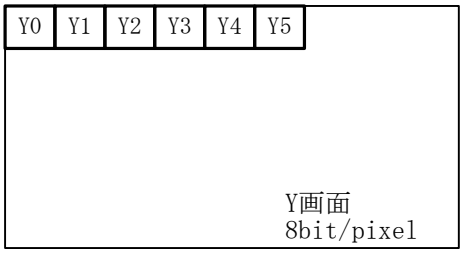
❑ エラーコード

２０	画面サイズ設定値範囲外
４６	画面ＩＤ空きなし
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

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方向サイズを必ず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のX方向サイズを必ず128の倍数とし、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)；

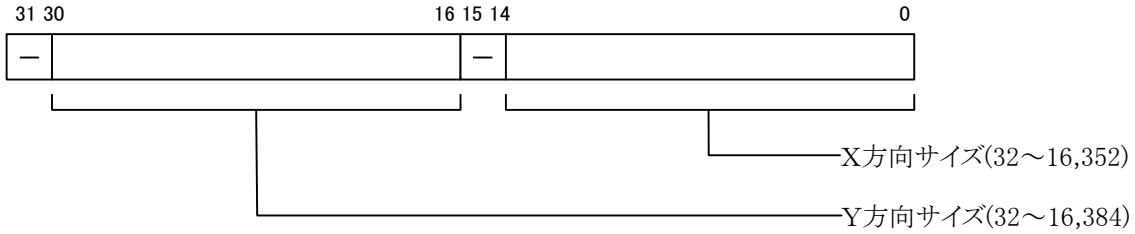
❑ パラメータ

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

画面サイズ(size)はX方向サイズ、Y方向サイズを任意に設定することが可能です。表中の画面サイズを使用しない場合は、下記フォーマットに従い、32byteアラインで設定してください。

size



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	画面番号または異常終了（－１）

❑ エラーコード

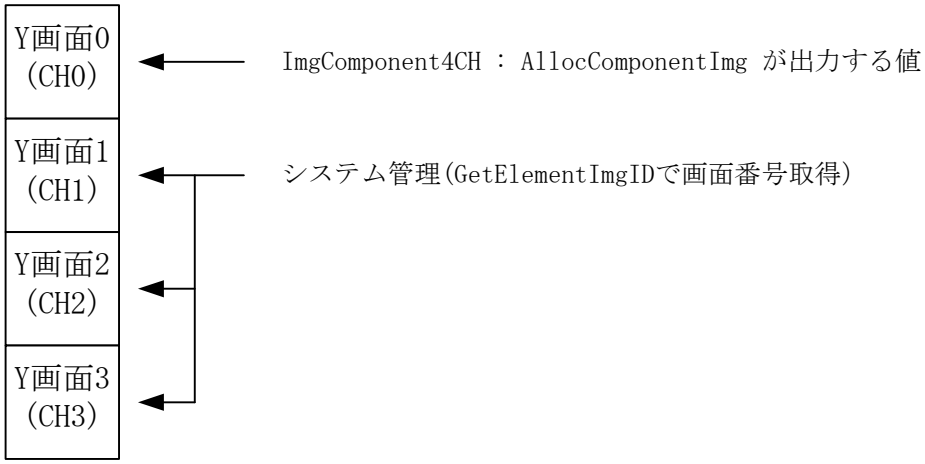
２０	画面サイズ設定値範囲外
２１	画面チャネル数設定値範囲外
４６	画面ＩＤ空きなし
４８	動的メモリ確保エラー
－	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

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(Dual)、4TAP、または、8TAPの場合、X方向サイズを必ず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のX方向サイズを必ず128の倍数とし、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
１０	画面番号が有効範囲外（不当画面番号エラー）
１３１	ＶＩＮドライバエラー
１３２	ＤＵドライバエラー
１４１	ＶＰＰドライバエラー
－	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
１０	画面番号が有効範囲外（不当画面番号エラー）
１３１	ＶＩＮドライバエラー
１３２	ＤＵドライバエラー
１４１	ＶＰＰドライバエラー
－	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（－１２８～１２７）
                                             １：符号なし（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：キャッシュパージ要求クリア
                                             １：デスティネーション画面となる場合のみ
                                                キャッシュパージ要求
                                             ２：キャッシュパージ要求
                                             ３：キャッシュパージ要求
  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ネル
} IMGTBL;
```

❑ エラーコード

１０	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	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づ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(1～3)

❑ 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

１０	指定画面空き領域、または範囲外（不当画面番号エラー）
２１	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

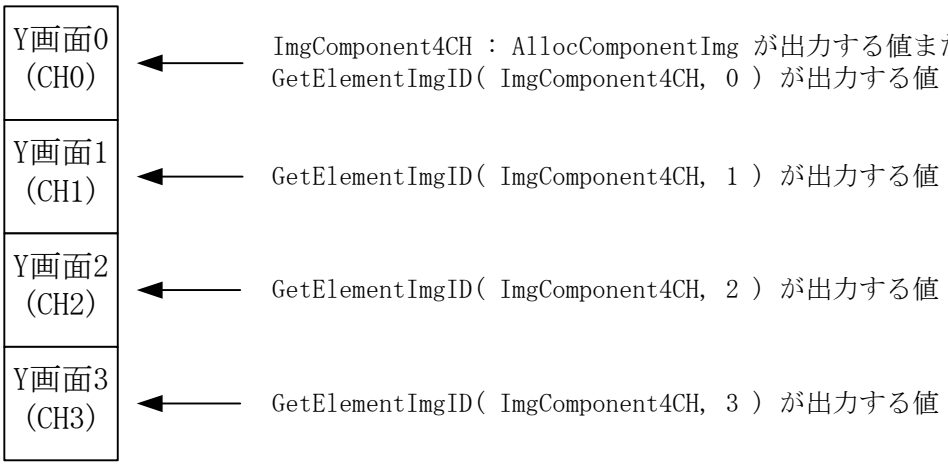
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 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レメント番号(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

１０	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	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（sx＞0）
int	sy	始点Ｙ座標（sy＞0）
int	ex	終点Ｘ座標（ex＞sx）
int	ey	終点Ｙ座標（ey＞sy）

❑ 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	ウィンドウ種類設定値範囲外
２１	始点Ｘ座標範囲外
２２	始点Ｙ座標範囲外
２３	終点Ｘ座標範囲外、始点Ｘ座標＞終点Ｘ座標と設定
２４	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。

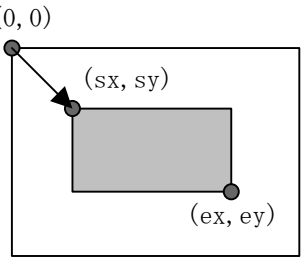
❑ 機能

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 ）（ SetAllWindow ）

❑ C言語API

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

❑ パラメータ

int	sx	始点Ｘ座標（sx＞0）
int	sy	始点Ｙ座標（sy＞0）
int	ex	終点Ｘ座標（ex＞sx）
int	ey	終点Ｙ座標（ey＞sy）

❑ 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）

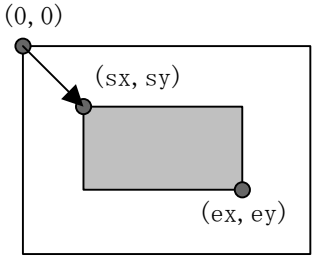
❑ エラーコード

２０	始点Ｘ座標範囲外
２１	始点Ｙ座標範囲外
２２	終点Ｘ座標範囲外、始点Ｘ座標＞終点Ｘ座標と設定
２３	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（ ResetAllWindow ）

❑ C言語API

```
int ret = ResetAllWindow();
```

❑ 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

１３３	アクティブビデオポートが未設定
１３６	システム管理データが不正(ビデオフレームサイズ未設定)
－	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

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

１３３	アクティブビデオポートが未設定
１３６	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

２０	ウィンドウ種類設定値範囲外
２１	始点Ｘ座標を返すアドレスが不正
２２	始点Ｙ座標を返すアドレスが不正
２３	終点Ｘ座標を返すアドレスが不正
２４	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

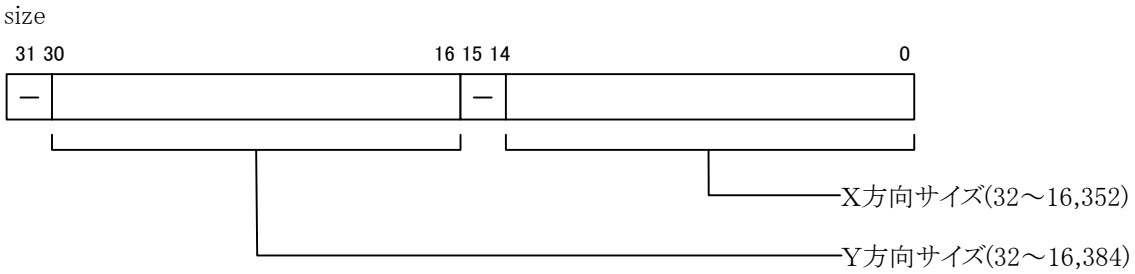
```
IMGID ImgID = AllocRGBImg(enum ImageFrameSize size);
```

❑ 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

画面サイズ(size)はX方向サイズ、Y方向サイズを任意に設定することが可能です。表中の画面サイズを使用しない場合は、下記フォーマットに従い、32byteアラインで設定してください。



❑ 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
ImgID	－	○	－	－	－	－	◆	-	-

❑ リターンパラメータ

IMGID	ImgID	画面番号または異常終了（－１）

❑ エラーコード

２０	画面サイズ設定値範囲外
４６	画面ＩＤ空きなし
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。

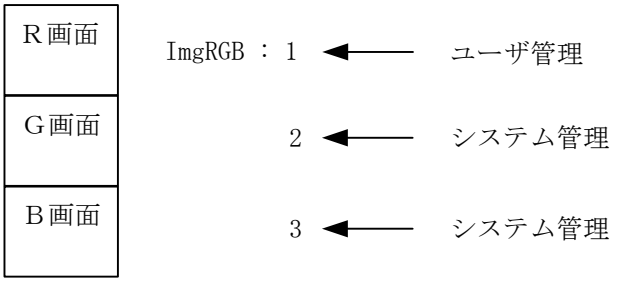
❑ 機能

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。
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、５画面分空きエリアとすることを推奨します。

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方向サイズを必ず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のX方向サイズを必ず128の倍数とし、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

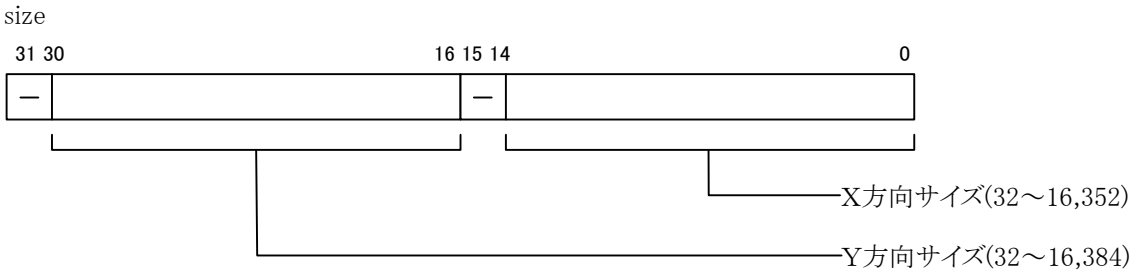
```
IMGID ImgID = AllocRGB16Img(enum ImageFrameSize ImgSize);
```

❑ 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

画面サイズ(size)はX方向サイズ、Y方向サイズを任意に設定することが可能です。表中の画面サイズを使用しない場合は、下記フォーマットに従い、32byteアラインで設定してください。



❑ 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
ImgID	－	－	－	－	－	○	◆	-	-

❑ リターンパラメータ

IMGID	ImgID	画面番号または異常終了（－１）

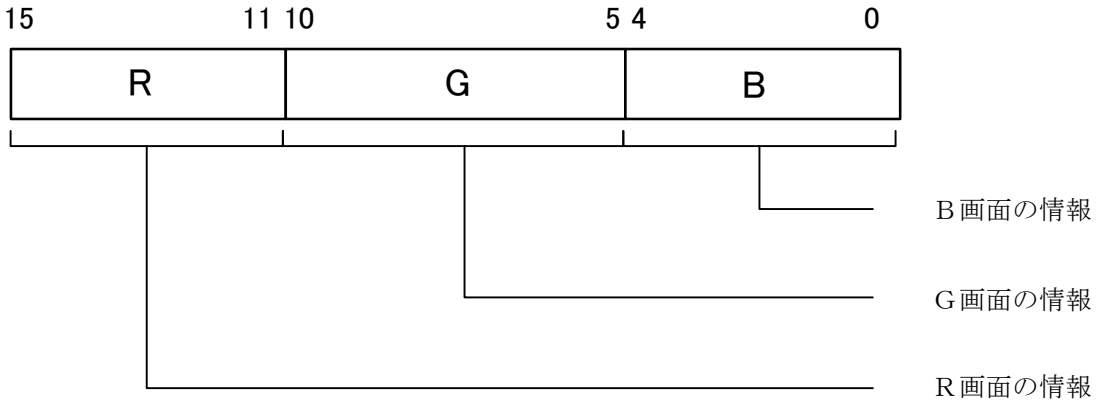
❑ エラーコード

２０	画面サイズ設定値範囲外
４６	画面ＩＤ空きなし
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

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方向サイズを必ず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のX方向サイズを必ず128の倍数とし、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	×	○	×	×	×	×	○	○	○
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

１０	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

１０	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。

2.2.18　画像メモリ領域確保(コンポーネントRGBY画面)（ AllocRGBYImg ）

❑ C言語API

```
IMGID ImgID = AllocRGBYImg(enum ImageFrameSize size, int opt);
```

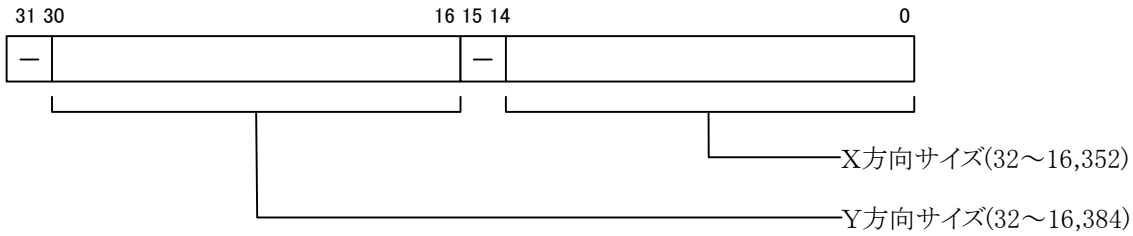
❑ パラメータ

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

画面サイズ(size)はX方向サイズ、Y方向サイズを任意に設定することが可能です。表中の画面サイズを使用しない場合は、下記フォーマットに従い、32byteアラインで設定してください。

size



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

	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　　画面番号または異常終了（－１）

❑ エラーコード

２０	画面サイズ設定値範囲外
２１	オプション設定値範囲外
４６	画面ＩＤ空きなし
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。

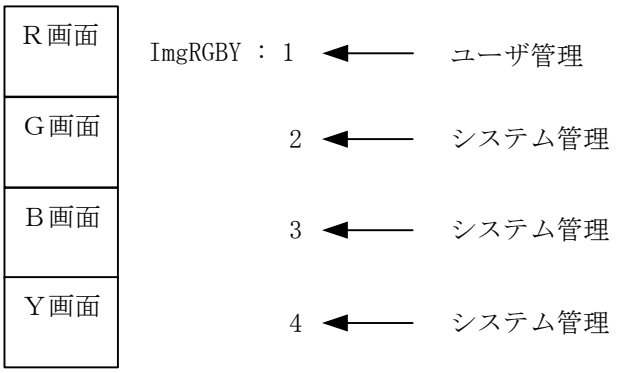
❑ 機能

画像メモリに、パラメータで指定されたサイズのＲＧＢＹ画面領域を確保します。

ＲＧＢＹ画面はＲＧＢ画面（カラー画面）の３画面とＹ画面（モノクロ画面）の１画面で１セットになります。本コマンドで確保したＲＧＢＹ画像メモリの属性はＲＧＢです。

［ＲＧＢＹ画像メモリ確保例］

```
 int  ImgRGBY;
ImgRGBY= AllocRGBYImg( devID, IMG_FS_512H_512V );
```



ＲＧＢＹ画面は、SelectCamera()でマルチコンポーネントモードのRGBLUT8BPP_BAYER24を設定した場合、GetCamera()の映像入力画面に使用します。

2.2.19　Y画面番号抽出（ GetYImgID ）

❑ C言語API

```
IMGID  YImgID = GetGYImgID(IMGID RGBYImgID);
```

❑ パラメータ

IMGID	RGBYImgID	対象画面のＲＧＢＹ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
RGBYImgID	×	○	×	×	×	×	○	○	○
YImgID	○	－	－	－	－	－	○	－	－

ただし、YImgIDはAllocRGBYImg()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	YImgID	Ｙ画面の画面ＩＤ(≧1) または、異常終了（≦０）

❑ エラーコード

１０	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

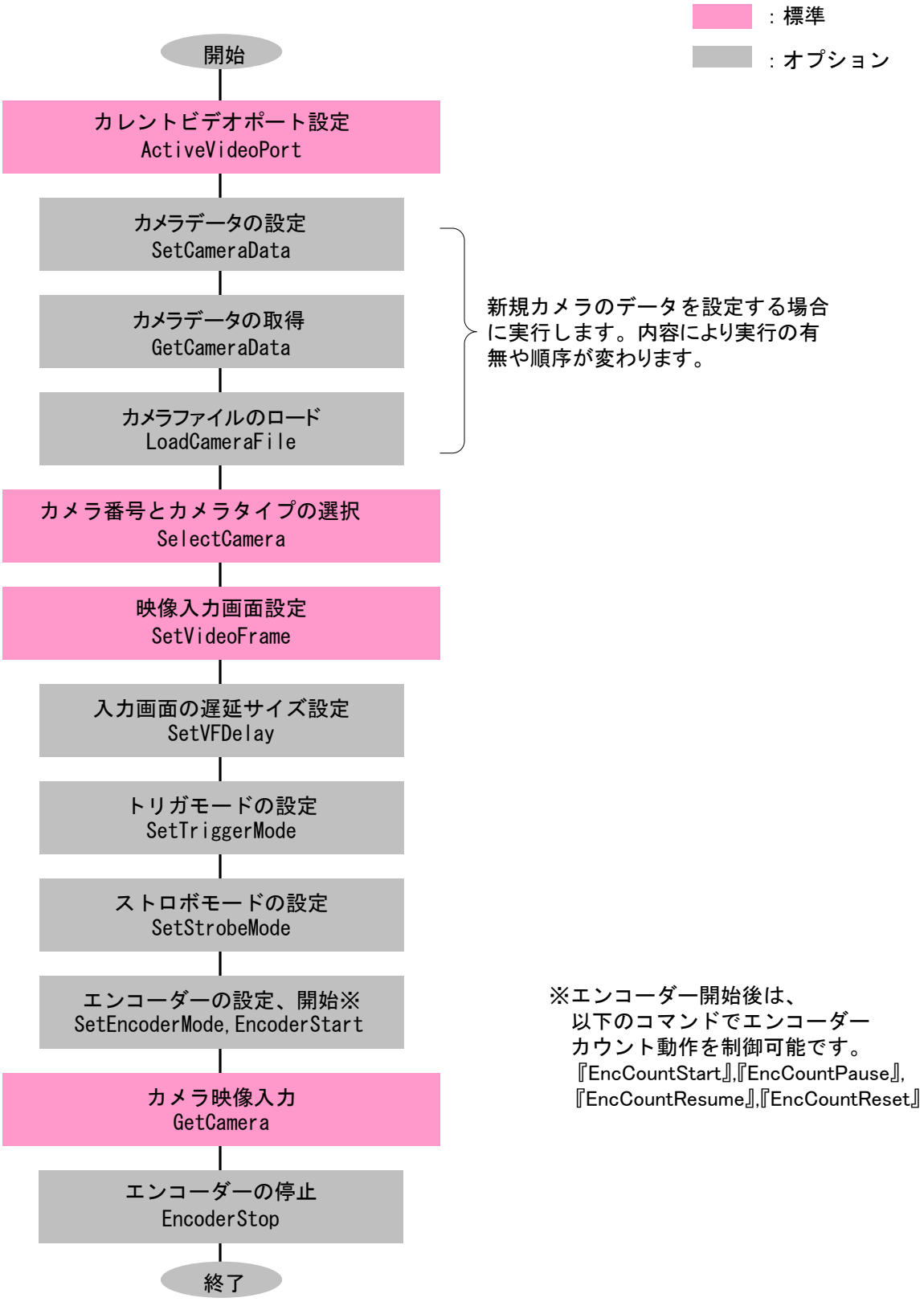
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

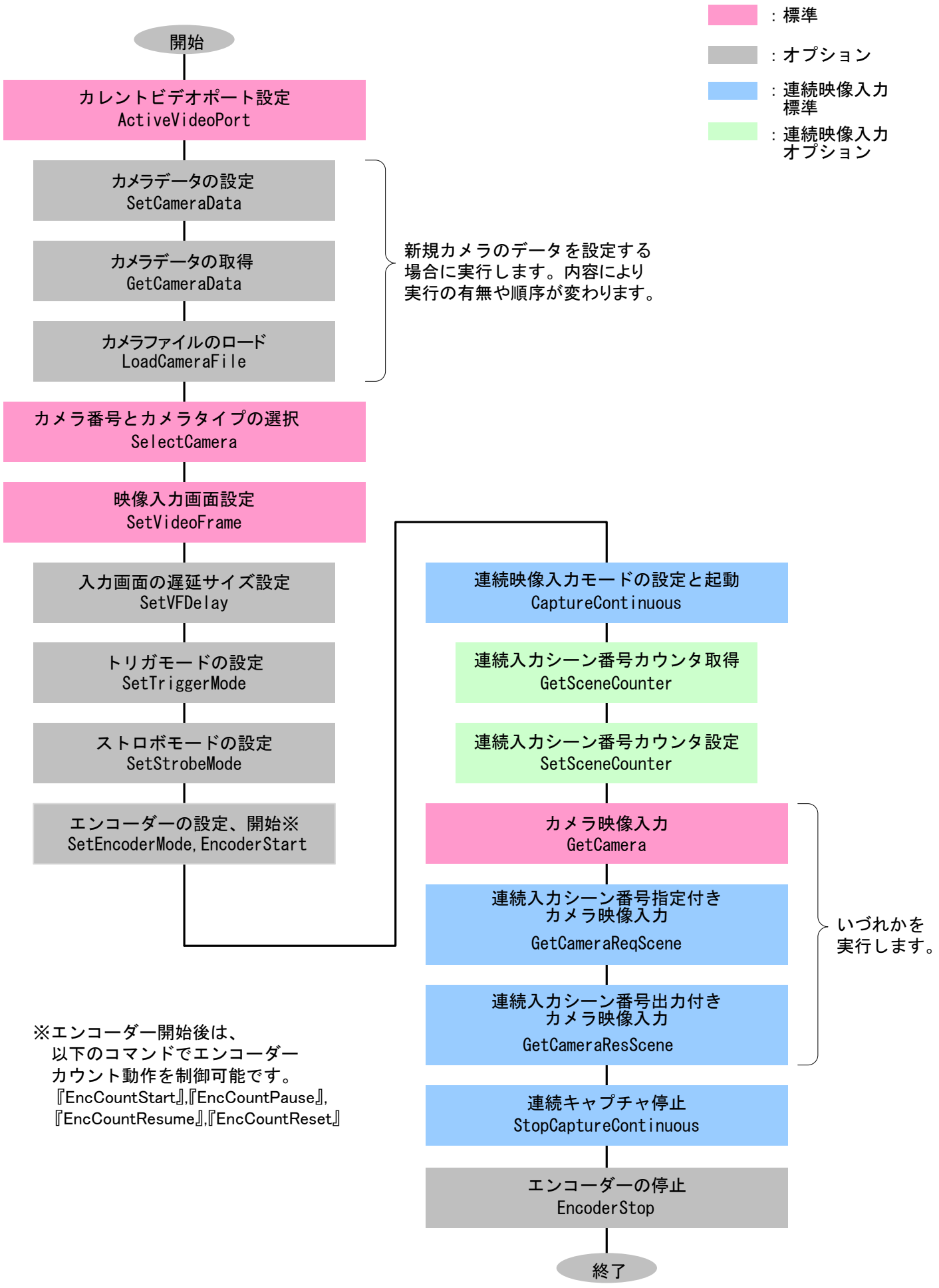
ＲＧＢＹ画面の画面番号からＹ画面の画面番号を読み出します。
マルチコンポーネント映像入力を行い、Ｙ画面の処理が必要な場合、GetYImgID()でＹ画面番号を取得し処理を行います。

2.3 映像入力

＜単一映像入力手順＞



＜連続映像入力手順＞


：標準
：オプション
：連続映像入力 標準
：連続映像入力 オプション
開始
カレントビデオポート設定
ActiveVideoPort
カメラデータの設定
SetCameraData
カメラデータの取得
GetCameraData
カメラファイルのロード
LoadCameraFile
新規カメラのデータを設定する場合に実行します。内容により実行の有無や順序が変わります。
カメラ番号とカメラタイプの選択
SelectCamera
映像入力画面設定
SetVideoFrame
入力画面の遅延サイズ設定
SetVFDelay
トリガモードの設定
SetTriggerMode
ストロボモードの設定
SetStrobeMode
エンコーダーの設定、開始※
SetEncoderMode, EncoderStart
※エンコーダー開始後は、
以下のコマンドでエンコーダー
カウント動作を制御可能です。
『EncCountStart』,『EncCountPause』,
『EncCountResume』,『EncCountReset』
連続映像入力モードの設定と起動
CaptureContinuous
連続入力シーン番号カウンタ取得
GetSceneCounter
連続入力シーン番号カウンタ設定
SetSceneCounter
カメラ映像入力
GetCamera
連続入力シーン番号指定付き
カメラ映像入力
GetCameraReqScene
連続入力シーン番号出力付き
カメラ映像入力
GetCameraResScene
いづれかを実行します。
連続キャプチャ停止
StopCaptureContinuous
エンコーダーの停止
EncoderStop
終了

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

２０　　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

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。

ボードによってカメラチャネル数が異なります。ＮＶＰ－Ａｘ４３５ＣＬは、ビデオポート＃０～＃３、その他のボードは、＃０～＃１で指定してください。

2.3.2 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_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）

マルチコンポーネントモード	対応定数	内容
CAMERA	0x00000000	カメラ画像
CAMERA_RGBLUT8	0x00000100	RAWカメラの場合： カメラ画像(8bpp)＋RGB LUT変換データ(8bpp) RGBカメラの場合： カメラ画像(24bpp)＋RGB LUT変換データ(8bpp)
CAMERA_BAYER24	0x00000200	カメラ画像(8bpp)＋RAW-RGB変換データ(24bpp)
BAYER24_RGBLUT8BPP	0x00000300	RAW-RGB変換データ(24bpp)＋RGB-LUT変換データ(8bpp)
BAYER24	0x00000400	RAW-RGB変換データ(24bpp)

enum CameraType type　　カメラタイプ

カメラタイプNo.	対応定数	内容　※1	備考　※2
SONY_XCL_SG510	2	モノクロ／ＲＡＷ	8bit, 2TAP(dual), 85MHz, 2448x2048
NED_RMSL4K100CL	6	ラインカメラ、モノクロ	8bit, 8TAP(quad), 40MHz, 4096
NED_RMSL8K76CL	7	ラインカメラ、モノクロ	8bit, 8TAP(quad), 40MHz, 8192
TAKEX_TL_4096ACL	8	ラインカメラ、モノクロ	8bit, 8TAP(quad), 50MHz, 4096
TAKEX_TL_8K35ACL	9	ラインカメラ、モノクロ	8bit, 8TAP(quad), 50MHz, 8192
TAKEX_TL_16KACL	10	ラインカメラ、モノクロ	8bit, 8TAP(quad), 40MHz, 16384
-	11～127	リザーブ	
-	128～255 (0x80～0xFF)	ユーザ定義カメラ	

※1　”モノクロ／ＲＡＷ”と記載しているカメラタイプは、モノクロ入力カメラとＲＡＷ入力カメラで共通のカメラタイプになります。例えば、XCL_SG510Cカメラ(ＲＡＷ)を使用する場合はSONY_XCL_SG510を選択してください。

※2　カメラのモード（画素フォーマット,ＴＡＰ,周波数,有効画素）を示します。カメラの設定をこれに合わせてください。 ()は、カメラデータのＶＩＮモード設定値です。

❑ 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

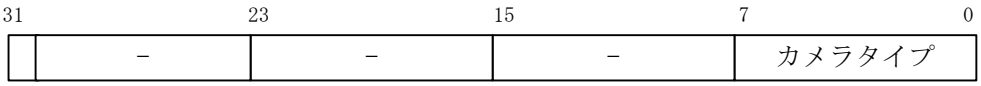
２	ＯＳシステムエラー
７	指定したカメラのカメラデータ設定範囲外 または指定カメラでのマルチコンポーネントモード設定範囲外
２０	カメラポート番号設定値範囲外
２１	カメラタイプ設定値範囲外
１３１	ＶＩＮドライバでのエラー
１３２	ＤＵドライバでのエラー
１３３	アクティブビデオポートが未設定
１３８	連続映像入力動作時（CaptureContinuous）
－	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

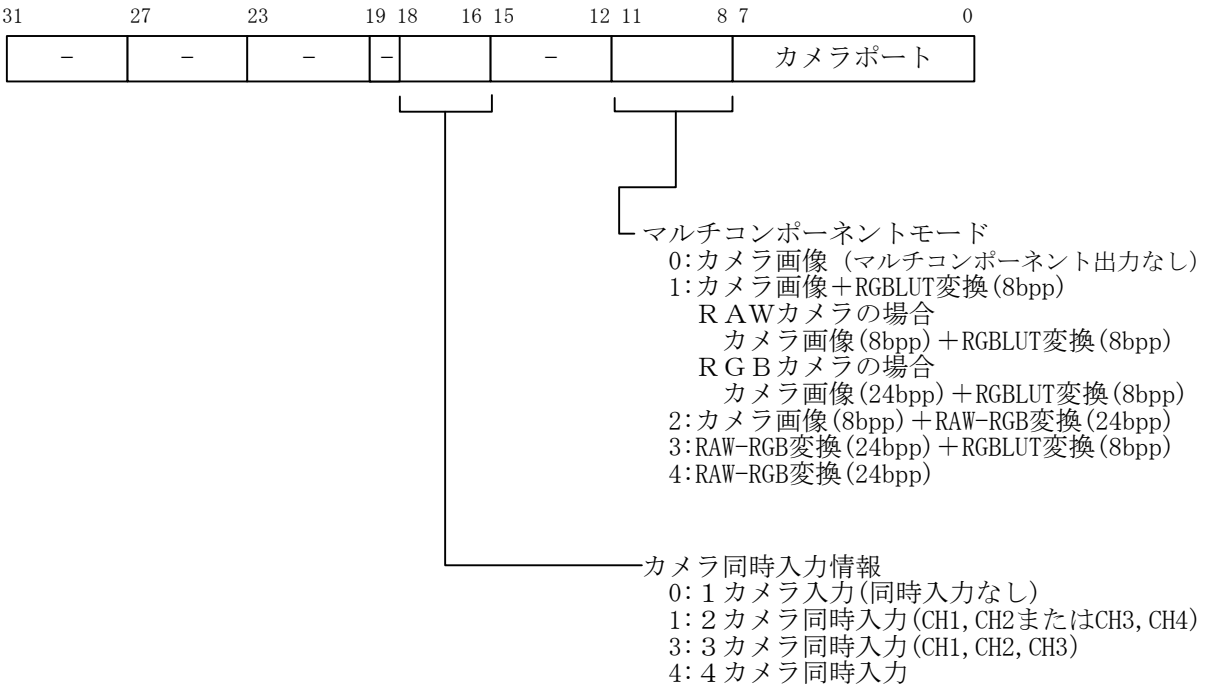
❑ 機能

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bit8～bit11にマルチコンポーネントモード、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、マルチコンポーネントモードと同時入力情報のＯＲをとり、idに設定してください。
同時入力は、同一カメラおよび同一カメラ設定で使用可能です。
登録済みのサポートカメラ以外のカメラタイプ 番号は使用できませんので指定しないでください。但し、１２８～２５５は、ユーザー定義カメラをとして登録し使用できます。
ユーザー定義カメラは、『LoadCameraFile』『SetCameraData』で登録します。

type



id



type, idのその他ビットは０を指定してください。

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	

選択したカメラを複数同時に映像入力する場合はカメラ同時入力情報を指定してください。また、同時取込み用画像メモリ領域確保コマンド『AllocComponentImg』で確保した画像メモリで映像入力をしてください。ＲＡＷカメラ、モノクロカメラで同時入力が可能です。

マルチコンポーネントモードはＣＨ１,ＣＨ２で使用可能です。カメラ画像の他にＲＧＢ－ＬＵＴ変換画像、ＲＡＷ－ＲＧＢ変換画像を同時に出力することができます。ＲＧＢカメラは、1:カメラ画像(24bpp)＋ＲＧＢ－ＬＵＴ変換(8bpp)のみ使用可能です。また、マルチコンポーネントモード使用時（0：カメラ画像以外）は同時入力はできません。
ＲＧＢ－ＬＵＴ変換を行う場合は、『SetVideoLUTMode』も合わせて設定してください。

●映像入力できる画像メモリについて

下記に示す画像メモリコマンドで映像入力の画面を確保してください。
画像メモリのサイズ制限があります。『SetCameraData』のＶＩＮモード詳細をご確認下さい。

ＲＡＷカメラ

マルチコンポーネントモード	画像メモリ領域確保コマンド
0:カメラ画像(8bpp)	AllocImg()
1:カメラ画像(8bpp)＋RGBLUT変換(8bpp)	AllocComponentImg() :画面チャネル数=2※1
2:カメラ画像(8bpp)＋RAW-RGB変換(24bpp)	AllocRGBYImg() ※2
3:RAW-RGB変換(24bpp)＋RGBLUT変換(8bpp)	AllocRGBYImg() ※3
4:RAW-RGB変換(24bpp)	AllocRGBImg()

※1 カメラ画像(8bpp)はＹ画面０、ＲＧＢ－ＬＵＴ変換データ(8bpp)はＹ画面１に格納されます。
※2 ＲＡＷ－ＲＧＢ変換データ(24bpp) はR,G,B画面、カメラ画像(8bpp)はY画面に格納されます。
※3 ＲＡＷ－ＲＧＢ変換データ(24bpp) はR,G,B画面、ＲＧＢ－ＬＵＴ変換データ(8bpp)はＹ画面に格納されます。

ＲＧＢカメラ

マルチコンポーネントモード	画像メモリ領域確保コマンド※
0:カメラ画像(24bpp)	AllocRGBImg()
1:カメラ画像(24bpp)＋RGBLUT変換(8bpp)	AllocRGBYImg()※1

※1 カメラ画像(24bpp)はＲ、Ｇ、Ｂ画面、ＲＧＢ－ＬＵＴ変換データ(8bpp)はＹ画面に格納されます。

●接続可能カメラＣＨおよび同時入力

登録済みのカメラタイプの接続可能ＣＨを以下に示します。
接続可能ＣＨでの組み合わせで同時入力が可能です。

〇:可、×:不可

カメラタイプNo.	対応定数	CH1	CH2	CH3	CH4	備考
SONY_XCL_SG510	2	**〇※1**	**〇※2**	×	×	
NED_RMSL4K100CL	6	**〇※3**	×	×	×	
NED_RMSL8K76CL	7	**〇※3**	×	×	×	
TAKEX_TL_4096ACL	8	**〇※3**	×	×	×	
TAKEX_TL_8K35ACL	9	**〇※3**	×	×	×	
TAKEX_TL_16KACL	10	**〇※3**	×	×	×	

※1 ＣＨ１を使用した場合、ＣＨ３は使用できません。
※2 ＣＨ２を使用した場合、ＣＨ４は使用できません。
※3 ＣＨ３を使用した場合、ＣＨ２、ＣＨ３、ＣＨ４は使用はできません。
NVP-Ax435FCLのＣＨ１でのみ使用可能です。

上記は、マルチコンポーネントモードをしない（0：カメラ画像）場合です。詳細は次ページを参照してください。マルチコンポーネントモードは、エリアカメラでのみ使用可能です。ラインカメラでの使用はできません。

●カメラの接続制限

接続制限は、以下の接続パターンＡ～Ｄの４パターンいずれかになります。

・接続可能CHと接続制限

接続 パターン	接続可能CH	他CHの接続制限
A	CH1,CH2※1	なし
B	CH1,CH2	CH1での使用でCH3が、CH2での使用でCH4が使用できません。
C	CH1	CH3,CH4は使用できません。
D	CH1	CH2,CH3,CH4は使用できません。

※1 マルチコンポーネントモードを使用しない（0:カメラ画像）場合はＣＨ３，ＣＨ４も接続可能です。

以下に各カメラにおける接続制限を示します。カメラデータ（ＴＡＰ、周波数など）、マルチコンポーネントモードなどで異なります。（表記記載がないＶＩＮ拡張モードは０です）ＶＩＮモード、ＶＩＮ拡張モード、周波数はカメラデータに設定されています。『SetCameraData』のＶＩＮモード詳細を参照してください。

ＲＡＷカメラ 1TAP

ＶＩＮ モード	周波数	マルチコンポーネントモード	接続 パターン
0:single	1:50MHz以下	0:カメラ画像(8bpp)	A
		1:カメラ画像(8bpp)＋RGBLUT変換(8bpp)	非対応 (接続不可)
		2:カメラ画像(8bpp)＋RAW-RGB変換(24bpp)	
		3:RAW-RGB変換(24bpp)＋RGBLUT変換(8bpp)	
		4:RAW-RGB変換(24bpp)	
	0:85MHz以下	0:カメラ画像(8bpp)	A
		1:カメラ画像(8bpp)＋RGBLUT変換(8bpp)	B
		2:カメラ画像(8bpp)＋RAW-RGB変換(24bpp)	D
		3:RAW-RGB変換(24bpp)＋RGBLUT変換(8bpp)	D
		4:RAW-RGB変換(24bpp)	C

ＲＡＷカメラ 2TAP

ＶＩＮ モード	周波数	マルチコンポーネントモード	接続 パターン
0:single	1:50MHz以下	0:カメラ画像(8bpp)	A
		1:カメラ画像(8bpp)＋RGBLUT変換(8bpp)	B
		2:カメラ画像(8bpp)＋RAW-RGB変換(24bpp)	D
		3:RAW-RGB変換(24bpp)＋RGBLUT変換(8bpp)	D
		4:RAW-RGB変換(24bpp)	C
1:dual	0:85MHz以下	0:カメラ画像(8bpp)	B
		1:カメラ画像(8bpp)＋RGBLUT変換(8bpp)	非対応 (接続不可)
		2:カメラ画像(8bpp)＋RAW-RGB変換(24bpp)	
		3:RAW-RGB変換(24bpp)＋RGBLUT変換(8bpp)	
		4:RAW-RGB変換(24bpp)	

ＲＡＷカメラ 3TAP

ＶＩＮ モード	周波数	マルチコンポーネントモード	接続 パターン
1:dual	1:50MHz以下	0:カメラ画像(8bpp)	B
		1:カメラ画像(8bpp)＋RGBLUT変換(8bpp)	非対応 (接続不可)
		2:カメラ画像(8bpp)＋RAW-RGB変換(24bpp)	
		3:RAW-RGB変換(24bpp)＋RGBLUT変換(8bpp)	
		4:RAW-RGB変換(24bpp)	

ＲＡＷカメラ 4TAP

ＶＩＮ モード	周波数	マルチコンポーネントモード	接続 パターン
3:quad	0:85MHz以下	0:カメラ画像(8bpp)	D
		1:カメラ画像(8bpp)＋RGBLUT変換(8bpp)	非対応 (接続不可)
		2:カメラ画像(8bpp)＋RAW-RGB変換(24bpp)	
		3:RAW-RGB変換(24bpp)＋RGBLUT変換(8bpp)	
		4:RAW-RGB変換(24bpp)	

ＲＡＷカメラ 8TAP

ＶＩＮ モード	周波数	マルチコンポーネントモード	接続 パターン
3:quad	1:50MHz以下	0:カメラ画像(8bpp)	D
		1:カメラ画像(8bpp)＋RGBLUT変換(8bpp)	非対応 (接続不可)
		2:カメラ画像(8bpp)＋RAW-RGB変換(24bpp)	
		3:RAW-RGB変換(24bpp)＋RGBLUT変換(8bpp)	
		4:RAW-RGB変換(24bpp)	

ＲＧＢカメラ

ＶＩＮ モード	周波数	マルチコンポーネントモード	接続 パターン
1:dual	1:50MHz以下	0:カメラ画像(24bpp)	B
		1:カメラ画像(24bpp)＋RGBLUT変換(8bpp)	B
		2:カメラ画像(8bpp)＋RAW-RGB変換(24bpp)	非対応 (接続不可)
		3:RAW-RGB変換(24bpp)＋RGBLUT変換(8bpp)	
		4:RAW-RGB変換(24bpp)	
2:triple	0:85MHz以下	0:カメラ画像(24bpp)	C
		1:カメラ画像(24bpp)＋RGBLUT変換(8bpp)	D
		2:カメラ画像(8bpp)＋RAW-RGB変換(24bpp)	非対応 (接続不可)
		3:RAW-RGB変換(24bpp)＋RGBLUT変換(8bpp)	
		4:RAW-RGB変換(24bpp)	

2.3.3 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_512H_240V	0x00F00200	512	240	
VIDEO_FS_640H_240V	0x00F00280	640	240	
VIDEO_FS_256H_480V	0x01E00100	256	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

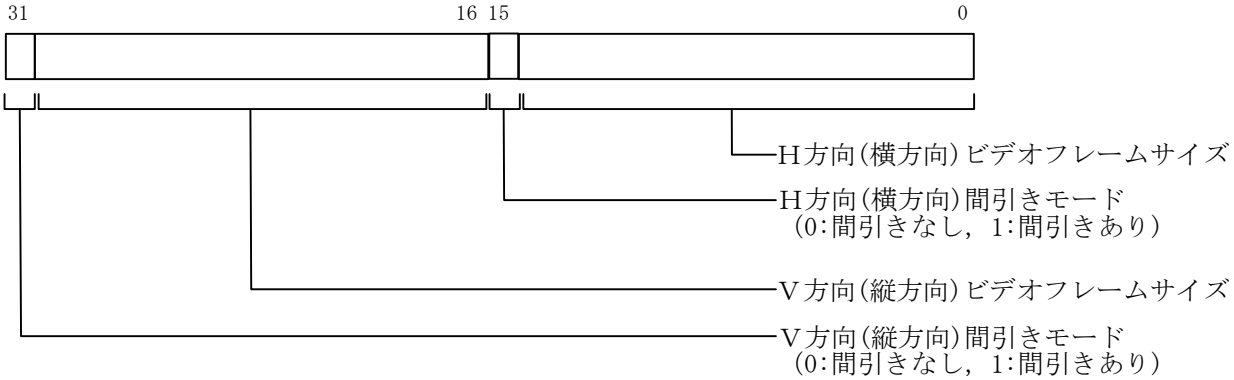
２	ＯＳシステムエラー
２０	走査方式設定値範囲外
２１	映像入力サイズ設定値範囲外
１３１	ＶＩＮドライバでのエラー
１３３	アクティブビデオポートまたは、連続映像入力時(CaptureContinuous)、未設定
１３８	カメラ選択未実施
－	エラーリセットコマンド（ClearIPError）未発行

※ 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。

size



● 走査方式について

カメラリンク対応カメラの走査方式はNONINTERLACEで使用して下さい。

● ビデオフレームサイズ、画像メモリサイズについて

カメラタイプ別の設定可能な最大ビデオフレームサイズを以下に示します。また、備考に、ＶＩＮモード、ＶＩＮ拡張モード値を示します。

カメラタイプNo.	対応定数	ビデオフレームサイズ 最大 水平×垂直	備考 (VINモード、 VIN拡張モード)
SONY_XCL_SG510	2	2432×2048	(dual,垂直拡張)
NED_RMSL4K100CL	6	3968×16384	(quad,垂直拡張)
NED_RMSL8K76CL	7	8192(※１)×4096	(quad,水平拡張)
TAKEX_TL_4096ACL	8	3968×16384	(quad,垂直拡張)
TAKEX_TL_8K35ACL	9	8192(※１)×4096	(quad,水平拡張)
TAKEX_TL_16KACL	10	16352(※１)×4096	(quad,水平拡張)

※１ 水平方向は記載値のみ設定可能です。

ＶＩＮモード、ＶＩＮ拡張モードによってビデオフレームサイズ、画像メモリサイズに制限があります。詳細は、『SetCameraData』コマンドのＶＩＮモード詳細を参照してください。

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	○	×	×	×	×	×	◆	-	-
	×	○	×	×	×	×	◆	-	-

❑ 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	ＶＩＮ入力タイムアウト
１０	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）、
１３１	ＶＩＮドライバでのエラー
１３２	ＤＵドライバでのエラー
１３３	アクティブビデオポートが未設定
１３５	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
１３６	システム管理データが不正
１３７	画面管理データが不正
１３８	映像入力画面設定コマンド（SetVideoFrame）未発行
１３９	映像入力タイムアウト時間設定値不正
－	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

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は入力画面にY画面を指定してください。
ＲＧＢカラーカメラの映像入力は入力画面にＲＧＢ画面を指定してください。
SelectCamera()でマルチコンポーネントモードを設定した場合は次の入力画面を指定してください。

CAMERAモード	:Y画面
CAMERA_RGBLUT8(RAWカメラ)	:コンポーネントＹ画面
CAMERA_RGBLUT8(RGBカメラ)	:コンポーネントＲＧＢＹ画面
BAYER24モード	:コンポーネントＲＧＢ画面 (AllocRGBImg()で画面確保)
RGBLUT8BPP_BAYER24,CAMERA_BAYER24モード	:コンポーネントＲＧＢＹ画面 (AllocRGBYImg()で画面確保)

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）
・１２８画素アライン
詳細はSetVideoFreame()を参照してください。

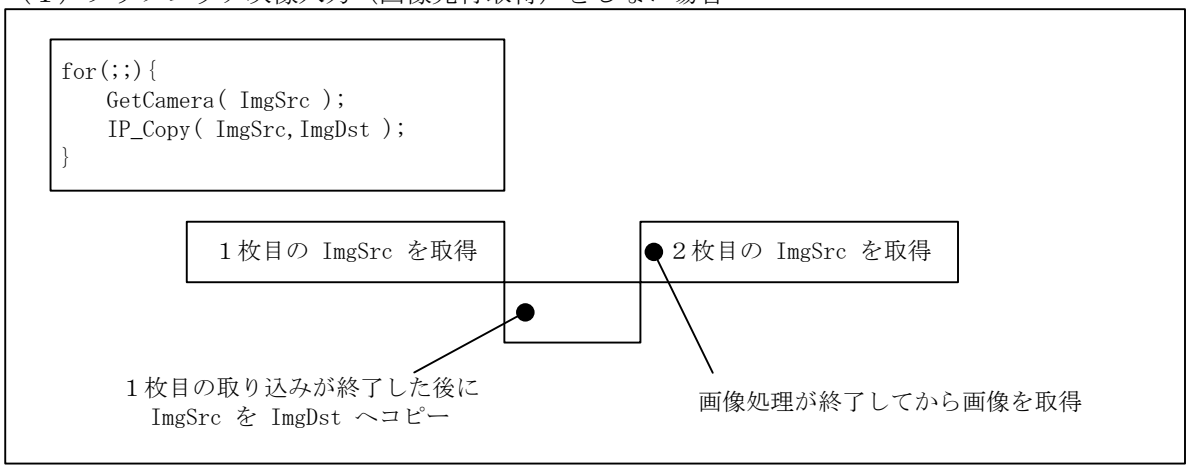
本ライブラリの映像入力コマンドでは、単一キャプチャモードと連続キャプチャモードをサポートしています。

● 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。

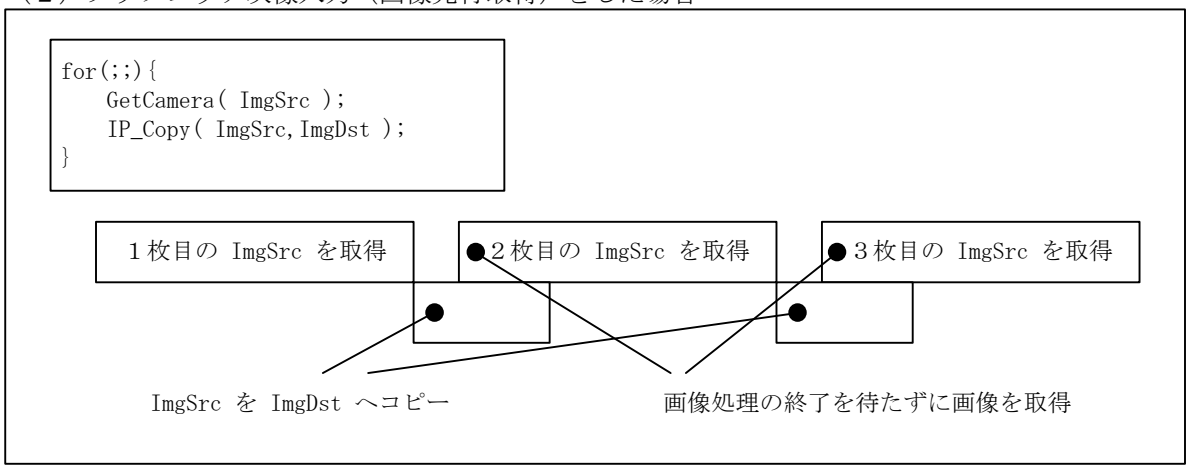
● 連続キャプチャモード：
『CaptureContinuous()』コマンドで指定した画像メモリ（３画面）に常に映像入力が行われており、『GetCamera()』コマンド発行時に取込が終了した最新のフレームの映像が取り込まれます。常に映像を先行入力しているため、『GetCamera()』コマンド発行から映像入力取得までの時間は０ｍｓ～カメラ毎のフレームレートで映像入力します。連続で映像入力を行った場合はカメラ毎のフレームレートで映像入力します。
また、連続キャプチャモードを使用し、プリフェッチ映像入力が可能です。
詳細については、『CaptureContinuous()』コマンドを参照して下さい。

＜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



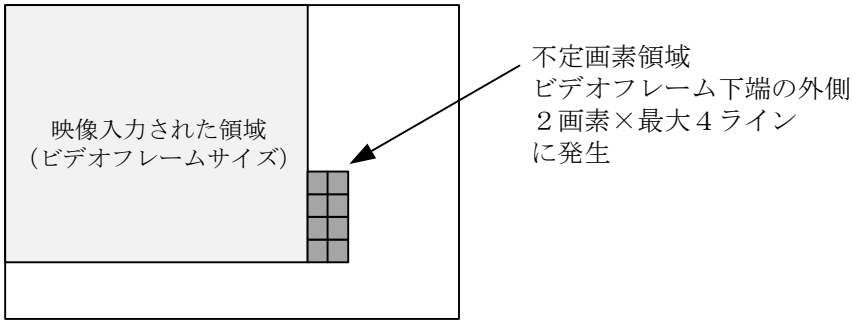
（２）プリフェッチ映像入力（画像先行取得）をした場合



● 映像入力時の不定画素

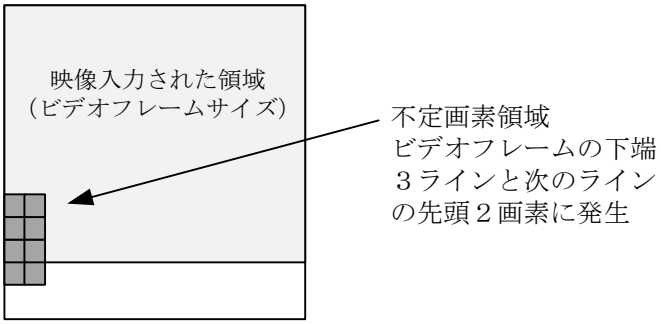
映像入力した画像メモリにおいて不定画素となる領域が発生します。不定画素となる領域は、以下の通りです。

（1）画像メモリの幅が、ビデオフレームの幅よりも大きい場合は、以下の領域になります。



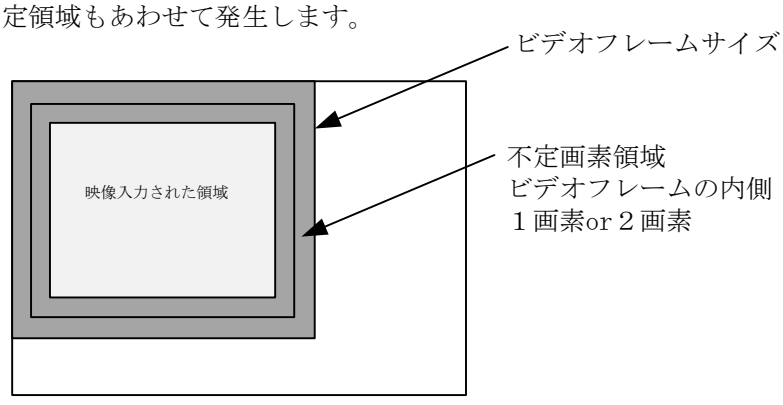
画像メモリ

（2）画像メモリの幅＝ビデオフレームの幅の場合ビデオフレームの下端３ラインと次のラインの先頭２画素（４ラインに発生）が不定となります。



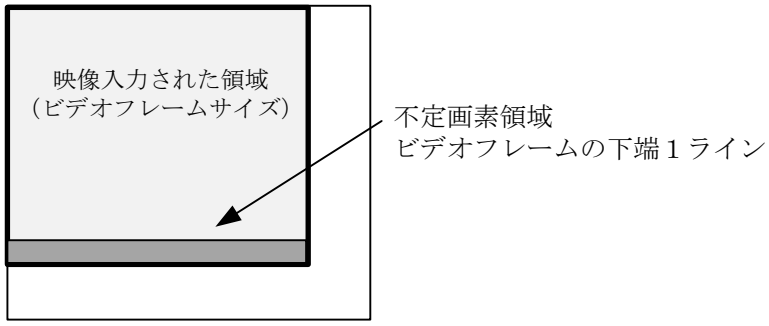
画像メモリ

（3）ＲＡＷカメラでマルチコンポーネントモードを使用する場合、出力画像はビデオフレームの内側周辺n画素が不定値になります。
（ＶＩＮモードがsingleの場合周辺1画素,dualの場合周辺２画素）
（1）（2）の不定領域もあわせて発生します。



画像メモリ

（4）ラインカメラを水平拡張モードで使用する場合は（1）（2）に合わせて下端１ライン
が不定となります。
カメラによっては先頭数ラインが不定となる場合があります（カメラの仕様に従います）。



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）

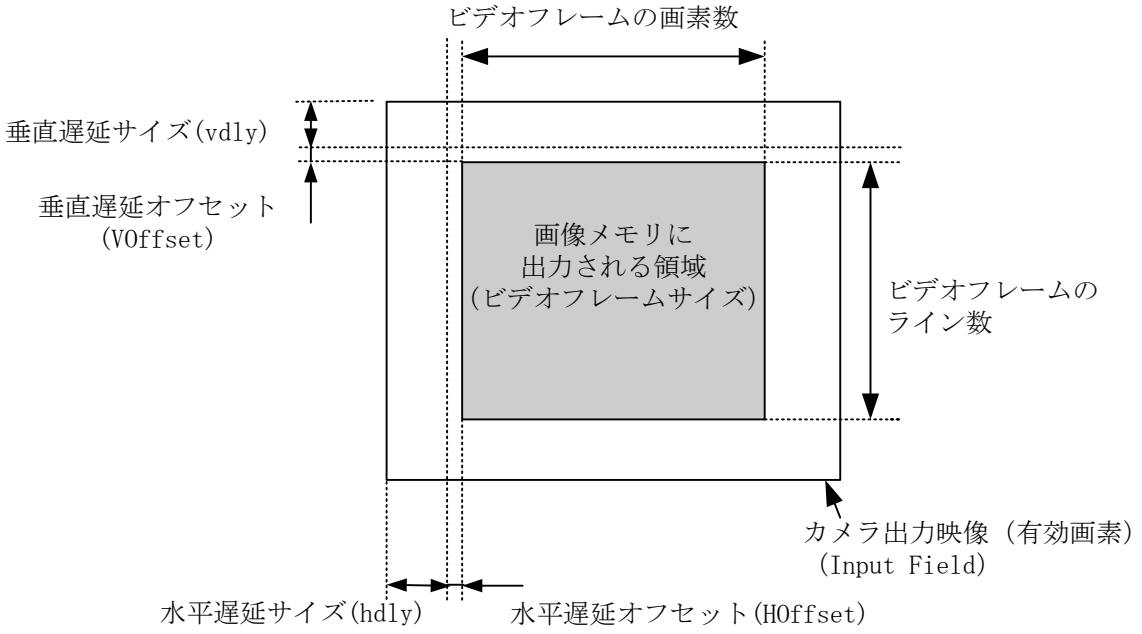
❑ エラーコード

２０	走査方式設定値範囲外
２１	映像入力サイズ設定値範囲外
１３３	アクティブビデオポートが未設定
１３６	システム管理データが不正
１３８	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

映像入力の遅延サイズを設定します。
下図に示す通り、遅延サイズ(hdly,vdly)＋遅延サイズオフセット(Hoffset,Voffset)の位置からビデオフレームサイズの領域を取り出し画像メモリに出力します。
遅延サイズ(hdly,vdly)は、『SetVideoFrame』コマンド実行時にカメラデータの有効画素サイズとビデオフレームサイズからカメラ出力映像の中央部分を画像メモリに出力するように標準値が設定されます。
本コマンドで遅延サイズオフセット(Hoffset,Voffset)を設定することで画像メモリに出力する領域を調整できます。本コマンドを設定しない場合は、遅延サイズオフセットは０です。
ラインカメラの場合は、垂直遅延サイズvdly = 0となります。垂直遅延オフセットVoffsetも０固定としてください。
ビデオフレームがカメラ出力映像からはみ出る遅延量の場合はエラーとなります。



● 標準値の計算

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

※ラインカメラの場合は、vdlyは０固定です。

● 遅延量制限

カメラタイプ別の遅延量制限を以下に示します。設定値がこれを満たさない場合調整されます。

カメラタイプNo.	対応定数	水平遅延量	垂直遅延量	備考 (VINモード、 VIN拡張モード)
SONY_XCL_SG510	2	2画素単位	2ライン単位	(dual,垂直拡張)
NED_RMSL4K100CL	6	2画素単位	0固定	(quad,垂直拡張)
NED_RMSL8K76CL	7	0固定	0固定	(quad,水平拡張)
TAKEX_TL_4096ACL	8	2画素単位	0固定	(quad,垂直拡張)
TAKEX_TL_8K35ACL	9	0固定	0固定	(quad,水平拡張)
TAKEX_TL_16KACL	10	0固定	0固定	(quad,水平拡張)

ex. カメラタイプがSONY_XCL_SG510（有効画素サイズが2448 x 2048、ＶＩＮモード = dual）で、ビデオフレームサイズを2432 x 1024とした場合、

遅延横サイズ(hdly) = (2448 - 2432) / 2 = 8

遅延縦サイズ(vdly) = (2048 - 1024) / 2 = 512

となります。（標準値）

この時、例えば垂直遅延オフセット(Voffset)に対し垂直遅延量は以下のようになります。

VOffset = -2 → 垂直遅延量=512-2 = 510

VOffset = -1 → 垂直遅延量=512-1 = 511→510（２ライン単位に調整）

VOffset = 0 → 垂直遅延量=512+0 = 512

VOffset = 1 → 垂直遅延量=512+1 = 513→512（２ライン単位に調整）

VOffset = 2 → 垂直遅延量=512+2 = 514

Hoffsetの場合は、±標準値の範囲で設定可能です。カメラ出力映像からはみ出るためエラーとなります。

Voffsetの場合は、－標準値～＋(標準値－垂直方向遅延量制限※)の範囲で設定可能です。

※垂直方向遅延量制限は、『SetCameraData』コマンドのＶＩＮモード詳細を参照してください。
ＶＩＮモード、ＶＩＮ拡張モードによって遅延量制限が変わります。

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	ＤＩトリガ入力起動（正極性）
DI_NEGA	0x11	ＤＩトリガ入力起動（負極性）
DI_THROUGH	0x100000	ＤＩスルー出力 ※1
ENCODER1	0x20	エンコーダーＣＨ１トリガ入力起動※2
ENCODER2	0x21	エンコーダーＣＨ２トリガ入力起動※2

※1 ＤＩスルー出力は、他の設定とＯＲ指定が可能です。
※2 エンコーダートリガを使用する場合、SetEncoderMode()を設定してください。

int speed　　シャッタスピード（μS）(10～640000)

※シャッタスピードは10μS単位で設定されます（10μS未満切り捨て）。
例えば、speed = 85μSとした場合シャッタスピードは80μS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
２０	トリガモード設定値範囲外
２１	シャッタスピード設定値範囲外
１３１	ＶＩＮドライバでのエラー
１３２	ＤＵドライバでのエラー
１３３	アクティブビデオポートが未設定
１３８	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（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

フレームシャッタカメラのトリガモードとカメラのシャッタ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TRIGER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TRIGER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。NOTRIGER（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。

・TRIGER（トリガ内部起動）
トリガ起動は、コマンドで生成されます。『CaputureContinuous』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。ラインカメラは、本トリガモードでシングル取り込みはできません。

・DI_POSI/DI_NEGA(ＤＩトリガ入力起動)
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

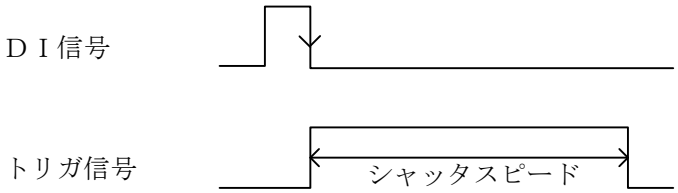
・DI_THROUGH（ＤＩスルー出力）
ＤＩに入力された信号をカメラにそのまま出力します。ＤＩスルー信号によってトリガ信号の起動は行いません。また、本設定はほかのトリガモードとＯＲ指定が可能です。
ＤＩスルー出力時のカメラに出力するＤＩ信号は以下の対応となります。
CH1 － DI4、 CH2 － DI5、 CH3 － DI6、 CH4 － DI7

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TRIGERに再設定して下さい。

・ENCODER1,ENCODER2（エンコーダートリガ）
エンコーダーに同期したトリガをカメラに発行します。ENCODER1は、エンコーダーチャネル１(CH1)は、ENCODER2は、エンコーダーチャネル２(ＣＨ２)となります。
エンコーダートリガを使用する場合は、『SetEncoderMode』コマンドでモードを設定して下さい。

ＤＩトリガ、ＤＩスルー、エンコーダーで使用するＤＩ信号は、一部共用で使用します。
エンコーダーモードを同時に使用する場合は、使用可能な組み合わせをご確認の上ご使用ください。『SetEncoderMode』コマンドリファレンスをご確認ください。

（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:ﾋﾞﾃﾞｵﾎﾟｰﾄ#0～3:ﾋﾞﾃﾞｵﾎﾟｰﾄ#3)
      int        Delay;              出力ディレイ（μS）(0～640000)
      int        Length;             ストロボ出力長（μS）(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１０μＳ単位で設定されます(１０μＳ未満切り捨て)。例えば、Length = ８５μＳとした場合ストロボ出力長は８０μＳになります。

❑ エラーコード

２０　　ストロボチャネル設定値範囲外
２１　　ストロボモード設定値範囲外
２２　　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（ClearIPError）未発行

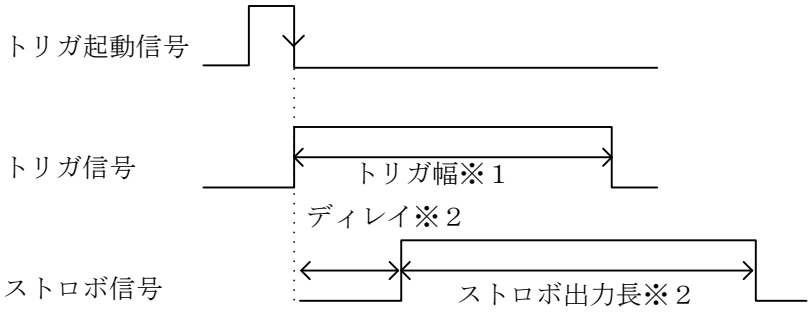
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

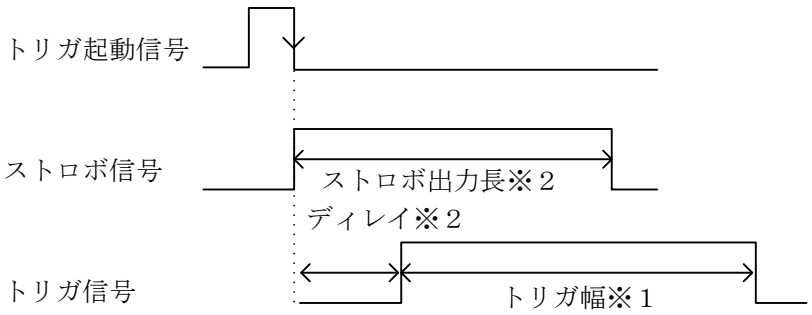
ストロボモードで映像入力する場合、SetTriggerMode()でトリガモードを設定し、本コマンドでストロボ情報を設定してから映像入力を行います。SelectCamera( )コマンドを発行するとトリガモードが解除されますので、注意して下さい。
ストロボは４チャネルあり、それぞれ独立にストロボ情報を設定することができます（但しＮＶＰ－Ａｘ４３０ＣＬは２チャネル）。
ストロボのチャネルとストロボ情報テーブルのビデオポート番号は同じにしてください（ビデオポート#0に対しストロボを使用する場合はストロボチャネルＣＨ０を指定します）。
同時取り込みの場合、アクティブビデオポートに設定されたストロボ信号を出力します。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ネル番号を指定して下さい。

単一映像入力においてストロボモードをストロボ有効（STROBE）で使用する場合は、ストロボ出力期間中であっても、映像入力が終了した時点でストロボ出力は停止します。
連続映像入力においてトリガ内部起動（TRIGER）かつストロボモードをストロボ有効（STROBE）で使用する場合は、『SetCameraData』または『LoadCameraFile』で設定するトリガ周期でトリガおよびストロボを起動します。



トリガ先行モード



ストロボ先行モード

※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 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　　モード
0：ＬＵＴ無効モード
1：ＬＵＴモード
　映像入力でカメラ映像を設定したLUTで変換し出力します。
2：ＲＧＢ-ＬＵＴモード
　ＲＧＢ－ＬＵＴ変換のＬＵＴを設定します。（SelectCameraでマルチコンポーネントモードのＲＧＢ－ＬＵＴ変換を選択した場合のＬＵＴ）

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(mode=1), 262144(mode=2)

int　opt　　オプション。
mode=0,mode=1の場合、０を指定してください。
mode=2の場合、ＲＧＢ－ＬＵＴ変換のデータを指定します。

bit 14-12 ： RGBConvert RGB-LUT変換のデータを指定します。

bit 11- 8 ： Rform[3:0]
RGB-LUT変換時のREDデータ有効ビット設定します。

bit 7- 4 ： Gform[3:0]
RGB-LUT変換時のGREENデータ有効ビット設定します。

bit 3- 0 ： Bform[3:0]
RGB-LUT変換時のBLUEデータ有効ビット設定します。

❑ 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

２０	ＬＵＴ番号設定値範囲外
２１	ＬＵＴモード設定値範囲外
２２	ＬＵＴテーブルがNULL
２３	書き込みデータ数設定値範囲外
２４	オプション設定値範囲外
１４１	ＶＰＰ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

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。
本コマンドは、他タスクも含め映像入力を実施する前に設定してください。
『SelectCamera』でマルチコンポーネントモードオフ（カメラ画像）の場合は、ＬＵＴモードまたは、ＬＵＴ無効モードでご使用ください。マルチコンポーネントモードのＲＧＢ－ＬＵＴ変換を指定する場合は、必ずＲＧＢーＬＵＴモードを設定してください。その他の場合は、ＬＵＴ無効モードに設定してください。

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) ※1	CH1	○		○	
		CH2		○		○
3	RGB, 3TAP ※1 ※2	CH1	○		○	○
4	4TAP, 8TAP ※1	CH1	○	○	○	○

※1　2TAP(Dual),3TAP,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。

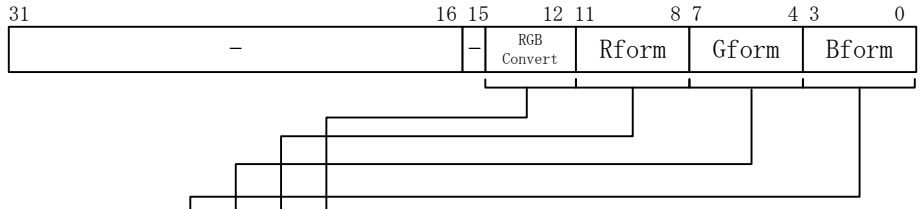
※2　ＲＧＢの場合、ＬＵＴ０～ＬＵＴ２がＲ、Ｇ、Ｂのどのデータを変換対象にするかは以下の通りです。

CameraLink PortA　：LUT0
CameraLink PortC　：LUT2
CameraLink PortB　：LUT3

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。

マルチコンポーネントモードのＲＧＢ－ＬＵＴ変換では、以下のＬＵＴ設定が必要です。
マルチコンポーネントモードのＲＧＢ－ＬＵＴ変換を行う場合は、ＬＵＴ０、ＬＵＴ１両方に同じＬＵＴ設定を行ってください。ただし、ＣＨ１とＣＨ２でそれぞれでＲＧＢ－ＬＵＴ変換を行う場合は、ＬＵＴ０はＣＨ１、ＬＵＴ１はＣＨ２のＬＵＴ設定を行ってください。

optの機能を以下に示します。



RGBConvert[2:0]：ＲＧＢ－ＬＵＴ変換の有効／無効データ設定
111： Ｒ有効、Ｇ有効、Ｂ有効
others： 設定しないでください。

Rform[3:0]：RGB-LUT変換時のREDデータ有効ビット
rform[3]：REDデータ(8bit)の有効ビット設定
0：固定(上位ビット有効)
※8bitすべて有効の場合０としてください。
rform[2:0]：REDデータ(8bit)の有効ビット設定
111：ＲＥＤデータ(8bit)の上位7bit有効
110：ＲＥＤデータ(8bit)の上位6bit有効
100：ＲＥＤデータ(8bit)の上位4bit有効
010：ＲＥＤデータ(8bit)の上位2bit有効
000：ＲＥＤデータ(8bit)の上位8bit有効
others： 設定しないでください。

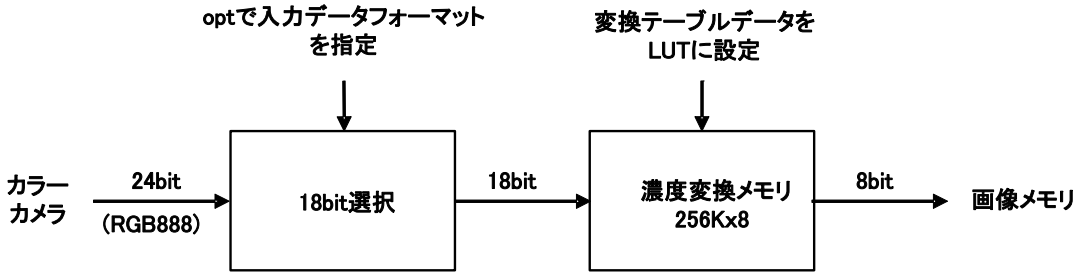
Gform[3:0]： ＲＧＢ－ＬＵＴ変換時のＧＲＥＥＮデータ有効ビット
機能はRformと同じ。

Bform[3:0]： ＲＧＢ－ＬＵＴ変換時のＢＲＵＥデータ有効ビット
機能はRformと同じ。

※optの未使用ビットは０を指定してください。
Rform、Gform, Bformの有効ビット数の合計が18bitとなるように設定してください。
以下のbit組み合わせで使用してください。
Rform、Gform, Bform = 6,6,6 (RGB666)
Rform、Gform, Bform = 0,0,2 (RGB882)
Rform、Gform, Bform = 0,2,0 (RGB828)
Rform、Gform, Bform = 2,0,0 (RGB288)
Rform、Gform, Bform = 7,7,4 (RGB774)
Rform、Gform, Bform = 7,4,7 (RGB747)
Rform、Gform, Bform = 4,7,7 (RGB477)

（1）ＲＧＢ－ＬＵＴモードについて

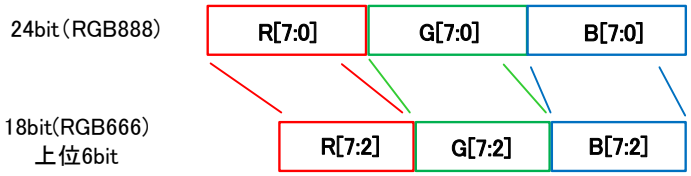
マルチコンポーネントモードのＲＧＢ－ＬＵＴ変換を使用する場合、本コマンドでmodeをＲＧＢ-ＬＵＴモードを指定した上でoptとLUTで下図の入力データフォーマットおよび濃度変換メモリを設定します。



RGB-LUT変換

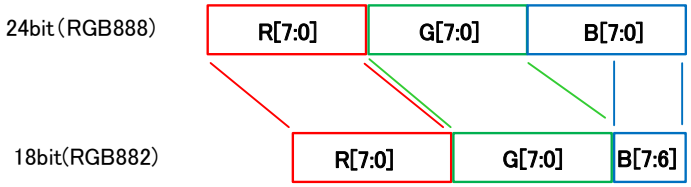
・入力データフォーマット設定例

①RGB888→RGB666上位



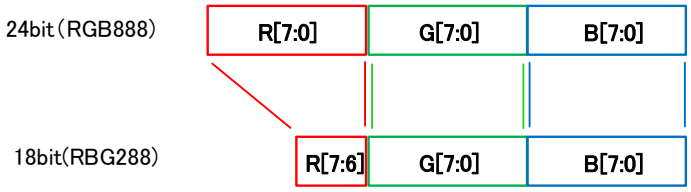
opt指定
RGBConvert = B’111: R有効、G有効、B有効
RForm = B’0110: R上位6bit
GForm = B’0110: G上位6bit
BForm = B’0110: B上位6bit

②RGB888→RGB882



opt指定
RGBConvert = B’111: R有効、G有効、B有効
RForm = B’0000: R8bit
GForm = B’0000: G8bit
BForm = B’0002: B上位2bit

③RGB888→RGB288



opt指定
RGBConvert = B’111: R有効、G有効、B有効
RForm = B’0002: R上位2bit
GForm = B’0000: G8bit
BForm = B’0000: B8bit

※RGBの有効データの合計が18bitとなるように設定してください。

・LUT設定

入力データフォーマットの変換後の18bitデータ並びをＬＵＴテーブルのアドレスとして指定してください。
例えば 上記入力データフォーマット設定例①RGB888→RGB666上位の場合に、元の24bit(RGB888)の映像の、R = 4、G = 252、B = 0を20に変換する場合は、

LUT[8128] = 20

※8128 = h’1FC0 = R:1(b’000001), G:63(b’111111), B:0(b’000000)

(R、G、Bは8bitの内上位6bit(下位2bitを捨て)選択しているので2bit右シフトした値で指定)

と指定します。

2.3.9 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

２０	シーン番号のカウンタ値格納アドレス範囲外
１３１	ＶＩＮドライバでのエラー
１３３	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、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。

2.3.10　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

１３１	ＶＩＮドライバでのエラー
１３３	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。

2.3.11 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

２０	カメラタイプ設定値範囲外
２１	映像入力用カメラデータテーブル定範囲外

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

ユーザ定義カメラを登録します。
カメラの設定にあわせたカメラデータを設定します。設定が異なると映像入力が正しくできません。
すべてのカメラの接続を保証するものではありません。弊社までお問い合わせください。

ユーザー定義カメラは、カメラタイプ(type)の0x80～0xFFのいずれかに登録し『SelectCamera()』で指定します。

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 = 0x00000003;
　　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を設定して下さい。

・カメラの有効画素（最大フレームサイズ）（maxfs）
　カメラから出力されるサイズを設定してください。パーシャルモードなどで出力サイズが変わる場合はそれに合わせ設定してください。１～１６３８４で設定可能です。ただし、モードによっては使用できる有効画素の最大値が異なります。
　エリアカメラの垂直出力サイズ（= Vsize）は使用するVINモードにアラインを合わせてください（VINモード＝single:１ライン単位、dual:２ライン単位、triple:３ライン単位、quad:４ライン単位）。後述のVINモード詳細を確認して下さい。

（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。

・水平周波数（Hcycle）
　０を設定して下さい。

・オプション
　０を指定して下さい。

・リザーブ(Reserve)
　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                              */
         unsigned char      scantype; /* 0:Area Scan 1: Line Scan                       */
         unsigned char      tapform;  /* TAPフォーマット                                */
                            /* 0x14 : 1TAP                                              */
                            /* 0x23 : 2TAP-2X1Y-PXL                                     */
                            /* 0x33 : 3TAP-3X1Y-PXL                                     */
                            /* 0x43 : 4TAP-4X1Y-PXL                                     */
                            /* 0x83 : 8TAP-8X1Y-PXL                                     */
                            /* 0x84 : 8TAP-8X1Y-BLK2                                    */
                            /* 0x88 : 8TAP-8X1Y-BLK3                                    */
         unsigned char      pxlform;  /* 0:8bpp 3:24bpp(RGB)                            */
         unsigned char      pxlconv;  /* 0                                              */
         unsigned char      seldval;  /* 0:DVAL 1:LVAL                                  */
         unsigned char      vinmode;  /* 0:single, 1:dual, 2:triple, 3:quad             */
         unsigned char      reserve[13];
                            /* reserve[1] : RAWカメラフォーマット                       */
                            /*              0x00 : RG                                   */
                            /*              0x01 : GR                                   */
                            /*              0x02 : GB                                   */
                            /*              0x03 : BG                                   */
                           /* reserve[2] : 拡張モード   0:垂直拡張, 1:水平拡張          */
                           /* reserve[3] : 周波数       0:85MHz以下,1:50MHz以下         */
                           /* reserve[0],[4-12] : 0固定                                */
} VIDEO_CLINK_MODE_TBL, *PVIDEO_CLINK_MODE_TBL;
```

・カメラリンクタイプ（camtype）
PoCL、CameraLinkの場合０を指定してください。
PoCL-Liteは接続できません。

・スキャンタイプ（scantype）
エリアスキャンカメラの場合０を指定して下さい。
ラインスキャンカメラの場合１を指定して下さい。

・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。
※RGBカメラの場合１ＴＡＰを指定してください。

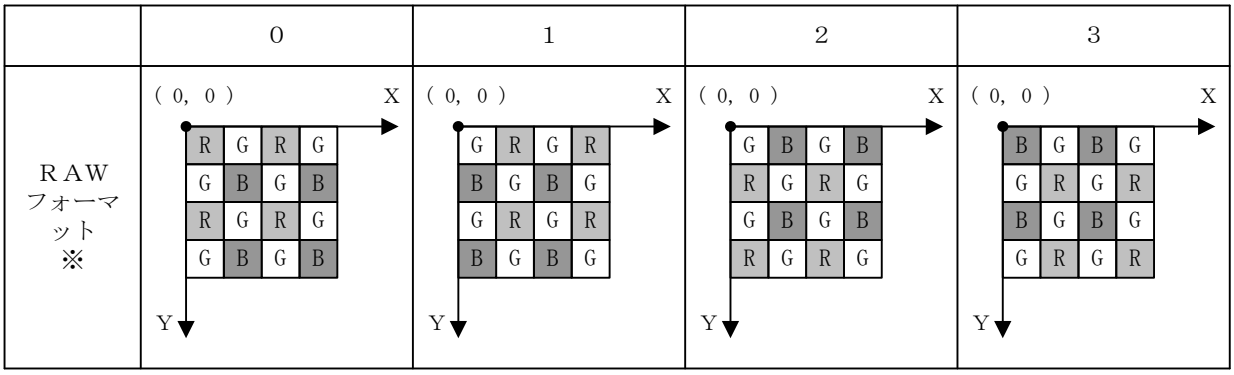
・画素フォーマット（pxlform）
カメラの画素フォーマットを指定します。
8bpp ： 0、　 24bpp(RGB) ： 3
※ＲＧＢカメラの場合３を指定してください。
上記以外は指定できません。

・画素変換（pxlconv）
０を指定して下さい。

・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）
ＶＩＮ拡張モード(reserve[2])と合わせて指定します。後述のＶＩＮモード詳細を参照してください。
0　：single
1　：dual
2　：triple
3　：quad

・ＲＡＷフォーマット(reserve[1])
ＲＡＷカメラの場合にカメラ出力にあわせて指定してください。
0x00　：RG
0x01　：GR
0x02　：GB
0x03　：BG
本設定は、マルチコンポーネントモードのＲＡＷｔｏＲＧＢ変換で使用されます。ＲＡＷカメラでない場合は０を指定してください。



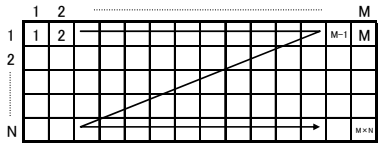
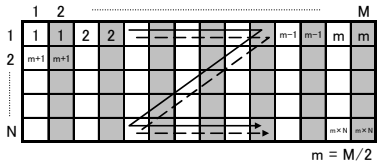
※カメラから出力される映像データの左上が始点（０，０）です。

・ＶＩＮ拡張モード（reserve[2]）
ＶＩＮモードとあわせて指定します。後述のＶＩＮモード詳細を参照してください。
０：垂直拡張，1:水平拡張

・周波数（reserve[3]）
カメラの周波数に合わせて指定してください。
０：85MHz以下，　１：50MHz以下
ＴＡＰフォーマットを３ＴＡＰ、８ＴＡＰとした場合は１を設定してください。（５０ＭＨｚを超える周波数のカメラ入力はできません。）４ＴＡＰとした場合は、５０ＭＨｚ未満のカメラであっても０を設定して下さい。

・reserve[0,4～12]
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-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。
3TAP-3X1Y-PXL	0x33	2TAP-2X1Y-PXLが２画素単位に対し、３画素単位で転送するフォーマットです。
4TAP-4X1Y-PXL	0x43	2TAP-2X1Y-PXLが２画素単位に対し、４画素単位で転送するフォーマットです。
8TAP-8X1Y-PXL	0x83	2TAP-2X1Y-PXLが２画素単位に対し、８画素単位で転送するフォーマットです。
8TAP-8X1Y-BLK2	0x84	カメラタイプTAKEX_TL_8K35ACL,TAKEX_TL_16KACL用8TAPフォーマットです。２TAP単位で奇数、偶数画素を転送します。 ・水平8192画素の場合 TAP1 ： 1～2047 TAP2 ： 2～2048 TAP3 ： 2049～4095 TAP4 ： 2050～4096 TAP5 ： 4097～6143 TAP6 ： 4098～6144 TAP7 ： 6145～8191 TAP8 ： 6146～8192 ・水平16384画素の場合 TAP1 ： 1～4095 TAP2 ： 2～4096 TAP3 ： 4097～8191 TAP4 ： 4098～8192 TAP5 ： 8193～12287 TAP6 ： 8194～12288 TAP7 ： 12289～16383 TAP8 ： 12290～16384
8TAP-8X1Y-BLK3	0x88	カメラタイプNED_RMSL8K76CL用8TAPフォーマットです。 ２TAP単位で奇数、偶数画素を転送します。TAP1～4,TAP5～8で転送順序が逆になります。 ・水平8192画素の場合 TAP1 ： 1～2047 TAP2 ： 2～2048 TAP3 ： 2049～4095 TAP4 ： 2050～4096 TAP5 ： 6143～4097 TAP6 ： 6144～4098 TAP7 ： 8191～6145 TAP8 ： 8192～6146

●ＶＩＮモード詳細

ＶＩＮモード仕様を以下に示します。

－ エリアカメラ －

・使用可能なカメラおよび有効画素最大

VIN拡張モード	ＶＩＮモード	使用可能なカメラTAP（周波数）※1 ※2	使用可能カメラの有効画素最大
0：垂直拡張(0固定)	0:single	1TAP(85Mz以下),2TAP(50MHz以下)	4096×4096
	1:dual	2TAP(85Mz以下),3TAP/RGBカメラ(50MHz以下)	
	2:triple	RGBカメラ(85Mz以下)	
	3:quad	4TAP(85Mz以下),8TAP(50MHz以下)	

※1 すべてのカメラの接続を保証するものではありません。
※2 4TAP/8TAPはNVP-Ax435FCLでのみ使用可能です。

・ビデオフレームサイズ制限

VIN拡張モード	ＶＩＮモード	ビデオフレームサイズ 水平方向	ビデオフレームサイズ 垂直方向 ※1※2
0：垂直拡張(0固定)	0:single	有効画素数(最大4096)以下、128画素単位 ※3	有効画素数(最大4096)以下、1ライン単位
	1:dual		有効画素数(最大4096)以下、2ライン単位
	2:triple		有効画素数(最大4095)以下、3ライン単位
	3:quad	有効画素数(最大3968)以下、128画素単位	有効画素数(最大4096)以下、4ライン単位

※1 設定可能な最小値は single:2,dual:4,triple:6,quad:8となります。
※2 ＲＧＢカメラは、ＶＩＮモード 0：singleと同じ制限になります
※3 ＲＧＢ(50MHz以下,ＶＩＮモード＝１)カメラの場合水平方向は最大２０４８となります。

・接続可能CHと接続制限

ＶＩＮモード	接続可能CH ※1	他CHの接続制限
0:single	CH1～CH4※2	なし
1:dual	CH1,CH2	CH1での使用でCH3が、CH2での使用でCH4が使用不可になります。
2:triple	CH1	CH3,CH4は使用不可になります。
3:quad	CH1 ※3	CH2,CH3,CH4は使用不可になります。

※1 マルチコンポーネントモードを使用しない（0：カメラ画像）場合です。
　　マルチコンポーネントモードでの接続制限も合わせてご確認下さい。（『SelectCamera』）
※2 ＣＨ３、ＣＨ４はNVP-Ax435CLでのみ接続可能です。
※3 NVP-Ax435FCLのＣＨ１でのみ使用可能です。

・遅延量制限※

VIN拡張モード	ＶＩＮモード	水平方向	垂直方向
0：垂直拡張(0固定)	0:single	２画素単位	1ライン単位
	1:dual		2ライン単位
	2:triple		3ライン単位
	3:quad		4ライン単位

※ 『SetVFDelay』を参照してください。
　ＲＧＢカメラの場合は、水平方向２画素単位、垂直方向１ライン単位となります。

・映像入力に使用する画像メモリ

画像メモリ水平方向サイズ：ビデオフレームサイズ～最大以下
　　(最大sinle:16352,dual:8064,triple:5376,quad:3968)
　　かつ１２８画素単位（サイズが最大の場合を除く）
画像メモリ垂直方向サイズ：ビデオフレームサイズ以上（最大１６３８４）

- ラインカメラ -

・使用可能なカメラおよび有効画素最大

VIN拡張 モード	ＶＩＮ モード	使用可能なカメラTAP（周波数）※1 ※2	使用可能カメラの 有効画素最大
0：垂直拡張	0:single	2TAP(50MHz以下)	4096
	1:dual	2TAP(85Mz以下)	
	2:triple	使用できません。	
	3:quad	4TAP(85Mz以下),8TAP(50MHz以下)	
1：水平拡張	0:single	使用できません。	-
	1:dual	使用できません。	-
	2:triple	使用できません。	-
	3:quad	8TAP(50MHz以下),2TAP(85MHz以下)	16384

※1 すべてのカメラの接続を保証するものではありません。
※2 4TAP/8TAPはNVP-Ax435FCLでのみ使用可能です。

・ビデオフレームサイズ制限

VIN 拡張モード	ＶＩＮ モード	ビデオフレームサイズ 水平方向	ビデオフレームサイズ 垂直方向 ※1 ※2
0：垂直拡張	0:single	有効画素数 (最大4096)以下、 128画素単位	(最大4096)以下、1ライン単位
	1:dual		(最大8192)以下、2ライン単位
	2:triple		(最大12288)以下、3ライン単位
	3:quad	有効画素数 (最大3968)以下、 128画素単位	(最大16384)以下、4ライン単位
1：水平拡張 ※5	3:quad	有効画素数 (最大16352)以下、 512画素単位 ※3,4	(最大4096)以下、1ライン単位

※1 設定可能な最小値は垂直拡張の場合、single:2,dual:4,triple:6,quad:8となります。
　水平拡張の場合 2となります。
※2 RGBカメラは、ＶＩＮモード 0：singleと同じ制限になります
※3 水平拡張の場合、設定可能な値はquad: 4096～16352です。
　4096以下での使用は垂直拡張モードに設定してください。
※4 ビデオフレームサイズ16352未満の場合512画素単位です。
※5 水平拡張の場合はカメラ有効画素の水平方向(maxfs.Hsize)とビデオフレームサイズは同値である必要があります。
　ただしビデオフレームサイズが水平16352の場合は、maxfs.Hsize=16384となります。

・接続可能CHと接続制限

エリアカメラと同じです。
但しマルチコンポーネントモードは使用できません。（0：カメラ画像のみ）『SelectCamera』）

・遅延量制限

水平方向は、２画素単位です。垂直方向は遅延量を調整できません。0としてください。
水平拡張モードの場合は、水平方向も遅延量を調整できません。0としてください。

・映像入力に使用する画像メモリ

エリアカメラと同じです。
VIN拡張モードが水平拡張の場合は水平方向は最大16352となります。

（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μＳ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。
１０μＳ～６４０ｍＳの範囲で設定可能です。１０μＳ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。
１～４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。
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。

2.3.12　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

２０	カメラタイプ設定値範囲外
２１	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。

2.3.13　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

２０	カメラタイプ設定値範囲外
２１	ファイル名不正、カメラデータ設定値範囲外
４８	動的メモリ確保エラー
１００	カメラデータファイルオープンエラー、
１０１	カメラデータファイルのフォーマット、または設定値が不正

※ 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(type)は０ｘ８０～０ｘＦＦとしてください。カメラタイプはSelectCamera()で指定します。

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例を以下に示します。カメラデータは[]で示されるでセクションを記述し、次行からデータ（キー=値）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、セクション行、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 Versiom]
Version=3.0

[Camera Info]
name=xxxxxxxx
attr=CAMATR_CLINK
option=0

[Video Frame]
pluslng_vsize=0
pluslng_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
vin_exmode=VEXT
freq_mode=50MHZ
raw_format=RG

[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ついては、動作の保証はできません。

❑ カメラファイルーカメラデータ対応表

カメラファイル			カメラデータ ※			
セクション名	キー	値	テーブル名	変数	値	備考
[Camera Data Version]	Version	3.0	VIDEO_CAM_FRM_INFO	inf1,inf2	3,0	
[Camera Info]	name	文字列（半角63文字まで）	VIDEO_CAM_TBL	name[64]	文字列（半角63文字まで）	
	attr	CAMATR_CLINK		CamAtr	0x07	
		CAMATR_CLINKRGB			0x08	
	option	0		OptInfo	0	
[Video Frame]	plusIng_vsize	0	VIDEO_FRM_TBL	plusIng.Vsize	0	
	plusIng_hsize	0		plusIng.Hsize	0	
	maxfm_vsize	1～16384		maxfs.Vsize	1～16384	
	maxfm_hsize	1～16384		maxfs.Hsize	1～16384	
[Camera Link]	camera_type	CAMERA_LINK	VIDEO_CLINK_MODE_TBL	camtype	0	
	scan_type	AREA_SCAN		scantype	0	
		LINE_SCAN			1	
	tap_format	1TAP		tapform	0x14	
		2TAP-2X1Y-PXL			0x23	
		3TAP-3X1Y-PXL			0x33	
		4TAP-4X1Y-PXL			0x43	
		8TAP-8X1Y-PXL			0x83	
		8TAP-8X1Y-BLK2			0x84	
		8TAP-8X1Y-BLK3			0x88	
	pixel_format	8BPP		pxlform	0	
		24BPP_RGB			3	
	pixel_convert	NON		pxlconv	0	
	dval	DVAL		seldval	0	
		LVAL			1	
	vin_mode	SINGLE		vinmode	0	
		DUAL			1	
		RGB			2	
		TRIPLE			2	
		QUAD			3	
	freq_mode	50MHZ		reserve[3]	1	
		85MHZ			0	
	raw_format	NON		reserve[1]	0x00	
		RG			0x00	
		GR			0x01	
		GB			0x02	
		BG			0x03	
	vin_exmode	NON		reserve[2]	0	
		VEXT			0	
		HEXT			1	
[Trigger]	cycle	10-640000	VIDEO_VPP_DATA_TBL	data[0]	10-640000	
	out_cc	CC1		data[1]	1	
		CC2			2	
		CC3			3	
		CC4			4	
	diout_cc	CC1		data[2]	1	
		CC2			2	
		CC3			3	
		CC4			4	
	cc1_mode	NORMAL		data[3]	0x0	bit OR設定
		INVERT			0x1	
	cc2_mode	NORMAL			0x0	
		INVERT			0x2	
	cc3_mode	NORMAL			0x0	
		INVERT			0x4	
	cc4_mode	NORMAL			0x0	
		INVERT			0x8	
	voffset	0～16384		data[4]	0～16384(bit[31:16])	bit OR設定
	hoffset	0～16384			0～16384(bit[15:0])	

※ 記載のないカメラデータのメンバは0が設定されます。

2.3.14　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
１０	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	キャプチャ先画面ＩＤテーブル設定値範囲外
１３１	ＶＩＮドライバでのエラー
１３２	ＤＵドライバでのエラー
１３３	アクティブビデオポートが未設定
１３５	画面サイズまたは画面タイプが不正
１３６	システム管理データが不正
１３７	画面管理データが不正
１３８	映像入力画面未設定
－	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

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のメモリベー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。

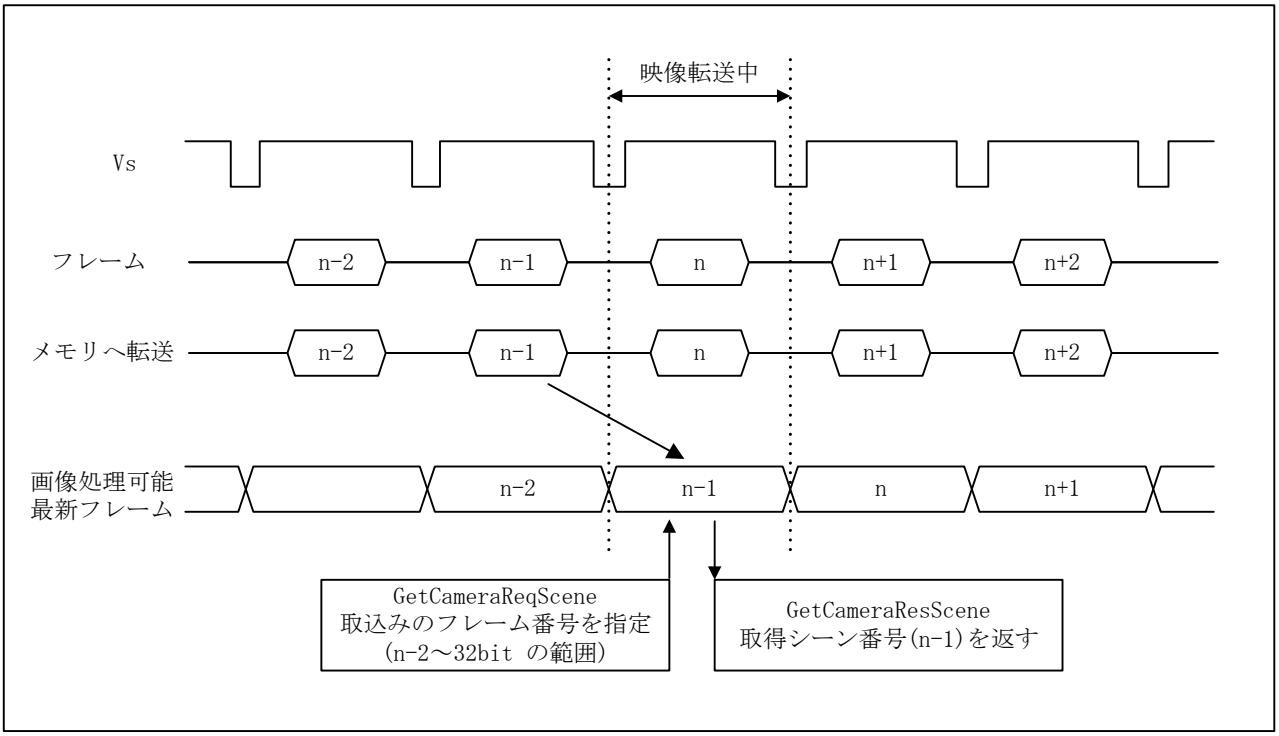
本コマンドの２重起動を行った場合は、パラメータチェックだけ行い、何もせずに正常終了します。
また、画面番号テーブルに指定した３面分の画面番号は本コマンドのワーク画面扱いとなりますので、これらの画面番号は画像処理に使用しないで下さい。

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）～３２ｂｉｔ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あります。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
```

2.3.15　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
１３１	ＶＩＮドライバでのエラー
１３３	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。

2.3.16　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	○	×	×	×	×	×	◆	-	-
	×	○	×	×	×	×	◆	-	-

❑ 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	ＶＩＮ入力タイムアウト
１０	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	シーン番号を返すアドレスが不正
２３	映像入力タイムアウト設定値範囲外（０未満）
１３１	ＶＩＮドライバでのエラー
１３２	ＤＵドライバでのエラー
１３３	アクティブビデオポートが未設定
１３５	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
１３６	システム管理データが不正
１３７	画面管理データが不正
１３８	映像入力画面未設定　または、単一映像入力動作時
－	エラーリセットコマンド（ClearIPError）未発行

※ 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。

2.3.17　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	○	×	×	×	×	×	◆	－	－
	×	○	×	×	×	×	◆	－	－

❑ 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	ＶＩＮ入力タイムアウト
１０	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	シーン番号を返すアドレスが不正
１３１	ＶＩＮドライバでのエラー
１３２	ＤＵドライバでのエラー
１３３	アクティブビデオポートが未設定
１３５	画面サイズが不正、または画面タイプが不正
１３６	システム管理データが不正
１３７	画面管理データが不正
１３８	映像入力画面未設定　または、単一映像入力動作時
１３９	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（ClearIPError）未発行

※ 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。

2.3.18　カメラ映像入力エンコーダーモードの設定（ SetEncoderMode ）

❑ C言語API

int ret = SetEncoderMode(int ch , ENC_INFO *encinfo);

❑ パラメータ

Int　　ch	エンコーダーチャネル（0:CH1, 1:CH2）
ENC_INFO *encinfo	エンコーダー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  videoch;   0固定
        int encA;       Ａ相入力指定　0:ラインドライバ, 1: IODI
        int encB;       Ｂ相入力指定　0:ラインドライバ, 1: IODI
        int encZ;       Ｚ相入力指定　0:ラインドライバ, 1: IODI
        int encstart;   開始モード
                        0:Ｚ相立ち上がり入力で開始(Z相入力毎カウントはリセットされる)
                        1:Ｚ相立ち上がり入力で開始（開始後はＺ相入力は無効）
                        2:コマンド『EncCountStart』で開始（Ｚ相無効）
        int match;      0固定
        int encmode;    エンコーダモード
                         0:Ａラインのみ使用（Ｂライン使用しない）
                         1:Ｂラインのみ使用（Ａライン使用しない）
                         2:CW(時計回り）方向カウントアップ
                         3:CCW（反時計回り）方向カウントアップ
        int cntmode;    カウントモード
                         0:x1倍でカウントする。
                         1:x2倍でカウントする。
                         2:x4倍でカウントする。
        int cntdown;    カウントダウンモード
                         0:カウントダウンする。
                         1:カウントダウンしない。
        int ecnt1;      トリガ開始カウント設定(1～32767 )
        int ecnt2;      トリガ出力間隔カウント設定(1～32767 )
        int ecnt3;      トリガ数カウント設定(0～32767)
        int opt;         0固定
} ENC_INFO;
```

❑ エラーコード

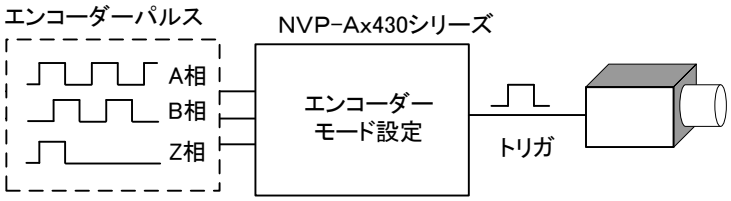
２０	エンコーダーチャネル設定値範囲外
２１	エンコーダー情報テーブル設定値範囲外
１４１	VPPドライバでのエラー
－	エラーリセットコマンド（ClearIPError）未発行

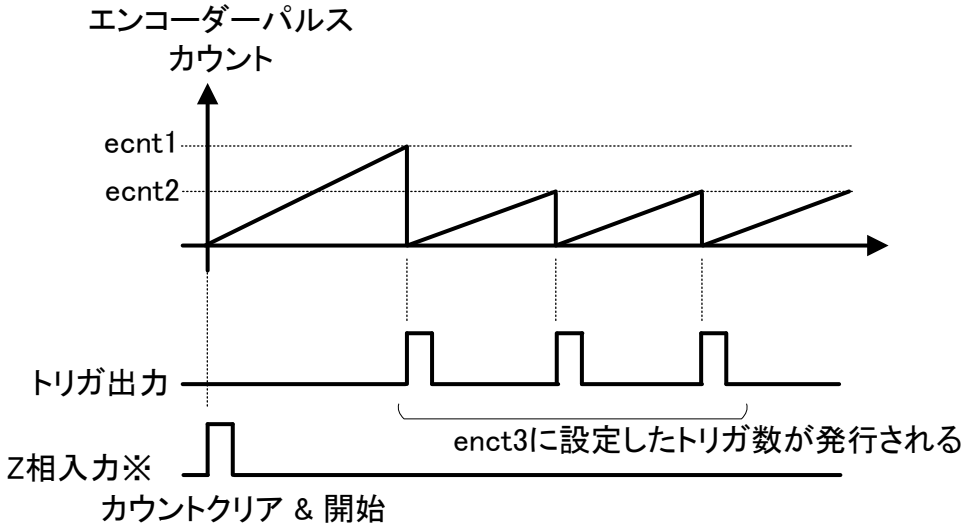
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

エンコーダーに同期した映像入力が可能です。本設定に従ってエンコーダーパルスをカウントし、カメラにトリガを出力します。エンコーダーを使用するには、本設定および『SetTriggerMode』でエンコーダートリガを指定し、『EncoderStart』でエンコーダーを開始してください。

本コマンドによる設定はエンコーダーが停止中に行ってください。





※コマンドでカウントを開始、クリアすることも可能です。

Ｚ相入力で、エンコーダーのカウントが開始されます。カウント値は0クリアされecnt1からカウント動作します。Ｚ相入力はエンコーダーのカウント開始、クリアを目的としています。エンコーダーのＺ相との接続を示すものではありませんのでご注意ください。
カウントは、Ａ相、Ｂ相のパルスをencmodeおよびcntmode、cntdownの設定に従って行われます。カウントがecnt1に達するまでトリガは出力されません。ecnt1に達した後、ecnt2のカウント間隔でトリガを出力します。ecnt3 >0の場合トリガ数がecnt3を超えた場合トリガは出力されません。ecnt3 = 0の場合トリガ数は制限されません。
カウントの開始条件はencstartで指定できます。
映像入力(『GetCamera』) のタイミングは、ＤＩ割り込みを使って外部から入力してください。

(1)videoch
0固定として下さい。

(2)エンコーダーパルス指定(encA, encB, encZ)
A、B、Z相のエンコーダーパルスの入力I／Fを選択します。A、B、Z相すべて同じI/Fを選択し使用してください。
・0：ラインドライバ
・1：IODI
各相の設定値に対する入力信号を以下に示します。

No.	エンコーダー入力	設定変数	enc* 設定値	
			0	1
1	A相	encA	ENC1A(ENC2A)	IODI[0](IODI4])
2	B相	encB	ENC1B(ENC2B)	IODI[1](IODI5])
3	Z相	encZ	ENC1Z(ENC2Z)	IODI[2](IODI6])

()内はCH2の対応信号です。

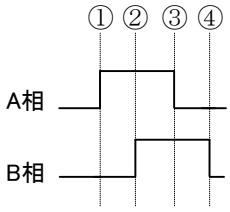
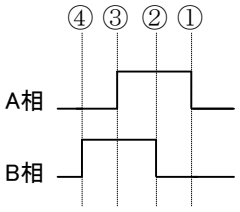
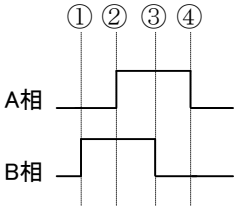
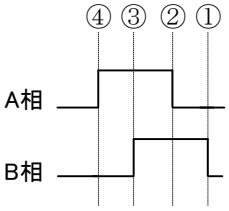
ラインドライバI/Fは、NVP-Ax430ACL/Ax435CL/Ax435FCLで使用可能です。
IODI I/Fは、下記の通りDIトリガ入力と信号ラインが共通です。エンコーダーパルスで使用する場合、当該信号での、DIトリガ入力(DI_POS,DI_NEG,DI_THROUTH)はできませんのでご注意ください。
IODI[0]：カメラCH1のDIトリガ入力または、エンコーダーCH1のA相入力
IODI[1]：カメラCH2のDIトリガ入力または、エンコーダーCH1のB相入力
IODI[2]：カメラCH3のDIトリガ入力または、エンコーダーCH1のZ相入力
IODI[4]：カメラCH1のDIスルー入力または、エンコーダーCH2のA相入力
IODI[5]：カメラCH2のDIスルー入力または、エンコーダーCH2のB相入力
IODI[6]：カメラCH3のDIスルー入力または、エンコーダーCH2のZ相入力

(3)開始モード(encstart)
エンコーダーのカウントリセット&開始条件を指定します。
・0:Ｚ相立ち上がり入力毎開始(Ｚ相入力毎カウントはリセットされる)
・1:Ｚ相立ち上がり入力で開始（開始後はＺ相入力は無効)
・2:コマンド『EncCountStart』で開始（Ｚ相無効)

(4) match
0固定としてください。

(5)エンコーダモード(encmode)
エンコーダーパルスのカウント方法を指定します。
・0 ： Ａ相立ち上がりエッジでカウントアップ
・1 ： Ｂ相立ち上がりエッジでカウントアップ
・2 ： ＣＷ(時計回り）方向カウントアップ
・3 ： ＣＣＷ（反時計回り）方向カウントアップ

(6)カウントモード(cntmode)
カウントの倍数
・0 ： x1倍でカウントする。（カウント条件②)
・1 ： x2倍でカウントする。（カウント条件②④)
・2 ： x4倍でカウントする。（カウント条件①②③④)
※encmode = 0, 1の場合本設定は無効となります。０を指定して下さい。

カウント条件	カウントアップ	カウントダウン
CW(時計回り) encmode = 2		
CCW(反時計回り) encmode = 3		

(7)カウントダウンモード(cntdown)
・0：カウントダウンする、 1:カウントダウンしない
cntdown = 0の場合、cntmodeの設定に従いカウントダウンも行います。ただしencmode = 0,1の場合は、cntdown = 1として下さい。

(8)トリガ開始カウント設定(ecnt1)
カウント開始後トリガ出力するまでのカウント値を設定します。１～３２７６７の範囲で設定してください。
トリガ開始カウンタは符号付１６ｂｉｔのカウンタです。

(9)トリガ出力間隔カウント設定(ecnt2)
トリガ出力する間隔をカウント値で設定します。１～３２７６７の範囲で設定してください。
トリガ出力間隔カウンタは符号付16bitのカウンタです。

(10)トリガ数カウント設定(ecnt3)
トリガを出力する最大数を設定します。 １～３２７６７を設定した場合、エンコーダーカウントされても設定した回数を超えるトリガ信号の出力をしません。カウンタがリセットされると解除されます。０を指定した場合トリガ回数の制限はしません。トリガ出力間隔カウンタは符号付１６ｂｉｔのカウンタです。

(11)opt
０固定としてください。

2.3.19　エンコーダー動作開始（ EncoderStart ）

❑ C言語API

```
int ret = EncoderStart( int ch );
```

❑ パラメータ

Int	ch	エンコーダーチャネル（0:CH1, 1:CH2）

❑ 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	エンコーダーチャネル設定値範囲外
１４１	ＶＰＰ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

エンコーダーの動作を開始します。あらかじめ『SetEncoderMode』を設定してください。
エンコーダーを使用して映像入力を行う場合は、映像入力を開始する前にエンコーダーを開始してください。

2.3.20　エンコーダー動作停止（ EncoderStop ）

❑ C言語API

int ret = EncoderStop(int ch);

❑ パラメータ

Int	ch	エンコーダーチャネル（0:CH1, 1:CH2）

❑ 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	エンコーダーチャネル設定値範囲外
１４１	VPP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

エンコーダーの動作を停止します。エンコーダーを使用して映像入力を行っている場合は、映像取り込み中に停止させないでください。本コマンドによりエンコーダーカウント動作もリセットされます。

2.3.21　エンコーダーカウント開始（ EncCountStart ）

❑ C言語API

int ret = EncCountStart(int ch);

❑ パラメータ

Int	ch	エンコーダーチャネル（0:CH1, 1:CH2）

❑ 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	エンコーダーチャネル設定値範囲外
１４１	ＶＰＰ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

『SetEncoderMode』で開始モード(startmode = 2)を設定した場合、エンコーダーを起動後本関数呼び出しでエンコーダーパルスのカウントを開始します。カウント値は０クリアされecnt1からカウント動作します。

2.3.22　エンコーダーカウントー時停止（ EncCountPause ）

❑ C言語API

int ret = EncCountPause(int ch);

❑ パラメータ

Int	ch	エンコーダーチャネル（0:CH1, 1:CH2）

❑ 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	エンコーダーチャネル設定値範囲外
１４１	ＶＰＰ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

エンコーダーパルスのカウントを一時停止します。カウントを再開する場合『EncCountResume』を使用します。

2.3.23　エンコーダーカウント再開（ EncCountResume ）

❑ C言語API

int ret = EncCountResume(int ch);

❑ パラメータ

Int	ch	エンコーダーチャネル（0:CH1, 1:CH2）

❑ 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	エンコーダーチャネル設定値範囲外
１４１	ＶＰＰ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

『EncCountPause』で一時停止したエンコーダーパルスのカウントを再開します。

2.3.24　エンコーダーカウントリセット（ EncCountReset ）

❑ C言語API

int ret = EncCountReset(int ch);

❑ パラメータ

Int	ch	エンコーダーチャネル（0:CH1, 1:CH2）

❑ 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	エンコーダーチャネル設定値範囲外
１４１	ＶＰＰ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

エンコーダーパルスのカウント動作をリセットします。Ｚ相の開始信号または『EncCountStart』でecnt1からカウント動作が始まります。

2.3.25　映像入力タイムアウト時間設定（ SetVideoTimeout ）

❑ C言語API

int ret = SetVideoTimeout (int tmo);

❑ パラメータ

Int	tmo	タイムアウト時間[mS]（tmo>0）

❑ 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	タイムアウト時間範囲外
－	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

映像入力タイムアウト時間を変更します。デフォルトは３００ｍＳです。
タイムアウト時間は、InitIP()ではデフォルト値に初期化されませんのでご注意下さい。
タイムアウト時間の変更は、カメラ映像入力『GetCamera』『GetCameraResScene』前に行ってください。

2.3.26　カメラリンクシリアルI/F接続設定（ SetCLComCH ）

❑ C言語API

```
int ret = SetCLComCH( int Ch );
```

❑ パラメータ

int	Ch	カメラチャネル番号（０～３）

❑ 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	チャネル番号設定範囲外
１４１	ＶＰＰドライバエラー
－	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

ＳＣＩのＣＨ２を使用して、カメラとシリアル通信（カメラリンクシリアル）が可能です。
本コマンドで接続するカメラリンクのシリアルI/Fチャネルを指定してください。
また、スイッチＳＷ２の１～３をＯＦＦとしてください。

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
２０	表示画面の構成制御テーブル設定値範囲外
１３１	ＶＩＮドライバエラー
１３２	ＤＵドライバエラー
１４１	ＶＰＰドライバエラー
－	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
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。

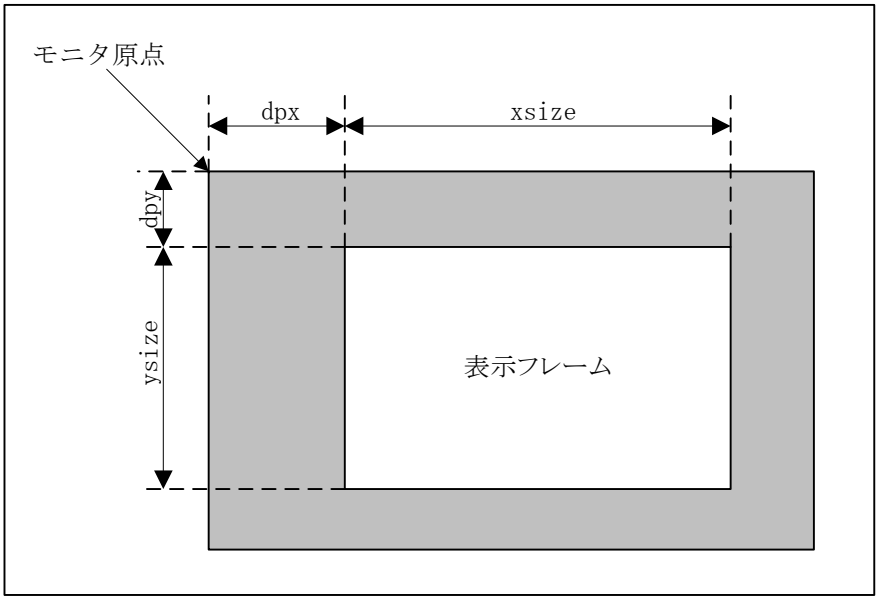
・xsize / ysize ： 表示フレームサイズ

設定項目	動作	設定範囲	備考
xsize	Ｘ方向の表示サイズ	１～４０９５　※	
ysize	Ｙ方向の表示サイズ	１～２０４７　※	

・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	０～４０９５　※	
dpy	Ｙ方向の表示開始位置	０～２０４７　※	

※　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。



・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

・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．７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

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	
	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
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

２０	パレット番号範囲外
２１	パレット情報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

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。

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	（２）	擬似カラー表示：パレット１
		PAL2_DISPLAY	（３）	擬似カラー表示：パレット２
		PAL3_DISPLAY	（４）	擬似カラー表示：パレット３
		PAL4_DISPLAY	（５）	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

２０	映像出力形式値範囲外
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（ClearIPError）未発行

※ 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

４６	画面ＩＤ空きなし
４８	動的メモリ確保エラー
５５	同時入力が設定されている
１３１	ＶＩＮドライバエラー
１３２	ＤＵドライバエラー
１３３	アクティブビデオポートが未設定
１３５	表示画面サイズエラー
１３６	システム管理データが不正
１３８	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
１４１	ＶＰＰドライバエラー
－	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

『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。

『SelectCamera()』コマンドでマルチコンポーネントモードが0：カメラ画像の場合に表示可能です。それ以外は表示できません。

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、『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。

なお、X方向が4096画素を超えるフレームサイズの表示はできません。

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
１３１	ＶＩＮドライバエラー
１３２	ＤＵドライバエラー
１４１	ＶＰＰドライバエラー
－	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
１０	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）
１３１	ＶＩＮドライバエラー
１３２	ＤＵドライバエラー
１３５	表示画面サイズエラー
１３８	表示画面の構成制御設定コマンド（SetConfigDisp）未発行
１４１	ＶＰＰドライバエラー
－	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

『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。
なお、X方向が4096画素を超える画像の表示はできません。

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
１３１	ＶＩＮドライバエラー
１３２	ＤＵドライバエラー
１４１	ＶＰＰドライバエラー
－	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

１０	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	重ね合せ表示属性設定値範囲外
１３２	ＤＵドライバエラー
１３５	表示画面サイズエラー
１３８	表示画面の構成制御設定コマンド（SetConfigDisp）未発行、またはオーバレイプレーン番号、オーバレイパレット番号が範囲外
－	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()』コマンドでオーバレイプレーン番号、及びオーバ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

１３２	ＤＵドライバエラー
－	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
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（ClearIPError）未発行

※ 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）

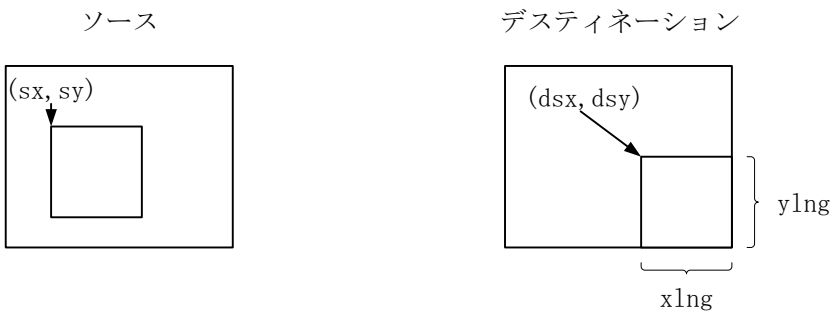
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理実行タイムアウト
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（ClearIPError）未発行

※ 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
－	エラーリセットコマンド（ClearIPError）未発行

※ 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
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）
－　エラーリセットコマンド（ClearIPError）未発行

※ 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
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）
２１	定数値設定範囲外
－	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
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)
Y→RGBY	Y画面をRGBY.R画面にコピー
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)
RGB→RGBY	RGB.R,RGB.G,RGB.B画面をRGBY.R,RGBY.G,RGBY.B画面へそれぞれコピー
RGB16→Y	エラー(13)
RGB16→RGB	エラー(13)
RGB16→RGB16	RGB16画面へコピー
RGB16→RGBY	エラー(13)
RGBY→Y	RGBY.R画面をY画面へコピー
RGBY→RGB	RGBY.R,RGBY.G,RGBY.B画面をRGB.R,RGB.G,RGB.B画面へそれぞれコピー
RGBY→RGB16	エラー(13)
RGBY→RGBY	RGBY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
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）
１６	ソース画面とデスティネーション画面の属性不一致
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

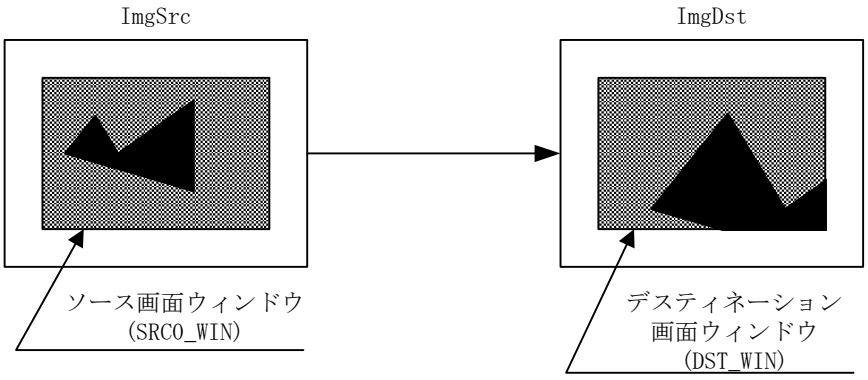
パラメータで指定の変換倍率に従い、拡大／縮小をします。倍率は実数です。

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。

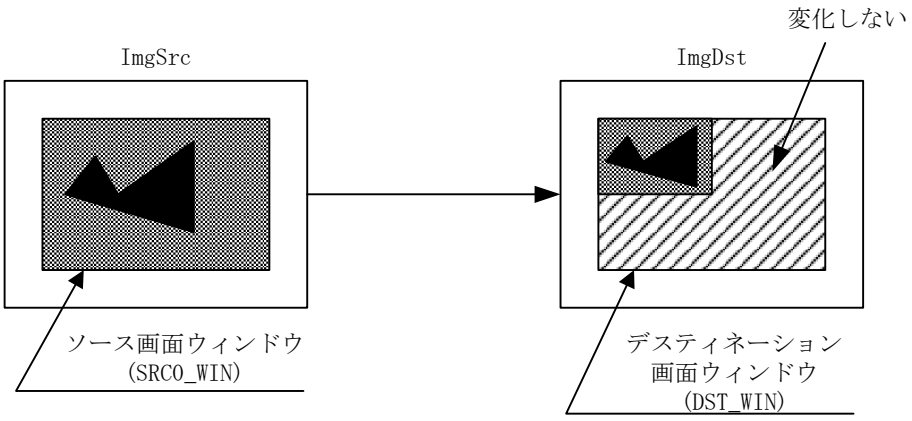
なお、このコマンドを実行したときの出力画面サイズは、デスティネーション画面サイズ、またはデスティネーション画面ウィンドウサイズ（DST_WIN）となります。拡大／縮小後の画面サイズがDST_WINサイズより大きい場合、DST_WINでクリッピングされます。拡大／縮小後の画面サイズがDST_WINサイズより小さい場合、転送対象以外の部分は変化しません。

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
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）
１６	ソース画面とデスティネーション画面の属性不一致
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

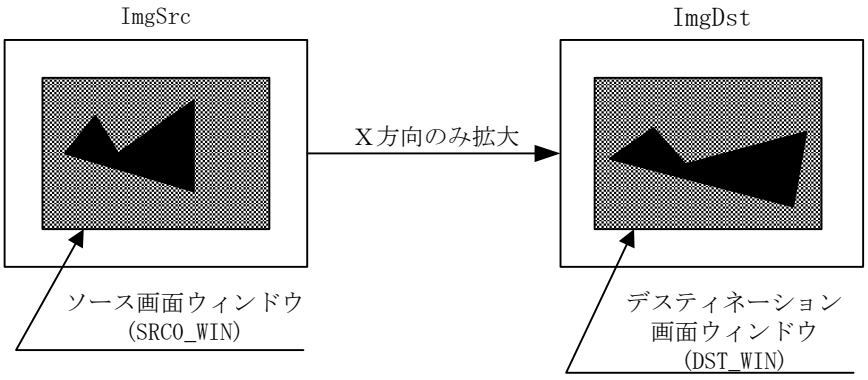
パラメータで指定のＸ方向とＹ方向の変換倍率に従い、拡大／縮小をします。倍率は実数です。

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。

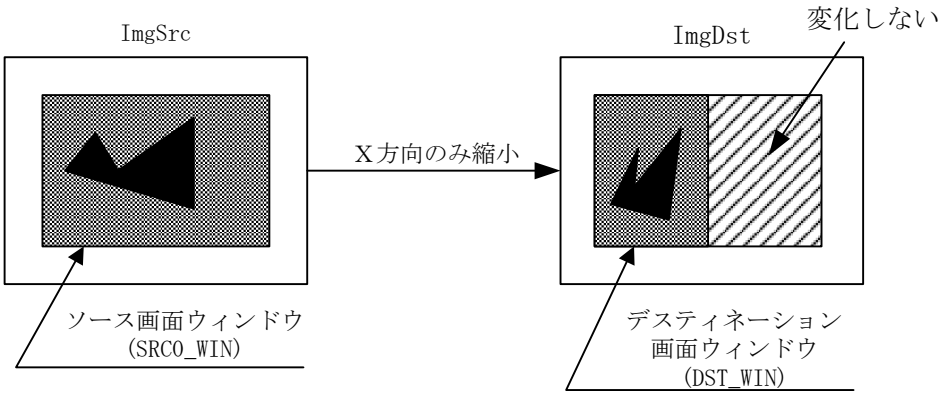
なお、このコマンドを実行したときの出力画面サイズは、デスティネーション画面サイズ、またはデスティネーション画面ウィンドウサイズ（DST_WIN）となります。拡大／縮小後の画面サイズがDST_WINサイズより大きい場合、DST_WINでクリッピングされます。拡大／縮小後の画面サイズがDST_WINサイズより小さい場合、転送対象以外の部分は変化しません。

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
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(13)
RGB16→RGB	エラー(13)
RGB16→RGB16	RGB16画面へ拡大/縮小

[注意事項]
ソース画面の画像メモリのX 方向サイズと変換倍率は以下の条件で指定してください。
ソース画面の画像メモリのX 方向サイズ（バイト） × 変換倍率 ≦ 32767

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
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(13)
RGB16→RGB	エラー(13)
RGB16→RGB16	RGB16画面へ拡大/縮小

[注意事項]
ソース画面の画像メモリのX 方向サイズと変換倍率は以下の条件で指定してください。
ソース画面の画像メモリのX 方向サイズ（バイト） × 変換倍率 ≦ 32767

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

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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、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(13)
RGB16→RGB	エラー(13)
RGB16→RGB16	RGB16画面へ拡大/縮小

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

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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(13)
RGB16→RGB	エラー(13)
RGB16→RGB16	RGB16画面へ拡大/縮小

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。

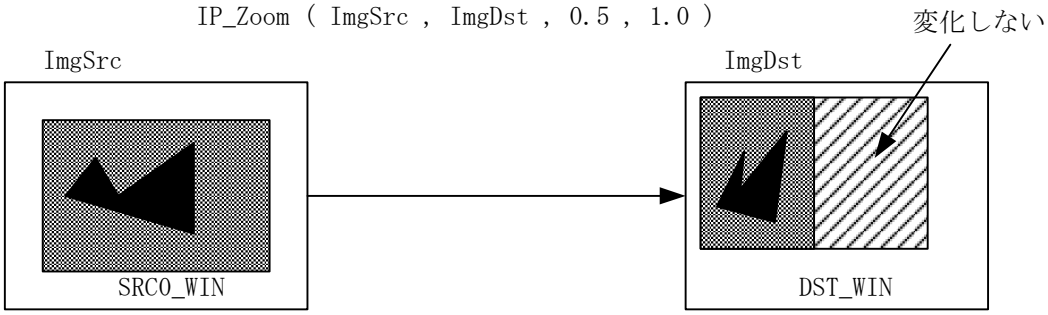
［実行例］

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
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

ソース画面のウイ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。

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 ≧ デスティネーション画面のウィンドウ長

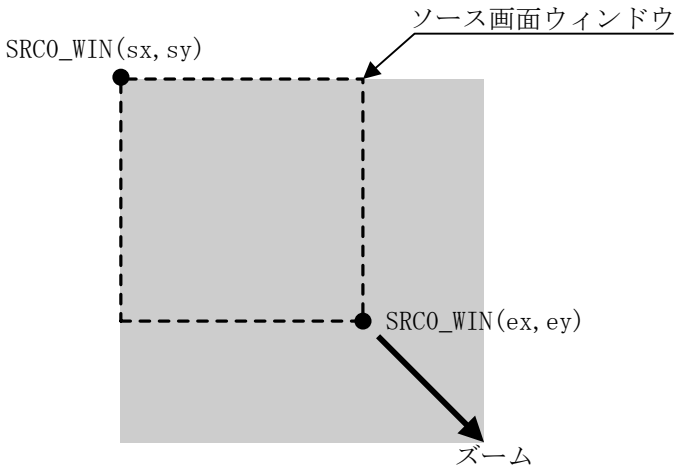
シフト量＜0： シフト量の絶対値 ≧ ソース画面のウィンドウ長×変換倍率

「ソース画面のウィンドウ長×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×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。

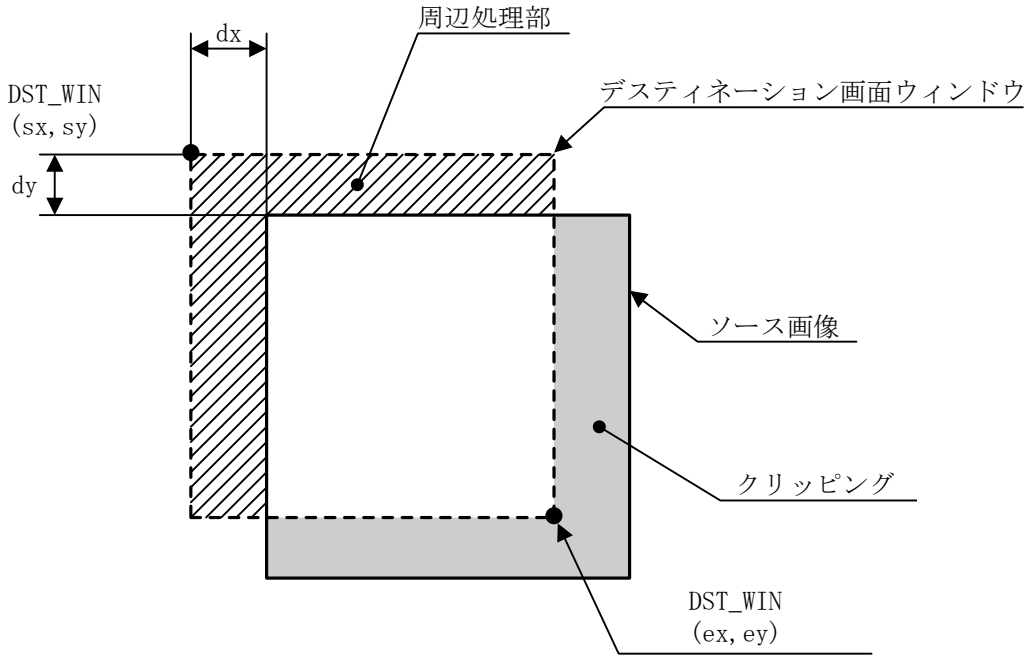
周辺処理オプションにSPACE_128CLEARを指定した場合、システムデータタイプが符号付８ビットのとき周辺部は－１２８になり、符号なし８ビットのとき周辺部は１２８になります。

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_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	○	△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
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。

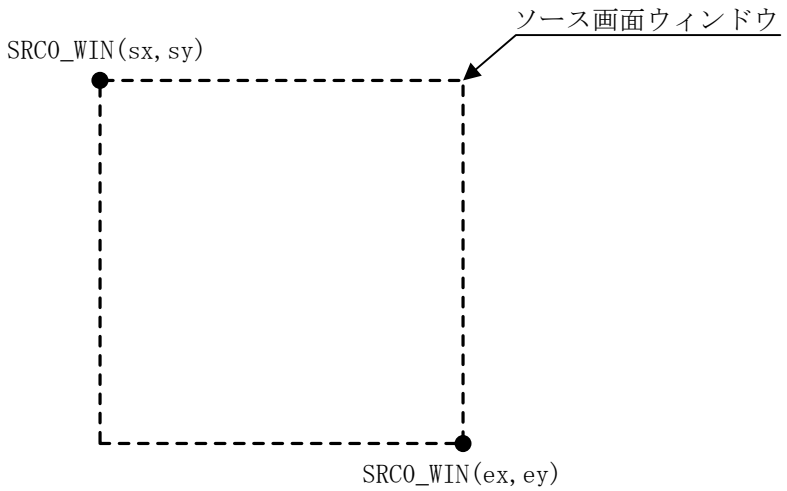
❑ 機能

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。

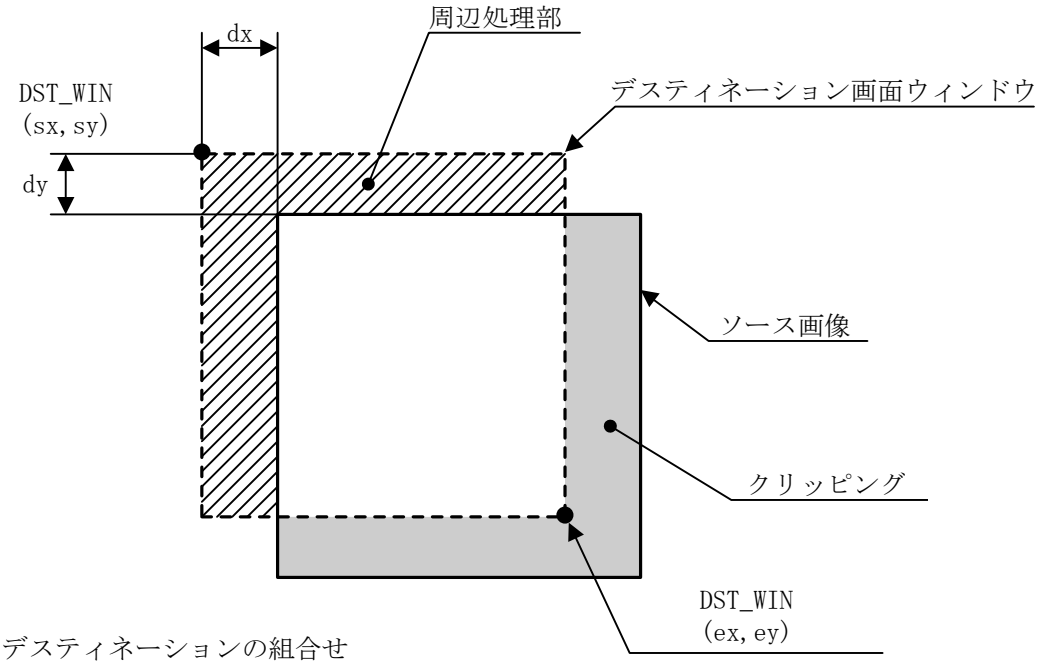
画面サイズもしくはデスティネーションのウインドウ画面サイズ以上のシフト量を設定した場合にはエラーとなります。

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_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
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	回転角度設定範囲外
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。

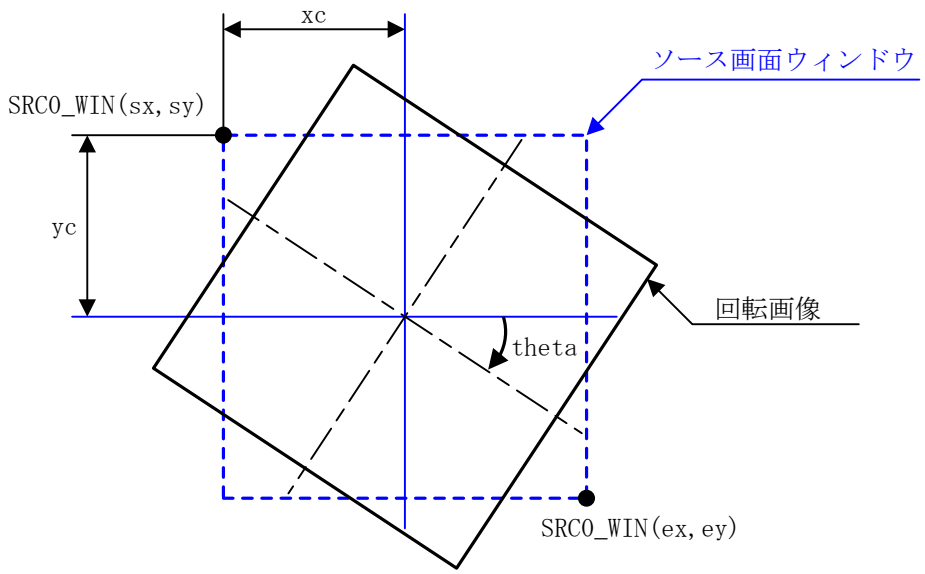
❑ 機能

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。

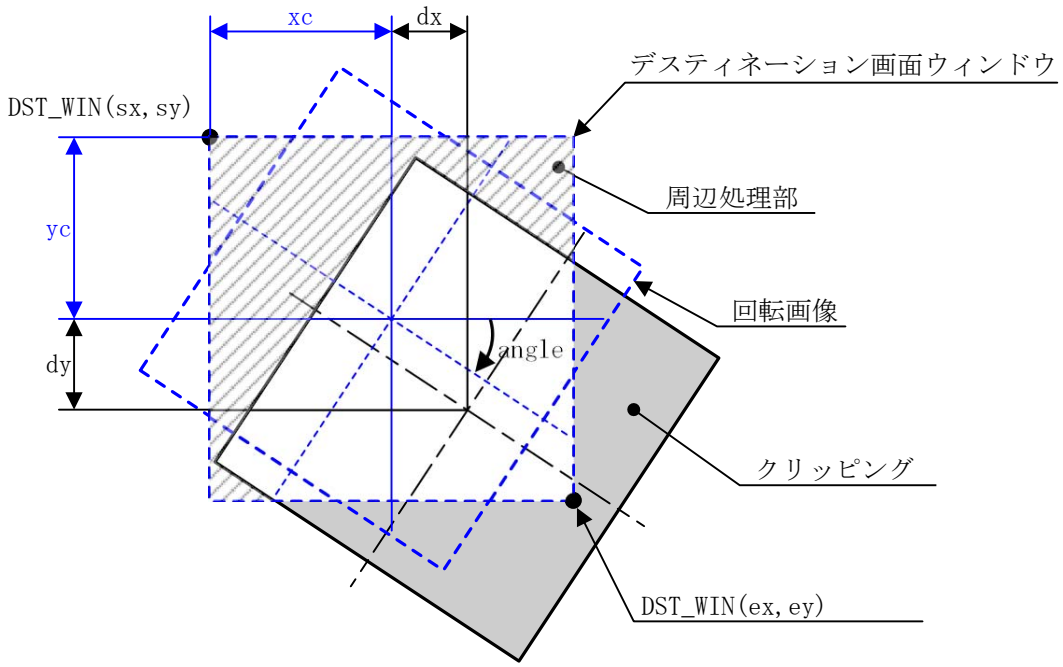
本機能はソフトウェアにより処理します。実際にはデスティネーション画面ウィンドウ範囲内の座標についてソース画面の座標を算出し、算出座標がソース画面ウィンドウ範囲内であればソース画面データをデスティネーション画面に書き込みます。ソース画面座標の小数点以下は整数型への型変換による切り捨て処理を行います。

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 ≧ 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 ≧ ソース画面のウィンドウ長×変換倍率
「ソース画面のウィンドウ長×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×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
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

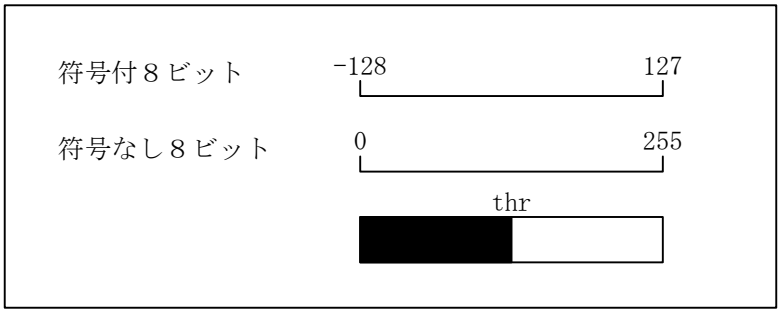
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 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
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	しきい値最小値範囲設定エラー
２３	しきい値最大値範囲設定エラー
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
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

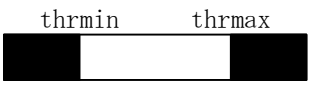
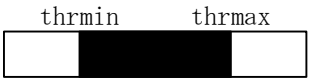
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 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		白
1		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
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 = 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
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
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（－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
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
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 ＋ 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
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
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 － 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（－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
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
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 － 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
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 × 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（－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
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
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）。
定数には－２５６～２５５の値が設定できますが、定数値が－２５６～－１２９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－１２８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 ＝ MIN (f , 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（－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
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
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）。
定数には－２５６～２５５の値が設定できますが、定数値が－２５６～－１２９の範囲では、結果はソース画面になります。

演算式	F ＝ MAX (f , 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
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

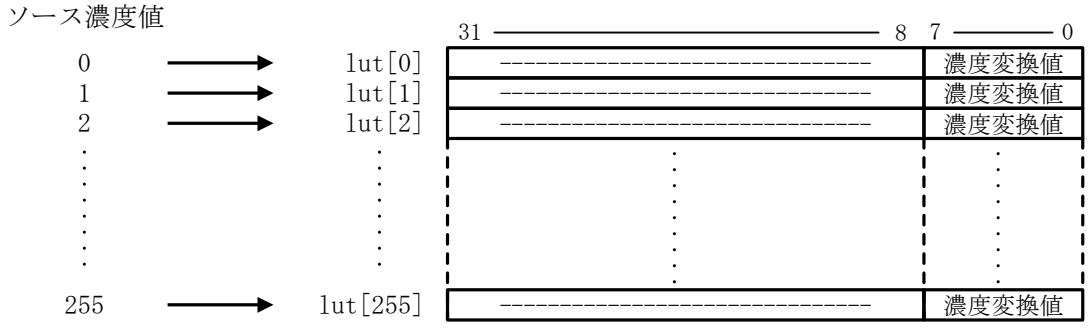
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 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８bitのみ有効で、上位２４bit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
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

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
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

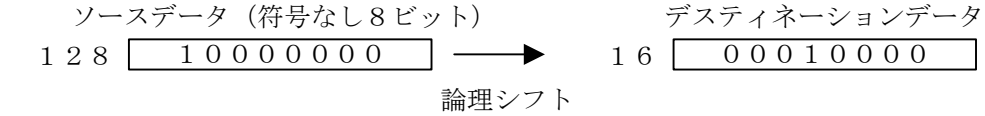
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

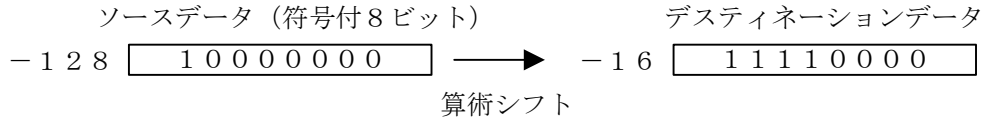
❑ 機能

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





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
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	シフトアップ量設定値範囲外
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

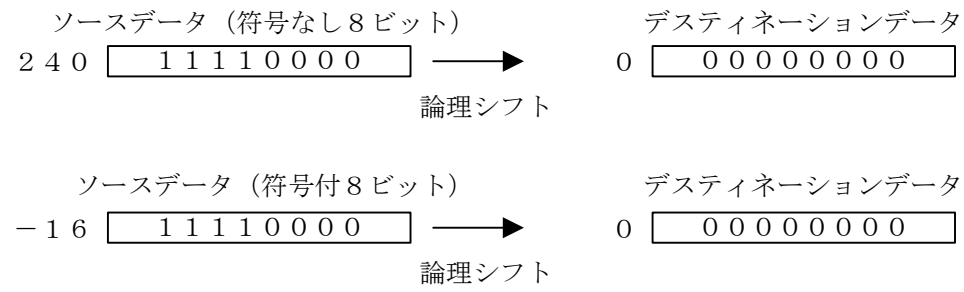
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 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



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
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）
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 ＋ 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
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）
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 － 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
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）
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 － 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（０～１５）
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
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。
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 × 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

2.9.5 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
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 × 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

2.9.6 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
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
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／－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 × 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
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）
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 ＋ 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
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）
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 ＝ 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
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）
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 ＝ 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
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
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）×（ｆ － ａ） ＋ （ｇ － ｂ）×（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
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 － 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

2.9.13　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
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 ＋ 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

2.10 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
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）
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 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
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）
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 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
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）
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 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
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）
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 ＝ ~f 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
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）
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 ＝ ~f 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
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）
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 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
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

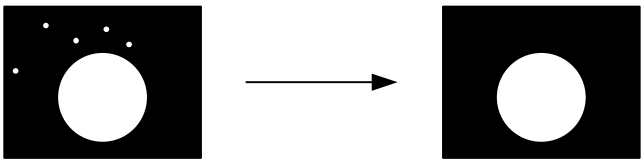
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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。

４連結と８連結の処理内容を次ページに示します。

<table>
<tr><th></th><th>４連結</th><th>８連結</th></tr>
<tr><td>処理内容</td><td>処理前 → 処理後
<table><tr><td></td><td>1</td><td></td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td></td><td>0</td><td></td></tr></table> → <table><tr><td></td><td>1</td><td></td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td></td><td>0</td><td></td></tr></table>
<table><tr><td></td><td>0</td><td></td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td></td><td>0</td><td></td></tr></table> → <table><tr><td></td><td>0</td><td></td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td></td><td>0</td><td></td></tr></table>
<table><tr><td></td><td>1</td><td></td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td></td><td>1</td><td></td></tr></table> → <table><tr><td></td><td>1</td><td></td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td></td><td>1</td><td></td></tr></table>
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。</td><td>処理前 → 処理後
<table><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table> → <table><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table>
<table><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table> → <table><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table>
<table><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table> → <table><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>
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。</td></tr>
<tr><td>処理例</td><td colspan="2"> 
白の独立点除去</td></tr>
</table>

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
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

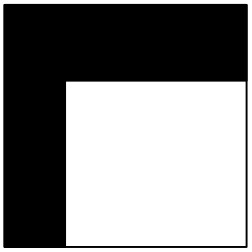
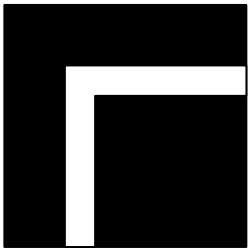
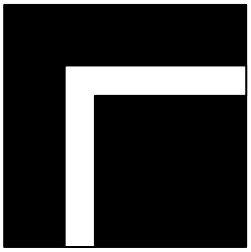
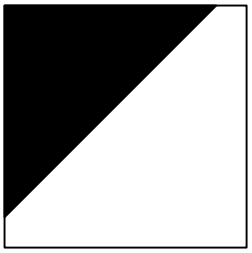
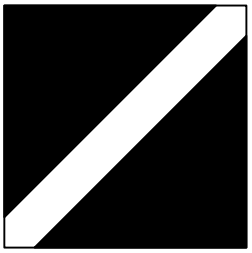
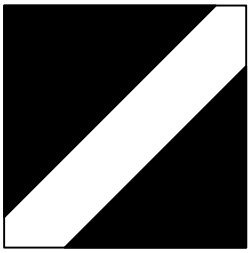
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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。

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	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 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 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 
処理例２	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 	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 	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 

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
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

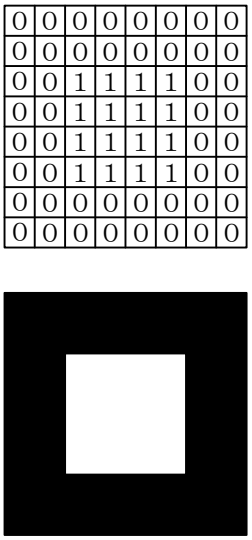
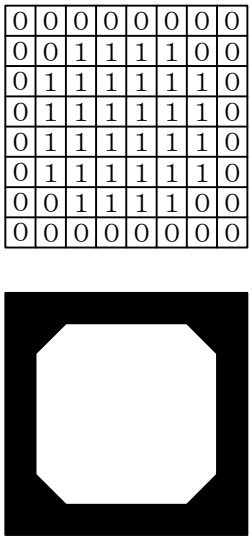
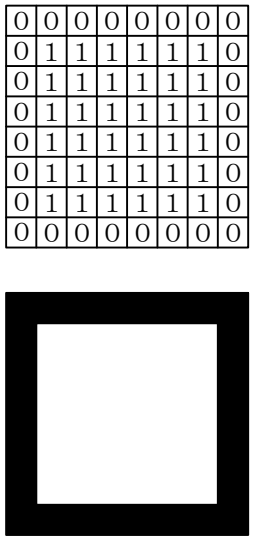
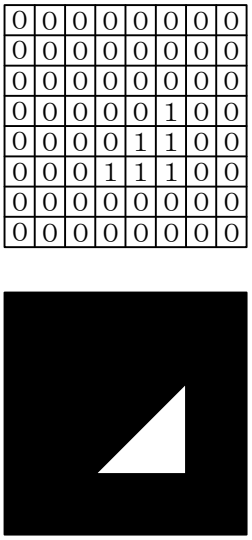
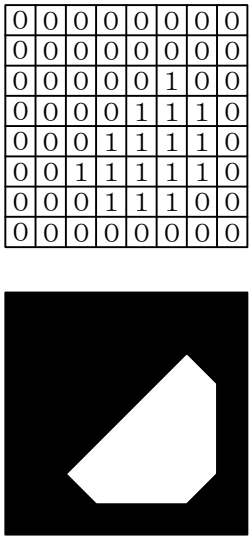
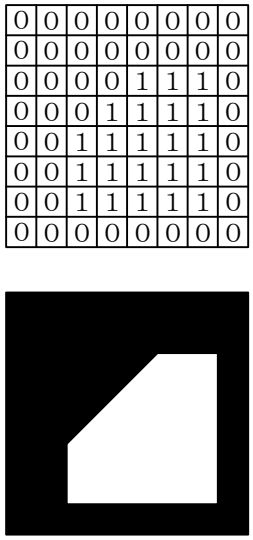
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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。

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			
処理例2			

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
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

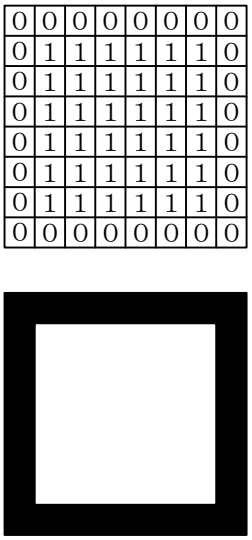
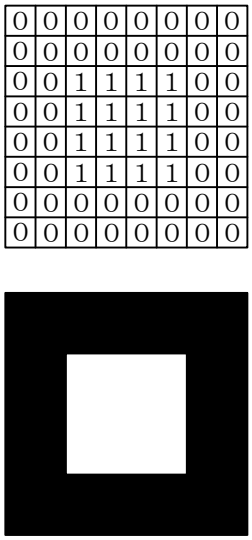
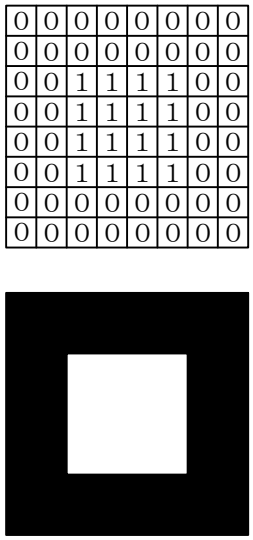
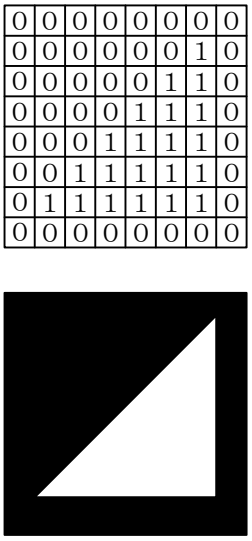
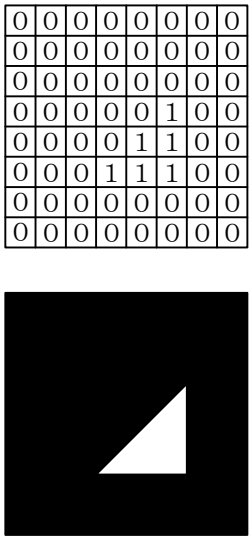
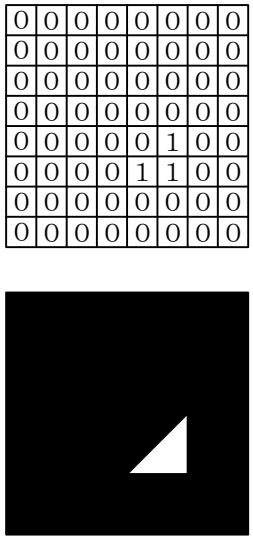
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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。

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			
処理例２			

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
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）

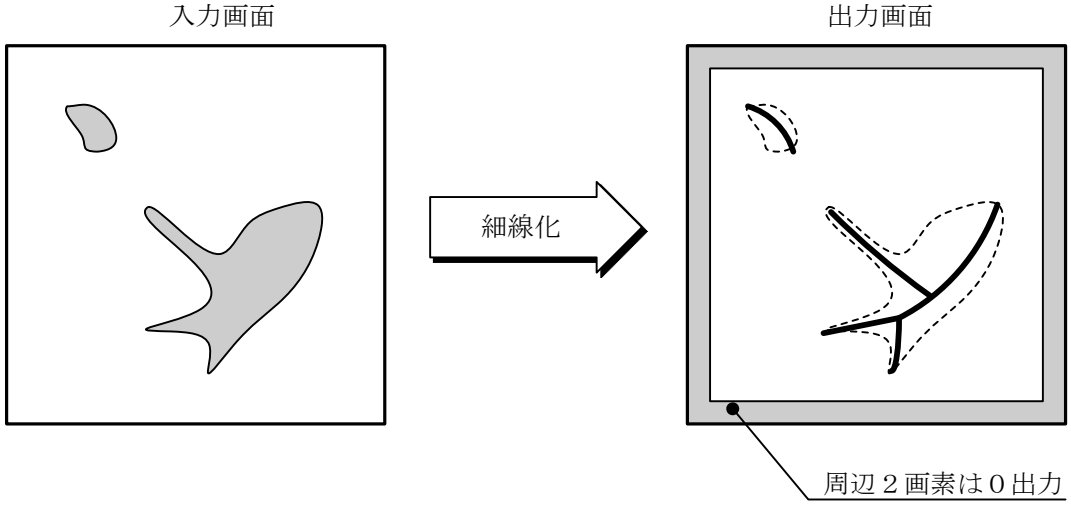
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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

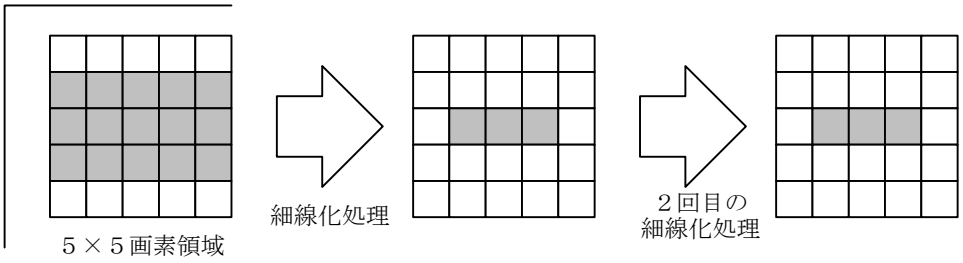
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 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
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）

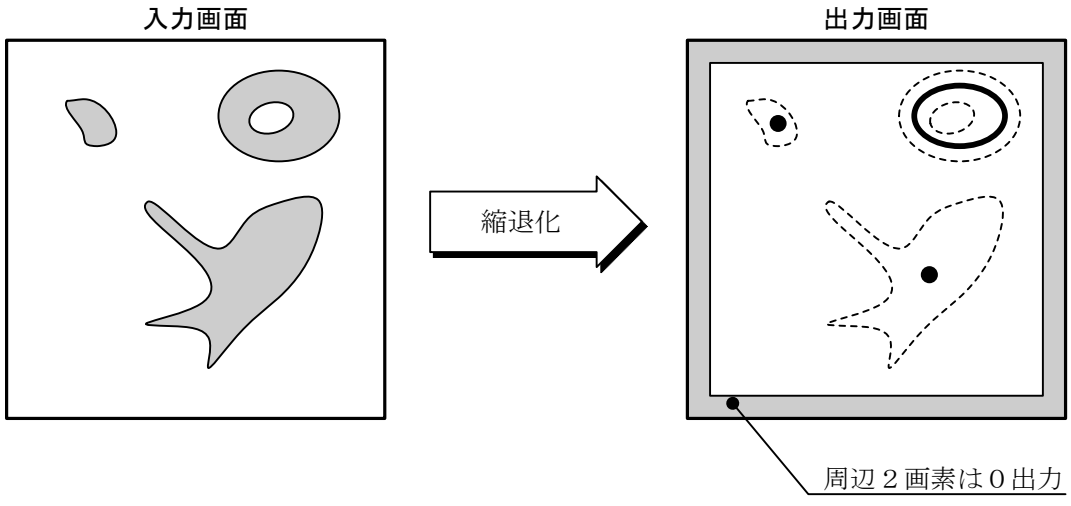
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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

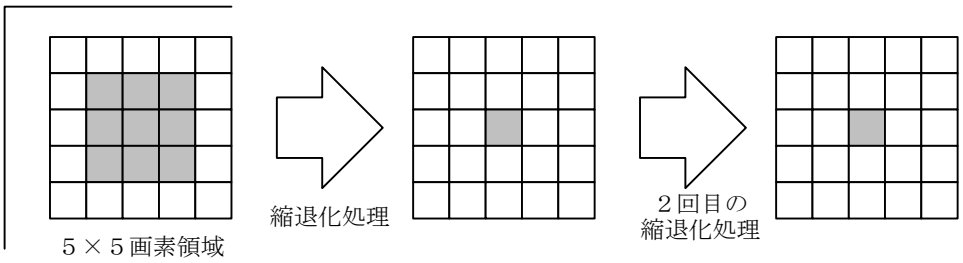
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

２値画像に対し、8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。
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化処理を行います。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。また、縮退化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_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
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。

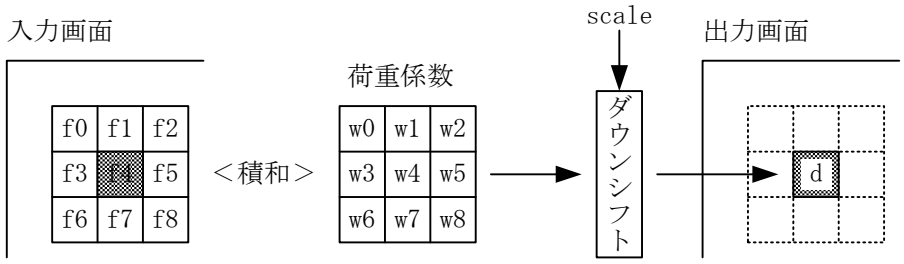
❑ 機能

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。

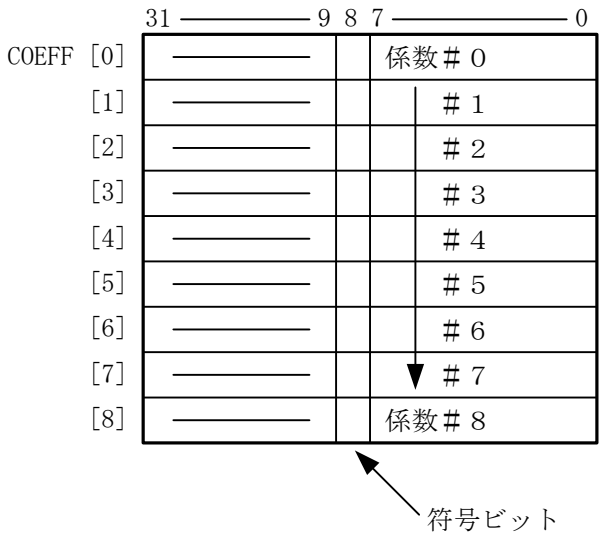



```
d = ( f0 × w0 + f1 × w1 + f2 × w2 + f3 × w3 + f4 × w4
              + f5 × w5 + f6 × w6 + f7 × w7 + f8 × w8 ) >> scale
```

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

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、ダウンシフト量は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
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。

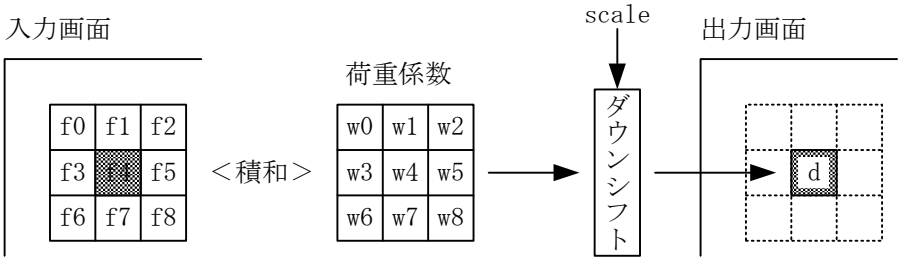
❑ 機能

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。



$$d = (f0 \times w0 + f1 \times w1 + f2 \times w2 + f3 \times w3 + f4 \times w4 + f5 \times w5 + f6 \times w6 + f7 \times w7 + f8 \times w8) \gg scale$$

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
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。

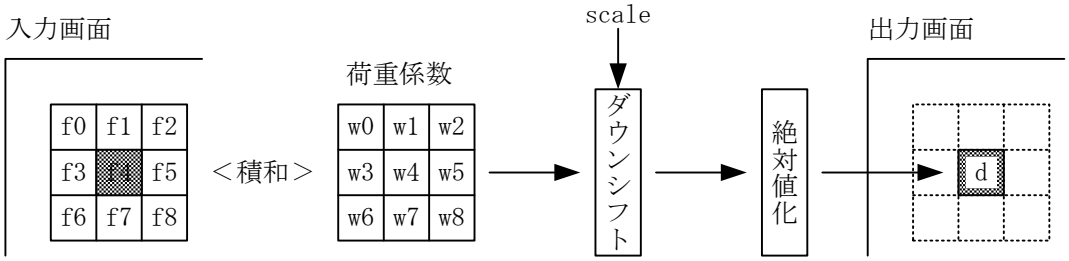
❑ 機能

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１２７、符号なし８ビットの場合は２５５を出力します。



d = | (f0 × w0 + f1 × w1 + f2 × w2 + f3 × w3 + f4 × w4
+ f5 × w5 + f6 × w6 + f7 × w7 + f8 × w8) >> 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
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

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
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

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
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

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
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

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
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。

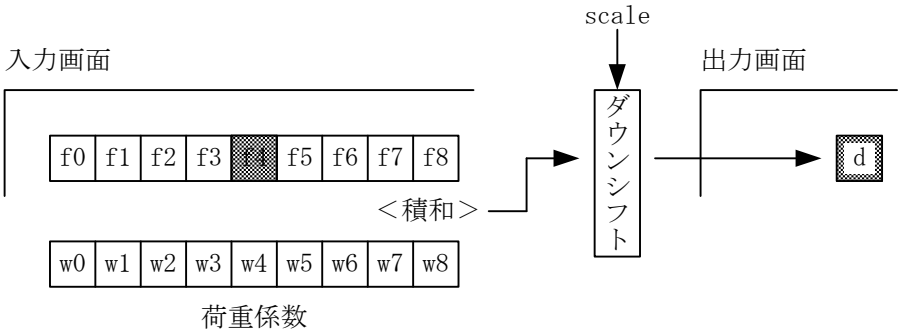
❑ 機能

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 × w0 + f1 × w1 + f2 × w2 + f3 × w3 + f4 × w4
            + f5 × w5 + f6 × w6 + f7 × w7 + f8 × w8 ) >> 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。

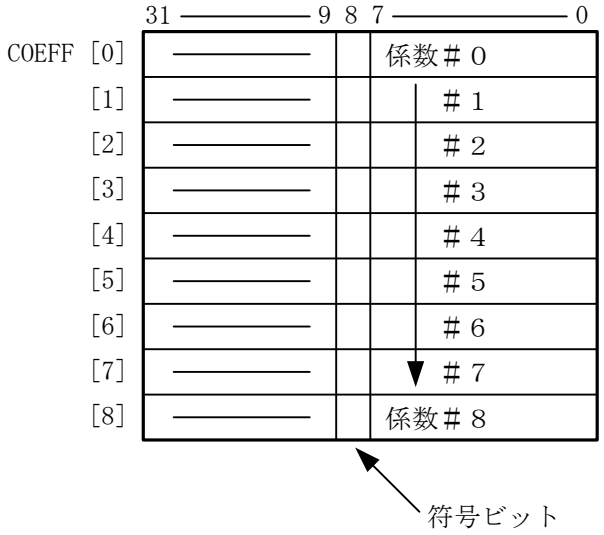
荷重係数

#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、ダウンシフト量は６です。



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
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。

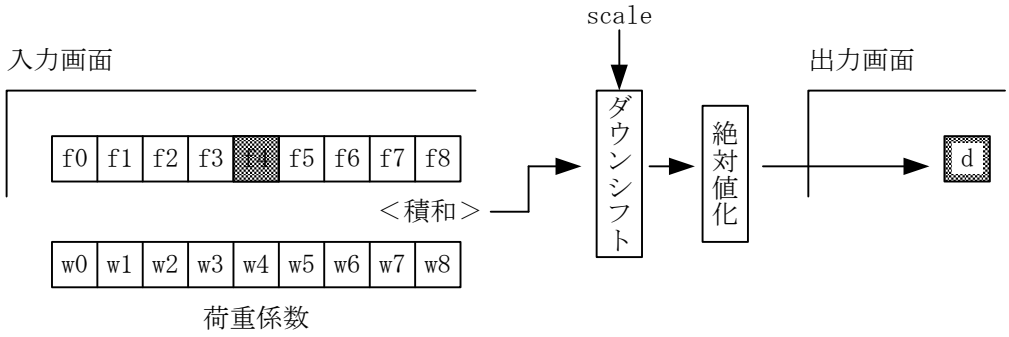
❑ 機能

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１２７、符号なし８ビットの場合は２５５を出力します。




```
d = |( f0 × w0 + f1 × w1 + f2 × w2 + f3 × w3 + f4 × w4
              + f5 × w5 + f6 × w6 + f7 × w7 + f8 × w8 ) >> 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

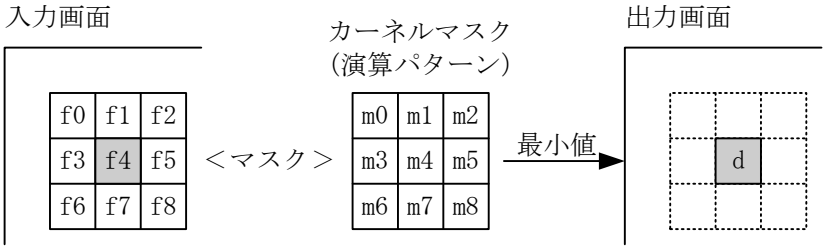
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

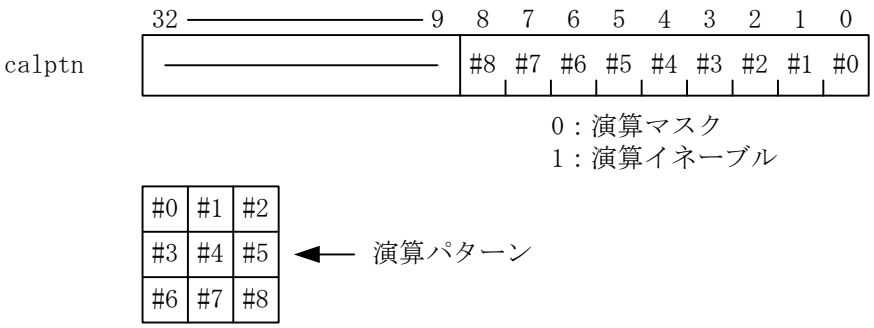
ソース画像に対し、パラメータで指定されたパターンで最小濃度値を見つけ、デスティネーション画面に出力します。局所最小値フィルタでは、入力画面の指定領域内の全画素に対して、注目画素を中心とした３×３近傍の局所領域を与えられた演算パターンでマスク処理し、有効となった画素の中から最小の画素を出力します。指定領域の周辺(内側1画素の領域)は、ソース画面の画素をそのまま出力します。

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。



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

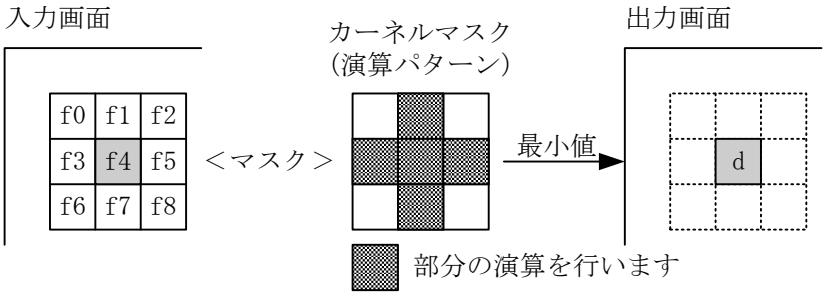
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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

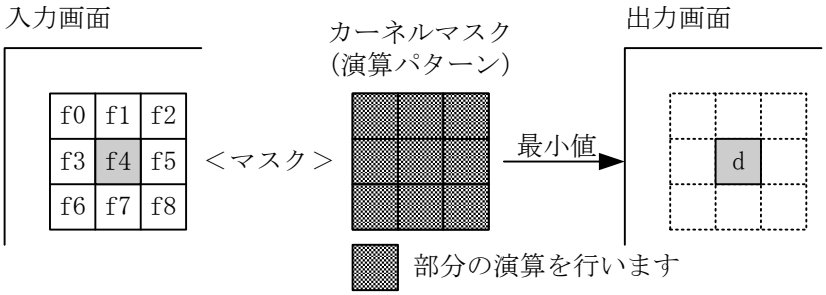
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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
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

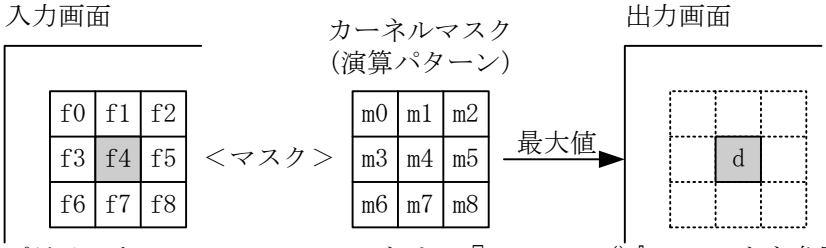
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 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

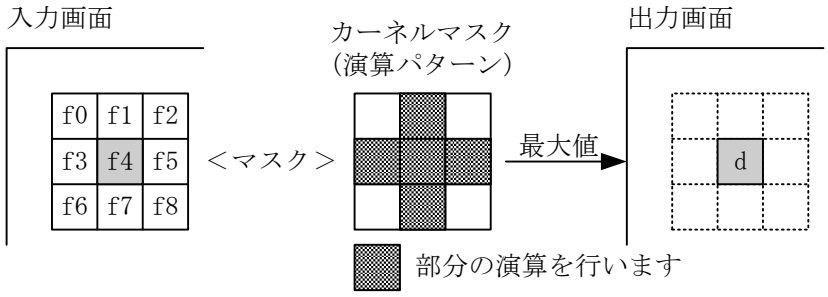
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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

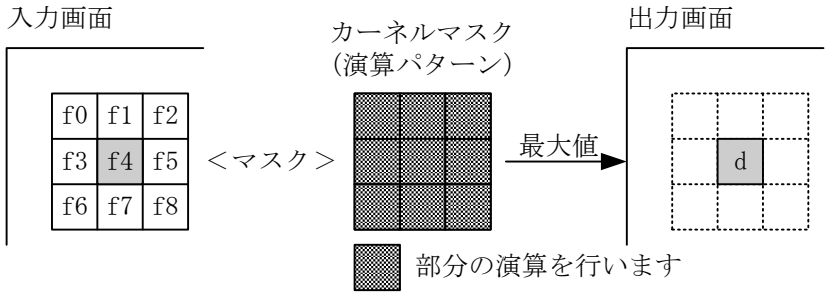
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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
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

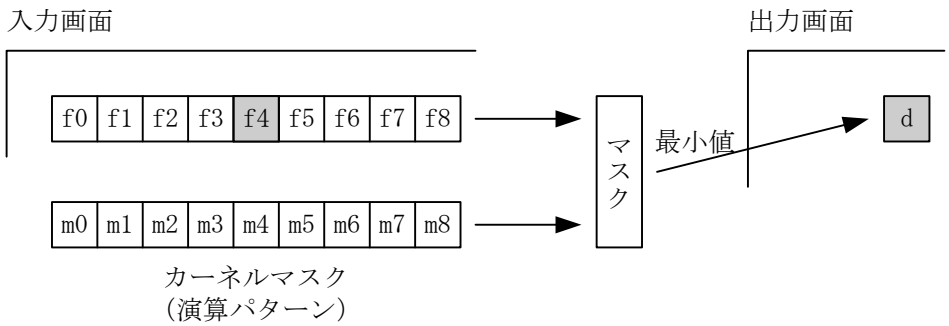
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

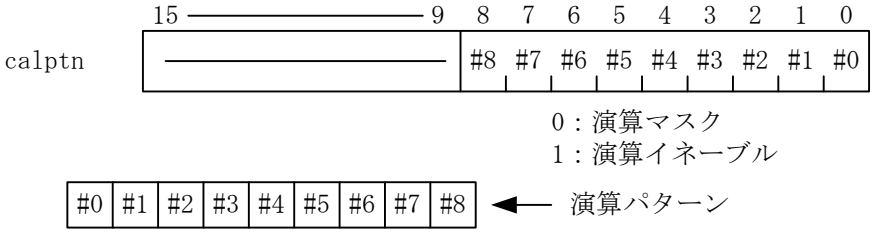
ソース画像に対し、パラメータで指定されたパターンで最小濃度値を見つけ、デスティネーション画面に出力します。ライン局所最小値フィルタでは、入力画面の指定領域内の全画素に対して、注目画素を中心とした１×９近傍の局所領域を与えられた演算パターンでマスク処理し、有効となった画素の中から最小の画素を出力します。指定領域の周辺(内側1画素の領域)は、ソース画面の画素をそのまま出力します。

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

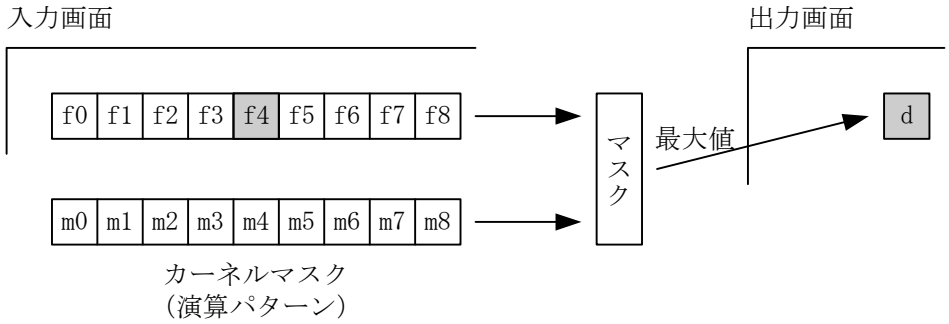
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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）

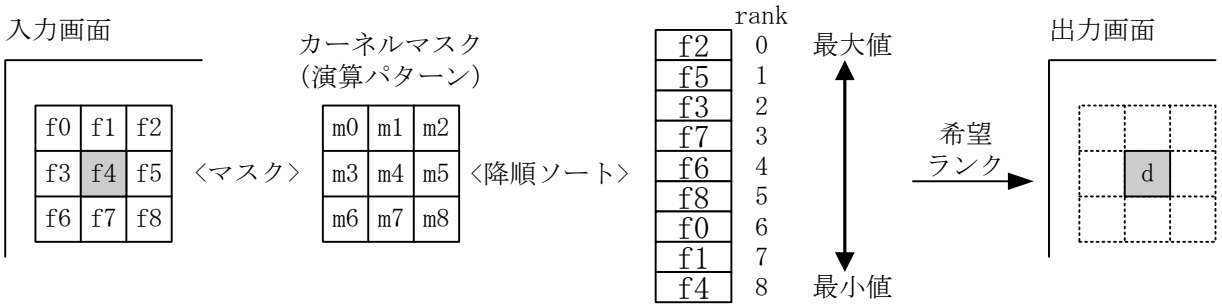
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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）
２３	ランク指定範囲外
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

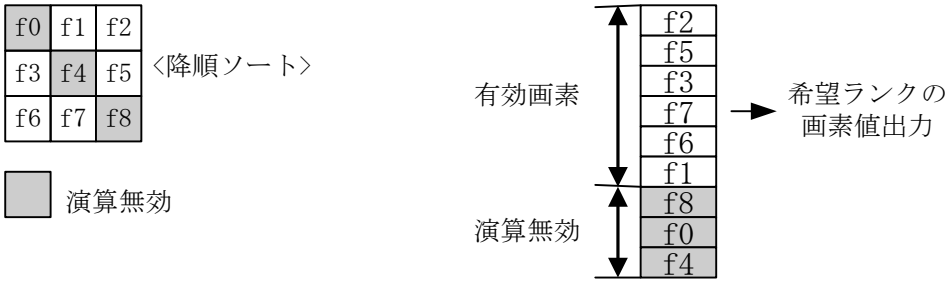
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

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）。

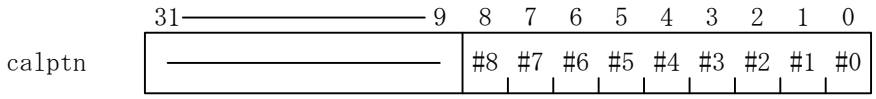


注）マスクされた画素の値は最小値として扱われます。したがって、ランク値を指定するときは、最大値（ランク０）からの順番を指定して下さい。calptn=0を指定した場合や指定されたランクがcalptnマスク後の有効画素の範囲を超えている場合は、周辺部を除いてシステムデータタイプが符号なし８ビットの場合には０、システムデータタイプが符号付８ビットの場合には－１２８が出力されます。周辺部については周辺処理が優先します。



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）

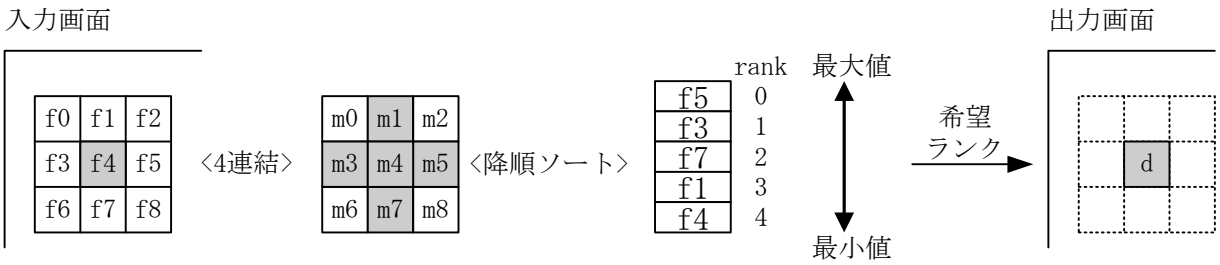
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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	ランク指定範囲外
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

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）

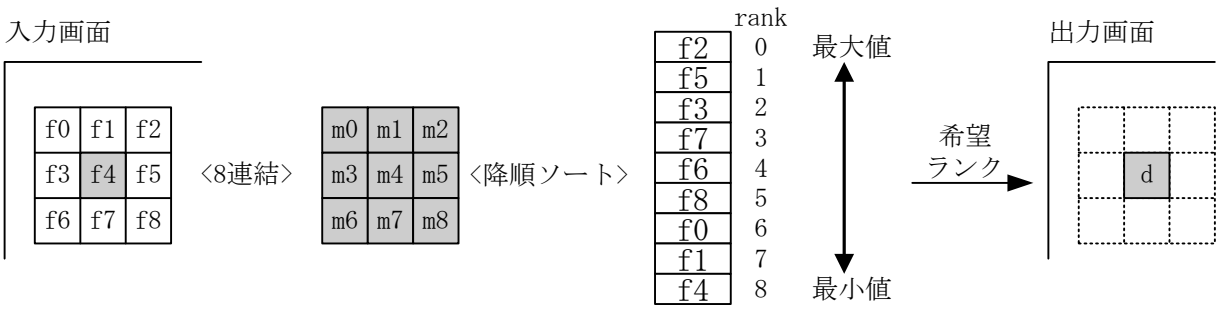
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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	ランク指定範囲外
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

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	ランク０（rank = 0）
中間値の場合	ランク４（rank = 4）
最小値の場合	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

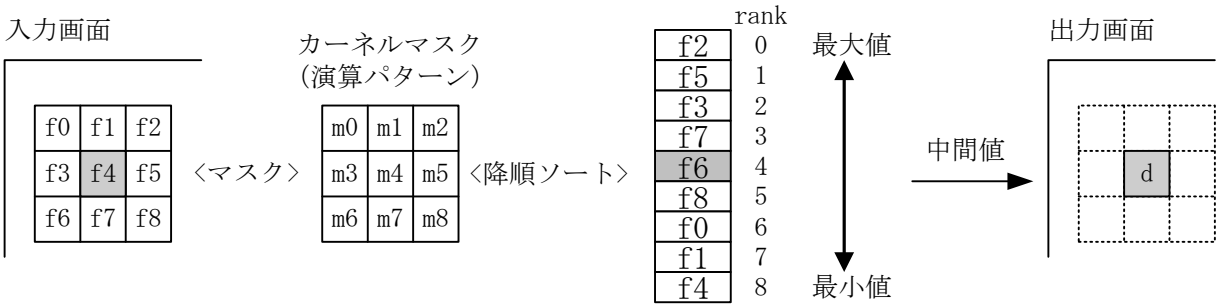
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

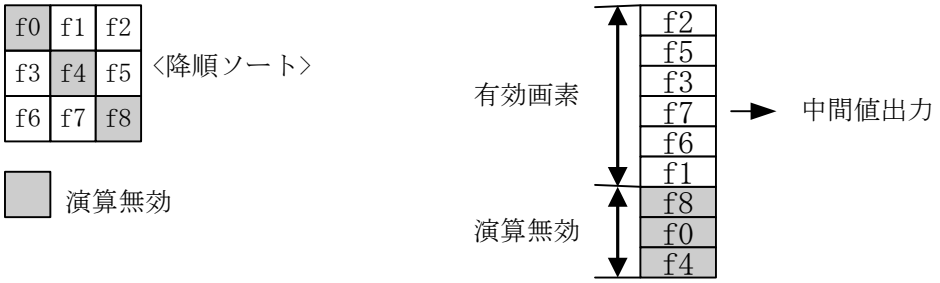
❑ 機能

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。

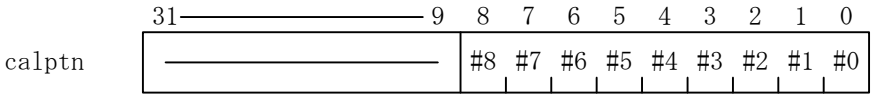
また、ソース画面データタイプが２値の場合は、符号付８ビットに変換して処理します（２５５は－１として処理します）。



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：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)

2.14.5　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

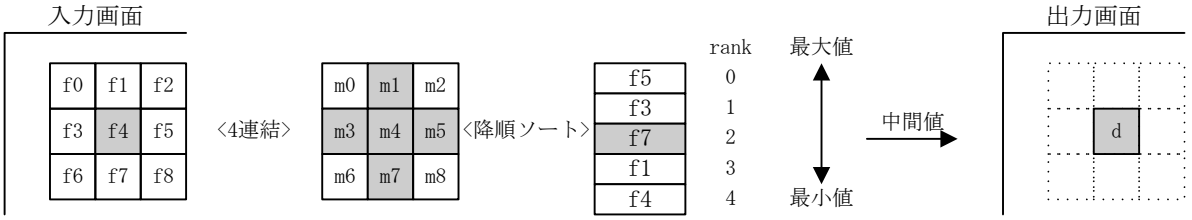
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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、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

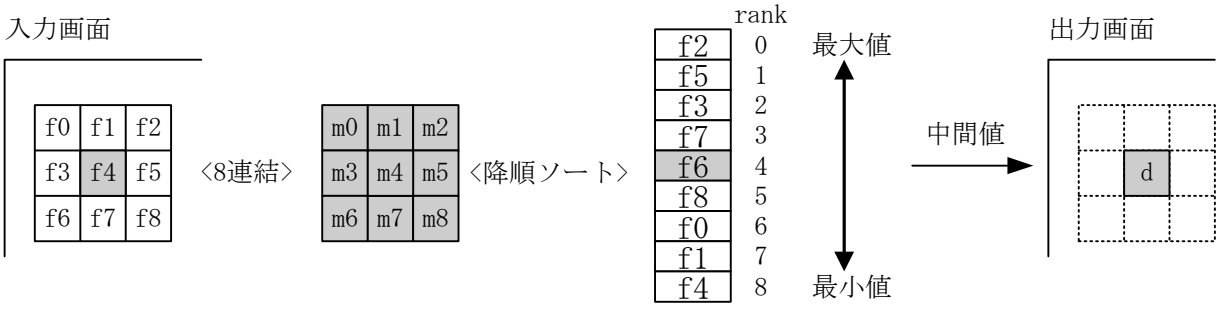
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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（０ｘ７ＦＦＦ）

❑ 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
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	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

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４０９５個以上、合流対が２０４７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_Label8()』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（０ｘ７ＦＦＦ）

❑ エラーコード

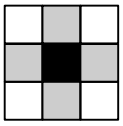
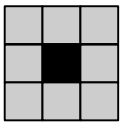
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
２２	オプション設定値範囲外
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４８	動的メモリ確保エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

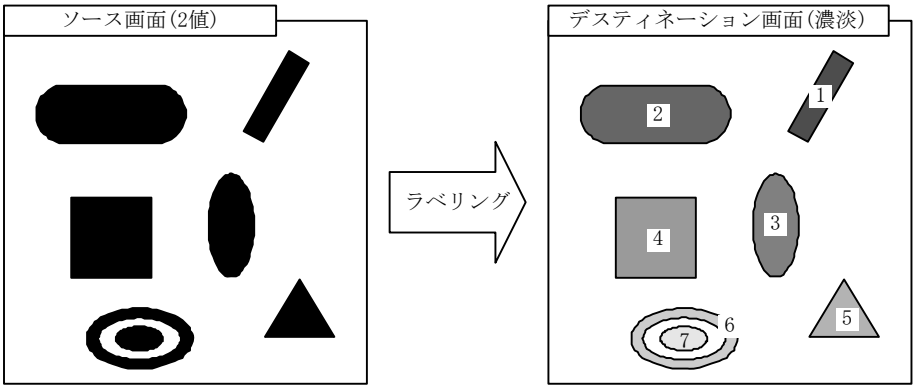
❑ 機能

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４０９５個以上、合流対が２０４７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＞



⟶ ラスタスキャン方向

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。

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（０ｘ７ＦＦＦ）

❑ エラーコード

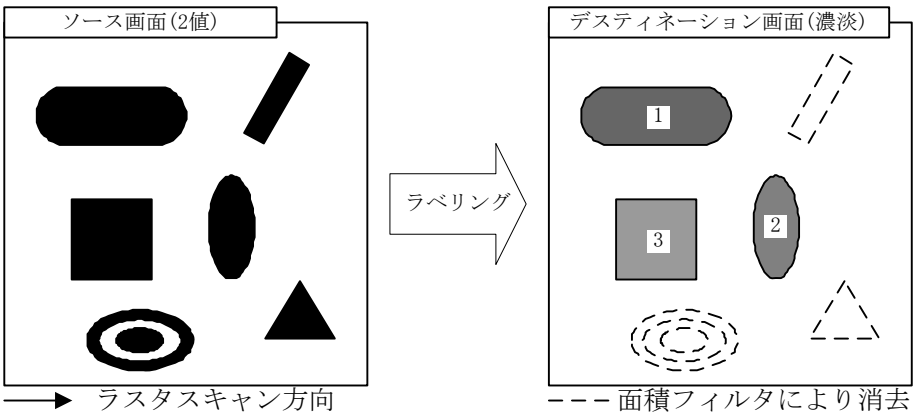
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
２２	面積フィルタ最小しきい値範囲設定エラー
２３	面積フィルタ最大しきい値範囲設定エラー
２４	オプション設定値範囲外
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４８	動的メモリ確保エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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、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４０９５個以上、合流対が２０４７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４０９５を超えるとラベルオーバ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thrmin≦面積≦thrmaxのラベルに対し、左上から順に番号が付けられます。

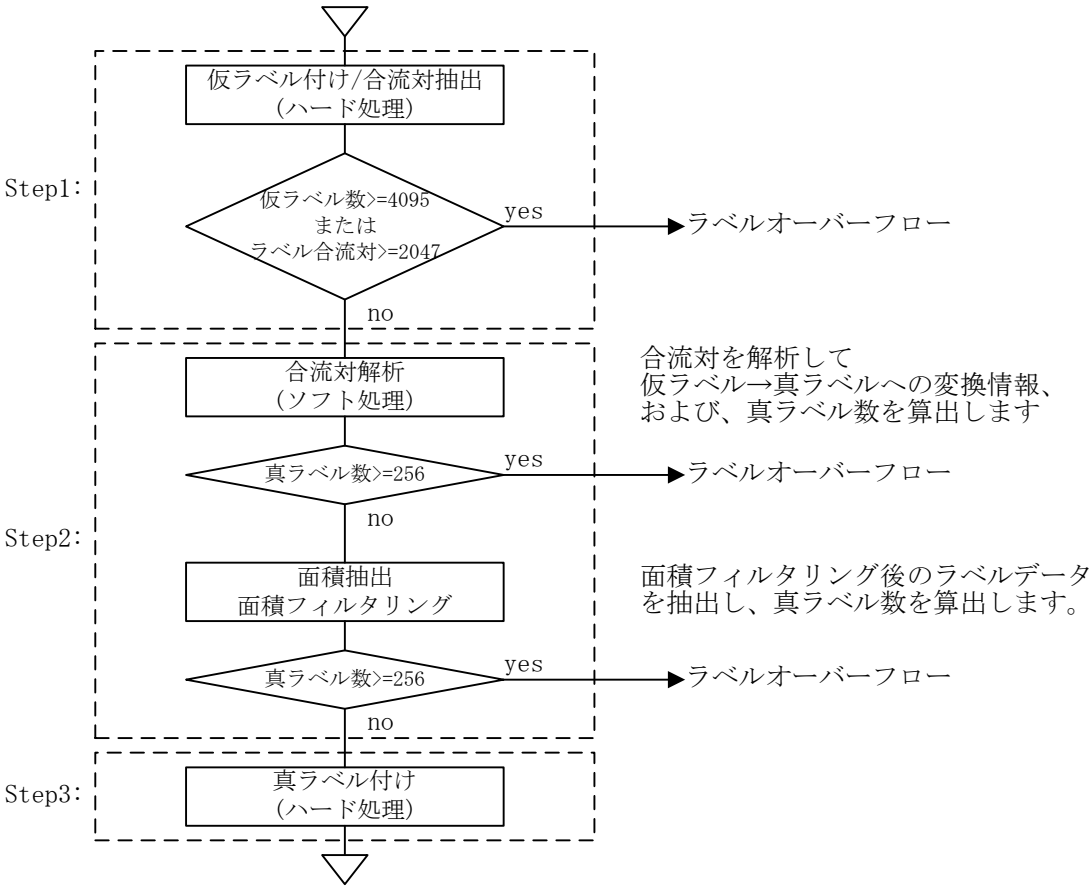
＜ラベル付け例＞



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（０ｘ７ＦＦＦ）

❑ 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
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	面積フィルタ最小しきい値範囲設定エラー
２３	面積フィルタ最大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４０９５個以上、合流対が２０４７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４０９５を超えるとラベルオーバ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、thrmin≦面積≦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。

2.15.5　面積フィルタ付きラベリング（ 面積ソート4連結 ）（ IP_Label4withAreaFLTSort ）

❑ C言語API

```
int ret = IP_Label4withAreaFLTSort(IMGID ImgSrc, IMGID ImgDst, long thrmin, long thrmax,
                        enum IP_Label_opt opt1, enum IP_Label_Sort_opt opt2);
```

❑ 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_opt　opt1　ラベリング処理オプション１

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　opt2　ラベリング処理オプション２

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
LABEL_SORT_NON	（２）	ソートなし（未使用）

❑ 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（０ｘ７ＦＦＦ）

❑ 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
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	面積フィルタ最小しきい値範囲設定エラー
２３	面積フィルタ最大しきい値範囲設定エラー
２４	オプション１設定値範囲外
２５	オプション２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

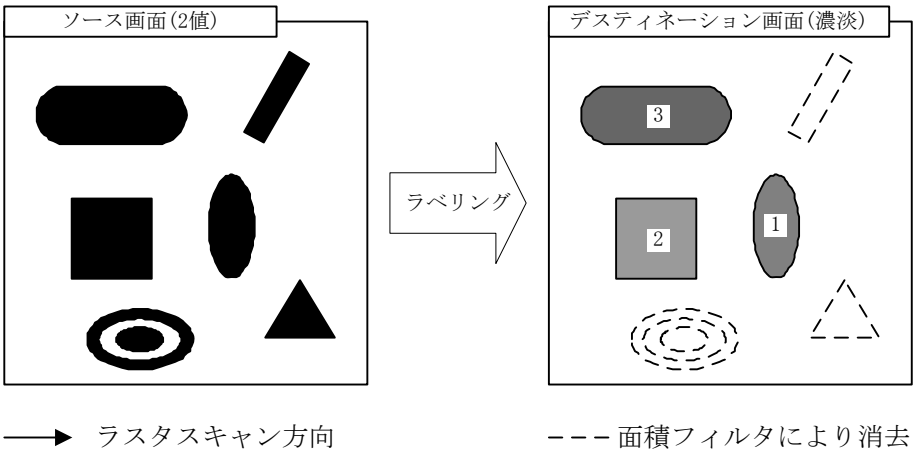
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

ソース画面（２値）に対し、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４０９５個以上、合流対が２０４７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４０９５を超えるとラベルオーバ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thrmin≦面積≦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。

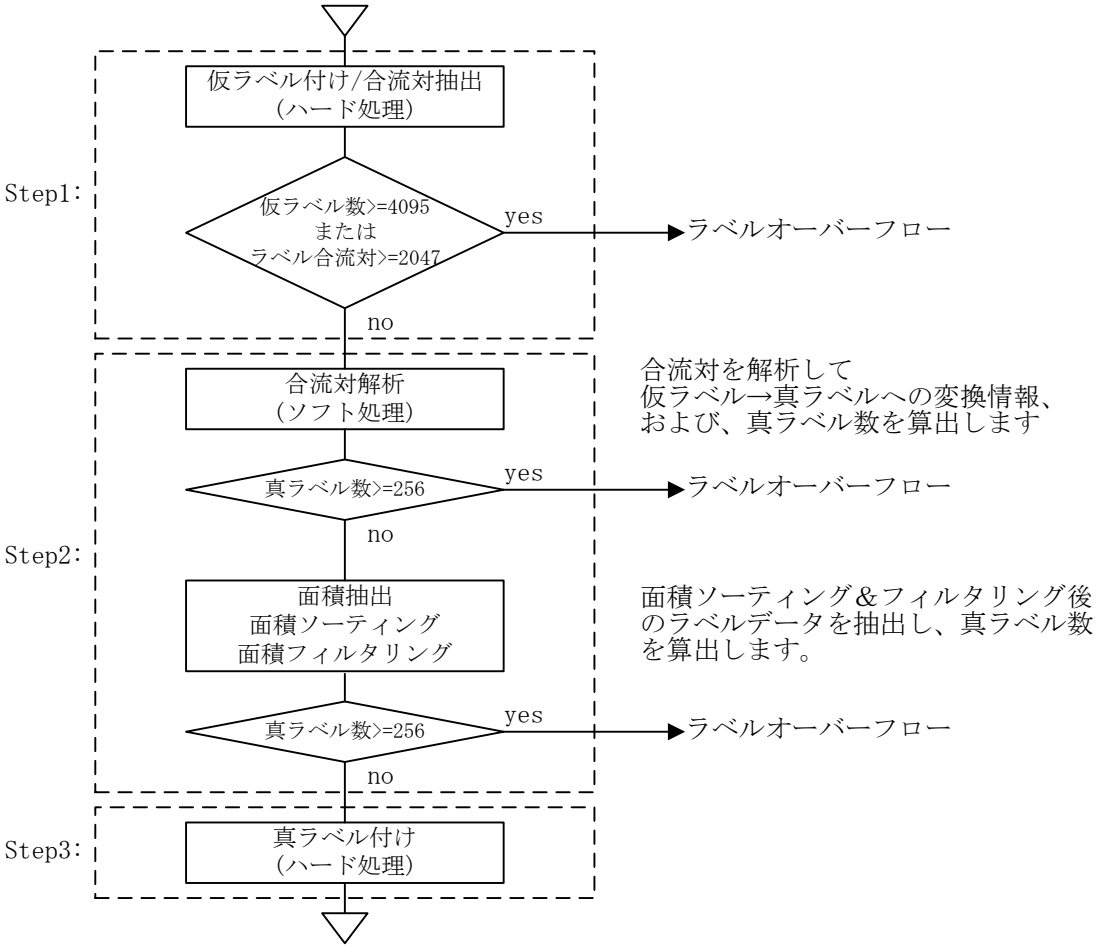
＜ラベル付け例：面積が小さい順にソート＞



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＞



2.15.6　面積フィルタ付きラベリング（ 面積ソート8連結 ）（ IP_Label8withAreaFLTSort ）

❑ C言語API

```
int ret = IP_Label8withAreaFLTSort(IMGID ImgSrc, IMGID ImgDst, long thrmin, long thrmax,
                        enum IP_Label_opt opt1, enum IP_Label_Sort_opt opt2);
```

❑ 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_opt　opt1　ラベリング処理オプション１

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　opt2　ラベリング処理オプション２

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
LABEL_SORT_NON	（２）	ソートなし（未使用）

❑ 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（０ｘ７ＦＦＦ）	

❑ 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
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	面積フィルタ最小しきい値範囲設定エラー
２３	面積フィルタ最大しきい値範囲設定エラー
２４	オプション１設定値範囲外
２５	オプション２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４０９５個以上、合流対が２０４７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４０９５を超えるとラベルオーバ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thrmin≦面積≦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

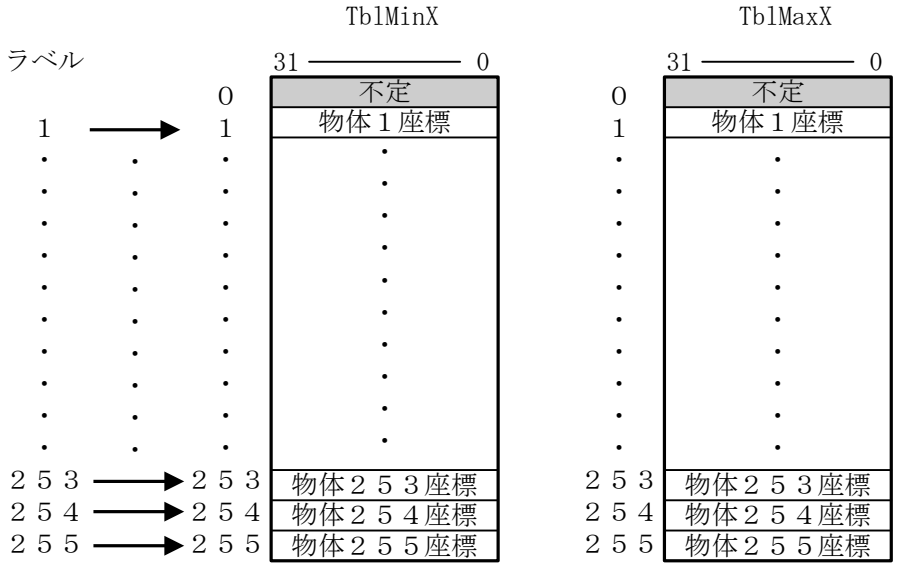
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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、

```
最小Ｘ座標 ＝ TblMinX［ラベル番号］ ＋ sx
最大Ｘ座標 ＝ TblMaxX［ラベル番号］ ＋ sx
```

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

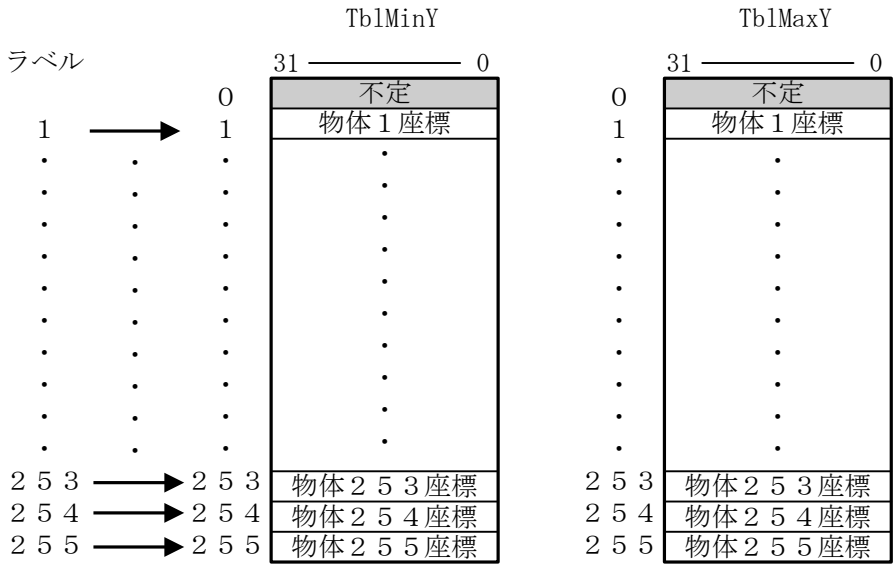
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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、

```
最小Ｙ座標 ＝ TblMinY［ラベル番号］ ＋ sy
最大Ｙ座標 ＝ TblMaxY［ラベル番号］ ＋ sy
```

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

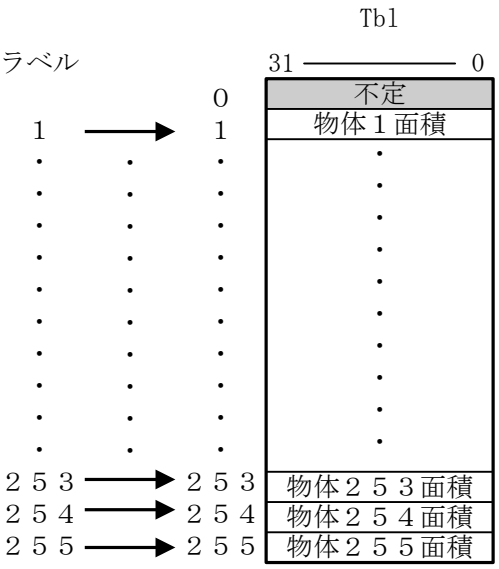
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
２１	面積格納テーブルがアドレス不正
３０	ソース画面ウィンドウ設定エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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, IPLOGravityTbl *Tbl);
```

❑ パラメータ

IMGID	ImgSrc	ソース画面番号
IPLOGravityTbl	*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IPLOGravityTbl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）
IPLOGravityTbl	*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
} IPLOGravityTbl;
```

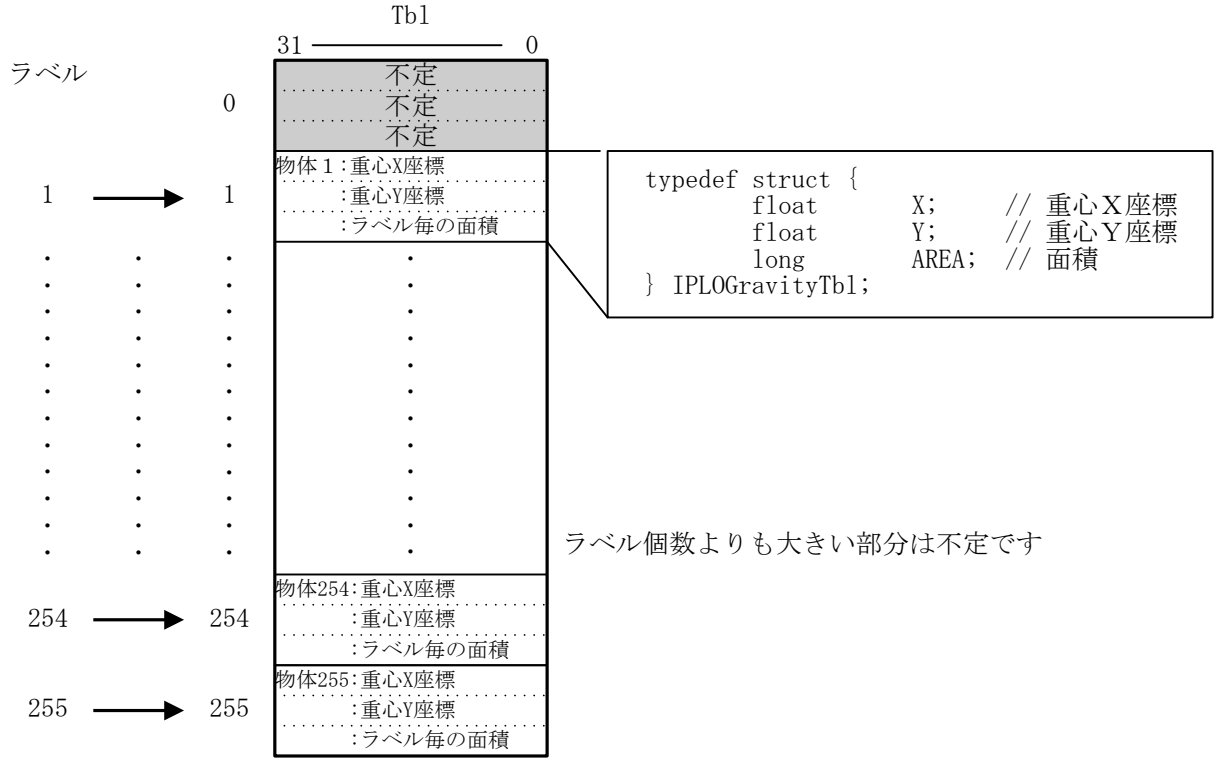
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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;
```

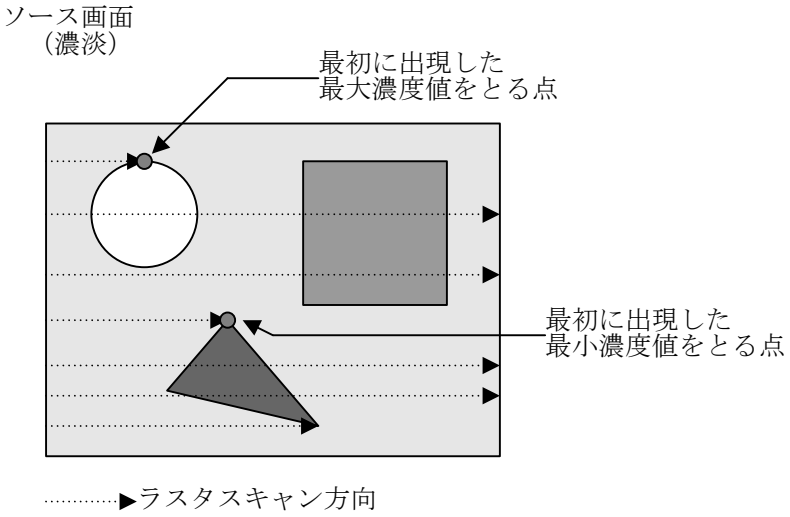
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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, IPGOFeatureTbl *FtrTbl, int opt);
int ret = IP_HistogramShort(IMGID ImgSrc, int *Tbl, IPGOFeatureTbl *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。
IPGOFeatureTbl	*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
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

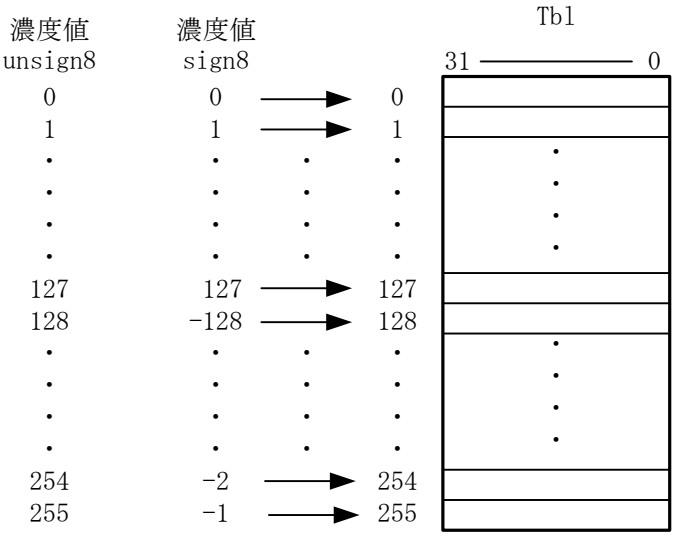
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
２１	濃度頻度分布格納アドレス不正
２２	基本特徴量格納アドレス不正
２３	ヒストグラム処理オプション不正
３０	ソース画面ウィンドウ設定エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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）『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_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

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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）
２２	ヒストグラム特徴量を格納アドレス不正
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

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。

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

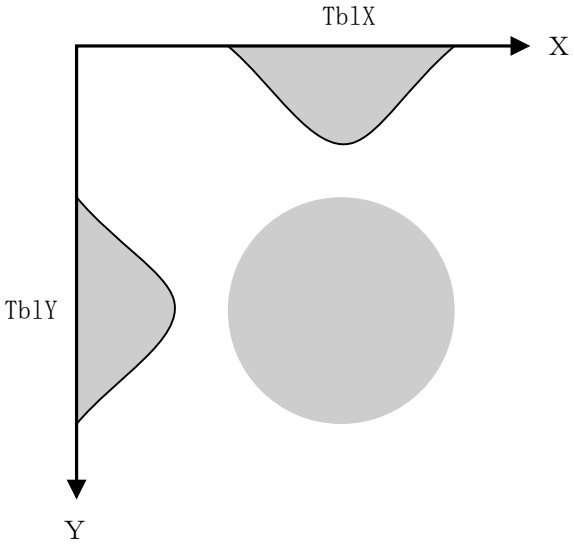
❑ エラーコード

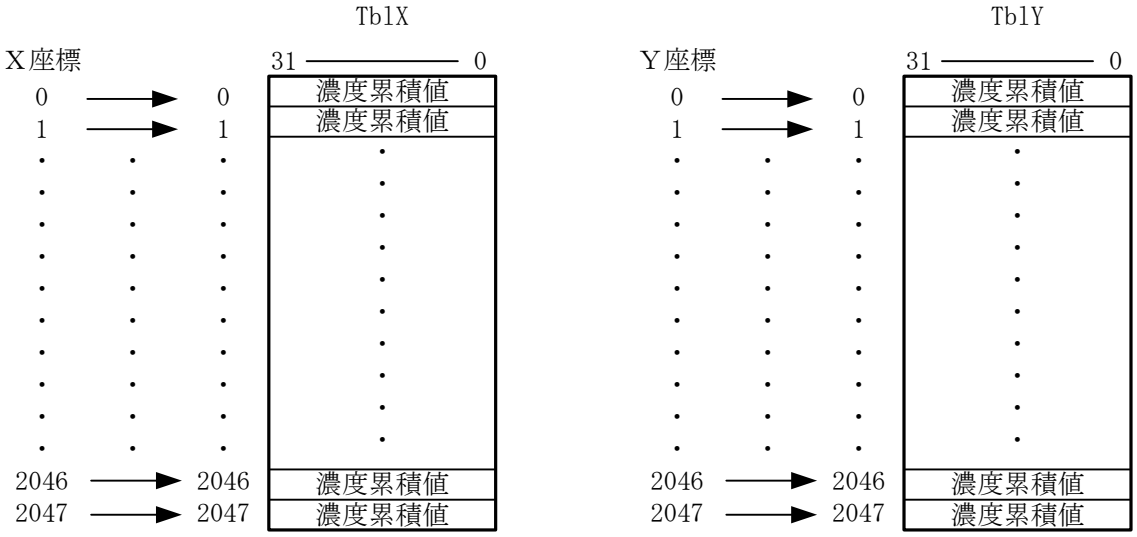
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
２１	濃度累積値（X軸への投影）格納アドレス不正
２２	濃度累積値（Y軸への投影）格納アドレス不正
２３	基本特徴量格納アドレス不正
３０	ソース画面ウィンドウ設定エラー
４８	動的メモリ確保エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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。





（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、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それぞれ、

```
X軸濃度累積値 ＝ TblX [ X - sx ]
Y軸濃度累積値 ＝ TblY [ Y - sy ]
```

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で確保して下さい。
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

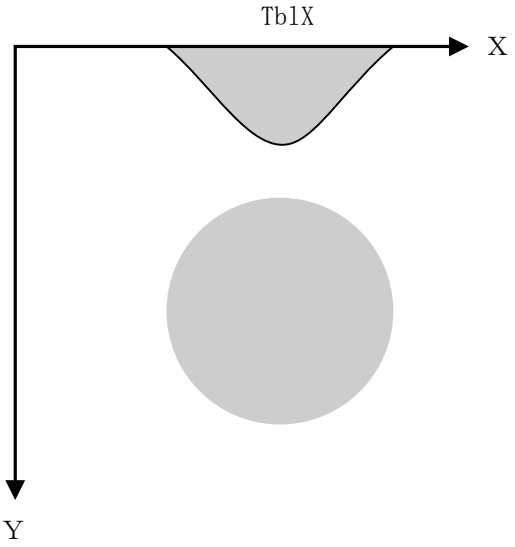
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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	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
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

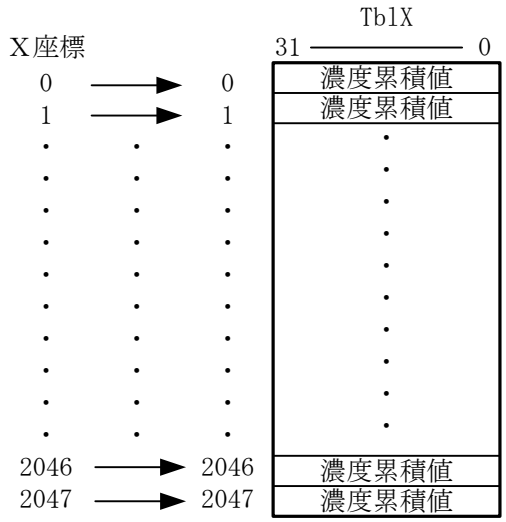
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 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。



(注)ウイ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、

X軸濃度累積値 ＝ TblX [X - 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で確保して下さい。
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

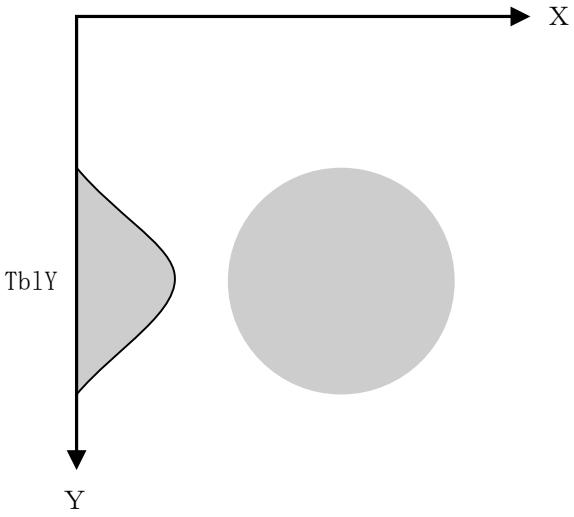
❑ エラーコード

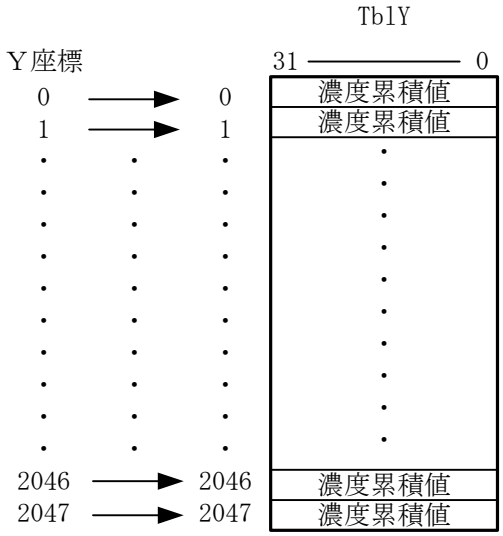
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
２１	濃度累積値（Y軸への投影）格納アドレス不正
２２	基本特徴量格納アドレス不正
３０	ソース画面ウィンドウ設定エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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。





(注)ウイ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軸濃度累積値 ＝ 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

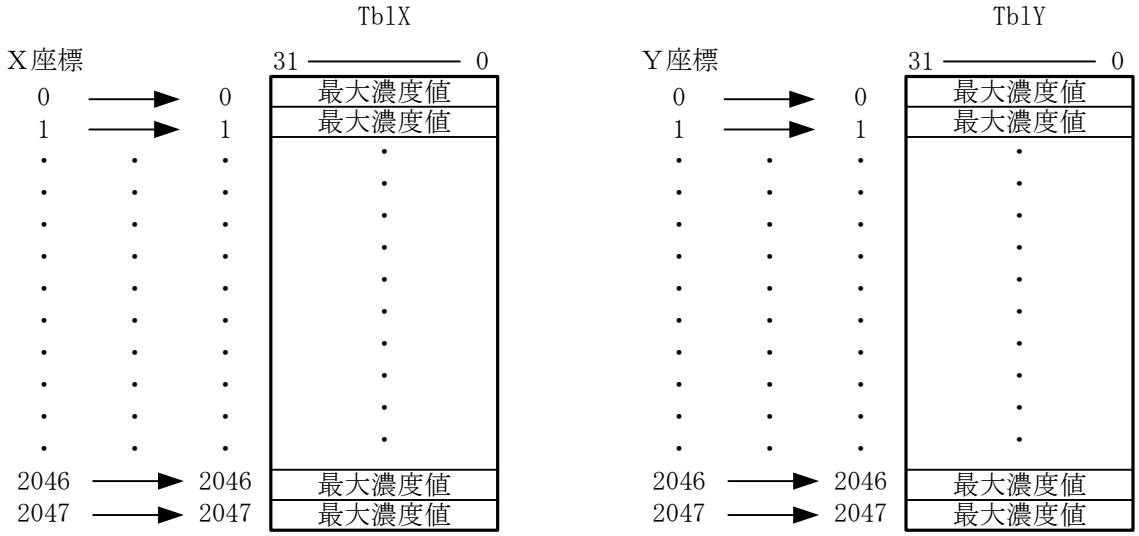
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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３２７６８（０ｘ８０００）、符号付８ビットの場合は－３２７６８（０ｘＦＦＦＦ８０００）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。

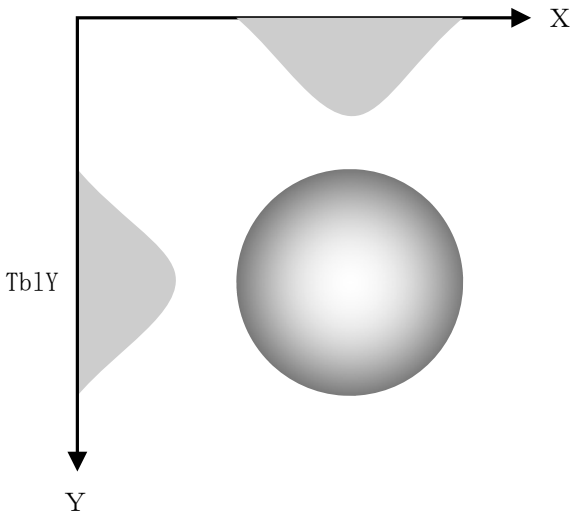


（注）ウィンドウサイズを越える部分は不定です。

本コマンドで得られるTblX、TblY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、

```
X軸最大濃度値 ＝ TblX [ X - sx ]
Y軸最大濃度値 ＝ TblY [ Y - sy ]
```

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

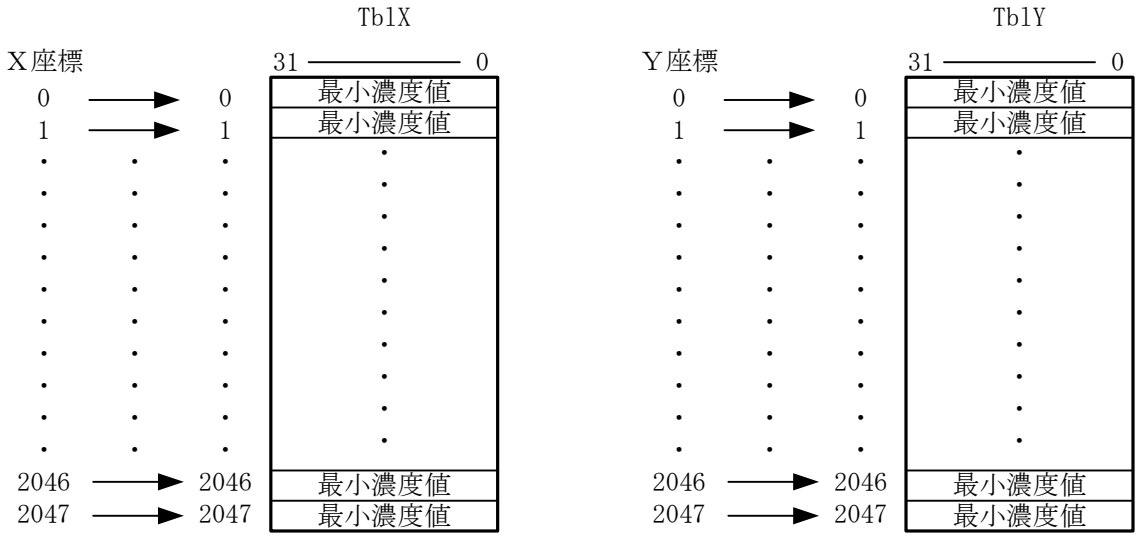
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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。

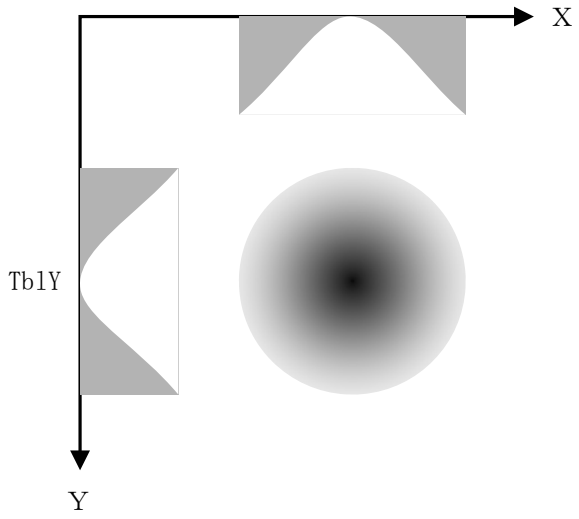


（注）ウィンドウサイズを越える部分は不定です。

本コマンドで得られるTblX、TblY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、

```
X軸最小濃度値 ＝ TblX [ X - sx ]
Y軸最小濃度値 ＝ TblY [ Y - sy ]
```

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
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）＊ 2048/（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

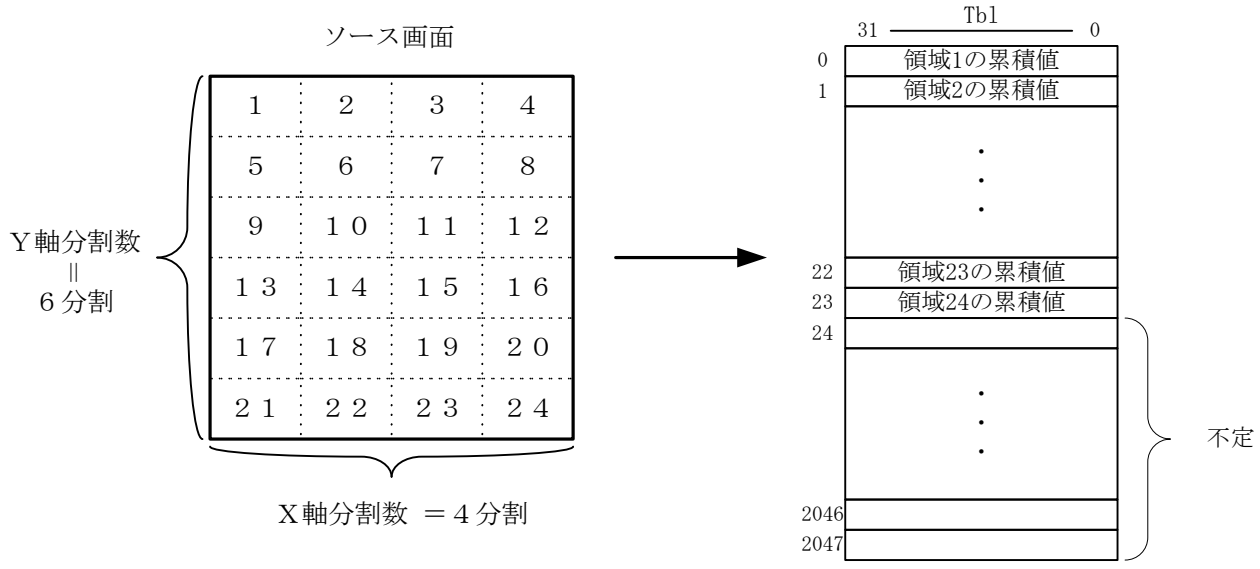
これは、『IP_ProjectBlockB0()』コマンド及び『IP_ProjectBlockG0()』コマンドを行う場合のヒストグラムメモリ（HM）への画像処理ＬＳＩ内部のアドレス発生回路が固定小数点演算で行っており、上記の条件を満たさないとHMのアドレスに誤差が生じる為です。なお、『IP_ProjectBlockB0()』コマンド、『IP_ProjectBlockG0MinMaxValue()』コマンド、及び『IP_ProjectBlockG0()』コマンド以外では、このアドレッシングを使用しないので、他のコマンドには影響ありません。

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下さい。
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	１～１０２４
３、４	１～５１２
５～８	１～２５６
９～１６	１～１２８
１７～３２	１～６４

Ｘ分割数	Ｙ分割数
３３～６４	１～３２
６５～１２８	１～１６
１２９～２５６	１～８
２５７～５１２	１～４
５１３～７０４	１

分割後の処理領域の長さは2048 以下にしてください。



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_ProjectBlockGO()』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
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）＊ 2048/（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

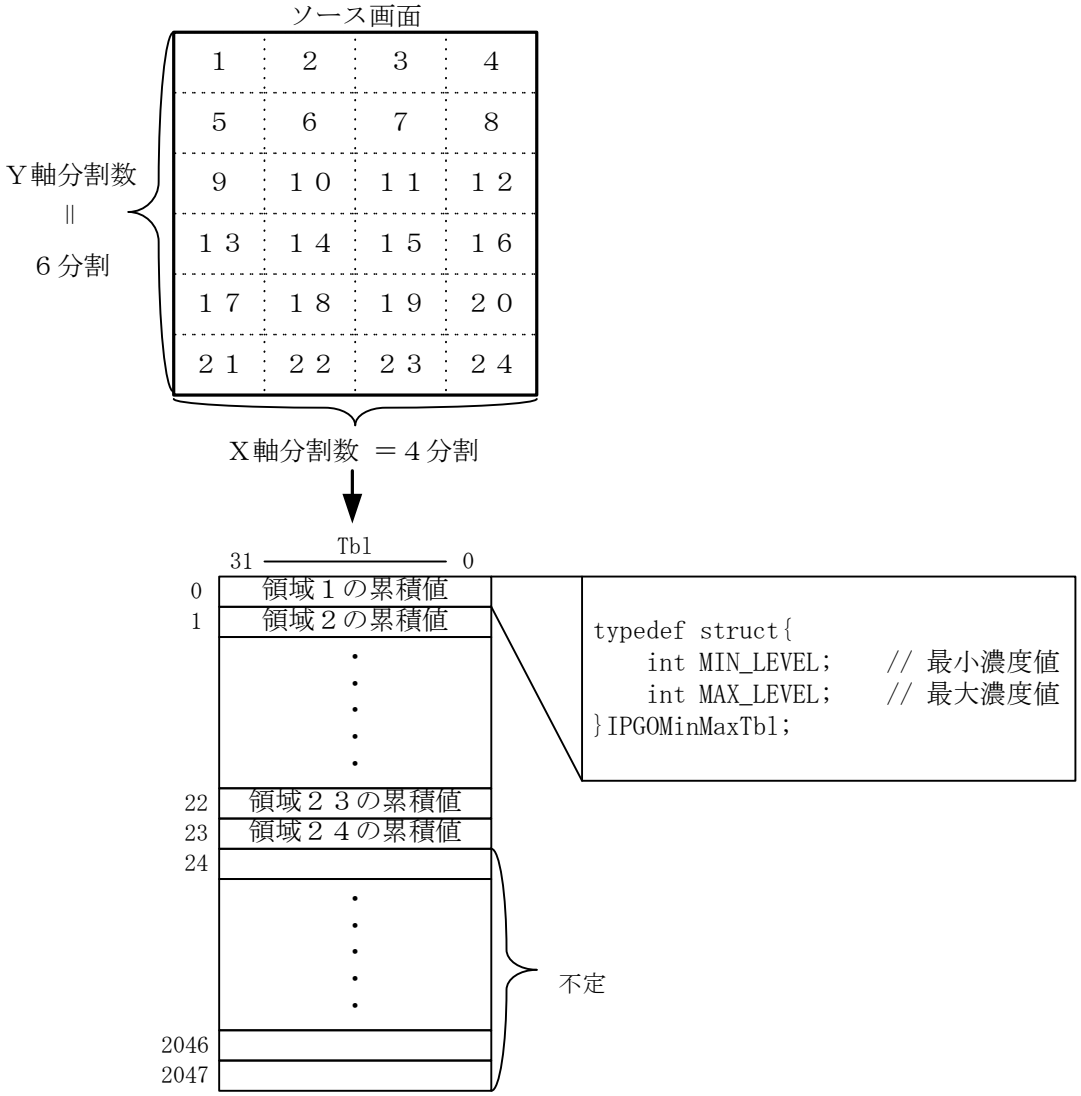
これは、『IP_ProjectBlockB0()』コマンド及び『IP_ProjectBlockG0()』コマンドを行う場合のヒストグラムメモリ（HM）への画像処理ＬＳＩ内部のアドレス発生回路が固定小数点演算で行っており、上記の条件を満たさないとHMのアドレスに誤差が生じる為です。なお、『IP_ProjectBlockB0()』コマンド、『IP_ProjectBlockG0MinMaxValue()』コマンド、及び『IP_ProjectBlockG0()』コマンド以外では、このアドレッシングを使用しないので、他のコマンドには影響ありません。

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下さい。
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	１～１０２４
３、４	１～５１２
５～８	１～２５６
９～１６	１～１２８
１７～３２	１～６４

Ｘ分割数	Ｙ分割数
３３～６４	１～３２
６５～１２８	１～１６
１２９～２５６	１～８
２５７～５１２	１～４
５１３～７０４	１

分割後の処理領域の長さは２０４８以下にしてください。



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

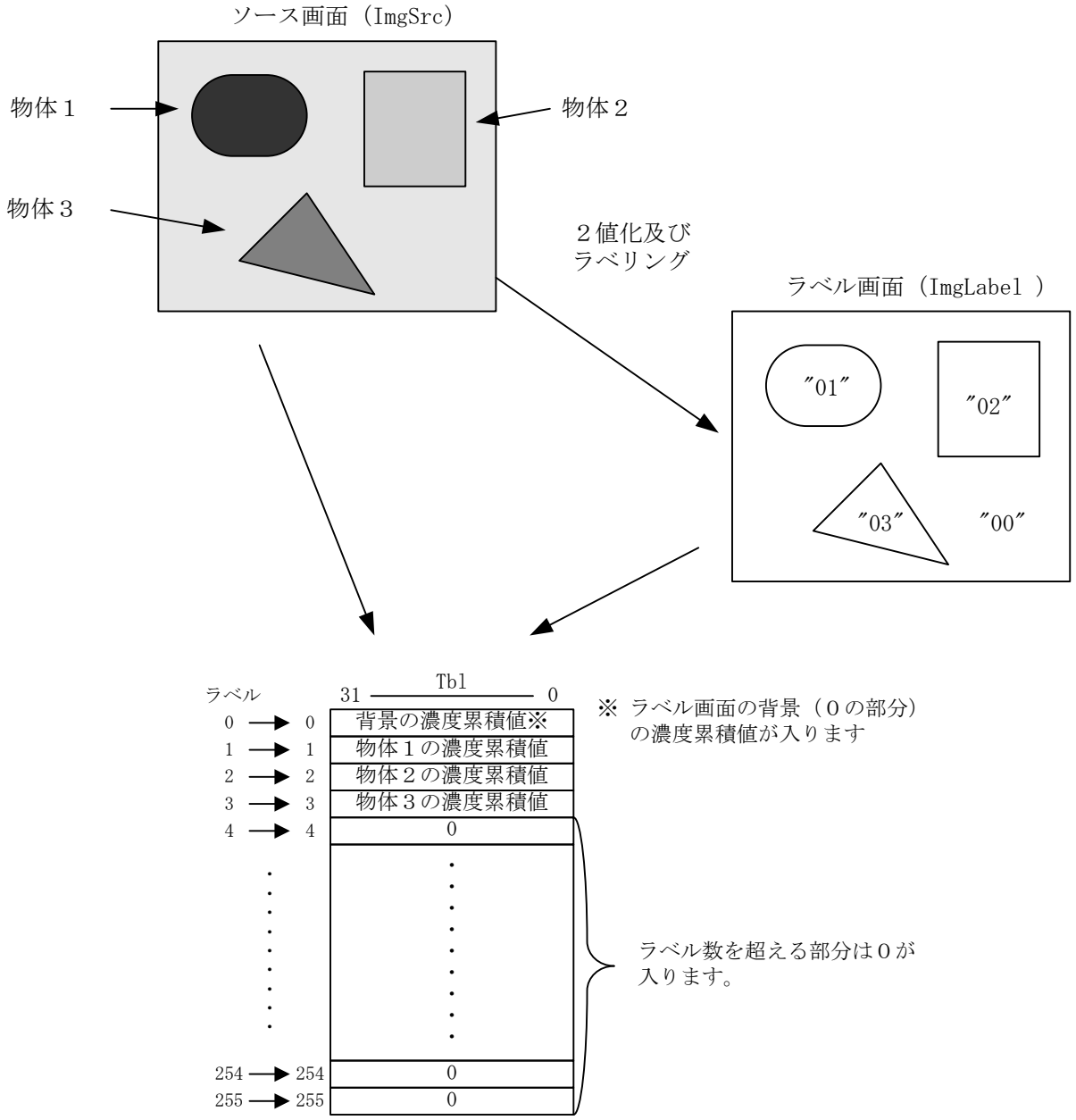
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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
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１０、１３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。

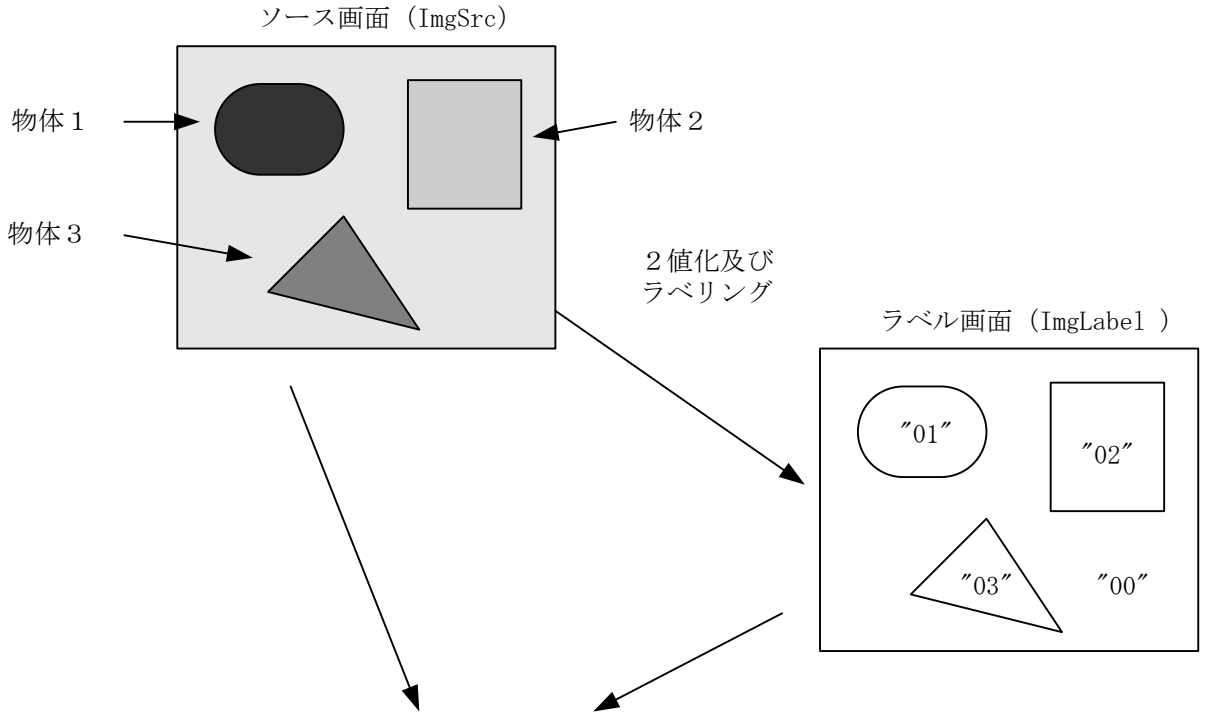
❑ エラーコード

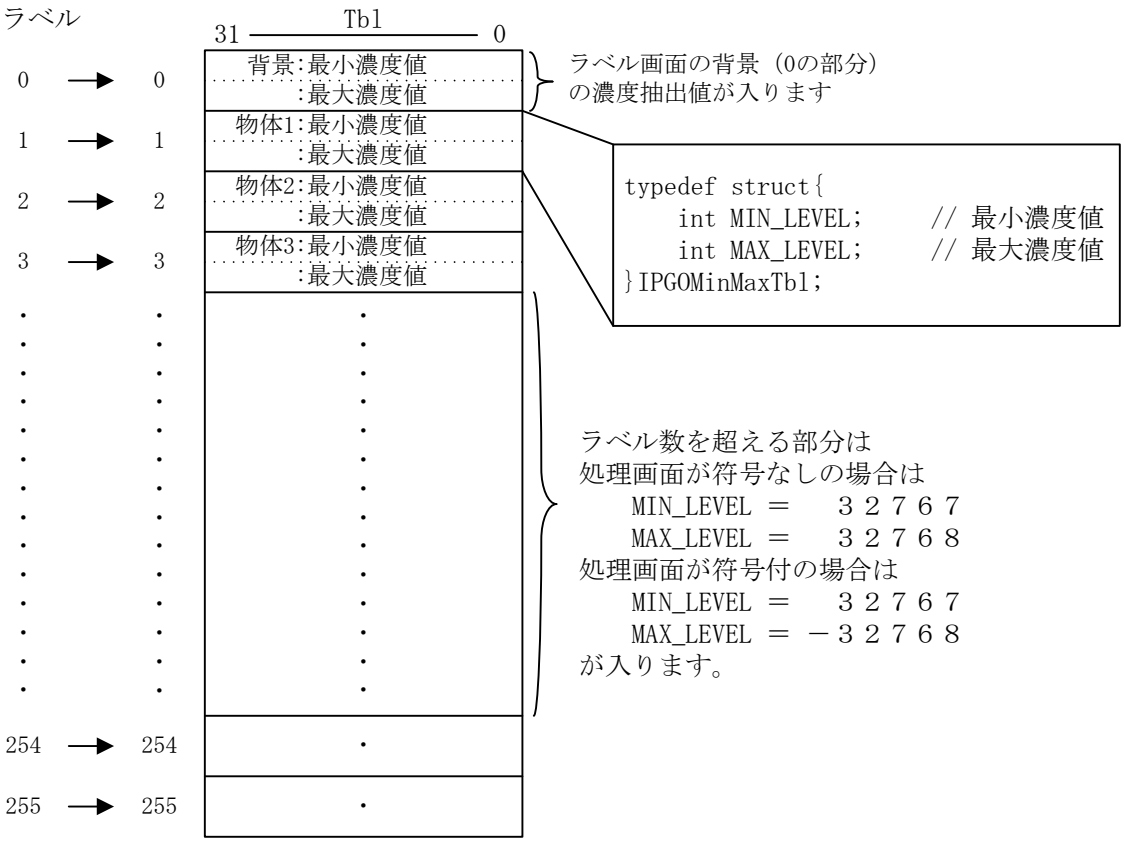
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１１	ラベル画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
２２	ラベル毎最小／最大濃度値格納アドレス不正
３０	ソース画面ウィンドウ設定エラー
３１	ラベル画面ウィンドウ設定エラー
４８	動的メモリ確保エラー
４９	ソース画面またはラベル画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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１０、１３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

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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	２値画像データの面積格納アドレス不正
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

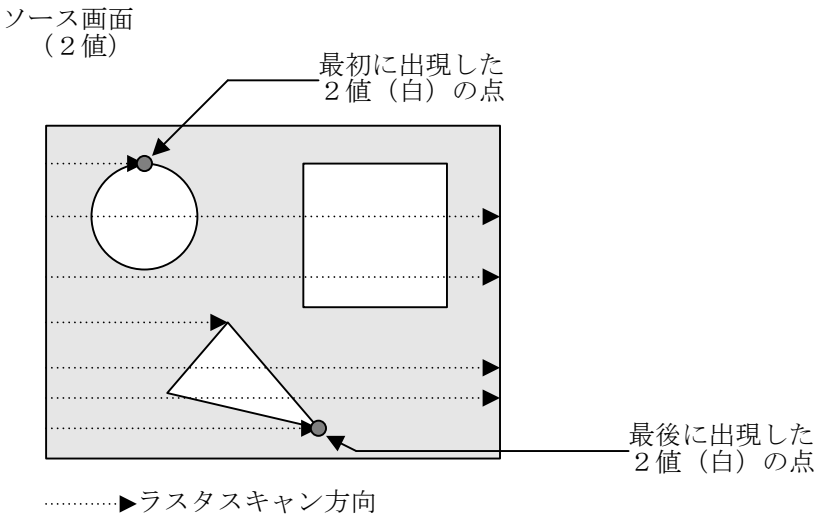
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
２１	基本特徴量格納アドレス不正
２２	オプション設定値範囲外
３０	ソース画面ウィンドウ設定エラー
４８	動的メモリ確保エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

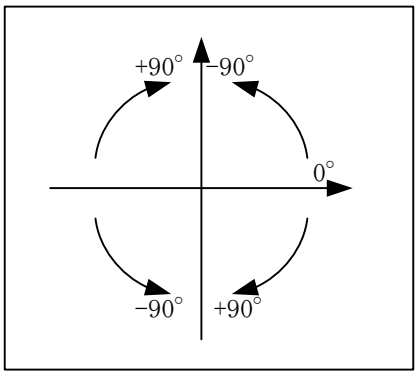
❑ 機能

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。



（※）IPBOFeatureTbl.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

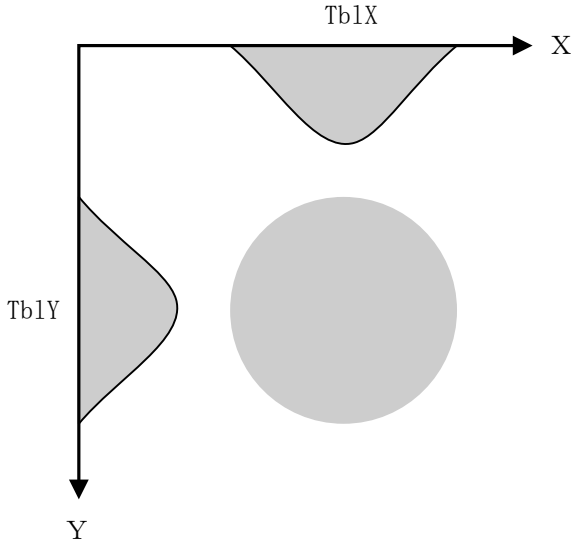
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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

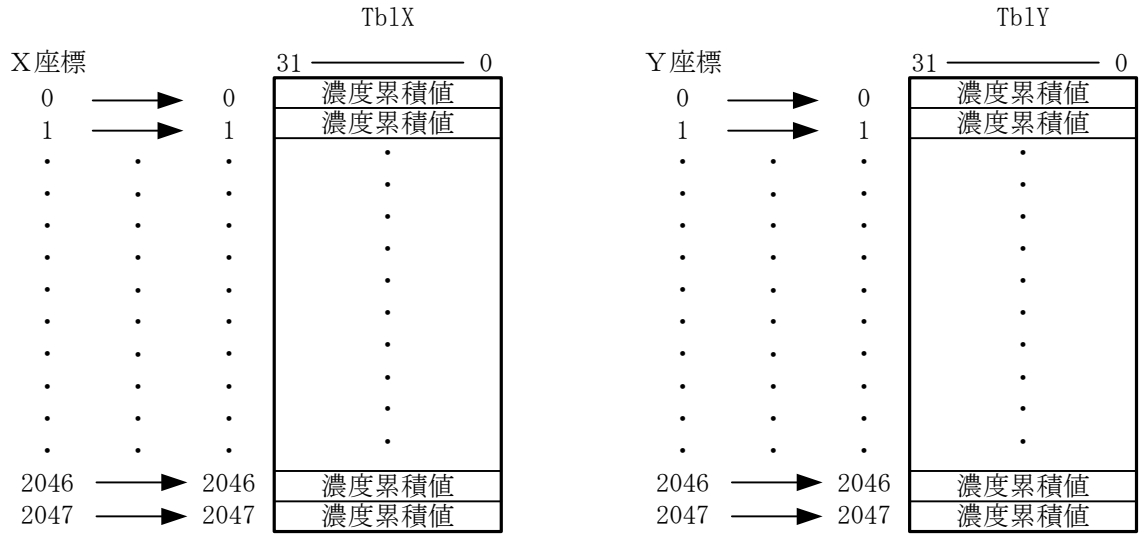
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 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。



TblX、TblYのフォーマットを下記に示します。



（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TblX、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、

```
Ｘ軸濃度累積値 ＝ TblX [ X - sx ]
Ｙ軸濃度累積値 ＝ TblY [ Y - sy ]
```

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で確保して下さい。
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で確保して下さい。

❑ 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

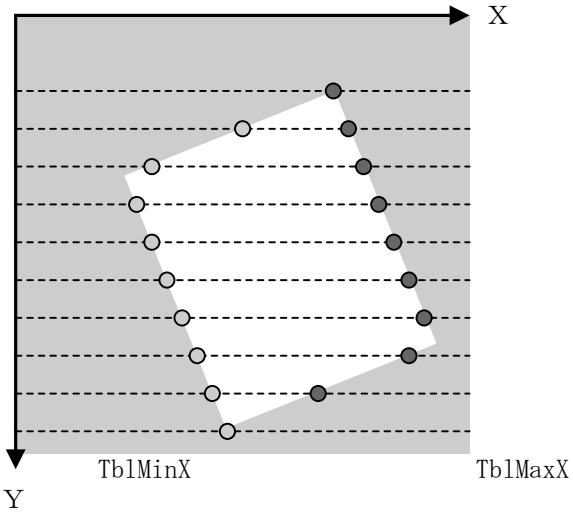
❑ エラーコード

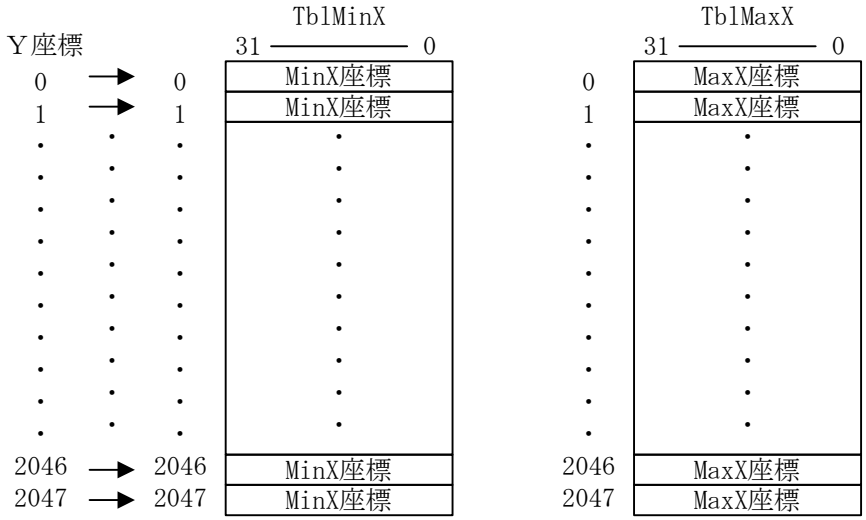
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
２１	MinX座標格納アドレス不正
２２	MaxX座標格納アドレス不正
３０	ソース画面ウィンドウ設定エラー
４８	動的メモリ確保エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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３２７６７（０ｘ７ＦＦＦ）、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。





（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、

```
最小Ｘ座標 ＝ TblMinX [ Y - sy ] + sx
最大Ｘ座標 ＝ TblMaxX [ Y - sy ] + sx
```

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で確保して下さい。
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で確保して下さい。

❑ 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

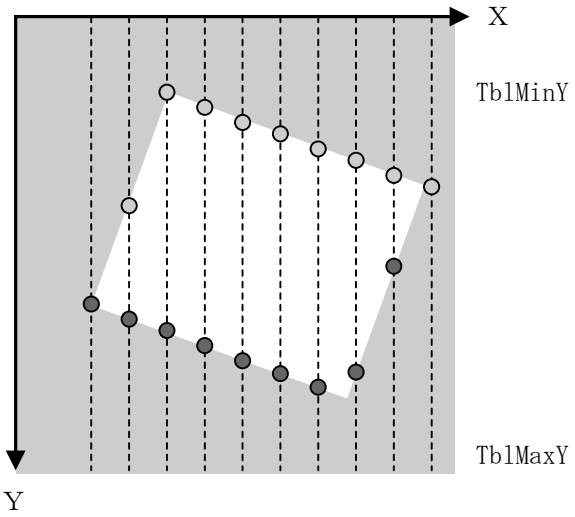
❑ エラーコード

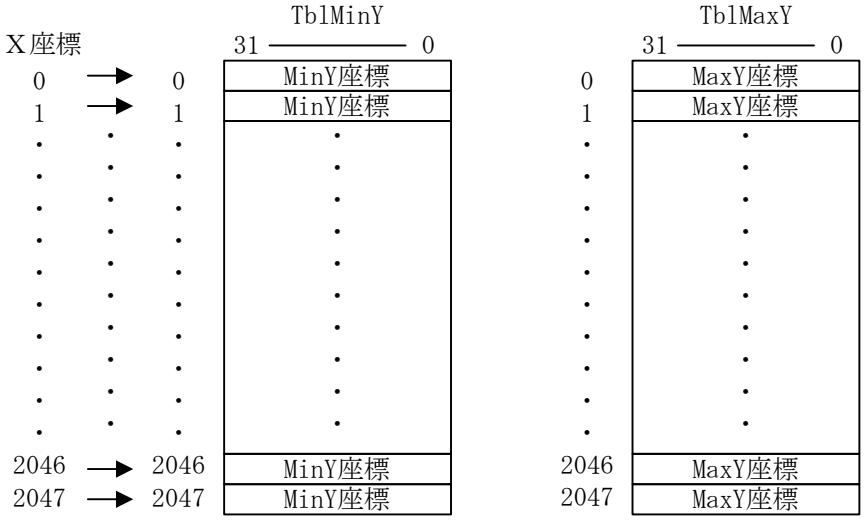
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
２１	MinY座標格納アドレス不正
２２	MaxY座標格納アドレス不正
３０	ソース画面ウィンドウ設定エラー
４８	動的メモリ確保エラー
４９	ソース画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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３２７６７（０ｘ７ＦＦＦ）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。





（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、

```
最小Ｙ座標 ＝ TblMinY [ X - sx ] + sy
最大Ｙ座標 ＝ TblMaxY [ X - sx ] + sy
```

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２０４８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
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）＊ 2048/（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

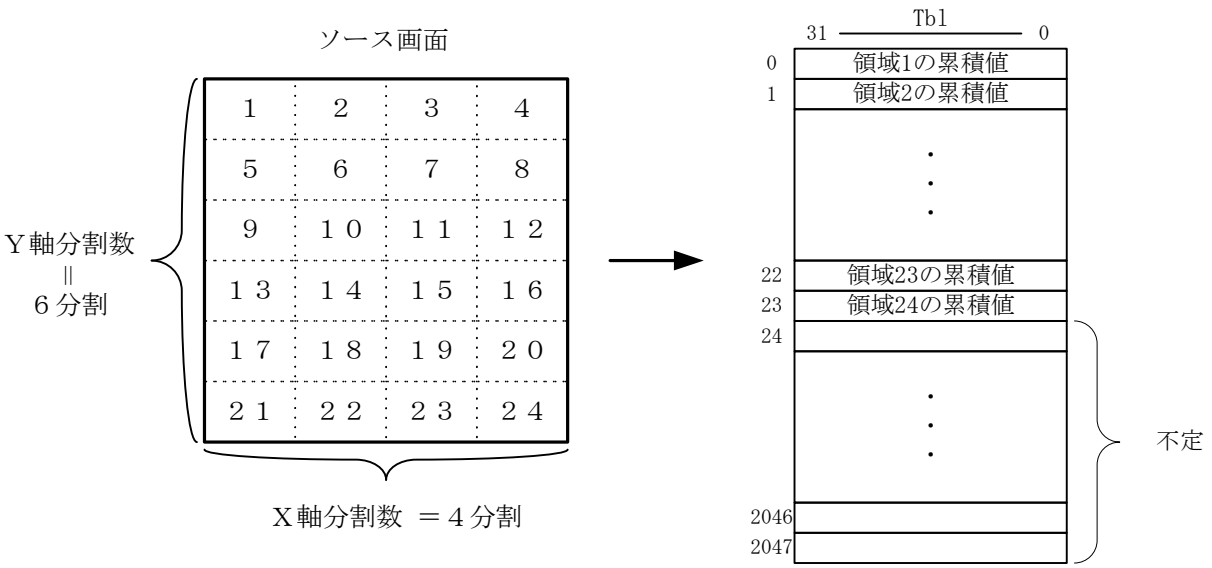
これは、『IP_ProjectBlockB0()』コマンド及び『IP_ProjectBlockG0()』コマンドを行う場合のヒストグラムメモリ（HM）への画像処理ＬＳＩ内部のアドレス発生回路が固定小数点演算で行っており、上記の条件を満たさないとHMのアドレスに誤差が生じる為です。なお、『IP_ProjectBlockB0()』コマンド、『IP_ProjectBlockG0MinMaxValue()』コマンド、及び『IP_ProjectBlockG0()』コマンド以外は、このアドレッシングを使用しないので影響ありません。

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下さい。
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	１～１０２４
３、４	１～５１２
５～８	１～２５６
９～１６	１～１２８
１７～３２	１～６４

Ｘ分割数	Ｙ分割数
３３～６４	１～３２
６５～１２８	１～１６
１２９～２５６	１～８
２５７～５１２	１～４
５１３～７０４	１

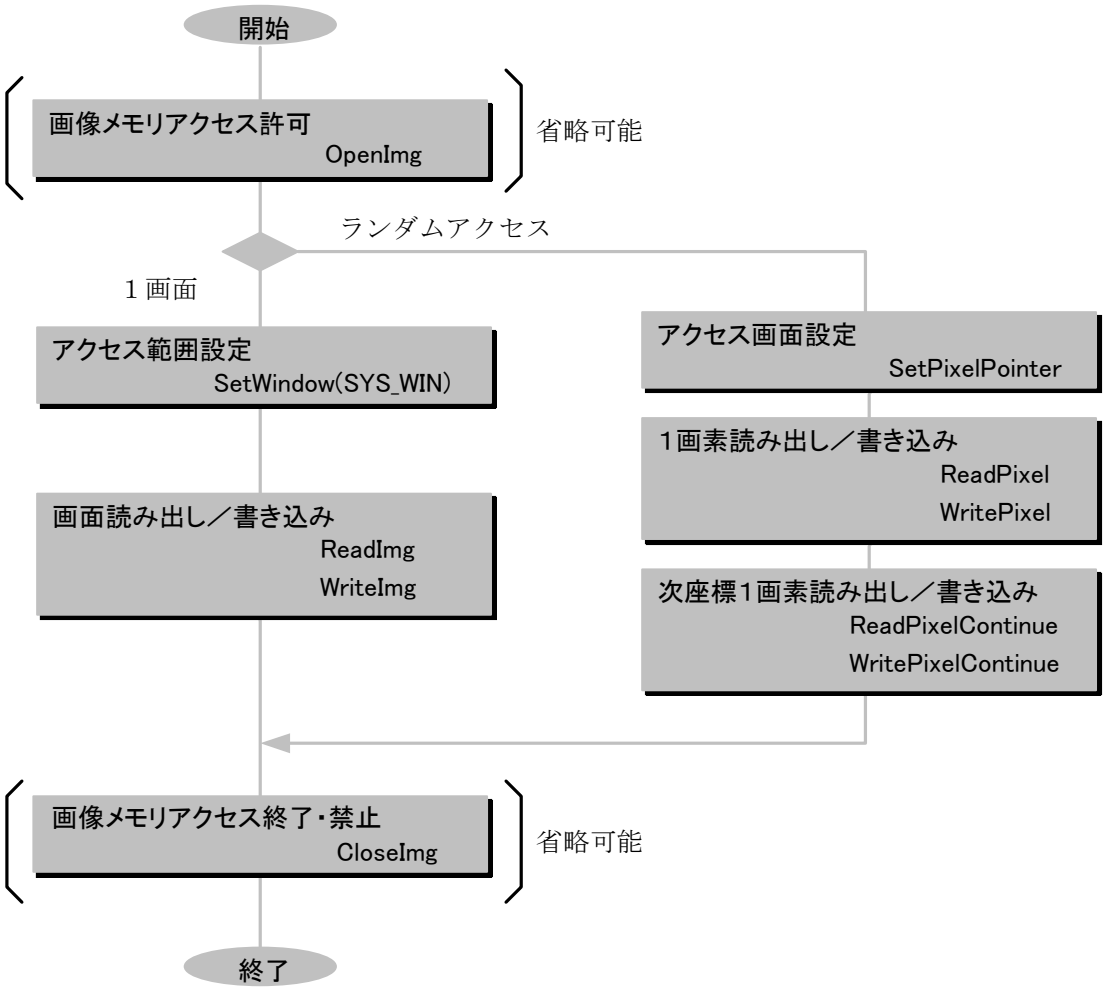
分割後の処理領域の長さは2048 以下にしてください。



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
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
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
－	エラーリセットコマンド（ClearIPError）未発行

※ 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
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	読み出しデータを返すアドレスが不正
２２	読み出し画素数範囲外
３４	画像メモリアクセスウィンドウ設定エラー
４９	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。

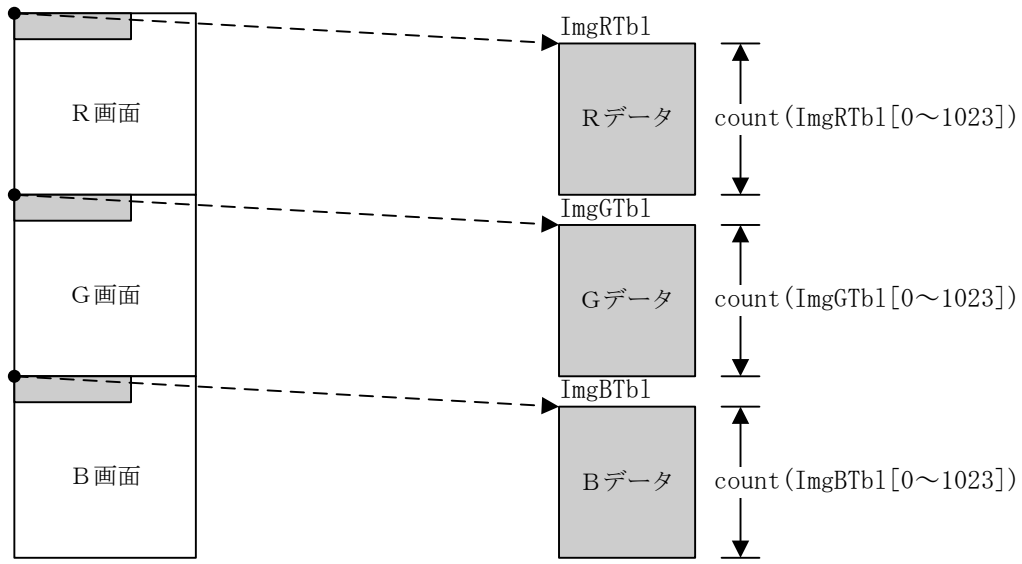
❑ 機能

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。

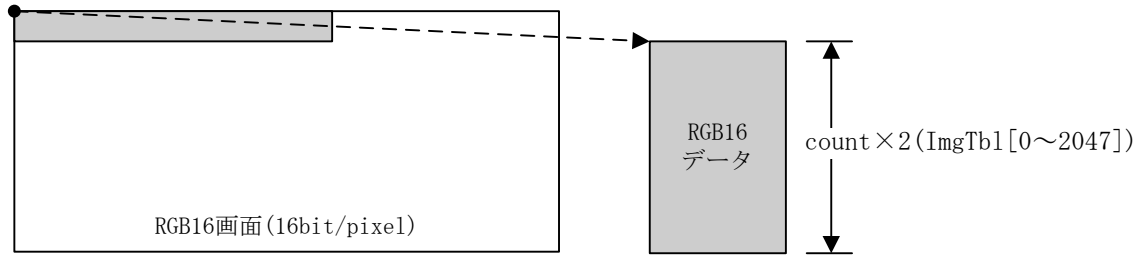
[ＲＧＢ画面読み出し例]

```
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 );
```



[ＲＧＢ１６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
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	書き込みデータを格納したアドレスが不正
２２	書き込み画素数範囲外
３４	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

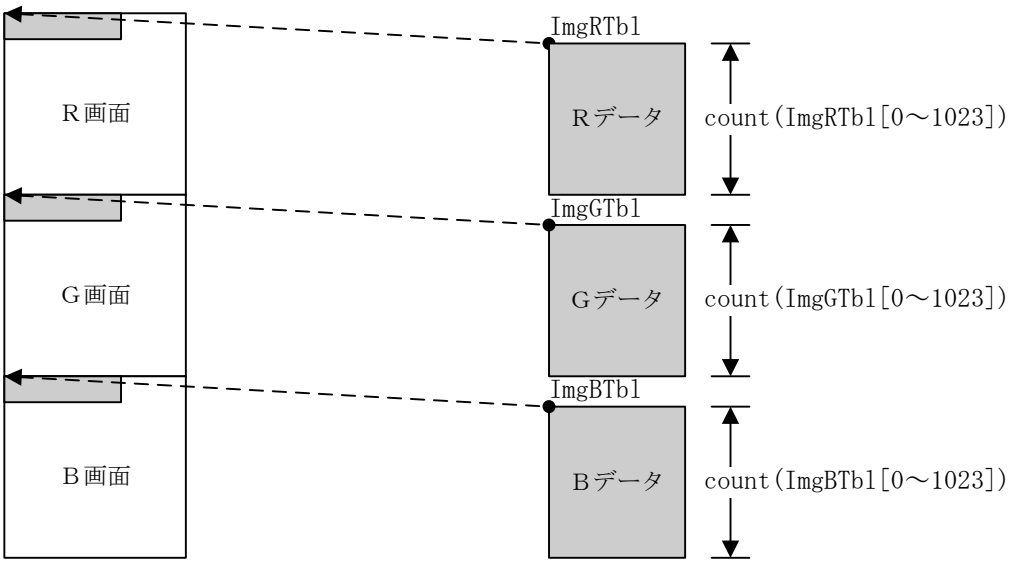
パラメータで指定された画面番号の画像メモリに指定画素数分書き込みます。指定画素数が画面（ウィンドウ）を超える場合は、越える直前までのデータが書き込まれます。画像メモリへの書き込みサイズは、ウィンドウが有効の時は画像メモリアクセスウィンドウサイズ（SYS_WIN）、無効の時は画面サイズに従います。
ＲＧＢ１６画面の書込みは、１６bit/pixel単位で行います。書込みデータ格納配列(ImgTbl)は、count×２バイト分確保して下さい。

ＲＧＢ画面は、Ｒ、Ｇ、Ｂの画面をそれぞれＹ画面として書き込みます。ＲＧＢ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、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 >= 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。

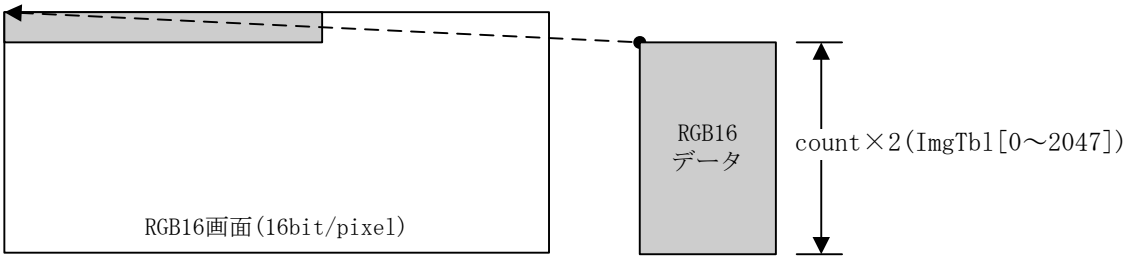
[ＲＧＢ画面書込み例]

```
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 );
```



[ＲＧＢ１６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
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）
３４	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

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

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
－	エラーリセットコマンド（ClearIPError）未発行

※ 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

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	読み出しデータを返すアドレスが不正
４４	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

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）
４４	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
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ＣＰＵ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ＣＰＵ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

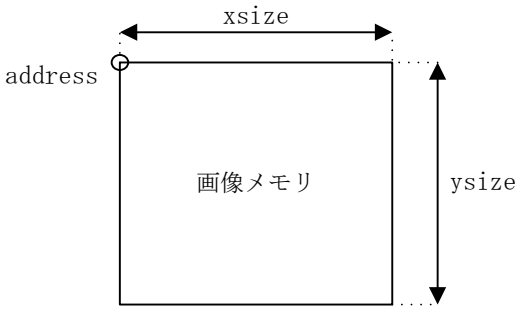
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	オープン対象画面番号に対応する画面が未確保、または番号が有効範囲外（不当画面番号エラー）
２１	オープン対象画面のＸ方向の長さを返すアドレスが不正
２２	オープン対象画面のＹ方向の長さを返すアドレスが不正
２３	オープン対象画面の先頭アドレスを返すアドレスが不正
４９	オープン対象画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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 * 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
１０	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）
４９	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 PipelineModepmode　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

２０	パイプライン処理出力段数／演算設定値範囲外
２１	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

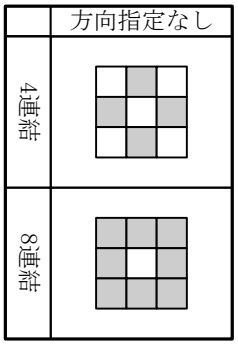
形状変換パイプライン（８段２値フィルタ）の処理内容を設定します。
各処理演算内容をStageFunction列挙型から選択し、PipelineStageMode構造体に設定して下さい。

・StageFunction列挙型

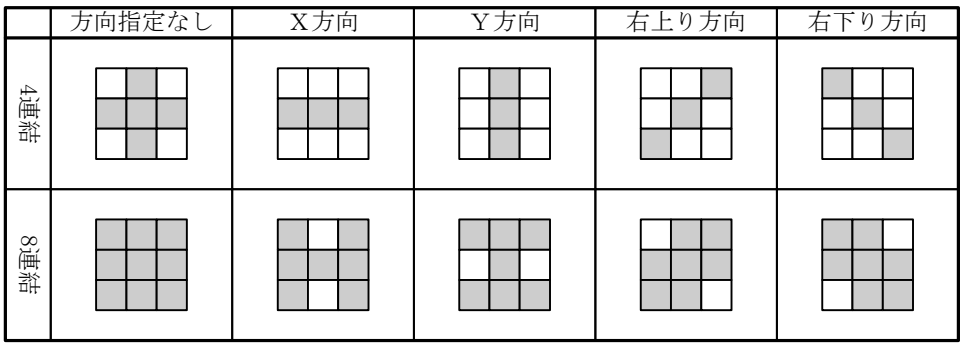
STAGE_NOOP	（０）	処理無し
STAGE_NOOP1	（１）	処理無し
DELETENOISE4	（２）	ノイズ除去４連結
DELETENOISE8	（３）	ノイズ除去８連結
DILATION4	（４）	４連結膨張
DILATION8	（５）	８連結膨張
EROSION4	（６）	４連結収縮
EROSION8	（７）	８連結収縮
OUTLINE4	（８）	４連結輪郭抽出
OUTLINE8	（９）	８連結輪郭抽出
DILATION4X	（１０）	４連結Ｘ方向膨張
DILATION8X	（１１）	８連結Ｘ方向膨張
EROSION4X	（１２）	４連結Ｘ方向収縮
EROSION8X	（１３）	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］



［膨張、収縮］



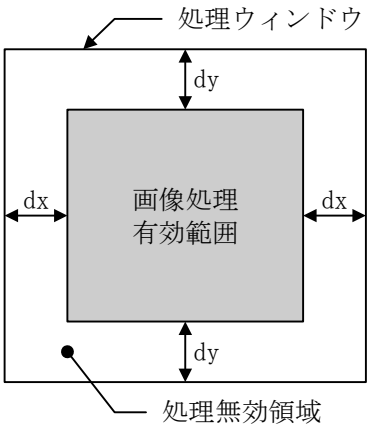
注目画素

❑ 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
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
４５	形状変換パイプライン処理設定エラー
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。

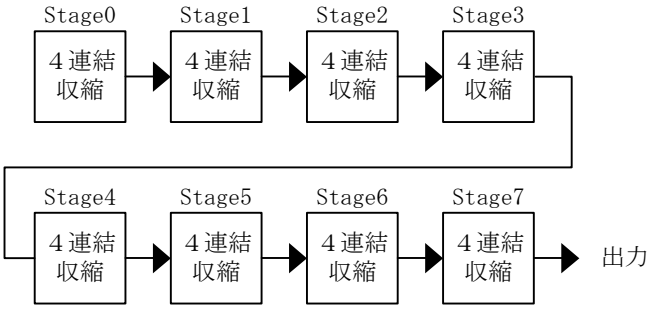
❑ 機能

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。

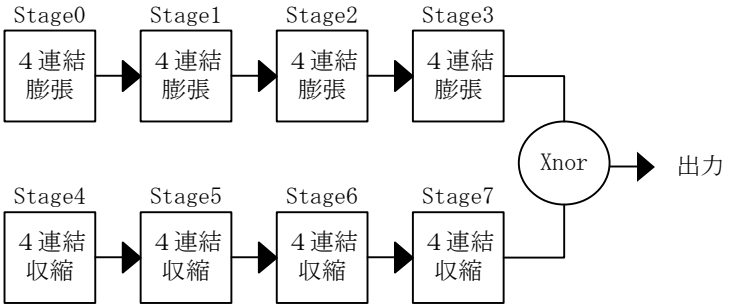
（実行例１）８段全て４連結収縮とする場合

```
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	関数名	備考
画像転送／アフィン変換	IP_Copy	
	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_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_Lapl4FLT	
	IP_Lapl8FLT	
	IP_Lapl4FLTAbs	
	IP_Lapl8FLTAbs	
	IP_LineFLT	
	IP_LineFLTAbs	
ミニ／マックスフィルタ	IP_MinFLT	
	IP_MinFLT4	
	IP_MinFLT8	
	IP_MaxFLT	
	IP_MaxFLT4	
	IP_MaxFLT8	
	IP_LineMinFLT	
	IP_LineMaxFLT	
ランクフィルタ	IP_RankFLT	
	IP_Rank4FLT	
	IP_Rank8FLT	
	IP_MedFLT	
	IP_Med4FLT	
	IP_Med8FLT	
２値マッチングフィルタ	IP_BinMatchFLT	

項目	コマンド名	備考
拡張コンボリューション	IP_SmoothFLT5x5	
	IP_SmoothFLT7x7	
	IP_EdgeFLT5x5	
	IP_EdgeFLT7x7	
	IP_MinFLT5x5	
	IP_MaxFLT5x5	
	IP_SmoothFLT5x5Ext	
	IP_SmoothFLT7x7Ext	
	IP_EdgeFLT5x5Ext	
	IP_EdgeFLT7x7Ext	

● 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_ExtractGOFeatures	注意事項3
	IP_Histogram	
	IP_HistogramShort	
	IP_HistogramFeatures	
	IP_ProjectGO	
	IP_ProjectGOonX	
	IP_ProjectGOonY	
	IP_ProjectGOMaxValue	
	IP_ProjectGOMinValue	
	IP_ProjectBlockGO	注意事項6
	IP_ProjectBlockGOMinMaxValue	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_ExtractBOArea	
	IP_ExtractBOFeatures	
	IP_ProjectBO	
	IP_ProjectBORegionX	
	IP_ProjectBORegionY	
	IP_ProjectBlockBO	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（パージ）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。
ペンディングされているコマンドを実行（パージ）してから動作するコマンドを以下に示します。

項目	コマンド名	備考
映像出力	CombineRGB	
	CombineRGBEx	
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_ConvertLUT	
	IP_ShiftUp	
濃淡画像特徴量抽出	IP_ProjectLabelGO	
	IP_ProjectLabelGOMinMaxValue	
画像メモリアクセス	OpenImg	
	OpenImgExt	
	CloseImg	
	ReadImg	
	WriteImg	
	SetPixelPointer	
	RefreshImg	
	OpenImgDirect	
	CloseImgDirect	
パイプライン制御	DisablePipeline	
正規化相関	SetCorrTemplate	
	SetCorrTemplateExt	
	IP_Corr	
	IP_CorrPrecise	
グラフィックス	RefreshGraphics	
	DrawString	
	DrawLine	
	DrawSegments	
	DrawLines	
	DrawRectangle	
	DrawPolygon	
	DrawArc	
拡張コンボリューション	IP_SmoothFLTExt	
	IP_EdgeFLTAbsExt	
線分化	ExtractPolyline	
RAW/RGB変換	IP_ConvertRAWtoRGB	
	IP_ConvertRAWtoRGBEx	
VP互換 正規化相関	vpxSetCorrTemplate	
	vpxSetCorrTemplateExt	
	vpxIP_CorrStep	
	vpxIP_CorrPoint	
	vpxIP_CorrPrecise	

［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（パージ）されます。

６．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、SetAllWindow、ResetAllWindow、
　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
－	エラーリセットコマンド（ClearIPError）未発行

※ 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

２０	テンプレート登録テーブルを格納したアドレスが不正
８１	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

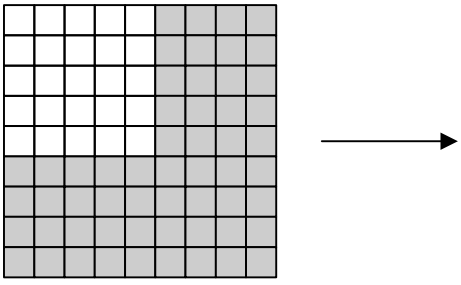
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 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

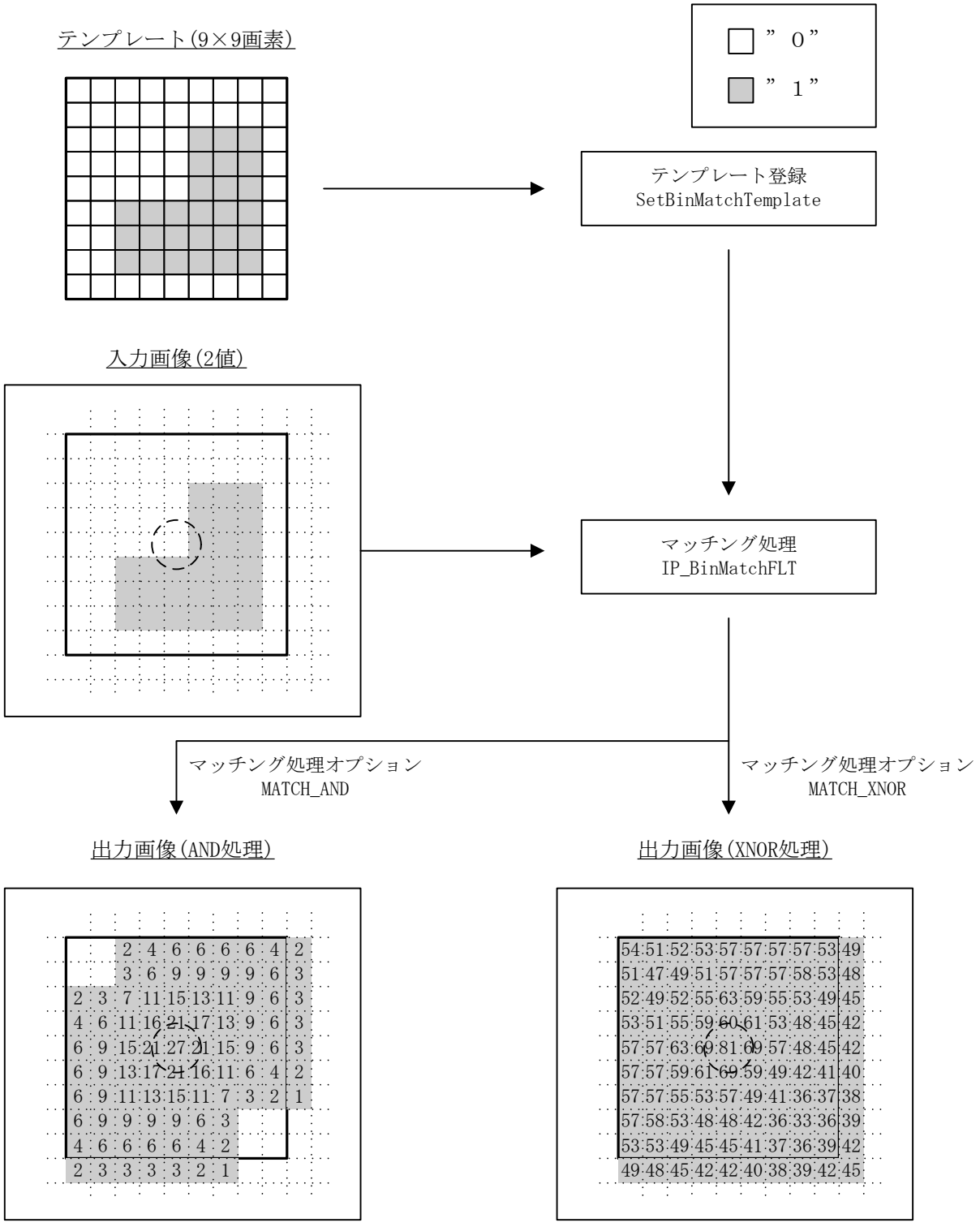
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
２２	テンプレート番号設定値範囲外
２３	オプション設定値範囲外
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４９	ソース画面が不定画面
８７	テンプレート未設定エラー
－	エラーリセットコマンド（ClearIPError）未発行

※ 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。

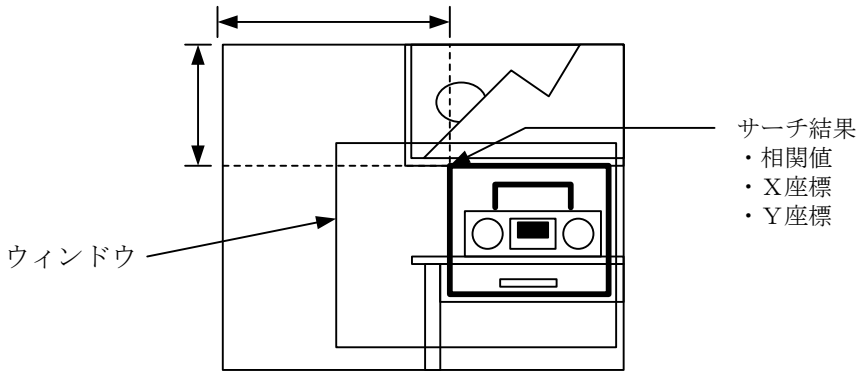
＜例＞



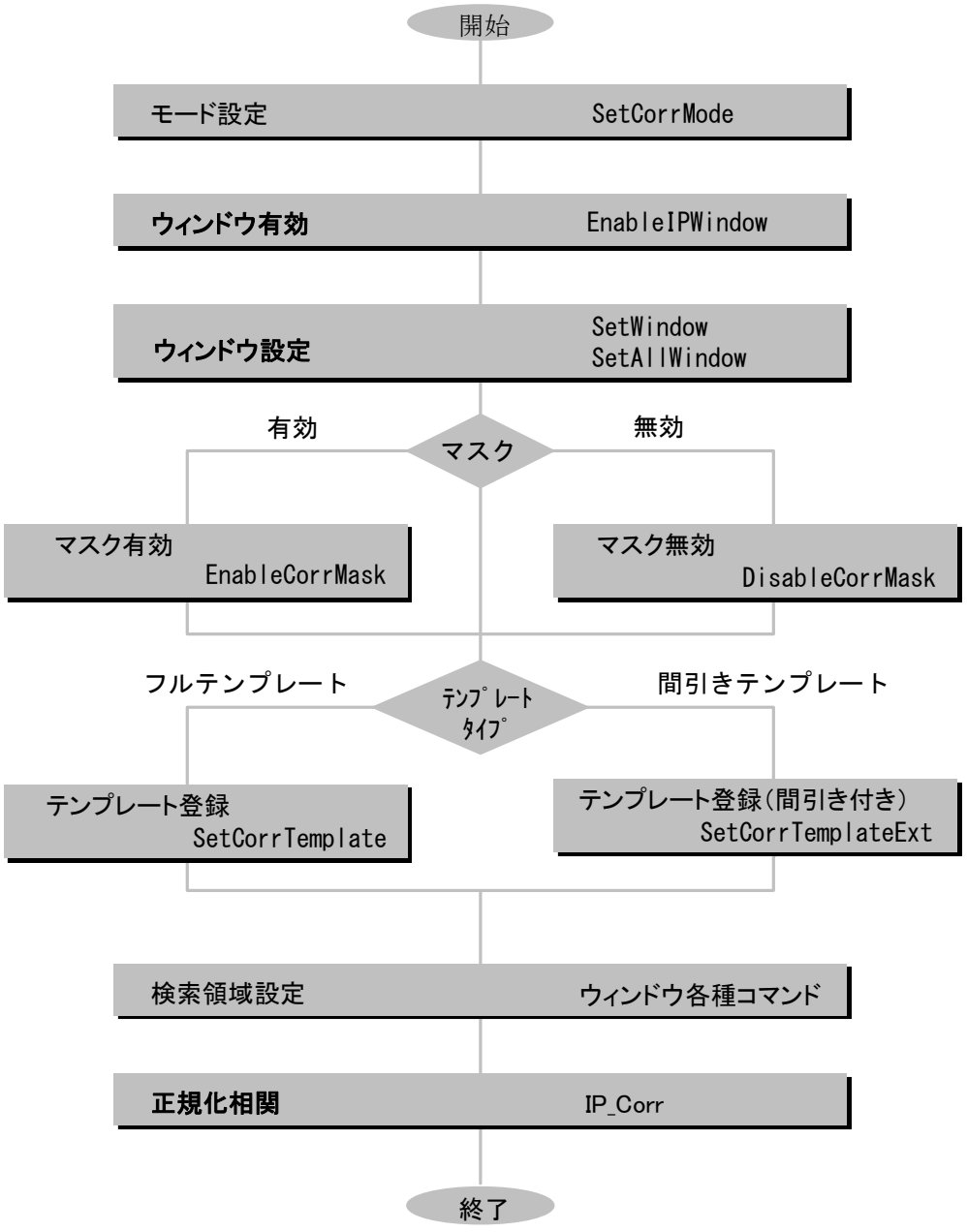
入力画像の（ ）を中心とした場合、オプションMATCH_ANDでは入力画像とテンプレートのＡＮＤ処理結果から“１”の総数である”２７”が出力され、オプションMATCH_XNORではＸＮＯＲ処理結果から“１”の総数である“８１”が出力されます。
※ 指定領域の周辺(内側４画素の領域)は、０を出力します。

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

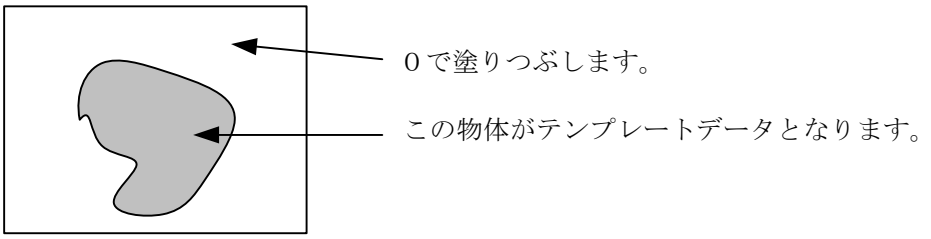
SetCorrTemplateExt()コマンドで登録されたテンプレートです。間引きテンプレートは、X／Y方向の間引き率を登録できます。この間引き率により、テンプレートデータおよび処理対象画面を間引いて演算処理するので、精度はフルテンプレートに対して落ちますが、演算回数が減るため、高速処理ができます。

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()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

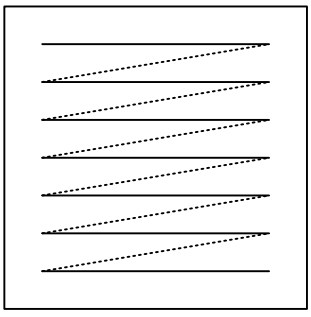
２０	走査方法設定値範囲外
８４	検知物体数エラー
８５	相関値エラー
－	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

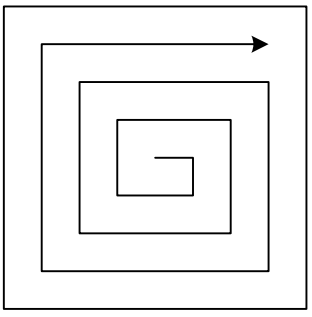
❑ 機能

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　正規化相関マスク有効（ 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

正規化相関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と同じになります。
『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、正規化相関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 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

２０	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。

2.22.5　正規化相関テンプレート登録（ 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
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

正規化相関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（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。

2.22.6　正規化相関テンプレート登録(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
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の、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（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

[注意事項]
ソース画面の画像メモリのX 方向サイズと変換倍率は以下の条件で指定してください。
ソース画面の画像メモリのX 方向サイズ（バイト） × 変換倍率 ≦ 32767

2.22.7 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
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）
２２	相関値および座標を返すアドレスが不正
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
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
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

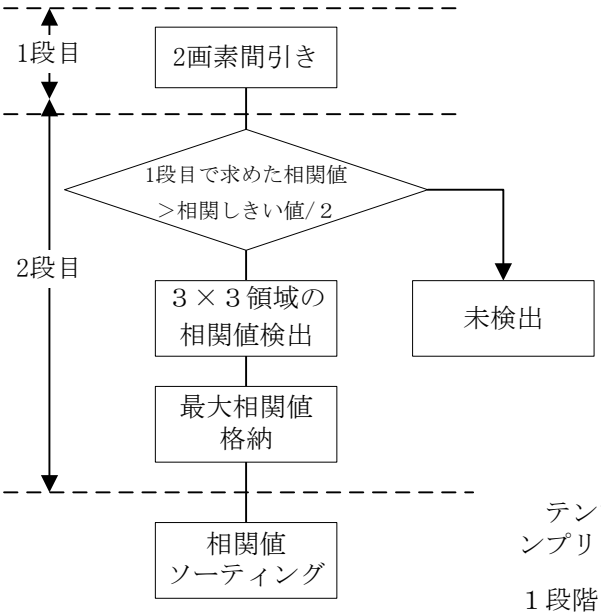
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

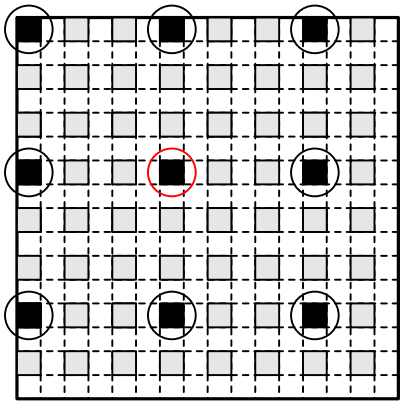
正規化相関を実行し、検知した物体数をリターン値として返します。また、そのときの相関値および座標を、ユーザテーブルへセットします。
『IP_Corr()』コマンドでは、使用するテンプレートタイプにより、処理内容が異なります。フルテンプレート（SetCorrTemplate()で登録）の場合は、処理領域内全画素を演算処理します。間引きテンプレート（SetCorrTemplateExt()で登録）の場合は、登録された間引き率で処理領域内を間引いて処理します。

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で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的です。
・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。



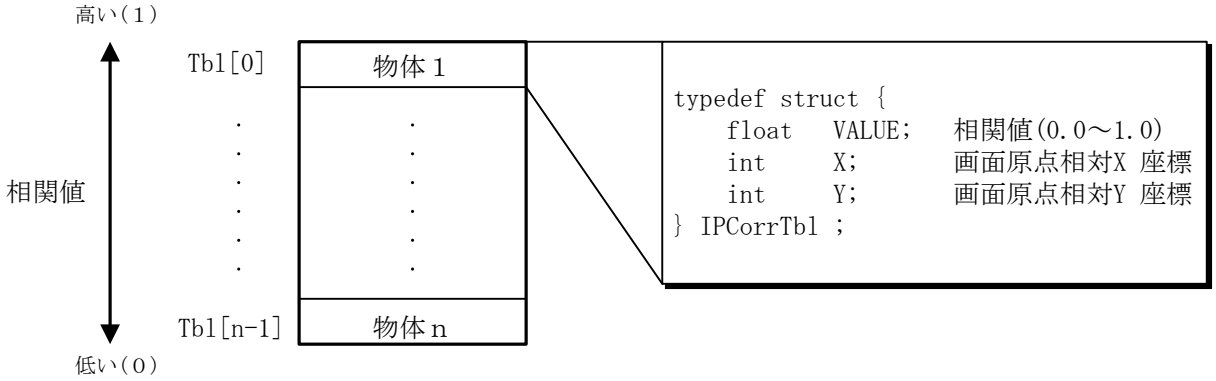
＜例：間引き率２＞



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。



2.22.8 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
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。

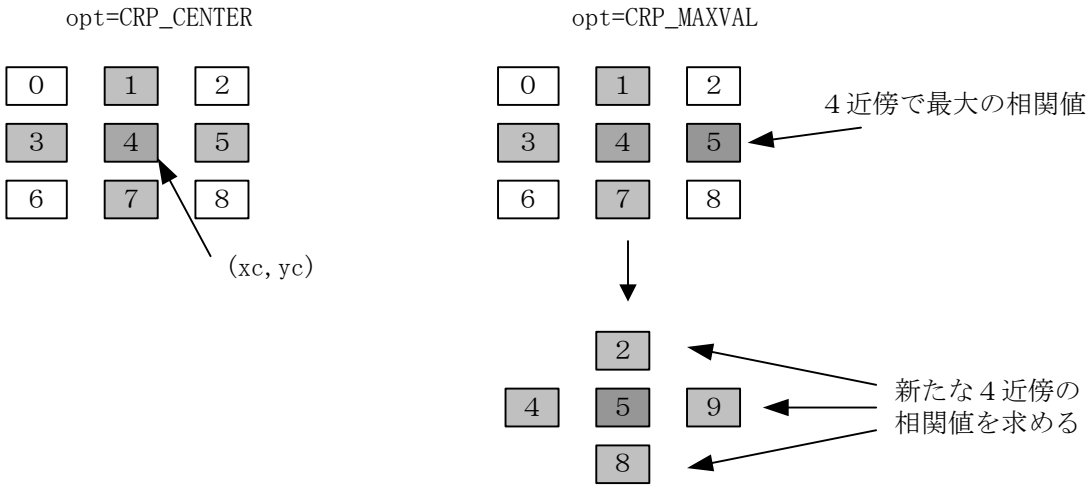
●（xc，yc）は、画面相対座標で指定して下さい。

●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。

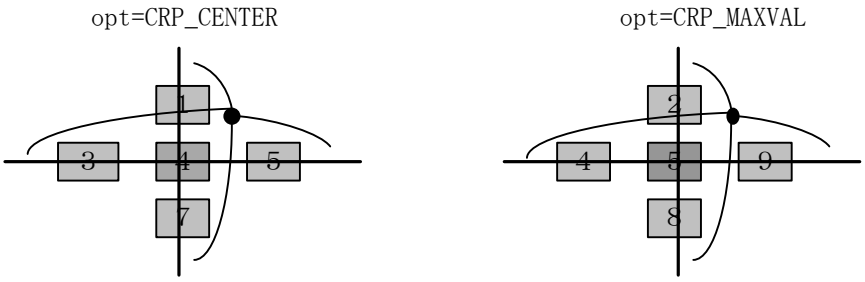
●ウィンドウ指定、『SetCorrMode()』コマンドは無視されます。

●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。



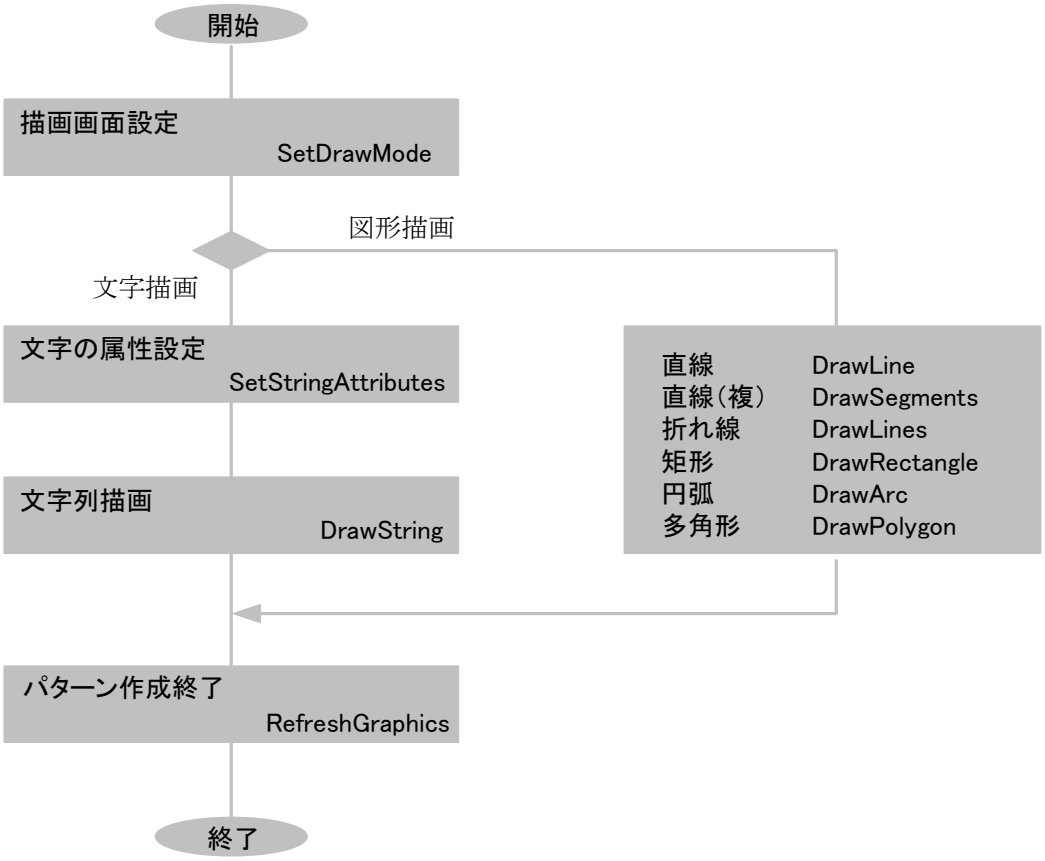
（B）求めたＸ方向の３点とＹ方向の３点の相関値を用いて、２次近似により最大値の存在するＸ座標とＹ座標を得る。



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



[注意事項]
開始から終了の間に画像処理のコマンドを入れないでください。

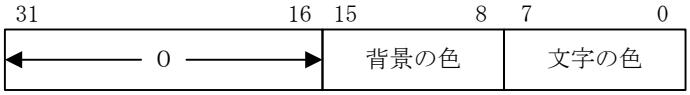
2.23.1 描画画面設定（ SetDrawMode ）

❑ C言語API

```
int ret = SetDrawMode(IMGID ImgSrc, enum IPX_DRAW_MODE mode, int color);
```

❑ パラメータ

IMGID	ImgSrc	ソース画面番号
enum IPX_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IPX_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

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	描画モード設定値範囲外
２２	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）

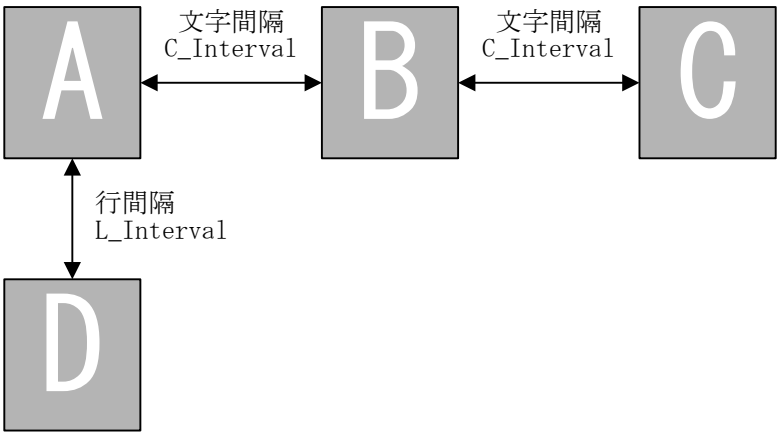
❑ エラーコード

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
４９	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パージを行います。

（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
１０	描画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	文字列を格納したアドレスが不正
３４	画像メモリアクセスウィンドウ（SYS_WIN）設定エラー
４４	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

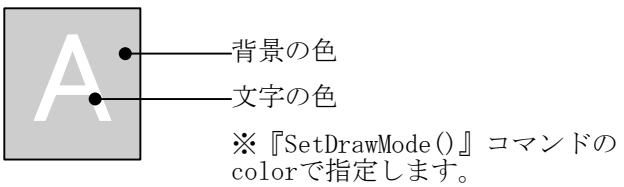
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 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。



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

上位4ビット	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
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	−	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）

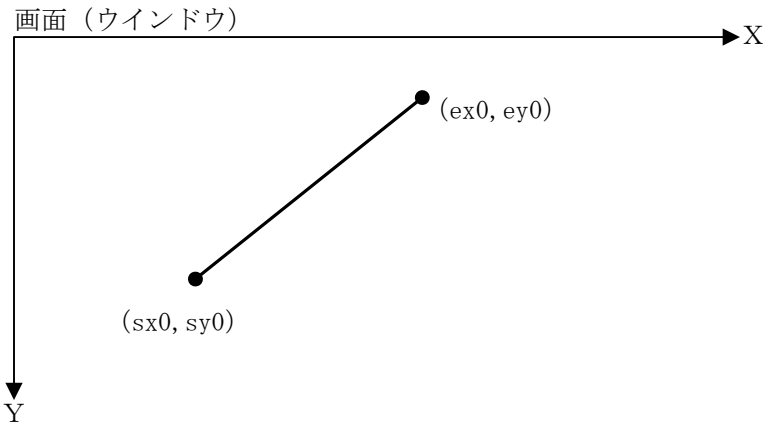
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	描画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	終点Ｘ座標設定エラー
２３	終点Ｙ座標設定エラー
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（ClearIPError）未発行

※ 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;
```

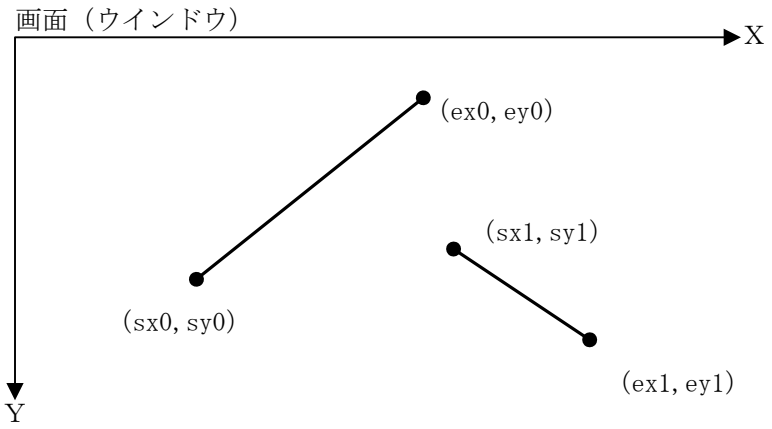
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	描画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	線分設定テーブルを格納したアドレスが不正
２１	描画線分数が０以下
３４	画像メモリアクセスウィンドウ（SYS_WIN）設定エラー
４４	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 ;
```

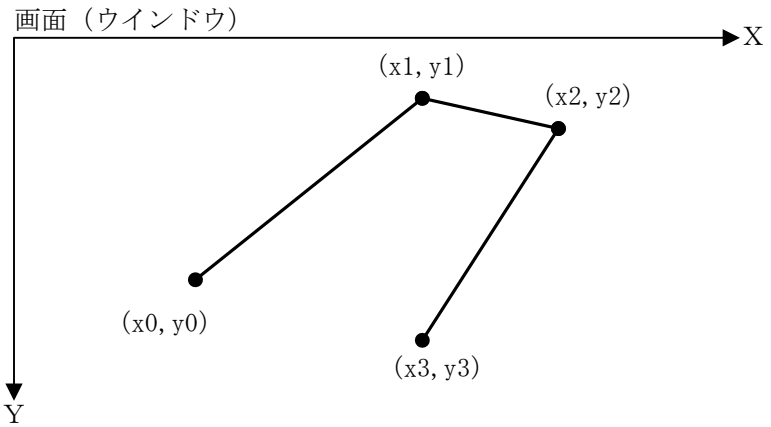
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	描画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	線分設定テーブルを格納したアドレスが不正
２１	設定座標数が１以下
３４	画像メモリアクセスウィンドウ（SYS_WIN）設定エラー
４４	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）

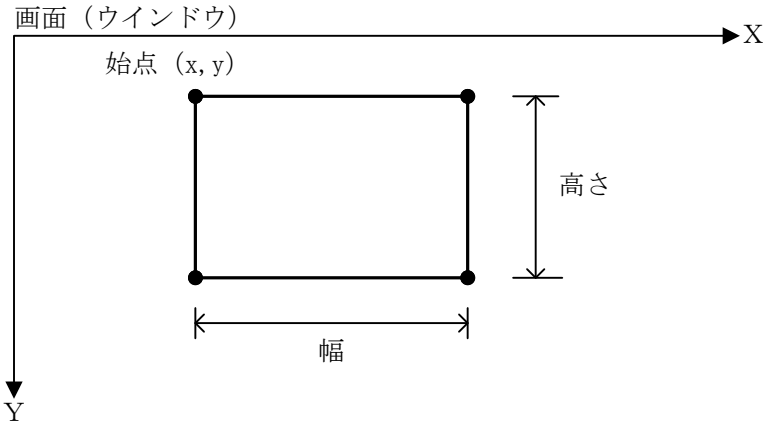
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	描画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（ClearIPError）未発行

※ 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。

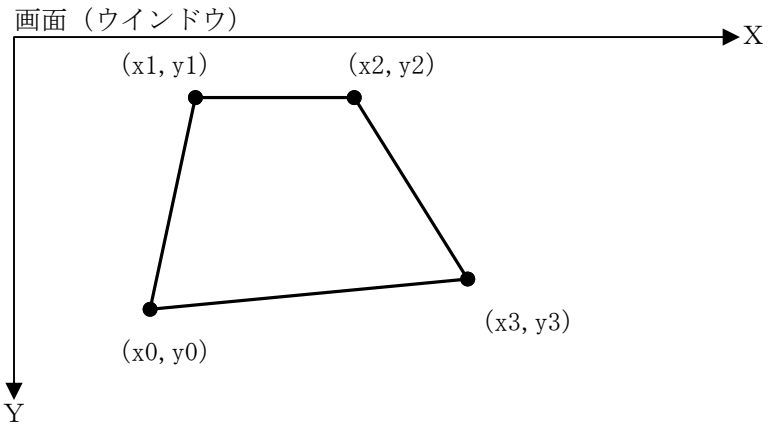
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	描画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	線分設定テーブルを格納したアドレスが不正
２１	設定座標数が１以下
３４	画像メモリアクセスウィンドウ（SYS_WIN）設定エラー
４４	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）

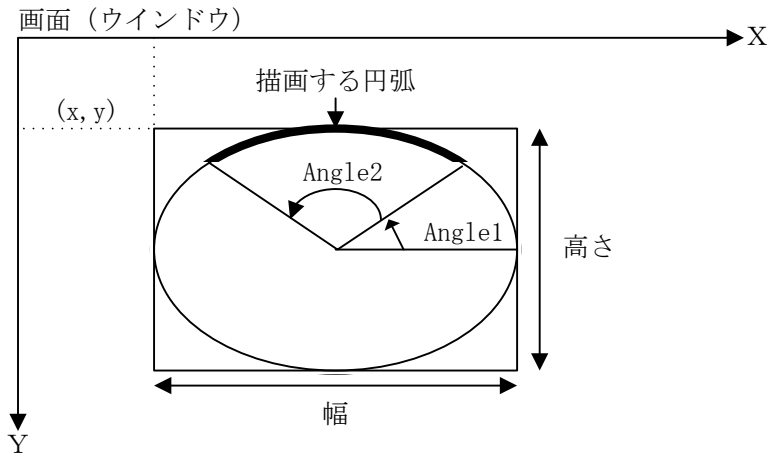
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	描画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座標設定エラー
２１	Ｙ座標設定エラー
２２	幅設定エラー（４以下）
２３	高さ設定エラー（４以下）
２４	弧の開始角度設定エラー
２５	弧の存在する角度設定エラー
３４	画像メモリアクセスウィンドウ（SYS_WIN）設定エラー
４５	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	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	描画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
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

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	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	描画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	線分設定テーブルを格納したアドレスが不正
２１	設定座標数が１以下
３４	画像メモリアクセスウィンドウ（SYS_WIN）設定エラー
４４	座標設定エラー
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

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

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）
３４	画像メモリアクセスウィンドウ（SYS_WIN）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
１００	画像ファイルオープンエラー
１０１	不当画像ファイルエラー
１３５	画面サイズ又は画面タイプが不正
－	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

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

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）
２２	転送モード指定範囲外
３４	画像メモリアクセスウィンドウ（SYS_WIN）設定エラー
４８	動的メモリ確保エラー
４９	セーブを行う画面が不定画面
１００	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。

Linuxは、ファイルをクローズしても直ぐにディスク装置にデータが書き込まれず、カーネルのディスクキャッシュにデータが書き込まれます。Linuxの「sync」システムコールを使用しディスクキャッシュからディスク装置にデータを書き込んでください。

なお、ディスク装置自体にもバッファキャッシュを持っており、最終的に物理的な記憶部に書き込みが終了するまで時間がかかる場合がありますので注意してください。

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 ランレングス・ラベリング

2.25.1 ランレングス・ラベリング（4連結）（ IP_Label4byRL ）

❑ C言語API

```
int ret = IP_Label4byRL( IMGID ImgSrc , IMGID ImgDst , enum IP_Label_opt 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

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

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	結果のﾃﾞｽﾃｨﾈｰｼｮﾝ画面未出力
LABEL_DST	1	結果のﾃﾞ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≦２５４　：正常終了（ラベル数）
		－１　：異常終了
		０ｘ７ｆｆｆ　：ラベルオーバーフロー
IPLabelBasicTbl	*BasicTbl	基本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
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
２２	ラベリング処理オプション１設定値範囲外
２３	基本特徴量テーブルを格納したアドレスが不正
２４	ラベリング処理オプション２設定値範囲外
３０	ソース画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
３３	デスティネーション画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
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)
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（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[ﾗﾍﾞﾙ番号].LEFT_UP_POINT_X + sx
MinY = BasicTbl[ﾗﾍﾞﾙ番号].LEFT_UP_POINT_Y + sy
MaxX = BasicTbl[ﾗﾍﾞﾙ番号].RIGHT_DOWN_POINT_X + sx
MaxY = BasicTbl[ﾗﾍﾞﾙ番号].RIGHT_DOWN_POINT_Y + sy
Gx   = BasicTbl[ﾗﾍﾞﾙ番号].CENTER_of_GRAVITY_X + sx
Gy   = BasicTbl[ﾗﾍﾞﾙ番号].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。

2.25.2 ランレングス・ラベリング（8連結）（ IP_Label8byRL ）

❑ C言語API

```
int ret = IP_Label8byRL( IMGID ImgSrc , IMGID ImgDst , enum IP_Label_opt 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

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

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	結果のﾃﾞｽﾃｨﾈｰｼｮﾝ画面未出力
LABEL_DST	1	結果のﾃﾞ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≦２５４　：正常終了（ラベル数）
　　－１　：異常終了
　　０ｘ７ｆｆｆ　：ラベルオーバーフロー
IPLabelBasicTbl　*BasicTbl　基本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
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
２２	ラベリング処理オプション１設定値範囲外
２３	基本特徴量テーブルを格納したアドレスが不正
２４	ラベリング処理オプション２設定値範囲外
３０	ソース画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
３３	デスティネーション画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
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)
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（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、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。

2.25.3 ランレングス・面積フィルタ付きラベリング（4連結）(IP_Label4byRLwithAreaFLT)

❑ C言語API

```
int ret = IP_Label4byRLwithAreaFLT( IMGID ImgSrc , IMGID ImgDst , long thrmin , long thrmax ,
          enum IP_Label_opt 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

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	結果のﾃﾞｽﾃｨﾈｰｼｮﾝ画面未出力
LABEL_DST	1	結果のﾃﾞ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≦２５４　：正常終了（ラベル数） －１　：異常終了 ０ｘ７ｆｆｆ　：ラベルオーバーフロー
IPLabelBasicTbl	*BasicTbl	基本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
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
２２	面積フィルタしきい値（最小値）設定値範囲外
２３	面積フィルタしきい値（最大値）設定値範囲外
２４	ラベリング処理オプション１設定値範囲外
２５	基本特徴量テーブルを格納したアドレスが不正
２６	ラベリング処理オプション２設定値範囲外
３０	ソース画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
３３	デスティネーション画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
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)
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（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、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。

2.25.4 ランレングス・面積フィルタ付きラベリング（8連結）(IP_Label8byRLwithAreaFLT)

❑ C言語API

```
int ret = IP_Label8byRLwithAreaFLT( IMGID ImgSrc , IMGID ImgDst , long thrmin , long thrmax ,
          enum IP_Label_opt 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

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	結果のﾃﾞｽﾃｨﾈｰｼｮﾝ画面未出力
LABEL_DST	1	結果のﾃﾞ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≦２５４　：正常終了（ラベル数） －１　：異常終了 ０ｘ７ｆｆｆ　：ラベルオーバーフロー
IPLabelBasicTbl	*BasicTbl	基本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
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
２２	面積フィルタしきい値（最小値）設定値範囲外
２３	面積フィルタしきい値（最大値）設定値範囲外
２４	ラベリング処理オプション１設定値範囲外
２５	基本特徴量テーブルを格納したアドレスが不正
２６	ラベリング処理オプション２設定値範囲外
３０	ソース画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
３３	デスティネーション画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
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)
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（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。

2.25.5 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。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	結果のﾃﾞｽﾃｨﾈｰｼｮﾝ画面未出力
LABEL_DST	1	結果のﾃﾞ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≦２５４　：正常終了（ラベル数）
		－１　：異常終了
		０ｘ７ｆｆｆ　：ラベルオーバーフロー
IPLabelBasicTbl	*BasicTbl	基本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
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
２２	面積フィルタしきい値（最小値）設定値範囲外
２３	面積フィルタしきい値（最大値）設定値範囲外
２４	ラベリング処理オプション１設定値範囲外
２５	ソート処理オプション設定値範囲外
２６	基本特徴量テーブルを格納したアドレスが不正
２７	ラベリング処理オプション２設定値範囲外
３０	ソース画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
３３	デスティネーション画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
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)
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（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。

2.25.6 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	結果のﾃﾞｽﾃｨﾈｰｼｮﾝ画面未出力
LABEL_DST	1	結果のﾃﾞ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≦２５４　：正常終了（ラベル数）
		－１　：異常終了
		０ｘ７ｆｆｆ　：ラベルオーバーフロー
IPLabelBasicTbl	*BasicTbl	基本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
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
２２	面積フィルタしきい値（最小値）設定値範囲外
２３	面積フィルタしきい値（最大値）設定値範囲外
２４	ラベリング処理オプション１設定値範囲外
２５	ソート処理オプション設定値範囲外
２６	基本特徴量テーブルを格納したアドレスが不正
２７	ラベリング処理オプション２設定値範囲外
３０	ソース画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
３３	デスティネーション画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
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)
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（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。

2.25.7 ランレングス・ラベリング（4連結・拡張）（ IP_Label4byRLExt ）

❑ C言語API

```
int ret = IP_Label4byRLExt( IMGID ImgSrc , IMGID ImgDst , 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

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。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	結果のﾃﾞｽﾃｨﾈｰｼｮﾝ画面未出力
LABEL_DST	1	結果のﾃﾞ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≦２５４　：正常終了（ラベル数）
　　－１　：異常終了
　　０ｘ７ｆｆｆ　：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
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
２２	ラベリング処理オプション１設定値範囲外
２３	拡張特徴量テーブルを格納したアドレスが不正
２４	ラベリング処理オプション２設定値範囲外
３０	ソース画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
３３	デスティネーション画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
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)
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（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 */
   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 */
}IPLabelExtTbl;
```

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[ﾗﾍﾞﾙ番号].LEFT_UP_POINT_X + sx
MinY = ExtTbl[ﾗﾍﾞﾙ番号].LEFT_UP_POINT_Y + sy
MaxX = ExtTbl[ﾗﾍﾞﾙ番号].RIGHT_DOWN_POINT_X + sx
MaxY = ExtTbl[ﾗﾍﾞﾙ番号].RIGHT_DOWN_POINT_Y + sy
Gx   = ExtTbl[ﾗﾍﾞﾙ番号].CENTER_of_GRAVITY_X + sx
Gy   = ExtTbl[ﾗﾍﾞﾙ番号].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。

2.25.8 ランレングス・ラベリング（8連結・拡張）（ IP_Label8byRLExt ）

❑ C言語API

```
int ret = IP_Label8byRLExt( IMGID ImgSrc , IMGID ImgDst , 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

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。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	結果のﾃﾞｽﾃｨﾈｰｼｮﾝ画面未出力
LABEL_DST	1	結果のﾃﾞ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≦２５４　：正常終了（ラベル数）
　　－１　：異常終了
　　０ｘ７ｆｆｆ　：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
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
２２	ラベリング処理オプション１設定値範囲外
２３	拡張特徴量テーブルを格納したアドレスが不正
２４	ラベリング処理オプション２設定値範囲外
３０	ソース画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
３３	デスティネーション画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
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)
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（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、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、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。詳細はIP_Label4byRLExt()コマンドを参照して下さい。

2.25.9 ランレングス・拡張面積フィルタ付きラベリング（4連結）
（ IP_Label4byRLwithAreaFLTExt ）

❑ C言語API

```
int ret = IP_Label4byRLwithAreaFLTExt( 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。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	結果のﾃﾞｽﾃｨﾈｰｼｮﾝ画面未出力
LABEL_DST	1	結果のﾃﾞ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≦２５４　：正常終了（ラベル数）
　　－１　：異常終了
　　０ｘ７ｆｆｆ　：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
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
２２	面積フィルタしきい値（最小値）設定値範囲外
２３	面積フィルタしきい値（最大値）設定値範囲外
２４	ラベリング処理オプション１設定値範囲外
２５	拡張特徴量テーブルを格納したアドレスが不正
２６	ラベリング処理オプション２設定値範囲外
３０	ソース画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい)
３３	デスティネーション画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい)
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)
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（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、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。

2.25.10 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。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	結果のﾃﾞｽﾃｨﾈｰｼｮﾝ画面未出力
LABEL_DST	1	結果のﾃﾞ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≦２５４ ：正常終了（ラベル数） －１ ：異常終了 ０ｘ７ｆｆｆ ：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
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
２２	面積フィルタしきい値（最小値）設定値範囲外
２３	面積フィルタしきい値（最大値）設定値範囲外
２４	ラベリング処理オプション１設定値範囲外
２５	拡張特徴量テーブルを格納したアドレスが不正
２６	ラベリング処理オプション２設定値範囲外
３０	ソース画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい)
３３	デスティネーション画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい)
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)
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（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。

2.25.11　ランレングス・拡張面積フィルタ付ソートラベリング（4連結）
（ 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	結果のﾃﾞｽﾃｨﾈｰｼｮﾝ画面未出力
LABEL_DST	1	結果のﾃﾞ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≦２５４　：正常終了（ラベル数）
		－１　：異常終了
		０ｘ７ｆｆｆ　：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
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
２２	面積フィルタしきい値（最小値）設定値範囲外
２３	面積フィルタしきい値（最大値）設定値範囲外
２４	ラベリング処理オプション１設定値範囲外
２５	ソート処理オプション設定値範囲外
２６	拡張特徴量テーブルを格納したアドレスが不正
２７	ラベリング処理オプション２設定値範囲外
３０	ソース画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
３３	デスティネーション画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
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)
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（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。

2.25.12 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。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	結果のﾃﾞｽﾃｨﾈｰｼｮﾝ画面未出力
LABEL_DST	1	結果のﾃﾞ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≦２５４　：正常終了（ラベル数）
		－１　：異常終了
		０ｘ７ｆｆｆ　：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
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
２２	統合ラベリングオプション設定値範囲外
２３	２値化しきい値設定値範囲外
２４	ラベリング管理テーブル設定値範囲外
２５	統合管理テーブル設定値範囲外
２６	基本特徴量テーブルを格納したアドレスが不正
２７	拡張特徴量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（ウィンドウ幅は２以上にして下さい）
３３	デスティネーション画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
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)
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（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。

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。

2.25.13 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 ﾃﾞｽﾃｨﾈｰｼｮﾝ画面にﾗﾍﾞﾘﾝｸﾞ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 ﾃﾞｽﾃｨﾈｰｼｮﾝ画面にﾗﾍﾞﾘﾝｸﾞ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 ～ ４００１）　基本特徴量テーブルのポインタ

IPCombineTblExt　*combineTblExt （２ ～ ４００１）　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

❑ 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	○	○	○	×	◆	-	-

❑ リターンパラメータ

int	ret	０≦ret≦４０００　：正常終了（ラベル数）
		－１　：異常終了
		０ｘ７ｆｆｆ　：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

❑ 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	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
２２	統合ラベリングオプション設定値範囲外
２３	２値化しきい値設定値範囲外
２４	ラベリング管理テーブル設定値範囲外
２５	統合管理テーブル設定値範囲外
２６	基本特徴量テーブルを格納したアドレスが不正
２７	拡張特徴量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（ウィンドウ幅は２以上にして下さい）
３３	デスティネーション画面ウィンドウ設定エラー（ウィンドウ幅は２以上にして下さい）
４９	アクセス画面が不定画面
５１	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（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 ～ 横ｳｨﾝﾄﾞｳAND サイズ     */
                                    /*       =0 ：統合なし )                  */
     int                   height;  /* 統合高さ範囲 (0 ～ 縦ｳｨﾝﾄﾞｳAND サイズ) */
     enum IPCombineInc  including; /* 包含による統合                         */
}IPCombineCtl;
```

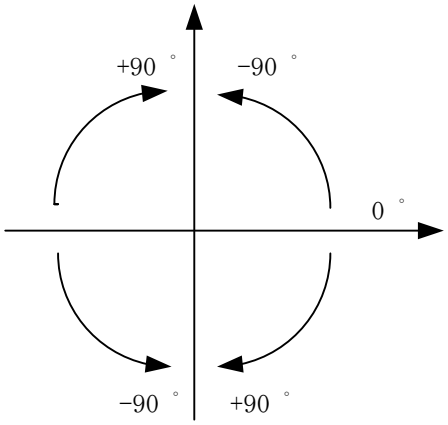
IPCombineTbl ：基本特徴量テーブルのポインタ

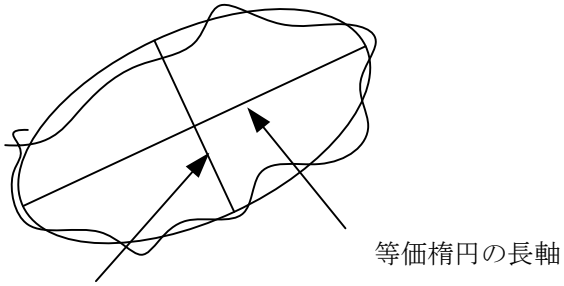
```
typedef struct {
   int           area;   /* 物体面積                 */
   int           xmin;   /* 物体を囲む最小Ｘ座標     */
   int           ymin;   /* 物体を囲む最小Ｙ座標     */
   int           xmax;   /* 物体を囲む最大Ｘ座標     */
   int           ymax;   /* 物体を囲む最大Ｙ座標     */
   int           length; /* 物体周囲長               */
}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４０００個です。ラベルオーバーフローは、仮ラベル数４００１個以上で発生します。真ラベル付けの最大数は４０００個です。

・ 現状でのラベル画像のデータタイプはUNSIGN8 なので、２５５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。

2.26 2値化閾値算出支援

2.26.1 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（lower≧０）
int	upper	上位レベル（lower＜upper≦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	≧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

２０	判別分析テーブルを格納したアドレスが不正
２１	下位レベル設定値範囲外
２２	上位レベル設定値範囲外
－	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。

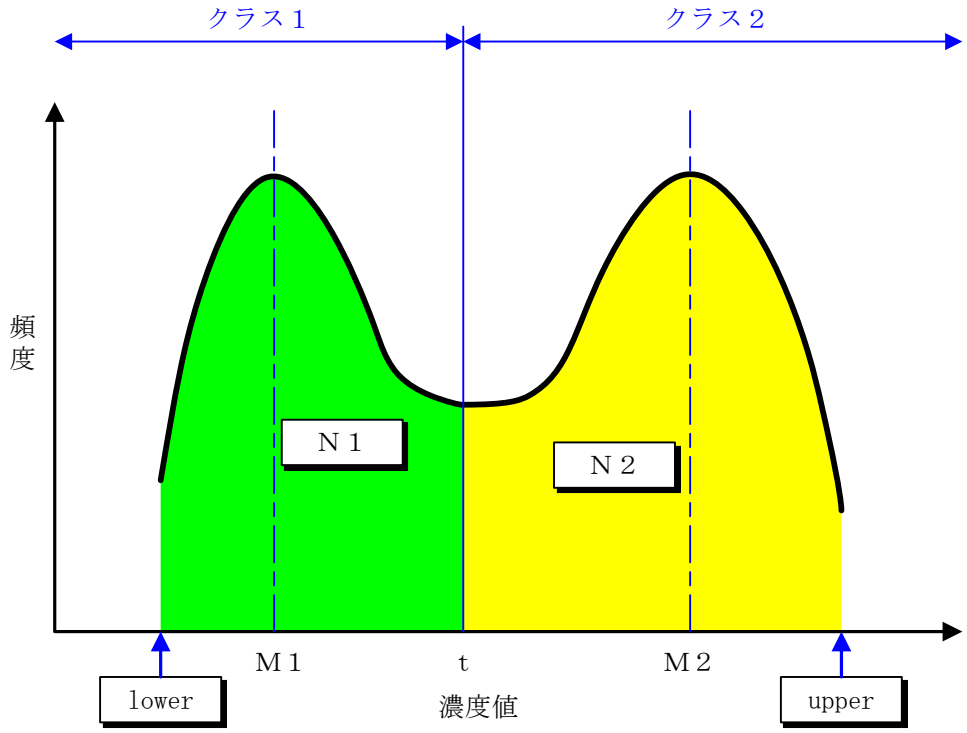
❑ 機能

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）	ｍ２ － ｍ１（簡易コントラスト）



上図において

Ｎ１	：クラス１の総画素数
Ｎ２	：クラス２の総画素数
Ｍ１	：クラス１の平均濃度値
Ｍ２	：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、

$$\sigma^2 = N1 \cdot N2 \cdot (M1 - M2)^2$$

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Ｎ１，Ｎ２，Ｍ１，Ｍ２を求めます。

2.27 拡張コンボリューション

2.27.1 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 ～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

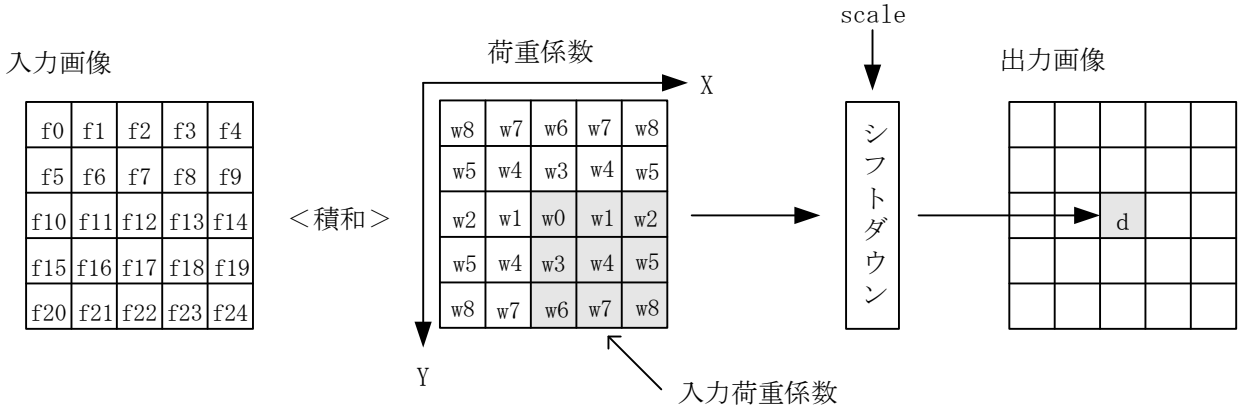
２	ＯＳでのエラー(タイムアウト以外)
３	IMP処理実行タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー)
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
２２	ダウンシフト量設定エラー
２３	荷重係数設定エラー
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４８	動的メモリ確保エラー (ユーザープログラムによるメモリ破壊がないか確認して下さい)
４９	アクセス画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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（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。

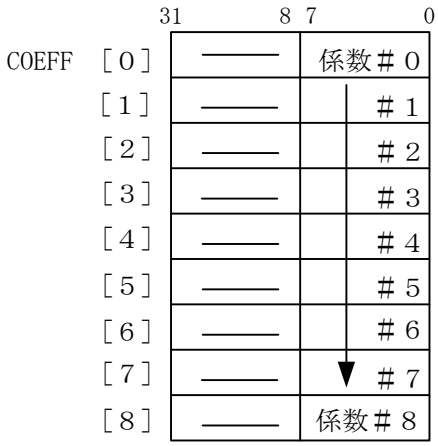



```
d =(  f0 × w8 +  f1 × w7 +  f2 × w6 +  f3 × w7 +  f4 × w8
   +  f5 × w5 +  f6 × w4 +  f7 × w3 +  f8 × w4 +  f9 × w5
   + f10 × w2 + f11 × w1 + f12 × w0 + f13 × w1 + f14 × w2
   + f15 × w5 + f16 × w4 + f17 × w3 + f18 × w4 + f19 × w5
   + f20 × w8 + f21 × w7 + f22 × w6 + f23 × w7 + f24 × w8
   )>>scale
```

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



＜ 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。

2.27.2 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	係数テーブルを格納したアドレス テーブルはint型で１６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

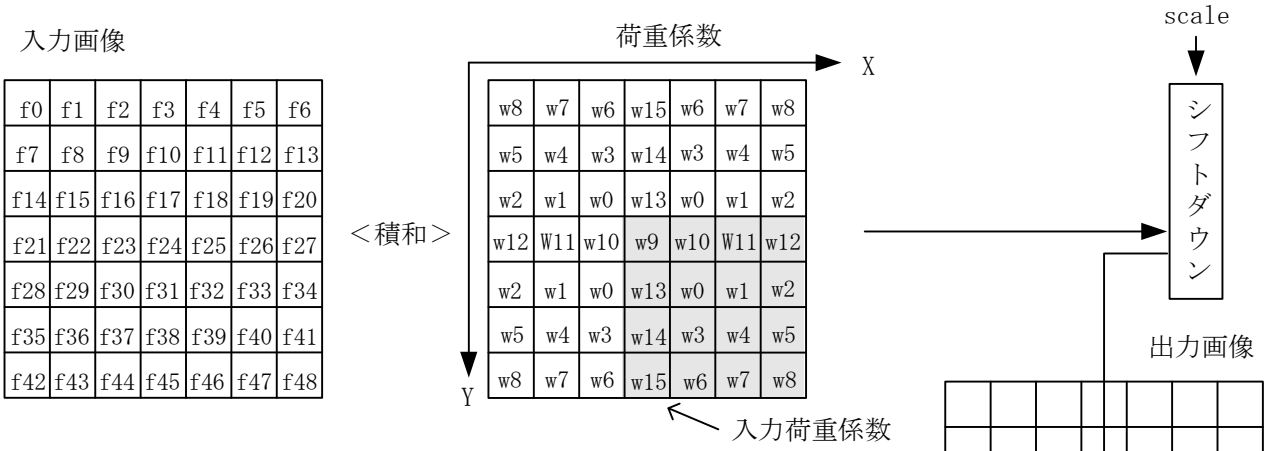
２	ＯＳでのエラー(タイムアウト以外)
３	IMP処理実行タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー)
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
２２	ダウンシフト量設定エラー
２３	荷重係数設定エラー
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４８	動的メモリ確保エラー (ユーザープログラムによるメモリ破壊がないか確認して下さい)
４９	アクセス画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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（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。

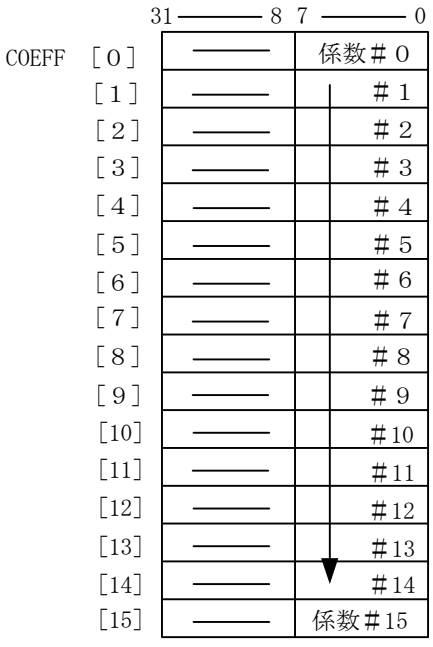


f0	f1	f2	f3	f4	f5	f6
f7	f8	f9	f10	f11	f12	f13
f14	f15	f16	f17	f18	f19	f20
f21	f22	f23	f24	f25	f26	f27
f28	f29	f30	f31	f32	f33	f34
f35	f36	f37	f38	f39	f40	f41
f42	f43	f44	f45	f46	f47	f48

w8	w7	w6	w15	w6	w7	w8
w5	w4	w3	w14	w3	w4	w5
w2	w1	w0	w13	w0	w1	w2
w12	W11	w10	w9	w10	W11	w12
w2	w1	w0	w13	w0	w1	w2
w5	w4	w3	w14	w3	w4	w5
w8	w7	w6	w15	w6	w7	w8

```
d =(  f0 ×w8 +f1 ×w7 + f2 ×w6 + f3 ×w15 + f4 ×w6 + f5 ×w7 + f6 ×w8
    + f7 ×w5 +f8 ×w4 + f9 ×w3 +f10 ×w14 +f11 ×w3 +f12 ×w4 +f13 ×w5
                              ・
                              ・
                              ・
    +f21 ×w12 +f22 ×w11 +f23 ×w10 +f24 ×w9 +f25 ×w10 +f26 ×w11 +f20 ×w12
                              ・
                              ・
                              ・
    +f42 ×w8 +f43 ×w7 + f44 ×w6 + f45 ×w15 + f46 ×w6 + f47 ×w7 + f48 ×w8
  )>>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	#9	#10	#11
#12	#13	#14	#15

〈 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	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	0
0	2	2	2	2	2	0
0	2	2	2	2	2	0
0	2	2	2	2	2	0
0	2	2	2	2	2	0
0	2	2	2	2	2	0
0	0	0	0	0	0	0

この例の場合、ダウンシフト量は６です。

2.27.3 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 ～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

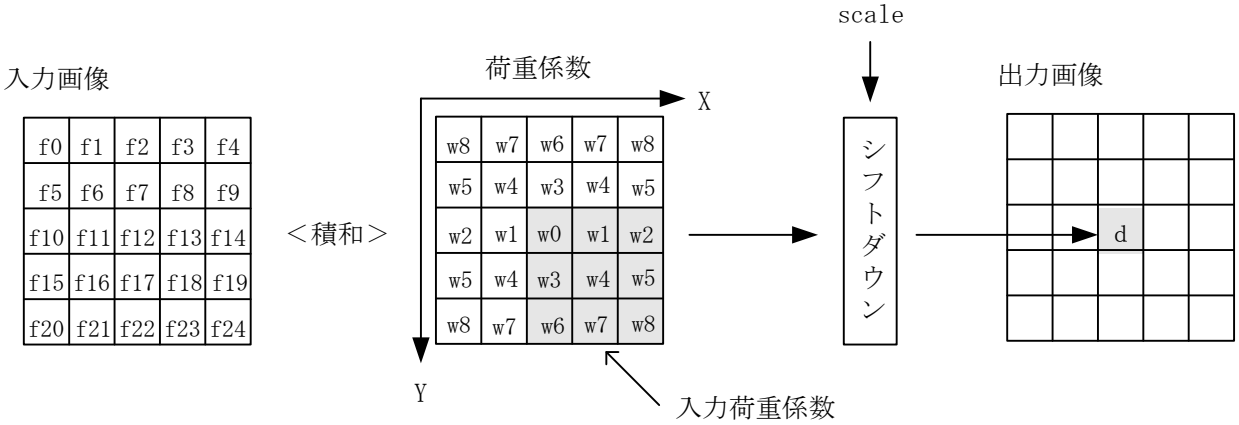
２	ＯＳでのエラー(タイムアウト以外)
３	IMP処理実行タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー)
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
２２	ダウンシフト量設定エラー
２３	荷重係数設定エラー
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４８	動的メモリ確保エラー (ユーザープログラムによるメモリ破壊がないか確認して下さい)
４９	アクセス画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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（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。




```
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。

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。

2.27.4 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	係数テーブルを格納したアドレス テーブルはint型で１６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

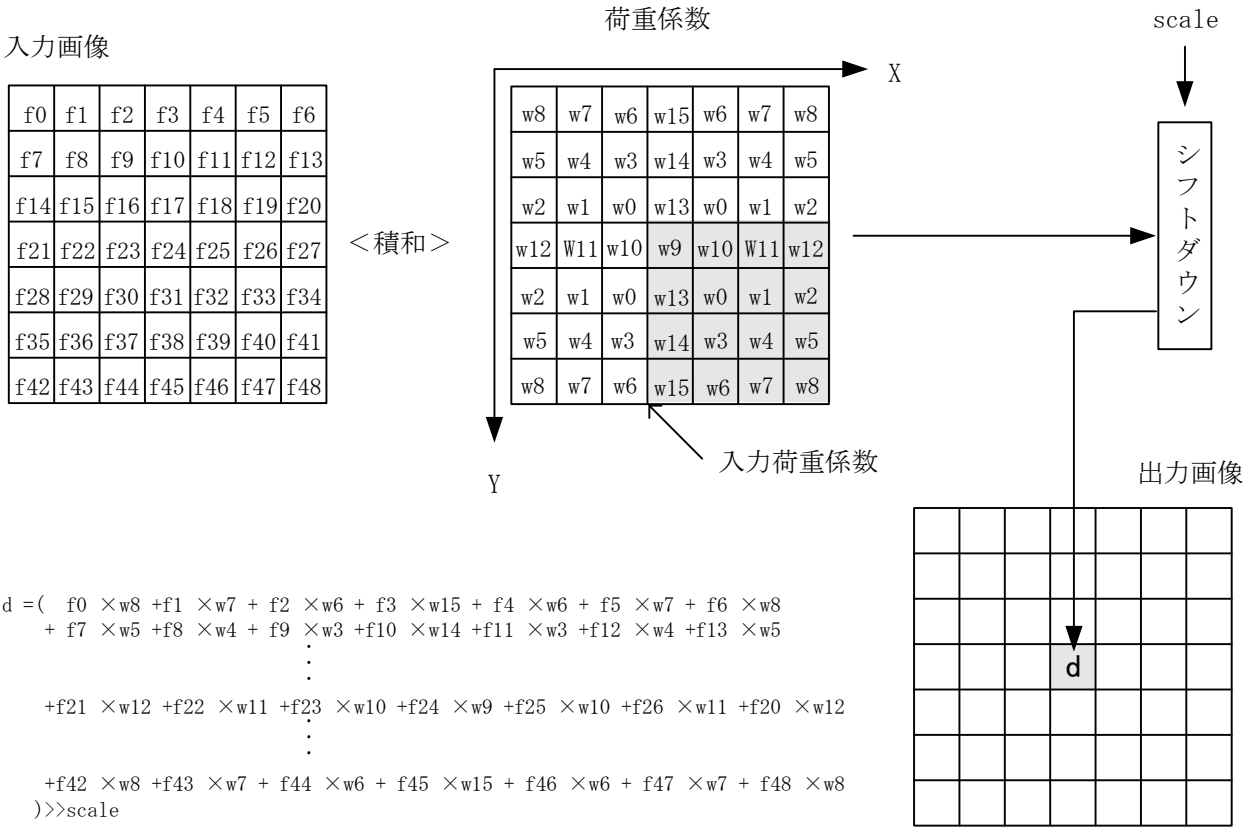
２	ＯＳでのエラー(タイムアウト以外)
３	IMP処理実行タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー)
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
２２	ダウンシフト量設定エラー
２３	荷重係数設定エラー
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４８	動的メモリ確保エラー (ユーザープログラムによるメモリ破壊がないか確認して下さい)
４９	アクセス画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 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（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。



f0	f1	f2	f3	f4	f5	f6
f7	f8	f9	f10	f11	f12	f13
f14	f15	f16	f17	f18	f19	f20
f21	f22	f23	f24	f25	f26	f27
f28	f29	f30	f31	f32	f33	f34
f35	f36	f37	f38	f39	f40	f41
f42	f43	f44	f45	f46	f47	f48

w8	w7	w6	w15	w6	w7	w8
w5	w4	w3	w14	w3	w4	w5
w2	w1	w0	w13	w0	w1	w2
w12	W11	w10	w9	w10	W11	w12
w2	w1	w0	w13	w0	w1	w2
w5	w4	w3	w14	w3	w4	w5
w8	w7	w6	w15	w6	w7	w8

```
d =(  f0 ×w8 +f1 ×w7 + f2 ×w6 + f3 ×w15 + f4 ×w6 + f5 ×w7 + f6 ×w8
    + f7 ×w5 +f8 ×w4 + f9 ×w3 +f10 ×w14 +f11 ×w3 +f12 ×w4 +f13 ×w5
                          :
                          :
    +f21 ×w12 +f22 ×w11 +f23 ×w10 +f24 ×w9 +f25 ×w10 +f26 ×w11 +f20 ×w12
                          :
                          :
    +f42 ×w8 +f43 ×w7 + f44 ×w6 + f45 ×w15 + f46 ×w6 + f47 ×w7 + f48 ×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_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。

2.27.5　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

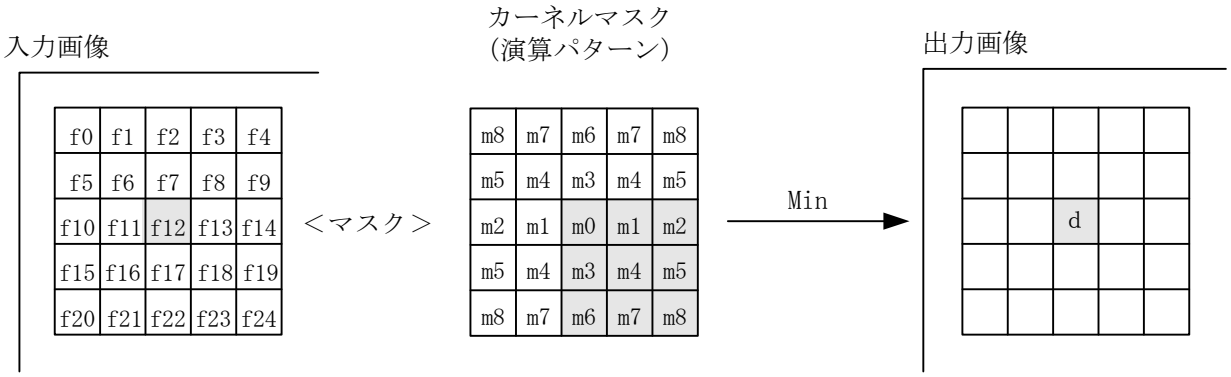
２	ＯＳでのエラー(タイムアウト以外)
３	IMP処理実行タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４８	動的メモリ確保エラー (ユーザープログラムによるメモリ破壊がないか確認して下さい)
４９	アクセス画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

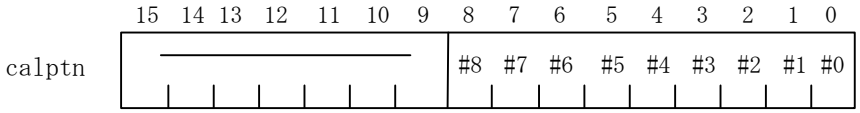
❑ 機能

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。

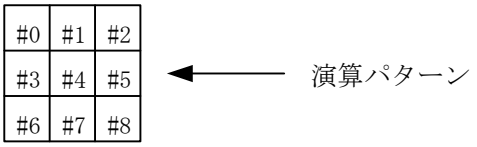
局所最小値フィルタ（５×５近傍）では、入力画面の指定領域内の全画素に対して、注目画素を中心とした５×５近傍の局所領域で与えられた演算パターンでマスク処理し、有効となった画素の中から最小の画素を出力します。
また、対象領域の周辺（内側３画素の領域）処理の場合は不定値を出力します。



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



2.27.6 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

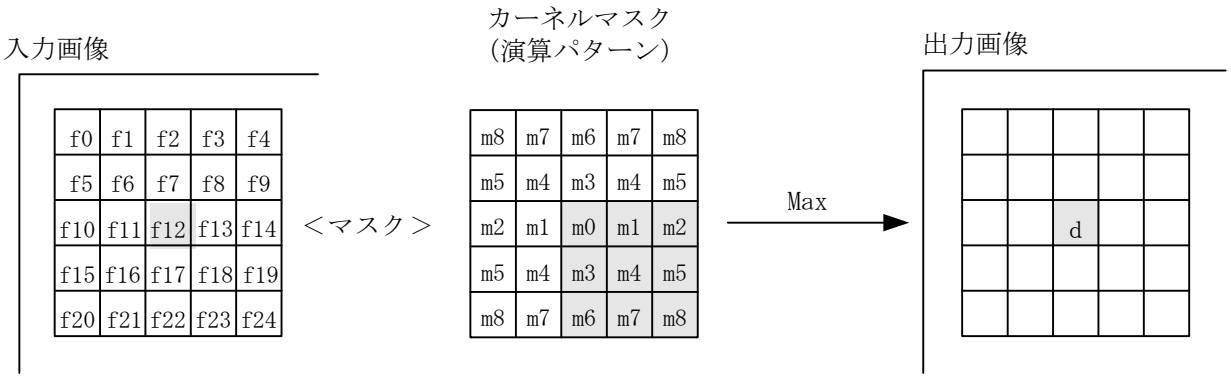
２	ＯＳでのエラー(タイムアウト以外)
３	IMP処理実行タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー)
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４８	動的メモリ確保エラー (ユーザープログラムによるメモリ破壊がないか確認して下さい)
４９	アクセス画面が不定画面
－	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

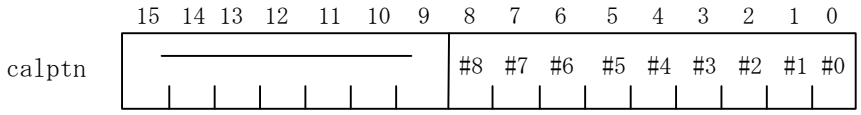
❑ 機能

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。

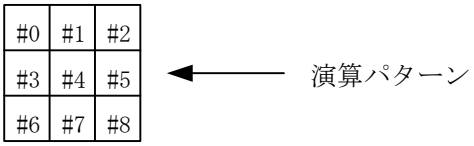
局所最大値フィルタ（５×５近傍）では、入力画面の指定領域内の全画素に対して、注目画素を中心とした５×５近傍の局所領域で与えられた演算パターンでマスク処理し、有効となった画素の中から最大の画素を出力します。
また、対象領域の周辺（内側３画素の領域）処理の場合は不定値を出力します。



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



2.27.7 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ＵＰ フィルタ：詳細情報参照
SENSUP4	４倍感度ＵＰ 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	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
２２	繰り返し回数設定エラー
２３	平滑化方式設定エラー
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
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)
４９	アクセス画面が不定画面
５１	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「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。

2.27.8　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 IPMEdgeMethod 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(｜水平エッジ｜,｜垂直エッジ｜)：詳細情報（A),（B）参照
EDGEADDVH	｜水平エッジ｜＋｜垂直エッジ｜　：詳細情報（A),（B）参照
EDGEV	｜垂直エッジ｜：詳細情報（A),（B）参照
EDGEH	｜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_SOBEL	｜水平エッジ｜＋｜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）

int　lutno　　拡張濃度変換テーブル登録番号

０　　：NOOP
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
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
２２	輪郭強調手法設定エラー
２３	輪郭強調ゲインまたは乗算のダウンシフト量設定エラー
２４	拡張濃度変換テーブル登録番号
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	アクセス画面が不定画面
５１	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

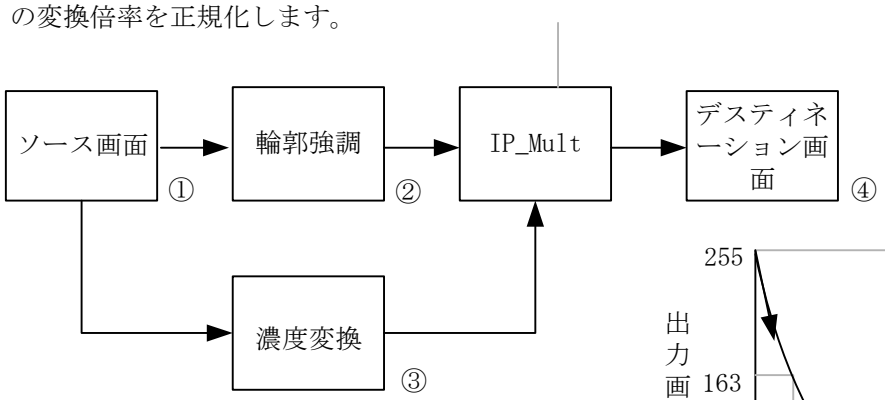
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

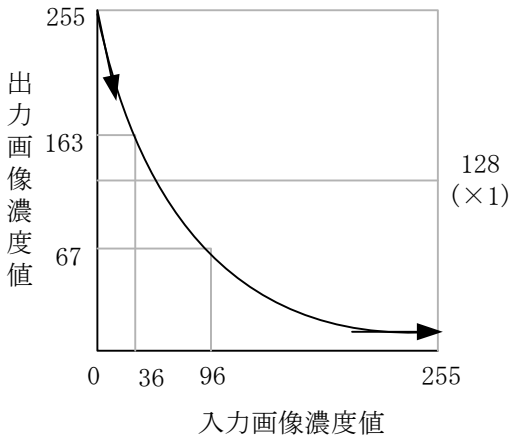
❑ 機能

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（÷１２８）し、濃度補正の変換倍率を正規化します。

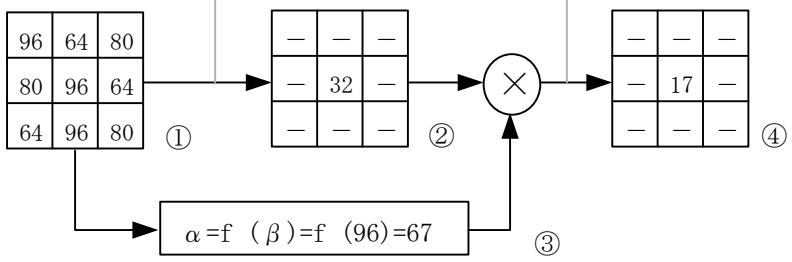




startvalue:255 startgrad:-3.0
endvalue :16 endgrad :-0.025

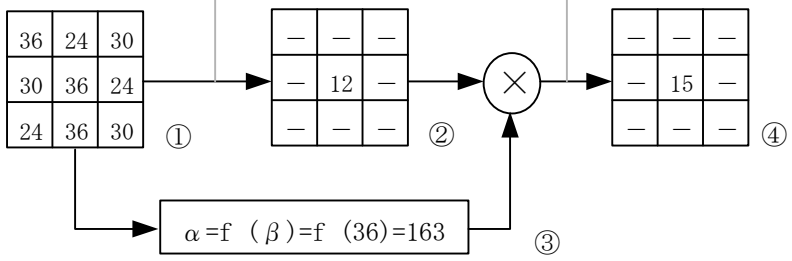
垂直エッジ(gain:1)（７ビットシフトダウン）

例１



垂直エッジ(gain:1)（７ビットシフトダウン）

例２

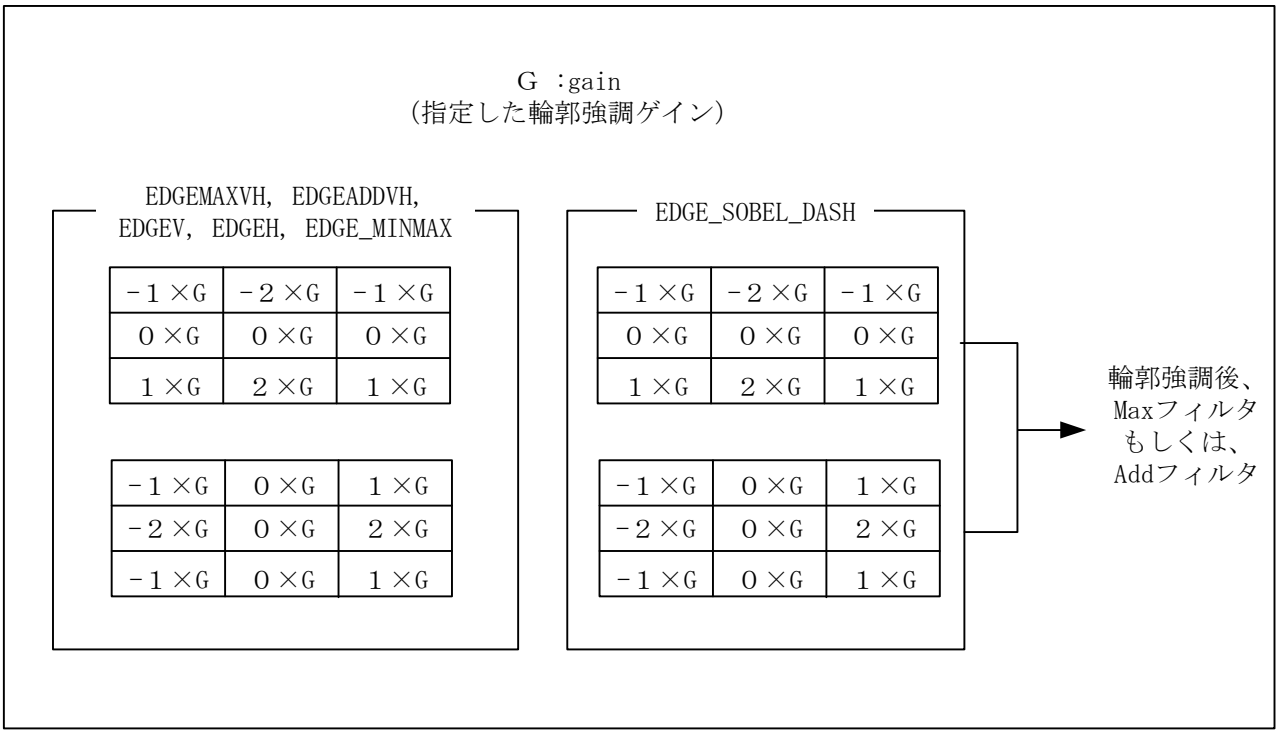


左図の例１、２の様に濃度補正を行う事で、同一パターンで出力される値が均一化されます。
濃度変換は、上図のテーブルを指定します。
①ソース画面のサンプル
②垂直エッジの濃度値
③濃度変換α=f（β）（右上図）
④濃度適応処理結果（四捨五入されます）

［B］ 係数テーブル

輪郭強調フィルタで、EDGEMAXVH,EDGEADDVH,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



2.27.9 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

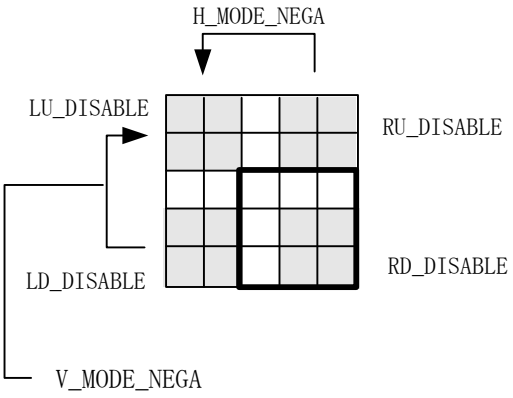
IMGID　ImgSrc　　ソース画面の画面番号

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　　拡張コンボリューションオプション

オプション	対応定数	内容
H_MODE_NEGA	３２	負対称（水平方向）
V_MODE_NEGA	６４	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　　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

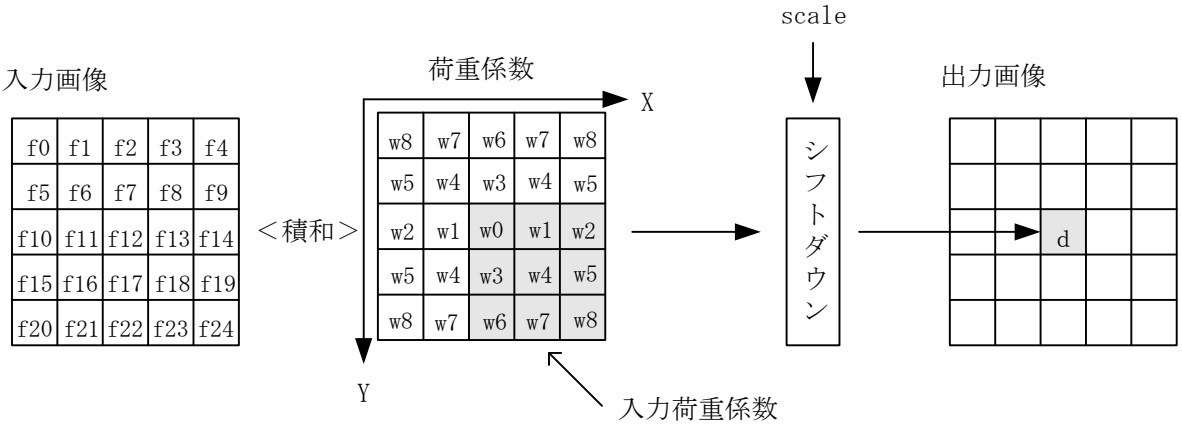
２	ＯＳでのエラー(タイムアウト以外)
３	IMP処理実行タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
２２	ダウンシフト量設定エラー
２３	拡張コンボリューションオプション設定エラー
２４	荷重係数設定エラー
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４８	動的メモリ確保エラー (ユーザープログラムによるメモリ破壊がないか確認して下さい)
４９	アクセス画面が不定画面
－	エラーリセットコマンド（ClearIPError）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（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。



f0	f1	f2	f3	f4
f5	f6	f7	f8	f9
f10	f11	f12	f13	f14
f15	f16	f17	f18	f19
f20	f21	f22	f23	f24

w8	w7	w6	w7	w8
w5	w4	w3	w4	w5
w2	w1	w0	w1	w2
w5	w4	w3	w4	w5
w8	w7	w6	w7	w8

```
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。

荷重係数

＜ 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

opt == 0

＜ 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

opt == 0

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

2.27.10　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

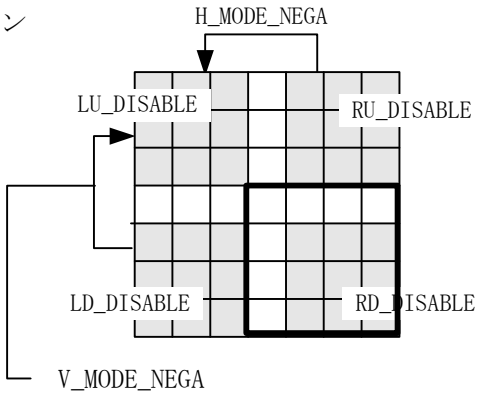
IMGID　ImgSrc　　ソース画面の画面番号

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 ～１５ ）

enum IPConvExtOpt opt　　拡張コンボリューションオプション

オプション	対応定数	内容
H_MODE_NEGA	３２	負対称（水平方向）
V_MODE_NEGA	６４	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　　係数テーブルを格納したアドレス
テーブルはint型で１６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　　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
２２　　ダウンシフト量設定エラー
２３　　拡張コンボリューションオプション設定エラー
２４　　荷重係数設定エラー
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
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)
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

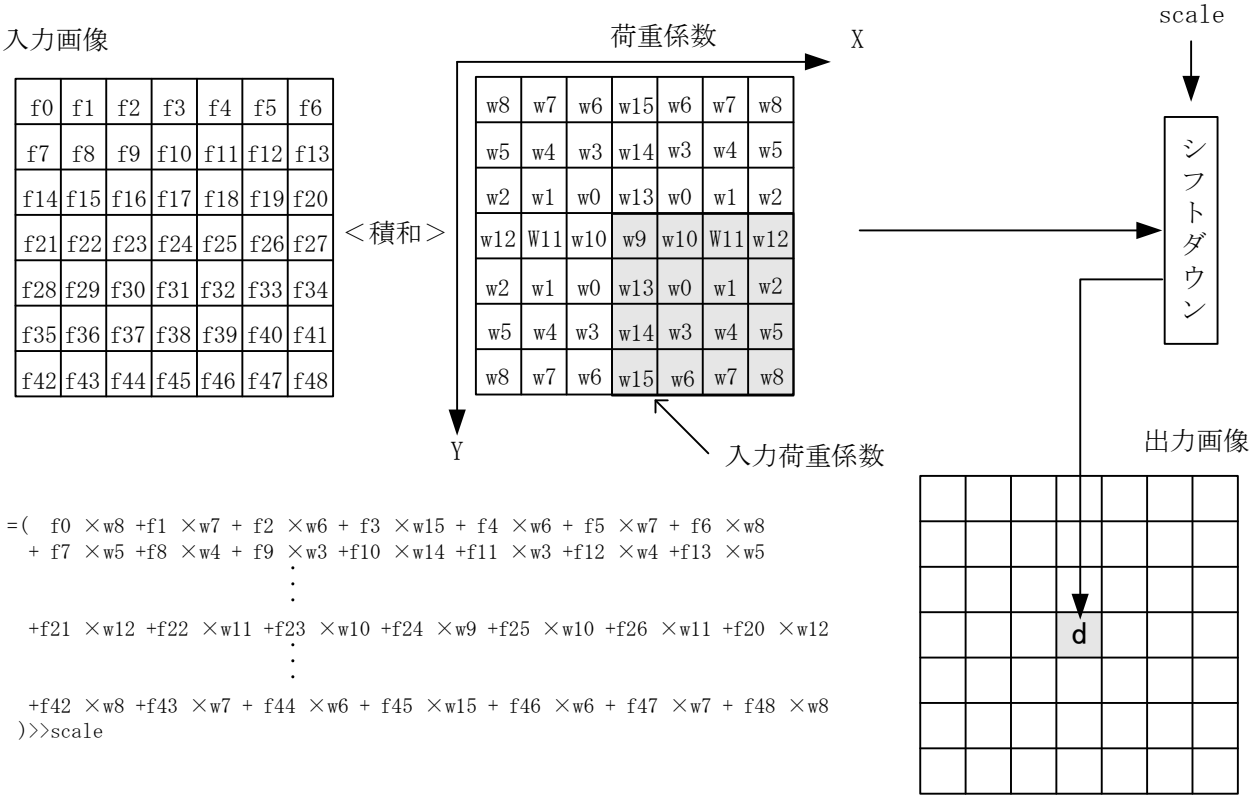
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

ソース画面（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。



f0	f1	f2	f3	f4	f5	f6
f7	f8	f9	f10	f11	f12	f13
f14	f15	f16	f17	f18	f19	f20
f21	f22	f23	f24	f25	f26	f27
f28	f29	f30	f31	f32	f33	f34
f35	f36	f37	f38	f39	f40	f41
f42	f43	f44	f45	f46	f47	f48

w8	w7	w6	w15	w6	w7	w8
w5	w4	w3	w14	w3	w4	w5
w2	w1	w0	w13	w0	w1	w2
w12	W11	w10	w9	w10	W11	w12
w2	w1	w0	w13	w0	w1	w2
w5	w4	w3	w14	w3	w4	w5
w8	w7	w6	w15	w6	w7	w8

```
d =(  f0 ×w8 +f1 ×w7 + f2 ×w6 + f3 ×w15 + f4 ×w6 + f5 ×w7 + f6 ×w8
    + f7 ×w5 +f8 ×w4 + f9 ×w3 +f10 ×w14 +f11 ×w3 +f12 ×w4 +f13 ×w5
                              :
                              :
    +f21 ×w12 +f22 ×w11 +f23 ×w10 +f24 ×w9 +f25 ×w10 +f26 ×w11 +f20 ×w12
                              :
                              :
    +f42 ×w8 +f43 ×w7 + f44 ×w6 + f45 ×w15 + f46 ×w6 + f47 ×w7 + f48 ×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_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

2.27.11　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

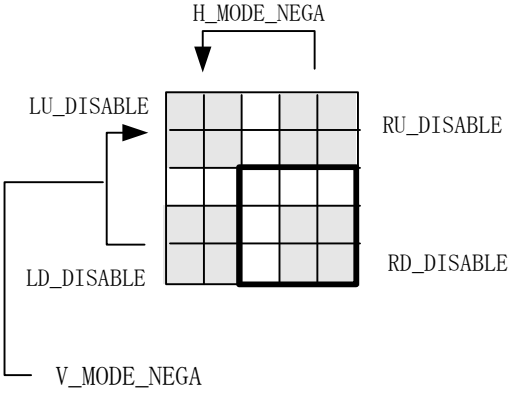
IMGID　ImgSrc　　ソース画面の画面番号

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 ～１５ ）

enum IPConvExtOpt opt　　拡張コンボリューションオプション

オプション	対応定数	内容
H_MODE_NEGA	３２	負対称（水平方向）
V_MODE_NEGA	６４	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　　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

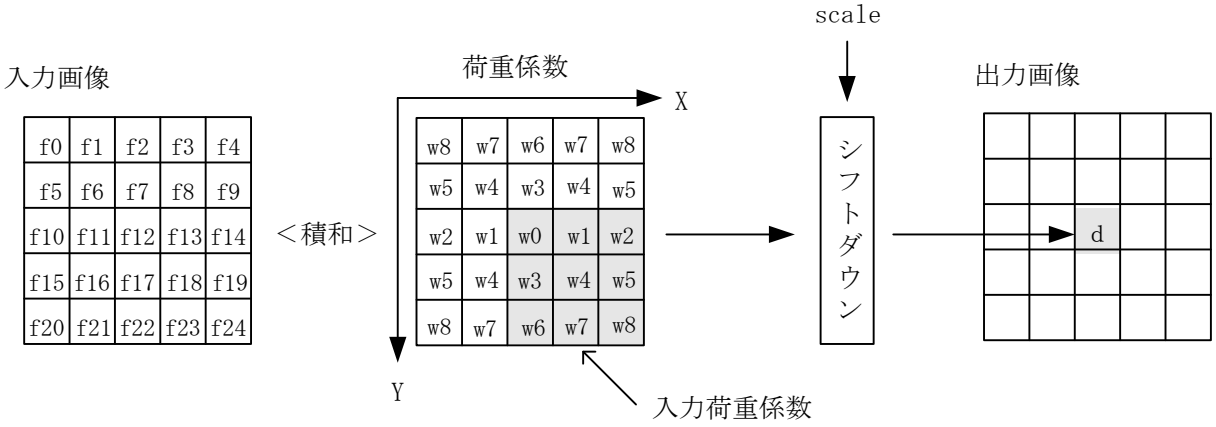
２	ＯＳでのエラー(タイムアウト以外)
３	IMP処理実行タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
２２	ダウンシフト量設定エラー
２３	拡張コンボリューションオプション設定エラー
２４	荷重係数設定エラー
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４８	動的メモリ確保エラー (ユーザープログラムによるメモリ破壊がないか確認して下さい)
４９	アクセス画面が不定画面
－	エラーリセットコマンド（ClearIPError）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（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。




```
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_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。

2.27.12　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

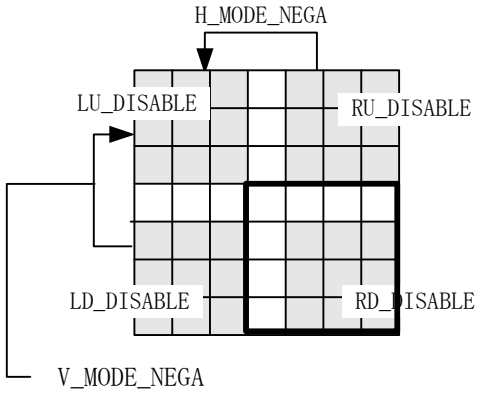
IMGID　ImgSrc　　ソース画面の画面番号

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 ～１５ ）

enum IPConvExtOpt opt　　拡張コンボリューションオプション

オプション	対応定数	内容
H_MODE_NEGA	３２	負対称（水平方向）
V_MODE_NEGA	６４	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　　係数テーブルを格納したアドレス
テーブルはint型で１６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

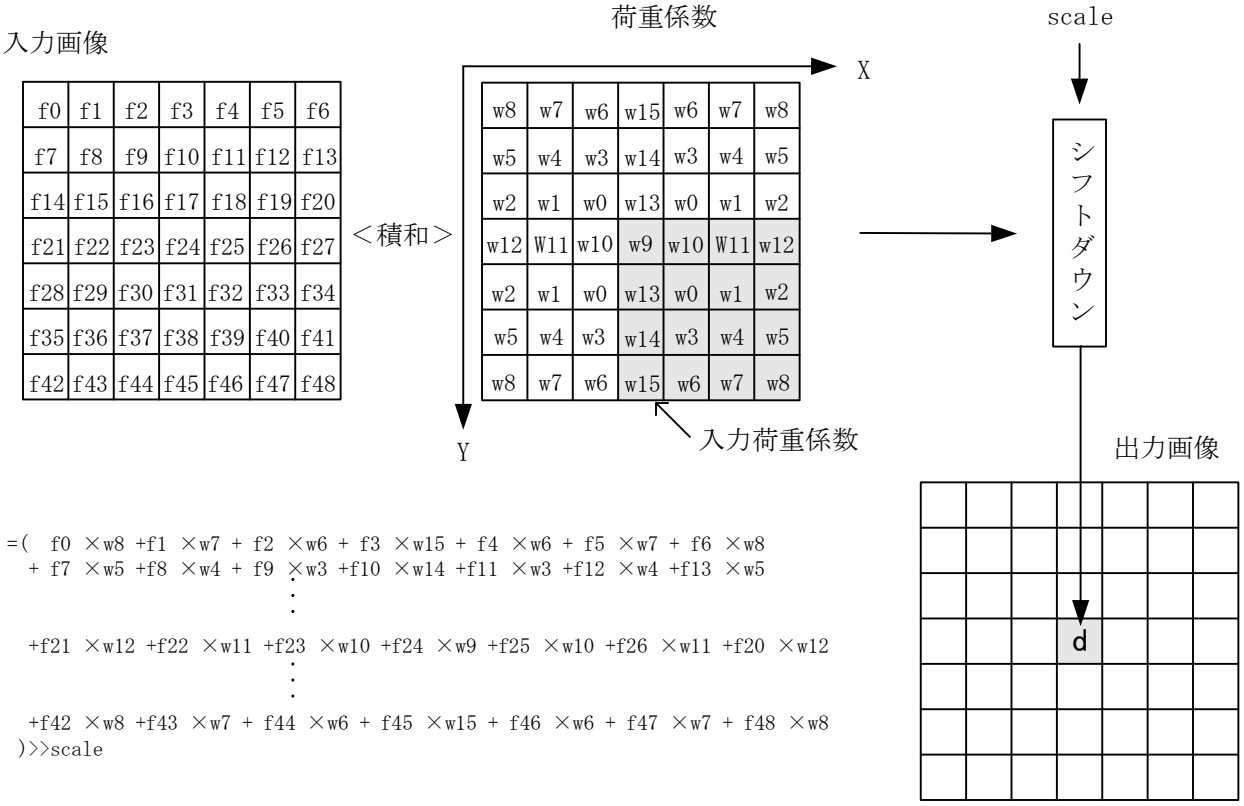
２	ＯＳでのエラー(タイムアウト以外)
３	IMP処理実行タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
２２	ダウンシフト量設定エラー
２３	拡張コンボリューションオプション設定エラー
２４	荷重係数設定エラー
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４８	動的メモリ確保エラー (ユーザープログラムによるメモリ破壊がないか確認して下さい)
４９	アクセス画面が不定画面
－	エラーリセットコマンド（ClearIPError）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（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。



f0	f1	f2	f3	f4	f5	f6
f7	f8	f9	f10	f11	f12	f13
f14	f15	f16	f17	f18	f19	f20
f21	f22	f23	f24	f25	f26	f27
f28	f29	f30	f31	f32	f33	f34
f35	f36	f37	f38	f39	f40	f41
f42	f43	f44	f45	f46	f47	f48

w8	w7	w6	w15	w6	w7	w8
w5	w4	w3	w14	w3	w4	w5
w2	w1	w0	w13	w0	w1	w2
w12	W11	w10	w9	w10	W11	w12
w2	w1	w0	w13	w0	w1	w2
w5	w4	w3	w14	w3	w4	w5
w8	w7	w6	w15	w6	w7	w8

```
d =(  f0 ×w8 +f1 ×w7 + f2 ×w6 + f3 ×w15 + f4 ×w6 + f5 ×w7 + f6 ×w8
    + f7 ×w5 +f8 ×w4 + f9 ×w3 +f10 ×w14 +f11 ×w3 +f12 ×w4 +f13 ×w5
                          :
                          :
    +f21 ×w12 +f22 ×w11 +f23 ×w10 +f24 ×w9 +f25 ×w10 +f26 ×w11 +f20 ×w12
                          :
                          :
    +f42 ×w8 +f43 ×w7 + f44 ×w6 + f45 ×w15 + f46 ×w6 + f47 ×w7 + f48 ×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_EdgeFLT7x7()の備考の欄を参照してください。

＜ 例1 ＞　　荷重係数　　＜ 例2 ＞

-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

-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

2.28 拡張画像処理

2.28.1 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

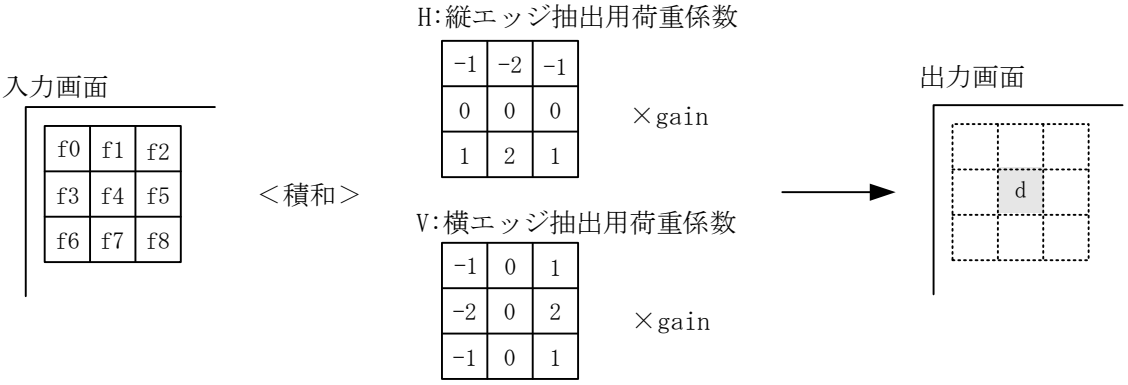
２	ＯＳでのエラー(タイムアウト以外)
３	IMP処理実行タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
２２	ゲイン値設定範囲外
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４８	動的メモリ確保エラー(ユーザープログラムによるメモリ破壊がないか確認して下さい)
４９	アクセス画面が不定画面
５１	ワーク画面確保エラー(不要な画面を解放してリトライして下さい)
－	エラーリセットコマンド（ClearIPError）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。



$$d = |f0\times(-gain) + f1\times(-2)\times gain + f2\times(-gain) + f6\times gain + f7\times 2\times gain + f8\times gain| + |f0\times(-gain) + f3\times(-2)\times gain + f6\times(-gain) + f2\times gain + f5\times 2\times gain + f8\times gain|$$

2.28.2　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

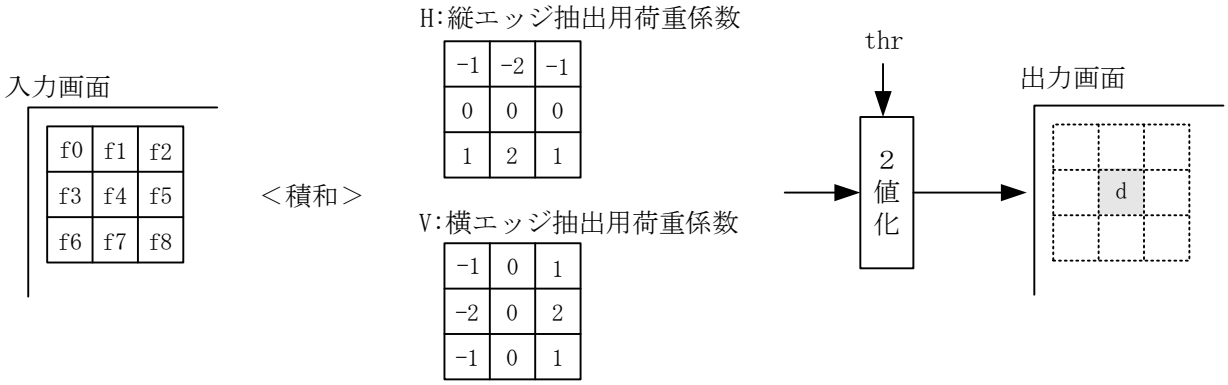
２	ＯＳでのエラー(タイムアウト以外)
３	IMP処理実行タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー)
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
２２	２値化しきい値設定範囲外
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４８	動的メモリ確保エラー(ユーザープログラムによるメモリ破壊がないか確認して下さい)
４９	アクセス画面が不定画面
５１	ワーク画面確保エラー(不要な画面を解放してリトライして下さい)
－	エラーリセットコマンド（ClearIPError）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。

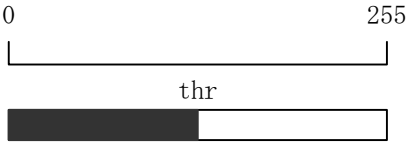
❑ 機能

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。



$$d = |f0\times(-1) + f1\times(-2) + f2\times(-1) + f6 + f7\times2 + f8| + |f0\times(-1) + f3\times(-2) + f6\times(-1) + f2 + f5\times2 + f8|$$



2.28.3 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

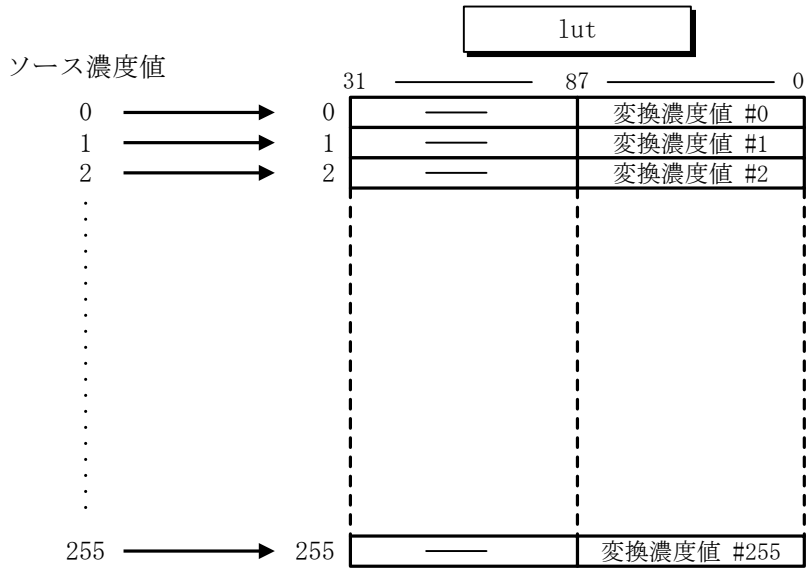
２０	濃度変換テーブル登録番号設定範囲外
２１	濃度変換データ格納テーブル設定エラー
－	エラーリセットコマンド（ClearIPError）未発行

※ 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



2.28.4 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

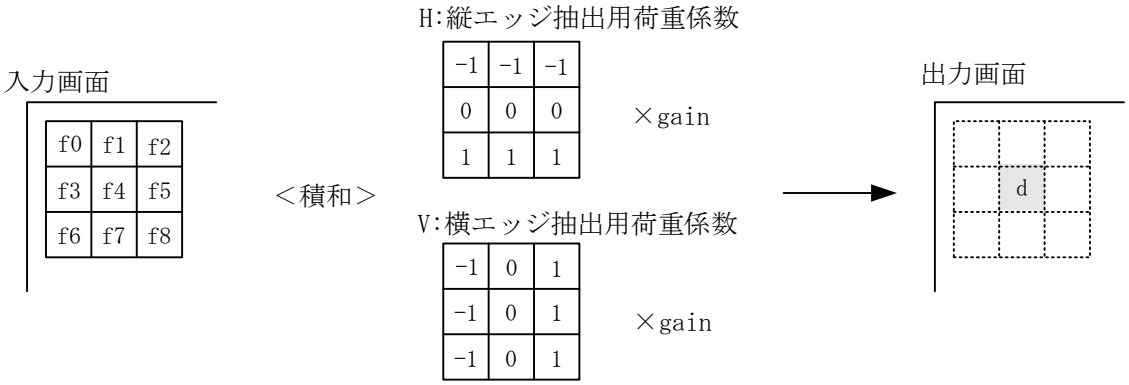
２	ＯＳでのエラー(タイムアウト以外)
３	IMP処理実行タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
２２	ゲイン値設定範囲外
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４８	動的メモリ確保エラー(ユーザープログラムによるメモリ破壊がないか確認して下さい)
４９	アクセス画面が不定画面
５１	ワーク画面確保エラー(不要な画面を解放してリトライして下さい)
－	エラーリセットコマンド（ClearIPError）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×(-gain) ＋ f1×(-2)×gain ＋ f2×(-gain) ＋ f6 ×gain ＋ f7×2×gain ＋ f8×gain |
＋|f0×(-gain) ＋ f3×(-2)×gain ＋ f6×(-gain) ＋ f2 ×gain ＋ f5×2×gain ＋ f8×gain |

2.28.5　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

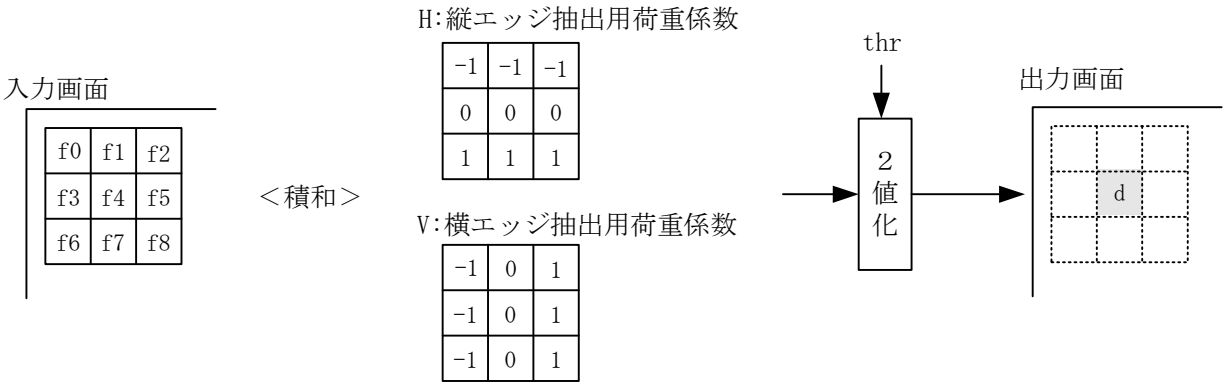
２	ＯＳでのエラー(タイムアウト以外)
３	IMP処理実行タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー)
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外
２２	２値化しきい値設定範囲外
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４８	動的メモリ確保エラー(ユーザープログラムによるメモリ破壊がないか確認して下さい)
４９	アクセス画面が不定画面
５１	ワーク画面確保エラー(不要な画面を解放してリトライして下さい)
－	エラーリセットコマンド（ClearIPError）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。

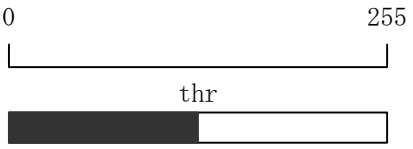
❑ 機能

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。



$$d = |f0\times(-1) + f1\times(-2) + f2\times(-1) + f6 + f7\times2 + f8| + |f0\times(-1) + f3\times(-2) + f6\times(-1) + f2 + f5\times2 + f8|$$



2.29 線分化

2.29.1 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

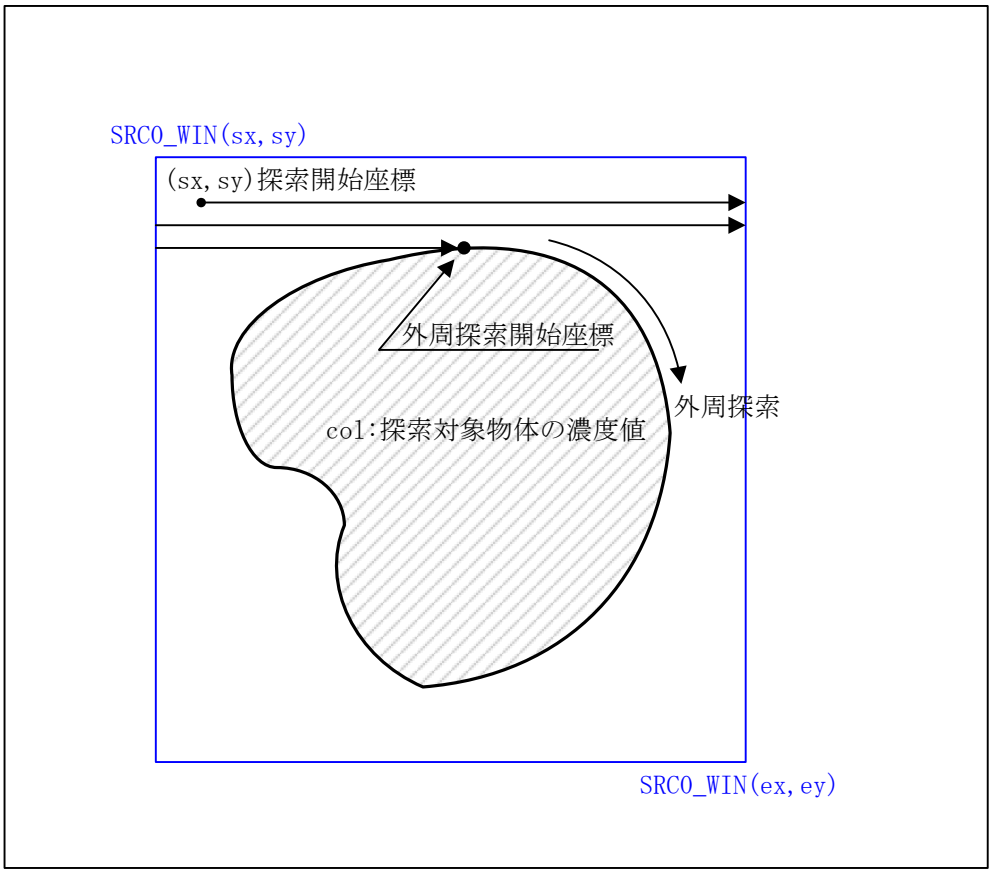
２	ＯＳでのエラー(タイムアウト以外)
３	IMP処理実行タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
２１	探索開始Ｘ座標設定エラー
２２	探索開始Ｙ座標設定エラー
２３	探索対象物体の濃度値設定エラー
２４	探索オプション設定エラー
２５	線分列抽出座標格納テーブルヲ格納したアドレスが不正
２６	抽出線分列最大数設定エラー
３０	ソース画面ウィンドウ設定エラー
４９	アクセス画面が不定画面
５６	線分列未検出エラー
－	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

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SRCO_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;
```

2.29.2　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

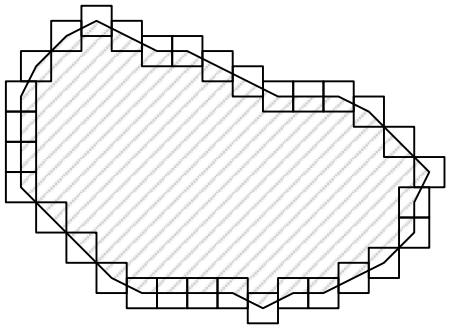
２０	線分列抽出座標を格納したアドレスが不正、または有効データ数範囲外
２１	面積を格納したアドレスが不正
－	エラーリセットコマンド（ClearIPError）未発行

※ 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。

2.29.3　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

２０	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
２１	周囲長（画素）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

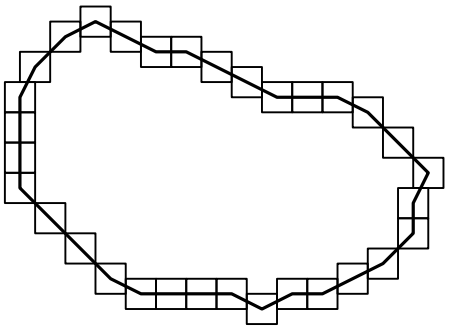
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 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。

2.29.4　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

２０	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
２１	重心Ｘ座標を格納したアドレスが不正
２２	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。

2.29.5　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	外接座標＋周囲長＋面積＋重心 ＋モーメント＋慣性主軸角度
POLY_AXIS_LENG	2	外接座標＋周囲長＋面積＋重心 ＋モーメント＋慣性主軸角度 ＋慣性主軸長＋慣性主軸長距離＋慣性主軸長座標
POLY_MAX_LENG	3	外接座標＋周囲長＋面積＋重心 ＋モーメント＋慣性主軸角度 ＋慣性主軸長＋慣性主軸長距離＋慣性主軸長座標 ＋絶対最大長＋絶対最大長距離＋絶対最大長座標

❑ 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）
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

２０	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
２１	ヒストグラム特徴量テーブルのアドレスが不正
２２	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。

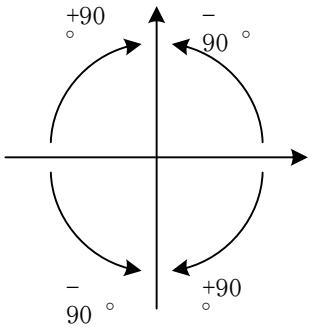
POLY_FEATURE のフォーマット及び内容を、下記に示します。

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	絶対最大長の方向と水平線との角度（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）



2.30 2値画像の穴埋め

2.30.1 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	○	○	○	×	△Ｂ２	△Ｂ２	○
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（０ｘ７ＦＦＦ）

❑ 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　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
２２　オプション設定エラー
２３　穴埋めする穴の色設定エラー
２４　穴埋めする穴の面積閾値設定エラー
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
５１　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、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。

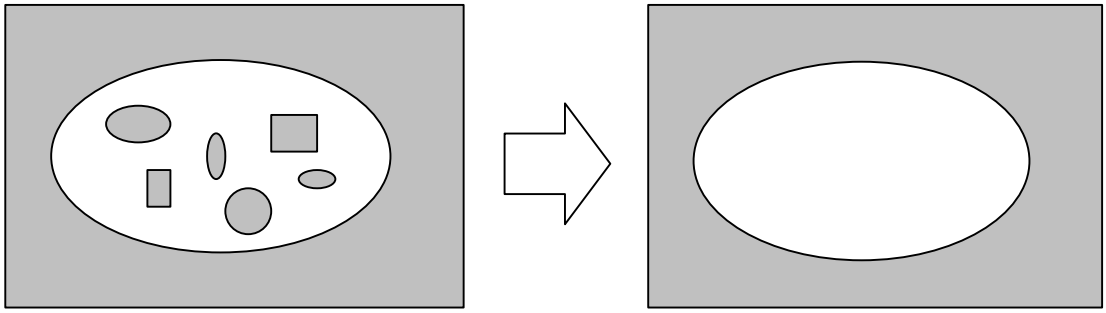
OBJ_AREA_OPTのフォーマットを下記に示します。

ipxsys.hで宣言しています。

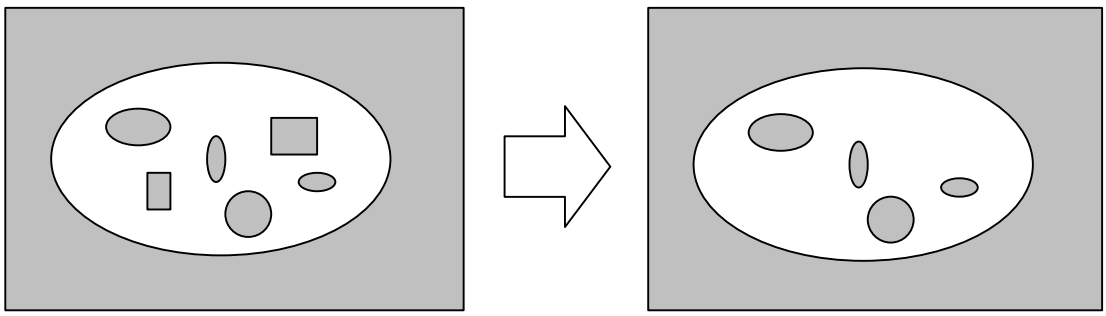
```
typedef struct {
   int      min;     // 最小面積
   int      max;     // 最大面積
} OBJ_AREA_OPT;
```

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



2.30.2　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	○	○	○	×	△Ｂ２	△Ｂ２	○
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（０ｘ７ＦＦＦ）

❑ 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　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
２２　オプション設定エラー
２３　穴埋めする穴の色設定エラー
２４　穴埋めする穴の面積閾値設定エラー
２５　穴埋めする対象物体の抽出面積閾値設定エラー
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
５１　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

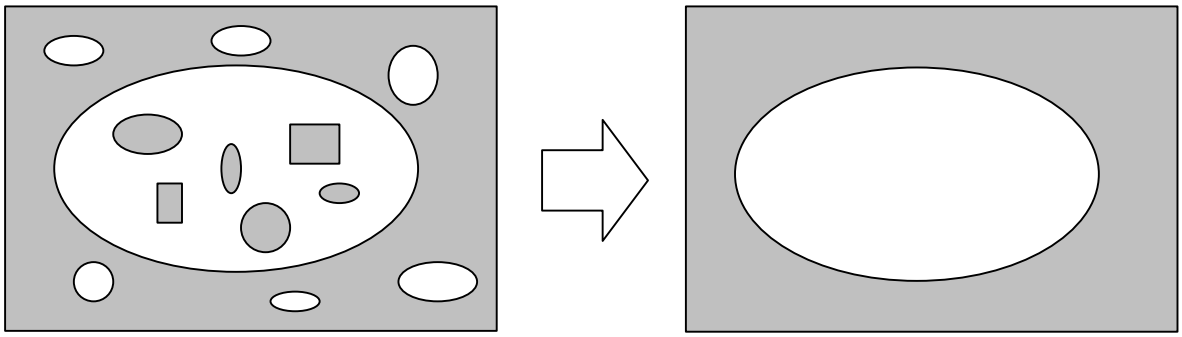
２値画像の対象物体の抽出を行い抽出した対象物の穴埋めを行います。拡張機能として穴埋めする対象物体の抽出面積閾値の設定ができます。

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。

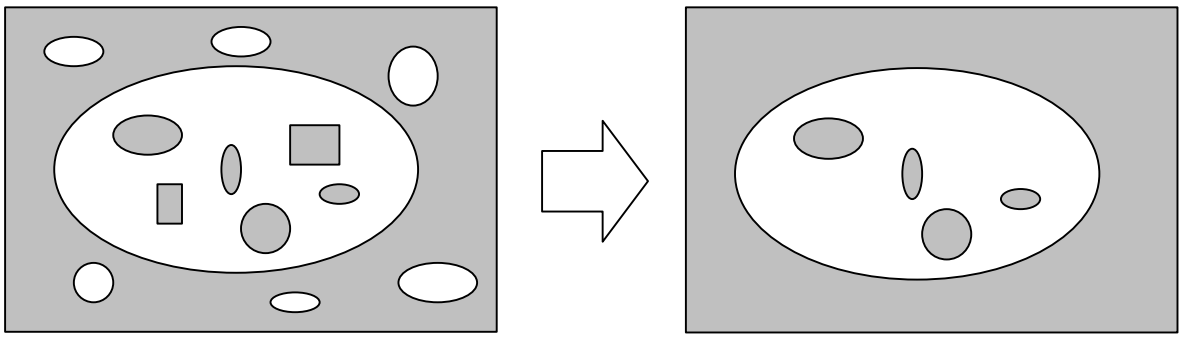
なお、本コマンドではラベリング処理を用いているため、２５５個以上の穴埋めはできません。

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



2.31 RGBLUT変換

2.31.1 RGBLUTオブジェクトの生成（ CreateRGBLUT ）

❑ C言語API

```
HRGBLUT ret = CreateRGBLUT( int mode , int opt );
```

❑ パラメータ

int　　mode　　ＬＵＴモード

ＬＵＴモード	定数値	内　容
RGBLUT888	0	R:8bit / G:8bit / B:8bit (16,777,216byte)
RGBLUT777	1	R:7bit / G:7bit / B:7bit (2,097,152byte)
RGBLUT666	2	R:6bit / G:6bit / B:6bit (262,144byte)
RGBLUT887	3	R:8bit / G:8bit / B:7bit (8,388,608byte)
RGBLUT878	4	R:8bit / G:7bit / B:8bit (8,388,608byte)
RGBLUT788	5	R:7bit / G:8bit / B:8bit (8,388,608byte)
RGBLUT787	6	R:7bit / G:8bit / B:7bit (4,194,304byte)
RGBLUT2CH88	7	R:8bit / G:8bit (65,536byte)
RGBLUT565	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

HRGBLUT　　ret　　－１、０以外：ＲＧＢＬＵＴハンドル
－１、０　　：異常終了

❑ エラーコード

２０　　ＬＵＴモードの設定範囲外
２１　　オプションの設定範囲外
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。

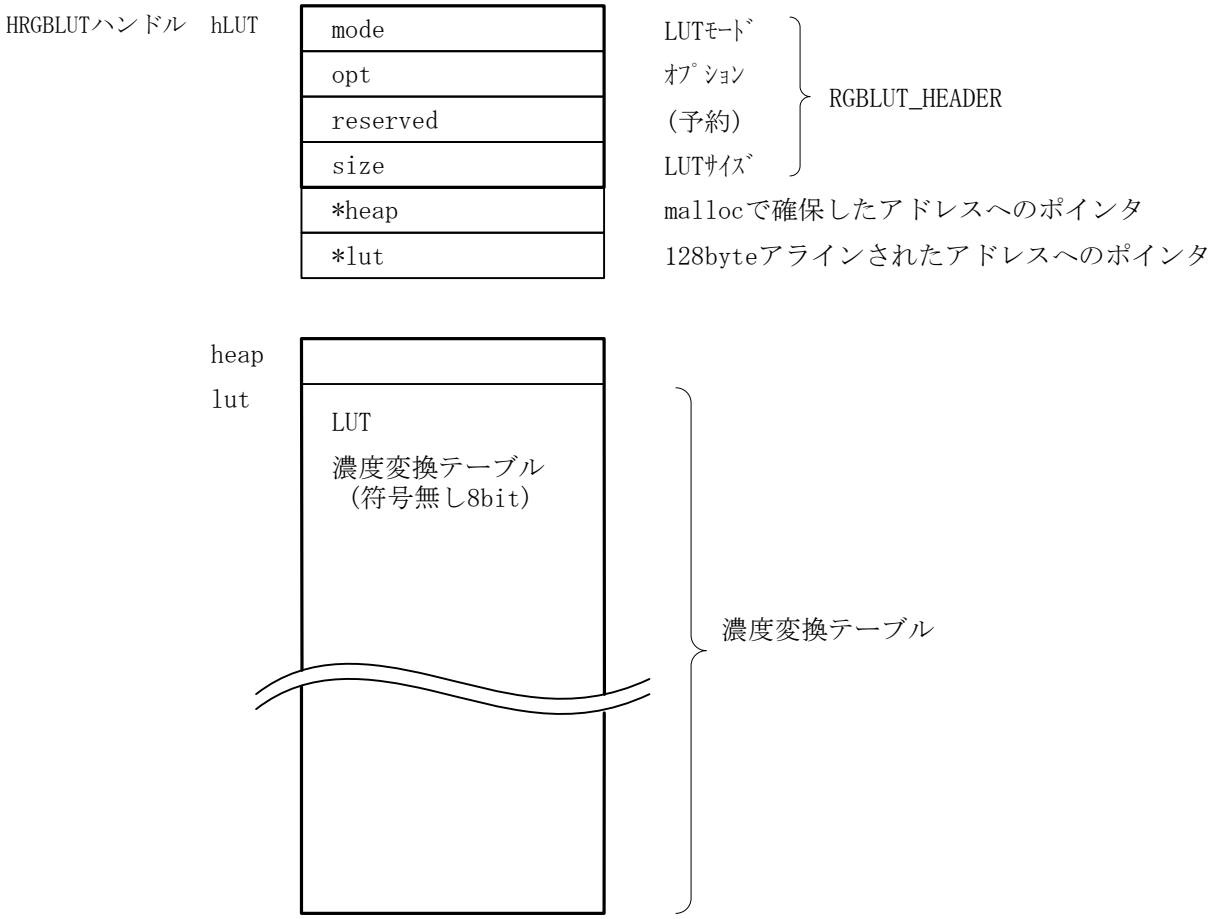
❑ 機能

ＲＧＢＬＵＴオブジェクトの生成を行います。ＲＧＢＬＵＴオブジェクトはオンボードＣＰＵ上に生成され、ＲＧＢＬＵＴハンドルで管理されます。ＲＧＢＬＵＴオブジェクトの構成を以下に示します。

ＬＵＴモード(mode)は、濃度変換処理と濃度変換テーブル(ＬＵＴ)の大きさを決定します。例えば、RGBLUT888は、Ｒデータが０～２５５(８bit)、Ｇデータが０～２５５(８bit)、Ｂデータが０～２５５(８bit)のＲＧＢカラー画像を濃淡画像０～２５５(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、OpenRGBLUT()の実行によりＬＵＴの先頭アドレスを取得して、直接、ＬＵＴをアクセスすることができます。ＬＵＴの設定後、IP_ConvertRGBLUT()およびIP_ConvertRGBLUTEx()で、ＲＧＢのＬＵＴ変換を実行します。

ＲＧＢＬＵＴ変換が終了したら、DeleteRGBLUT()でＲＧＢＬＵＴオブジェクトを削除してください。ＲＧＢＬＵＴ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。



2.31.2　RGBLUTオブジェクトの削除（ DeleteRGBLUT ）

❑ C言語API

```
int ret = DeleteRGBLUT(HRGBLUT hLUT);
```

❑ パラメータ

HRGBLUT	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

ＲＧＢＬＵＴオブジェクトの削除を行います。
CleateRGBLUT()を実行した場合は、必ず、本コマンドを実行してください。

2.31.3　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ＣＰＵ側からＰＣ側へ読み出さない。 １：ＬＵＴデータをＣＰＵ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

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
－	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

ＲＧＢＬＵＴハンドルで示されるＲＧＢＬＵＴオブジェクトの濃度変換テーブル(ＬＵＴ)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。

ＬＵＴ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ＬＵＴの先頭アドレス、ｒ、ｇ、ｂはそれぞれＲＧＢ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ＬＵＴのアドレスを示すパラメータになります。

ＬＵＴテーブルへのデータ書き込みマクロ

```
#define SetRGBLUT888(lut,r,g,b,d) lut[r << 16 | g << 8 | b]= (d)
#define SetRGBLUT777(lut,r,g,b,d) lut[(r & 0xfe) << 13 | (g & 0xfe) << 6 | b >> 1]= (d)
#define SetRGBLUT666(lut,r,g,b,d) lut[(r & 0xfc) << 10 | (g & 0xfc) << 4 | b >> 2]= (d)
#define SetRGBLUT887(lut,r,g,b,d) lut[r << 15 | g << 7 | b >> 1]= (d)
#define SetRGBLUT878(lut,r,g,b,d) lut[r << 15 | (g & 0xfe) << 7 | b]= (d)
#define SetRGBLUT788(lut,r,g,b,d) lut[(r & 0xfe) << 15 | g << 8 | b]= (d)
#define SetRGBLUT787(lut,r,g,b,d) lut[(r & 0xfe) << 14 | g << 7 | b >> 1]= (d)
#define SetRGBLUT2CH88(lut,r,g,d) lut[r << 8 | g]= (d)
#define SetRGBLUT565(lut,r,g,b,d) lut[(r & 0xf8) << 8  | (g & 0xfc) << 5 | b >> 3]= (d)
```

ＬＵＴテーブルへのデータ読み出しマクロ

```
#define GetRGBLUT888(lut,r,g,b)   lut[r << 16 | g << 8 | b]
#define GetRGBLUT777(lut,r,g,b)   lut[(r & 0xfe) << 13 | (g & 0xfe) << 6 | b >> 1]
#define GetRGBLUT666(lut,r,g,b)   lut[(r & 0xfc) << 10 | (g & 0xfc) << 4 | b >> 2]
#define GetRGBLUT887(lut,r,g,b)   lut[r << 15 | g << 7 | b >> 1]
#define GetRGBLUT878(lut,r,g,b)   lut[r << 15 | (g & 0xfe) << 7 | b]
#define GetRGBLUT788(lut,r,g,b)   lut[(r & 0xfe) << 15 | g << 8 | b]
#define GetRGBLUT787(lut,r,g,b)   lut[(r & 0xfe) << 14 | g << 7 | b >> 1]
#define GetRGBLUT2CH88(lut,r,g)   lut[r << 8 | g]
#define GetRGBLUT565(lut,r,g,b)   lut[(r & 0xf8) <<  8 | (g & 0xfc) << 5 | b >> 3]
```

リモートコマンドではＬＵＴ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ＣＰＵに書き込まれているＬＵＴデータを読み出してデータの変更を行います。ＬＵＴをオンボードＣＰＵから読み出す必要がない場合には、ＬＵＴ読み出しモード(mode)を"0"にしてください。ＬＵＴを読み出す時間を短縮することができます。

CloseRGBLUT()では、ＬＵＴデータをオンボードＣＰＵに書き戻し、ワーク領域を解放しますので、ＬＵＴの設定が終了したら、必ず、CloseRGBLUT()でＬＵＴのアクセスを終了してください。

2.31.4　RGBLUTオブジェクトの使用終了（ CloseRGBLUT ）

❑ C言語API

```
int ret = CloseRGBLUT(HRGBLUT hLUT , void *RGBTbl);
```

❑ パラメータ

HRGBLUT	hLUT	ＬＵＴハンドル
void	*RGBTbl	ＲＧＢＬＵＴ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

２０	ＲＧＢＬＵＴ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

ＲＧＢＬＵＴハンドルで示されるＲＧＢＬＵＴ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、OpenRGBLUT()で確保したワーク領域を解放します。OpenRGBLUT()実行後は、必ず、CloseRGBLUT()を実行してください。

2.31.5 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	ＲＧＢ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
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
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
２３	ＬＵＴの先頭アドレス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。

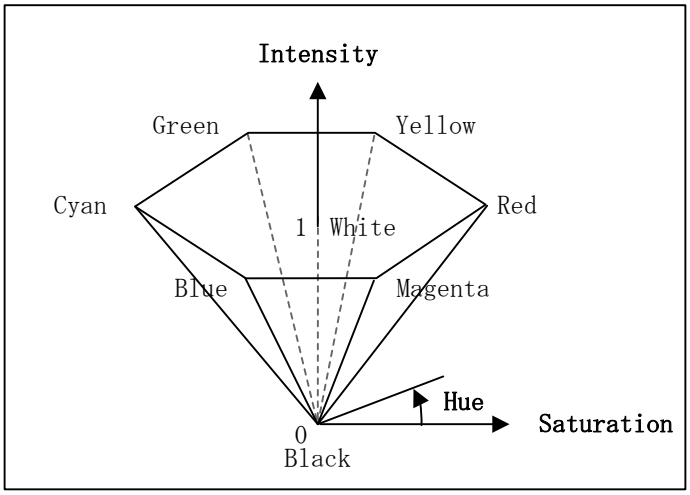
❑ 機能

ＲＧＢ画像のＬＵＴ変換を行います。ＬＵＴハンドルで管理されるＲＧＢＬＵＴ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	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(不定)
	I≠ 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。

ＲＧＢ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 );

.....................................

/* RGBLUTオブジェクト作成 */
mode = RGBLUT888;
hLUT = CreateRGBLUT( mode , 0 );

/*********************************************
  ＲＧＢ画像から色相(Hue)画像へ変換
*********************************************/
/* RGBLUTのLUTテーブルアクセス可能 */
mode = 0;
OpenRGBLUT( hLUT, &size, &lut, mode );

/* LUTテーブル設定[色相(Hue)] */
for( r = 0; r < 256; r++ ){                    ┐ r,g,b値(0～255)で
   for( g = 0; g < 256; g++ ){                 │ H(Hue)値が決定されます
      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;
         }
         SetRGBLUT888( lut, r, g, b, H );      ← r,g,b値のときのH(Hue)値
      }                                          をlutへ設定します
   }
}

/* RGBLUTのLUTテーブルアクセス禁止 */
CloseRGBLUT( hLUT, lut );

/* RGBLUT変換 */
IP_ConvertRGBLUT( ImgRGB, ImgID , 0 , hLUT);

.....................................

/* RGBLUTオブジェクト削除 */
DeleteRGBLUT( hLUT );

/* 画面解放 */
FreeImg( ImgID );
FreeImg( ImgRGB );
```

ＲＧＢ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 );

.....................................

/* RGBLUTオブジェクト作成 */
mode = RGBLUT888;
hLUT = CreateRGBLUT( mode , 0 );

/*********************************************
  ＲＧＢ画像から彩度(Saturation)画像へ変換
*********************************************/
/* RGBLUTのLUTテーブルアクセス可能 */
mode = 0;
OpenRGBLUT( 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;
         }
         SetRGBLUT888( lut, r, g, b, S );
      }
   }
}

/* RGBLUTのLUTテーブルアクセス禁止 */
CloseRGBLUT( hLUT, lut );

/* RGBLUT変換 */
IP_ConvertRGBLUT( ImgRGB, ImgID , 0 , hLUT);

.....................................

/* RGBLUTオブジェクト削除 */
DeleteRGBLUT( hLUT );

/* 画面解放 */
FreeImg( ImgID );
FreeImg( ImgRGB );
```

ＲＧＢ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 );

.....................................

/* RGBLUTオブジェクト作成 */
mode = RGBLUT888;
hLUT = CreateRGBLUT( mode , 0 );

/*********************************************
  ＲＧＢ画像から明度(Intensity)画像へ変換
*********************************************/
/* RGBLUTのLUTテーブルアクセス可能 */
mode = 0;
OpenRGBLUT( 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 ) );
         SetRGBLUT888( lut, r, g, b, I);
      }
   }
}

/* RGBLUTのLUTアクセス禁止 */
CloseRGBLUT( hLUT, lut );

/* RGBLUT変換 */
IP_ConvertRGBLUT( ImgRGB, ImgID , 0 , hLUT);

.....................................

/* RGBLUTオブジェクト削除 */
DeleteRGBLUT( hLUT );

/* 画面解放 */
FreeImg( ImgID );
FreeImg( ImgRGB );
```

2.31.6 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
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
２３	ＬＵＴの先頭アドレス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

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	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)]

ＲＧＢＬＵＴ変換例は、IP_ConvertRGBLUT()の機能の欄を参照してください。

2.32 RAW/RGB変換

2.32.1 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

２０	ＲＡＷ／ＲＧＢ変換データＸ方向長設定値範囲外
２１	ＲＡＷ／ＲＧＢ変換データＹ方向長設定値範囲外
４６	画面ＩＤ空きなし
４８	動的メモリ確保エラー
１３８	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。

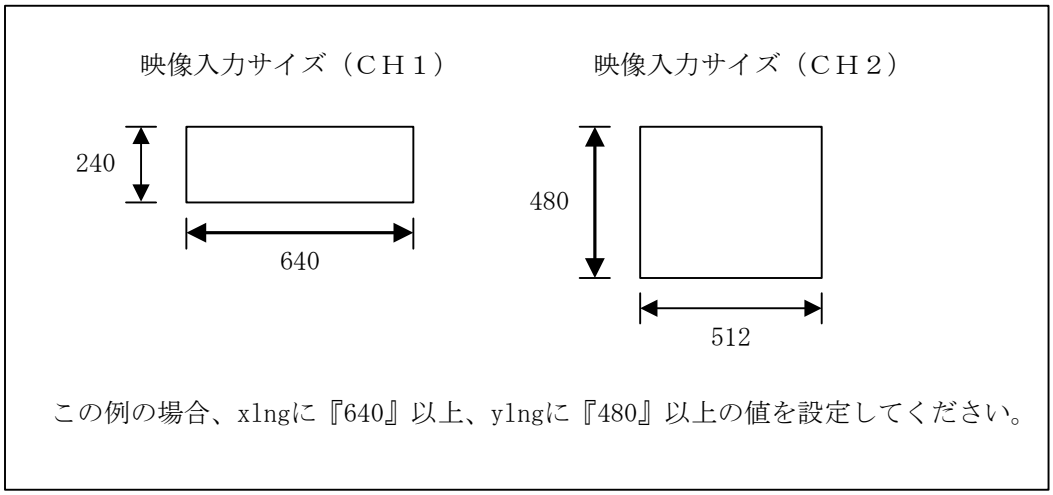
ＲＡＷ／ＲＧＢ変換コマンド

```
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：１３８）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。



この例の場合、xlngに『640』以上、ylngに『480』以上の値を設定してください。

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。

2.32.2　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（ClearIPError）未発行

※ 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。

2.32.3　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） （但し ＮＶＰ－Ａｘ４３５ＣＬは０～３）
float	Rgain	Ｒゲイン（０～２５５）
float	Ggain	Ｇゲイン（０～２５５）
float	Bgain	Ｂゲイン（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	カメラチャネル番号設定値範囲外
２１	Ｒゲイン設定値範囲外
２２	Ｇゲイン設定値範囲外
２３	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：１．２３ Ｇ：１．００ Ｂ：１．９５

2.32.4　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） （但し ＮＶＰ－Ａｘ４３５ＣＬは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
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

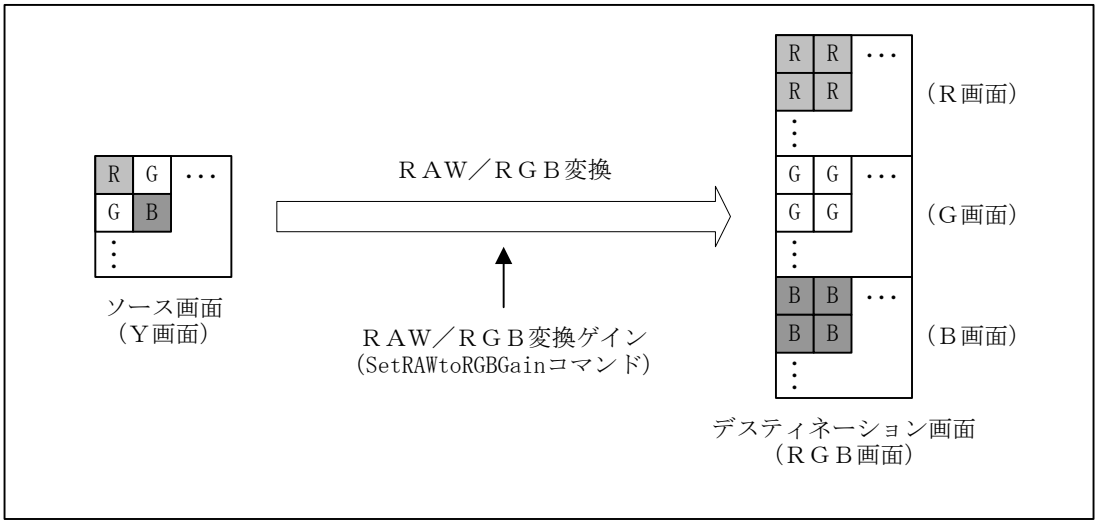
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	ソース画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
１３	デスティネーション画面番号に対応する画面が未確保、または番号が有効範囲外 （不当画面番号エラー）
２２	変換モード設定値範囲外
２３	カメラチャネル番号設定値範囲外
３０	ソース画面ウィンドウ設定エラー
３３	デスティネーション画面ウィンドウ設定エラー
４９	ソース画面が不定画面
１３８	ＲＡＷ／ＲＧＢ変換データ未生成（CreateConvertRAWData未発行）、またはＲＡＷ／ＲＧＢ変換データのワーク画面が映像入力サイズより小さい、または映像入力サイズ未設定（SetVideoFrame未発行）
－	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

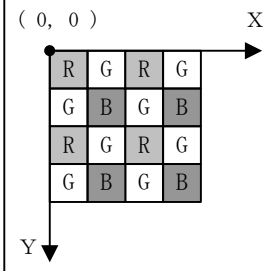
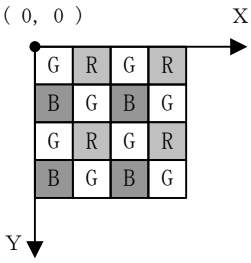
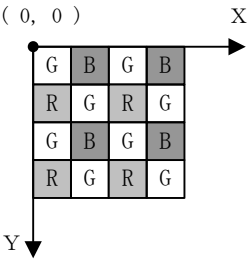
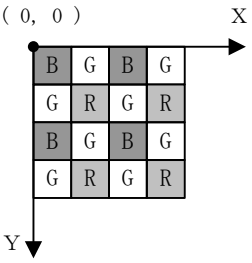
❑ 機能

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：１３８）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：１３８）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				

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 */
```

2.32.5　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） （但し ＮＶＰ－Ａｘ４３５ＣＬは０～３）
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）

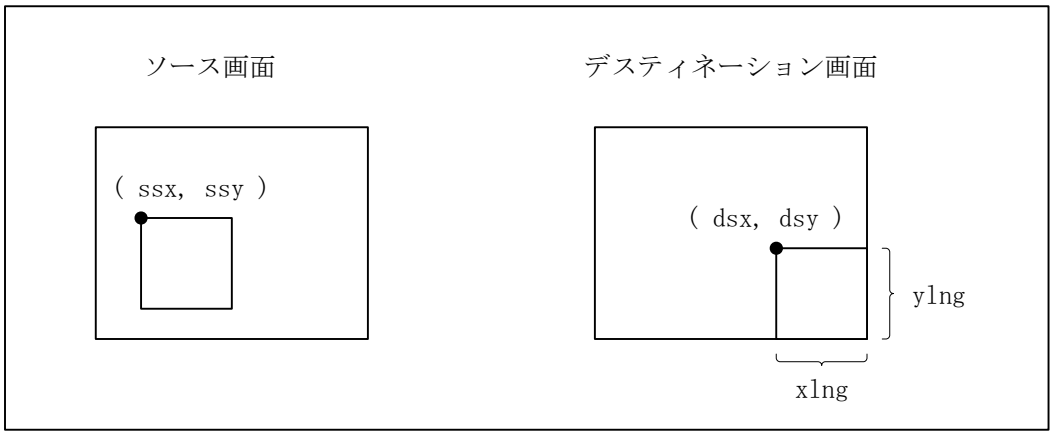
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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
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
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（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

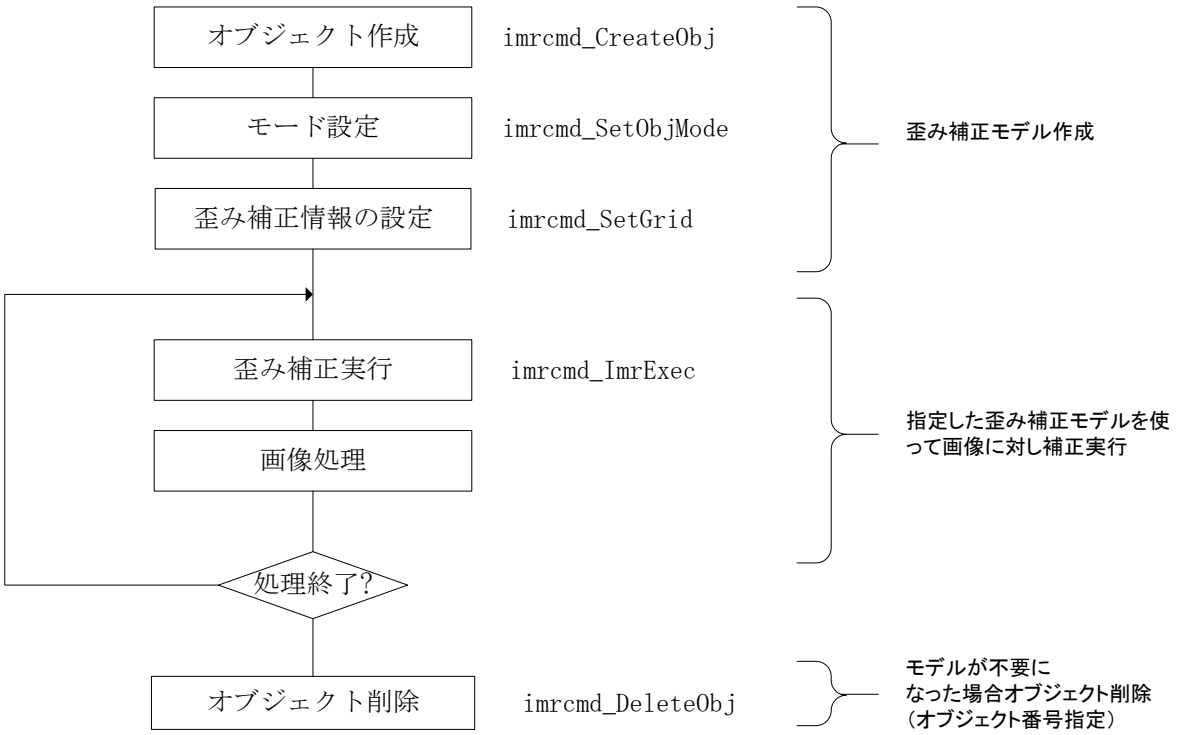
❑ 機能

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。

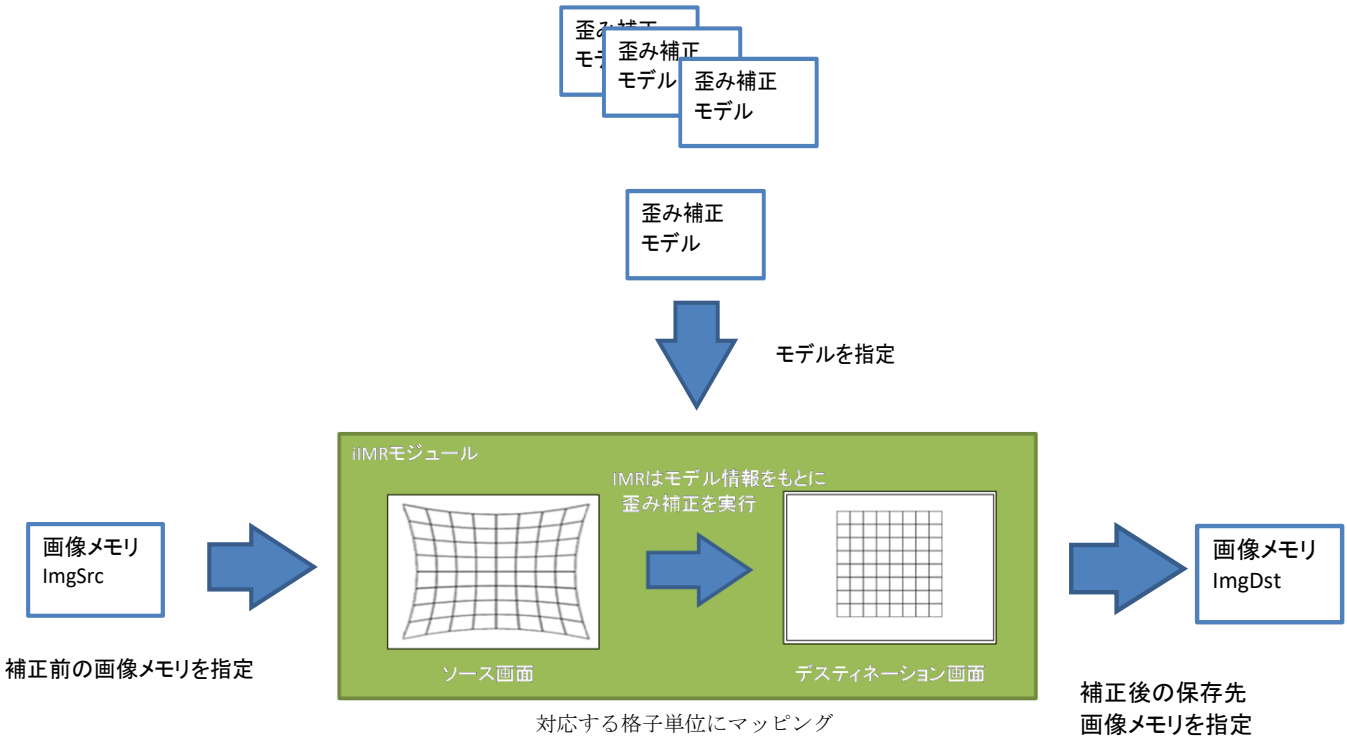


2.33 歪み補正

(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。

2.33.1 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

２	ＯＳシステムエラー
２０	オブジェクトタイプ指定エラー
２１	オブジェクトサイズ指定エラー
４８	メモリ確保エラー
１３８	手続きエラー
１６０	内部エラー
１６１	IMR処理タイムアウト
１６２	IMR資源確保タイムアウト
１６７	オブジェクトの複数タスク同時呼び出し
－	エラーリセットコマンド (ClearIPError) 未発行

※ 「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

2.33.2　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

２	ＯＳシステムエラー
２０	歪み補正オブジェクトハンドル指定エラー
１３８	手続きエラー
１６０	内部エラー
１６１	IMR処理タイムアウト
１６２	IMR資源確保タイムアウト
１６７	オブジェクトの複数タスク同時呼び出し
－	エラーリセットコマンド（ClearIPError）未発行

※ 「付録A 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。

2.33.3 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

２	ＯＳシステムエラー
２０	歪み補正オブジェクトハンドル指定エラー
２１	オブジェクト動作モード指定エラー
１３８	手続きエラー
１６０	内部エラー
１６１	IMR処理タイムアウト
１６２	IMR資源確保タイムアウト
１６６	オブジェクト不正
１６７	オブジェクトの複数タスク同時呼び出し
－	エラーリセットコマンド（ClearIPError）未発行

※ 「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。

2.33.4 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≦x≦32767、2≦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）

❑ エラーコード

－　　エラーリセットコマンド (ClearIPError) 未発行

※ 「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 {
  long x;                   /* X方向頂点数 */
  long y;                   /* Y方向頂点数 */
} IMR_VERTEX;
```

❑ エラーコード

２	ＯＳシステムエラー
２０	歪み補正オブジェクトハンドル指定エラー
２１	頂点データ数指定エラー
２２	補正前格子座標リスト指定エラー
２３	補正後格子座標リスト指定エラー
２４	補正前格子座標リスの固定小数点表記指定エラー
１３８	手続きエラー
１６０	内部エラー
１６１	IMR処理タイムアウト
１６２	IMR資源確保タイムアウト
１６５	オブジェクトサイズ不足
１６６	オブジェクト不正
１６７	オブジェクトの複数タスク同時呼び出し

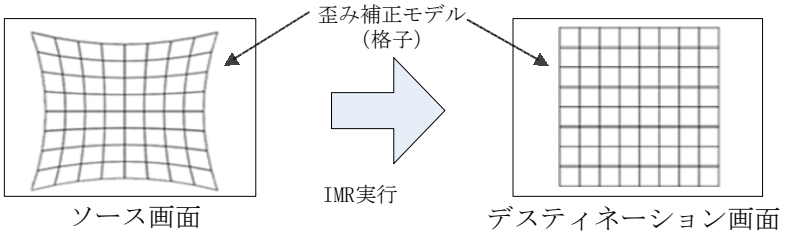
※ 「付録A エラーコード一覧」を参照して下さい。

❑ 機能

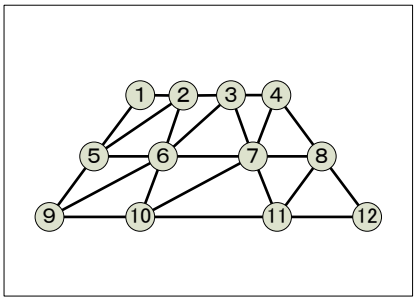
指定したオブジェクトに歪み補正モデルを生成します。歪み補正はソース・デスティネーション画面それぞれに対応するVtxSizeに示す頂点数(X×Y)からなる格子を指定することで実行されます。この格子を本コマンドで設定します。ソース・デスティネーションそれぞれの格子の頂点座標を一次元配列に配置して、SrcVtxListに、DstVtxListに渡します。

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。

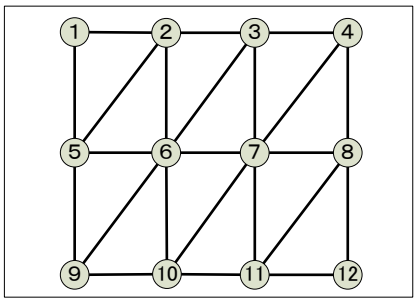
頂点数が多いほどオブジェクトサイズが大きくなります。『imrcmd_CreateObj』でオブジェクトを生成する際に必要な頂点数を指定して生成してください。指定した値を超える頂点数を本コマンドで設定した場合オブジェクトサイズが足りずエラーとなります。



（1）台形補正の設定例を示します。
頂点数は、X(4)×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

SrcVtxList

[0]	①Src座標
[1]	②Src座標
⋮	⋮
[10]	⑪Src座標
[11]	⑫Src座標

4×3

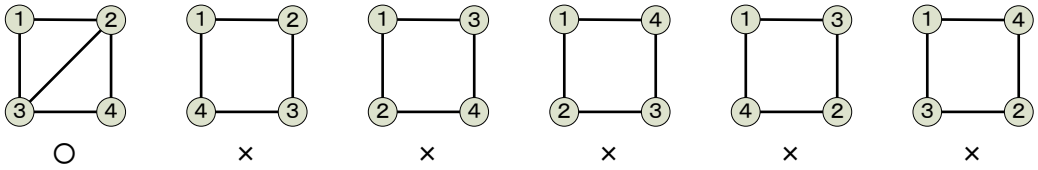
DstVtxList

[0]	①Dst座標
[1]	②Dst座標
⋮	⋮
[10]	⑪Dst座標
[11]	⑫Dst座標

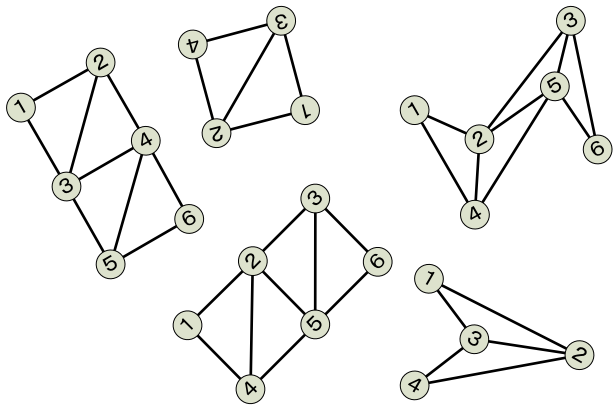
4×3

台形補正の設定例

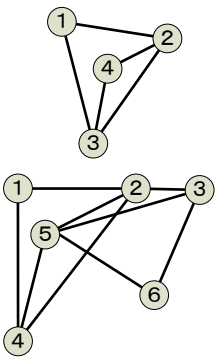
（2）座標リストへの登録順序について
格子の頂点を座標リストへ登録する順序に注意が必要です。逸脱した場合描画されません。また、格子領域が重なる場合も描画されません。以下に例を示します。



頂点の座標リストへの登録順序



○（すべて可）



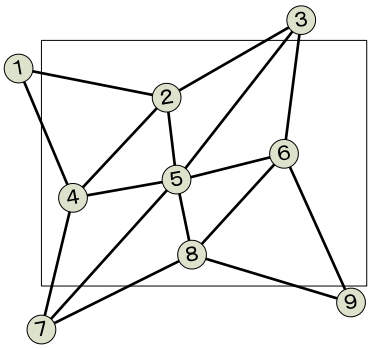
×（格子領域が重なる場合不可）

格子の例

（３）ソース座標について

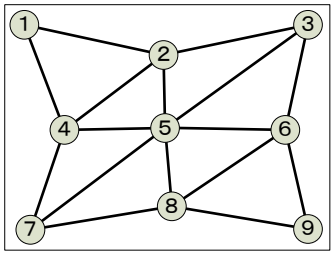
座標値は、ソースウィンドウ始点からの相対座標で設定してください。必ず使用するソース画面かつソースウィンドウ範囲内の座標（0≦X,Y≦2047）で設定してください。それを超える座標を指定した場合、画面外の領域が参照され動作が不定となります。（ソースウィンドウおよびソース画面については『imrcmd_ImrExec』を参照ください。）

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

×



ソース画面ウィンドウ

○

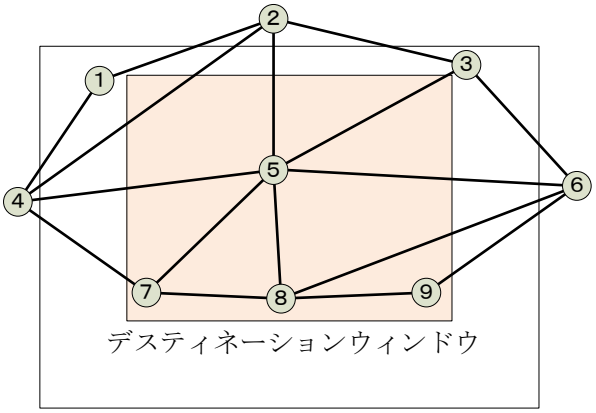
ソース画面と座標指定

（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≦X,Y≦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画面

○

デスティネーション画面と座標指定

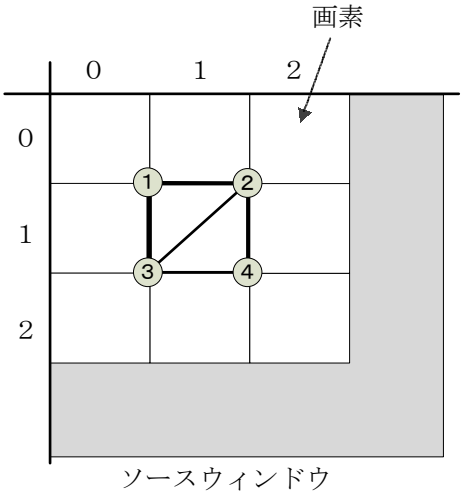
（５）固定小数点表記について
ソース座標は、整数型または、固定小数点１～５桁での設定が可能です。
以下に float型(浮動小数点)から固定小数点データへの変換方法を示します。

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 = (unsigned long)((2^n) * x)

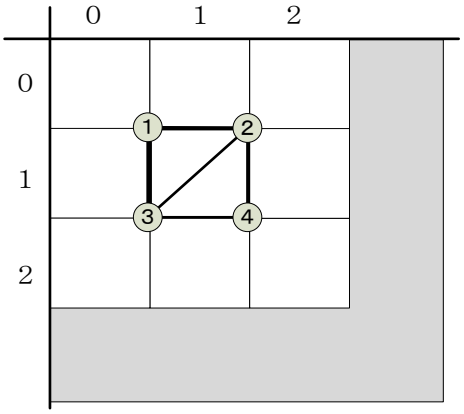
変換の際に、整数型へのキャストにより小数点以下が切り捨てられますので誤差となります。ご理解の上ご使用ください。

（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→ディ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。



ソースウィンドウ

頂点座標
①(1,1)
②(2,1)
③(1,2)
④(2,2)



デスティネーションウィンドウ

頂点座標
①(1,1)
②(2,1)
③(1,2)
④(2,2)

2.33.5 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_CH0 （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	ディ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_0CLEAR （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１２８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１６画素アラインとして下さい。
それ以外の場合は３２画素アラインとして下さい。

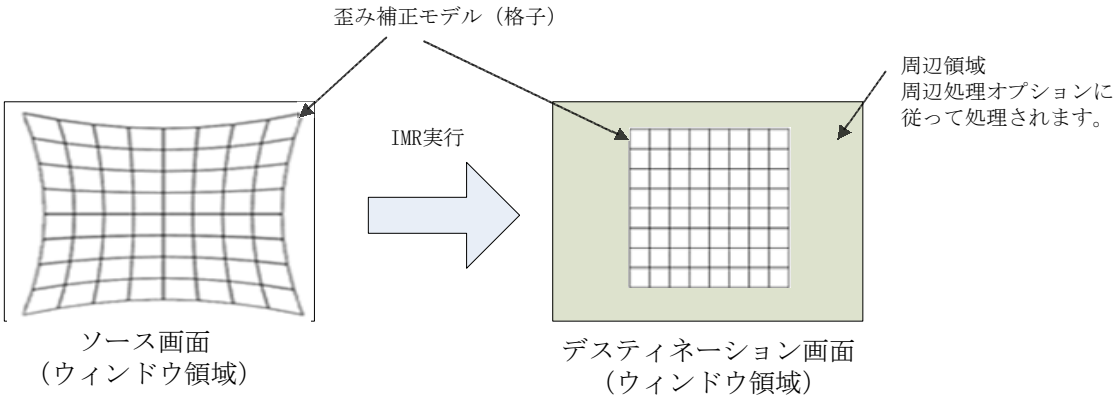
❑ エラーコード

２	ＯＳシステムエラー
２０	チャンネル指定エラー
２１	歪み補正オブジェクトハンドル指定エラー
２２	ソース画面幅エラー
２３	ディスティネーション画面幅エラー
２４	IMRウィンドウ設定エラー
２５	歪み補正実行オプション指定エラー
１３８	手続きエラー
１６０	内部エラー
１６１	IMR処理タイムアウト
１６２	IMR資源確保タイムアウト
１６６	オブジェクト不正
１６７	オブジェクトの複数タスク同時呼び出し
－	エラーリセットコマンド（ClearIPError）未発行

※ 「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CPUのキャッシュ上に存在しない場合は不要です。詳細は「パイプライン制御」、および『RefreshImg』コマンドを参照ください。

3. VP互換コマンド

3.1 正規化相関(高速ラスタスキャン)

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

４８	動的メモリ確保エラー
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

正規化相関サーチのテンプレート特徴量データ領域の確保を行います。デフォルトでは、256個の領域が確保されていますが、それ以上のテンプレートを使用する場合は、本コマンドにより、必要なテンプレート特徴量データ領域を確保してください。

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。本コマンド実行後にVP互換コマンドの正規化相関コマンドを使用する場合は、必ず「vpxAllocCorrTemplate」コマンドで再度、テンプレート特徴量データ領域を確保してください。

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

２１	テンプレートデータテーブルを返すアドレスが不正
８１	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           xleng;
        short           yleng;
        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           xleng;
        short           yleng;
} 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
xleng	テンプレートのアフェイン変換前のＸ方向の長さ
yleng	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
xleng	テンプレートのＸ方向長さ
yleng	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

２１	テンプレートデータテーブルを返すアドレスが不正
８１	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

正規化相関サーチのテンプレートパターンのマスクを有効にします。

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、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と同一となります。
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

正規化相関サーチのテンプレートパターンの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	Y16	Y32	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

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	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
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	ソース０画面ウィンドウの始点座標が画面外
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（サイズが２５６×２５６より大きい）
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（分散値エラー）
８６	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	Y16	Y32	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

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	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
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	ソース０画面ウィンドウの始点座標が画面外
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（サイズが２５６×２５６より大きい）
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（分散値エラー）
８６	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（SRCO_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 , CORRTBL *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
CORRTBL	*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	Y16	Y32	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	*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

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	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
２２	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
３０	ソース０画面ウィンドウの始点座標が画面外
４５	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）
４８	動的メモリ確保エラー
４９	サーチ画面が不定画面
８０	テンプレートサイズがサーチ領域を超える
８１	テンプレート番号が有効範囲外
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

矩型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１００００倍した値です。

```
score = r * r * 10000
```

SRCO_WINで指定された矩型領域をStepX、StepYで指定された間隔でラスターサーチします。
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	Y16	Y32	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

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	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
２２	入力処理ポイントテーブルを返すアドレスが不正
２４	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
３０	ソース０画面ウィンドウの始点座標が画面外
４５	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）
４９	サーチ画面が不定画面
８０	テンプレートサイズがサーチ領域を超える
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（有効なサーチ点がない）
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

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１００００倍した値です。

```
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	Y16	Y32	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

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	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
２２	入力処理ポイントテーブルを返すアドレスが不正
２３	サブピクセル計算結果テーブルを返すアドレスが不正
３０	ソース０画面ウィンドウの始点座標が画面外
４９	サーチ画面が不定画面
８０	テンプレートサイズがサーチ領域を超える
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（有効なサーチ点がない）
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。

```
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　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。

3.1.13　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。

3.1.14　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

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

２０	Ｘ，Ｙ座標配列テーブルを格納したアドレスが不正
２１	検索角度範囲テーブルを格納したアドレスが不正
２２	ハフ変換直線算出テーブルを格納したアドレスが不正
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、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;
```

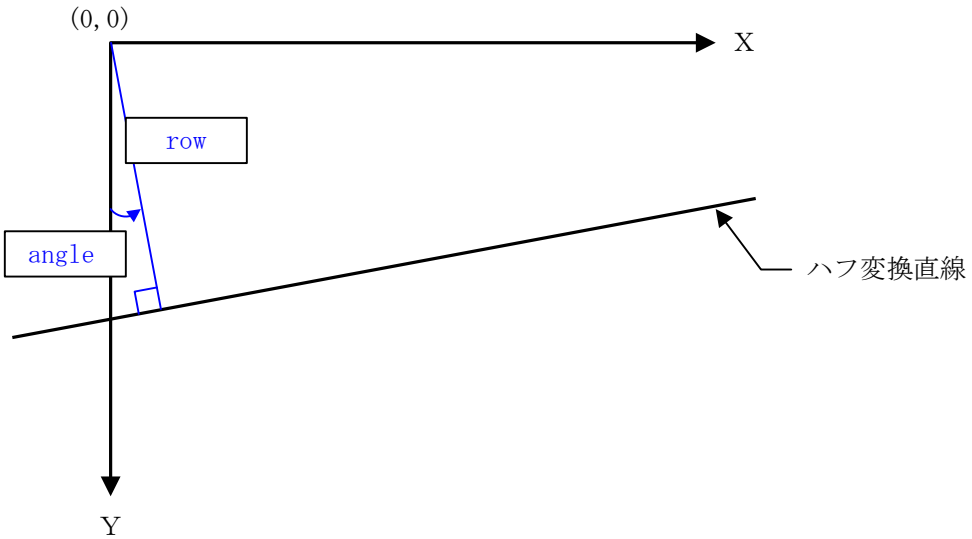
POINTTBL	説明
num	有効データ数
opt	未使用
point[_POINTTBL_DATA_NUM]	X, Y座標配列

```
typedef struct {
      short       lower;
      short       upper;
} LINERANGE;
```

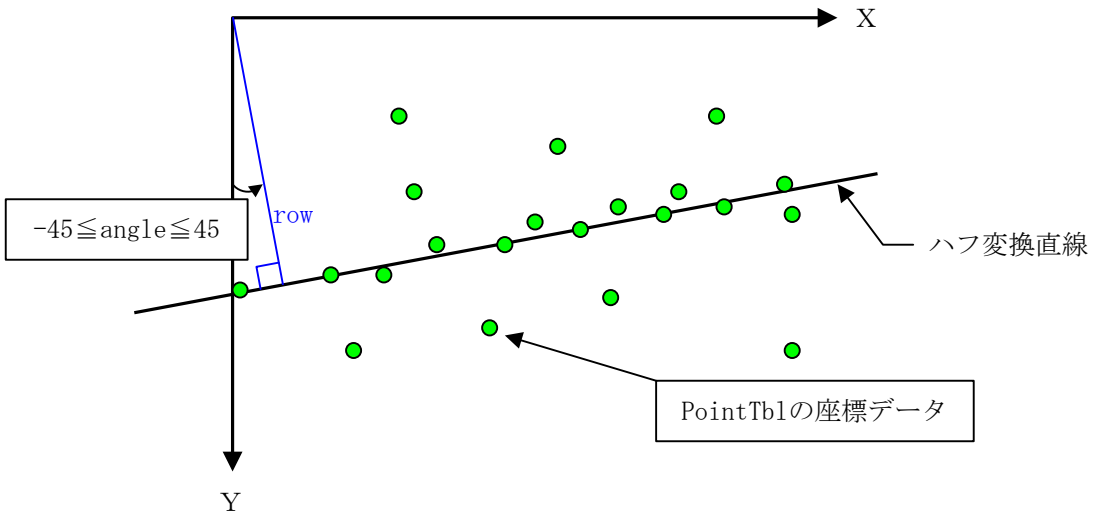
LINERANGE	説明
lower	検索範囲　下限
upper	検索範囲　上限

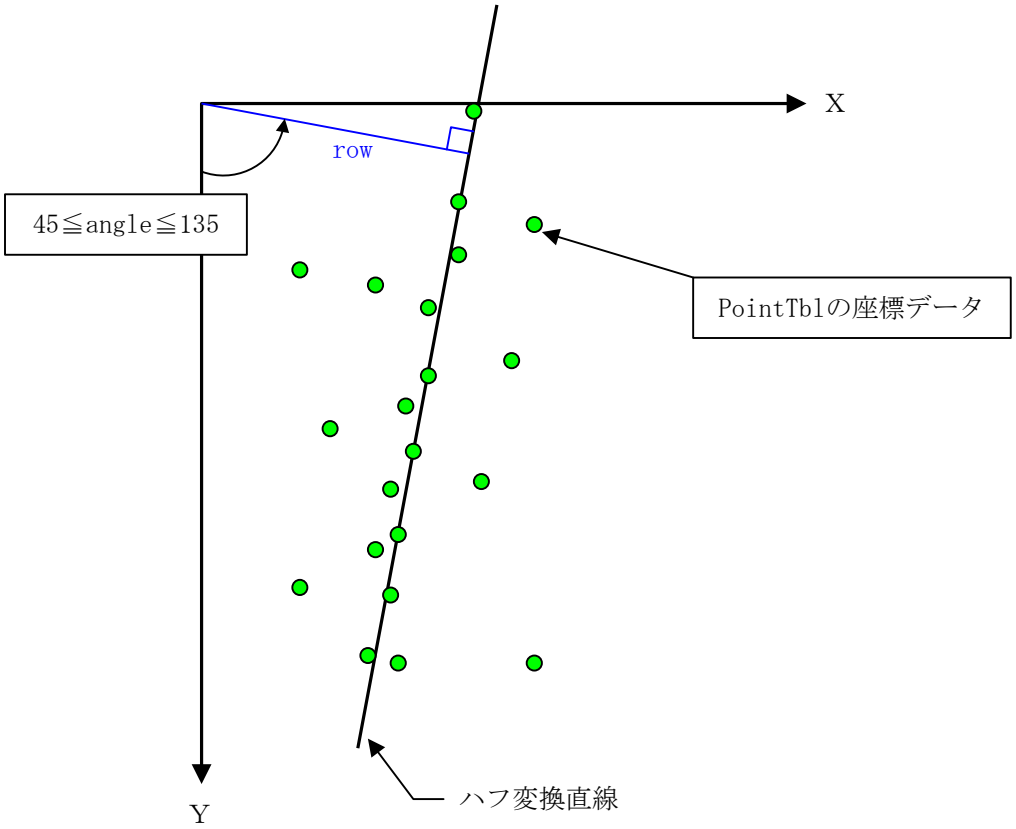
```
typedef struct {
      float       row;
      float       angle;
} HOUGHLINE;
```

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_ProjectBORegionX())、領域Ｙ座標抽出(IP_ProjectBORegionY())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０～４０９５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 */
POINTTBL RegionYMin,RegionYMax,RegionXMin,RegionXMax;
POINTTBL *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_ProjectBORegionY( 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_ProjectBORegionX( 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

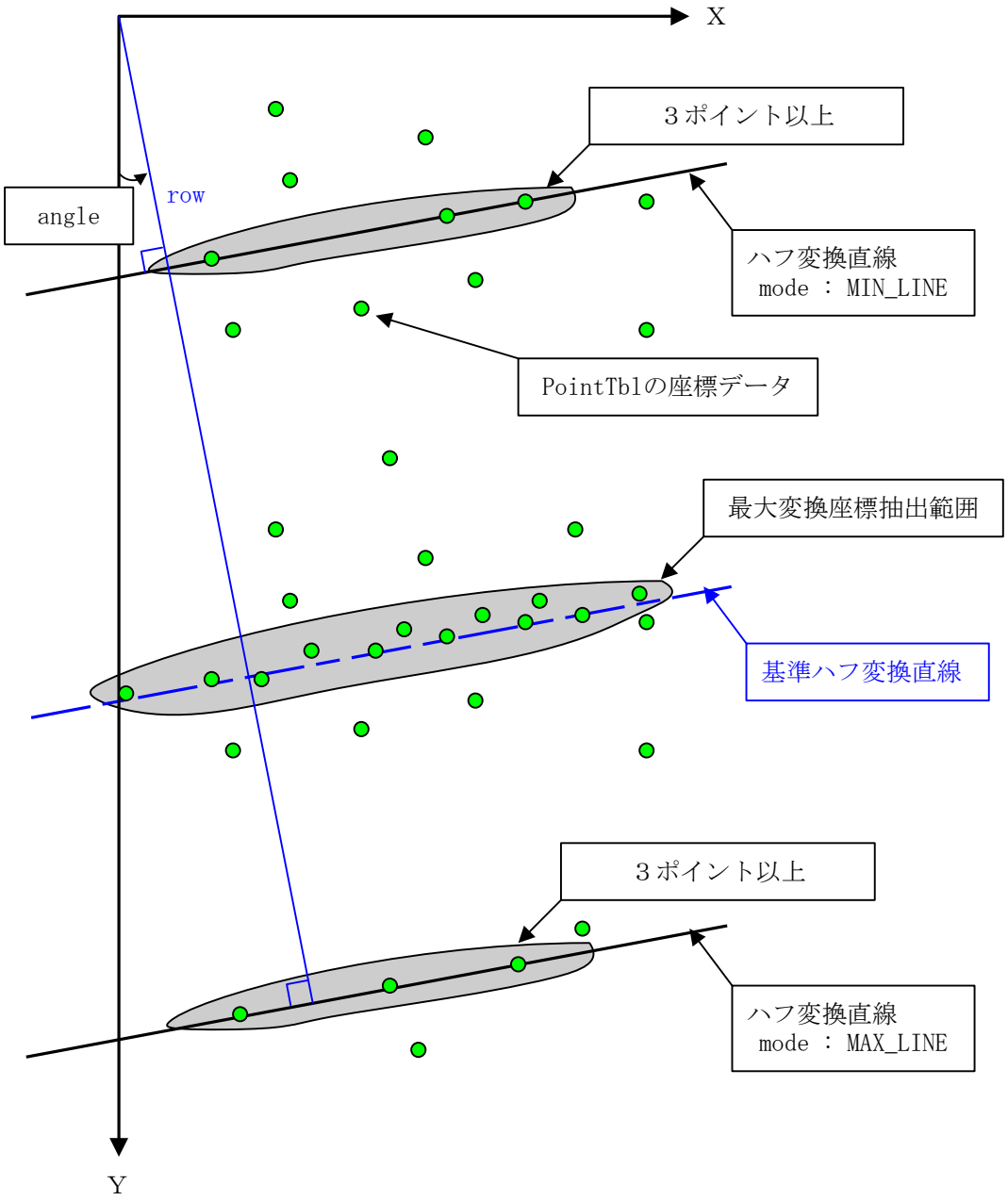
２０	Ｘ，Ｙ座標配列テーブルを格納したアドレスが不正
２１	検索角度範囲テーブルを格納したアドレスが不正
２２	ハフ変換直線算出テーブルを格納したアドレスが不正
４８	メモリ確保エラー
２０１	直線のポイント数不足による直線抽出エラー
－	エラーリセットコマンド（ClearIPError）未発行

※ 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	X，Y座標配列テーブルを格納したアドレスが不正
２１	検索角度範囲テーブルを格納したアドレスが不正
２２	ハフ変換直線算出テーブルを格納したアドレスが不正
２４	ライン検出モード設定エラー
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。

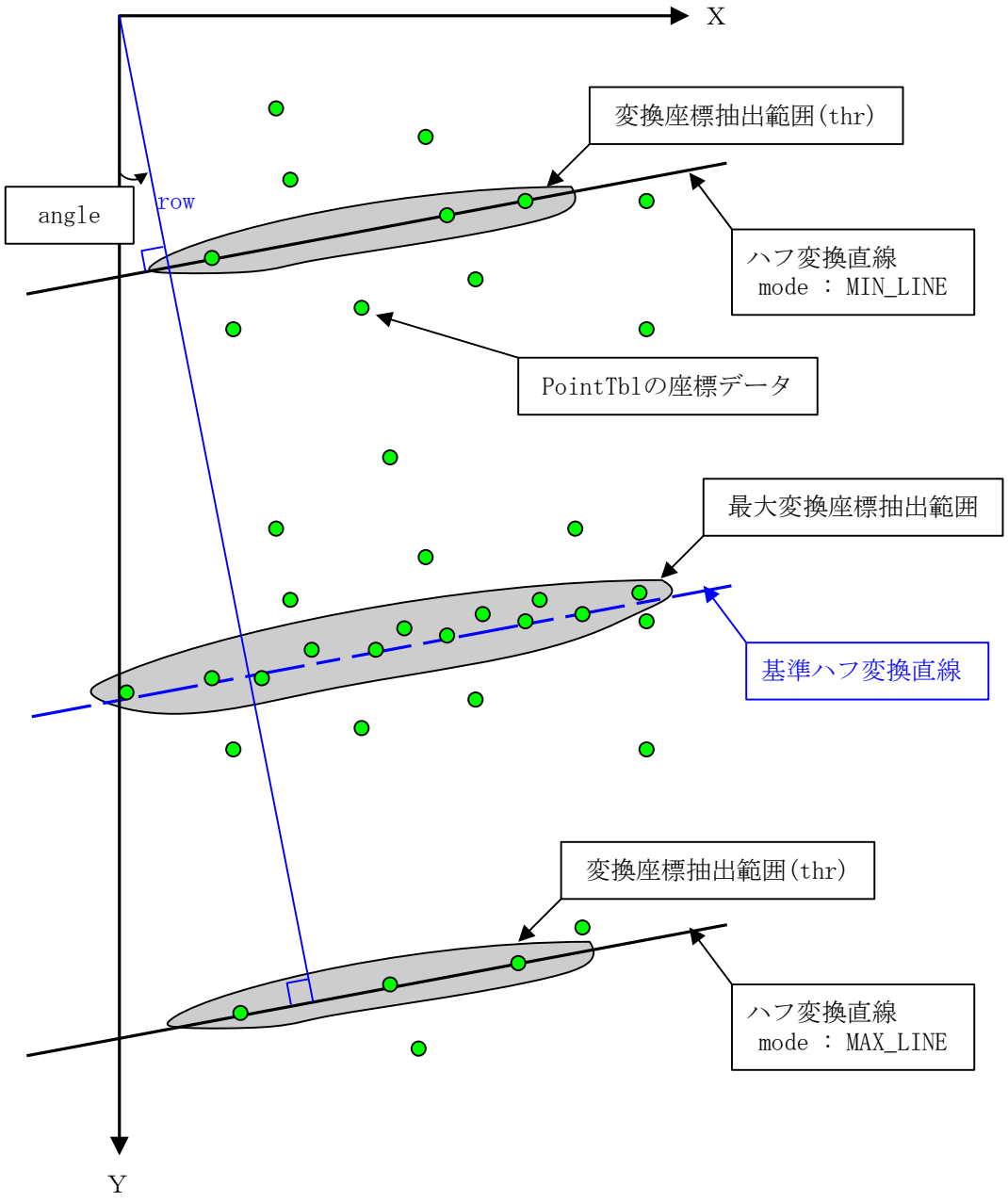
❑ 機能

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	X，Y座標配列テーブルを格納したアドレスが不正
２２	ハフ変換直線算出テーブルを格納したアドレスが不正
２４	ライン検出モード設定エラー
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

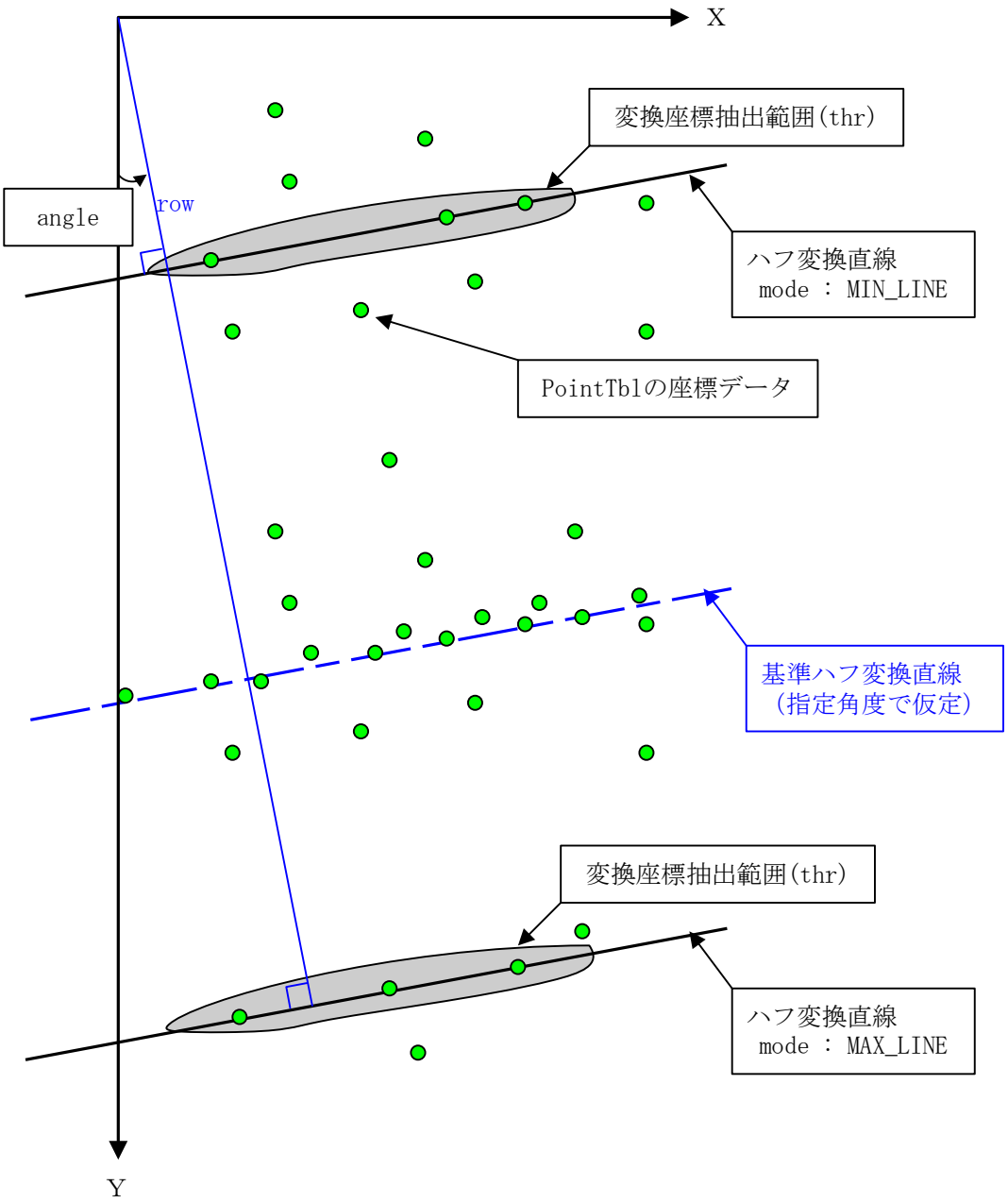
※ 詳細は、付録A「エラーコード一覧」を参照して下さい。

❑ 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° ≦angle≦ 45° までのハフ変換直線
HOUGHLINE	*yline	45° ≦angle≦ 135° 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	xlineハフ変換直線テーブルを格納したアドレスが不正
２２	ylineハフ変換直線テーブルを格納したアドレスが不正
２４	直線の交点座標テーブルを格納したアドレスが不正
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。

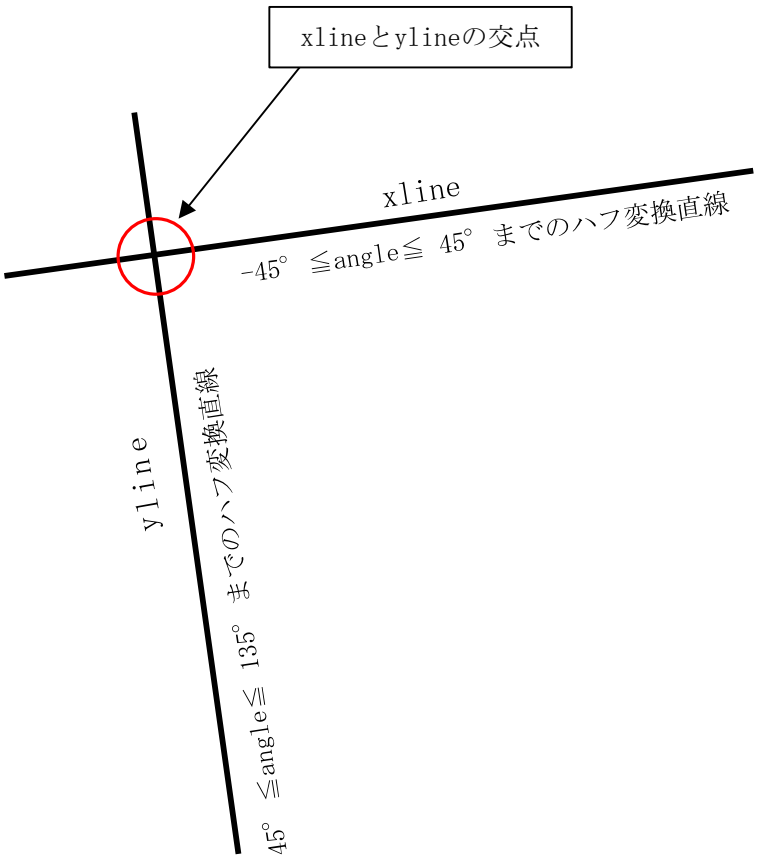
❑ 機能

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° ≦angle≦45° までのブロック０のハフ変換直線

HOUGHLINE *block1-45° ≦angle≦45° までのブロック１のハフ変換直線

HOUGHLINE *block245° ≦angle≦135° までのブロック２のハフ変換直線

HOUGHLINE *block345° ≦angle≦135° 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

２０	ブロック０のハフ変換直線テーブルを格納したアドレスが不正
２１	ブロック１のハフ変換直線テーブルを格納したアドレスが不正
２２	ブロック２のハフ変換直線テーブルを格納したアドレスが不正
２３	ブロック３のハフ変換直線テーブルを格納したアドレスが不正
２４	直線の交点座標テーブルを格納したアドレスが不正
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

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。

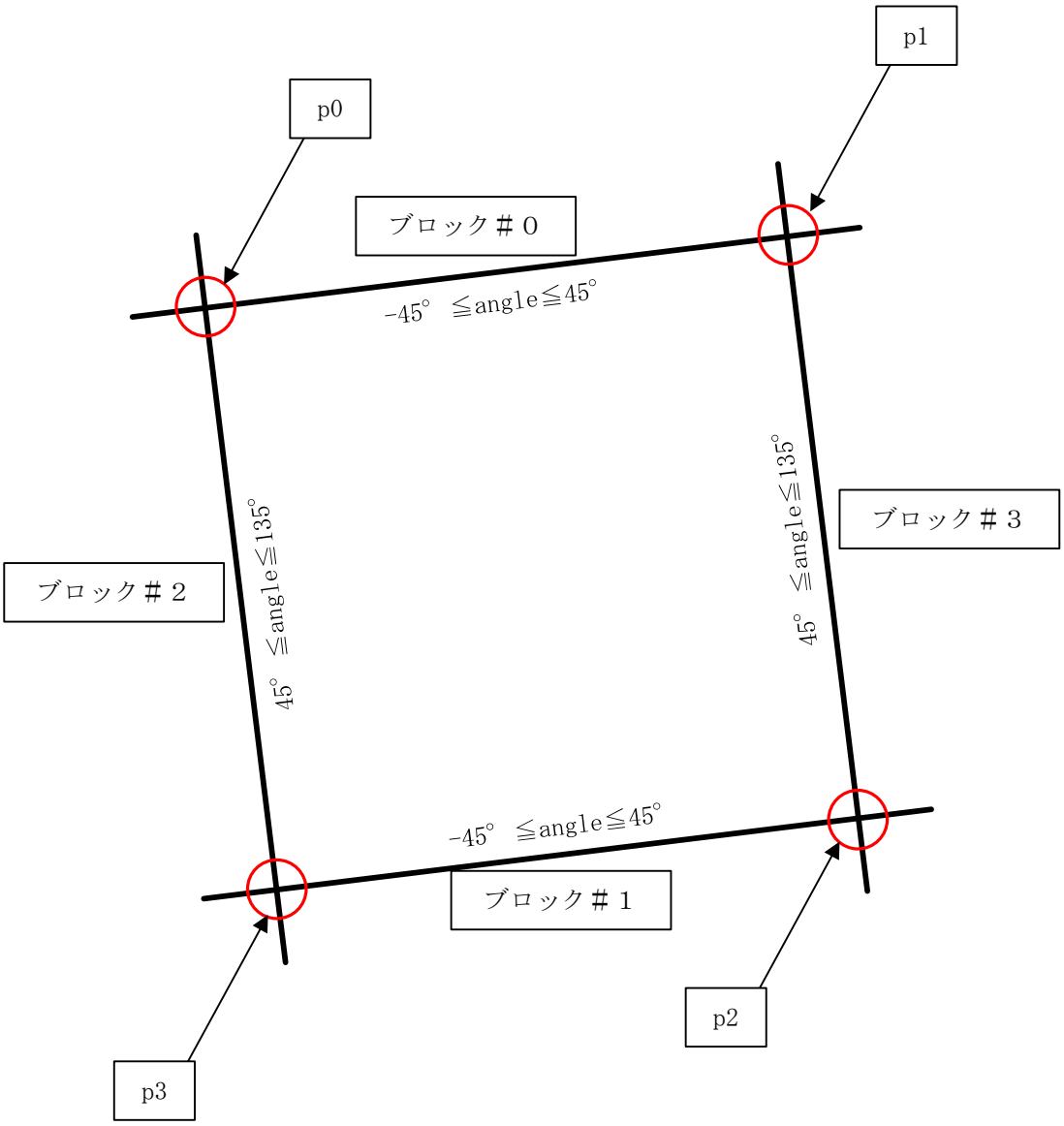
HOUGHLINEのフォーマットは、GetHoughLine()の詳細情報の欄を参照して下さい。
FRECTPOINTのフォーマットを下記に示します。

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

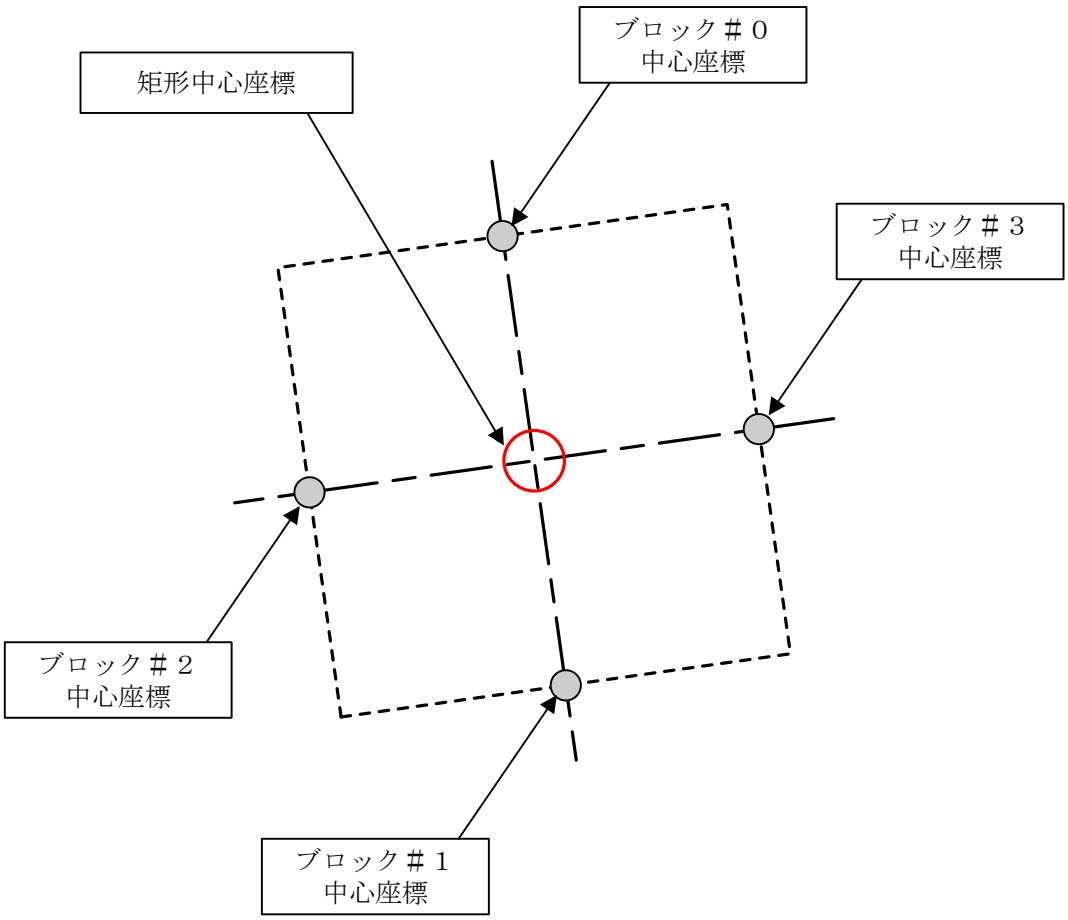
２０	ブロック０の頂点中心座標テーブルを格納したアドレスが不正
２１	ブロック１の頂点中心座標テーブルを格納したアドレスが不正
２２	ブロック２の頂点中心座標テーブルを格納したアドレスが不正
２３	ブロック３の頂点中心座標テーブルを格納したアドレスが不正
２４	直線の交点座標テーブルを格納したアドレスが不正
２０１	０除算エラー
－	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

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

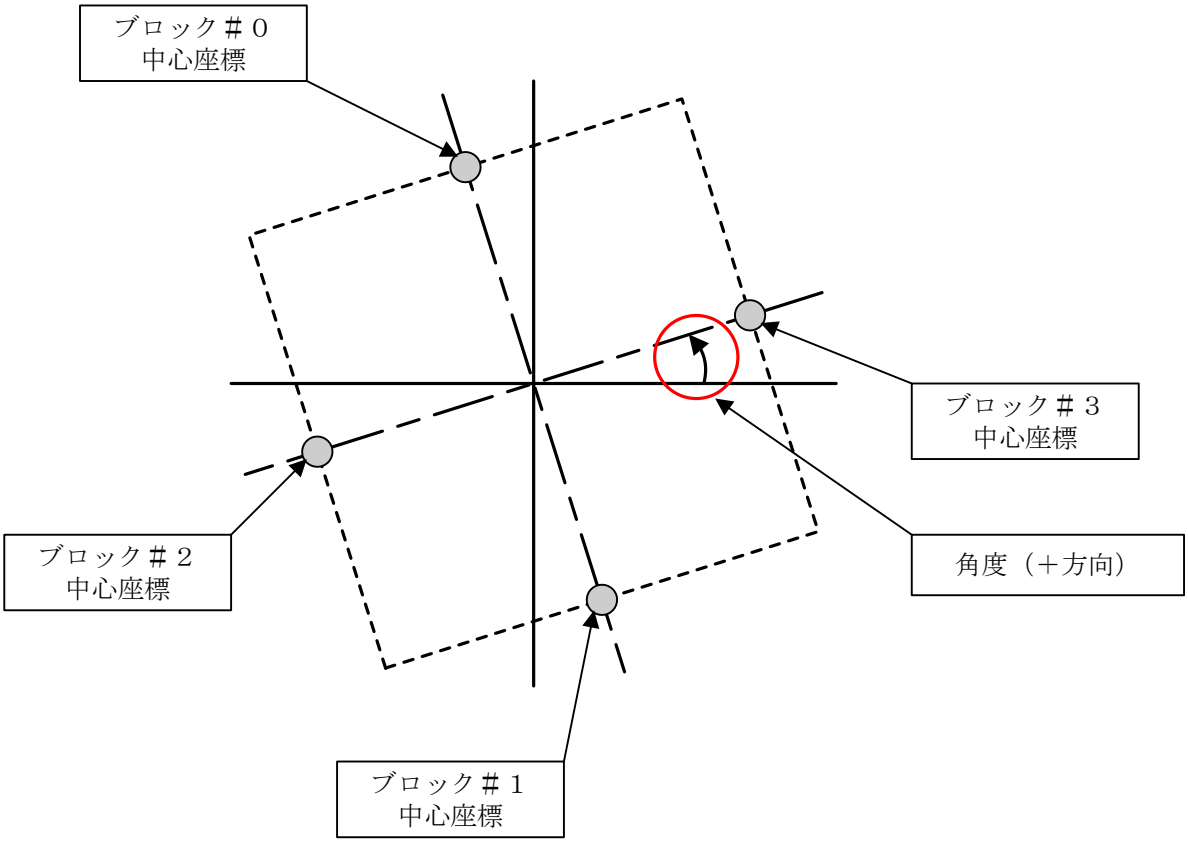
２０	ブロック０の頂点中心座標テーブルを格納したアドレスが不正
２１	ブロック１の頂点中心座標テーブルを格納したアドレスが不正
２２	ブロック２の頂点中心座標テーブルを格納したアドレスが不正
２３	ブロック３の頂点中心座標テーブルを格納したアドレスが不正
２４	矩形の傾き角度テーブルを格納したアドレスが不正
２０１	０除算エラー
－	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

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

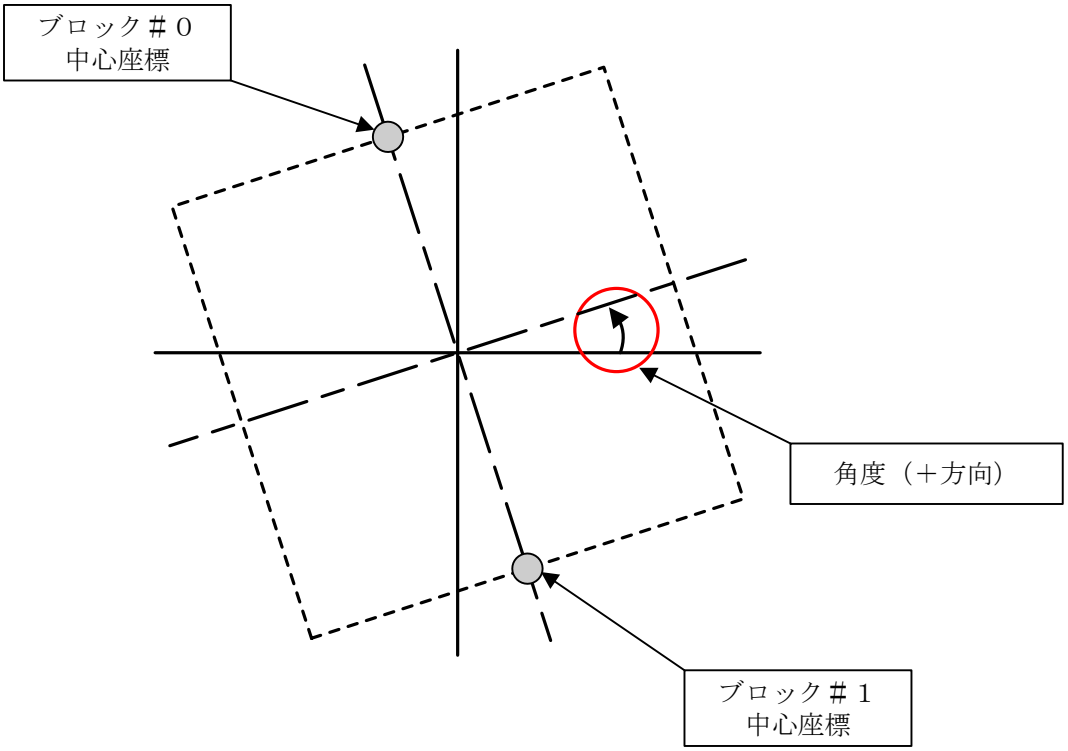
２０	ブロック#０、#２の頂点中心座標テーブルを格納したアドレスが不正
２１	ブロック#１、#３の頂点中心座標テーブルを格納したアドレスが不正
２２	矩形の傾き角度テーブルを格納したアドレスが不正
２０１	０除算エラー
－	エラーリセットコマンド（ClearIPError）未発行

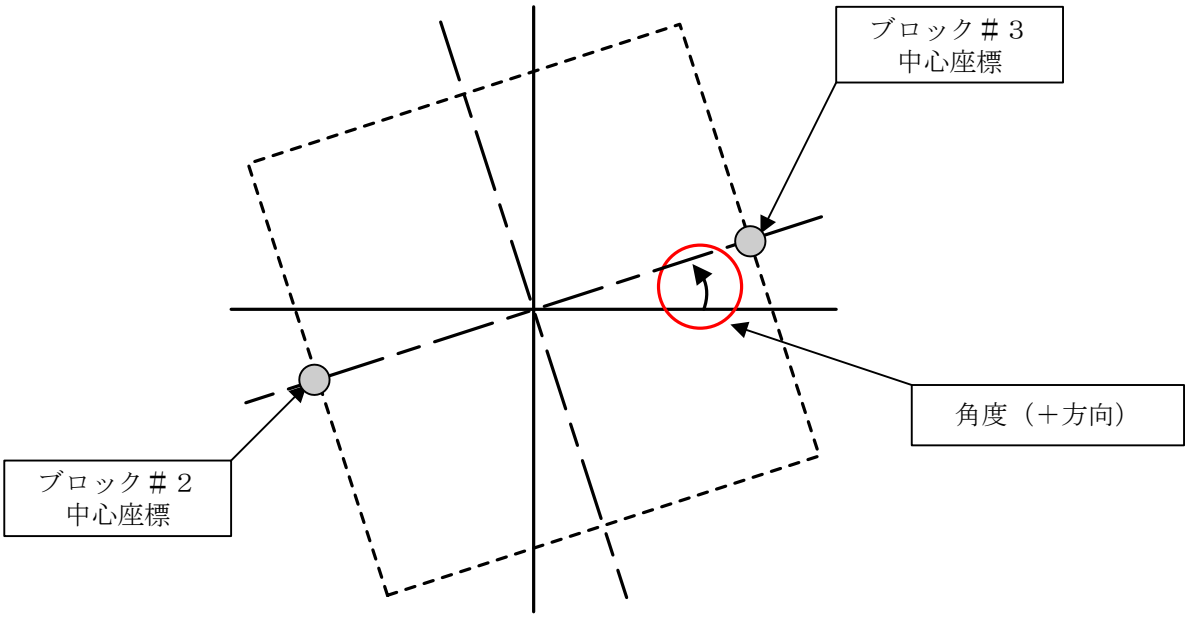
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

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	Y16	Y32	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）
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
１０	画面空き領域、または範囲外（不当画面番号エラー）
２１	読み出しデータテーブルを格納したアドレスが不正
２２	モード設定エラー
２３	ラインウィンドウを格納したアドレスが不正、または投影領域データ不正
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

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

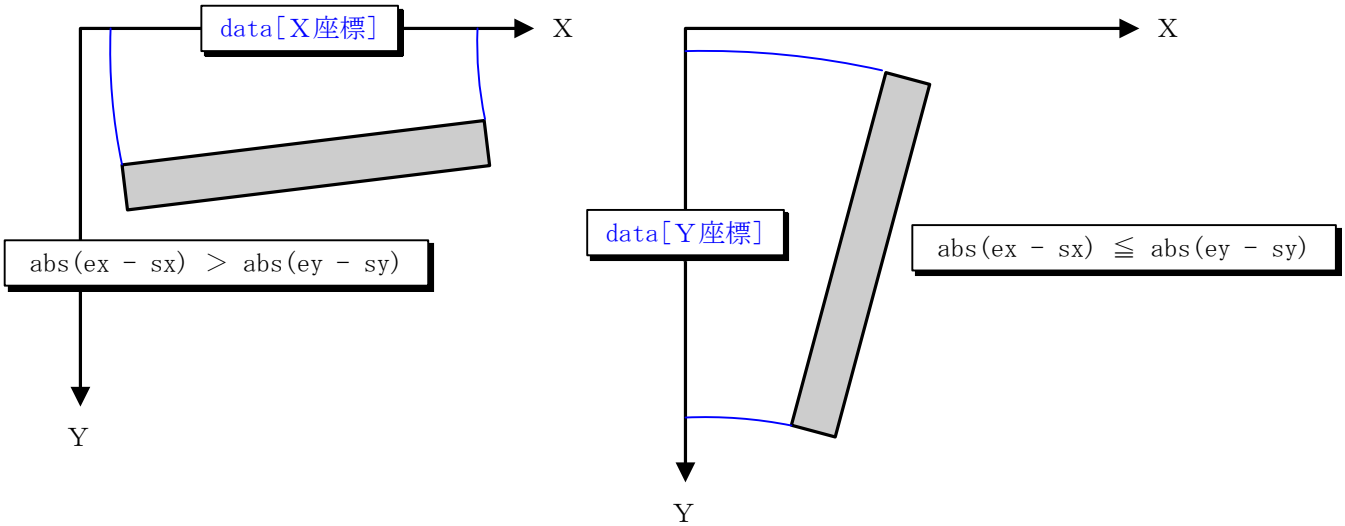
```
typedef struct {
       short            sx;
       short            sy;
       short            ex;
       short            ey;
       short            leng;
       short            opt;
} LINEWINDOW;
```

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

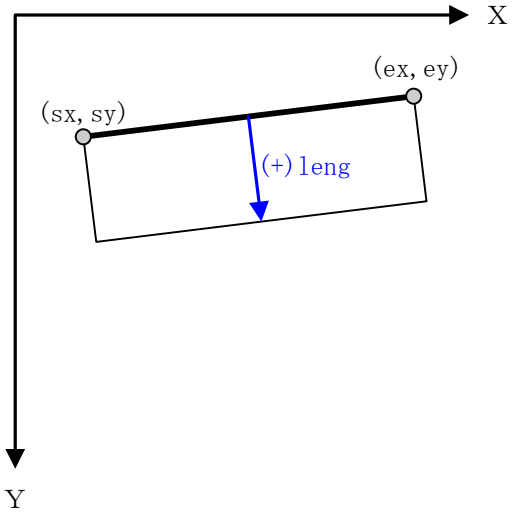
```
abs(ex - sx) ＞ abs(ey - sy)  :  data[ X座標 ]
abs(ex - sx) ≦ abs(ey - sy)  :  data[ Y座標 ]
```

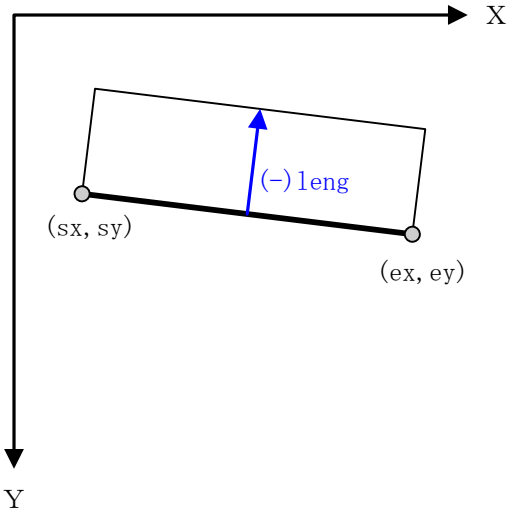
になります。



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。

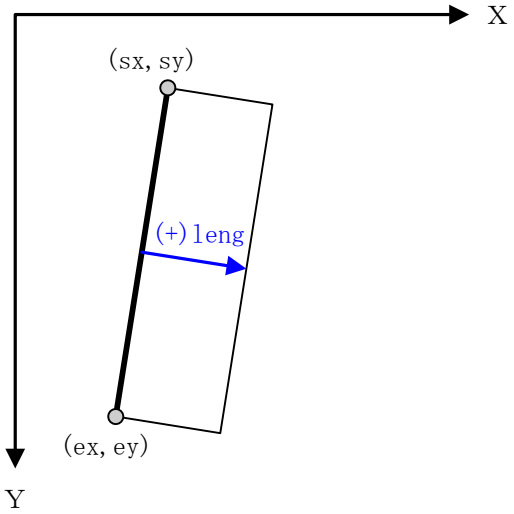
abs(ex - sx) ＞ abs(ey - sy)

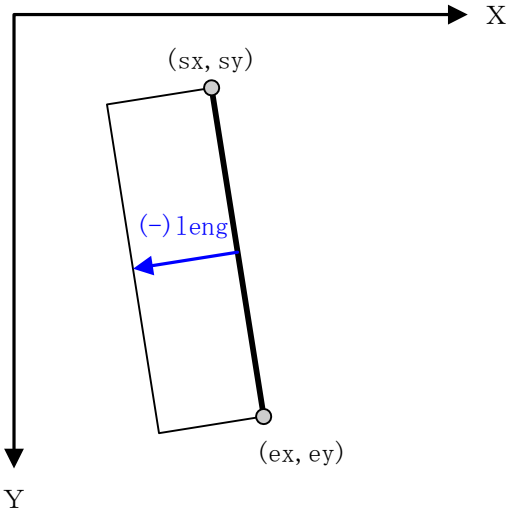




ラインテーブルに出力される値
LineTbl.lower = sx
LineTbl.num = ex - sx + 1
LineTbl.data[] = X軸上に投影されるデータ

abs(ex - sx) ≦ 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	投影データテーブルを格納したアドレスが不正
２１	出力エッジテーブル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

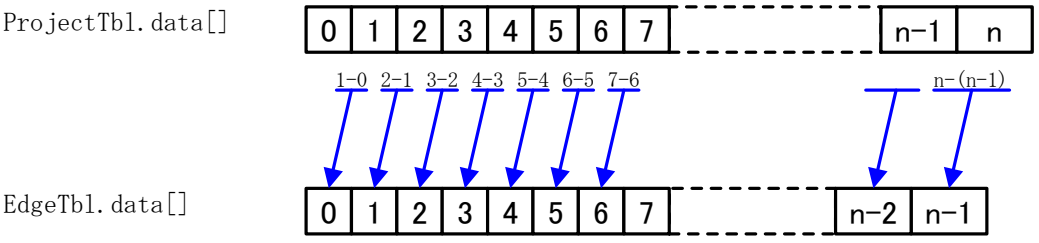
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

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。



エッジテーブルの有効データ数

EdgeTbl.num → ProjectTbl.num - 1

エッジテーブルの先頭アドレス

EdgeTbl.lower → ProjectTbl.lower

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、INT(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	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

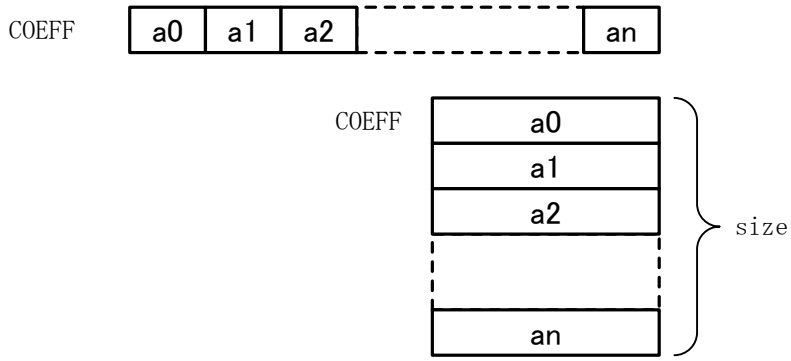
２０	投影データテーブルを格納したアドレスが不正、投影データ数エラー
２１	出力エッジテーブルを格納したアドレスが不正
２２	オフセット設定エラー
２３	スケールダウン量設定エラー
２４	荷重係数のサイズ設定エラー
２５	荷重係数テーブルを格納したアドレスが不正
－	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

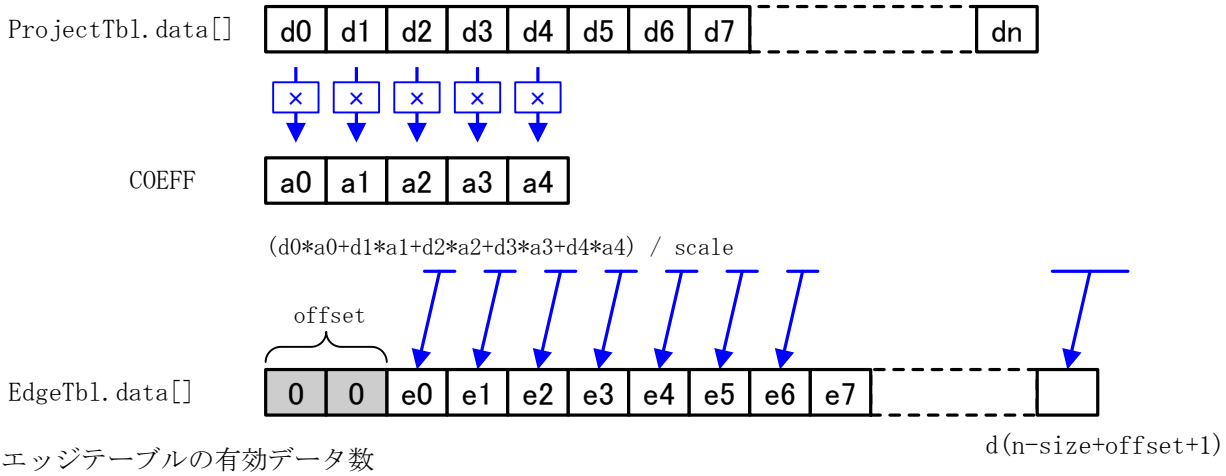
ProjectTblに設定された画像データ及び投影データをCOEFFテーブルの空間オペレータでフィルタリンクし、EdgeTblに出力します。
LINETBLのフォーマット及び内容は、ProjectLine()の詳細情報の欄を参照して下さい。

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 → ProjectTbl.num - size + offset + 1

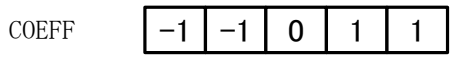
エッジテーブルの先頭アドレス

EdgeTbl.lower → ProjectTbl.lower

offset の設定値の目安は、以下のようになります。

```
offset = INT( size / 2 - 0.5 )
```

設定例



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	エッジ成分ラインテーブルを格納したアドレスが不正
２１	一次元論理座標のキャリパテーブルを格納したアドレスが不正
２２	抽出モード設定エラー
－	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

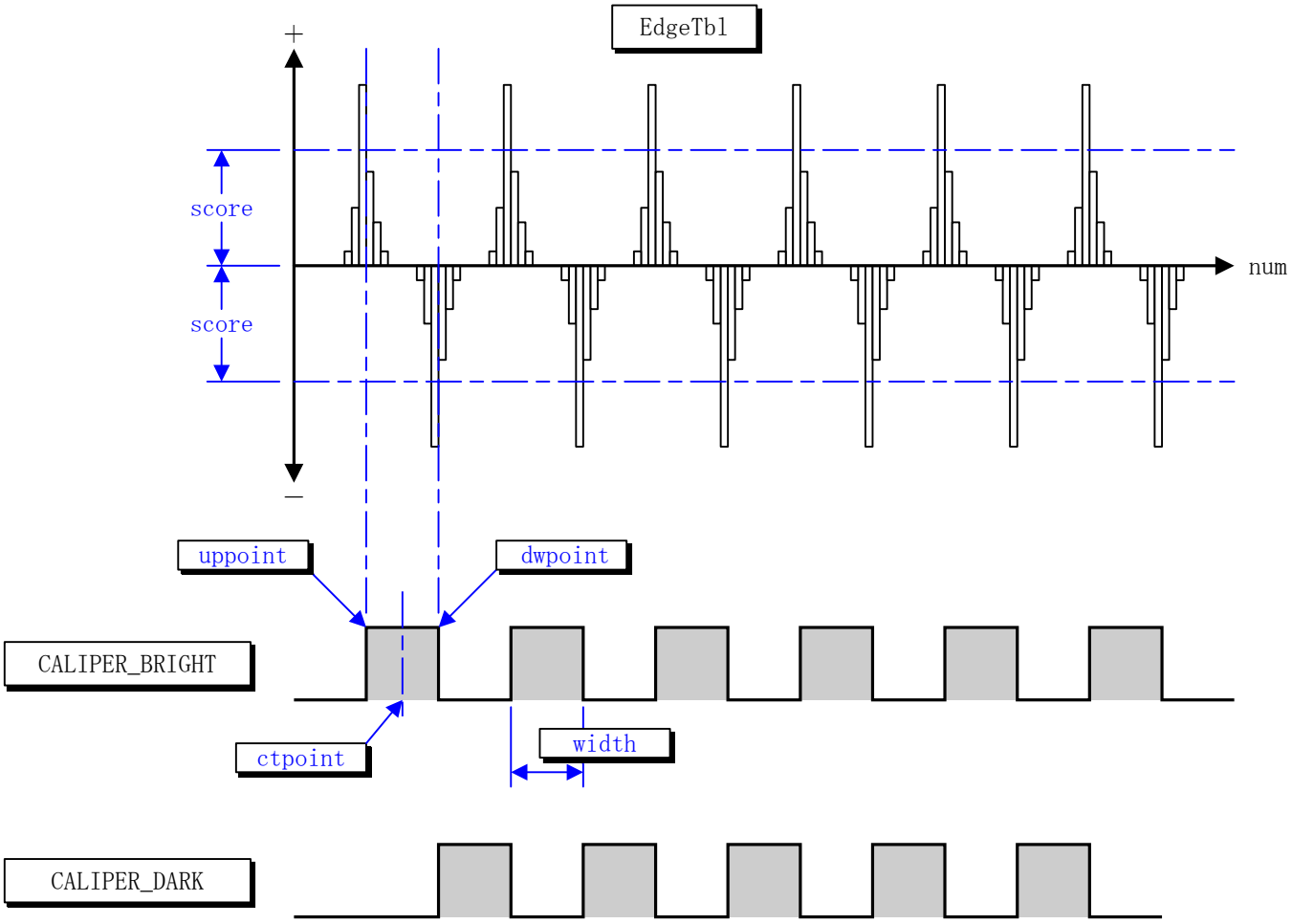
EdgeTblのエッジ成分からscoreをしきい値としてアップエッジとダウンエッジを検出し、抽出モードmodeに従いオブジェクトの開始点、終了点、中心点、幅を一次元論理座標で抽出し、LineCaliperTblに出力します。

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。
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）
CALIPERTBL	*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	一次元論理座標のラインキャリパテーブルを格納したアドレスが不正、抽出オブジェクト数が範囲外
２１	二次元論理座標のキャリパテーブルを格納したアドレスが不正
２２	投影を行ったラインウィンドウテーブルを格納したアドレスが不正を格納したアドレスが不正
２３	キャリブレーションデータテーブルを格納したアドレスが不正
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：abs(dx)＞abs(dy)　１：abs(dx)≦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）

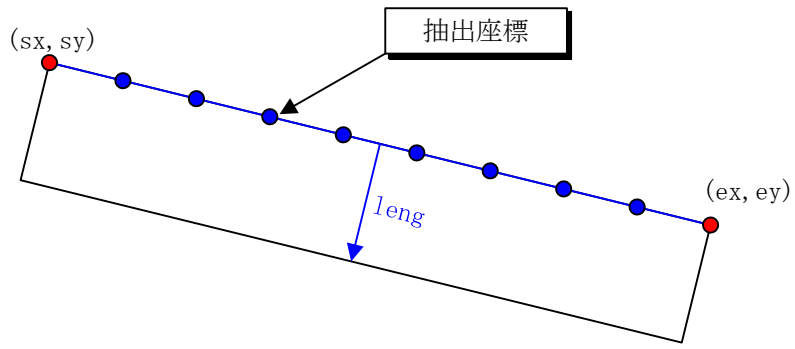
```
typedef struct {
        short           sx;
        short           sy;
        short           ex;
        short           ey;
        short           leng;
        short           opt;
} LINEWINDOW;
```

●ラインウィンドウテーブル

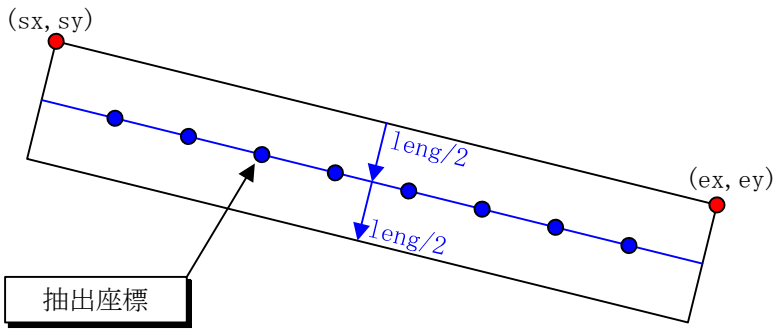
LINEWINDOW	説明
sx	開始点座標X
sy	開始点座標Y
ex	終了点座標X
ey	終了点座標Y
leng	投影幅
opt	オプション オプション：内容 0：ラインウィンドウ直線上の座標を抽出 1：ラインウィンドウ直線とleng幅との中心座標を抽出 2：ラインウィンドウ直線とleng幅間隔の座標を抽出 詳細を以下に示します。

● LINEWINDOWのメンバ変数 opt の詳細

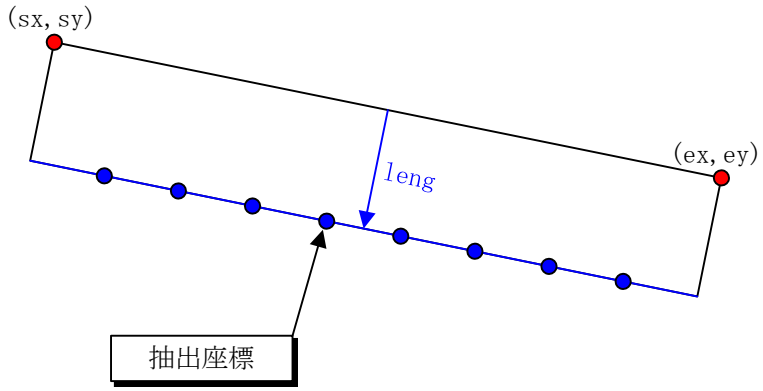
```
opt ： 0
```



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(atan(dy/dx)) //　mode(1)： sin(atan(dx/dy))
cos	mode(0)： cos(atan(dy/dx)) //　mode(1)： cos(atan(dx/dy))
Factor	mode(0)： cos(atan(dy/dx)) //　mode(1)： sin(atan(dx/dy))

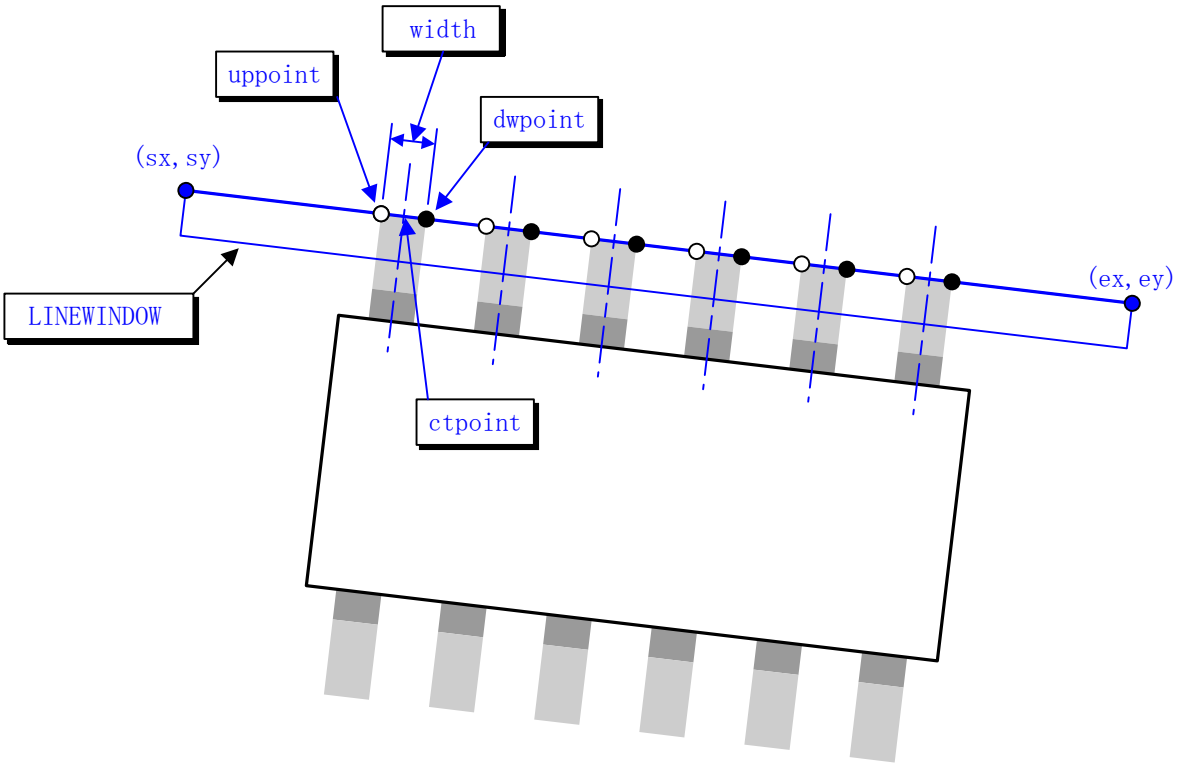
CFACTORのフォーマットを下記に示します。
vp900.hで宣言しています。

```
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

２０	エッジ成分ラインテーブルを格納したアドレスが不正
２０１	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

２０	キャリブレーションデータテーブルを格納したアドレスが不正
２１	ラインウィンドウテーブルを格納したアドレスが不正
２２	投影幅設定エラー
２３	ラインウィンドウ幅の方向設定エラー
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

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	エッジ成分ラインテーブルを格納したアドレスが不正
２１	一次元論理座標のラインエッジ抽出テーブルを格納したアドレスが不正
２２	抽出モード設定エラー
－	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

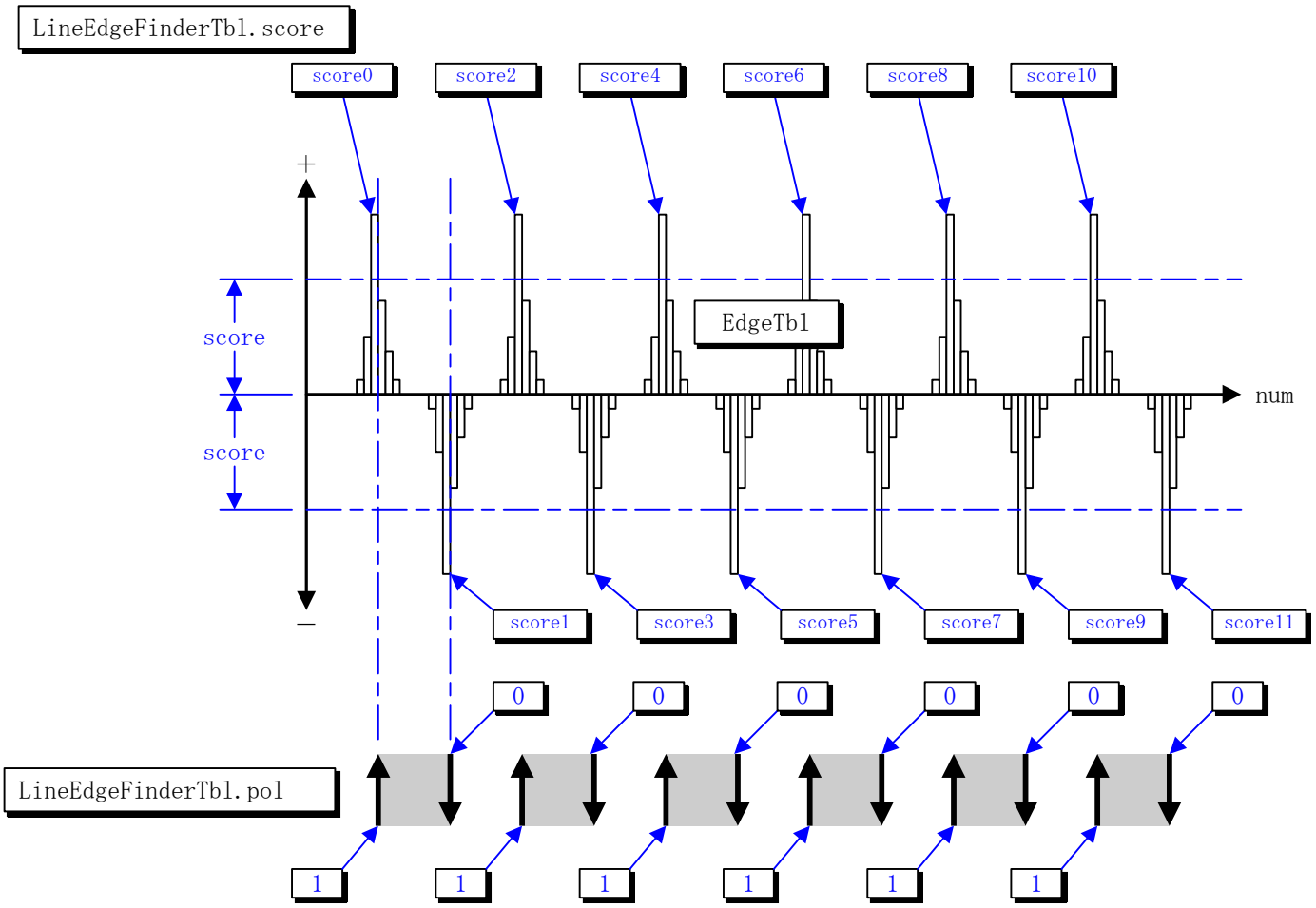
EdgeTblのエッジ成分から、scoreをしきい値としてエッジを検出し、そのエッジの位置を一次元論理座標で抽出し、LineEdgeFinderTblに出力します。
本コマンドでは、抽出したエッジの強さや極性を個別に抽出できます。

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;
```

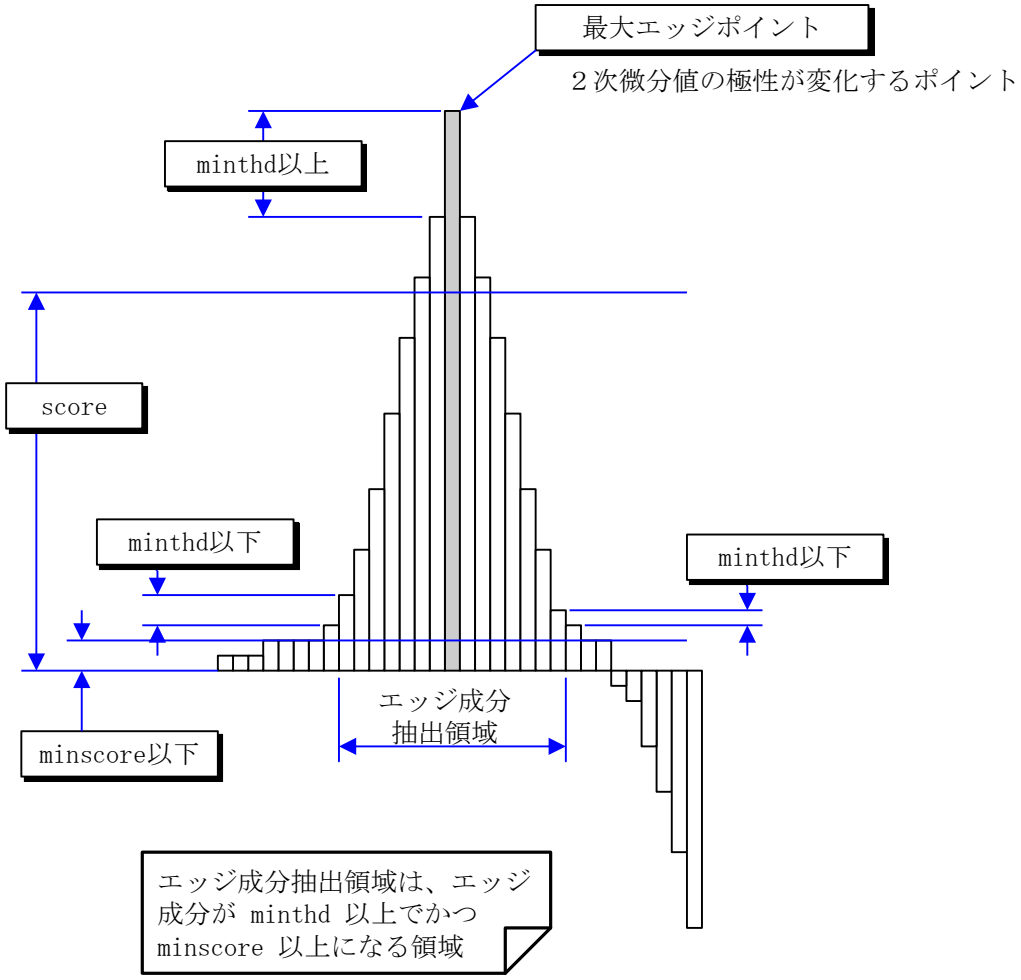
LINEEDGEFINDERTBL	説明
num	抽出エッジの個数
opt	未使用
pos[_EDGEFINDER_DATA_NUM]	抽出エッジの位置（一次元論理座標）
pol[_EDGEFINDER_DATA_NUM]	抽出エッジの属性 属性 / 内容 0 / 明 ⇒ 暗 1 / 暗 ⇒ 明
score[_EDGEFINDER_DATA_NUM]	抽出エッジのスコア

一次元論理座標ラインエッジ抽出テーブル（LINEEDGEFINDERTBL）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以上のポイントを求め、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もチェックしており、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次微分の値が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	一次元論理座標ラインエッジ抽出テーブルを格納したアドレスが不正
２１	二次元論理座標エッジ抽出テーブルを格納したアドレスが不正
２２	投影を行ったラインウィンドウテーブルを格納したアドレスが不正、ライン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

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;
```

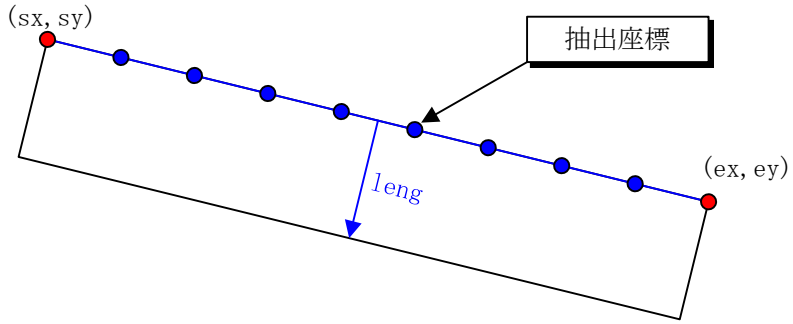
EDGEFINDERTBL	説明
num	抽出エッジの個数
opt	未使用
pos[_EDGEFINDER_DATA_NUM]	抽出エッジの位置（二次元論理座標）
pol[_EDGEFINDER_DATA_NUM]	抽出エッジの属性 属性：0 ⇒ 明 ⇒ 暗 属性：1 ⇒ 暗 ⇒ 明
score[_EDGEFINDER_DATA_NUM]	抽出エッジのスコア

```
typedef struct {
       short           sx;
       short           sy;
       short           ex;
       short           ey;
       short           leng;
       short           opt;
} LINEWINDOW;
```

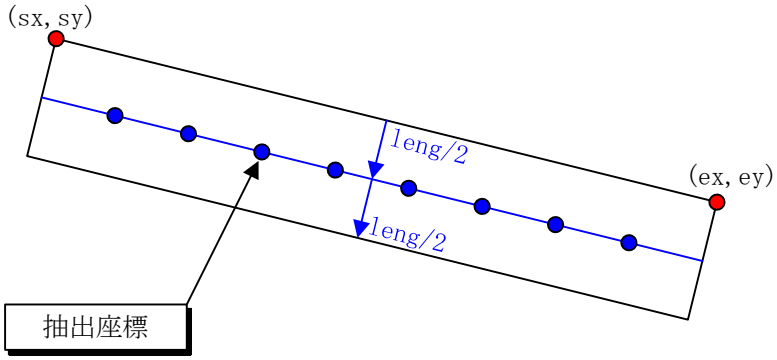
LINEWINDOW	説明
sx	開始点座標X
sy	開始点座標Y
ex	終了点座標X
ey	終了点座標Y
leng	投影幅
opt	オプション 0：ラインウィンドウ直線上の座標を抽出 1：ラインウィンドウ直線とleng幅との中心座標を抽出 2：ラインウィンドウ直線とleng幅間隔の座標を抽出 詳細を次ページに示します。

● LINEWINDOWのメンバ変数 opt の詳細

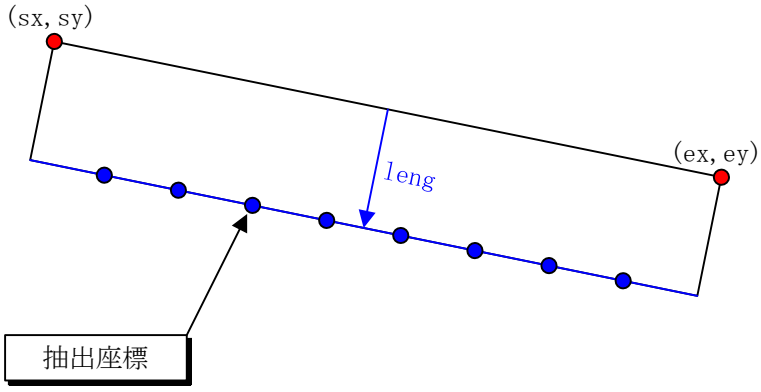
opt ： 0



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	Y16	Y32	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

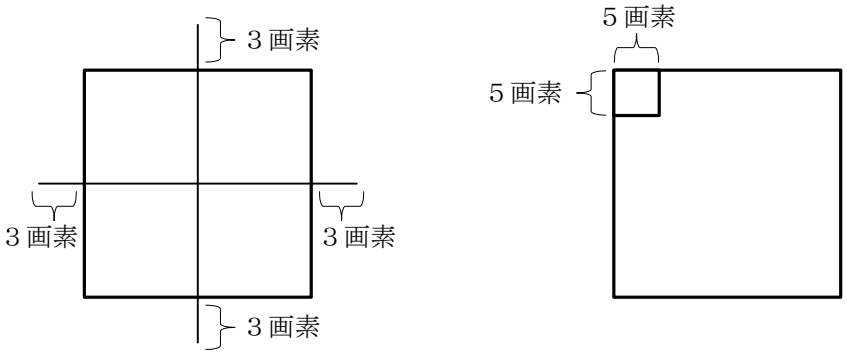
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	画面空き領域、または範囲外（不当画面番号エラー）、画面属性不正
２１	描画モード設定範囲外
－	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

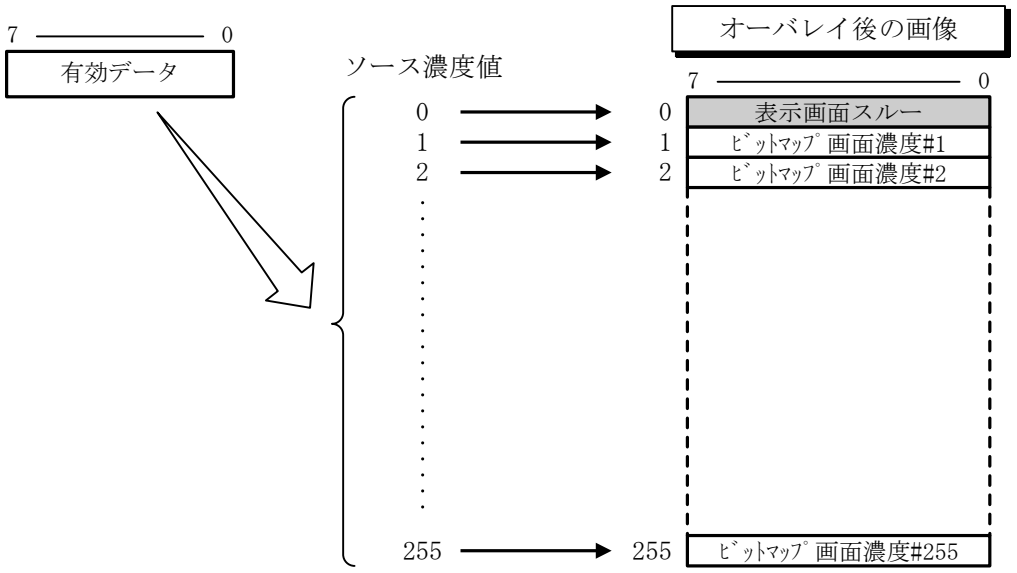
描画モードに従い線または矩形を描画します。
図形描画コマンドの描画は、ローカルＣＰＵで実行します。コマンド実行中はＣＰＵのキャッシュに一時的に蓄えられるため、実際の画像メモリ（ＩＭ）には書き込まれていない部分が出てきます。RefreshImg( )コマンドにより描画が全て終了した時点で、そのキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。

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（オーバレイ画面）のオーバ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	Y16	Y32	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
１０　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
２１　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

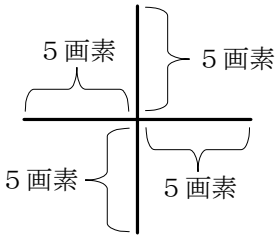
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

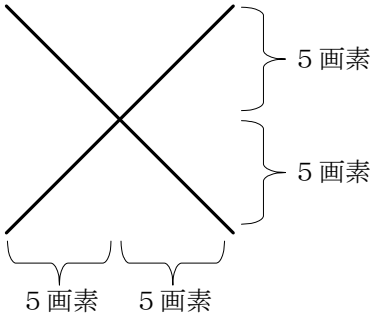
十字カーソルを描画します。
図形描画コマンドの描画は、ローカルＣＰＵで実行します。コマンド実行中はＣＰＵのキャッシュに一時的に蓄えられるため、実際の画像メモリ（ＩＭ）には書き込まれていない部分が出てきます。RefreshImg( )コマンドにより描画が全て終了した時点で、そのキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。

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)

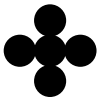


×印(mode ： 1)



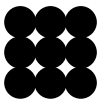
点(mode ： 2)

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	Y16	Y32	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
１０	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
２１	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で実行します。コマンド実行中はＣＰＵ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	Y16	Y32	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
１０　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
２１　描画モード設定エラー
２２　頂点Ｘ、Ｙ座標列を返すアドレスが不正
２３　頂点数が０未満
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

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で実行します。コマンド実行中はＣＰＵ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;
```

3.4.5 円描画（ PutCircle ）

❑ C言語API

```
int ret = PutCircle( IMGID ImgID , enum IPDrawLineMode mode , int xp , int yp ,int r ,
          int col );
```

❑ パラメータ

IMGID　ImgID　描画画面の画面番号

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

int　xp　中心Ｘ座標

int　yp　中心Ｙ座標

int　r　半径

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	Y16	Y32	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
１０	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
２１	描画モード設定範囲外
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

（xp,yp）を中心とする半径ｒの円形を描画し、中を塗りつぶします。
図形描画コマンドの描画は、ローカルＣＰＵで実行します。コマンド実行中はＣＰＵ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。

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

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	Y16	Y32	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
１０　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
２１　描画モード設定範囲外
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

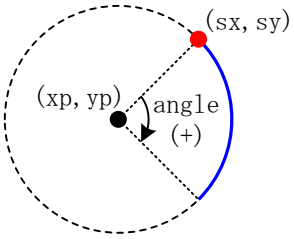
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

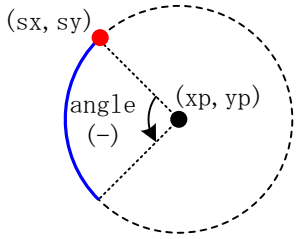
（xp, yp）を中心、（sx, sy）を始点とする角angleの円弧を描画します。
図形描画コマンドの描画は、ローカルＣＰＵで実行します。コマンド実行中はＣＰＵのキャッシュに一時的に蓄えられるため、実際の画像メモリ（ＩＭ）には書き込まれていない部分が出てきます。RefreshImg( )コマンドにより描画が全て終了した時点で、そのキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。

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

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	Y16	Y32	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

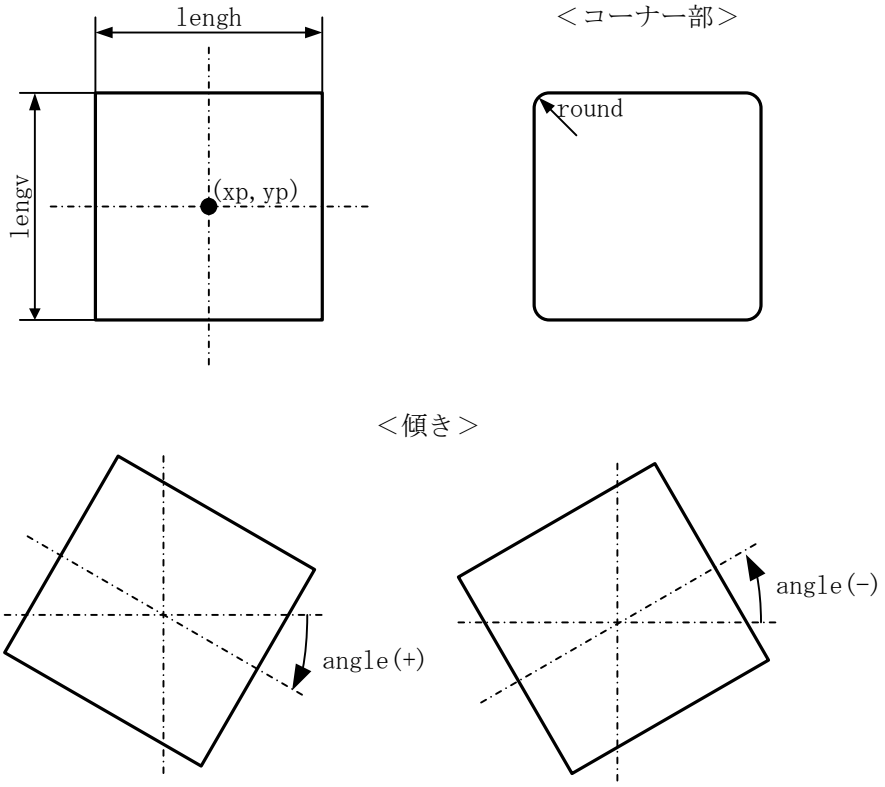
２　ＯＳシステムエラー
３　ＩＭＰ処理タイムアウト
１０　画面空き領域、または範囲外（不当画面番号エラー）、画面属性不正
２１　描画モード設定範囲外
２４　横幅が０未満
２５　縦幅が０未満
４８　動的メモリ確保エラー
－　エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

（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で実行します。コマンド実行中はＣＰＵ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 lengh ,
          int lengv , int round , float angle , int col );
```

❑ パラメータ

IMGID　ImgID　描画画面の画面番号

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

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	Y16	Y32	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

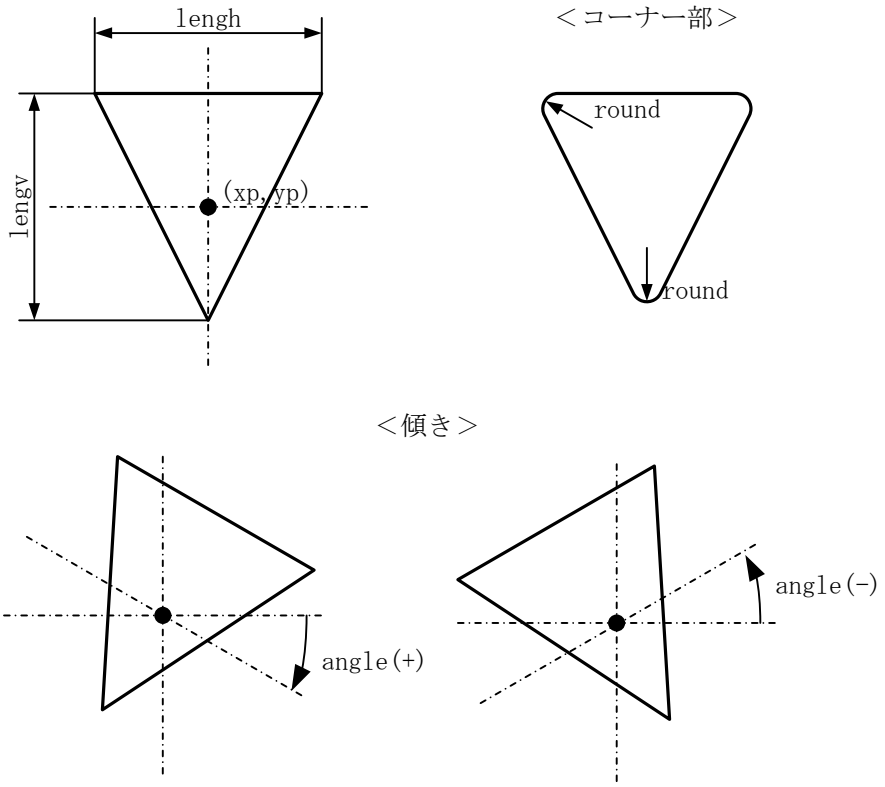
２　ＯＳシステムエラー
３　ＩＭＰ処理タイムアウト
１０　画面空き領域、または範囲外（不当画面番号エラー）、画面属性不正
２１　描画モード設定範囲外
２４　横幅が０未満
２５　縦幅が０未満
４８　動的メモリ確保エラー
－　エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

（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で実行します。コマンド実行中はＣＰＵ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

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	Y16	Y32	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

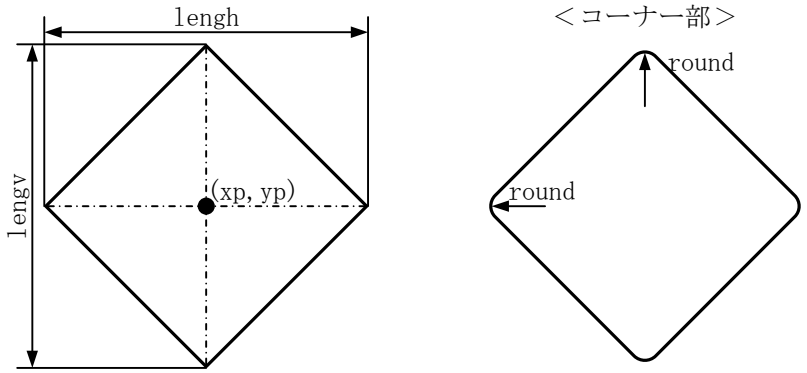
２　ＯＳシステムエラー
３　ＩＭＰ処理タイムアウト
１０　画面空き領域、または範囲外（不当画面番号エラー）、画面属性不正
２１　描画モード設定範囲外
２４　横幅が０未満
２５　縦幅が０未満
４８　動的メモリ確保エラー
－　エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

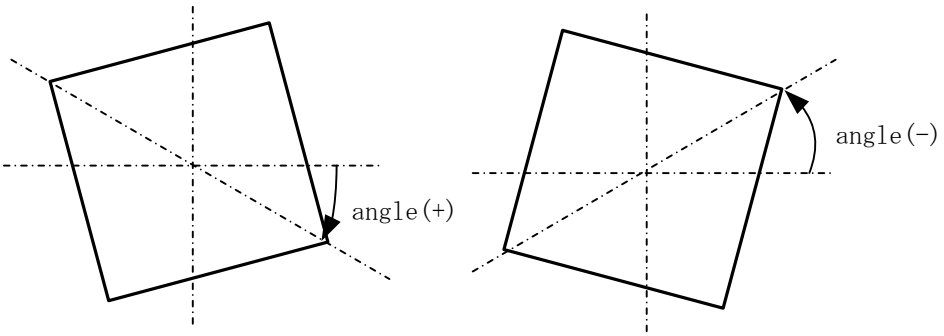
（xp,yp）を中心座標、横幅（lengh）、縦幅（lengv）、角の円弧の半径（round）、傾き（angle）とするひし形を描画します。
図形描画コマンドの描画は、ローカルＣＰＵで実行します。コマンド実行中はＣＰＵのキャッシュに一時的に蓄えられるため、実際の画像メモリ（ＩＭ）には書き込まれていない部分が出てきます。RefreshImg( )コマンドにより描画が全て終了した時点で、そのキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。

ひし形と各パラメータの関係を示します。



＜コーナー部＞

＜傾き＞



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
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	Y16	Y32	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

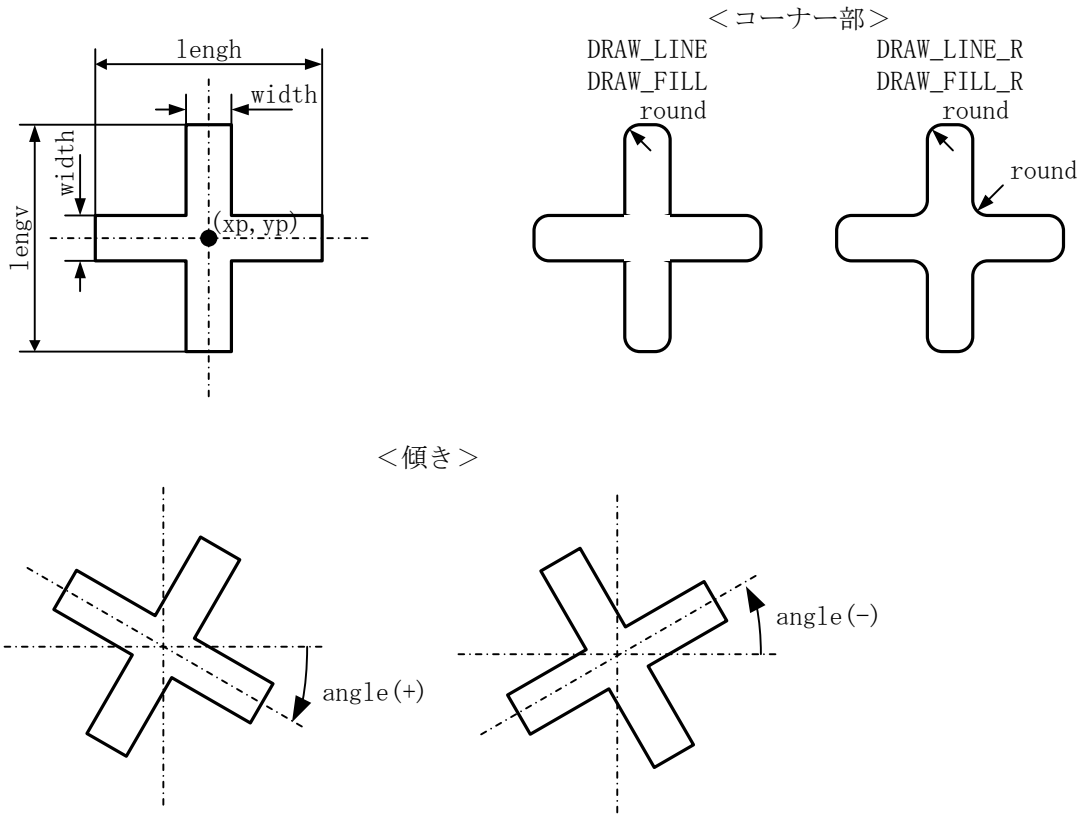
２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
１０	画面空き領域、または範囲外（不当画面番号エラー）、画面属性不正
２１	描画モード設定範囲外
２４	横幅が０未満
２５	縦幅が０未満
２６	腕幅が０未満
４８	動的メモリ確保エラー
－	エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

（xp,yp）を中心座標、横幅（lengh）、縦幅（lengv）、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で実行します。コマンド実行中はＣＰＵ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
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	Y16	Y32	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

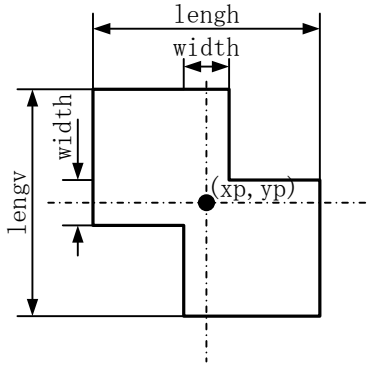
２　ＯＳシステムエラー
３　ＩＭＰ処理タイムアウト
１０　画面空き領域、または範囲外（不当画面番号エラー）、画面属性不正
２１　描画モード設定範囲外
２４　横幅が０未満
２５　縦幅が０未満
２６　腕幅が０未満
４８　動的メモリ確保エラー
－　エラーリセットコマンド（ClearIPError）未発行

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

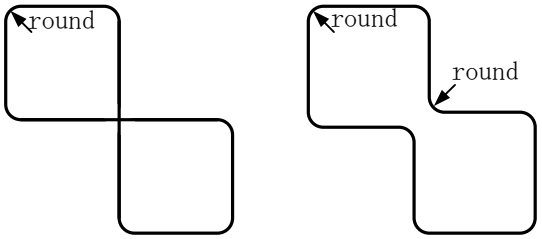
❑ 機能

（xp,yp）を中心座標、横幅（lengh）、縦幅（lengv）、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で実行します。コマンド実行中はＣＰＵ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。

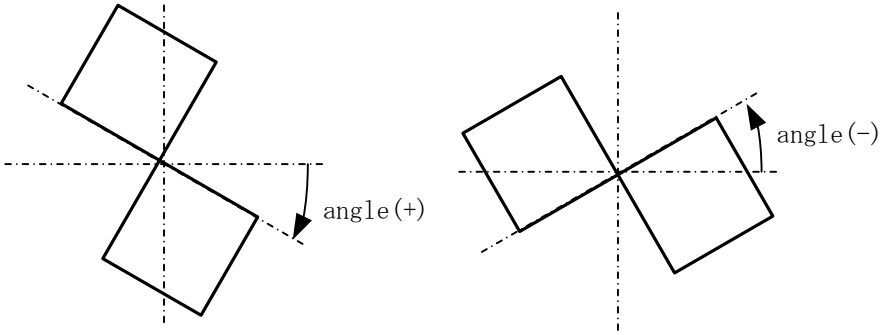
２四角形と各パラメータの関係を示します。



＜コーナー部＞



＜傾き＞



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	
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	Y16	Y32	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
１０　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
２１　描画モード設定範囲外
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

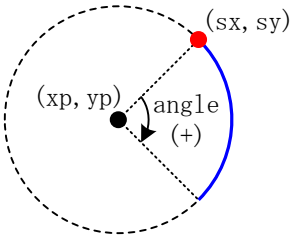
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

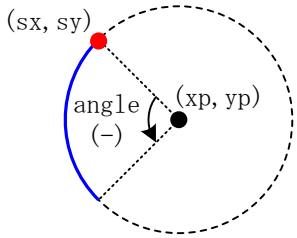
（xp,yp）を中心、（sx,sy）を始点とする角angleの円弧を描画します。なお、パラメータがフロート型である以外はPutArc()と同じです。
図形描画コマンドの描画は、ローカルＣＰＵで実行します。コマンド実行中はＣＰＵのキャッシュに一時的に蓄えられるため、実際の画像メモリ（ＩＭ）には書き込まれていない部分が出てきます。RefreshImg()コマンドにより描画が全て終了した時点で、そのキャッシュメモリから画像メモリに強制的にデータを書き出して下さい。

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）



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	Y16	Y32	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）

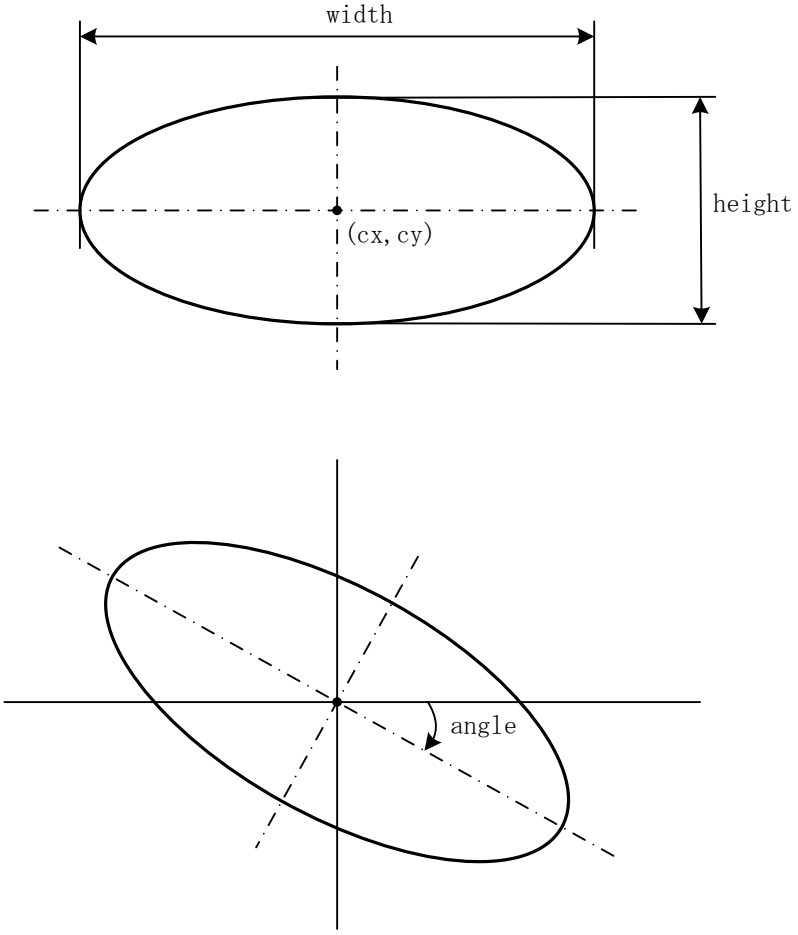
❑ エラーコード

２　ＯＳシステムエラー
３　ＩＭＰ処理タイムアウト
１０　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
２１　描画モード設定範囲外
４５　パラメータの設定条件違反
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

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で実行します。コマンド実行中はＣＰＵ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*(long)x_radius)-1) < 2*a
（条件２） 2*b >= a*((2*(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

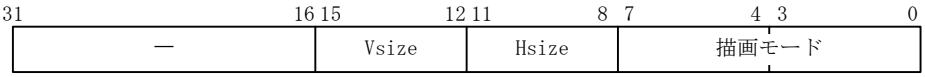
```
int ret = PutString( IMGID ImgID , char *string , int mode , int sx , int sy , int col);
```

❑ パラメータ

IMGID　ImgID　　描画画面の画面番号

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



※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	Y16	Y32	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
１０	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
２１	文字列テーブルを返すアドレスが不正
２２	文字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

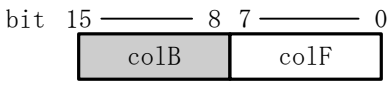
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

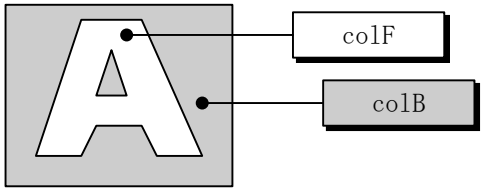
❑ 機能

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のコードで設定してください。

文字描画コマンドの描画は、ローカルＣＰＵで実行します。コマンド実行中はＣＰＵ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,"半角ｶﾅも使用できます",BACKCOL_NOWRITE,128,80,15)
```

4. リモート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 ターゲットアプリケーション情報取得（ GetTargetAppInfo ）

❑ C言語API

```
int ret = GetTargetAppInfo( void *pParam, int size );
```

❑ パラメータ

void	*pParam	NVPターゲットアプリケーション情報格納領域アドレス (NVPで設定した情報が格納される)
int	size	NVPターゲットアプリケーション情報サイズ （最小４バイト、最大６４Ｋバイト）

❑ 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 NVPターゲットアプリケーション情報サイズが64Kバイトを超えている。
－４１	内部オブジェクトエラー

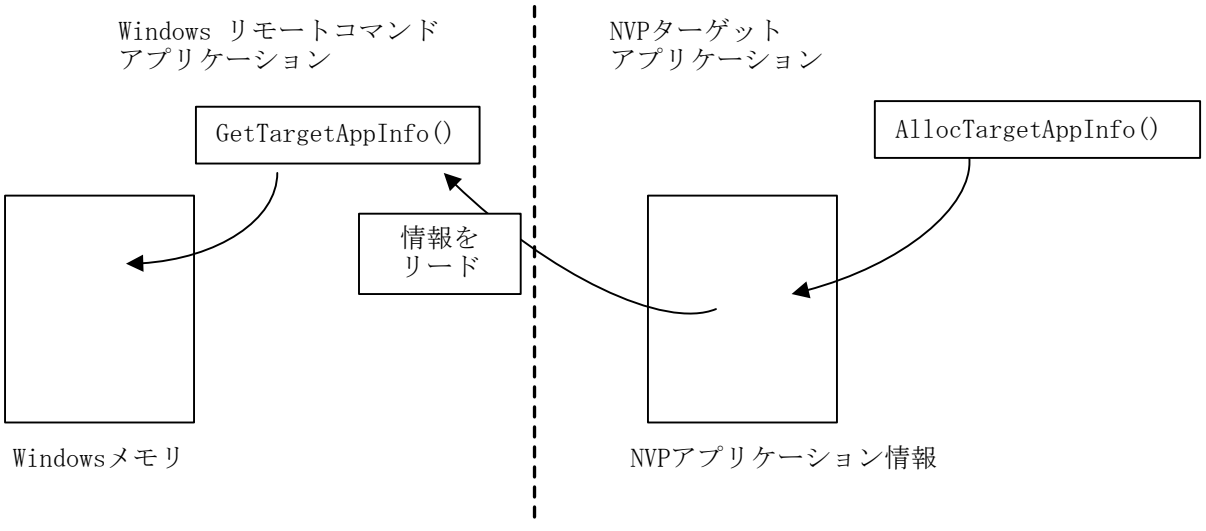
※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

NVPターゲットアプリケーション（NVPユーザーアプリケーション）内で「AllocTargetAppInfo()」コマンドで設定したデータをPC側で取得します。インテリジェントモジュール、画像処理タスクのアドレスなどを取得する際に使用します。情報格納領域(pParam)は、指定した取得サイズ(size)分確保してください。

本コマンドはPC側のコマンドです。NVP-Linux側では使用できません。
本コマンド実行前にNVP-Linux側の「AllocTargetAppInfo()」コマンドでデータを設定しておく必要があります。

NVP-Linux側でAllocTargetAppInfo()が実行されていない場合や本コマンドで指定した取得サイズ(size)がAllocTargetAppInfo()で確保したサイズより大きい場合はエラー（エラーコードー４１）になります。



4.1.2 ターゲットアプリケーションの情報設定(AllocTartgetAppInfo)

❑ C言語API

```
int ret = AllocTargetAppInfo( void *pParam, int size );
```

❑ パラメータ

void	*pParam	NVPターゲットアプリケーション情報格納領域アドレス (Windowsで取得する情報が格納された領域のアドレス)
int	size	NVPターゲットアプリケーション情報サイズ(バイト)

❑ リターンパラメータ

int	ret	正常終了（０）またはエラーコード

❑ エラーコード

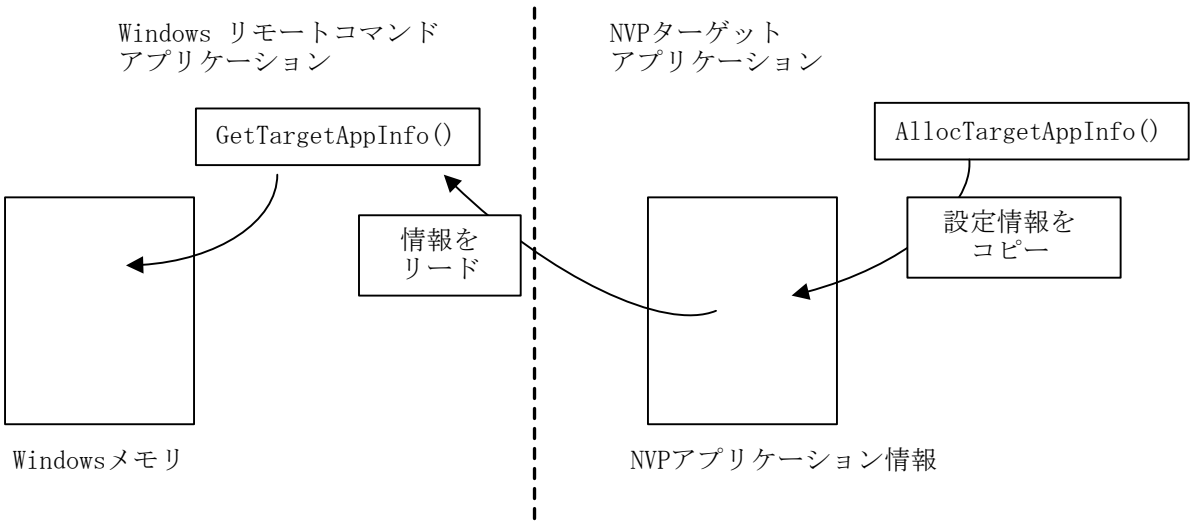
－１７	パラメータエラー
－４１	FreeTargetAppInfoを実行していない
－３３	NVPターゲットアプリケーション情報領域未確保

※ 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側のリモートコマンドを使用したアプリケーションに対して「GetTargetAppInfo()」で取得する情報格納領域を確保し指定された情報をコピーします。情報領域の先頭アドレスを「pParam」に指定し情報のサイズを「size」に指定してください。「FreeTargetAppInfo()」で確保した情報格納領域を解放することができます。なお、本コマンドはPC側のリモートコマンドからは実行できません。

PC側のアプリケーションで「GetTargetAppInfo()」の実行前に本コマンドでデータを設定しておく必要があります。本コマンドは、「StartIP()」実行前に使用することができます。



4.1.3　ターゲットアプリケーションの情報解放(FreeTargetAppInfo)

❑ C言語API

```
int ret = FreeTargetAppInfo( void );
```

❑ 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

「AllocTargetAppInfo()」で確保した情報格納領域を解放します。
「AllocTargetAppInfo()」で領域確保していない場合は、何もせず終了します。
本コマンドは、「StartIP()」実行前に使用することができます。また、本コマンドはNVP-Linux側とPC側の両方で使用できます。

4.1.4 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
０～１６個のパラメータをNVP-Linuxアプリケーションに
Ｃ言語のパラメータとして渡すことができます。

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	ＰＩＯ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

－１７　　パラメータエラー
－４１　　内部オブジェクトエラー
－４２　　タスク無効

※ 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ＰＩＯ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、NVP-Linux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。

4.1.5 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_16	15	第１６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。

4.1.6 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_16	15	第１６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。

4.1.7 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_16	15	第１６パラメータ

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

－１７　　パラメータエラー
－５８　　確保しようとしている容量がNVP-Linux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

ＰＣ上にある配列データをNVP-Linuxのプログラムにパラメータとして引き渡します。NVP-Linux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NVP-Linux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、NVP-Linux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、SetParamTableExt()コマンドでは、その制限はありません。

4.1.8 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_16	15	第１６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

－１７　　パラメータエラー
－５８　　確保しようとしている容量がNVP-Linux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NVP-Linuxシステムメモリに確保します。本コマンドでは、NVP-Linux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、NVP-Linux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、NVP-Linux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NVP-Linux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NVP-Linux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NVP-Linux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(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_16	15	第１６パラメータ

void　*pctbl　　NVP-Linux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　　NVP-Linux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

－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

NVP-LinuxのプログラムからAllocParamTable()及びAllocParamTableExt()コマンドでPC側に確保した領域にデータを転送します。その場合、NVP-Linux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て使用します。通常、AllocParamTable()及びAllocParamTableExt()コマンドにより、配列データ（入力）の領域をNVP-Linux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NVP-Linux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。

4.1.10　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_16	15	第１６パラメータ

unsigned long iptbl_addr　書き込みを行うNVP-Linux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。

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

－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

PC側にある配列データをNVP-Linuxのプログラムにパラメータとして引き渡します。NVP-Linux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NVP-Linux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、NVP-Linux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。

4.1.11　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_16	15	第１６パラメータ

unsigned long iptbl_addr　書き込みを行うNVP-Linux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NVP-Linuxのプログラムにパラメータとして引き渡します。NVP-Linux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NVP-Linux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、NVP-Linux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NVP-Linux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NVP-Linux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。

4.1.12　Linux-NVP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 mem_addr, unsigned long Count,
          int Mode );
```

❑ パラメータ

void *pPCMem	PC側データ領域を返すアドレス
unsigned long mem_addr	Linux－NVP側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 ０：４Ｂアクセス １：１Ｂアクセス ２：２Ｂ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	PC側データ領域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

NVP-Linuxシステムメモリからデータを読み出し、ＰＣ側のデータ領域に書き込みます。

4.1.13　Linux-NVP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 mem_addr, unsigned long Count,
          int Mode );
```

❑ パラメータ

void *pPCMem	PC側データ領域を返すアドレス
unsigned long mem_addr	Linux－NVP側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 ０：４Ｂアクセス １：１Ｂアクセス ２：２Ｂ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

－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

ＰＣ側のデータを NVP-Linux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＃２５５

unsigned long module_addr　　インテリジェントモジュールのアドレス。
NVP-Linux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

－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

NVP-Linux側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＃２５５

MODULE_PARAM_TBL *prmtbl　モジュールのパラメータテーブルのパケットアドレス。
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。
本コマンドはPC側でのみ実行可能です。

4.3 PIO割込オブジェクト制御

4.3.1 割込オブジェクトの生成（ CreateInterruptLink ）

❑ C言語API

```
int ret = CreateInterruptLink( INT_DEVICE_OBJ *obj , int taskID , int event , 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。 リンクする割込デバイスの情報を設定します。
int	taskID	割込デバイスオブジェクトとリンクするタスクのタスクＩＤ。
int	event	割込時の動作を指定します。

割込イベント	対応数	説明
INTEVENT_NOP	0	イベントなし
INTEVENT_WAKEUP	1	ウェイクアップ

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）またはエラーコード

❑ エラーコード

－５	Linuxライブラリまたはドライバエラー
－１７	パラメータエラー
－４１	内部データ不正
－４２	CreateIPTaskで生成したスレッド以外で実行された

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

割込リンクオブジェクトの生成。本コマンドにより、ＰＩＯ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;
```

<table>
<tr><th>割込デバイスオブジェクト</th><th>説明</th></tr>
<tr><td>intdev</td><td>割込デバイス番号
<table>
<tr><th>デバイス番号</th><th>対応数</th><th>説明</th></tr>
<tr><td>INTDEV_UNLINK</td><td>0</td><td>割込デバイスなし(現在指定不可)</td></tr>
<tr><td>INTDEV_PIO</td><td>2</td><td>ボード上のＰＩＯ割込</td></tr>
</table></td></tr>
<tr><td>intbit</td><td>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。
<table>
<tr><th>デバイス番号</th><th>対応数</th><th>説明</th></tr>
<tr><td>INTDEV_PIO</td><td>0-15</td><td>ボード上のＰＩＯのビットに対応しています。</td></tr>
</table></td></tr>
<tr><td>evtpri</td><td>「０」を設定して下さい。</td></tr>
<tr><td>opt</td><td>オプション（「０」を指定）</td></tr>
</table>

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）またはエラーコード

❑ エラーコード

－５	Linuxライブラリまたはドライバエラー
－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

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、ＰＩＯ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）またはエラーコード

❑ エラーコード

－５	Linuxライブラリまたはドライバエラー
－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

本コマンドにより、ＰＩＯ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）またはエラーコード

❑ エラーコード

－５	Linuxライブラリまたはドライバエラー
－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

本コマンドにより、ＰＩＯ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

4.4.1 ユーザタスクの生成（ CreateIPTask ）

❑ C言語API

```
int tskid = 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（スレッド）を生成します。

❑ リターンパラメータ

int	tskid	＞０ ：タスクＩＤ ≦０ ：エラーコード

❑ エラーコード

－５	Linuxライブラリまたはドライバエラー
－１７	パラメータエラー
－３３	タスク情報テーブル確保エラー
－３４	生成可能な画像処理タスク数を超えた
－４１	内部データ不正（タスク管理リソース確保失敗）

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

ユーザタスク(スレッド)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。なお、ユーザータスクは、Linuxのスレッドとして生成されますが、タスクＩＤは、LinuxスレッドＩＤではなく、本コマンドシステムの管理ＩＤになります。

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;
       int              affinity;
       unsigned long    arg;

} CREATE_TASK_TBL;
```

<table>
<tr><th>タスク情報生成テーブル</th><th>説明</th></tr>
<tr><td>task_addr</td><td>タスクの先頭アドレス
<table>
<tr><th>task_addr</th><th>内容</th></tr>
<tr><td>0</td><td>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。</td></tr>
<tr><td>0以外</td><td>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。</td></tr>
</table></td></tr>
<tr><td>priority</td><td>タスクプライオリティが「０」の場合、スケジューリングポリシーがタイムシェアスケジューリングになります。「１」以上の場合は、SCHED_FIFOのリアルタイムスケジューリングになります。なお、「TASK_PRI_NORMAL」は「３３」です。
タスクのプライオリティ
<table>
<tr><th>プライオリティ</th><th colspan="2">説明</th></tr>
<tr><td>TASK_PRI_HIGHEST</td><td>ノーマル+２</td><td rowspan="5">高い
▲
低い</td></tr>
<tr><td>TASK_PRI_ABOVE_NORMAL</td><td>ノーマル+１</td></tr>
<tr><td>TASK_PRI_NORMAL</td><td>ノーマル（初期値）</td></tr>
<tr><td>TASK_PRI_BELOW_NORMAL</td><td>ノーマル-１</td></tr>
<tr><td>TASK_PRI_LOWEST</td><td>ノーマル-２</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>pbuff_size</td><td>パラメータバッファのサイズ（バイト）</td></tr>
<tr><td>stack_size</td><td>スタックサイズ（バイト）</td></tr>
<tr><td>param_opt</td><td>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。
<table>
<tr><th>param_opt</th><th>内容</th></tr>
<tr><td>0</td><td>パラメータを渡さない</td></tr>
<tr><td>1</td><td>パラメータを渡す</td></tr>
</table></td></tr>
<tr><td>task_opt</td><td>タスクオプション
<table>
<tr><th>task_opt</th><th>内容</th></tr>
<tr><td>0</td><td>ＣＰＵアフィニティパラメータ無効
親プロセスの値が継承されます</td></tr>
<tr><td>０ｘ１０</td><td>ＣＰＵアフィニティパラメータ有効
「affinity」パラメータを設定してください</td></tr>
</table></td></tr>
<tr><td>affinity</td><td>ＣＰＵアフィニティ
使用するＣＰＵが複数の場合、下表の値を論理和してください。
また、「０」を指定すると親プロセスの設定が継承されます。
<table>
<tr><th>affinity</th><th>内容</th></tr>
<tr><td>1</td><td>ＣＰＵ０ が有効</td></tr>
<tr><td>2</td><td>ＣＰＵ１ が有効</td></tr>
<tr><td>4</td><td>ＣＰＵ２ が有効</td></tr>
<tr><td>8</td><td>ＣＰＵ３ が有効</td></tr>
</table></td></tr>
<tr><td>arg</td><td>task_addr 情報に「０」以外のアドレスを設定したタスクモジュールに対して引き渡すパラメータを指定します</td></tr>
</table>

【注意事項】

(1) 本Linuxシステムは４コアのマルチプロセッサのスケジューラーが動作しているため、プライオリティ設定したスレッドが複数のＣＰＵで実行されている場合、別ＣＰＵの間でのプライオリティの値による優先度制御は機能しません。必要な場合は、各スレッドで同一のＣＰＵとなるよう、ＣＰＵアフィニティを設定してください。

(2) リアルタイムスケジューリングを使用するとアプリケーション実行中にリアルタイムクラスへ割り当てるCPU時間が制限される場合があります(RTスロットリング機能)。RTスロットリングによるリアルタイムクラスへのCPU割り当て時間の制限をLinuxの「sysctl」コマンドで適切に調整してください。

例：リアルタイムクラスへのCPU時間の制限をなくす場合

```
sysctl -w kernel.sched_rt_runtime_us=-1
```

(3) リアルタイムスケジューリングを多用すると、Linuxのシステムサービス等が動作できなくなり、アプリケーションの正しい動作ができなくなる可能性があります。十分評価、検討していただき使用してください。

(4) 画像処理コマンドを使用するタスク(スレッド)のスタックサイズは64KB以上確保してください。

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 pioint_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。 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 pioint_task	ＰＩＯからの割込で起動されるモジュールのアドレス。 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。「０」を指定して下さい。

❑ リターンパラメータ

int ret	０：正常終了またはエラーコード

❑ エラーコード

－１７	パラメータエラー
－４１	内部データ不正
－４２	CreateIPTaskで生成したスレッド以外のタスクが指定された

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

NVP-Linuxアプリケーション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ＩＤ。 CreateIPTask()コマンドで生成しRegistIPTask()コマンドで登録された タスクＩＤ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パケットアドレス。 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：正常終了またはエラーコード

❑ エラーコード

－５	Linuxライブラリまたはドライバエラー
－１７	パラメータエラー
－４２	タスク無効

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 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渡すことができます。

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ＩＤ。CreateIPTask()コマンドで生成し RegistIPTask()コマンドで登録されたタスクＩＤ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パケットアドレス。 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	ＰＩＯ入力の割込で起動します

❑ リターンパラメータ

int	ret	０：正常終了またはエラーコード

❑ エラーコード

－５	Linuxライブラリまたはドライバエラー
－１７	パラメータエラー
－４２	タスク無効

※ 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ＰＩＯ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渡すことができます。

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。
NVP-LinuxアプリケーションからＰＣへのシグナルを送信する場合、NVP-Linuxアプリケーション側からの 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

NVP-Linuxアプリケーションの割込モジュールとタスクモジュールの終了シグナルを受信するまでウェイトします。割込モジュールは、SendSignaltoPC()コマンドで終了シグナルを送信し、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。

本コマンドの前に必ずReadyforWaitIPTaskSignal()コマンドを実行し、シグナル番号を設定してください。

本コマンドを使用する場合、本コマンドとReadyforWaitIPTaskSignal()コマンドは他コマンドとは別のデバイスＩＤをオープンし、そのデバイスＩＤを使用して処理を行ってください。

本コマンドはPC側からのみ実行可能です。

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。
WaitforIPTaskSignal()コマンドでウェイトするsignalパラメータ
と同じ値を設定してください。

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

終了シグナルウェイトのシグナル番号の設定を行います。WaitforIPTaskSignal()コマンドでウェイトするデバイスＩＤに対して必ず１回行ってください。
WaitforIPTaskSignal()コマンドでシグナルウェイトを行う場合、WaitforIPTaskSignal()コマンド実行前に本コマンドを実行し、同じユーザシグナル番号を設定してください。
本コマンドはPC側からのみ実行可能です。

4.4.7 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

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。

本コマンド実行により、ＴＣＰ接続がクローズされ切断されます。再度リモートコマンドを使用する場合は、切断された接続のデバイスＩＤをクローズし、再度オープンしたデバイスＩＤを使用してください。

4.4.8 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。

❑ リターンパラメータ

int	ret	０：正常終了またはエラーコード

❑ エラーコード

－５	Linuxライブラリまたはドライバエラー
－１７	パラメータエラー
－４２	タスク無効

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 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。

4.4.9 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。

❑ リターンパラメータ

int	ret	０：正常終了またはエラーコード

❑ エラーコード

－５	Linuxライブラリまたはドライバエラー
－１７	パラメータエラー
－４２	タスク無効

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 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。

4.4.10　タスクモジュールの削除（ CancelIPTask ）

❑ C言語API

```
int ret = CancelIPTask( int taskID );
```

❑ パラメータ

int taskID	タスク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。

❑ リターンパラメータ

int	ret	０：正常終了またはエラーコード

❑ エラーコード

－５	Linuxライブラリまたはドライバエラー
－１７	パラメータエラー
－４２	タスク無効
－４９	CancelIPTask()が発行された

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

ＰＩＯ割込モジュールとタスクモジュールのウェイト状態をキャンセルします。
StartIPTask()コマンド発行前のタスクに本コマンドを発行すると、対象タスクが終了します。
SleepIPTask()コマンドでウェイトしている場合、ウェイトが解除されリターンコード「－４９」が出力されます。
RegistIPTask()コマンドで登録されたタスクは、ウェイト状態の場合タスクが終了します。

4.4.11　タスクモジュールのスリープ（ SleepIPTask ）

❑ C言語API

```
int ret = SleepIPTask( void );
```

❑ パラメータ

なし

❑ リターンパラメータ

int	ret	０：WakeupIPTaskで待ち解除またはエラーコード

❑ エラーコード

－５	Linuxライブラリまたはドライバエラー
－４１	内部データ不正
－４２	タスク無効
－４９	CancelIPTask()が発行された

※ 詳細は、付録Ａ「エラーコード一覧」を参照して下さい。

❑ 機能

ＰＩＯ割込モジュール（自タスク）をスリープ状態にします。このスリープ状態は、他のタスクや割込みハンドラからのウェイクアップ(CancelIPTask()コマンドが発行)まで継続します。
他タスクからCancelIPTask()コマンドが発行された場合、本コマンドはリターンコードが「－４９」になります。このリターンコードでスリープが解除されることでタスクを終了することができます。
このコマンドは、NVP-Linux側アプリケーションでのみ有効です。

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　　NVP-LinuxアプリケーションからＰＣ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：正常終了またはエラーコード

❑ エラーコード

－５　　Linuxライブラリまたはドライバエラー
－１７　　パラメータエラー
－４１　　内部データ異常
－４２　　ReadyforWaitIPTaskSignal()でのユーザシグナル番号未設定(PC側のみ)
－５１　　ＬＡＮ通信異常

※ 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-Linux側アプリケーションのＰＩＯ割込モジュールからPC側に終了シグナルを送信する場合に使用します。PC側は、WaitforIPTaskSignal()コマンド等でNVP-Linux側アプリケーション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NVP-Linux側アプリケーション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。

5. 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

－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ポートからデータ入力を行います。
ＮＶＰ－Ａｘ４３０ＣＬのＩ／ＯポートはＤ０～Ｄ７の８ビット、ＮＶＰ－Ａｘ４３０ＡＣＬ／ＮＶＰ－Ａｘ４３５ＣＬ／ＮＶＰ－Ａｘ４３５ＦＣＬはＤ０～Ｄ１５の１６ビットが有効です。他のビットは無効で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

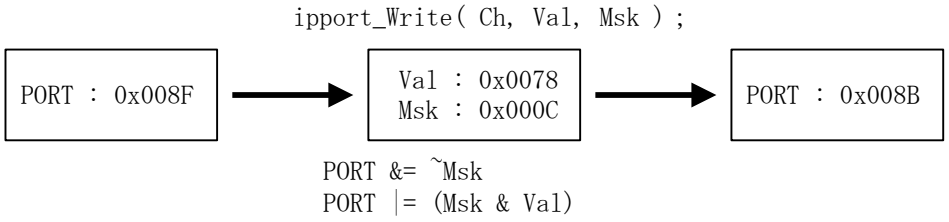
－１７　　パラメータエラー

※ 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－Ａｘ４３０ＣＬはＤ０～Ｄ７の８ビット、ＮＶＰ－Ａｘ４３０ＡＣＬ／ＮＶＰ－Ａｘ４３５ＣＬ／ＮＶＰ－Ａｘ４３５ＦＣＬはＤ０～Ｄ１５の１６ビットが有効です。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

5.2 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のチャネル（０～２）
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
                              SCICOM_115200 (115200) 115200 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(未使用 「0」を指定してください)

  int        SndBufSiz;   送信バッファサイズ(未使用 「0」を指定してください)

} SCICOM_SETUP_INFO;
```

❑ 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

エラーコードはGetLinuxErrno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 12.Linuxエラー」を参照して下さい。

❑ 機能

ＳＣＩ(Serial Communication Interface)デバイスに対して、デバイスのオープン、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。

接続先は、Ch:0はCOM1、Ch:1はCOM2、Ch:2はカメラリンクとなります。なお、Ch:0(COM1)は、出荷設定でLinux起動メッセージ出力とログインシェルが使用しているため、使用する場合はユーザーズマニュアル「3.9.2 COM1でのログインシェルを無効にする方法」を参照し無効設定してください。

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（０～２）

❑ 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

エラーコードはGetLinuxErrno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 12.Linuxエラー」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（０～２）
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 異常終了（－１）
char	*pBuf	受信データを格納したアドレス

❑ エラーコード

エラーコードはGetLinuxErrno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 12.Linuxエラー」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。

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のチャネル（０～２）
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 異常終了（－１）
char	*pBuf	受信データを格納したアドレス

❑ エラーコード

エラーコードはGetLinuxErrno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 12.Linuxエラー」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。

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（０～２）
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 異常終了（－１）

❑ エラーコード

エラーコードはGetLinuxErrno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 12.Linuxエラー」を参照して下さい。

❑ 機能

指定したＳＣＩのＣｈの指定されたバイト数のデータを指定したＳＣＩのＣｈに出力します。

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	μＳ 単位の計測
M_SEC	1	ｍＳ 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 異常終了（－１）

❑ エラーコード

エラーコードはGetLinuxErrno()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 12.Linuxエラー」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 ｍＳ 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);          // μＳ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 異常終了（－１）

❑ エラーコード

エラーコードはGetLinuxErrno()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 12.Linuxエラー」を参照して下さい。

❑ 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
		0xFFFFFFFF	：異常終了

❑ エラーコード

エラーコードはGetLinuxErrno()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 12.Linuxエラー」を参照して下さい。

❑ 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ｍＳ単位ですが、SetIPTimeMode()コマンドにより、μＳ，ｍＳ，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);
```

タイマスタート(startGetIPTime)を実行せず本APIをコールした場合、不定値が読み出されます。

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）

❑ エラーコード

―

❑ 機能

ダミーで実装しています。正常終了を返します。

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（Ｈｚ）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（Ｈｚ）を格納したアドレス

❑ エラーコード

エラーコードはGetLinuxErrno()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 12.Linuxエラー」を参照して下さい。

❑ 機能

パフォーマンスカウンタ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

エラーコードはGetLinuxErrno()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 12.Linuxエラー」を参照して下さい。

❑ 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)

❑ 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

エラーコードはGetLinuxErrno()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 12.Linuxエラー」を参照して下さい。

❑ 機能

指定した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	Linuxシステムクロックに登録する時刻(ローカル時刻) （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

❑ 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 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（０～９９９）

❑ エラーコード

エラーコードはGetLinuxErrno()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 12.Linuxエラー」を参照して下さい。

❑ 機能

NVPのLinuxに対して、システムクロック時刻を登録します。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。時刻データはタイムゾーンが"Asia/Takyo"のローカル時間(JST)です。

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。

32ビットのLinuxを使用しているため、設定できる範囲はJSTで1970年1月1日 9時00分00秒※1 から2038年1月19日 12時14分07秒（UTCでは1970年1月1日 0時00分00秒※1から2038年1月19日 3時14分07秒）までとなります。

本コマンドでは、NVPに搭載されているRTCへの登録はおこないません。RTCへの登録は、Linuxシェルで「hwclock -w --utc」で登録したシステムクロックをRTCへ反映してください。

※1 「UTCで1970年1月1日9時0分0秒＋Linuxが起動してからの累計時間」以前の時刻は設定できません

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	Linuxシステムクロックに登録する時刻(ローカル時刻)（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）または異常終了（－１）

❑ 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 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（０～９９９）

❑ エラーコード

エラーコードはGetLinuxErrno()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 12.Linuxエラー」を参照して下さい。

❑ 機能

Linuxシステ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。時刻データはタイムゾーンが"Asia/Takyo"のローカル時間です。
なお、SetIPTime()コマンド、又はLinuxシェルの「date」コマンドで時刻をあらかじめ登録する必要があります。

32ビットのLinuxを使用しているため、取得できる範囲は"Asia/Takyo"のローカル時間で1970年1月1日 9時00分00秒から2038年1月19日 12時14分07秒（UTCでは1970年1月1日 0時00分00秒から2038年1月19日 3時14分07秒）までとなります。

5.3.10　エラーコードの取得（ GetLinuxErrno ）

❑ C言語API

```
int  ret = GetLinuxErrno( void );
```

❑ パラメータ

なし

❑ リターンパラメータ

int	ret	保持しているエラー番号

❑ 機能

保持しているエラー番号を返します。詳細は、付録Ａ「エラーコード一覧 12.Linuxエラー」を参照して下さい。

5.4 WDTコマンド

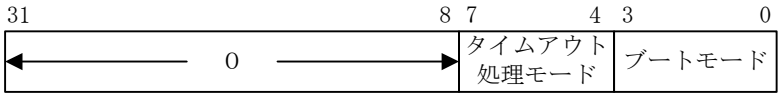
5.4.1 WDTの起動（ WDT_Start ）

❑ 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



int　　tmo　　タイムアウト時間（ｍＳ）

０～　　１：	Ｍｉｎ　１ｍＳ	Ｍａｘ　３ｍＳ
２～　１０：	Ｍｉｎ　１０ｍＳ	Ｍａｘ　３０ｍＳ
１１～　３０：	Ｍｉｎ　３０ｍＳ	Ｍａｘ　９０ｍＳ
３１～　１００：	Ｍｉｎ１００ｍＳ	Ｍａｘ ３００ｍＳ
１０１～　１０００：	Ｍｉｎ　１Ｓ	Ｍａｘ　３Ｓ
１００１～１００００：	Ｍｉｎ　１０Ｓ	Ｍａｘ　３０Ｓ
１０００１～６００００：	Ｍｉｎ　６０Ｓ	Ｍａｘ １８０Ｓ

❑ 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
－４１　　内部データ異常

※ 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。

5.4.2　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）またはエラーコード

❑ エラーコード

－４１	内部データ異常

❑ 機能

ＷＤＴ（Watchdog Timer）を停止します。

5.4.3　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）または異常終了（－１）

❑ 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。

5.4.4 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) 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。

ソフトリセットは、ResetIPコマンドを実行した場合に発生します。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)は、ResetIPコマンドの実行により起動されたことを示します。

5.4.5　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）またはエラーコード

❑ エラーコード

－４１	内部データ異常

❑ 機能

ＷＤＴ（Watchdog Timer）のステータスをクリアします。

6. ファイルアクセスコマンド

6.1 ファイルアクセス

6.1.1 ファイルオープン（ fmOpen ）

❑ C言語API

```
FM_HFILE ret = fmOpen( char *filename , int mode );
```

❑ パラメータ

```
char     *filename        ファイル名（パス名付き）
int      mode             アクセスモード
```

モード	対応数	説明
FM_DEFAULT	0x00	指定されたファイルが存在している場合、そのファイルを開きます。指定されたファイルが存在していない場合、新しいファイルを作成します。本モードは、FM_RDWR属性で開きます。そのため、指定されたファイルが存在している場合、オープン時に既存のデータは削除されません。既存ファイルよりも小さいサイズを上書きする場合は、「FM_CREATE | FM_RDWR」属性を指定して下さい。
FM_RDONLY	0x01	読み出しモードで開きます。ファイルがない場合は、エラーになります。データの書き換えはできません。
FM_WRONLY	0x02	書き込みモードで開きます。ファイルがない場合は、エラーになります。オープン時に既存のデータは削除されません。
FM_RDWR	0x04	読み出し、書き込みモードで開きます。ファイルがない場合は、エラーになります。オープン時に既存のデータは削除されません。
FM_CREATE	0x08	指定されたファイルが存在していない場合、新しいファイルを作成します。指定されたファイルが存在している場合、そのファイルのデータを削除します。
FM_APPEND	0x20	指定されたファイルが存在してる場合、追記モードで開きます。オープン時にファイルポインタをファイルの終端へ移動します。 指定されたファイルが存在していない場合、他モードの条件でファイルが生成されればファイルポインタは先頭となります。

※各モードはビットフィールドに対応していますので、複数組み合わせて使用可能です。

組み合わせ条件は以下となります。組み合わせ条件を逸脱した場合、エラーとなります。

(1)FM_RDONLY、FM_WRONLY、FM_RDWRはどれか1つのみ設定可能となります。
(2)FM_DEFAULTと他モード設定を組み合わせた場合、他モード設定が優先されます。
(3)FM_CREATEとFM_APPENDはどちらか一方のみ設定可能となります。

ファイルオープン時、指定したファイルが存在する場合以下となります。

(1)ファイルではなくパスを指定した場合はエラーとなります。
(2)ファイルの多重指定はしないでください。但し、エラーとはなりません。
(3)ファイル属性がリードオンリーの場合、FM_WRONLY、またはFM_RDWRを指定するとエラーとなります。
(4)ファイル属性がライトオンリーの場合、FM_RDONLY、またはFM_RDWRを指定するとエラーとなります。
(5)FM_CRERATEかFM_APPENDを指定しなかった場合、既存のファイルをオープンし、ファイルポインタはファイルの先頭へ移動します。
(6)FM_RDONLYとFM_APPENDを組み合わせた場合、リードのファイルポインタは先頭となります。
(7)シンボリックリンクを指定した場合、その参照先をオープンします。

❑ リターンパラメータ

FM_HFILE	ret	0以外 ：ファイルハンドル 0 ：異常終了

❑ エラーコード

異常終了した場合 GetLinuxErrnoでエラー情報を取得できます。

❑ 機能

指定されたファイルをアクセスモード(mode)に従いオープンします。新規作成や書き込みの場合はアクセスモードを「FM_RDWR | FM_CREATE」、読み出しのみの場合は「FM_RDONLY」を指定して下さい。
ファイル名及びパス名は、英半角大文字、小文字、数字のみ指定可能です。
ファイル名、または単一フォルダ名の最大長は255byteを、パス＋ファイル名の最大長は512byteをそれぞれ超えないで下さい。超えた場合はエラーとなります。
指定したパスが存在しない場合、エラーを返します。事前にディレクトリを作成してください。
パスに、相対パス(./、../)は指定できますが、チルダ(~/)は指定できません。チルダを指定した場合エラーとなります。

なお、Windows系のフォーマット(FAT,NTFS)は、アクセスモードを「FM_RDONRY」及び「FM_WRONLY」でファイルをオープンすることはできません。

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、GetLinuxErrno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 12.Linuxエラー」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。

Linuxは、ファイルをクローズしても直ぐにディスク装置にデータが書き込まれず、カーネルのディスクキャッシュにデータが書き込まれます。Linuxの「sync」システムコールを使用しディスクキャッシュからディスク装置にデータを書き込んでください。
なお、ディスク装置自体にもバッファキャッシュを持っており、最終的に物理的な記憶部に書き込みが終了するまで時間がかかる場合がありますので注意してください。

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、GetLinuxErrno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 12.Linuxエラー」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。
sizeに0を指定した場合、ファイルの内容は読み込まれず、正常終了(0)を返します。
ファイル読み込み後のファイルポインタは読み込んだデータの次のオフセットに移動します。
sizeの指定でファイルサイズより大きなサイズを指定した場合、またはファイルポインタ＋sizeがファイルサイズより大きい場合はエラーを返します。
エラーとなった場合、以降正しくファイル読み込みができませんので、当該ファイルハンドルのファイル読み込みを行いたい場合、ファイルクローズを行い、改めてファイルオープン処理からファイル読み込みを実施して下さい。

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、GetLinuxErrno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 12.Linuxエラー」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。
sizeに0を指定した場合、何も行われず正常終了(0)を返します。

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	0xFFFFFFFF 以外	：ファイルサイズ(byte)
		0xFFFFFFFF	：異常終了

❑ エラーコード

異常終了した場合、GetLinuxErrno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 12.Linuxエラー」を参照して下さい。

❑ 機能

ファイルハンドルで指定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、GetLinuxErrno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 12.Linuxエラー」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(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、GetLinuxErrno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 12.Linuxエラー」を参照して下さい。

❑ 機能

指定ファイル名を変更します。
ファイル名を変更する以外にも、同一ファイルシステム内のディレクトリ下へファイルを移動できます。

ファイル名、または単一フォルダ名の最大長は255byteを、パス＋ファイル名の最大長は512byteをそれぞれ超えないで下さい。超えた場合はエラーとなります。
dstfilenameが既存の場合は上書きされます。
srcfilenameとdstfilenameが同一ファイルの場合はなにも変更されずエラーにもなりません。
srcfilenameがディレクトリ名の場合は、dstfilenameにはsrcfilenameと同じディレクトリパスを指定できません。
変更前後どちらのパス名に対してもシンボリックリンクは辿りません。
srcfilenameがシンボリックリンクの場合は、シンボリックリンク自身の名前を変更します。
dstfilenameがシンボリックリンクの場合は、通常のパス名として処理し、dstfilenameの名前を変更します。
srcfilenameにディレクトリ名を指定すればディレクトリの名前を変更できます。
dstfilenameにパス付のファイル名を指定時する時、存在しないパスを指定した場合エラーとなります。
srcfilenameがディレクトリの場合は、dstfilenameもディレクトリとして下さい。
パス名のディレクトリ部分の名前を変更する場合（移動）は、dstfilenameには新規ファイル名を指定して下さい。但し、dstfilenameの示すディレクトリ内部にファイルが１つも無い場合、新規ファイル名としなくともパス名を変更することが出来ます(パスの上書き)。
srcfilenameとdstfilenameは同一ファイルシステムでなければなりません。

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、GetLinuxErrno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 12.Linuxエラー」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。
検索パスとファイル名にはワイルドカード(“*”,”?”)は使えません。
　検索ファイル名が255byteを超える場合、または検索ファイル名とパスの長さが512byteを超える場合、異常終了となります。
　シンボリックリンクを指定した場合、シンボリックリンクを検索します。この先のファイルは検索しません。

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_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	リザーブビットです。指定しても意味を持ちません

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、GetLinuxErrno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 12.Linuxエラー」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。
属性は、オーナー、グループ、ユーザ全てが変更されます。なお、Windows系(FAT,NTFS)フォーマットの場合、ファイル属性の変更はできません。
フラグがセット(FM_SET)の場合、本API実行後は指定したファイル属性となります。実行前と同じ属性だった場合、属性は変更されません。
フラグがリセット(FM_RESET)の場合、本API実行後はリードライト属性(FM_ATTR_RDWR)となります。実行前の属性がリードライト属性(FM_ATTR_RDWR)の場合、属性は変更されません。
属性はどれか１つのみ設定可能です。
ファイル属性を変更する場合は、指定するファイルが開かれていない状態で行って下さい。

6.1.10　PC―NVP-Linuxマウント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	PC側ディスク→NVP-Linuxマウントディスク
FLASH_TO_HDD	1	NVP-Linuxマウントディスク→PC側ディスク
FLASH_TO_FLASH	2	NVP-Linuxマウントディスク→ NVP-Linuxマウントディスク (fmCopyFile()がコールされます)

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

(1)コピーモード：HDD_TO_FLASH / FLASH_TO_HDD の場合

int　ret　　正常終了（０）または以下エラーコード

□エラーコード
-17　：パラメータエラー
-103 ：ファイルオープンエラー
-105 ：ファイルアクセスエラー
-106 ：PC側領域確保エラー

(2)コピーモード：FLASH_TO_FLASH の場合

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、GetLinuxErrno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 12.Linuxエラー」を参照して下さい。

❑ 機能

ＰＣ側ディスク→NVP-Linuxマウントディスク／NVP-Linuxマウントディスク→ＰＣ側ディスク／NVP-Linuxマウントディスク→NVP-Linuxマウント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NVPボード側からの呼び出しはできません。コピー先にすでにファイルがある場合、上書きされます。また、コピー元またはコピー先のファイルがライトオンリ属性またはコピー先のファイルがリードオンリまたはライトオンリ属性の場合はコピーできません。あらかじめ削除してください。
本コマンドを実行する前の準備としてコピー先のディスクに十分な空き容量があるか確認しコピーを行って下さい。

6.1.11　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、GetLinuxErrno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 12.Linuxエラー」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。

6.1.12　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、GetLinuxErrno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 12.Linuxエラー」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。
パス名、ディレクトリ名には、英半角大文字、小文字、数字のみ指定可能です。
パス名には絶対パスからの指定(″/”)と、カレントディレクトリからの相対パスの指定(“./”、”../”)が可能です。
パスにワイルドカード（″*”、″?”）を指定した場合エラーとなります。
ディレクトリの作成はオーナー属性で作成され、作成されたディレクトリは、オーナー属性、グループ属性、ユーザ属性全てが有効となります。
本APIはpathName全てのディレクトリを生成します。言い換えると、fmCreateDirectory(“aaa/bbb/ccc”, attr);は、aaa、bbb、cccディレクトリを順に作成します。なお、作成途中不正なディレクトリ名が存在する場合、正常なディレクトリ名までのディレクトリを作成し、リターンパラメータに-1を返します。
単一フォルダ名の最大長は255byteを、パス全体の最大長は512byteをそれぞれ超えないで下さい。超えた場合はエラーとなります。

6.1.13　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、GetLinuxErrno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 12.Linuxエラー」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。パス名、ディレクトリ名には、英半角大文字、小文字、数字のみ指定可能です。

6.1.14　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）

❑ エラーコード

異常終了した場合、GetLinuxErrno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 12.Linuxエラー」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。
検索パスとファイル名にはワイルドカード(“*”,”?”)が指定可能です。
検索ファイル名が255byteを超える場合、または検索ファイル名とパスの長さが512byteを超える場合、異常終了となります。
本関数は内部でワークエリアを確保します。検索を終了する場合、内部ワークエリアを解放するため、必ずfmFindClose関数を実行して下さい。

FM_FIND_TBL のフォーマット及び内容を、下記に示します。

```
typedef struct {
   unsigned long    attr;                 // ﾌｧｲﾙ属性
   unsigned long    size;                 // ﾌｧｲﾙｻｲｽﾞ
   time_t           time;                 // 最終書き込み時間
   unsigned long    reserved[2];          // ﾘｻﾞｰﾌﾞ
   char             name[C_FSZ_FILENAME]; // ﾌｧｲﾙ名(255byte)
} FM_FIND_TBL;
```

6.1.15　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） 異常終了（－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、GetLinuxErrno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 12.Linuxエラー」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()を呼び出します。
検索ファイル名が255byteを超える場合、または検索ファイル名とパスの長さが512byteを超える場合、異常終了となります。

FM_FIND_TBL のフォーマット及び内容を、下記に示します。

```
typedef struct {
   unsigned long    attr;                 // ﾌｧｲﾙ属性
   unsigned long    size;                 // ﾌｧｲﾙｻｲｽﾞ
   time_t           time;                 // 最終書き込み時間
   unsigned long    reserved[2];          // ﾘｻﾞｰﾌﾞ
   char             name[C_FSZ_FILENAME]; // ﾌｧｲﾙ名(255byte)
} FM_FIND_TBL;
```

6.1.16　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、GetLinuxErrno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 12.Linuxエラー」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、内部で確保されたワークエリアを解放します。

検索途中で検索を終了（中断）する場合も、必ず本APIを実行し検索ハンドルをクローズして下さい。

6.1.17　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_DIRECTORY	0x1000	ディレクトリ

❑ エラーコード

異常終了した場合、GetLinuxErrno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 12.Linuxエラー」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。

6.1.18　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 *pathname, long long *devicesize, long long *freesize );
```

❑ パラメータ

char	*pathname	マウントファイルシステムのパス名
long long	*disksize	デバイスのディスクサイズを格納する領域のアドレス
long long	*freesize	デバイス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	デバイスのディスクサイズ(byte)
long long	*freesize	デバイスの空きエリア容量(byte)

❑ エラーコード

異常終了した場合、GetLinuxErrno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 12.Linuxエラー」を参照して下さい。

❑ 機能

NVP-Linuxにマウントされているファイルシステムのディスクサイズとファイルシステムの空きエリア容量を取得します。

6.1.19　NVP-Linuxマウント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）または異常終了（－１）

❑ エラーコード

異常終了した場合、GetLinuxErrno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 12.Linuxエラー」を参照して下さい。

❑ 機能

NVP-Linuxにマウントされているディスク上のファイルをコピーします。コピー先にすでにファイルがある場合、上書きされます。また、コピー元のファイルがライトオンリ属性またはコピー先のファイルがリードオンリまたはライトオンリ属性の場合はコピーできません。あらかじめ削除してください。

本コマンドを実行する前の準備としてコピー先のディスクに十分な空き容量があるか確認しコピーを行って下さい。

7. Windowsアプリケーションサポートコマンド

7.1 Windowsアプリケーションサポート

7.1.1 IMデータを BMPに変換し転送(ReadBMPImg)

❑ C言語API

```
int ret = ReadBMPImg( IMGID ImgID, IMSizeInfo *pIMSize, char *pBmpImg,
                                        void *pBmpInf, enum BITMAP_MODE mode );
```

❑ 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（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
} IMSizeInfo;
```

❑ 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
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）
２２	ビットマップ画像データを返すアドレス不正、転送モード指定範囲外
４８	動的メモリ確保エラー
４９	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

ＩＭ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。

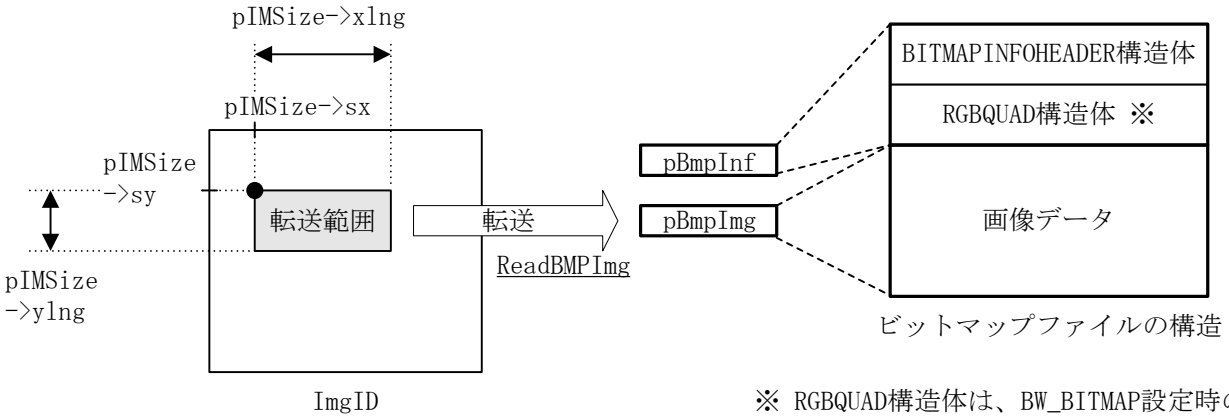
ビットマップ情報ヘッダ（BITMAPINFOHEADER）とLUT情報（RGBQUAD）の構造体を以下に示します。

BITMAPINFOHEADER構造体

フィールド	バイト数	内容
biSize	4	構造体サイズ（４０Ｂ）
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。

画面＼mode	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.xlng = width;
IMSize.ylng = 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 );
```

7.1.2 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;
```

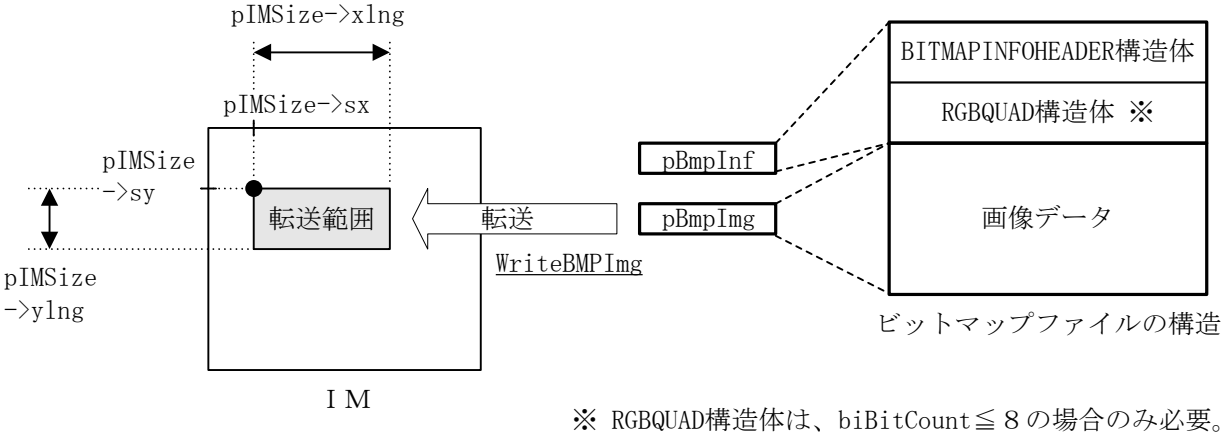
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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）
２２	ビットマップ画像データを返すアドレス不正
２３	ビットマップ情報ヘッダ＆ＬＵＴを返すアドレス不正
４８	動的メモリ確保エラー
４９	不定画面エラー
１３５	画面サイズ又は画面タイプが不正
－	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

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いるデータ）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	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	ＲＧＢ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１２８でクリアする	ＹＵＶデータをＹＵＶ画面に書き込む	ＲＧＢデータをＹＵＶデータに変換し、ＹＵＶ画面に書き込む
ＣＹＵＶ	有	パレットでB/WデータをＣＹＵＶデータに変換し、ＣＹＵＶ画面に書き込む	－	－
	無	B/WデータをY画面に書き込む。ＵＶ画面は１２８でクリアする	ＹＵＶデータをＣＹＵＶ画面に書き込む	ＲＧＢデータをＣＹＵＶデータに変換し、ＣＹＵＶ画面に書き込む
ＲＧＢ	有	パレットでB/WデータをＲＧＢデータに変換し、ＲＧＢ画面に書き込む	－	－
	無	B/WデータをＲＧＢ画面（R，G，Bそれぞれ）に書き込む。	エラー	ＲＧＢデータをＲＧＢ画面に書き込む。
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画面に書き込む	－	－
	無	B/WデータをＲＧＢ_Ｒ画面に書き込む。	YデータをＲＧＢ_Ｒ画面に書き込む	ＲＧＢデータをYデータに変換し、ＲＧＢ_Ｒ画面に書き込む
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画面に書き込む	－	－
	無	B/WデータをＲＧＢ_Ｇ画面に書き込む。	YデータをＲＧＢ_Ｇ画面に書き込む	ＲＧＢデータをYデータに変換し、ＲＧＢ_Ｇ画面に書き込む
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画面に書き込む	－	－
	無	B/WデータをＲＧＢ_Ｂ画面に書き込む。	YデータをＲＧＢ_Ｂ画面に書き込む	ＲＧＢデータをYデータに変換し、ＲＧＢ_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.xlng = width;
IMSize.ylng = height;

/* ＢＭＰデータをＩＭに転送 */
WriteBMPImg( ImgID, &IMSize, pBmpImg, pBmpInf, NULL );
```

7.1.3 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,
                              IMSizeInfo *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
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。
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
} IMSizeInfo;
```

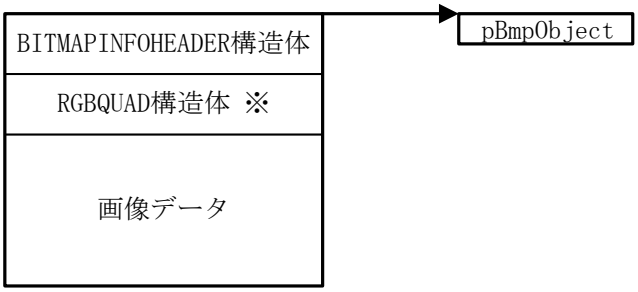
❑ エラーコード

２	ＯＳシステムエラー
３	ＩＭＰ処理タイムアウト
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）
２０	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不正
２２	転送モード指定範囲外
４８	動的メモリ確保エラー
４９	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。



画像メモリビットマップオブジェクト

※ 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。

・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.xlng = width;
IMSize.ylng = 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 );
```

7.1.4　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

２０	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不正
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

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。

7.1.5 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

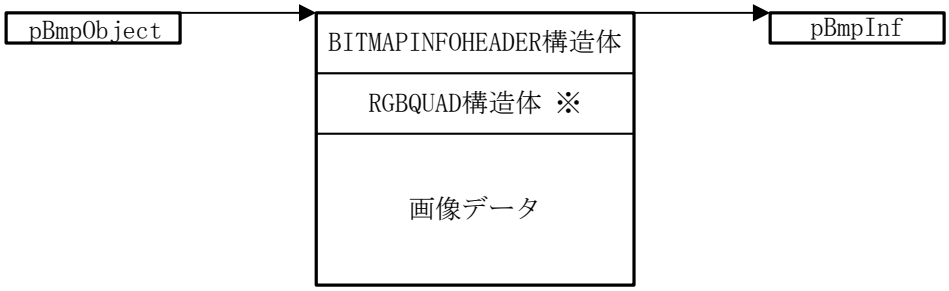
void *pBmpInf	!= NULL	ビットマップ情報ヘッダの先頭アドレス
	== NULL	エラー

❑ 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の先頭アドレスを取得します。



画像メモリビットマップオブジェクト

※ RGBQUAD構造体は、BW_BITMAP設定時のみ必要

7.1.6 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

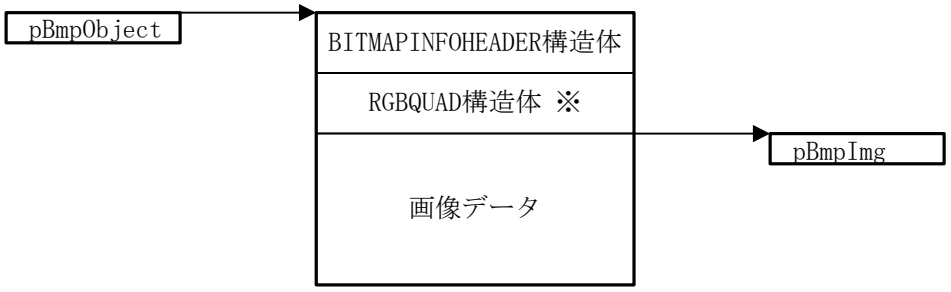
void *pBmpImg	!= NULL	ビットマップ画像データの先頭アドレス
	== NULL	エラー

❑ 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。



画像メモリビットマップオブジェクト

※ RGBQUAD構造体は、BW_BITMAP設定時のみ必要

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		
16	0x0010	ソース画面とデスティネーション画面の属性不一致エラー	画面属性を確認して下さい	

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 ウインドウ設定エラー(30 ～ 39)

エラー番号		エラー内容	対策	備考
30	0x001E	Src0ウインドウ設定エラー	ウインドウの設定値を確認して下さい	
31	0x001F	Src1ウインドウ設定エラー		
32	0x0020	Src2ウインドウ設定エラー		
33	0x0021	Dstウインドウ設定エラー		
34	0x0022	Sysウインドウ設定エラー		
35	0x0023	Extウインドウ設定エラー		
36	0x0024	Dst拡張ウインドウ設定エラー		

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	同時映像入力が設定されている	SelectCameraの設定を確認して下さい	
56	0x0038	線分列未検出エラー	処理対象画像を確認して下さい	

6 正規化相関エラー(80 ～ 89)

エラー番号		エラー内容	対策	備考
80	0x0050	正規化相関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	－	－	

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。またはカメラファイル（カメラデータ）のフォーマットを確認して下さい。	

8 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	―	―	

9 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	
141	0x008D	VPP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	
202	0x00CA	―	―	
203	0x00CB	画像処理スレッドシステムテーブル空きなし	実行条件を確認してください	
204	0x00CC	IMP2重オープンエラー	実行条件を確認してください	
205	0x00CD	IMPがオープンしていません	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。

10 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	
196610	0x00030002	コマンドタイムアウト	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	
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ＰＣＩ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１６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ＩＤのリソースが足りません	不要なデバイスＩＤをクローズして下さい	
131336	0x00020108	ドライバーエラーまたは通信エラー	ドライバーでエラーが発生しました	
131337	0x00020109	オンボードシステムでエラーが発生した	-	
131338	0x0002010A	システム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	システム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	システム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	ＮＶＰシステムが動作していない	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	-	

11 制御系エラー

エラー番号	エラー内容	備考
-1	内部データエラー 通信ドライバーエラー	
-2	OSシステムエラー	
-5	Linuxライブラリまたはドライバーエラー	
-9	ドライバ初期化エラー	
-10	サービスコールが組み込まれていない	
-11	属性が不正	
-17	パラメータエラー	
-18	不正ＩＤ番号	
-20	パラメータエラー（ボード番号エラー、ＩＭＰ番号範囲外）	
-25	コンテキストエラー	
-28	サービスコール不正使用	
-33	メモリ不足	
-34	ＩＤ番号不足	
-41	内部オブジェクトエラー	
-42	オブジェクトが存在しない	
-43	キューイングまたはネストのオーバーフロー	
-49	待ち状態強制解除	
-50	ポーリング失敗またはタイムアウト	
-51	待ちオブジェクトが削除された	
-52	待ちオブジェクトの状態が変化	
-57	ノンブロッキング受付	
-58	バッファオーバーフロー	
-100	ドライバインタフェースエラー(PC側のみ)	
-101	オンボードシステムエラー(PC側のみ)	
-102	画像処理コマンドエラー	
-103	ファイルオープンエラー	
-104	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	ファイルアクセスエラー	
-106	作業領域メモリ確保エラー	
-107	ダウンロード領域のメモリ確保エラー	
-108	ベリファイエラー	
-109	エントリシンボル無し	
-110	フラッシュメモリディスク容量不足	
-131	VINドライバーエラー	
-132	DUドライバーエラー	
-138	手続きエラー（StartIP多重実行、マルチプロセッサモード未設定、クローズされないままマルチモードからシングルモードへ設定された）	
-141	VPPドライバーエラー	
-203	画像処理スレッドシステムテーブル空きなし	
-204	OpenIMP未発行	
-205	２重オープン(OpenIMPの２重発行)	

12 Linuxエラー(errno.h)

エラー番号		エラー内容	備考
1	EPERM	操作が許可されていない	
2	ENOENT	そのようなファイルやディレクトリは無い	
3	ESRCH	そのようなプロセスは無い	
4	EINTR	関数呼び出しが割り込まれた	
5	EIO	入出力エラー	
6	ENXIO	そのようなデバイスやアドレスはない	
7	E2BIG	引き数リストが長過ぎる	
8	ENOEXEC	実行ファイル形式のエラー	
9	EBADF	ファイルディスクリプターが不正である	
10	ECHILD	子プロセスが無い	
11	EAGAIN	リソースが一時的に利用不可 ①システムのプロセステーブルがいっぱいなために fork関数が失敗した ②ユーザがそれ以上プロセスを作成することを許されていない ③メモリやスワップ領域の不足により呼び出しが失敗した	
12	ENOMEM	カーネルのメモリが足りない	
13	EACCES	許可がない	
14	EFAULT	アドレスが不正である ①アドレスが間違っている ② path が割り当てられたアドレス空間外を指している	
15	ENOTBLK	ブロックデバイスが必要である	
16	EBUSY	リソースが使用中である ①すでにマウントされているデバイスをマウントしようとした ②アクティブなファイルが存在するデバイスをアンマウントしようとした	
17	EEXIST	ファイルが存在する 既存のファイルが不適切な場面で指定されている	
18	EXDEV	不適切なリンク	
19	ENODEV	そのようなデバイスは無い	
20	ENOTDIR	ディレクトリではない	
21	EISDIR	ディレクトリである	
22	EINVAL	引数が無効である	
23	ENFILE	システム全体でオープンされているファイルが多過ぎる	
24	EMFILE	オープンしているファイルが多過ぎる	
25	ENOTTY	I/O 制御操作が適切でない	
26	ETXTBSY	テキストファイルが使用中である ①現在書込み用に開かれているピュア手続きプログラムを実行しようとした ②現在使用中のピュア手続きプログラムを書込み用に開いたり削除したりしようとした	
27	EFBIG	ファイルが大き過ぎる	
28	ENOSPC	デバイスに空き領域が無い	
29	ESPIPE	無効なシーク	
30	EROFS	読み出し専用のファイルシステムである	
31	EMLINK	リンクが多過ぎる	
32	EPIPE	パイプが壊れている	
33	EDOM	数学関数で引き数が領域外である	
34	ERANGE	結果が大き過ぎる	
35	EDEADLK	リソースのデッドロックを回避した	
36	ENAMETOOLONG	ファイル名が長過ぎる	
37	ENOLCK	利用できるロックが無い	
38	ENOSYS	関数が実装されていない	
39	ENOTEMPTY	ディレクトリが空ではない	
40	ELOOP	シンボリックリンクの回数が多過ぎる	
42	ENOMSG	要求された型のメッセージが存在しない	
43	EIDRM	識別子が削除された	
44	ECHRNG	チャンネル番号が範囲外	
45	EL2NSYNC	同期できていない（レベル 2）	
46	EL3HLT	停止（レベル 3）	
47	EL3RST	停止（レベル 3）	
48	ELNRNG	リンク番号が範囲外	
49	EUNATCH	プロトコルのドライバが付与（attach）されていない	
50	ENOCSI	利用可能なCSI構造なし	
51	EL2HLT	停止（レベル 2）	
52	EBADE	不正なやり取り（exchange）である	
53	EBADR	不正なリクエストディスクリプタ	

エラー番号		エラー内容	備考
54	EXFULL	変換テーブルが一杯である	
55	ENOANO	Exchangeフル	
56	EBADRQC	不正なリクエストコード	
57	EBADSLT	不正なスロット	
59	EBFONT	不適切なフォントファイル形式	
60	ENOSTR	ストリームではない	
61	ENODATA	ストリームの読み出しキューの先頭に読み出し可能なメッセージがない	
62	ETIME	時間が経過した	
63	ENOSR	指定されたストリームリソースが存在しない	
64	ENONET	マシンがネットワーク上にない	
65	ENOPKG	パッケージがインストールされていない	
66	EREMOTE	オブジェクトがリモートにある	
67	ENOLINK	リンクが切れている	
68	EADV	宣言済みのリソースの宣言、宣言されたリソースが存在する RFS の停止、または宣言されたリソースの強制アンマウントを行おうとした	
69	ESRMNT	①リソースがまだリモートノードにマウントされた状態で RFS を停止しようとした ②現時点でリソースがマウントされているリモートノードを含まないクライアントリストによりリソースが再宣言された	
70	ECOMM	送信時に通信エラーが発生した	
71	EPROTO	プロトコルエラー	
72	EMULTIHOP	マルチホップ（multihop）を試みた 直接アクセスが不可能なリモートリソースにアクセスしようとした	
73	EDOTDOT	プロセスがマウントポイントから戻されたことを、サーバからクライアントに伝える方法	
74	EBADMSG	メッセージが不正である	
75	EOVERFLOW	指定されたデータ型に格納するには値が大き過ぎる	
76	ENOTUNIQ	名前がネットワークで一意ではない	
77	EBADFD	ファイルディスクリプターが不正な状態である ①ファイル記述子で指定したファイルが開かれていない ②読込み要求が書込み専用に開かれたファイルに対して行われている	
78	EREMCHG	リモートアドレスが変わった	
79	ELIBACC	必要な共有ライブラリにアクセスできなかった	
80	ELIBBAD	壊れた共有ライブラリにアクセスしようとした	
81	ELIBSCN	a.out のライブラリセクションが壊れている	
82	ELIBMAX	リンクしようとした共有ライブラリが多過ぎる	
83	ELIBEXEC	共有ライブラリを直接実行できなかった	
84	EILSEQ	不正なバイト列	
85	ERESTART	システムコールが中断され再スタートが必要である	
86	ESTRPIPE	ストリームパイプエラー	
87	EUSERS	ユーザー数が多過ぎる	
88	ENOTSOCK	ソケットではない	
89	EDESTADDRREQ	宛先アドレスが必要である	
90	EMSGSIZE	メッセージが長過ぎる	
91	EPROTOTYPE	ソケットに指定できないプロトコルタイプである	
92	ENOPROTOOPT	指定されたプロトコルが利用できない	
93	EPROTONOSUPPORT	プロトコルがサポートされていない	
94	ESOCKTNOSUPPORT	サポートされていないソケット種別である	
95	EOPNOTSUPP	ソケットでサポートしていない操作である	
96	EPFNOSUPPORT	サポートされていないプロトコルファミリーである	
97	EAFNOSUPPORT	アドレスファミリーがサポートされていない	
98	EADDRINUSE	アドレスがすでに使用されている	
99	EADDRNOTAVAIL	アドレスが使用できない	
100	ENETDOWN	ネットワークが不通である	
101	ENETUNREACH	ネットワークが到達不能である	
102	ENETRESET	接続がネットワーク側から中止された	
103	ECONNABORTED	接続が中止された	
104	ECONNRESET	接続がリセットされた	
105	ENOBUFS	使用可能なバッファー空間がない	
106	EISCONN	ソケットが接続されている	
107	ENOTCONN	ソケットが接続されていない	
108	ESHUTDOWN	通信相手がシャットダウンされて送信できない	

エラー番号		エラー内容	備考
109	ETOOMANYREFS	参照が多過ぎてスプライスできない	
110	ETIMEDOUT	操作がタイムアウトした ①タイムアウト時間が経過しても接続先が適切な応答を返さなかったため connect() または send() 要求が失敗した。 ②ファイルがソフトオプションでマウントされた NFS ファイルシステム上にあるため write()または fsync() 要求が失敗した	
111	ECONNREFUSED	接続が拒否された	
112	EHOSTDOWN	ホストがダウンしている	
113	EHOSTUNREACH	ホストに到達不能である	
114	EALREADY	接続が既に処理中である	
115	EINPROGRESS	操作が実行中である	
116	ESTALE	ファイルハンドルが古い状態になっている	
117	EUCLEAN	Structureのcleaningが必要	
118	ENOTNAM	XENIX型ファイルではない	
119	ENAVAIL	XENIXセマフォは使用できない	
120	EISNAM	名前付きのファイルである	
121	EREMOTEIO	リモート I/O エラー	
122	EDQUOT	ディスククォータ (quota) を超過した	
123	ENOMEDIUM	メディアが見つからない	
124	EMEDIUMTYPE	間違ったメディア種別である	
125	ECANCELED	操作がキャンセルされた	
126	ENOKEY	要求された鍵が利用できない	
127	EKEYEXPIRED	鍵が期限切れとなった	
128	EKEYREVOKED	鍵が無効となった	
129	EKEYREJECTED	鍵がサーバにより拒否された	
130	EOWNERDEAD	Lockによってプロセスはdiedとなった	
131	ENOTRECOVERABLE	Lockは回復不可能だった	
132	ERFKILL	RFをkillすることで動作できなかった	
133	EHWPOISON	メモリページにハードウェアエラーがある	
512	ERESTARTSYS	システムコールがシグナルによって割り込まれ、何も読み出せなかった	
513	ERESTARTNOINTR	システムコールはシグナルによって中断され、再起動される	
514	ERESTARTNOHAND	ハンドラがない場合は再起動する	
515	ENOIOCTLCMD	ioctlコマンドなし	
516	ERESTART_RESTARTBLOCK	sys_restart_syscallを呼び出して再起動した	
517	EPROBE_DEFER	ドライバがプローブの再試行を要求した	
518	EOPENSTALE	open found a stale dentry	
521	EBADHANDLE	不正なNFSファイルハンドル	
522	ENOTSYNC	同期の不一致を更新した	
523	EBADCOOKIE	Cookieは古くなっている	
524	ENOTSUPP	操作はサポートされていない	
525	ETOOSMALL	バッファまたは要求が小さ過ぎる	
526	ESERVERFAULT	翻訳できないエラーが発生した	
527	EBADTYPE	サーバでサポートされていないタイプ	
528	EJUKEBOX	リクエストは開始されたがタイムアウト前に完了しなかった	
529	EIOCBQUEUED	キューに入れられたiocbが完了イベントを取得した	

付録B カメラデーター一覧

記載のない変数は0です。
VIDEO_FRM_TBL(pluslng.Vsize,pluslng.HSize),VIDEO_CAM_TBL(Hcycle,OptInfo,Reserve),
VIDEO_CLINK_MODE_TBL(reserve[0,4-12]), VIDEO_VPP_DATA_TBL(Data[5-19])

1. SONY_XCL_SG510

テーブル名	要素	値	備考
VIDEO_CAM_FRM_INFO	inf1	3	
	inf2	0	
VIDEO_FRM_TBL	maxfs.Vsize	2048	有効画素（最大フレームサイズ）※1
	maxfs.Hsize	2448	
VIDEO_CAM_TBL	name	"SONY_XCL_SG510"	
	CamAtr	0x07	CAMATR_CLINK
VIDEO_CLINK_MODE_TBL	camtype	0	PoCL/CameraLink
	scantype	0	Area Scan
	tapform	0x23	TAPフォーマット：2TAP-2X1Y-PXL ※2
	pxlconv	0	画素フォーマット：8bpp ※2
	seldval	0	DVAL
	vinmode	1	VINモード：dual
	reserve[1]	0	Bayerフォーマット：RG
	reserve[2]	0	VIN拡張モード：垂直拡張
	reserve[3]	0	周波数：85MHz以下 ※2
VIDEO_VPP_DATA_TBL	Data[0]	40000	内部トリガ間隔[uS]
	Data[1]	1	トリガ出力先：CC1 ※3
	Data[2]	2	DIスルー出力先：CC2 ※3
	Data[3]	0x1	CC極性：CC1負極,CC2～CC4正極 ※3
	Data[4]	0x00000000	有効画素オフセット

※1 カメラデータ設定値 ＝ カメラ側設定 となる様にカメラを設定してください。(WIDTH/HEIGHT)
※2 TAPフォーマット,画素フォーマット、周波数に合わせてカメラを設定してください。
(CAMERALINK-TAP、PIXEL-DEPTH、BASE-CLOCK)
※3 使用するトリガモードに合わせてカメラ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CCに出力されます。

2. NED_RMSL4K100CL

テーブル名	要素	値	備考
VIDEO_CAM_FRM_INFO	inf1	3	
	inf2	0	
VIDEO_FRM_TBL	maxfs.Vsize	1	有効画素（最大フレームサイズ）※1
	maxfs.Hsize	4096	
VIDEO_CAM_TBL	name	"NED_RMSL4K100CL"	
	CamAtr	0x07	CAMATR_CLINK
VIDEO_CLINK_MODE_TBL	camtype	0	PoCL/CameraLink
	scantype	1	Line Scan
	tapform	0x83	TAPフォーマット：8TAP-8X1Y-PXL ※2
	pxlform	0	画素フォーマット：8bpp ※2
	seldval	1	LVAL
	vinmode	3	VINモード：quad
	reserve[1]	0	Bayerフォーマット：RG
	reserve[2]	0	VIN拡張モード：垂直拡張
	reserve[3]	1	周波数：50MHz以下
VIDEO_VPP_DATA_TBL	Data[0]	60	内部トリガ間隔[uS]
	Data[1]	1	トリガ出力先：CC1 ※3
	Data[2]	2	DIスルー出力先：CC2 ※3
	Data[3]	0x0	CC極性：CC1正極,CC2～CC4負極 ※3
	Data[4]	0x00000000	有効画素オフセット

※1 カメラデータ設定値 ＝ カメラ側設定 となる様にカメラを設定してください。
※2 TAPフォーマット,画素フォーマット,周波数に合わせてカメラを設定してください。(カメラリンク出力Tapの変更、出力BIT数の変更、ピクセル周波数の変更)
※3 使用するトリガモードに合わせてカメラを設定してください。(カメラ動作モード、露光モード設定)
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CCに出力されます。

3. NED_RMSL8K76CL

テーブル名	要素	値	備考
VIDEO_CAM_FRM_INFO	inf1	3	
	inf2	0	
VIDEO_FRM_TBL	maxfs.Vsize	1	有効画素（最大フレームサイズ）※1
	maxfs.Hsize	8192	
VIDEO_CAM_TBL	name	"NED_RMSL8K76CL"	
	CamAtr	0x07	CAMATR_CLINK
VIDEO_CLINK_MODE_TBL	camtype	0	PoCL/CameraLink
	scantype	1	Line Scan
	tapform	0x88	TAPフォーマット：8TAP-8X1Y-BLK3 ※2
	pxlform	0	画素フォーマット：8bpp ※2
	seldval	1	LVAL
	vinmode	3	VINモード：quad
	reserve[1]	0	Bayerフォーマット：RG
	reserve[2]	1	VIN拡張モード：水平拡張
	reserve[3]	1	周波数：50MHz以下
VIDEO_VPP_DATA_TBL	Data[0]	60	内部トリガ間隔[uS]
	Data[1]	1	トリガ出力先：CC1 ※3
	Data[2]	2	DIスルー出力先：CC2 ※3
	Data[3]	0x0	CC極性：CC1正極,CC2～CC4負極 ※3
	Data[4]	0x00000000	有効画素オフセット

※1 カメラデータ設定値 ＝ カメラ側設定 となる様にカメラを設定してください。
※2 TAPフォーマット,画素フォーマット,周波数に合わせてカメラを設定してください。(カメラリンク出力Tapの変更、出力BIT数の変更、ピクセル周波数の変更)
※3 使用するトリガモードに合わせてカメラを設定してください。(カメラ動作モード、露光モード設定)
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CCに出力されます。

4. TAKEX_TL_4096ACL

テーブル名	要素	値	備考
VIDEO_CAM_FRM_INFO	inf1	3	
	inf2	0	
VIDEO_FRM_TBL	maxfs.Vsize	1	有効画素（最大フレームサイズ）※1
	maxfs.Hsize	4096	
VIDEO_CAM_TBL	name	"TAKEX_TL_4096ACL"	
	CamAtr	0x07	CAMATR_CLINK
VIDEO_CLINK_MODE_TBL	camtype	0	PoCL/CameraLink
	scantype	1	Line Scan
	tapform	0x83	TAPフォーマット：8TAP-8X1Y-PXL ※2
	pxlform	0	画素フォーマット：8bpp ※2
	seldval	1	LVAL
	vinmode	3	VINモード：quad
	reserve[1]	0	Bayerフォーマット：RG
	reserve[2]	0	VIN拡張モード：垂直拡張
	reserve[3]	1	周波数：50MHz以下
VIDEO_VPP_DATA_TBL	Data[0]	60	内部トリガ間隔[uS]
	Data[1]	1	トリガ出力先：CC1 ※3
	Data[2]	2	DIスルー出力先：CC2 ※3
	Data[3]	0x0	CC極性：CC1正極,CC2～CC4負極 ※3
	Data[4]	0x00000000	有効画素オフセット

※1 カメラデータ設定値 ＝ カメラ側設定 となる様にカメラを設定してください。
※2 TAPフォーマット,画素フォーマット,周波数に合わせてカメラを設定してください。(カメラリンク出力Tapの変更、出力BIT数の変更、ピクセル周波数の変更)
※3 使用するトリガモードに合わせてカメラを設定してください。(カメラ動作モード、露光モード設定)
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CCに出力されます。

5. TAKEX_TL_8K35ACL

テーブル名	要素	値	備考
VIDEO_CAM_FRM_INFO	inf1	3	
	inf2	0	
VIDEO_FRM_TBL	maxfs.Vsize	1	有効画素（最大フレームサイズ）※1
	maxfs.Hsize	8192	
VIDEO_CAM_TBL	name	"TAKEX_TL_8K35ACL"	
	CamAtr	0x07	CAMATR_CLINK
VIDEO_CLINK_MODE_TBL	camtype	0	PoCL/CameraLink
	scantype	1	Line Scan
	tapform	0x84	TAPフォーマット：8TAP-8X1Y-BLK2※2
	pxlform	0	画素フォーマット：8bpp ※2
	seldval	1	LVAL
	vinmode	3	VINモード：quad
	reserve[1]	0	Bayerフォーマット：RG
	reserve[2]	1	VIN拡張モード：水平拡張
	reserve[3]	1	周波数：50MHz以下 ※2
VIDEO_VPP_DATA_TBL	Data[0]	60	内部トリガ間隔[uS]
	Data[1]	1	トリガ出力先：CC1 ※3
	Data[2]	2	DIスルー出力先：CC2 ※3
	Data[3]	0x0	CC極性：CC1正極,CC2～CC4負極 ※3
	Data[4]	0x00000000	有効画素オフセット

※1 カメラデータ設定値 ＝ カメラ側設定 となる様にカメラを設定してください。
※2 TAPフォーマット,画素フォーマット, 周波数に合わせてカメラを設定してください。(カメラリンク出力設定)
※3 使用するトリガモードに合わせてカメラを設定してください。(同期設定、露光モード設定)
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CCに出力されます。

6. TAKEX_TL_16KACL

テーブル名	要素	値	備考
VIDEO_CAM_FRM_INFO	inf1	3	
	inf2	0	
VIDEO_FRM_TBL	maxfs.Vsize	1	有効画素（最大フレームサイズ）※1
	maxfs.Hsize	16384	
VIDEO_CAM_TBL	name	"TAKEX_TL_16KACL"	
	CamAtr	0x07	CAMATR_CLINK
VIDEO_CLINK_MODE_TBL	camtype	0	PoCL/CameraLink
	scantype	1	Line Scan
	tapform	0x84	TAPフォーマット：8TAP-8X1Y-BLK2※2
	pxlform	0	画素フォーマット：8bpp ※2
	seldval	1	LVAL
	vinmode	3	VINモード：quad
	reserve[1]	0	Bayerフォーマット：RG
	reserve[2]	1	VIN拡張モード：水平拡張
	reserve[3]	1	周波数：50MHz以下 ※2
VIDEO_VPP_DATA_TBL	Data[0]	60	内部トリガ間隔[uS]
	Data[1]	1	トリガ出力先：CC1 ※3
	Data[2]	2	DIスルー出力先：CC2 ※3
	Data[3]	0x0	CC極性：CC1正極,CC2～CC4負極 ※3
	Data[4]	0x00000000	有効画素オフセット

※1 カメラデータ設定値 ＝ カメラ側設定 となる様にカメラを設定してください。
※2 TAPフォーマット,画素フォーマット,周波数に合わせてカメラを設定してください。(カメラリンク出力設定)
※3 使用するトリガモードに合わせてカメラを設定してください。(同期設定、露光モード設定)
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CCに出力されます。

画像処理ユニット NVP-Ax430シリーズ　ソフトウェア開発キット

NVP-Ax430SDK

コマンドリファレンス（第３版）

(C) マクセルフロンティア株式会社

開発元

マクセルフロンティア株式会社

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 www.frontier.maxell.co.jp
mail : vp-support@maxell.co.jp