お客様各位

カタログ等資料中の旧社名の扱いについて

拝啓　時下ますますご清栄のこととお喜び申し上げます。平素は格別のご高配を賜り厚く御礼申し上げます。

さて、２０１７年５月１日を以ってルネサス セミコンダクタ パッケージ&テスト ソリューションズ株式会社の半導体製造装置をはじめとする各種産業用制御ボードの受託開発・製造および画像認識システム開発・製造・販売事業を日立マクセル株式会社へ譲渡したことにより、当該事業は日立マクセル株式会社の子会社として新設されるマクセルシステムテック株式会社に承継されております。

従いまして、ドキュメント等資料中には、旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

敬具

２０１７年５月１日

マクセルシステムテック株式会社

【発行】マクセルシステムテック（http://www.systemtech.maxell.co.jp/）

【お問い合わせ先】denki-support@maxell.co.jp

maxell

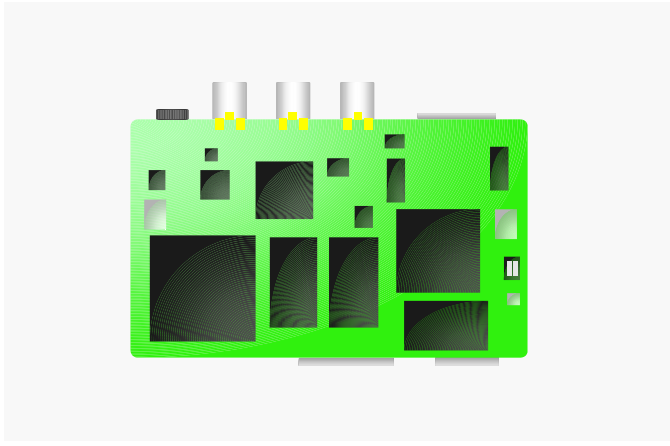
マクセルシステムテック株式会社

超小型
画像処理ボード SVP-330

Network Vision Processor

Software Development Kit

ユーザーズ ガイド



HIRECON

はじめに

このたびは、「超小型画像処理ボード：ＳＶＰ－３３０」および「画像処理ソフトウェア開発キット：ＳＶＰ－３３０ＳＤＫ」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、ＳＶＰ－３３０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、「画像処理ソフトウェア開発キット：ＳＶＰ－３３０ＳＤＫ」について記載しております。ハードウェアについては、「超小型画像処理ボード：ＳＶＰ－３３０ ハードウェアマニュアル」をご参照ください。

⚠ ご注意

●システムの構築やプログラム作成などの操作を行う前に、本マニュアルの記載内容をよく読み、書かれている指示や注意を十分理解して下さい。誤った操作によりシステムの故障が発生することがあります。

●本マニュアルの記載内容について理解できない内容、疑問点または不明点がございましたら、弊社営業窓口までお知らせ下さい。
尚、下記アドレスにてｅ－ｍａｉｌによるお問い合わせも受け付けておりますのでご利用ください。
e-mail:vp.support@kitasemi.renesas.com

●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。

目次

第5章　画像処理コマンドの基礎

第6章　画像処理コマンド

第7章　インテリジェントモジュール

第8章　割込起動モジュール

第9章　ビデオレート処理

第10章　非同期映像入力

第11章　エラー発生時の対策と手順

第12章　マルチポート映像入力

第13章　YUVカラー処理コマンド

第14章　RGBカラー処理コマンド

第15章 フラッシュメモリの活用

第16章 IP5000コマンドの使用方法

第17章 イーサネットでの通信

第18章 ユニット内アプリケーションの開発

付録

第１章

画像処理ボード(SVP−330)の概要

１．１ ハードウェア構成

ＳＶＰ－３３０画像処理ボードでは、画像処理プロセッサによる画像処理と浮動小数点演算コプロセッサを持つＲＩＳＣタイプのＣＰＵ（ＳＨ４）により、柔軟性の高い画像処理を高速に処理可能です。

画像処理プロセッサは、画像間演算、フィルタリング、正規化相関、ヒストグラム、ラベリング等豊富な画像処理機能を備えており、基本機能は７．５ナノ秒／画素で処理します。

画像メモリは、８ビットで５１２×５１２画素のサイズの映像を６０面格納することができます。

ＳＶＰ－３３０はＮＴＳＣモノクロカメラとＣＶＢＳカラーカメラを使用することができます。

イーサネットによるネットワーク接続でＳＶＰ－３３０複数ユニットをコントロール可能です。画像処理の並列化を図り、システムのパフォーマンスの向上が期待できます。

1.1.1 SVP−330 ハードウェア構成

ＳＶＰ－３３０は、ＮＴＳＣモノクロ／カラーカメラ対応の画像処理ユニットです。
図１－１に示すように

・画像処理プロセッサ
・画像メモリ／システムメモリ（UMA方式）
・ＲＩＳＣタイプＣＰＵ（ＳＨ４）
・ＣＰＵフラッシュメモリ
・映像入力部（独立２チャネル・カラー対応）
・映像出力部
・イーサネット装備（１００ＢＡＳＥ-ＴＸ／１０ＢＡＳＥ-Ｔ）
・フォトアイソレーションＤＩ/DO装備 （入力４ＣＨ、出力４ＣＨ）
・シリアル（ＲＳ２３２Ｃ）装備

で構成されています。

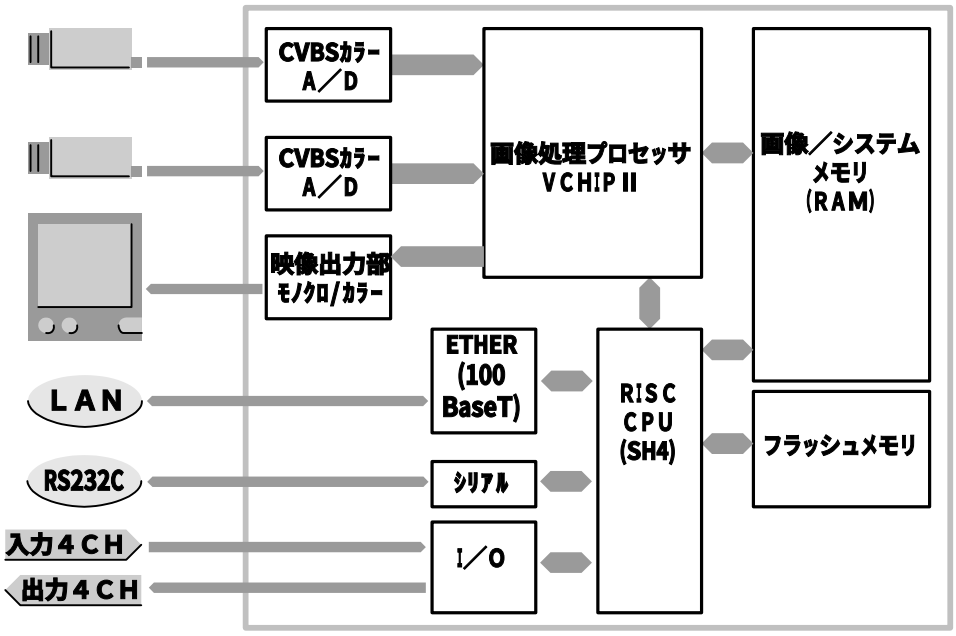


図１－１　ＳＶＰ－３３０ハードウェア構成

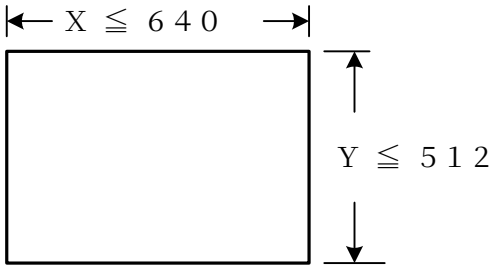
1．2 画像処理の制限事項

ＳＶＰ－３３０では、画像処理プロセッサの制限により画像処理を行う際、以下の制限があります。

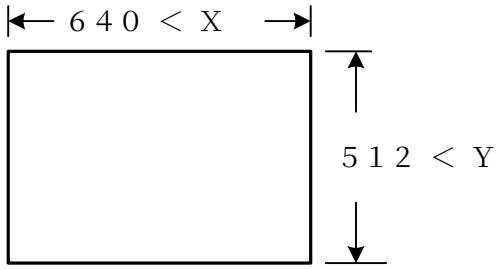
1.2.1 画面サイズの制限事項

画面の大きさが６４０(X)×５１２(Y)より大きいサイズの画面で画像処理を行うことはできません。

処理可能な画面サイズ

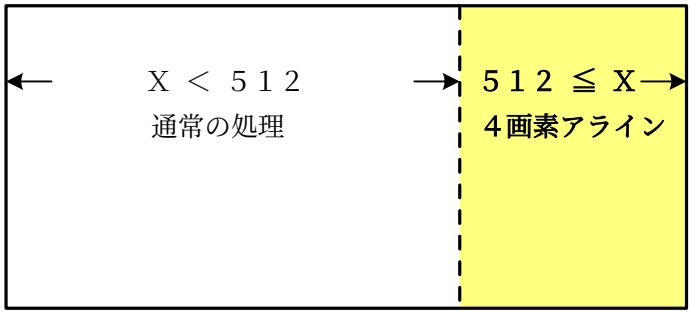


処理できない画面サイズ



1.2.2 ウィンドウサイズの制限事項

処理ウィンドウのX方向の大きさが５１２以上が指定された場合、X方向のサイズが４画素アラインサイズに切り捨てられて処理されます。対象のコマンドは下表のデスティネーション画面に処理結果が出力されるコマンドです。



X方向４バイトアラインコマンド

	対象コマンド	備考
1	画像クリアコマンド	
2	画像転送・アフィン変換コマンド	IP_Zoom(),IP_ZoomExt(),IP_Rotate()コマンド以外
3	２値化コマンド	
4	画素変換コマンド	
5	画像間算術演算コマンド	
6	画像間論理演算コマンド	
7	２値画像形状変換コマンド	
8	コンボリューションコマンド	
9	ミニ／マックスフィルタコマンド	
10	ランクフィルタコマンド	
11	ラベリングコマンド	ラベル毎特徴量抽出以外
12	拡張コンボリューションコマンド	
13	拡張画像処理コマンド	
14	擬似カラーコマンド	

1.2.3 画像処理機能の制限事項

ＳＶＰ－３３０では、ハードウェアの制約により以下のコマンドは使用できません。

使用できないコマンド

	対象コマンド	備考
1	２値パイプラインフィルタ	
2	２値マッチングフィルタ	
3	ＲＧＢＬＵＴ変換コマンド	

第２章

SVP-330SDKの概要

ＳＶＰ－３３０ＳＤＫ（SVP-330 Software Development Kit）は、画像処理ユニットＳＶＰ－３３０のソフトウェア開発キットです。弊社で提供するＳＶＰ－３３０のアプリケーションソフトウェア開発に必要なライブラリ、ツール、ドキュメントが全て含まれています。

２．１ ＳＶＰ－３３０ＳＤＫの特徴

・Ｗｉｎｄｏｗｓ、Ｖｉｓｕａｌ Ｃ／Ｃ＋＋環境でのアプリケーションの構築が可能です。

・ＰＣドライバはＤＬＬで供給されるため、ＤＬＬをサポートしている他の言語からも使用可能です。

・オンボードＣＰＵでのインテリジェント処理モジュールを容易に構築可能です。

２．２ 動作環境及び開発環境

ＳＶＰ－３３０ＳＤＫを使用したアプリケーションプログラムの開発には、次の環境が必要です。

2．2．1　PCドライバを使用したアプリケーション開発

ＰＣドライバを使用したアプリケーション開発では以下の環境が必要です。

表２－１　ＰＣドライバを使用したアプリケーションの開発環境

項目		仕様	備考
ハードウェア	DOS/Vマシン	シングルＣＰＵのみ対応 メモリ：６４Mバイト以上推奨	
	SVP-330	最大 １６台（認識可能なボード番号）	
ＯＳ		Microsoft Windows 2000 Service Pack 1 以上	
		Microsoft Windows XP Service Pack 1 以上	
コンパイラ		Microsoft Visual C/C++ 6.0(Service Pack 3)	
インストールディスク容量		５０Mバイト以上推奨	

2．2．2　オンボードCPU上のアプリケーション開発

オンボードＣＰＵで動作するインテリジェントモジュール、割込み起動モジュール、ユニット内アプリケーションの開発は、上記のＰＣドライバを使用したアプリケーション開発で用意する環境の他に表２－２の環境が必要です。

表２－２　オンボードＣＰＵインテリジェントモジュールの開発環境

項目	内容	備考
コンパイラ	SuperH RISC engineファミリ C/C++コンパイラパッケージ V.8.00 Release 04 以降	日立ＳＫ

2．3 パッケージ内容

ＳＶＰ－３３０ＳＤＫをインストールするとハードディスクに図２－１に示すディレクトリが作成されます。表２－３にインストールされるファイルの一覧を示します。

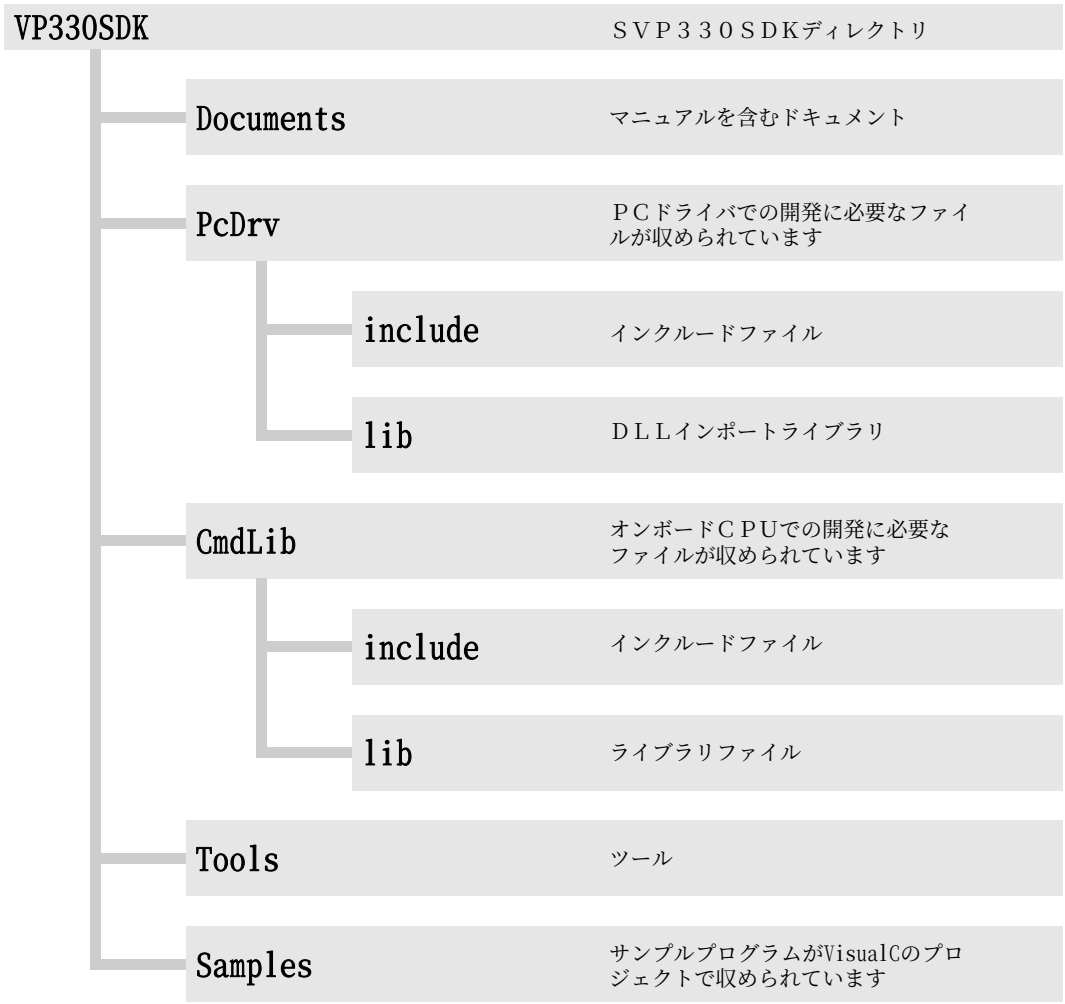


図２－１　ディレクトリ構成

表２－３　インストールファイル一覧（１／３）

ディレクトリ	ファイル名	内容
Documents	使用手引.pdf	ＳＶＰ－３３０ 使用の手引き
	簡単ｾｯﾄｱｯﾌﾟ.pdf	ＳＶＰ－３３０ セットアップ手順
	環境設定ガイド.pdf	ＳＶＰ－３３０ 環境設定ガイド
	ユーザーズガイド.pdf	ＳＶＰ－３３０ＳＤＫ ユーザーズガイド
	コマンドリファレンス.pdf	ＳＶＰ－３３０ＳＤＫ コマンドリファレンス
	ＳＨＣ設定ガイド.pdf	ＳＨＣ Ｃｏｍｐｉｌｅｒ設定ガイド
	VPMaster取説.pdf	VPMaster操作説明書
	SHELLリファレンス.pdf	ＳＨＥＬＬ コマンドリファレンス
	vNavi操作説明書.pdf	ｖＮａｖｉ 操作説明書
	SVP330ﾊｰﾄﾞｳｪｱﾏﾆｭｱﾙ.pdf	ＳＶＰ－３３０ ハードウェアマニュアル
PcDrv\include	vpxdef.h	ＰＣドライバ用 define,enum定義
	vpxsys.h	ＰＣドライバ用 構造体定義
	vpxfnc.h	ＰＣドライバ用 プロトタイプ定義
	vpxerr.h	ＰＣドライバ用 エラー定義（VP810互換ダミー）
	vpxver.h	ＰＣドライバ用 バージョン定義
	vpxcnv.h	ＰＣドライバ用 オンボード－ＰＣコンバート定義
	vpxadm.h	ＰＣドライバ用 define,構造体定義
	vpxmacro.h	ＰＣドライバ用 enum定義
	ipxdef.h	ＰＣドライバ用 define,enum定義
	ipxsys.h	ＰＣドライバ用 構造体定義
	ipxprot.h	ＰＣドライバ用 プロトタイプ定義
	ipxmacro.h	ＰＣドライバ用 マクロ定義
	ipxfunc.h	ＰＣドライバ用 プロトタイプ定義２
	ipxmac.h	ＰＣドライバ用 define,enum定義
	vpxvin.h	ＰＣドライバ用 ビデオ登録コマンド定義
PcDrv\lib	vp900cmd.lib	ＰＣドライバ用 ＤＬＬインポートライブラリ(SVP-330コマンド)
	vp900std.lib	ＰＣドライバ用 ＤＬＬインポートライブラリ(IP5000互換コマンド)
CmdLib\include	vpxdef.h	オンボードＣＰＵ用 define,enum定義
	vpxsys.h	オンボードＣＰＵ用 構造体定義
	vpxfnc.h	オンボードＣＰＵ用 プロトタイプ定義
	vpxerr.h	オンボードＣＰＵ用 エラー定義（VP810互換ダミー）
	vpxver.h	オンボードＣＰＵ用 バージョン定義
	vpxcnv.h	ＰＣドライバ → オンボードＣＰＵ 関数変換マクロ
	vpxadm.h	オンボードＣＰＵ用 define,構造体定義
	vpxsock.h	オンボードＣＰＵ用 define,構造体,プロトタイプ定義
	ipxdef.h	オンボードＣＰＵ用 define,enum定義
	ipxsys.h	オンボードＣＰＵ用 構造体定義
	ipxprot.h	オンボードＣＰＵ用 プロトタイプ定義
	ipxmacro.h	オンボードＣＰＵ用 マクロ定義
	vpxvin.h	オンボードＣＰＵ用 ビデオ登録コマンド定義
	cnvcmd.h	画像処理関数置換マクロ

表２－３　インストールファイル一覧（２／３）

ディレクトリ	ファイル名	内容
CmdLib\ include	windows.h	Windows プログラムからの置換用ダミー
	malloc.h	Windows プログラムからの置換用ダミー
	conio.h	Windows プログラムからの置換用ダミー
CmdLib\lib	cmdlib.lib	オンボードＣＰＵ用 コマンドライブラリ
	stdlib.lib	オンボードＣＰＵ用 STDLIB,STDIOライブラリ
Tools	VPSetup.exe	ＳＶＰ－３３０のセットアップアプリケーション
	VPSetreg.exe	ＰＣ環境の設定アプリケーション
	VPMaster.exe	評価ツール
	vNavi.exe	ファイル管理ツール
Samples\console	command	画像処理コマンドのコーディングサンプル Visual C/C++ コンソールアプリケーション用 プロジェクト
	module	インテリジェントモジュール（ＰＣ側）サンプル Visual C/C++ コンソールアプリケーション用 プロジェクト ボードＣＰＵ側のプログラム Samples\Hew8\intelimod をコントロールします。
	piotask	割込起動モジュール（ＰＣ側）サンプル Visual C/C++ コンソールアプリケーション用 プロジェクト ボードＣＰＵ側のプログラム Samples\Hew8\piotask Samples\Hew8\piotask1 Samples\Hew8\piotask2 Samples\Hew8\piotask3 をコントロールします。
	Download	ユニット内アプリケーションのＰＣからのダウンロードと 実行サンプル ボードＣＰＵ側のプログラム Samples\Hew8\unitapl Samples\Hew8\unittsk Samples\Hew8\unitpio をコントロールします。
Samples\mfc	degicam	Windowsビットマップファイルのセーブ／ロードサンプル Visual C/C++ Windowsアプリケーション用 プロジェクト
	VPCapture	様々なカメラから映像を入力しWindows画面への表示するサンプル Visual C/C++ Windowsアプリケーション用 プロジェクト

表２－３　インストールファイル一覧（３／３）

ディレクトリ	ファイル名	内容
Samples\Hew8 ※１(SHC V8)	intelimod	インテリジェントモジュール（ボードＣＰＵ側）サンプル ＳＨＣコンパイラ Ver6.0用 プロジェクト ＰＣ側のプログラム Samples\console\module からコントロールされます。
	piotask	ＰＩＯ割込起動モジュール（ボードＣＰＵ側）サンプル１ ＳＨＣコンパイラ Ver6.0用 プロジェクト ＰＣ側のプログラム Samples\consol\piotask からコントロールされます。
	piotask1	ＰＩＯ割込起動モジュール（ボードＣＰＵ側）サンプル２ ＳＨＣコンパイラ Ver6.0用 プロジェクト ＰＣ側のプログラム Samples\console\piotask からコントロールされます。
	piotask2	ＰＩＯ割込起動モジュール（ボードＣＰＵ側）サンプル３ ＳＨＣコンパイラ Ver6.0用 プロジェクト ＰＣ側のプログラム Samples\console\piotask からコントロールされます。
	piotask3	ＰＩＯ割込起動モジュール（ボードＣＰＵ側）サンプル４ ＳＨＣコンパイラ Ver6.0用 プロジェクト ＰＣ側のプログラム Samples\console\piotask からコントロールされます。
	unitapl	ユニット内アプリケーション　サンプル アプリケーションタスク内での実行。 ＰＣ側のプログラム Samples\console\Download からコントロールされます。
	unittsk	ユニット内アプリケーション　サンプル 別タスクでのアプリケーションの実行。 ＰＣ側のプログラム Samples\console\Download からコントロールされます。
	unitpio	ユニット内アプリケーション　サンプル 別タスクでのPIO割込みモジュールの実行。 ＰＣ側のプログラム Samples\console\Download からコントロールされます。

※１　「Samples\Hew8」フォルダには、SHC V8用のプロジェクトが納められています。

2．4 アプリケーションの開発

オンボードＣＰＵで動作するインテリジェントモジュール、割込み起動モジュール、ユニット内アプリケーションの開発は、ＪＴＡＧ－ＩＣＥなどの開発装置などが必要になり大掛かりなものになってしまいます。そこで弊社では、Ｃ言語のソースレベルで互換性を持った画像処理関数をＰＣホストとオンボードＣＰＵに開発することでＰＣホストのＣ言語のデバッガでインテリジェントモジュール部分をデバッグすることを可能にしました。通常、オンボードＣＰＵで動作するモジュール部分のデバッグは、画像処理アルゴリズムの検証程度の簡単なものなので、それで開発したソースコードをそのままオンボードＣＰＵのＣ言語コンパイラでコンパイルすることでアプリケーションが完成します。

2．4．1 インテリジェントモジュールの開発

インテリジェントモジュールの開発フローを図２－２に示します。ＰＣドライバの画像処理コマンドでアプリケーションを作成し、ＶｉｓｕａｌＣ／Ｃ＋＋でコンパイル、デバッグをＰＣ上で行い、アルゴリズムの検証等を行います。それらの設計が終了後、ＳＨＣコンパイラでコンパイルし、実行ファイルを作成します。また、インテリジェントモジュールはＰＣとのインタフェースが必要ですので、ＰＣからのパラメータの受け渡しや起動の部分をＰＣ側のアプリケーション内に作成します。

なお、インテリジェントモジュールの詳細は第７章を参照して下さい。

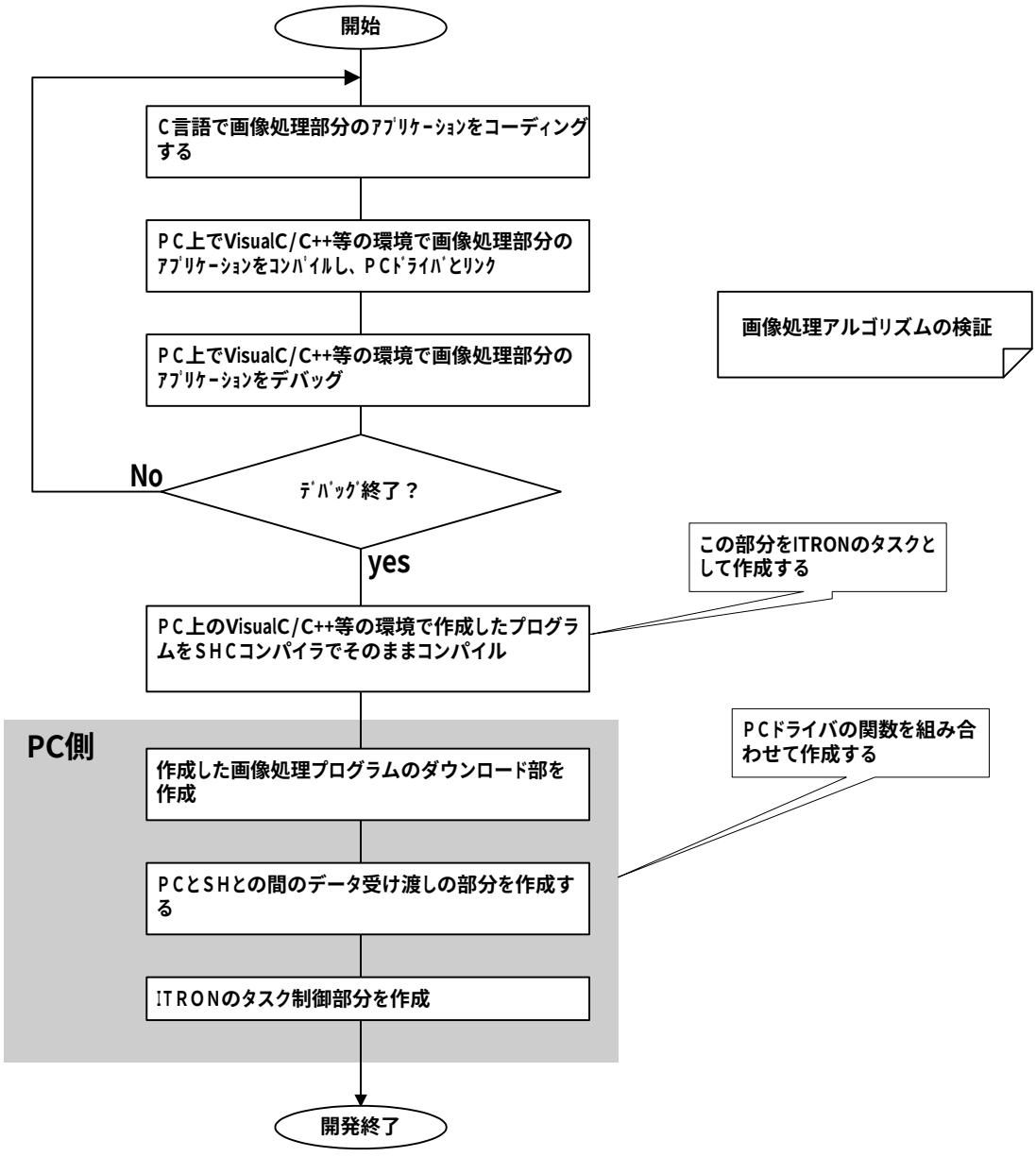


図２－２　インテリジェントモジュール開発フロー

2．4．2　割込起動モジュールの開発

割込起動モジュールの開発フローを図２－３に示します。ＰＣドライバの画像処理コマンドでアプリケーションを作成し、ＶｉｓｕａｌＣ／Ｃ＋＋でコンパイル、デバッグをＰＣ上で行い、アルゴリズムの検証等を行います。それらの設計が終了後、ＳＨＣコンパイラでコンパイルし、実行ファイルを作成します。また、割込起動モジュールでＰＣとのインタフェースが必要な場合、ＰＣからのパラメータの受け渡しや起動の部分をＰＣ側のアプリケーション内に作成します。

なお、割込起動モジュールの詳細は第８章を参照して下さい。

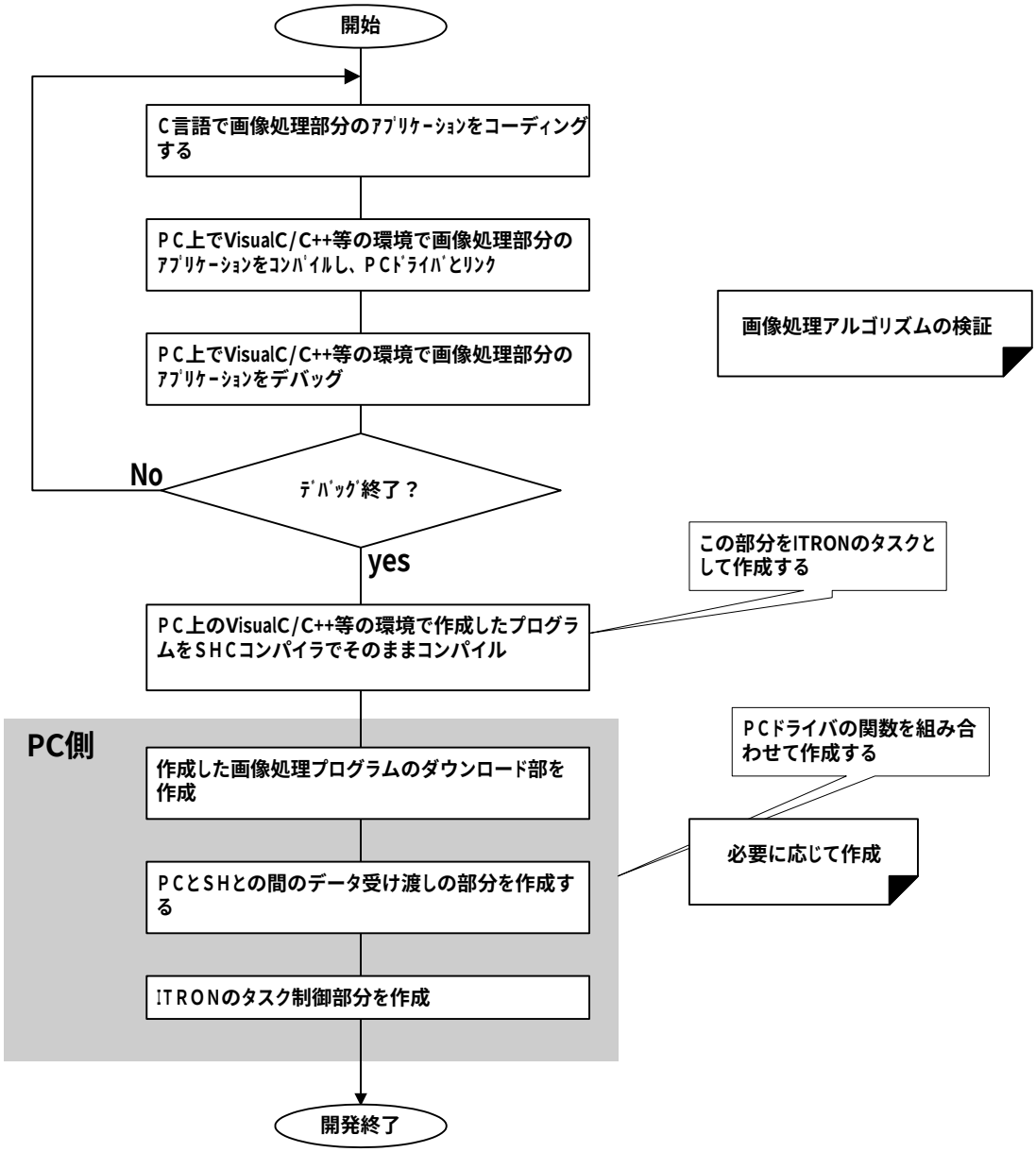


図２－３　割込起動モジュール開発フロー

2．4．3　ユニット内アプリケーションの開発

ユニット内アプリケーションは、ＳＶＰ－３３０のオンボードＣＰＵ上で動作するアプリケーションです。電源投入と同時にＳＶＰ－３３０上のフラッシュメモリに書き込んでおいたユーザアプリケーションを動作させることが出来ます。

ユニット内アプリケーションの開発フローを図２－４に示します。ＰＣドライバの画像処理コマンドでアプリケーションを作成し、ＶｉｓｕａｌＣ／Ｃ＋＋でコンパイル、デバッグをＰＣ上で行い、アルゴリズムの検証等を行います。それらの設計が終了後、ＳＨＣコンパイラでコンパイルし、実行ファイルを作成します。スタートアップファイルにアプリケーションの自動起動を設定することにより、電源投入と同時にそのアプリケーションを自動起動することができます。

なお、ユニット内アプリケーションの詳細は第１８章を参照して下さい。

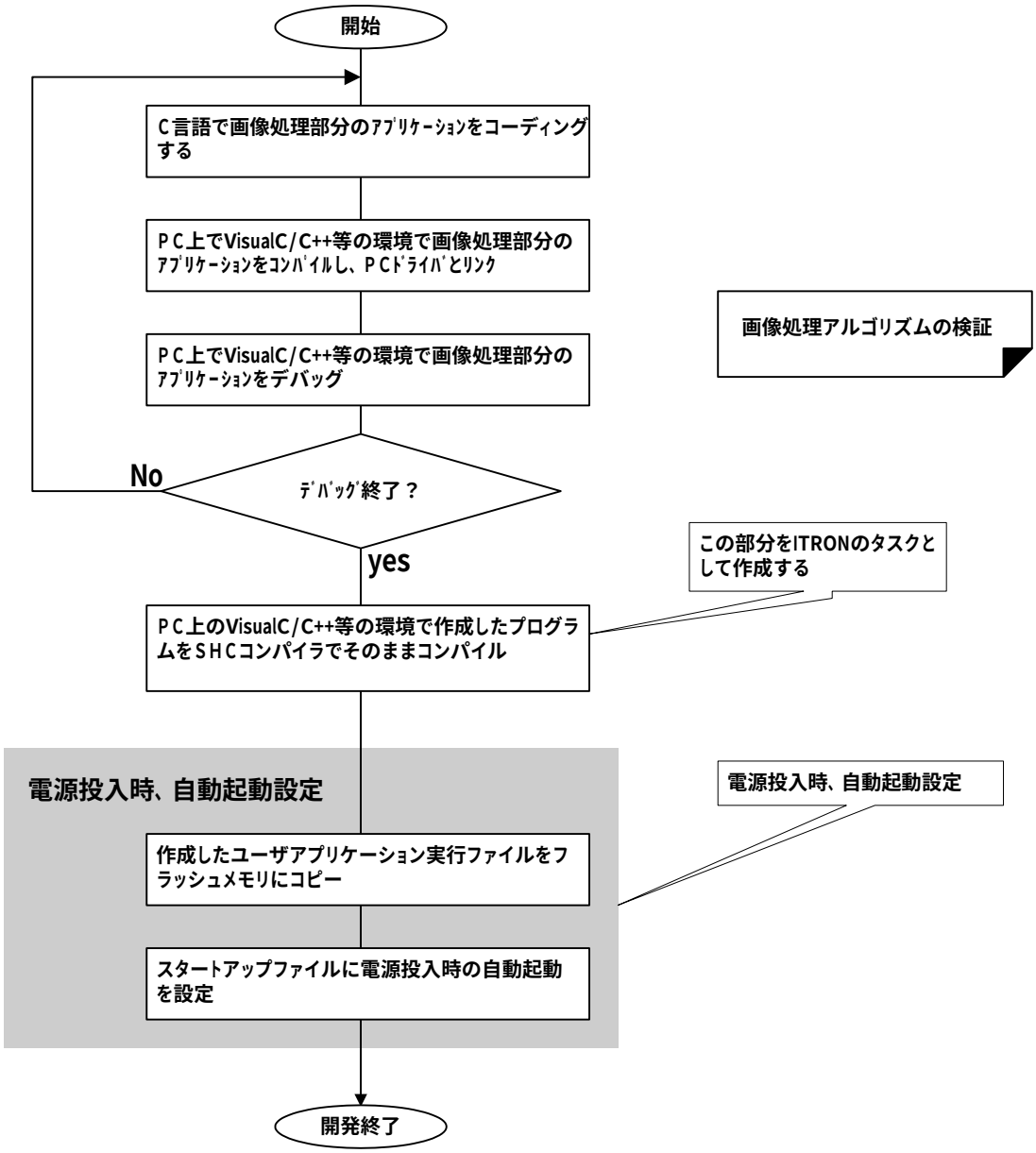


図２－４　ユニット内アプリケーション開発フロー

第３章

PCドライバの概要

３．１ 画像処理コマンドとＰＣドライバ

画像処理コマンドとは、画像処理を行なうための画像入力、画像２値化といった個々のサブルーチンで、Ｃ言語でいえば関数であり画像処理関数と言う場合もあります。また画像処理コマンドは、ユーザーがＳＶＰ－３３０で画像処理アプリケーションを作成する場合の最小の実行単位であり、画像処理アプリケーションは画像処理コマンドを複数組み合わせて作成されます。実行モジュールはＳＶＰ－３３０のＲＯＭやＲＡＭ上にあり、オンボードＣＰＵで実行されます。

一方、ＤＯＳ／Ｖパソコンから画像処理を行なう場合、ＳＶＰ－３３０ではＤＯＳ／Ｖパソコンからコマンド通信により、画像処理プロセッサとオンボードＣＰＵに画像処理を実行させています。特にその部分を”ＰＣドライバ”と呼んでいます。

また、画像処理コマンドは、通常ＰＣドライバを含んでいますが、ＰＣドライバから実行する場合、特に”ＰＣコマンド”という呼び方をする場合があります。

3．2 PCドライバの動作概要

ＳＶＰ－３３０では、実行モジュールはＳＶＰ－３３０のＲＯＭやＲＡＭ上にあり、ＰＣから実行する画像処理コマンドは画像処理プロセッサとオンボードＣＰＵにより行なわれます。また、割込みや複数のインテリジェントモジュールのタスクを管理するためにμＩＴＲＯＮ４．０（ＨＩ７７５０／４）を採用し、ＰＣアドインシステムでの画像処理を実現しています。ＰＣドライバは図３－１に示すようにＰＣ（パソコン）とＳＶＰ－３３０のオンボードＣＰＵとの間でコマンド通信することによりオンボードＣＰＵで画像処理を実行します。

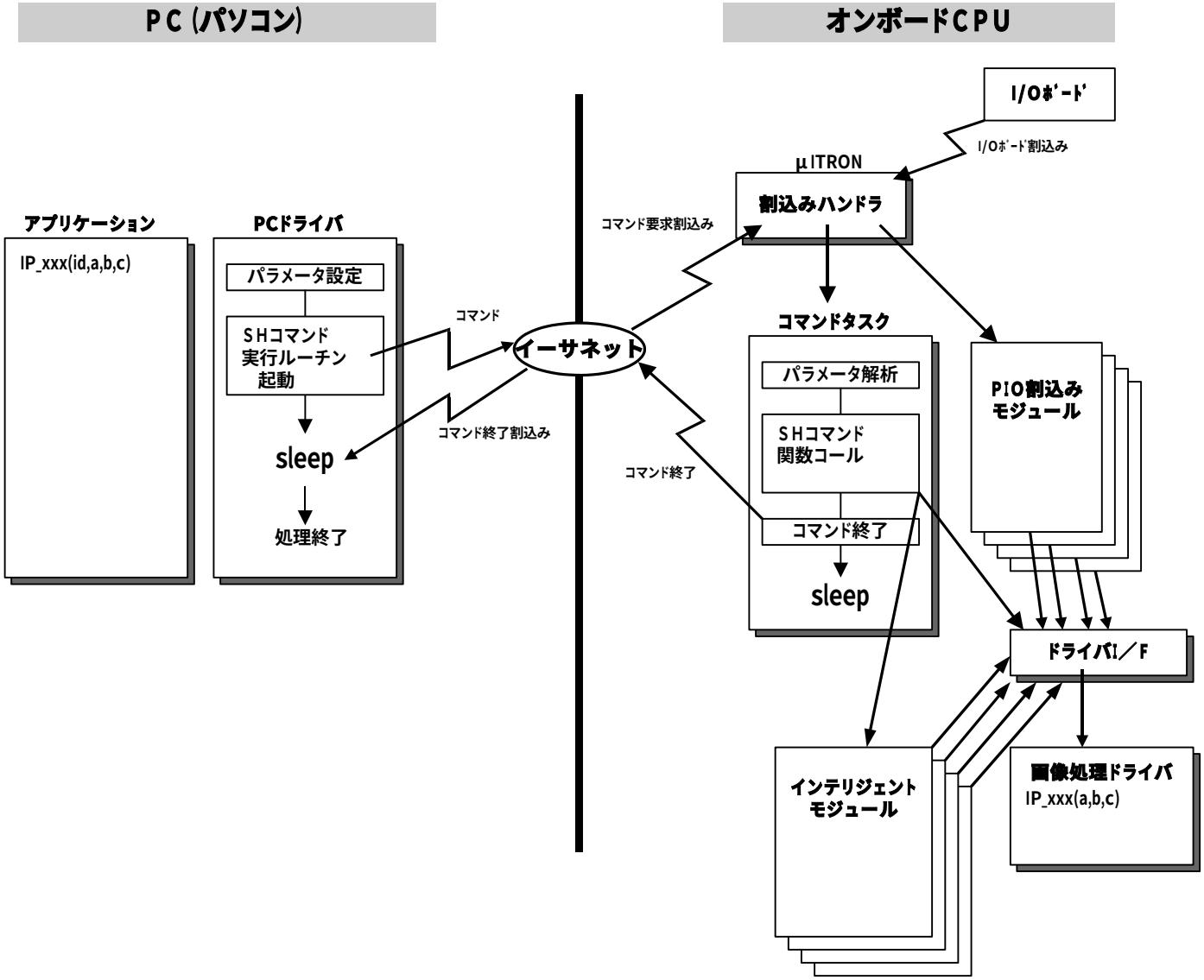


図３－１　ＰＣドライバの仕組み

3．3 インテリジェントモジュール

インテリジェントモジュールとは、画像処理コマンドを複数組み合わせて作成するユーザーオリジナルの画像処理コマンドのことです。

ＳＶＰ－３３０を使用したＰＣでは、画像処理は画像処理プロセッサとオンボードＣＰＵにより行なっているため、ユーザーの画像処理アプリケーションの一部をインテリジェントモジュールとして実装することにより、処理の分散化を実現できます。

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登録し実行されます。もちろんＰＣドライバとしてダウンロード、モジュール登録、実行などを簡単に行なうことができるように関数を用意しています。

3．4 割込起動モジュール

割込起動モジュールとは、ＳＶＰ－３３０のパラレルＩ／Ｏ（ＰＩＯ）等のデバイスの割込みにより起動することができるモジュールです。画像処理コマンドを複数組み合わせて作成したユーザーオリジナルの画像処理モジュールをＰＩＯの割込みにより起動するモジュールとして簡単に登録することができます。

ＰＩＯ割込起動モジュールはμＩＴＲＯＮの１つのタスクとして登録し実行されます。もちろんＰＣドライバとしてダウンロード、タスク登録、実行などを簡単に行なうことができるように関数を用意しています。

3．5 WindowsでのPCドライバの構成

ＷｉｎｄｏｗｓでのＰＣドライバは、図３－２に示すように、ＤＬＬ（Dynamic Link Library）とカーネルモードのデバイスドライバ、オンボードＣＰＵの画像処理コマンド実行ファイルで構成されています。画像処理コマンドやユーザが作成したオンボードＣＰＵ上のプログラムを使用する場合は、オンボードＣＰＵの画像処理コマンド実行ファイルをＳＶＰ－３３０にダウンロードします。

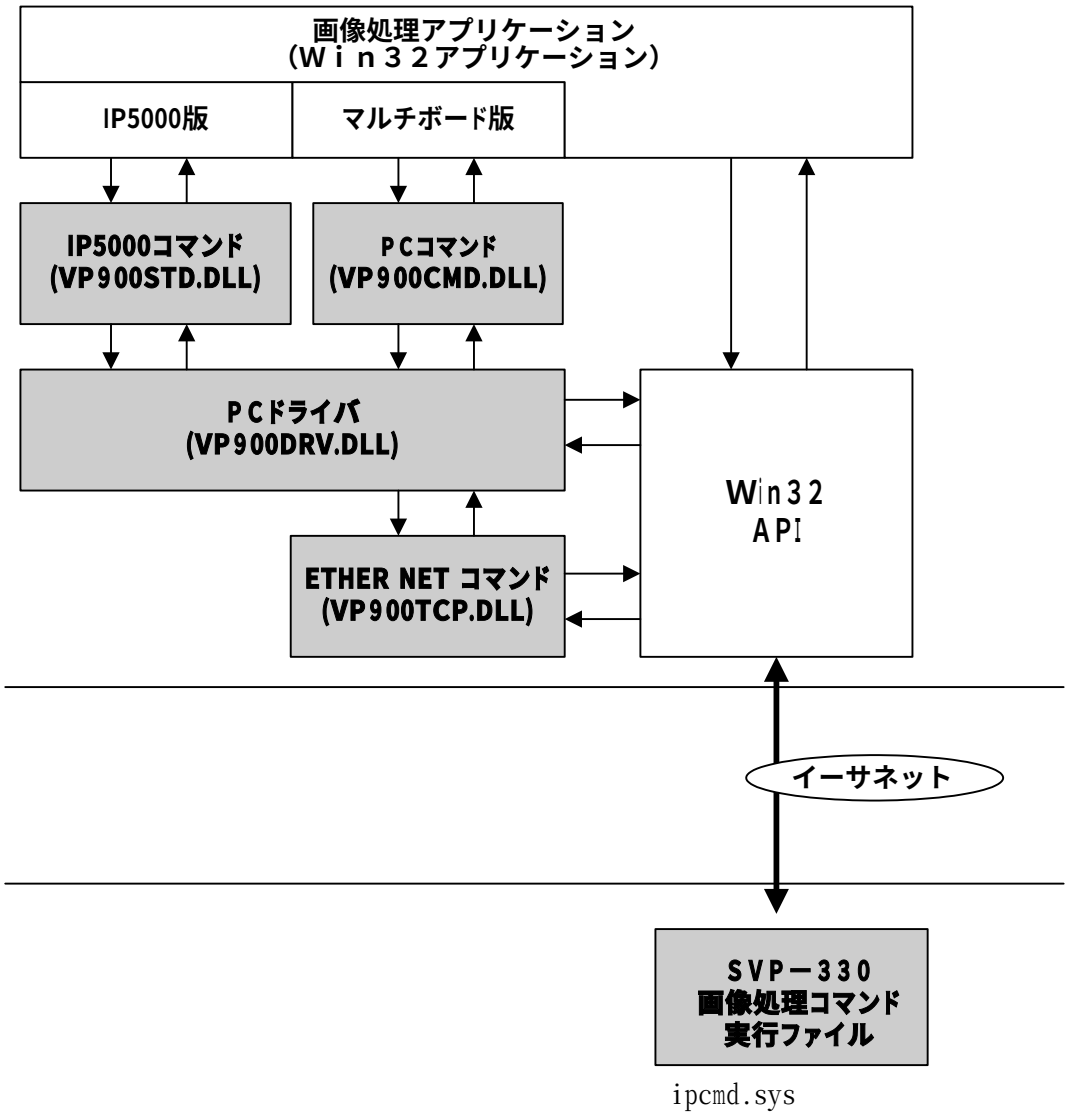


図３－２　ＷｉｎｄｏｗｓでのＰＣドライバの構成

第４章

PCドライバでのプログラミング

Ｗｉｎｄｏｗｓ上でＰＣドライバとVisual C/C++で作成できるアプリケーションには次のようなものがあります。

（1）コンソールアプリケーション

最上位の関数が ｍａｉｎ（ ）で記述される標準入出力がコンソールベースのアプリケーションです。Ｗｉｎｄｏｗｓのグラフィカルインタフェースを使用しないため、最も手軽に作成できます。

（2）Ｗｉｎｄｏｗｓアプリケーション

Ｗｉｎｄｏｗｓのグラフィカルインタフェースを使用したＣ言語で作成されるアプリケーション。

（3）ＭＦＣアプリケーション

Ｗｉｎｄｏｗｓのグラフィカルインタフェースを ＭＦＣ（Microsoft Foundation Class）クラスライブラリを使用してＣ＋＋言語で作成されるアプリケーション。

4.1 プログラムのコンパイルとリンク

ＰＣドライバを使用する場合、ＳＶＰ－３３０ＳＤＫで用意しているインクルードファイルをＣのソースプログラムからインクルードすることと、ＰＣドライバのＤＬＬインポートライブラリをリンクする必要があります。

MicrosoftのVisual C/C++ 6.0でプログラムをコンパイル、リンクする場合、下記の要領でインクルードファイルのディレクトリパスの設定とライブラリのプロジェクトへの追加を行なって下さい。

4.1.1 インクルードファイルのディレクトリパスの設定

プログラムをコンパイルする場合、Visual C/C++のVisualStudio上でインクルードファイルのディレクトリパスを設定します。

［プロジェクト］メニューから［設定］を選択します。［プロジェクト設定］のダイアログボックスが表示されたら、［Ｃ／Ｃ＋＋］のタブを選択します。カテゴリのコンボボックスから［プリプロセッサ］を選択すると”**インクルードファイルのパス**”という項目が表示されます。ここに、ＳＶＰ－３３０ＳＤＫをインストールしたディレクトリを入力します。インストール時にディレクトリの変更がなければ

Ｃ：¥ＶＰ３３０ＳＤＫ¥ＰｃＤｒｖ¥ｉｎｃｌｕｄｅ

と入力します。

4.1.2 ライブラリのプロジェクトへの追加

ＰＣドライバのＤＬＬインポートライブラリをリンクする場合、Visual C/C++のVisualStudio上でライブラリをプロジェクトに追加します。

［プロジェクト］メニューの［プロジェクトへ追加］から［ファイル］を選択します。［プロジェクトへ追加］のダイアログボックスが表示されたら、［ファイルの種類］のコンボボックスから［ライブラリファイル（ｌｉｂ）］を選択しＳＶＰ－３３０ＳＤＫのインストールされたディレクトリからＰＣドライバのＤＬＬインポートライブラリを選択します。インストール時にディレクトリの変更がなければ

Ｃ：¥ＶＰ３３０ＳＤＫ¥ＰｃＤｒｖ¥ｌｉｂ¥ＶＰ９００ＣＭＤ．ｌｉｂ

を選択します。

4．2 コーディング規約

ＰＣドライバ画像処理コマンドを使用したアプリケーションプログラム作成における、プログラムの記述規約、使用規約について説明します。

4．2．1 インクルードファイルの記述

表４－１に示すインクルードファイルを表の順番でプログラムにインクルードしてください。尚、

```
vpxdef.h
vpxsys.h
vpxfnc.h
```

のファイルは必ずインクルードして下さい。

表４－１　インクルードファイル一覧

ファイル名	内容	摘要
vpxdef.h	define,enum定義	必ずインクルードして下さい
vpxsys.h	構造体定義	必ずインクルードして下さい
vpxfnc.h	プロトタイプ定義	必ずインクルードして下さい

4．2．2 画像処理コマンドのセットアップ

ＰＣドライバ画像処理コマンドを使用する場合は、必ずユーザーのプログラムで下記の画像処理コマンドのセットアップを行なって下さい。簡単なプログラム例を図４－１に示します。

① 画像処理ボードのオープン･クローズ

ＰＣドライバ画像処理コマンドを使用する場合は、必ずプログラムの最初で画像処理ボードのオープン処理（OpenIPDevExt( )）を実行しデバイスＩＤを取得して下さい。そしてプログラムの終了時に画像処理ボードのクローズ処理（CloseIPDev( )）を実行して下さい。

```
devID =  OpenIPDevExt(IPBOARD_0,NULL);
         *
         *
         *

CloseIPDev(devID);
```

② 画像処理コマンドの初期化

ＳＶＰ－３３０のハードウェアの初期化と画像処理コマンドの初期化を
InitIP( )
または
vpxInitIP( )
コマンドにより行います。

```
#include     "vpxdef.h"
#include     "vpxsys.h"
#include     "vpxfnc.h"

viod main( )
{
/* ローカル変数の設定  */
    DEVID   devID;
    int     ImgID1,ImgID2,ImgID3;

/*  プログラム          */

    /* 画像処理ボードのオープン */
    devID = OpenIPDevExt(IPBOARD_0,NULL);
    if(devID == ISPX_NULL){
        printf("デバイスのオープンに失敗しました\n");
        exit(0);
    }

    /* 画像処理コマンドの初期化 */
    InitIP(devID);

    SetVideoFrame(devID,INTERLACE,VIDEO_FS_512H_480V);
    ImgID1 = AllocImg(devID,IMG_FS_512H_512V);
    ImgID2 = AllocImg(devID,IMG_FS_512H_512V);
    ImgID3 = AllocImg(devID,IMG_FS_512H_512V);
    GetCamera(devID,ImgID1);
    IP_AddConst(devID,ImgID1,ImgID2,20);
    IP_Binarize(devID,ImgID2,ImgID3,120);
    DispImg(devID,ImgID3);

    /* 画像処理ボードのクローズ */
    CloseIPDev(devID);

}
```

このインクルードファイルは必ずインクルードして下さい

画像処理ボードのオープンと画像処理コマンドのセットアップ

画像処理コマンド初期化

画像処理ボードのクローズ

図４－１　コーディング例

4．2．3 定数 (define) 定義

ＰＣドライバ画像処理コマンドでは、プログラムの保守性を高めるため、define宣言で数値を文字列として定義しています。プログラムを作成する場合は画像処理コマンドリファレンスを参照し、関数のパラメーターとしてdefine定義された文字列が指定されている場合には数値を直接設定せずにdefine定義された文字列を使用して下さい。また、インクルードファイルであらかじめdefine定義している文字列がありますので、ユーザーのプログラムの中でそれらを再定義しないでください。define定義は vpxdef.h ファイルで定義していますので、参照してください。

4．2．4 型 (typedef) 定義

ＰＣドライバ画像処理コマンドでは、プログラムの保守性を高めるため、typedef宣言でＣ言語であらかじめ定義されている変数の型を別の文字列として定義しています。プログラムを作成する場合は画像処理コマンドリファレンスを参照し、関数のパラメーターとしてtypedef定義された文字列が指定されている場合には指定された型で変数を定義して下さい。また、インクルードファイルであらかじめtypedef定義している文字列がありますので、ユーザーのプログラムの中でそれらを再定義しないでください。typedef定義は vpxdef.h ファイルで定義していますので、参照してください。

4．2．5 構造体 (struct) 定義

ＰＣドライバ画像処理コマンドでは、インクルードファイルであらかじめ構造体定義している文字列がありますので、ユーザーのプログラムの中でそれらを再定義しないでください。構造体定義は vpxsys.h ファイルで定義していますので、参照してください。

4．2．6　列挙型 (enum) 定義

ＰＣドライバ画像処理コマンドでは、プログラムの保守性を高めるため、define宣言で数値を文字列として定義していますが更にenum宣言で定義しているものもあります。プログラムを作成する場合は画像処理コマンドリファレンスを参照し、関数のパラメーターとしてenum定義された文字列が指定されている場合には数値を直接設定するとコンパイルエラーになります。原則的にはenum定義された文字列を使用することを勧めますが、どうしても直接数値を設定したい場合や変数を設定しなければならないときはenum型でキャストして下さい。

また、インクルードファイルであらかじめ enum 定義している文字列がありますので、ユーザーのプログラムの中でそれらを再定義しないでください。enum 定義は vpxdef.hファイルで定義していますので、参照してください。

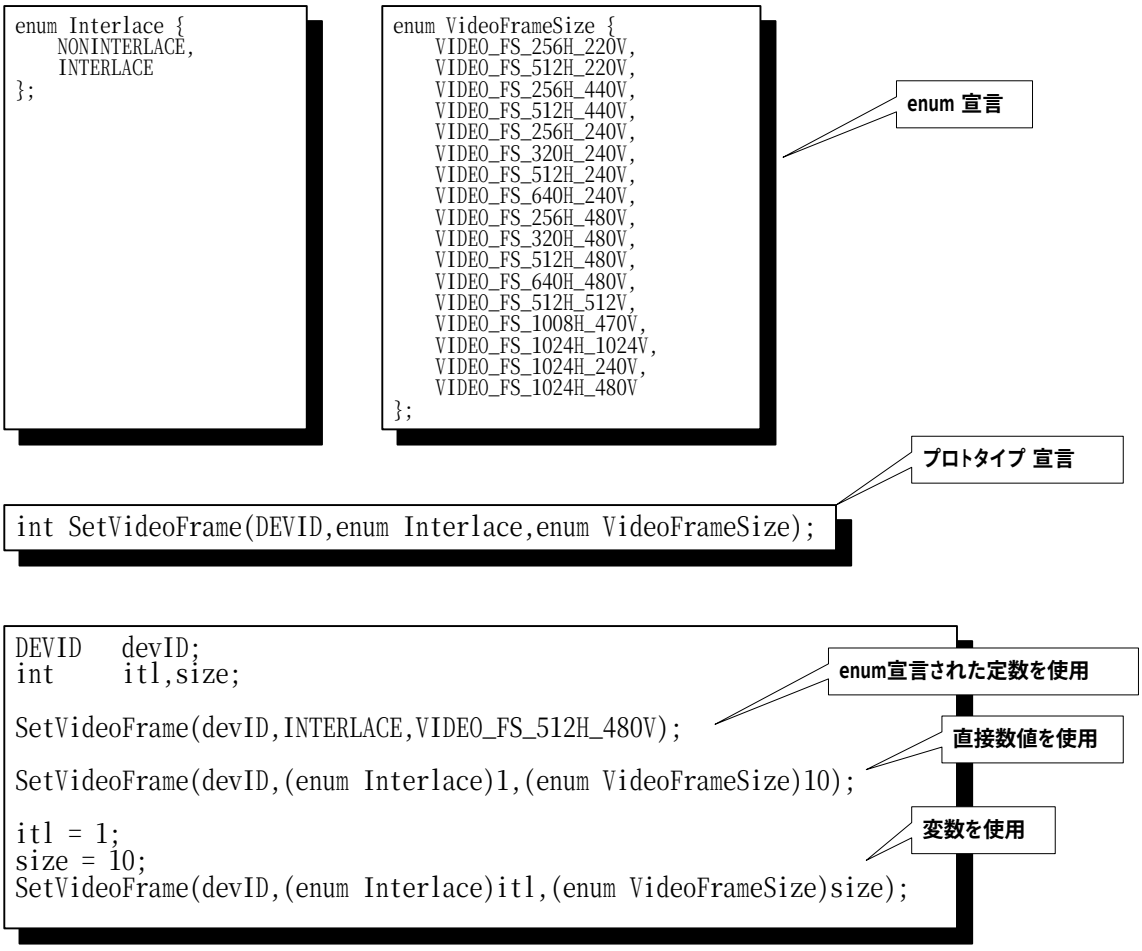


図４－２　enum型のキャスト

4．2．7　注釈 (コメント)

Ｃ言語の仕様では「 /* 」と「 */ 」で囲まれた領域に注釈（コメント）を書きます。「 // 」コメントの使用も可能です。

第５章

画像処理コマンドの基礎

５．１ 画像処理コマンドの構成

画像間演算、空間フィルタリングなどの画像処理コマンドは、サブルーチン形式のコマンドとしてＣ言語で利用できます。

図５－１　画像処理コマンドの構成

5．2 PCドライバと画像処理コマンドのセットアップ

ＰＣアドインシステムで画像処理コマンドを使用する場合、ＰＣドライバによる画像処理コマンドを使用します。ＰＣドライバ画像処理コマンドは、図５－２の要領でＰＣドライバのオープン、クローズと画像処理コマンドのセットアップを行なう必要があります。また、このとき OpenIPDevExt コマンドで画像処理コマンドで使用するデバイスＩＤ（devID）を取得します。

図５－２　画像処理ボードのセットアップ

5．3 PCドライバとデバイスID

ＰＣドライバでは、画像処理ボードのマルチボード構成やマルチスレッド、マルチプロセスに対応するため、内部で各々のリソースの管理を行なっています。そのため、ＰＣドライバ画像処理関数の第１パラメータには、デバイスＩＤを指定するようになっています。デバイスＩＤは OpenIPDevExt( ) 関数で取得します。

```
devID = OpenIPDevExt( BOARD_0, NULL );

IP_Invert( devID, ImgSrc, ImgDst );
```

5．4 画像処理コマンドのエラー処理

5．4．1 コマンドエラー

画像処理コマンドでパラメータエラー等のエラーが発生時した場合、Ｗｉｎｄｏｗｓのメッセージボックスでユーザーにそれを知らせる機能があります。

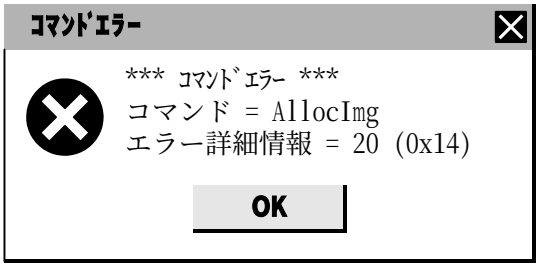


図５－３　コマンドエラー処理

5．4．2 エラー処理の制御

画像処理コマンドではシステムでエラーの制御を行っています。画像処理コマンドでは、一旦、画像処理コマンドエラーが発生すると引き続く画像処理コマンドで処理を実行しません。再び処理を再開するには ClearIPError コマンドで画像処理コマンドエラーをクリアして下さい。

また、画像処理コマンドでエラーが発生した際にエラーのメッセージボックスが表示されますが、EnableIPErrorMessage , DisableIPErrorMessageでその出力を制御することが可能です。

5．5 画像処理の手順

ＳＶＰ－３３０での画像処理の一般的な処理フローを示します。

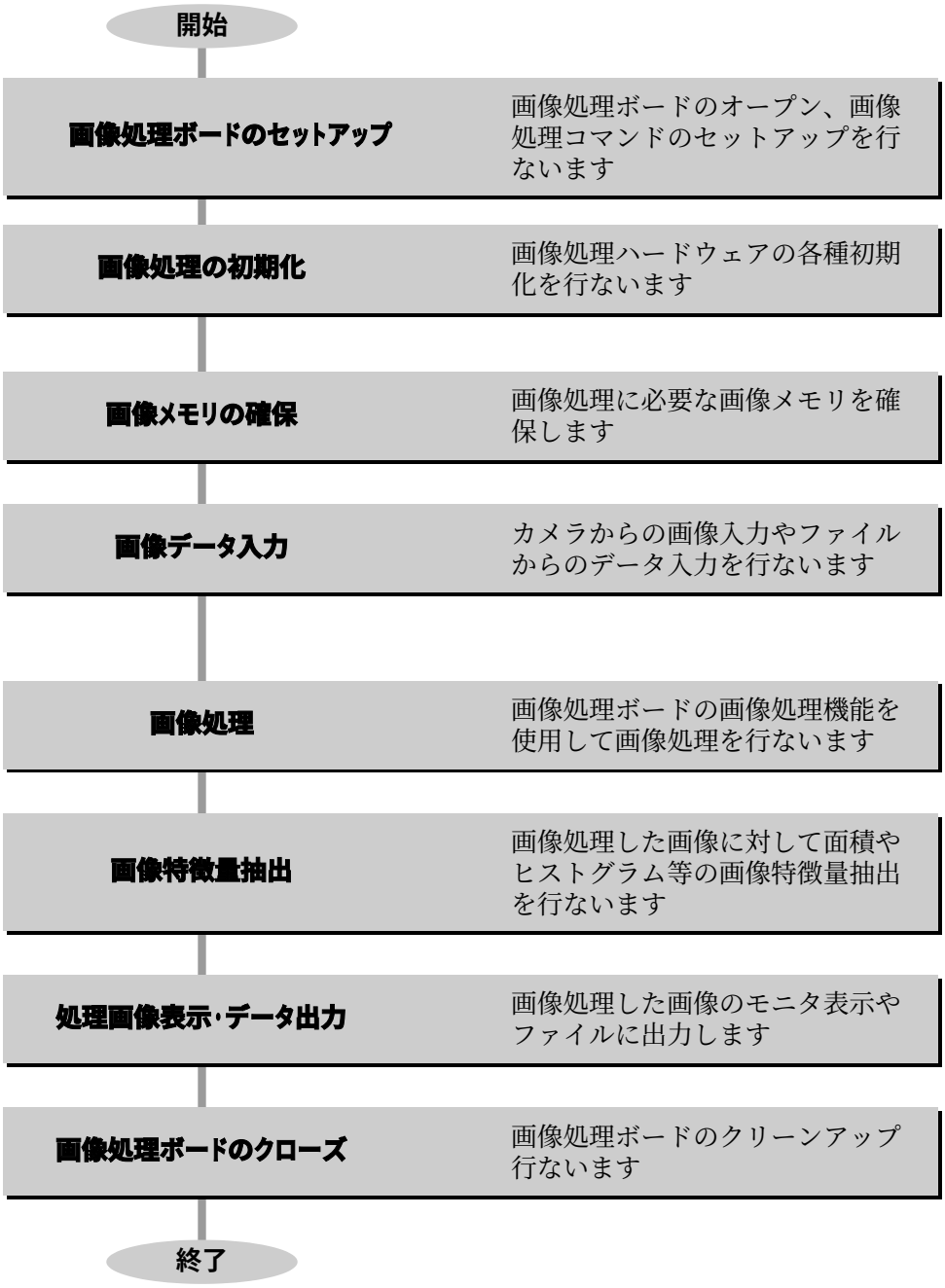


図５－４　画像処理フロー

5．6 画像処理の動作

画像処理は、処理対象画像メモリからのデータ読出し、画像処理プロセッサによる演算、処理結果画像の画像メモリへの書込みの順番で行われます。処理対象画像は、ソース画像又はソース画面、処理結果画像はデスティネーション画像又はデスティネーション画面と呼びます。

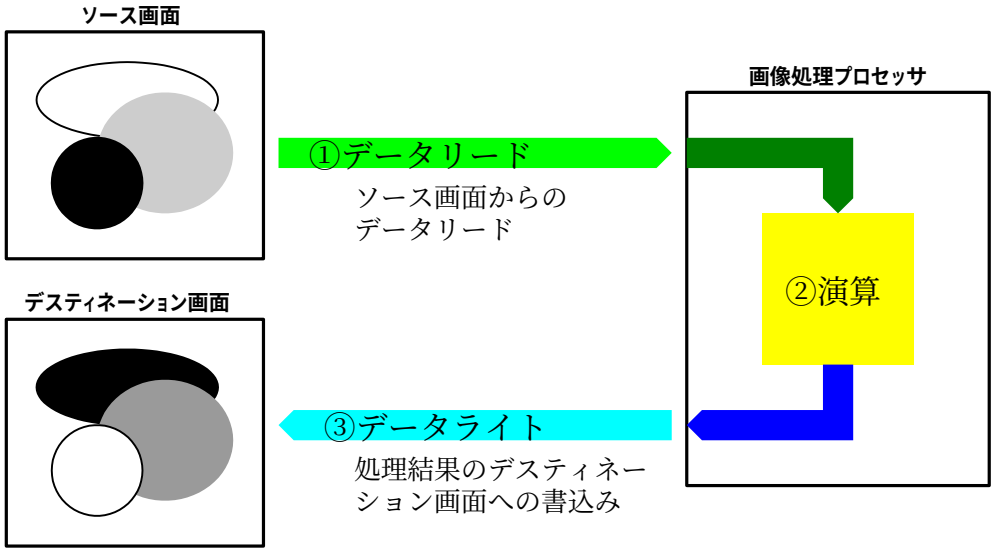
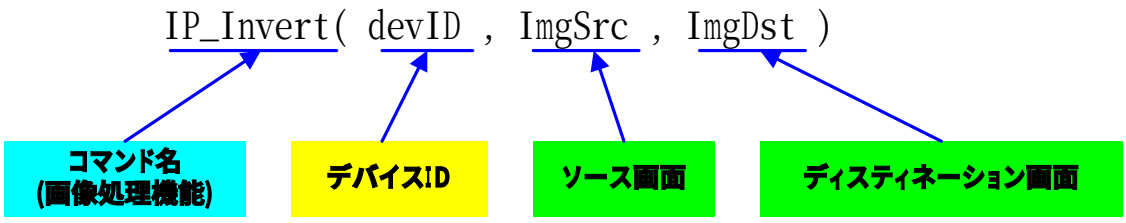


図５－５　画像処理の動作

ＰＣドライバ画像処理コマンドは、画像処理の機能をＣ言語のサブルーチンで提供しています。画像処理のプログラムでは、コマンド（画像処理機能）の選択、デバイスＩＤの設定、ソース画面とデスティネーション画面の選択によって画像処理を実行します。

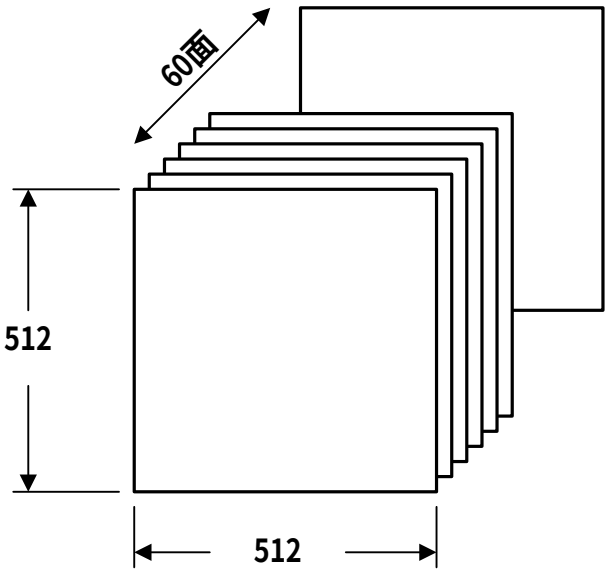


5．7 画像メモリの基礎

5．7．1 画像メモリのハードウェア構成

画像メモリは、カメラ映像の入力データや画像処理演算結果を格納するメモリです。画像メモリの容量は、８ビットで約１６Mバイトあり、５１２×５１２画素の画面であれば６０面程度確保できる容量です。

また、画像メモリには濃淡画像メモリと２値画像メモリの物理的な区別はなく、２値画像として使用する場合には、「 FF(h) 」を「１」として「 00(h) 」を「０」として処理されます。

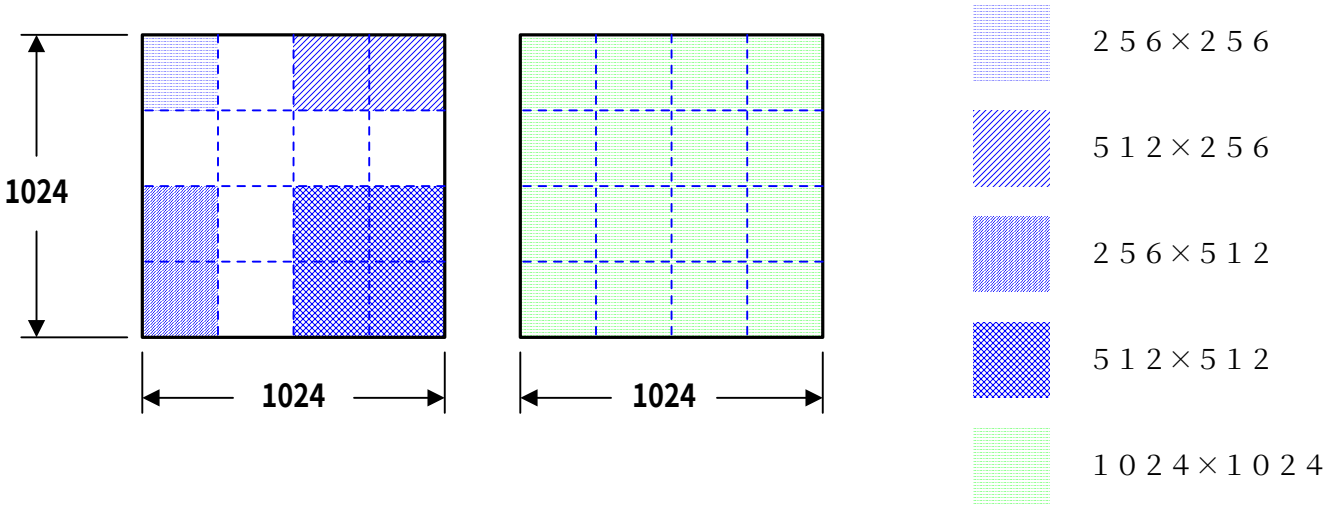


単位（画素）

図５－６　画像メモリ容量

5．7．2 画像メモリのサイズ

使用時の画面サイズは２５６×２５６画素を最小単位として最大６４０×５１２画素のサイズを選択することができます。以下にAllocLockImg()コマンドで確保できるサイズの例を示します。
AllocImg()コマンドでは、以下の単位サイズに関係なくリニアに確保します。



単位（画素）

図５－７　画像メモリサイズ

5．7．3　画像メモリの座標系

画面の座標系は、図５－８に示すように画面の左上隅を原点とした右回りの座標です。また、正規化相関サーチやキャリパーコマンドで得られるサブピクセルの座標を図５－９に示します。

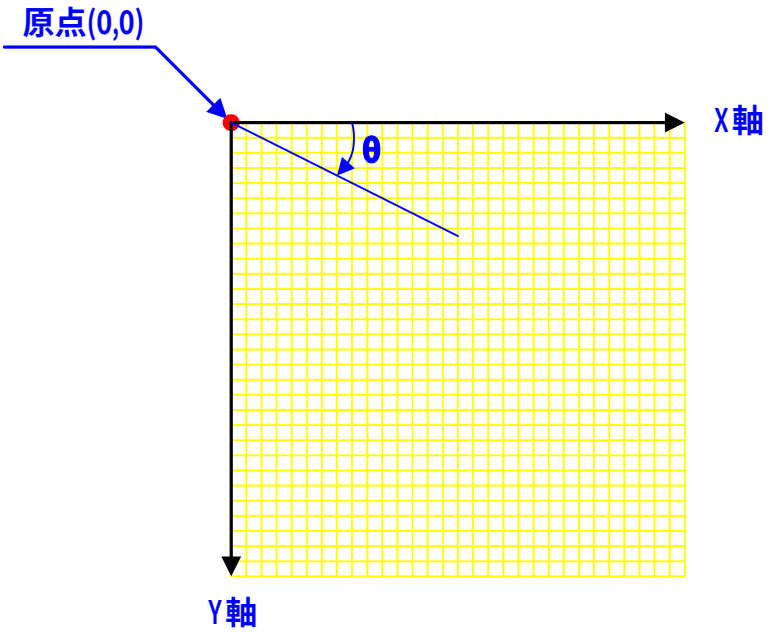


図５－８　画像メモリの座標系

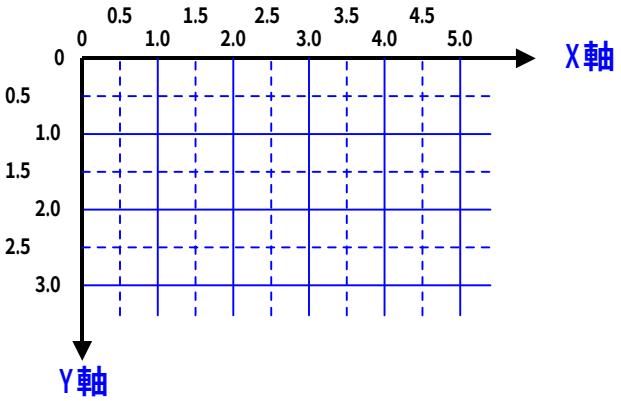


図５－９　サブピクセル座標

5．7．4　画像メモリのデータタイプ

画像メモリに格納されるデータは、濃淡データと２値データがあります。濃淡データは２５６階調を持ち、符号無し８ビット（unsign8）と符号付き８ビット（sign8）の２種類があり、符号無し８ビットは「０～２５５」、符号付き８ビットは「－１２８～１２７」として画像メモリに格納されます。また、２値データは、黒と白の２階調のデータです。黒は「 00(h) 」（０）として「 FF(h) 」（２５５）として画像メモリに格納されます。

5．7．5　画像メモリの使い方

カメラ映像の入力や、またその画像に対して画像処理を行う場合には、前もって必要なサイズの画面を確保する必要があります。

このためのコマンドとして、AllocImg(),AllocYUVImg(),AllocRGBImg()等のコマンドが用意されています。このコマンドを実行すると画像メモリ内で使用する領域を確保し、確保された画面の画面番号をユーザに返します。以降は、この画面番号により、ソース画面、デスティネーション画面を指定し、画像処理を実行します。

5．7．6　Allocimgによる画像メモリの確保

使用したい画像メモリのサイズを指定することによって、画像処理コマンドの管理システムにより、約３２Mバイトの画像メモリ内を検索し、使用する画像メモリを割り当てます。

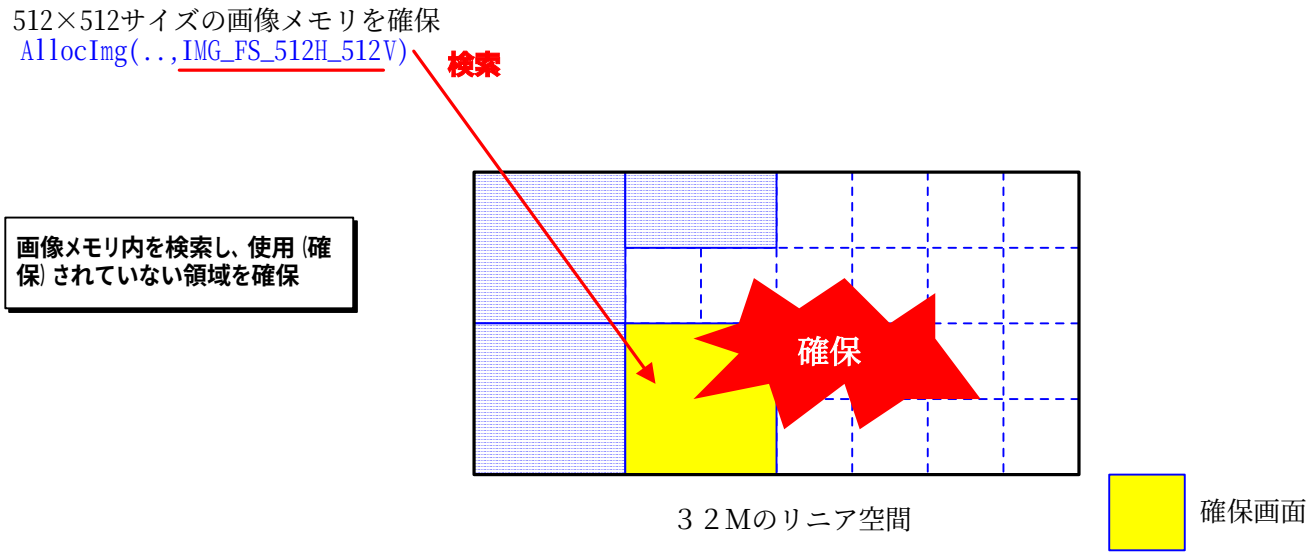


図５－１０　AllocImgによる画像メモリ確保

5．8 ウィンドウの基礎

ウィンドウとは、処理対象画像または、結果格納画面内の処理領域のことをさします。
画像処理は、画像メモリを確保した画面サイズまたは、ウィンドウとして指定された矩形領域で動作します。また、処理時間はウィンドウのサイズにより左右されます。画面のサイズが小さければ処理時間が短くなり、逆に画面のサイズが大きくなれば処理時間が長くなります。画像処理ボードで行われる大部分の画像処理では、処理時間はウィンドウ領域内の画素数に比例します。

ＳＶＰ－３３０での画像処理のウィンドウサイズに関し制限事項があります。
詳細は「第１章 １．２ 画像処理の制限事項」を参照して下さい。

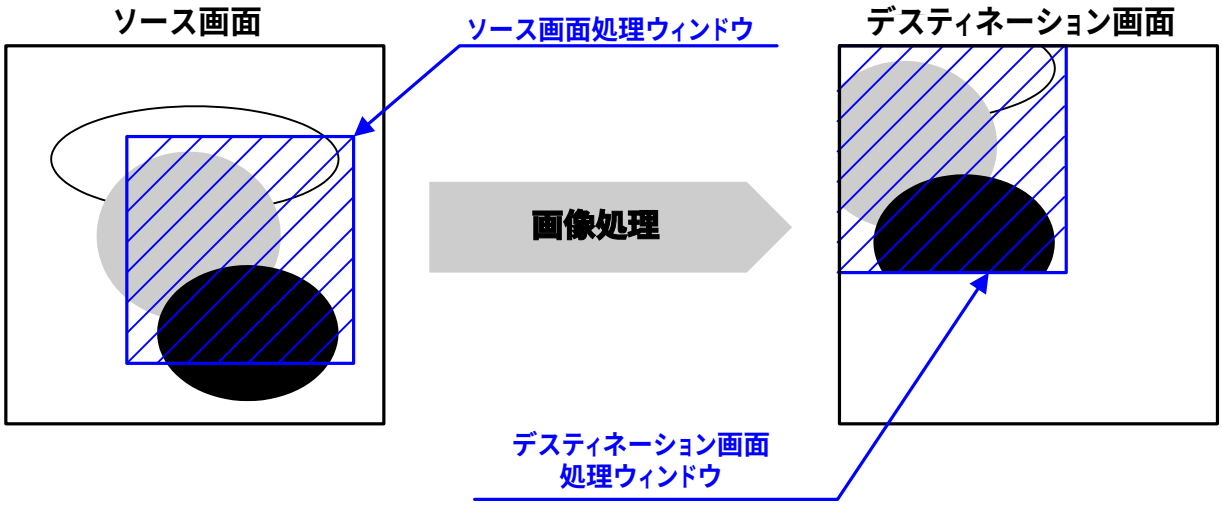


図５－１１ 画像処理とウィンドウ

5．8．1 ウィンドウの座標系

画面の座標系は、画面の左上隅を原点とした図５－１２のような座標系となります。ウィンドウ設定値は、画面の原点からの相対座標を指示して下さい。

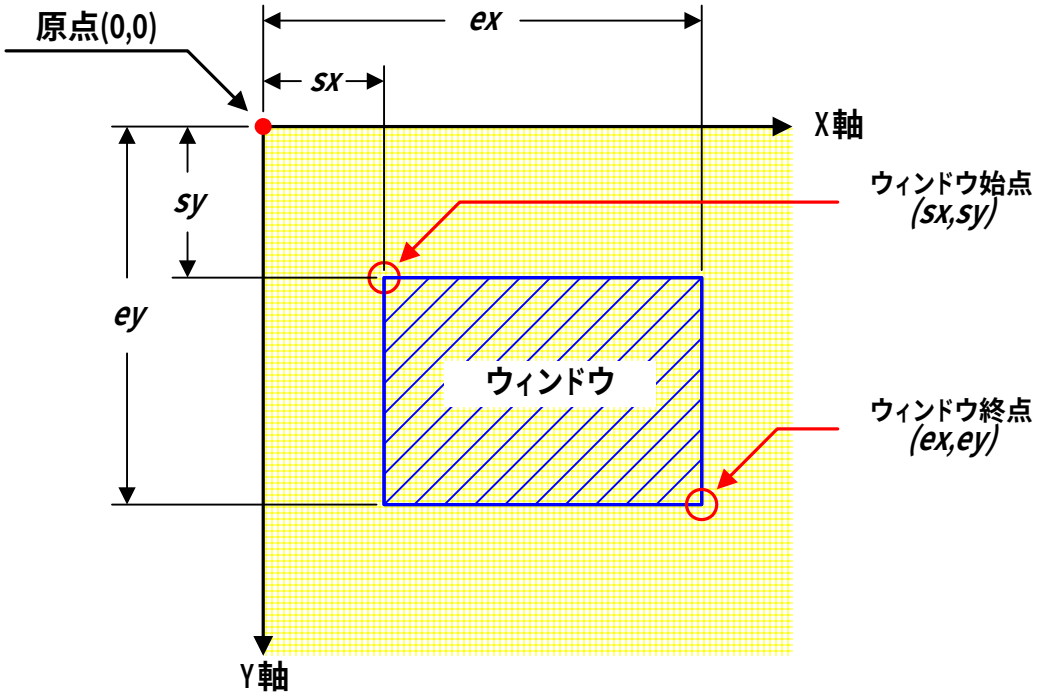


図５－１２ ウィンドウの座標系

5．8．2 ウィンドウの種類

ウィンドウには、表５－１に示す４種類があります。
ウィンドウの設定には、SetAllWindowまたは、SetWindowコマンドを使用します。SetAllWindowは、４種類のウィンドウすべてに対してウィンドウの設定を行います。SetWindowは種類ごとの座標設定が可能です。

表５－１　ウィンドウの種類

ウィンドウ種類	用途
ソース画面１ （ＳＲＣ０＿ＷＩＮ）	画像処理でのソース画面に使用します。
ソース画面２ （ＳＲＣ１＿ＷＩＮ）	画像間演算処理でのソース画面に使用します。 （ソース画面を２画面使用するときの第２ソース画面）
デスティネーション画面 （ＤＳＴ＿ＷＩＮ）	画像処理でのデスティネーション画面に使用します。
画像メモリアクセス （ＳＹＳ＿ＷＩＮ）	画像メモリアクセスコマンド（WriteImg,ReadImg, WritePixel,ReadPixel 等）で有効な領域を指定します。

ソース画面とデスティネーション画面で異なるサイズのウィンドウを設定した場合、それぞれのウィンドウサイズ内の最小のサイズで処理を行います。つまり、図５－１３に示すように目的のコマンドが使用する全てのウィンドウサイズが重なる部分が処理の対象となります。

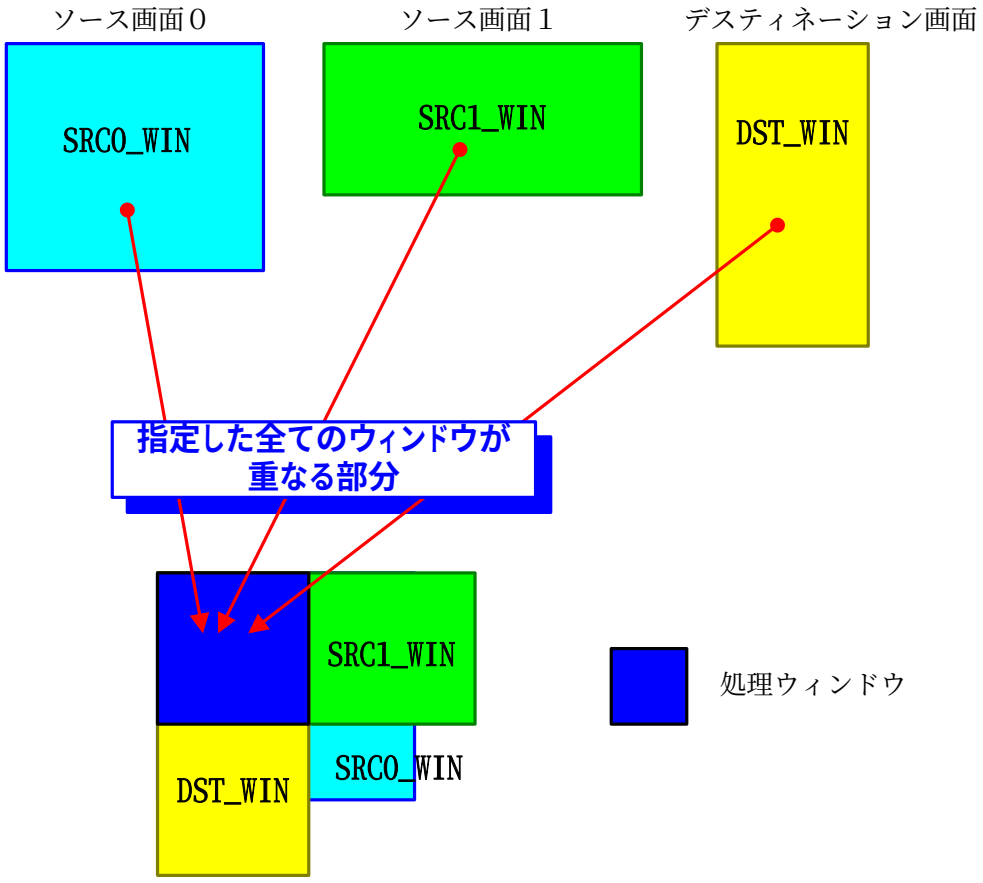


図５－１３　処理領域

5．8．3　ウィンドウの有効／無効

画像処理は、確保した画面のサイズまたは、ウィンドウのサイズで動作します。どちらで動作するかは、ウィンドウの状態で決まり、ウィンドウが無効の時は、確保した画面のデフォルトビデオフレームサイズ、有効のときは設定したウィンドウサイズとなります。この設定は、EnableIPWindow及びDisableIPWindowコマンドによって行います。ウィンドウ無効は、DisableIPWindowコマンド、有効はEnableIPWindowコマンドで行います。ビデオフレームサイズとは、カメラから取り込むまたは、出力する映像サイズのことであり、SetVideoFrameコマンドで設定します。

5．9 画面データタイプの属性

画像処理プロセッサは、映像入出力、画像処理等の処理データを以下のタイプで処理を行います。またデータタイプの種類としてシステムデータタイプと画面データタイプの２種類があります。

表５－２　データタイプ

データタイプ	値の範囲
符号付き８ビット（ＳＩＧＮ８＿ＤＡＴＡ）	－１２８～１２７
符号無し８ビット（ＵＮＳＩＧＮ８＿ＤＡＴＡ）	０～２５５
２値　（ＢＩＮＡＲＹ＿ＤＡＴＡ）	０（黒）または２５５（白）

（１）システムデータタイプ

システムとしてのデフォルト設定のデータタイプです。カメラからの映像取り込み、AllocImgコマンドによる画像メモリ確保時の画面データタイプ等に使用されます。デフォルト設定は、符号無し８ビットです。

システムデータタイプは、SetIPDataTypeコマンドで設定します。

（２）画面データタイプ

画面毎に設定されているデータタイプです。画像処理コマンドによっては、対象画面のデータタイプにより処理結果が変わるものもあります。

AllcImgコマンドで確保された画面の画面データタイプは、システムデータタイプに設定されます。また、ChangeImgDataTypeコマンドで変更可能です。

5．9．1　画像演算と演算結果の正規化

画像処理プロセッサで行う演算は、ソース画面のデータタイプに従って演算を行い、表５－３に示す出力画面データタイプに従い正規化された演算結果がデスティネーション画面に格納されます。ここでいう正規化とは、演算結果が符号付き８ビット(-128～127)や符号無し８ビット(0～255)の値を超える場合、その範囲の中に収まるように処理することです。

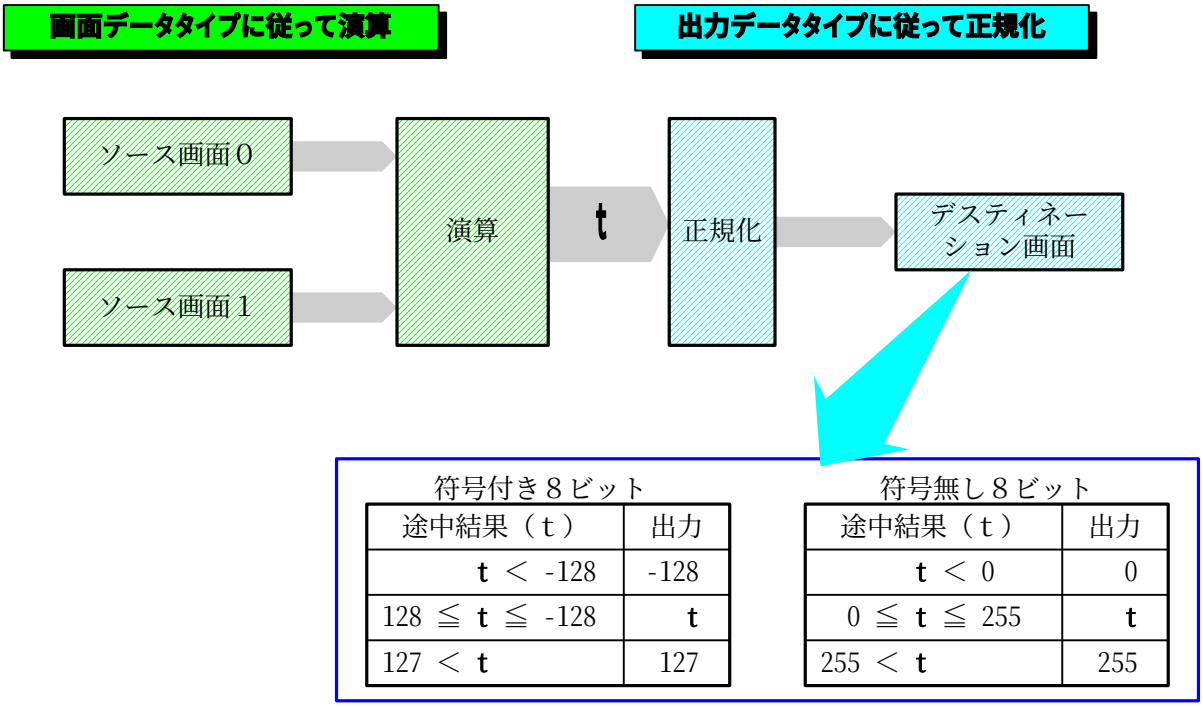


図５－１４　演算の正規化

表５－３　データタイプ（１／２）

コマンド名	出力画面データタイプ	コマンド名	出力画面データタイプ
IP_ClearAllImg	システムデータタイプ	IP_SubConstAbs	システムデータタイプ
IP_ClearCHImg	システムデータタイプ	IP_MultConst	システムデータタイプ
IP_ClearImg	システムデータタイプ	IP_MinConst	システムデータタイプ
IP_Const	システムデータタイプ	IP_MaxConst	システムデータタイプ
IP_Copy	ソース画面１データタイプ	IP_ConvertLUT	システムデータタイプ
IP_Zoom	ソース画面１データタイプ	IP_ShiftDown	システムデータタイプ
IP_ZoomExt	ソース画面１データタイプ	IP_ShiftUp	システムデータタイプ
IP_ZoomOut	ソース画面１データタイプ	IP_AndConst	システムデータタイプ
IP_ZoomOutExt	ソース画面１データタイプ	IP_Add	システムデータタイプ
IP_Shift	ソース画面１データタイプ	IP_Sub	システムデータタイプ
IP_ZoomS	ソース画面１データタイプ	IP_SubAbs	システムデータタイプ
IP_Rotate	ソース画面１データタイプ	IP_Comb	システムデータタイプ
IP_Binarize	２値	IP_CombAbs	システムデータタイプ
IP_BinarizeExt	２値	IP_Mult	システムデータタイプ
IP_Invert	システムデータタイプ	IP_Average	システムデータタイプ
IP_Minus	システムデータタイプ	IP_Min	システムデータタイプ
IP_Abs	システムデータタイプ	IP_Max	システムデータタイプ
IP_AddConst	システムデータタイプ	IP_SubConstAbsAdd	システムデータタイプ

表５－３　データタイプ（２／２）

コマンド名	出力画面データタイプ	コマンド名	出力画面データタイプ
IP_AubConstMultAdd	システムデータタイプ	IP_Med8FLT	システムデータタイプ
IP_SubConstMult	システムデータタイプ	IP_Label4	符号なし８ビット
IP_And	ソース画面１データタイプ	IP_Label8	符号なし８ビット
IP_Or	ソース画面１データタイプ	IP_Label4withAreaFLT	符号なし８ビット
IP_Xor	ソース画面１データタイプ	IP_Label8withAreaFLT	符号なし８ビット
IP_InvertAnd	ソース画面１データタイプ	IP_Label4withAreaFLTSort	符号なし８ビット
IP_InvertOr	ソース画面１データタイプ	IP_Label8withAreaFLTSort	符号なし８ビット
IP_Xnor	ソース画面１データタイプ		
IP_PickNoize4	２値		
IP_PickNoise8	２値		
IP_Outline4	２値		
IP_Outline8	２値		
IP_Dilation4	２値		
IP_Dilation8	２値		
IP_Erosion4	２値		
IP_Erosion8	２値		
IP_Shrink4	２値		
IP_Shrink8	２値		
IP_Thin4	２値		
IP_Thin8	２値		
IP_SmoothFLT	システムデータタイプ		
IP_EdgeFLT	システムデータタイプ		
IP_EdgeFLTAbs	システムデータタイプ		
IP_Lapl4FLT	システムデータタイプ		
IP_Lapl8FLT	システムデータタイプ		
IP_Lapl4FLTAbs	システムデータタイプ		
IP_Lapl8FLTAbs	システムデータタイプ		
IP_LineFLT	システムデータタイプ		
IP_LineFLTAbs	システムデータタイプ		
IP_MinFLT	システムデータタイプ		
IP_MinFLT4	システムデータタイプ		
IP_MinFLT8	システムデータタイプ		
IP_MaxFLT	システムデータタイプ		
IP_MaxFLT4	システムデータタイプ		
IP_MaxFLT8	システムデータタイプ		
IP_LineMinFLT	システムデータタイプ		
IP_LineMaxFLT	システムデータタイプ		
IP_RankFLT	システムデータタイプ		
IP_Rank4FLT	システムデータタイプ		
IP_Rank8FLT	システムデータタイプ		
IP_MedFLT	システムデータタイプ		
IP_Med4FLT	システムデータタイプ		

5.10 映像入出力の基礎

5.10.1 カメラ映像入力

以下にＳＶＰ－３３０のサポートカメラを示します。

表５－４　ＳＶＰ－３３０　サポートカメラ

ＳＶＰ	対応カメラ	フレーム シャッタ	備考
ＳＶＰ－３３０	ＮＴＳＣカメラ (モノクロ／カラー)	――	インタレース／ノンインタレース

図５－１５にカメラ映像入力系のハードウェア構成を示します。

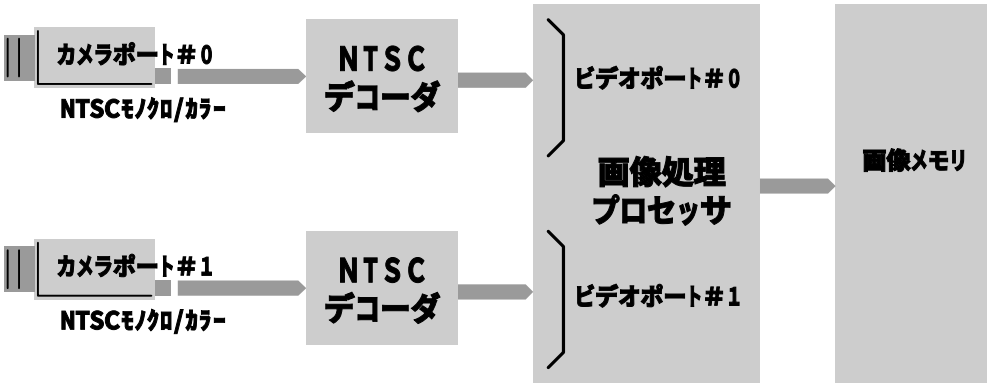


図５－１５　映像入力部の構成

カメラのチャネル切換及び、カメラタイプの切換は、SelectCamera コマンドで行います。
なお、ＳＶＰ－３３０のデフォルトはＹＵＶカメラに設定されます。

5.10.2 カメラ映像入力と画面サイズ

カメラ映像入力とは、カメラが出力する映像データを画像処理ボードの画像メモリに入力（書込む）することです。ビデオフレームサイズとは、カメラ映像を入力する際の解像度のことです。

ビデオフレームサイズは、使用するカメラとインタレース／ノンインタレースのモードにより決まります。カメラのフレームサイズとモードは SetVideoFrame コマンドで設定します。デフォルトでは、インタレースモードで５１２×４８０に設定されています。

また、入力される画像メモリの大きさはそのビデオフレームサイズの大きさ以上である必要があります。表５－５にカメラ解像度と画像メモリサイズの関係を示します。使用するカメラとフレームサイズに合わせて、AllocImg コマンドで画像メモリを確保して下さい。

表５－５　ビデオフレームサイズと画像メモリサイズ

カメラ分類	メーカ	型式	モード	ビデオフレームサイズ(H×V)	画像メモリサイズ(H×V)
ＮＴＳＣカメラ	――	――	インタレース	256×220	256×256
				256×240	256×256
				320×240	512×256
				512×220	512×256
				512×240	512×256
				640×240	640×256
				512×440	512×512
				512×480	512×512
				640×480	640×512
			ノンインタレース	256×220	256×256
				256×240	256×256
				320×240	512×256
				512×220	512×256
				512×240	512×256
				640×240	640×256

5.10.3 映像表示の概要

画像処理ボードは、ＮＴＳＣビデオモニタにカメラ映像や画像メモリの内容を表示することができます。図５－１６に映像出力部の構成を示します。また、それらの映像に線や文字などの映像を重ね合わせて表示することもできます。

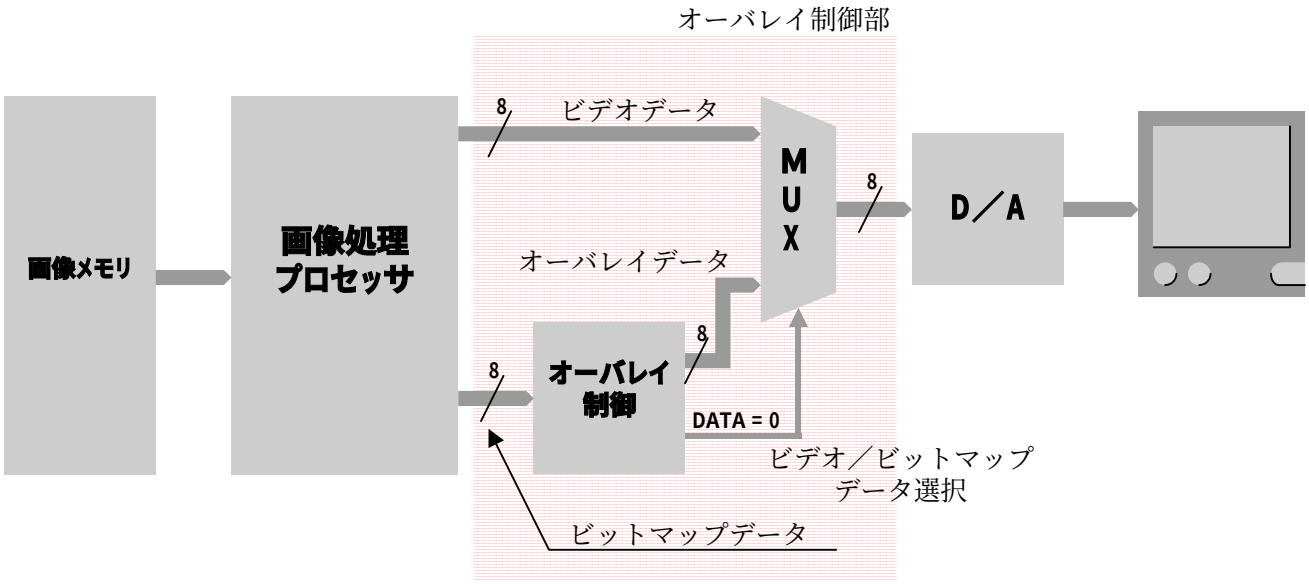


図５－１６　映像出力部の構成

5.10.4 画像メモリ表示とループ表示

画像メモリを表示する場合、画像処理プロセッサが画像メモリを設定された出力のビデオフレームに従いスキャンしてデータを読み出します。ビデオフレームが５１２×４８０の場合、１回のスキャンで５１２×４８０の映像データを出力します。しかし、１回のスキャンだけでは、人間の目に映像として認識できません。ですから画像メモリを表示している間は常にそのスキャンを繰り返しているわけです。そのことをループ表示と呼びます。

5.10.5 カメラ映像表示とループカメラ入力

ＳＶＰ－３３０では、カメラ映像を直接ビデオモニタに表示する場合でも必ず画像メモリにカメラ映像を入力します。そして、その入力した画像メモリをループ表示で表示します。従って、カメラ映像を直接表示している間は、画像入力を繰り返して行わなければなりません。このことをループカメラ入力と呼びます。カメラ映像の直接表示は、ループカメラ入力とループ表示を同時に行っているわけです。

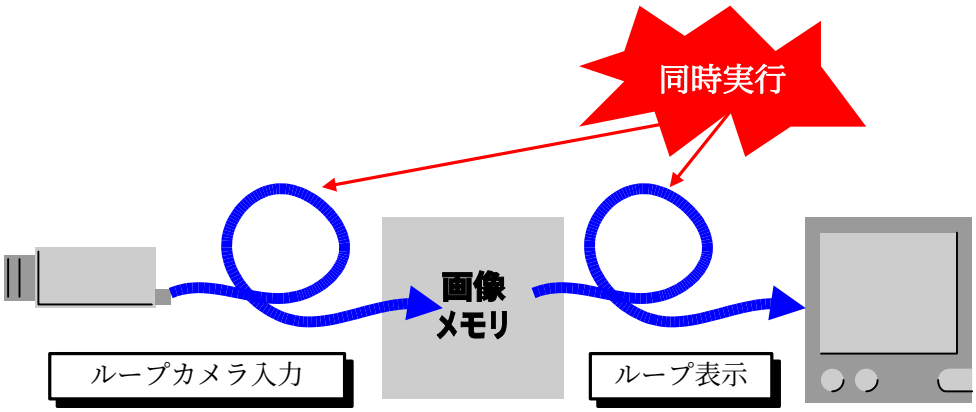


図５－１７　カメラ映像表示

5.10.6 映像表示とオーバラップ表示

画像処理ボードは、図５－１８のオーバレイ制御部により、ＮＴＳＣビデオモニタにカメラ映像や画像メモリの映像に線や文字などの映像を重ね合わせて表示することができます。そのことをオーバラップ表示と呼びます。実際にオーバラップされるのは、画像メモリの一部ですがその領域のことをビットマップ画面と呼びます。また、映像入出力する画像メモリの領域のことをディスプレイ画面と呼びます。

画像処理コマンドでは、ディスプレイ画面とビットマップ画面が割り当てられています。ビットマップ画面とディスプレイ画面を使用しない場合でかつ画像メモリが足りない場合のみ FreeDispImg コマンドでそれらの画面の領域を開放して下さい。

画像メモリを表示している間は、ループ表示を行っているため、常にその画像メモリに対してアクセスが発生しています。その状態で画像処理を行うと画像メモリの競合が発生し、画像処理の性能が低下します。画像表示が必要ない場合、NoDispコマンドでループ入力、ループ表示を止めて下さい。また、画像処理コマンドでは、デフォルトでカメラ映像の表示状態になっているので注意して下さい。

5.10.7 ディスプレイ画面とビットマップ画面のオーバレイ

ビットマップ画面のデータは、物理的に８ビットあります。図５－１８のオーバレイ制御部に示すように、ビットマップ画面のデータが０の時、ディスプレイ画面のデータが出力され、ビットマップ画面のデータが０以外の時、ビットマップ画面のデータが出力されます。

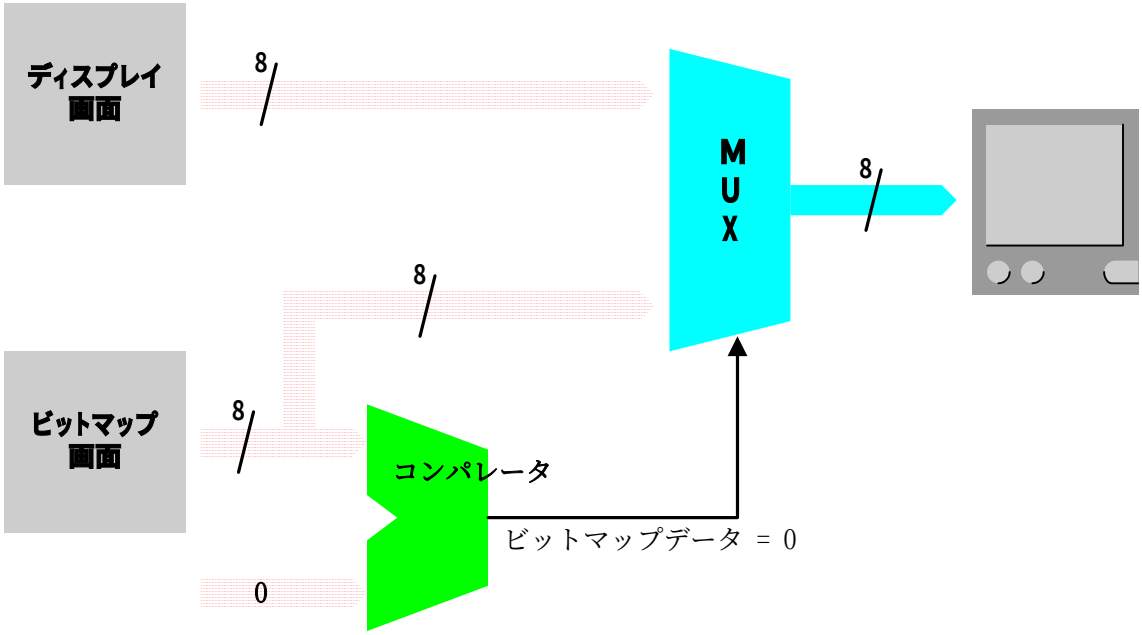


図５－１８　オーバレイ制御

5.10.8 ディスプレイ画面／ビットマップ画面への描画

ディスプレイ画面とビットマップ画面は画像処理ボードの画像メモリの一部です。AllocImgコマンドで確保される画面と物理的に同じものです。ただし、AllocImgコマンドで確保される画面番号（イメージＩＤ）は、ユーザが管理しますが、ディスプレイ画面とビットマップ画面は画像処理コマンドのシステムが管理しています。

ディスプレイ画面とビットマップ画面はAllocImgコマンドで確保される画面と全く同じ画像処理コマンドを使用することができます。このとき、ディスプレイ画面とビットマップ画面の画面番号は、画像処理コマンドのシステムで管理されているため、その画面番号を取得する必要があります。画像処理コマンドでは、GetDispImgIDコマンドでディスプレイ画面、GetBitmapIDコマンドでビットマップ画面の画面番号を取得することができます。

ビットマップ画面には、画像処理結果のポイントや結果として点、直線、矩形や文字を描画したい場合が多いと思います。その場合、ディスプレイ画面とビットマップ画面への描画は画像処理コマンドの描画コマンドを使用して行います。GetDispImgIDコマンドでディスプレイ画面、GetBitmapIDコマンドでビットマップ画面の画面番号を取得すること以外は、通常の処理画面と同じです。

5．11 画像処理機能

本項では、主な画像処理機能や処理例について簡単に説明します。

5．11．1　アフィン変換

アフィン変換とは、拡大・縮小、移動、回転などの幾何学変換のことです。ＳＶＰ－３３０では、式(11.1)の２次元のアフィン変換をオンボードＣＰＵ（ＳＨ４）によりソフト的に処理します。

$$X = ax + by + c$$
$$Y = dx + ey + f$$

(11-1)式

x, y	変換前の座標
X, Y	変換後の座標
a, b, c, d, e, f	任意定数

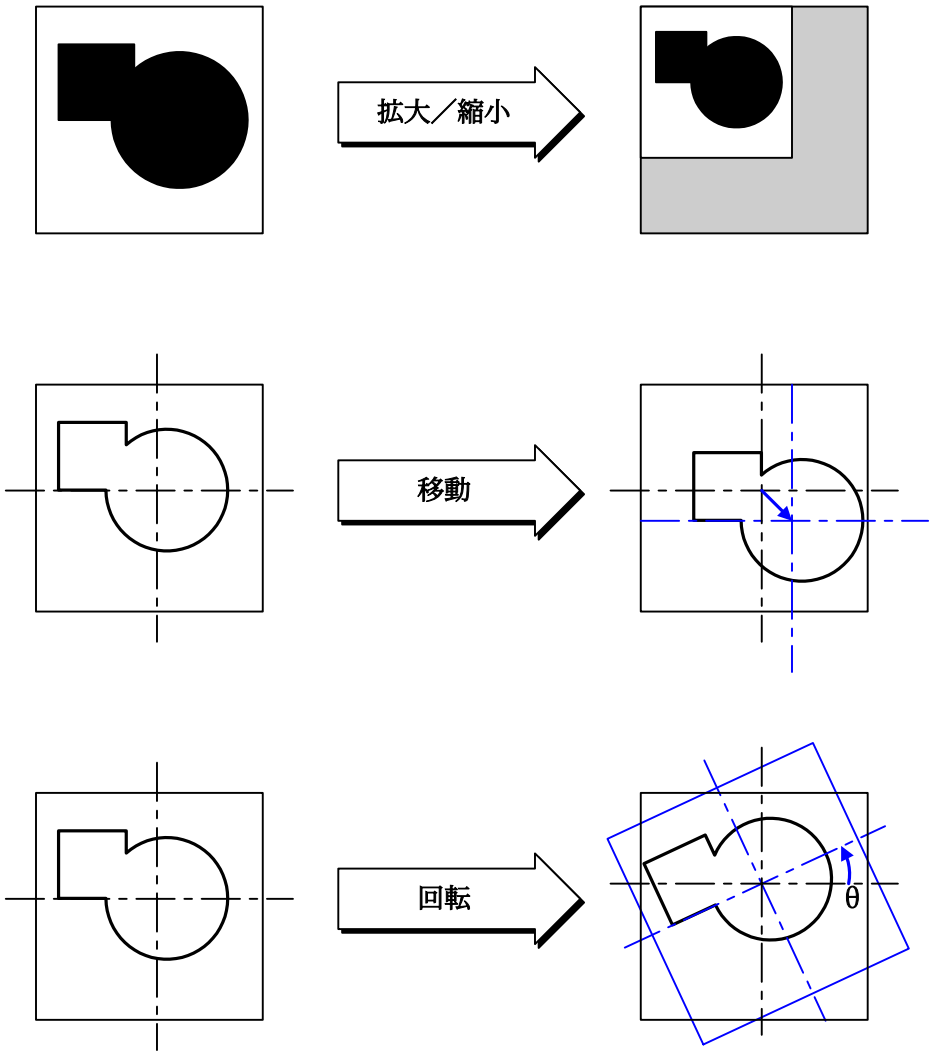


図５－１９　アフィン変換

5.11.2 2値化

２値化とは、濃淡画像（２５６階調）を黒（濃度０）と白（濃度２５５）の２階調の画像に変換することです。画像処理コマンドでは、任意の閾値以上の濃度値を白、それ以外の濃度値を黒とする通常の２値化と、任意の濃度値の範囲内と範囲外で白か黒にする範囲・反転付２値化をサポートしています。

また、２値化の閾値を算出するために判別分析法などによる２値化閾値算出支援コマンドを用意しています。

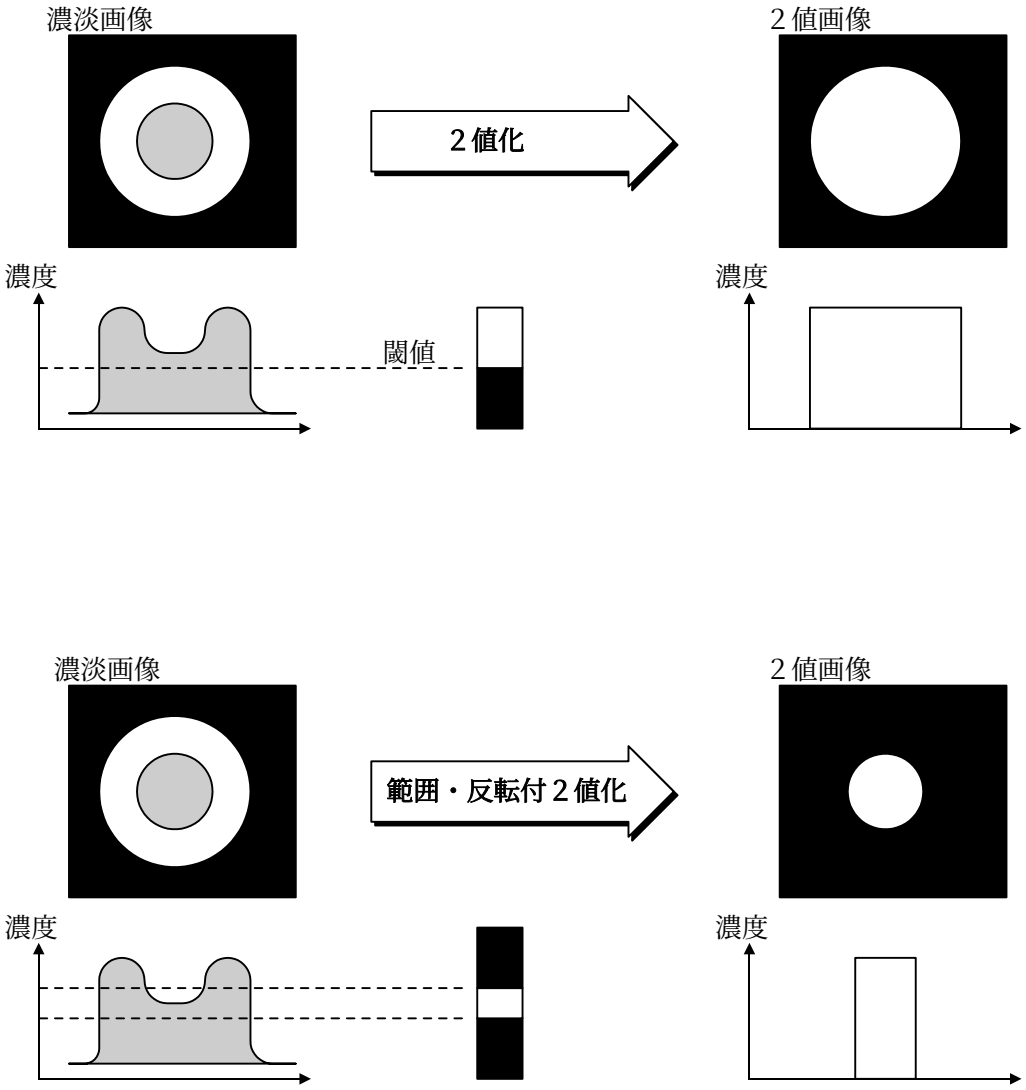


図５－２０　２値化

5.11.3 濃度変換

濃度変換とは、濃度（輝度・明るさ）を変換する処理です。

濃度変換のパターンを換えることにより、暗い部分の濃度を高くしたり、逆に明るい部分の濃度を低くしたりする画質改善が可能です。画質改善以外にも、等濃度縞（ストライプ）のデータを使用すれば、曲面の曲がり、くぼみなどを見つけることができます。さらに、この考えを拡大していけば、３値化、４値化という処理が行えます。濃度変換のパターンは、自由に設定可能です。

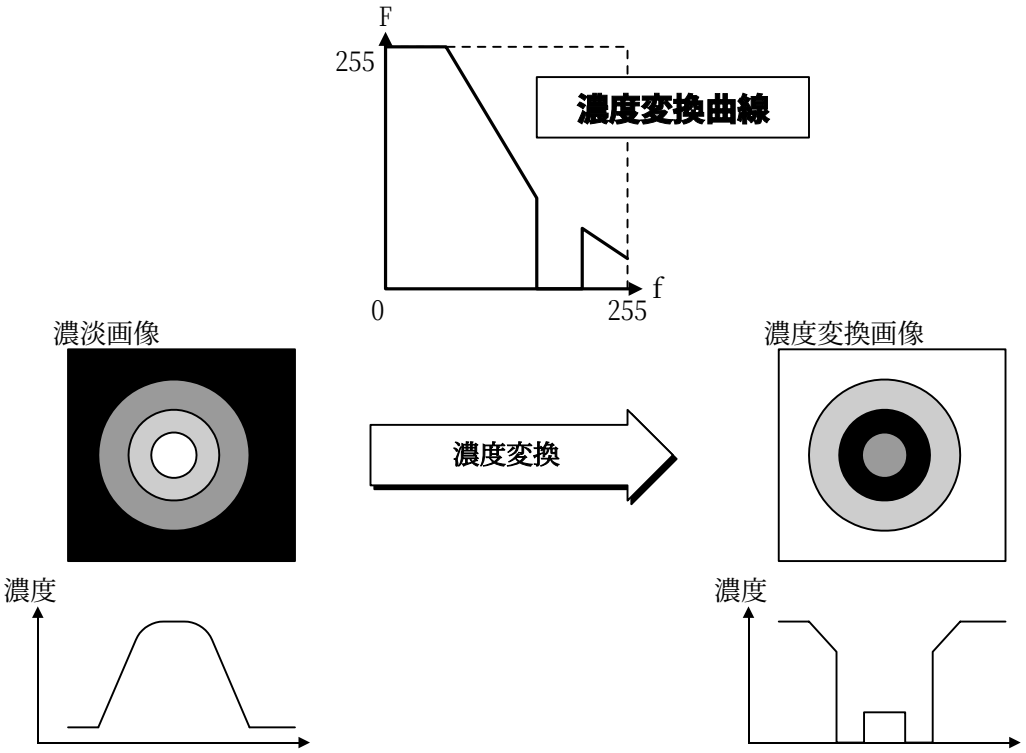
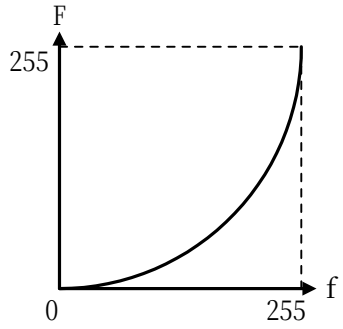
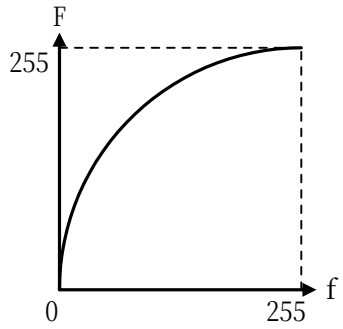
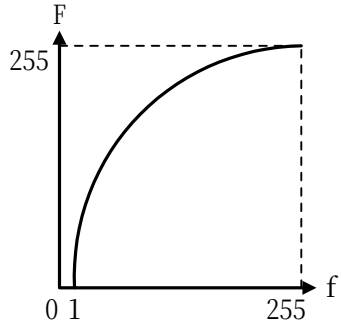
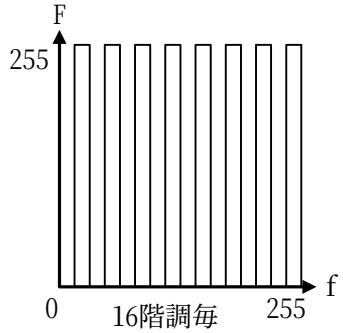
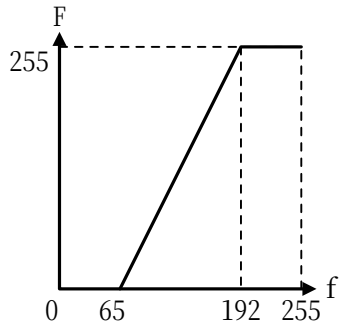


図５－２１　濃度変換

表５－６　濃度変換テーブルの例

種類	濃度変換曲線	詳細	効果
γ補正①		0≦f≦255において $f = af^{1.2}$ （但し $a = 255^{-0.2}$ ） f ： 処理対象画像濃度 F ： 処理結果画像濃度	低濃度部の強調
γ補正②		0≦f≦255において $f = af^{1/1.2}$ （但し $a = 255^{-0.2/1.2}$ ） f ： 処理対象画像濃度 F ： 処理結果画像濃度	高濃度部の強調
log変換		1≦f≦255において f = a*log(f) （但し a = 255/log(255)） f=0 において F=0 f ： 処理対象画像濃度 F ： 処理結果画像濃度	１以下を０とし高濃度部を強調
等濃度縞		0≦f≦255において 16階調毎に f=0 or F=255 f ： 処理対象画像濃度 F ： 処理結果画像濃度	１６階調単位に黒と白に変換
強調		0≦f≦255において f = 0 ～ 64 にて F=0 f = 65 ～ 191 にて F = 255/127*(f-65) f = 192 ～ 255にて F = 255 f ： 処理対象画像濃度 F ： 処理結果画像濃度	低濃度部と高濃度部の強調（コントラスト改善）

5.11.4 画像間算術演算

画像間演算とは、２画面の画像で演算を行い演算後別の１画面に合成することです。画像間演算には加算、減算、乗算などの画像間算術演算と論理和、論理積などの画像間論理演算があります。

画像間算術演算の用途としては、基準パターンとの減算による差画像から欠陥を検出したり、何度も同一画像を取込み加算することでランダムノイズ（ホワイトノイズ）を除去するといったことがあります。また、リアルタイムに２画面の画像の最大値をとることで、明るい物体を次から次へと１画面の画像に合成することができます。これは、移動物体の軌跡や移動量を検出するために有効です。

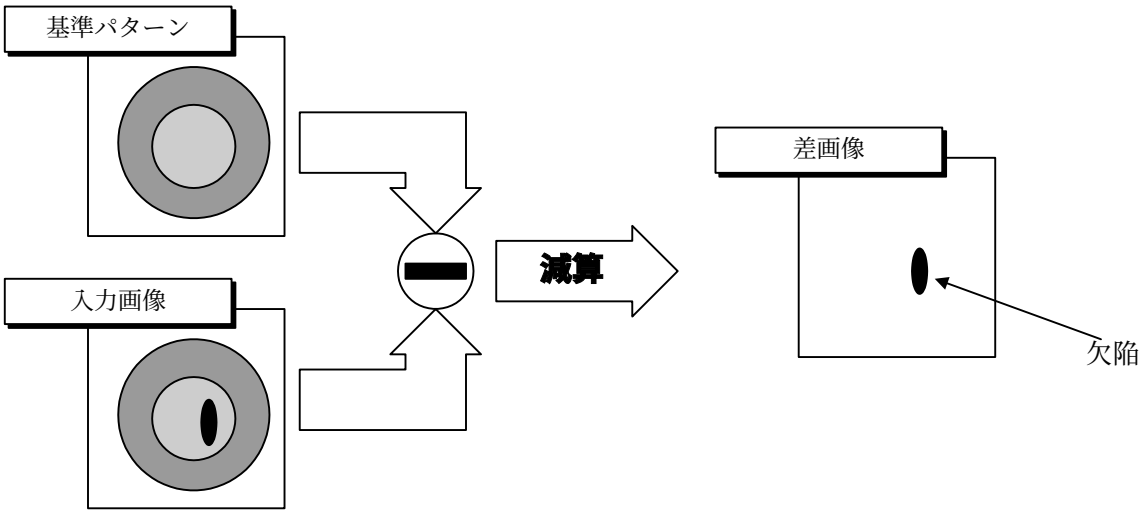


図５－２２　画像間算術演算

5.11.5 画像間論理演算

２画面間で１画素毎に論理和、論理積などの論理演算を行い結果を別の画像に格納します。

２値画像での基準パターンと検査パターンとのチェック（ＸＯＲ演算）、画像の合成（ＯＲ演算）、キズの検出に有効です。

また、画像に任意のパターンでマスクをかける場合にも任意パターンとターゲット画像のＡＮＤ演算を行います。

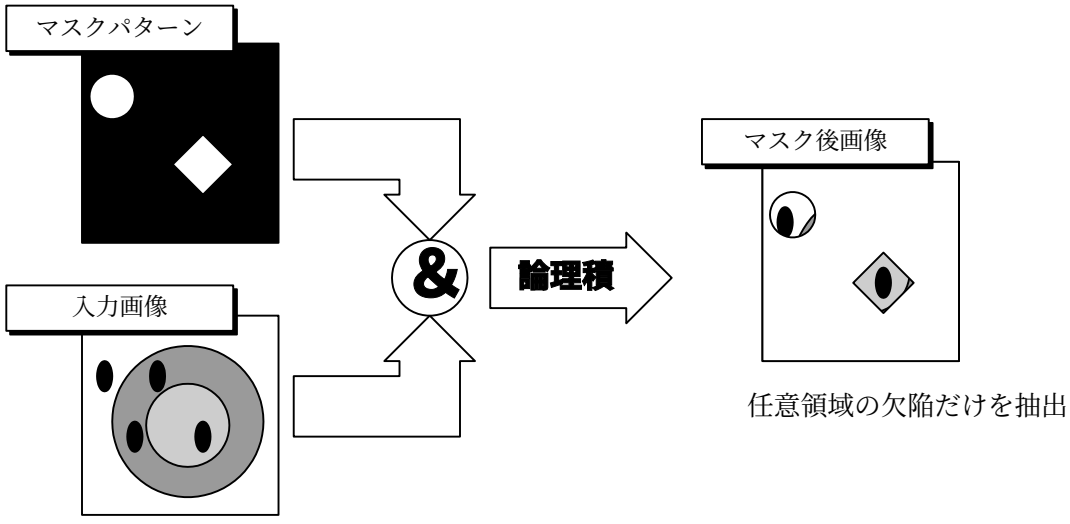


図５－２３　画像間論理演算

5.11.6 2値画像形状変換

２値画像に対してノイズ除去、輪郭抽出、膨張、収縮、細線化、縮退化といった処理を行うことができます。これらの処理は、目的に応じて４連結と８連結を選択できます。

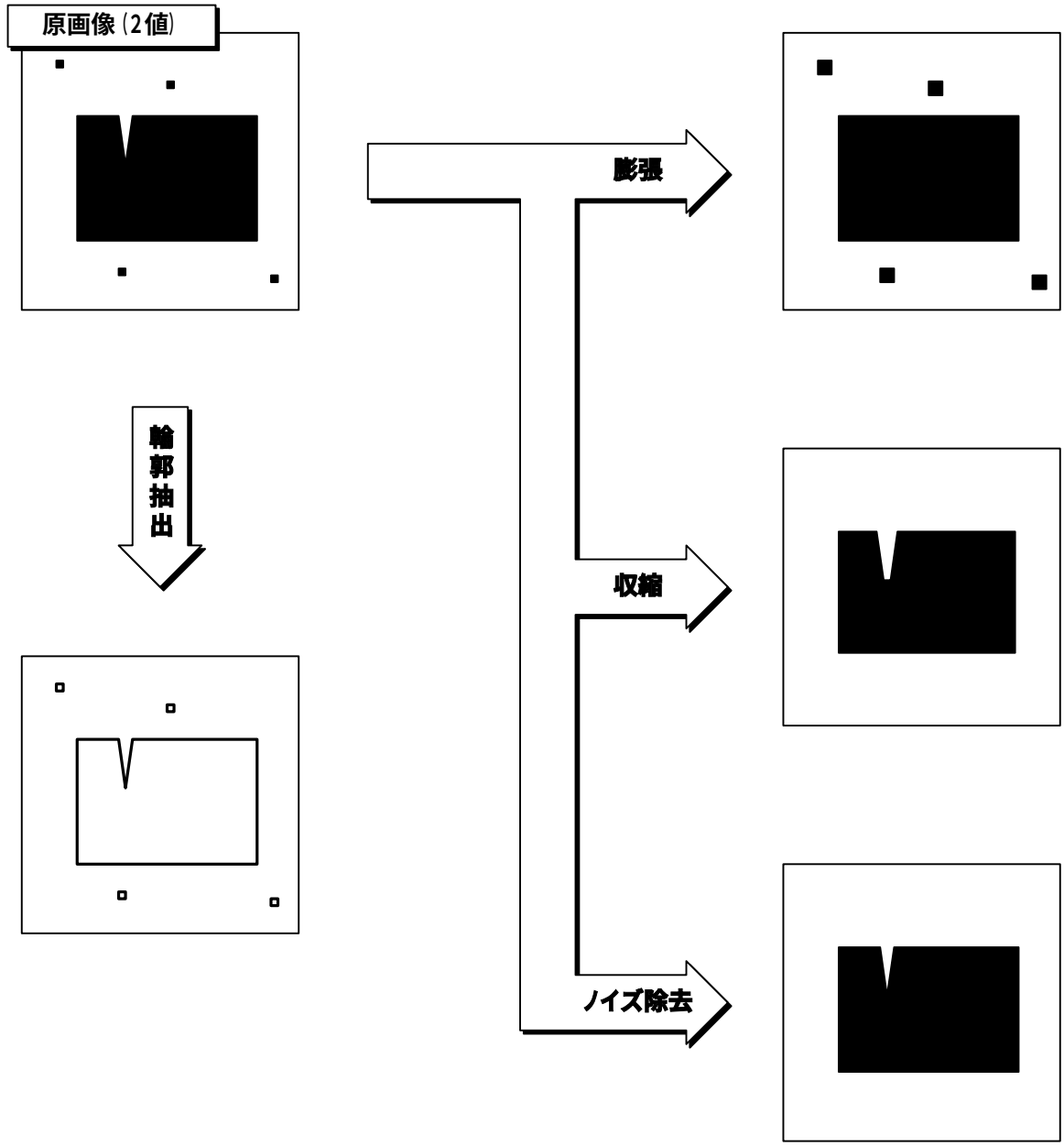


図５－２４　２値画像形状変換

5.11.7 コンボリューション

コンボリューションとは、近傍領域の積和演算のことです。３×３近傍の積和演算を画像処理プロセッサによりハード的演算することができます。

下図に示すように、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。

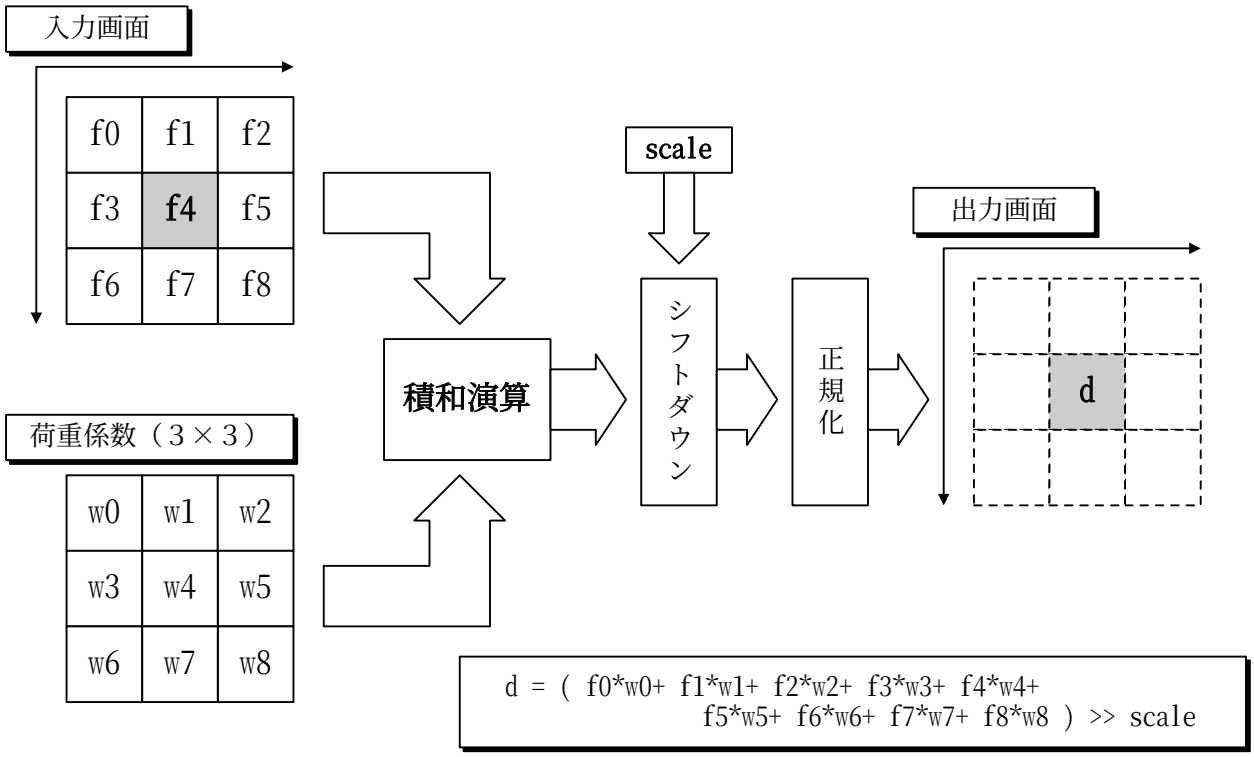


図５－２５　コンボリューション

この演算は、下表に示す荷重係数の設定により、濃淡画像での平滑化や輪郭強調などを行うことができます。

表５－７－１　荷重係数の例

種類・名称		荷重係数	scale	効果
平滑化	---	1 1 1 1 1 1 1 1 1	3	平滑化。画像の輪郭を滑らかにする。

表５－７－２　荷重係数の例

種類・名称		荷重係数	scale	効果
１次微分 グラディエント	微分	0 0 0 0 1 -1 0 0 0	0	輪郭強調（X方向）
		0 0 0 0 1 0 0 -1 0	0	輪郭強調（Y方向）
	Roberts	0 0 0 0 1 0 0 0 -1	0	輪郭強調（X方向）
		0 0 0 0 0 1 0 -1 0	0	輪郭強調（Y方向）
	Sobel	-1 0 1 -2 0 2 -1 0 1	0	輪郭強調（X方向）
		-1 -2 -1 0 0 0 1 2 1	0	輪郭強調（Y方向）
２次微分 ラプラシアン	４連結	0 -1 0 -1 4 -1 0 -1 0	0	輪郭強調
	８連結	-1 -1 -1 -1 8 -1 -1 -1 -1	0	輪郭強調

濃淡画像の輪郭強調には、１次微分と２次微分がありますが、１次微分はＸ／Ｙ方向それぞれ別の荷重係数です。画像処理プロセッサでは、１次微分の処理を１回で行う機能はありません。そこで１次微分を行う場合はＸ／Ｙ方向それぞれコンボリューションを行い、その画像を画像間算術演算でひとつにまとめます。

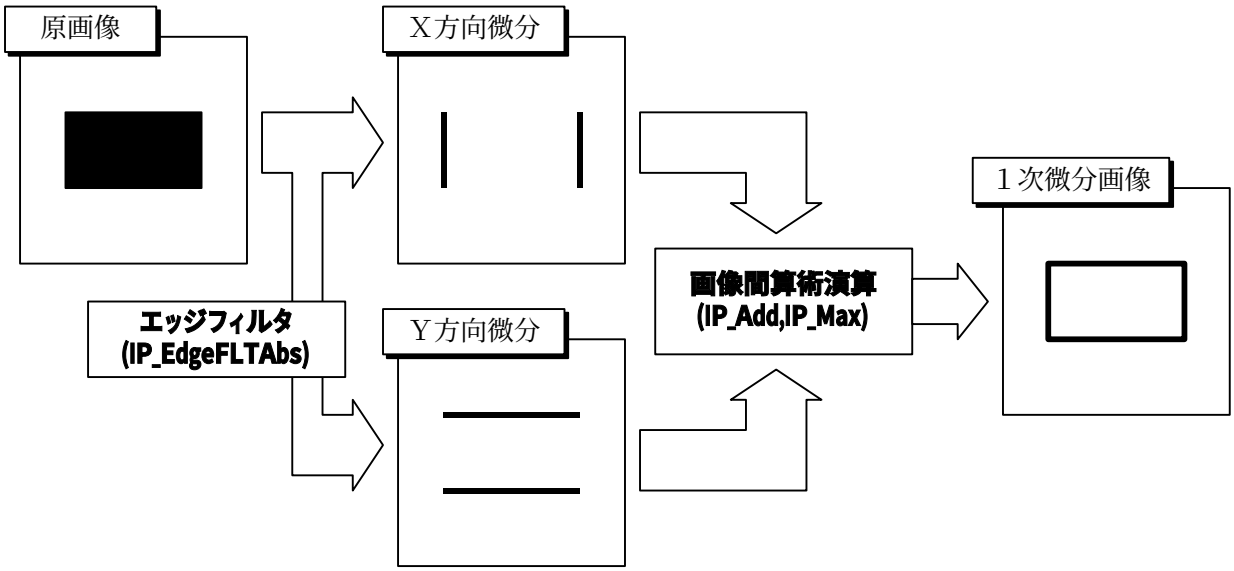


図５－２６　１次微分の方法

5.11.8 ランクフィルタ

ミニマムフィルタ、マックスフィルタ、メディアンフィルタなどをランクフィルタと呼びます。ランクフィルタ処理は、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近傍の局所領域内の画素データをソート（順に並べる）し、任意の順番の画素データを抽出します。また、３×３近傍の局所領域内で任意のカーネルマスクでマスク処理を行い、有効となった画素データ中でソートすることができます。

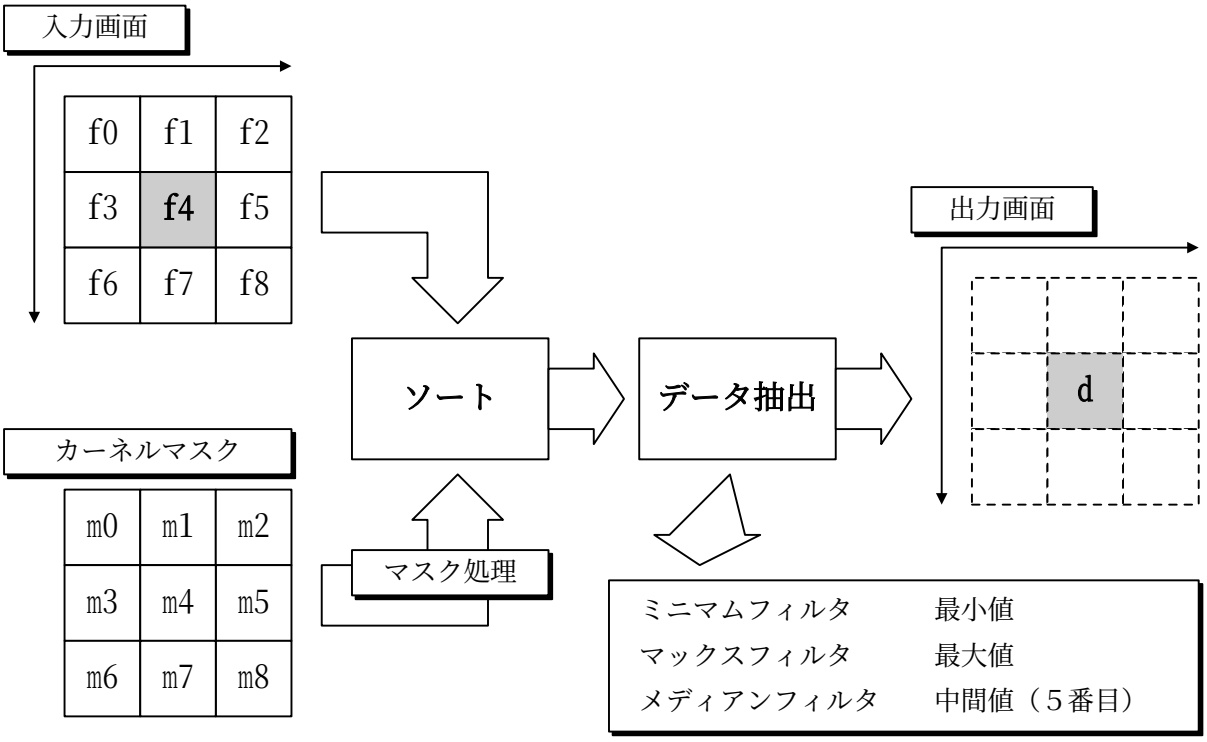
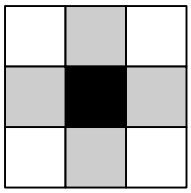
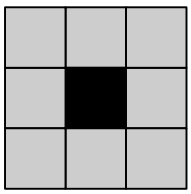


図５－２７　コンボリューション

5.11.9 ラベリング

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、異なった連結成分には異なった番号をつける処理です。また、連結成分を判定する方法に下表のように４連結と８連結があります。

表５－８　ラベリング連結方法

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

ラベリング処理のソース画面は２値画像を対象としていますが、ラベリングの処理結果は、ディスティネーション画面に連結成分毎の番号が濃度値として書込まれます。

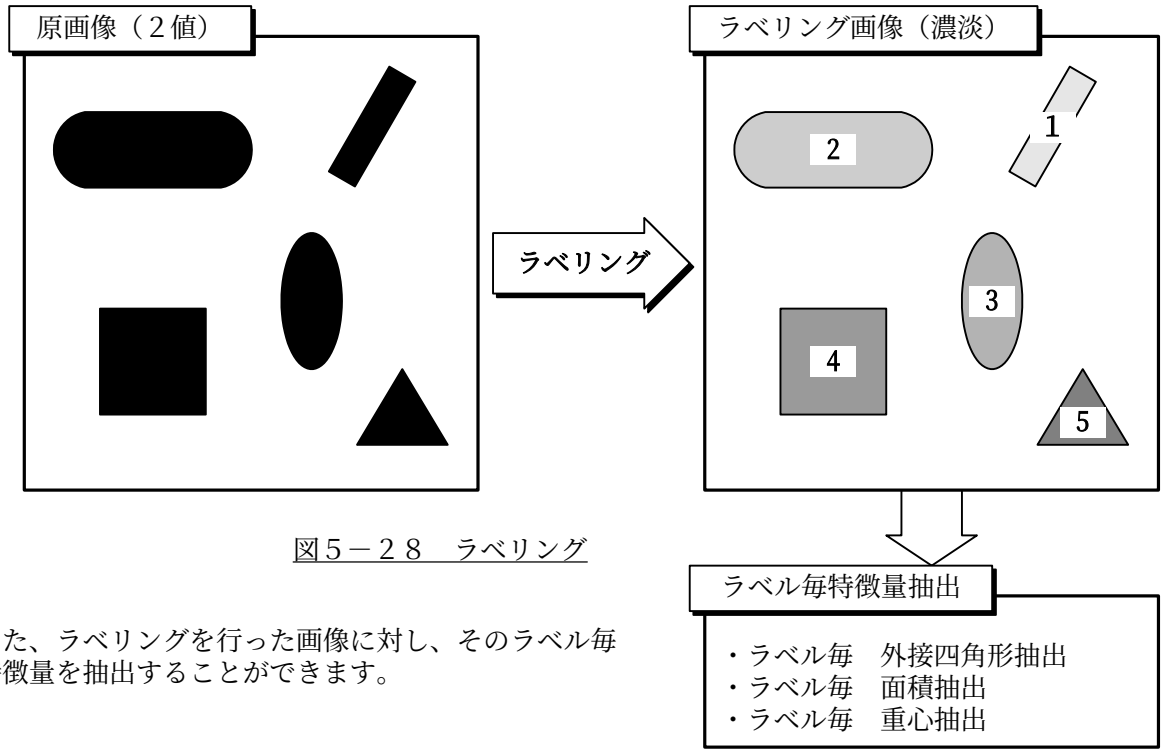


図５－２８　ラベリング

また、ラベリングを行った画像に対し、そのラベル毎の特徴量を抽出することができます。

ラベリング処理は、仮ラベル付け、合流対検出、真ラベル付けの３段階で行われます。
各々の処理には

・仮ラベル付け番号　：　最大　１０２２番
・合流対　　：　最大　１０２２件
・真ラベル付け　　：　最大　２５５個

という制約があり、この値を超えるとオーバーフローになり、正しい結果が得られません。ノイズの多い画像では前処理としてノイズ除去を行う必要があります。

5.11.10 濃淡画像特徴量抽出

濃淡画像に対し、濃度ヒストグラム、濃度累積値投影などの処理を画像処理プロセッサで高速に行います。

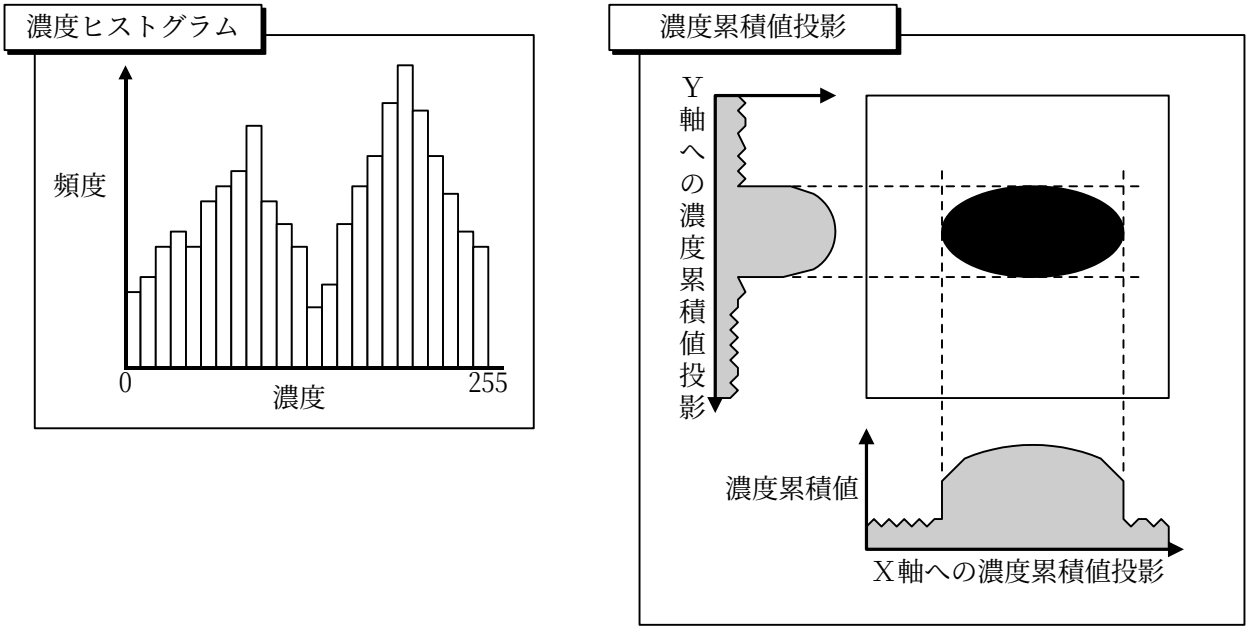


図５－２９　濃淡画像特徴量抽出

5.11.11 2値画像特徴量抽出

２値画像に対し、投影、領域抽出などの処理を画像処理プロセッサで高速に行います。

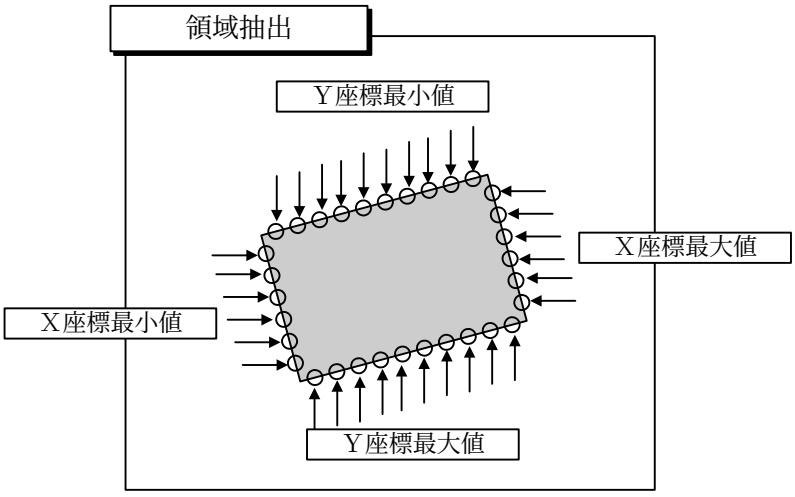


図５－３０　２値画像特徴量抽出

5.11.12 正規化相関サーチ

濃淡画像のテンプレートマッチングによるテンプレート画像サーチ（検索）の手法のひとつに正規化相関サーチがあります。

正規化相関サーチは、式(11-2)で示される相関演算をソース画像の指定領域内の全画素に対して行い相関係数（r）の値が最大（1.0）になるポイントをターゲットの位置として検出します。

$$r^2 = \frac{\left\{n\sum fg - \sum f \sum g\right\}^2}{\left\{n\sum f^2 - \left(\sum f\right)^2\right\}\left\{n\sum g^2 - \left(\sum g\right)^2\right\}} \qquad (11\text{-}2)$$

f ： ソース画像
g ： テンプレート画像
n ： テンプレート領域内有効画素数
（$1 < n \leqq 65536$：256×256相当）

この方法では、２値画像のパターンマッチングに比べ情報量は１２８×１２８のテンプレートの場合、２値画像では$32,\ 768$であるのに対し、正規化相関のパターンマッチングでは濃淡データが８ビットの場合は$4,\ 194,\ 304$と飛躍的に向上します。また、２値画像のパターンマッチングではサーチの結果がターゲット画像の明るさや２値化する時の閾値に大きく左右されますが正規化相関のパターンマッチングでは相関係数算出の計算式自体がデータを正規化しているという性質から明るさの変動に対して強い特徴あります。

正規化相関パターンマッチングの演算は、２値画像のパターンマッチングの演算に比べて計算が複雑なため、画像処理プロセッサによるハード演算とオンボードＣＰＵによる演算を組み合わせて処理しています。つまり、テンプレート画像とソース画像との間で式(11-3)～式(11-5)の積和演算を
画像処理プロセッサで行い、最後にテンプレート画像からあらかじめ計算しておいた式(11-6)～式(11-8)のデータを使用し、式(11-2)をオンボードＣＰＵで計算します。

積和演算

$$\sum fg \qquad (11\text{-}3)$$
$$\sum f \qquad (11\text{-}4)$$
$$\sum f^2 \qquad (11\text{-}5)$$

テンプレートデータ

$$\left(\sum g\right)^2 \qquad (11\text{-}6)$$
$$\sum g^2 \qquad (11\text{-}7)$$
$$n \qquad (11\text{-}8)$$

これらの演算は、テンプレート画像の領域（テンプレートカーネル）と対応するソース画像の一領域に対して行われます。ですから、サーチを行う場合は、テンプレートカーネルを移動しながら積和演算と相関の演算を行う必要があります。テンプレートカーネル内の積和演算と相関の演算を行う処理を１次走査、テンプレートカーネルを移動しながら最大の相関値を求める処理を２次走査と呼びます。

ソース画像の全領域でサーチを行う場合は、図５－３２のように１次走査をソース画像の全領域で繰り返し行います。

画像処理コマンドではこれらの動作をサーチコマンド内部で効率的に行います。

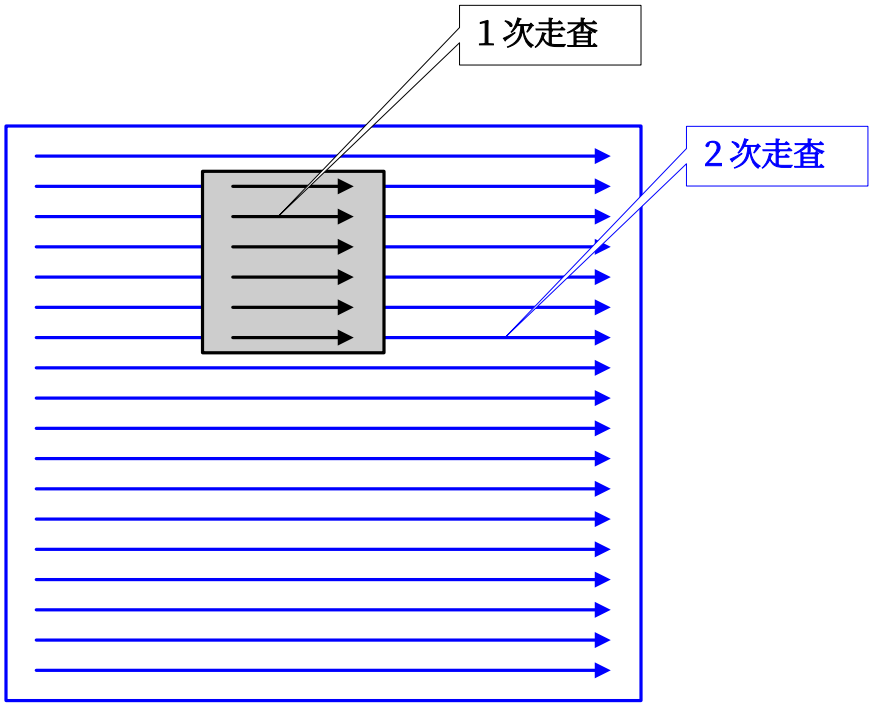


図５－３１　１次走査と２次走査

正規化相関サーチは、セットアップ、トレーニング、サーチという３つのステップで実行します。

セットアップとは、テンプレート画像を入力画像から切り出したり図形を描画したりしてテンプレート画像を作り出すことです。
トレーニングとは、セットアップで切り出した画像を正規化相関サーチのテンプレート画像として登録することです。
サーチとは、トレーニングで登録したテンプレートのサーチを行うことです。

下図にその簡単なフローを示します。

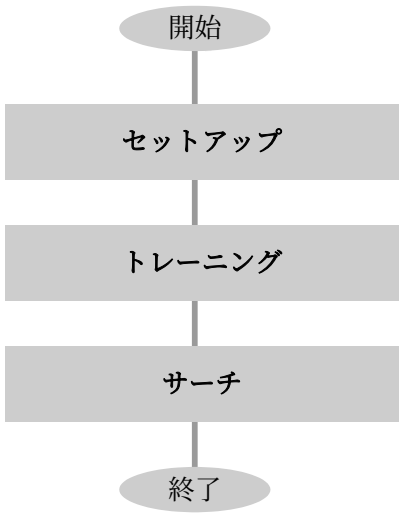


図５－３２　正規化相関サーチのフロー

5. 11. 13　直線抽出

画像処理コマンドでは、ハフ変換を用いて離散的な座標データから直線を抽出します。ハフ変換の演算はオンボードＣＰＵで高速に処理します。

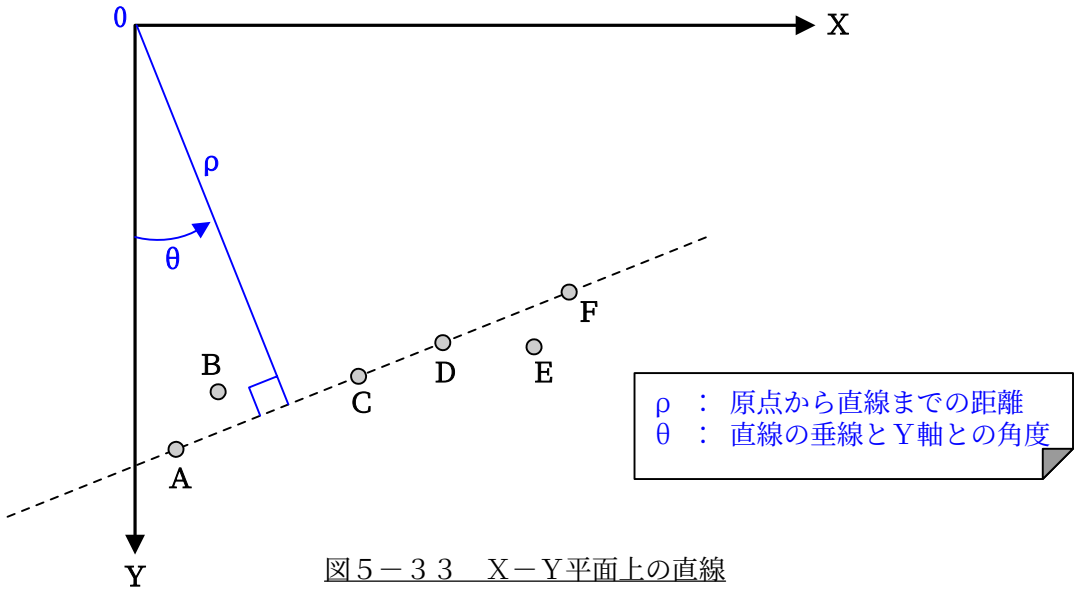


図５－３３　Ｘ－Ｙ平面上の直線

ハフ変換による直線抽出は、上図のようなＸ－Ｙ平面から式(11-9)によりρ－θ平面に変換し、ポイントＡ～Ｆそれぞれの正弦曲線を描き、その曲線が重なる度合いが一番大きいポイント（ρ_0, θ_0）を求め、そのポイントのρ，θ値の直線を理想の直線とする方法です。

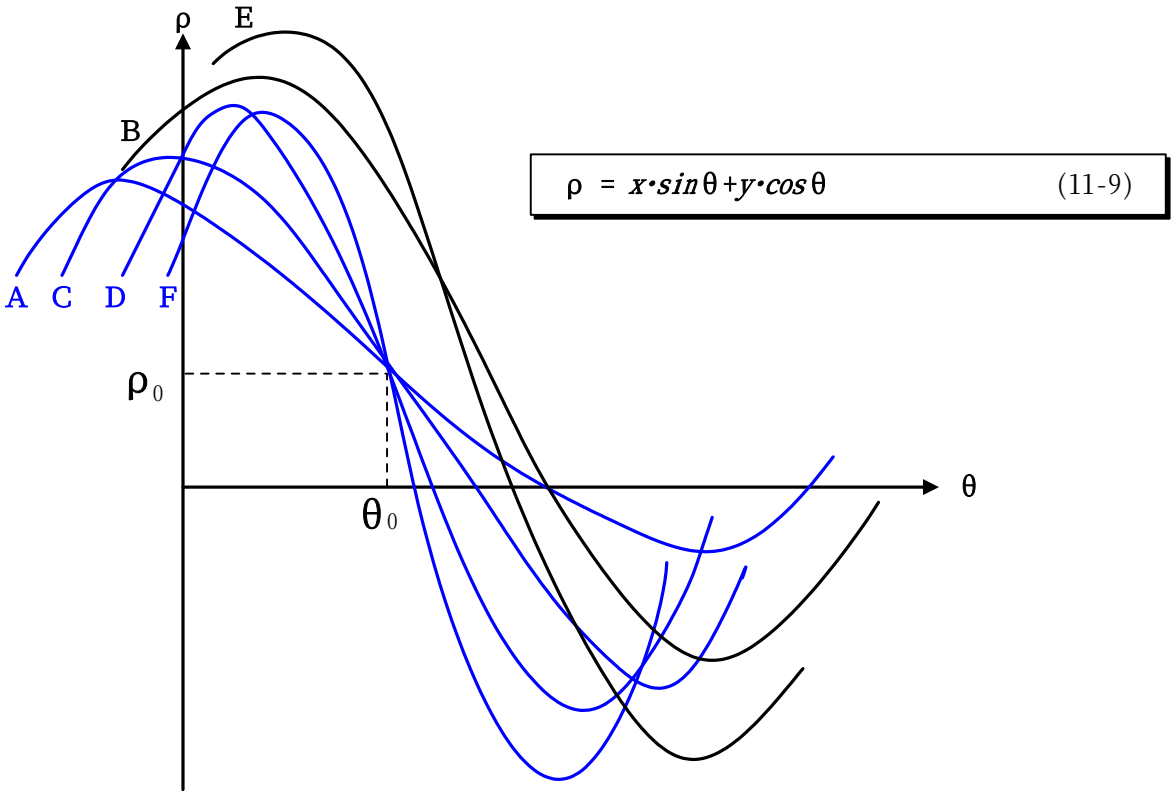


図５－３４　ρ－θ平面

この方法では、ポイントＢやＥといった直線からはずれているポイントが除外されるため、ノイズに影響され難い性質があります。欠点としては、最小自乗法などに比べるとθ値を変化させて繰返し処理しなければならず処理時間がかかりますが、オンボードＣＰＵと最適なアルゴリズムで高速に処理します。

5. 11. 14　イメージキャリパ

キャリパとはノギスのことで、画像処理により対象物のエッジを検出し、その対象物のエッジで平行なペアをサーチし、２つのエッジ間の距離やその中心位置を計測することができます。
イメージキャリパにより、以下のような集積回路（ＩＣ）パッケージのリード幅やリード間隔を求めることができます。

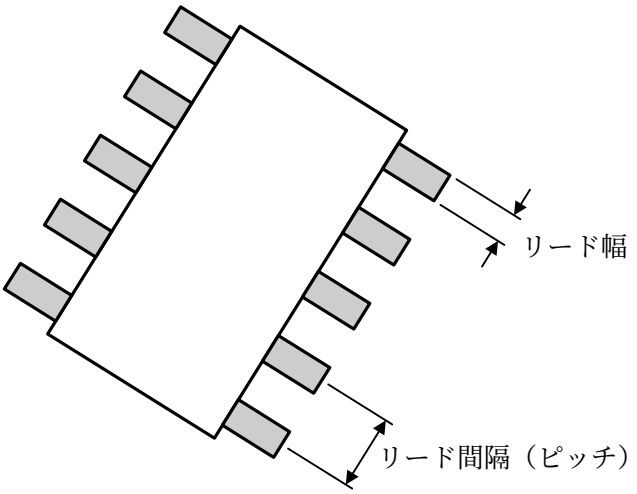


図５－３５　イメージキャリパによる寸法計測例

5. 11. 15　エッジファインダ

イメージキャリパでは画像処理により対象物のエッジを検出し、その対象物のエッジで平行なペアをサーチするため、エッジのペアがないパターンでは、エッジを抽出することはできません。そこで、エッジファインダではその対象物に含まれるあらゆるエッジを解析し、位置、ポラリティ、レベル（強さ）といった情報を抽出します。

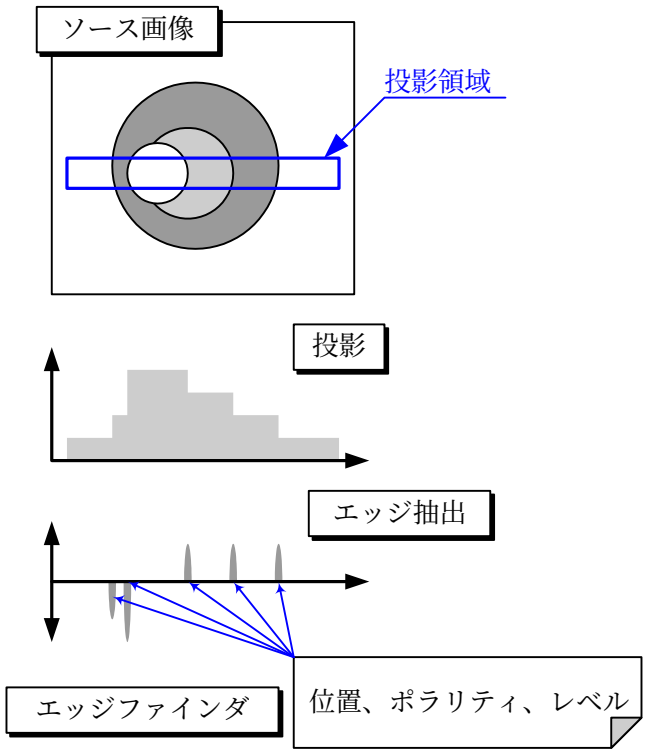


図５－３６　エッジファインダ

第６章

画像処理コマンド

６．１ ＰＣドライバセットアップコマンド

ＰＣからリモートで画像処理コマンドを使用し、アプリケーションを作成する場合やリモートで画像処理のデバッグを行う場合、ＰＣドライバによる画像処理コマンドを使用します。

６．１．１ ＰＣドライバのセットアップ

ＰＣドライバ画像処理コマンドを使用する場合、オープン、コマンドセットアップ、クローズの処理を図６－１の要領で行なう必要があります。

オープンでは、OpenIPDevExt()コマンドで指定するボード番号のボードのデバイスドライバの初期化とコマンドのセットアップ（「画像処理ボードのリセット」、「画像処理コマンドの実行ファイルのダウンロード」、「画像処理コマンドのブート」）を行います。このとき画像処理コマンドで使用するデバイスＩＤ（devID）を取得します。

クローズでは、画像処理ボードのデバイスドライバの終了処理を行います。セットアップを行ったアプリケーションが終了する場合、この処理を行います。
１枚のボードに複数のプロセスからアクセスする必要がある場合、オープンとクローズは、プロセス毎に行いそのデバイスＩＤで画像処理コマンドを実行します。しかし、セットアップは、１枚の画像処理ボードに対して１回だけ実行する必要があります。プロセス毎にセットアップを行うと、その都度画像処理コマンドのシステムが初期化されてしまいます。また、複数のプロセスから同時に画像処理コマンドが実行される場合は、排他制御を行い複数のプロセスから同時に画像処理コマンドが実行されることを避ける必要があります。
実際のコーディング例は、図４－１を参照して下さい。

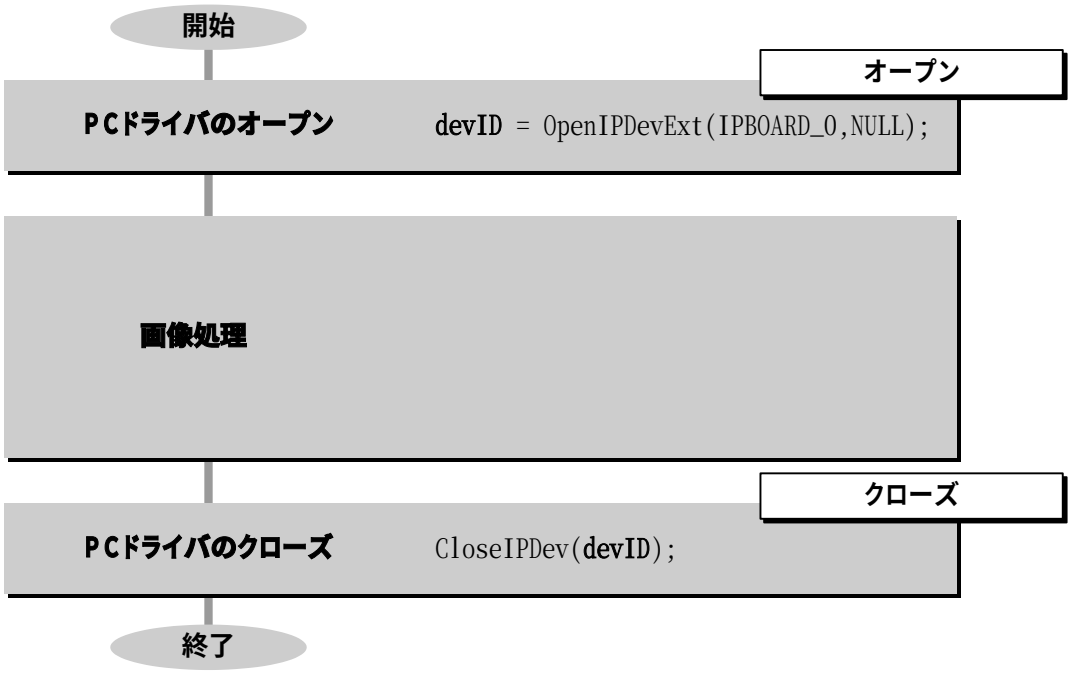


図６－１　画像処理ボードのセットアップ

6.1.2 PCドライバセットアップコマンド一覧

表６－１にＰＣドライバセットアップコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－１　ＰＣドライバセットアップコマンド一覧

コマンド名	機能	備考
OpenIPDevExt	ＰＣドライバのオープンとセットアップ	
OpenIPDev	ＰＣドライバのオープン	
CloseIPDev	ＰＣドライバのクローズ	
ResetIPSys	画像処理ボードのリセット	
LoadIPSys	画像処理コマンド実行ファイルのダウンロード	
BootIPSys	画像処理コマンドのブート	
GetIPDevNumber	ボード枚数の読み出し	
GetIPDevCount	オープンボード枚数の読み出し	
GetOpenIPDevCount	オープン回数の読み出し	
EnumAttachIPDev	装着ボードの列挙	
QueryAttachIPDev	装着ボードチェック	

6．2 コマンドエラー制御コマンド

画像処理コマンドでは、コマンド実行時のパラメータチェックや動作時のエラーを検出する機能を画像処理コマンドのシステムとして実装しています。
画像処理コマンドでは、コマンド実行後のリターンコードの値で正常終了か異常終了（エラー）かを判定します。通常の画像処理コマンドは、リターンコードが「０」の場合、正常終了で「－１」がエラーです。画像処理コマンドがエラーの場合、Ｗｉｎｄｏｗｓのメッセージボックスが出力されます。

6．2．1 コマンドエラー

画像処理コマンドでパラメータエラー等のエラーが発生時した場合、Ｗｉｎｄｏｗｓのメッセージボックスでユーザーにそれを知らせる機能があります。

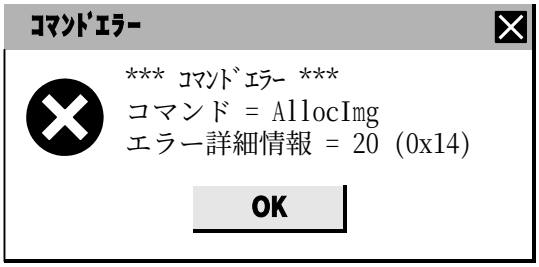


図６－２　コマンドエラー処理

6．2．2 エラー処理の制御

画像処理コマンドではシステムでエラーの制御を行っています。画像処理コマンドでは、一旦、画像処理コマンドエラーが発生すると引き続く画像処理コマンドで処理を実行しません。再び処理を再開するには ClearIPError コマンドで画像処理コマンドエラーをクリアして下さい。
また、画像処理コマンドでエラーが発生した際にエラーのメッセージボックスが表示されますが、EnableIPErrorMessage, DisableIPErrorMessageでその出力を制御することが可能です。

6.2.3 コマンドエラー制御コマンド一覧

表６－２にコマンドエラー制御コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－２　コマンドエラー制御コマンド一覧

コマンド名	機能	備考
ClearIPError	コマンドエラーのクリア	
CheckIPError	コマンドエラーのチェック	
ReadIPErrorTable	コマンドエラー情報の取得	
EnableIPErrorMessage	エラーメッセージの出力の許可	
DisableIPErrorMessage	エラーメッセージの出力の禁止	

6.3 システム制御コマンド

システム制御コマンドは、画像処理システムの初期化、画像処理の制御、画像処理のサポートを行うコマンドです。

6.3.1 システムの初期化

電源投入時は、画像処理コマンドが初期化されていません。アプリケーションの初期化部でInitIP()コマンドを実行し、画像処理コマンドの初期化を行って下さい。表６－３にInitIP()コマンドを実行した時のシステムの初期状態を示します。また、vpxInitIP()コマンドでは、種々のオプションを指定して初期化ができます。詳細は、コマンドリファレンス「ＶＰ－８１０互換コマンド」のvpxInitIP()を参照して下さい。

表６－３　システムの初期状態

	設定項目	設定内容
映像入力表示関連	走査方式	インタレース
	映像入力サイズ	５１２H×４８０V
	カメラポート番号	ポート＃０
	カメラタイプ	SVP-330 ： ＮＴＳＣカラーカメラ
	システムデータタイプ	符号なし８ビット
	映像同期信号	カメラ同期
	カメラトリガ出力	ＯＦＦ
	表示	カメラ入力映像の表示 （ループ入力・ループ表示）
画像制御関連	ウィンドウ	有効（５１２（X）×４８０（Y））

6.3.2 システム制御コマンド一覧

表６－４にシステム制御コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－４　システム制御コマンド一覧

コマンド名	機能	備考
InitIP	画像処理システムの初期化	
InitIPExt	画像処理システムの初期化（ハードウェアテスト付き）	
GetIPVersionInfo	バージョン情報の取得	
CheckIPVersion	バージョンチェック	
SetIPDataType	画像処理システムデータタイプの設定	
GetDispImgID	ディスプレイ画面の画面番号の取得	
GetBitmapImgID	ビットマップ画面の画面番号の取得	
ISP_BusyWait	画像処理プロセッサのビジーウェイト	
ImgBusyWait	処理画面に対するビジーウェイト	
Img2chBusyWait	２ｃｈ処理画面に対するビジーウェイト	
ISP_BusyCheck	ビジーステータスの読み出し	

6．4 画像メモリ領域管理コマンド

画像メモリ領域管理コマンドは、画像メモリ領域確保／開放、ウィンドウの制御を行うコマンドです。

画像処理ボードは、１６Mバイトの画像メモリを実装しています。カメラ映像の入力や映像表示、画像処理を行う場合には、前もって必要なサイズの画面を確保する必要があります。
ウィンドウとは、処理領域のことであり、処理を画面の一部に限定して行う場合に使用します。画像メモリ管理、ウィンドウ制御については、５章で説明していますので詳細はそちらを参照して下さい。

6．4．1 画像メモリ領域管理コマンド一覧

表６－５に画像メモリ領域管理コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－５　画像メモリ領域管理コマンド一覧

コマンド名	機能	備考
AllocImg	画像メモリ領域確保	
AllocLockImg	画像メモリ領域確保（固定領域）	
FreeImg	画像メモリ領域開放（指定画面）	
FreeAllImg	画像メモリ領域開放（全画面）	
ReadImgTable	画像メモリ管理テーブル読出し	
ChangeImgDataType	画面データタイプ変更	
SetWindow	ウィンドウ設定	
SetAllWindow	ウィンドウ設定（全種）	
ResetAllWindow	ウィンドウリセット	
EnableIPWindow	ウィンドウ有効化	
DisableIPWindow	ウィンドウ無効化	
ReadWindow	ウィンドウ設定座標読出し	
AllocDispImg	ディスプレイ画面の確保	
FreeDispImg	ディスプレイ画面の開放	

6．5 映像入力コマンド

映像入力コマンドは、カメラからの映像入力、映像入力のサポートを行うコマンドです。

画像処理ボードは、ＮＴＳＣ標準のモノクロカメラをはじめ、プログレッシブカメラ、フレームシャッターカメラ、高速カメラ、高精細カメラの映像入力が可能です。
映像の入力については、５章で説明していますので詳細はそちらを参照して下さい。
ＳＶＰ－３３０はＮＴＳＣカメラのみサポートしています。

6．5．1 NTSC標準カメラ映像の入力方法

カメラ映像を入力するには、まず、映像入力の前準備として
① SetVideoFrame コマンドで映像入力画面の設定を行う
② SelectCamera コマンドでカメラ番号とカメラタイプを選択する
を行います。その後
③ GetCamera コマンドで映像を指定した画像メモリに入力する
を行います。
なお、画像処理ボードでは、電源投入、ＰＣドライバのセットアップ、InitIP コマンド実行後は SetVideoFrame , SelectCamera コマンドで
・カメラ同期信号 ： 画像処理ボードからの外部同期
・フレームサイズ ： ５１２×４８０
・インタレースモード ： インタレース
・カメラ番号 ： ０
・カメラタイプ ： ＮＴＳＣカラーカメラ（YUV_CAMERA）
に設定されています。

6．5．2 GetCameraWithSelectPort コマンドによるNTSC標準カメラ映像の入力について

GetCameraWithSelectPort コマンドは SelectCamera コマンドと GetCamera コマンドを組み合わせたコマンドです。複数のカメラを切り替えて使用する時、多くの場合、カメラ同期の同期信号を外部同期で動作させて最小の切換え時間にしたいと考えるでしょう。ユーザのアプリケーションでは、SelectCamera コマンドの後に GetCamera コマンドを実行する処理を繰り返すようにするはずです。しかし、ＰＣドライバでそのようにプログラムを作成するとうまく動作しない（取込みが遅れる）場合があります。実は、SelectCamera コマンドはカメラの同期信号を監視して GetCamera コマンドで映像を入力するタイミングを決めているため、SelectCamera コマンドと GetCamera コマンドの間にドライバとコマンドコールのオーバヘッドが生じると最悪、１フレーム取込みが遅れることになるのです。そこで、SelectCamera コマンドと GetCamera コマンドをオンボードＣＰＵ側で実行すればオーバヘッドは「０」になります。

6．5．3　NTSC標準カメラ以外のカメラ映像の入力方法

ＮＴＳＣ標準カメラ以外のカメラ、プログレッシブカメラ、フレームシャッタカメラ、高速カメラ、高精細カメラの映像を入力するには、まず、映像入力の前準備として

① SelectCamera コマンドでカメラ番号とカメラタイプを選択する

を行います。その後

② GetCamera コマンドで映像を指定した画像メモリに入力する

を行います。

SelectCamera コマンドでＮＴＳＣ標準カメラ以外のカメラ選択すると本来、SetCameraSync ，SetVideoFrameコマンドで設定するべき値が自動的に以下のように設定されます。

・カメラ同期信号　　：　カメラ同期

・インタレースモード　：　ノンインタレース

また、フレームサイズは、高精細以外は SetVideoFrame コマンドで最後に設定された大きさになります。電源投入、ＰＣドライバのセットアップ、InitIP コマンド実行後のフレームサイズは、５１２×４８０です。高精細カメラでは、カメラタイプにより固有の大きさに設定されます。

なお、ＳＶＰ－３３０ではＮＴＳＣ以外のカメラ映像入力はできません。

6．5．4　フレームシャッタを使用するカメラ映像の入力方法

フレームシャッターをはじめとしてプログレッシブカメラ、フレームシャッタカメラ、高速カメラ、高精細カメラには多くの機種でフレームシャッタでの映像入力機能があります。フレームシャッタで映像入力するには、まず、映像入力の前準備として

① SelectCamera コマンドでカメラ番号とカメラタイプを選択する

② SetTrigerMode コマンドでトリガーモードに設定する

③ SetShutterSpeed コマンドでシャッタスピードを設定する（対応しているカメラのみ）

を行います。その後

④ GetCamera コマンドで映像を指定した画像メモリに入力する

を行います。②でのシャッタスピードの設定は対応しているカメラのみ設定して下さい。対応していないカメラのシャッタスピードは、カメラ本体で任意の値に設定して下さい。詳細は、付録の対応カメラ一覧と各カメラの付属マニュアルを参照して下さい。

SetTrigerMode コマンドでトリガーモードを選択すると多くカメラではカメラのスルー映像が出力されなくなるので注意して下さい。

なお、ＳＶＰ－３３０ではフレームシャッタ機能は使用できません。

6．5．5　高精細カメラの映像入力方法

高精細カメラの映像入力は、前項（６．５．３項及び６．５．４項）と同じです。ただし、高精細カメラは、画面サイズが１０２４×１０２４（カメラによって異なる）と標準のカメラより大きいので、入力する画面も１０２４×１０２４のサイズで画像メモリの領域をあらかじめ確保する必要があります。入力する画面の大きさがそれよりも小さいと画面サイズエラーになります。以下に、高精細カメラの映像画面設定と映像入力のサンプルを示します。

なお、ＳＶＰ－３３０では高精細カメラの映像入力はできません。

```
// 高精細カメラ映像入力画面設定
FreeDispImg(devID);
AllocDispImg(devID,IMG_FS_1024H_1024V,BW_DISPLAY);
SetVideoFrame(devID,NONINTERLACE, VIDEO_FS_1024H_1024V);

// 高精細カメラポート選択
SelectCamera(devID,CAMERA_PORT0,TELI_CS3910);

// 高精細カメラ映像入力
imgid= AllocImg(devID,IMG_FS_1024H_1024V);
GetCamera(devID,imgid);

// 高精細画像表示
sx= 0; sy= 0; xmag= 2; ymag= 2;
SetDispWindow(devID,sx,sy,xmag,ymag);
EnableHIREZDisp(devID);
DispImg(devID,imgid);
```

6．5．6 カメラ切換時のウェイト

ＳＶＰ－３３０には２チャネルのカメラを接続することができます。SelectCamera コマンドによりそれらに接続したカメラから１つをカメラ番号（カメラチャネル）とカメラタイプで選択することができます。ＳＶＰ－３３０に実装されている２つのポート切り替えは独立したビデオプロセッサ（ＶＰ）で行われているため、SelectCamera コマンドとGetCamera コマンドとの間にウェイトを挿入する必要はありません。

6．5．7 映像入力コマンド一覧

表６－６に映像入力コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

ＳＶＰ－３３０は、４カメラ映像の同時入力をサポートしていません。また、ＳＶＰ－３３０はランダムトリガ機能をサポートしていません。

表６－６　映像入力コマンド一覧

コマンド名	機能	備考
SetVideoFrame	映像入力画面設定	
SelectCamera	カメラ番号とカメラタイプの選択	
GetCamera	カメラ映像入力	
GetCameraWithSelectCamera	カメラ映像入力（カメラ番号とカメラタイプの選択付き）	
SetCameraSync	カメラ同期信号の設定	
SetVFDelay	映像入力画面の遅延サイズ設定	
SetShutterSpeed	カメラ映像入力シャッタスピードの設定	
Get2Camera	２カメラ映像の同時入力	
SetTrigerMode	カメラ映像入力トリガモードの設定	
GetCameraSts	カメラ接続のチェック	
Get4Camera	４カメラ映像の同時入力	
ResetCamera	リセットカメラ	

6.6 NTSCモニタへの映像表示コマンド

ＮＴＳＣモニタへの映像出力コマンドは、カメラ映像の表示、画像メモリの表示、ビットマップ画面のオーバレイ表示、映像表示のサポートを行うコマンドです。

映像表示については、５章で説明していますので詳細はそちらを参照して下さい。

6.6.1 カメラ映像の表示

ＮＴＳＣモニタへのカメラ映像の表示は、DispCamera コマンドにより行います。
原理的には、カメラの映像がそのままモニタへ表示されるわけですが、画像処理ボードは画像メモリに対してループ映像入力、ループ表示を行っています。

6.6.2 画像メモリの表示

ＮＴＳＣモニタへの画像メモリの表示は、DispImg コマンドにより行います。
指定された画面のループ表示を行っています。

6.6.3 ループ映像入力･ループ表示の終了

カメラ映像の表示や画像メモリの表示を行っている間は、常にループ映像入力、ループ表示が行われています。これらのループ処理の終了は、NoDisp コマンドにより行います。
NoDisp コマンドはループ映像入力とループ表示の両方を終了します。

6.6.4 ビットマップ画面のオーバレイ表示

画像処理の処理結果などの表示として、画面に文字や線を描画したい場合があります。BitmapOverlapコマンドにより、ビットマップ画面をカメラ映像表示や画像メモリ表示をモニタ画面にオーバレイ表示することができます。なお、カラー映像にはオーバーレイ表示することはできません。

6.6.5 映像表示コマンド一覧

表６－７に映像表示コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－７　映像表示コマンド一覧

コマンド名	機能	備考
DispCamera	カメラ映像表示	
DispImg	画像メモリ表示	
NoDisp	表示終了	
BitmapOverlap	オーバレイ表示制御	
DispOverlap	画像メモリのオーバレイ表示	
SetDFDlay	映像表示画面の遅延サイズ設定	
SetDispFrame	映像表示画面設定	
SelectDisp	映像出力形式の選択	
SetDispWindow	高精細映像表示のウィンドウ設定	
EnableHIREZDisp	高精細映像表示の有効化	
DisableHIREZDisp	高精細映像表示の無効化	

6.7 画像クリアコマンド

画像クリアコマンドは、画像メモリの「0」クリア、任意定数の発生を行うコマンドです。
処理はすべてハードウェアで処理されます。

6.7.1 ウィンドウの設定

IP_ClearAllImg , IP_ClearCHImg , IP_ClearImg コマンドはウィンドウを設定できません。
IP_Const コマンドは任意のウィンドウが設定できます。設定するウィンドウの種類は
ＤＳＴ＿ＷＩＮ（デスティネーション）ウィンドウです。

6.7.2 画面データタイプ

画像クリアコマンド実行後の画面は、すべてシステムデータタイプに設定されます。

6.7.3 画像クリアコマンド一覧

表６－８に画像クリアコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－８　画像クリアコマンド一覧

コマンド名	機能	備考
IP_ClearAllImg	画像メモリクリア（全チャネル）	
IP_ClearCHImg	画像メモリクリア（指定チャネル）	
IP_ClearImg	画像メモリクリア（指定画面）	
IP_Const	定数発生	

6．8 画像転送･アフィン変換コマンド

画像転送・アフィン変換コマンドは、画像メモリ間の転送、拡大／縮小、アフィン変換を行うコマンドです。

画像処理については、５章で説明していますので詳細はそちらを参照して下さい。

6.8.1 ウィンドウの設定

画像転送・アフィン変換コマンドは任意のウィンドウが設定できます。設定するウィンドウの種類はソース画面は、ＳＲＣ０＿ＷＩＮ（ソース０）、デスティネーション画面は、ＤＳＴ＿ＷＩＮ（デスティネーション）ウィンドウです。

6.8.2 画面データタイプ

画像転送・アフィン変換コマンド実行後のデスティネーション画面は、すべてソース画面１のデータタイプに設定されます。

6.8.3 画像転送･アフィン変換コマンド一覧

表６－９に画像転送・アフィン変換コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－９　画像転送・アフィン変換コマンド一覧

コマンド名	機能	備考
IP_Copy	転送	
IP_Zoom	ズーム	
IP_ZoomExt	ズーム（縦横任意倍率）	
IP_ZoomOut	ズームアウト	
IP_ZoomOutExt	ズームアウト（縦横任意倍率）	
IP_Shift	シフト	
IP_ZoomS	シフト付きズーム	
IP_Rotate	回転	
IP_ZoomIn	ズームイン	
IP_ZoomInExt	ズームイン（縦横任意倍率）	

6.9 2値化コマンド

２値化コマンドは、濃淡画像の２値化を行うコマンドです。
処理はすべてハードウェアで処理されます。

画像処理については、５章で説明していますので詳細はそちらを参照して下さい。

6.9.1 ウィンドウの設定

２値化コマンドは任意のウィンドウが設定できます。設定するウィンドウの種類はソース画面は、ＳＲＣ０_ＷＩＮ（ソース０）、デスティネーション画面は、ＤＳＴ_ＷＩＮ（デスティネーション）ウィンドウです。

6.9.2 画面データタイプ

２値化を行う画面（ソース画面）の画面データタイプは符号なし８ビットである必要があります。それ以外では正常に動作しない場合があるので注意が必要です。また、２値化コマンド実行後のデスティネーション画面は、すべて２値のデータタイプに設定されます。

6.9.3 2値化コマンド一覧

表６－１０に２値化コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－１０　２値化コマンド一覧

コマンド名	機能	備考
IP_Binarize	２値化	
IP_BinarizeExt	範囲・反転付き２値化	

6.10 画素変換コマンド

画素変換コマンドは、濃淡画像ソース画面に対して画素変換を行うコマンドです。
処理はすべてハードウェアで処理されます。

画像処理については、５章で説明していますので詳細はそちらを参照して下さい。

6.10.1 ウィンドウの設定

画素変換コマンドは任意のウィンドウが設定できます。設定するウィンドウの種類はソース画面は、ＳＲＣ０＿ＷＩＮ（ソース０）、デスティネーション画面は、ＤＳＴ＿ＷＩＮ（デスティネーション）ウィンドウです。

6.10.2 画面データタイプ

画素変換コマンド実行後のデスティネーション画面は、すべてシステムデータタイプに設定されます。

6.10.3 画素変換コマンド一覧

表６－１１に画素変換コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－１１　画素変換コマンド一覧

コマンド名	機能	備考
IP_Invert	論理反転	
IP_Minus	符号反転	
IP_Abs	絶対値	
IP_AddConst	定数加算	
IP_SubConst	定数減算	
IP_SubConstAbs	定数減算絶対値	
IP_MultConst	定数乗算	
IP_MinConst	定数比較 Ｍｉｎ	
IP_MaxConst	定数比較 Ｍａｘ	
WriteConvertLUT	濃度変換データ設定	
IP_ConvertLUT	濃度変換	
IP_ShiftDown	定数シフトダウン	
IP_ShiftUp	定数シフトアップ	
IP_AndConst	定数論理積	

6.11 画像間算術演算コマンド

画像間算術演算コマンドは、２つの濃淡画像画面間で算術演算を行うコマンドです。
処理はすべてハードウェアで処理されます。

画像処理については、５章で説明していますので詳細はそちらを参照して下さい。

6.11.1 ウィンドウの設定

画像間算術演算コマンドは任意のウィンドウが設定できます。設定するウィンドウの種類は、ソース画面は2画面あり、ＳＲＣ０_ＷＩＮ（ソース０）とＳＲＣ１_ＷＩＮ（ソース１）です。デスティネーション画面は、ＤＳＴ_ＷＩＮ（デスティネーション）ウィンドウです。

6.11.2 画面データタイプ

画像間算術演算コマンド実行後のデスティネーション画面は、すべてシステムデータタイプに設定されます。

6.11.3 画像間算術演算コマンド一覧

表６－１２に画像間算術演算コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－１２　画像間算術演算コマンド一覧

コマンド名	機能	備考
IP_Add	加算	
IP_Sub	減算	
IP_SubAbs	減算絶対値	
IP_Comb	係数付加算	
IP_CombAbs	係数付加算絶対値	
IP_Mult	乗算	
IP_Average	平均	
IP_Min	比較 Ｍｉｎ	
IP_Max	比較 Ｍａｘ	
IP_SubConstAbsAdd	定数減算絶対値和	
IP_SubConstMultAdd	定数減算自乗和	
IP_SubConstMult	定数減算積	
IP_CombDrop	係数付加算（切り捨て）	

6.12 画像間論理演算コマンド

画像間論理演算コマンドは、２つの画面間で論理演算を行うコマンドです。
処理はすべてハードウェアで処理されます。

画像処理については、５章で説明していますので詳細はそちらを参照して下さい。

6.12.1 ウィンドウの設定

画像間論理演算コマンドは任意のウィンドウが設定できます。設定するウィンドウの種類は、ソース画面は2画面あり、ＳＲＣ０_ＷＩＮ（ソース０）とＳＲＣ１_ＷＩＮ（ソース１）です。デスティネーション画面は、ＤＳＴ_ＷＩＮ（デスティネーション）ウィンドウです。

6.12.2 画面データタイプ

画像間論理演算コマンド実行後のデスティネーション画面は、すべてソース画面１のデータタイプに設定されます。

6.12.3 画像間論理演算コマンド一覧

表６－１３に画像間論理演算コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－１３　画像間論理演算コマンド一覧

コマンド名	機能	備考
IP_And	論理積	
IP_0r	論理和	
IP_Xor	排他的論理和	
IP_InvertAnd	否定論理積	
IP_Invert0r	否定論理和	
IP_Xnor	排他的否定論理和	

6.13 2値画像形状変換コマンド

２値画像形状変換コマンドは、２値のソース画面に対しノイズ除去、輪郭抽出、膨張、収縮等の２値物体の形状の変更を行うコマンドです。
処理はすべてハードウェアで処理されます。

画像処理については、５章で説明していますので詳細はそちらを参照して下さい。

6.13.1 ウィンドウの設定

２値画像形状変換コマンドは任意のウィンドウが設定できます。設定するウィンドウの種類はソース画面はＳＲＣ０_ＷＩＮ（ソース０）でデスティネーション画面は、ＤＳＴ_ＷＩＮ（デスティネーション）ウィンドウです。

6.13.2 画面データタイプ

２値画像形状変換を行う画面（ソース画面）の画面データタイプは２値で、「０」または「２５５」のデータのみの画像データである必要があります。それ以外では正常に動作しない場合があるので注意が必要です。また、２値画像形状変換コマンド実行後のデスティネーション画面は、すべて２値のデータタイプに設定されます。

6.13.3 2値画像形状変換コマンド一覧

表６－１４に２値画像形状変換コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－１４　２値画像形状変換コマンド一覧

コマンド名	機能	備考
IP_PickNoise4	２値画像ノイズ除去（４連結）	
IP_PickNoise8	２値画像ノイズ除去（８連結）	
IP_Outline4	２値画像輪郭抽出（４連結）	
IP_Outline8	２値画像輪郭抽出（８連結）	
IP_Dilation4	２値画像膨張（４連結）	
IP_Dilation8	２値画像膨張（８連結）	
IP_Erosion4	２値画像収縮（４連結）	
IP_Erosion8	２値画像収縮（８連結）	
IP_Shrink4	２値画像細線化（４連結）	
IP_Shrink8	２値画像細線化（８連結）	
IP_Thin4	２値画像縮退化（４連結）	
IP_Thin8	２値画像縮退化（８連結）	

6.14 コンボリューションコマンド

コンボリューションコマンドは、ソース画面の指定領域内の画素に対し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。この荷重係数を適切に選択することで平滑化、輪郭強調といった処理を行います。また、１×９近傍の積和演算（ラインフィルタ）もサポートしています。
処理はすべてハードウェアで処理されます。

画像処理については、５章で説明していますので詳細はそちらを参照して下さい。

6.14.1 ウィンドウの設定

コンボリューションコマンドは任意のウィンドウが設定できます。設定するウィンドウの種類はソース画面はＳＲＣ０＿ＷＩＮ（ソース０）でデスティネーション画面は、ＤＳＴ＿ＷＩＮ（デスティネーション）ウィンドウです。

6.14.2 画面データタイプ

コンボリューションを行う画面（ソース画面）の画面データタイプは符号なし８ビットか符号付き８ビットです。符号なし８ビットか符号付き８ビットで処理結果が異なる場合があるので注意が必要です。またコンボリューションコマンド実行後のデスティネーション画面は、すべてシステムデータタイプに設定されます。

6.14.3 コンボリューションコマンド一覧

表６－１５にコンボリューションコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－１５　コンボリューションコマンド一覧

コマンド名	機能	備考
IP_SmoothFLT	平滑化	
IP_EdgeFLT	濃淡画像輪郭強調	
IP_EdgeFLTAbs	濃淡画像輪郭強調（絶対値）	
IP_Lapl4FLT	ラプラシアン（４連結）	
IP_Lapl8FLT	ラプラシアン（８連結）	
IP_Lapl4FLTAbs	ラプラシアン（４連結・絶対値）	
IP_Lapl8FLTAbs	ラプラシアン（８連結・絶対値）	
IP_LineFLT	ラインフィルタ	
IP_LineFLTAbs	ラインフィルタ（絶対値）	

6.15 ミニ／マックスフィルタコマンド

ミニ／マックスフィルタ（局所最大値／最小値フィルタ）コマンドは、ソース画面の指定領域内の画素に対し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カーネルマスクで処理し、有効となった画素の中から最大値または最小値の画素の値の出力を行います。周辺では、中央の画素をそのまま出力します。また、１×９近傍での処理もサポートしています。
処理はすべてハードウェアで処理されます。

画像処理については、５章で説明していますので詳細はそちらを参照して下さい。

6.15.1 ウィンドウの設定

ミニ／マックスフィルタコマンドは任意のウィンドウが設定できます。設定するウィンドウの種類はソース画面はＳＲＣ０_ＷＩＮ（ソース０）でデスティネーション画面は、ＤＳＴ_ＷＩＮ（デスティネーション）ウィンドウです。

6.15.2 画面データタイプ

ミニ／マックスフィルタを行う画面（ソース画面）の画面データタイプは符号なし８ビットか符号付き８ビットです。符号なし８ビットか符号付き８ビットで処理結果が異なる場合があるので注意が必要です。また、ミニ／マックスフィルタコマンド実行後のデスティネーション画面は、すべてシステムデータタイプに設定されます。

6.15.3 ミニ／マックスフィルタコマンド一覧

表６－１６にミニ／マックスフィルタコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－１６　ミニ／マックスフィルタコマンド一覧

コマンド名	機能	備考
IP_MinFLT	局所最小値フィルタ	
IP_MinFLT4	局所最小値フィルタ（４連結）	
IP_MinFLT8	局所最小値フィルタ（８連結）	
IP_MaxFLT	局所最大値フィルタ	
IP_MaxFLT4	局所最大値フィルタ（４連結）	
IP_MaxFLT8	局所最大値フィルタ（８連結）	
IP_LineMinFLT	ライン局所最小値フィルタ	
IP_LineMaxFLT	ライン局所最大値フィルタ	

6.16 ランクフィルタコマンド

ランクフィルタコマンドは、ソース画面の指定領域内の画素に対し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カーネルマスクで処理し、有効となった画素の中で値のソートを行い指定された順番の画素の出力を行います。特に、ソートした順番の中間値を抽出する処理をメディアンフィルタと呼びます。周辺では、中央の画素をそのまま出力します。
処理はすべてハードウェアで処理されます。

画像処理については、５章で説明していますので詳細はそちらを参照して下さい。

6.16.1 ウィンドウの設定

ランクフィルタコマンドは任意のウィンドウが設定できます。設定するウィンドウの種類はソース画面はＳＲＣ０＿ＷＩＮ（ソース０）でデスティネーション画面は、ＤＳＴ＿ＷＩＮ（デスティネーション）ウィンドウです。

6.16.2 画面データタイプ

ランクフィルタを行う画面（ソース画面）の画面データタイプは符号なし８ビットか符号付き８ビットです。符号なし８ビットか符号付き８ビットで処理結果が異なる場合があるので注意が必要です。また、ランクフィルタコマンド実行後のデスティネーション画面は、すべてシステムデータタイプに設定されます。

6.16.3 ランクフィルタコマンド一覧

表６－１７にランクフィルタコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－１７　ランクフィルタコマンド一覧

コマンド名	機能	備考
IP_RankFLT	局所ランクフィルタ	
IP_Rank4FLT	局所ランクフィルタ（４連結）	
IP_Rank8FLT	局所ランクフィルタ（８連結）	
IP_MedFLT	局所メディアンフィルタ	
IP_Med4FLT	局所メディアンフィルタ（４連結）	
IP_Med8FLT	局所メディアンフィルタ（８連結）	

6.17 ラベリングコマンド

ラベリングは、２値画像の物体に対してラベル（番号）付けを行う処理です。２値のソース画面の物体に対してラベル付けしたラベル値をデスティネーション画面に出力します。ラベリングコマンドにはラベル付けの処理とラベル付けされた画面に対して行うラベル特徴量抽出処理があります。ラベリングコマンドでは、ラベル付け処理とラベル特徴量抽出処理を組み合わせて２値物体の面積や外接四角形等を求めることができます。

画像処理については、５章で説明していますので詳細はそちらを参照して下さい。

6.17.1 ラベル付け

ラベル付けのコマンドには、IP_Label4 ， IP_Label8 ， IP_Label4withAreaFLT ， IP_Label8withAreaFLT ， IP_Label8withAreaFLT ， IP_Label4withAreaFLTSort ， IP_Label8withAreaFLTSort という６種類のコマンドがあります。これらのコマンドは、ソース画面の２値物体のラベル付けを行い、デスティネーション画面にラベル値を出力します。デスティネーション画面は、８ビットであるためラベル値「０」を省いた「１～２５５」のラベル付けを行うことができます。

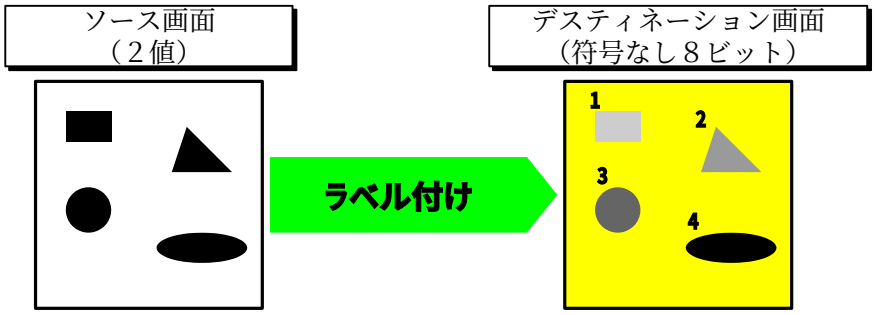


図６－３　ラベル付け

6.17.2 ラベル特徴量抽出

２値物体の面積、外接四角形、重心等を求めるには、
① ２値画像のラベル付けを行う
② ラベル付けされた画面に対してラベル特徴量抽出を行う
という操作を行います。①のラベル付けは前項で説明したとおりです。次に②の処理として IP_ExtractL0RegionX ， IP_ExtractL0RegionY ， IP_ExtractL0Area ， IP_ExtractL0Gravity コマンドでラベル付けされたデスティネーション画面に対してラベルの特徴量抽出を行います。

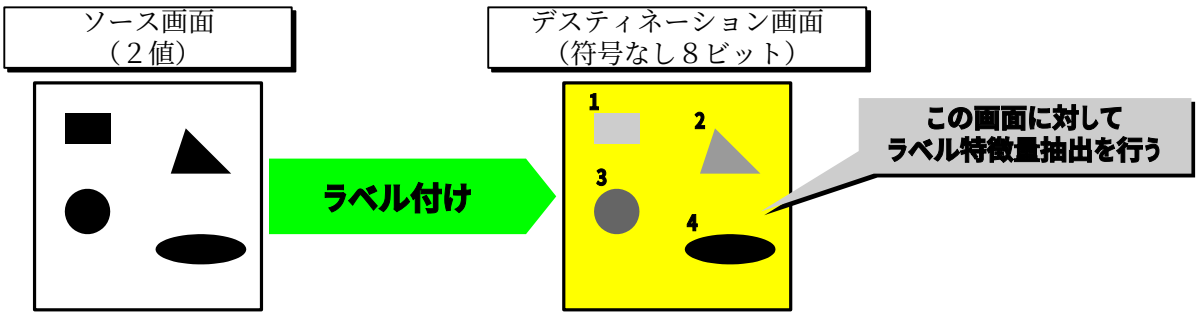


図６－４　ヒストグラム処理の対象画面

6.17.3 ウィンドウの設定

①ラベル付けコマンド

ラベル付けコマンドは任意のウィンドウが設定できます。設定するウィンドウの種類はソース画面はＳＲＣ０＿ＷＩＮ（ソース０）でデスティネーション画面は、ＤＳＴ＿ＷＩＮ（デスティネーション）ウィンドウです。

②ラベル特徴量抽出コマンド

ラベル特徴量抽出（IP_ExtractLORegionX , IP_ExtractLORegionY , IP_ExtractLOArea , IP_ExtractLOGravity）ではソース画面ＳＲＣ０＿ＷＩＮ（ソース０）の設定になります。デスティネーション画面はなく、ロングワード（４バイト）データ配列や指定された構造体配列に処理結果が格納されます。このとき、ラベル特徴量抽出コマンドの画面は、ラベル付けコマンド実行後のデスティネーション画面を指定して下さい。

また、処理結果として出力される座標はＳＲＣ０＿ＷＩＮ（ソース０）の相対座標で出力されますので注意して下さい。

6.17.4 画面データタイプ

ラベル付けコマンドでは、ラベル付けを行う画面（ソース画面）の画面データタイプは２値データで、「０」または「２５５」のデータのみの画像データである必要があります。それ以外では正常に動作しないことがあるので注意が必要です。また、ラベリ付けコマンド実行後のデスティネーション画面は、すべて符号なし８ビットに設定されます。

また、ラベル特徴量抽出コマンドは、符号なし８ビットの画面に対して処理を行うことを前提としています。

6.17.5 ラベリングコマンド一覧

表６－１８にラベリングコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－１８　ラベリングコマンド一覧

コマンド名	機能	備考
IP_Label4	ラベリング（４連結）	
IP_Label8	ラベリング（８連結）	
IP_Label4withAreaFLT	面積フィルタ付ラベリング（４連結）	
IP_Label8withAreaFLT	面積フィルタ付ラベリング（８連結）	
IP_Label4withAreaFLTSort	面積フィルタ付ラベリング（面積ソート４連結）	
IP_Label8withAreaFLTSort	面積フィルタ付ラベリング（面積ソート８連結）	
IP_ExtractLORegionX	ラベル毎領域Ｘ座標抽出（Min & Max Ｘ座標）	
IP_ExtractLORegionY	ラベル毎領域Ｙ座標抽出（Min & Max Ｙ座標）	
IP_ExtractLOArea	ラベル毎面積抽出	
IP_ExtractLOGravity	ラベル毎重心座標抽出	

6.18 濃淡画像特徴量抽出コマンド

濃淡画像特徴量抽出コマンドは、濃淡画像（符号無し８ビット／符号付き８ビット）の画面に対してヒストグラムや投影等を行うコマンドです。
処理はすべてハードウェアで処理されます。

画像処理については、５章で説明していますので詳細はそちらを参照して下さい。

6.18.1 ウィンドウの設定

濃淡画像特徴量抽出コマンドは任意のウィンドウが設定できます。設定するウィンドウの種類はソース画面はＳＲＣ０_ＷＩＮ（ソース０）です。デスティネーション画面はなく、ワード（２バイト）または、ロングワード（４バイト）データ配列や指定された構造体配列に処理結果が格納されます。
また、処理結果として出力される座標はＳＲＣ０_ＷＩＮ（ソース０）の相対座標で出力されますので注意して下さい。

6.18.2 画面データタイプ

濃淡画像特徴量抽出を行う画面（ソース画面）の画面データタイプは符号なし８ビットか符号付き８ビットです。符号なし８ビットか符号付き８ビットで処理結果が異なる場合があるので注意が必要です。

6.18.3 濃淡画像特徴量抽出コマンド一覧

表６－１９に濃淡画像特徴量抽出コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－１９　濃淡画像特徴量抽出コマンド一覧

コマンド名	機能	備考
IP_ExtractGOFeatures	濃淡画像基本特徴量抽出	
IP_Histogram	濃度ヒストグラム	
IP_ProjectGO	Ｘ＆Ｙ軸への濃度累積値投影（１６ビット）	
IP_ProjectGOonX	Ｘ軸への濃度累積値投影	
IP_ProjectGOonY	Ｙ軸への濃度累積値投影	
IP_ProjectGOMaxValue	Ｘ＆Ｙ軸への最大濃度値投影	
IP_ProjectGOMinValue	Ｘ＆Ｙ軸への最小濃度値投影	
IP_HistogramShort	濃度ヒストグラム（１６ ビット）	
IP_ProjectBlockGO	領域毎濃淡画像濃度累積	
IP_ProjectBlockGOMinMaxValue	領域毎濃淡画像最大・最小濃度値摘出	
IP_ProjectLabelGO	ラベル毎濃淡画像濃度累積	
IP_ProjectLabelGOMinMaxValue	ラベル毎濃淡画像最小／最大濃度値摘出	
EnableRotateProject	斜方投影処理有効	
DisableRotateProject	斜方投影処理無効	
IP_HistogramFeatures	濃度ヒストグラム特徴量抽出	

6.19 2値画像特徴量抽出コマンド

２値画像特徴量抽出コマンドは、２値画像の画面に対して投影をはじめとした特徴量の抽出を行うコマンドです。
処理はすべてハードウェアで処理されます。

画像処理については、５章で説明していますので詳細はそちらを参照して下さい。

6.19.1 ウィンドウの設定

２値画像特徴量抽出コマンドは任意のウィンドウが設定できます。設定するウィンドウの種類はソース画面はＳＲＣ０_ＷＩＮ（ソース０）です。デスティネーション画面はなく、ワード（２バイト）または、ロングワード（４バイト）データ配列や指定された構造体配列に処理結果が格納されます。
また、処理結果として出力される座標はＳＲＣ０_ＷＩＮ（ソース０）の相対座標で出力されますので注意して下さい。

6.19.2 画面データタイプ

２値画像特徴量抽出を行う画面（ソース画面）の画面データタイプは２値で、「０」または「２５５」のデータのみの画像データである必要があります。それ以外では正常に動作しない場合があるので注意が必要です。

6.19.3 2値画像特徴量抽出コマンド一覧

表６－２０に２値画像特徴量抽出コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－２０　２値画像特徴量抽出コマンド一覧

コマンド名	機能	備考
IP_ExtractBOFeatures	２値画像基本特徴量抽出	
IP_ProjectBO	Ｘ＆Ｙ軸への投影	
IP_ProjectBORegionX	領域Ｘ座標抽出（Ｙ軸へのMin&MaxＸ投影）	
IP_ProjectBORegionY	領域Ｙ座標抽出（Ｘ軸へのMin&MaxＹ投影）	
IP_ExtractBOArea	２値画像累積値抽出	
IP_ProjectBlockBO	領域毎２値画像濃度累積	

6.20 正規化相関サーチの概要

正規化相関サーチは、セットアップ、トレーニング、サーチという３つのステップで実行されます。また、使用したいテンプレート数に合わせてテンプレート特徴量データ領域を確保する必要があります。

画像処理については、５章で説明していますので詳細はそちらを参照して下さい。

6.20.1 正規化相関サーチのフロー

正規化相関サーチを行うためには、テンプレート特徴量データ領域の確保、セットアップ、トレーニング、サーチのを下図のフローで実行します。

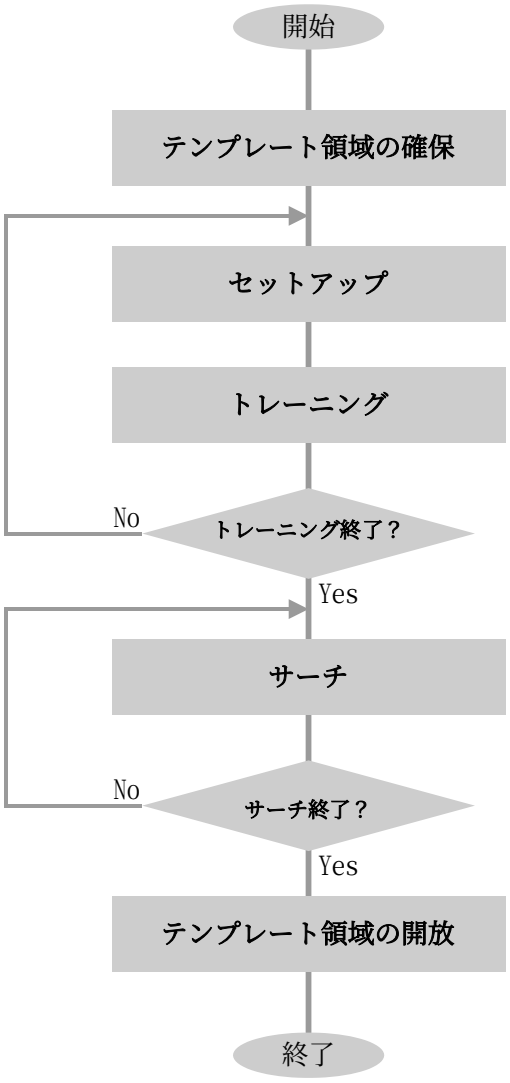


図６－５　正規化相関サーチのフロー

6.20.2 テンプレート特徴量データ領域の確保

正規化相関サーチのセットアップ、トレーニング、サーチを行う前に使用したいテンプレート数に合わせてテンプレート特徴量データ領域を確保する必要があります。また、確保した領域は、アプリケーションの終了時に開放します。

6.20.3 セットアップ

正規化相関サーチのセットアップとは、サーチを行う際のテンプレートを準備する操作のことです。
テンプレートとして登録したい画像を入力画像から切出したり、目的とする画像を画像描画コマンドで描画したりしてテンプレートを用意します。

6.20.4 トレーニング

正規化相関サーチのトレーニングとは、サーチを行う際のテンプレートを登録することです。トレーニングされたデータはテンプレート番号（テンプレートＩＤ）で管理されます。

6.20.5 サーチ

正規化相関サーチのサーチは、テンプレートとサーチしたい画面を指定して実際にサーチを行います。

6．21 テンプレートデータ領域管理コマンド

正規化相関サーチを行う際、テンプレート特徴量データ領域確保コマンドにより、必ずテンプレート特徴量データ領域を確保しなければなりません。

6．21．1 テンプレートデータについて

トレーニングによりテンプレートが登録されると正規化相関サーチに必要なデータはテンプレート番号（テンプレートＩＤ）で管理されます。

正規化相関サーチのテンプレートのデータは、テンプレート特徴量と画像が必要です。トレーニングを行うと指定された画像からテンプレート特徴量を計算し、テンプレート特徴量データ領域にセーブされます。指定された画像は、そのまま画像メモリにあり、その画面番号とウィンドウのデータだけがテンプレート特徴量データ領域にセーブされます。つまり、テンプレート特徴量データと画像データは別々の領域にあるということで、特に画像データは誤って別の画像に書換えられる可能性があります。テンプレートデータ内の特徴量データと画像メモリの内容が異なると正常にサーチできなくなるのでテンプレートに指定する画面の管理には十分注意が必要です。

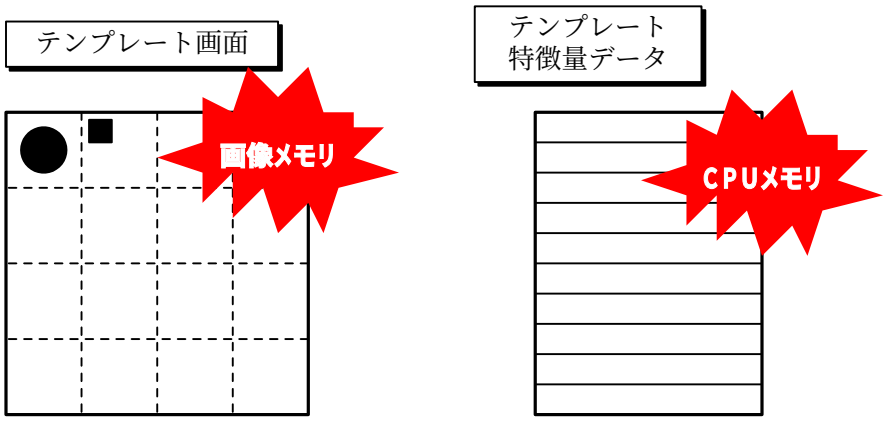


図６－６　テンプレートデータ

6．21．2 テンプレートデータ領域管理コマンド

表６－２１にテンプレート特徴量データ領域確保コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－２１　テンプレートデータ領域管理コマンド一覧

コマンド名	機能	備考
vpxAllocCorrTemplate	テンプレート特徴量データ領域確保	
vpxFreeCorrTemplate	テンプレート特徴量データ領域開放	
vpxReadCorrTemplate	テンプレート特徴量データ読出し	
vpxWriteCorrTemplate	テンプレート特徴量データ書込み	

6．22 セットアップとトレーニングコマンド

正規化相関サーチは、セットアップ、トレーニング、サーチという３つのステップで実行されます。ここでは、セットアップとトレーニングのコマンドについて説明します。

6．22．1 セットアップと画像の切出し

正規化相関サーチのセットアップとは、サーチを行う際のテンプレートを準備する操作のことです。
テンプレートとして登録したい画像を入力画像から切出したり、目的とする画像を画像描画コマンドで描画したりしてテンプレートを用意します。
正規化相関サーチでは、セットアップコマンドとして特別に用意しているものはありません。画像転送、画像縮小コマンドで画像をテンプレート画面として確保した画面に画像を切出し（転送）して下さい。
また、画像描画コマンドを使用して目的の図形の画像をテンプレート画面に直接描画するか、別の画面に描画したものを画像転送、画像縮小コマンドでその画像を切出して下さい。

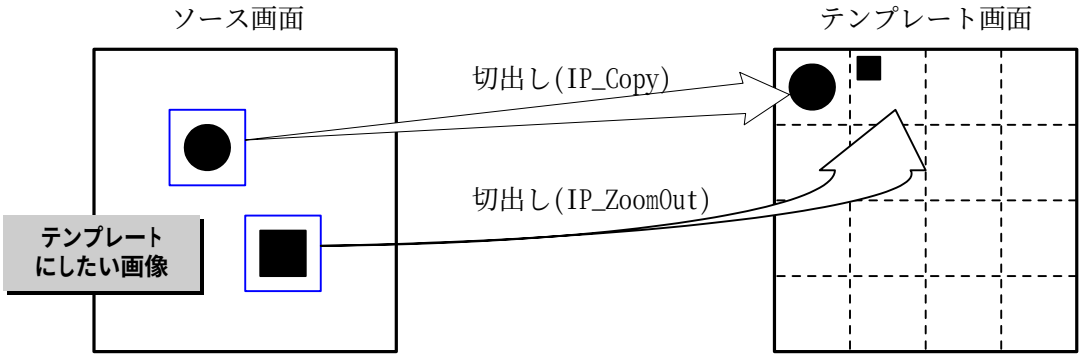


図６－７　画像の切出し

6．22．2 テンプレートの情報量と処理時間

正規化相関サーチは、画像処理プロセッサとオンボードＣＰＵにより行っています。画像処理プロセッサではテンプレートカーネル内の積和演算を行います。オンボードＣＰＵでは、積和演算の結果から相関値を計算し、テンプレートカーネルを移動します。

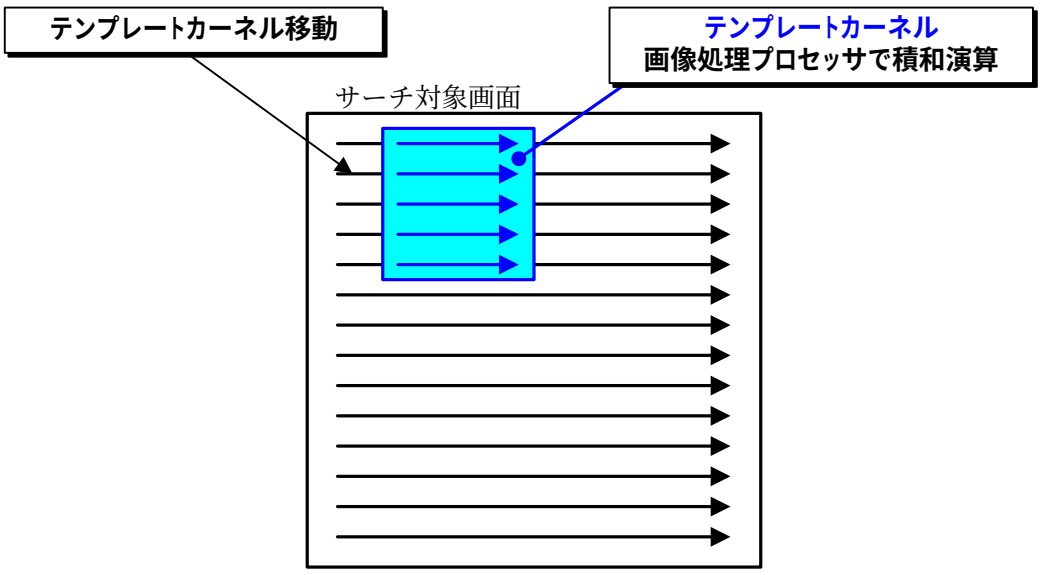


図６－８　正規化相関処理

画像処理プロセッサは、テンプレートカーネル内の積和演算を約７ｎＳ／画素で実行します。そしてオンボードＣＰＵでテンプレートカーネルを移動しながら、それぞれのポイントでの相関値を求め、最大の相関値とその座標を求めるわけです。ですから、サーチ時間はハードのオーバヘッド等を無視して単純に計算すると

サーチ時間 ＝ ７ｎＳ × テンプレートの大きさ × カーネルの移動回数　（式６－１）

の式で計算できます。例えば、テンプレートの大きさが１２８×１２８画素で５１２×４８０のサーチ対象画面をサーチするとすると

7e-9 × (128 × 128) × (512 × 480) = 28.1秒

２８秒もかかることになります。
処理時間を縮めるには式６－１からテンプレートの大きさを小さくするかカーネルの移動回数を減らせばいいことになります。
画像処理コマンドでは、テンプレートに関しては、テンプレートの大きさを小さくする代わりに情報量を減らすことで等価的にテンプレートの大きさを小さくできるようにしています。情報量を減らすということは、サーチのときのカーネル内の積和演算のデータのサンプリングをハード的に間引いて実行するということです。また、カーネルの移動も間引いて実行できるようになっています。
上記の例で、テンプレート情報量を縦８画素、横８画素間引き、カーネル移動を縦８画素、横８画素間引くと

7e-9 × (128/8 × 128/8) × (512/8 × 480/8) = 0.007秒

２８秒の処理が４０９６分の１の７ミリ秒（実際には、レジスタアクセスのオーバヘッドで１０ｍＳ程度）でできることになります。
また、このときのテンプレートの情報量は

128/8 × 128/8 = 256

になります。

このように、実際の処理は情報量を減らして高速化を図ります。

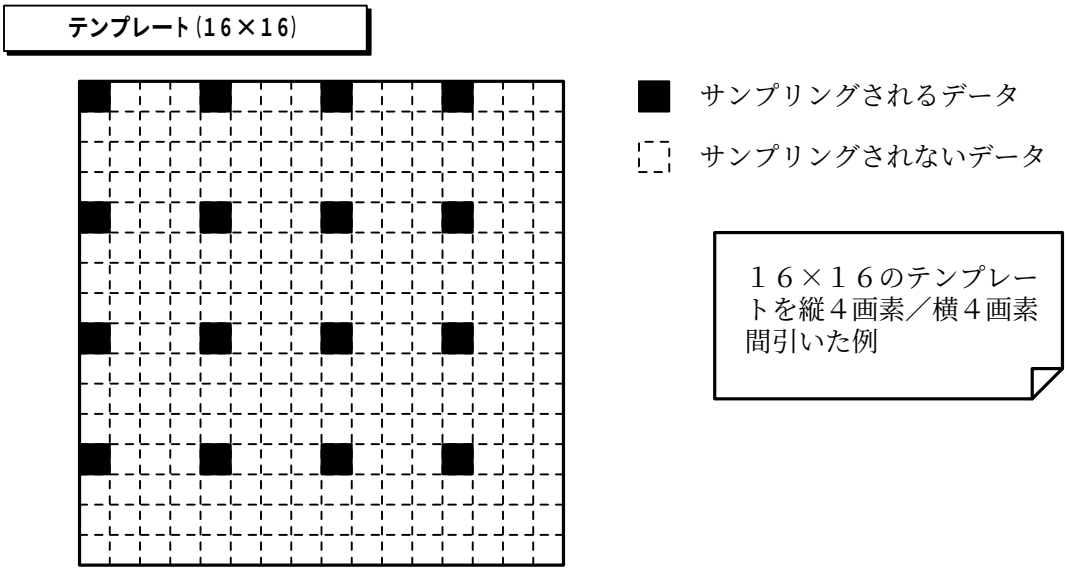


図６－９　テンプレートの間引き

6.22.3 テンプレートマスクの設定

テンプレート画像は矩形領域しか設定できません。しかし、テンプレート画像として矩形以外の形の領域を設定したい場合があります。そこで、テンプレート画像にはマスク（不感帯）領域を設定できる機能があります。テンプレート画像のマスク設定された領域は、ハード処理される積和演算で計算の対象から除外されます。

設定したいマスク領域のテンプレート画像の値を「0」にすることでその領域がマスク領域になります。通常は、円等の簡単な図形でマスクしますが、複雑なマスクも設定可能です。

下図にマスクの設定例を示します。

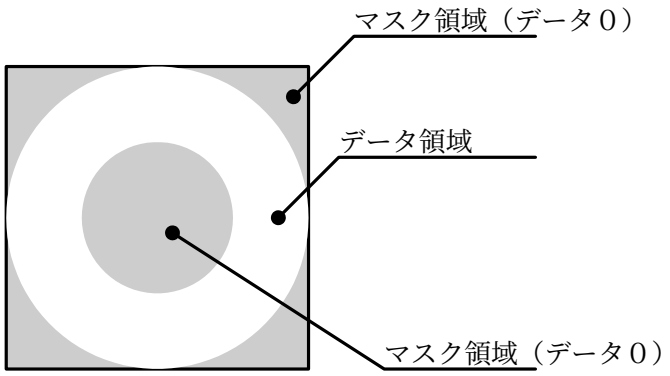


図６－１０　マスクの設定例

次にマスクの設定手順を説明します。まず、テンプレートを切出し、テンプレートのマスクしたい領域に「0」を書込みます。次にvpxEnableCorrMaskコマンドでマスクモードを有効にします。そして、vpxSetCorrTemplateまたは、vpxSetCorrTemplateExtコマンドでトレーニング（テンプレートの登録）を行います。マスクを解除する場合は、vpxDisableCorrMaskコマンドでマスクモードを解除して下さい。

マスクなしのテンプレートに戻したい場合は、テンプレート画像をもう一度切出し、トレーニングを行って下さい。

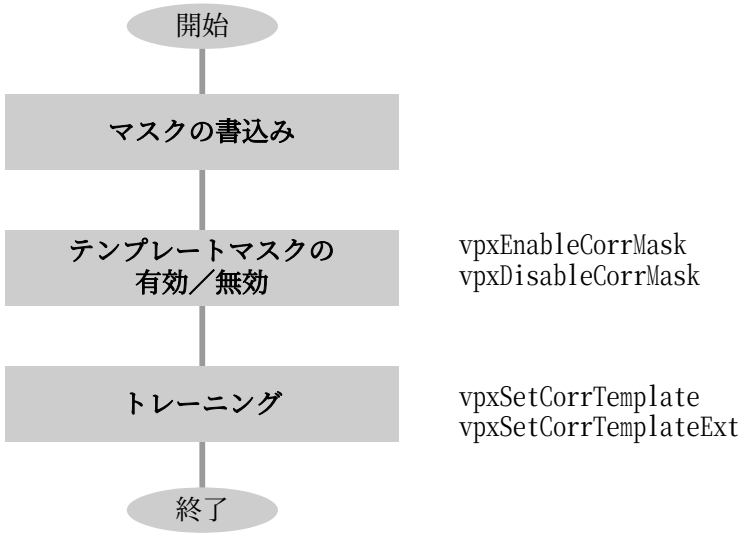


図６－１１　テンプレートマスクの設定フロー

6.22.4 トレーニングコマンド

トレーニングとは、テンプレートデータの登録のことです。トレーニングコマンドには vpxSetCorrTemplateと vpxSetCorrTemplateExt コマンドの２種類があります。２種類のコマンドの違いは、サーチを実行する際のテンプレートカーネル間引きの取り扱いです。

vpxSetCorrTemplate コマンドでトレーニングされたテンプレートは、サーチの際、テンプレートカーネルの画素データのサンプリングは間引かずにサーチする画面のサンプリングだけを間引きます。つまり、テンプレートカーネルの画像の大きさが実際にサーチされる画像の１／（間引き数）に小さくなり、その大きさがそのまま情報量になります。普通、vpxSetCorrTemplateコマンドでトレーニングする場合、画像の切出しは、切出す画像を縮小しなければならないので IP_ZoomOut コマンドで行います。

vpxSetCorrTemplateExtコマンドでトレーニングされたテンプレートは、サーチの際、テンプレートカーネルとサーチする画面の画素データのサンプリングを同じ間隔で間引きます。つまり、テンプレートカーネルの画像とサーチする画像は同じ大きさになりということです。普通、vpxSetCorrTemplateExtコマンドでトレーニングする場合、画像の切出しは、IP_Copyコマンドで行います。

また、間引き間隔は、パラメータ xmag , ymag で設定します。

6.22.5 ウィンドウの設定

トレーニングコマンドは登録するテンプレート領域をウィンドウにより設定します。設定するウィンドウの種類はソース画面はＳＲＣ０＿ＷＩＮ（ソース０）です。デスティネーション画面はなく、テンプレートデータ領域にテンプレートデータが格納されます。

6.22.6 画面データタイプ

トレーニングを行う画面（ソース画面）の画面データタイプは符号なし８ビットです。
また、トレーニングコマンドは、符号なし８ビットの画面に対して処理を行うことを前提としています。

6.22.7 トレーニングコマンド一覧

表６－２２にトレーニングコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－２２　トレーニングコマンド一覧

コマンド名	機能	備考
vpxEnableCorrMask	テンプレートマスクの有効化	
vpxDisableCorrMask	テンプレートマスクの無効化	
vpxSetCorrTemplate	トレーニング	
vpxSetCorrTemplateExt	トレーニング	

6．23 正規化相関サーチコマンド

正規化相関サーチは、セットアップ、トレーニング、サーチという３つのステップで実行されます。ここでは、サーチのコマンドについて説明します。

6.23.1 高速サーチの方法

サーチコマンドには、vpxIP_CorrStep ，vpxIP_CorrPoint ，vpxIP_CorrPrecise コマンドがあります。
サーチコマンドは、テンプレートカーネルを移動しながら相関値を演算し最大の相関値と座標を見つけ出す処理を行っています。
最も単純なサーチでは、前項（６．２２．２項）で説明しているように処理時間がユーザの要求に応えられるものではありません。そのため、テンプレートカーネルの間引きとサーチの間引きを行うわけです。前項（６．２２）ではテンプレートカーネルの間引きについて説明しましたが、ここではテンプレートカーネルを移動する時の間引きについてと高速化の方法を説明します。

vpxIP_CorrStep コマンドは、任意の間引き間隔でラスターサーチを行うコマンドです。このとき相関値（スコア）とその座標のソートを行いながら任意の候補点を抽出します。

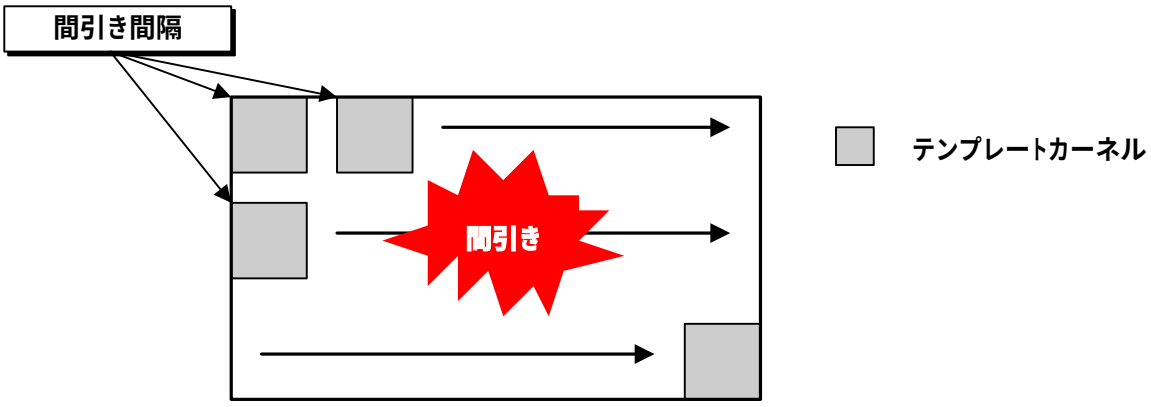


図６－１２　vpxIP_CorrStepコマンド

vpxIP_CorrPointコマンドは、入力された座標の近傍領域でサーチを行います。通常は、vpxIP_CorrStepコマンドで間引いたぶんの補間をするためのサーチとして使用します。

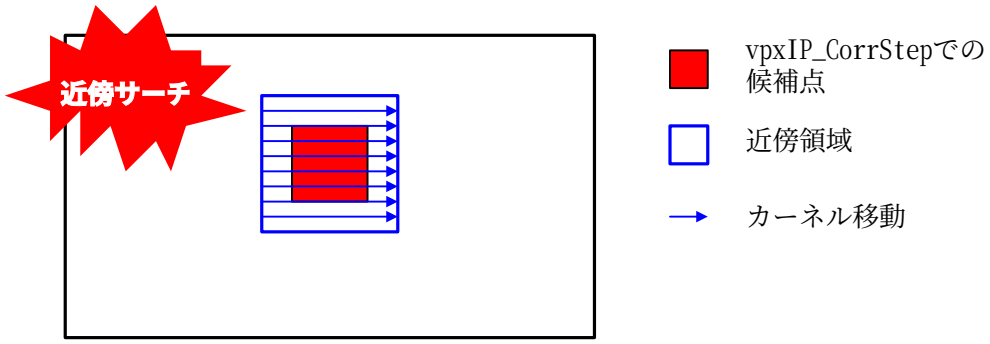


図６－１３　vpxIP_CorrPointコマンド

vpxIP_CorrPrecise コマンドは、入力された座標の近傍領域の相関値からサブピクセルの位置を計算します。通常は、vpxIP_CorrPoint コマンドで得られた最終位置座標を vpxIP_CorrPrecise コマンドに入力します。

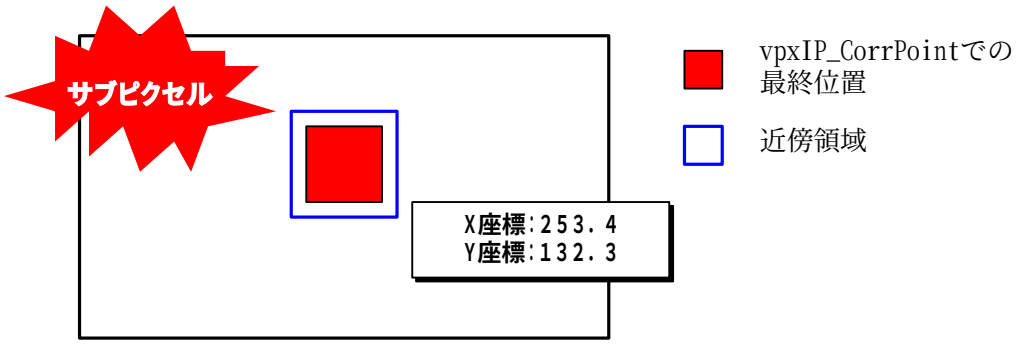


図６－１４　vpxIP_CorrPreciseコマンド

6.23.2 並列演算カーネルによるサーチの高速化

画像処理プロセッサは、正規化相関演算用の積和演算器を１６組内臓しています。その並列演算カーネルを使用して１つのテンプレートカーネルに対して１６組（４×４）の積和演算を並列に実行することができます。

並列演算カーネルを使用すると処理時間は原理的に並列演算カーネルを使わない時とくらべ１／１６になります。また、原理的に４×４カーネルの間隔は通常は１画素ですが、テンプレートカーネルの間引きを行うと４×４カーネルの間隔はテンプレートカーネルの間引き間隔になります。

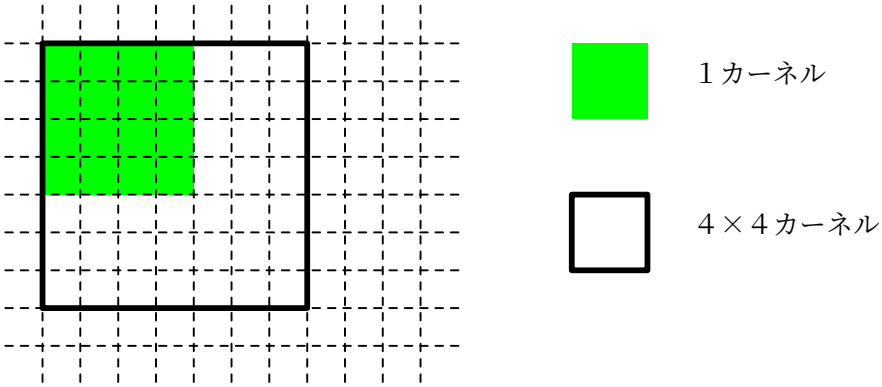


図６－１５　４×４カーネル

サーチの高速化には上記の並列演算カーネルを使用する方法とテンプレートとサーチ移動の間引きによる方法がありますが、両方の方法を使用することで画像処理ボードの最大の性能を引き出すことができます。しかし、並列演算カーネルとテンプレートとサーチ移動の間引きを同時に使用するには以下に示す条件を満足しなければなりません。

① サーチ移動の間引き間隔がテンプレートカーネルの間引き間隔と同じ
　または、その整数分の１
　または、その２倍

①の条件は、テンプレートカーネルの間引きを行うと４×４カーネルの間隔はテンプレートカーネルの間引き間隔になることに起因しています。

vpxIP_CorrStep ，vpxIP_CorrPoint コマンドは、上記①の条件のとき自動的に並列演算カーネルモードで処理を実行します。

6.23.3 相関演算途中打切りによるサーチの高速化

正規化相関の演算として画像処理プロセッサによりテンプレートカーネル内の積和演算を行っていますが、その他にもう一つ差分累積演算を行う演算器を持っています。これは、テンプレートカーネル内の画像と差分累積演算を行い累積誤差が指定閾値よりも大きくなったとき、正規化相関の積和演算の処理を途中で打切るためのものです。正規化相関の積和演算の処理を途中で打切ることにより、正規化相関サーチの高速化を図ることができます。
サーチ対象画像がテンプレート画像とあきらかに異なる場合、相関演算実行中に累積誤差が非常に大きくなると考えられます。そこで、この累積誤差がある閾値を超えた時点で、ミスマッチと判断し、途中で演算を中止します。
途中で演算を中止することにより処理の高速化を図ることができますが、照明の変動による許容値が低くなるので、注意して下さい。

● 閾値の設定

閾値とモードの設定は vpxSetCorrBreak コマンドで行います。
実際に演算を途中で打ち切るための評価値である累積誤差閾値は、
累積誤差閾値 ＝ ｔｈｒ × テンプレート画素数
で計算されます。
また、ＳＳＤＡ法による自動閾値変更モードもサポートしています。これは、常に最小を閾値として、演算の打ち切りを行います。この場合、照明変動にはかなり弱くなりますので十分に注意して下さい。

● 相関演算打切りの設定

相関演算途中打切りを行う場合は、vpxIP_CorrStep , vpxIP_CorrPoint コマンドを実行する前に vpxEnableCorrBreak , vpxDisableCorrBreak コマンドで相関演算途中打切りモードを設定して下さい。
画像処理コマンドシステムの初期設定は、相関演算途中打切り無効になっています。

6.23.4 サーチ除外領域設定とマッチング候補点

vpxIP_CorrStep コマンドでは、テンプレートカーネルを移動しながら相関値の計算を行い、マッチングポイントを探します。通常は、間引いてサーチを行うので、探し出したポイントは正確なマッチングポイントではありません。その正確ではないマッチングポイントをマッチング候補点と呼びます。
vpxIP_CorrStep コマンドでは、探し出すマッチング候補点の個数を任意に設定できるようになっています。しかし、複数のマッチング候補点をサーチすると、近傍領域にその候補点が集中します。
そこで、マッチング候補点がその近傍領域に集中しないようにサーチ除外領域を設定します。サーチ除外領域として設定した距離以内にマッチング候補点が２つ以上存在しないようにポイントを探します。
サーチ除外領域を設定／解除は、vpxSetSearchDistance コマンドで行います。

6.23.5 ウィンドウの設定

サーチコマンドはサーチする領域をウィンドウにより設定すことができます。設定するウィンドウの種類はソース画面はＳＲＣ０_ＷＩＮ（ソース０）です。デスティネーション画面はなく、マッチングポイントとスコア値が構造体配列に格納されます。出力されるマッチングポイントの座標は画面（０，０）相対座標です。ＳＲＣ０_ＷＩＮ相対ではないので注意して下さい。

6.23.6 画面データタイプ

サーチを行う画面（ソース画面）の画面データタイプは符号なし８ビットです。
また、トレーニングコマンドは、符号なし８ビットの画面に対して処理を行うことを前提としています。

6.23.7 サーチコマンド一覧

表６－２３にサーチコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－２３　サーチコマンド一覧

コマンド名	機能	備考
vpxIP_CorrStep	ラスターサーチ	
vpxIP_CorrPoint	近傍サーチ	
vpxIP_CorrPrecise	サブピクセルサーチ	
vpxEnableCorrBreak	相関演算途中打切りの有効化	
vpxDisableCorrBreak	相関演算途中打切りの無効化	
vpxSetCorrBreakThr	相関演算途中打切り閾値設定	
vpxSetSearchDistance	サーチ除外領域の設定	
vpxSetCorrMode	相関サーチモード設定	
vpxSetCorrPrecise	サブピクセル算出桁数設定	
vpxIP_CorrMap	相関値分布の抽出	
vpxGetCorrMapSize	相関値分布格納テーブルサイズ取得	

6．24 2値化閾値算出支援コマンド

画像処理コマンドでは２値化閾値算出支援コマンドを用意しています。以下にその使用方法を説明します。

6.24.1 判別分析法による2値化閾値算出

判別分析法による２値化閾値算出のフローを下図に示します。判別分析法による２値化閾値算出を行うまえに２値化する画面のヒストグラムを求めておく必要があります。

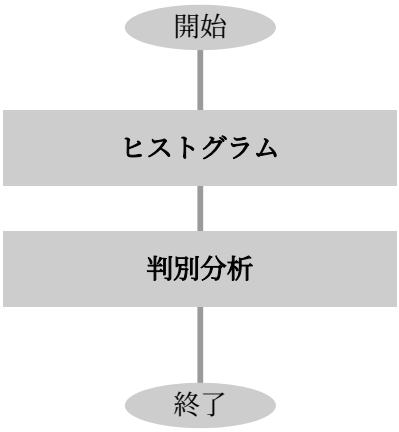


図６－１６　２値化閾値算出フロー

6.24.2 2値化閾値算出支援コマンド一覧

表６－２４に２値化閾値算出支援コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－２４　２値化閾値算出支援コマンド一覧

コマンド名	機能	備考
HistAnalyze	判別分析法	

6．25 直線抽出コマンド

画像処理コマンドではハフ変換による直線・矩形抽出コマンドを用意しています。以下にその使用方法を説明します。

6．25．1 ハフ変換による直線抽出

画像処理コマンドでは、ハフ変換を用いて離散的な座標データから直線を抽出します。ハフ変換の演算はオンボードＣＰＵで高速に処理します。
また、直線抽出コマンドでは下図のようにρ－θで示される直線が抽出されます。

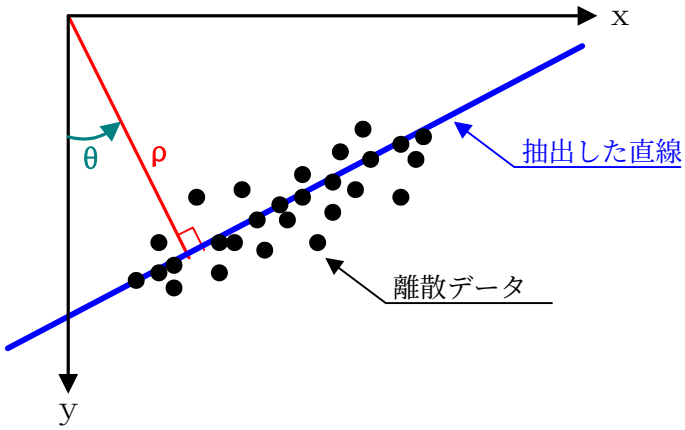


図６－１７　離散データからの直線抽出

離散データからの直線抽出では、まず、図６－１７に示すような離散データの座標を２値化した画像から IP_ProjectBORegionX ， IP_ProjectBORegionY コマンドなどの座標抽出を行い抽出します。次に、その座標データ群からハフ変換によりρ－θで示される直線を抽出します。

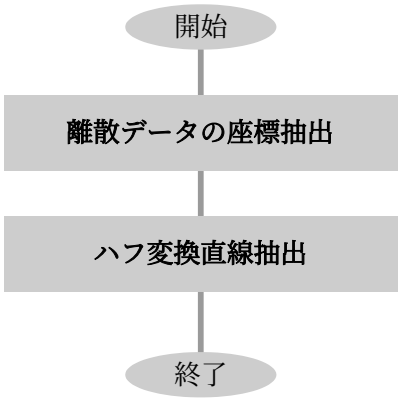


図６－１８　離散データからの直線抽出フロー

6.25.2 ハフ変換直線からの矩形算出

画像処理コマンドでは、ハフ変換で抽出されたρ－θで示される４つの直線からそれぞれの交点座標を算出し、矩形の頂点座標や角度等を算出することができます。

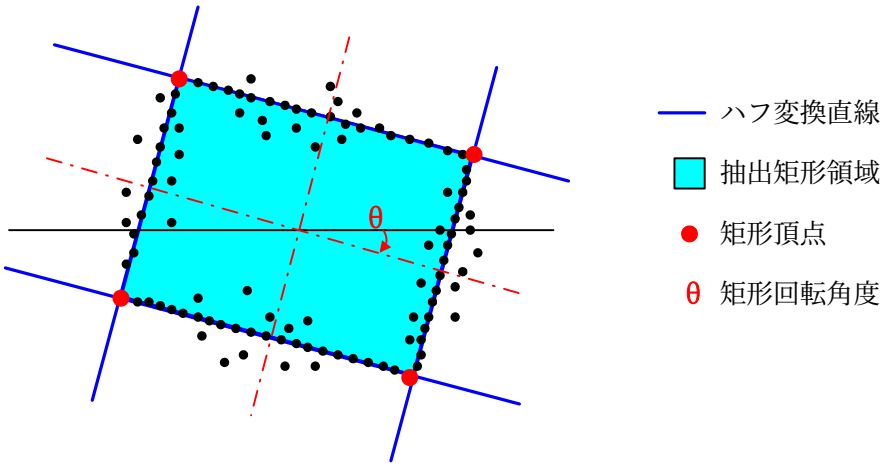


図６－１９　ハフ変換直線からの矩形算出

GetCrossPoint,GetRectPoint,GetRectCenter,GetAnglePoint4,GetAnglePoint2コマンドにより、ハフ変換で抽出されたρ－θで示される２つまたは４つの直線からそれぞれの交点座標の算出、矩形の頂点座標や角度等の算出を行うことができます。

6.25.3 直線抽出コマンド一覧

表６－２５に直線抽出コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－２５　直線抽出コマンド一覧

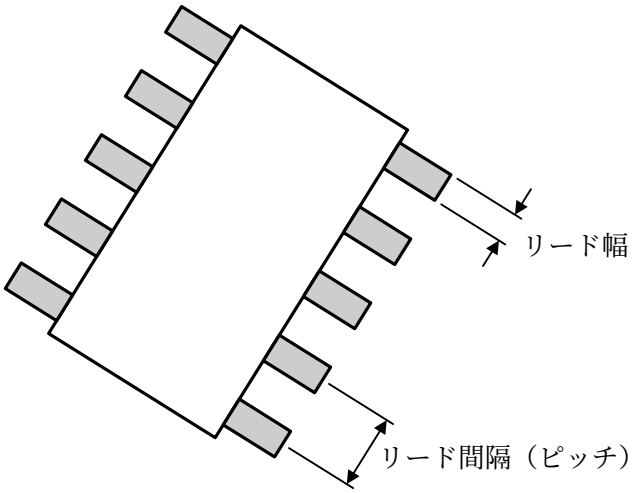
コマンド名	機能	備考
GetHoughLine	離散データ座標からの直線抽出	
GetHoughLineRow	離散データ座標からの２直線抽出	
GetHoughLineRowExt	離散データ座標からの２直線抽出（拡張）	
GetSideHoughLine	離散データ座標からの直線抽出（角度固定）	
GetCrossPoint	ハフ変換直線の交点座標算出	
GetRectPoint	ハフ変換直線の矩形交点座標の算出	
GetRectCenter	矩形頂点中心座標の算出	
GetAnglePoint4	矩形頂点間中心座標４ポイントからの角度算出	
GetAnglePoint2	矩形頂点間中心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

6．26 イメージキャリパコマンド

画像処理コマンドではキャリパコマンドを用意しています。キャリパとはノギスのことで、画像処理により対象物のエッジを検出し、その対象物のエッジで平行なペアをサーチし、２つのエッジ間の距離やその中心位置を計測することができます。
以下にその使用方法を説明します。

6．26．1 イメージキャリパコマンドによる寸法計測

以下にイメージキャリパコマンドによる寸法計測のフローを示します。例は、集積回路（ＩＣ）パッケージのリード幅やリード間隔を求める場合です。



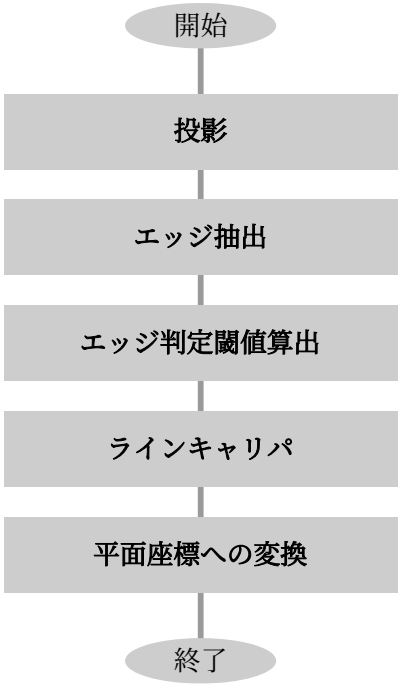


図６－２０　寸法計測フロー

まず、ProjectLineコマンドにより、ＩＣのリードに沿って投影を行い、平面のデータをラインのデータに変換します。次にLineEdgeFilter, LineEdgeFilterExtコマンドによりエッジ抽出を行います。エッジデータからGetCaliperScoreコマンドでエッジ判定閾値を算出し、LineCaliperコマンドでエッジ判定閾値に従い、アップエッジとダウンエッジのペアをサーチします。そして、CaliperLPtoSPコマンドによりラインデータを平面のデータに変換して処理が完了します。そのデータからリード幅とリード間隔を計算します。

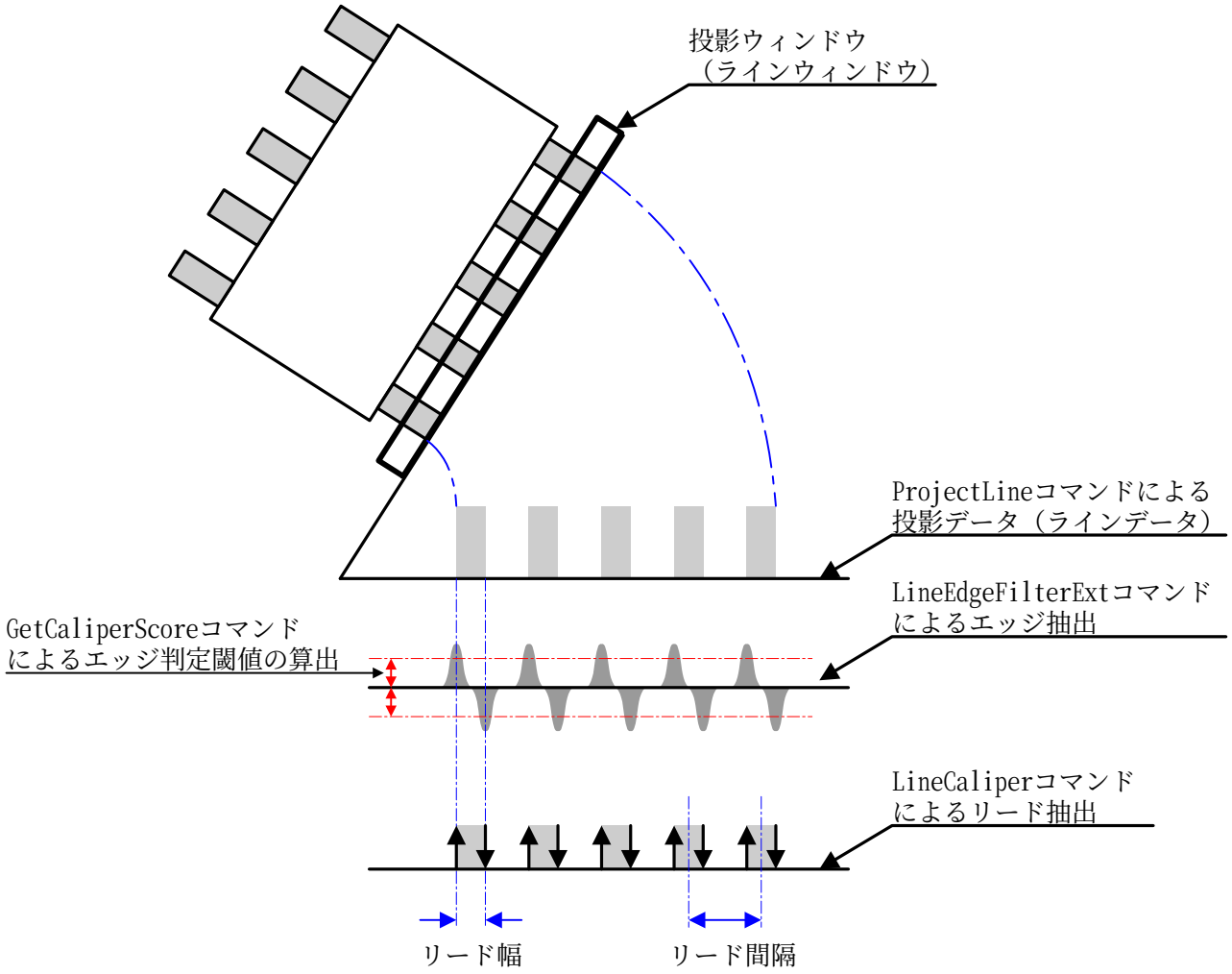


図６－２１　寸法計測方法

6.26.2 ラインウィンドウについて

ProjectLineコマンドで投影を行う場合やCaliperLPtoSPコマンドによりラインデータを平面のデータに変換する場合、投影を行う領域を指定する必要があります。その場合の領域は、ラインウィンドウにより指定します。通常の画像処理コマンドのウィンドウは、傾斜した矩形領域を設定することはできませんが、ProjectLineコマンドでは、ラインウィンドウにより傾斜した矩形領域を設定し、傾斜した領域を投影することができます。

以下に、ラインウィンドウの構造体を示します。

```
typedef struct {
    short      sx;       // 開始点Ｘ座標
    short      sy;       // 開始点Ｙ座標
    short      ex;       // 終了点Ｘ座標
    short      ey;       // 終了点Ｙ座標
    short    leng;       // 投影幅
    short     opt;       // オプション（未使用）
} LINEWINDOW;
```

次に、ラインウィンドウの例を挙げながら構造体の各メンバーについて説明します。

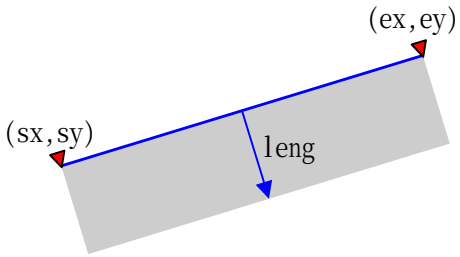
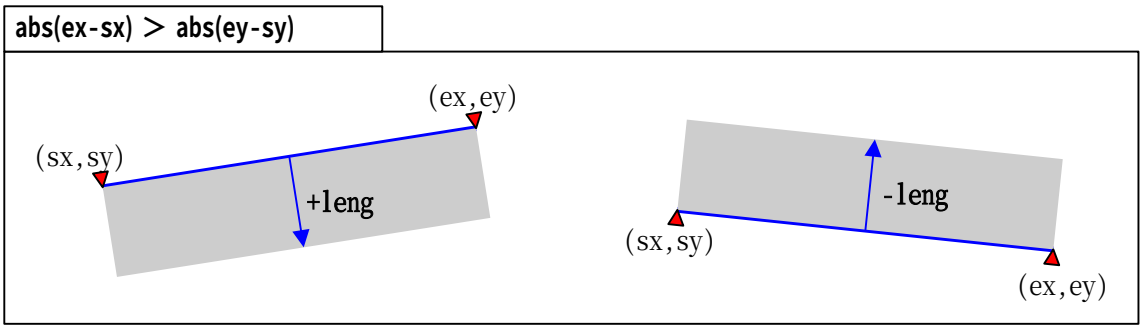


図６－２２　ラインウィンドウ

ラインウィンドウでは、(sx,sy)と(ex,ey)で結ばれる直線で投影領域を指定し、lengで投影する画素数を指定します。(sx,sy)と(ex,ey)で結ばれる直線上のポイントが投影の開始ポイントになり、直線上のポイントに沿って(sx,sy)と(ex,ey)で結ばれる直線と垂直な方向にlengで指定される画素数分画素データの濃度累積が行われなす。

また、lengは(sx,sy)と(ex,ey)で結ばれる直線の状態と符号により濃度累積する方向が変わります。



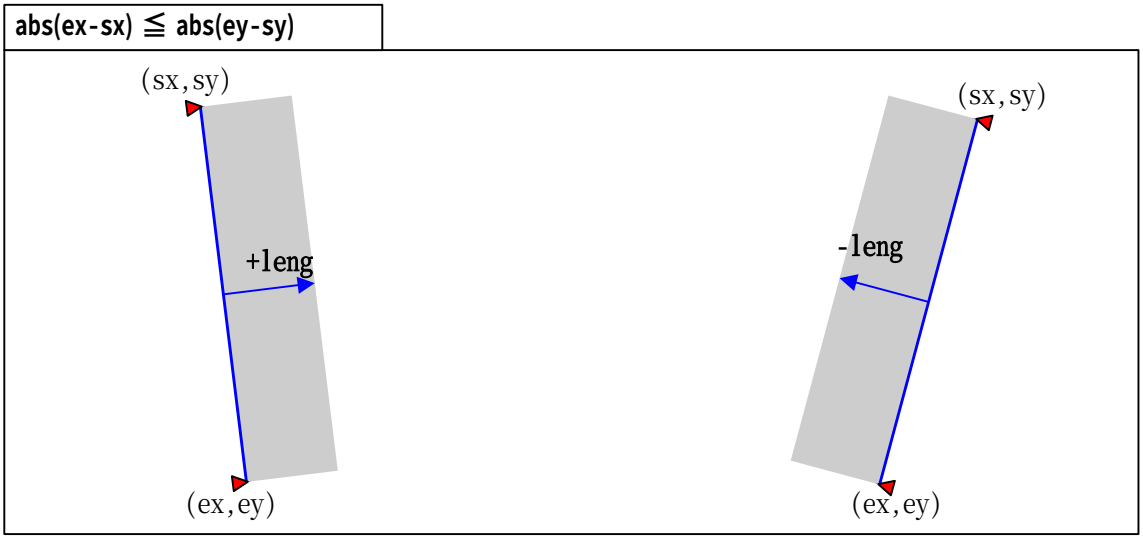


図６－２３　lengの符号

6．26．3 イメージキャリパコマンド一覧

表６－２６にイメージキャリパコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－２６　イメージキャリパコマンド一覧

コマンド名	機能	備考
ProjectLine	傾斜矩形領域の投影	
LineFilter	エッジ抽出	
LineFilterExt	フィルター付きエッジ抽出	
LineCaliper	ラインキャリパー	
CaliperLPtoSP	ライン座標から平面座標への変換	
GetCaliperScore	エッジ判定閾値の算出	
SetCaliperWidth	投影幅の算出	

6.27 エッジファインダコマンド

画像処理コマンドではエッジファインダコマンドを用意しています。イメージキャリパでは画像処理により対象物のエッジを検出し、その対象物のエッジで平行なペアをサーチするため、エッジのペアがないパターンでは、エッジを抽出することはできません。そこで、エッジファインダーではその対象物に含まれるあらゆるエッジを解析し、位置、ポラリティ、レベルといった情報を抽出します。
以下にその使用方法を説明します。

6.27.1 ラインエッジファインダコマンドによるエッジ抽出

以下にラインエッジファインダコマンドによるエッジ抽出のフローを示します。処理手順は基本的にイメージキャリパコマンドと同じですので、そちらも参照して下さい。

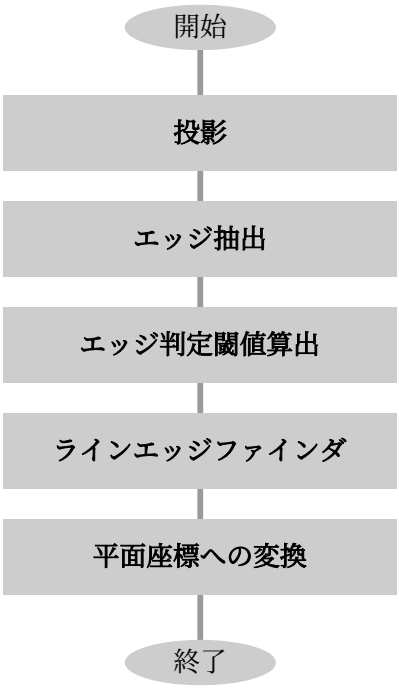


図６－２４　エッジ抽出フロー

6.27.2 エッジファインダコマンド一覧

表６－２７にエッジファインダコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－２７　エッジファインダコマンド一覧

コマンド名	機能	備考
LineEdgeFinder	ラインエッジファインダ	
EdgeFinderLPtoSP	ライン座標から平面座標への変換	

6.28 I／O入出力コマンド

画像処理コマンドでは、デジタル入出力ボードにアクセスするためのＩ／Ｏ入出力コマンドを用意しています。

6.28.1 I／O入出力コマンド一覧

表６－２８にＩ／Ｏ入出力コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－２８　Ｉ／Ｏ入出力コマンド一覧

コマンド名	機能	備考
InIPPort	Ｉ／Ｏポートからのデータ入力	
OutIPPort	Ｉ／Ｏポートへのデータ出力	

6.29 画像メモリアクセスコマンド

画像処理コマンドでは画像メモリのデータを１画素単位または、ブロック単位でアクセスするためのコマンドを用意しています。画像メモリアクセス手順についての詳細は、コマンドリファレンスを参照して下さい。

6.29.1 ウィンドウの設定

画像メモリアクセスコマンドは任意のウィンドウが設定できます。設定するウィンドウの種類はＳＹＳ＿ＷＩＮです。ReadImg, ReadImgReverseコマンドはＳＹＳ＿ＷＩＮで設定された矩形領域の画素データをローカルメモリにブロック単位に読込み、WriteImg, WriteImgReverseコマンドはローカルメモリ上のデータをＳＹＳ＿ＷＩＮで設定された矩形領域の画像メモリにブロック単位に書込みます。

6.29.2 画面データタイプ

画像メモリアクセスを行う画面の画面データタイプは符号なし８ビット、符号付き８ビット、２値です。２値データ０は画素データ「０」、２値データ１は画素データ「２５５」です。

6.29.3 画像メモリアクセスコマンド一覧

表６－２９に画像メモリアクセスコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－２９　画像メモリアクセスコマンド一覧

コマンド名	機能	備考
OpenImg	画像メモリアクセス許可	
CloseImg	画像メモリアクセス禁止	
ReadImg	画像メモリ読出し（矩形領域）	
WriteImg	画像メモリ書込み（矩形領域）	
ReadImgReverse	画像メモリ読出し（ビットマップ矩形領域）	
WriteImgReverse	画像メモリ書込み（ビットマップ矩形領域）	
SetPixelPointer	画像メモリポインタ設定	
ReadPixel	画像メモリ１画素読出し	
WritePixel	画像メモリ１画素書込み	
ReadPixelContinue	画像メモリ連続１画素読出し	
WritePixelContinue	画像メモリ連続１画素書込み	
ReadImgLine	画像メモリ１ライン読出し	
WriteImgLine	画像メモリ１ライン書込み	
GetImVirtualAddress	画像メモリのアドレス取得	

6.30 図形描画コマンド

画像処理コマンドでは線、矩形、円、多角形等の図形を画像メモリに描画するためのコマンドを用意しています。

6.30.1 描画領域

図形描画コマンドでの描画領域は指定した画面の領域全体です。任意にウィンドウを設定することはできません。また、指定された画面の領域をはみ出す部分はクリッピングされます。

6.30.2 画面データタイプ

図形描画を行う画面の画面データタイプは符号なし８ビット、符号付き８ビット、２値です。２値データ０は画素データ「０」、２値データ１は画素データ「２５５」です。

6.30.3 図形描画コマンド一覧

表６－３０に図形描画コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－３０　図形描画コマンド一覧

コマンド名	機能	備考
PutLine	線・矩形描画	
PutCross	十字カーソル描画	
PutLineWindow	ラインウィンドウ描画	
PutPolygon	多角形描画	
PutCircle	円描画	
PutArc	円弧描画	
PutRectangle	矩形描画	
PutTriangle	三角形描画	
PutDiamond	菱形描画	
PutCrossRect	十字形描画	
PutTwinRectangle	２連矩形描画	
PutArcExt	円描画（拡張）	
RefreshImg	画像メモリのリフレッシュ	
PutEllipse	楕円描画	

6．31 文字描画コマンド

画像処理コマンドでは英数文字を画像メモリに描画するためのコマンドを用意しています。

6.31.1 描画領域

図形描画コマンドでの描画領域は指定した画面の領域全体です。任意にウィンドウを設定することはできません。また、指定された画面の領域をはみ出す部分はクリッピングされます。

6.31.2 画面データタイプ

文字描画を行う画面の画面データタイプは符号なし８ビット、符号付き８ビット、２値です。２値データ０は画素データ「０」、２値データ１は画素データ「２５５」です。

6.31.3 文字描画コマンド一覧

表６－３１に文字描画コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－３１　文字描画コマンド一覧

コマンド名	機能	備考
PutAnkString	全角文字列の描画（英数文字）	
PutHalfString	半角文字列の描画（英数文字）	
PutAnkChar	全角文字の描画（英数文字）	
PutHalfChar	半角文字の描画（英数文字）	
PutString	全／半角文字列（漢字含む）の描画	

6.32 画面描画コマンド

ビットマップ画面には、画像メモリアクセスコマンド、図形描画コマンド、文字描画コマンドを使用して画像の描画ができますが、画像処理コマンドでは画面の矩形塗り潰しや画面クリア（０クリア）をするためのコマンドとして画面描画コマンドを用意しています。

6.32.1 ウィンドウの設定

ClearBitmapコマンドは任意のウィンドウが設定できます。ClearScreenコマンドは、指定された画面の画面サイズに従いクリアします。

6.32.2 画面データタイプ

文字描画を行う画面の画面データタイプは符号なし８ビット、符号付き８ビット、２値です。２値データ０は画素データ「０」、２値データ１は画素データ「２５５」です。

6.32.3 画面描画コマンド一覧

表６－３２に画面描画コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－３２　画面描画コマンド一覧

コマンド名	機能	備考
ClearBitmap	画面描画（範囲・色指定）	
ClearScreen	画面クリア	

6.33 タイマ／RTCコマンド

画像処理コマンドでは時間計測とウェイトを行うためのタイマコマンドとＲＴＣの時間をアクセスするコマンドを用意しています。タイマコマンドはオンボードＣＰＵのハードウェアタイマを使用し、μＳ～ｍＳ単位でコントロールできます。

6.33.1 タイマコマンドによる時間計測

タイマコマンドによる時間計測は、startGetIPTime, timeGetIPTime, stopGetIPTimeコマンドにより行います。

時間計測を始める前に、startGetIPTimeコマンドでタイマを起動し、timeGetIPTimeコマンドで計測個所の時間を読出します。一通り時間計測が終了したら stopGetIPTimeコマンドでタイマを停止させます。

以下にそのフローを示します。

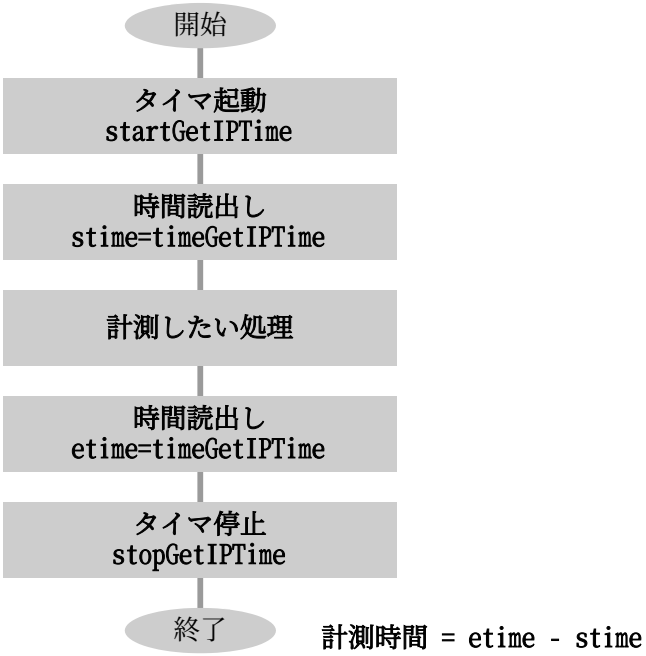


図６－２５　時間計測フロー

6.33.2 タイマ／RTCコマンド一覧

表６－３３にタイマ／ＲＴＣコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－３３　タイマ／ＲＴＣコマンド一覧

コマンド名	機能	備考
startGetIPTime	タイマスタート	
timeGetIPTime	タイマ時間読出し	
stopGetIPTime	タイマストップ	
IPSleep	タイマウェイト	
SetIPTimeMode	時間計測モード設定	
SetIPTime	時刻の登録	
GetIPTime	時刻の取得	

6.34 SCIコントロールコマンド

画像処理コマンドではシリアル通信のコマンドを用意しています。

6.34.1 SCIコントロールコマンドによるシリアル通信

ＳＣＩコントロールコマンドによるシリアル通信は、InitSCI、ReadSCIBuffer、WriteSCIBufferコマンドにより行います。
シリアル通信を始める前に、InitSCIコマンドでＳＣＩ(Serial Communication Interface)デバイスの初期化を行い、ReadSCIBufferコマンドでＳＣＩからデータ入力、WriteSCIBufferコマンドでＳＣＩへデータ出力します。
以下にそのフローを示します。

図６－２６　シリアル通信フロー

6.34.2 SCIコントロールコマンド一覧

表６－３４にＳＣＩコントロールコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－３４　ＳＣＩコントロールコマンド一覧

コマンド名	機能	備考
InitSCI	ＳＣＩデバイスの初期化	
SetSCIBuffer	ＳＣＩバッファの設定	
FlushSCIBuffer	ＳＣＩバッファのフラッシュ	
ReadSCIBuffer	ＳＣＩバッファからのデータリード	
ReadSCIBufferWithTimeout	ＳＣＩバッファからのデータリード(ﾀｲﾑｱｳﾄ付)	
WriteSCIBuffer	ＳＣＩへのデータ出力	
IsSCIBuffer	ＳＣＩバッファデータ数の取得	
IsSCIError	ＳＣＩ通信エラーの取得	
IsSCIBufferFull	ＳＣＩバッファフルエラーの取得	
SetCOMMode	カメラリンク用シリアルポート切替え	

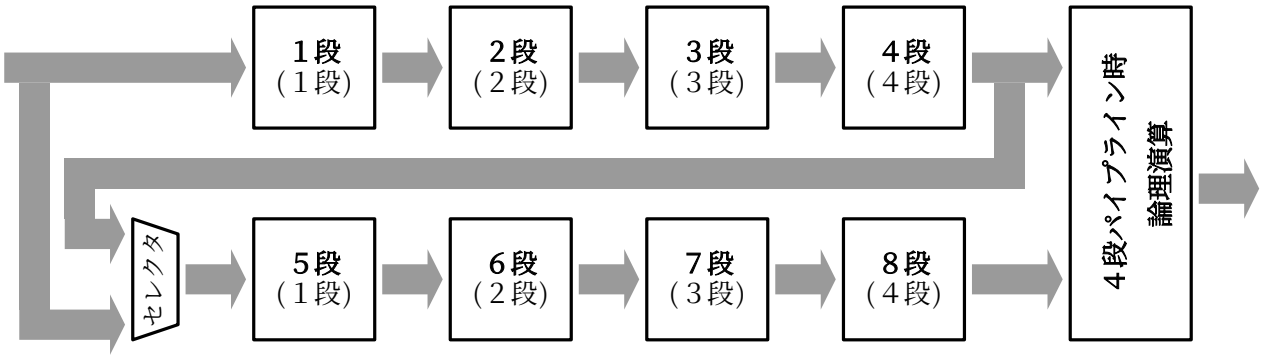
6．35　2値パイプラインフィルタコマンド

２値パイプラインフィルタコマンドは、２値画像に対し、３×３カーネルを用いてノイズ除去、膨張、収縮、輪郭の４つの画像処理を組み合わせて、最大８段までのフィルタ処理のパイプライン処理を行います。
処理はすべてハードウェアで処理されます。

ＳＶＰ－３３０では、ハードウェアの制約により「２値パイプラインフィルタコマンド」は、サポートされていません。

6．35．1　2値パイプラインフィルタ処理の概要

２値パイプラインフィルタは、１段毎に、ノイズ除去、各種膨張、各種収縮、輪郭抽出処理を設定することができます。また、８段パイプラインを４段×２の構成にして、４段目の出力結果を論理演算して出力することもできます。

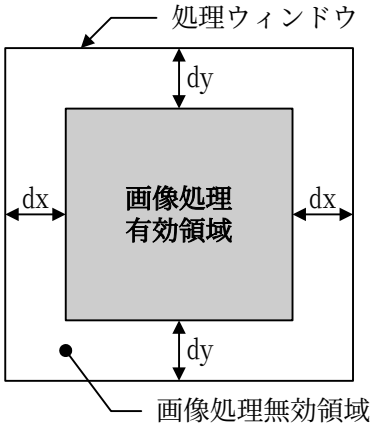


注：（ｘ段）は４段パイプライン×２のとき

図６－２７　２値パイプラインフィルタ処理

6．35．2　2値パイプラインフィルタ処理領域

２値パイプラインフィルタ処理は、３×３カーネルを用いて画像処理を行うため、画像処理対象領域の境界部分に画像処理結果無効の領域が生じます。この画像処理無効範囲は、パイプラインの段数に依存します。



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段パイプライン時は４段の値になります

図６－２８　２値パイプラインフィルタ処理領域

6.35.3 ウィンドウの設定

２値パイプラインフィルタコマンドは任意のウィンドウが設定できます。設定するウィンドウの種類はソース画面はＳＲＣ０_ＷＩＮ（ソース０）でデスティネーション画面は、ＤＳＴ_ＷＩＮ（デスティネーション）ウィンドウです。

6.35.4 画面データタイプ

２値パイプラインフィルタを行う画面（ソース画面）の画面データタイプは２値で、「０」または「２５５」のデータのみの画像データである必要があります。それ以外では正常に動作しない場合があるので注意が必要です。
また、２値パイプラインのフィルタコマンド実行後のディスティネーション画面は、すべて２値のデータタイプに設定されます。

6.35.5 2値パイプラインフィルタコマンド一覧

表６－３５に２値パイプラインフィルタコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－３５　２値パイプラインフィルタコマンド一覧

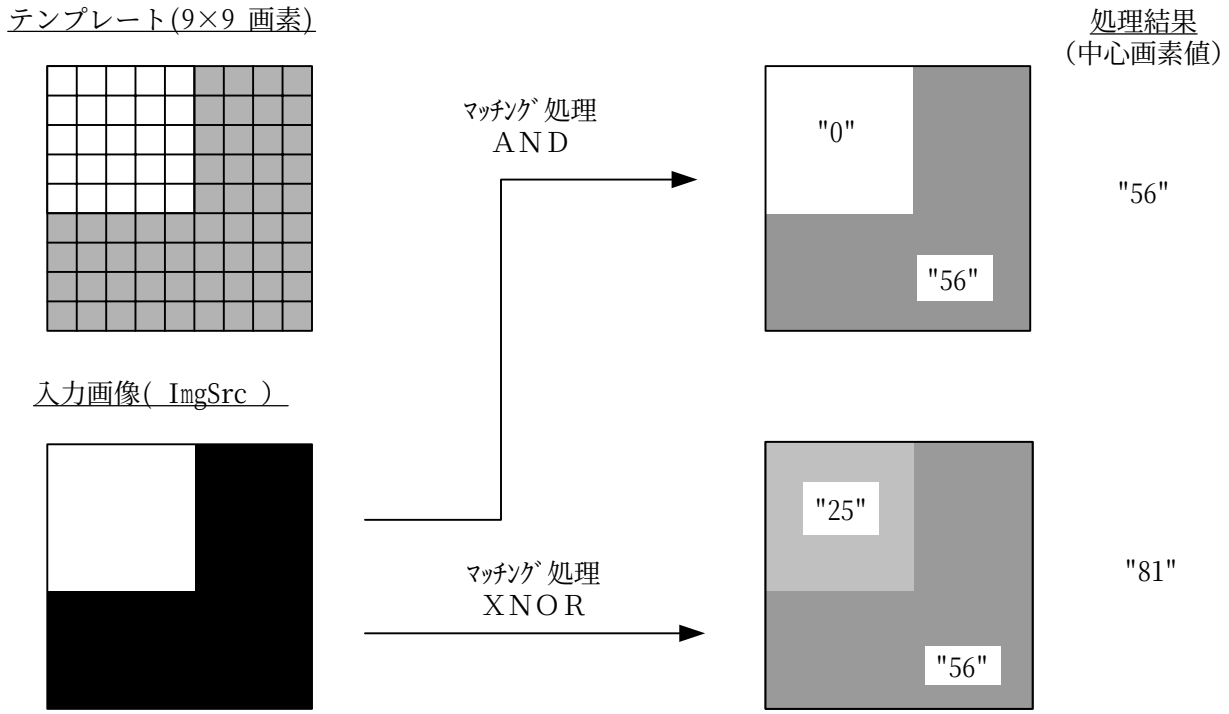
コマンド名	機能	備考
SetTrsPipelineFLTMode	パイプラインフィルタのモード設定	
IP_TrsPipelineFLT	２値パイプラインフィルタの実行	

6．36　2値マッチングフィルタコマンド

２値マッチングフィルタコマンドは、２値画像に対してテンプレートでマッチングフィルタ処理を行い、その処理結果を出力します。
処理はすべてハードウェアで処理されます。
ＳＶＰ－３３０では、ハードウェアの制約により「２値マッチングフィルタコマンド」は、サポートされていません。

6．36．1　2値マッチングフィルタ処理の概要

２値マッチングフィルタは、９×９画素のテンプレートでＡＮＤ／ＸＮＯＲのマッチングフィルタ処理を行います。処理結果は、”１”となる画素の総数です。
また、IP_BinMatchFLTコマンド以外は、規定のテンプレートでＡＮＤのマッチングフィルタ処理を行い、処理結果を指定したしきい値で２値化して出力します。



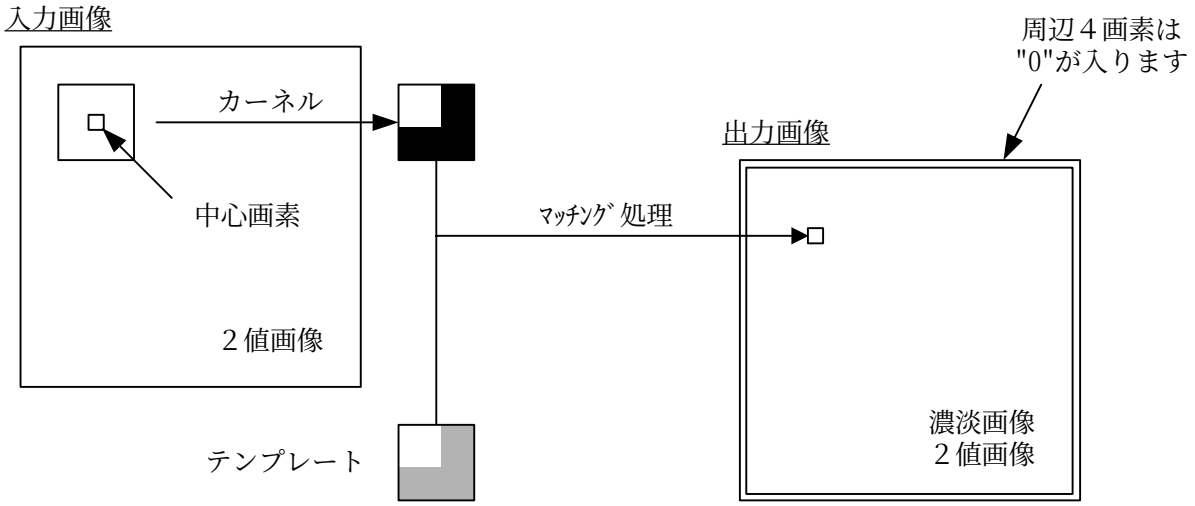


図６－２９　２値マッチングフィルタ処理

6.36.2 2値マッチングフィルタ処理領域

２値マッチングフィルタ処理は、９×９テンプレートを用いて画像処理を行うため、画像処理対象領域の周辺４画素に画像処理結果無効の領域が生じます。この画像処理無効領域は”０”となります。

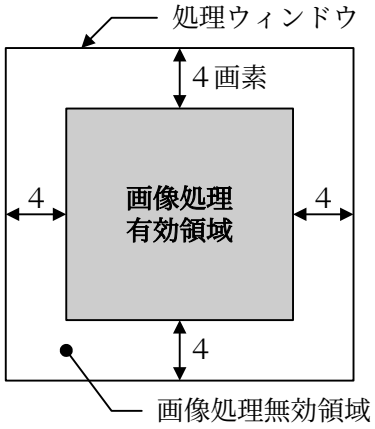


図６－３０　２値マッチングフィルタ処理領域

6.36.3 ウィンドウの設定

２値マッチングフィルタコマンドは任意のウィンドウが設定できます。設定するウィンドウの種類はソース画面はＳＲＣ０_ＷＩＮ（ソース０）でデスティネーション画面は、ＤＳＴ_ＷＩＮ（デスティネーション）ウィンドウです。

6.36.4 画面データタイプ

２値マッチングフィルタを行う画面（ソース画面）の画面データタイプは２値です。それ以外では、正常に動作しない場合があるので注意が必要です。また、２値マッチングフィルタコマンド実行後のデスティネーション画面は、符号なし８ビットまたは２値に設定されます。

6.36.5 2値マッチングフィルタコマンド一覧

表６－３７に２値マッチングフィルタコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－３７　２値マッチングフィルタコマンド一覧

コマンド名	機能	備考
SetBinMatchTemplate	２値画像マッチングフィルタテンプレート登録	
IP_BinMatchFLT	２値画像マッチングフィルタ実行	
IP_BinarizePTM3x3	２値画像マッチングフィルタ（３×３）	
IP_BinarizePTM5x5	２値画像マッチングフィルタ（５×５）	
IP_BinarizePTM7x7	２値画像マッチングフィルタ（７×７）	
IP_BinarizePTM9x9	２値画像マッチングフィルタ（９×９）	
IP_BinarizePTM3x5	２値画像マッチングフィルタ（３×５）	
IP_BinarizePTM5x7	２値画像マッチングフィルタ（５×７）	
IP_BinarizePTM7x9	２値画像マッチングフィルタ（７×９）	

6．37 画像ファイリングコマンド

画像処理コマンドでは、ファイルへの画像データの読み出し／書き込みを行うコマンドを用意しています。

6．37．1 ウィンドウの設定

画面内の転送領域はウインドウ（ＳＹＳ_ＷＩＮ）で指定できます。ただし、データ転送領域のＸ方向始点座標を偶数，終点座標を奇数に設定する必要があります。

6．37．2 画面データタイプ

ソース画面のデータタイプは保存されません。このため、画像メモリに書き込まれた映像データのデータタイプは転送先画面のデータタイプに依存します。データタイプの異なる画面を混在して使用する場合は、ChangeImgDataType() で任意のデータタイプに設定してください。

6．37．3 画像ファイリングコマンド一覧

表６－３７に画像ファイリングコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－３７　画像ファイリングコマンド一覧

コマンド名	機能	備考
LoadBMPFile	ＢＭＰファイルのロード	
SaveBMPFile	ＢＭＰファイルのセーブ	

6.38 拡張コンボリューションコマンド

拡張コンボリューションコマンドは、ソース画面の指定領域内の画素に対し注目画素を中心とした５×５近傍または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。この荷重係数を適切に選択することで平滑化、輪郭強調、最小値／最大値フィルタといった処理を行います。
処理はすべてハードウェアで処理されます。

6.38.1 ウィンドウの設定

拡張コンボリューションコマンドは任意のウィンドウが設定できます。設定するウィンドウの種類はソース画面はＳＲＣ０＿ＷＩＮ（ソース０）でデスティネーション画面は、ＤＳＴ＿ＷＩＮ（デスティネーション）ウィンドウです。

6.38.2 画面データタイプ

拡張コンボリューションを行う画面（ソース画面）の画面データタイプは、符号なし８ビットか符号付き８ビットです。符号なし８ビットか符号付き８ビットで処理結果が異なる場合があるので注意が必要です。また、コンボリューションコマンド実行後のデスティネーション画面は、すべてシステムデータタイプに設定されます。

6.38.3 拡張コンボリューションコマンド一覧

表６－３８に拡張コンボリューションコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－３８　拡張コンボリューションコマンド一覧

コマンド名	機能	備考
IP_SmoothFLT5x5	平滑化（５×５）	
IP_SmoothFLT7x7	平滑化（７×７）	
IP_EdgeFLT5x5	濃淡画像輪郭強調（５×５）	
IP_EdgeFLT7x7	濃淡画像輪郭強調（７×７）	
IP_MinFLT5x5	局所最小値フィルタ（５×５）	
IP_MinFLT44	局所最小値フィルタ（５×５、４４連結）	
IP_MinFLT48	局所最小値フィルタ（５×５、４８連結）	
IP_MinFLT88	局所最小値フィルタ（５×５、８８連結）	
IP_MaxFLT5x5	局所最大値フィルタ（５×５）	
IP_MaxFLT44	局所最大値フィルタ（５×５、４４連結）	
IP_MaxFLT48	局所最大値フィルタ（５×５、４８連結）	
IP_MaxFLT88	局所最大値フィルタ（５×５、８８連結）	
IP_SmoothFLT5x5Ext	平滑化（５×５）	
IP_SmoothFLT7x7Ext	平滑化（７×７）	
IP_EdgeFLT5x5Ext	濃淡画像輪郭強調（５×５）	
IP_EdgeFLT7x7Ext	濃淡画像輪郭強調（７×７）	

6.39 拡張画像処理コマンド

拡張画像処理コマンドは、ソース画面の指定領域内の画素に対し注目画素を中心とした３×３近傍の局所領域で指定した手法により平滑化、輪郭強調といった処理を行います。
処理はすべてハードウェアで処理されます。

6.39.1 ウィンドウの設定

拡張画像処理コマンドは任意のウィンドウが設定できます。設定するウィンドウの種類は、ソース画面はＳＲＣ０＿ＷＩＮ（ソース０）でデスティネーション画面は、ＤＳＴ＿ＷＩＮ（デスティネーション）ウィンドウです。

6.39.2 画面データタイプ

拡張画像処理を行う画面（ソース画面）の画面データタイプは、符号なし８ビットか符号付き８ビットです。符号なし８ビットか符号付き８ビットで処理結果が異なる場合があるので注意が必要です。また、拡張画像処理コマンド実行後のデスティネーション画面は、システムデータタイプまたは２値に設定されます。

6.39.3 拡張画像処理コマンド一覧

表６－３９に拡張画像処理コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－３９　拡張画像処理コマンド一覧

コマンド名	機能	備考
IP_SmoothFLTExt	平滑化フィルタ（拡張版）	
IP_SmoothFLTAbsExt	輪郭強調フィルタ（拡張版）	
IP_Sobel	輪郭強調（ソーベル）	
IP_SobelBinaerize	輪郭強調２値化（ソーベル２値化）	
IP_RegisterLUT	濃度変換テーブルの登録	
IP_Prewitt	輪郭強調（プレビット）	
IP_PrewittBinarize	輪郭強調２値化（プレビット２値化）	

6.40 ランレングス･ラベリングコマンド

ランレングス・ラベリングは、２値画像の物体に対してラベル（番号）付けを行う処理です。２値のソース画面の物体に対してラベル付けしたラベル値をデスティネーション画面に出力するとともに、基本特徴量の抽出を行います。

画像処理については、ラベリングコマンドを参照して下さい。

6.40.1 ウィンドウの設定

ランレングス・ラベリングコマンドは任意のウィンドウが設定できます。設定するウィンドウの種類は、ソース画面はＳＲＣ０_ＷＩＮ（ソース０）でデスティネーション画面は、ＤＳＴ_ＷＩＮ（デスティネーション）ウィンドウです。

6.40.2 画面データタイプ

ラベル付けコマンドでは、ラベル付けを行う画面（ソース画面）の画面データタイプは２値データで、「０」または「２５５」のデータのみの画像データである必要があります。それ以外では正常に動作しないことがあるので注意が必要です。また、ラベル付けコマンド実行後のデスティネーション画面は、すべて符号なし８ビットに設定されます。

6.40.3 ランレングス･ラベリングコマンド一覧

表６－４０にランレングス・ラベリングコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－４０　ランレングス・ラベリングコマンド一覧

コマンド名	機能	備考
IP_Label4byRL	ﾗﾝﾚﾝｸﾞｽ･ﾗﾍﾞﾘﾝｸﾞ処理（４連結）	
IP_Label8byRL	ﾗﾝﾚﾝｸﾞｽ･ﾗﾍﾞﾘﾝｸﾞ処理（８連結）	
IP_Label4byRLwithAreaFLT	面積ﾌｨﾙﾀ付ﾗﾝﾚﾝｸﾞｽ･ﾗﾍﾞﾘﾝｸﾞ処理（４連結）	
IP_Label8byRLwithAreaFLT	面積ﾌｨﾙﾀ付ﾗﾝﾚﾝｸﾞｽ･ﾗﾍﾞﾘﾝｸﾞ処理（８連結）	
IP_Label4byRLwithAreaFLTSort	面積ﾌｨﾙﾀ付ﾗﾝﾚﾝｸﾞｽ･ﾗﾍﾞﾘﾝｸﾞ処理（ｿｰﾄ、４連結）	
IP_Label8byRLwithAreaFLTSort	面積ﾌｨﾙﾀ付ﾗﾝﾚﾝｸﾞｽ･ﾗﾍﾞﾘﾝｸﾞ処理（ｿｰﾄ、８連結）	
IP_Label4byRLExt	ﾗﾝﾚﾝｸﾞｽ･ﾗﾍﾞﾘﾝｸﾞ処理（４連結、拡張）	
IP_Label8byRLExt	ﾗﾝﾚﾝｸﾞｽ･ﾗﾍﾞﾘﾝｸﾞ処理（８連結、拡張）	
IP_Label4byRLwithAreaFLTExt	面積ﾌｨﾙﾀ付ﾗﾝﾚﾝｸﾞｽ･ﾗﾍﾞﾘﾝｸﾞ処理（４連結、拡張）	
IP_Label8byRLwithAreaFLTExt	面積ﾌｨﾙﾀ付ﾗﾝﾚﾝｸﾞｽ･ﾗﾍﾞﾘﾝｸﾞ処理（８連結、拡張）	
IP_Label4byRLwithArea FLTshortExt	面積ﾌｨﾙﾀ付ﾗﾝﾚﾝｸﾞｽ･ﾗﾍﾞﾘﾝｸﾞ処理（ｿｰﾄ、４連結、拡張）	
IP_Label8byRLwithArea FLTshortExt	面積ﾌｨﾙﾀ付ﾗﾝﾚﾝｸﾞｽ･ﾗﾍﾞﾘﾝｸﾞ処理（ｿｰﾄ、８連結、拡張）	
IP_LabelCombine	統合ラベリング	

6.41 ビデオ拡張コマンド

画像処理コマンドでは、カメラから入力する映像データをＬＵＴ（ルックアップテーブル）で変換するビデオ拡張コマンドを用意しています。ビデオ拡張コマンドは、１０ビットの入力映像データを８ビットに濃度変換する場合やミラー映像入力する場合に使います。

6.41.1 カメラ入力映像のデータ変換

カメラから入力する映像のデータ変換は、WriteVideoLUTでビデオ入力濃度変換メモリへのデータ書き込みを行い、SetVideoOptでビデオオプションを設定後、映像入力を行います。
以下にそのフローを示します。

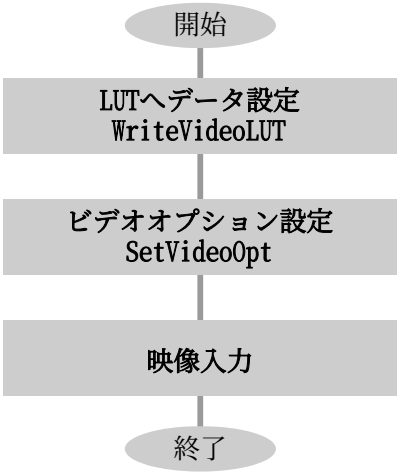


図６－３１　カメラ映像入力のデータ変換フロー

6.41.2 ビデオ拡張コマンド一覧

表６－４１にビデオ拡張コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－４１　ビデオ拡張コマンド一覧

コマンド名	機能	備考
WriteVideoLUT	ビデオ入力ＬＵＴへのデータ書き込み	
SetVideoOpt	ビデオオプションの設定	
GetVideoOpt	ビデオオプションの参照	

6．42 線分化コマンド

画像処理コマンドでは線分化コマンドを用意しています。線分化は、２値画像の物体に対して線分列外周座標を抽出し、線分列外周座標から特徴量の抽出を行います。

6．42．1 線分化処理の概要

線分列外周座標の抽出は、ExtractPolylineコマンドで行います。ExtractPolylineコマンドでは、指定画面の探索開始座標(sx,sy)からラスタースキャンで指定した濃度の探索対象物体の外周探索開始座標を取得し、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て、線分列外周座標を取得します。

対象物体の特徴量は、PolyArea、PolyPerm、PolyGrav，PolyFeatureコマンドで抽出します。

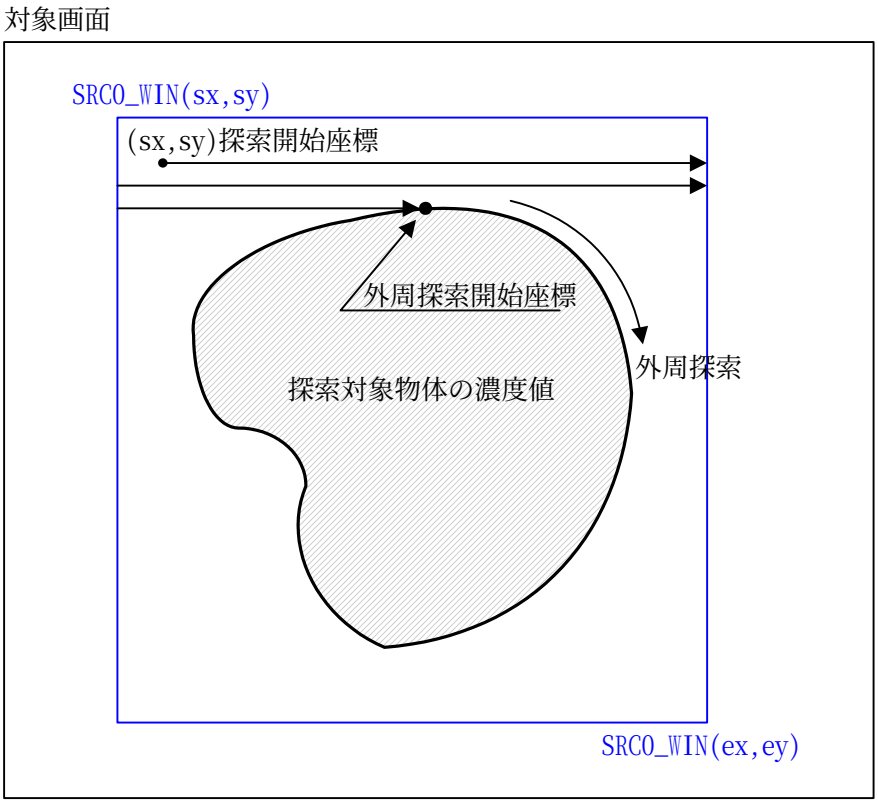


図６－３２　線分化処理

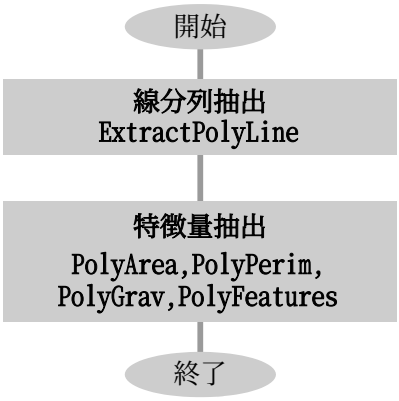


図６－３３　線分化処理フロー

6．42．2　ウィンドウの設定

線分化コマンドは任意のウィンドウが設定できます。設定するウィンドウの種類はソース画面はＳＲＣ０＿ＷＩＮ（ソース０）です。デスティネーション画面はなく、８バイトデータ配列や指定された構造体配列に処理結果が格納されます。
また、処理結果として出力される座標は画面絶対座標で出力されますので注意して下さい。

6．42．3　画面データタイプ

線分化コマンドでは、ラベル付けを行う画面（ソース画面）の画面データタイプは２値データで、「０」または「２５５」のデータのみの画像データである必要があります。それ以外では正常に動作しないことがあるので注意が必要です。

6．42．4　線分化コマンド一覧

表６－４２に線分化コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－４２　線分化コマンド一覧

コマンド名	機能	備考
ExtractPolyLine	線分列抽出	
PolyArea	線分列から面積抽出	
PolyPerim	線分列からの周囲長抽出	
PolyGrav	線分列から重心抽出	
PolyFeatures	線分列から形状特徴量抽出	

6．43 2値画像の穴埋めコマンド

画像処理コマンドでは穴埋めコマンドを用意しています。穴埋めは、２値画像の物体に対してラベリング処理を行い、指定した面積しきい値の条件を満たすオブジェクト（穴）を塗りつぶす処理を行います。

6．43．1 2値画像の穴埋め処理

図６－３４は、対象画面中の穴埋め対象物が”白”オブジェクト、穴が”黒”オブジェクトの例です。
穴埋め処理は、”黒”オブジェクトに対してラベリング処理を行い、面積が最小面積以上最大面積以下のものを塗りつぶします。
尚、穴埋め処理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。

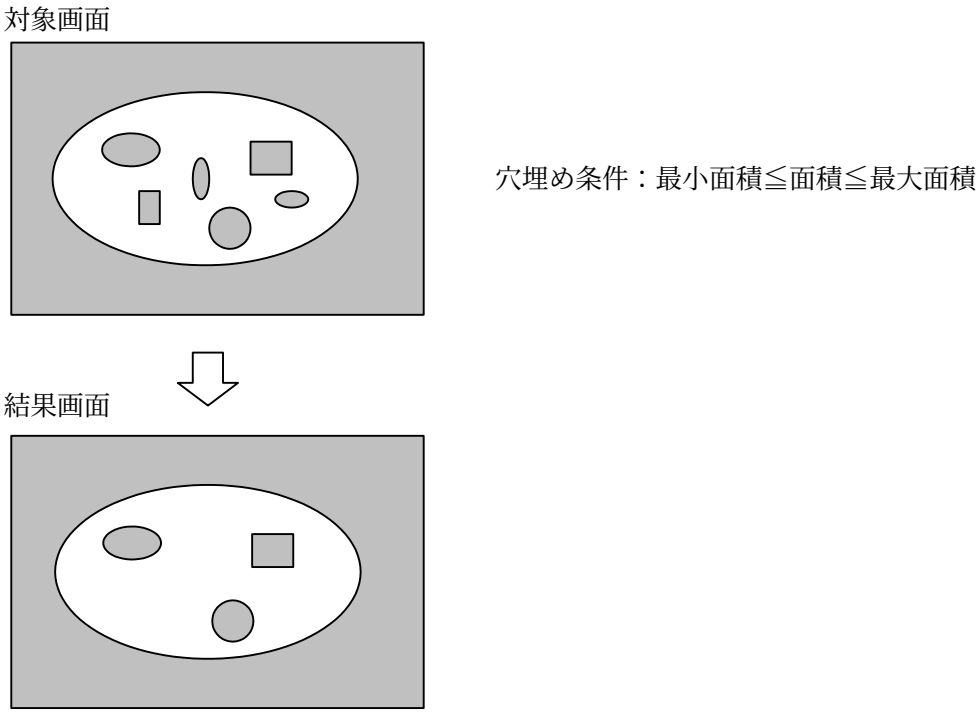


図６－３４　２値画像穴埋め処理

6．43．2 ウィンドウの設定

２値画像の穴埋めコマンドは任意のウィンドウが設定できます。設定するウィンドウの種類はソース画面はＳＲＣ０_ＷＩＮ（ソース０）で、デスティネーション画面はＤＳＴ_ＷＩＮ（デスティネーション）ウィンドウです。

6．43．3 画面データタイプ

２値画像の穴埋めを行う画面（ソース画面）の画面データタイプは２値で、「０」または「２５５」のデータのみの画像データである必要があります。それ以外では正常に動作しない場合があるので注意が必要です。また、２値画像の穴埋めコマンド実行後のデスティネーション画面はすべて２値のデータタイプに設定されます。

6．43．4 2値画像の穴埋めコマンド一覧

表６－４３に２値画像の穴埋めコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－４３　２値画像の穴埋めコマンド一覧

コマンド名	機能	備考
IP_FillHole	２値画像穴埋め	
IP_FillHoleExt	２値画像穴埋め（拡張）	

6.44 RGBLUT変換コマンド

画像処理コマンドではＲＧＢＬＵＴ変換コマンドを用意しています。ＲＧＢＬＵＴ変換コマンドは、ＲＧＢ画像に対して指定されたＬＵＴ（ルックアップテーブル）で画素変換を行うコマンドです。

ＳＶＰ－３３０では、ハードウェアの制約により「ＲＧＢＬＵＴ変換コマンド」は、サポートされていません。

6.44.1 RGB画像の画素変換

以下に、ＲＧＢ画像のＬＵＴ（ルックアップテーブル）による画素変換のフローを示します。

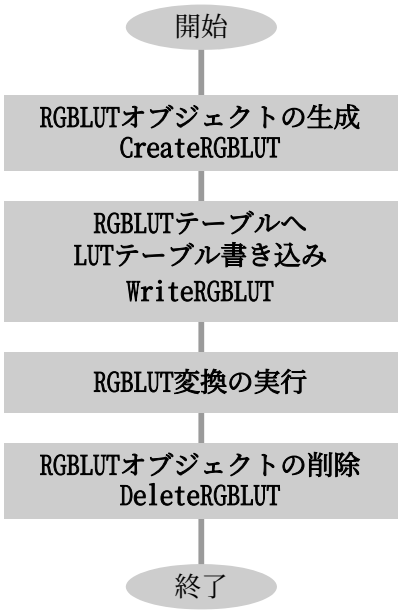


図６－３５　ＲＧＢ画像の画素変換フロー

6.44.2 ウィンドウの設定

ＲＧＢＬＵＴ変換コマンドは任意のウィンドウが設定できます。設定するウィンドウの種類は、ソース画面はＳＲＣ０_ＷＩＮ（ソース０）でデスティネーション画面は、ＤＳＴ_ＷＩＮ（デスティネーション）ウィンドウです。

6.44.3 画面データタイプ

ＲＧＢＬＵＴ変換を行う画面（ソース画面）の画面データタイプは、符号なし８ビットか符号付き８ビットです。符号なし８ビットか符号付き８ビットで処理結果が異なる場合があるので注意が必要です。また、ＲＧＢＬＵＴ変換コマンド実行後のデスティネーション画面は、すべてシステムデータタイプに設定されます。

6.44.4 RGBLUT変換コマンド一覧

表６－４４にＲＧＢＬＵＴ変換コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－４４　ＲＧＢＬＵＴ変換コマンド一覧

コマンド名	機能	備考
CreateRGBLUT	RGBLUTオブジェクトの生成	
DeleteRGBLUT	RGBLUTオブジェクトの削除	
SetRGBLUT	RGBLUTテーブルへ値設定	
GetRGBLUT	RGBLUTテーブルから値取得	
GetRGBLUTAddr	RGBLUTテーブルからLUTテーブル先頭ｱﾄﾞﾚｽ取得	
GetRGBLUTSize	RGBLUTテーブルからLUTテーブルサイズ取得	
WriteRGBLUT	RGBLUTテーブルへLUTテーブル書き込み	
ReadRGBLUT	RGBLUTテーブルからLUTテーブル読み出し	
IP_ConvertRGBLUT	RGBLUT変換の実行	
IP_ConvertRGBLUTEx	RGBLUT変換の実行（拡張）	

6．45 擬似カラーコマンド

画像処理コマンドでは擬似カラーコマンドを用意しています。擬似カラーコマンドは、濃淡画像に対して指定した擬似カラー変換テーブルでカラー変換を行うコマンドです。

6．45．1 濃淡画像の擬似カラー変換

濃淡画像のＲＧＢ擬似カラー変換は、図６－３６に示すように、濃淡画像の濃度値に対応したＲ画像、Ｇ画像、Ｂ画像の画像データテーブルを作成し、この画像データテーブルに従って擬似カラー画像に変換します。

以下に、濃淡画像のＲＧＢ画像データテーブルによる擬似カラー変換のフローを示します。

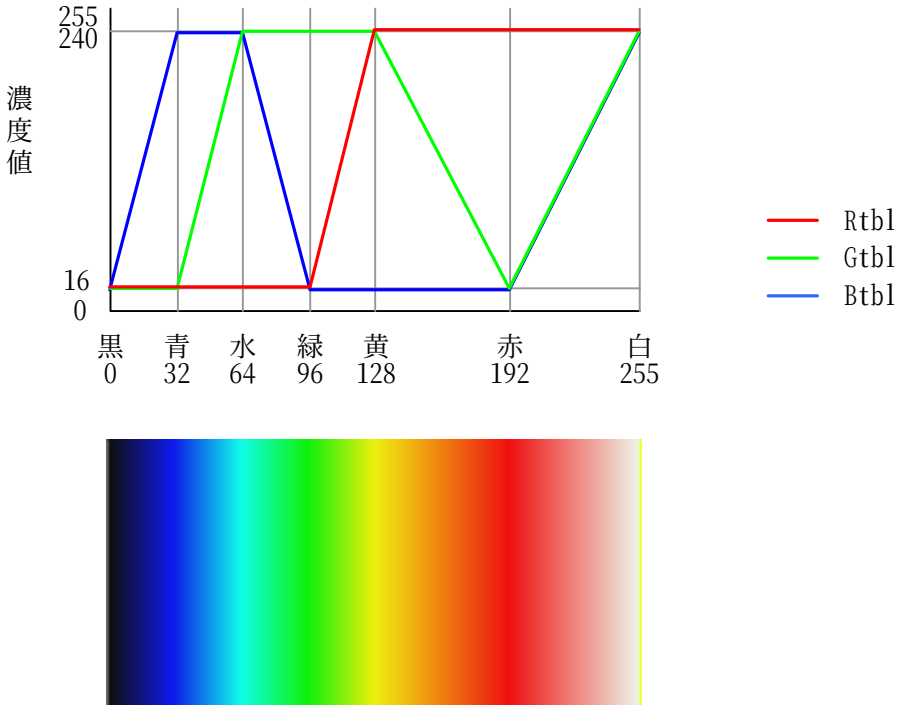


図６－３６　濃淡画像データとＲＧＢ画像データの対応

図６－３７　濃淡画像の擬似カラー変換フロー

6.45.2 ウィンドウの設定

擬似カラー変換コマンドは、ウィンドウ設定に関係なく画面サイズとなります。

6.45.3 画面データタイプ

擬似カラー変換を行う画面（ソース画面）の画面データタイプは、符号なし８ビットか符号付き８ビットです。符号なし８ビットか符号付き８ビットで処理結果が異なる場合があるので注意が必要です。また、擬似カラー変換コマンド実行後のデスティネーション画面は、すべてシステムデータタイプに設定されます。

6.45.4 擬似カラー変換コマンド一覧

表６－４５に擬似カラー変換コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－４５　擬似カラー変換コマンド一覧

コマンド名	機能	備考
IP_PseudoColor	擬似カラー変換	
SetPseudoColor	RGB擬似カラーテーブル作成	
SetPseudoColorExt	RGB擬似カラーテーブル作成(拡張)	

6.46 WDTコマンド

画像処理コマンドではシステムが正常動作しているかをチェックするためのＷＤＴ（Watch Dog Timer）コマンドを用意しています。本コマンドは、オンボードＣＰＵ側専用のコマンドです。

6.46.1 WDTコマンドによるシステムチェック

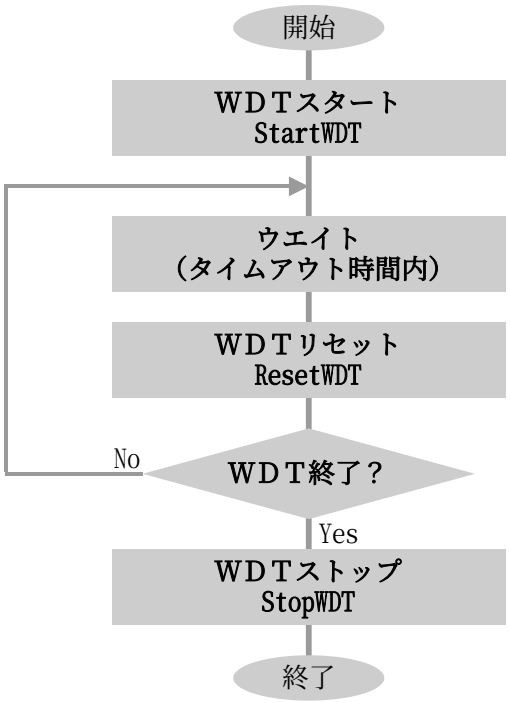
ＷＤＴコマンドによるシステムの正常動作チェックは、StartWDT, ResetWDT, StopWDT, GetWDTStatusコマンドにより行います。

システムの正常動作チェックは、StartWDTコマンドでタイマを起動し、タイムアウト以内の間隔でResetWDTコマンドを実行します。システム異常でResetWDTコマンドが実行されない場合は、約600～1500ms後にＷＤＴエラーが発生してシステムリセットされます。

システムリセットがＷＤＴエラーによるものかどうかは、システムの再起動後、GetWDTStatusコマンドでＷＤＴステータスを確認します。

ＷＤＴを停止させるには、StopWDTコマンドを実行します。

以下にそのフローを示します。



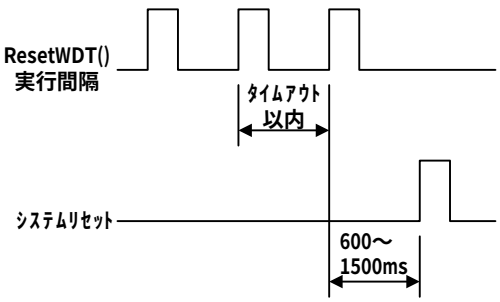


図６－３８　ＷＤＴ実行フロー

6.46.2 WDTコマンド一覧

表６－４６にＷＤＴコマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表６－４６　ＷＤＴコマンド一覧

コマンド名	機能	備考
StartWDT	ＷＤＴスタート	
ResetWDT	ＷＤＴリセット	
StopWDT	ＷＤＴストップ	
GetWDTStatus	ＷＤＴステータスの読み出し	

第7章

インテリジェントモジュール

画像処理コマンドでは画像処理アプリケーションのモジュール化として、
・インテリジェントモジュール
・割込起動モジュール
という２通りの手法を提供しています。
本章では、インテリジェントモジュールについて説明します。第３章で概略を説明していますので、そちらも参照して下さい。

7．1 モジュール化の概要

ＳＶＰ－３３０では、実行モジュールはＳＶＰ－３３０のＲＯＭやＲＡＭ上にあり、ＰＣから実行する画像処理コマンドでの処理は画像処理プロセッサとオンボードＣＰＵにより行われています。
ＰＣ（パソコン）側から実行される画像処理コマンドでは、コマンド１つ１つについて以下のフローに従いオンボードＣＰＵで処理が実行されます。そのため、画像処理コマンドを実行するたびにオンボードＣＰＵの処理時間以外にＬＡＮ経由のデータ転送やコマンド起動にオーバヘッドが加わります。

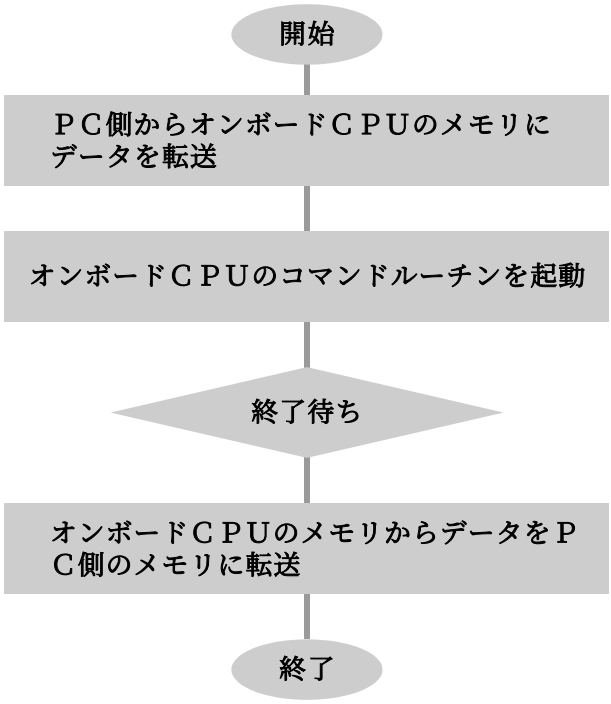


図７－１　コマンド実行フロー

そこで、画像処理コマンドでは、ＰＣとのＩ／Ｆでのオーバヘッド削減と処理の分散化のために、ユーザーオリジナルの画像処理アプリケーションをオンボードＣＰＵで実行できるように
・インテリジェントモジュール
・割込起動モジュール
という２つのフレームモデルを実装しています。このフレームモデルの特徴は、Ｃ言語の関数形式でＰＣ側からオンボードＣＰＵ側に簡単にデータを渡すことができることと、モジュールの実行制御を簡単に行うことができるということです。

また、オンボードＣＰＵで実行されるユーザーオリジナルの画像処理アプリケーションは、オンボードＣＰＵ（ＳＨ４）のコンパイラでコンパイルし実行オブジェクトを作成します。

7．2 インテリジェントモジュールの概要

インテリジェントモジュールとは、画像処理コマンドを複数組み合わせて作成するユーザーオリジナルの画像処理コマンドのことです。

ＳＶＰ－３３０では、ＰＣから実行する画像処理は画像処理プロセッサとオンボードＣＰＵにより行なっているため、ユーザーの画像処理アプリケーションの一部をインテリジェントモジュールとして実装することにより、処理の分散化を実現できます。

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登録し実行されます。最大２５６モジュールの登録が可能です。

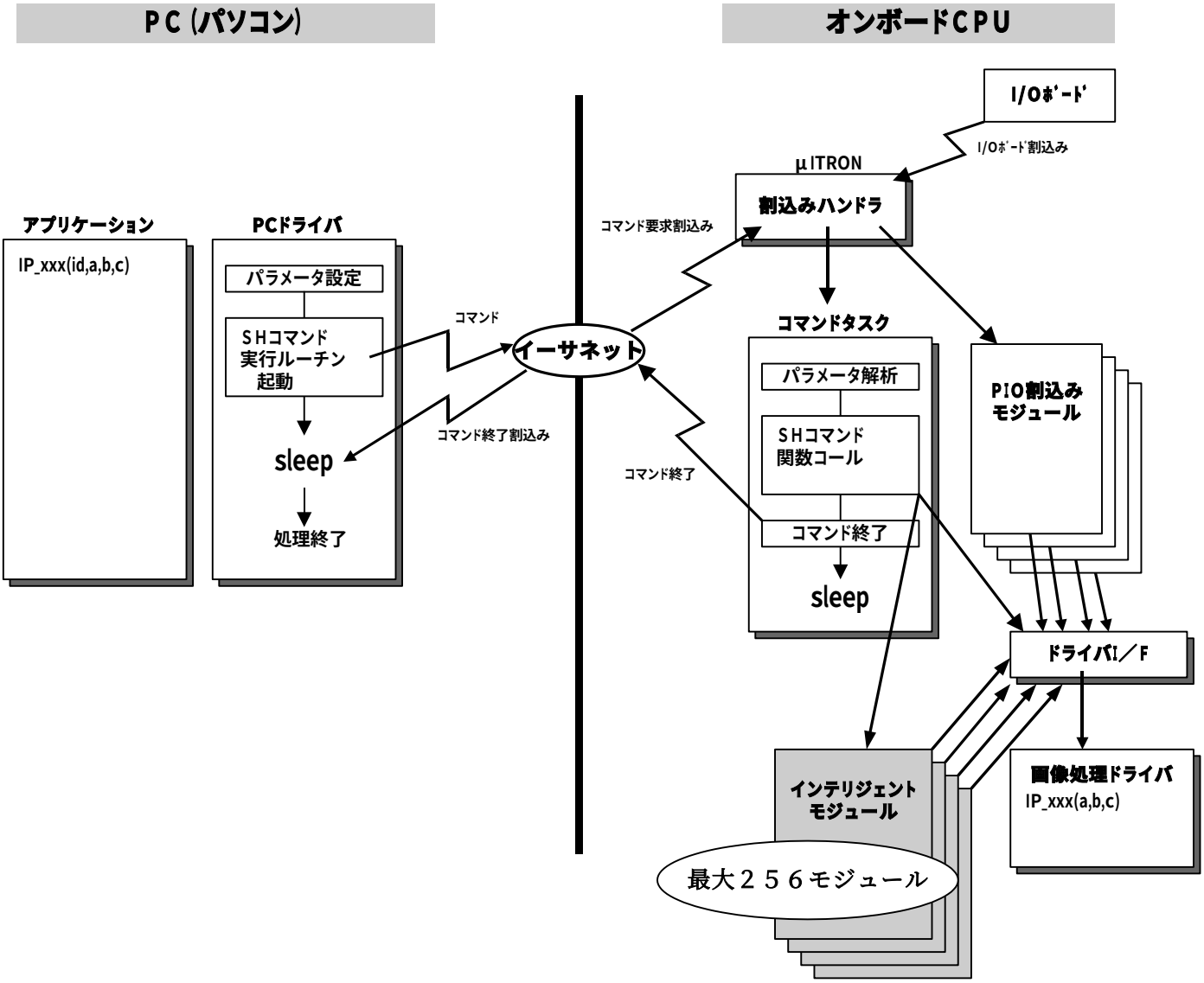


図７－２　インテリジェントモジュールの仕組み

7．3 インテリジェントモジュールの作成

本項では、オンボードＣＰＵ側のプログラムについて説明します。

7．3．1 プログラム

ＰＣ（パソコン）側から起動されるオンボードＣＰＵのプログラムは、Ｃ言語の関数（サブルーチン）形式で記述して下さい。また、疑似命令「#pragma」でセクションを指定して下さい。リンク時に指定したセクション名でプログラムの開始アドレスを直接指定できます。

```
#pragma   section   module

void imgprc(int a,float b,short *tbl)
{
   ..........
   ..........
}
```

関数名は任意でパラメータはint型、float型、ポインタ型で０から最大３２個まで指定できます。これらのパラメータは、ＰＣ側のプログラムで「モジュールサポートコマンド」によりデータが設定されます。
それ以外は、ＰＣでのプログラムと同一にして下さい。

7．3．2 インクルードファイル

ＰＣコマンドの場合と同じように、

vpxdef.h,vpxsys.h,vpxfnc.h

をインクルードします。それ以外に

vpxcnv.h

をインクルードしてください。vpxcnv.hはＰＣ側にもダミーで用意してあります。ＰＣ側でプログラムのデバッグを行う場合、そのままvpxcnv.hをインクルードして下さい。

7．3．3 メモリマップ

オンボードＣＰＵシステムメモリでユーザーが使用できるメモリ空間は

0x8C80:0000 ～ 0x8CFF:FFFF

の８Ｍバイトです。この領域にプログラム、スタティックデータなどをマッピングするように、リンク時に指定して下さい。また、この領域内にmalloc()等のヒープ領域やＲＡＭディスク領域も含んでいますので、「プログラム領域＋ヒープ等の領域」が８Ｍバイトを超えないようにマッピングして下さい。

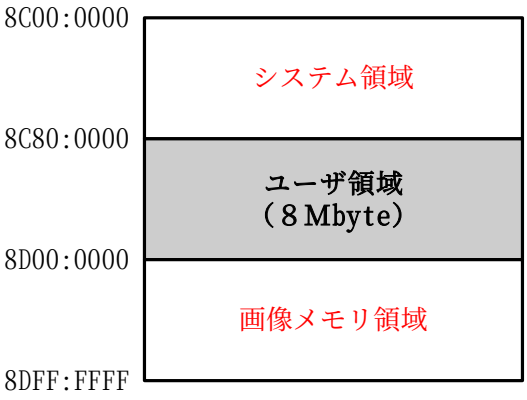


図７－３　ＲＡＭメモリマップ

7．3．4 プログラムローダー

ユーザー領域(0x8C80:0000 ～ 0x8CFF:FFFF)は、システムにより管理されています。
プログラムのローダーは、実行ファイル(.abs)からプログラムがロードされる領域を計算し、その領域をユーザ領域から確保し、その領域に実行コードをダウンロードします。実行コードがダウンロードされる領域がすでに確保されている場合、ローダーは実行ファイルの実行コードをダウンロードできません。その場合は、すでに実行しているアプリケーションを終了しその領域を開放するか、アドレスのマッピングを変更してください。また、この領域内にmalloc()等のヒープ領域やＲＡＭディスク領域も含んでいます。特に、ＲＡＭディスクを確保した場合は、0x8C80:0000番地からマッピングできない場合がありますので、注意してください。

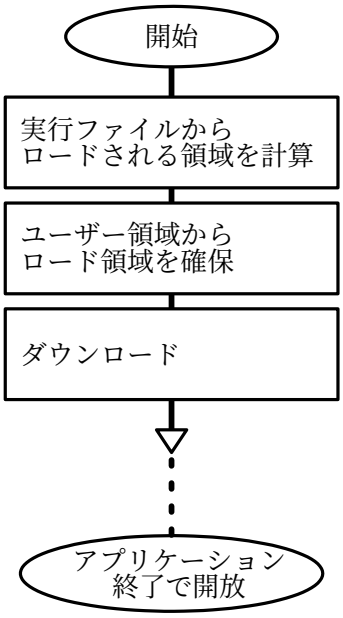


図７－４ ロードフロー

7．3．5 スタック

プログラムで使用できるスタック領域は６４Ｋバイトです。これ以上のスタックを使用するとシステム領域を破壊してしまいますので注意して下さい。

7．3．6 プロジェクトのビルド

ＳＨ４のコンパイラでコンパイル、リンクを行いＥＬＦ形式の実行ファイル(.abs)ファイルをビルドします。
なお、ＳＨ４コンパイラのプロジェクトの設定等の詳細は、「ＳＨＣ Ｃｏｍｐｉｌｅｒ設定ガイド」を参照して下さい。

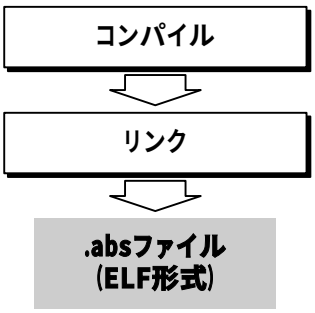


図７－５ プロジェクトのビルド

7．3．7 デバッグ

オンボードＣＰＵインテリジェントモジュールの開発環境や開発手法については第２章で説明していますので、まず、そちらを参照してください。
オンボードＣＰＵインテリジェントモジュールのデバッグは、ＪＴＡＧ－ＩＣＥを使用して行う方法もありますが、ＰＣ側である程度動作しているモジュールであれば、その必要性は低いと思われます。しかし、データの確認などの簡単なデバッグ作業は、プログラム開発上できるにこしたことはありません。そこで画像処理コマンドでは、オンボードＣＰＵ側にＲＳ２３２Ｃのターミナルを対象にしたprintf()やscanf()などの標準入出力を実装しています。ＳＶＰ－３３０で使用できる標準入出力コマンドは、以下に示します。

表７－１ オンボードＣＰＵ標準入出力関数

関数名	内容	
printf	Ｃの標準関数のprintfと同じです。	
scanf	Ｃの標準関数のscanfと同じです。	
charget	Ｃの標準関数のgetcharと同じです。	
charput	Ｃの標準関数のputcharと同じです。	
malloc	Ｃの標準関数のmallocと同じです。	
free	Ｃの標準関数のfreeと同じです。	

⚠ これらの標準入出力関数を使用する場合は、インクルードファイル「vpxcnv.h,stdio.h,stdlib.h」をＣのソースファイルでインクルードして下さい。

関数の詳細は、コマンドリファレンスのオンボードデバッグサポートコマンドを参照して下さい。

シリアルポートの初期設定を以下に示します。これ以外の設定が必要な場合、InitSCI()コマンドで設定して下さい。

表７－２　シリアルポートの初期設定

項目	初期値	備考
ビットレート	９６００ｂｐｓ	
キャラクタ長	８ｂｉｔ	
パリティ	パリティなし	
ストップビット	１ｂｉｔ	

7．4 インテリジェントモジュールの実行

本項では、オンボードＣＰＵ側のインテリジェントモジュールを実行するためのＰＣ（パソコン）側のプログラムについて説明します。
以下に実行フローを示します。

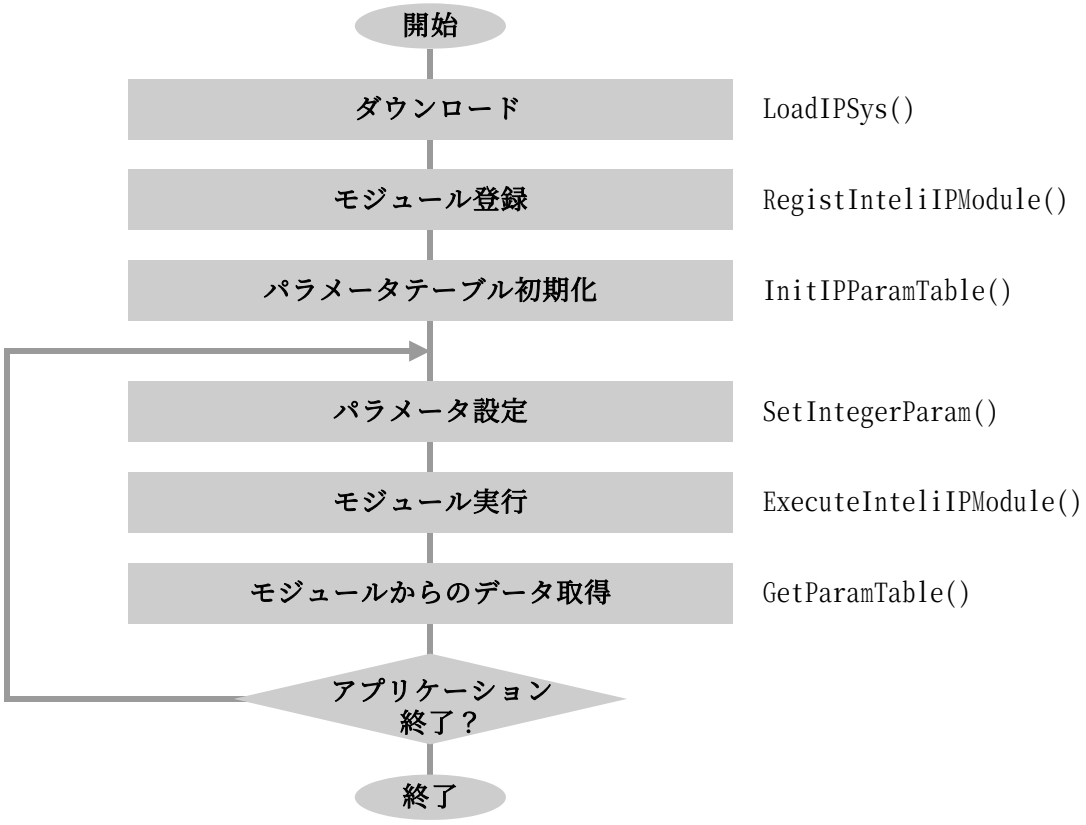


図７－６　モジュール実行フロー

7．4．1 インテリジェントモジュールのダウンロード

７．３項でビルドした「abs」ファイルをLoadIPSys()コマンドを使用してＳＶＰ－３３０にダウンロードします。

```
LoadIPSys(_devID,"imgprc.abs",DIRECT_PATH);
```

また、同じアドレス空間に再度ダウンロードする場合は「UNMAP_APLBITS」オプションを指定して下さい。システムはその空間のマッピングを開放し、再度領域を確保しダウンロードします。

```
LoadIPSys(_devID,"imgprc.abs",UNMAP_APLBITS | DIRECT_PATH);
```

7．4．2 インテリジェントモジュールの登録

ダウンロードしたインテリジェントモジュールをRegistInteliIPModule()コマンドで登録します。

```
RegistInteliIPModule(_devID,MODULE_0,0x8C200000,0);
```

7．4．3 インテリジェントモジュールへのパラメータ渡し

登録したインテリジェントモジュールにパラメータを渡す方法を説明します。

(1) パラメータテーブルの初期化

InitIPParamTable()コマンドでパラメータテーブルを初期化します。

```
MODULE_PARAM_TBL   prmtbl;

InitIPParamTable(_devID,&prmtbl,3,INTELI_MODULE,0,0);
```

(2) パラメータの設定

インテリジェントモジュールのパラメータがint型やfloat型の場合、SetIntegerParam(), SetFloatParam()コマンドでパラメータテーブルにデータを設定します。

```
SetIntegerParam(_devID,&prmtbl,PARAM_1,123);
SetFloatParam(_devID,&prmtbl,PARAM_2,4.567);
```

(3) テーブル（配列）パラメータの設定

インテリジェントモジュールのパラメータが配列の場合、SetParamTable()コマンドでパラメータテーブルにデータを設定します。

```
SetParamTable(_devID,&prmtbl,PARAM_3,&tbl,sizeof(tbl));
```

7．4．4 インテリジェントモジュールの実行

ExecuteInteliIPModule()コマンドにより、パラメータのデータを転送し、登録したインテリジェントモジュールを実行します。

```
ExecuteInteliIPModule(_devID,MODULE_0,&prmtbl,NULL);
```

このコマンドはオンボードＣＰＵのインテリジェントモジュールを起動し、その処理が終了するまでスリープします。そして、処理が終了した時点でウェイクアップしてこのコマンドが終了します。

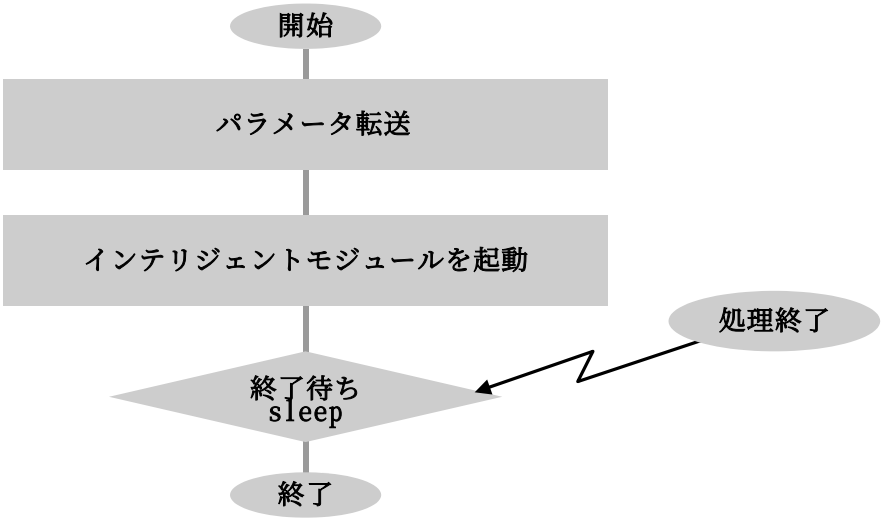


図７－７　コマンド実行フロー

7．4．5 インテリジェントモジュールからのデータ取得

インテリジェントモジュール実行後にデータ取得する方法を説明します。

(1) 指定した配列パラメータが入出力データの場合

インテリジェントモジュールのパラメータが配列でインテリジェントモジュールにデータを渡してなおかつインテリジェントモジュール実行終了後にデータを取得する場合、SetParamTable()コマンドでパラメータテーブルにデータを設定し、インテリジェントモジュール実行終了後、GetParamTable()コマンドでデータを取得して下さい。

```
SetParamTable(_devID,&prmtbl,PARAM_3,&tbl,sizeof(tbl));
ExecuteInteliIPModule(_devID,MODULE_0,&prmtbl,NULL);
GetParamTable(_devID,&prmtbl,PARAM_3,&tbl,sizeof(tbl));
```

(2) 指定した配列パラメータが出力のみのデータの場合

インテリジェントモジュールのパラメータが配列でインテリジェントモジュール実行終了後にデータを取得する場合、AllocParamTable()コマンドでパラメータテーブルの領域を確保し、インテリジェントモジュール実行終了後、GetParamTable()コマンドでデータを取得して下さい。

```
AllocParamTable(_devID,&prmtbl,PARAM_3,sizeof(tbl));
ExecuteInteliIPModule(_devID,MODULE_0,&prmtbl,NULL);
GetParamTable(_devID,&prmtbl,PARAM_3,&tbl,sizeof(tbl));
```

7．5 インテリジェントモジュールコマンド

インテリジェントモジュール関連コマンドの一覧を示します。

7．5．1 インテリジェントモジュール制御コマンド一覧

コマンド名	機能	備考
RegistInteliIPModule	インテリジェントモジュールの登録	
ExecuteInteliIPModule	インテリジェントモジュールの実行	

7．5．2 モジュールサポートコマンド一覧

コマンド名	機能	備考
InitIPParamTable	モジュールのパラメータテーブルの初期化	
SetIntegerParam	整数型パラメータのセット	
SetFloatParam	フロート型パラメータのセット	
SetParamTable	配列データ（入力）パラメータのセット	
AllocParamTable	配列データ（出力）パラメータ領域の確保	
GetParamTable	配列データ（出力）パラメータの読み出し	
SetParamTableExt	配列データ（入力）パラメータのセット	
AllocParamTableExt	配列データ（出力）パラメータ領域の確保	
ReadIPMemDatabyLong	ボード側システムメモリデータリード（４バイト）	
ReadIPMemDatabyWord	ボード側システムメモリデータリード（２バイト）	
ReadIPMemDatabyByte	ボード側システムメモリデータリード（１バイト）	
WriteIPMemDatabyLong	ボード側システムメモリデータライト（４バイト）	
WriteIPMemDatabyWord	ボード側システムメモリデータライト（２バイト）	
WriteIPMemDatabyByte	ボード側システムメモリデータライト（１バイト）	
ExitIPTask	自タスクの終了	
GetIPTaskID	実行状態の自タスクＩＤの参照	
CreateMBX	メールボックスの生成	
DeleteMBX	メールボックスの削除	
SendMBX	メールボックスへの送信	
RecvMBX	メールボックスからの受信	
pRecvMBX	メールボックスからの受信（ポーリング）	
tRecvMBX	メールボックスからの受信（タイムアウト付き）	
ChgIMS	割込マスクの変更	
getmpl	メモリブロックの確保	
relmpl	メモリブロックの確保	

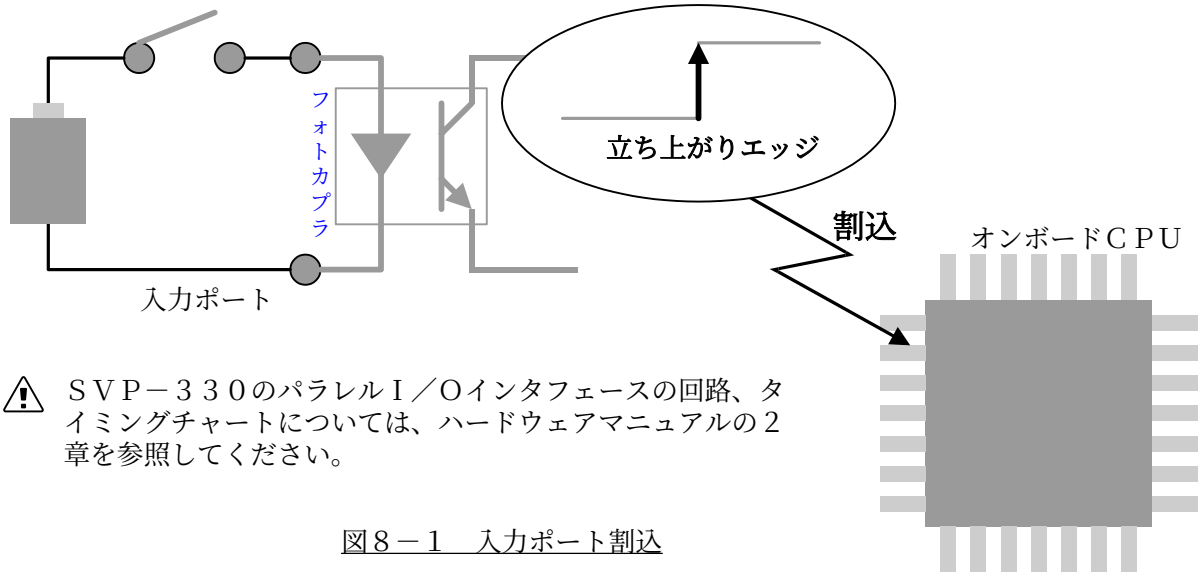
第8章

割込起動モジュール

画像処理コマンドでは画像処理アプリケーションのモジュール化として、
・インテリジェントモジュール
・割込起動モジュール
という２通りの手法を提供しています。
本章では、割込起動モジュールについて説明します。第３章で概略を説明していますので、そちらも参照して下さい。また、モジュール化については第７章で説明していますので、そちらもあわせて参照して下さい。

8．1 PIO割込について

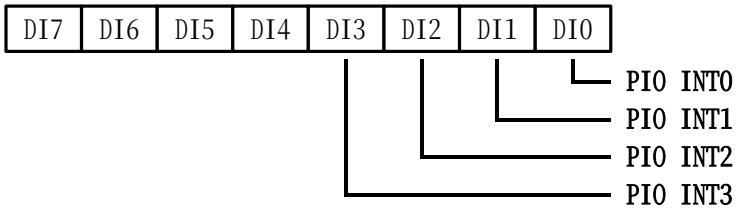
ＳＶＰ－３３０では、パラレルＩ／Ｏを実装しています。フォトカプラによりアイソレートされたパラレルの入力／出力ポート（ＰＩＯ）を入力４ビット、出力４ビット持っています。
入力ポートはビット毎に割込に対応しており、入力ポートのレベルが「Ｌ」→「Ｈ」になるときの立ち上がりエッジによりオンボードＣＰＵに対して割込が発生します。



⚠ ＳＶＰ－３３０のパラレルＩ／Ｏインタフェースの回路、タイミングチャートについては、ハードウェアマニュアルの２章を参照してください。

図８－１　入力ポート割込

画像処理コマンドでは、そのＰＩＯ割込に対する処理をオンボードＣＰＵのユーザオリジナルの画像処理アプリケーションで簡単に実行できるように割込起動モジュールというフレームモデルを実装しています。そのフレームモデルにより、オンボードＣＰＵのモジュールを登録するだけで、任意のＰＩＯ割込でそのモジュールを起動することができます。
ＰＩＯの入力ポートは４ビットあり、ＤＩ０～ＤＩ３ビットが割込起動モジュールに割り当てられます



8．2 割込起動モジュールの概要

割込み起動モジュールとは、ＳＶＰ－３３０のパラレルＩ／Ｏ（ＰＩＯ）等の割込みにより起動することができるモジュールです。画像処理コマンドを複数組み合わせて作成したユーザーオリジナルの画像処理モジュールをＰＩＯの割込み等により起動するモジュールとして簡単に登録することができます。

ＰＩＯからの割込で起動できる割込起動モジュールはμＩＴＲＯＮの１つのタスクとして登録され実行されます。ＳＶＰ－３３０ではＰＩＯ入力ポートのデータビット０～データビット３（ＤＩ０～ＤＩ３）の割込にそれぞれ任意のタスクを対応させたモジュールとして登録することができます。

ＳＶＰ－３３０では、ユーザは全部で３２個のタスクを生成することができますが、そのうち４つを割込起動モジュールとして管理することが可能です。

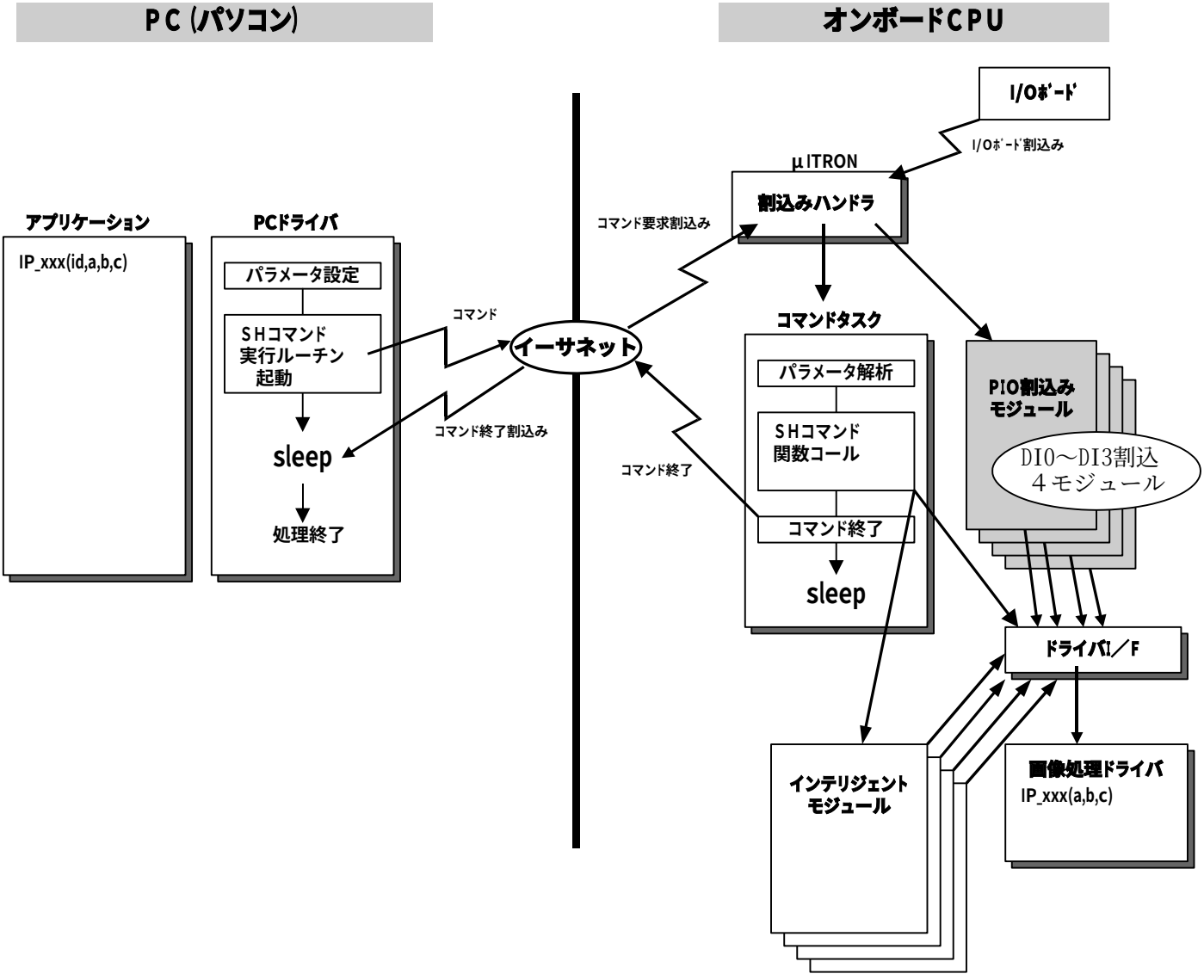


図８－２　割込モジュールの仕組み

8．3 割込起動モジュールの動作

割込起動モジュールはμＩＴＲＯＮの１つのタスクとして登録され実行されます。そのため、画像処理コマンドでは、そのモジュールをμＩＴＲＯＮのタスクの動作に適合するように実装します。

割込起動モジュールは、初期化部モジュールと実行部モジュールのペアのモジュールで１つのタスクを構成します。

初期化部モジュールは、StartIPTaskwithParam()コマンドで起動されTerminateIPTask()コマンドでタスクが終了するまで１度だけ動作します。初期化部は、ユーザオリジナルアプリケーションのグローバル変数などの初期化に使用します。

実行部モジュールは、WakeupIPTaskwithParam()コマンドや対応するＰＩＯの入力割込により起床（ウェイクアップ）します。実際の画像処理アプリケーションはこの実行部に登録されます。

また、初期化部モジュール、実行部モジュールには、インテリジェントモジュールと同様にパラメータを渡すことが可能です。

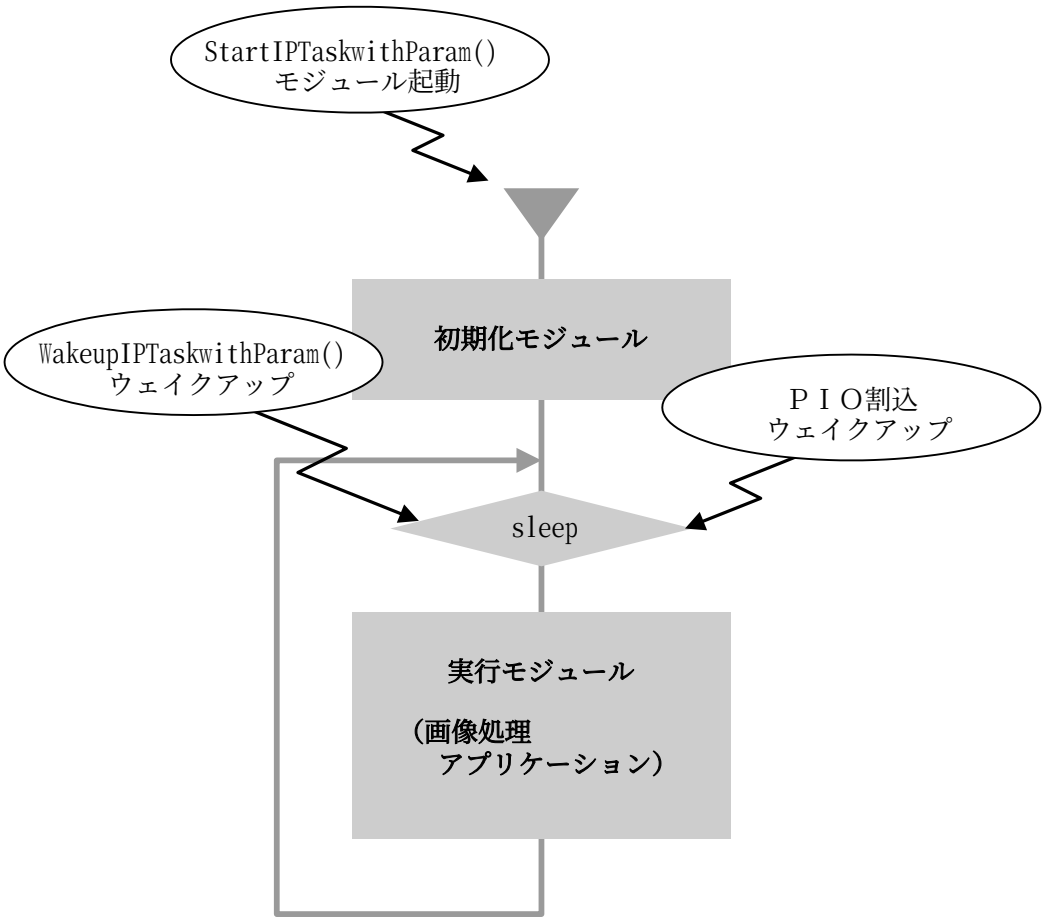


図８－３　割込モジュールの動作

8．4 割込起動モジュールの作成

本項では、オンボードＣＰＵ側のプログラムについて説明します。

8．4．1 プログラム

割込起動モジュールでは、初期化部モジュールと実行部モジュールの２つのモジュールを作成します。
ＰＣ（パソコン）側から起動されるオンボードＣＰＵのプログラムは、Ｃ言語の関数（サブルーチン）形式で記述して下さい。また、疑似命令「#pragma」でセクションを指定して下さい。リンク時に指定したセクション名でプログラムの開始アドレスを直接指定できます。

初期化部モジュール

```
#pragma   section   task0init
void piоint0_task_init(int a,float b)
{
   ..........
   ..........
}
```

実行部モジュール

```
#pragma   section   task0main
void piоint0_task(int a,float b,short *tbl)
{
   ..........
   ..........
   SendSignaltoPC(..........);
}
```

関数名は任意でパラメータはint型、float型、ポインタ型で０から最大３２個まで指定できます。これらのパラメータは、ＰＣ側のプログラムで「モジュールサポートコマンド」によりデータが設定されます。
それ以外は、ＰＣでのプログラムと同一にして下さい。

8．4．2 ＰＣとの同期

画像処理コマンドでは、割込起動モジュールに対して、ＰＣ（パソコン）側の処理とオンボードＣＰＵとの同期をとる方法としてWaitforIPTaskSignal()とSendSignaltoPC()コマンドを用意しています。
まず、ＰＣ側はWaitforIPTaskSignal()コマンドでオンボードＣＰＵの処理が終了するのを待ちます。そして、オンボードＣＰＵ側からSendSignaltoPC()コマンドを実行すると、ＰＣ側に終了シグナルが送信され、ＰＣ側はWaitforIPTaskSignal()コマンドのウェイト処理から抜けます。

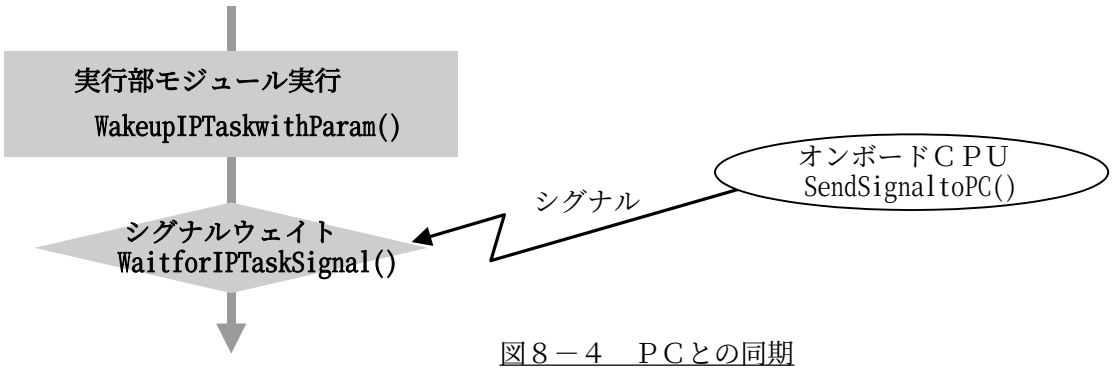


図８－４　ＰＣとの同期

なお、SendSignaltoPC()コマンドは、モジュールの最後に実行して下さい。モジュールの中でSendSignaltoPC()コマンド実行後に画像処理コマンドを実行すると正常に動作しないことがあります。

8．4．3　インクルードファイル

ＰＣコマンドの場合と同じように、

vpxdef.h,vpxsys.h,vpxfnc.h

をインクルードします。それ以外に

vpxcnv.h

をインクルードしてください。vpxcnv.hはＰＣ側にもダミーで用意してあります。ＰＣ側でプログラムのデバッグを行う場合、そのままvpxcnv.hをインクルードして下さい。

8．4．4　メモリマップ

オンボードＣＰＵシステムメモリでユーザーが使用できるメモリ空間は

0x8C80:0000 ～ 0x8CFF:FFFF

の８Ｍバイトです。この領域にプログラム、スタティックデータなどをマッピングするように、リンク時に指定して下さい。また、この領域内にmalloc()等のヒープ領域やＲＡＭディスク領域も含んでいますので、「プログラム領域＋ヒープ等の領域」が８Ｍバイトを超えないようにマッピングして下さい。

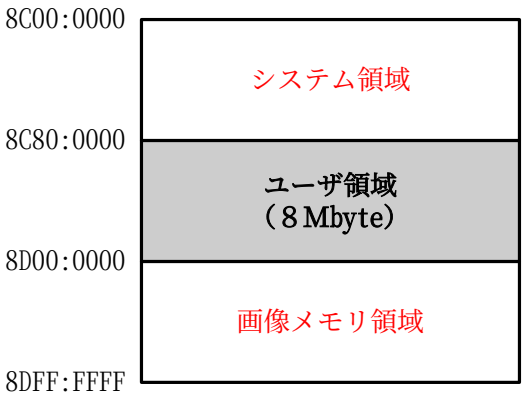


図８－５　ＲＡＭメモリマップ

8．4．5　プログラムローダー

ユーザー領域(0x8C80:0000 ～ 0x8CFF:FFFF)は、システムにより管理されています。
プログラムのローダーは、実行ファイル(.abs)からプログラムがロードされる領域を計算し、その領域をユーザ領域から確保し、その領域に実行コードをダウンロードします。実行コードがダウンロードされる領域がすでに確保されている場合、ローダーは実行ファイルの実行コードをダウンロードできません。その場合は、すでに実行しているアプリケーションを終了しその領域を開放するか、アドレスのマッピングを変更してください。また、この領域内にmalloc()等のヒープ領域やＲＡＭディスク領域も含んでいます。特に、ＲＡＭディスクを確保した場合は、0x8C80:0000番地からマッピングできない場合がありますので、注意してください。

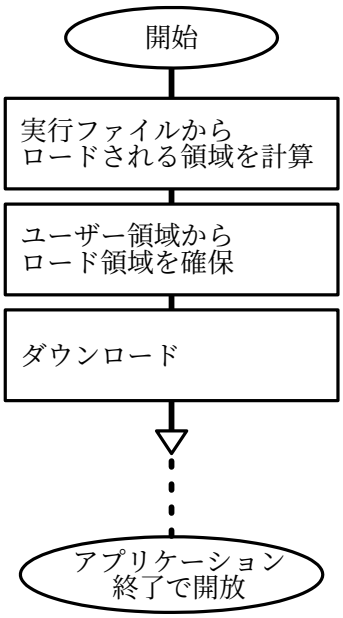


図８－６　ロードフロー

8.4.6 スタック

プログラムで使用できるスタック領域は、CreateIPTaskコマンドでタスクを生成した時の設定値になります。CreateIPTaskでタスクを生成する時に必要な容量を確保して下さい。

8.4.7 プロジェクトのビルド

ＳＨ４のコンパイラでコンパイル、リンクを行いＥＬＦ形式の実行ファイル(.abs)ファイルをビルドします。
なお、ＳＨ４コンパイラのプロジェクトの設定等の詳細は、「ＳＨＣ Ｃｏｍｐｉｌｅｒ設定ガイド」を参照して下さい。

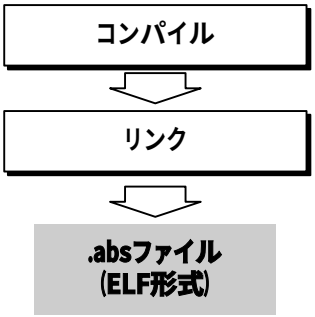


図８－７　プロジェクトのビルド

8.4.8 デバッグ

オンボードＣＰＵの割込起動モジュールの開発環境や開発手法については第２章で説明していますので、まず、そちらを参照してください。オンボードＣＰＵの割込起動モジュールのデバッグは、ＪＴＡＧ－ＩＣＥを使用して行う方法もありますが、ＰＣ側である程度動作しているモジュールであれば、その必要性は低いと思われます。しかし、データの確認などの簡単なデバッグ作業は、プログラム開発上できるにこしたことはありません。そこで画像処理コマンドでは、オンボードＣＰＵ側にＲＳ２３２Ｃのターミナルを対象にしたprintf()やscanf()などの標準入出力を実装しています。ＳＶＰ－３３０で使用できる標準入出力コマンドは、以下に示します。

表８－１　オンボードＣＰＵ標準入出力関数

関数名	内容	
printf	Ｃの標準関数のprintfと同じです。	
scanf	Ｃの標準関数のscanfと同じです。	
charget	Ｃの標準関数のgetcharと同じです。	
charput	Ｃの標準関数のputcharと同じです。	
malloc	Ｃの標準関数のmallocと同じです。	
free	Ｃの標準関数のfreeと同じです。	

⚠ これらの標準入出力関数を使用する場合は、インクルードファイル「vpxcnv.h,stdio.h,stdlib.h」をＣのソースファイルでインクルードして下さい。

シリアルポートの初期設定を以下に示します。これ以外の設定が必要な場合、InitSCI()コマンドで設定して下さい。

表８－２　シリアルポートの初期設定

項目	初期値	備考
ビットレート	９６００ｂｐｓ	
キャラクタ長	８ｂｉｔ	
パリティ	パリティなし	
ストップビット	１ｂｉｔ	

8．5 割込起動モジュールの制御

本項では、オンボードＣＰＵ側の割込起動モジュールを制御するためのＰＣ（パソコン）側のプログラムについて説明します。
以下に制御フローを示します。

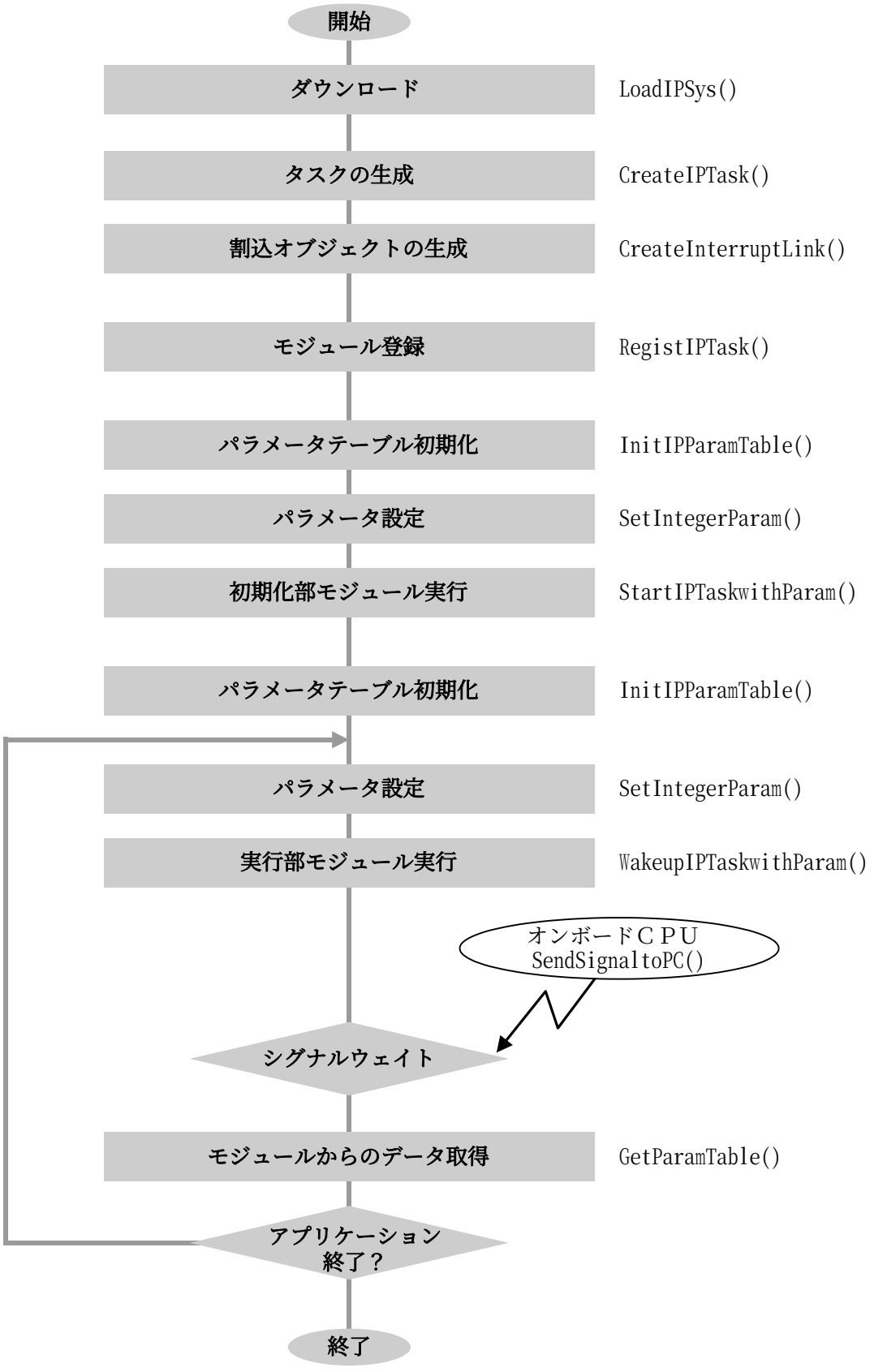


図８－８　割込起動モジュール制御フロー

8.5.1 割込起動モジュールのダウンロード

８．４項でビルドした「abs」ファイルをLoadIPSys()コマンドを使用してＳＶＰ－３３０にダウンロードします。

```
LoadIPSys(_devID,"piotask.abs",DIRECT_PATH);
```

また、同じアドレス空間に再度ダウンロードする場合は「UNMAP_APLBITS」オプションを指定して下さい。システムはその空間のマッピングを開放し、再度領域を確保しダウンロードします。ただし、対象のアプリケーション（タスク）が完全に終了してから行って下さい。

```
LoadIPSys(_devID,"piotask.abs",UNMAP_APLBITS | DIRECT_PATH);
```

8.5.2 タスクの生成

ダウンロードしたＰＩＯ割込起動モジュールを動作させるためのタスクを生成します。

```
int tskid;
CREATE_TASK_TBL   iptsk;

iptsk.task_addr  = 0;
iptsk.priority   = 0;
iptsk.pbuff_size = 0x2000;
iptsk.stack_size = 0x1000;
iptsk.task_opt   = 0;
iptsk.param_opt  = 1;

tskid= CreateIPTask(_devID,&iptsk);
```

8.5.3 割込オブジェクトの生成

生成したタスクを割込元のデバイスに登録します。

```
INT_DEVICE_OBJ   intobj;

intobj.intdev = INTDEV_PIO;
intobj.evtpri = 0;
intobj.intbit = 1;
intobj.opt    = 0;

CreateInterruptLink(_devID,&intobj,tskid,INTEVENT_WAKEUP,0);
```

8.5.4 割込起動モジュールの登録

ダウンロードした割込起動モジュールをRegistIPTask()コマンドで登録します。

```
RegistIPTask(_devID,tskid,0x8C200000,0x8C201000,TASK_NO_OPTION);
```

8.5.5 割込起動モジュールへのパラメータ渡し

登録した割込起動モジュールにパラメータを渡す方法を説明します。初期化モジュール、実行モジュールの両方が同じ方法でパラメータの設定ができます。

(1) パラメータテーブルの初期化

InitIPParamTable()コマンドでパラメータテーブルを初期化します。

```
MODULE_PARAM_TBL   prmtbl;

InitIPParamTable(_devID,&prmtbl,3,PIOINT_MODULE,tskid,0);
```

(2) パラメータの設定

割込起動モジュールのパラメータがint型やfloat型の場合、SetIntegerParam(),SetFloatParam()コマンドでパラメータテーブルにデータを設定します。

```
SetIntegerParam(_devID,&prmtbl,PARAM_1,123);
SetFloatParam(_devID,&prmtbl,PARAM_2,4.567);
```

(3) テーブル（配列）パラメータの設定

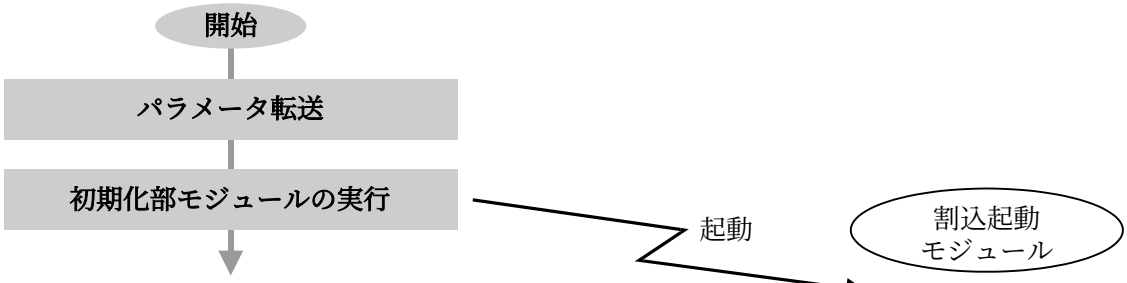
割込起動モジュールのパラメータが配列の場合、SetParamTable()コマンドでパラメータテーブルにデータを設定します。

```
SetParamTable(_devID,&prmtbl,PARAM_3,&tbl,sizeof(tbl));
```

8.5.6 初期化部モジュールの実行

RegistIPTask()コマンドにより、パラメータのデータを転送し、登録した割込起動モジュールの初期化部モジュールを実行します。

StartIPTaskwithParam(_devID,tskid,&prmtbl);

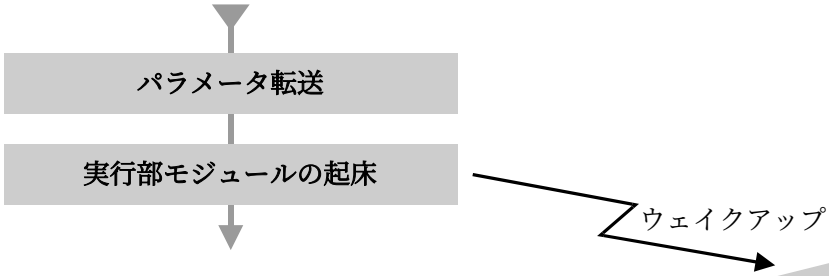


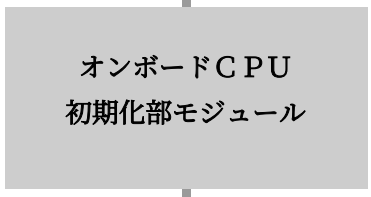
8.5.7 実行部モジュールのウェイクアップ

WakeupIPTaskwithParam()コマンドにより、パラメータのデータを転送し、登録した割込起動モジュールの実行部モジュールをウェイクアップします。

このコマンドはオンボードＣＰＵの割込起動モジュールの実行部をウェイクアップするだけです。処理終了ウェイトは行いません。

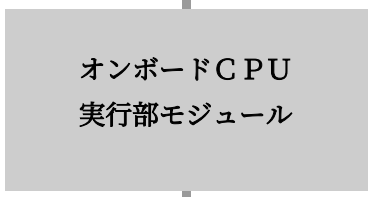
WakeupIPTaskwithParam(_devID,tskid,&prmtbl,SELF_WAKEUP);



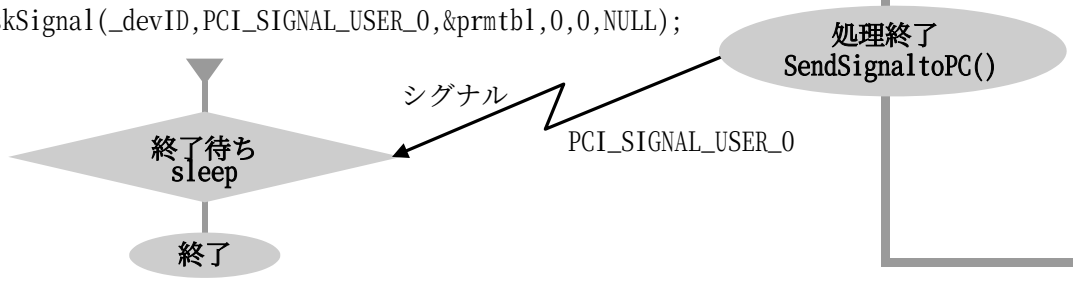


8.5.8 モジュールの終了ウェイト

WaitforIPTaskSignal()コマンドにより、ウェイクアップされたオンボードＣＰＵの割込起動モジュールからのSendSignaltoPC()コマンドによる終了シグナルをウェイトします。そして、オンボードＣＰＵでSendSignaltoPC()コマンドを実行したした時点でウェイクアップしてこのコマンドが終了します。



WaitforIPTaskSignal(_devID,PCI_SIGNAL_USER_0,&prmtbl,0,0,NULL);



8.5.9 割込起動モジュールからのデータ取得

割込起動モジュール実行後にデータ取得する方法を説明します。

(1) 指定した配列パラメータが入出力データの場合

割込起動モジュールのパラメータが配列で割込起動モジュールにデータを渡してなおかつ割込起動モジュール実行終了後にデータを取得する場合、SetParamTable()コマンドでパラメータテーブルにデータを設定し、割込起動モジュール実行終了後、GetParamTable()コマンドでデータを取得して下さい。

```
SetParamTable(_devID,&prmtbl,PARAM_3,&tbl,sizeof(tbl));
WakeupIPTaskwithParam(_devID,tskid,&prmtbl,PIO_WAKEUP);
WaitforIPTaskSignal(_devID,PCI_SIGNAL_USER_0,&prmtbl,0,0,NULL);
GetParamTable(_devID,&prmtbl,PARAM_3,&tbl,sizeof(tbl));
```

(2) 指定した配列パラメータが出力のみのデータの場合

割込起動モジュールのパラメータが配列で割込起動モジュールにデータを渡してなおかつ割込起動モジュール実行終了後にデータを取得する場合、AllocParamTable()コマンドでパラメータテーブルの領域を確保し、割込起動モジュール実行終了後、GetParamTable()コマンドでデータを取得して下さい。

```
AllocParamTable(_devID,&prmtbl,PARAM_3,sizeof(tbl));
WakeupIPTaskwithParam(_devID,tskid,&prmtbl,PIO_WAKEUP);
WaitforIPTaskSignal(_devID,PCI_SIGNAL_USER_0,&prmtbl,0,0,NULL);
GetParamTable(_devID,&prmtbl,PARAM_3,&tbl,sizeof(tbl));
```

8．6 割込起動モジュールコマンド

割込起動モジュール関連コマンドの一覧を示します。

8．6．1 割込起動モジュール制御コマンド一覧

コマンド名	機能	備考
CreateIPTask	ユーザタスクの生成	
CreateInterruptLink	割り込みリンクオブジェクトの生成	
DeleteInterruptLink	割り込みリンクオブジェクトの削除	
EnableInterruptObject	割り込み動作の有効化	
DisableInterruptObject	割り込みオブジェクトの無効化	
RegistIPTask	割込モジュールの登録	
StartIPTaskwithParam	割込モジュールのサービス開始	
WakeupIPTaskwithParam	割込モジュールの実行	
WaitforIPTaskSignal	割込モジュールからの終了シグナルウェイト	
ReadyforWaitIPTaskSignal	割込モジュールからの終了シグナルウェイト準備	
WakeupIPTaskExt	割り込みモジュールの実行２	
CancelIPTaskWait	終了シグナルウェイトの解除	
StartIPTask	割込モジュールのサービス開始	
WakeupIPTask	割込モジュールの実行	
TerminateIPTask	割込モジュールのサービス終了	
ResumeIPTask	割込モジュールサービスの再開	
SuspendIPTask	割込モジュールサービスの一時停止	
DeleteIPTask	割込モジュールの削除	

8．6．2 割込起動モジュール!／Fコマンド(ボードCPU側)一覧

コマンド名	機能	備考
SendSignaltoPC	割込モジュールからのシグナル送信	

8．6．3 モジュールサポートコマンド一覧

コマンド名	機能	備考
InitIPParamTable	モジュールのパラメータテーブルの初期化	
SetIntegerParam	整数型パラメータのセット	
SetFloatParam	フロート型パラメータのセット	
SetParamTable	配列データ（入力）パラメータのセット	
AllocParamTable	配列データ（出力）パラメータ領域の確保	
GetParamTable	配列データ（出力）パラメータの読み出し	
SetParamTableExt	配列データ（入力）パラメータのセット	
AllocParamTableExt	配列データ（出力）パラメータ領域の確保	
ReadIPMemDatabyLong	ボード側システムメモリデータリード（４バイト）	
ReadIPMemDatabyWord	ボード側システムメモリデータリード（２バイト）	
ReadIPMemDatabyByte	ボード側システムメモリデータリード（１バイト）	
WriteIPMemDatabyLong	ボード側システムメモリデータライト（４バイト）	
WriteIPMemDatabyWord	ボード側システムメモリデータライト（２バイト）	
WriteIPMemDatabyByte	ボード側システムメモリデータライト（１バイト）	
ExitIPTask	自タスクの終了	
GetIPTaskID	実行状態の自タスクＩＤの参照	
CreateMBX	メールボックスの生成	
DeleteMBX	メールボックスの削除	
SendMBX	メールボックスへの送信	
RecvMBX	メールボックスからの受信	
pRecvMBX	メールボックスからの受信（ポーリング）	
tRecvMBX	メールボックスからの受信（タイムアウト付き）	
ChgIMS	割込マスクの変更	
getmpl	メモリブロックの確保	
relmpl	メモリブロックの確保	

第9章

ビデオレート処理

ビデオレートの処理では、映像入力を行っている間に画像処理を並列に行うということが求められます。画像処理コマンドでは、そのアプローチの１つとして
　・プリフェッチ処理
という手法を提供しています。

また、ビデオレートの処理では、映像の入力タイミングである３３ｍＳ以内に全ての処理を行う必要があります。そのため、画像処理コマンドでは、画像処理アプリケーション全体の高速化として
　・パイプライン処理
という個々の処理を並列実行するという手法も提供しています。
　本章では、それらの処理について説明します。
　実際のアプリケーションでは、７章、８章で述べているオンボードＣＰＵでの処理と併せて実装する場合が多いと思います。

9.1 プリフェッチ処理の概要

　通常の画像処理は、カメラからの映像入力を行い、映像入力が終了してからその入力された画像に対して画像処理を行います。そのため、画像処理時間は、

（映像入力：３３ｍＳ）＋（画像処理時間）

になります。

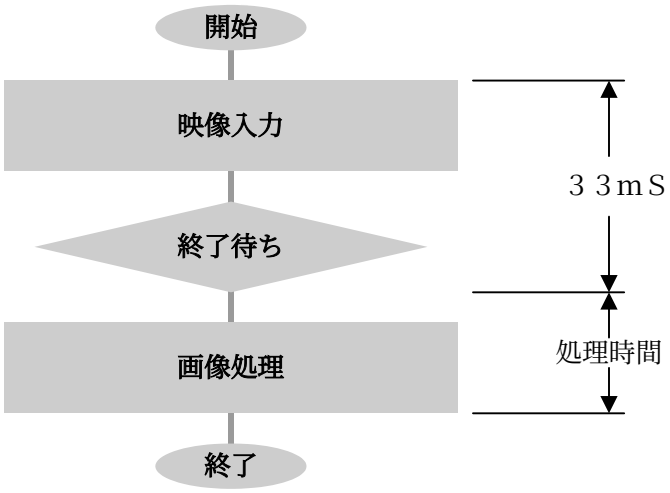


図９－１　画像処理時間

　ビデオレートでの処理（動画処理）は、映像入力の３３ｍＳに同期して処理を行う必要があり、上記のような処理では不可能です。つまり、ビデオレートで処理を行うには、映像入力している間に、画像処理を行わなければならないということです。
　画像処理コマンドでは、そのような要求に対しプリフェッチ処理という手法を提供しています。

9.1.1 プリフェッチ処理

画像処理プロセッサは、ハード的にビデオ入力系のプロセッサと画像処理系のプロセッサが独立しています。そのため、映像入力と画像処理をそれぞれ別のフレームの画像メモリで行うことにより、お互いの処理に干渉することなく、完全に並列に処理を行うことができます。

そこで、下図で画像メモリのフレーム＃０とフレーム＃１を画像入力の度に交互に切り換えて映像入力と画像処理が並列に処理するようにします。そうすることで、画像処理の結果は１フレーム遅れることになりますがビデオレートで画像処理できることになります。

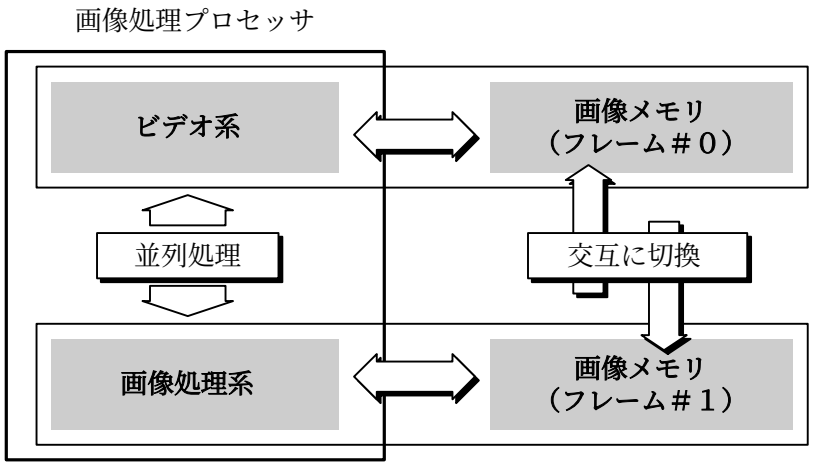


図９－２　並列処理

9.1.2 プリフェッチ処理フロー

下図にプリフェッチ処理のフローを示します。画像メモリフレーム＃０と＃１を交互に切換えて、映像入力と画像処理を並列に行います。

画像メモリフレーム＃０の画像処理を行う場合は、画像メモリフレーム＃０への映像入力を待ち、入力が終了したら、すぐ、画像メモリフレーム＃１への映像入力を起動し、すでに映像入力が完了している画像メモリフレーム＃０の画像処理を実行します。そして、画像メモリフレーム＃０の画像処理が終了したら、さきほど映像入力を起動した画像メモリフレーム＃１に対する映像入力の終了を待ちます。入力が終了したら、すぐ、画像メモリフレーム＃０への映像入力を起動し、すでに映像入力が完了している画像メモリフレーム＃１の画像処理を実行します・・・・・。そのような処理を延々と繰り返すわけです。

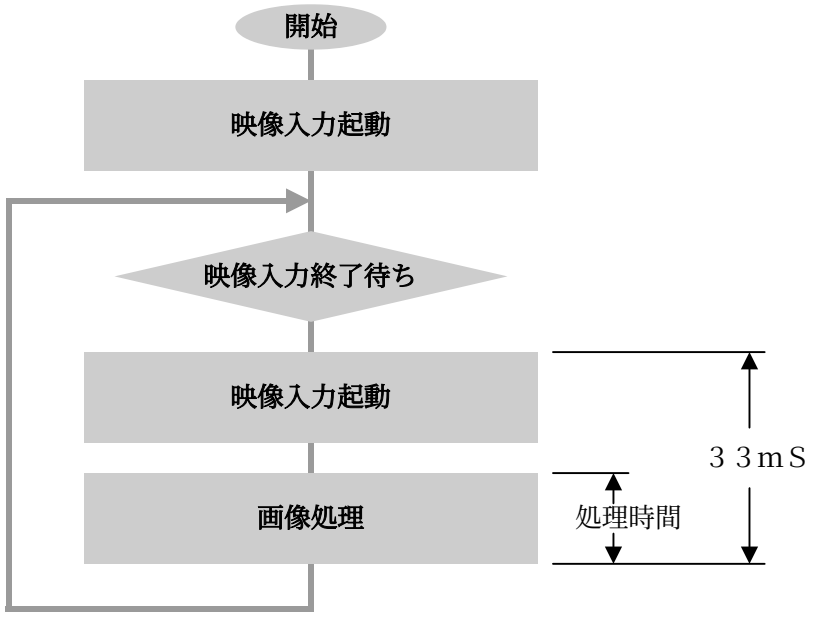


図９－３　プリフェッチ処理フロー

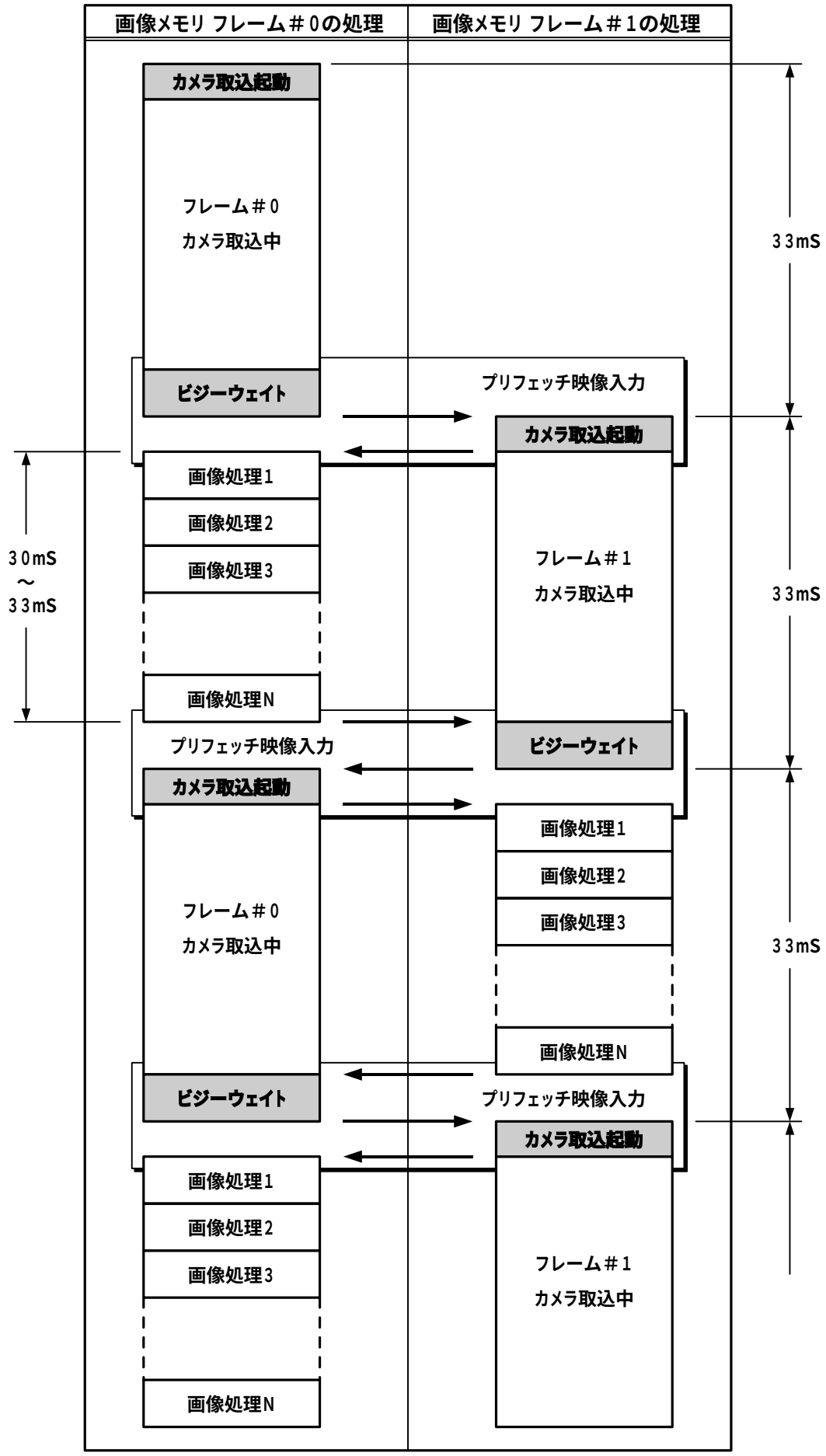


図９－４　プリフェッチ処理フロー詳細

9.1.3 プリフェッチ処理コマンド

画像処理コマンドでは、表９－１に示すようなプリフェッチ処理を簡単化するためのコマンドを用意していますが、GetCamara()コマンドやGet2Camera()コマンドでも同じ処理が可能です。GetCamara()コマンドやGet2Camera()コマンドは映像入力起動後、映像入力の終了を待たずに処理を終了します。プリフェッチ処理コマンドでは、映像入力を行う画像メモリと画像処理を行う画像メモリを画像処理コマンド側で管理しているだけです。

9.1.4 プリフェッチ処理コマンド一覧

表９－１にプリフェッチ処理コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表９－１　プリフェッチ処理コマンド一覧

コマンド名	機能	備考
vpxEnableCameraPrefetch	プリフェッチ処理の有効化	
vpxGetCameraPrefetch	プリフェッチ映像入力	
vpxDisableCameraPrefetch	プリフェッチ処理の終了	

9．2 パイプライン処理の概要

画像処理プロセッサは、画像処理、２値化、ヒストグラムのプロセッサを独立して持っています。そして、その３つの処理を画像処理→２値化→ヒストグラムの順番でパイプライン処理を行うことが可能です。画像処理、２値化、ヒストグラムプロセッサの処理を１画面の処理時間で実行できるわけです。

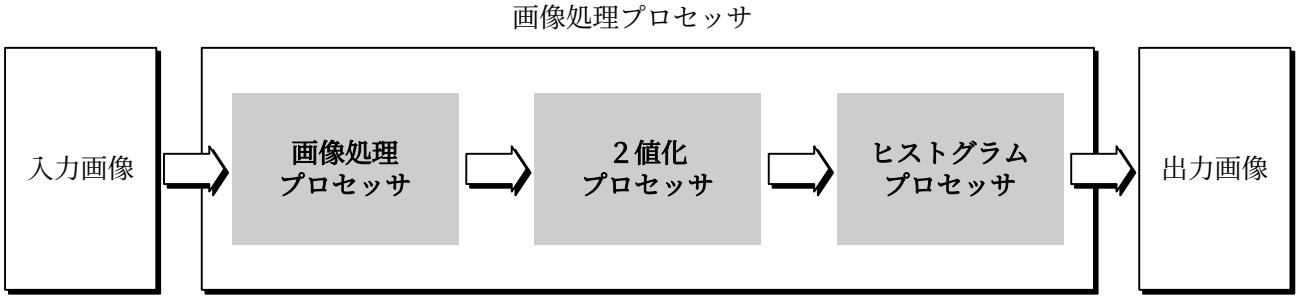


図９－５　画像処理プロセッサパイプライン処理部

また、下記の組み合わせでパイプライン処理で実行可能です。

(1) 画像処理 ＋ ２値化
(2) 画像処理 ＋ ヒストグラム処理
(3) 画像処理 ＋ ２値化 ＋ ヒストグラム処理

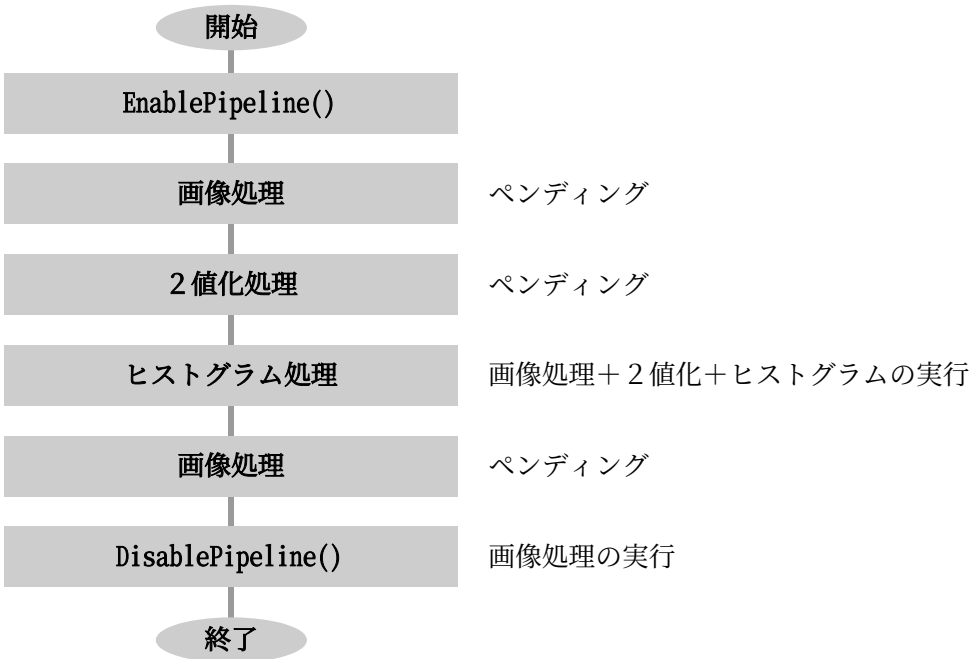
9．2．1 パイプライン処理の実行方法

画像処理コマンドでは、パイプラインモード制御コマンドとしてEnablePipeline(),DisablePipeline()コマンドを用意しています。

EnablePipeline()コマンドを実行すると、パイプラインモードになります。

パイプラインモードでは、最初に発行された画像処理コマンドを実行せず、次のコマンド処理へ移行します。そのコマンドの処理がパイプラインで実行できない場合は、パイプライン制御により実行を待たされていた（ペンディング状態）画像処理コマンド実行します。つまり、コマンドの処理がパイプラインでつながる場合、その時点での処理を行わずペンディング状態にして、コマンドバッファにバッファリングを行い、パイプラインのチェインが切れた時点で処理を行うわけです。

DisablePipeline()コマンドを実行すると、コマンドバッファにバッファリングされていたペンディング処理を実行し、パイプラインモードを解除し、通常の処理に戻ります。



9．2．2　パイプライン処理の実行条件

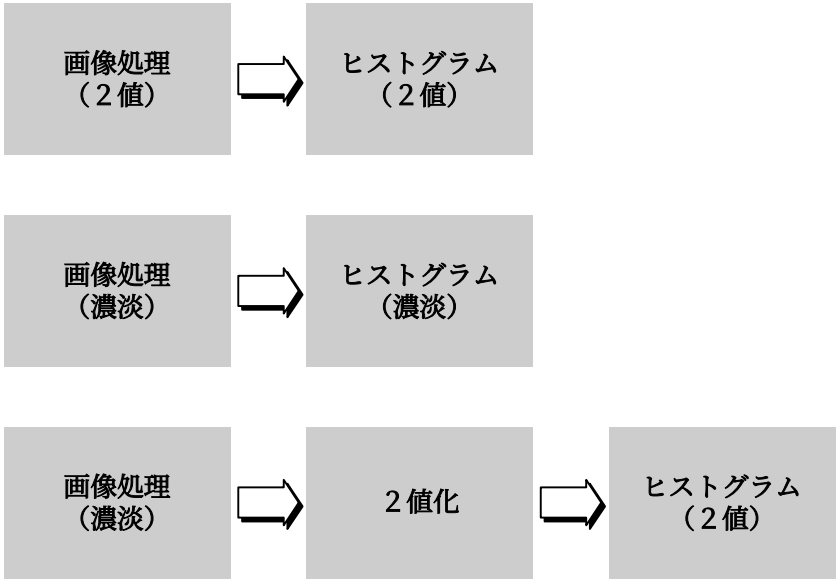
パイプライン処理が行われるには、以下の条件があります。

(1) 画像処理の対象画面

先に実行されるコマンドのデスティネーション画面と、次に実行されるコマンドのソース画面が同一であること。

(2) パイプラインが構成可能な処理と順番

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。



上記の画像処理、ヒストグラム処理に対応するコマンドの項目を以下に示します。

処理項目	コマンド項目
画像処理 （濃淡）	画像転送，アフィン変換，画素変換，画像間算術演算，画像間論理演算，コンボリューション，ミニ／マックスフィルタ，ランクフィルタ
画像処理 （２値）	２値画像形状変換
ヒストグラム （濃淡）	濃淡特徴量抽出 ラベリング特徴量抽出（IP_ExtractL0xxxx()コマンド）
ヒストグラム （２値）	２値画像特徴量抽出

9．2．3　パイプライン制御コマンド一覧

表９－３にパイプライン処理コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表９－３　パイプライン制御コマンド一覧

コマンド名	機能	備考
EnablePipeline	パイプラインモードの有効化	
DisablePipeline	パイプラインモードの無効化	

9．2．4 パイプライン処理の処理例

以下にパイプライン可能／不可能の例を３例示します。

画像処理＋２値化
パイプライン動作可能な場合

```
EnablePipeline(_devID);                          // パイプラインモード起動
IP_Add(_devID,ImgSrc0,ImgSrc1,ImgID);            // IP_Addコマンドはペンディング
IP_Binarize(_devID,ImgID,ImgDst,thr);            // IP_AddとIP_Binarizeをパイプライン実行
DisablePipeline(_devID);                         // パイプラインモード終了
```

IP_Add()のデスティネーション画面とIP_Binarize()のソース画面が同じ場合なので、パイプライン処理を行います。

画像処理＋２値化
パイプライン動作不可能な場合

```
EnablePipeline(_devID);                          // パイプラインモード起動
IP_Add(_devID,ImgSrc0,ImgSrc1,ImgID);            // IP_Addコマンドはペンディング
IP_Binarize(_devID,ImgSrc0,ImgDst,thr);          // パイプライン実行できずIP_Addと
                                                 // IP_Binarizeをシリーズに実行
DisablePipeline(_devID);                         // パイプラインモード終了
```

IP_Add()のデスティネーション画面とIP_Binarize()のソース画面が異なるので、パイプライン処理できません。

画像処理＋ヒストグラム
パイプライン動作可能な場合

```
EnablePipeline(_devID);                          // パイプラインモード起動
IP_Add(_devID,ImgSrc0,ImgSrc1,ImgID);            // IP_Addコマンドはペンディング
IP_Histogram(_devID,ImgID,&tbl,&regs,opt);       // IP_AddとIP_Histogramをパイプライン実行
DisablePipeline(_devID);                         // パイプラインモード終了
```

IP_Add()のデスティネーション画面とIP_Binarize()のソース画面が同じ場合なので、パイプライン処理を行います。

画像処理＋ヒストグラム
パイプライン動作不可能な場合

```
EnablePipeline(_devID);                          // パイプラインモード起動
IP_Add(_devID,ImgSrc0,ImgSrc1,ImgID);            // IP_Addコマンドはペンディング
IP_Histogram(_devID,ImgSrc0,&tbl,&regs,opt);     // パイプライン実行できずIP_Addと
                                                 // IP_Histogramをシリーズに実行
DisablePipeline(_devID);                         // パイプラインモード終了
```

IP_Add()のデスティネーション画面とIP_Binarize()のソース画面が異なるので、パイプライン処理できません。

画像処理＋２値化＋ヒストグラム
パイプライン動作可能な場合

```
EnablePipeline(_devID);                                  // パイプラインモード起動
IP_Add(_devID,ImgSrc0,ImgSrc1,ImgID);                    // IP_Addコマンドはペンディング
IP_Binarize(_devID,ImgID,ImgID2,thr);                    // IP_Binarizeコマンドはペンディング
IP_Histogram(_devID,ImgID2,&tbl,&regs,opt);              // IP_AddとIP_BinarizeとIP_Histogramを
DisablePipeline(_devID);                                 // パイプライン実行
                                                         // パイプラインモード終了
```

IP_Add()のデスティネーション画面とIP_Binarize()のソース画面、IP_Binarize()のデスティネーション画面とIP_Histogram()のソース画面が同じ場合なので、パイプライン処理を行います。

画像処理＋２値化＋ヒストグラム
パイプライン動作不可能な場合

```
EnablePipeline(_devID);                                  // パイプラインモード起動
IP_Add(_devID,ImgSrc0,ImgSrc1,ImgID);                    // IP_Addコマンドはペンディング
IP_Binarize(_devID,ImgID,ImgID2,thr);                    // IP_Binarizeコマンドはペンディング
IP_Histogram(_devID,ImgID,&tbl,&regs,opt);               // IP_BinarizeとIP_Histogramがパイプライン
                                                         // 実行できないため、IP_AddとIP_Binarizeを
                                                         // パイプライン実行後、IP_Histogramを実行
DisablePipeline(_devID);                                 // パイプラインモード終了
```

IP_Binarize()のデスティネーション画面とIP_Histogram()のソース画面が異なるので、パイプライン処理はできません。

9.2.5 パイプライン処理による不定画面

パイプライン処理で、デスティネーション画面にデータが書き込まれなかった画面を不定画面といいます。不定画面は、パイプライン処理で結果が格納されなかったデータの不定な画面であるため、次の処理で画像処理のソース画面として使用すると期待する結果が得られません。不定画面を画像処理ソース画面として使用するとエラーになります。

```
EnablePipeline(_devID);

IP_Add(_devID,ImgSrc0,ImgSrc1,ImgID);

IP_Binarize(_devID,ImgID,ImgDst,thr);

DisablePipeline(_devID);
```

ImgSrc0とImgSrc1をＡＤＤ後、２値化し、結果をImgDstに格納

ImgIDには、処理結果が格納されない
ImgIDは不定画面になる

第10章

非同期映像入力

第９章で説明しているプリフェッチ処理では、カメラからの映像入力は、カメラの同期信号に同期したタイミングで映像を入力することを想定しています。しかし、マシンビジョンに於いては、カメラの同期信号に同期したタイミングで映像を入力するということは、ほとんどなく、マシンからのトリガによりカメラから映像を入力するというということが前提になるでしょう。

本章では、マシンからの非同期のトリガによるプリフェッチ処理について説明します。本章を読む前に、第８章の割込起動モジュール、第９章のビデオレート処理を理解しておいて下さい。

１０．１　非同期映像入力の概要

第９章で説明しているプリフェッチ処理では、カメラからの映像入力待ちの時間に別の画像メモリのチャネルで画像処理を行い、映像入力時間の３３ｍＳという時間を実質「０」にするという処理ですが、カメラの同期信号に同期して処理を行うことを前提にしたものです。しかし、マシンビジョンに於いては、マシンからのトリガ信号でカメラから映像入力をすることの方が普通です。

マシンからのトリガ信号でカメラから映像入力することとプリフェッチ処理で映像入力時間の３３ｍＳという時間を実質「０」にすることができないのか？。そうすればマシンビジョンのパフォーマンスは最大になるはずです。

そこで、画像処理コマンドでは割込起動モジュールとプリフェッチ処理を組み合わせて上記のような非同期映像入力処理の要求に対応できる仕組みを提供しています。

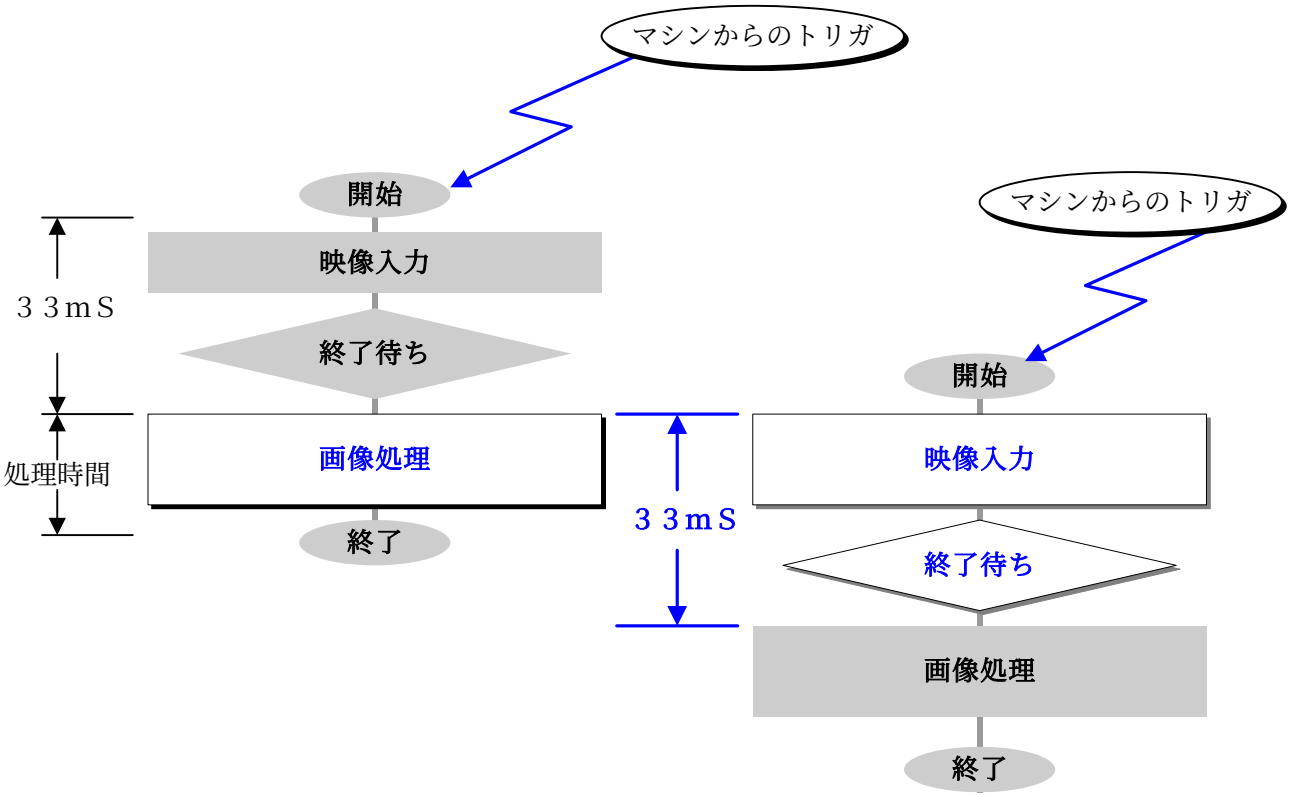


図１０－１　画像処理時間

10．1．1　映像入力タスクからの画像処理タスクの起動

映像入力タスク（タスク１）と画像処理タスク（タスク２）を別々のタスクにして、映像入力タスクから、画像処理タスクを起動します。

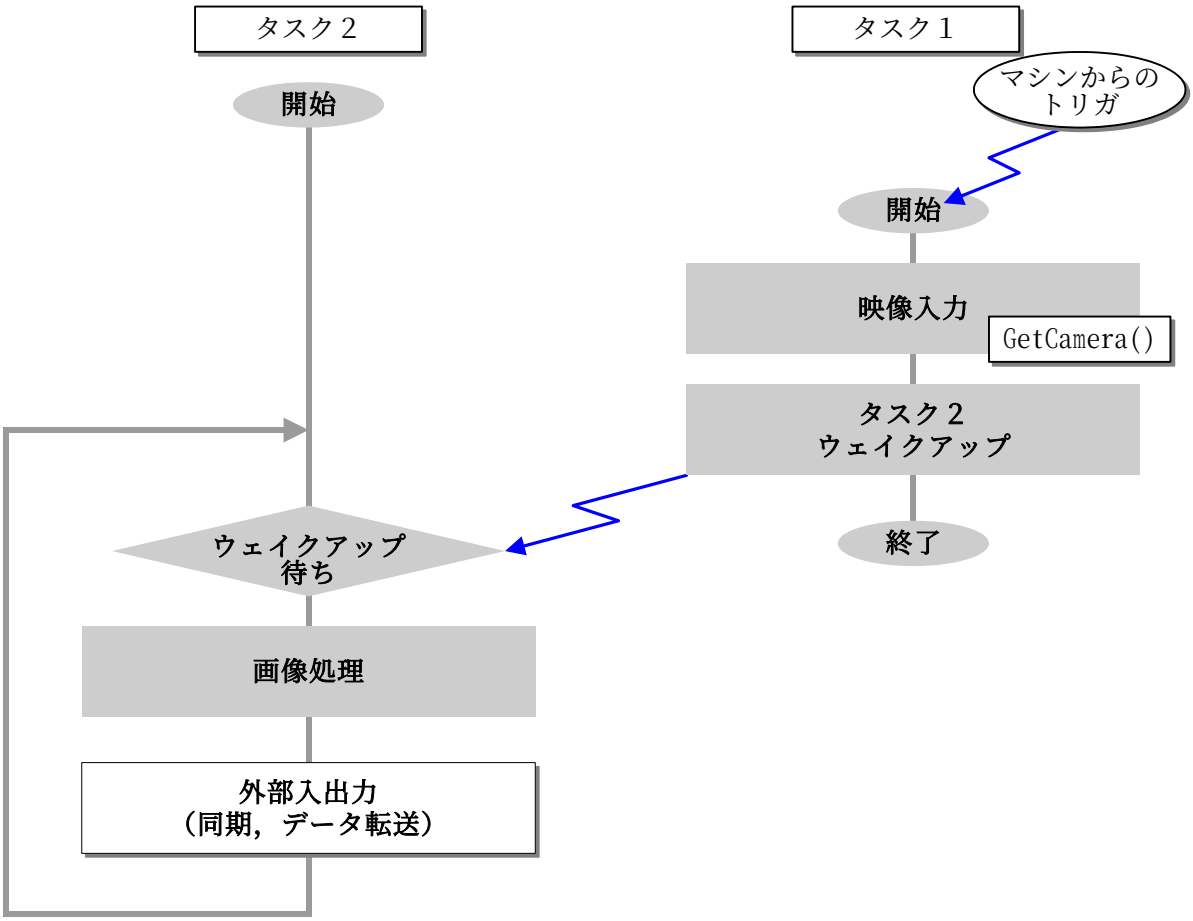


図１０－２　映像入力タスク起動のフロー

映像入力タスクからの画像処理タスクの起動による非同期映像入力のタイムチャートを以下に示します。

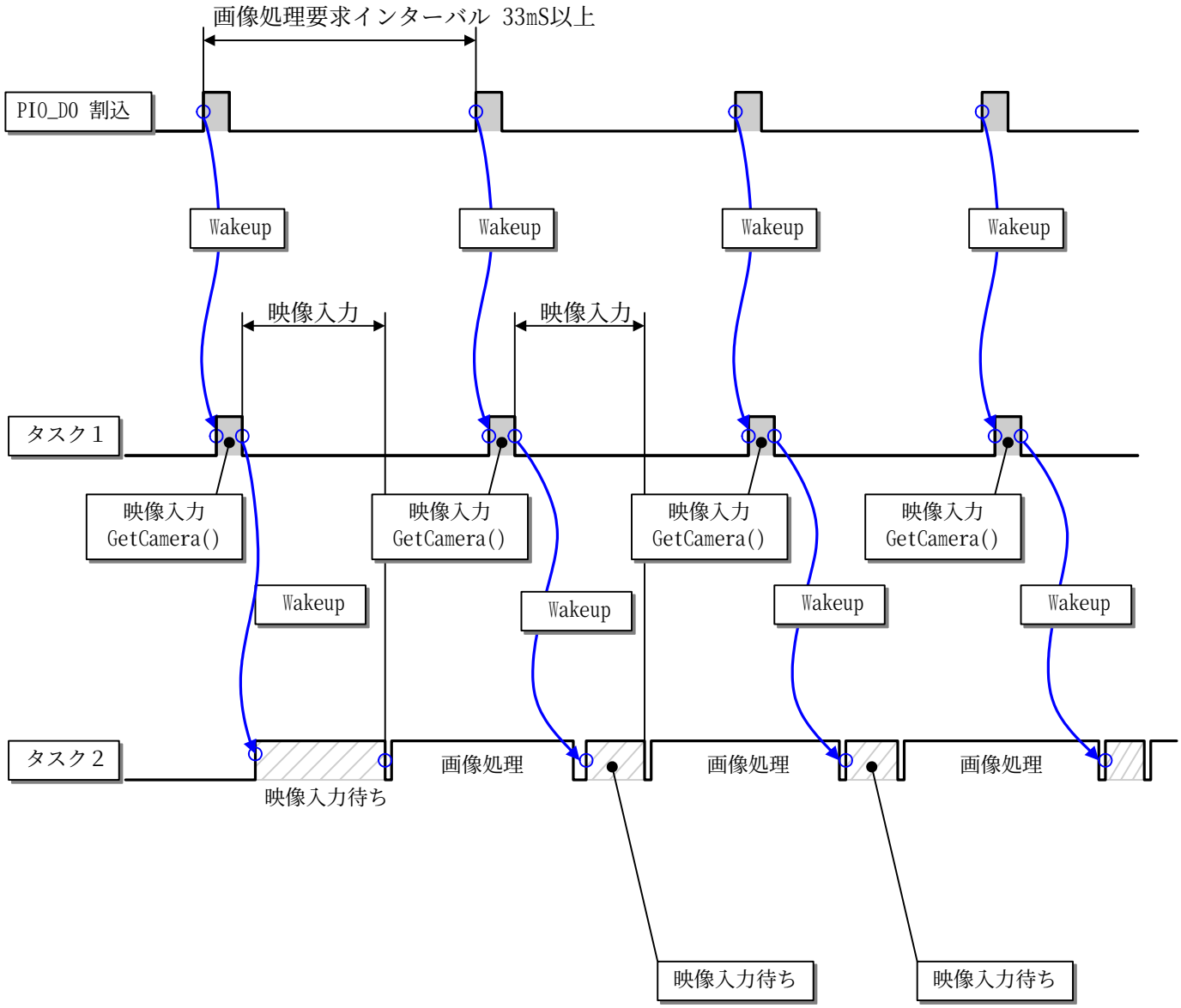


図１０－３　非同期映像入力のタイムチャート

10．1．2　タスクプライオリティの変更

非同期映像入力を実現するためには、割込起動タスク毎のタスクの優先順位（プライオリティ）を変更する必要があります。初期状態では、すべてのタスクが同じプライオリティになっています。

ランダムトリガによる非同期映像入力では、トリガ出力タスクを画像処理のタスクよりもプライオリティを高くする必要があります。また、映像入力タスクからの画像処理タスクの起動による非同期映像入力では、映像入力タスクを画像処理タスクよりもプライオリティを高くする必要があります。

タスクのプライオリティは、ChangeIPTaskPriority()コマンドで行います。

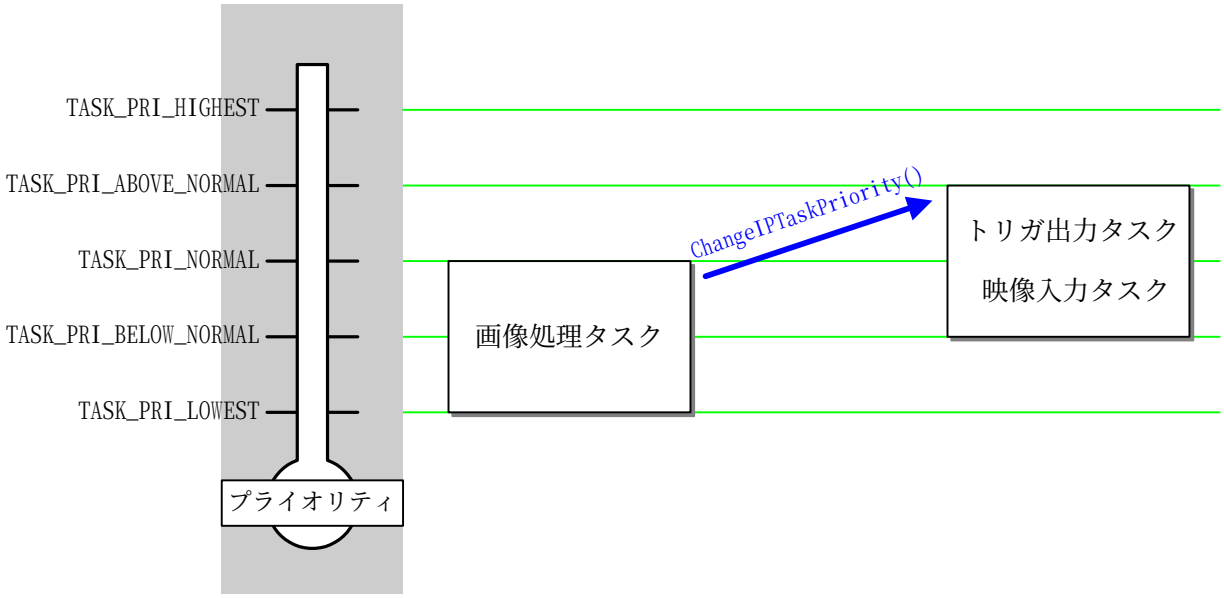


図１０－４　タスクプライオリティの変更

10.1.3 タスクコントロールコマンド一覧

非同期映像入力をサポートするコマンドとして、以下のタスクコントロールコマンドを用意しています。詳細は、コマンドリファレンスを参照して下さい。

表１０－１　タスクコントロールコマンド一覧

コマンド名	機能	備考
ChangeIPTaskPriority	タスクプライオリティの変更	
EnablePIOInterrupt	ＰＩＯ割込マスクの解除	
DisablePIOInterrupt	ＰＩＯ割込のマスク	
SleepIPTask	モジュールのスリープ	
WaitIPTask	モジュールの時間指定スリープ	
GetIPTaskStatus	タスクステータスの取得	
CancelWakeup	ウェイクアップ要求無効化	
WaitSemaphore	セマフォウェイト	
SignalSemaphore	セマフォシグナル	
SemStatus	セマフォステータスの取得	
SignalIPSemaphore	ＰＣからのセマフォシグナル	
SuspendSystemTimer	システムタイマの一時停止	
ResumeSystemTimer	システムタイマの一時停止解除	
CreateIPSemaphore	セマフォの生成	
DeleteIPSemaphore	セマフォの削除	
ReadSemaphore	セマフォの状態参照	
WaitSemaphoreWithTimeOut	タイムアウト付きセマフォウェイト	
CreateEventFlag	イベントフラグの生成	
DeleteEventFlag	イベントフラグの削除	
SetEventFlag	イベントフラグのセット	
ClearEventFlag	イベントフラグのクリア	
WaitEventFlag	イベントフラグウェイト	
WaitEventFlagWithTimeOut	タイムアウト付きイベントフラグウェイト	
ReadEventFlag	イベントフラグの状態参照	
DelayIPTask	タスク遅延	
ReleaseWait	待ち状態の強制解除	

第11章

エラー発生時の対策と手順

11.1 エラー発生時の対策と手順

コマンドがエラーを検出した場合、ユーザに対して報告する手段とその内容、及びエラーの回復について説明します。

11.1.1 エラー発生時の処理

画像処理コマンドでパラメータエラー等のエラーが発生時した場合、Ｗｉｎｄｏｗｓのメッセージボックスでユーザーにそれを知らせる機能があります。

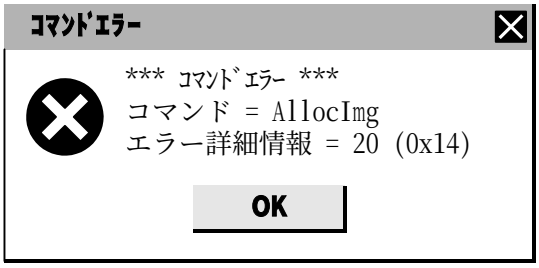


図１１－１　コマンドエラー処理

11.1.2 エラー処理の制御

画像処理コマンドではシステムでエラーの制御を行っています。画像処理コマンドでは、一旦、画像処理コマンドエラーが発生すると引き続く画像処理コマンドで処理を実行しません。再び処理を再開するには ClearIPError コマンドで画像処理コマンドエラーをクリアして下さい。

また、画像処理コマンドでエラーが発生した際にエラーのメッセージボックスが表示されますが、EnableIPErrorMessage , DisableIPErrorMessageでその出力を制御することが可能です。

11.1.3 エラーの回復とエラー情報の読み込み

ドライバがユーザに報告するエラー情報を読み込むには、ReadIPErrorTableコマンドを発行します。また、同コマンドを発行することにより、エラーは自動的に回復します。

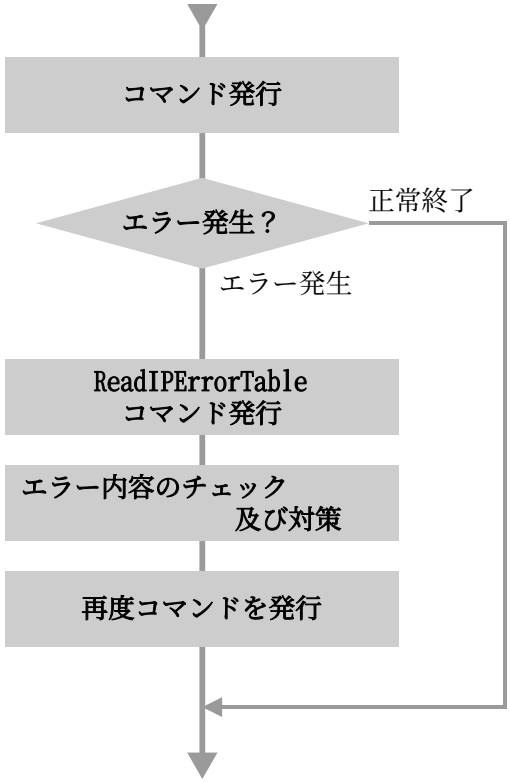


図１１－２　エラー情報読み込みフロー

11.2 コマンドエラー

以下にコマンドエラーのエラー番号と内容、対策を示します。

11.2.1 ハードウェア／システムエラー

エラー番号		エラー内容	対策	備考
1	0x01	画像処理ボード未実装エラー	画像処理ボードが実装されているか確認して下さい	
2	0x02	ＰＣドライバシステムエラー	ＰＣドライバがインストールされているかどうか確認して下さい	
3	0x03	ハードウェアエラー	開発元へ連絡して下さい	
4	0x04	レジスタテストエラー	InitIPExt内のレジスタテストでエラーが発生しました。	
5	0x05	コマンドテストエラー	InitIPExt内のコマンドテストでエラーが発生しました。	
6	0x06	画像メモリテストエラー	InitIPExt内のコマンドテストでエラーが発生しました。	

11.2.2 不当画面番号エラー

エラー番号		エラー内容	対策	備考
10	0x0A	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	0x0B	ImgSrc1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	
12	0x0C	ImgSrc2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	
13	0x0D	ImgDst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	
14	0x0E	ImgYUV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	
15	0x0F	ImgRGB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	

11.2.3 パラメータエラー

以下のパラメータエラーのエラー内容は、デバイスＩＤ以外引数の順番です。

エラー番号		エラー内容	対策	備考
20	0x14	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	0x15	第２引数の設定エラー	引数の設定値を確認して下さい	
22	0x16	第３引数の設定エラー	引数の設定値を確認して下さい	
23	0x17	第４引数の設定エラー	引数の設定値を確認して下さい	
24	0x18	第５引数の設定エラー	引数の設定値を確認して下さい	
25	0x19	第６引数の設定エラー	引数の設定値を確認して下さい	
26	0x1A	第７引数の設定エラー	引数の設定値を確認して下さい	
27	0x1B	第８引数の設定エラー	引数の設定値を確認して下さい	
28	0x1C	第９引数の設定エラー	引数の設定値を確認して下さい	
29	0x1D	第１０引数の設定エラー	引数の設定値を確認して下さい	

11.2.4 ウィンドウ設定エラー

エラー番号		エラー内容	対策	備考
30	0x1E	ＳＲＣ０ウィンドウ設定エラー	ウィンドウの設定値を確認して下さい	
31	0x1F	ＳＲＣ１ウィンドウ設定エラー	ウィンドウの設定値を確認して下さい	
32	0x20	ＳＲＣ２ウィンドウ設定エラー	ウィンドウの設定値を確認して下さい	
33	0x21	ＤＳＴウィンドウ設定エラー	ウィンドウの設定値を確認して下さい	
34	0x22	ＳＹＳウィンドウ設定エラー	ウィンドウの設定値を確認して下さい	
35	0x23	ＥＸＴウィンドウ設定エラー	ウィンドウの設定値を確認して下さい	
36	0x24	ＤＳＴ拡張ウィンドウ設定エラー	ウィンドウの設定値を確認して下さい	

11.2.5 一般エラー

エラー番号		エラー内容	対策	備考
40	0x28	画像メモリオープンエラー	画像メモリオープン処理を 行って下さい	
41	0x29	使用手続きエラー	関係するルーチンの説明を 読み直して下さい	
42	0x2A	カメラ入力での 画面サイズミスマッチエラー	カメラ入力する画面サイズを 確認して下さい	
43	0x2B	映像表示での 画面サイズミスマッチエラー	映像表示する画面サイズを 確認して下さい	
44	0x2C	座標設定エラー	座標の設定値を確認して下さい	
45	0x2D	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	0x2E	画面Allocエラー（空領域なし）	不要な画面を開放して リトライして下さい	
47	0x2F	画面Allocエラー（重複）	不要な画面を開放して リトライして下さい	
48	0x30	動的メモリ確保エラー （ワークエリアがmallocで確保できなかった）	ユーザ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	0x31	不定画面エラー （パイプライン処理で不定画面に設定された）	ユーザプログラムによる画像処理 （パイプライン）を確認して下さい	
50	0x32	画像メモリのチャネル競合エラー	動的画面の場合 　不要な画面を開放して再配置できるようにして下さい 静的画面の場合 　画像処理のソース画面とデスティネーション画面が同一チャネルでないかどうか確認して下さい	
51	0x33	ワーク画面確保エラー	不要な画面を開放して リトライして下さい	
52	0x34	カメラ入力でのビデオ画面 サイズミスマッチエラー	カメラ入力するビデオ画面 サイズを確認して下さい	
53	0x35	カメラ入力画面エラー	Get2Cameraの画面制約に従った配置で画面を確保して下さい	
54	0x36	カメラ映像入出力での再配置エラー	不要な画面を開放して 再配置できるようにして下さい	
55	0x37	Get2Cameraでのカメラタイプエラー	Get2Cameraは、BW_CAMERA以外のカメラでは使用できません	

11.2.6 正規化相関サーチでのエラー

エラー番号		エラー内容	対策	備考
80	0x50	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	
81	0x51	テンプレートの不当番号エラー	確保した番号が正しく設定されているか確認して下さい	
82	0x52	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	0x53	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	0x54	リザーブ		
85	0x55	リザーブ		
86	0x56	マスク処理での テンプレート登録エラー	テンプレート画像が「0」データの可能性があります。確認して下さい。	
87	0x57	正規化相関用テンプレート 未設定エラー	SetCorrTemplate等のコマンドで正しくテンプレートを設定して下さい	
88	0x58	テンプレート登録番号エラー	登録されたテンプレート画面が有効になっているか確認して下さい	
89	0x59	リザーブ		

11.2.7 画像ファイリングエラー

エラー番号		エラー内容	対策	備考
100	0x64	画像ファイルオープンエラー	画像ファイルのファイル名を 確認して下さい	
101	0x65	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どうか確認して下さい	

11.2.8 Windows API エラー

エラー番号		エラー内容	対策	備考
120	0x78	WindowsAPI コマンドエラー	マニュアルを確認して下さい	
121	0x79	Win32 API CreateXXX関数オープン （内部コマンド）エラー	開発元に連絡して下さい	

11.2.9 ファイルロードエラー

エラー番号		エラー内容	対策	備考
200	0xC8	Windowsレジストリアクセスエラー	Windowsへのレジストリのアクセスができない状態です。レジストリのアクセス権が適切かどうかとドライバが正常にインストールされているかどうか確認して下さい。	
201	0xC9	ファイルオープンエラー	指定されたファイルがあるかどうか確認して下さい。	
202	0xCA	ファイルアクセスエラー	指定されたファイルが正しいかどうかと壊れていないかどうか確認して下さい。	
203	0xCB	Windowsメモリエラー	開発元へ連絡して下さい	
204	0xCC	Windowsメモリ確保エラー	開発元へ連絡して下さい	
205	0xCD	ダウンロードベリファイエラー	ダウンロードファイルが壊れていないかどうか確認して下さい	
206	0xCE	ターゲットコードエラー	コマンドファイルがターゲットと一致していません。ファイルを確認下さい	
207	0xCF	ダウンロード領域メモリ確保エラー	ダウンロード領域のメモリ確保に失敗しました。領域を確認して下さい	

11.2.10 不当表示画面番号エラー

エラー番号		エラー内容	対策	備考
210	0xD2	ビデオ表示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	
211	0xD3	DISP画面の不当画面番号エラー	DISP画面の画面番号を確保しているか 確認して下さい	

11.2.11 一般処理エラー

エラー番号		エラー内容	対策	備考
220	0xDC	0除算エラー	パラメータ正しいかどうか 確認して下さい	
221	0xDE	対象画素またはデータがありません	パラメータ正しいかどうか 確認して下さい	
222	0xDF	プリフェッチモードが有効になっていません	カメラプリフェッチモードを有効にして下さい	

11.2.12 モジュール･タスク管理エラー

エラー番号		エラー内容	対策	備考
260	0x104	タスク管理システムエラー	ＩＴＲＯＮのリソース等の確認をして下さい	
261	0x105	タスク生成オーバーフロー	不要なタスクを削除して下さい	
262	0x106	２重登録エラー	以前に登録していたオブジェクトを削除して下さい	
263	0x107	未登録タスク指定エラー	登録されているタスクを指定して下さい。	
264	0x108	パラメータバッファ不足エラー	パラメータバッファ容量を大きくして下さい	

11.2.13 ITRONサービスコールエラー

割込モジュール制御コマンド及びタスクコントロールコマンドで発生するエラーです。

エラーコード		エラー内容	備考
-9	0xFFFF:FFF7	未サポート機能	
-10	0xFFFF:FFF6	サービスコールが組み込まれていない	
-11	0xFFFF:FFF5	予約属性（属性が不正）	
-17	0xFFFF:FFEF	パラメータエラー	
-18	0xFFFF:FFEE	不正ＩＤ番号	
-25	0xFFFF:FFE7	コンテキストエラー	
-28	0xFFFF:FFF4	サービスコール不正使用	
-33	0xFFFF:FFDF	メモリ不足	
-34	0xFFFF:FFDE	ＩＤ番号不足	
-41	0xFFFF:FFD7	オブジェクトの状態不正	
-42	0xFFFF:FFD6	オブジェクトが存在しない	
-43	0xFFFF:FFD5	キューイングまたはネストのオーバフロー	
-49	0xFFFF:FFCF	待ち状態強制解除	
-50	0xFFFF:FFCE	ポーリング失敗またはタイムアウト	
-51	0xFFFF:FFCD	待ちオブジェクトが削除された	

第12章

マルチポート映像入力

ＳＶＰ－３３０には、モノクロ／カラーのビデオプロセッサが２ユニットあります。２つのビデオプロセッサはそれぞれ独立して映像入力を行うことができます。ＳＶＰ－３３０の画像処理コマンドでは、それらの入力をサポートするコマンドを用意しています。

12.1 マルチポート映像入力の概要

図１２－１にＳＶＰ－３３０の映像入力系のハードウェア構成を示します。ＳＶＰ－３３０は、カメラポート＃０、＃１は、それぞれカラー映像のＮＴＳＣデコーダ＃０、＃１に接続されます。ＮＴＳＣデコーダ＃０はビデオプロセッサ(ビデオポート)＃０、ＮＴＳＣデコーダ＃１はビデオプロセッサ(ビデオポート)＃１に接続されており、それぞれのビデオプロセッサ(ビデオポート)は独立して映像入力が可能です。また、ＳＶＰ－３３０は、カラー映像とモノクロ映像を同時あるいは非同期に入力することができます。

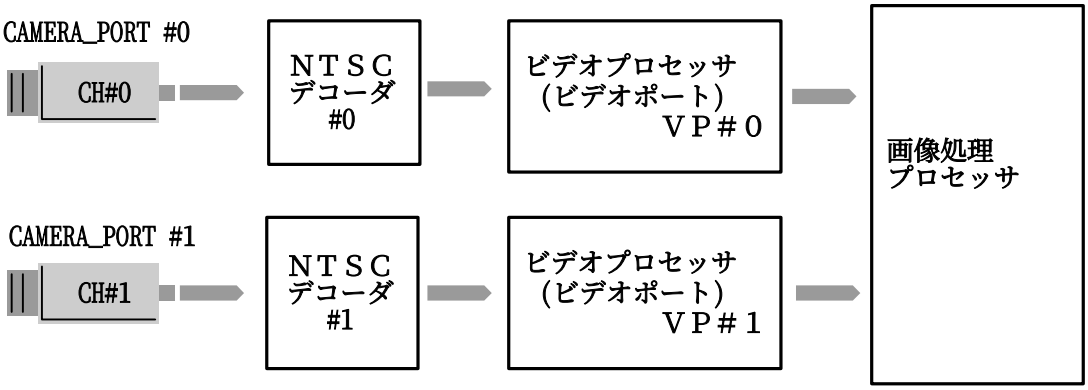


図１２－１　映像入力ハードウェア構成

12.1.1 カメラポート配置の設定

InitIP( )コマンド発行後は「PORTCFG_NORMAL」設定（ノーマルモード）になります。

ＳＶＰ－３３０は、カメラポートをSelectCamera( )コマンドでカメラ選択する場合、カメラポート#0、#1を指定することで、自動的にビデオポート＃０／＃１が選択され、選択されたポートがカレントのビデオポートになります。

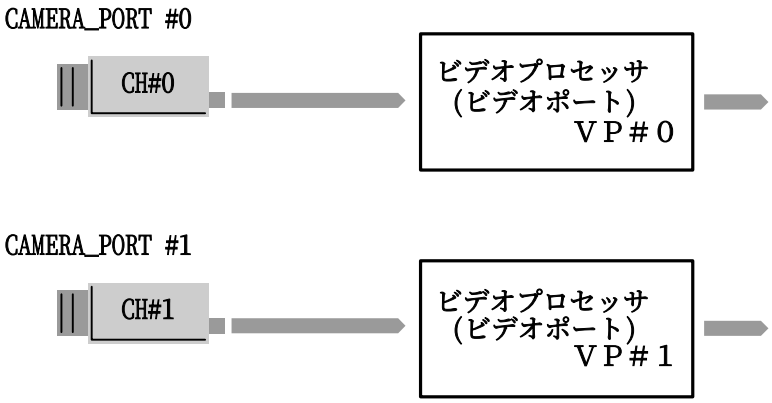


図１２－２　PORTCFG_NORMAL 構成

12.1.2 マルチポート映像入力コマンド一覧

マルチポート映像入力をサポートするコマンドとして、以下のマルチポート映像入力コマンドを用意しています。詳細は、コマンドリファレンスを参照して下さい。

表１２－１　マルチポート映像入力コマンド一覧

コマンド名	機能	備考
SetCameraPortConfig	カメラポート配置の設定	
ActiveVideoPort	カレントビデオポートの設定	
SetVideoFrameMltPort	マルチポート対応映像入力画面設定	
SelectCameraMltPort	マルチポート対応カメラ番号選択	
GetCameraMltPort	マルチポート対応映像入力	
DispCameraMltPort	マルチポート対応カメラの表示	
SetTriggerModeMltPort	マルチポート対応トリガモードの設定	SVP-330では未ｻﾎﾟｰﾄ
SetShutterSpeedMltPort	マルチポート対応シャッタースピードの設定	SVP-330では未ｻﾎﾟｰﾄ
Get2CameraMltPort	マルチポート対応２カメラ同時入力	SVP-330では未ｻﾎﾟｰﾄ
EnableLoopCameraMltPort	マルチポート対応ループ映像入力開始	
SuspendLoopCameraMltPort	マルチポート対応ループ映像入力停止	

第13章

YUVカラー処理コマンド

13．1　YUVカラー処理機能

カラー処理機能には、以下の３点の機能があります。

- カラーカメラ（NTSC）からの映像取り込み
- カラー映像の表示出力
- カラー処理
  - ・ＹＵＶカラー抽出（ＹＵＶ→２値）
  - ・ＹＵＶカラー抽出（色彩距離・色相変換・色相→２値）
  - ・ＲＧＢカラー抽出（ＲＧＢ→２値）
  - ・ＲＧＢカラー抽出（色相・彩度・明度→２値）

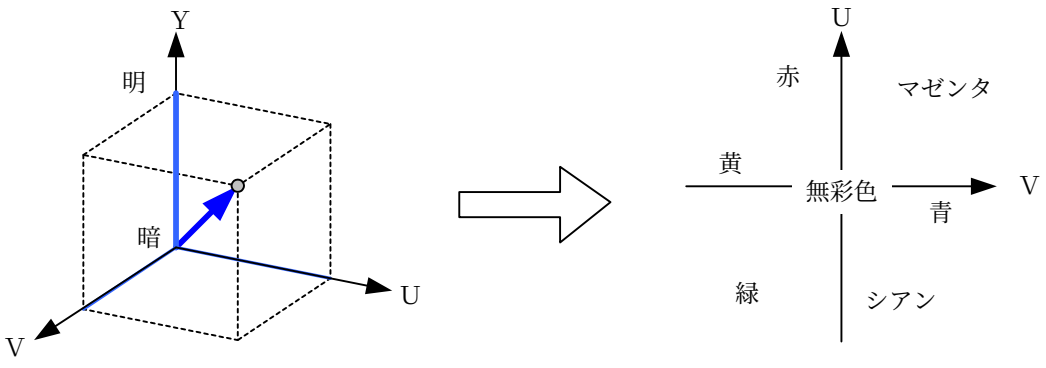
画像処理については、１３．２で説明していますので詳細はそちらを参照して下さい。

13.1.1　カラーNTSC信号について

ＮＴＳＣ信号は、複合映像信号であり、同期信号、輝度信号（白黒信号）、色信号が複合されています。輝度信号は、輝度情報を表し、モノクロデータと同じです。色差信号は、色情報を表します。よく知られているように色はＲＧＢの３色で表しますが、カラーＮＴＳＣでは、これを、輝度と色差によって表します。色差信号は２つあり、１つはＲ（赤）からＹ（輝度）を引いた信号でＲ－Ｙ信号ともＣr信号ともＵ信号とも呼ばれます。もう１つはＢ（青）からＹ（輝度）を引いた信号でＢ－Ｙ 信号ともＣb 信号ともＶ信号とも呼ばれます。ＲＧＢとＹＵＶの関係は下記式で表わされます。このマニュアルでは、色差情報をＵとＶで表記します。ＲＧＢとＹＵＶの関係は下式で表わされます。

ＲＧＢとＹＵＶの関係

```
R ＝Y ＋1.3707 ×（U －128 ）
G ＝Y －0.6980 ×（U －128 ）－0.3364 ×（V －128 ）
B ＝Y ＋1.7324 ×（V －128 ）
```



ＹＵＶの立体表示図　　色差平面図

図１３－１　Ｙ（輝度）、ＵＶ（色差）の概念図

［参考］カラーバーに対するＹＵＶ値

	Y	U	V		Y	U	V
White	235	128	128	Blue	41	110	240
Black	16	128	128	Yellow	210	146	16
Red	82	240	90	Cyan	170	16	166
Green	145	34	54	Magenta	106	222	202

13.1.2 カラーカメラ(ＮＴＳＣ)からの映像取り込み

YUVカラー映像では、輝度情報の他に色差情報が必要になります。カラー映像を取り込むと、カラー映像はディジタル化され輝度（Ｙ）成分と色差（Ｕ,Ｖ）成分に分けて、画像メモリ上にＹ画面／ＵＶ画面の２画面として取り込まれます。ＹとＵＶは４：２：２で画像メモリに取り込まれ、色差成分Ｕ／Ｖはデコード時に時分割処理されてＵＶ画面に交互に格納されます。

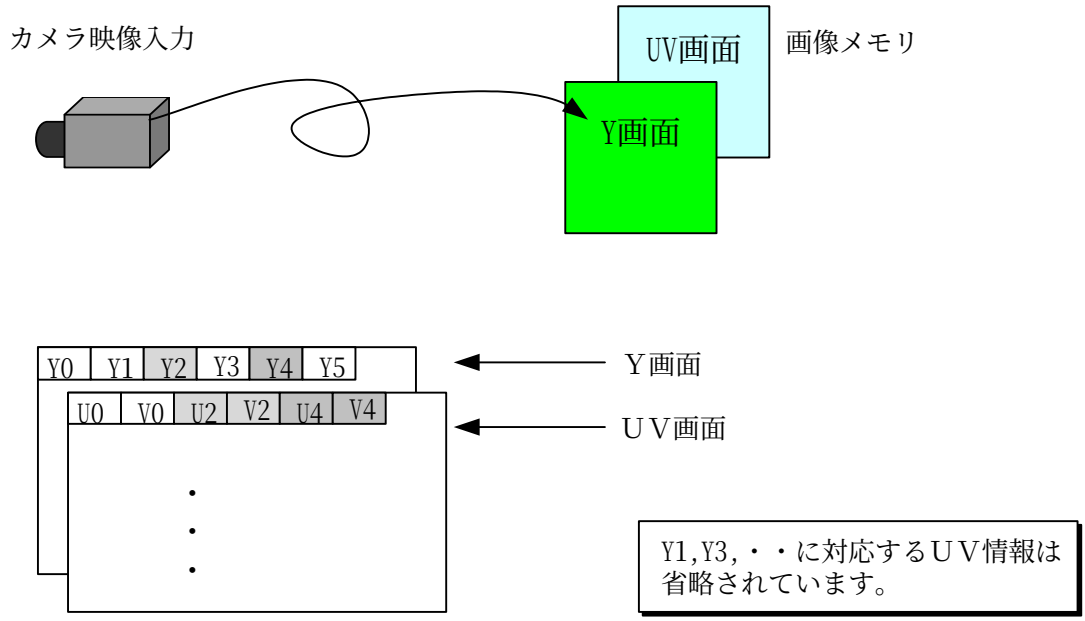


図１３－２ ＮＴＳＣカラー映像入力

13.1.3 YUVカラー映像の表示出力

カラーNTSC カメラから取り込んだYUVカラー映像を表示、出力します。

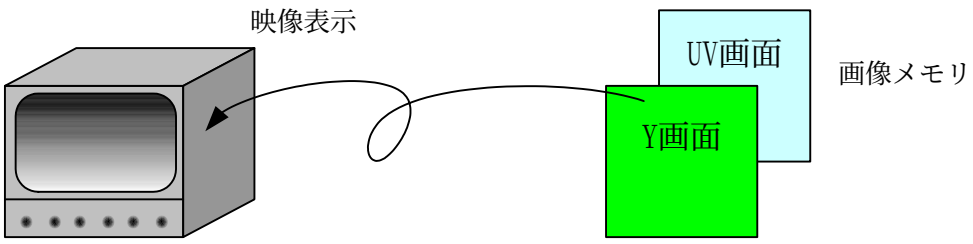


図１３－３ ＹＵＶ画像の表示

13．2　YUVカラー抽出処理

13．2．1　YUVカラー抽出

ＹとＵＶデータから構成されたカラー画像から指定されたＹ／Ｕ／Ｖ色の部分を抽出し、２値画像としてモノクロ画面に転送します。

13．2．2　YUVカラー抽出（色彩距離･色相）

ＹとＵＶデータから構成されたカラー画像から指定されたＹ(輝度)、色彩距離(ρ)、色相(φ)の部分を抽出し、２値画像としてモノクロ画面に転送します。

（１）色彩距離・色相について

ＮＴＳＣ信号のＹＵＶによる色の表示系は、物理色の組み合わせによる色の表現であり、人間には、意識しにくい色の表現方法です。

人間の視感覚に順応した色の表現方法として、明るさ(輝度)、あざやかさ(彩度)、色合い／色調（色相）による色の表現方法があります。

・明るさ／輝度（Ｙ）
　色の明るさを示します。ＹＵＶのＹ(輝度)を示します。
・色彩距離（ρ）
　色のあざやかさを示します。無彩色を0として、値が大きいほどあざやかであることを示します。色彩距離は下式で表わされます。ρの範囲は 0～181 です。

$$\rho = \sqrt{\{(U-128)^2 + (V-128)^2\}} \qquad U=V=0 \text{ のとき、} \rho=181$$

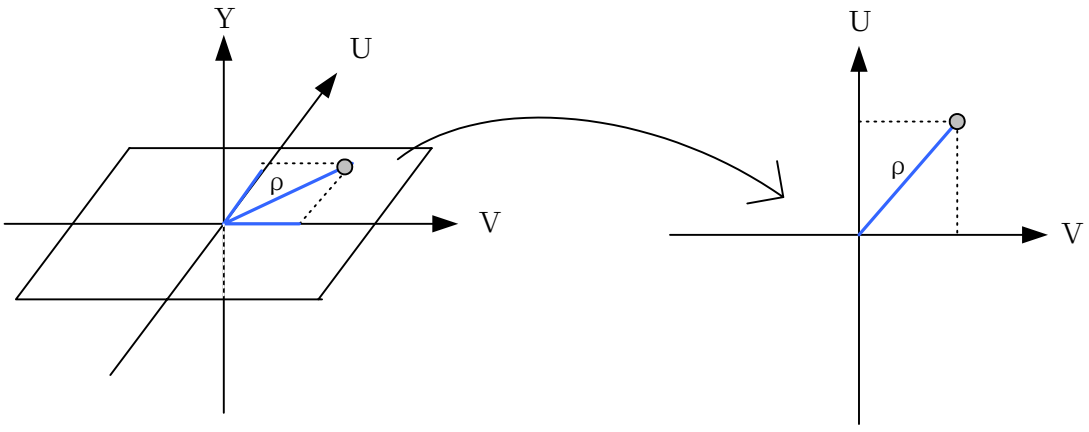


図１３－４　色彩距離（ρ）の概念図

・色相（θ）
　色を円周上に配列していると考え、色あい／色調を角度で表わしたものです。色相は下式で表わされます。θの範囲は0 ～360 です。

$$\theta = \arctan \frac{(U - 128)}{(V - 128)}$$

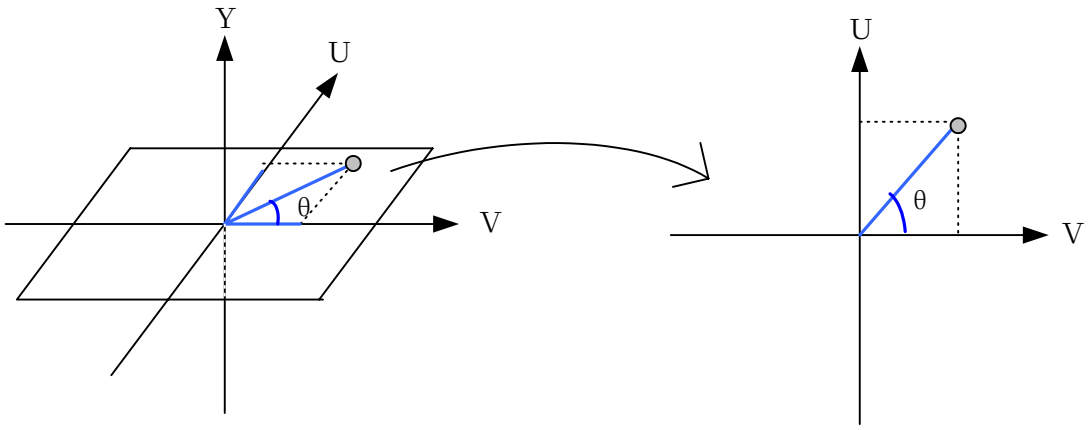
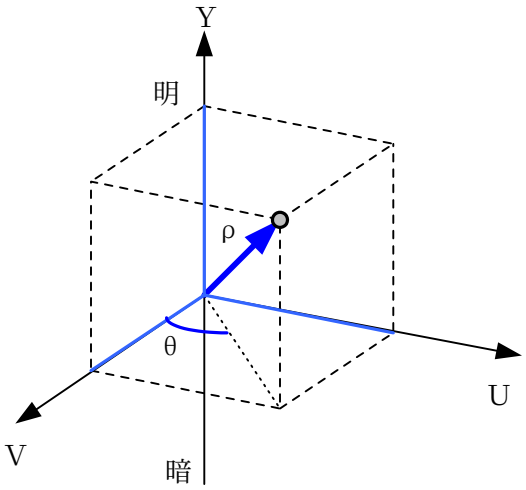


図１３－５　色相（θ）の概念図

　色彩距離（ρ）、色相（θ）にて色差を表現することにより、ＵＶという物理量より、人間の視覚感覚に近い色表現ができるようになります。



$$\rho = \sqrt{\{(U-128)^2 + (V-128)^2\}}$$

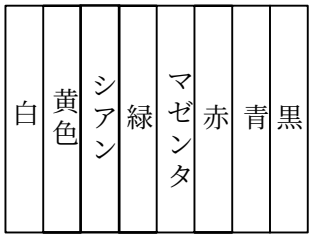
$$\theta = \arctan \frac{(U - 128)}{(V - 128)}$$

図１３－６　色彩距離（ρ）と色相（θ）の概念図

13.2.3　サンプルプログラム

① カラーバーパターン作成

画像メモリにカラーバーのパターンを書き込み表示します。




```
#include   <stdio.h>
#include   <stdlib.h>

#include   "vpxdef.h"
#include   "vpxsys.h"
#include   "vpxfnc.h"

#define  XSIZE  512
#define  YSIZE  512


void main()
{
   int devID,ImgYUV;
   int i,j;
   unsigned char *data;

   /* システムイニシャライズ*/
   devID = OpenIPDevExt( IPBOARD_0, NULL );
   InitIP( devID);

   SelectCamera( devID, CAMERA_PORT0, YUV_CAMERA);


   /* 画面確保 */
   ImgYUV=AllocYUVImg( devID, IMG_FS_512H_512V);
   data = (unsigned char *) malloc( sizeof( unsigned char)* XSIZE* YSIZE* 2 );


   /* Yパターン作成 */
   for(i=0; i<480; i++){
      memset(data+ (Xsize*i)    , 0xEB, 64);    /* White   */
      memset(data+ (Xsize*i)+ 64, 0xD2, 64);    /* Yellow  */
      memset(data+ (Xsize*i)+128, 0xAA, 64);    /* Cyan    */
      memset(data+ (Xsize*i)+192, 0x91, 64);    /* Green   */
      memset(data+ (Xsize*i)+256, 0x6A, 64);    /* Magenta */
      memset(data+ (Xsize*i)+320, 0x52, 64);    /* Red     */
      memset(data+ (Xsize*i)+384, 0x29, 64);    /* Blue    */
      memset(data+ (Xsize*i)+448, 0x10, 64);    /* Black   */
   }
```

このインクルードファイルは必ずインクルードして下さい

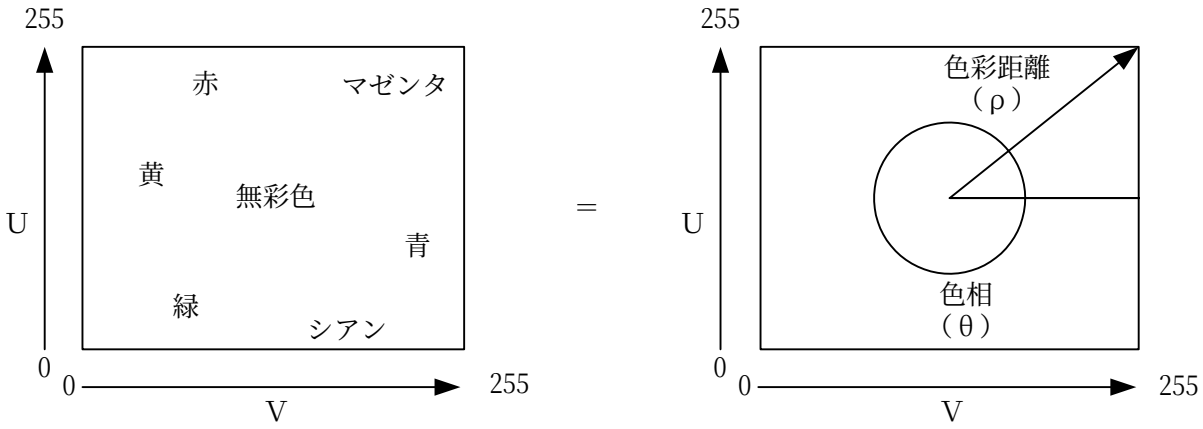
カラーNTSC カメラに設定

カラー映像用画像メモリ領域確保

```
    /* UVパターン作成 */
    for(i=0; i<480; i++){
       /* White */
       for(j=0; j<64; j+=2){
          *(data+ (Xsize* Ysize)+ (Xsize* i)+ j)  = 0x80;    /* U= 128 */
          *(data+ (Xsize* Ysize)+ (Xsize* i)+ j+1)= 0x80;    /* V= 128 */
       }
       /* Yellow */
       for(j=64; j<128; j+=2){
          *(data+ (Xsize* Ysize)+ (Xsize* i)+ j)  = 0x92;    /* U= 146 */
          *(data+ (Xsize* Ysize)+ (Xsize* i)+ j+1)= 0x10;    /* V= 10 */
       }
       /* Cyan */
       for(j=128; j<192; j+=2){
          *(data+ (Xsize* Ysize)+ (Xsize* i)+ j)  = 0x10;    /* U= 16 */
          *(data+ (Xsize* Ysize)+ (Xsize* i)+ j+1)= 0xA6;    /* V= 166 */
       }
       /* Green */
       for(j=192; j<256; j+=2){
          *(data+ (Xsize* Ysize)+ (Xsize* i)+ j)  = 0x22;    /* U= 34 */
          *(data+ (Xsize* Ysize)+ (Xsize* i)+ j+1)= 0x36;    /* V= 34 */
       }
       /* Magenta */
       for(j=256; j<320; j+=2){
          *(data+ (Xsize* Ysize)+ (Xsize* i)+ j)  = 0xDE;    /* U= 222 */
          *(data+ (Xsize* Ysize)+ (Xsize* i)+ j+1)= 0xCA;    /* V= 202 */
       }
       /* Red */
       for(j=320; j<384; j+=2){
          *(data+ (Xsize* Ysize)+ (Xsize* i)+ j)  = 0xF0;    /* U= 240 */
          *(data+ (Xsize* Ysize)+ (Xsize* i)+ j+1)= 0x5A;    /* V= 90 */
       }
       /* Blue */
       for(j=384; j<448; j+=2){
          *(data+ (Xsize* Ysize)+ (Xsize* i)+ j)  = 0x6E;    /* U= 110 */
          *(data+ (Xsize* Ysize)+ (Xsize* i)+ j+1)= 0xF0;    /* V= 240 */
       }
       /* Black */
       for(j=448; j<512; j+=2){
          *(data+ (Xsize* Ysize)+ (Xsize* i)+ j)  = 0x80;    /* U= 128 */
          *(data+ (Xsize* Ysize)+ (Xsize* i)+ j+1)= 0x80;    /* V= 128 */
       }
    }

    /* 画像メモリのカラーパターン書き込み */
    OpenImg( devID,WAIT_FOREVER );
    WriteImg( devID,ImgYUV, data, Xsize* Ysize );
    CloseImg( devID);

    free(data);

    DispImg( devID,ImgYUV);
}
```

② カラーグラデーションパターン作成

画像メモリにカラーのグラデーションパターンを書き込み表示します。左下（0、511）を始点とし、Ｙ方向のＵとＸ方向のＶを0 ～255 に変化させたパターンを書き込みます。
このパターンで色彩距離（ρ）と色相（θ）の関係が表わせます。




```
#include   "vpxdef.h"
#include   "vpxsys.h"
#include   "vpxfnc.h"

void pattern_make(int,int);

void main()
{
   int devID,ImgYUV;

   /* システムイニシャライズ */
   devID = OpenIPDevExt( IPBOARD_0, NULL );
   InitIP( devID);

   SelectCamera( devID,CAMERA_PORT0, YUV_CAMERA );

   /* 画面確保 */
   ImgYUV= AllocYUVImg( devID,IMG_FS_512H_512V );

   /* カラーパターン作成 */
   pattern_make(devID,ImgYUV);
}
```

このインクルードファイルは必ずインクルードして下さい

カラーNTSC カメラに設定

カラー映像用画像メモリ領域確保

```
/* カラーパターン作成 */
void pattern_make  (int devID, int ImgYUV)
{
   int   ImgUV;
   unsigned char data[512], data1[512];
   int   i,j;

   ImgUV= GetUVImgID(devID,ImgYUV);

   /* Uパターン作成 */
   for(i=0,j=255; i<512; i+=2,--j){
      data[i]= j;
      data[i+1]= j;
   }

   OpenImg(devID,WAIT_FOREVER);
   for(i=0,j=0; i<511; i+=2,j++){
      /* Vパターン作成 */
      memset(data1,j,512);

      /* Uパターン描画 */
      SetAllWindow(devID,i,0,i,511);
      WriteImg(devID,ImgUV,data,512);
      DispImg(devID,ImgYUV);

      /* Vパターン描画 */
      SetAllWindow(devID,i+1,0,i+1,511);
      WriteImg(devID,ImgUV,data1,512);
      DispImg(devID,ImgYUV);
   }
   CloseImg(devID);


   /* Yパターン描画 */
   ResetAllWindow(devID);
   IP_Const(devID, ImgYUV, 0x80);

   /* カラー画面表示 */
   DispImg(devID,ImgYUV);
}
```

13.2.4 YUVカラー処理コマンド一覧

表１３-１にYUVカラー処理コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表１３－１ YUVカラー処理コマンド一覧

コマンド名	機能	備考
GetCamera	カメラ映像入力	
DispCamera	カメラ映像表示	
IP_ClearImg	画像メモリクリア（指定画面）	
IP_Copy	画像転送	
ReadImg	画像メモリ読み出し（矩形領域）	
WriteImg	画像メモリ書き込み（矩形領域）	
AllocYUVImg	カラー映像用画像メモリ領域確保	
GetUVImgID	ＵＶ画面番号取り出し	
ReadYUVImgTable	カラー画面管理テーブル読み出し	
IP_Mask	マスク処理	
IP_ClearColor	カラー画面の色情報クリア（指定画面）	
IP_ExtractColor	カラー抽出処理	
IP_ExtractColorRhoTheta	カラー抽出処理－色彩距離・色相変換	
IP_ConvertRho	色彩距離変換	
IP_ConvertTheta	色相変換	
IP_ConvertRhoTheta	色彩距離・色相変換	
IP_ConvertYUVtoRGB	カラー疑似変換（YUV →RGB ）	
IP_ConvertYUVtoRGBfast	カラー疑似変換（YUV →RGB ）	
OpenMultiColor	マルチカラー抽出処理可能	
CloseMultiColor	マルチカラー処理不可	
ClearMultiColorYRT	Ｙρθマルチカラーテーブルクリア	
SetMultiColorYRT	Ｙρθマルチカラー抽出データ設定	
IP_ExtractMultiColorRhoTheta	Ｙρθマルチカラー抽出	
ClearMultiColor	マルチカラーテーブルクリア	
SetMultiColor	マルチカラー抽出データ設定	
IP_ExtractMultiColor	マルチカラー抽出	

13.3 カラーカメラ(NTSC)からの映像取り込み

カラー映像では、輝度情報の他に色差情報が必要になります。カラー映像は輝度（Y）成分と色差（U,V）成分に分けて、画像メモリ上にY画面／ＵＶ画面の２画面として取り込まれます。色差成分U／Vはデコード時に分割処理され、ＵＶ画面に交互に格納されます。

13.3.1 YUVカラー時の画像メモリ使用方法

前に述べたようにYUVカラー映像データは、ＹとＵＶの２ つのデータで構成されるため、画像メモリも２画面を使用します。

カラー用の画像メモリを確保するコマンドがAllocYUVImg です。AllocYUVImg では、カラー映像用にＹ画面とＵＶ画面の２画面を確保します。ユーザにはこのうちＹ画面の画面番号を返し、ＵＶ用の画面はシステム内部で管理されます。

	確保コマンド	確保例
モノクロ画面	AllocImg ／AllocLockImg	（例）AllocImg （IMG_FS_512H_512V ） 512 512 画面番号 512 ×512 の画面を １画面確保
YUVカラー 画面	AllocYUVImg	（例）AllocYUVImg （IMG_FS_512H_512V ） 512 512 Ｙ画面 画面番号 ＵＶ画面 512 ×512 の画面を ２画面確保 Ｙ画面の画面番号 をユーザに返す

13.3.2 YUVカラー映像用画像メモリの確保

AllocYUVImg で確保されたカラー画面を使用して、カメラ映像を取り込むと、カラーＮＴＳＣ映像はディジタル化され輝度（Y）成分と色差（U，V）成分に分けて、画像メモリ上にＹ画面／ＵＶ画面の２画面として取り込まれます。ＹとＵＶは４：２：２で画像メモリに取り込まれ、ＵＶは２画素分が圧縮されます。

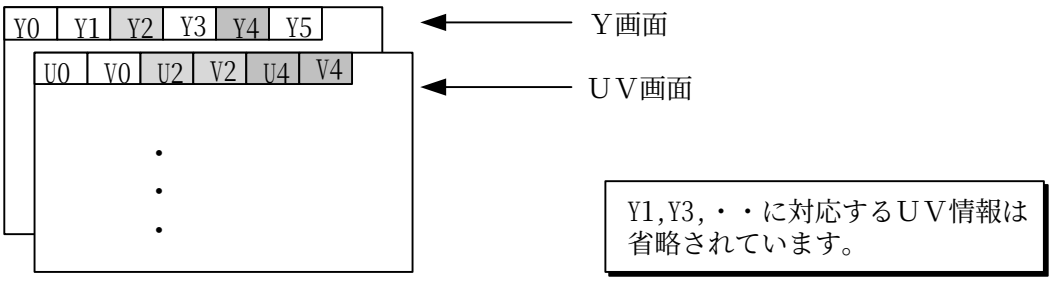


図１３－７ ＹＵＶ映像用画像メモリ

13．4　カラーカメラからのカメラ映像入力手順

カラーカメラ使用時のカメラ映像入力の手順を以下に示します。

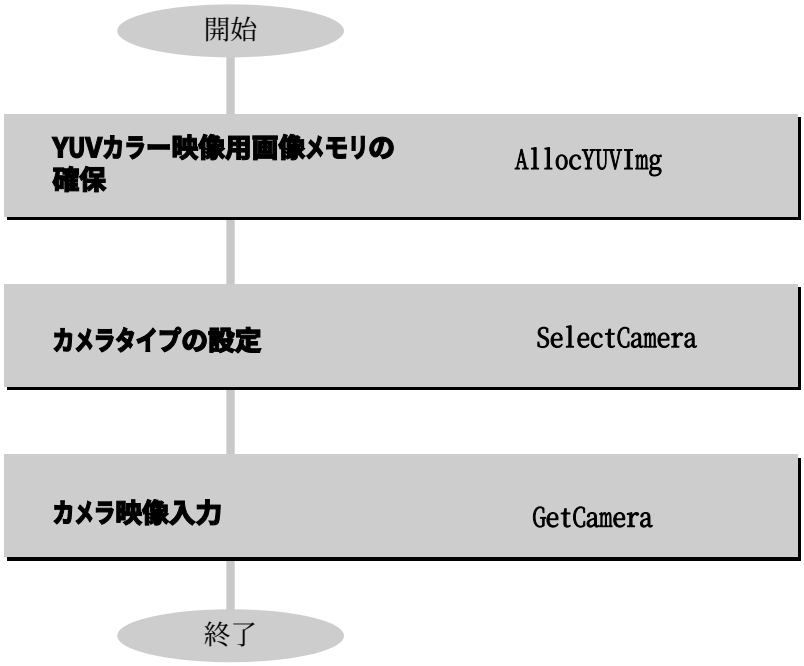


図１３－８ ＹＵＶカメラ映像入力フロー

13.4.1　カメラタイプの設定

接続するカメラの種類を指定します。カメラタイプをカラーカメラに指定します。

13.4.2　ビデオフレームサイズの設定

映像の入出力サイズであるビデオフレームサイズを設定します。

13.4.3　カメラ映像入力

カメラからの映像を入力します。設定例を以下に示します。

```
    InitIPExt(devID);

①   ImgYUV1=AllocYUVImg(devID,IMG_FS_512H_512V);
    ImgYUV2=AllocYUVImg(devID,IMG_FS_512H_512V);

②   SelectCamera(devID,CAMERA_PORT0,YUV_CAMERA);

③   GetCamera(devID,ImgYUV1);
```

① カラー映像用画像メモリを確保します。

② カメラタイプをカラーNTSCカメラに設定します。

③ カメラ映像入力を起動します。

13.4.4 YUVカラー画面の画像処理

AllocYUVImg で確保したカラー画面番号を使用することで、カラー画面に対する画像処理ができます。ただし、カラー画面に対して画像処理する場合は、画面転送（IP_Copy）を除いてY画面のみ処理します。ＵＶ画面は処理されません。

```
[実行例]

ImgYUV1=  AllocYUVImg(devID,IMG_FS_512H_512V);
ImgYUV2=  AllocYUVImg(devID,IMG_FS_512H_512V);
IP_Zoom(devID,ImgYUV1,ImgYUV2,2.0);
```

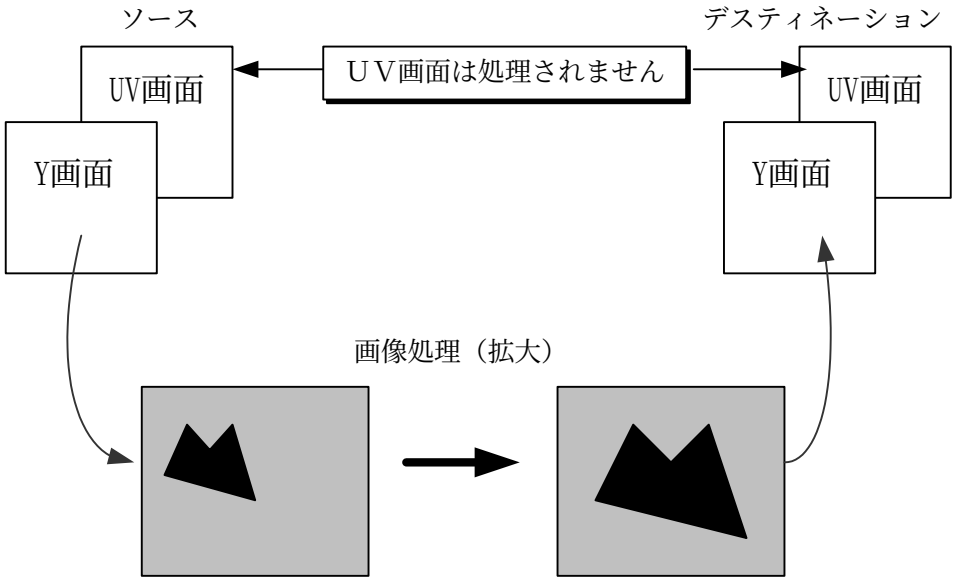
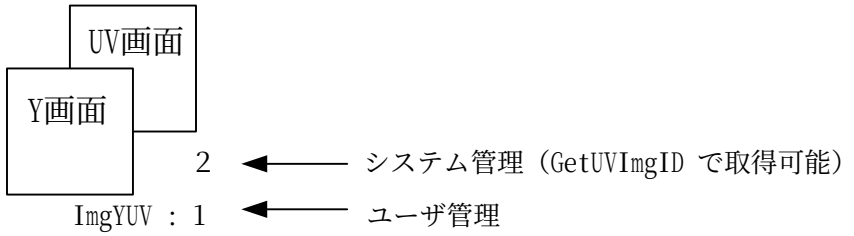


図１３－９ ＹＵＶの画像処理

ＵＶ画面に対して処理する場合は、Ｙ画面の画面番号で管理するカラー画面のＵＶ画面番号を取得し、このＵＶ画面番号を使用して実行できます。GetUVImgID コマンドでカラー画面のＵＶ画面番号を取得できます。




```
［実行例］

  ImgYUV1= AllocYUVImg(devID,IMG_FS_512H_512V);
  ImgYUV2= AllocYUVImg(devID,IMG_FS_512H_512V);

  ImgUV1= GetUVImgID(devID,ImgYUV1);  /* ImgYUV1 のUV 画面No.を取得 */
  ImgUV2= GetUVImgID(devID,ImgYUV2);  /* ImgYUV2 のUV 画面No.を取得 */

  IP_Zoom(devID,ImgUV1,ImgUV2,2.0);
```

● カラー画面の画像処理における注意事項

カラー画面では、UVデータはUV画面にUとVの順番で１画素おきに格納されています。
よって、ウィンドウの始点座標を奇数や、ウィンドウの横方向サイズを奇数にして画像処理を実行した場合に、UとVの順番が入れ替わり、カラーの色情報がくずれます。ウィンドウの始点座標は偶数に、ウィンドウの横方向サイズも偶数にしてください。

［実行例］

```
SetWindow(devID,SRC0_WIN, 1, 0, 100, 200);
IP_Copy(devID,ImgYUV1, ImgYUV2);
```

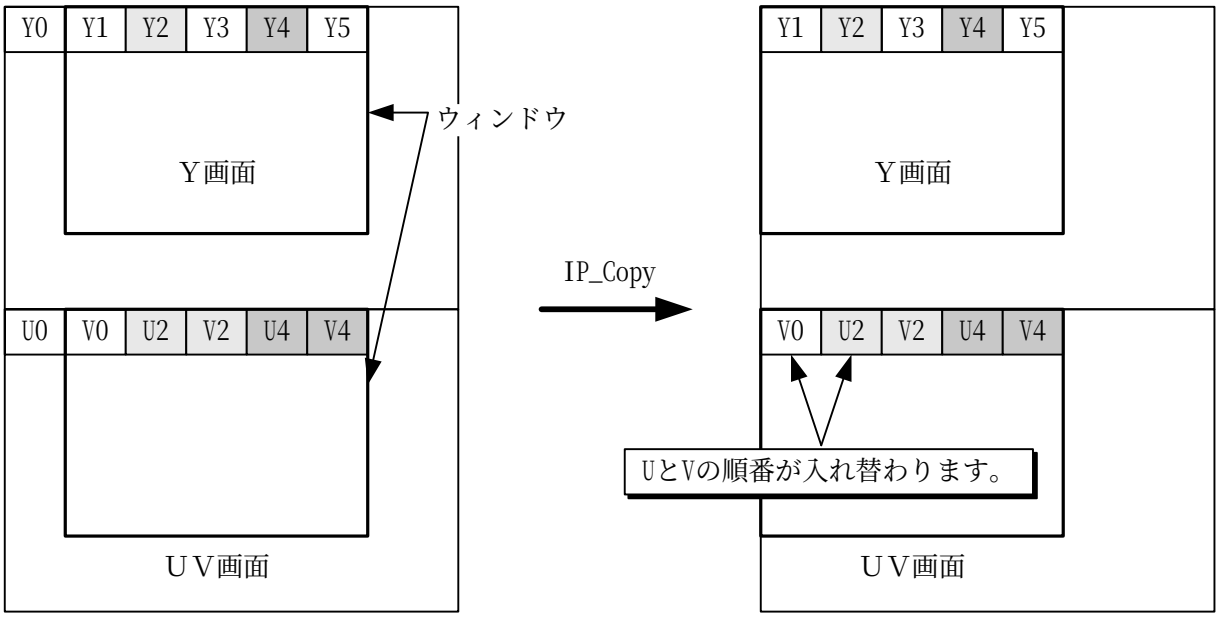


図１３－１０ ＹＵＶ画像処理における注意

第14章

RGBカラー処理コマンド

14．1　RGBカラー処理機能

ＲＧＢカラー処理機能には、以下の２点の機能があります。

- カラー画像の変換（ＲＧＢ→ＹＵＶ）
- カラー処理
  - ・ＲＧＢカラー抽出（ＲＧＢ→２値）
  - ・ＲＧＢカラー抽出（色相・彩度・明度→２値）

画像処理については、１４．２で説明していますので詳細はそちらを参照して下さい。

14.1.1　RBGとYUVの関係

ＮＴＳＣ信号は、複合映像信号であり、同期信号、輝度信号（白黒信号）、色信号が複合されています。輝度信号は、輝度情報を表し、モノクロデータと同じです。色差信号は、色情報を表します。よく知られているように色はＲＧＢの３色で表しますが、カラーＮＴＳＣでは、これを、輝度と色差によって表します。色差信号は２つあり、１つはＲ（赤）からＹ（輝度）を引いた信号でＲ－Ｙ信号ともＣr信号ともＵ信号とも呼ばれます。もう１つはＢ（青）からＹ（輝度）を引いた信号でＢ－Ｙ 信号ともＣb 信号ともＶ信号とも呼ばれます。このマニュアルでは、色差情報をＵとＶで表記します。ＲＧＢとＹＵＶの関係は下式で表わされます。

ＲＧＢとＹＵＶの関係

Ｙ ＝0.29900 ×Ｒ ＋0.58700 ×Ｇ ＋0.14400 ×Ｂ
Ｕ ＝0.72955 ×(Ｒ －Ｙ) +128
Ｖ ＝0.57722 ×(Ｂ －Ｙ) +128

表１４－１ カラーバーに対するＲＧＢ値

	R	G	B		R	G	B
White	255	255	255	Blue	0	0	255
Black	0	0	0	Yellow	255	255	0
Red	255	0	0	Cyan	0	255	255
Green	0	255	0	Magenta	255	0	255

14.1.2 RGBカラー映像の表示出力

ＳＶＰ－３３０ではＲＧＢカラー映像を直接ＮＴＳＣカラーモニタに表示できません。そのため、IP_ConvertRGBtoYUVfast()コマンドを使用してＲＧＢ画像をＹＵＶ画像に変換する必要があります。

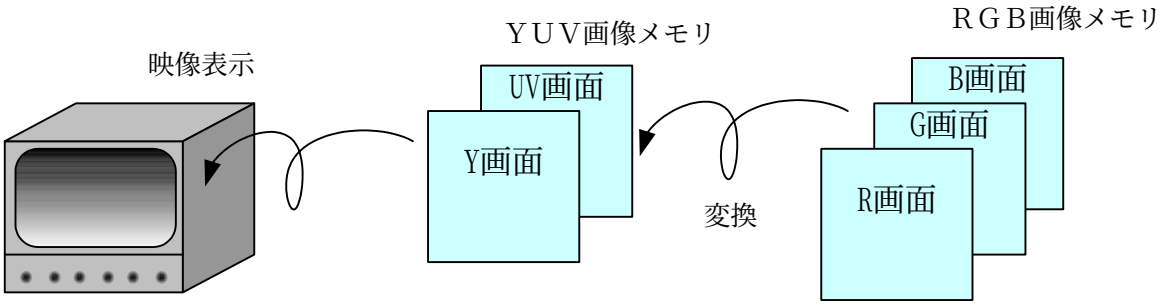


図１４－１ ＲＧＢ画像の表示

以下に、ＲＧＢ映像出力の例を示します。

```
// 画面確保

      // ＲＧＢ映像入力ワーク画面確保
      ImgBW = AllocImg(devID,IMG_FS_512H_512V);
      // ＹＵＶ画面確保
      ImgYUV = AllocYUVImg(devID,IMG_FS_512H_512V);
      // ＲＧＢ表示画面確保
      ImgRGB = AllocRGBImg(devID,IMG_FS_512H_512V);


// ＲＧＢ画像をロード
      LoadBMPFile(devID, ImgRGB, filename);


// ＲＧＢカラー映像表示

      // YUV -> RGB 変換
      IP_ConvertRGBtoYUVfast(devID,ImgRGB,ImgYUV);
      // YUV カラー映像表示
      DispImg(devID,ImgYUV);
```

14.2 RGBカラー抽出処理

14.2.1 RGBカラー抽出

RとG、Bデータから構成されたカラー画像から指定されたR/G/B色の部分を抽出し、２値画像としてモノクロ画面に転送します。

14.2.2 RGBカラー抽出(色相･彩度･明度)

RとG、Bデータから構成されたカラー画像から指定されたH、S、Ｉの成分を抽出し、２値画像としてモノクロ画面に転送します。

○ 色相・彩度・明度について

できるだけ定量的に色を表現するため、色の性質を次の３つの要素に分けて表現することがあります。人間の視感覚に順応した色の表現方法として、色合い／色調(色相)、あざやかさ(彩度)、明るさ(明度)による色の表現方法があります。

・色相（Hue）
　色を円周上に配列していると考え、色あい／色調を角度で表わしたものです。
　Hの範囲は 0～359 です。
・彩度（Saturation）
　色のあざやかさを示します。Sの範囲は 0 ～255 です。
・明度（Intensity）
　色の明るさを示します。Iの範囲は 0 ～255 です。

```
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)
```

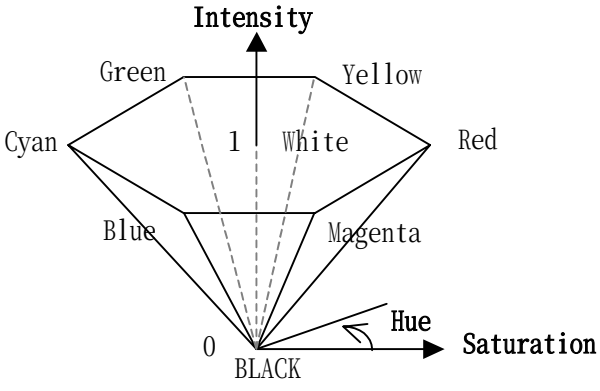
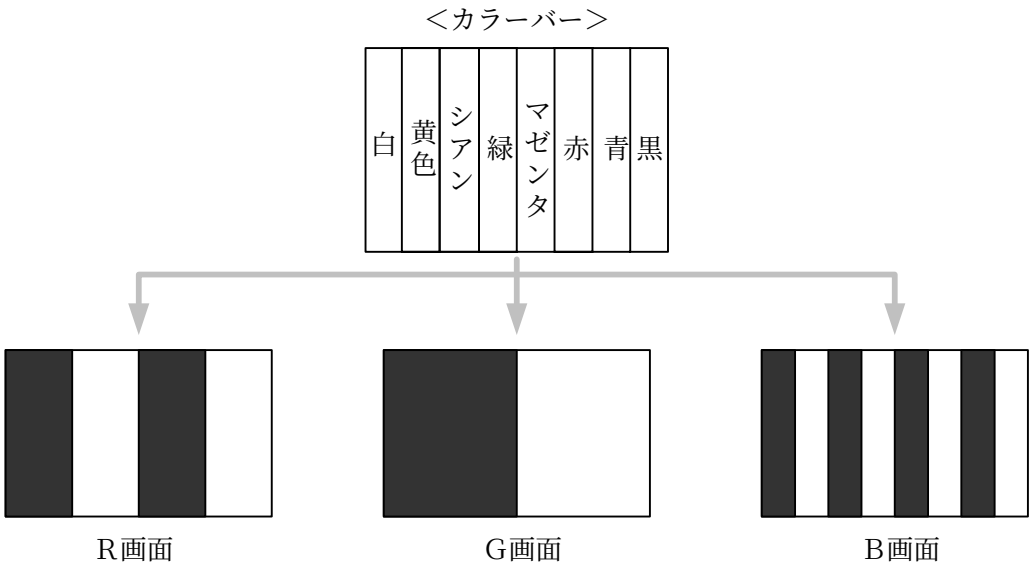


図１４－２　色相(H)、彩度（S)、明度(I)の概念図

14.2.3 サンプルプログラム

カラーバーパターン作成（RGB）

画像メモリに下図のようなカラーバーのパターンを書き込み、表示します。




```
#include   <stdio.h>
#include   <stdlib.h>

#include   "vpxdef.h"
#include   "vpxsys.h"
#include   "vpxfnc.h"

#define    Xsize 512
#define    Ysize 480

void main()
{
   int devID, i, ImgRGB, ImgG, ImgB;
   unsigned char *data1, *data2, *data3;

   /* システムイニシャライズ */
   devID = OpenIPDevExt( IPBOARD_0, NULL );
   InitIP( devID );

   /* 映像表示選択 */
   SelectDisp( devID, YUV_DISPLAY );

   /* 画面確保 */
   ImgRGB=AllocRGBImg( devID,IMG_FS_512H_512V );
   ImgYUV=AllocYUVImg( devID,IMG_FS_512H_512V );

   ImgG = GetGImgID( devID,ImgRGB );    /* G画面画像番号取得 */
   ImgB = GetBImgID( devID,ImgRGB );    /* B画面画像番号取得 */

   data1= (unsigned char *)malloc(sizeof(unsigned char)* Xsize* Ysize);
   data2= (unsigned char *)malloc(sizeof(unsigned char)* Xsize* Ysize);
   data3= (unsigned char *)malloc(sizeof(unsigned char)* Xsize* Ysize);
```

このインクルードファイルは必ずインクルードして下さい

YUV映像表示に設定

カラー映像用画像メモリ領域確保

```
    OpenImg(devID,WAIT_FOREVER);

    /* パターン作成 */
    for(i=0; i<480; i++){
       /* Rパターン作成 */
       memset(data1+ (Xsize*i)    , 0xFF, 128);      /* White  - Yellow */
       memset(data1+ (Xsize*i)+128, 0x00, 128);      /* Cyan   - Green  */
       memset(data1+ (Xsize*i)+256, 0xFF, 128);      /* Magenta- Red    */
       memset(data1+ (Xsize*i)+384, 0x00, 128);      /* Blue   - Black  */

       /* Gパターン作成 */
       memset(data2+ (Xsize*i)    , 0xFF, 256);      /* White  - Green  */
       memset(data2+ (Xsize*i)+256, 0x00, 256);      /* Magenta- Black  */

       /* Bパターン作成 */
       memset(data3+ (Xsize*i)    , 0xFF, 64);       /* White   */
       memset(data3+ (Xsize*i)+ 64, 0x00, 64);       /* Yellow  */
       memset(data3+ (Xsize*i)+128, 0xFF, 64);       /* Cyan    */
       memset(data3+ (Xsize*i)+192, 0x00, 64);       /* Green   */
       memset(data3+ (Xsize*i)+256, 0xFF, 64);       /* Magenta */
       memset(data3+ (Xsize*i)+320, 0x00, 64);       /* Red     */
       memset(data3+ (Xsize*i)+384, 0xFF, 64);       /* Blue    */
       memset(data3+ (Xsize*i)+448, 0x00, 64);       /* Black   */

       SetAllWindow(devID,0,i,511,i);

       /* Rパターン描画 */
       WriteImg(devID,ImgRGB,data1,128*480);

       /* Gパターン描画 */
       WriteImg(devID,ImgG,data2,256*480);

       /* Bパターン描画 */
       WriteImg(devID,ImgB,data3,64*256);
    }

    CloseImg(devID);

    free(data3);
    free(data2);
    free(data1);

    // RGB -> YUV 変換
    IP_ConvertRGBtoYUVfast(devID,ImgRGB,ImgYUV);
    // YUV カラー映像表示
    DispImg(devID,ImgYUV);

}
```

14.2.4　RGBカラー処理コマンド一覧

表１４－２ にＲＧＢカラー処理コマンドの一覧を示します。詳細はコマンドリファレンスを参照して下さい。

表１４－２ ＲＧＢカラー処理コマンド一覧

コマンド名	機能	備考
GetCamera	RGBカメラ映像入力	※１
AttachRGBWorkImg	RGB映像入力ワークフレームの設定	
DetachRGBWorkImg	RGB映像入力ワークフレームの解除	
AllocRGBImg	画像メモリ領域確保(RGB画面)	
ReadRGBImgTable	RGB画像メモリ管理テーブル読み出し	
GetGImgID	G画面番号抽出	
GetBImgID	B画面番号抽出	
IP_ConvertHue	色相変換	
IP_ConvertSaturation	彩度変換	
IP_ConvertIntensity	明度変換	
IP_ConvertRGBtoYUV	カラー疑似変換（RGB →YUV）	
IP_ConvertRGBtoYUVfast	カラー疑似変換（RGB →YUV）	
IP_ExtractColorRGB	RGBカラー抽出処理	
IP_ExtractColorHSI	カラー抽出処理－色相・彩度・明度変換	
OpenMultiColor	マルチカラー抽出処理可能	
CloseMultiColor	マルチカラー処理不可	
ClearMultiColorHSI	HSIマルチカラーテーブルクリア	
SetMultiColorHSI	HSIマルチカラー抽出データ設定	
IP_ExtractMultiColorHSI	HSIマルチカラー抽出	
ClearMultiColorRGB	RGBマルチカラーテーブルクリア	
SetMultiColorRGB	RGBマルチカラー抽出データ設定	
IP_ExtractMultiColorRGB	RGBマルチカラー抽出	

※１　ＳＶＰ－３３０では、ＲＧＢカメラ映像入力には対応していません。

14.3 RGBカラーの画像処理

14.3.1 RGBカラー時の画像メモリ使用方法

前に述べたようにＲＧＢカラー映像データは、ＲとＧ、Ｂの３つのデータで構成されるため、画像メモリは３画面を使用します。

ＲＧＢカラー用の画像メモリを確保するコマンドがAllocRGBImg()です。AllocRGBImg()では、カラー映像用にＲ画面とＧ画面とＢ画面の３画面を確保します。ユーザにはこのうちＲ画面の画面番号を返し、Ｇ、Ｂ用の画面はシステム内部で管理されます。

	確保コマンド	確保例
モノクロ画面	AllocImg ／AllocLockImg	（例）AllocImg （IMG_FS_512H_512V ） 512 512 画面番号 512 ×512 の画面を １画面確保
RGBカラー 画面	AllocRGBImg	（例）AllocRGBImg （IMG_FS_512H_512V ） 512 512 R画面 画面番号 G画面 B画面 512 ×512 の画面を ３画面確保 Ｒ画面の画面番号 をユーザに返す

14.3.2 RGBカラー映像用画像メモリの確保

ＲＧＢ画面は、AllocRGBImg( )コマンドで確保します。AllocRGBImg( )では、Ｒ画面、Ｇ画面、Ｂ画面のモノクロ画面を３画面確保し、一元管理します。

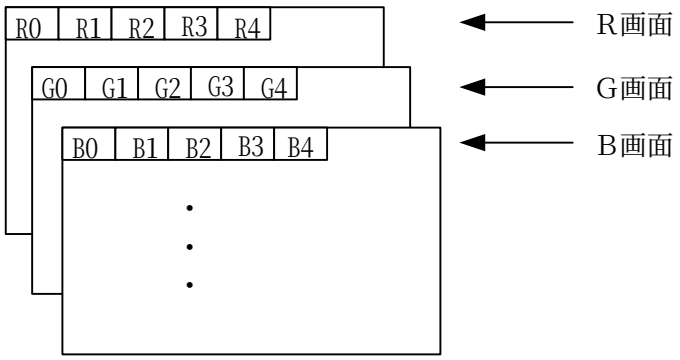


図１４－３ ＲＧＢ映像用画像メモリ

14．3．3　RGBカラー画面の画像処理

AllocRGBImg()で確保したカラー画面番号を使用することで、ＲＧＢカラー画面に対する画像処理ができます。ただし、カラー画面に対して画像処理する場合は、Ｒ画面のみ処理します。

```
[実行例]

ImgRGB1= AllocRGBImg(devID,IMG_FS_512H_512V);
ImgRGB2= AllocRGBImg(devID,IMG_FS_512H_512V);
IP_Zoom(devID,ImgRGB1,ImgRGB2,2.0);
```

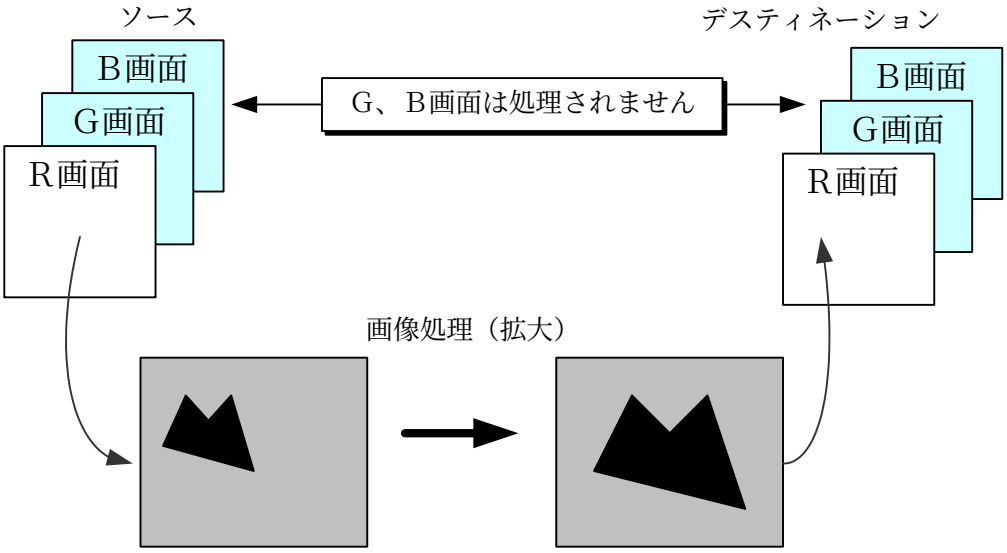
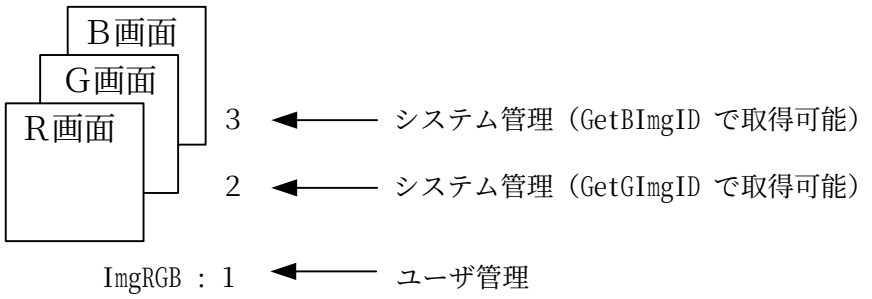


図１４－４　ＲＧＢの画像処理

Ｇ、Ｂ画面に対して処理する場合は、Ｒ画面の画面番号で管理するＲＧＢカラー画面のＧ、Ｂ画面番号を取得し、このＧ、Ｂ画面番号を使用して実行できます。GetGImgID コマンドでカラー画面のＧ画面番号を、GetBImgID コマンドでカラー画面のＢ画面番号を取得できます。




```
[実行例]

   ImgRGB1= AllocRGBImg(devID,IMG_FS_512H_512V);
   ImgRGB2= AllocRGBImg(devID,IMG_FS_512H_512V);

   ImgG1= GetGImgID(devID,ImgRGB1);     /* ImgRGB1 のG画面No.を取得 */
   ImgB1= GetBImgID(devID,ImgRGB1);     /* ImgRGB1 のB画面No.を取得 */
   ImgG2= GetGImgID(devID,ImgRGB2);     /* ImgRGB2 のG画面No.を取得 */
   ImgB2= GetBImgID(devID,ImgRGB2);     /* ImgRGB2 のB画面No.を取得 */

   IP_Zoom(devID,ImgRGB1,ImgRGB2,2.0);
   IP_Zoom(devID,ImgG1,  ImgG2,  2.0);
   IP_Zoom(devID,ImgB1,  ImgB2,  2.0);
```

第15章

フラッシュメモリの活用

15.1 概要

ＳＶＰ－３３０はボード上に４Mバイトのフラッシュメモリを実装しています。フラッシュメモリは、ＳＶＰ－３３０のシステムブートモジュールが含まれていますが、それ以外の部分（約１．５M）は未使用の領域です。画像処理コマンドは、そのフラッシュメモリの未使用領域を使用してデータの保存や読み出しを行うために、簡単なフラッシュメモリディスクの機能を実装し、ファイルとしてのアクセスを可能にしました。

なお、フラッシュメモリは、その物理的な特性により、消去、書き込みできる回数に制限があります。通常は１０万回程度ですので、常に消去、書き込みを繰り返すようなアプリケーションには使用できませんので注意して下さい。

15．2 フラッシュメモリファイル

ＳＶＰ－３３０のフラッシュメモリには、ファイルとしてデータを保存しておくことができます。
なお、ＳＶＰ－３３０のフラッシュメモリファイルは、約７．５Ｍバイトまで作成できます。

15.2.1 フラッシュメモリファイルコマンド一覧

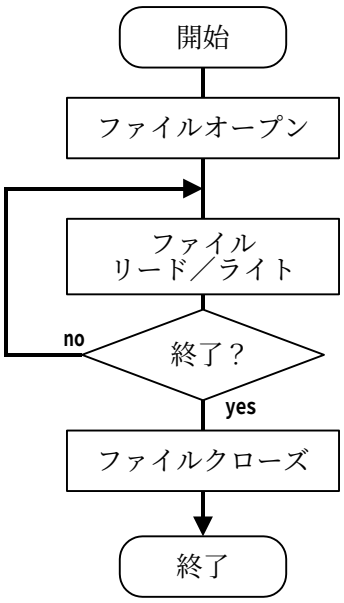
以下にフラッシュメモリコマンドの一覧を示します。詳細は、コマンドリファレンスを参照して下さい。

表１５－１　フラッシュメモリファイルコマンド一覧

コマンド名	機能	備考
fmOpen	ファイルオープン	
fmClose	ファイルクローズ	
fmRead	ファイルからのデータ読み込み	
fmWrite	ファイルへのデータ書き込み	
fmGetFileSize	ファイルサイズ取得	
fmDelete	ファイル削除	
fmRename	ファイル名変更	
fmFileList	ファイルリスト取得	
fmFormat	フラッシュメモリのフォーマット	
fmDiskSize	ディスクサイズ取得	
fmDiskFree	未使用領域サイズの取得	
fmLoardProgram	プログラムダウンロード	
fmFindFile	ファイル検索	
fmChgAttr	ファイル属性の変更	
fmSetAttr	ファイル属性の設定	
fmFileCopy	PC-ﾌﾗｯｼｭﾒﾓﾘ間のファイルコピー	

15．2．2 フラッシュメモリファイルアクセスフロー

以下にフラッシュメモリファイルアクセスのフローを示します。



(注)
フラッシュメモリへの書込は、デバイスの特性上「消去－書込」のサイクルが１０万回程度に制限されています。

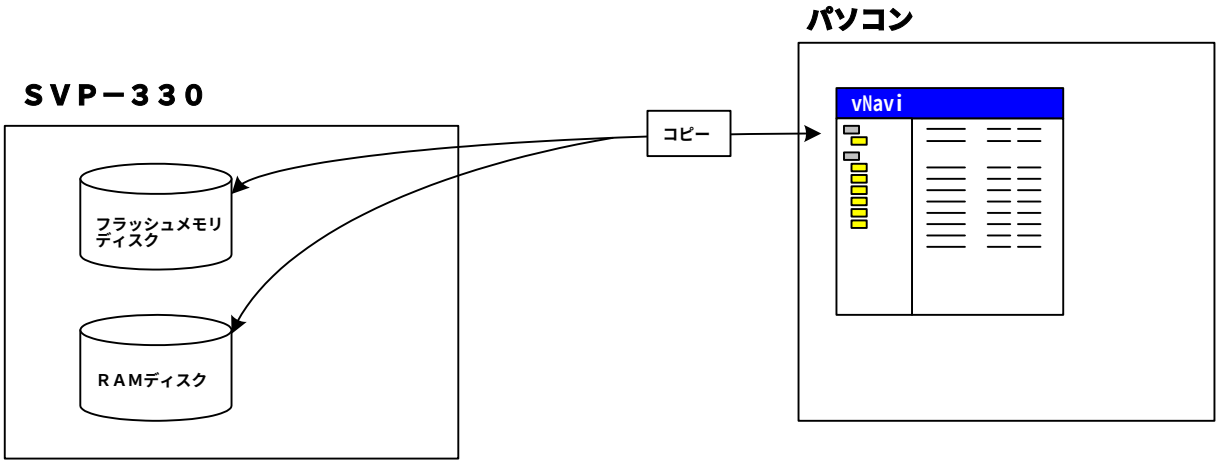
15．2．3 フラッシュメモリファイルアクセスサンプル

以下にフラッシュメモリファイルアクセスのサンプルを示します。

```
int data[128];
FM_HFILE  hFile;

    // 書き込み
    hFile = fmOpen(devID,"sample.dat",FM_DEFAULT);
    if(hFile == 0){
       // エラー
    }
    fmWrite(devID,hFile,data,sizeof(data));
    fmClose(devID,hFile);

    // 読み込み
    hFile = fmOpen(devID,"sample.dat",FM_DEFAULT);
    if(hFile == 0){
       // エラー
    }
    fmRead(devID,hFile,data,sizeof(data));
    fmClose(devID,hFile);
```

15．3 ファイル操作

ＳＶＰ－３３０のフラッシュメモリディスクのファイルをＰＣから操作するためのＧＵＩツール「ｖＮａｖｉ」を用意しました。ｖＮａｖｉを使用することでフラッシュメモリディスクからＰＣに、ＰＣからフラッシュメモリに簡単にファイルのコピーやその他の操作が可能です。
ｖＮａｖｉの操作については、「ｖＮａｖｉ 操作説明書」を参照して下さい。



第16章

IP5000互換コマンドの使用方法

ＳＶＰ－３３０ＳＤＫでは、IP5000互換コマンド（Ｃ言語ソース互換）を用意しています。以下にその概要と使用方法を説明します。

16.1 概要

IP5000互換コマンドでは、アプリケーションのイニシャライズで行っていたＰＣドライバのセットアップ操作が不要になります。

```
devID = OpenIPDevExt( IPBOARD_0, NULL );

InitIP( );

ImgID = AllocImg( IMG_FS_512H_512V );

・・・・・・・・・・・・
```

また、IP5000互換コマンドでは、すべてのコマンドでデバイスＩＤが不要になります。

```
InitIP( devID )                          --->  InitIP( )

AllocImg( devID, IMG_FS_512H_512V )      --->  AllocImg( IMG_FS_512H_512V )
```

16.1.1 プリプロセッサ定義

IP5000コマンドを使用する場合は、コンパイラのプリプロセッサ定義で

```
STD_LIB
```

を定義するか又は、
Ｃ言語ソースファイル上のヘッダファイルインクルード宣言の前で

```
#define   STD_LIB
```

を定義して下さい。

16.1.2 インクルードファイル

IP5000と同様に

```
#include  "ipxdef.h"
#include  "ipxsys.h"
#include  "ipxprot.h"
```

をインクルードして下さい。このとき、インクルードファイルのパスは、

```
C:\VP330SDK\PcDrv\include
```

に設定して下さい。

16.1.3 リンクライブラリ

```
C:\VP330SDK\PcDrv\lib\VP900STD.LIB
```

をリンクして下さい。

16.1.4 ＳＨモジュール化について

IP5000互換コマンドでＰＣ用に作成したプログラムをＳＶＰ－３３０コマンドと同様にＳＨモジュール化できます。プリプロセッサ定義で「STD_LIB」を指定するだけで、それ以外はＳＶＰ－３３０コマンドと同じ方法でＳＨモジュール化できます。

16.2 注意事項

16.2.1 InitIP( )コマンドについて

IP5000のInitIP( ),InitIPExt( )コマンドは、512×440に設定し、デフォルトのウィンドウサイズも512×440に設定します。ところが、ＳＶＰ－３３０のInitIP( ),InitIPExt( )コマンドは、ビデオサイズを512×480に設定し、デフォルトのウィンドウサイズも512×480に設定します。そこで、IP5000と同じビデオサイズで初期化する場合は、InitIP( )及びInitIPExt( )コマンドを以下のようにユーザプログラムで変更して下さい。

```
InitIP( )     --->  vpxInitIP(IP5K_CONFIG,0,INIT_ONLY)

InitIPExt( )  --->  vpxInitIP(IP5K_CONFIG,0,SELF_TEST)
```

16.2.2 カメラ選択について

ＳＶＰ－３３０ではＮＴＳＣモノクロカメラやＣＶＢＳカメラを使用することが可能です。また、IP5000のカメラデファインは、

① BW_CAMERA
② YUV_CAMERA

以外は使用できません。詳細は、コマンドリファレンスのSelectCamera()コマンドを参照して下さい。

16.2.3 SVP-330のオリジナルコマンドについて

ＩＰ５０００のコマンド以外にＳＶＰ－３３０のオリジナルコマンドも使用可能です。その場合は、ＳＶＰ－３３０コマンドからデバイスＩＤを削除したコマンド形式になります。また、それとは逆に、ＩＰ５０００のコマンドをデバイスＩＤを付加した形式でＳＶＰ－３３０のコマンドとしても使用可能です。

第17章

イーサネットでの通信

17.1 概要

イーサネットの通信はＵＮＩＸのＢＳＤソケットのイメージでプログラミングが可能です。
なお、ＢＳＤソケットコマンドを使用する場合は、"VPXSOCK.H"をインクルードして下さい。

17.1.1 ソケットインタフェース

ＢＳＤソケットモデルは、ネットワーク通信を２つの端点（ソケット）間で発生するものとして概念化しています。電化製品を使用する場合に電源ケーブルを電源ソケットに接続することと同じように、アプリケーションをソケットを介してネットワークにつなぐということです。１つのソケット(socket)は、２つのプロセス間で起きる通信の一方の端点を定義します。通信を開始するには、クライアントソケットとサーバソケットという２つのソケットが存在する必要があります。

17.1.2 クライアント／サーバーモデル

ＢＳＤソケットモデルは、通信に対してクライアント／サーバー方式のアプローチをとります。クライアントとサーバーの２つのネットワークアプリケーションは同時に起動するのではなく、サーバー側のアプリケーションは常に使用可能な状態に置き、クライアント側のアプリケーションは必要に応じてサービスをサーバー側のアプリケーションに要求するようにします。サーバーはソケットを作成し、それに名前を付けてクライアントが識別できるようにします。その後、サービスの要求を待機します。クライアントアプリケーションは、ソケットを作成し、サーバーソケットをアドレスで識別してからプラグインして対話を開始します。いったん対話を始めた後は、データを双方向に送信することができます。

17.1.3 クライアント／サーバーアプリケーション

クライアント／サーバーアプリケーションは、基本的に接続型アプリケーションと非接続型アプリケーションの２種類に大別されます。アプリケーションが接続型と非接続型のどちらに分類されるかは、通常はアプリケーションが通信に使用するプロトコルによって決まります。

17.1.4 接続型アプリケーション

データをバイトストリームとして送受信するアプリケーションは、一般に接続型アプリケーションです。データを転送したり受信したりするには、まず２つのプロセス間で接続を確立しなければなりません。各端点(ソケット)の識別情報は、接続の開始時に一度だけ確認されこれ以降のデータの送受信は、その接続の２点の間で行うものとみなされます。そのため、各アプリケーションは、データ転送のたびに識別情報を確認する必要がありません。接続型アプリケーションは通常ＴＣＰ(Transmission Control Protocol)で送受信を行うアプリケーションを指します。

まずサーバーアプリケーションが起動し、socket()を使ってソケットを作成し、bind()でソケットに名前を付けます。ソケットに名前を付けることにより、サーバーのアドレス、接続先のポート、使用するプロトコルを特定することができます。ソケットに名前を付けたら、サーバーはlisten()を使ってクライアントからの要求を待機し、accept()でそれを受け入れます。

これでクライアントアプリケーションを実行する準備が整ったことになります。クライアントアプリケーションはsocket()を使ってソケットを作成し、connect()を使ってサーバーにプラグインし、サービス要求を送ります。この時点で両アプリケーションの識別情報が確認されて仮想回線が確立されます。両方のアプリケーションでは、少なくとも１つのソケットをお互いの通信専用に使用する必要があります。この後は、send()とrecv()を使ってどちらの方向にもデータを送受信することができます。アプリケーション間の通信が終わったら両アプリケーションでshutdown()とclosesocket()を使ってそれぞれのソケットを閉じます。

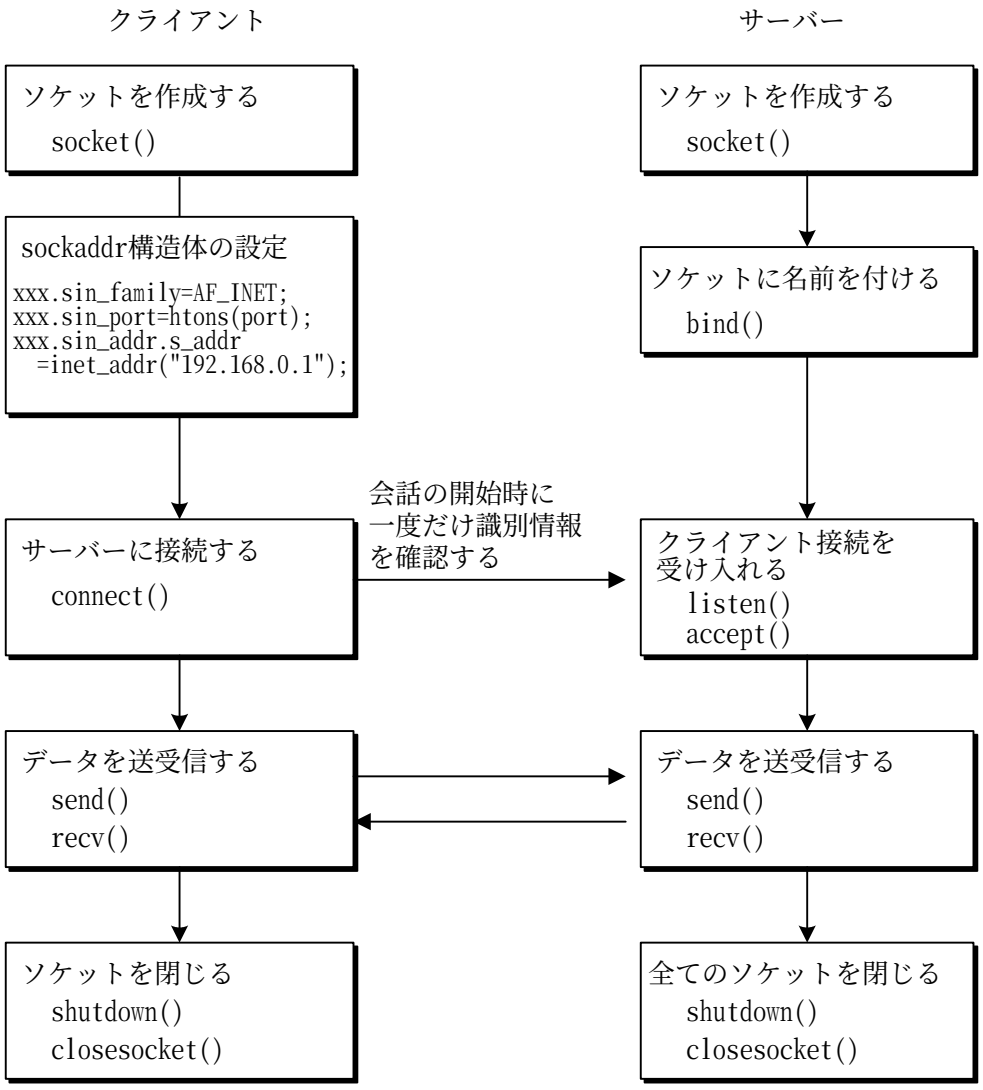


図１７－１ 接続型アプリケーション

サンプル１、２にＴＣＰ接続でのサーバー側とクライアント側のプログラム例を示します。

サンプル１ ： ＴＣＰでのサーバー側のプログラム例

```
#include <string.h>
#include "vpxdef.h"
#include "vpxsys.h"
#include "vpxfnc.h"
#include "vpxcnv.h"
#include "vpxsock.h"

typedef int SOCKET;
#define INADDR_NONE      0
#define SOCKET_ERROR     -1

#define   BUFSIZE         1000
#define   DEFAULT_PORT    5320

void TCPServer()
{
struct sockaddr_in client, server;
int             port, len, i, n, j, k;
unsigned char   buf[BUFSIZE];
SOCKET          wait_sock,sock;

   port = DEFAULT_PORT;

   if((wait_sock = socket(AF_INET, SOCK_STREAM, 0)) < 0){
      printf("socket error !!\n");
      return;
   }

   // 受付クライアントの設定
    memset((char *)&server, 0x00, sizeof(server));
   server.sin_family      = AF_INET;
   server.sin_addr.s_addr = htonl(INADDR_ANY);
   server.sin_port        = htons((unsigned short)port);

   // バインド
   if(bind(wait_sock, (struct sockaddr *)&server, sizeof(server)) < 0){
      printf("bind error !!\n");
      closesocket(wait_sock);
      return;
   }

   // 受付けキューの設定
   if(listen(wait_sock, 1) < 0){
      printf("listen error !!\n");
      closesocket(wait_sock);
      return;
   }

   // TCPコネクションの受付け
   len = sizeof(client);
   if((sock = accept(wait_sock, (struct sockaddr *)&client, (socklen_t*)&len)) < 0){
      printf("accept error !!\n");
      closesocket(wait_sock);
      return;
   }

   // TCPを利用したデータの送受信
   while((n = recv(sock, buf, BUFSIZE-1, 0)) > 0){
      for(i = 0;i < n;) {
         printf("Receive [ ");
         for (j = 0 ; i < n && j < 10 ; i++ ,j++) {
            printf("H'%02X ", buf[i]);
         }
         printf("]\n");
      }

      k= 0;
      for(i = 0;i < n;) {
         printf("Send    [ ");
         for (j = 0 ; i < n && j < 10 ; i++ ,j++) {
            printf("H'%02X ", buf[i]);
         }
         printf("]\n");
      }

      send(sock, buf, n, 0);
   }

   shutdown(sock,SHUT_RDWR);
   closesocket(sock);
   closesocket(wait_sock);
}
```

サンプル２ ： ＴＣＰでのクライアント側のプログラム例

```
#include <stdio.h>
#include "vpxdef.h"
#include "vpxsys.h"
#include "vpxfnc.h"
#include "vpxcnv.h"

#include "vpxsock.h"

typedef int SOCKET;
#define INADDR_NONE      0
#define SOCKET_ERROR     -1

#define  DEFAULT_PORT    5320

void TCPClient()
{
struct sockaddr_in svr;
int             ret;
char server[] = "192.168.0.3";
unsigned long   ipaddr;
unsigned short  port= DEFAULT_PORT;
SOCKET sock;
char op_msg[] = "Connect TCPServer OK !!\r\nYou can communicate with TCPServer ...\r\n";

   // サーバーのIPアドレス
   ipaddr= inet_addr(server);
   if(ipaddr == INADDR_NONE){
      printf("inet_addr error !!\n");
      return;
   }

   // ソケットの作成
   sock= socket(AF_INET,SOCK_STREAM,0);
   if(sock == SOCKET_ERROR){
      printf("socket error !!\n");
      return;
   }

   // サーバーの設定
   memset((char*)&svr, 0x00, sizeof(svr));
   svr.sin_family      = AF_INET;
   svr.sin_port        = htons(port);
   svr.sin_addr.s_addr = ipaddr;

   // サーバーに接続
   ret= connect(sock,(struct sockaddr*)&svr,sizeof(svr));
   if(ret<0){
      printf("connect error !!\n");
      closesocket(sock);
      return;
   }

   ret= send(sock, op_msg, strlen(op_msg), 0);

   closesocket(sock);

}
```

17.1.5 非接続型アプリケーション

データをデータグラムの形で送受信するアプリケーションは一般に非接続型アプリケーションです。アプリケーションは、データ送信のたびに識別情報を確認します。非接続型アプリケーションは通常ＵＤＰ(User Datagram Protocol)で送受信を行うアプリケーションを指します。

まずサーバーアプリケーションが起動し、socket()を使ってソケットを作成し、bind()でソケットに名前を付け、クライアントから要求が送られてくるのを待機します。この種のサーバーは一般にlisten()やaccept()を使わず、recvfrom()を呼び出すことによってクライアント要求を待機します。recvfrom()は接続型サーバーで使うrecv()の持つ機能に加えて、サーバーにデータの送信元を知らせる役割も果たします。サーバーはこの情報を元にsendto()を使ってクライアントに応答を返します。アプリケーション間の通信が終わったら両アプリケーションでclosesocket()を使ってそれぞれのソケットを閉じます。

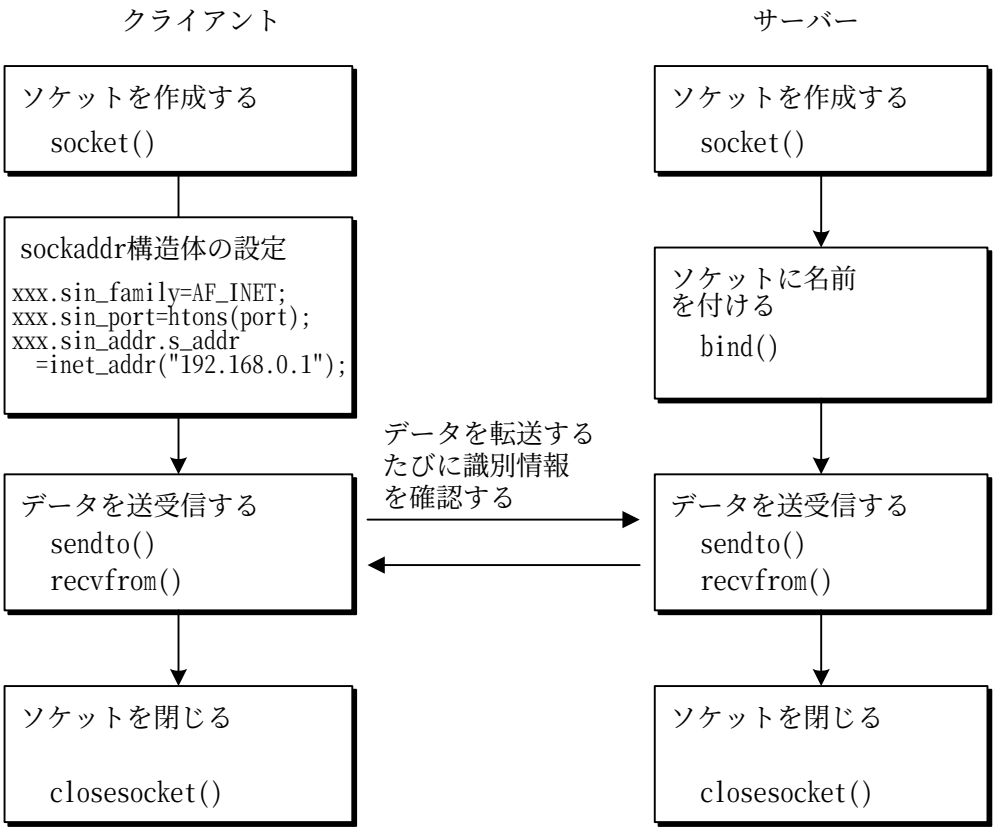


図１７－２ 非接続型アプリケーション

サンプル３、４にＵＤＰ接続でのサーバー側とクライアント側のプログラム例を示します。

サンプル３ ： ＵＤＰでのサーバー側のプログラム例

```
#include <stdio.h>
#include "vpxdef.h"
#include "vpxsys.h"
#include "vpxfnc.h"
#include "vpxcnv.h"

#include "vpxsock.h"

typedef int SOCKET;
#define INADDR_NONE      0
#define SOCKET_ERROR     -1

#define BUFSIZE          1000
#define DEFAULT_PORT     5320
#define END_CODE         0

void UDPServer()
{
struct sockaddr_in client, server;
int             port, len, i, n, j, k;
unsigned char   buf[BUFSIZE];
SOCKET          sock;

   port = DEFAULT_PORT;

   if((sock = socket(AF_INET, SOCK_DGRAM, 0)) < 0){
      printf("socket error !!\n");
      return;
   }

   // 受付クライアントの設定
   memset((char *)&server, 0x00, sizeof(server));
   server.sin_family      = AF_INET;
   server.sin_addr.s_addr = htonl(INADDR_ANY);
   server.sin_port        = htons((unsigned short)port);

   // バインド
   if(bind(sock, (struct sockaddr *)&server, sizeof(server)) < 0){
      printf("bind error !!\n");
      closesocket(sock);
      return;
   }

   // UDPを利用したデータの送受信
   while(1){
      len= sizeof(client);
      n = recvfrom(sock, buf, BUFSIZE-1, 0, (struct sockaddr *)&client, (socklen_t*)&len);
      if(n <= 0)
         break;
      for(i = 0;i < n;) {
         printf("Receive [ ");
         for (j = 0 ; i < n && j < 10 ; i++ ,j++) {
            if(buf[i] == END_CODE)
               goto CLOSE_SOKET;
            printf("H'%02X ", buf[i]);
         }
         printf("]\n");
      }

      k= 0;
      for(i = 0;i < n;) {
         printf("Send    [ ");
         for (j = 0 ; i < n && j < 10 ; i++ ,j++) {
            printf("H'%02X ", buf[i]);
         }
         printf("]\n");
      }

      sendto(sock, buf, n, 0, (struct sockaddr *)&client, sizeof(client));
   }

CLOSE_SOKET:

   closesocket(sock);

}
```

サンプル４ ： ＵＤＰでのクライアント側のプログラム例

```
#include <string.h>
#include "vpxdef.h"
#include "vpxsys.h"
#include "vpxfnc.h"
#include "vpxcnv.h"

#include "vpxsock.h"

typedef int SOCKET;
#define INADDR_NONE      0
#define SOCKET_ERROR     -1

#define DEFAULT_PORT     5320

void UDPClient()
{
struct sockaddr_in svr;
int             ret;
char server[] = "192.168.0.3";
unsigned long   ipaddr;
unsigned short  port= DEFAULT_PORT;
SOCKET sock;
char op_msg[] = "Connect UDPServer OK !!\r\nYou can communicate with UDPServer ...\r\n";
char buf[4]= {0};

   // サーバーのIPアドレス
   ipaddr= inet_addr(server);
   if(ipaddr == INADDR_NONE){
      printf("inet_addr error !!\n");
      return;
   }

   // ソケットの作成
   sock= socket(AF_INET,SOCK_DGRAM,0);
   if(sock == SOCKET_ERROR){
      printf("socket error !!\n");
      return;
   }

   // サーバーの設定
   memset((char*)&svr, 0x00, sizeof(svr));
   svr.sin_family      = AF_INET;
   svr.sin_port        = htons(port);
   svr.sin_addr.s_addr = ipaddr;

   sendto(sock, op_msg, strlen(op_msg), 0, (struct sockaddr*)&svr, sizeof(svr));
   sendto(sock, buf, 1, 0, (struct sockaddr*)&svr, sizeof(svr));

   closesocket(sock);

}
```

17.1.6 イーサネット通信の環境設定

ＳＶＰ－３３０のイーサネット通信を使用する前にあらかじめユーザーが使用する環境にあわせて環境設定値を設定する必要があります。システム起動時に「BOOT.INI」ファイルに記述されているイーサネット通信の環境設定値により、通信に必要なメモリの確保やその他の環境変数を設定しイーサネット通信のプロトコルスタックを起動します。以下の環境設定値の変更は、環境設定ガイドの「システムブートファイルの設定」を参照して下さい。

表１７－１　イーサネット通信環境設定

セクション名	内容
IPADR	ＩＰアドレスを指定します。 デフォルトは、”１９２．１６８．０．２００”が設定されます。
IPMSK	サブネットマスクを指定します。 デフォルトは、”０．０．０．０”が設定されます。
GWADR	デフォルトゲートウェイを指定します。 デフォルトは、”０．０．０．０”が設定されます。
PORTSTR	自動割り当てにしようするポート番号の先頭を指定します。使用可能なポート番号は、１～６５５３５ です。デフォルトは、”１５０００”が設定されます。
PORTCNT	自動割り当てにしようするポート番号の数を指定します。使用可能なポート番号は、１～６５５３５ です。デフォルトは、”１６”が設定されます。
TCPACT	ＴＣＰで接続するソケットの総数を指定します。設定範囲は、０ ～ ３２です。 但し、(TCPACT+TCPCON) <= 32に設定して下さい。デフォルトは、”８”が設定されます。
TCPLST	ＴＣＰで接続待ちに使用するソケットの総数を指定します。設定範囲は、 ０ ～ ３２ です。デフォルトは、”８”が設定されます。
UDPCNT	ＵＤＰで接続するソケットの総数を指定します。設定範囲は、０ ～ ３２です。 デフォルトは、”４”が設定されます。
TCPCON	ＴＣＰで同時に受け入れ可能なコネクション要求の総数を指定します。設定範囲は、０ ～ ３２ です。但し、(TCPACT+TCPCON) <= 32に設定して下さい。デフォルトは、”８”が設定されます。
RCVWB	ＴＣＰの受信バッファのデフォルト長を指定します。１つあたりのＴＣＰソケットで使用する受信バッファの最大長です。設定範囲は、２０４８ ～ ８１９２です。 デフォルトは、”４０９６”が設定されます。
SNDWB	ＴＣＰの送信バッファのデフォルト長を指定します。１つあたりのＴＣＰソケットで使用する送信バッファの最大長です。設定範囲は、２０４８ ～ ８１９２です。 デフォルトは、”４０９６”が設定されます。 なお、各ＴＣＰソケットの受信バッファ、送信バッファの長さはsetsockopt()により指定することが可能です。このとき、[RCVWB]セクション、[SNDWB]セクションで指定した長さを超える指定はできません。
UDPDGM	ＵＤＰの最大受信データグラム長を指定します。ＵＤＰソケットが受信するデータグラムの最大長（通常は５７６から１４７２）を指定します。設定範囲は、０ ～ １４７２ です。デフォルトは、”１４７２”が設定されます。
UDPQUE	ＵＤＰの受信バッファキューのデフォルト数を指定します。１つあたりのＵＤＰソケットで用意するデータグラム受信キューのデフォルトの数（通常はrecvまたはrecvfromによって取得する前に受信できるデータグラムの数＋１）を指定します。設定範囲は、０ ～ ２５５ です。デフォルトは、”５”が設定されます。

注）画像処理コマンドサーバのポート番号がデフォルトで”３００００”に設定されています。
ユーザーでこのポート番号を使用する場合は「PORT」を変更して下さい。

17.1.7 ＢＳＤソケットコマンド一覧

以下にＢＳＤソケットコマンドの一覧を示します。詳細は、コマンドリファレンスを参照して下さい。

表１７－２　ＢＳＤソケットコマンド一覧

コマンド名	機能	備考
socket	ソケット生成	
bind	通信アドレス情報の指定	
listen	受動モード設定	
accept	ソケットに対するコネクションの受入れ	
connect	ソケットの接続の開始	
getpeername	相手側通信アドレス情報取得	
getsockname	自通信アドレス情報取得	
recv	ソケットからの受信データ取得	
recvform	受信データと送信者アドレス取得	
send	ソケットへの送信データ設定	
sendto	送信データと送信先アドレス設定	
closesocket	ソケットのクローズ	
shutdown	コネクション閉鎖	
getsockopt	プロトコルのオプション取得	
setsockopt	プロトコルのオプション変更	
selectsocket	ソケットの利用可能待ち	
set_blocking_socket	ブロッキングモード設定	
get_errno	エラーコード取得	
get_thread_errno	スレッドエラーコード取得	
set_sock_timewait	TIMEWAIT 時間の変更	
get_sock_recvlen	受信データ長の取得	
set_sock_keepalive	キープアライブ制御情報の変更	

第18章

ユニット内アプリケーションの開発

ＳＶＰ－３３０でのアプリケーションの作成方法及び実行方法について説明します。

18.1 概要

ＳＶＰ－３３０はｓｈｅｌｌ（コマンド解釈して実行するプログラム）を実装しており、アプリケーションの主記憶へのロードや実行をｓｈｅｌｌのコマンドで実行します。また、ＳＶＰ－３３０では、フラッシュメモリディスクやＲＡＭディスクマネージャを実装しており、アプリケーションの実行ファイルをそれらのディスクに書き込み、ｓｈｅｌｌの実行コマンドで実行します。現在、ＳＶＰ－３３０の実行ファイルとして、ＥＬＦ形式のＡｂｓｏｌｕｔｅファイル（．ａｂｓ）のみサポートしています。

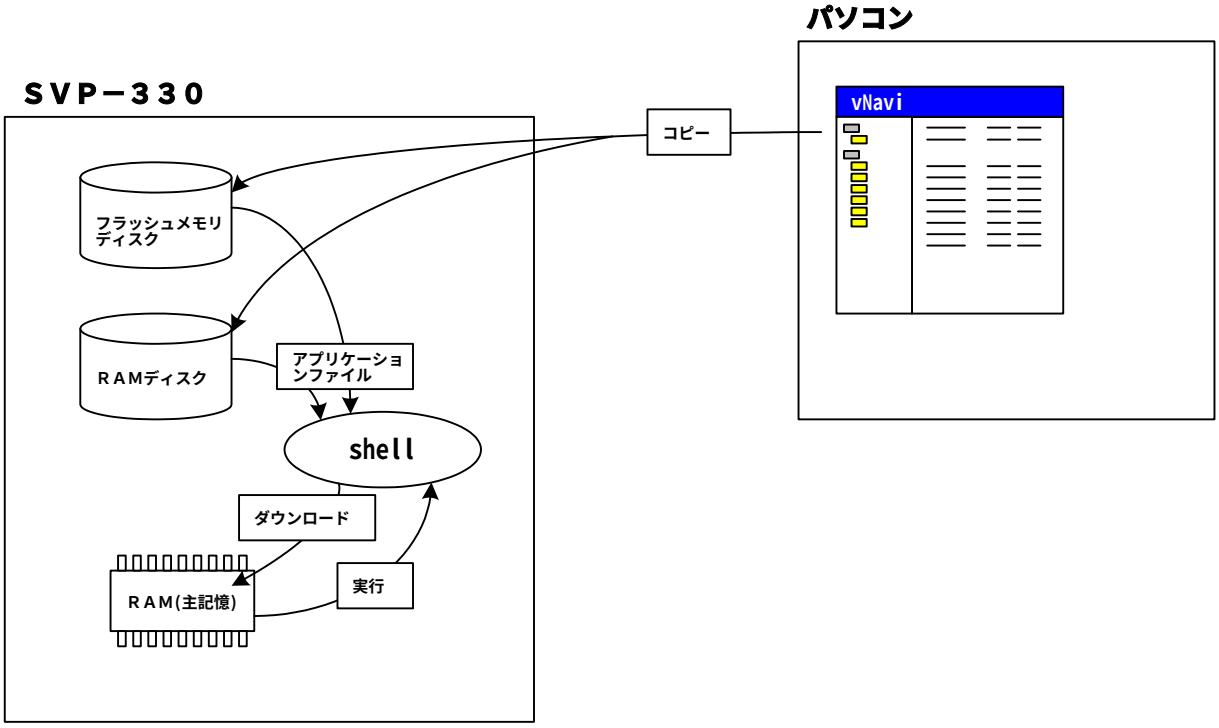


図１８－１ アプリケーション開発

ＳＶＰ－３３０では、アプリケーションを実行する際にフラッシュメモリディスクやＲＡＭディスクにアプリケーションの実行ファイルを書き込むことが必要です。ＳＶＰ－３３０ではＧＵＩツール「vNavi」を使用し、パソコンのディスクからフラッシュメモリディスクやＲＡＭディスクにアプリケーションの実行ファイルを簡単にコピーすることができます。

18.2 アプリケーションの動作

18.2.1 メイン関数と終了関数

アプリケーションは必ず、メイン関数と終了関数を持ちます。それぞれの関数仕様は、以下のように決められています。アプリケーションの実行マネージャーは、アプリケーションの実行時、アプリケーションの実行ファイルの以下の関数エントリを検索し、タスクをクリエイトしてMainエントリアドレスから実行を開始します。そして、Main関数からのリターンでTerminate関数を実行し、アプリケーションのタスクを終了、削除します。従って、実行ファイルに以下の関数エントリが無い場合は、実行できません。

```
メイン関数      :      void Main( int argc, char *argv[] )
終了関数        :      void Terminate( void )
```

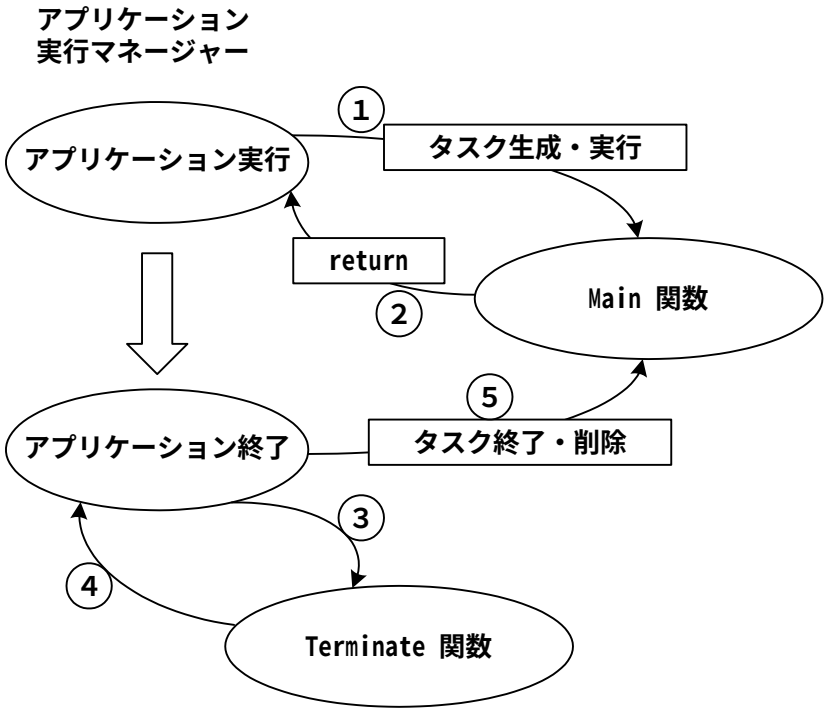


図１８－２ 実行マネージャー

18.2.2 メイン(Main)関数

アプリケーション実行マネージャーは、アプリケーション実行ファイルからMain関数のエントリを検索し、タスク生成・実行を行います。そのとき、Main関数の引数 argc, argv
には、ｓｈｅｌｌのコマンドラインで指定したパラメータが渡されます。このMain関数からのリターンはアプリケーションの自立的な終了となり、アプリケーション実行マネージャーは、アプリケーションタスクの終了と削除を行います。

18.2.3 終了(Terminate)関数

Terminate関数は、Main関数のリターン時、アプリケーション実行マネージャーからコールされます。Terminate関数終了後アプリケーション実行マネージャーは、アプリケーションタスクの終了と削除を行います。Terminate関数には、アプリケーションが終了する場合に特に必要な処理を行います。Terminate関数は、Terminate時のデッドロックを防ぐため、タイムアウト（３秒）を設けています。特に必要の無い場合は、空の関数にして下さい。

18.3 アプリケーションの作成

18.3.1 プログラム

アプリケーションは、少なくともメイン(Main)関数と終了(Terminate)関数を持つ必要があります。
また、実行ファイルがＡｂｓｏｌｕｔｅ（実アドレス）ファイルなので、プログラム領域を限定するためにセクションの指定が必要です。

セクション指定

```
#pragma   section   _main
```

メイン関数

```
void Main(int argc, char *argv[])
{


   ...........
   ...........
}
```

終了関数

```
void Terminate(void)
{
   ...........
   ...........
}
```

18.3.2 インクルードファイル

ＰＣコマンドの場合と同じように、

vpxdef.h,vpxsys.h,vpxfnc.h

をインクルードします。それ以外に

vpxcnv.h

をインクルードしてください。

18.3.3 メモリマップ

オンボードＣＰＵシステムメモリでユーザーが使用できるメモリ空間は

0x8C80:0000 ～ 0x8CFF:FFFF

の８Ｍバイトです。この領域にプログラム、スタティックデータなどをマッピングするように、リンク時に指定して下さい。また、この領域内にmalloc()等のヒープ領域やＲＡＭディスク領域も含んでいますので、「プログラム領域＋ヒープ等の領域」が８Ｍバイトを超えないようにマッピングして下さい。

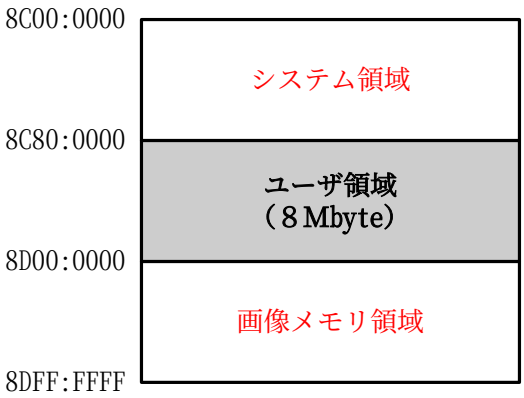


図１８－３ ＲＡＭメモリマップ

18.3.4 プログラムローダー

上記のユーザー領域(0x8C80:0000 ～ 0x8CFF:FFFF)は、システムにより管理されています。プログラムのローダーは、実行ファイル(.abs)からプログラムがロードされる領域を計算し、その領域をユーザ領域から確保し、その領域に実行コードをダウンロードします。実行コードがダウンロードされる領域がすでに確保されている場合、ローダーは実行ファイルの実行コードをダウンロードできません。その場合は、すでに実行しているアプリケーションを終了するか、アドレスのマッピングを変更してください。また、この領域内にmalloc()等のヒープ領域やＲＡＭディスク領域も含んでいます。特に、ＲＡＭディスクを確保した場合は、0x8C80:0000番地からマッピングできない場合がありますので、注意してください。

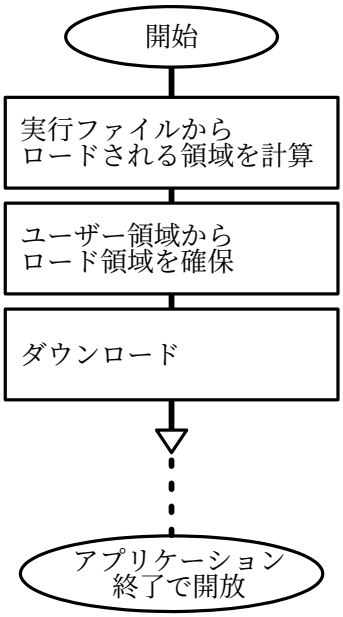


図１８－４ ロードフロー

18.3.5 スタック

アプリケーションで使用できるスタック領域は、デフォルトで６４Ｋバイトです。変更する場合は、ｓｈｅｌｌのコマンドラインからアプリケーション実行オプション「#s」で必要な容量を指定して下さい。

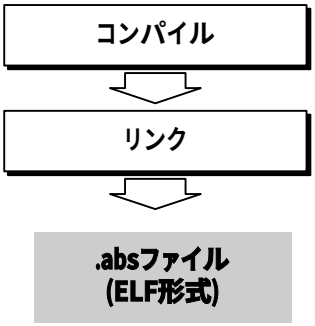
スタックオプション ： #s<スタックサイズ(16進数)>

例　スタックサイズを２５６Ｋバイトにする場合

```
sample.abs #s40000
```

18.3.6 プロジェクトのビルド

ＳＨ４のコンパイラでコンパイル、リンクを行いＥＬＦ形式の実行ファイル(.abs)をビルドします。
なお、ＳＨ４コンパイラのプロジェクトの設定等の詳細は、「ＳＨＣＣｏｍｐｉｌｅｒ設定ガイド」を参照して下さい。



18.3.7 デバッグ

オンボードＣＰＵの割込起動モジュールの開発環境や開発手法については第２章で説明していますので、まず、そちらを参照してください。

オンボードＣＰＵの割込起動モジュールのデバッグは、ＪＴＡＧ－ＩＣＥを使用して行う方法もありますが、ＰＣ側である程度動作しているモジュールであれば、その必要性は低いと思われます。しかし、データの確認などの簡単なデバッグ作業は、プログラム開発上できるにこしたことはありません。そこで画像処理コマンドでは、オンボードＣＰＵ側にＲＳ２３２Ｃのターミナルを対象にしたprintf()やscanf()などの標準入出力を実装しています。ＳＶＰ－３３０で使用できる標準入出力コマンドは、以下に示します。

表１８－１　オンボードＣＰＵ標準入出力関数

関数名	内容	
printf	Ｃの標準関数のprintfと同じです。	
scanf	Ｃの標準関数のscanfと同じです。	
charget	Ｃの標準関数のgetcharと同じです。	
charput	Ｃの標準関数のputcharと同じです。	
malloc	Ｃの標準関数のmallocと同じです。	
free	Ｃの標準関数のfreeと同じです。	

⚠ これらの標準入出力関数を使用する場合は、インクルードファイル「vpxcnv.h,stdio.h,stdlib.h」をＣのソースファイルでインクルードして下さい。

関数の詳細は、コマンドリファレンスのオンボードデバッグサポートコマンドを参照して下さい。

シリアルポートの初期設定を以下に示します。これ以外の設定が必要な場合、InitSCI()コマンドで設定して下さい。

表１８－２　シリアルポートの初期設定

項目	初期値	備考
ビットレート	９６００ｂｐｓ	
キャラクタ長	８ｂｉｔ	
パリティ	パリティなし	
ストップビット	１ｂｉｔ	

18.4 アプリケーションの実行

18.4.1 スタートアップファイル

ＳＶＰ－３３０は、システムが立ち上ると、最初にスタートアップファイル(startup.bat)実行のためのｓｈｅｌｌが起動し、スタートアップファイルに記述されているｓｈｅｌｌコマンドやアプリケーションを実行します。電源投入時又は、リセット時にユーザーアプリケーションを実行する場合は、このスタートアップファイルにコマンドを記述します。スタートアップファイルは、「startup.bat」のファイル名のテキストファイルでフラッシュメモリの「fmd:」ドライブのルートに書き込まれてある必要があります。他のドライブやディレクトリに存在してもサーチしませんので注意してください。

記述 例１

```
shell↵
```

cns1(SCI1)からのｓｈｅｌｌを起動します

記述 例２

```
sample.abs↵
```

sample.abs を実行します

※改行は必ず入れてください。

※ボード出荷時、startup.bat は書き込まれていません。

※スタートアップファイルでアプリケーションが起動される場合、アプリケーションは、システムがイーサネット通信の初期化が終了する前にアプリケーションが起動されます。そのため、アプリケーションでイーサネット通信を行っている場合、イーサネット通信がセットアップされるまで、通信の接続をウェイトして下さい（通常１０～２０秒）。

※startup.bat は『vNavi』で編集可能です。

18.4.2 shell からの実行

スタートアップファイル(startup.bat)にＳＣＩ１からのｓｈｅｌｌを起動（記述例１）するとＳＣＩ１から対話形式でコマンドを実行できるようになります。
起動時、「fmd:\>」のプロンプトが出力され、コマンド入力待ち状態になります。

```
fmd:\>
```

ｓｈｅｌｌの詳細は、『ｓｈｅｌｌコマンドリファレンス』を参照して下さい。

18.5 アプリケーションサンプル

電源投入やリセットでアプリケーションを起動する場合のユニット内アプリケーションのサンプルを以下に示します。

18.5.1 基本的なアプリケーション

以下に、基本的なユニット内ユーザーアプリケーションを場合のサンプルを示します。

```
#include <stdio.h>
#include "vpxdef.h"
#include "vpxsys.h"
#include "vpxfnc.h"
#include "vpxcnv.h"

#pragma section _main
void Main(int argc,char *argv[])
{
    int imgID[2];
    HISTANATBL HistAna;
    IPGOFeatureTbl RegTbl;

    printf("MAIN PROGRAM START !!\n");

    /*******************************
       画像処理システムの初期化
    *******************************/
    InitIP();

    /************************************
       カメラ番号とカメラタイプの選択
    ************************************/
    SelectCamera(CAMERA_PORT0, BW_CAMERA);

    /***************************
       画像メモリ領域の確保
    ***************************/
    imgID[0] = AllocImg(IMG_FS_512H_512V);
    imgID[1] = AllocImg(IMG_FS_512H_512V);

    while(1){
        /***************
           映像入力
        ***************/
        GetCamera(imgID[0]);

        /***************
           画像処理
        ***************/
        IP_Histogram(imgID[0], HistAna.HistTbl, &RegTbl, 0);
        HistAnalyze(&HistAna, 0, 255);
        IP_Binarize(imgID[0], imgID[1], HistAna.thr);

        /***************
           画像表示
        ***************/
        DispImg(imgID[1]);

        /***************
           ウェイト
        ***************/
        DelayIPTask(30);
    }
}

void Terminate()
{
}
```

18.5.2 複数タスクのアプリケーション

以下に、複数タスクのユニット内ユーザーアプリケーションのサンプルを示します。

```
#include <stdio.h>
#include "vpxdef.h"
#include "vpxsys.h"
#include "vpxfnc.h"
#include "vpxcnv.h"

void apltsk(void);

#pragma section _main
void Main(int argc,char *argv[])
{
    int taskID;
    CREATE_TASK_TBL iptsk;

    printf("MAIN PROGRAM START !!\n");

    /*******************************
       画像処理システムの初期化
    *******************************/
    InitIP();

    /************************************
       カメラ番号とカメラタイプの選択
    ************************************/
    SelectCamera(CAMERA_PORT0, BW_CAMERA);

    /*****************
       タスク生成
    *****************/
    iptsk.task_addr  = (unsigned long)apltsk;
    iptsk.priority   = TASK_PRI_NORMAL;
    iptsk.pbuff_size = 0;
    iptsk.stack_size = 0x1000;
    iptsk.task_opt   = 0;
    iptsk.param_opt  = 0;
    taskID = CreateIPTask(&iptsk);

    /***********************
       タスク スタート
    ***********************/
    StartIPTask(taskID);

    /***********************
       メインタスクの終了
    ***********************/
}

void Terminate()
{
}
```

```
void apltsk(void)
{
    int imgID[2];
    HISTANATBL HistAna;
    IPGOFeatureTbl RegTbl;

    printf("APPLICATION TASK WAKEUP !!\n");

    /***************************
       画像メモリ領域の確保
    ***************************/
    imgID[0] = AllocImg(IMG_FS_512H_512V);
    imgID[1] = AllocImg(IMG_FS_512H_512V);

    while(1){
        /****************
           映像入力
        ****************/
        GetCamera(imgID[0]);

        /****************
           画像処理
        ****************/
        IP_Histogram(imgID[0],
        HistAna.HistTbl, &RegTbl, 0);
        HistAnalyze(&HistAna, 0, 255);
        IP_Binarize(imgID[0], imgID[1], HistAna.thr);

        /****************
           画像表示
        ****************/
        DispImg(imgID[1]);

        /****************
           ウェイト
        ****************/
        DelayIPTask(30);
    }

}
```

18.5.3 PIO割込みタスクのアプリケーション

以下に、ユニット内ユーザーアプリケーションをＰＩＯ割込みタスクとしてで実行する場合のサンプルを示します。

```
#include <stdio.h>
#include "vpxdef.h"
#include "vpxsys.h"
#include "vpxfnc.h"
#include "vpxcnv.h"

#pragma  section  _main
void Main(int argc,char *argv[])
{
CREATE_TASK_TBL iptsk;
INT_DEVICE_OBJ  intobj;

/*****************************************************************************
    画像処理コマンド初期化
*****************************************************************************/
   InitIP();

/*****************************************************************************
    画像メモリの領域確保
*****************************************************************************/
   _ImgID[0]= AllocImg(IMG_FS_512H_512V);
   _ImgID[1]= AllocImg(IMG_FS_512H_512V);

/*****************************************************************************
    割込起動タスクの生成
*****************************************************************************/
   // ＰＩＯ#0 割込タスクの生成
   iptsk.task_addr   = 0;
   iptsk.priority    = 0;
   iptsk.pbuff_size  = 0x1000; // 16K
   iptsk.stack_size  = 0x1000; // 16K
   iptsk.task_opt    = 0;
   iptsk.param_opt   = 0;
   _tskid[0]= CreateIPTask(&iptsk);
   intobj.intdev     = INTDEV_PIO;
   intobj.evtpri     = 0;
   intobj.intbit     = 0;// bit0
   CreateInterruptLink(&intobj,_tskid[0],INTEVENT_WAKEUP,0);
   // ＰＩＯ#1 割込タスクの生成
   iptsk.task_addr   = 0;
   iptsk.priority    = 0;
   iptsk.pbuff_size  = 0x1000; // 16K
   iptsk.stack_size  = 0x1000; // 16K
   iptsk.task_opt    = 0;
   iptsk.param_opt   = 0;
   _tskid[1]= CreateIPTask(&iptsk);
   intobj.intdev     = INTDEV_PIO;
   intobj.evtpri     = 0;
   intobj.intbit     = 1;// bit1
   CreateInterruptLink(&intobj,_tskid[1],INTEVENT_WAKEUP,0);

/*****************************************************************************
    割込起動タスクの登録
*****************************************************************************/
   /* 映像入力 タスク                          */
   RegistIPTask(_tskid[0],0x8C802000,0x8C803000,0);
   /* 画像処理 タスク                          */
   RegistIPTask(_tskid[1],0x8C804000,0x8C805000,0);

/*****************************************************************************
    割込起動タスクのセットアップ
*****************************************************************************/
   /* 映像入力 タスク スタート                 */
   StartIPTask(_tskid[0]);
   /* 画像処理 タスク スタート                 */
   StartIPTask(_tskid[1]);
}

void Terminate()
{
}
```

18.6 PCからのダウンロードと実行

ユニット内アプリケーションをＰＣからダウンロードし実行させることができます。

18.6.1 ダウンロード、実行フロー

以下に、ユニット内アプリケーションをＰＣからダウンロードし実行させる場合のフローを示します。

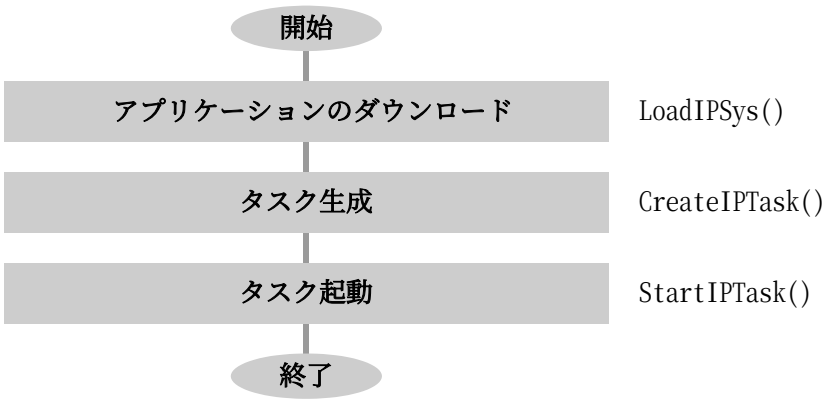


図１８－５　ダウンロード、実行フロー

18．6．2　ダウンロードと実行のアプリケーション

以下に、ユニット内アプリケーションをＰＣからダウンロードし実行させるプログラム例を示します。

```
#include <stdio.h>
#include <stdlib.h>

#include "vpxdef.h"
#include "vpxsys.h"
#include "vpxfnc.h"

void main()
{
int rc;
CREATE_TASK_TBL iptsk;
int taskid;
int devID;

   devID = OpenIPDevExt(IPBOARD_0, NULL);
   if(devID == ISPX_NULL){
      printf("デバイスのオープンに失敗しました\n");
      return;
   }

   /*********************************************
      ユーザーアプリケーションのダウンロード
   *********************************************/
   rc= LoadIPSys(devID,"USER.ABS",DIRECT_PATH);
   if(rc){
      printf("File Not Found !!\n");
      return;
   }
   printf("Download [USER.APL]\n");

   /*********************************************
      ユーザーアプリケーションタスクの生成
   *********************************************/
   iptsk.task_addr    = 0x8C800000;
   iptsk.priority     = 1;
   iptsk.pbuff_size   = 0;
   iptsk.stack_size   = 0x10000;
   iptsk.task_opt     = 0;
   iptsk.param_opt    = 0;

   taskid= CreateIPTask( devID,&iptsk);

   /*********************************************
      ユーザーアプリケーションタスクの起動
   *********************************************/
   rc = StartIPTask(devID,taskid);
   printf("StartIPTask [%d]\n",taskid);
}
```

付録１　ＩＰ５０００互換コマンド一覧

分類	No	コマンド名称	機能	備考
システム制御	1	InitIP	画像処理システムの初期化	
	2	InitIPExt	画像処理システムの初期化(ハードウェアテスト付き)	
	3	IPLibVersion	画像処理ライブラリのバージョン問い合わせ	
	4	IPLibType	画像処理ライブラリ種別とバージョン問い合わせ	
	5	IPBoardVersion	画像処理ボード、デバイスドライバのバージョン問い合わせ	
	6	SetIPDataType	画像処理システムデータタイプ設定	
	7	ReadIPDataType	画像処理システムデータタイプ読み出し	
	8	ISP_BusyWait	画像処理プロセッサのビジーウェイト	
	9	VP_BusyWait	映像入力待ち制御(VPビジーウェイト)	
	10	ActiveIP	マルチポート処理選択/NO.リード	
エラー制御	11	ClearIPError	コマンドエラーのクリア	
	12	ReadIPErrorTable	コマンドエラー情報の取得	
	13	EnableIPErrorMessage	エラーメッセージ出力許可	
	14	DisableIPErrorMessage	エラーメッセージ出力禁止	
画像メモリ管理	15	AllocImg	画像メモリ領域確保	
	16	AllocLockImg	画像メモリ領域確保(番地指定)	
	17	AllocYUVImg	画像メモリ領域確保(YUV画面	
	18	AllocRGBImg	画像メモリ領域確保(RGB画面	
	19	FreeImg	画像メモリ領域解放(指定画面	
	20	FreeAllImg	画像メモリ領域解放(全画面	
	21	ReadImgTable	画像メモリ管理テーブル読みだし	
	22	ReadImgTableEType	画像メモリ管理テーブル(画像メモリ領域管理フラグ)読みだし	
	23	ReadYUVImgTable	画像メモリ管理テーブル読みだし(YUV画面	
	24	ReadRGBImgTable	画像メモリ管理テーブル読みだし(RGB画面	
	25	GetUVImgID	UV画面番号抽出	
	26	GetGImgID	G画面番号抽出	
	27	GetBImgID	B画面番号抽出	
	28	ChangeImgDataType	画面データタイプ変更	
	29	ReadImgDataType	画面データタイプ読み出し	
	30	EnableIPWindow	ウィンドウ有効	
	31	DisableIPWindow	ウィンドウ無効	
	32	SetWindow	ウィンドウ設定(種類指定)	
	33	SetAllWindow	ウィンドウ設定(全種	
	34	ResetAllWindow	ウィンドウリセット	
	35	ReadWindow	ウィンドウ設定座標読出し	
	36	SetDefaultWindow	デフォルトウィンドウ設定	
画像メモリ制御	37	OpenImg	画像メモリアクセス可能	
	38	OpenImgExt	画像メモリアクセス可能(指定画面	
	39	CloseImg	画像メモリアクセス不可	
	40	ReadImg	画像メモリ読出し(指定画素数	
	41	WriteImg	画像メモリ書込み(指定画素数	
	42	SetPixelPointer	画像メモリ1画素アクセス画面設定	
	43	ReadPixel	画像メモリ読出し(座標指定 1画素	
	44	WritePixel	画像メモリ書込み(座標指定 1画素	
	45	ReadPixelContinue	画像メモリ読出し(次座標 1画素	
	46	WritePixelContinue	画像メモリ書込み(次座標 1画素	
	47	RefreshImg	画像メモリのリフレッシュ	
映像入力	48	SetVideoFrame	映像入力画面設定	
	49	SelectCamera	カメラ番号とカメラタイプの選択	
	50	GetCamera	カメラ映像入力	
	51	SetVFDelay	映像入力画面の遅延サイズ設定	
	52	Get2Camera	2カメラ映像の同時入力	
	53	ActiveVideoPort	ビデオポート選択	
	54	Get4Camera	4カメラ映像入力	
映像出力	55	DispCamera	カメラ映像表示	
	56	DispImg	画像メモリ表示	
	57	NoDisp	表示終了	
	58	BitmapOverlap	ビットマップ画面のオーバレイ	
	59	SetOverlayLUT	オーバレイテーブルの設定	
	60	SetDFDelay	映像表示画面の遅延サイズ設定	
	61	SetDispFrame	映像表示画面設定	
パイプライン制御	62	EnablePipeline	パイプラインモード設定	
	63	DisablePipeline	パイプラインモード解除	
画像クリア	64	IP_ClearAllImg	画像メモリクリア(全チャネル)	
	65	IP_ClearCHImg	画像メモリクリア(指定チャネル)	
	66	IP_ClearImg	画像メモリクリア(指定画面	
	67	IP_ClearColor	画像メモリクリア(カラー画面	
	68	IP_Const	定数発生	
画像転送	69	IP_Copy	転送	
アフィン変換	70	IP_Zoom	ズーム	
	71	IP_ZoomExt	ズーム(縦横任意倍率)	
	72	IP_Shift	シフト	
	73	IP_ZoomS	シフト付きズーム	
	74	IP_Rotate	回転	

2値化	75	IP_Binarize	2値化	
	76	IP_BinarizeExt	反転・反転付き2値化	
画素変換	77	IP_Invert	論理反転	
	78	IP_Minus	符号反転	
	79	IP_Abs	絶対値	
	80	IP_AddConst	定数加算	
	81	IP_SubConst	定数減算	
	82	IP_SubConstAbs	定数減算絶対値	
	83	IP_MultConst	定数乗算	
	84	IP_MinConst	定数比較　Min	
	85	IP_MaxConst	定数比較　Max	
	86	WriteConvertLUT	濃度変換データ設定	
	87	IP_ConvertLUT	濃度変換	
	88	IP_ShiftDown	定数シフトダウン	
	89	IP_ShiftUp	定数シフトアップ	
	90	IP_AddConst	定数論理積	
画像間算術演算	91	IP_Add	加算	
	92	IP_Sub	減算	
	93	IP_SubAbs	減算絶対値	
	94	IP_Comb	係数付加算	
	95	IP_CombAbs	係数付加算絶対値	
	96	IP_CombDrop	係数付加算（切り捨て）	
	97	IP_Mult	乗算	
	98	IP_Average	平均	
	99	IP_Min	比較　Min	
	100	IP_Max	比較　Max	
	101	IP_SubConstAbsAdd	定数減算絶対値和	
	102	IP_SubConstMultAdd	定数減算自乗和	
	103	IP_SubConstMult	定数減算積	
画像間論理演算	104	IP_And	論理積	
	105	IP_Or	論理和	
	106	IP_Xor	排他的論理和	
	107	IP_InvertAnd	否定論理積	
	108	IP_InvertOr	否定論理和	
	109	IP_Xnor	排他的否定論理和	
2値画像形	110	IP_PickNoise4	2値画像ノイズ除去（4連結）	
状変換	111	IP_PickNoise8	2値画像ノイズ除去（8連結）	
	112	IP_Outline4	2値画像輪郭抽出（4連結）	
	113	IP_Outline8	2値画像輪郭抽出（8連結）	
	114	IP_Dilation4	2値画像膨張（4連結）	
	115	IP_Dilation8	2値画像膨張（8連結）	
	116	IP_Erosion4	2値画像収縮（4連結）	
	117	IP_Erosion8	2値画像収縮（8連結）	
	118	IP_Thin4	2値画像細線化（4連結）	
	119	IP_Thin8	2値画像細線化（8連結）	
	120	IP_Shrink4	2値画像縮退化（4連結）	
	121	IP_Shrink8	2値画像縮退化（8連結）	
ｺﾝﾎﾞﾘｭｰｼｮﾝ	122	IP_SmoothFLT	平滑化	
	123	IP_EdgeFLT	濃淡画像輪郭強調（絶対値）	
	124	IP_EdgeFLTAbs	ラプラシアン（4連結）	
	125	IP_Lapl4FLT	ラプラシアン（8連結）	
	126	IP_Lapl8FLT	ラプラシアン（4連結・絶対値）	
	127	IP_Lapl4FLTAbs	ラプラシアン（8連結・絶対値）	
	128	IP_Lapl8FLTAbs	ラインフィルタ	
	129	IP_LineFLT	ラインフィルタ（絶対値）	
	130	IP_LineFLTAbs	平滑化（周辺部処理なし）	
ﾐﾆ/ﾏｯｸｽﾌｨﾙﾀ	131	IP_MinFLT	局所最小値フィルタ	
	132	IP_MinFLT4	局所最小値フィルタ（4連結）	
	133	IP_MinFLT8	局所最小値フィルタ（8連結）	
	134	IP_MaxFLT	局所最大値フィルタ	
	135	IP_MaxFLT4	局所最大値フィルタ（4連結）	
	136	IP_MaxFLT8	局所最大値フィルタ（8連結）	
	137	IP_LineMinFLT	ライン局所最小値フィルタ	
	138	IP_LineMaxFLT	ライン局所最大値フィルタ	
ﾗﾝｸﾌｨﾙﾀ	139	IP_RankFLT	局所ランクフィルタ	
	140	IP_Rank4FLT	局所ランクフィルタ（4連結）	
	141	IP_Rank8FLT	局所ランクフィルタ（8連結）	
	142	IP_MedFLT	局所メディアンフィルタ	
	143	IP_Med4FLT	局所メディアンフィルタ（4連結）	
	144	IP_Med8FLT	局所メディアンフィルタ（8連結）	
2値ﾊﾟｲﾌﾟﾗｲﾝﾌｨﾙ	145	SetTrsPipelineFLTMode	パイプラインフィルタのモード設定	
	146	IP_TrsPipelineFLT	2値パイプラインフィルタの実行	

2値ﾏｯﾁﾝｸﾞﾌｨﾙﾀ	147	SetBinMatchTemplate	2値画像マッチングフィルタ	
	148	IP_BinMatchFLT	2値画像テンプレート登録	
ラベリング	149	IP_Label4	ラベリング（4連結）	
	150	IP_Label8	ラベリング（8連結）	
	151	IP_Label4withAreaFLT	面積フィルタ付きラベリング（4連結）	
	152	IP_Label8withAreaFLT	面積フィルタ付きラベリング（8連結）	
	153	IP_Label4withAreaFLTSort	面積フィルタ付きラベリング（面積ソート4連結）	
	154	IP_Label8withAreaFLTSort	面積フィルタ付きラベリング（面積ソート8連結）	
	155	IP_ExtractLORegionX	ラベル毎領域X座標抽出（Min/Max　X座標）	
	156	IP_ExtractLORegionY	ラベル毎領域Y座標抽出（Min/Max　Y座標）	
	157	IP_ExtractLOArea	ラベル毎面積抽出（16ビット）	
	158	IP_ExtractLOAreaExt	ラベル毎面積抽出（32ビット）	
	159	IP_ExtractLOGravity	ラベル毎重心座標抽出	
濃淡画像	167	IP_ExtractGOFeatures	濃淡画像基本特徴量抽出	
特徴量抽出	168	IP_Histogram	濃度ヒストグラム（32ビット）	
	169	IP_ProjectGO	X&Y軸への濃度累積値投影（16ビット）	
	170	IP_ProjectGOonX	X軸への濃度累積値投影	
	171	IP_ProjectGOonY	Y軸への濃度累積値投影	
	172	IP_ProjectGOMaxValue	X&Y軸への最大濃度値投影	
	173	IP_ProjectGOMinValue	X&Y軸への最小濃度値投影	
2値画像	174	IP_ExtractBOArea	2値画像累積値抽出	
特徴量抽出	175	IP_ExtractBOFeatures	2値画像基本特徴量抽出	
	176	IP_ProjectBO	X&Y軸への投影	
	177	IP_ProjectBORegionX	領域X座標抽出　（Y軸へのMin&MaxX投影）	
	178	IP_ProjectBORegionY	領域Y座標抽出　（X軸へのMin&MaxY投影）	
ヒストグラム	179	EnableRotateProject	斜方投影処理有効	
その他	180	DisableRotateProject	斜方投影処理無効	
	181	IP_ProjectBlockBO	2値画像での領域毎の画像累積	
	182	IP_ProjectBlockGO	濃淡画像での領域毎の濃度累積値	
	183	IP_ProjectBlockGOMinMaxValue	濃淡画像での領域毎の最小／最大濃度値抽出累積値	
	184	IP_ProjectLabelGO	濃淡画像とラベル画像とのラベル毎の濃度累積値	
	185	IP_ProjectLabelGOMinMaxValue	濃淡画像での領域毎の最小／最大濃度値抽出	
正規化相関	186	SetCorrMode	相関サーチモード設定	
	187	DisableCorrMask	テンプレートマスク無効	
	188	EnableCorrMask	テンプレートマスク有効	
	189	ReadCorrMask	テンプレートマスク（無効/有効）読み出し	
	190	SetCorrBreakThr	打切り閾値設定	
	191	DisableCorrBreak	相関演算途中打切りの無効化	
	192	EnableCorrBreak	相関演算途中打切りの有効化	
	193	ReadCorrBreak	打切り閾値（禁止/許可）読み出し	
	194	SetCorrControl	正規化相関制御	
	195	SetCorrTemplate	トレーニング	
	196	SetCorrTemplateExt	トレーニング	
	197	IP_Corr	正規化相関	
	198	IP_CorrPecise	サブピクセルサーチ	
	199	SetOptFlowMode	オプティカルフローモード設定	
	200	IP_OptFlow	オプティカルフロー	
グラフィックス	201	SetDrawMode	描画画面設定	
	202	SetStringAttributes	文字の属性設定	
	203	DrawString	文字列描画	
	204	DrawLine	直線描画	
	205	DrawSegments	直線描画（複数）	
	206	DrawLines	折れ線描画	
	207	DrawRectangle	矩形描画	
	208	DrawPolygon	多角形描画	
	209	DrawArc	円弧描画	
	210	FillRectangle	矩形描画（塗りつぶし）	
	211	FillPolygon	多角形描画（塗りつぶし）	
画像ファイリング	212	LoadBMPFileInfo	BMPファイルヘッダー情報読出し	
	213	LoadBMPFile	BMPファイル画像データ読出し	
	214	SaveBMPFile	BMPファイル画像データ書込み	

カラー処理	215	IP_Mask	画像マスク	
	216	IP_ConvertRho	色彩距離変換	
	217	IP_ConvertTheta	色相変換	
	218	IP_ConvertRhoTheta	色彩距離・色相変換	
	219	IP_ExtractColor	色抽出処理	
	220	IP_ExtractColorRhoTheta	色抽出処理（色彩距離・色相変換）	
	221	IP_ConvertYUVtoRGB	カラー擬似変換（YUV→PGB）	
	222	IP_ConvertRGBtoYUV	カラー擬似変換（PGB→YUV）	
	223	IP_ConvertYUVtoRGBfast	カラー擬似変換（YUV→PGB）	
	224	IP_ConvertRGBtoYUVfast	カラー擬似変換（PGB→YUV）	
	225	IP_ConvertHue	色相変換	
	226	IP_ConvertSaturation	彩度変換	
	227	IP_ConvertIntensity	明度変換	
	228	IP_ExtractColorRGB	RGBカラー抽出処理	
	229	IP_ExtractColorHSI	カラー抽出処理 ― 色相・彩度・明度変換	
	230	OpenMultiColor	マルチカラー抽出処理可能	
	231	CloseMultiColor	マルチカラー処理不可	
	232	ClearMultiColorYRT	Ｙρθマルチカラーテーブルクリア	
	233	SetMultiColorYRT	Ｙρθマルチカラー抽出データ設定	
	234	IP_ExtractMultiColorRhoTheta	Ｙρθマルチカラー抽出	
	235	ClearMultiColor	マルチカラーテーブルクリア	
	236	SetMultiColor	マルチカラー抽出データ設定	
	237	IP_ExtractMultiColor	マルチカラー抽出	
	238	ClearMultiColorHSI	HISマルチカラーテーブルクリア	
	239	SetMultiColorHSI	HISマルチカラー抽出データ設定	
	240	IP_ExtractMultiColorHSI	HISマルチカラー抽出	
	241	ClearMultiColorRGB	RGBマルチカラーテーブルクリア	
	242	SetMultiColorRGB	RGBマルチカラー抽出データ設定	
	243	IP_ExtractMultiColorRGB	RGBマルチカラー抽出	
ｺﾝﾎﾞﾘｭｰｼｮﾝ	244	IP_SmoothFLT5x5	平滑化（5× 5）	
ﾌｨﾙﾀ拡張	245	IP_SmoothFLT7x7	平滑化（7× 7）	
	246	IP_EdgeFLT5x5	濃淡画像輪郭強調（5× 5）	
	247	IP_EdgeFLT7x7	濃淡画像輪郭強調（7× 7）	
	248	IP_MinFLT5x5	局所最小値フィルタ（5× 5）	
	249	IP_MinFLT44	局所最小値フィルタ（5× 5，44連結）	
	250	IP_MinFLT48	局所最小値フィルタ（5× 5，48連結）	
	251	IP_MinFLT88	局所最小値フィルタ（5× 5，88連結）	
	252	IP_MaxFLT5x5	局所最大値フィルタ（5× 5）	
	253	IP_MaxFLT44	局所最大値フィルタ（5× 5，44連結）	
	254	IP_MaxFLT48	局所最大値フィルタ（5× 5，48連結）	
	255	IP_MaxFLT88	局所最大値フィルタ（5× 5，88連結）	
	256	IP_SmoothFLT5x5Ext	平滑化(5× 5)	
	257	IP_SmoothFLT7x7Ext	平滑化(7× 7)	
	258	IP_EdgeFLT5x5Ext	画像輪郭強調（5× 5）	
	259	IP_EdgeFLT7x7Ext	画像輪郭強調（7× 7）	
拡張画像処理	260	IP_SmoothFLTExt	平滑化フィルタ（拡張版）	
	261	IP_EdgeFLTAbsExt	輪郭強調フィルタ・絶対値（拡張版）	
	262	IP_Sobel	エッジソーベル	
	263	IP_SobelBinarize	エッジソーベル（2 値化）	

付録２　画像処理時間

画像処理コマンドの処理時間を以下に示します。なお、時間はReadImg(),WriteImg()コマンド以外はＳＨ側での処理時間でＰＣ－ＳＨコマンド間のオーバヘッド（数百μs～数十 ms)は含まれていません。
以下の処理時間は、映像入力・表示をしない状態で計測したものですので、映像入力と並列に処理が行われる場合は処理時間が増加しますので注意して下さい。

付録２-１　処理時間一覧

処理サイズ：512×480

分類	No	コマンド名称	時間[ms]	備考
ｺﾏﾝﾄﾞｴﾗｰ制御	1	ClearIPError	0.1	
	2	CheckIPError	0.1	
	3	ReadIPErrorTable	0.1	
	4	EnableIPErrorMessage	0.1	
	5	DisableIPErrorMessage	0.1	
ｼｽﾃﾑ制御	6	InitIP	42.9	
	7	InitIPExt	2798.3	
	8	GetIPVersionInfo	0.1	
	9	CheckIPVersion	0.1	
	10	SetIPDataType	0.1	
	11	GetDispImgID	0.1	
	12	GetBitmapImgID	0.1	
	13	ISP_BusyWait	0.1	
	14	ImgBusyWait	0.1	
	15	Img2chBusyWait	0.1	
	16	ISP_BusyCheck	0.1	
	17	IPBoardType	0.1	
	18	ReadIPDataType	0.1	
	19	VP_BusyWait	0.1	
	20	ActiveIP	0.1	
画像ﾒﾓﾘ管理	21	AllocImg	0.3	
	22	AllocLockImg	0.3	
	23	FreeImg	0.1	
	24	FreeAllImg	0.1	12画面
	25	ReadImgTable	0.1	
	26	ChangeImgDataType	0.1	
	27	SetWindow	0.1	
	28	SetAllWindow	0.1	
	29	ResetAllWindow	0.1	
	30	EnableIPWindow	0.1	
	31	DisableIPWindow	0.1	
	32	ReadWindow	0.1	
	33	AllocDispImg	0.1	
	34	FreeDispImg	0.1	
	35	AllocImgExt	0.3	
	36	ReadImgTableType	0.1	
	37	ReadImgDataType	0.1	
	38	SetDefaultWindow	0.1	
	39	GetImPhysicalAddress	0.1	
	40	CacheFlush	0.1	
	41	ImCacheFlash	0.2	
映像入力	42	SetVideoFrame	0.1	
	43	SelectCamera	0.1	
	44	GetCamera	33.0	33～66ms
	45	GetCameraWithSelectPort	33.0	
	46	SetVFDelay	0.1	
	47	Get2Camera	33.0	33～66ms
	48	GetCameraSts	0.1	
	49	AttachRGBWorkImg	0.1	
	50	DetachRGBWorkImg	0.1	
	51	Get2CameraOpt	33.0	33～66ms
	52	ResetCamera	0.1	

映像出力	53	DispCamera	0.1	
	54	DispImg	0.1	
	55	NoDisp	0.1	
	56	BitmapOverlap	0.1	
	57	DispOverlap	0.1	
	58	SetDFDelay	0.1	
	59	SetDispFrame	0.1	
	60	SelectDisp	0.1	
	61	SetDispWindow	0.1	
	62	EnableHIREZDisp	0.1	
	63	DisableHIREZDisp	0.1	
	64	SetDispMode	0.1	
画像クリア	65	IP_ClearAllImg	51.1	12画面
	66	IP_ClearImg	4.3	
	67	IP_Const	4.0	
画像転送	68	IP_Copy	2.4	
ｱﾌｨﾝ変換	69	IP_Zoom	29.0	mag=1.0
	70	IP_ZoomExt	28.9	xmag=1.0, ymag=1.0
	71	IP_ZoomOut	2.5	mag=1
	72	IP_ZoomOutExt	2.5	xmag=1, ymag=1
	73	IP_Shift	5.5	shift x=100, shift=100, opt=0
	74	IP_ZoomS	26.6	mag=1.0, shift x=100, shift=100, opt=0
	75	IP_Rotate	34.4	mag=1.0, theta=45.0, dx=0, dy=0, opt=0
	76	IP_ZoomIn	2.5	mag=1
	77	IP_ZoomInExt	2.5	xmag=1, ymag=1
2値化	78	IP_Binarize	2.5	
	79	IP_BinarizeExt	2.5	
画素変換	80	IP_Invert	2.5	
	81	IP_Minus	2.5	
	82	IP_Abs	2.5	
	83	IP_AddConst	2.5	
	84	IP_SubConst	2.5	
	85	IP_SubConstAbs	2.5	
	86	IP_MultConst	2.5	
	87	IP_MinConst	2.5	
	88	IP_MaxConst	2.5	
	89	WriteConvertLUT	0.1	
	90	IP_ConvertLUT	2.5	
	91	IP_ShiftDown	2.5	
	92	IP_ShiftUp	2.5	
	93	IP_AndConst	2.5	
画像間算術演算	94	IP_Add	3.7	
	95	IP_Sub	3.7	
	96	IP_SubAbs	3.7	
	97	IP_Comb	3.7	
	98	IP_CombAbs	3.7	
	99	IP_Mult	3.7	
	100	IP_Average	3.7	
	101	IP_Min	3.7	
	102	IP_Max	3.7	
	103	IP_SubConstAbsAdd	3.7	
	104	IP_SubConstMultAdd	3.7	
	105	IP_SubConstMult	3.7	
	106	IP_CombDrop	3.7	
	107	IP_DivideConst	3.7	
画像間論理演算	108	IP_And	3.7	
	109	IP_Or	3.7	
	110	IP_Xor	3.7	
	111	IP_InvertAnd	3.7	
	112	IP_InvertOr	3.7	
	113	IP_Xnor	3.7	
2値画像形	114	IP_PickNoise4	2.5	
状変換	115	IP_PickNoise8	2.5	
	116	IP_Outline4	2.5	
	117	IP_Outline8	2.5	
	118	IP_Dilation4	2.5	
	119	IP_Dilation8	2.5	
	120	IP_Erosion4	2.5	
	121	IP_Erosion8	2.5	
	122	IP_Thin4	2.5	
	123	IP_Thin8	2.5	
	124	IP_Shrink4	2.5	
	125	IP_Shrink8	2.5	

ｺﾝﾎﾞﾘｭｰｼｮﾝ	126	IP_SmoothFLT	2.5	
	127	IP_EdgeFLT	2.5	
	128	IP_EdgeFLTAbs	2.5	
	129	IP_Lapl4FLT	2.5	
	130	IP_Lapl8FLT	2.5	
	131	IP_Lapl4FLTAbs	2.5	
	132	IP_Lapl8FLTAbs	2.5	
	133	IP_LineFLT	2.5	
	134	IP_LineFLTAbs	2.5	
ﾐﾆ/ﾏｯｸｽﾌｨﾙﾀ	135	IP_MinFLT	2.5	
	136	IP_MinFLT4	2.5	
	137	IP_MinFLT8	2.5	
	138	IP_MaxFLT	2.5	
	139	IP_MaxFLT4	2.5	
	140	IP_MaxFLT8	2.5	
	141	IP_LineMinFLT	2.5	
	142	IP_LineMaxFLT	2.5	
ﾗﾝｸﾌｨﾙﾀ	143	IP_RankFLT	2.5	
	144	IP_Rank4FLT	2.5	
	145	IP_Rank8FLT	2.5	
	146	IP_MedFLT	2.5	
	147	IP_Med4FLT	2.5	
	148	IP_Med8FLT	2.5	
ラベリング	149	IP_Label4	4.4	
	150	IP_Label8	4.4	
	151	IP_Label4withAreaFLT	6.3	
	152	IP_Label8withAreaFLT	6.3	
	153	IP_Label4withAreaFLTSort	6.3	
	154	IP_Label8withAreaFLTSort	6.3	
	155	IP_Label	4.4	
	156	IP_LabelwithAreaFLT	4.4	
	157	IP_ExtractLORegionX	2.0	
	158	IP_ExtractLORegionY	2.0	
	159	IP_ExtractLOArea	2.0	
	160	IP_ExtractLOAreaExt	2.0	
	161	IP_ExtractLOGravity	5.9	
濃淡画像 特徴量抽出	162	IP_ExtractGOFeatures	1.9	
	163	IP_Histogram	2.0	
	164	IP_ProjectGO	2.0	
	165	IP_ProjectGOonX	2.0	
	166	IP_ProjectGOonY	2.0	
	167	IP_ProjectGOMaxValue	2.0	
	168	IP_ProjectGOMinValue	2.0	
	169	IP_HistogramShort	2.0	
	170	IP_ProjectBlockGO	2.0	
	171	IP_ProjectBlockGOMinMaxValue	2.0	
	172	IP_ProjectLabelGO	2.6	
	173	IP_ProjectLabelGOMinMaxValue	2.6	
	174	EnableRotateProject	0.1	
	175	DisableRotateProject	0.1	
	176	IP_HistogramFeatures	2.0	
2値画像 特徴量抽出	177	IP_ExtractBOFeatures	4.5	
	178	IP_ProjectBO	2.0	
	179	IP_ProjectBORegionX	2.0	
	180	IP_ProjectBORegionY	2.0	
	181	IP_ExtractBOArea	2.0	
	182	IP_ProjectBlockBO	2.0	
ﾃﾝﾌﾟﾚｰﾄﾃﾞｰﾀ 領域管理	183	vpxAllocCorrTemplate	0.1	
	184	vpxFreeCorrTemplate	0.1	
	185	vpxReadCorrTemplate	0.1	
	186	vpxWriteCorrTemplate	0.1	
ﾄﾚｰﾆﾝｸﾞ	187	vpxEnableCorrMask	0.1	
	188	vpxDisableCorrMask	0.1	
	189	vpxSetCorrTemplate		付録2-3
	190	vpxSetCorrTemplateExt		
ｻｰﾁ	191	vpxIP_CorrStep		
	192	vpxIP_CorrPoint		
	193	vpxIP_CorrPrecise		
	194	vpxEnableCorrBreak	0.1	
	195	vpxDisableCorrBreak	0.1	
	196	vpxSetCorrBreakThr	0.1	
	197	vpxSetSearchDistance	0.1	
	198	vpxSetCorrMode	0.1	
	199	vpxSetCorrPrecise	0.1	
	200	vpxIP_CorrMap		付録2-3
	201	vpxGetCorrMapSize		

2値化閾値算出支	202	HistAnalyze	0.1	
直線抽出	203	GetHoughLine	2.5	Range(low=45, up=135)
	204	GetHoughLineRow	2.5	Range(low=45, up=135)
	205	GetHoughLineRowExt	2.5	Range(low=45, up=135), mode=MAX_LINE, thr=50
	206	GetSideHoughLine	0.1	
	207	GetCrossPoint	0.1	
	208	GetRectPoint	0.1	
	209	GetRectCenter	0.1	
	210	GetAnglePoint4	0.1	
	211	GetAnglePoint2	0.1	
イメージキャリパ	212	ProjectLine	13.8	Window(sx=5, sy=200, ex=505, ey=203, leng=1, opt=0)
	213	LineEdgeFilter	0.1	LineTble(lower=5, num=501)
	214	LineEdgeFilterExt	0.4	offset=2, scale=1, size=5
	215	LineCaliper	0.1	mode=0, score=4, Edge(lower=5, num=500)
	216	CaliperLPtoSP	0.1	
	217	GetCaliperScore	0.2	
	218	SetCaliperWidth	0.1	
ｴｯｼﾞﾌｧｲﾝﾀﾞ	219	LineEdgeFinder	0.1	
	220	EdgeFinderLPtoSP	0.1	
画像ﾒﾓﾘ制御	221	OpenImg	0.1	
	222	OpenImgExt	0.1	
	223	CloseImg	0.1	
	224	ReadImg	118.5	PC
	225		10.0	SH
	226	WriteImg	173.7	PC
	227		11.0	SH
	228	ReadImgReverse	15.0	
	229	WriteImgReverse	15.0	
	230	SetPixelPointer	0.1	
	231	ReadPixel	0.1	
	232	WritePixel	0.1	
	233	ReadPixelContinue	0.1	
	234	WritePixelContinue	0.1	
	235	ReadImgLine	0.1	
	236	WriteImgLine	0.1	
	237	GetImVirtualAddress	0.1	
	238	OpenImgDirect	0.1	
	239	CloseImgDirect	0.3	
	240	RefreshImg	0.1	
図形描画	241	PutLine		付録2-4
	242	PutCross		
	243	PutLineWindow		
	244	PutPolygon		
	245	PutCircle		
	246	PutArc		
	247	PutRectangle		
	248	PutTriangle		
	249	PutDiamond		
	250	PutCrossRect		
	251	PutTwinRectangle		
	252	PutArcExt		
	253	PutEllipse		
文字描画	254	PutAnkString		
	255	PutHalfString		
	256	PutAnkChar		
	257	PutHalfChar		
	258	PutString		
画面描画	259	ClearBitmap		付録2-6
	260	ClearScreen		
ﾌﾟﾘﾌｪｯﾁ処理	261	vpxEnableCameraPrefetch	0.1	
	262	vpxDisableCameraPrefetch	0.1	
	263	vpxResetCameraPrefetch	0.1	
	264	vpxGetCameraPrefetch	33.0	33～66ms
ﾊﾟｲﾌﾟﾗｲﾝ制御	265	EnablePipeline	0.1	
	266	DisablePipeline	0.1	

ﾌﾗｯｼｭﾒﾓﾘｱｸｾｽ	267	fmOpen	3.8	
	268	fmClose	0.1	書込みが無い場合
	269	fmRead	0.1	
	270	fmWrite	0.1	
	271	fmGetFileSize	0.1	
	272	fmDelete	540.0	
	273	fmRename	56.0	
	274	fmFileList	3.7	
	275	fmDiskSize	0.1	
	276	fmDiskFree	0.3	
	277	fmFindFile	0.2	
	278	fmChgAttr	503.0	
	279	fmSetAttr	0.0	
VP810互換	280	vpxInitIP	42.7	
	281	vpxEnableLoopCamera	0.1	
	282	vpxDisableLoopCamera	0.1	
	283	vpxSuspendLoopCamera	0.1	
	284	vpxResumeLoopCamera	0.1	
	285	EnableLoopDisp	0.1	
	286	DisableLoopDisp	0.1	
	287	SuspendLoopDisp	0.1	
	288	ResumeLoopDisp	0.1	
	289	vpxIP_Rotate	33.3	
	290	vpxWriteConvertLUT	0.1	
	291	vpxIP_SmoothFLT	2.5	
	292	vpxIP_EdgeFLT	2.5	
	293	vpxIP_EdgeFLTAbs	2.5	
	294	vpxIP_LineFLT	2.5	
	295	vpxIP_LineFLTAbs	2.5	
	296	vpxIP_SmoothFLTF	2.5	
	297	vpxIP_ExtractLOArea	2.0	
	298	vpxIP_ExtractLORegionX	2.0	
	299	vpxIP_ExtractLORegionY	2.0	
	300	vpxIP_ProjectGO	2.0	
	301	vpxIP_ProjectGOMinValue	2.0	
	302	vpxIP_ProjectGOMaxValue	2.0	
	303	vpxIP_ProjectBO	2.0	
	304	vpxIP_ProjectBORegionX	2.0	
	305	vpxIP_ProjectBORegionY	2.0	
	306	vpxIP_HistogramShort	2.0	
ﾏﾙﾁﾎﾟｰﾄ 映像入力	307	SetCameraPortConfig	0.1	
	308	ActiveVideoPort	0.1	
	309	SetVideoFrameMltPort	0.1	
	310	SelectCameraMltPort	0.1	
	311	GetCameraMltPort	33.0	33～66ms
	312	DispCameraMltPort	0.1	
	313	EnableLoopCameraMltPort	6.0	
	314	SuspendLoopCameraMltPort	0.1	

YUVｶﾗｰ処理	315	AllocYUVImg	0.6	
	316	GetUVImgID	0.1	
	317	ReadYUVImgTable	0.1	
	318	IP_Mask	3.4	
	319	IP_ClearColor	3.5	
	320	IP_ExtractColor	8.6	
	321	IP_ExtractColorRhoTheta	12.3	
	322	IP_ConvertRho	4.1	
	323	IP_ConvertTheta	7.6	
	324	IP_ConvertRhoTheta	11.7	
	325	IP_ConvertYUVtoRGB	351.4	
	326	IP_ConvertYUVtoRGBfast	21.5	
	327	OpenMultiColor	0.1	
	328	CloseMultiColor	0.1	
	329	ClearMultiColorYRT	0.1	
	330	SetMultiColorYRT	0.1	
	331	IP_ExtractMultiColorRhoTheta	12.6	
	332	ClearMultiColor	0.1	
	333	SetMultiColor	0.1	
	334	IP_ExtractMultiColor	9.3	
RGBｶﾗｰ処理	335	AllocRGBImg	0.9	
	336	ReadRGBImgTable	0.1	
	337	GetGImgID	0.1	
	338	GetBImgID	0.1	
	339	IP_ConvertHue	82.3	
	340	IP_ConvertSaturation	24.8	
	341	IP_ConvertIntensity	7.0	
	342	IP_ConvertRGBtoYUV	352.3	
	343	IP_ConvertRGBtoYUVfast	27.2	
	344	IP_ExtractColorRGB	9.3	
	345	IP_ExtractColorHSI	92.5	
	346	ClearMultiColorHSI	0.1	
	347	SetMultiColorHSI	0.1	
	348	IP_ExtractMultiColorHSI	92.5	
	349	ClearMultiColorRGB	0.1	
	350	SetMultiColorRGB	0.1	
	351	IP_ExtractMultiColorRGB	9.3	
画像ﾌｧｲﾘﾝｸﾞ	352	ReadBMPFileInfo	4.5	
	353	LoadBMPFile	97.0	
	354	SaveBMPFile	2990.0	
ｺﾝﾎﾞﾘｭｰｼｮﾝ ﾌｨﾙﾀ拡張	355	IP_SmoothFLT5x5	2.7	
	356	IP_SmoothFLT7x7	2.7	
	357	IP_EdgeFLT5x5	2.7	
	358	IP_EdgeFLT7x7	2.7	
	359	IP_MinFLT5x5	2.7	
	360	IP_MinFLT44	2.7	
	361	IP_MinFLT48	2.7	
	362	IP_MinFLT88	2.7	
	363	IP_MaxFLT5x5	2.7	
	364	IP_MaxFLT44	2.7	
	365	IP_MaxFLT48	2.7	
	366	IP_MaxFLT88	2.7	
	367	IP_SmoothFLT5x5Ext	2.7	
	368	IP_SmoothFLT7x7Ext	2.7	
	369	IP_EdgeFLT5x5Ext	2.7	
	370	IP_EdgeFLT7x7Ext	2.7	
	371	IP_SmoothFLTExt	2.7	
	372	IP_EdgeFLTAbsExt	7.2	

正規化相関 (IP5000互換)	373	SetCorrMode	0.1	
	374	DisableCorrMask	0.1	
	375	EnableCorrMask	0.1	
	376	ReadCorrMask	0.1	
	377	SetCorrBreakThr	0.1	
	378	DisableCorrBreak	0.1	
	379	EnableCorrBreak	0.1	
	380	ReadCorrBreak	0.1	
	381	SetCorrControl	0.1	
	382	SetCorrTemplate		付録2-2
	383	SetCorrTemplateExt		
	384	IP_Corr		
	385	IP_CorrPecise		
	386	SetOptFlowMode		
	387	IP_OptFlow		
グラフィックス	388	SetDrawMode		付録2-5
	389	SetStringAttributes		
	390	DrawString		
	391	DrawLine		
	392	DrawSegments		
	393	DrawLines		
	394	DrawRectangle		
	395	DrawPolygon		
	396	DrawArc		
	397	FillRectangle		
	398	FillPolygon		
拡張画像処理	399	IP_Sobel	7.2	method=1, n=1
	400	IP_SobelBinarize	7.2	method=1, gain=5.0, lutno=0
	401	IP_RegisterLUT	0.1	
	402	IP_Prewitt	7.2	
	403	IP_PrewittBinarize	7.2	
ﾗﾝﾚﾝｸﾞｽ・ﾗﾍﾞﾘﾝｸﾞ	404	IP_Label4byRL		付録2-7
	405	IP_Label8byRL		
	406	IP_Label4byRLwithAreaFLT		
	407	IP_Label8byRLwithAreaFLT		
	408	IP_Label4byRLwithAreaFLTSort		
	409	IP_Label8byRLwithAreaFLTSort		
	410	IP_Label4byRLExt		
	411	IP_Label8byRLExt		
	412	IP_Label4byRLwithAreaFLTExt		
	413	IP_Label8byRLwithAreaFLTExt		
	414	IP_Label4byRLwithAreaFLTSortExt		
	415	IP_Label8byRLwithAreaFLTSortExt		
	416	IP_LabelCombine		付録2-8
ビデオ拡張	417	WriteVideoLUT	0.1	
	418	SetVideoOpt	0.1	
	419	GetVideoOpt	0.1	
線分化	420	ExtractPolyline	1.3	
	421	PolyArea	0.2	
	422	PolyPerim	0.2	
	423	PolyGrav	0.5	
	424	PolyFeatures	2.8	
穴埋め	425	IP_FillHole	9.3	
	426	IP_FillHoleExt	9.3	
擬似ｶﾗｰ	427	IP_PseudoColor	19.3	
	428	SetPseudoColor	0.1	
	429	SetPseudoColorExt	0.1	

付録２-２　正規化相関(IP5000互換)コマンド

・テンプレート登録　・処理ｻｲｽﾞ：512ｘ480　・ﾃﾝﾌﾟﾚｰﾄｻｲｽﾞ：32x32，64x64　・SetCorrMode：N=4,thr=0.8

No.	コマンド名称	処理時間 [ms]		備考
		32x32	64x64	
1	SetCorrTemplate	0.1	0.1	
2	SetCorrTemplateExt	0.1	0.1	xmag=1，ymag=1

・SetCorrTemplate

No.	コマンド名称	走査方法	処理時間 [ms]		備考
			32x32	64x64	
1	IP_Corr	ﾗｽﾀ型	109.1	247.6	
		ｽﾊﾟｲﾗﾙ型	504.7	976.9	
2	IP_CorrPrecise	ﾗｽﾀ型	0.1	0.1	
		ｽﾊﾟｲﾗﾙ型	0.1	0.1	

・SetCorrTemplateExt (xmag=1，ymag=1)

No.	コマンド名称	走査方法	処理時間 [ms]		備考
			32x32	64x64	
1	IP_Corr	ﾗｽﾀ型	109.1	247.6	
		ｽﾊﾟｲﾗﾙ型	504.7	976.9	
2	IP_CorrPrecise	ﾗｽﾀ型	0.1	0.1	
		ｽﾊﾟｲﾗﾙ型	0.1	0.1	

・オプティカルフロー　・ﾃﾝﾌﾟﾚｰﾄｻｲｽﾞ：32x32，40x40

No.	コマンド名称	走査方法	処理時間 [ms]		備考
			32x32	40x40	
1	SetOptFlowMode	OPT_FULL_SEARCH_15	0.1	0.1	variance= 1880600，thr_c_value= 0
2	IP_OptFlow		3.6	4.6	num=4

付録２-３　正規化相関コマンド

・テンプレート登録　・処理ｻｲｽﾞ：512ｘ480　・ﾃﾝﾌﾟﾚｰﾄｻｲｽﾞ：32x32，64x64

No.	コマンド名称	処理時間 [ms]		備考
		32x32	64x64	
1	vpxSetCorrTemplate	0.1	0.1	xmag=1，ymag=1
2	vpxSetCorrTemplateExt	0.1	0.1	xmag=1，ymag=1

・サーチ（SetCorrTemplateExt）　・走査方法：ノーマル

No.	コマンド名称	xmag	ymag	StepX	StepY	処理時間 [ms]		備考
						32x32	64x64	
1	vpxIP_CorrStep	1	1	1	1	454.8	892.3	
		2	2	2	2	80.6	140.0	
		3	3	3	3	29.4	50.3	
		4	4	4	4	16.9	24.8	
		5	5	5	5	10.4	13.5	
		6	6	6	6	7.2	8.8	
		7	7	7	7	4.7	6.1	
		8	8	8	8	3.6	5.1	
2	vpxIP_CorrPoint	1	1	1	1	0.1	0.1	LengX= 4，LengY= 4
		2	2	2	2	0.1	0.1	
		3	3	3	3	0.1	0.1	
		4	4	4	4	0.1	0.1	
		5	5	5	5	0.1	0.1	
		6	6	6	6	0.1	0.1	
		7	7	7	7	0.1	0.1	
		8	8	8	8	0.1	0.1	
3	vpxIP_CorrPrecise	1	1	1	1	0.1	0.1	
		2	2	2	2	0.2	0.2	
		3	3	3	3	0.1	0.1	
		4	4	4	4	0.1	0.1	
		5	5	5	5	0.1	0.1	
		6	6	6	6	0.1	0.1	
		7	7	7	7	0.1	0.1	
		8	8	8	8	0.1	0.1	

・正規化相関

No.	コマンド名称	処理時間 [ms]		備考
		32x32	64x64	
1	vpxGetCorrMapSize	0.0	0.0	
2	vpxIP_CorrMap	358.0	686.9	

付録２-４　図形/文字描画コマンド

・処理ｻｲｽﾞ：512ｘ480　・mode：DRAW_FILL, BACKCOL_NOWRITE

No.	コマンド名称	処理時間 [ms]	備考
1	PutLine	0.089	DRAW_LINE (座標:始点 (0, 0)　終点 (511, 479)), mode=0
2	PutCross	0.004	十字
3	PutLineWindow	0.213	length=0, PutLine()と同領域
4	PutPolygon	5.964	PutLine()と同領域
5	PutCircle	9.329	中心 (255, 239) , 半径:240
6	PutArc	9.291	angle=360, 始点(255,0), PutCircle()と同条件
7	PutRectangle	5.675	round=0, angle=0, PutLine()と同条件
8	PutTriangle	3.519	round=0, angle=0, 中心 (255, 239) , lengh=511, lengv=480
9	PutDiamond	3.211	PutTriangle()と同条件
10	PutCrossRect	1.982	width=1, PutTriangle()と同条件
11	PutTwinRectangle	3.900	width=1, PutTriangle()と同条件
12	PutArcExt	6.936	angle=360, PutCircle()と同条件
13	PutEllipse	8.923	中心 (255, 239) , wideth=512, height=480
14	PutAnkString	0.043	一文字描画
15	PutHalfString	0.016	
16	PutAnkChar	0.022	
17	PutHalfChar	0.014	
18	PutString	0.025	

付録２-５　グラフィックス（IP5000互換）コマンド

・処理ｻｲｽﾞ：512ｘ480　・mode：DRAW_DIRECT, COLOR_WHITE

No.	コマンド名称	処理時間 [ms]	備考
1	SetDrawMode	0.003	
2	SetStringAttributes	0.003	
3	RefreshGraphics	0.004	
4	DrawString	0.026	STRING_16x16
		0.066	STRING_32x32
		0.126	STRING_48x48
		0.215	STRING_64x64
5	DrawLine	0.137	始点座標 (0,0)　終点座標 (511,479)
6	DrawSegments	0.475	ns=4, DrawLines()と同条件
7	DrawLines	0.506	np=4, 座標((0,0), (511,0), (511,479), (0,479))
8	DrawRectangle	0.782	座標:始点 (0, 0)　wideth=512, height=480
9	DrawPolygon	0.784	np=4, 座標((0,0), (511,0), (511,479), (0,479))
10	DrawArc	1.855	始点 (0,0) , width=511, height=479, angle1=0, angle2=360
11	FillRectangle	58.763	DrawRectangle()と同条件
12	FillPolygon	58.925	DrawPolygon()と同条件

付録２-６　画面描画コマンド

・処理ｻｲｽﾞ：512ｘ480

No.	コマンド名称	処理時間 [ms]	備考
1	ClearBitmap	3.5	画面全領域指定
2	ClearScreen	3.5	

付録２-７　ランレングス・ラベリングコマンド

・処理ｻｲｽﾞ：512ｘ480

No.	コマンド名称	ラベル数	処理時間 [ms]		備考
			画面出力	画面無出力	
1	IP_Label4RL	1	10.1	4.1	
		3	10.1	4.1	
		254	12.8	6.2	
		255	8.5	5.7	Label Overflow
2	IP_Label8RL	1	10.1	4.1	
		3	10.1	4.1	
		254	12.8	6.2	
		255	8.5	5.7	Label Overflow
3	IP_Label4RLwithArea	1	10.1	4.1	
		3	10.1	4.1	
		254	12.8	6.2	
		255	8.5	5.7	Label Overflow
4	IP_Label8RLwithArea	1	10.1	4.1	
		3	10.1	4.1	
		254	12.8	6.4	
		255	8.5	6.0	Label Overflow
5	IP_Label4RLwithAreaSort	1	10.1	4.1	
		3	10.1	4.1	
		254	13.1	6.4	
		255	8.9	6.0	Label Overflow
6	IP_Label8RLwithAreaSort	1	10.1	4.1	
		3	10.1	4.1	
		254	13.1	8.5	
		255	8.9	6.5	Label Overflow
7	IP_Label4RLExt	1	10.1	4.1	
		3	10.1	4.1	
		254	15.1	8.5	
		255	9.3	6.4	Label Overflow
8	IP_Label8RLExt	1	10.1	4.1	
		3	10.1	4.1	
		254	15.1	8.5	
		255	9.3	6.4	Label Overflow
9	IP_Label4RLwithAreaExt	1	10.1	4.1	
		3	10.1	4.1	
		254	15.1	8.5	
		255	9.4	6.4	Label Overflow
10	IP_Label8RLwithAreaExt	1	10.1	4.1	
		3	10.1	4.1	
		254	15.1	8.5	
		255	9.4	6.4	Label Overflow
11	IP_Label4RLwithAreaSortExt	1	10.1	4.1	
		3	10.1	4.1	
		254	15.3	8.7	
		255	9.7	6.8	Label Overflow
12	IP_Label8RLwithAreaSortExt	1	10.1	4.1	
		3	10.1	4.1	
		254	15.3	8.7	
		255	9.7	6.8	Label Overflow

付録２-８ 統合ラベリングコマンド

・処理ｻｲｽﾞ：512ｘ480

No.	連結手段	統合ラベリングオプション	ラベル数	処理時間 [ms]	備考
1	LABEL4	NORMAL_FEATURE	1	10.5	
			6	10.5	
			256	12.7	
			512	15.4	
			768	18.5	
			1024	21.2	
			1280	24.0	
			1536	26.7	
			1792	29.5	
			2000	31.7	
			2001	14.2	Label Overflow
2		EXTRA_FEATURE	1	13.2	
			6	13.3	
			256	19.7	
			512	26.6	
			768	33.9	
			1024	41.0	
			1280	47.8	
			1536	54.6	
			1792	61.5	
			2000	67.1	
			2001	14.2	Label Overflow
3		NORMAL_FEATURE_NODST	1	4.9	
			6	4.9	
			256	6.8	
			512	9.3	
			768	12.1	
			1024	14.8	
			1280	17.3	
			1536	19.8	
			1792	22.4	
			2000	24.4	
			2001	14.2	Label Overflow
4		EXTRA_FEATURE_NODST	1	7.8	
			6	7.8	
			256	14.1	
			512	21.0	
			768	28.3	
			1024	35.3	
			1280	42.1	
			1536	49.0	
			1792	55.9	
			2000	61.5	
			2001	14.2	Label Overflow

5	LABEL8	NORMAL_FEATURE	1	10.5	
			6	10.5	
			256	12.7	
			512	15.4	
			768	18.5	
			1024	21.3	
			1280	24.0	
			1536	26.8	
			1792	29.4	
			2000	31.7	
			2001	14.2	Label Overflow
6		EXTRA_FEATURE	1	13.2	
			6	13.3	
			256	19.7	
			512	26.6	
			768	34.0	
			1024	40.9	
			1280	47.8	
			1536	54.7	
			1792	61.6	
			2000	67.2	
			2001	14.2	Label Overflow
7		NORMAL_FEATURE_NODST	1	4.9	
			6	4.9	
			256	6.8	
			512	9.3	
			768	12.2	
			1024	14.8	
			1280	17.3	
			1536	19.8	
			1792	22.3	
			2000	24.4	
			2001	14.2	Label Overflow
8		EXTRA_FEATURE_NODST	1	7.8	
			6	7.9	
			256	14.2	
			512	21.0	
			768	28.3	
			1024	35.3	
			1280	42.1	
			1536	48.9	
			1792	55.9	
			2000	61.5	
			2001	14.2	Label Overflow

超小型画像処理ボード ＳＶＰ－３３０
Software Development Kit
ユーザーズガイド Version 2.0

開発元

株式会社　ルネサス北日本セミコンダクタ

電子機器本部　〒992-0021　山形県米沢市花沢3091-6
TEL 0238-22-7755　FAX 0238-22-6570
電子機器営業部　〒105-0004　東京都港区新橋5-11-3
（新橋住友ビル８階）
TEL 03-5733-4550（代）　FAX 03-5733-4660
技術サポート窓口　E_Mail　vp.support@kitasemi.renesas.com
URL　http://www.kitasemi.renesas.com